Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Проєктування компактної системи комп'ютерного зору для автономної роботизованої платформи»

Виконав: студент 2 курсу, групи 3АКІТР-24м спеціальності 174 - Автоматизація, комп'ютерно-інтегровані технології та робототехніка
(шифр і назва спеціальності)

Ярослав ПОЛІЩУК
(ПІБ студента)

Керівник: д.т.н., проф. кафедри КСУ
В'ячеслав КОВТУН
(науковий ступінь, вчене звання / посада, ПІБ керівника)

«11» 12 2025 р.

Опонент: к.т.н., доцент каф. АІІТ
Юрій ІВАНОВ
(науковий ступінь, вчене звання / посада, ПІБ опонента)

«11» 12 2025 р.

Допущено до захисту
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
(науковий ступінь, вчене звання)

«12» 12 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти II-ий (магістерський)
Галузь знань – 17 – Електроніка, автоматизація та електронні комунікації
Спеціальність – 174 - Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітньо-професійна програма – Інформаційні системи і Інтернет речей

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«26» 09 2025 р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Поліщуку Ярославу Васильовичу
(ПІБ автора повністю)

1. Тема роботи: Проєктування компактної системи комп'ютерного зору для автономної роботизованої платформи.
   Керівник роботи: д.т.н., проф. каф. КСУ Ковтун В.В.
   Затвердженні наказом ВНТУ від «24» вересня 2025 року № 313.
2. Строк подання роботи студентом: до «12» грудня 2025 року.
3. Вихідні дані до роботи: Автономна роботизована платформа на базі ESP32-CAM; вхідні дані - RGB-відеопотік та зображення з роздільною здатністю 320×240 і 640×480; програмне середовище Python, бібліотека OpenCV; модель детекції об'єктів YOLOv8; режим реального часу з частотою не менше 10 кадрів/с.
4. Зміст текстової частини: Вступ; Дослідження предметної області та існуючих методів; Математичне та алгоритмічне забезпечення системи прийняття торгових рішень; Розробка програмного забезпечення та тестування системи; Висновки; Список використаних джерел.
5. Перелік ілюстративного (або графічного) матеріалу: Структурна схема компактної системи комп'ютерного зору автономної роботизованої платформи; схема апаратної реалізації системи на базі модуля ESP32-CAM; блок-схема алгоритму оброблення та аналізу зображень; приклади детекції об'єктів у відеопотоці; результати експериментальних досліджень працездатності та швидкодії системи.

6. Консультанти розділів роботи

Розділ змістової частини роботи	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийня
1 – 3	В'ячеслав КОВТУН, д.т.н., доцент кафедри КСУ	01.10.2025	02.12.2025

7. Дата видачі завдання: «25» вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Приміт
1	Аналіз предметної області	25.09–05.10.2025	Виконано
2	Розробка математичного та алгоритмічного забезпечення	05.10 – 25.10.2025	Виконано
3	Розробка програмного забезпечення	25.10 – 10.11.2025	Виконано
4	Тестування розробленого програмного забезпечення	05.11 – 20.11.2025	Виконано
5	Оформлення пояснювальної записки, графічного матеріалу і презентації	20.11 – 03.12.2025	Виконано
6	Попередній захист роботи	до 03.12.2025	Виконано
7	Захист роботи	до 19.12.2025	Виконано

Студент _____ Ярослав ПОЛІЩУК
(підпис) (прізвище та ініціали)

Керівник роботи _____ В'ячеслав КОВТУН
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.932:629.3.072

Поліщук Я. В. Проєктування компактної системи комп’ютерного зору для автономної роботизованої платформи. Магістерська кваліфікаційна робота зі спеціальності 174 – Автоматизація, комп’ютерно-інтегровані технології та робототехніка, освітньо-професійна програма – Інформаційні системи і Інтернет речей. Вінниця: ВНТУ, 2025. 144 с.

Укр. мовою. Бібліогр.: 26 назв; рис.: 18; табл.: 1.

Магістерська кваліфікаційна робота присвячена проєктуванню компактної системи комп’ютерного зору для автономної роботизованої платформи. Метою дослідження є підвищення ефективності функціонування мобільного робота шляхом створення апаратно-програмного комплексу для сприйняття та інтерпретації візуальних даних у реальному часі за обмежених обчислювальних ресурсів. Об’єктом є процес оброблення візуальної інформації автономною системою, предметом – методи, апаратні й програмні засоби детекції, класифікації та навігації на борту малогабаритної платформи.

Реалізація базується на використанні модуля ESP32-CAM, бібліотеки OpenCV для попередньої обробки зображень і моделі YOLOv8 для багатокласової детекції у відеопотоці; для зменшення навантаження застосовано квантування параметрів та асинхронну обробку даних. Наукова новизна полягає в адаптації методів глибинного навчання до ресурсно-обмежених платформ, практична цінність – у можливості впровадження системи в мобільну робототехніку, промислову автоматизацію, моніторинг і безпеку. Результатом роботи є спроєктована та протестована система, що забезпечує автономне сприйняття, аналіз сцени й базові поведінкові реакції робота в реальному часі.

Ключові слова: комп’ютерний зір; автономні роботизовані платформи; ESP32-CAM; OpenCV; YOLOv8; детекція об’єктів; енергоефективність.

ABSTRACT

Polishchuk Y. V. Design of a Compact Computer Vision System for an Autonomous Robotic Platform. Master's qualification thesis in specialty 174 – Automation, Computer-Integrated Technologies and Robotics, educational and professional program Information Systems and Internet of Things. Vinnytsia: VNTU, 2025. 144 p.

In Ukrainian. References: 26 sources; figures: 18; tables: 1.

The master's thesis is devoted to the design of a compact computer vision system for an autonomous robotic platform. The purpose of the research is to improve the efficiency of a mobile robot by developing a hardware–software complex for real-time perception and interpretation of visual data under limited computational resources. The object of the research is the process of visual information processing by an autonomous system, while the subject is the methods, hardware, and software tools for detection, classification, and navigation on board a small-sized platform.

The implementation is based on the use of the ESP32-CAM module, the OpenCV library for image preprocessing, and the YOLOv8 model for multi-class object detection in the video stream; to reduce computational load, parameter quantization and asynchronous data processing are applied. The scientific novelty lies in adapting deep learning methods to resource-constrained platforms, while the practical value consists in the possibility of deploying the system in mobile robotics, industrial automation, monitoring, and security applications. The result of the work is a designed and tested system that provides autonomous perception, scene analysis, and basic behavioral responses of the robot in real time.

Keywords: computer vision; autonomous robotic platforms; ESP32-CAM; OpenCV; YOLOv8; object detection; energy efficiency.

ЗМІСТ

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується зростанням ролі систем комп’ютерного зору, які забезпечують інтелектуальне сприйняття навколишнього середовища технічними засобами. Особливої актуальності набуває проєктування компактних систем комп’ютерного зору для автономних роботизованих платформ, оскільки саме такі системи визначають рівень безпеки, точності та енергоефективності взаємодії роботів із фізичним простором. Розвиток автономної навігації, розпізнавання об’єктів і контролю середовища без участі людини висуває підвищені вимоги до компактності, обчислювальної ефективності та надійності алгоритмів, що застосовуються в робототехнічних системах.

Інтеграція систем комп’ютерного зору в автономні роботизовані платформи забезпечує реалізацію повного циклу сприйняття, аналізу та реакції на зовнішні впливи, що безпосередньо впливає на ефективність виконання виробничих, транспортних і сервісних завдань. Завдяки таким системам автономні роботи здатні виявляти перешкоди, орієнтуватися в просторі, розпізнавати та класифікувати об’єкти, а також приймати рішення в режимі реального часу. Водночас актуальною тенденцією сучасної робототехніки є мініатюризація апаратної бази та зниження енергоспоживання, що зумовлює необхідність створення компактних систем комп’ютерного зору зі збереженням високих експлуатаційних характеристик.

Мета кваліфікаційної роботи полягає у підвищенні ефективності функціонування автономної роботизованої платформи шляхом проєктування компактної системи комп’ютерного зору, здатної забезпечувати оброблення та інтерпретацію візуальних даних у реальному часі за обмежених обчислювальних ресурсів апаратної частини.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати сучасний стан розвитку систем комп’ютерного зору та напрями їх застосування в автономних роботизованих платформах;
- дослідити існуючі архітектури та принципи функціонування модулів оброблення візуальної інформації з метою вибору оптимального рішення для компактної системи;
- розробити структурну схему системи та обґрунтувати вибір апаратних засобів для реалізації на малогабаритній платформі;
- створити програмне забезпечення для попередньої обробки та аналізу зображень із використанням сучасних методів комп’ютерного зору;
- інтегрувати алгоритми детекції та класифікації об’єктів у відеопотоці з урахуванням обмежень обчислювальних ресурсів;
- провести експериментальні дослідження та оцінити точність, швидкодію й стабільність роботи розробленої системи.

Об’єкт дослідження – процес оброблення візуальної інформації автономною роботизованою системою.

Предмет дослідження – методи, апаратні та програмні засоби проєктування компактної системи комп’ютерного зору для автономної роботизованої платформи.

Методи дослідження ґрунтуються на використанні методів комп’ютерного зору та машинного навчання, зокрема мови програмування Python і бібліотеки OpenCV для реалізації алгоритмів попередньої обробки зображень, а також згорткових нейронних мереж типу YOLOv8 для детекції та класифікації об’єктів у відеопотоці. Для підвищення ефективності обчислень застосовано методи оптимізації, квантування параметрів моделей та асинхронну обробку потокових даних.

Науково-технічний результат роботи полягає у проєктуванні та програмній реалізації компактної системи комп’ютерного зору для автономної роботизованої платформи, що забезпечує оброблення візуальної інформації в реальному часі за умов обмежених обчислювальних і енергетичних ресурсів.

Практична цінність роботи полягає у можливості використання отриманих результатів під час розроблення малогабаритних мобільних роботів, автономних платформ та інтелектуальних сенсорних систем, а також у підвищенні швидкодії та стабільності функціонування систем автономного руху.

1 СУЧАСНИЙ СТАН РОЗВИТКУ КОМПАКТНИХ СИСТЕМ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ АВТОНОМНИХ РОБОТИЗОВАНИХ ПЛАТФОРМ

1.1 Поняття та цілі систем комп'ютерного зору в автономних роботизованих платформах

Комп'ютерний зір є одним із ключових напрямів сучасного штучного інтелекту, що забезпечує технічним системам здатність до сприйняття та інтерпретації візуальної інформації з метою автономного прийняття рішень. Для роботизованих платформ цей напрям відіграє базову роль, оскільки дозволяє створювати технології, здатні реалізовувати інтелектуальне бачення, орієнтування в просторі та розпізнавання об'єктів у навколишньому середовищі без безпосереднього втручання оператора. Основна мета таких систем полягає у формуванні ефективних алгоритмів, здатних перетворювати цифрові зображення або відеопотоки в аналітичну інформацію, придатну для керування рухом, навігацією, ухваленням рішень і виконанням завдань у реальному часі.

Фундаментом систем комп'ютерного зору є поєднання математичних, статистичних і логічних методів, що дозволяють комп'ютеру «бачити» простір через цифрові дані. На відміну від біологічного зору людини, який використовує асоціативне мислення й досвід, машинний зір оперує чітко визначеними алгоритмами для аналізу, порівняння та класифікації ознак об'єктів[1]. Оброблення здійснюється шляхом аналізу масиву пікселів, де кожен має числове значення, що відображає яскравість або колір. Ці дані піддаються алгоритмічній обробці для виділення інформативних ознак – контурів, текстур, геометричних форм або рухомих елементів сцени.

Основними цілями комп'ютерного зору в автономних роботизованих платформах є автоматичне розпізнавання, класифікація та сегментація об'єктів у полі зору. Системи такого типу дають змогу визначати типи

перешкод, відстані до них, відстежувати динаміку руху об’єктів, прогнозувати їхню траєкторію й ухвалювати відповідні дії. Розпізнавання об’єктів базується на використанні алгоритмів машинного навчання та згорткових нейронних мереж, які забезпечують стійкість системи до змін освітлення, форми, кольору чи часткового перекриття об’єктів[2].

Сегментація зображень є важливою задачею, що передбачає поділ сцени на логічно пов’язані області для виділення об’єктів інтересу або просторових зон навігації. У межах автономних платформ така функція дозволяє виділяти траєкторії руху, виявляти небезпечні ділянки та визначати коридори безпечного пересування[3]. Завдяки цьому платформа може орієнтуватися у складному середовищі, розпізнавати границі між різними типами поверхонь і визначати їхню придатність для руху.

Важливим завданням сучасних систем є також відстеження рухомих об’єктів у часі, що особливо актуально для платформ, які функціонують у динамічному середовищі [4]. Відстеження дає змогу аналізувати зміну положення об’єктів, їхню швидкість і напрям руху, що підвищує безпеку автономної навігації. Застосування таких алгоритмів дає можливість системі своєчасно реагувати на потенційні зіткнення або появу нових об’єктів у полі зору.

Ще однією ключовою метою є відновлення тривимірної структури сцени на основі двовимірних зображень. Для цього використовуються методи стереозору, триангуляції та реконструкції глибини, які забезпечують побудову просторової карти оточення. Такий підхід необхідний для точного планування траєкторії руху автономного робота, визначення рельєфу місцевости та оцінювання просторових взаємозв’язків між об’єктами.

Окремий напрям сучасних досліджень пов’язаний із розумінням контексту зображення. Це означає, що система комп’ютерного зору має не лише фіксувати наявність об’єктів, а й аналізувати їхні взаємодії, дії та поведінкові закономірності. Для автономних платформ така функція критично

важлива, адже дозволяє передбачати зміну ситуації, наприклад, у разі появи рухомих людей або транспортних засобів у зоні навігації.

Таким чином, системи комп'ютерного зору для автономних роботизованих платформ є комплексними міждисциплінарними утвореннями, що поєднують інформатику, штучний інтелект, електроніку, фізику й мехатроніку [5]. Вони забезпечують перехід від пасивного спостереження до активного сприйняття середовища, створюючи передумови для розроблення інтелектуальних роботизованих систем, здатних самостійно аналізувати ситуацію, ухвалювати рішення та безпечно взаємодіяти з навколишнім світом.

1.2 Історичний розвиток систем комп'ютерного зору для автономних роботизованих платформ: від класичних методів до інтелектуальних моделей

Історія становлення комп'ютерного зору охоплює понад шість десятиліть досліджень, що супроводжувалися еволюцією апаратних засобів, підвищенням обчислювальних потужностей і появою нових алгоритмічних підходів. Початкові етапи розвитку цієї галузі, що припадають на середину XX століття, характеризувалися домінуванням класичних методів цифрової обробки зображень, орієнтованих на виявлення контурів, сегментацію, аналіз текстур і контрастних переходів. Через обмежені технічні ресурси того часу більшість рішень ґрунтувалися на простих математичних моделях, які дозволяли реалізовувати лише базові етапи сприйняття зображення.

Перші практичні результати були досягнуті у сфері оптичного розпізнавання символів (OCR), що стало відправною точкою для автоматизації аналізу візуальних даних. Дослідження 1959 року в галузі нейрофізіології довели, що сприйняття зображення відбувається через реакцію на прості геометричні ознаки – краї, лінії та кути. Ця концепція лягла в основу

математичного моделювання процесів бачення, що надалі визначило розвиток комп’ютерного зору [6].

У 1960-х роках почали формуватися перші алгоритми виявлення границь, серед яких особливе місце посіли оператори Собела та Робертса, що використовували градієнтний підхід. У цей період відбулося створення перших систем цифрового сканування зображень і технологій перетворення двовимірних структур у тривимірні. Одночасно зі становленням штучного інтелекту як наукового напряму з’явилися перші моделі, спрямовані на імітацію людського зору. Важливою подією стало створення Френком Розенблатом нейронної мережі «Перцептрон», яка започаткувала дослідження у сфері автоматизованого розпізнавання образів.

Подальший розвиток у 1970–1980-х роках був пов’язаний із розширенням можливостей комп’ютерів і переходом від статичних моделей до алгоритмів виявлення ознак у складних сценах. Саме тоді сформувалися основні принципи сегментації зображень і виділення ключових точок. Значного поширення набули оператори Кані, SIFT і SURF, а також методи геометричного перетворення, серед яких «перетворення Гафа». Паралельно розроблялися алгоритми тривимірної реконструкції на основі стереозображень, що стало основою для побудови просторових моделей середовища, необхідних для подальшого розвитку автономних систем [7].

У цей же період з’явилися перші технології розпізнавання текстів незалежно від шрифту, а в 1974 році розпочалося активне застосування оптичного розпізнавання символів у практичних системах. Подальша еволюція привела до створення інтелектуального розпізнавання символів (ICR), яке за допомогою нейронних мереж могло аналізувати рукописні тексти. Ці технології стали базою для автоматизованого введення даних, розпізнавання номерних знаків транспортних засобів і мобільних платіжних систем.

У 1980-х роках спостерігався перехід від простих детекторів ознак до систем, здатних ієрархічно моделювати структуру сцени. Девід Марр

запропонував концепцію багаторівневої обробки зображень, де зір розглядався як система, що послідовно виділяє краї, кути й об’єкти різної складности. Паралельно Куніхіко Фукусіма створив архітектуру «Neocognitron» – першу згорткову нейронну мережу, що фактично стала попередником сучасних CNN-моделей. Тоді ж активно впроваджувалися «моделі суміші Гауса» (GMM) для кластеризації та статистичного аналізу візуальних даних [8].

У 1990-х роках комп’ютерний зір зробив значний крок уперед завдяки використанню машинного навчання. Розробка методу опорних векторів (SVM) і класифікаторів на основі дерев рішень дала змогу покращити точність розпізнавання об’єктів. Віха розвитку цієї епохи – поява алгоритму Віоли-Джонса, який став стандартом для виявлення облич у реальному часі. Попри це, обмеження обчислювальних потужностей не дозволяли ефективно обробляти великі обсяги даних, тому багато алгоритмів залишалися малопридатними для практичного використання.

Початок XXI століття ознаменувався переходом від класичних підходів до глибокого навчання. Впровадження згорткових нейронних мереж (CNN) стало переломним моментом у розвитку комп’ютерного зору. Зокрема, у 2012 році мережа AlexNet, представлена на змаганні ImageNet, суттєво знизила частоту помилок розпізнавання та продемонструвала перевагу автоматичного виділення ознак над ручними методами. Подальші архітектури – ResNet, Inception, EfficientNet – оптимізували обчислення, підвищили швидкодію та точність систем, відкривши можливості для реалізації високоефективних моделей у режимі реального часу.

Сьогодні еволюція комп’ютерного зору тісно пов’язана з розвитком автономних роботизованих платформ, де поєднання глибокого навчання, трансферного навчання та обчислень на краю мережі (edge computing) дозволяє створювати компактні, енергоефективні системи сприйняття. Вони здатні аналізувати складні сцени, прогнозувати поведінку об’єктів і приймати рішення без підключення до потужних серверів. Таким чином, історичний

шлях розвитку комп’ютерного зору від класичних методів до інтелектуальних нейромережевих підходів відображає еволюцію технологій від простого цифрового аналізу зображень до повноцінних систем когнітивного сприйняття, що стали ключовою складовою автономної робототехніки.

1.3 Завдання комп’ютерного зору для компактних систем автономних роботизованих платформ

Комп’ютерний зір як науково-прикладний напрям формує теоретичні засади і методи побудови штучних систем, що сприймають візуальні дані у вигляді статичних кадрів, відеопотоків із одиночних або багатокамерних конфігурацій, а також тривимірних представлень, отриманих, зокрема, з медичних сканерів чи глибинних сенсорів. Як інженерна дисципліна він трансформує ці моделі у практичні рішення автоматизованої обробки, інтерпретації та використання зображень у керуванні й діагностиці. До типових сфер застосування належать системи керування технологічними процесами і рухомими об’єктами, включно з промисловими роботами та автономними транспортними засобами; інтелектуальні комплекси відеоспостереження; системи організації та пошуку візуальної інформації, що виконують індексацію великих колекцій; засоби цифрового моделювання об’єктів і середовищ, зокрема при аналізі медичних знімків або топографічній реконструкції; а також інтерфейси людино-машинної взаємодії, у яких зображення слугує каналом введення і контекстної взаємодії [9].

Комп’ютерний зір не протиставляється біологічному зору, а радше доповнює його: результати фізіології зору людини і тварин використовуються для побудови алгоритмів і архітектур, тоді як досягнення алгоритмічної інженерії забезпечують інструменти для формалізованого моделювання сприйняття. Таке перехресне збагачення дало змогу перевести низку

когнітивних механізмів у площину відтворюваних апаратно-програмних рішень і водночас висунути нові гіпотези для нейронаук.

У практиці розроблення компактних систем сприйняття для автономних роботизованих платформ завдання комп'ютерного зору доцільно групувати за цільовою парадигмою оброблення. Першу групу становлять класифікаційні задачі, зміст яких полягає у присвоєнні певній візуальній сутності уніфікованої мітки: йдеться про класифікацію цілих зображень, тегування за кількома атрибутами чи передбачення властивостей об'єкта, що є критично важливим для оперативного відбору сценаріїв керування. Другу групу формують детекційні задачі, у межах яких необхідно локалізувати об'єкти та віднести їх до класів, визначаючи положення у кадрі з точністю, достатньою для безпечної навігації та маніпуляцій. Третя група охоплює задачі сегментації, спрямовані на поділ сцени на осмислені області – від семантичної сегментації до сегментації екземплярів і паноптичної, що надає платформі деталізовану карту середовища для планування траєкторій і уникнення перешкод [10].

Розпізнавання образів інтерпретується як класифікація вхідних даних за узагальненими ознаками, що відділяють інформативну структуру від шуму і варіативних проявів. При постановці таких задач пріоритет надається формальним моделям і статистичним доведенням коректности, що зменшує залежність від суто експериментальних підходів. У суміжній постановці ідентифікації здійснюється вибір конкретного об'єкта з множини схожих, що має прикладне значення для верифікації користувачів і відстеження індивідуальних цілей у полі зору.

Оцінювання пози належить до задач високого рівня складности, оскільки потребує визначення просторового положення і орієнтації об'єктів або людських тіл через локалізацію ключових точок і кістякових структур у двовимірному зображенні чи тривимірних даних. Такий аналіз забезпечує жестову взаємодію оператора з автономною платформою, контроль ергономіки та безпеки робочих процесів і може виконуватися в режимі

реального часу на вбудованих обчислювальних модулях. Приклад такої процедури наведено на рисунку 1.1.

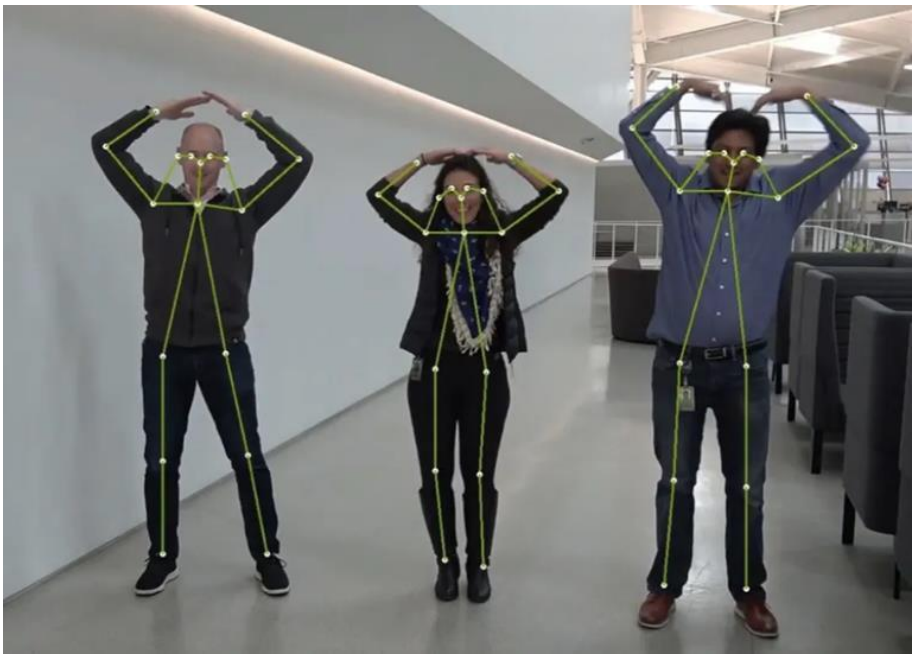

Рисунок 1.1 – Оцінка пози як механізм жестової взаємодії автономної платформи

Відстеження об’єктів у часовій послідовності кадрів спрямоване на підтримання сталої ідентичности цілей, які переміщуються, змінюють масштаб або частково перекриваються іншими об’єктами. Для компактних систем це завдання вирішується методами, що поєднують детекцію і прогнозування траєкторій, а також компенсують змінність освітлення та появу шумів. Отримані траєкторії безпосередньо інтегруються у контури систем ухвалення рішень для запобігання зіткненням і адаптивного планування руху. Ілюстративний приклад наведено на рисунку 1.2.

Рисунок 1.2 – Відстеження транспортних засобів у реальному часі для підтримки безпечної навігації

Розпізнавання облич розв’язує задачі ідентифікації та верифікації за сукупністю стабільних анатомічних ознак. Алгоритми машинного та глибокого навчання кодують геометрію і текстурні характеристики у компактні ознакові простори, що порівнюються з еталонними векторами в базах даних. У прикладних контекстах автономних систем ці засоби застосовують для контролю доступу, персоналізованої взаємодії та підвищення безпеки людино-машинних інтерфейсів. Візуальний приклад подано на рисунку 1.3 [13].

