МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп’ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувачка кафедри,
д-р техн. наук, проф.

_____Ірина ЖУРАВСЬКА

« __ » _____ 202__ р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

Система детектування автономерів на основі Raspberry Pi, OpenCV та Spring Boot

Спеціальність 123 Комп’ютерна інженерія

Освітня програма «Комп’ютерна інженерія»

Здобувач _____Владислав ВОЛЯНИК
підпис

«__» _____ 2025 р.

Керівник канд. фіз-мат. наук, доц. _____ Сергій ПУЗИРЬОВ
підпис

«__» _____ 2025 р.

Миколаїв – 2025

Факультет	Комп’ютерних наук
Кафедра	Комп’ютерної інженерії
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	123 Комп’ютерна інженерія
Освітня програма	Комп’ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри комп’ютерної інженерії
_____ Ірина ЖУРАВСЬКА
«_____» _____ 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Воляник Владислав Сергійович
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи
Система детектування автономерів на основі Raspberry Pi, OpenCV та Spring Boot

Затверджена наказом по ЧНУ ім. Петра Могили від 30.10.2024 № 297.

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Очікуваний результат роботи – функціональна система автоматичного розпізнавання номерних знаків, яка здатна в режимі реального часу виявляти транспортні засоби, зчитувати та зберігати їхні номерні знаки у базі даних.
Початкові дані – відеопотік або зображення з камери, попередньо налаштована база даних, програмне середовище (Python, OpenCV, Tesseract, Java, PostgreSQL), а також тестовий набір зображень номерних знаків для перевірки точності системи.

4. Перелік питань, що підлягають розробці:

– провести аналіз сучасних технологій і рішень у сфері автоматичного розпізнавання номерних знаків;
– розробити загальну архітектуру системи з урахуванням обмежень апаратної платформи Raspberry Pi;
– реалізувати модуль захоплення та попередньої обробки зображень за допомогою OpenCV;

– розробити серверну частину на основі Spring Boot для обробки результатів розпізнавання;
– забезпечити збереження та обробку отриманих даних у базі даних;
– провести тестування розробленої системи та оцінити її ефективність;
– надати рекомендації щодо подальшої оптимізації та розвитку системи.

5. Перелік графічних матеріалів
Презентація

6. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи _____ Сергій ПУЗИРЬОВ
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Владислав ВОЛЯНИК
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання «29» __жовтня__ 2024

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної бакалаврської роботи

Тема: Система детектування автономерів на основі Raspberry Pi, OpenCV ra Spring Boot

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КБР	29.10.2024	30.10.2024	Виконано
2	Огляд літератури за темою роботи	08.11.2024	30.11.2024	Виконано
3	Складання календарного плану КБР	02.01.2025	03.01.2025	Виконано
4	Аналіз предметної області	04.01.2025	20.02.2025	Виконано
5	Розробка проєктних рішень	03.02.2025	25.02.2025	Виконано
6	Моделювання та конструювання АПЗ	07.04.2025	21.03.2025	Виконано
7	Перевірка працездатності, тестування та апробація розробленого АПЗ, аналіз результатів тестування	06.04.2025	20.04.2025	Виконано
8	Відгук керівника КБР	15.06.2025	15.06.2025	Виконано
9	Оформлення КБР та презентації	18.05.2025	16.06.2025	Виконано
10	Попередній захист	21.05.2025	04.06.2025	Виконано
11	Рецензування	14.06.2025	16.06.2025	Виконано
12	Завершення оформлення КБР та презентації	10.06.2025	16.06.2025	Виконано
13	Захист кваліфікаційної бакалаврської роботи	24.06.2025	24.06.2025	Виконано

Керівник роботи _____ Сергій ПУЗИРЬОВ

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Владислав ВОЛЯНИК

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

«____» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи
«Система детектування автономерів на основі Raspberry Pi, OpenCV та Spring Boot»

Студент 405 гр.: Воляник Владислав Сергійович
Керівник: канд. фіз-мат. наук, доц. Пузирьов Сергій Володимирович

Актуальність теми зумовлена постійним зростанням потреби у сучасних та надійних засобах автоматизації процесів контролю транспортних засобів. Це пов’язано з необхідністю підвищення рівня безпеки на дорогах, ефективності обліку автомобілів, моніторингу транспортних потоків, зниження рівня правопорушень, а також оптимізації роботи служб, відповідальних за дорожній контроль. Особливої актуальності дана тема набуває у великих містах, де щодня фіксується значна кількість транспортних подій, і ручне відстеження є малоефективним.

Об’єктом дослідження виступає процес цифрової обробки зображень транспортних засобів, що включає етапи виявлення об'єкта на зображенні, його сегментацію, виділення ключових ознак та підготовку зображення для подальшої обробки алгоритмами машинного навчання. Особливу увагу приділено обробці номерних знаків, що є основним ідентифікатором транспортного засобу.

Предметом дослідження є методи, моделі та алгоритми, що використовуються для автоматичного виявлення і розпізнавання державних реєстраційних номерних знаків транспортних засобів. Розглядаються алгоритми комп’ютерного зору, машинного та глибокого навчання, які дозволяють здійснити точне, стабільне та швидке розпізнавання у різних умовах освітлення, кутах нахилу, забрудненості знака тощо.

Метою роботи є розробка ефективної апаратно-програмної системи, здатної в автоматичному режимі з високим рівнем точності виявляти та розпізнавати номерні знаки транспортних засобів на зображеннях або відеопотоці. Система має забезпечувати стійкість до зовнішніх факторів та відповідати вимогам до продуктивності в умовах реального часу.

Практична значимість дослідження полягає у можливості впровадження розробленої системи у різноманітні практичні сфери: системи відеоспостереження, інтелектуальні системи дорожнього контролю, платні паркувальні майданчики, автоматизовані пункти пропуску, митні та прикордонні служби, а також у сфері смарт-міст. Це дозволить суттєво підвищити рівень автоматизації та зменшити потребу у людському факторі.

Практичне значення роботи також полягає у використанні сучасних технологій комп’ютерного зору, згорткових нейронних мереж і методів глибокого навчання, що дозволяють досягти високої точності при розпізнаванні навіть заскладнених зображень. Реалізація таких підходів сприяє покращенню швидкодії системи та підвищенню її адаптивності до змін у зовнішньому середовищі.

Основні результати роботи:

У першому розділі було здійснено детальний аналіз предметної області. Розглянуто існуючі програмно-апаратні рішення в галузі автоматичного розпізнавання номерних знаків, проаналізовано їх переваги та недоліки. На основі проведеного аналізу обґрунтовано вибір програмних засобів, архітектурних підходів і технологій, які використано в реалізації.

У другому розділі описано загальну архітектуру створеної системи, включаючи опис апаратної платформи (Raspberry Pi, камера, блок живлення, модулі зберігання даних) та використаного програмного забезпечення. Також наведено діаграми взаємодії компонентів системи та обґрунтовано вибір методів обробки та розпізнавання.

У третьому розділі представлено результати експериментальних досліджень, які охоплюють вимірювання точності розпізнавання, часу обробки одного зображення та стійкості роботи системи при змінних умовах освітлення, відстані та кутах зйомки. Оцінено ефективність системи та підтверджено доцільність її практичного використання.

Обсяг роботи: 62 сторінки (без додатків), 19 рисунків, 20 джерел посилання та 2 додатки.

Ключові слова*: комп'ютерний зір, номерний знак, розпізнавання, обробка зображень, глибоке навчання, Python, система моніторингу.*

ABSTRACT

of the Bachelor's Thesis

"License Plate Detection System Based on Raspberry Pi, OpenCV, and Spring Boot"

Student: Vladyslav Volianyk

Supervisor: Ph.D. Phys. & Math. Sci. Doc. Serhiy Puzyryov

Relevance of the topic is driven by the growing demand for modern and reliable means of automating the control of vehicles. This is connected to the need to improve road safety, streamline vehicle accounting, monitor traffic flow, reduce the number of violations, and optimize the work of road control services. The importance of this topic is especially evident in large urban areas, where the volume of daily transport events is high, and manual monitoring becomes ineffective.

The object of the study is the process of digital image processing of vehicles, which includes the stages of object detection in an image, segmentation, feature extraction, and preparing the image for further analysis using machine learning algorithms. Special attention is given to license plate processing, as it is the primary identifier of a vehicle.

The subject of the study includes the methods, models, and algorithms used for the automatic detection and recognition of vehicle license plates. The study examines computer vision techniques as well as machine and deep learning approaches that enable accurate, robust, and fast recognition under various conditions such as poor lighting, skewed angles, and dirty plates.

The aim of the work is to develop an efficient hardware-software system capable of detecting and recognizing vehicle license plates automatically with a high degree of accuracy. The system should be resistant to external factors and meet real-time performance requirements.

The practical significance of the research lies in the possibility of implementing the developed system in various applications such as video surveillance systems, intelligent traffic control systems, paid parking areas, automated checkpoints, customs, and border control. This can significantly improve automation levels and reduce dependence on human effort.

The practical value of the work also lies in the use of modern computer vision technologies, convolutional neural networks, and deep learning methods. These enable high recognition accuracy even with complex images. The implementation of such approaches improves processing speed and enhances the system's adaptability to environmental changes.

Main results of the work:

The first chapter provides an in-depth analysis of the subject area. It reviews existing hardware and software solutions for automatic license plate recognition, analyzing their advantages and limitations. Based on the analysis, appropriate tools, technologies, and architectural approaches were selected for implementation.

The second chapter describes the overall architecture of the developed system, including the hardware platform (Raspberry Pi, camera, power supply, storage modules) and the software stack (OpenCV, TensorFlow, etc.). Diagrams of component interactions are provided, and the choice of processing and recognition methods is justified.

The third chapter presents the results of experimental research, covering metrics such as recognition accuracy, image processing time, and system reliability under varying lighting conditions, distances, and camera angles. The system's effectiveness was evaluated, confirming its practical viability.

Scope: 62 pages (excluding appendices), 19 figures, 20 references, and 2 appendices.

Keywords*: computer vision, license plate, recognition, image processing, deep learning, Python, monitoring system.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АПЗ – Апаратно програмне забезпечення

АРНЗ – Автоматичне розпізнавання номерних знаків

API – Application Programming Interface

ALPR – Automatic License Plate Recognition,

ANPR – Automatic Number Plate Recognition

LPR – License Plate Recognition

OCR – Optical Character Recognition

REST API – Representational State Transfer Application Programming Interface

ROI –Region of Interest

UI – User Interface

YOLO – You Only Look Once

ВСТУП

Розвиток інтелектуальних систем обробки зображень і комп’ютерного зору в сучасних умовах стає важливим напрямом науково-технічного прогресу. Серед прикладних завдань цього напряму особливе місце займають системи автоматичного розпізнавання державних реєстраційних номерних знаків транспортних засобів. Актуальність теми пояснюється широким практичним застосуванням таких систем у галузях безпеки, контролю доступу, автоматизації парковок та транспортної аналітики. Науково-практичне значення роботи полягає у створенні ефективної, компактної, енергоощадної та недорогої системи детектування номерних знаків з використанням апаратної платформи Raspberry Pi, бібліотеки OpenCV і серверної частини на базі Spring Boot.

Об'єкт дослідження – процес автоматичної ідентифікації транспортних засобів за допомогою розпізнавання державних реєстраційних номерних знаків.

Предмет дослідження – апаратно-програмна система детектування і розпізнавання номерних знаків на базі Raspberry Pi з використанням інструментів комп’ютерного зору та серверної обробки даних через Spring Boot.

Мета роботи – підвищення ефективності автоматичної ідентифікації транспортних засобів за рахунок розробки системи детектування номерних знаків на основі Raspberry Pi, OpenCV та Spring Boot.

Для досягнення визначеної мети необхідно вирішити такі завдання:

- провести аналіз сучасних технологій і рішень у сфері автоматичного розпізнавання номерних знаків;
- розробити загальну архітектуру системи з урахуванням обмежень апаратної платформи Raspberry Pi;
- реалізувати модуль захоплення та попередньої обробки зображень за допомогою OpenCV;
- розробити серверну частину на основі Spring Boot для обробки результатів розпізнавання;
- забезпечити збереження та обробку отриманих даних у базі даних;
- провести тестування розробленої системи та оцінити її ефективність;

- надати рекомендації щодо подальшої оптимізації та розвитку системи.

Обґрунтування необхідності нової розробки. Аналіз сучасного стану проблеми за даними вітчизняної та зарубіжної науково-технічної літератури показує, що існуючі рішення в області розпізнавання номерних знаків переважно орієнтовані на використання потужних серверів або комерційних хмарних сервісів (OpenALPR, PlateRecognizer, Amazon Rekognition), що значно підвищує вартість системи та залежить від стабільного інтернет-з'єднання. Провідні фірми у цій галузі пропонують дорогі рішення для великих підприємств і державних структур. Наявні прогалини включають недостатню кількість недорогих автономних рішень для невеликих об'єктів та відсутність інтеграції таких систем з компактними пристроями типу Raspberry Pi [1].

Обґрунтування основних проєктних рішень. Вибір Raspberry Pi обумовлений потребою у мобільності, низькій вартості та енергоефективності пристрою. Бібліотека OpenCV дозволяє реалізувати складні алгоритми обробки зображень навіть на обмежених ресурсах. Серверна частина на базі Spring Boot забезпечує масштабованість та можливість інтеграції з іншими інформаційними системами.

Сфера застосування результатів. Розроблена система може бути використана у сфері безпеки для контролю доступу на закриті об'єкти, у системах автоматизації паркінгів, в муніципальних службах контролю транспорту, а також в приватному секторі для моніторингу прибуття і відправлення транспортних засобів.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ ІДЕНТИФІКАЦІЇ ТА ОБЛІКУ ТРАНСПОРТНИХ ЗАСОБІВ

1.1 Актуальність теми

У сучасних умовах динамічного розвитку транспортної інфраструктури питання ідентифікації та обліку транспортних засобів набувають особливої актуальності та стратегічної важливості. Стрімке збільшення кількості автомобілів на дорогах України та світу створює безпрецедентні виклики для систем контролю, безпеки та управління транспортними потоками. Ця тенденція вимагає впровадження інноваційних, високоефективних систем моніторингу, автоматизованого контролю доступу, оптимізації паркувальних сервісів та забезпечення надійної безпеки на об'єктах різного призначення – від бізнес-центрів та промислових підприємств до житлових комплексів та об'єктів критичної інфраструктури.