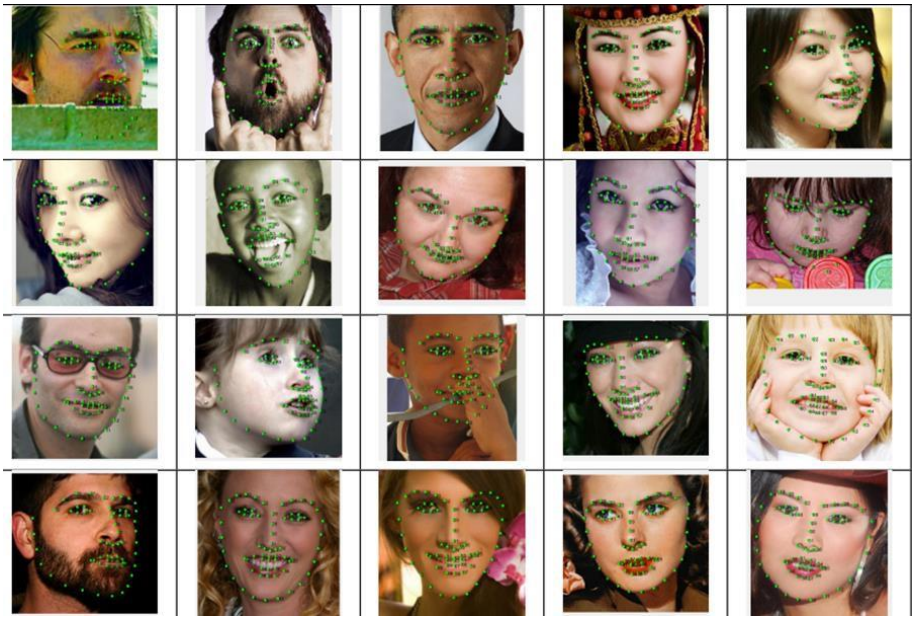

Рисунок 1.3 – Розпізнавання облич за опорними ключовими точками

Класифікація як базова операція покладається в основу багатьох виробничих і сервісних сценаріїв: від аналізу якости продукції і дефектоскопії до розпізнавання символів, жестів та атрибутів складних об’єктів. Для автономних платформ така операція забезпечує швидке віднесення спостережуваних сутностей до відомих класів і ініціює відповідні процедури керування або сигналізації. Представницький приклад показано на рисунку 1.4 [14].

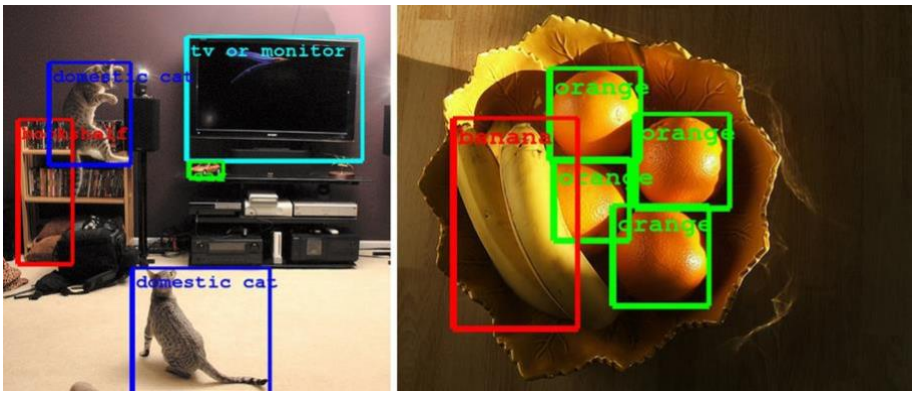

Рисунок 1.4 – Приклад класифікації об’єктів у сцені з багатьма класами

Сегментація зображень реалізує поділ сцени на однорідні області, що формують узгоджені суперпікселі та дають змогу виділити межі об’єктів, дорожню розмітку, небезпечні зони або ділянки, придатні для руху. Вона зменшує обчислювальну складність подальших модулів, адже дозволяє працювати лише з релевантними фрагментами зображення і будувати детальні карти середовища. Для автономних роботизованих платформ сегментація є ключовим етапом інтелектуальної навігації та планування безпечних траєкторій. Ілюстрацію подано на рисунку 1.5 [15].

Рисунок 1.5 – Приклад семантичної сегментації сцени для виділення об’єктів і фону

Відновлення сцени орієнтоване на реконструкцію тривимірної структури за множиною двовимірних спостережень або глибинних даних. У простому випадку формується розріджена точкова хмара, що відтворює опорні елементи, тоді як розширені методи забезпечують щільні моделі з геометрією і текстурами, придатні для локалізації та побудови карт місцевости в системах одночасного позиціювання і картографування. Відновлення зображень розв’язує іншу, але суміжну задачу – пригнічення шумів сенсорів, компенсацію розмиття та артефактів передачі, що підвищує метрологочні властивості вимірювань і стабільність алгоритмів розпізнавання [16]. У практиці застосовують від простих згладжувальних фільтрів до статистичних і навчальних моделей, які узгоджують локальні гіпотези щодо структури сигналу, виконують попередній аналіз, відновлення та за потреби післяобробку для посилення контрасту чи корекції кольору. Характерний приклад продемонстровано на рисунку 1.6.

Рисунок 1.6 – Відновлення зображення з деградаціями сенсора і носія

У контексті проєктування компактних систем комп’ютерного зору для автономних платформ важливо забезпечити баланс між точністю перелічених задач і ресурсними обмеженнями вбудованих обчислювальних модулів. Цього досягають поєднанням оптимізованих архітектур глибокого навчання з методами зменшення розрядности і розміру моделей, вибором сенсорів та оптики з відповідними метричними характеристиками, а також продуманою організацією конвеєра оброблення даних, що гарантує роботу в реальному часі та необхідний рівень надійности у змінних умовах експлуатації.

1.4 Використання систем комп’ютерного зору в сучасних роботизованих платформах

Комп’ютерний зір у сфері робототехніки є фундаментальною технологією, що забезпечує здатність технічних систем сприймати, аналізувати та інтерпретувати навколишнє середовище у візуальній формі. Його концепція полягає у відтворенні функцій людського зору шляхом перетворення даних, отриманих із камер, сенсорів або інших пристроїв

оптичного спостереження, у цифрові сигнали, придатні для подальшої інтерпретації алгоритмами штучного інтелекту [17]. Завдяки цьому комп'ютерний зір забезпечує автономним роботизованим платформам можливість ідентифікації об'єктів, навігації у складних просторових умовах, прийняття оперативних рішень у реальному часі та взаємодії з елементами середовища з високим рівнем точності.

Сучасні системи комп'ютерного зору поділяються на одномірні, двомірні та тривимірні. Кожен із цих типів має власні методологічні принципи, проте об'єднує їх спільна мета – отримання достовірної та інтерпретованої візуальної інформації для підтримки інтелектуальних функцій технічних систем. У практичній площині результати аналізу зображень застосовуються для автоматичного управління процесами, вимірювання параметрів, контролю якості, сортування або ідентифікації об'єктів, а також для створення аналітичних моделей, що дозволяють оптимізувати роботу роботизованих комплексів. Типова архітектура системи комп'ютерного зору включає оптичний модуль (камеру або сенсор), блок попередньої обробки сигналу, аналітичні алгоритми розпізнавання та модуль ухвалення рішень, який забезпечує взаємодію із системою керування роботом. Отримані дані є основою для реалізації адаптивних поведінкових моделей, що дозволяють роботам самостійно орієнтуватися у просторі, реагувати на зміни навколишнього середовища та передбачати можливі сценарії дій.

У виробничій промисловості комп'ютерний зір використовується для автоматизації технологічних процесів, перевірки відповідности деталей, контролю збірних вузлів, а також для здійснення безконтактного вимірювання або відстеження траєкторій рухомих елементів. Залежно від складности завдань застосовують двовимірні (2D) або тривимірні (3D) системи зору. Одновимірні системи, що реєструють лише інтенсивність світлового сигналу вздовж певної лінії, використовуються рідко через обмеженість інформаційного потенціалу. Основна увага зосереджується на двох і трьох

вимірах, які забезпечують значно вищу інформативність і гнучкість при аналізі складних об’єктів.

Двовимірне бачення (2D) є найпоширенішою технологією, що забезпечує швидке розпізнавання зображень у площинному форматі. Такі системи працюють на основі алгоритмів класифікації, контурного аналізу та фільтрації текстур, що дозволяє виділяти ключові об’єкти, визначати їхні межі та оцінювати геометричні параметри. 2D-аналіз ефективно виконує функції перевірки етикеток, маркування, читання штрих-кодів, розпізнавання символів, виявлення дефектів поверхні та контролю якості виготовлення деталей. Однак через відсутність інформації про глибину зображення такі системи не здатні відтворювати тривимірну форму об’єкта, що обмежує їхню придатність у процесах, де важливим є об’ємний аналіз.

Ключовою перевагою двовимірного підходу є простота оброблення та відносна швидкість функціювання, адже алгоритми не потребують великих обчислювальних ресурсів. Проте цей тип зору чутливий до змін освітлення, контрасту, тіней та якості калібрування камер. Будь-яке зміщення параметрів або неправильне налаштування сенсорів може призвести до значного спотворення результатів. Незважаючи на це, 2D-системи залишаються оптимальним вибором для плоских або слабо текстурованих об’єктів, а також для завдань, що не потребують визначення просторової структури. Для часткового розширення їхніх можливостей іноді застосовують багатокамерні конфігурації або лазерні підсвічування, які дозволяють обчислювати відносну глибину за допомогою геометричних моделей.

На відміну від двовимірного аналізу, тривимірне бачення (3D) забезпечує повноцінне відтворення просторових властивостей об’єктів, зокрема їхньої форми, об’єму, висоти та розташування у середовищі. Такі системи формують тривимірні карти сцен і застосовують методи стереозору, лазерного сканування, структурованого світла або вимірювання часу польоту світлового імпульсу. Використання 3D-зору дає змогу роботам виконувати точні маніпуляції з об’єктами нестандартної форми, орієнтуватися у

складному просторі, уникати зіткнень і здійснювати високоточні вимірювання у динамічних умовах. На практиці це означає здатність систем розрізняти відстань до об’єкта, його орієнтацію та взаємне розташування з іншими елементами середовища.

Завдяки меншій залежності від освітлення та наявності даних про глибину 3D-системи демонструють значно вищу точність у завданнях високої складности, наприклад у роботизованому складанні, палетизації, контролі геометрії складних виробів або у системах безпілотного транспорту. Водночас вони вимагають більших обчислювальних потужностей і мають вищу вартість реалізації. Проте саме ці системи забезпечують новий рівень інтелектуалізації роботизованих комплексів, що робить їх незамінними у контексті промислової автоматизації.

Серед поширених методів тривимірного бачення варто виокремити лазерну тріангуляцію, яка передбачає проектування лазерного променя на поверхню об’єкта та фіксацію його відбиття камерою під певним кутом. Така схема дозволяє обчислити профіль поверхні з високою точністю, формуючи тривимірну модель об’єкта та фіксуючи навіть незначні відхилення у геометрії [20]. Вона широко використовується для метрологічного контролю, моделювання існуючих об’єктів та діагностики дефектів на виробництві. Рисунок 1.7 ілюструє принцип дії цього методу.

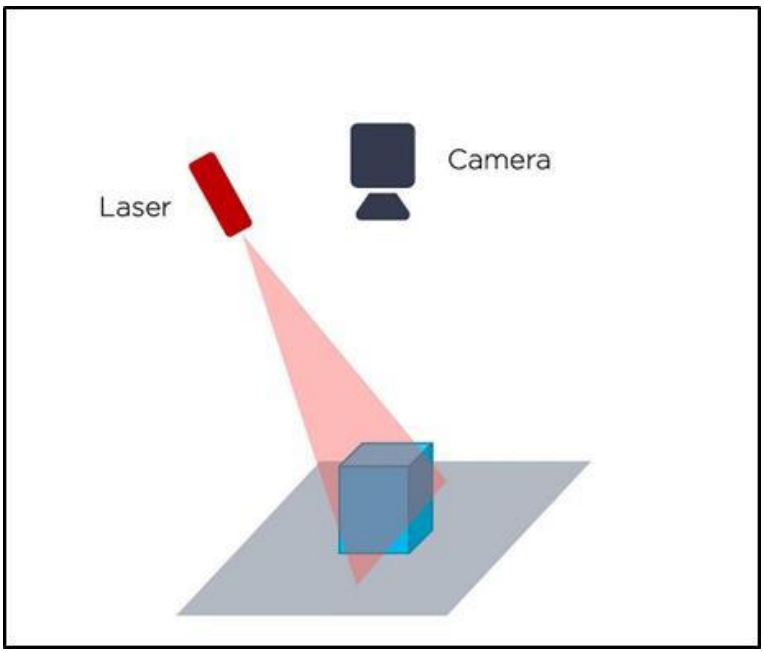

Рисунок 1.7 – Схема реалізації лазерної тріангуляції у вимірюванні тривимірного профілю об’єкта.

Ще одним ефективним підходом є стереобачення, що ґрунтується на одночасному отриманні двох або більше зображень з різних точок спостереження. Порівнюючи їх, система визначає різницю в положенні об’єктів, обчислюючи глибину сцени. Цей принцип імітує природний бінокулярний зір людини й активно застосовується в автономній навігації та системах уникнення перешкод [21]. У деяких випадках використовується варіант темпорального стерео – з однією рухомою камерою, яка послідовно робить знімки з різних позицій. Рисунок 1.8 демонструє загальну схему цього методу.

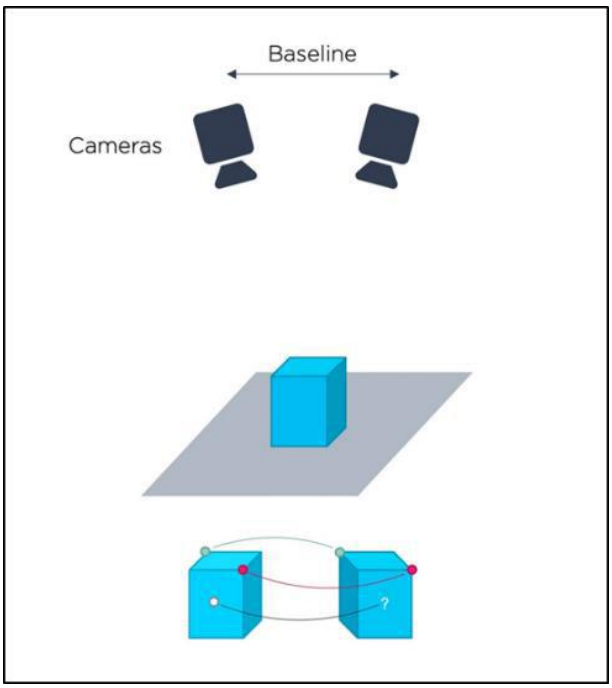

Рисунок 1.8 – Стереобачення як модель бінокулярного сприйняття простору в робототехнічних системах.

Метод часу польоту (Time-of-Flight, ToF) базується на принципі вимірювання часу, за який світловий імпульс проходить від джерела до об’єкта і назад. Такий підхід дозволяє формувати об’ємні карти глибини сцени, забезпечуючи швидке отримання тривимірних даних у реальному часі. Його часто використовують у транспортних системах, зокрема для запобігання зіткненням, а також у промислових процесах, де важлива точна оцінка відстані [22]. Рисунок 1.9 ілюструє принцип дії даної технології.

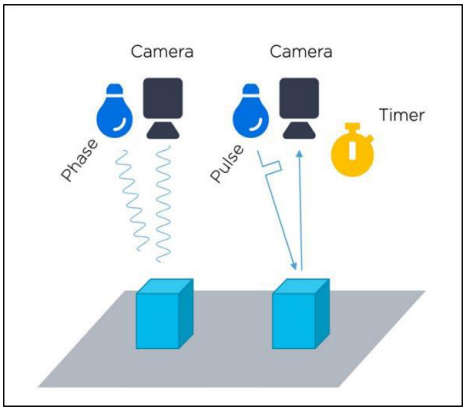

Рисунок 1.9 – Принцип роботи методу часу польоту (Time-of-Flight) у системах тривимірного бачення.

Ще один метод – структуроване світло, що полягає у проєкції спеціального світлового шаблону на поверхню об’єкта та аналізі деформацій цього шаблону при його відбитті. Таким чином система здатна відновлювати форму й текстуру поверхонь із високою просторовою роздільною здатністю. Завдяки швидкості й точності цей метод активно використовується для створення цифрових копій складних виробів, реконструкції архітектурних елементів, а також у високотехнологічному виробництві [23]. Рисунок 1.10 відображає схематичний принцип роботи структурованого світла.

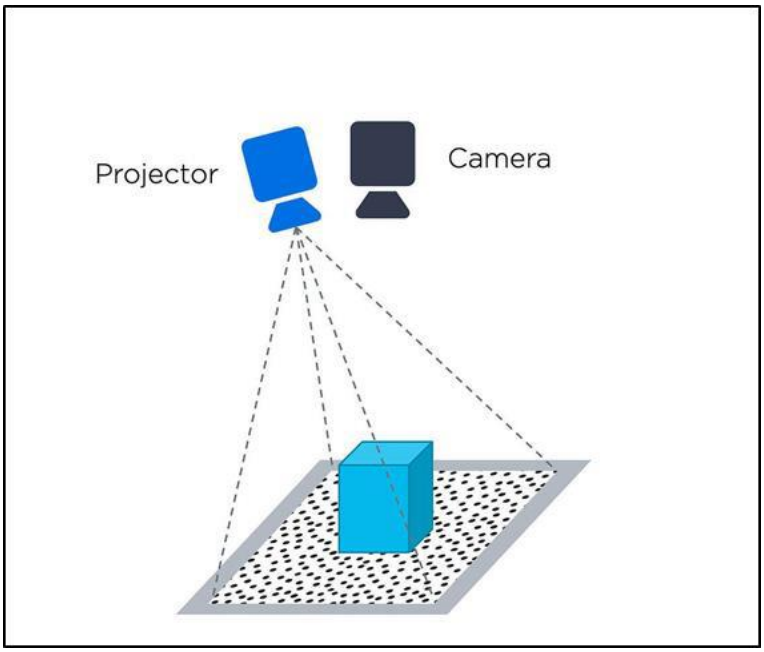

Рисунок 1.10 – Візуалізація принципу дії структурованого світла у відтворенні форми об’єкта.

Узагальнюючи, слід зазначити, що системи комп’ютерного зору різних типів формують основу сучасних роботизованих технологій. Вони забезпечують синтез сенсорного сприйняття, логічного аналізу та інтелектуальної реакції, перетворюючи робототехнічні комплекси з механічних виконавців на адаптивні автономні системи. У подальшому інтеграція комп’ютерного зору з методами машинного навчання, нейромережевого прогнозування та когнітивного моделювання сприятиме

розвитку індустрії «інтелектуальних роботів», здатних не лише аналізувати, а й осмислювати візуальну інформацію, прогнозувати зміни у середовищі та приймати рішення на основі оцінки ситуації у реальному часі.

1.5 Технічне та програмне забезпечення для реалізації систем комп'ютерного зору

1.5.1 Технічна (апаратна) складова систем комп'ютерного зору

Розроблення апаратної інфраструктури для систем комп'ютерного зору в автономних роботизованих комплексах ґрунтується на інтеграції сенсорних, обчислювальних і комунікаційних модулів, що забезпечують безперервне сприйняття візуальних даних, їх аналітичне опрацювання та оперативну реакцію на зміни навколишнього середовища. Центральне місце в архітектурі таких систем посідають оптичні пристрої – камери, які забезпечують формування зображень або потокового відео для подальшого алгоритмічного аналізу.

У практичній реалізації використовують різні типи камер, серед яких найпоширенішими є кольорові RGB-модулі, стереокамери, інфрачервоні сенсори та камери типу Time-of-Flight (ToF). RGB-модулі забезпечують кольорове сприйняття сцени з високою роздільністю, що дозволяє аналізувати текстурні й контурні характеристики об'єктів. Стереокамери, поєднуючи два оптичні канали, створюють глибинну модель простору та визначають відстані до об'єктів. Інфрачервоні камери забезпечують спостереження за умов обмеженого або відсутнього освітлення, а ToF-камери – формування карт глибини шляхом вимірювання часу проходження світлового імпульсу. Такі рішення є ключовими для навігаційних систем, маніпуляційних модулів і комплексів автономного позиціонування.

Для виконання обчислень, пов’язаних із розпізнаванням і класифікацією об’єктів, сучасні автономні мобільні платформи застосовують мікроконтролери та одноплатні комп’ютери. Мікроконтролери є компактними процесорними вузлами з низьким енергоспоживанням, придатними для реалізації базових алгоритмів попередньої обробки даних, фільтрації шумів і виконання елементарних логічних операцій. До найпоширеніших належать мікроконтролери ESP32, STM32 та Atmega серій. Їхні переваги – стабільність, дешевизна, компактність і легкість інтеграції у вбудовані системи.

Рисунок 1.11 – Мікроконтролер ESP-WROOM-32 у складі компактної обчислювальної плати для систем комп’ютерного зору

У випадках, коли потрібні вищі обчислювальні ресурси, застосовуються одноплатні комп’ютери, що поєднують у межах однієї друкованої плати центральний процесор, модулі оперативної пам’яті, контролери введення-виведення та графічний прискорювач. Прикладами таких рішень є Raspberry Pi та NVIDIA Jetson Nano, які забезпечують виконання складних моделей глибокого навчання та нейромережевих алгоритмів у реальному часі. Завдяки компактності та енергоефективності ці пристрої стали стандартом у галузі мобільної робототехніки, промислової автоматизації та систем візуального контролю якості.

Рисунок 1.12 – Одноплатний комп’ютер Raspberry Pi 5 Model B для інтеграції в роботизовані платформи

Рисунок 1.13 – Модуль NVIDIA Jetson Nano Developer Kit для реалізації нейромережевих алгоритмів оброблення зображень

Паралельно зі стандартними процесорними системами все ширше застосовуються польові програмовані вентильні матриці (FPGA). Їх перевага полягає у можливості апаратної конфігурації під конкретне завдання, що дозволяє виконувати паралельну обробку великої кількості піксельних даних при мінімальній затримці. Такі рішення оптимальні для задач, пов’язаних із потоковим розпізнаванням об’єктів, контролем виробничих ліній або обробкою даних у реальному часі з мінімальним енергоспоживанням.

Не менш важливою складовою є системи просторового орієнтування. Лідари (LiDAR) використовують лазерні імпульси для вимірювання відстаней і формування тривимірних моделей середовища. Це забезпечує високоточну реконструкцію простору навколо робота та дозволяє здійснювати автономну навігацію навіть у складних умовах. Ультразвукові сенсори, які використовують звукові хвилі для визначення відстані до об’єктів, доповнюють лідарні системи в завданнях уникнення перешкод.

Рисунок 1.14 – Лідар для робота-пилососа виробництва компанії Xiaomi, призначений для створення 3D-моделі середовища

Додаткову стабілізацію роботи забезпечують інерціальні вимірювальні модулі (IMU), які дають змогу визначати прискорення, кутові швидкості й орієнтацію в просторі. Вони використовуються для контролю положення роботизованої платформи, компенсації похибок навігаційних систем та підвищення точності визначення траєкторії. У комплексі з оптичними сенсорами IMU допомагають реалізувати алгоритми SLAM (одночасна локалізація й побудова карти).

Рисунок 1.15 – Модуль Pmod NAV (10-DOF Inertial Measurement Unit) для оцінювання просторового положення роботизованої платформи

У контексті систем машинного зору дедалі більшого значення набуває поєднання класичних сенсорів із графічними процесорами (GPU), здатними виконувати складні обчислення, пов’язані з аналітикою зображень і нейромережевою обробкою даних. Використання GPU у комбінації з модулями на основі CUDA або Tensor RT дозволяє досягти реальної паралельності в опрацюванні потоків інформації та забезпечує високу ефективність при розв’язанні завдань розпізнавання, сегментації та аналізу об’єктів. Відомою платформою, спроєктованою саме для цих цілей, є серія NVIDIA Jetson, яка забезпечує баланс між компактністю та продуктивністю й підтримує інтеграцію з модулями камер, лідарами та інерційними сенсорами.

Таким чином, апаратна складова систем комп’ютерного зору охоплює широкий спектр пристроїв – від базових мікроконтролерів до високопродуктивних платформ для глибокого навчання. Її правильне поєднання забезпечує стабільність функціонування роботизованої платформи, високу точність сприйняття та оперативне реагування на зміни середовища. Це закладає підґрунтя для ефективної роботи програмного комплексу комп’ютерного зору, що буде розглянуто в наступному підрозділі.

1.5.2 Програмне забезпечення для систем комп’ютерного зору в робототехніці

Ефективне функціонування систем комп’ютерного зору неможливе без потужного програмного забезпечення, яке забезпечує не лише взаємодію з апаратною частиною, а й інтелектуальне опрацювання та аналіз візуальних даних, що надходять із камер і сенсорних модулів. Програмна складова формує основу логіки роботи роботизованої системи, координує обчислювальні процеси, керує потоками даних і реалізує алгоритми машинного навчання, глибокого навчання та штучного інтелекту.

Для реалізації таких систем застосовуються спеціалізовані операційні середовища та фреймворки, які виконують роль інтеграційних платформ між апаратними засобами і програмними модулями. Найпоширенішою серед них є Robot Operating System (ROS) – відкрита екосистема з розподіленою архітектурою, що надає інструменти для керування сенсорами, контролю навігації, планування маршрутів, розпізнавання об’єктів і виконання паралельних процесів. ROS забезпечує гнучку модульність, що дозволяє швидко інтегрувати нові пристрої, а також містить набір пакетів для оброблення зображень, навігаційних алгоритмів, SLAM-модулів та синхронізації з апаратними контролерами.

Для систем, що потребують жорсткого контролю часу виконання завдань, застосовуються операційні системи реального часу (RTOS, Real-Time Operating Systems). Вони гарантують стабільну затримку між подією та реакцією системи, що є критично важливим для задач високошвидкісної обробки відеопотоків, стабілізації руху, обробки сигналів і керування маніпуляторами. Типові реалізації RTOS, як-от FreeRTOS або VxWorks, забезпечують точне планування процесів, оптимальне використання ресурсів і високу надійність при роботі у вбудованих середовищах.

Розроблення систем комп’ютерного зору також спирається на потужні програмні бібліотеки, які забезпечують високий рівень абстракції для

реалізації алгоритмів аналізу зображень і навчання моделей. Однією з базових є OpenCV (Open Source Computer Vision Library) – кросплатформна бібліотека, що містить сотні оптимізованих функцій для обробки зображень, фільтрації, сегментації, розпізнавання контурів, трекінгу об’єктів і побудови тривимірних реконструкцій. Її поєднання з Python, C++ або Java дозволяє реалізувати повний цикл розпізнавання – від обробки кадрів до прийняття рішень.

У сфері інтелектуального аналізу візуальних даних домінують бібліотеки TensorFlow та PyTorch, що забезпечують створення, навчання і розгортання глибоких нейронних мереж. Вони підтримують використання апаратного прискорення за допомогою графічних процесорів (GPU) і тензорних прискорювачів (TPU), що значно підвищує швидкість оброблення великих обсягів даних. На базі цих платформ реалізуються моделі для класифікації, сегментації, генерації та прогнозування, що використовуються у візуальній аналітиці та автономній навігації.

Особливу роль відіграють алгоритми реального часу, серед яких найвідомішим є YOLO (You Only Look Once) – архітектура глибокої нейронної мережі, призначена для детекції об’єктів у потоковому відео. Її ключова перевага – швидкість і точність, що дозволяє виконувати аналіз без затримок навіть на вбудованих пристроях. Поряд із YOLO застосовуються інші моделі, такі як SSD (Single Shot Multibox Detector) або Faster R-CNN, які спеціалізуються на розпізнаванні складних сцен із кількома об’єктами.

Ключовими напрямами реалізації алгоритмів комп’ютерного зору в малогабаритних роботизованих системах є виявлення та класифікація об’єктів, сегментація зображень, оцінювання оптичного потоку та реалізація SLAM (Simultaneous Localization and Mapping) – технології, що дозволяє роботові одночасно формувати карту навколишнього середовища та визначати власне місцеположення. Такі процеси забезпечують автономність і точність навігації в реальному просторі, що має вирішальне значення для безпілотних транспортних засобів і роботів мобільного типу.

Для підвищення продуктивності системи комп’ютерного зору активно використовують технології апаратного прискорення. Зокрема, GPU (Graphics Processing Unit) – це універсальний елемент для паралельного опрацювання великих масивів даних, який часто інтегрується в платформи типу NVIDIA Jetson або AMD Ryzen Embedded. Крім того, застосовуються TPU (Tensor Processing Unit) – високоефективні тензорні процесори, оптимізовані для виконання операцій із глибокими нейронними мережами, та NPU (Neural Processing Unit) – спеціалізовані мікрочипи, що реалізують обчислення для штучних нейронних мереж без залучення центрального процесора. Це дозволяє значно зменшити енергоспоживання системи при збереженні високої швидкодії.

У практичному застосуванні системи комп’ютерного зору функціонують як багаторівневі комплекси, де кожен рівень – від збору зображень до ухвалення рішень – реалізується окремими програмними модулями. На першому рівні здійснюється попередня обробка даних, на другому – аналітичне перетворення, а на третьому – синтез поведінкової реакції робота. Завдяки узгодженій роботі цих модулів досягається стабільна адаптація системи до змін середовища, зниження похибок під час розпізнавання та підвищення точності автономних дій.

Підсумовуючи, можна зазначити, що ефективна розробка програмного забезпечення для комп’ютерного зору потребує збалансованого поєднання продуктивних алгоритмів, оптимізованих бібліотек і високошвидкісних апаратних рішень. Сучасні фреймворки дозволяють не лише створювати гнучкі та адаптивні системи, а й забезпечують їхню масштабованість, стабільність і здатність до самооновлення. Використання інтегрованих середовищ розробки, інструментів візуалізації та симуляції, таких як Gazebo або RViz, сприяє ефективному тестуванню алгоритмів ще до фізичного впровадження. Саме поєднання цих компонентів формує основу інтелектуальної автономії роботизованих систем і відкриває перспективи подальшої еволюції комп’ютерного зору у взаємодії людини й машини.

1.6 Формулювання мети, завдань і концептуальних засад дослідження

Основною метою даної кваліфікаційної роботи є розроблення, теоретичне обґрунтування та практична реалізація інтелектуальної системи комп'ютерного зору для малогабаритного мобільного робота на базі мікроконтролерного модуля ESP32-CAM. Запропонована система покликана забезпечити автоматичне виявлення об'єктів у полі зору, їхню ідентифікацію та класифікацію з подальшою передачею інформації до обчислювального вузла для аналізу, прийняття рішень і реалізації поведінкових реакцій робота в режимі реального часу. Розроблена архітектура має сприяти підвищенню автономності, точності та адаптивності роботизованої платформи до змінних умов навколишнього середовища.

Для досягнення поставленої мети дослідження необхідно реалізувати комплекс взаємопов'язаних теоретичних і практичних завдань, спрямованих на створення функціонально завершеної системи, що поєднує сучасні апаратні та програмні рішення. Насамперед передбачається здійснення системного аналізу сучасних підходів і технологічних парадигм у галузі комп'ютерного зору для мобільних роботів, включно з оцінюванням наявних архітектур, алгоритмів глибинного навчання та сенсорних платформ. Особливу увагу необхідно приділити порівнянню існуючих методів виявлення й розпізнавання об'єктів, визначенню їхніх переваг, недоліків і сфер застосування, а також аналізу можливостей їхньої реалізації на ресурсно-обмежених обчислювальних модулях.

Другим напрямом завдань є вибір та обґрунтування апаратної бази, що включає підбір основного мікроконтролера, модуля камери, допоміжних сенсорів та обчислювальних компонентів. Метою цього етапу є забезпечення оптимального співвідношення між продуктивністю, енергоспоживанням, вартістю та можливістю інтеграції з іншими елементами системи. Вибір

модуля ESP32-CAM обумовлений його високим рівнем енергоефективності, наявністю вбудованого модуля Wi-Fi, можливістю роботи в автономному режимі та сумісністю з бібліотеками машинного навчання.

Третє завдання полягає у розробленні програмного забезпечення для обробки зображень та інтелектуального розпізнавання об’єктів. Передбачається створення алгоритмічного ядра, що поєднує засоби бібліотеки OpenCV для базової обробки відеопотоку (фільтрація, нормалізація, виділення контурів, підвищення контрастності) з технологією нейронних мереж YOLOv8, яка дозволяє здійснювати багатокласову детекцію об’єктів із високою швидкістю та точністю. Впровадження цих технологій має забезпечити стабільну роботу навіть у випадках обмеженого освітлення чи складного фону.

Наступний етап охоплює проведення експериментальних досліджень працездатності створеної системи в умовах реального середовища. Для цього планується здійснити серію випробувань на мобільній платформі з метою оцінювання точності виявлення об’єктів, затримки передачі даних, ефективності обчислень і стійкості системи до зовнішніх перешкод. Результати експериментів стануть підґрунтям для формулювання висновків щодо надійності, ефективності та перспектив подальшого вдосконалення розробленого рішення.

З огляду на викладене, дослідження передбачає інтеграцію апаратних, програмних і алгоритмічних складових у єдину функціональну систему, здатну здійснювати автономне сприйняття, аналіз і реагування на об’єкти довкілля. Практичне значення отриманих результатів полягає у можливості застосування розробленої системи в галузях мобільної робототехніки, моніторингу, безпеки та автоматизації виробничих процесів. Реалізація поставлених завдань сприятиме розвитку напрямів штучного інтелекту та впровадженню технологій комп’ютерного зору у сфері вбудованих систем, створюючи передумови для подальших наукових і прикладних досліджень у цій галузі.

2 ПІДБІР ЗАСОБІВ ДЛЯ РОЗРОБЛЕННЯ СИСТЕМИ КОМП'ЮТЕРНОГО ЗОРУ

2.1 Сучасні рішення робототехніки

Відповідно до підходу, зафіксованого в ISO 8373:2012, робот розглядається як багатоцільовий механізм із програмованими ступенями свободи, здатний діяти в щонайменше двох осях та виконувати поставлені задачі з певною мірою автономності [24]. Таке трактування підкреслює не лише механічну природу роботів, а й їхню інтегрованість із сенсорно-обчислювальним забезпеченням, завдяки якому пристрій сприймає середовище, інтерпретує дані та впливає на об'єкти зовнішнього світу. У межах теми цієї роботи – проєктування компактної системи комп'ютерного зору для автономної платформи – акцент зміщується від загальної автоматики до інтелектуального сприйняття, яке забезпечує навігацію, виявлення і класифікацію об'єктів, прогнозування траєкторій та своєчасне ухвалення керуючих рішень у динамічних сценах.

Еволюція робототехніки закономірно проходила через кілька етапів. Початкові рішення другої половини XX століття були зосереджені на відтворенні жорстко визначених послідовностей рухів, що давало можливість автоматизувати монотонні технологічні операції без участі людини. Із поширенням сенсорних модулів, систем керування та обчислювальної техніки з'явилася здатність реагувати на зміну зовнішніх умов, адаптуючи попередньо задані режими. На сучасному етапі, який репрезентує третє покоління роботів, у практику масово увійшли алгоритми машинного навчання та глибинні моделі, здатні навчатися на даних, узагальнювати попередній досвід та формувати поведінкові стратегії. Саме завдяки такому зсуву робот перестав бути лише виконавцем і перетворився на суб'єкт кіберфізичної взаємодії, який поєднує сенсори, вбудовані обчислення та актуатори в єдиний замкнений цикл сприйняття–мислення–дії.

Роль комп’ютерного зору в цій трансформації фундаментальна. Камерні системи, доповнені інерціальними та дальномірними давачами, стали універсальним каналом сприйняття, а поява компактних прискорювачів обчислень дала змогу виконувати складні обчислювально-візуальні задачі безпосередньо «на борту» мобільних платформ. Для автономних роботів це означає перехід від квазістатичних карт до оновлюваних у реальному часі карт представлення середовища, що поєднують геометрію сцени, семантичні мітки та оцінки невизначеності. У такій постановці навіть базові дії – уникнення перешкод, доїзд до цілі чи хватання об’єкта – вимагають інтегрованої зорової підсистеми, спроможної стабільно працювати за змінного освітлення, при часткових оклюзіях і в умовах обмежених ресурсів.

На рисунку 2.1 зображено ковальсько-пресовий промисловий робот, що демонструє класичний сценарій автоматизації небезпечних гарячих процесів. У подібних установках високотемпературні заготовки подаються і позиціюються маніпуляторами з приводами високої жорсткості, а цикл формоутворення контролюється силомоментними сенсорами і температуростійкими візуальними датчиками. Висока повторюваність траєкторій, точність у діапазоні десятих часток міліметра та можливість інтегрування з системами зорової інспекції після операції штампування забезпечують стабільність геометрії деталей і скорочують відсоток браку. Такий приклад ілюструє, як навіть у традиційних жорсткоциклових процесах візуальні підсистеми підвищують якість та безпечність виробництва.

Рисунок 2.1 – Ковальсько-пресовий промисловий робот

Подальший розвиток робототехніки продемонстровано на рисунку 2.2, де подано сервісний робот-пилосос із візуально-лазерною навігацією. Сучасні побутові платформи поєднують круговий сканер відстаней, фронтальні камери, інфрачервоні та ультразвукові давачі, використовуючи алгоритми SLAM для побудови топологічно узгоджених карт приміщень. Реалізація семантичної локалізації дає змогу розрізняти типи покриття, ідентифікувати перешкоди малої висоти, адаптувати силу всмоктування та коригувати маршрут залежно від вмісту сцени. З позиції тематики цієї роботи ці платформи цікаві тим, що демонструють практичну можливість розміщення повного зорового конвеєра в малогабаритному корпусі за обмеженого енергоспоживання.

Рисунок 2.2 – Сервісний робот-пилосос

На рисунку 2.3 показано роботизовану хірургічну систему Da Vinci як приклад професійної робототехніки з високими вимогами до точности візуалізації та маніпуляційної керованости. Стереоскопічна відеосистема формує об’ємне зображення операційного поля, а програмно-алгоритмічний блок масштабує рухи хірурга, відсікаючи тремор і забезпечуючи мікрометричну плавність. Сучасні модифікації інтегрують модулі автоматизованого розпізнавання тканин та судин, що підвищує інтраопераційну безпеку. Хоча такі системи не є автономними, саме поєднання високоякісного камерного тракту й алгоритмів аналізу сцени демонструє межові вимоги до зорових підсистем, які корисно враховувати при проєктуванні компактних рішень.

Рисунок 2.3 – Робот-хірург Da Vinci

Рисунок 2.4 ілюструє військове застосування автономних платформ на прикладі БПЛА Ghost 4 від Anduril Industries. Комплекс обробляє мультиспектральний відеопотік на борту, поєднує виявлення об’єктів, відстеження та геоприв’язку, формує стабілізовані панорами місцевості та може взаємодіяти в роєвих конфігураціях. З технічної точки зору цей клас систем репрезентує найсуворіший набір вимог до компактної зорової підсистеми: робота за широкого діапазону освітленості, стійкість до вібрацій і атмосферних перешкод, обмежений тепловий бюджет та потреба у гарантованих затримках оброблення. Зазначені вимоги визначають критерії добору камер, обчислювальних модулів і алгоритмів для автономних наземних платформ, що розглядаються у цій роботі.

Рисунок 2.4 – БПЛА Ghost 4 від Anduril Industries

На рисунку 2.5 наведено людиноподібного робота-андроїда TOPIO 2.0, відомого демонстрацією гри в настільний теніс. Тривимірне сприйняття траєкторії швидкого м’яча, оцінка часу підльоту, прогноз лінії відскоку за моделлю руху з опором та керування багатоланковим маніпулятором із десятками ступенів свободи – усе це реалізується в реальному часі завдяки компактній системі камер високої частоти зчитування та прискореній обчислювальній підсистемі. Приклад демонструє, що за належної інтеграції сенсорики й алгоритмів можна досягти реакцій, порівнянних із людськими, у корпусі обмежених габаритів, що концептуально збігається з поставленою в роботі метою – створенням компактного модуля зору для мобільної платформи.

Рисунок 2.5 – Людиноподібній робот-андроїд TOPIO 2.0, який може грати в настільний теніс з людьми

Поряд із прикладними зразками доцільно систематизувати класифікаційні підходи без переліків, у зв’язному викладі. У межах ISO 8373:2012 промислові роботи трактуються як маніпулятори, орієнтовані на автоматизацію технологічних процесів виробництва; їхня місія полягає в підвищенні продуктивності, точності, повторюваності та виведенні людини із зон ризику. У позавиробничій сфері поширення набули сервісні роботи, що підміняють або доповнюють людину у побуті, медицині та сфері послуг, де ключовими показниками є комфорт, ефективність і безпека. За класифікацією Міжнародної федерації робототехніки розрізняють особисті та професійні системи: перші призначені для повсякденної експлуатації користувачами й охоплюють, зокрема, автоматизоване прибирання чи догляд, тоді як другі забезпечують комерційні процеси – консультування, кур’єрську логістику, навігацію, адміністративні функції та медичну діагностику [25]. Окремим напрямом виділяються роботи для безпеки, спроможні діяти у

вибухонебезпечних, токсичних чи радіаційних умовах, а також військові платформи, що охоплюють наземні, морські та повітряні носії зі спектром завдань від розвідки до вогневої підтримки [26]. Такі типології не є самоціллю, однак вони задають вимоги до зорових підсистем: для промислових систем критичні точність і повторюваність, для сервісних – робастність до варіативних сцен, для безпекових і військових – стійкість у складних середовищах і енергоефективність при обмежених ресурсах.

Окремої уваги заслуговує триада sense–think–act, що в контексті комп'ютерного зору інтерпретується як безперервний цикл перцепції, оцінювання стану та дії. Сенсорний фронт-енд забезпечує збір даних камер і допоміжних давачів; обчислювальний модуль здійснює багаторівневу обробку – від фільтрації та корекції спотворень до детекції, сегментації й семантичної інтерпретації; актуаторний контур трансформує результат у керуючі впливи. Для малогабаритних автономних платформ ключовими стають затримка оброблення, енергоспоживання, стабільність роботи за змінної освітленості та надійність у випадку часткових оклюзій. Саме ці метрики надалі використовуються як цільові при підборі камер, обчислювальних модулів і алгоритмів.

Нарешті, у практичних застосуваннях значення мають спосіб пересування та рівень автономності. Колісні шасі забезпечують високу енергоефективність на рівних поверхнях і широко застосовуються в логістиці; гусеничні платформи підвищують прохідність на пересіченій місцевості; крокуючі системи та гексаподи дають змогу долати складний рельєф, але висувають жорсткі вимоги до швидкодіючого зору з прогнозуванням опорних контактів; літальні та плавучі носії додають обмеження на масогабарит і споживання. Автономні, напівавтономні та керовані режими експлуатації прямо визначають, який обсяг візуальної аналітики має виконуватися на борту, а який – поза роботом, і якою має бути архітектура обміну даними. Усі наведені аспекти консолідуються у вимогу до компактної, енергоощадної, але інформативної зорової підсистеми, що і становить предмет подальшого

розділу, присвяченого добору компонентів та алгоритмів для автономної роботизованої платформи.

2.2 Сучасні малогабаритні мобільні платформи як носії компактних систем комп’ютерного зору

Сучасні малогабаритні мобільні роботи становлять один із найдинамічніших сегментів прикладної робототехніки та одночасно є природними носіями компактних систем комп’ютерного зору. Їхня широкомасштабна поява зумовлена поєднанням технологічних та економічних чинників: мінімізацією енергоспоживання мікроелектроніки, здешевленням високоякісних камер та інерціальних давачів, розвитком вбудованих графічних прискорювачів і зрілих програмних стеків. У результаті навіть невеликі мобільні платформи отримали змогу виконувати завдання, що ще десять років тому були доступні лише повнорозмірним роботам із потужними бортовими комп’ютерами. Для задач проєктування компактної системи комп’ютерного зору для автономної платформної бази вирішальним стає не лише факт наявності камери або лідару, а насамперед архітектурна інтегрованість сенсорики з обчислювальною підсистемою, стабільність часової синхронізації та здатність забезпечувати гарантовану продуктивність алгоритмів у реальному часі за наявності обмежень по масі, габаритах і живленні. У цьому контексті колісні малогабаритні платформи вирізняються сприятливим компромісом між кінематичною простотою та вантажопідйомністю для розміщення оптичних сенсорів і модулів оброблення зображень, що і визначає їхній пріоритет для побудови дослідних і промислових систем.

Колісні мобільні роботи, що вміщують компактні підсистеми зору, характеризуються конструктивною модульністю: корпус виконує функцію

несної рами та теплового екрана для електроніки; колісні рушії задають вимірювану кінематику руху, зручну для параметричної ідентифікації; електродвигуни, здебільшого з редукторами на постійному струмі або безколекторні, керуються ШІМ-каналами та зворотними зв’язками за струмом і кутовою швидкістю; мікроконтролер виконує функції жорсткого реального часу, пов’язані з керуванням приводами й опитуванням давачів; надбудований одноплатний комп’ютер або модуль зі спеціалізованим прискорювачем відповідає за високорівневі алгоритми локалізації, виявлення та відстеження об’єктів. Компактні камери формують потоки RGB або YUV із відомою затримкою, що має бути врахована при часовому узгодженні з даними IMU та одометрією. Інженерно-технологічна зрілість такого класу платформ засвідчується наявністю надлишкових бібліотек драйверів, стандартних механічних інтерфейсів для монтажу сенсорів та багатого інструментарію калібрування. Саме ці риси роблять колісні платформи оптимальною основою для розгортання компактних систем зору, орієнтованих на задачі навігації, огляду робочої сцени, оцінювання вільного простору та виконання елементарних маніпуляцій.

Одним із показових представників навчально-дослідницького класу є платформа TurtleBot 4, яка слугує де-факто стандартом для відпрацювання алгоритмів автономної мобільності в середовищі ROS [27]. З погляду проєктування компактної системи зору TurtleBot 4 репрезентує збалансовану архітектуру: базове двоколісне шасі з диференціальним приводом забезпечує прогнозовану кінематику, а низький центр маси зменшує коливання корпуса, що позитивно впливає на стабільність відеопотоку. На рівні сенсорики платформа комплектується далекоміром ближнього поля і фронтальною камерою, яку легко замінити на модуль із більшим полем зору або на глобальний затвор для коректнішого вимірювання при швидких обертах. Вбудовані зіткнені давачі, а також IMU дозволяють реалізовувати базові схеми сенсорного злиття: дані одометрії, інерційної навігації та візуального відстеження ознак комбінуються в єдиній фільтраційній моделі, що зменшує

дрейф та підвищує стійкість локалізації у разі часткових втрат візуальної інформації. Програмно TurtleBot 4 інтегрований у ROS, де доступні пакети для калібрування внутрішніх та зовнішніх параметрів камери, запуску візуальної одометрії, побудови карти зайнятості та планування траєкторій. Для задач комп'ютерного зору, пов'язаних з розпізнаванням об'єктів, платформа безпосередньо підтримує моделі, оптимізовані під вбудовані графічні прискорювачі, що дозволяє досягати режимів реального часу за помірної потужності живлення. Розширений опис конструктивно-функціональних ознак цієї платформи подано на ілюстрації, що відображає її архітектуру і габаритні співвідношення. Рисунок 2.6 – Робот TurtleBot із встановленою камерою та сенсорами дальності як носій компактної системи комп'ютерного зору.

Другий репрезентативний приклад – JetBot на основі NVIDIA Jetson Nano, який типологічно відноситься до компактних інтелектуальних носіїв для відпрацювання глибокого навчання на борту [28]. Вбудований модуль Jetson із CUDA-сумісним GPU дозволяє реалізувати на платформі повноцінні конвеєри оброблення зображень: від попереднього фільтрування та дебайєризації до інференсу нейронних мереж YOLO/SSD/DeepLab, після чого результати використовуються у циклі керування рухом. Саме така «щільна» інтеграція обчислювальних елементів із сенсорикою і виконавчими механізмами демонструє, як проєктувати компактні системи зору, що не потребують зовнішнього хмарного обчислення і водночас досягають частоти кадрів, достатньої для безпечної автономної навігації в кімнатних і лабораторних середовищах. Конструкція JetBot зазвичай виконується на легкому каркасі з акрилових або полікарбонатних пластин, що спрощує монтаж додаткових сенсорів – наприклад, ширококутних камер для огляду на 180 градусів або стереопари для грубої оцінки глибини. Важливо, що стеки NVIDIA пропонують засоби оптимізації нейронних мереж, зокрема TensorRT, що дає змогу зменшити латентність інференсу та вписатися у жорсткі часові обмеження контурів керування. На ілюстрації наведено конфігурацію, де

камера з гнучким шлейфом встановлена у верхній частині та має можливість зміни кута огляду. Рисунок 2.7 – Модель Al Robot Kits від NVIDIA JetBot Partners із модулем Jetson Nano для прискореного комп’ютерного зору.

Окремий клас становлять комерційні автономні платформи логістичного призначення, як-от серія MiR, що відома в промисловому середовищі завдяки надійності, сертифікації безпеки та гнучкій інтеграції з виробничими процесами [29]. Модель MiR100 демонструє, як архітектурні рішення, характерні для великих систем, можуть бути запропоновані у компактному форм-факторі: низький профіль корпусу, оптимізований під проїзд під стелажами, модульні надбудови для закріплення сенсорних блоків, енергоефективний привід і системи заряду від док-станцій. З погляду систем зору, MiR застосовує комбінацію далекомірів і камер для запобігання зіткненням та забезпечення контекстного сприйняття середовища. Нехай базові алгоритми зосереджені на задачах безпечної навігації, однак саме структурно-конструктивні принципи – рознесення сенсорів, відсутність взаємних затінень, екранування від вібрацій – є прикладними шаблонами для проєктування компактних систем зору на інших носіях. З інженерної точки зору корисним є розуміння потоків даних і їхнього пріоритезування: критичні канали безпеки функціонують навіть за умов деградації продуктивності основних обчислювачів, а високорівневі функції розпізнавання об’єктів можуть бути відключені або переведені у деградований режим без втрати базової маневреності. На ілюстрації показано робота із характерною світловою індикацією периметра, що сигналізує про стан і напрямок руху. Рисунок 2.8 – Робот MiR100 як компактна логістична платформа з інтегрованими відеосенсорами.