Автоматичне розпізнавання державних реєстраційних номерних знаків транспортних засобів стало одним із найперспективніших напрямів розвитку інтелектуальних транспортних систем. Технології комп'ютерного зору та машинного навчання відкривають широкі можливості для створення систем, що здатні в реальному часі ідентифікувати транспортні засоби за їхніми номерними знаками з високою точністю та надійністю [2].

Впровадження таких систем дозволяє досягти низки суттєвих переваг:

- мінімізація впливу людського фактору на процеси контролю та реєстрації;
- багаторазове підвищення швидкості обробки та аналізу інформації;
- забезпечення цілодобового безперервного моніторингу та контролю доступу;
- створення комплексних систем збору та аналізу даних у режимі реального часу;
- інтеграція з іншими системами безпеки та управління інфраструктурою.

Важливо відзначити, що такі системи знаходять застосування у різноманітних сферах:

- контроль доступу на територію підприємств, житлових комплексів, паркінгів;
- автоматизація роботи платних парковок та паркувальних комплексів;
- моніторинг транспортних потоків у містах та на магістралях;
- виявлення викрадених автомобілів та транспортних засобів у розшуку;
- контроль дотримання правил дорожнього руху та автоматична фіксація порушень;
- забезпечення безпеки на об'єктах критичної інфраструктури [3].

В умовах описаних обмежень особливої актуальності набуває створення ефективної, доступної та автономної апаратно-програмної системи для розпізнавання автономерів на базі відкритих технологій. Використання таких компонентів як Raspberry Pi (компактний та енергоефективний одноплатний комп'ютер), бібліотека комп'ютерного зору OpenCV та фреймворк Spring Boot дозволяє побудувати рішення, що має низку переваг:

1) економічна доступність – використання недорогих компонентів та відкритого програмного забезпечення істотно знижує вартість системи;

2) енергоефективність – низьке енергоспоживання дозволяє використовувати систему в умовах обмежених енергетичних ресурсів;

3) автономність – можливість функціонування без постійного підключення до мережі інтернет;

4) гнучкість та адаптивність – відкрита архітектура дозволяє модифікувати систему під специфічні потреби;

5) масштабованість – можливість поступового нарощування кількості пристроїв у мережі;

6) інтеграція з іншими системами – відкриті стандарти та інтерфейси спрощують взаємодію з існуючою інфраструктурою.

Науково-практичне значення теми полягає у розробці оптимізованого підходу до обробки зображень і розпізнавання символів в умовах обмеженої

обчислювальної потужності, що сприятиме поширенню систем комп'ютерного зору в нових сферах застосування: смарт-міста, розумні парковки, безпека територій, автоматизація пропускних пунктів.

Отже, для досягнення визначеної мети необхідно вирішити такі завдання:

- провести аналіз сучасних технологій і рішень у сфері автоматичного розпізнавання номерних знаків та сформувати вимоги до апаратно – програмного забезпечення ;
- розробити загальну архітектуру системи з урахуванням обмежень апаратної платформи Raspberry Pi;
- реалізувати модуль захоплення та попередньої обробки зображень за допомогою OpenCV;
- розробити серверну частину на основі Spring Boot для обробки результатів розпізнавання.
- забезпечити збереження та обробку отриманих даних у базі даних.
- провести тестування розробленої системи та оцінити її ефективність.
- надати рекомендації щодо подальшої оптимізації та розвитку системи.

У межах проєкту передбачено виконання комплексу завдань для створення ефективної системи розпізнавання номерних знаків. Основна увага приділена аналізу сучасних технологій, розробці архітектури, реалізації ключових модулів та оцінці ефективності системи.

1.1.1 Об'єкт і предмет дослідження

Об'єктом дослідження виступає багатокомпонентний процес автоматичної ідентифікації транспортних засобів за державними реєстраційними номерними знаками як комплексне науково-технічне явище. Цей процес є фундаментальним елементом сучасних інтелектуальних транспортних систем і набув безпрецедентної актуальності в умовах стрімкої урбанізації, діджиталізації міської інфраструктури та нових викликів у сфері безпеки й організації дорожнього руху.

Загострення потреби в ефективних системах автоматичної ідентифікації транспортних засобів обумовлюється низкою взаємопов'язаних факторів.

Насамперед слід відзначити експоненційне зростання кількості транспортних засобів у містах та на автомагістралях, що створює безпрецедентне навантаження на транспортну інфраструктуру та контролюючі органи. Згідно з статистикою, в більшості розвинених країн щорічне збільшення автопарку становить 3–5 %, а в країнах, що розвиваються, цей показник може сягати 7–10 %. Це призводить до суттєвого ускладнення задач контролю та моніторингу транспортного потоку традиційними методами, що вимагає впровадження автоматизованих рішень [4].

Одночасно з цим відбувається посилення вимог до комплексної безпеки об'єктів інфраструктури, яке спонукає до впровадження автоматизованих систем контролю доступу з високим рівнем надійності та мінімальною затримкою в обробці даних. Сучасні стандарти безпеки передбачають багаторівневі системи верифікації з використанням різноманітних біометричних та технічних ідентифікаторів, серед яких розпізнавання номерних знаків займає особливе місце завдяки неінвазійному характеру та високій ефективності.

Паралельно з цими процесами відбувається інтенсифікація заходів з контролю дорожнього руху та правозастосування у сфері дотримання правил дорожнього руху, що неможливо ефективно реалізувати без комплексних автоматизованих систем моніторингу.

Статистика свідчить про те, що впровадження автоматичних систем фіксації порушень ПДР знижує кількість аварій на контрольованих ділянках на 15–30 % протягом перших років експлуатації, що підтверджує критичну важливість таких систем для забезпечення безпеки дорожнього руху. Революційні зміни охопили й організацію паркувальних просторів, які трансформуються з простих майданчиків для стоянки в інтелектуальні системи управління з функціями моніторингу, оптимізації заповнення, автоматичного розрахунку вартості та безготівкової оплати, що неможливо уявити без надійної ідентифікації транспортних засобів за номерними знаками [5].

Концепція «розумного міста» також активно розвивається, і в ній автоматична ідентифікація транспортних засобів постає як невід'ємна складова інтегрованої системи управління міським транспортом, що дозволяє оптимізувати

транспортні потоки, зменшувати затори та знижувати рівень забруднення повітря, сприяючи покращенню екологічної ситуації та якості життя в міських агломераціях.

На фоні цих позитивних трансформацій не можна ігнорувати і зростання загроз, пов'язаних з тероризмом та організованою злочинністю, що вимагає створення ефективних систем раннього виявлення та попередження потенційних загроз, серед яких системи автоматичного розпізнавання номерних знаків відіграють важливу роль у забезпеченні національної безпеки та протидії протиправним діям.

Об'єктом дослідження виступає багатокомпонентний процес автоматичної ідентифікації транспортних засобів за державними реєстраційними номерними знаками як комплексне науково-технічне явище. Цей процес є фундаментальним елементом сучасних інтелектуальних транспортних систем і набув безпрецедентної актуальності в умовах стрімкої урбанізації, діджиталізації міської інфраструктури та нових викликів у сфері безпеки й організації дорожнього руху.

Предметом дослідження є розробка інтегрованого апаратно-програмного комплексу для детектування, розпізнавання та передачі інформації про автономери, який базується на використанні компактної обчислювальної платформи Raspberry Pi та реалізується за допомогою сучасних технологій комп'ютерного зору OpenCV у поєднанні з серверною технологією Spring Boot для забезпечення надійної обробки та збереження даних. Цей комплекс представляє собою цілісне рішення, що охоплює всі етапи обробки візуальної інформації від моменту захоплення зображення до формування структурованої бази даних розпізнаних номерних знаків та надання програмних інтерфейсів для інтеграції з іншими системами.

Дослідження безпосередньо фокусується на розробці та оптимізації методів обробки вхідних зображень, що є первинною ланкою в ланцюгу розпізнавання. Ця складова включає комплекс алгоритмічних рішень для попередньої обробки зображень з метою покращення їх якості (нормалізація яскравості та контрасту, усунення шумів, стабілізація зображення при динамічному захопленні), що є критично важливим для подальших етапів роботи системи, особливо в умовах

різної освітленості, несприятливих погодних умов та недостатньої якості вхідних даних. Підхід до обробки зображень розробляється з урахуванням обмежених обчислювальних можливостей платформи Raspberry Pi та необхідності забезпечення роботи системи в режимі реального часу, що вимагає нетривіальних рішень у галузі оптимізації алгоритмів комп'ютерного зору [6].

Наступним ключовим аспектом предмету дослідження є розробка ефективних методів детекції номерних знаків на оброблених зображеннях транспортних засобів. Ця задача вирішується шляхом розробки спеціалізованих алгоритмів локалізації областей інтересу, що базуються на комбінації класичних методів комп'ютерного зору (каскадні класифікатори, фільтрація за геометричними ознаками) та сучасних підходів з використанням згорткових нейронних мереж, адаптованих для роботи на пристрої з обмеженими ресурсами. Особлива увага приділяється стійкості методів детекції до різних кутів огляду, часткового затінення номерного знаку та варіативності типів номерних знаків, що зустрічаються в реальних умовах експлуатації.

Після успішної локалізації номерного знаку на зображенні дослідження зосереджується на розпізнаванні текстової інформації, що містить літерно-цифрові символи державного реєстраційного номера. Ця складова вимагає розробки алгоритмів сегментації символів на номерному знаку та їх подальшої класифікації з використанням методів машинного навчання, адаптованих для функціонування в умовах обмежених обчислювальних ресурсів Raspberry Pi. Дослідження охоплює техніки нормалізації зображень символів, виділення ключових ознак та побудови ефективних класифікаторів, здатних розрізняти символи навіть при частковому пошкодженні або забрудненні номерного знаку, що є типовим сценарієм у реальних умовах експлуатації.

Невід'ємною складовою предмету дослідження є також розробка архітектури серверної частини системи на базі технології Spring Boot, що забезпечує ефективну обробку, збереження та надання доступу до розпізнаних даних. Це включає проектування оптимальної структури бази даних для збереження інформації про зафіксовані номерні знаки з урахуванням вимог до швидкодії, масштабованості та

захисту інформації. Особлива увага приділяється розробці REST API (Representational State Transfer Application Programming Interface) для інтеграції системи з іншими компонентами інформаційної інфраструктури, а також механізмам синхронізації даних між розподіленими пристроями розпізнавання та центральним сервером, що працюють в умовах нестабільного мережевого з'єднання [6].

Таким чином, предмет дослідження представляє собою комплексний аналіз та розробку цілісної системи від апаратного рівня до програмної реалізації, що охоплює всі аспекти процесу автоматичної ідентифікації транспортних засобів – від отримання вхідних зображень до надання структурованих даних для подальшого використання в різних прикладних сценаріях. Практична цінність такого дослідження полягає у створенні доступного, енергоефективного та гнучкого рішення, що може бути адаптоване до різних умов експлуатації та інтегроване в існуючі системи контролю доступу, моніторингу транспортних потоків та забезпечення безпеки об'єктів інфраструктури.

1.1.2 Структурні і функціональні особливості об'єкта дослідження

Процес автоматичного розпізнавання автономерів транспортних засобів являє собою складний багатоетапний технологічний цикл, що об'єднує різноманітні методи комп'ютерного зору, алгоритми обробки зображень та технології машинного навчання в єдину функціональну систему. Цей процес розпочинається із захоплення зображень за допомогою спеціалізованої відеокамери, підключеної безпосередньо до обчислювальної платформи Raspberry Pi, яка виступає центральним компонентом апаратної частини системи. Важливо зазначити, що відеокамера може функціонувати як у стаціонарному режимі при встановленні на фіксованих об'єктах (наприклад, на в'їзних воротах, контрольно-пропускних пунктах або елементах дорожньої інфраструктури), так і в мобільному варіанті розміщення на транспортних засобах або портативних установках для тимчасового моніторингу. Конфігурація та технічні параметри відеокамери обираються з урахуванням конкретних умов експлуатації, зокрема, враховується

необхідність роботи при різному освітленні, включаючи нічний режим, а також можливість функціонування в динамічному середовищі з рухомими об'єктами [7].

Після захоплення зображення система переходить до етапу попередньої обробки отриманого візуального матеріалу, який виступає критично важливою підготовчою фазою, що суттєво впливає на успішність подальшого розпізнавання. На цьому етапі застосовується комплекс алгоритмічних рішень, спрямованих на підвищення якості вхідного зображення та його адаптацію до подальшого аналізу.

Спершу виконується фільтрація різноманітних шумів, які неминуче виникають при цифровій фіксації зображення та можуть бути спричинені недостатнім освітленням, атмосферними явищами або технічними особливостями сенсора камери. Далі здійснюється оптимізація розміру зображення з метою зменшення обчислювального навантаження на систему при збереженні достатньої деталізації для розпізнавання дрібних елементів номерного знаку. Важливим компонентом попередньої обробки є також адаптивне покращення контрасту зображення, що дозволяє підкреслити межі між символами та фоном номерного знаку навіть у складних умовах освітлення.

Наступним кроком виступає перетворення кольорового зображення в градації сірого, що значно спрощує подальший аналіз при збереженні інформативності зображення. Завершальною стадією попередньої обробки є бінаризація – трансформація зображення в чорно-білий формат з адаптивним порогом, що оптимально розділяє символи від фону та мінімізує вплив нерівномірного освітлення або забруднення номерного знаку.

Після завершення попередньої обробки система переходить до найбільш важливого і технологічно складного етапу – виявлення області розташування номерного знаку на загальному зображенні транспортного засобу. Для вирішення цього завдання використовується комбінований підхід, що включає пошук контурів на зображенні з подальшим аналізом їх геометричних характеристик для виділення прямокутних об'єктів з пропорціями, типовими для номерних знаків. Цей базовий метод доповнюється більш точними алгоритмами машинного зору, такими як каскадні класифікатори Haar, які дозволяють ефективно виявляти об'єкти

певного класу на основі їх характерних ознак. У більш складних умовах або для підвищення точності розпізнавання можуть застосовуватися сучасні згорткові нейронні мережі, оптимізовані для функціонування на платформі з обмеженими обчислювальними ресурсами. Такий комбінований підхід забезпечує високу надійність виявлення номерного знаку навіть при частковому загородженні, нестандартних ракурсах зйомки або складних умовах освітлення, при цьому зберігаючи прийнятну швидкодію на апаратній платформі Raspberry Pi.