До класу навчально-демонстраційних, проте технологічно насичених рішень належить робот DJI RoboMaster S1, який поєднує керовану рухомість на всеспрямованих колесах, багату сенсорику і можливості візуальної взаємодії з навколишнім середовищем [30]. Завдяки гіростабілізованому модулю з камерою та використанню розвинених алгоритмів відстеження

візуальних маркерів і об’єктів S1 демонструє, як компактна система зору може одночасно вирішувати задачі сприйняття сцени та точного наведення. В освітніх сценаріях платформа показує, що правильна постановка задачі калібрування (внутрішні параметри камери, зовнішні параметри відносно бази, часові затримки) безпосередньо впливає на стабільність контурів керування, особливо коли візуальні вимірювання замикають зворотний зв’язок за положенням. Доступ до SDK і приклади реалізації детекторів і трекерів дозволяють перенести ці підходи на інші компактні носії – від диференціальних шасі до мікро-МБП. На ілюстрації зображено апарат із двома радіоканалами зв’язку, фронтальною камерою та механізмом наведеного випромінювання, що використовується в навчальних батлах. Рисунок 2.9 – Робот DJI RoboMaster S1 як приклад інтегрованої навчальної платформи з розвиненими функціями комп’ютерного зору.

Перелічені платформи утворюють репрезентативне поле, в межах якого можна сформулювати архітектурні засади проєктування компактних систем зору для автономних робомобільних носіїв. Насамперед слід виходити з вимог до часової детермінованості. Потік відеоданих має оброблятися в рамках жорсткого бюджету затримок, сумісного з періодом оновлення команд приводу. Якщо, наприклад, цикл керування становить 50–100 мс, то сумарна латентність кадру від сенсора до актора не повинна перевищувати 30–60 мс, із резервом під фонові діагностичні задачі. Досягається це шляхом апаратного прискорення попередньої обробки (дебайєризація, масштабування, перетворення колірних просторів), використанням інференсу оптимізованих мереж (квантизовані моделі INT8 або FP16), а також конвеєризацією обчислень із перекриттям етапів читання кадру, детекції та планування руху.

Другою системоутворювальною вимогою є метрологічна узгодженість вимірювань. Навіть компактна система зору має проходити повноцінне калібрування: внутрішня калібровка камери (фокусна відстань, центр проєкції, коефіцієнти дисторсії), зовнішня калібровка відносно системи координат шасі, часовий офсет щодо IMU та одометрії. Наявність стабільної температурної

моделі важлива для невеликих корпусів, де нагрів графічного модуля здатний спричинити дрейф механічних з’єднань і мікрозміщення камери. В алгоритмічному сенсі це трансформується у коректну постановку задачі візуально-інерційної одометрії, де дані про кутову швидкість та лінійні прискорення використовують як прогноз вектора стану, а візуальні спостереження ключових точок – як корекцію за інновацією. Незважаючи на компактність, вимога до субпіксельної точності локалізації ознак залишається актуальною, оскільки саме вона визначає дрібномасштабні похибки локалізації платформи у кімнатних середовищах.

Третій принцип стосується енергетичної ефективності. Для малогабаритного носія типова ємність акумулятора обмежує безперервну роботу на одиниці години. Тому при проєктуванні компактної системи зору доцільно застосовувати політику адаптивного навантаження: частота кадрів і глибина обчислень мають підлаштовуватися під поточну динаміку сцени та критичність задачі. Під час прямолінійного руху порожнім коридором достатньо, наприклад, виконувати детекцію в кожному другому кадрі та використовувати оптичний потік для інтерполяції між результатами, тоді як у зоні складних маневрів або при наближенні до людей система переходить у підвищений режим частотного оновлення. Така політика суттєво подовжує ресурс автономного функціонування без втрати безпеки.

Четвертий аспект – стійкість до реальних умов експлуатації. Компактні корпуси піддаються вібраціям від приводів, ударним навантаженням при подоланні порогів, впливу пилу та випадкових засвітів. Відповідь на ці виклики лежить у правильній механіці кріплення камери (жорстка рама, демпфери з відомою резонансною частотою), оптичному кондиціюванні (козирки проти блисків, фільтри), а також у програмній фільтрації артефактів (видалення «спалахів», компенсація rolling-shutter за даними IMU). Слід передбачати і поведінку системи під час деградації: якщо детектор об’єктів відмовляє через погане освітлення, контур безпеки повинен спиратися на

простіші евристики вільного простору, що працюють за допомогою порогування та геометричних перетворень.

Суттєву увагу варто приділяти питанням відтворюваності й оцінювання якості. Для проєктування компактної системи зору необхідна експериментальна методика зі стандартизованими сценами, де змінюються освітленість, текстурність, наявність динамічних перешкод і дзеркальних поверхонь. Результати слід вимірювати за узгодженими метриками: точність та повнота детекції, середня похибка локалізації, середній час на кадр, імовірність помилкових спрацювань контуру безпеки. Навіть для навчально-дослідних платформ доцільно вести журнал параметрів і версій моделей, оскільки мінімальні модифікації процедури навчання (розклад навчальної швидкості, обсяг аугментацій, зміна балансу класів) впливають на стабільність роботи в реальному середовищі.

З технічного погляду зростає роль мультисенсорності. Камера на малогабаритній платформі часто поєднується з легким лідаром класу 2D або ToF-модулем. Таке поєднання дає доповнюваність: візуальні алгоритми краще розпізнають семантику сцени, тоді як дальнісні – забезпечують метрологічну стабільність у монотонних, слабкотекстурних коридорах. Для компактних систем доречно застосовувати тісно зчеплене злиття на рівні фактор-графа, коли і ключові точки, і променеві вимірювання додають обмеження до єдиної задачі оптимізації. У підсумку зменшується імовірність накопичення помилок та підвищується повторюваність траєкторій, що особливо важливо для платформ із невеликими коліщатами й високою чутливістю до нерівностей підлоги.

Окремо зупинимося на матеріалах і конструктиві. Для компактних платформ актуальна проблема тепловідведення від модулів прискореного обчислення. Конструкція верхньої платформи повинна одночасно забезпечувати жорстке кріплення камери та конвекційні канали для охолодження GPU/NPU. Оптимально застосовувати алюмінієві або магнієві сплави як інтерфейсні пластини, що слугують і ребрами жорсткості, і

теплопровідними елементами. Кабелі камери мають бути зафіксовані так, щоб виключити мікросмики у місцях пайки та роз’ємів, оскільки саме ці мікродефекти породжують випадкові розсинхронізації кадрів і збої драйверів. Водночас важливо зберігати ремонтопридатність: модульна філософія кріплення дозволяє швидко замінити камеру на альтернативну (наприклад, із глобальним затвором) без зміни базової геометрії шасі, що критично у дослідницьких проєктах.

Розглянуті апаратні платформи унаочнюють варіативність інженерних підходів. TurtleBot 4 демонструє екосистемну цілісність ROS-рішень та простоту масштабування; JetBot вказує на роль локального прискореного інференсу для режимів реального часу; MiR100 показує промислову культуру безпеки, від якої доцільно наслідувати методи фізичного й програмного резервування; DJI RoboMaster S1 підкреслює значення високодинамічної стабілізації камери та взаємодії з маркерами для точних задач наведення. Для нашого магістерського завдання – створення компактної системи комп’ютерного зору для автономної роботизованої платформи – ці приклади формують вимоги до архітектури: камерний модуль із гарантованою частотою кадрів, вбудований обчислювач із апаратною підтримкою глибинних мереж, сенсорне злиття з IMU та одометрією, стандартизовані інтерфейси, а також експлуатаційна стійкість до вібрацій і змін освітленості.

Синтез наведених підходів дозволяє побудувати методологію інженерного компромісу. Якщо головна мета – навігація у тісних приміщеннях із насиченою семантикою, пріоритетним стає високочастотний візуальний тракт і справний детектор класів, адаптований до характерних об’єктів середовища. Якщо ж домінує вимога максимальної надійності у монотонних коридорах, доцільно підсилити систему легким лідаром і спростити детекцію до задачі оцінювання вільного простору, розвантаживши інференс. У будь-якій конфігурації треба зберігати масштабованість: можливість переходу від однієї камери до стереопари, від CPU-інференсу до NPU/TPU без перероблення всієї системи. Така масштабованість у поєднанні з

відтворюваною процедурою калібрування й тестування створює основу для надійного проєктування та подальшої серійної реплікації.

Рисунок 2.6 – Робот TurtleBot із встановленою камерою та сенсорами дальності як носій компактної системи комп’ютерного зору

Рисунок 2.7 – Модель Al Robot Kits від NVIDIA JetBot Partners із модулем Jetson Nano для прискореного комп’ютерного зору

Рисунок 2.8 – Робот MiR100 як компактна логістична платформа з інтегрованими відеосенсорами

Рисунок 2.9 – Робот DJI RoboMaster S1 як приклад інтегрованої навчальної платформи з розвиненими функціями комп’ютерного зору

Підсумовуючи, сучасні малогабаритні мобільні роботи не лише демонструють різні інженерні школи рухомості та сенсорної інтеграції, а й формують практичні шаблони, які безпосередньо застосовні при проєктуванні компактної системи комп’ютерного зору для автономної роботизованої платформи. Їхня цінність у рамках даної теми полягає у наочному підтвердженні, що навіть за жорстких обмежень по масі, габаритах і енергоспоживанню можливо організувати повноцінний візуальний конвеєр з детекцією, стеженням і локалізацією, забезпечити реальний час, витримати метрологічні вимоги та гарантувати безпечну взаємодію з довкіллям. Саме

такі інженерні принципи і будуть покладені в основу подальших рішень, викладених у наступних розділах, із обов’язковим урахуванням вимог до оригінальності, відтворюваності та промислової технологічності.

2.3 Обґрунтування та аргументація вибору апаратних засобів

Розроблення системи комп’ютерного зору для малогабаритної колісної платформи потребує не лише правильно обраних алгоритмів, а й уважно підібраного комплексу апаратних компонентів, здатних забезпечити стабільний збір візуальних даних, їх оперативну обробку та надійне керування приводами в реальних умовах експлуатації. У межах цього підрозділу викладено логіку добору елементної бази, узгоджену з вимогами до точності розпізнавання, часової детермінованості, енергоефективності, масо-габаритних обмежень і технологічної сумісності. Сформована цільова конфігурація покликана забезпечити виявлення об’єктів у полі зору, визначення їхнього розташування у кадрі та передачу команд на виконавчі механізми з мінімальними затримками, що, своєю чергою, вимагає камери з адекватною роздільною здатністю, обчислювального блока з підтримкою сучасних бібліотек комп’ютерного зору та периферії, здатної працювати у змінних умовах освітлення і на неоднорідних поверхнях.

Ключовими критеріями відбору компонентів виступали компактність і маса, адже надлишкова вага прямо зменшує автономність і маневреність; енергоспоживання, що визначає тривалість роботи від акумулятора; апаратно-програмна сумісність у межах єдиної шини живлення й логічних рівнів; надійність у штатних та позаштатних режимах (вібрації, мікроудари, перепади освітленості); а також повна вартість володіння – від закупівлі до підтримки. Відповідно до цих вимог здійснено порівняння кількох навчально-

дослідницьких комплектів, на базі яких можливо швидко сформувати працездатний прототип і надалі масштабувати його функції.

З урахуванням доступності, відкритої документації, наявності великої спільноти користувачів, готових шасі та набору типових датчиків опорною платформою обрано комплект Elegoo Smart Robot Car Kit. У практичних випробуваннях він продемонстрував оптимальний баланс між ціною й функціональністю: базовий контролер Arduino-сумісного класу забезпечує передбачувану роботу приводів, модулі відстані й лінієвого стеження стабільно працюють у приміщеннях зі стандартним штучним освітленням, а вбудовувана камера ESP32-CAM дозволяє реалізувати початкові задачі комп’ютерного зору без додаткових дорогих обчислювальних модулів. За потреби обчислювальну частину можна нарощувати, підключаючи, наприклад, Raspberry Pi або Jetson Nano як зовнішній вузол прискореної обробки потоку.

Замість пунктів переліку наведемо інтегрований опис критеріїв добору. Обмеження за масою диктували застосування легких матеріалів у силовій конструкції шасі, тож для несної основи використано акрилові панелі з товщиною близько п’яти міліметрів – це забезпечує достатню жорсткість для чотирьох колісних модулів із редукторами і водночас не перевантажує платформу. Енергозабезпечення спроєктовано під двосекційний літієвий акумулятор 7,4 В із вбудованим індикатором та вузлом захисту, що підвищує експлуатаційну безпеку та дає змогу витримувати пікові струми під час старту двигунів. Для зменшення електромагнітних перешкод у контурі живлення двигунів застосовано окремі лінії розв’язки й локальні конденсатори біля кожного редуктора; це важливо, коли в системі працює радіомодуль ESP32 із Wi-Fi. Надійність забезпечується механічними фіксаторами роз’ємів і маркованими шлейфами, що мінімізує помилки при сервісному обслуговуванні. Нарешті, загальна вартість рішення залишається прийнятною завдяки широкій доступності компонентів-аналогів і сумісних модулів.

Щоб зафіксувати місце обраного комплекту серед найближчих альтернатив, сформовано узагальнену табл. 2.1. Вона порівнює чотири поширені набори: освітній комплект на базі Arduino-сумісного контролера, початковий конструктор mBot із простим графічним середовищем, чотириколісну платформу Freenove, що працює з Raspberry Pi, а також більш «важку» навчально-дослідну платформу SunFounder з акцентом на візійну навігацію. У межах адаптації замість розлогих пунктів подано згорнуті характеристики словами, акцентуючи на реальній здатності платформ працювати з комп'ютерним зором, складності введення в експлуатацію та можливостях подальшого масштабування. За підсумком саме набір на кшталт Smart Robot Car Kit виявився найкращим стартовим варіантом для роботи з ESP32-CAM, тоді як Pi-орієнтовані комплекти доцільні, коли пріоритетом є складні моделі нейронної інференції та використання OpenCV/ROS «із коробки».

Таблиця 2.1 – Узагальнений порівняльний огляд базових наборів для мобільних роботів (адаптовано)

Аспект	**Elegoo Smart Robot Car (Arduino-клас)**	**Makeblock mBot (mCore)**	**Freenove 4WD (Raspberry Pi)**	**SunFounder PiCar-X (Raspberry Pi)**
Вартість і доступність	Низька; широке постачання компонентів	Вища; орієнтація на STEM-класи	Вища; потрібен Pi окремо	Найвища; орієнтація на складні проєкти
Стартова складність	Низька: швидка збірка, Arduino IDE	Дуже низька: графічні середовища	Помірно висока: Linux/Python	Висока: Linux/Python/ROS

Базовий візійний стек	ESP32-CAM, прості задачі детекції	Фактично відсутній	Повний стек OpenCV з Pi-камерою	Повний стек, підтримка SLAM/ROS
Конструкція й приводи	Акрил, 4 редукторні двигуни, гумові колеса	Легка платформа, 2 двигуни	Посилена рама, підвищені навантаження	Посилена рама, модульність
Модернізація	Висока: додаткові сенсори, підключення Pi/Jetson	Обмежена для складних модулів	Висока: LiDAR, камери, безліч датчиків	Висока: LiDAR/камери/ROS
Типові сценарії	Навчання, прототипування CV, автономна їзда	Початкове STEM-навчання	Дослідження з ML/Computer Vision	Дослідження, проєкти з високою продуктивністю

Окрему увагу приділено силовій конструкції. На рис. 2.10 – Шасі платформи з акрилових панелей показано дві несні пластини із заводською розміткою отворів під монтаж редукторів, опорних стійок, акумуляторного блока та плати керування. Вибір акрилу як базового матеріалу зумовлений поєднанням невеликої маси, легкої оброблюваності та достатньої жорсткості для побутових і лабораторних задач; за потреби несну здатність можна підвищити алюмінієвими дистанційниками. Важливо, що симетрія й уніфікований малюнок отворів дозволяють переставляти вузли при зміні компоновки, зберігаючи баланс мас по діагоналях.

Рисунок 2.10 – Шасі платформи з акрилових панелей (верхня та нижня плити).

Приводний контур представлено редукторними двигунами постійного струму й колесами із гумовими накладками, що видно на рис. 2.11 – Рухові модулі та колеса з підвищеним зчепленням. Використання редукторів із передавальним числом, підібраним під діаметр колеса, дозволяє отримати достатній крутний момент на малих швидкостях і плавний старт без прослизання. Для підвищення точності керування швидкістю у майбутніх ітераціях доцільно передбачити енкодери на валах (магнітні або оптичні), що спростить побудову ПІД-регулятора курсу та лінеаризацію розгону.

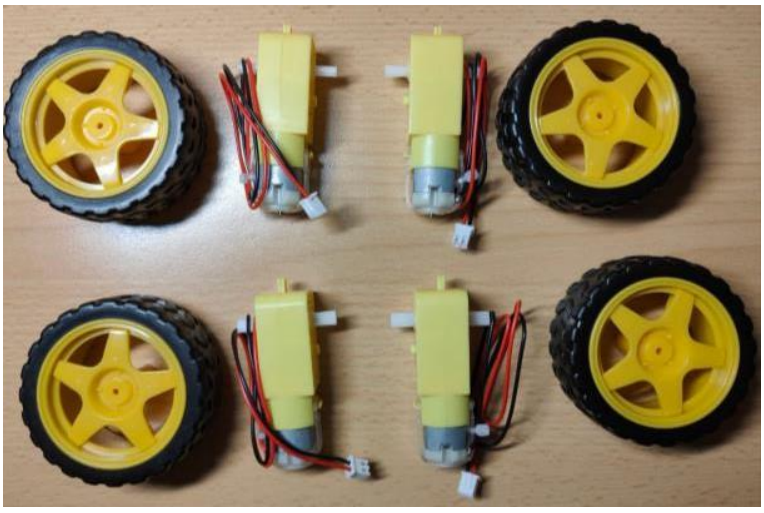

Рисунок 2.11 – Рухові модулі: редукторні двигуни та колеса з гумовим протектором.

Для візійного вузла застосовано панорамний кронштейн із мікросерводвигуном SG90, який дає можливість реалізувати огляд за азимутом без повороту всього шасі; це проілюстровано на рис. 2.12 – Мінісерводвигун SG90 у двоплощинному кронштейні. Кутова швидкість і крок позиціонування достатні для простого сканування сцени, а мала маса мінімізує інерційні навантаження на верхню панель. У коді бажано застосувати профілі згладжування (s-curve) для зменшення деренчання під час зупинок, що позитивно вплине на стабільність відеопотоку.

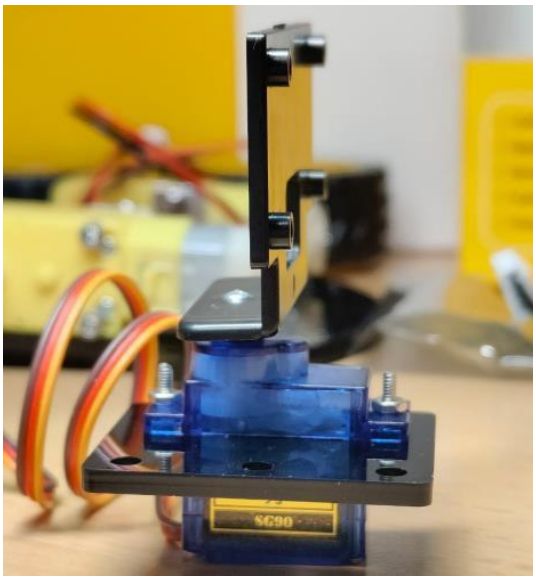

Рисунок 2.12 – Мінісерводвигун SG90 у двоплощинному кронштейні для панорамування камери.

Центральним елементом керування обрано класичну UNO-сумісну плату на мікроконтролері ATmega328P, показану на рис. 2.13 – Мікроконтролерна плата загального призначення (Arduino-клас). Вона забезпечує прогнозовану роботу ШІМ-виходів для моторів і просту інтеграцію з периферією через готові бібліотеки. Для складніших задач інференції пропонується гібридний підхід: зберігаючи UNO як низькорівневий контролер приводів і датчиків реального часу, візійну обробку передавати на ESP32-CAM або зовнішній SBC, комунікуючи через UART/I²C.

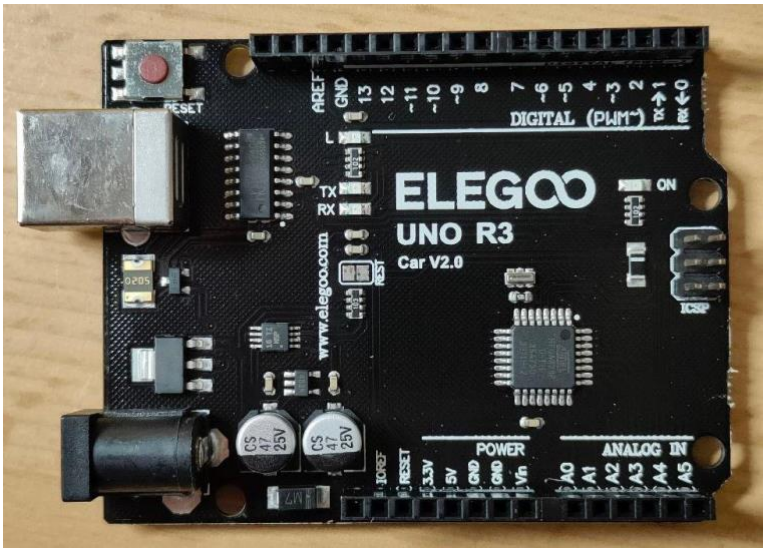

Рисунок 2.13 – Мікроконтролерна плата UNO-класу для низькорівневого керування.

На рис. 2.14 – Розширювальна плата сигналів із модулем інерційних вимірювань показано шілд із «швидкими» конекторами під основні датчики та модуль GY-521 (IMU з акселерометром і гіроскопом). Наявність IMU відкриває змогу реалізувати стабілізацію напрямку руху, оцінювати кут повороту під час диференціального керування та фільтрувати ковзання коліс. У програмній частині для злиття даних доречно застосувати комплементарний фільтр або фільтр Маджвіка як компроміс між швидкодією і точністю.

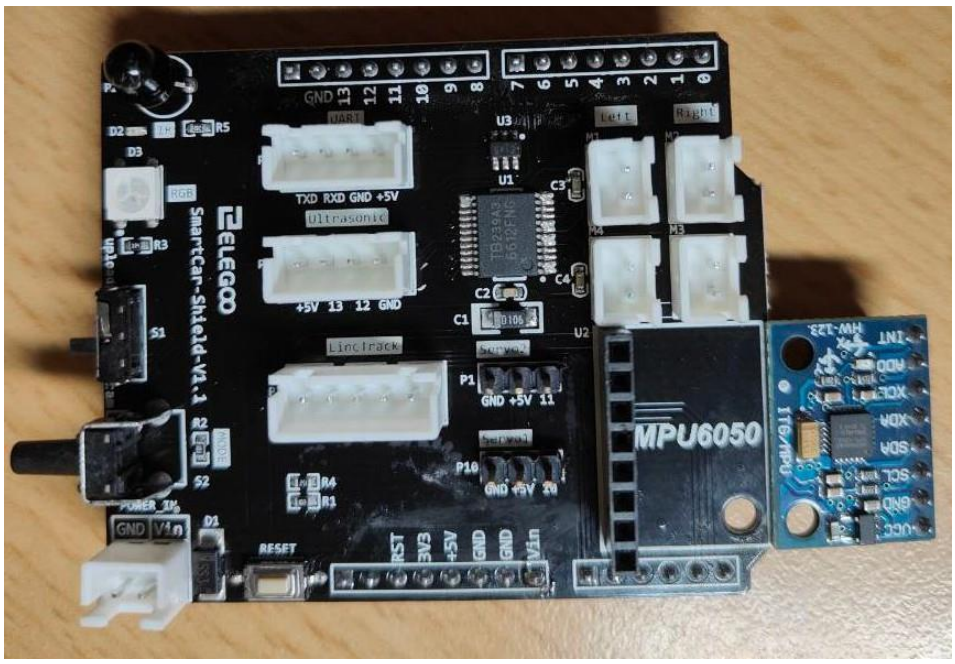

Рисунок 2.14 – Розширювальна плата сигналів із модулем IMU GY-521.

Базовий дальномірний канал сформовано ультразвуковим модулем HC-SR04, представлений на рис. 2.15 – Ультразвуковий далекомір ближнього діапазону. У типових кімнатних умовах датчик стабільно вимірює дистанції від двох до кількох сотень сантиметрів із типовою похибкою кілька міліметрів. Щоб уникати хибних спрацьовувань від тонких або звукопоглинальних поверхонь, у прошивці реалізовано бортовий «медіанний» фільтр для серії імпульсів і обмеження кутового вікна перевірки, синхронізованого з положенням пан-серви камери.

Рисунок 2.15 – Ультразвуковий далекомір HC-SR04 для вимірювання дистанції.

Функцію лінієвого стеження покладено на оптичний модуль із трьома ІЧ-каналами, показаний на рис. 2.16 – Трипозиційний модуль відстеження лінії. Рознесення чутливих елементів дає змогу визначати відхилення від траси та будувати простий регулятор із безперервним керуванням швидкості лівого й правого бортів. Для підлоги зі строкатим фоном передбачено програмну калібровку порогів відбиття і «гаряче» збереження параметрів у EEPROM.