Наступним етапом процесу є сегментація та розпізнавання окремих символів, розташованих на виявленому номерному знаку. Після виділення області номерного знаку зображення додатково обробляється для забезпечення максимальної чіткості символів, а потім застосовуються спеціалізовані алгоритми сегментації, які дозволяють розділити цілісне зображення номерного знаку на окремі символи – літери та цифри. Для кожного виділеного символу виконується нормалізація розміру та орієнтації, після чого застосовуються методи оптичного розпізнавання символів. У розробленій системі OCR реалізується за допомогою комбінації класичних методів виділення ознак (таких як HOG-дескриптори або LBP) та класифікаторів на основі машинного навчання, спеціально оптимізованих для швидкої роботи на платформі з обмеженими ресурсами. Для підвищення точності розпізнавання додатково враховуються специфічні особливості та обмеження формату державних реєстраційних номерних знаків, такі як фіксований набір допустимих символів та характерна послідовність літер і цифр, що дозволяє суттєво зменшити кількість помилок розпізнавання.

Завершальним етапом технологічного процесу є формування та передача результатів розпізнавання на сервер, реалізований на базі фреймворку Spring Boot. Розпізнаний номерний знак, разом з додатковими метаданими (часова мітка, геолокація, зображення транспортного засобу, рівень достовірності розпізнавання), передається через захищене з'єднання на серверну частину системи. Сервер забезпечує валідацію отриманих даних, їх структуроване збереження у реляційній або NoSQL базі даних, а також надає програмний інтерфейс (англ. Application Programming Interface, API) для інтеграції з іншими системами та сервісами.

Важливою особливістю серверного компонента є підтримка асинхронної взаємодії з пристроями розпізнавання, що дозволяє системі функціонувати навіть в умовах нестабільного мережевого з'єднання – дані можуть тимчасово зберігатися локально на пристрої Raspberry Pi і синхронізуватися з сервером при відновленні зв'язку, що забезпечує безперервність роботи системи в різних умовах експлуатації [7].

Особливо важливою характеристикою розробленої системи є її здатність функціонувати в режимі реального часу, забезпечуючи обробку відеопотоку та розпізнавання номерних знаків з мінімальною затримкою, що критично важливо для багатьох практичних застосувань, таких як контроль доступу на територію або моніторинг транспортного потоку. Досягнення такої продуктивності на платформі з обмеженими обчислювальними ресурсами, якою є Raspberry Pi, потребує комплексного підходу до оптимізації всіх компонентів системи. Цей підхід включає використання ефективних алгоритмів обробки зображень, оптимізацію коду з урахуванням особливостей апаратної архітектури, застосування методів паралельної обробки даних і, за необхідності, квантизацію моделей машинного навчання для зменшення обчислювального навантаження. Крім того, для підвищення продуктивності системи застосовується адаптивний підхід до обробки зображень, при якому інтенсивність обчислень динамічно коригується в залежності від поточного навантаження на систему та вимог до швидкодії в конкретному сценарії використання.

Таким чином, розроблений технологічний процес автоматичного розпізнавання автономерів представляє собою цілісну, гнучку та ефективну систему, що поєднує переваги сучасних технологій комп'ютерного зору та машинного навчання з доступністю та економічністю платформи Raspberry Pi, забезпечуючи оптимальне співвідношення продуктивності, точності та вартості для широкого спектра практичних застосувань у сфері контролю транспортних засобів та управління доступом.

Отже, процес розпізнавання автономерів включає в себе кілька етапів:

- захоплення зображень за допомогою камери, підключеної до Raspberry Pi;
- попередня обробка зображення;

- виявлення області номерного знаку на зображенні через пошук контурів та застосування методів машинного зору;
- сегментація і розпізнавання символів;
- формування та передача результатів розпізнавання на сервер Spring Boot.

Процес розпізнавання номерних знаків складається з послідовних етапів – від захоплення зображення до передачі результатів на сервер. Кожен етап виконує важливу функцію: обробку, виявлення, розпізнавання символів та передачу інформації для подальшої обробки.

1.2 Огляд існуючих рішень системи, що розробляється

Сучасний ринок систем автоматичного розпізнавання державних реєстраційних номерних знаків транспортних засобів представлений широким спектром технологічних рішень різного рівня складності, вартості та функціональних можливостей. Детальний аналіз існуючих розробок виявляє значну різноманітність підходів до вирішення завдання ідентифікації транспортних засобів – від масштабних комерційних продуктів провідних технологічних корпорацій до відкритих програмних бібліотек, що можуть бути використані для створення власних систем.

Однією з найбільш відомих та технологічно розвинених систем у цій галузі є OpenALPR (яка нині функціонує під брендом Rekor Scout). Це рішення завоювало значну популярність завдяки високій точності розпізнавання номерних знаків у різноманітних умовах експлуатації та при різній якості вхідних зображень. Суттєвою перевагою OpenALPR є гнучкість розгортання – система підтримує як локальну інсталяцію на серверах замовника, так і роботу через хмарну інфраструктуру, що дозволяє обирати оптимальний варіант залежно від конкретних потреб та обмежень. Проте, незважаючи на технічну досконалість, це рішення має суттєві недоліки, які обмежують його застосування у багатьох сценаріях. Головними з них є висока вартість ліцензій, що може становити значну перешкоду для малого та середнього бізнесу, а також освітніх та дослідницьких проєктів. Крім того, для забезпечення оптимальної продуктивності OpenALPR потребує доступу

до потужних обчислювальних ресурсів, що значно збільшує сукупну вартість володіння та ускладнює впровадження в умовах обмеженого бюджету або при необхідності розміщення системи на компактних та енергоефективних пристроях [8].

Рисунок 1.1 – Система детекції OpenALPR [12]

Іншим популярним рішенням на ринку є PlateRecognizer – хмарний сервіс, що спеціалізується на розпізнаванні автомобільних номерів з використанням сучасних технологій машинного навчання. Ключова особливість даного сервісу полягає в наданні доступу до функціоналу через чітко структурований REST API, що значно спрощує інтеграцію з існуючими системами та дозволяє швидко реалізувати базові сценарії розпізнавання без глибокого занурення в технічні деталі алгоритмів комп'ютерного зору. PlateRecognizer демонструє високу точність розпізнавання номерних знаків різних країн та форматів, а також забезпечує постійне оновлення моделей для підтримки нових типів номерних знаків. Однак суттєвим недоліком цього рішення є критична залежність від надійного та високошвидкісного Інтернет-з'єднання, оскільки всі обчислення виконуються на віддалених серверах провайдера. Це робить систему вразливою до мережевих збоїв та непридатною для використання в локаціях з обмеженим доступом до мережі або в сценаріях, де критично важлива конфіденційність даних, що не можуть передаватися за межі контрольованої інфраструктури [8].

Рисунок 1.2 – Система детектування PlateRecognizer [13]

Значний інтерес представляє також система Sighthound ALPR, що базується на використанні технологій глибинного навчання для високоточного розпізнавання номерних знаків. Це рішення відрізняється здатністю працювати локально, без залежності від хмарних сервісів, що є важливою перевагою для багатьох замовників. Система забезпечує високу якість розпізнавання навіть у складних умовах – при різному освітленні, різних кутах огляду та наявності часткових перешкод. Проте ключовим обмеженням Sighthound ALPR є його недостатня пристосованість до роботи на вбудованих системах з обмеженими ресурсами.

Рисунок 1.3 – База даних системи Sighthound ALPR [14]

Рисунок 1.4 – Порівняльна характеристика наведених вище систем [4]

Високопродуктивні нейромережеві моделі, що використовуються в цьому рішенні, потребують значних обчислювальних потужностей, що ускладнює їх розгортання на платформах типу Raspberry Pi або інших компактних обчислювальних пристроях, які часто є оптимальним вибором для розподілених систем відеоспостереження та контролю доступу.

Окремий підхід до побудови систем розпізнавання автономерів представлений комбінованим використанням відкритих програмних бібліотек, таких як EasyOCR та OpenCV. Ця стратегія дозволяє розробникам створювати власні рішення з мінімальними фінансовими витратами, маючи повний контроль над архітектурою системи та можливість її гнучкої адаптації під конкретні умови експлуатації. OpenCV надає потужний інструментарій для обробки зображень та базових операцій комп'ютерного зору, в той час як EasyOCR забезпечує високоякісне розпізнавання текстової інформації з використанням сучасних нейромережевих моделей. Однак цей підхід містить і суттєвий виклик – необхідність самостійної адаптації та налаштування алгоритмів для конкретних сценаріїв використання, що вимагає наявності відповідних технічних компетенцій та може значно збільшувати час розробки системи порівняно з готовими комерційними рішеннями [9].

Слід зазначити, що на глобальному ринку систем розпізнавання номерних знаків провідні позиції займають кілька великих технологічних корпорацій. Серед них важливо відзначити NEC Corporation – японську компанію, яка має багаторічний досвід у розробці систем комп'ютерного зору та біометричної ідентифікації. Її рішення для розпізнавання номерних знаків широко використовуються в проєктах «розумних міст» та системах безпеки по всьому світу.

Не менш значущим гравцем на цьому ринку є Bosch Security Systems, що пропонує інтегровані рішення для відеоспостереження з функцією автоматичного розпізнавання номерних знаків, які відрізняються високою надійністю та стійкістю до складних умов експлуатації. Німецька компанія Jenoptik AG спеціалізується на системах контролю дорожнього руху та фіксації порушень правил, де технології розпізнавання номерних знаків відіграють ключову роль.

Ще одним впливовим розробником є Kapsch TrafficCom – австрійська компанія, яка створює комплексні рішення для управління транспортними потоками, включаючи системи електронної оплати проїзду та моніторингу дорожнього руху, що активно використовують технології автоматичної ідентифікації транспортних засобів [10].

Незважаючи на технічну досконалість та високу ефективність комерційних рішень, вони мають ряд спільних обмежень, що перешкоджають їх широкому впровадженню в певних сценаріях використання. Більшість таких систем характеризується закритою архітектурою та обмеженими можливостями для модифікації, що ускладнює їх адаптацію до нестандартних умов експлуатації або інтеграцію з існуючими інформаційними системами замовника.

Висока вартість ліцензій та обладнання робить їх економічно недоцільними для невеликих проєктів або експериментальних розробок, де бюджет суттєво обмежений. Крім того, багато комерційних систем орієнтовані на використання потужних серверних рішень і не оптимізовані для роботи на малопотужних вбудованих платформах, таких як Raspberry Pi, що особливо важливо для створення компактних та енергоефективних пристроїв. Додатковим обмеженням є недостатня

гнучкість цінової політики, коли замовник змушений платити за весь комплекс функцій, навіть якщо для конкретного застосування потрібна лише частина з них [10].

Враховуючи виявлені обмеження існуючих рішень, актуальним та практично значущим завданням залишається розробка власної легкої, економічно доступної та функціонально гнучкої системи для розпізнавання номерних знаків на основі відкритих технологій. Така система повинна забезпечувати достатню точність розпізнавання при мінімальних вимогах до обчислювальних ресурсів, працювати автономно без постійного підключення до мережі Інтернет, мати відкриту архітектуру для легкої інтеграції з існуючими інформаційними системами та бути економічно доступною для широкого кола потенційних користувачів – від дослідницьких лабораторій та стартапів до малих підприємств та громадських організацій.

1.3 Обґрунтування та вибір підходів щодо виконання завдань КБР

Успішність розробки системи автоматичного розпізнавання автономерів значною мірою залежить від правильного вибору технологічного стеку. При формуванні комплексу технологій для проєкту були враховані критично важливі вимоги до фінансової доступності рішення, його здатності функціонувати автономно та обчислювальної ефективності при обмежених ресурсах. Проведений аналіз доступних платформ, бібліотек та інструментів дозволив сформувати оптимальну технологічну базу для реалізації поставлених завдань.

В якості основи апаратної частини системи було обрано компактну одноплатну обчислювальну платформу Raspberry Pi, що пропонує оптимальне співвідношення продуктивності, енергоефективності та вартості. Цей вибір обумовлений кількома ключовими перевагами даної платформи:

– доступність та демократична ціна: Raspberry Pi має суттєво нижчу вартість порівняно з промисловими обчислювальними системами аналогічного призначення, що робить його ідеальним рішенням для проєктів з обмеженим

бюджетом та для створення мереж з великою кількістю розподілених точок розпізнавання;

– розвинена екосистема периферійних пристроїв: платформа забезпечує нативну підтримку різноманітних камер через інтерфейс CSI (Camera Serial Interface), що дозволяє підключати як офіційні модулі Raspberry Pi Camera, так і широкий спектр сумісних відеокамер, включаючи моделі з інфрачервоним режимом для нічної зйомки;

– компактні розміри та низьке енергоспоживання: плата Raspberry Pi може бути встановлена практично в будь-якому місці, включаючи обмежені простори, а її енергоефективність дозволяє працювати від альтернативних джерел живлення (акумулятори, сонячні панелі) у віддалених локаціях;

– достатня обчислювальна потужність: сучасні моделі Raspberry Pi (особливо Raspberry Pi 4 та новіші) оснащені багатоядерними процесорами та достатнім об'ємом оперативної пам'яті для виконання оптимізованих алгоритмів комп'ютерного зору в реальному часі;

– розвинені мережеві можливості: вбудована підтримка Ethernet та Wi-Fi забезпечує гнучкість у розгортанні системи та різноманітні сценарії підключення до мереж передачі даних;

– активна спільнота розробників: значна кількість доступних ресурсів, бібліотек та прикладів коду для Raspberry Pi суттєво прискорює процес розробки та спрощує вирішення технічних проблем.

Для реалізації алгоритмів обробки зображень та виявлення об'єктів обрано бібліотеку OpenCV (Open Source Computer Vision Library), яка є визнаним стандартом у галузі комп'ютерного зору та має ряд переваг для застосування в розроблюваній системі:

– універсальність та функціональна повнота: бібліотека містить понад 2500 оптимізованих алгоритмів для різноманітних задач комп'ютерного зору, обробки зображень та машинного навчання, що дозволяє реалізувати весь технологічний ланцюжок розпізнавання номерних знаків;

- висока продуктивність: алгоритми OpenCV оптимізовані для швидкого виконання, включаючи багатопоточну обробку та апаратне прискорення, що критично важливо для роботи на платформі з обмеженими ресурсами;
- кросплатформеність: бібліотека підтримує різні операційні системи, включаючи Linux, який є основною операційною системою для Raspberry Pi, що забезпечує гнучкість у розгортанні та масштабуванні системи;
- активний розвиток та підтримка: постійні оновлення бібліотеки забезпечують впровадження новітніх алгоритмів та технологій комп'ютерного зору, що дозволяє підтримувати систему на сучасному технологічному рівні;
- інтеграція з популярними мовами програмування: наявність API для Python та Java значно спрощує розробку та забезпечує зручну інтеграцію з іншими компонентами системи;
- вбудована підтримка глибоких нейронних мереж: модуль DNN в OpenCV дозволяє використовувати попередньо навчені моделі для виявлення об'єктів та інших задач комп'ютерного зору, що суттєво підвищує точність розпізнавання.

Для вирішення завдання розпізнавання символів на номерних знаках обрано відкриту систему Tesseract OCR, яка має ряд переваг, що роблять її оптимальним вибором для розроблюваної системи:

- **висока точність розпізнавання**: Tesseract OCR використовує сучасні алгоритми на основі нейронних мереж (LSTM), що забезпечують високу якість розпізнавання символів навіть при складних умовах зйомки;
- **підтримка багатьох мов та символів**: система може ефективно працювати з різними алфавітами та наборами символів, що важливо для розпізнавання номерних знаків різних країн;
- **можливість навчання та адаптації**: Tesseract дозволяє створювати спеціалізовані моделі для розпізнавання конкретних типів шрифтів, що може бути використано для підвищення точності розпізнавання символів на номерних знаках;
- **відкритий вихідний код**: можливість модифікації та оптимізації алгоритмів під конкретні завдання, що особливо важливо при роботі на платформі з обмеженими ресурсами;

– **легка інтеграція з OpenCV**: існують відпрацьовані методики та інструменти для ефективної взаємодії між OpenCV та Tesseract OCR, що забезпечує створення цілісного технологічного ланцюжка розпізнавання.

Для реалізації серверної частини системи, що відповідає за збір, обробку та збереження даних, обрано сучасний фреймворк Spring Boot, який має ряд ключових переваг:

– **швидкість розробки**: фреймворк забезпечує автоматичну конфігурацію, вбудовану підтримку вебсерверів та спрощений процес розгортання, що значно прискорює створення серверної частини системи;

– **висока надійність та масштабованість** – архітектура Spring Boot забезпечує стабільну роботу сервісів під навантаженням та легке горизонтальне масштабування при зростанні обсягів даних;

– **багата функціональність**: фреймворк надає готові компоненти для роботи з базами даних, реалізації REST API, забезпечення безпеки, моніторингу стану системи та інших критично важливих функцій;

– **гнучкість інтеграції**: Spring Boot легко інтегрується з різними базами даних, системами обміну повідомленнями та іншими інформаційними системами, що забезпечує широкі можливості для розширення функціональності;

– **розвинені механізми безпеки**: фреймворк включає компоненти для автентифікації, авторизації та захисту даних, що критично важливо для систем, пов'язаних з ідентифікацією транспортних засобів;

– **активна спільнота розробників**: доступність значної кількості документації, навчальних матеріалів та готових рішень для типових задач суттєво знижує час та витрати на розробку.

Основна причина вибору вказаного комплексу технологій полягає у забезпеченні повної автономності функціонування системи без необхідності постійного з'єднання з мережею Інтернет або залежності від сторонніх хмарних сервісів. Така автономність є критично важливою для багатьох сценаріїв використання. Як приклад робота в локаціях з обмеженим доступом до мережі Інтернет, або віддалені об'єкти чи підземні паркінги. Забезпечується

конфіденційність даних – коли обробка інформації повинна відбуватися виключно в межах власної інфраструктури організації без передачі даних зовнішнім сервісам. Мінімізується затримка при розпізнаванні – локальна обробка даних забезпечує мінімальний час відгуку системи, що критично важливо для систем контролю доступу в реальному часі. Знижується сукупна вартість володіння – відсутня потреба в оплаті підписок на хмарні сервіси розпізнавання суттєво знижує витрати при довгостроковій експлуатації системи

Крім того, вибраний стек технологій забезпечує повну відкритість та гнучкість системи, що дозволяє адаптувати її до різноманітних вимог та специфічних умов експлуатації без необхідності значних змін в архітектурі або повторної розробки ключових компонентів.

Таким чином, комбінація Raspberry Pi, OpenCV, Tesseract OCR та Spring Boot створює оптимальну технологічну основу для розробки доступної, автономної та ефективної системи автоматичного розпізнавання державних реєстраційних номерних знаків, що відповідає сучасним вимогам до таких систем з урахуванням обмежень бюджету та обчислювальних ресурсів.

1.4 Формування вимог до апаратно-програмного забезпечення

Для ефективної реалізації системи автоматичного розпізнавання номерних знаків транспортних засобів необхідно чітко визначити функціональні та нефункціональні вимоги до апаратно-програмного забезпечення. Це дозволить забезпечити високу якість роботи системи, її стабільність, а також зручність у використанні та обслуговуванні.

Функціональні вимоги:

- інтеграція камери з Raspberry Pi: система має забезпечувати підключення цифрової камери до мікрокомп’ютера Raspberry Pi та можливість керування нею, зокрема ініціалізація зйомки, отримання кадрів та налаштування параметрів (яскравість, контраст, фокус тощо);
- попередня обробка зображень: перед застосуванням алгоритмів розпізнавання, система повинна здійснювати обробку кадрів для покращення їх

якості. Зокрема, йдеться про зменшення цифрового шуму, підвищення чіткості та контрастності, а також інші методи покращення видимості номерного знаку в кадрі;

– виявлення номерного знаку: алгоритм повинен вміти автоматично знаходити номерні знаки на отриманих зображеннях у реальному часі, незалежно від положення автомобіля в кадрі, освітлення чи фону;

– розпізнавання символів: після локалізації номерного знаку система має здійснювати розпізнавання символів за допомогою технології OCR. Результатом має бути текстовий рядок, що відповідає державному номерному знаку;

– передача даних на сервер: отримана інформація про номерний знак повинна передаватися на серверну частину системи через REST API. Передача повинна включати як сам текст номеру, так і, за потреби, зображення кадру та мітку часу;

– збереження інформації в базі даних: серверна частина повинна зберігати розпізнану інформацію у базі даних. Кожен запис повинен містити номерний знак, дату та час розпізнавання, а також інші метадані (наприклад, місце розташування або ідентифікатор камери);

– вебінтерфейс для перегляду даних: має бути реалізовано інтуїтивно зрозумілий вебінтерфейс, який дозволить авторизованим користувачам переглядати історію розпізнань, здійснювати пошук за номером чи датою, а також мати доступ до фотофіксацій.

Нефункціональні вимоги

– швидкість обробки кадру: час повного циклу обробки одного зображення (від отримання кадру до збереження даних на сервері) не повинен перевищувати 2 секунд, щоб забезпечити можливість роботи в реальному часі;

– стабільність у температурному діапазоні: уся система (камери, Raspberry Pi, блок живлення) повинна функціонувати без збоїв у температурному діапазоні від -15 до 50 °C, що дозволить використовувати її в різних кліматичних умовах;

– працездатність при низькому освітленні: система повинна коректно працювати за умов слабкого освітлення (вночі, у сутінках або в похмуру погоду). Це може потребувати додаткового ІЧ-підсвічування або налаштувань камери;

– автономність роботи: система має мати можливість працювати без підключення до Інтернету протягом не менше ніж 12 годин. Це передбачає використання локального збереження даних з подальшою синхронізацією після відновлення з’єднання;

– захищений доступ до даних: доступ до збережених даних та конфігурацій серверної частини повинен бути обмеженим і захищеним. Для цього потрібно впровадити механізми автентифікації користувачів, шифрування даних при передачі та зберіганні, а також логування дій користувачів.

Система автоматичного розпізнавання номерних знаків повинна забезпечувати повний цикл обробки – від захоплення зображень до збереження та перегляду результатів. Важливими аспектами є якісна обробка кадрів, точне розпізнавання символів, стабільна передача даних і надійний захист інформації. Усі функціональні й нефункціональні вимоги мають бути враховані для ефективної роботи системи в реальному часі та різних умовах.

Висновки до 1-го розділу

У першому розділі було здійснено всебічний системний аналіз предметної області, що стосується автоматичного розпізнавання номерних знаків транспортних засобів із використанням недорогих апаратних платформ та відкритого програмного забезпечення.

Розкрито об’єкт і предмет дослідження: об'єктом є процес ідентифікації транспортних засобів, а предметом – апаратно-програмна система для автоматичного детектування та розпізнавання номерних знаків на базі Raspberry Pi, OpenCV та Spring Boot.

Було розглянуто основні структурні та функціональні особливості процесу розпізнавання номерів, зокрема: попередню обробку зображення, локалізацію номерного знаку, сегментацію символів та їхню класифікацію. Зазначено, що при

побудові таких систем важливими аспектами є якість зображення, стійкість алгоритмів до зовнішніх умов (освітлення, погодні умови, забруднення номерів) та швидкість обробки даних.

У ході огляду існуючих рішень було встановлено, що на ринку присутні як комерційні дорогі рішення (наприклад, від компаній Hikvision, Axis, Bosch), так і відкриті проєкти, орієнтовані на дослідників та розробників (наприклад, OpenALPR, EasyOCR). Проте більшість готових рішень вимагають або високопродуктивних серверів, або значних фінансових витрат. Це підтверджує доцільність створення власної системи на основі доступного обладнання.

Таким чином, перший розділ заклав фундаментальні теоретичні й практичні основи для подальшої розробки системи детектування автономерів. Отримані результати визначили чіткий напрям подальших досліджень та проєктно-конструкторських робіт, спрямованих на створення ефективного та доступного рішення для задач автоматичного розпізнавання номерних знаків.

2 МАТЕМАТИЧНІ МЕТОДИ, МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ СИСТЕМИ ДЕТЕКТУВАННЯ НОМЕРІВ

У сучасному світі автоматизовані системи розпізнавання номерних знаків транспортних засобів (англ. Automatic License Plate Recognition, ALPR) набувають все більшого значення в контексті інтелектуальних транспортних систем, систем безпеки та контролю доступу. Даний розділ присвячено детальному опису математичного апарату, методів комп'ютерного зору та підходів до проєктування системи автоматичного розпізнавання номерних знаків на основі мініатюрної обчислювальної платформи Raspberry Pi із застосуванням бібліотеки комп'ютерного зору OpenCV та фреймворку Spring Boot для реалізації серверної частини.

Система, що розробляється, представляє собою інтегрований програмно-апаратний комплекс, здатний функціонувати в режимі реального часу, детектувати та розпізнавати автомобільні номерні знаки з подальшою можливістю використання цієї інформації в різноманітних прикладних задачах. Варто відзначити, що розробка такої системи вимагає глибокого розуміння фундаментальних принципів цифрової обробки зображень, комп'ютерного зору та машинного навчання, а також практичних аспектів реалізації подібних систем на апаратних платформах з обмеженими обчислювальними ресурсами.

2.1 Постановка завдання

Головною метою даної роботи є створення ефективної та надійної системи автоматичного розпізнавання номерних знаків, яка має функціонувати на платформі Raspberry Pi. З урахуванням специфіки досліджуваної проблеми та технічних обмежень обраної платформи, визначено такі функціональні вимоги до системи:

– детектування номерного знаку на зображенні: система повинна автоматично виявляти наявність автомобільного номерного знаку на вхідному зображенні, отриманому з камери або з файлу;

– локалізація та сегментація номерного знаку: після виявлення номерного знаку система має виділити його як окрему область інтересу (англ. Region of Interest, ROI) для подальшої обробки;

– попередня обробка зображення: застосування комплексу методів цифрової обробки зображень для покращення якості виділеного сегменту з номерним знаком, мінімізації шумів, корекції геометричних спотворень та підвищення контрастності;

– оптичне розпізнавання символів (OCR): перетворення графічного представлення номерного знаку в текстовий формат з високою точністю розпізнавання;

– інтеграція з вебінтерфейсом та API: забезпечення можливості передачі результатів розпізнавання в зручному форматі для їх подальшого використання в прикладних системах;

– функціонування в умовах обмежених обчислювальних ресурсів: оптимізація алгоритмів для забезпечення прийнятної швидкодії на апаратній платформі Raspberry Pi.

Ефективність вирішення поставленої задачі оцінюватиметься за різними критеріями, а саме:

– точність детектування номерних знаків на зображенні;

– точність розпізнавання символів номерного знаку;

– швидкість обробки одного зображення;

– стійкість системи до зміни умов освітлення, кута нахилу камери та забруднення номерного знаку.

Якщо буде дотримано всі вище перелічені критерії, то точність поставленої задачі буде високою та ефективною.

2.2 Математичні методи та підходи

Основою будь-якої системи комп'ютерного зору є методи цифрової обробки зображень, які забезпечують підготовку візуальної інформації для подальшого аналізу. У контексті розпізнавання номерних знаків цифрова обробка зображень

набуває особливого значення, оскільки якість вхідного зображення безпосередньо впливає на точність розпізнавання. Розглянемо детально математичні методи, що застосовуються на різних етапах обробки зображення.

2.2.1 Методи обробки зображень

Першим і надзвичайно важливим етапом обробки є зменшення шумів у зображенні за допомогою фільтра Гауса. Цей метод базується на згортці зображення з ядром, що має форму двовимірного нормального розподілу. Математично фільтр Гауса описується формулою [15]:

$$G(x,y) = \frac{\frac{1}{(2p\sigma^2)} \times e^{-(x^2+y^2)}}{2\sigma^2}, \quad (2.1)$$

де σ – стандартне відхилення розподілу Гауса, що контролює ступінь розмиття.

Фільтр Гауса ефективно видаляє високочастотні компоненти зображення, що відповідають шумам, одночасно зберігаючи важливі структурні особливості. Це критично важливо для подальшого аналізу, оскільки шуми можуть значно погіршити якість виявлення контурів та розпізнавання символів. При застосуванні фільтра відбувається усереднення значень пікселів з їх околицями, причому вага сусідніх пікселів зменшується з відстанню згідно з розподілом Гауса.

Після фільтрації шумів зображення конвертується у відтінки сірого. Цей процес передбачає перетворення кольорового зображення (зазвичай у форматі RGB) в одноканальне зображення, де кожен піксель представлений одним значенням інтенсивності. Стандартна формула для конвертації [16]:

$$Y = 0.299R + 0.587G + 0.114B, \quad (2.2)$$

де R, G, B – значення червоного, зеленого та синього каналів відповідно;

Y – результуюча інтенсивність сірого.