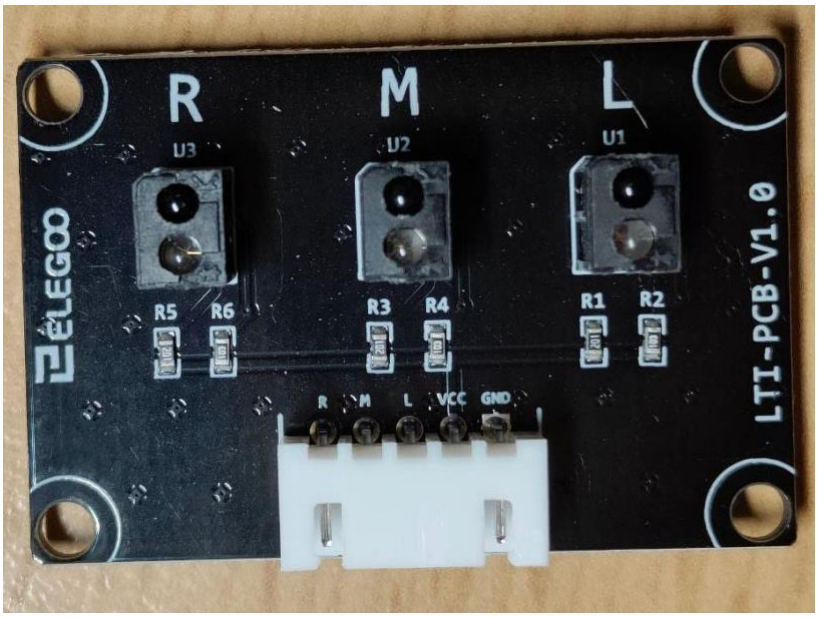

Рисунок 2.16 – Трипозиційний модуль відстеження лінії на ІЧ-оптопарах.

Акумуляторний блок на літієвій базі з апаратним вимикачем та світловим індикатором стану зображено на рис. 2.17 – Живлення мобільної платформи з індикацією заряду. Він забезпечує близько двох годин змішаної роботи за умови помірного використання відеостріму. Для безпеки рекомендовано встановити в прошивці поріг напруги з м’яким відключенням приводів і зупинкою Wi-Fi у разі глибокого розряду, що істотно продовжує ресурс акумулятора.

Рисунок 2.17 – Акумуляторний блок живлення з індикацією стану та портом заряджання.

Візійний сенсорний вузол реалізовано на ESP32-CAM, який поєднує двоядерний мікроконтролер родини ESP32 з камерою OV2640; модуль показано на рис. 2.18 – Камерний модуль ESP32-CAM для потокового відео. Конфігурація підтримує знімання до UXGA й передавання кадрів через Wi-Fi у локальну мережу, де на зовнішньому вузлі може виконуватися «важка» інференція (наприклад, YOLOv8-nano). На самому ESP32 можливо реалізувати легкі детектори руху, апаратні фільтри, порогову бінаризацію, виділення контурів та публікацію подій через MQTT – цього достатньо для задач стеження за мітками, базового виявлення перешкод або запуску сервісних дій.

Рисунок 2.18 – Камерний модуль ESP32-CAM з сенсором OV2640.

Комплексне керування здійснюється прошивкою, написаною в Arduino IDE з використанням відкритих бібліотек для мотор-драйверів, ультразвуку, IMU і сервоприводів. Логіка побудована модульно: низькорівневий цикл підтримує детерміноване опитування датчиків і ШІМ-керування з періодом у кілька мілісекунд; високорівневі поведінкові стани перемикаються подієво – отриманням результатів зору, зміною дистанції, перетином лінії тощо. Це дозволяє інтегрувати нові сценарії без суттєвого перероблення базового коду.

Щоб дати повніший контекст обґрунтування, нижче наведено розширений опис типових вузлів у прив’язці до реальних експлуатаційних обмежень. Конструктив шасі проєктувався з огляду на розміщення акумулятора ближче до геометричного центру нижньої плити, що знижує момент інерції при різких маневрах і мінімізує «клевки» під час старту. Кріплення камери винесено на окремий кронштейн, відокремлений від двигунів через гумові демпфери – таким чином зменшено високочастотні вібрації у відеопотоці, які негативно впливають на стабільність детектора ознак. Проводка живлення моторів відмежована від слабкострумових ліній датчиків, а силові дроти скручені попарно для зменшення наведень.

Алгоритмічно система підтримує три репрезентативні сценарії: автономне уникнення перешкод на основі ультразвуку з корекцією курсу за

IMU; стеження за лінією з м’яким об’їздом локальних перешкод завдяки панорамному скануванню камерою; та режим телеметрії, коли ESP32-CAM передає низько-бітрейтний стрім оператору, а керування здійснюється через Wi-Fi. Кожен сценарій реалізує однаковий абстрактний інтерфейс до приводів, що спрощує тестування й порівняння.

З точки зору подальшого росту складності запропоновано гібридну архітектуру: UNO/STM32 рівня «реального часу» лишається керувати приводами, датчиками і безпекою, тоді як SBC-вузол (Raspberry Pi 4 або Jetson Nano) під’єднується через UART/Ethernet і виконує детекцію об’єктів, семантичну сегментацію чи локалізацію за візуальними ознаками. У такому випадку ESP32-CAM може працювати як IP-камера або як попередній препроцесор кадрів (маскування, змінення розміру, стиск). Це дозволяє еволюційно нарощувати можливості без повної заміни базової платформи.

Важливо також зафіксувати ризики та заходи їхнього пом’якшення. Обмежена обчислювальна спроможність ESP32-CAM накладає верхню межу на розмір і частоту кадру при локальній обробці; для критичних задач рекомендовано передавати кадри на зовнішній вузол або застосовувати квантовані, малі за параметрами моделі (типу MobileNet/YOLO-Nano). Ультразвук чутливий до складної геометрії й звукопоглинальних матеріалів, тому для надійної навігації доцільно комбінувати його з візійними підказками (оптичний потік, маркери ArUco) або з короткофокусним ToF-сенсором. У плані енергетики помітною статтею витрат є Wi-Fi-передавання – для тривалих автономних місій варто зменшувати частоту кадрів і тривалість стріму, переводячи камеру в подієвий режим.

Підсумовуючи, обраний апаратний стек забезпечує послідовний шлях еволюції від базового рівня – із локальною передобробкою кадрів на ESP32-CAM і простими поведінковими реакціями – до більш складної архітектури з окремим вузлом високопродуктивної візійної інференції. Усі елементи конструкції підібрані так, щоб виконувати ключові вимоги проєкту: досягати прийнятної точності детекції, гарантувати керованість платформи в реальному

часі, працювати автономно протягом достатнього проміжку та залишатися ремонтопридатними й економічно ефективними. Така конфігурація створює резерв для подальшого розширення – від додавання інерційної стабілізації пан-тилт вузла до інтеграції комп'ютерного зору, заснованого на семантичній сегментації та позиційній оцінці, що в перспективі дозволить переходити від реактивної поведінки до планувальників траєкторій на основі карти середовища.

2.4 Вибір програмних засобів і обґрунтування програмного стека

Під час розроблення прототипу колісної мобільної платформи з функціями комп'ютерного зору було сформовано цілісний програмний стек, у якому роль операційного середовища виконує Windows 11, а основними інструментами є Arduino IDE для низькорівневого керування, інтерпретатор Python як базове середовище алгоритмічної логіки, бібліотека OpenCV для опрацювання зображень і відеопото́ків та модельний ряд YOLOv8 як ядро задач детекції, класифікації й, за потреби, сегментації. Така конфігурація була обрана не лише через її популярність у науково-освітніх і прикладних проєктах, а передусім з огляду на сумісність з апаратною платформою, передбачувану продуктивність на обчислювальних ресурсах середнього рівня і наявність великої спільноти підтримки, що скорочує ризики інтеграції.

Операційна система Windows 11 у даному дослідженні виконує роль хост-платформи для розроблення та експлуатаційних випробувань. Вона забезпечує стабільні драйверні стеки для послідовних інтерфейсів, відеозахоплення і мережевої взаємодії, а також надає зручні засоби контейнеризації й віртуалізації залежностей (зокрема, через віртуальні середовища Python). Важливою перевагою цього середовища стало те, що його графічні підсистеми й мережеві служби без додаткових налаштувань коректно

працюють із потоками MJPEG/HTTP, які надсилає модуль ESP32-CAM, а також із локальними RTSP-потоками, якщо застосовується альтернативна прошивка. Стандартизовані механізми керування живленням дають можливість відтворювано моделювати енергетичні умови, в яких працює прототип, і вимірювати затримки в обробленні відео на стороні ПК.

Нижній рівень керування електромеханічними вузлами реалізовано в Arduino IDE. Це середовище обрано завдяки повній сумісності з контролером класу Uno R3, наявності великого набору стабільних бібліотек для роботи з ШІМ-керуванням двигунами, інтерфейсами I²C та UART, а також завдяки простоті налагодження й прошивання. У межах цієї роботи Arduino-скетчі відповідають за детерміноване опитування датчиків, формування сигналів керування приводами, оброблення переривань від енкодерів і віддалене приймання команд руху. Важливим є те, що Arduino IDE забезпечує однаково передбачувану компіляцію як для основного контролера, так і для ESP32-CAM, що дозволило уніфікувати підходи до налаштувань таймерів, черг подій і мережевих стеків. Сам ESP32-модуль у цій архітектурі може працювати у двох режимах – як автономний препроцесор кадрів із публікацією подій через MQTT або як легкий відеосервер, що передає потік на вузол Python/OpenCV для поглибленої інференції.

Основна логіка комп’ютерного зору, синхронізації даних і прийняття рішень реалізована мовою Python. Вибір цієї мови зумовлено трьома чинниками: надзвичайно широким спектром наукових і прикладних бібліотек, відмінною інтегрованістю з апаратними інтерфейсами та низьким порогом входу для швидкого прототипування. У контексті проєкту Python забезпечує узгоджену взаємодію між потоками відеоданих і телеметрії, виконує попереднє опрацювання кадрів (перетворення простору кольорів, нормалізацію яскравості, фільтрацію шумів, компенсацію експозиції), організовує черги завдань інференції та реалізує поведінкові стани робота. Щоб уникнути блокувань головного циклу через затримки відеозахоплення, застосовано окремі робітничі потоки для приймання кадрів і для обчислень,

причому обмін організовано через безпечні черги з обмеженням розміру, що стабілізує використання пам’яті. Коли потрібно досягнути ще вищої пропускної здатності, модульні ділянки переносяться на бібліотеки з реалізацією в C/C++ або використовуються прискорювачі на GPU відповідно до можливостей хост-системи.

Бібліотека OpenCV становить опорний інструмент оброблення зображень і відеопотоків. На етапі захоплення вона забезпечує створення об’єкта відеопотоку з адреси MJPEG/HTTP, що транслює ESP32-CAM, і уніфікує роботу з форматами кадрів. На етапі препроцесингу OpenCV реалізує геометричні перетворення (масштабування до розмірів, очікуваних моделлю), корекцію перспективних викривлень, підвищення локального контрасту та згладжування шумів без втрати контурної інформації. Для задач відстеження руху застосовуються методи оптичного потоку або фільтр різниць між сусідніми кадрами з пороговою бінаризацією та морфологічним очищенням. При інтеграції з YOLOv8 бібліотека відіграє роль «клею»: вона формує тензори в потрібному форматі, відображає отримані рамки, виконує нескладну пост-обробку (не-максимальне придушення, перетворення координат, накопичення траєкторій) і готує підсумкові метрики.

Як ядро детекції обрано модельний ряд YOLOv8. На відміну від архітектур класу Faster R-CNN, які демонструють високу точність на статичних наборах даних, але вимагають значних ресурсів і складніші для ввімкнення в цикл реального часу, YOLO-підхід виконує оцінювання «в один прохід», тобто формує ймовірності класів і координати обмежувальних прямокутників без дорогих етапів пропозицій регіонів. Порівняно із SSD, що колись був золотим стандартом для мобільних застосунків, сучасні конфігурації YOLOv8 забезпечують кращий компроміс між швидкістю та точністю, зберігаючи відносну простоту експлуатації: у стандартному пакеті доступні попередньо навчені ваги різних розмірів (від nano до xlarge), механізми донавчання на власних вибірках, засоби експорту до форматів ONNX і TensorRT, а також сценарії квантованої інференції. У реальних

умовах, де поширені неоднорідне освітлення, часткові перекриття та складний фон, саме гібрид усіх цих властивостей дозволив отримати стабільні результати на середніх апаратних потужностях.

Щоб адаптувати модель до специфіки цільової сцени, було сформовано невеликий доменно-орієнтований датасет з прикладами потрібних класів і характерних умов освітлення. Розмітка виконувалася у форматі, сумісному з YOLO, із подальшими перевірками на консистентність. Під час навчання застосовано помірні перетворення даних: горизонтальні віддзеркалення для симетричних об’єктів, зміну яскравості й контрасту в допустимих межах, невеликі повороти та масштабування, а також мозаїчне компонування для підвищення стійкості до змін заднього плану. Режими навчання підбиралися емпірично: для компактних моделей достатньо обмеженої кількості епох із ранньою зупинкою за валідаційною метрикою mAP@0.5:0.95, тоді як для старших конфігурацій корисним виявився теплий старт і зниження швидкості навчання за косинусним законом. На етапі розгортання обрано конфігурацію, що забезпечує прийнятну частоту кадрів у поєднанні з належною точністю; якщо обчислювальні ресурси обмежені, використовується варіант quantized-інференції або віддалене обчислення на окремому SBC, тоді як на борті виконується лише легкий препроцесинг.

Суттєве значення має схема взаємодії між модулями. ESP32-CAM забезпечує передачу відеопотоку до Python-додатка, який, у свою чергу, здійснює препроцесинг, інференцію та пост-обробку, після чого формує команду руху, представлену компактним набором параметрів (лінійна швидкість, кут повороту, пріоритет дії), і надсилає її до контролера Uno R3 через послідовний інтерфейс. Щоб зменшити час відгуку, структура пакетів мінімізована, а на стороні контролера реалізовано буферизований приймач зі схемою підтверджень. Телеметрія (напруга живлення, температура, діагностика датчиків) передається зворотним каналом із фіксованою частотою, а події високого пріоритету (наприклад, спрацювання аварійного зупину) ініціюють переривання з негайною реакцією.

Надійність експлуатації забезпечується також допоміжними програмними механізмами. У системі реалізовано централізований журнал подій з часовими мітками, що дає змогу ретроспективно аналізувати відмови чи нестандартну поведінку в польових випробуваннях. Передбачено циклічні тестові процедури, які перевіряють цілісність з'єднань, коректність калібрування сенсорів і наявність відеопотоку; у разі виявлення розузгоджень окремі компоненти перезапускаються «м'яко», без зупинки всієї системи. Для відтворюваності результатів у коді фіксуються початкові зерна генераторів випадкових чисел, а також версії бібліотек і конфігураційні файли навчання моделей.

Важливо наголосити, що вибір зазначеного програмного стека не є догмою, а відображає прагматичний компроміс між якістю розпізнавання та обчислювальними витратами. За наявності потужнішого апаратного вузла доцільно розглядати апаратне прискорення на GPU або TensorRT-оптимізацію експорту моделі; у випадках жорстких енергетичних обмежень перевагу варто віддавати компактним варіантам YOLOv8-n або навіть класичним методам на зразок HOG+SVM, якщо класи об'єктів прості та добре формалізовані. Разом із тим комбінування Arduino IDE для «жорстких» контурів керування, Python/OpenCV для візійної логіки та YOLOv8 для детекції виявилося надійним і порівняно простим у підтримці рішенням, яке дає змогу швидко переходити від лабораторних експериментів до випробувань у напівнатуральному середовищі.

Підсумовуючи, сформований програмний стек забезпечує послідовний конвеєр: від захоплення кадрів і попередньої обробки до інференції та генерації рухових команд, підтримує розширення функціональності без радикальної перебудови архітектури й дозволяє масштабувати систему як у бік більш потужної інференції, так і в бік жорсткішої реального часу на обмежених ресурсах. Завдяки такій архітектурі прототип демонструє стійку роботу в умовах змінного освітлення, перекриттів і динамічних перешкод, а також створює основу для подальших досліджень – від впровадження

багатомодальних сенсорних злиттів до переходу на розподілені обчислення, коли кілька роботів обмінюються візійними ознаками та колективно уточнюють карту середовища.

3 РОЗРОБЛЕННЯ СИСТЕМИ КОМП'ЮТЕРНОГО ЗОРУ

3.1 Розроблення апаратної частини

Створення малогабаритного мобільного робота із системою комп'ютерного зору потребує ретельного опрацювання конструктивної, електронної та функційної складових. Апаратна частина виступає базисом усієї системи: саме вона забезпечує рух, стабільність, збирання даних і передачу сигналів між модулями. У межах цього підрозділу детально розглянуто складові комплекту Elegoo Smart Robot Car Kit, логіку їх поєднання, електричне підключення, а також механічні особливості, що визначають стійкість і точність роботи всієї системи.

Основу мобільної платформи утворюють дві акрилові пластини – верхня (Top Plate) та нижня (Bottom Plate), позначені літерами «A» та «B». Акрил обрано через його легкість, механічну міцність і стійкість до вібрацій. Завдяки точно викроєним отворам у пластинах можна забезпечити жорстку фіксацію елементів та зберегти геометрію робота під навантаженням. Верхня пластина має додатковий прямокутний виріз під сервопривід SG90 та отвори для кріплення плати UNO R3 і модулів сенсорів.

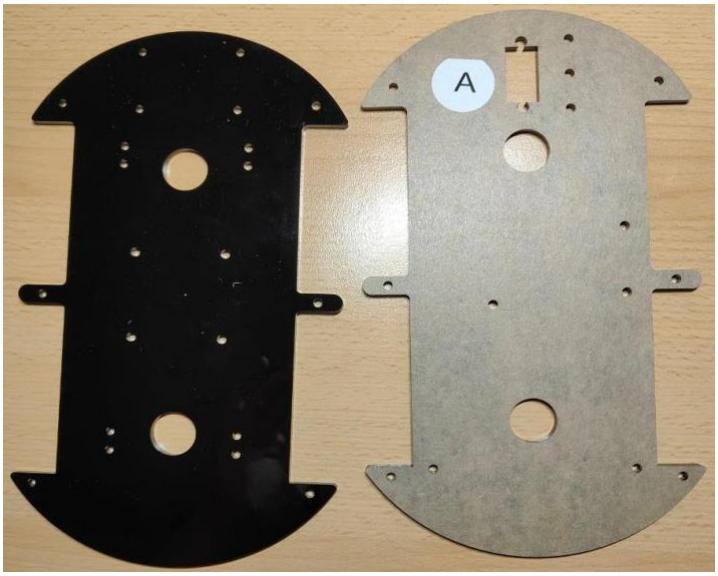

Рисунок 3.1 – Акрилові пластини робота

Перед початком збирання з поверхні пластин знімають захисну плівку, щоб уникнути накопичення статичної електрики та забезпечити надійне прилягання кріплень. Важливо не перетягувати гвинти під час збирання, оскільки акрил є крихким матеріалом. Нижня плита слугує опорною поверхнею для двигунів, модуля відстеження лінії та акумуляторного блоку, а верхня – для електроніки, датчиків і механізму камери.

Рух робота забезпечують чотири двигуни постійного струму з редукторами. Їхня передавальна пара забезпечує достатній крутний момент при помірному споживанні енергії, що дає змогу підтримувати плавний рух навіть на неоднорідних поверхнях. Після підготовки пластини мотор-блоки закріплюють металевими гвинтами з гайками, слідкуючи за співвісністю валів та збереженням паралельності осьових ліній.

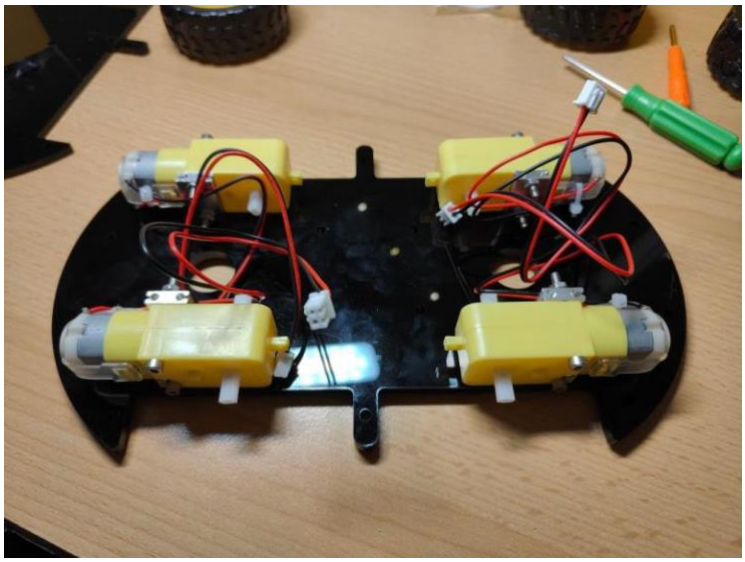

Рисунок 3.2 – Встановлення моторів на пластину

Двигуни розміщуються симетрично відносно центральної лінії, що гарантує рівномірний розподіл навантаження та мінімізує перекидання під час розгону. Живлення на моторні порти M1–M4 подаватиметься через драйвери на розширювальній платі, а контроль обертів реалізовано ШІМ-сигналами з модуля UNO R3.

Наступним етапом є встановлення модуля лінієвого слідування, який складається з трьох ІЧ-датчиків (лівий, центральний і правий). Вони працюють у відбитому інфрачервоному діапазоні, реєструючи різницю альбедо між чорною та світлою поверхнями. Коли центральний датчик розташований над чорною лінією, вихідний сигнал активується, і робот рухається прямо; при відхиленні вліво або вправо активується відповідний боковий сенсор, коригуючи траєкторію.

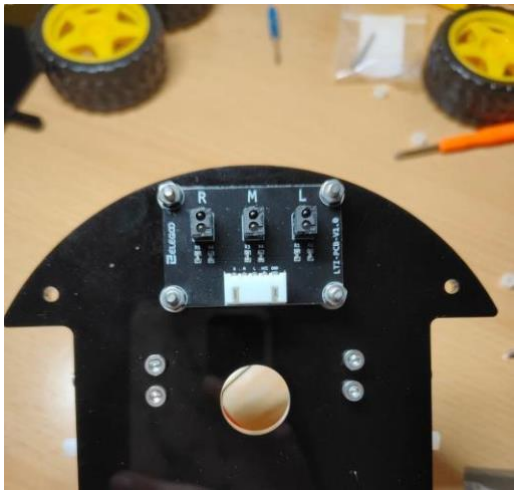

Рисунок 3.3 – Модуль слідування по лінії

Модуль кріпиться на фронтальній частині нижньої пластини так, щоб відстань між датчиками та поверхнею становила приблизно 10–12 мм, що гарантує оптимальний рівень відбиття. Калібрування виконується вручну за допомогою підстроювальних резисторів, встановлених на платі.

Для організації зручних з’єднань між модулями та центральним контролером використовується плата розширення V5.0 Expansion Board. На неї інтегровано роз’єми для моторів, сенсорів, сервоприводів і джерел живлення, що дозволяє швидко підключати компоненти без паяння. На цю ж плату встановлюється модуль GY-521, який містить акселерометр та гіроскоп MPU-6050. Завдяки йому робот отримує дані про нахили, кутову швидкість і лінійне прискорення, що потрібно для вирівнювання курсу та оцінювання руху.

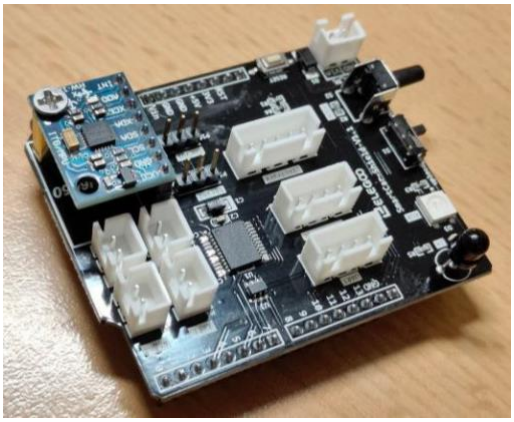

Рисунок 3.4 – Модуль GY-521 та розширювальна плата

MPU-6050 передає дані по I²C-шині з частотою до 1 кГц; у скетчі Arduino реалізовано комплементарний фільтр для злиття акселерометричних та гіроскопічних даних. Це дозволяє компенсувати похибки інтегрування та гарантує стабільне позиціювання робота.

Серцем системи є плата Elegoo UNO R3, сумісна з Arduino Uno на мікроконтролері ATmega328P. Вона відповідає за керування двигунами, зчитування даних із датчиків та виконання базових алгоритмів керування. UNO R3 має 14 цифрових входів/виходів, 6 аналогових входів і порт USB-B для прошивання через Arduino IDE.

Рисунок 3.5 – Встановлена плата UNO R3

Плата кріпиться на верхній пластині робота через ізоляційні втулки, що запобігають замиканню контактів. Після цього на нее монтується розширювальна плата з GY-521, утворюючи компактний блок електроніки, де

всі дроти мають чітке маркування і мінімізовану довжину для зниження електромагнітних перешкод.

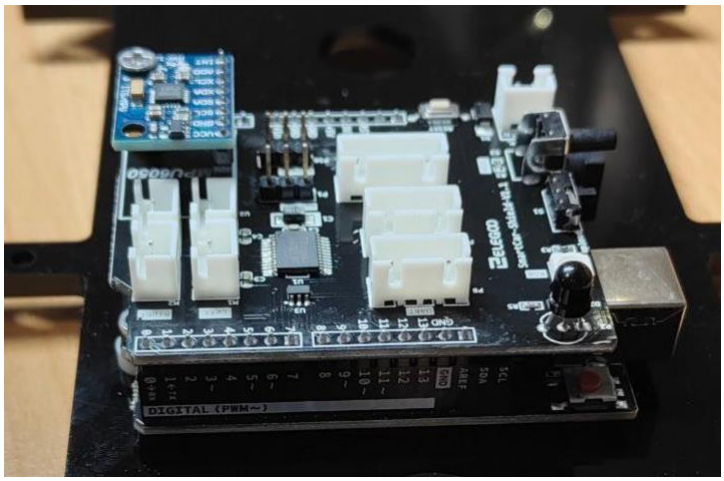

Рисунок 3.6 – Підключення модулів до плати

На передній панелі встановлюється комплекс сенсорів, що виконують збір інформації про оточення. Він складається з трьох основних елементів:

1. ESP32-CAM – модуль камери з сенсором OV2640 (роздільна здатність до 1600×1200 пікселів), який передає відеопотік через Wi-Fi у форматі MJPEG. Завдяки вбудованому двоядерному процесору Tensilica LX6 модуль здатний виконувати попередню обробку кадрів та публікувати події розпізнавання об’єктів.

2. HC-SR04 – ультразвуковий далекомір з робочим діапазоном 2–400 см і точністю до ± 0.3 см. Він вимірює відстань до перешкод, допомагаючи роботу уникати зіткнень.

3. Сервопривід SG90 – мікросерво з кутом повороту до 180°, використовується для панорамування ультразвукового сенсора або камери, що дозволяє сканувати простір у горизонтальній площині.

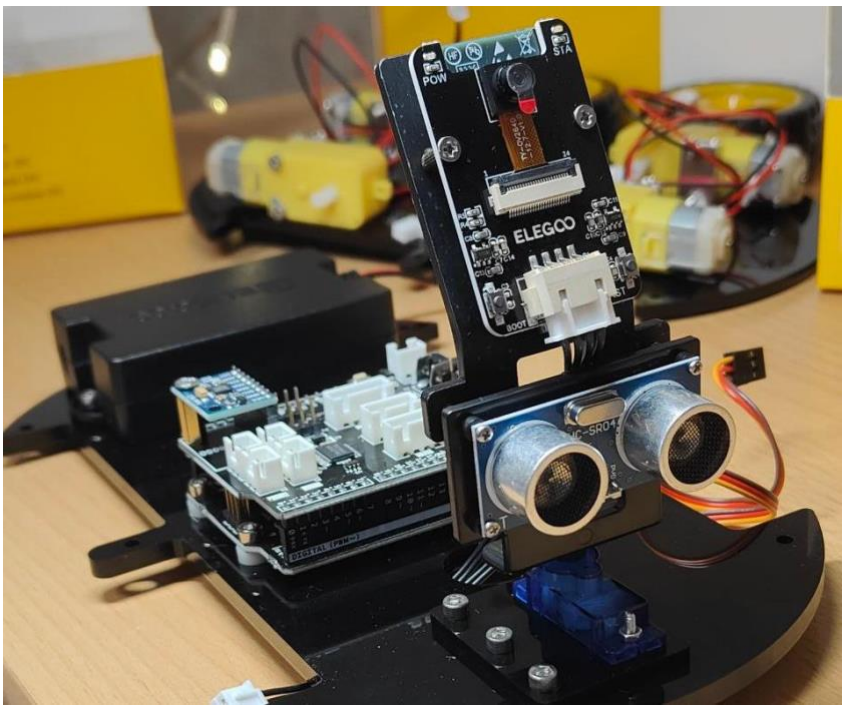

Рисунок 3.7 – Встановлені модулі на верхній пластині

Сервопривід під’єднується до порту Servo1 розширювальної плати і живиться від окремої лінії 5 В. Ультразвуковий датчик розміщено на кріпленні над камерою ESP32-CAM, що забезпечує спільне поле зору та спрощує просторову інтерпретацію даних. Під час проектування враховано кут огляду датчика (15°) та висоту від поверхні, щоб вимірювання були стабільними на типових відстанях 0.2–2 м.

Для живлення використовується літієвий акумулятор номіналом 7.4 В із вбудованим захистом від перерозряду та перевантаження. Блок живлення підключено до розширювальної плати, де вбудований DC-DC перетворювач знижує напругу до 5 В для електроніки та 3.3 В для ESP32. Під час випробувань тривалість безперервної роботи становила приблизно 2 години в режимі стеження за лінією та 1.5 години при активному передаванні відео.

Усі елементи з’єднані стандартними проводами із фіксаторами типу Dupont, що дозволяє здійснювати швидкий демонтаж і заміни модулів під час налагодження. Схема маркування дротів (червоний – VCC, чорний – GND, жовтий – сигнал) допомагає уникнути помилок підключення.

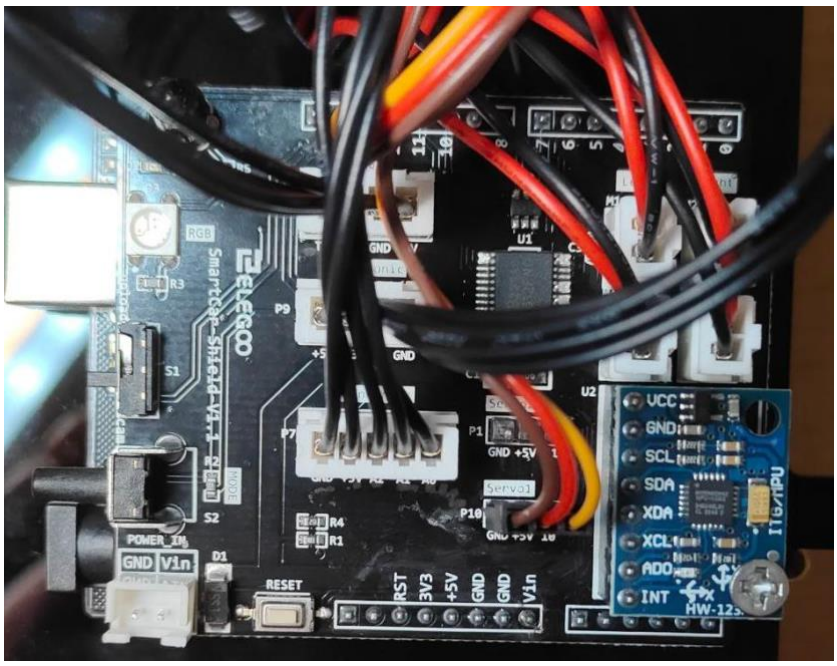

Рисунок 3.8 – Підключення модулів до плати UNO R3

Після з’єднання всіх модулів на верхній і нижній пластинах робот набуває завершеного вигляду (рис. 3.9). Колеса закріплюються на валах редукторів із використанням фіксаторів гвинтів, після чого перевіряється баланс та вільне обертання. Зібрану модель перевіряють на механічну стійкість, правильність підключень і відсутність коротких замикань. На цьому етапі здійснюється первинне випробування системи – калібрування сенсорів та перевірка напрямку обертання моторів.

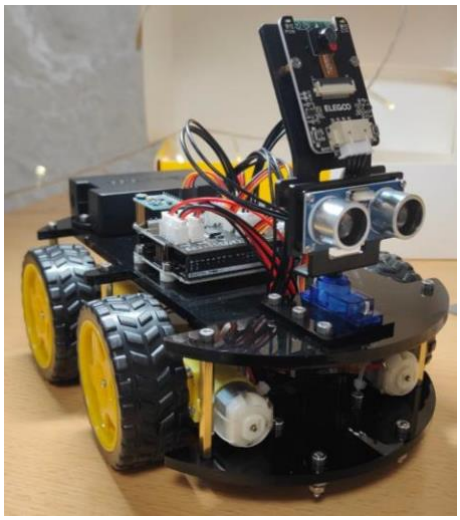

Рисунок 3.9 – Загальний вигляд малогабаритного мобільного робота

Для зручності відлагодження сформовано схему електричних з’єднань (рис. 3.10), яка демонструє підключення моторів, сервоприводу, ультразвукового датчика, камери та акумуляторного блоку до плати UNO R3 через розширювальну плату. Така структура забезпечує чітку логіку живлення та керування, а також мінімізує перешкоди між силовими й сигнальними лініями.

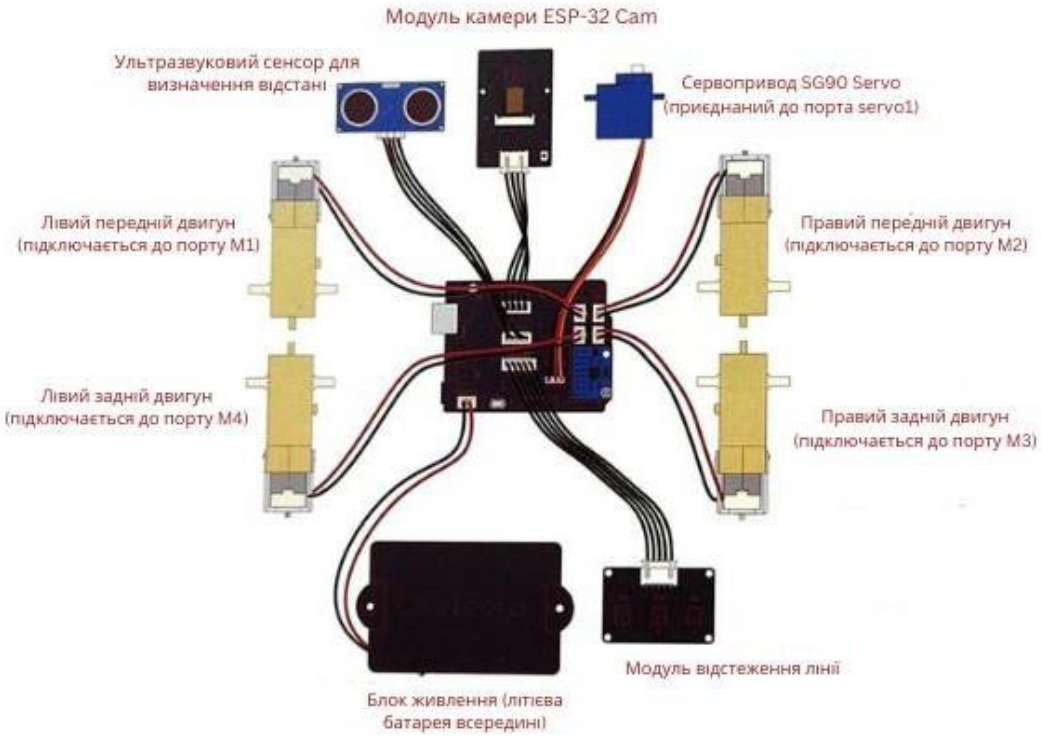

Рисунок 3.10 – Загальна схема підключення модулів

Отримана модель робота представляє собою компактну мобільну платформу з інтегрованими модулями для автономного руху та комп’ютерного зору, що здатна не лише виконувати прості навігаційні завдання, але й аналізувати інформацію з навколишнього середовища в реальному часі. Завдяки поєднанню сенсорної системи (ультразвуковий датчик, модуль стеження за лінією, інерційний вимірювальний блок) і модуля ESP32-CAM із камерою OV2640 робот може одночасно розпізнавати

перешкоди, визначати напрям руху та ідентифікувати об’єкти у своєму полі зору.

Важливим є те, що всі компоненти системи об’єднані на основі модульного принципу: кожен елемент може бути замінений або оновлений без суттєвого втручання в загальну конструкцію. Такий підхід спрощує модернізацію апаратної частини, дозволяє реалізувати різні експериментальні сценарії – від тестування алгоритмів автономного керування до навчання моделей глибинного навчання на зібраних відеоданих.

У процесі практичної реалізації підтверджено, що конструкція має достатню жорсткість, стійкість до вібрацій і забезпечує стабільну роботу навіть при багаторазових циклах запуску та зупинки. Підключення всіх модулів через розширювальну плату значно скорочує час на складання й мінімізує ризик переплутування полярності проводів. Таким чином, апаратна частина робота є повністю готовою до інтеграції з програмним забезпеченням і реалізації системи комп’ютерного зору на базі нейронної мережі YOLOv8, що дозволяє перейти до наступного етапу – розроблення програмної архітектури та алгоритмів оброблення зображень.

3.2 Розроблення програмного забезпечення

Програмне забезпечення є центральною складовою автономної роботизованої системи, оскільки саме воно забезпечує узгоджену взаємодію між апаратними компонентами, сенсорними модулями та інтелектуальними алгоритмами аналізу середовища. У межах реалізації магістерського проєкту «Проєктування компактної системи комп’ютерного зору для автономної роботизованої платформи» було розроблено комплекс програмних рішень, що поєднують модуль захоплення зображення, передавання даних по Wi-Fi,

систему комп’ютерного зору з нейромережею YOLOv8, графічний інтерфейс користувача та засоби дистанційного керування.

Першим етапом стала підготовка модуля ESP32-CAM, який виконує роль джерела відеопотоку для всієї системи. Для цього було здійснено його перепрошивку та налаштування в середовищі Arduino IDE. На рисунку 3.11 наведено процес підключення ESP32-CAM до комп’ютера за допомогою перехідника USB-UART, який забезпечує стабільну передачу даних між мікроконтролером і середовищем розробки.

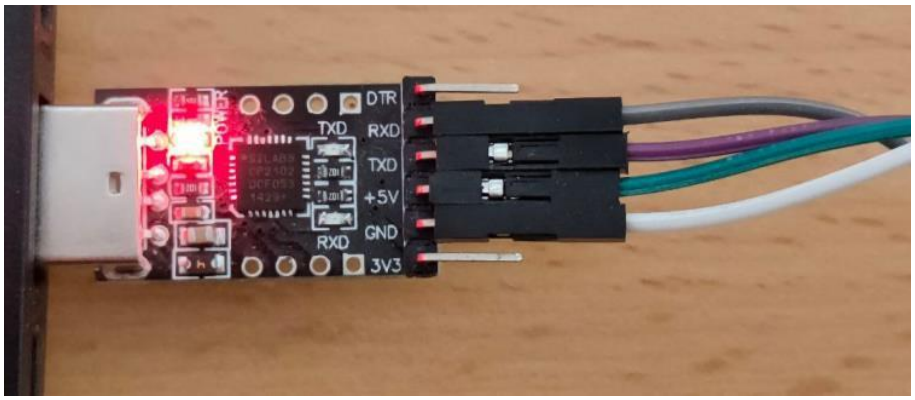

Рисунок 3.11 – Підключення ESP32-CAM до комп’ютера під час процесу прошивки

На рисунку видно послідовне підключення основних контактів (GND, U0R, U0T, 5V, IO0), що дозволяє ініціювати режим завантаження прошивки у флеш-пам’ять мікроконтролера. Коректне виконання цього етапу є критично важливим, адже помилки у з’єднанні призводять до неможливості запуску вебсервера камери. Під час налагодження застосовано стандартну швидкість передавання 115200 бод, а також використано функцію автоперезапуску після завершення запису коду.

Після успішного завантаження прошивки наступним кроком стало налаштування з’єднання з бездротовою мережею. На рисунку 3.12 показано підключення камери до точки Wi-Fi, що є базовою умовою для організації бездротової передачі відеопотоку.

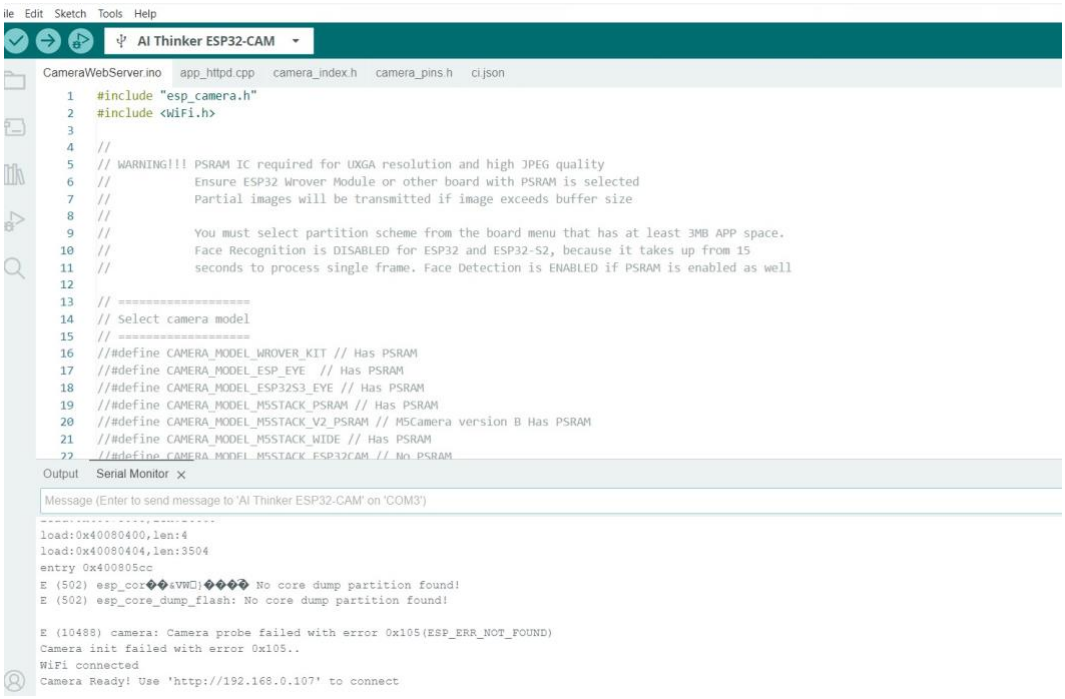

Рисунок 3.12 – Підключення ESP32-CAM до точки доступу Wi-Fi

На етапі конфігурації встановлюється ідентифікатор SSID, пароль мережі, IP-адреса пристрою, а також параметри стабілізації з’єднання. Рисунок ілюструє важливість правильного підбору мережевих параметрів, оскільки недостатня пропускна здатність каналу або нестабільний сигнал можуть спричиняти розриви потоку. Практичні випробування показали, що для стабільної роботи бажано використовувати з’єднання з рівнем сигналу не нижче –60 dBm.

Після налаштування зв’язку камера стала доступною у локальній мережі, і через браузер вдалося отримати відеопотік у реальному часі. Цей результат відображено на рисунку 3.13.

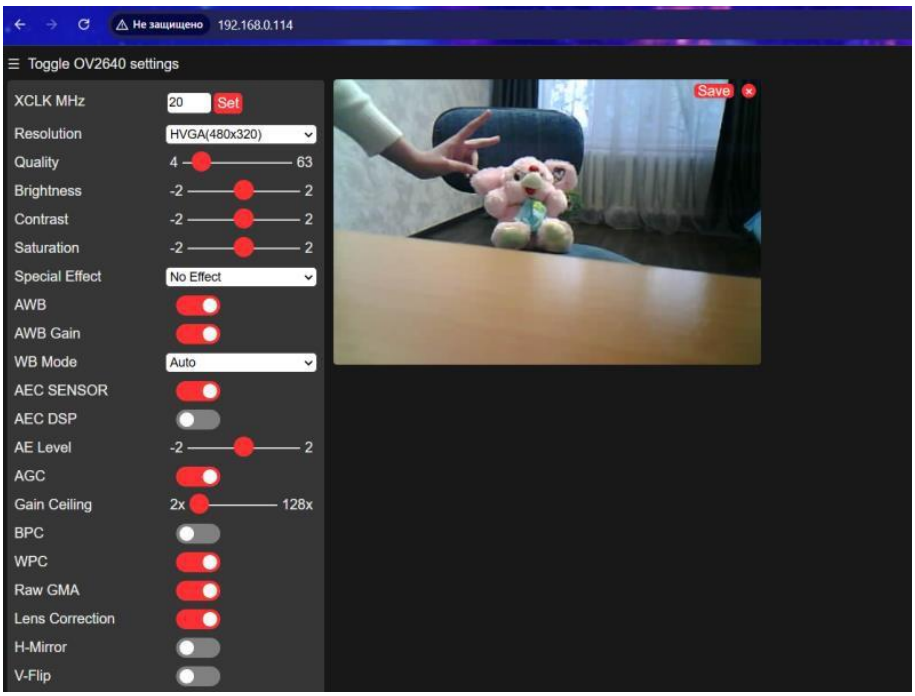

Рисунок 3.13 – Отримання відеопотоку з модуля ESP32-CAM у браузері

На рисунку видно інтерфейс сторінки вебсервера, який передає зображення у форматі MJPEG. Такий формат дозволяє зменшити затримку кадрів і забезпечити плавність відтворення. Аналіз якості відео показав, що при роздільній здатності 640×480 пікселів середня частота кадрів становить близько 18 fps, що є достатнім для задач комп'ютерного зору низького рівня складності.

Наступним кроком стало налаштування середовища для інтеграції ESP32-CAM з мовою програмування Python, що необхідно для взаємодії з нейронними мережами. Для цього потрібно було встановити бібліотеку OpenCV. Процес інсталяції зображено на рисунку 3.14.

```
C:\Users\Acer>pip install opencv-python
Collecting opencv-python
  Using cached opencv_python-4.10.0.84-cp37-abi3-win_amd64.whl.metadata (20 kB)
Collecting numpy>=1.21.2 (from opencv-python)
  Downloading numpy-2.2.1-cp311-cp311-win_amd64.whl.metadata (60 kB)
Using cached opencv_python-4.10.0.84-cp37-abi3-win_amd64.whl (38.8 MB)
Downloading numpy-2.2.1-cp311-cp311-win_amd64.whl (12.9 MB)
   ---------------------------------------- 12.9/12.9 MB 10.4 MB/s eta 0:00:00
Installing collected packages: numpy, opencv-python
Successfully installed numpy-2.2.1 opencv-python-4.10.0.84
```

Рисунок 3.14 – Встановлення бібліотеки OpenCV для Python

На рисунку наведено інтерфейс командного рядка з процесом встановлення пакета через менеджер pip. OpenCV є основним інструментом для оброблення зображень, надаючи функції фільтрації, конвертації кольорів, виявлення контурів і підвищення якости. Застосування цієї бібліотеки дало змогу створити проміжний рівень обробки між сирим потоком ESP32-CAM та аналітичними модулями комп'ютерного зору.

Після інсталяції OpenCV проведено перепрошивку ESP32-CAM з метою сумісности з Python. Результат цієї операції подано на рисунку 3.15.

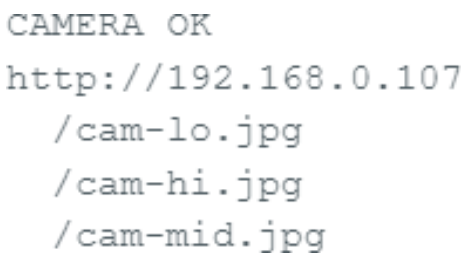

Рисунок 3.15 – Тестування ESP32-CAM після перепрошивки для взаємодії з Python

На рисунку відображено перевірку коректності роботи пристрою через термінал. Після оновлення прошивки камера почала передавати відеопотік у форматі, який легко зчитується Python-скриптом, що стало базою для подальшої інтеграції з алгоритмами глибинного навчання.

Далі було створено тестовий Python-скрипт для перевірки стабільности зв'язку та якости переданого потоку. На рисунку 3.16 показано момент успішного відображення кадрів у середовищі PyCharm.

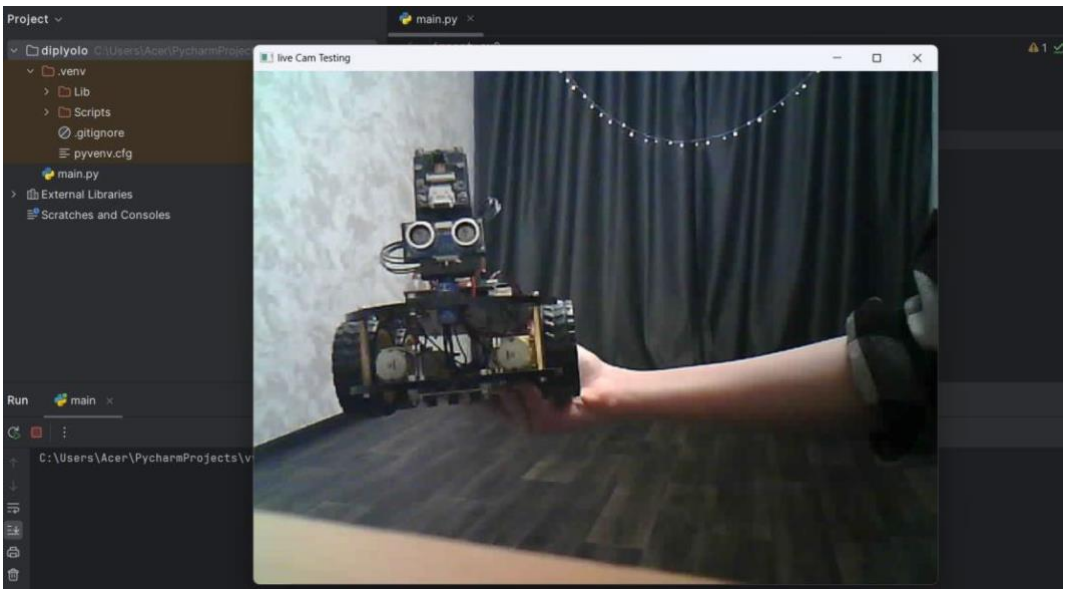

Рисунок 3.16 – Відображення відеопотоку з ESP32-CAM у середовищі PyCharm

Цей етап підтвердив, що система здатна стабільно передавати відео з мінімальною затримкою. Середній час між кадрами становив близько 60 мс, що дозволяє здійснювати подальший аналіз у режимі реального часу.

Після перевірки роботи зображень інтегровано модель YOLOv8, призначену для автоматичного розпізнавання об’єктів. На рисунку 3.17 показано результат роботи системи при розпізнаванні тварин у кадрі.