Таке перетворення дозволяє суттєво зменшити обчислювальну складність подальших операцій (оскільки обробляється один канал замість трьох) без втрати ключової інформації про структурні елементи зображення. Крім того, більшість алгоритмів виявлення країв та контурів оптимізовані саме для роботи з одноканальними зображеннями.

Наступним етапом є порогова обробка, яка переводить зображення у бінарний формат, де кожен піксель може приймати лише два значення: 0 (чорний) або 255 (білий). Процес базується на порівнянні значення інтенсивності кожного пікселя з певним пороговим значенням T.

Існує декілька методів визначення порогового значення T, серед яких найбільш поширеними є:

- глобальне порогове значення (фіксоване для всього зображення);
- адаптивне порогове значення (різне для різних частин зображення);
- метод Оцу, що автоматично обирає оптимальне порогове значення шляхом мінімізації дисперсії між двома класами пікселів.

Бінаризація має критичне значення для розпізнавання номерних знаків, оскільки вона дозволяє чітко відокремити символи номерного знака від фону, що спрощує подальше розпізнавання.

Після бінаризації застосовується **оператор Кані (Canny)**, що представляє собою багатоетапний алгоритм виявлення країв на зображенні. Цей метод включає:

- застосування фільтра Гауса для зменшення шумів;
- обчислення градієнтів інтенсивності по горизонталі та вертикалі (зазвичай за допомогою оператора Собеля);
- визначення величини та напрямку градієнта;
- прорідження немаксимумів для отримання тонких країв;
- порогову обробку з гістерезисом для видалення несуттєвих країв.

Математично, величина градієнта у кожній точці обчислюється як [17]:

$$|G| = \sqrt{(Gx^2 + Gy^2}\,, \qquad (2.3)$$

де Gx та Gy – градієнти по горизонталі та вертикалі.

Оператор Кані забезпечує високу точність виявлення країв при відносно низькій чутливості до шумів. Виявлені краї об'єктів формують основу для подальшого пошуку контурів номерних знаків.

На завершальному етапі обробки зображення система здійснює **пошук контурів**. Алгоритм findContours аналізує бінарне зображення та виявляє замкнуті області, що відповідають певним геометричним характеристикам. Для виявлення

номерних знаків особлива увага приділяється контурам, що мають прямокутну форму з певним співвідношенням сторін, характерним для номерних знаків транспортних засобів.

Процес пошуку контурів включає:

– сканування зображення для виявлення зовнішніх контурів (послідовностей пікселів, що утворюють межу об'єкта);

– аналіз геометричних властивостей контурів (площа, периметр, співвідношення сторін);

– фільтрацію контурів на основі заданих критеріїв (мінімальна та максимальна площа, співвідношення сторін тощо);

– апроксимацію (наближене вираження будь-яких величин через інші, відоміші або простіші величини) контурів геометричними примітивами (наприклад, прямокутниками).

Знайдені контури, що відповідають заданим критеріям, розглядаються як потенційні області номерних знаків і передаються на подальшу обробку для сегментації окремих символів та їх розпізнавання.

Усі описані методи обробки зображень є реалізацією класичних підходів комп'ютерного зору, де основними математичними принципами виступають градієнтний аналіз, фільтрація, морфологічні перетворення та геометричне виявлення об'єктів. Їх послідовне застосування дозволяє ефективно виділити номерні знаки транспортних засобів на складних зображеннях з різними умовами освітлення, кутами зйомки та присутністю сторонніх об'єктів.

2.3 Оптичне розпізнавання символів (OCR)

Після того як система виявляє і локалізує область, що містить номерний знак, наступним критичним етапом є оптичне розпізнавання символів (OCR) – перетворення графічного зображення символів у машинозчитуваний текст. Для цього у розробці використовується одна з найпотужніших і найбільш поширених відкритих бібліотек – Tesseract OCR. Вона має відкритий вихідний код, активно

підтримується спільнотою, і включає у свій склад інструменти попередньої обробки, сегментації та глибокого розпізнавання символів.

Tesseract спочатку виконує попередню обробку зображення, що включає нормалізацію контрасту, зменшення шумів, адаптивну бінаризацію, вирівнювання текстових рядків і корекцію перекосу (deskewing). Ці кроки мають за мету покращити якість вхідного зображення, щоб зменшити ймовірність помилок при розпізнаванні символів.

Далі система переходить до сегментації символів – процесу виділення окремих символів з суцільного зображення текстової області. У випадку номерних знаків це особливо критично, оскільки символи часто мають фіксовану ширину, але можуть бути розміщені близько один до одного або містити дефекти (подряпини, засвітки, перекриття).

Після сегментації запускається етап безпосереднього розпізнавання. У Tesseract OCR реалізовано гібридний підхід: поєднання згорткових нейронних мереж для екстракції ознак зображення символів і рекурентних нейронних мереж зокрема Long Short-Term Memory, для обробки послідовностей символів. Цей підхід дозволяє враховувати не лише форму окремого символа, а й його контекстне розташування у рядку, тобто попередній і наступний символи, що значно підвищує точність розпізнавання. Наприклад, якщо модель розпізнає символи «AA1234BX», вона може використати мовну модель (англ. language model), яка враховує типові шаблони номерних знаків в Україні – це допомагає зменшити кількість помилок, спричинених шумами або артефактами.

Математично цей процес описується як задача багатокласової класифікації, де кожен вхід (зображення символу) класифікується в один із заздалегідь визначених класів (цифри, літери тощо). Згорткові шари відповідають за виявлення просторових закономірностей у зображенні, а рекурентні – за аналіз послідовності символів у рядку.

Окрім цього, Tesseract використовує вбудовану мовну модель, яка після початкового розпізнавання перевіряє, наскільки отримана послідовність символів відповідає допустимим словам або шаблонам. Наприклад, якщо розпізнано

«AA1234BK», система може відкинути альтернативи, які не підпадають під типову структуру українського номерного знака, навіть якщо деякі символи розпізнані з низькою впевненістю [10].

В результаті поєднання класичних підходів комп’ютерного зору, глибокого навчання та мовного контексту, Tesseract OCR забезпечує достатню точність для розпізнавання номерних знаків в умовах реального світу: на зображеннях з різною якістю, освітленням, кутом нахилу та шумами. Це робить його оптимальним вибором для інтеграції в апаратно-програмну систему автоматичного розпізнавання номерів.

У підсумку, для розпізнавання номерного знака використовується бібліотека Tesseract OCR, яка поєднує методи попередньої обробки зображення, сегментацію символів та глибоке навчання. Основу розпізнавання складають згорткові нейронні мережі для виділення ознак та рекурентні мережі для аналізу послідовності символів. Також застосовується мовна модель, що враховує структуру номерних знаків, що значно підвищує точність. Цей підхід дозволяє ефективно розпізнавати навіть складні чи зіпсовані зображення номерів у реальних умовах.

2.4 Логіка роботи системи та апаратно-програмна реалізація

Для опису логіки функціонування системи автоматичного розпізнавання номерних знаків було розроблено блок-схему алгоритму, яка відображає основні етапи – від захоплення зображення до передавання результату на сервер.

Основні етапи роботи системи (рис. 2.1):

- захоплення зображення з камери: перший етап передбачає отримання зображення з камери, підключеної до Raspberry Pi. Для цього використовується камера типу Raspberry Pi Camera v2 або сумісна USB-камера. Захоплення кадрів відбувається у реальному часі за допомогою бібліотеки OpenCV, яка забезпечує інтерфейс до пристроїв відеозахоплення.
- попередня обробка зображення: отримане зображення проходить етап фільтрації та нормалізації;

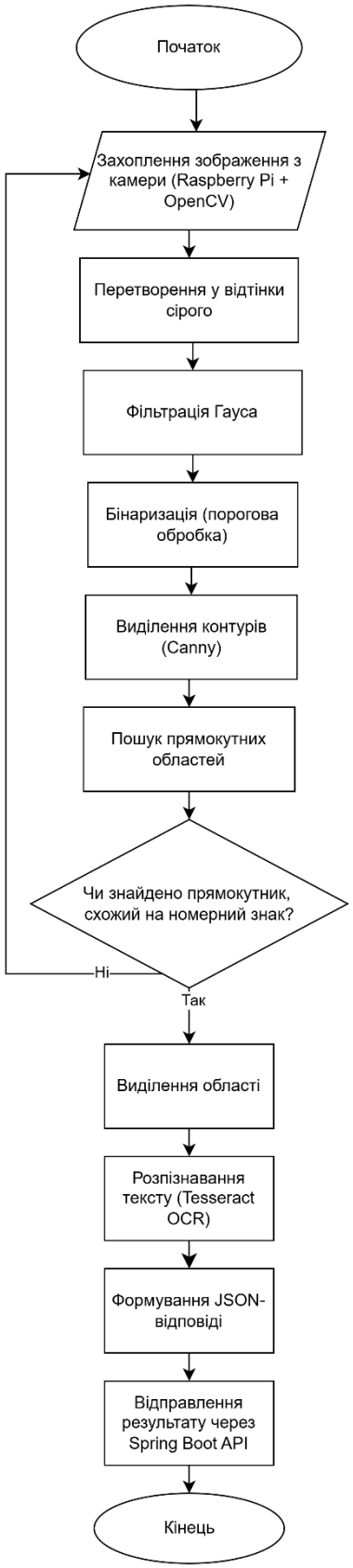

Рисунок 2.1 – Блок-схема алгоритму розпізнавання номерного знаку

– виділення області номерного знака: із знайдених контурів вибирається той, що відповідає геометричним ознакам номерного знака (прямокутна форма, співвідношення сторін). Ця область обрізається та готується до OCR;

– оптичне розпізнавання символів (OCR): виділена область обробляється за допомогою бібліотеки Tesseract OCR, яка використовує нейромережеву архітектуру (CNN + LSTM) для точного розпізнавання символів у заданій послідовності. Мовна модель дозволяє розпізнати як окремі символи, так і цілий номерний знак у форматі, наприклад, AA1234BB.

– передача результату на сервер: отриманий текст номерного знака передається на серверну частину, реалізовану на фреймворку Spring Boot. Система реалізована у вигляді RESTful API, яке дозволяє отримати результат у форматі JSON.

Апаратна платформа

Основу системи складає Raspberry Pi 4 Model B, яка забезпечує достатню обчислювальну потужність для обробки зображень і запуску моделей OCR. Завдяки низькому енергоспоживанню, компактності та можливості підключення різних пристроїв, Raspberry Pi виступає ідеальним вибором для вбудованих систем технічного зору.

Переваги апаратної реалізації:

– робота в режимі реального часу без підключення до хмари;

– відсутність необхідності в зовнішньому сервері;

– можливість розміщення в автономному середовищі (шлагбауми, стоянки, КПП тощо).

Блок-схема наочно відображає логіку роботи системи: від захоплення зображення камерою до розпізнавання номерного знаку та передачі результату через API. Такий підхід дозволяє структуровано представити алгоритм, виділити ключові етапи обробки зображення та забезпечити інтеграцію з іншими компонентами системи.

Під час проєктування системи автоматичного розпізнавання номерних знаків одним із ключових етапів є правильний вибір апаратних компонентів. Від їхньої

якості, потужності та взаємодії залежить як точність, так і стабільність роботи пристрою. Розглянемо основні складові системи, які були обрані, а також порівняємо їх із можливими аналогами.

Обчислювальний модуль: Raspberry Pi 4 Model B

Рисунок 2.2 – Raspberry Pi 4 Model B

Характеристики:

- процесор: 4-ядерний Cortex-A72 @1.5GHz;
- оперативна пам'ять: 4 GB RAM (можна обрати й 8GB);
- виходи: HDMI, USB 3.0, USB-C (живлення), GPIO;
- підтримка камери через CSI, SD-карти до 256GB.

Raspberry Pi 4 є компактною, енергоефективною, недорогою платформою, що ідеально підходить для комп’ютерного зору. Завдяки потужному ARM-процесору вона здатна виконувати одночасно завдання з обробки зображень (OpenCV) та передачі даних через мережу (REST API, Wi-Fi, Ethernet).

Порівняння з аналогами: Jetson Nano та Orange Pi 5

Рисунок 2.2 – Jetson Nano

Характеристики Jetson Nano:

- центральний процесор: чотириядерний arm a57;
- графічний процесор: gpu 128-ядерний графічний процесор maxwell™;
- оперативна пам'ять: 4 гб, 64-бітна lpddr4 25,6 гб/с;
- мережеве підключення: 10/100/1000base-t ethernet.;
- пам'ять програм: microsd (не входить у комплект);
- ціна 4000 грн.

Raspberry Pi 4 дешевший, енергоефективніший і має значно ширшу спільноту підтримки для швидкої розробки.

Характеристики Orange Pi 5:

- soc: rockchip rk3588s (процес на 8 нм);
- процесор: 8-ядерний 64-бітний процесор;
- велике ядро має частоту 2.4 ггц, а мале ядро - 1.8 ггц;
- відеокарта: arm mali-g610 mp4 "odin" gpu;
- 3d-графічний двигун та 2d-графічний двигун.

Рисунок 2.2 – Orange Pi 5

Raspberry Pi 4 стабільніший у роботі з периферією, має офіційну документацію та краще підтримується з боку бібліотек для розпізнавання номерів.

Використана камера: USB-камера EMIX

Рисунок 2.3 – USB-камера EMIX

Альтернатива: RPi Camera Module v2.1

Рисунок 2.4 – RPi Camera Module v2.1

Порівняння: USB-камера має просте підключення, але більшу затримку та нижчу стабільність фреймрейту. RPi Camera використовує високошвидкісний інтерфейс CSI, підтримує захоплення відео з мінімальною затримкою.

Живлення: Блок живлення 5V 3A + стабілізатор живлення

Raspberry Pi вимагає стабільного живлення, особливо при підключенні камер та модулів.

Вибраний блок живлення: 5V 3A USB-C адаптер

Рисунок 2.5 – 5V 3A USB-C адаптер

Альтернатива**:** використання PowerBank для віддалених інсталяцій.

RTC-модуль: DS3231

Перевага обраного блоку живлення полягає в його стабільній напрузі та достатній потужності, що забезпечує надійну роботу навіть при підключенні декількох периферійних пристроїв, включно з камерою, модулем RTC та активним охолодженням.

Модуль реального часу: DS3231

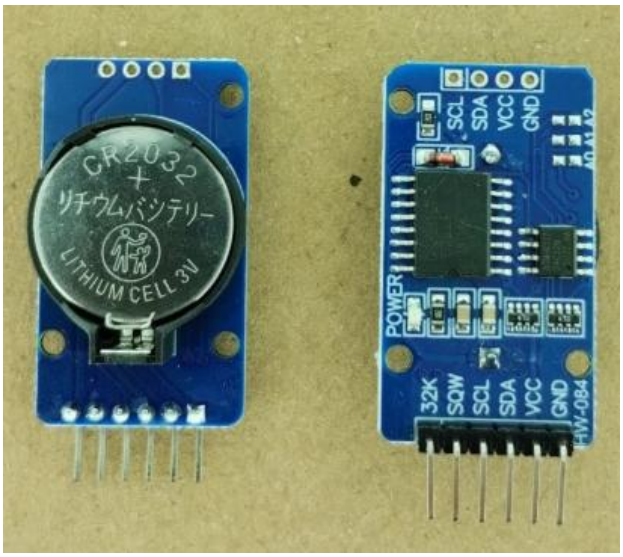

Рисунок 2.6 – DS3231

Функція: забезпечення точного зчитування часу навіть при відключенні живлення.

Переваги DS3231:

- висока точність (до ±2 ppm);
- вбудована батарейка (VBAT);
- протокол I2C — легка інтеграція з Raspberry Pi;

Альтернативи: DS1307

Характеристики DS1307:

- точність: ±2 хвилини/місяць (без термокомпенсації);
- живлення: 5V, потрібен зовнішній кварц;
- не має термокомпенсації, тому точність сильно залежить від температури;

– потребує зовнішнього кристалу, який збільшує нестабільність.

Рисунок 2.7 – DS1307

Чому гірше: Менша точність та чутливість до температури в порівнянні з DS3231.

Корпус та охолодження:

Рисунок 2.8 – Корпус

Для забезпечення роботи в умовах від -15°C до +50°C використовується металевий корпус із вентиляційними отворами, модульний кулер або пасивний радіатор (якщо очікується перегрів).

Висновки до розділу 2

У другому розділі було розглянуто математичні методи, алгоритми та засоби проєктування, необхідні для реалізації системи автоматичного розпізнавання номерних знаків. Було визначено ключові етапи обробки зображення: фільтрація, сегментація, виділення контурів, а також методи розпізнавання тексту з використанням OCR. Особливу увагу приділено бібліотеці Tesseract, яка реалізує глибокі нейронні мережі для точного розпізнавання символів.

Крім того, розглянуто особливості попередньої обробки зображень, які дозволяють покращити контрастність, зменшити шум і підвищити якість символів, що надходять на вхід OCR. У роботі було експериментально підібрано параметри алгоритмів порогового фільтрування та морфологічних операцій для досягнення найкращих результатів при різних умовах освітлення.

Також описано архітектуру системи та реалізацію алгоритмів на апаратній платформі Raspberry Pi, що забезпечує автономність, компактність і енергоефективність рішення. Для обробки зображень використано бібліотеку OpenCV, яка забезпечує широкий спектр функцій комп’ютерного зору. Серверна частина системи побудована на базі Spring Boot, що дозволяє не лише передавати результати розпізнавання у зовнішні системи за допомогою REST API, а й масштабувати рішення для роботи в багатокомпонентному середовищі.

Розроблена блок-схема чітко ілюструє логіку роботи системи від моменту отримання зображення до передачі результату, що забезпечує структурований підхід до реалізації проєкту. Додатково, було проаналізовано можливості розширення функціоналу системи для інтеграції з базами даних, системами контролю доступу та смарт-відеонаглядом.

Отже, представлена методика створення системи є цілісною, обґрунтованою та придатною для впровадження в реальних умовах, особливо в умовах, де необхідна автоматизація процесу ідентифікації транспортних засобів.

3 АПАРАТНО-ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ТА ТЕСТУВАННЯ

Під час створення системи автоматичного розпізнавання номерних знаків транспортних засобів було розроблено структурну архітектуру апаратно-програмного комплексу, що дозволяє забезпечити ефективне захоплення зображень, їх обробку, розпізнавання номерів та збереження результатів.

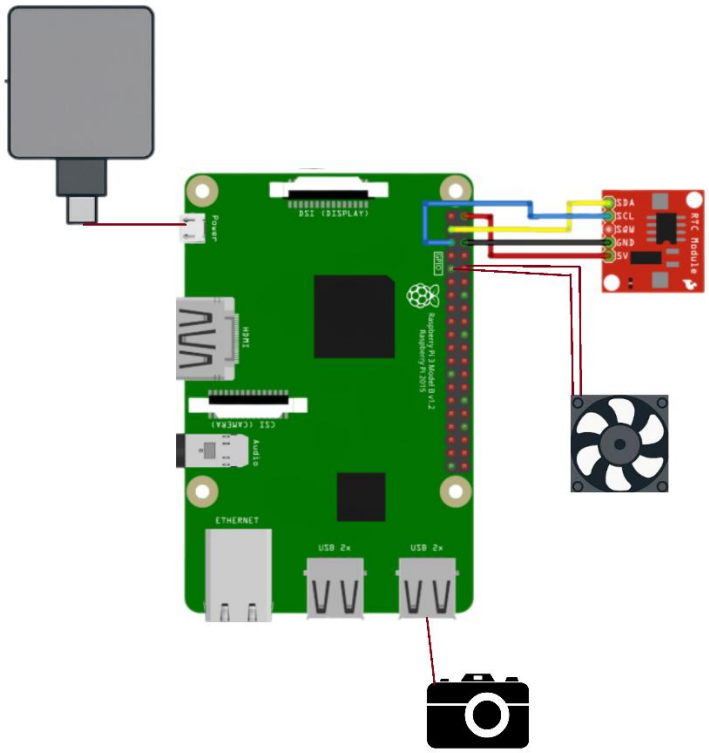

Рисунок 3.1 – Схема підключення пристрою

Рисунок 3.1 – Підключення камери до пристрою

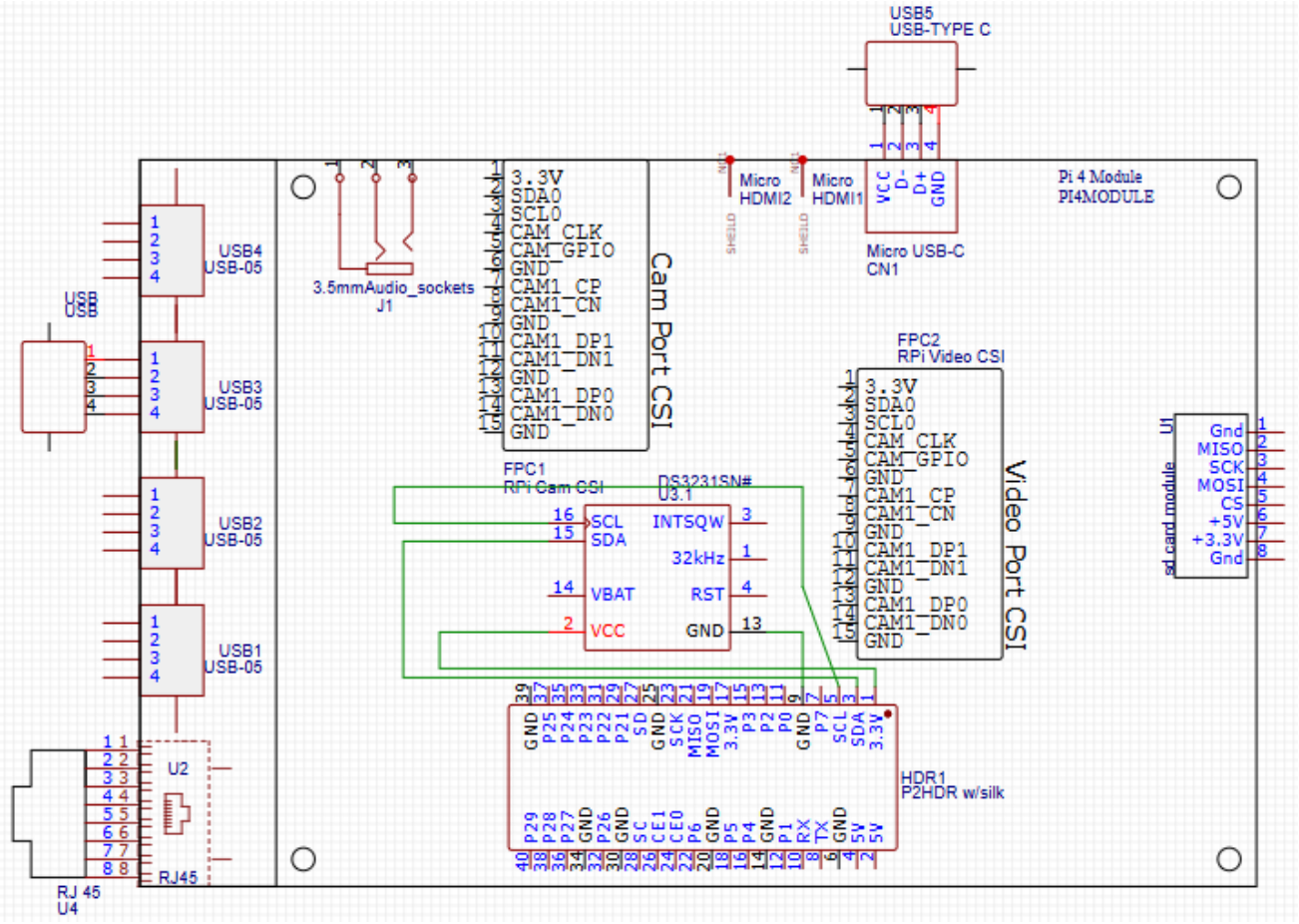

Рисунок 3.2 – Макет системи керування

Архітектура системи орієнтована на модульність, що дозволяє за необхідності замінювати або модернізувати окремі її компоненти без впливу на загальну працездатність.

3.1 Розробка архітектури апаратно-програмного забезпечення

Загалом архітектуру можна поділити на дві основні частини – апаратну та програмну. Кожна з них виконує чітко визначені функції, взаємодіє через стандартизовані інтерфейси та є критично важливою для забезпечення надійної роботи всієї системи.

Апаратна частина

До складу апаратної частини входять такі основні елементи:

– камера EMIX, яка використовується для фіксації транспортних засобів у контрольованій зоні. Камера забезпечує достатню роздільну здатність та частоту

кадрів для коректного розпізнавання номерного знаку незалежно від погодних умов або часу доби;

– обчислювальний пристрій, а саме вбудований мінікомп’ютер типу Raspberry Pi 4 Model B, на якому безпосередньо виконується обробка зображень і програмна частина системи;

– блок живлення, що забезпечує стабільне електроживлення камери та обчислювального пристрою.

– камера підключається до обчислювального пристрою за допомогою інтерфейсу USB. Для забезпечення безперервної фіксації подій використовується режим постійної зйомки з буфером кадрів.

Програмна частина

Програмне забезпечення розроблено з використанням мови програмування **Python**, яка обрана завдяки її гнучкості, наявності великої кількості бібліотек для комп’ютерного зору та машинного навчання, а також активної спільноти.

Структурно програмна частина складається з таких основних модулів:

– модуль захоплення відеопотоку, який відповідає за отримання зображень з камери у реальному часі;

– модуль попередньої обробки зображення – виконує масштабування, фільтрацію шумів, підвищення контрасту та перетворення в чорно-біле зображення;

– модуль виявлення номерного знаку – здійснює локалізацію області номерного знаку за допомогою методів сегментації та алгоритмів, зокрема YOLO;

– модуль оптичного розпізнавання символів (OCR), який виконує аналіз символів на виявленому номерному знаку. У роботі використано бібліотеку Tesseract OCR від Google;

– модуль збереження результатів – відповідає за запис розпізнаних номерів у базу даних або файл, а також фіксацію дати та часу.

Кожен модуль взаємодіє з іншими через внутрішні API, що забезпечує незалежність компонентів і полегшує тестування.

Архітектурна модель

Система реалізована за принципом **клієнт-серверної архітектури**, де камера виступає в ролі джерела даних, а програмне забезпечення обчислювального пристрою виконує роль сервера, який обробляє і зберігає дані.

У якості схеми обміну між модулями використано стандарт передачі зображень у вигляді потоків (OpenCV VideoCapture) та локальних функціональних викликів.

Окремо розроблено логіку взаємодії з користувачем у вигляді графічного інтерфейсу, створеного за допомогою бібліотеки Tkinter.

Інтерфейс дозволяє:

- запускати та зупиняти обробку відеопотоку;
- переглядати останнє розпізнане зображення з номером;
- експортувати розпізнані номери;
- переглядати журнал збережених результатів.

Розроблена архітектура забезпечує масштабованість рішення, дає змогу адаптувати його під різні сценарії (наприклад, контроль в’їзду на парковку, ведення обліку транспорту на підприємстві тощо) та інтегрувати з іншими інформаційними системами.

3.2 Технології та мови програмування

При створенні системи автоматичного розпізнавання номерних знаків транспортних засобів було прийнято рішення розділити функціональність на два основні компоненти – клієнтську частину, що займається комп’ютерним зором, і серверну частину, яка обробляє запити, зберігає дані та надає інтерфейс для взаємодії з користувачем. Такий підхід дозволяє оптимізувати ресурси системи, спростити масштабування та поліпшити підтримку коду.

Мова програмування Python

Для реалізації клієнтської частини, яка виконує функції детекції номерних знаків, попередньої обробки зображень та оптичного розпізнавання символів (OCR), обрано мову програмування Python. Основними причинами цього вибору є:

- наявність великої кількості бібліотек для роботи з зображеннями та відео;
- зручність реалізації прототипів алгоритмів комп’ютерного зору;
- сумісність з апаратними платформами на базі ARM
- активна спільнота та регулярна підтримка бібліотек.

Основними використаними бібліотеками Python у проєкті є:

- **OpenCV** для захоплення відеопотоку, обробки зображень, виділення контурів та підготовки кадрів до OCR;
- **Tesseract OCR** для розпізнавання символів номерного знаку. Використовується через обгортку `pytesseract`;
- **NumPy** для роботи з масивами та оптимізації математичних операцій;
- **Tkinter** для створення простого графічного інтерфейсу на стороні Raspberry Pi, що дозволяє запускати обробку та переглядати результати.

Захист від температур:

Підігрів:

- плівковий нагрівач 12V, 5–10W (з термореле);
- термостат-реле (W1209) – керує включенням підігрівача.