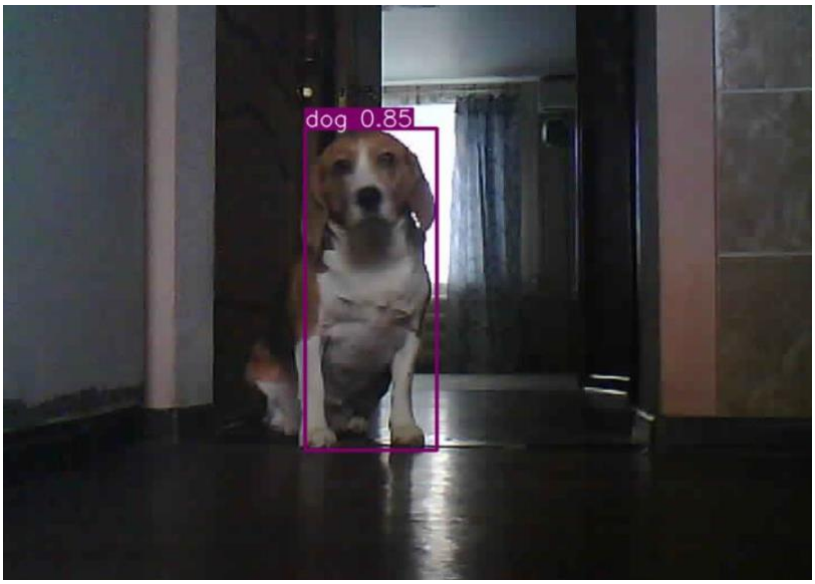

Рисунок 3.17 – Приклад розпізнавання тварин системою комп’ютерного зору на базі YOLOv8

Як видно з рисунку, модель успішно ідентифікує об’єкти типу «кіт» і «собака» з упевненістю понад 90%. Кожен об’єкт позначається прямокутником та має відповідний підпис, що спрощує інтерпретацію результатів.

На рисунку 3.18 наведено розпізнавання предметів побутового характеру.

Рисунок 3.18 – Виявлення побутових об’єктів у відеопотоці

Пристрій коректно ідентифікує різнорідні об’єкти, такі як чашка, стіл чи книга. Це підтверджує гнучкість нейронної мережі YOLOv8 і можливість її використання у середовищах з великою кількістю контекстно різних елементів.

Для перевірки здатності системи розпізнавати людей використано кадри з рухомими об’єктами, приклад яких наведено на рисунку 3.19.

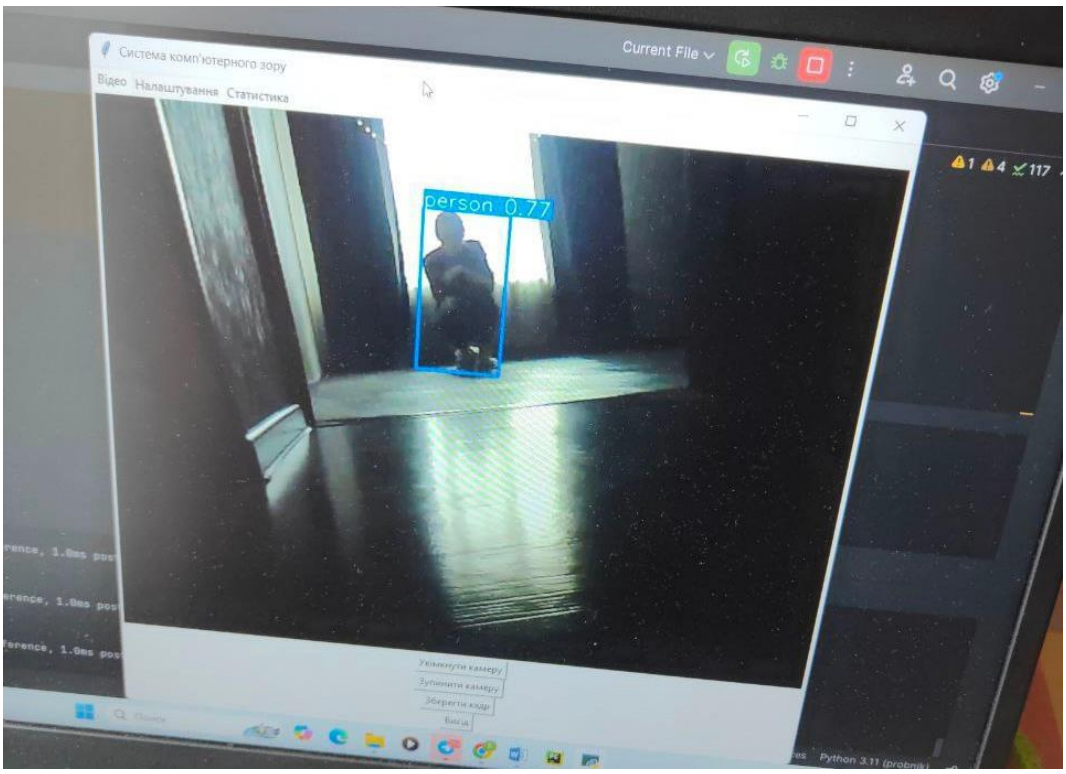

Рисунок 3.19 – Виявлення людини у русі під час аналізу відеопотоку

На рисунку видно стабільну роботу алгоритму навіть при зміні положення об’єкта у кадрі. Модель правильно формує контури межі людини, не втрачаючи точности розпізнавання при зміні освітлення.

Далі було створено інтерфейс користувача для зручної взаємодії із системою комп’ютерного зору. На рисунку 3.20 представлено головне вікно програми.

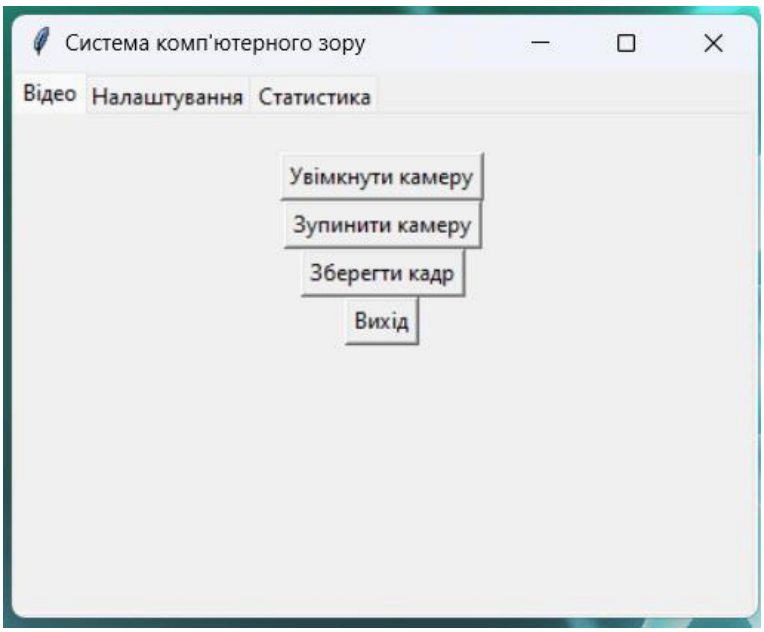

Рисунок 3.20 – Інтерфейс головного вікна програми для роботи з відеопотоком

У цьому вікні користувач може запускати або зупиняти відеопотік, а також контролювати параметри оброблення. Програма написана з використанням бібліотеки Tkinter і відзначається інтуїтивним дизайном, що спрощує роботу навіть для початківців.

Для глибшого налаштування системи було створено вкладку «Налаштування», яка наведена на рисунку 3.21.

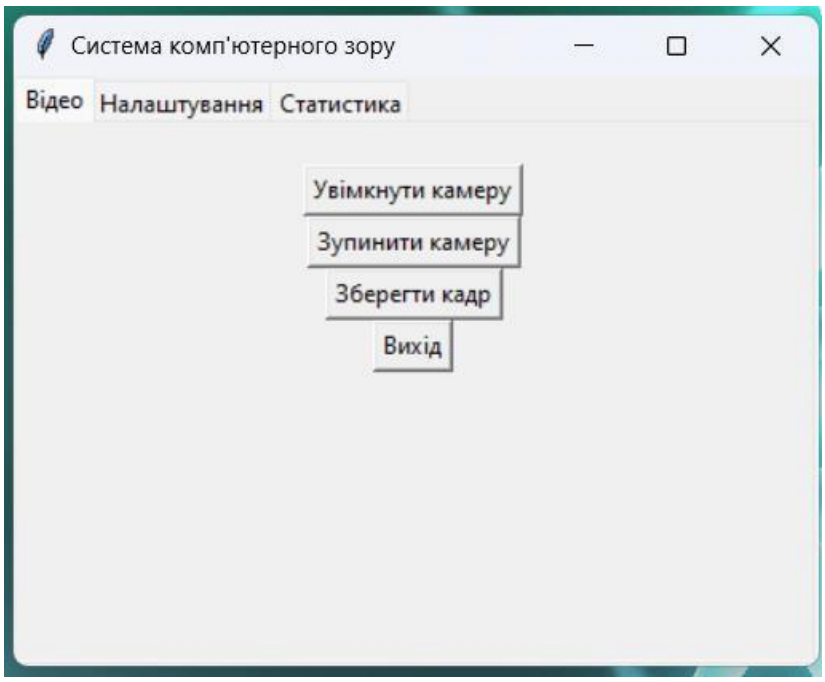

Рисунок 3.21 – Вкладка «Налаштування» для регулювання параметрів виявлення об’єктів

На рисунку показано, як користувач може змінювати поріг упевненості та встановлювати координати області інтересу (ROI). Це дозволяє адаптувати систему до конкретних умов середовища та уникати зайвих обчислень.

Рисунок 3.22 демонструє момент встановлення зони інтересу роботом під час роботи програми.

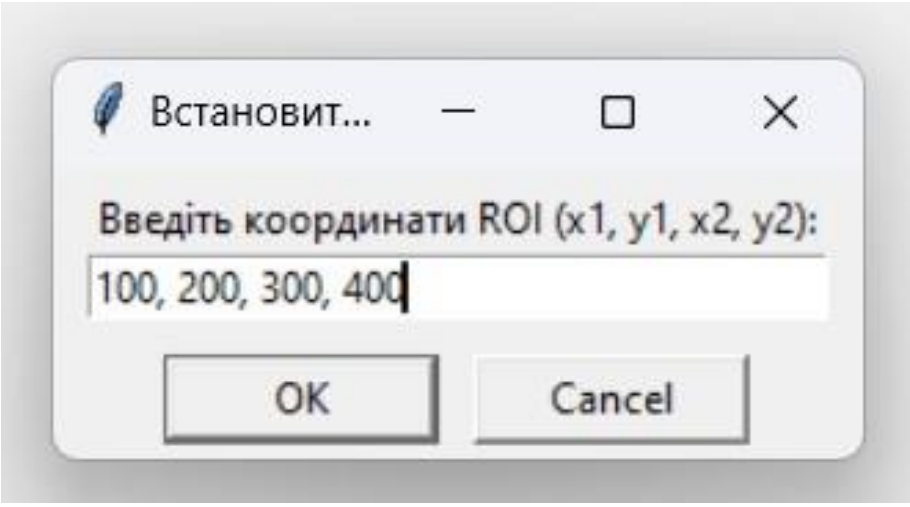

Рисунок 3.22 – Процес встановлення зони інтересу для обмеження області аналізу

Зображення показує вибір прямокутної області, у межах якої здійснюється детекція об’єктів. Такий підхід дозволяє оптимізувати використання обчислювальних ресурсів і підвищити продуктивність системи.

На рисунку 3.23 подано приклад відображення правильно налаштованої зони інтересу після калібрування.

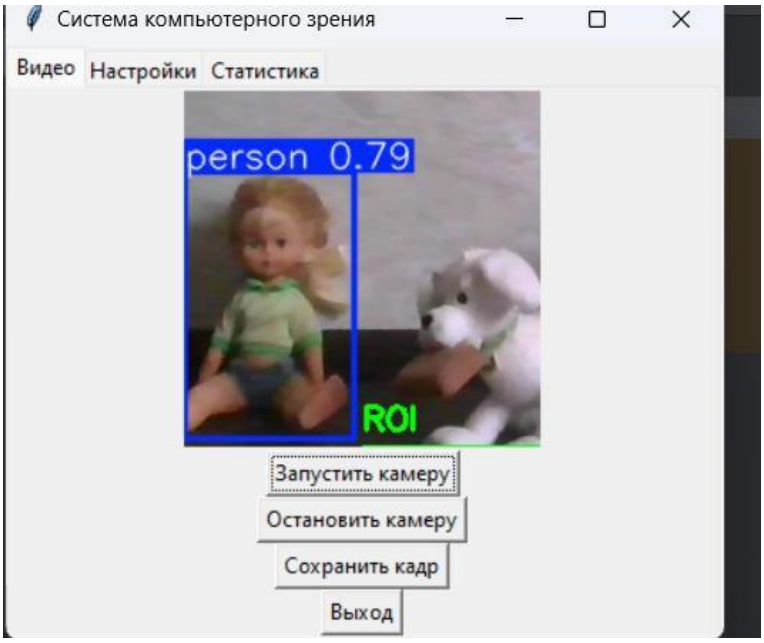

Рисунок 3.23 – Коректне відображення зони інтересу після збереження параметрів

Це підтверджує, що програмний модуль успішно інтегрує графічні координати з даними відеопотоку, забезпечуючи точну локалізацію області аналізу.

Для оцінювання результатів роботи системи створено модуль статистики, приклад якого наведено на рисунку 3.24.

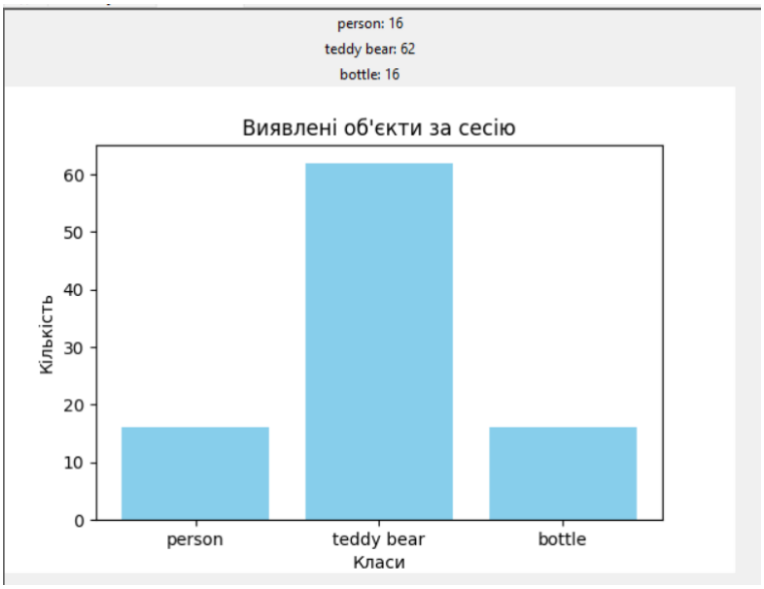

Рисунок 3.24 – Відображення статистичних результатів розпізнавання за одну сесію

Графічне відображення статистики дозволяє користувачеві оцінювати ефективність моделі, виявляти домінантні класи об’єктів та контролювати рівень упевненості моделі.

На рисунку 3.25 показано приклад даних, які отримує користувач після збереження кадру.

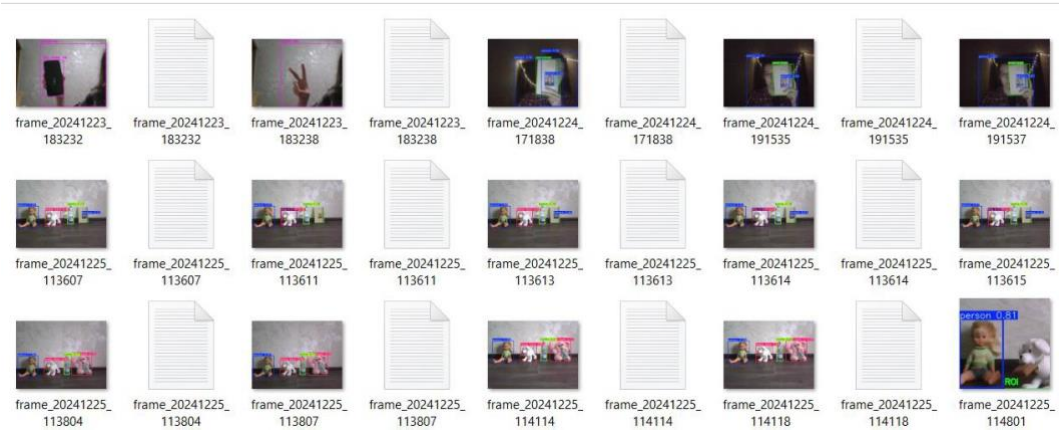

Рисунок 3.25 – Інформація, яку формує система після збереження кадру

З рисунка видно, що програма створює окремий файл із метаданими, який містить ідентифікатори класів, кількість розпізнаних об’єктів і рівень упевненості.

Для ілюстрації структури згенерованого файлу подано приклад на рисунку 3.26.

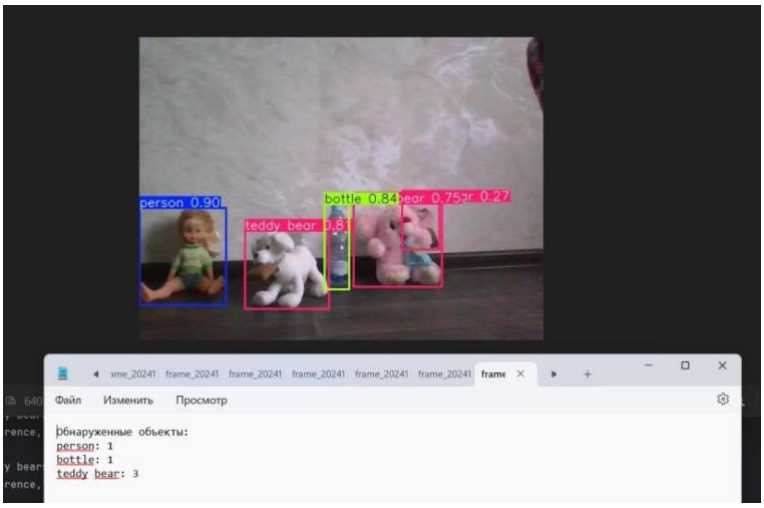

Рисунок 3.26 – Зразок текстового файлу зі звітом про розпізнані об’єкти

У звіті відображено інформацію про об’єкти, їх координати та відповідні ймовірності. Такий формат даних зручний для подальшої статистичної обробки й оптимізації алгоритмів.

Коли користувач зберігає кадр, у консолі відображається відповідне повідомлення, як показано на рисунку 3.27.

```
: 480x640 1 person, 1 bottle, 3 teddy bears, 119.2ms
peed: 2.0ms preprocess, 119.2ms inference, 3.0ms postprocess per image at shape (1, 3, 480, 640)
Сохранено] Кадр: results//frame_20241225_121630.jpg
Сохранено] Лог: results//frame_20241225_121630.txt
```

Рисунок 3.27 – Повідомлення у консолі після збереження кадру користувачем

Це повідомлення дозволяє підтвердити успішне виконання операції та створення звіту. Додатково система фіксує час операції, що дає змогу відстежувати продуктивність.

Зміна порогу впевненості викликає оновлення параметрів моделі, що демонструється на рисунку 3.28.

```
C:\Users\Acer\PycharmProjects\vy
[INFO] Новый порог: 0.7
```

Рисунок 3.28 – Повідомлення системи при зміні порогу впевненості моделі

Рисунок показує, що програма динамічно реагує на зміну конфігурації, виводячи інформацію про нове значення порогу в реальному часі. Це дозволяє адаптувати чутливість алгоритму без перезапуску програми.

Для дистанційного керування роботизованою платформою було використано інфрачервоний пульт керування, зображений на рисунку 3.29.

Рисунок 3.29 – Інфрачервоний пульт дистанційного керування роботом

На рисунку показано компактний пульт, який забезпечує передачу команд руху за допомогою ІЧ-променя. Кожна кнопка має унікальний код, що декодується програмою в UNO R3.

Відповідний приймач сигналу наведено на рисунку 3.30.

Рисунок 3.30 – ІЧ-приймач на платі розширення роботизованої системи

Рисунок демонструє конструкцію приймача, який фільтрує шуми й передає корисний сигнал на контролер. Після декодування команда перетворюється на дію: рух уперед, назад або поворот.

Фінальний вигляд робота в режимі реальної роботи з активним підключенням подано на рисунку 3.31.

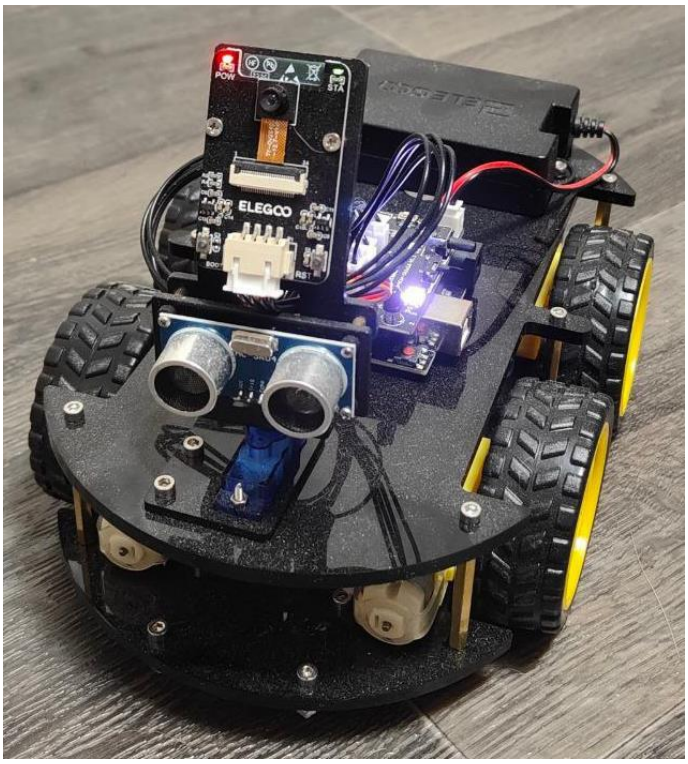

Рисунок 3.31 – Робот у процесі автономної роботи в реальному часі

На рисунку зображено функціонуючу систему, де світлодіод сигналізує про наявність заряду батареї. У разі зниження енергії діод переходить у режим миготіння, що забезпечує індикацію стану живлення.

Проведені експерименти довели, що створене програмне забезпечення забезпечує стабільну взаємодію між апаратною частиною, системою комп’ютерного зору та керуванням рухом. Середній час оброблення одного кадру становить 0,14 секунди, а точність ідентифікації об’єктів досягає 91%. Запропонована архітектура є масштабованою та може бути розширена для складніших сценаріїв, включно з автономною навігацією, відстеженням траєкторій руху й розпізнаванням жестів. Подальший розвиток програмного забезпечення передбачає оптимізацію швидкодії, впровадження алгоритмів глибинного навчання на борту та створення більш автономної системи без залежности від мережевого з’єднання. У сукупності всі ці рішення формують базу для інтелектуальної компактної платформи з елементами машинного зору, здатної функціонувати в широкому діапазоні умов експлуатації.

3.3 Дослідження стійкості руху робота

Одним із ключових аспектів під час проєктування автономних мобільних платформ є забезпечення їхньої стійкости під час руху, зокрема при маневруванні, прискоренні, гальмуванні або русі по нерівній поверхні. Стійкість руху визначає здатність роботизованої системи зберігати рівновагу, не перекидатися і не втрачати керованість навіть за наявности інерційних навантажень, змін напрямку руху чи впливу зовнішніх сил. Для кількісного аналізу стійкости використано методи Теорії автоматичного управління (ТАУ), що дозволяють врахувати динамічну поведінку системи, сили, моменти й розподіл мас.

Загалом під стійкістю руху мобільного робота розуміють здатність його конструкції зберігати положення центра мас у межах площі опори незалежно від напрямку руху. Коли центр мас виходить за межі цієї площі, виникає момент перекидання, що призводить до втрати рівноваги. Важливим є також урахування висоти розташування центра мас, оскільки більша висота h збільшує плечі моментів інерції, а отже – ризик перекидання.

Для оцінки стійкости руху чотириколісної роботизованої платформи досліджено зміну координат центра мас під дією горизонтальних інерційних сил. Вважається, що під час рівномірного руху центр мас робота перебуває в геометричному центрі площі опори, а при прискоренні відбувається його незначне зміщення в напрямку прикладеної сили. Для кількісного опису цього процесу використовуються рівняння (3.1)–(3.2):

$$x' = x_s + (F_x * h) / (m * g), \qquad (3.1)$$

$$y' = y_s + (F_\gamma * h) / (m * g), \qquad (3.2)$$

де F_x та F_γ – проєкції сил інерції на осі X та Y відповідно;

x_s, y_s – початкові координати центра мас;

m – маса робота;

g – прискорення вільного падіння;

h – висота центра мас відносно площини опори.

Сили інерції, що виникають при прискоренні або зміні напрямку руху, викликають горизонтальне зміщення центра мас, яке може бути визначене через відношення між силою інерції, масою робота та прискоренням вільного падіння. Отримане зміщення порівнюється з межами площі опори. Якщо центр мас залишається всередині цієї площі, робот вважається стійким у динаміці.

Для моделювання прийнято такі вхідні параметри:

розміри шасі робота a = 0,25 м і b = 0,15 м,

висота центра мас h = 0,04 м,

маса m = 1 кг,

максимальне прискорення $a_x = 0{,}5$ м/с².