Охолодження:

- алюмінієвий радіатор для CPU;
- термокерований вентилятор Sunon;
- термопаста.

Програма, написана на Python, виконується безпосередньо на Raspberry Pi 4, що дозволяє отримувати відео з камери, обробляти зображення та надсилати результат до центрального сервера.

Захоплення зображення з камери (USB / CSI)

Функція: Отримання кадру в реальному часі.

```
import cv2

cap = cv2.VideoCapture(0)  # 0 – перша підключена USB-камера
ret, frame = cap.read()
if ret:
    cv2.imwrite("car.jpg", frame)
cap.release()
```

Попередня обробка зображення

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (5, 5), 0)
thresh = cv2.adaptiveThreshold(blurred, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                               cv2.THRESH_BINARY_INV, 11, 2)
```

Виділення контурів та знаходження номерної області

```
contours, _ = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
for c in contours:
    x, y, w, h = cv2.boundingRect(c)
    aspect_ratio = w / h
    if 2 < aspect_ratio < 6:  # типові пропорції номерного знаку
        plate = frame[y:y+h, x:x+w]
        break
```

Функція: Фільтрація шуму, перетворення у відтінки сірого, підвищення контрасту.

OCR розпізнавання символів

Функція: Перетворення зображення номерного знаку у текст.

```
config = '--psm 8 --oem 3 -c tessedit_char_whitelist=ABCEHKMOPTX1234567890'
text = pytesseract.image_to_string(plate, config=config)
print("Розпізнаний номер:", text)
```

Відправка даних на сервер (REST API)

Функція: Передача тексту, зображення та часу фіксації на сервер Spring Boot.

```
import requests
from datetime import datetime

data = {
    "number": text.strip(),
    "timestamp": datetime.now().isoformat()
}
files = {"image": open("car.jpg", "rb")}

response = requests.post("http://localhost:8080/api/plates", data=data,
files=files)
print(response.status_code)
```

Мова програмування Java і фреймворк Spring Boot

Для реалізації серверної частини обрано мову Java та фреймворк Spring Boot. Цей вибір зумовлений такими перевагами:

- можливість створення повноцінного REST API;

- легка інтеграція з базами даних;
- висока продуктивність та стабільність у довготривалих запусках;
- гнучке конфігурування й підтримка безпеки доступу до API.
- Сервер, створений на базі Spring Boot, виконує наступні функції:
- приймає результати розпізнавання номерних знаків від Raspberry Pi через HTTP-запити;
- зберігає дані у базі даних (PostgreSQL);
- надає вебінтерфейс для перегляду журналу виявлених номерів;
- забезпечує простий вебінтерфейс адміністрування через браузер.

Взаємодія між компонентами

Зв’язок між Python-частиною на Raspberry Pi та Java-сервером реалізовано за допомогою REST-запитів. Після завершення обробки зображення клієнтська частина формує JSON-об’єкт із розпізнаним номером та поточною датою і часом, а потім надсилає його на сервер. Сервер обробляє цей запит, перевіряє його коректність та додає запис до бази даних.

Переваги обраного технологічного стеку:

- висока ефективність обробки зображень завдяки поєднанню OpenCV і Tesseract;
- надійна серверна логіка завдяки Spring Boot;
- гнучка структура дає змогу легко оновлювати або змінювати компоненти;
- можливість розширення функціоналу, наприклад, додавання модулів розсилки повідомлень, інтеграції з системою контролю доступу тощо.

Таким чином, використання Python і Java у поєднанні з сучасними фреймворками дозволило створити надійну, модульну та легко розширювану систему для автоматичного розпізнавання номерних знаків.

3.3 Вибір компонентів апаратно-програмного забезпечення

У процесі розробки системи автоматичного розпізнавання автомобільних номерних знаків ключовим етапом стало обґрунтоване формування набору апаратних і програмних компонентів. Вибір кожного з них був здійснений з урахуванням вимог до точності, швидкодії, масштабованості та інтеграції з іншими елементами системи.

Основною обчислювальною платформою обрано одноплатний комп’ютер Raspberry Pi. Його перевагами стали компактні розміри, енергоефективність, достатній рівень продуктивності для обробки зображень у режимі реального часу, а також підтримка бібліотек Python і сумісність із камерою. Пристрій легко інтегрується в будь-яке середовище – від стаціонарних точок контролю до мобільних платформ.

Для захоплення зображення та передачі його до обробки використано стандартну USB-камеру EMIX. Вибір зумовлений її доступністю та достатньою якістю зображення, а також підтримкою у середовищі OpenCV, яке було обрано як основний інструмент для комп’ютерного зору.

З точки зору програмної реалізації, головною мовою розробки обрано Python. Вона дозволяє швидко реалізовувати алгоритми, використовувати численні бібліотеки, серед яких ключову роль відіграє OpenCV – для виявлення номерних знаків, попередньої обробки зображень та підготовки їх до розпізнавання. Для етапу розпізнавання тексту було інтегровано OCR-движок Tesseract, що забезпечує високу точність виділення символів навіть у складних умовах, таких як нахил, тінь або часткове перекриття.

Для забезпечення комунікації між пристроєм збору даних (Raspberry Pi) та вебінтерфейсом було розроблено API на базі Spring Boot. Цей фреймворк дозволив організувати надійний REST-сервіс для передачі результатів розпізнавання та перегляду їх через браузер. Java була обрана для серверної частини з огляду на її стабільність і широкі можливості масштабування.

Вибрані компоненти ефективно взаємодіють один з одним, формуючи надійний цикл: зображення з камери обробляється Python-модулем, передається

через API на сервер, де зберігається та виводиться у вебінтерфейсі. Таке рішення забезпечує автономність роботи системи, легкість налаштування і можливість її подальшого розвитку.

3.4 Опис інтерфейсів апаратно-програмного забезпечення

Інтерфейси системи автоматичного розпізнавання номерних знаків відіграють ключову роль у забезпеченні взаємодії між користувачем, апаратною частиною та програмними модулями. Вони поділяються на два основні типи: апаратні інтерфейси, що забезпечують фізичну комунікацію між пристроями, та програмні – для взаємодії між окремими програмними модулями і з користувачем.

Центральним вузлом апаратної частини є Raspberry Pi, до якого підключена USB-камера. Взаємодія камери з комп’ютером здійснюється через USB-інтерфейс, що забезпечує стабільну передачу зображення з мінімальною затримкою. Також можливе підключення додаткових компонентів через GPIO-піни (наприклад, індикаторів, кнопок або модулів живлення), однак у базовій конфігурації цього не передбачено.

Живлення Raspberry Pi здійснюється через порт micro-USB. Передбачено підключення пристрою до мережі Ethernet або Wi-Fi, що дозволяє організувати обмін даними з віддаленим сервером або користувачем.

Програмні інтерфейси

У програмній частині реалізовано кілька рівнів інтерфейсної взаємодії:

– інтерфейс збору даних: Програма на Python ініціює роботу камери, отримує відеопотік, проводить обробку зображення (фільтрація, масштабування, виділення контурів), виявляє номерний знак і передає область інтересу на модуль розпізнавання;

– інтерфейс розпізнавання тексту: Виділена частина зображення передається в OCR-модуль (Tesseract), який повертає текстове значення номерного знака. Далі цей результат фіксується в базі даних і формує об’єкт відповіді;

– API для зв’язку з вебінтерфейсом: Оброблені дані через HTTP-запити передаються на сервер, розроблений на Spring Boot. API надає доступ до

результатів розпізнавання, включно з часом, номером, зображенням, а також підтримує фільтрацію та перегляд історії запитів;

– графічний вебінтерфейс користувача: Реалізований як браузерна сторінка, він дозволяє переглядати розпізнані номери, сортувати їх за часом, переглядати відповідні зображення та фіксувати аномалії. Користувач не взаємодіє напряму з кодом – уся взаємодія відбувається через інтуїтивно зрозумілу вебформу.

Нижче подано приклад зовнішнього вигляду вебінтерфейсу, де користувач бачить таблицю з розпізнаними номерами, часом фіксації та мініатюрою зображення транспортного засобу:

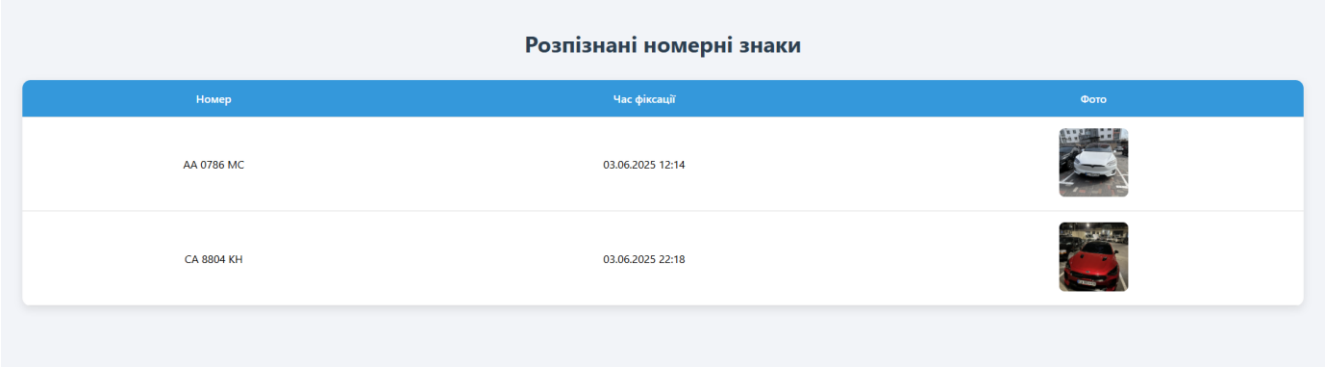

Рисунок 3.3 – Таблиця з розпізнаними номерами

Інтерфейси розроблені таким чином, щоб система могла працювати автономно, а всі дані були доступні з будь-якого пристрою в локальній мережі або через VPN-з’єднання. Це робить її придатною як для стаціонарного використання (наприклад, на паркінгах), так і для мобільних рішень, встановлених у патрульних автомобілях.

3.5 Тестування

Для перевірки працездатності та надійності розробленої системи було проведено комплексне тестування її функціональних модулів. Основна мета тестування полягала у виявленні можливих помилок при розпізнаванні номерних знаків, перевірці стабільності взаємодії між модулями, а також у перевірці коректності відображення результатів у вебінтерфейсі.

Методика тестування

Обрано метод функціонального тестування, оскільки необхідно було перевірити, як система виконує завдання відповідно до технічного завдання, не заглиблюючись у внутрішню реалізацію. Також було частково використано тестування чорного ящика – на вхід подавались різні зображення автомобілів, а на виході оцінювався результат розпізнавання.

Проведення тестування

Було підготовлено набір тестових зображень, які відрізнялись за якістю, кутом нахилу, освітленням та розміром номерного знака. Зображення оброблялись Python-скриптом з використанням OpenCV і Tesseract OCR. Після обробки результати автоматично надсилались на вебсервер через REST API, реалізований на Spring Boot.

У вебінтерфейсі перевірялась правильність:

- відображення розпізнаного номерного знаку;
- відповідності часу фіксації;
- коректного підвантаження зображення транспортного засобу;
- роботи таблиці з кількома запитами підряд.

Результати тестування

У більшості випадків система працювала стабільно та коректно. За результатами тестів було виявлено такі особливості:

- при чіткому зображенні номер розпізнається правильно з імовірністю понад **98 %**;
- при низькій якості або поганому освітленні точність розпізнавання знижується до **87 %**;
- всі запити успішно передаються до API та зберігаються у вебінтерфейсі без помилок;
- час від моменту обробки до збереження результату не перевищував **2 секунд** при звичайному навантаженні.

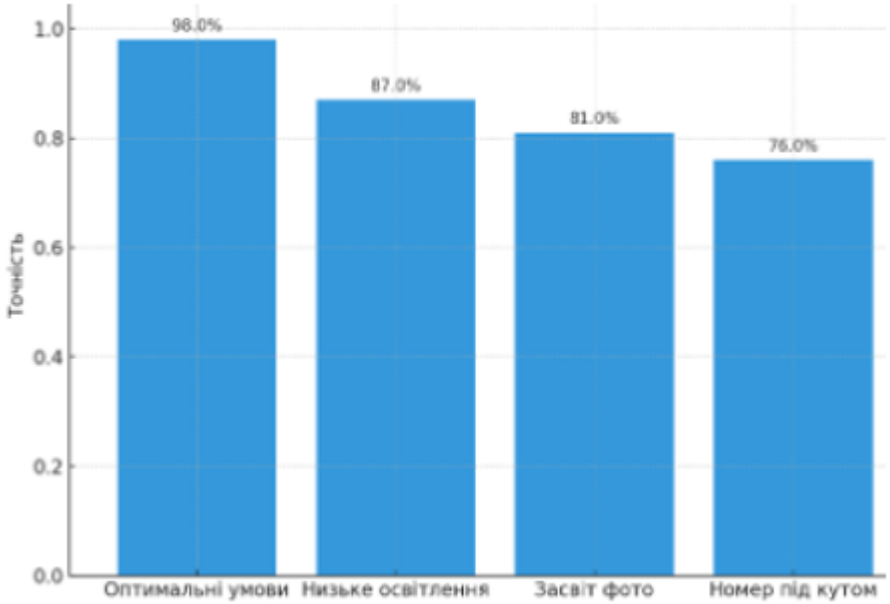

Рисунок 3.4 – Результати тестування

Якість розробленої системи можна вважати задовільною для MVP-етапу. Надалі можливо поліпшити точність OCR за рахунок покращення попередньої обробки зображень (контрастування, фільтрація шумів), а також впровадження алгоритмів машинного навчання для локалізації номерних знаків.

Висновки до розділу 3

У процесі розробки системи автоматичного розпізнавання номерних знаків транспортних засобів було створено комплексне апаратно-програмне рішення з чітко визначеною структурною архітектурою, що забезпечує ефективне захоплення, обробку, розпізнавання та збереження даних. Архітектура системи модульна, що дозволяє гнучко замінювати або модернізувати окремі компоненти без шкоди для загальної функціональності.

Апаратна частина базується на надійному та компактному одноплатному комп’ютері Raspberry Pi 4 у поєднанні з високоякісною USB-камерою, що забезпечує стабільний відеопотік з достатньою роздільною здатністю для коректного розпізнавання номерних знаків у різних умовах. Програмна частина реалізована на мові Python із застосуванням бібліотек OpenCV і Tesseract OCR, що дозволяє ефективно виконувати обробку зображень, локалізацію номерних знаків і оптичне розпізнавання символів. Серверна складова створена з використанням Java

та Spring Boot, що забезпечує надійну обробку даних, їх збереження у базі PostgreSQL і зручний вебінтерфейс для користувача.

Взаємодія між апаратною і програмною частинами організована через стандартизовані інтерфейси, включаючи USB для підключення камери та REST API для обміну даними між клієнтом на Raspberry Pi і сервером. Такий підхід гарантує масштабованість і можливість інтеграції з іншими інформаційними системами. Розроблений графічний інтерфейс користувача спрощує керування системою, надає можливість перегляду результатів розпізнавання та експорту даних. Система адаптована як для стаціонарного використання, так і для мобільних рішень.

Таким чином, створена система є ефективним, гнучким і масштабованим інструментом автоматичного розпізнавання номерних знаків, здатним задовольнити сучасні вимоги до точності, швидкодії та зручності експлуатації.

ВИСНОВКИ

В результаті виконання кваліфікаційної бакалаврської роботи було розроблено систему детектування автономерів на основі Raspberry Pi, OpenCV та Spring Boot. Зазначену мету досягнуто завдяки розв’язанню наступних завдань:

- проведено аналіз сучасних технологій і рішень у сфері автоматичного розпізнавання номерних знаків;
- розроблено загальну архітектуру системи з урахуванням обмежень апаратної платформи Raspberry Pi;
- реалізовано модуль захоплення та попередньої обробки зображень за допомогою OpenCV;
- розроблено серверну частину на основі Spring Boot для обробки результатів розпізнавання;
- забезпеченозбереження та обробку отриманих даних у базі даних;
- проведено тестування розробленої системи та оцінити її ефективність;
- надано рекомендації щодо подальшої оптимізації та розвитку системи.

У ході виконання роботи було проведено комплексний системний аналіз предметної області, що дозволив детально дослідити особливості та специфіку процесу автоматичного детектування номерних знаків транспортних засобів. На основі отриманих результатів було обґрунтовано вибір ефективних методів і сучасних засобів розробки апаратно-програмного комплексу, який відповідає сучасним вимогам автономності, продуктивності та економічної доцільності.

Одним із важливих етапів роботи стало ґрунтовне вивчення сучасних інформаційних технологій, що застосовуються у сфері розпізнавання номерних знаків. Зокрема, було здійснено аналіз актуальних апаратних платформ, їх функціональних можливостей, а також дослідження бібліотек комп’ютерного зору, серед яких ключове місце посідає OpenCV. Це дозволило сформувати повноцінне уявлення про оптимальні інструменти, які можуть бути використані у процесі розробки системи.

Особливу увагу було приділено обґрунтуванню актуальності створення автономної та економічно доступної системи на базі платформи Raspberry Pi у

поєднанні з OpenCV. Такий підхід має низку переваг, серед яких – суттєве зниження витрат на обладнання, мінімізація потреби у підключенні до високопродуктивних серверів, а також забезпечення можливості ефективної роботи системи в умовах обмежених ресурсів. Це робить розроблюваний апаратно-програмний комплекс практично застосовним у реальних умовах.

У процесі роботи було здійснено порівняльний аналіз існуючих технічних рішень у цій галузі, що дозволило виявити їхні сильні та слабкі сторони. Завдяки цьому було розроблено оптимальну архітектуру майбутньої системи, яка враховує обмеження обчислювальних ресурсів і необхідність автономного функціонування, забезпечуючи при цьому високу ефективність і надійність.

Також у роботі сформульовано вимоги до апаратно-програмного комплексу, які охоплюють як функціональні, так і нефункціональні характеристики системи. Було визначено основні сценарії використання розробленого рішення та розроблено критерії оцінки його якості, що є необхідною умовою для подальшої реалізації та тестування.

Для забезпечення комплексного функціонування системи було розглянуто та запропоновано підходи до інтеграції серверної частини на основі Spring Boot. Ця серверна складова призначена для організації збору, обробки та збереження інформації про розпізнані номерні знаки, що відкриває широкі можливості для подальшого аналізу та використання даних.

Практичні аспекти роботи включали освоєння та застосування бібліотеки OpenCV для обробки зображень, що є ключовою складовою процесу розпізнавання номерних знаків. Окрім того, здобуто значний досвід роботи з платформою Raspberry Pi як обчислювальним модулем, що підвищило рівень компетентності у створенні автономних вбудованих систем.

Таким чином, усі поставлені у межах бакалаврської роботи завдання було виконано у повному обсязі, що свідчить про готовність розробленого апаратно-програмного комплексу до подальшої впроваджувальної та дослідницької діяльності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Plate Recognizer. URL: https://platerecognizer.com/ (Last accessed: 12.05.2025).

2. OpenALPR. URL: https://www.openalpr.com/ (Last accessed: 13.05.2025).

3. Automatic Number Plate Recognition (ANPR) – A Complete Guide. Viso Suite. URL: https://viso.ai/computer-vision/automatic-number-plate-recognition-anpr/ (Last accessed: 13.05.2025).

4. Khan, M. Z. Automatic License Plate Recognition using YOLOv8 [GitHub]. URL: https://github.com/Muhammad-Zeerak-Khan/Automatic-License-Plate-Recognition-using-YOLOv8 (Last accessed: 14.05.2025).

5. Automated License Plate Readers (ALPRs) – Senstarpedia. Senstar. URL: https://senstar.com/senstarpedia/automated-license-plate-readers-alprs/ (Last accessed: 14.05.2025).

6. What is Automatic Number Plate Recognition (ANPR)? Shaip. URL: https://uk.shaip.com/blog/automatic-number-plate-recognition-anpr/ (Last accessed: 14.05.2025).

7. Стандарти ідентифікації, розпізнавання і детектування людей. InfoTech. URL: https://infotech.com.ua/article/standarty-identifikacii-raspoznavanii-i-detektirovanii-ludei (дата звернення: 14.05.2025).

8. Kavitha K., Amboru K. Detecting Vehicles Number Plates using CNN and Particle Swarm Optimization. *2025 International Conference on Intelligent Computing and Control Systems (ICICCS)*. Erode, India, Mar. 19–21 2025. P. 1256-1261. DOI: 10.1109/ICICCS65191.2025.10984880.

9. Zhang J., Wang Y., Zhou X. Improving License Plate Recognition Accuracy in Motion and Out-of-Focus Blur Environments. *Sensors*. 2025. Vol. 23, No. 5. P. 23–45. DOI: 10.1109/ICOIN63865.2025.10992703.

10. Muthukrishnan, G. Automated Traffic Ticket Generation System for Speed Violations Using YOLOv9 and DeepSORT *International Journal of Engineering Research & Technology (IJERT)*. Т. 13, № 12. 2024. URL: https://www.ijert.org/research/automated-traffic-ticket-generation-system-for-speed-

violations-using-yolov9-and-deepsort-IJERTV13IS120135.pdf (Last accessed: 20.05.2025).

11. License Plate Recognition. Inkryptis. 2023. URL: https://www.inkryptis.com/license-plate-recognition (Last access 19.05.2025)

12. Автоматизована система розпізнавання автомобільних номерів. URL: https://surl.li/njijbd (дата звернення: 19.05.2025).

13. Трофимчук, І. І. Інтелектуальні інформаційні технології : метод. вказівки до лабораторних робіт Вінниця: ВНТУ, 2022. URL: https://inmad.vntu.edu.ua/portal/static/F8003440-D6E5-4776-A571-15048258D9B6.pdf (дата звернення: 20.05.2025).

14. IEEE Xplore Digital Library: https://ieeexplore.ieee.org/search/searchresult.jsp?newsearch=true&queryText=automatic%20number%20plate%20recognition (Last accessed: 16.06.2025).

15. Знайти напрям та величину градієнта скалярного поля в точці. 2021. URL: https://yukhym.com/uk/funk2/znajti-napryam-ta-velichinu-gradienta-skalyarnogo-polya-v-tochtsi.html (дата звернення: 20.06.2025).

16. Rawat V., Sharma S. Automatic License Plate Recognition (ALPR) using YOLOv5 model and Tesseract OCR engine. *2025 International Conference on Intelligent Computing and Control Systems (ICICCS)*. Erode, India, Mar. 19–21, 2025. P. 1256–1261. DOI: 10.1109/ICICCS65191.2025.10984880

17. Haider J., Ahsan M., Rahman M. Intelligent System for Vehicles Number Plate Detection and Recognition Using Convolutional Neural Networks. 2025 International Conference on Intelligent Computing and Control Systems (ICICCS). Erode, India, Mar. 19–21, 2025. P. 1262–1267. DOI: 10.1109/ICICCS65191.2025.10984881

18. Nabizada E., Ahmad S. A Real-Time Vehicle License Plate Recognition System Using the YOLOv5 Model and Transfer Learning. 2025 International Conference on Intelligent Computing and Control Systems (ICICCS). Erode, India, Mar. 19–21, 2025. P. 1280–1285. DOI: 10.1109/ICICCS65191.2025.10984884

19. What grayscale conversion algorithm does OpenCV cvtColor() use. Stack Overflow. URL: https://stackoverflow.com/questions/19181323/what-grayscale-conversion-algorithm-does-opencv-cvtcolor-use (Last accessed: 20.05.2025).

20. Towards Data Science (Medium): https://towardsdatascience.com (Last accessed: 16.06.2025)

ДОДАТОК А

Код програми

```
import cv2
import pytesseract
from datetime import datetime
import sqlite3
import os
def init_db():
    conn = sqlite3.connect("plates.db")
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS plates (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            number TEXT,
            timestamp TEXT,
            image_path TEXT
        )
    ''')
    conn.commit()
    conn.close()

# Збереження в базу
def save_result(number, image_path):
    conn = sqlite3.connect("plates.db")
    cursor = conn.cursor()
    cursor.execute("INSERT INTO plates (number, timestamp, image_path) VALUES
(?, ?, ?)", (
        number,
        datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
        image_path
    ))
    conn.commit()
    conn.close()

# Основна функція
def detect_plate(image_path):
    image = cv2.imread(image_path)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Попередня обробка
    blur = cv2.GaussianBlur(gray, (5, 5), 0)
    edged = cv2.Canny(blur, 100, 200)

    # Знаходимо контури
    contours, _ = cv2.findContours(edged.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    contours = sorted(contours, key=cv2.contourArea, reverse=True) [:10]

    plate = None
    for c in contours:
        approx = cv2.approxPolyDP(c, 0.02 * cv2.arcLength(c, True), True)
        if len(approx) == 4:
            x, y, w, h = cv2.boundingRect(c)
            plate = gray [y:y+h, x:x+w]
            break

    if plate is not None:
        filename = f"plate_{datetime.now().strftime('%Y%m%d_%H%M%S')}.jpg"
        save_path = os.path.join("detected", filename)
```

```
            os.makedirs("detected", exist_ok=True)
            cv2.imwrite(save_path, plate)

            text = pytesseract.image_to_string(plate, config='--psm 8 -c
tessedit_char_whitelist=ABCEIKMHOPTX1234567890')

            cleaned_text = ''.join(filter(str.isalnum, text)).upper()
            print("Розпізнаний номер:", cleaned_text)

            save_result(cleaned_text, save_path)
        else:
            print("Номерний знак не знайдено")

    if __name__ == "__main__":
        init_db()
        detect_plate("car.jpg")
```

Бекенд на Java + Spring Boot

Entity Plate

```
package com.example.demo.entity;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

@Entity
@Table(name = "plates")
@Getter
@Setter
public class Plate {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String number;
    private String timestamp;
    private String imagePath;
}
```

PlateRepository

```
package com.example.demo.repository;

import com.example.demo.entity.Plate;
import org.springframework.data.jpa.repository.JpaRepository;

public interface PlateRepository extends JpaRepository<Plate, Long> {
}
```

REST-контролер

```
package com.example.demo.controller;

import com.example.demo.entity.Plate;
import com.example.demo.repository.PlateRepository;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/plates")
public class PlateController {
```

```
        private final PlateRepository repository;

        public PlateController(PlateRepository repository) {
            this.repository = repository;
        }

        @GetMapping
        public List<Plate> getAllPlates() {
            return repository.findAll();
        }
    }
```

```
<!DOCTYPE html>
    <html lang="uk">
    <head>
      <meta charset="UTF-8">
      <title>Розпізнані номерні знаки</title>
      <style>
        body {
          font-family: 'Segoe UI', sans-serif;
          background-color: #f2f4f8;
          margin: 0;
          padding: 2rem;
        }
        h1 {
          text-align: center;
          color: #2c3e50;
        }
        table {
          width: 100%;
          border-collapse: collapse;
          margin-top: 2rem;
          background-color: white;
          box-shadow: 0 4px 12px rgba(0,0,0,0.1);
          border-radius: 12px;
          overflow: hidden;
        }
        th, td {
          padding: 1rem;
          text-align: center;
          border-bottom: 1px solid #e0e0e0;
        }
        th {
          background-color: #3498db;
          color: white;
          font-weight: 600;
        }
        tr:hover {
          background-color: #f1f1f1;
        }
        img {
          width: 100px;
          border-radius: 8px;
          object-fit: cover;
        }
      </style>
    </head>
    <body>
      <h1>Розпізнані номерні знаки</h1>
      <table>
```

```
        <thead>
          <tr>
            <th>Номер</th>
            <th>Час фіксації</th>
            <th>Фото</th>
          </tr>
        </thead>
        <tbody id="plate-table-body">
          <tr><td colspan="3">Завантаження...</td></tr>
        </tbody>
      </table>

      <script>
        fetch('/api/plates')
          .then(res => res.json())
          .then(data => {
            const tbody = document.getElementById('plate-table-body');
            tbody.innerHTML = ''; // очищення
            data.forEach(plate => {
              const row = document.createElement('tr');
              row.innerHTML = `
                <td>${plate.number}</td>
                <td>${plate.timestamp}</td>
                <td><img src="${plate.imagePath}" alt="Авто"></td>
              `;
              tbody.appendChild(row);
            });
          })
          .catch(err => {
            document.getElementById('plate-table-body').innerHTML = '<tr><td
colspan="3">Помилка при завантаженні даних</td></tr>';
            console.error(err);
          });
      </script>
    </body>
```

ДОДАТОК Б

Блок-схеми роботи системи

Рисунок Б.1 – Блок-схема алгоритму роботи системи