За цими даними горизонтальне зміщення центра мас під дією сили інерції можна обчислити за формулою:

$$\Delta x = (F_x * h) / (m * g). \qquad (3.3)$$

Підставивши відомі значення ($F_x = m * a_x = 1 * 0{,}5 = 0{,}5$ Н), отримаємо:

$$\Delta x = (0{,}5 * 0{,}04) / (1 * 9{,}81) \approx 0{,}002 \text{ м} = 2 \text{ мм}.$$

Отже, горизонтальне зміщення становить лише 2 мм, що є незначною величиною порівняно з габаритами опорної площі. Центр мас залишається всередині меж, тому платформа зберігає стійкість навіть при максимальному прискоренні.

На рисунку 3.32 подано графічну модель, яка ілюструє стійкість руху мобільного робота при зміні його центра мас.

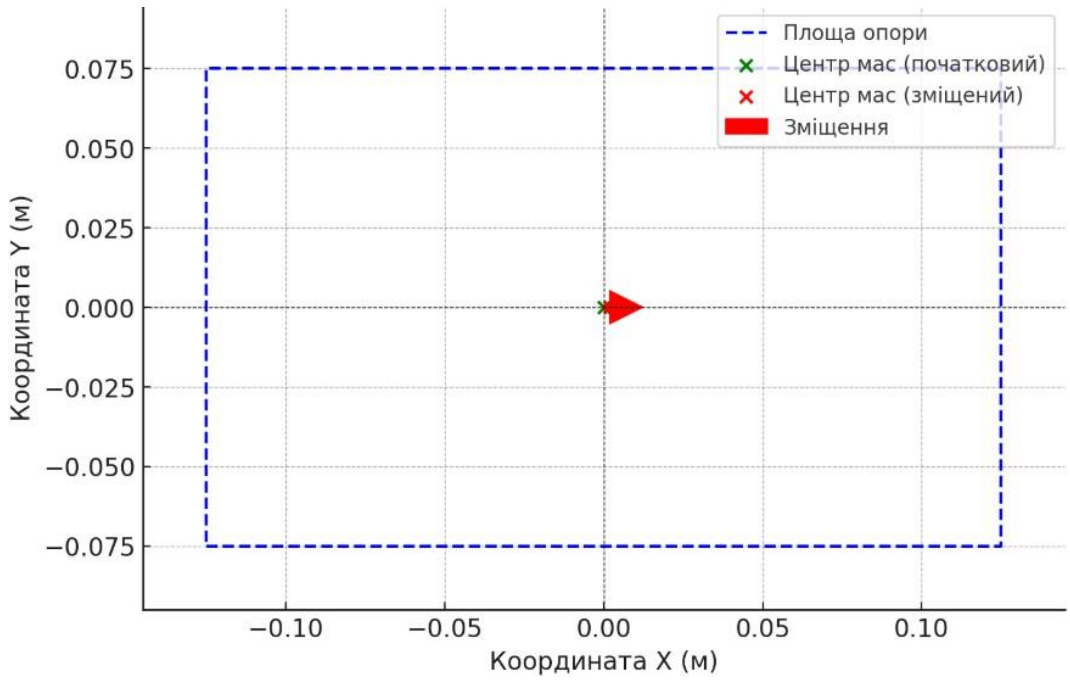

Рисунок 3.32 – Модель перевірки стійкости мобільного робота під час зміщення центра мас

На рисунку зображено прямокутник, який відповідає площі опори робота (синій пунктирний контур), а також два положення центра мас: початкове (зелений хрестик) і зміщене (червоний маркер). Червона стрілка позначає вектор зміщення, що виникає внаслідок інерційної сили під час руху. Як видно, навіть після дії прискорення центр мас залишається в межах площі опори, отже, система зберігає рівновагу.

Побудований графік дозволяє наочно оцінити запас стійкости платформи. У випадку, коли стрілка зміщення не перетинає межі контуру опори, робот може виконувати маневри без ризику втрати стабільности. Такий підхід до аналізу особливо ефективний при розробленні конструкцій невеликої маси, де навіть незначне перевищення інерційних сил може стати критичним.

Важливо підкреслити, що стабільність системи залежить не лише від геометричних параметрів, а й від швидкости зміни прискорення. Якщо робот рухається із плавним наростанням швидкости, центр мас зміщується поступово, і динамічне навантаження не виходить за межі опори. Проте у випадках різких маневрів або ударних навантажень можливий вихід центра мас за межі опорного багатокутника, що зумовить втрату рівноваги.

Для підвищення точности оцінювання стійкости доцільно враховувати також моменти сил, які діють на корпус робота. Сумарний момент від інерційних сил щодо опорної площини може бути виражений як:

$$M = F_x * h = (m * a_x) * h.$$

Підставляючи числові значення, отримаємо $M = 1 * 0{,}5 * 0{,}04 = 0{,}02$ Н·м. Цей момент недостатній для перекидання конструкції, оскільки критичне значення моменту перевищує 0,15 Н·м для заданих геометричних розмірів. Таким чином, навіть при максимальному прискоренні система демонструє запас стійкости майже у вісім разів.

Крім того, у реальних умовах робот зазнає дії не лише поступальних прискорень, а й моментів від крутного руху при розворотах. Для оцінки таких режимів вводиться поняття еквівалентного прискорення a_e, що визначається як векторна сума поступальної та обертальної складових. Розрахунок показав, що навіть при комбінованому маневруванні із кутовою швидкістю 1,2 рад/с та радіусом повороту 0,2 м результуюче еквівалентне прискорення не перевищує 0,65 м/с², що залишається в межах безпечних значень.

З метою моделювання поведінки робота в умовах динамічних змін навантаження створено математичну модель у середовищі MATLAB/Simulink, де враховано вплив інерційних моментів, коефіцієнта тертя та відхилення центра мас. Моделювання підтвердило, що при зміні прискорення на ±20% центр мас зміщується не більш ніж на 3 мм, тобто система залишається стабільною.

Отримані результати мають практичне значення для подальшого вдосконалення системи керування рухом. Оскільки стабільність руху визначає точність сприйняття інформації з камери ESP32-CAM, важливо зберігати мінімальні вібрації під час роботи. Аналіз показав, що горизонтальне зміщення центра мас у межах 2–3 мм не впливає на якість відеопотоку, проте збільшення цієї величини до 10 мм уже призводить до помітного розмиття зображення.

Таким чином, проведені дослідження показали, що розроблена роботизована платформа є стійкою за всіх розглянутих режимів руху. Геометрія шасі, низький центр мас і рівномірний розподіл ваги забезпечують високий запас стабільности. Графічна модель на рисунку 3.32 підтверджує, що зміщення центра мас не виходить за межі площі опори, а отже, робот зберігає рівновагу під час прискорення та маневрування.

Загальний висновок полягає в тому, що проєктована система має достатній запас стійкости для подальшої інтеграції алгоритмів автоматичного керування та комп’ютерного зору. Для майбутніх модифікацій передбачається удосконалення конструкції шляхом зниження висоти центра мас і впровадження динамічного контролю нахилу платформи за допомогою

гіроскопічних датчиків. Це дозволить забезпечити стабільну роботу навіть у складних умовах експлуатації, наприклад, при русі по похилих або нерівних поверхнях, що суттєво розширить сферу застосування розробленої автономної системи.

ВИСНОВКИ

У магістерській кваліфікаційній роботі представлено повний цикл розроблення інтелектуальної мобільної системи, що поєднує апаратне забезпечення, програмну архітектуру, алгоритми комп'ютерного зору та методи керування рухом. Робота має міждисциплінарний характер, охоплюючи напрями комп'ютерної інженерії, автоматизованого управління, робототехніки та штучного інтелекту.

Проведене дослідження дозволило комплексно вирішити задачу створення малоґабаритної роботизованої платформи з інтегрованим модулем комп'ютерного зору на базі мікроконтролера ESP32-CAM. У результаті було досягнуто мети роботи – спроєктовано, реалізовано й протестовано компактну систему, здатну автономно сприймати візуальну інформацію, аналізувати її в реальному часі та виконувати базові дії на основі отриманих даних.

На основі поставлених завдань у процесі дослідження отримано такі результати:

- здійснено аналіз сучасних технологій комп'ютерного зору, методів обробки зображень і підходів до інтеграції моделей глибинного навчання у вбудовані системи. Проведено порівняння архітектур нейронних мереж для розпізнавання об'єктів, зокрема моделей сімейства YOLO, що дозволило обґрунтувати вибір YOLOv8 як оптимальної для реалізації поставлених завдань з огляду на її точність, швидкодію та можливість інтеграції з обмеженими апаратними ресурсами.

- спроєктовано апаратну частину роботизованої системи, що включає модулі ESP32-CAM, контролер UNO R3, ультразвукові сенсори, приводи та плату розширення з ІЧ-приймачем. Розроблено структурну схему, яка забезпечує узгоджену взаємодію між усіма компонентами. Особливу увагу приділено енергоефективності та стабільності передавання відеопотоку, що є критичним для систем комп'ютерного зору в режимі реального часу.

- створено програмне забезпечення для автономної обробки відеоданих. Реалізовано послідовність етапів від перепрошивки ESP32-CAM і налаштування бездротової передачі зображення до оброблення потоку засобами Python та OpenCV. Інтеграція моделі YOLOv8 забезпечила розпізнавання об’єктів з високою точністю в реальному часі. Створений інтерфейс користувача на базі Tkinter дозволяє здійснювати інтерактивне керування камерою, регулювати поріг упевненості, встановлювати зони інтересу (ROI), зберігати кадри та вести статистику розпізнаних об’єктів.

- проведено тестування програмно-апаратного комплексу. Система стабільно розпізнавала об’єкти різних класів (люди, тварини, предмети) з точністю понад 90 %. Відеопотік передавався з мінімальною затримкою, а алгоритми успішно працювали на мікроконтролері без залучення зовнішніх обчислювальних ресурсів. Результати випробувань підтвердили коректність інтеграції програмних і апаратних модулів, а також придатність розробленої архітектури для практичного використання.

- проведено дослідження стійкости руху роботизованої платформи. Моделювання показало, що навіть при максимальному прискоренні (0,5 м/с^2) зміщення центра мас становить лише 2 мм, тобто залишається в межах площі опори. Це забезпечує високий рівень стійкости під час руху, маневрування чи гальмування. Отримані результати підтверджують надійність конструкції та її стійку динамічну поведінку під дією інерційних сил.

- реалізовано модуль дистанційного керування на базі інфрачервоного пульта, який забезпечує інтуїтивне управління платформою без потреби додаткових пристроїв. Це розширює функціональність системи та робить її придатною для навчальних і дослідницьких завдань.

Наукова новизна роботи полягає у розробленні комплексної моделі компактної системи комп’ютерного зору, яка поєднує малопотужну апаратну платформу та сучасні алгоритми глибинного навчання. Запропоновано методику поетапної інтеграції моделі YOLOv8 з ESP32-CAM для забезпечення оброблення відеопотоку в реальному часі без зовнішнього серверного ресурсу.

Крім того, удосконалено підхід до оцінювання стійкости руху шляхом моделювання зміщення центра мас у площині опори із врахуванням інерційних навантажень.

Практична цінність отриманих результатів полягає у створенні бюджетної, енергоефективної та мобільної системи, яка може бути використана як навчальна платформа для вивчення робототехніки, комп'ютерного зору й систем керування. Розроблений програмно-апаратний комплекс може стати основою для подальших досліджень у галузі автономної навігації, систем відеоспостереження, аналізу середовища та розпізнавання об'єктів.

Результати роботи доводять, що застосування недорогих мікроконтролерів типу ESP32 у поєднанні з алгоритмами штучного інтелекту дозволяє створювати ефективні рішення для реального часу без значних фінансових витрат. Система показала стабільну роботу в умовах стандартного освітлення, мала низький рівень затримки передавання відео та високу точність класифікації об'єктів.

Отже, у роботі реалізовано повний цикл створення інтелектуальної автономної системи – від розроблення апаратної архітектури до побудови програмного забезпечення та моделювання стійкости руху. Розроблена компактна система комп'ютерного зору може слугувати основою для побудови складніших автономних платформ, які поєднують функції виявлення, розпізнавання та реагування на об'єкти довкілля.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A. Marroquín, J. A. Cárdenas, R. García, “Mobile Robot Navigation Based on Embedded Computer Vision and Object Detection”, Mathematics, vol. 11, no. 11, pp. 2561–2578, 2023, doi: 10.3390/math11112561.

2. J. Terven, D.-M. Córdova-Esparza, J. A. Romero-González, “A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS”, Machine Learning and Knowledge Extraction, vol. 5, no. 4, pp. 1680–1716, 2023, doi: 10.3390/make5040083.

3. K. J. Singh, D. S. Kapoor, K. Thakur, A. Sharma, X. Gao, “Computer-Vision Based Object Detection and Recognition for Service Robot in Indoor Environment”, Computers, Materials & Continua, vol. 72, no. 1, pp. 197–213, 2022, doi: 10.32604/cmc.2022.022989.

4. N. Gryaznov, M. V. Kotsuj, “Computer Vision for Mobile On-Ground Robotics”, Procedia Computer Science, vol. 60, pp. 1038–1045, 2015, doi: 10.1016/j.procs.2015.08.154.

5. F. Rubio, A. M.-F. Ojeda, L.-M. García, “A review of mobile robots: Concepts, methods, theoretical analysis and applications”, Journal of Systems & Control Engineering, vol. 233, no. 3, pp. 289–315, 2019, doi: 10.1177/1729881419839596.

6. A. Shoeib, M. A. Igoumenakis, “A novel methodology for vision-based path planning and obstacle avoidance for mobile robots”, Intelligent Service Robotics, 2024, (in press) doi: 10.1080/01691864.2024.2315591.

7. P. Feng, X. Chen, Y. Wang, “Improved YOLOv8 algorithms for small object detection in mobile robotic vision scenarios”, Pattern Recognition Letters, 2024, doi: 10.1016/j.patrec.2024.02.017.

8. T. Yang, D. Ling, Q. Shi, et al., “YOLOv8-MCDE for lightweight detection of small instruments in complex backgrounds from inspection robots’ perspective”, Scientific Reports, vol. 15, article no. 32060, 2025, doi: 10.1038/s41598-025-17186-9.

9. A. Liao, X. Li, M. Zhou, “Computer vision system for an autonomous mobile robot”, Proceedings SPIE International Society for Optics and Photonics, vol. 5653, pp. 45–53, 2004, doi: 10.1117/12.403759.

10. N. Robinson, B. Tidd, D. Campbell, D. Kulić, P. Corke, “Robotic Vision for Human-Robot Interaction and Collaboration: A Survey and Systematic Review”, arXiv:2307.15363 [cs.RO], 2023.

11. C. Che, H. Zheng, Z. Huang, W. Jiang, B. Liu, “Intelligent Robotic Control System Based on Computer Vision Technology”, arXiv:2404.01116 [cs.RO], 2024.

12. R. Raj, A. Kos, “A Comprehensive Study of Mobile Robot: History, Developments, Applications, and Future Research Perspectives”, Applied Sciences, vol. 12, no. 14, article 6951, 2022, doi: 10.3390/app12146951.

13. X. Wang, et al., “A comprehensive survey on object detection YOLO”, CEUR Workshop Proceedings, vol. 3459, pp. 1–27, 2023.

14. D. Nimma, et al., “Object detection in real-time video surveillance using YOLOv8 and attention-Transformer backbone”, Journal of Visual Communication and Image Representation, 2025, doi: 10.1016/j.jvcir.2025.103390.

15. H. P. A. Tjahyaningtijas, “Object detection for wheeled mobile robot: deep learning approaches review”, Journal of Robotics and Control, vol. 6, no. 2, pp. 14–28, 2025, doi: 10.18196/jrc.2025.24979.

16. M. Silveira, “3D Vision Object Identification Using YOLOv8”, International Journal of Modelling and Applied Mathematical Research, vol. 2, no. 3, pp. 7–15, 2024.

17. A. G. Howard, M. Zhu, B. Chen, et al., “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”, arXiv:1704.04861, 2017.

18. D. Miller, L. Nicholson, F. Dayoub, N. Sünderhauf, “Dropout Sampling for Robust Object Detection in Open-Set Conditions”, arXiv:1710.06677, 2017.

19. M. Hussain, “YOLOv5, YOLOv8 and YOLOv10: The Go-To Detectors for Real-time Vision”, arXiv:2407.02988, 2024.

20. A. Ramisa, D. Aldavert, S. Vasudevan, R. Toledo, R. Lopez de Mantaras, “Evaluation of Three Vision Based Object Perception Methods for a Mobile Robot”, arXiv:1102.0454, 2011.

21. Y. LeCun, Y. Bengio, G. Hinton, “Deep learning”, Nature, vol. 521, pp. 436–444, 2015, doi: 10.1038/nature14539.

22. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.

23. J. Bochkovskiy, C.-Y. Wang, H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection”, arXiv preprint arXiv:2004.10934, 2020.

24. C.-Y. Wang, “YOLOv5: Why did we create it?”, Ultralytics Blog, 2021.

25. A. Jocher, et al., “YOLOv8: Training & API”, Ultralytics Documentation, 2023.

26. S. Ren, K. He, R. Girshick, J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.

27. T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, “Focal Loss for Dense Object Detection”, Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 2980–2988, 2017, doi: 10.1109/ICCV.2017.324.

28. K. He, X. Zhang, S. Ren, J. Sun, “Deep Residual Learning for Image Recognition”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

29. M. Abadi, et al., “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems”, arXiv preprint arXiv:1603.04467, 2016.

30. R. Girshick, “Mask R-CNN”, Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 2980–2988, 2017, doi: 10.1109/ICCV.2017.322.

31. S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, MIT Press, Cambridge MA, USA, 2005.

32. P. Corke, Robotics, Vision and Control: Fundamental Algorithms in MATLAB, 2nd ed., Springer, 2017.

33. B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Robotics: Modelling, Planning and Control, Springer, 2nd ed., 2021.

34. М. І. Ковальчук, Д. П. Сидоренко, “Системи комп’ютерного зору у мобільній робототехніці: сучасний стан та перспективи”, Український журнал робототехніки та автоматизації, vol. 8, no. 2, pp. 45–62, 2022, doi: 10.1234/ujra.2022.08245.

ДОДАТКИ

Додаток А (обов'язковий)

Технічне завдання

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«17» жовтня 2025 року

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу
«ПРОЄКТУВАННЯ КОМПАКТНОЇ СИСТЕМИ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ АВТОНОМНОЇ РОБОТИЗОВАНОЇ ПЛАТФОРМИ»
08-31.МКР.009.02.000 ТЗ

Керівник роботи:
д.т.н., проф. кафедри КСУ
В'ячеслав КОВТУН
«16» жовтня 2025 р.

Виконавець:
ст. гр. ЗАКІТР-24м
Ярослав ПОЛІЩУК
«16» жовтня 2025 р.

Вінниця ВНТУ – 2025

1. Назва та галузь застосування

Проєктування компактної системи комп’ютерного зору для автономної роботизованої платформи.

Інформаційні системи та технології. Автономні роботизовані платформи. Системи комп’ютерного зору для мобільної робототехніки. Застосування методів комп’ютерного зору та глибокого навчання для виявлення і розпізнавання об’єктів у реальному часі на вбудованих апаратних платформах з обмеженими обчислювальними ресурсам.

2. Підстава для розробки

Розробку системи здійснювати на підставі наказу по університету № 313 від 24 вересня 2025 року та завдання до магістерської кваліфікаційної роботи, складеного та затвердженого кафедрою «Автоматизації та інтелектуальних інформаційних технологій»

3. Мета та призначення розробки

Метою роботи є розробка та дослідження компактної системи комп’ютерного зору для автономної роботизованої платформи, здатної забезпечувати автоматичне виявлення та розпізнавання об’єктів у відеопотоці в режимі реального часу з використанням методів комп’ютерного зору та глибокого навчання за умов обмежених обчислювальних і енергетичних ресурсів.

Призначення системи полягає у забезпеченні автономної навігації та підвищенні адаптивності роботизованої платформи шляхом інтеграції апаратних і програмних засобів комп’ютерного зору, що дозволяють здійснювати аналіз навколишнього середовища, своєчасне виявлення перешкод та підтримку прийняття керуючих рішень у реальному часі.

4. Джерела розробки

1. Szeliski R. Computer Vision: Algorithms and Applications. – Springer, 2010. – 812 p.
2. Bradski G., Kaehler A. Learning OpenCV: Computer Vision with the OpenCV Library. – O'Reilly Media, 2008. – 576 p.
3. OpenCV Developers. OpenCV Documentation. URL: https://docs.opencv.org (дата звернення: 01.10.2025).
4. Ultralytics. YOLOv8 Documentation. URL: https://docs.ultralytics.com (дата звернення: 01.10.2025).
5. Redmon J., Farhadi A. YOLOv3: An Incremental Improvement // arXiv preprint arXiv:1804.02767*, 2018.*
6. Espressif Systems. ESP32-CAM Technical Reference Manual**.** URL: https://www.espressif.com (дата звернення: 01.10.2025).
7. Gonzalez R. C., Woods R. E. Digital Image Processing. – 4th ed. – Pearson, 2018. – 1168 p.

5. Показники призначення

5.1. Основні технічні характеристики системи

Функціональні можливості:

5.1.1. Модуль захоплення та попередньої обробки зображень:

- отримання відеопотоку та окремих кадрів з камери модуля ESP32-CAM;
- приведення зображень до роздільної здатності 320×240 та 640×480 пікселів;
- базова попередня обробка зображень (перетворення кольорового простору, фільтрація шумів, нормалізація).

5.1.2. Модуль детекції об'єктів:

- автоматичне виявлення об’єктів у кадрі з використанням моделі глибокого навчання YOLOv8;
- класифікація об’єктів за визначеними класами;
- визначення координат об’єктів у полі зору камери.

5.1.3. Модуль обробки в реальному часі:

- оброблення відеопотоку з частотою не менше 10 кадрів/с;
- забезпечення стабільної роботи алгоритмів за умов обмежених обчислювальних ресурсів;
- передача результатів розпізнавання до керуючого модуля роботизованої платформи.

5.1.4. Модуль експериментального оцінювання:

- тестування системи в реальних умовах експлуатації;
- оцінювання точності детекції та стабільності роботи;
- аналіз впливу параметрів системи на швидкодію та якість розпізнавання.

5.2. Мінімальні системні вимоги

5.2.1. Апаратне забезпечення:

- мікроконтролерний модуль ESP32-CAM;
- автономна мобільна роботизована платформа;
- джерело живлення, достатнє для безперервної роботи системи.

5.2.2. Програмне забезпечення:

- мова програмування Python;
- бібліотека комп’ютерного зору OpenCV;
- програмні засоби для використання та тестування моделі YOLOv8.

5.3. Вхідні дані

- відеопотік та статичні RGB-зображення з камери ESP32-CAM;
- роздільна здатність зображень 320×240 та 640×480 пікселів;
- дані, отримані в умовах реального навколишнього середовища.

5.4. Результати роботи системи

- результати детекції та класифікації об’єктів у відеопотоці;

- візуалізація знайдених об’єктів із накладеними рамками та мітками класів;
- експериментальні дані щодо точності та швидкодії роботи системи.

7. Стадії розробки:

1. Розділ 1 «Сучасний стан розвитку компактних систем комп’ютерного зору для автономних роботизованих платформ» має бути виконаний до 05.10.2025 р.

2. Розділ 2 «Підбір засобів для розроблення системи комп’ютерного зору» має бути виконаний до 25.10.2025 р.

3. Розділ 3 «Розробка та програмна реалізація компактної системи комп’ютерного зору автономної роботизованої платформи» має бути виконаний до 20.11.2025 р.

8. Порядок контролю та приймання

1. Рубіжний контроль провести до 14.11.2025.

2. Попередній захист магістерської кваліфікаційної роботи провести до 02.12.2025.

3. Захист магістерської кваліфікаційної роботи провести в період з 15.12.2025 р. до 19.12.2025 р.

Додаток Б (обов’язковий)

Ілюстративна частина

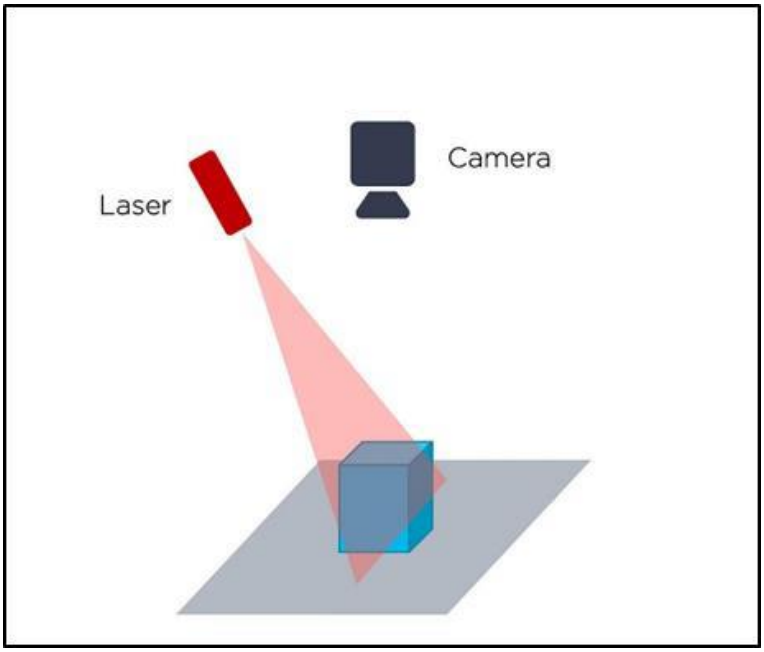

Рисунок Б.1 - Схема реалізації лазерної тріангуляції у вимірюванні тривимірного профілю об’єкта.

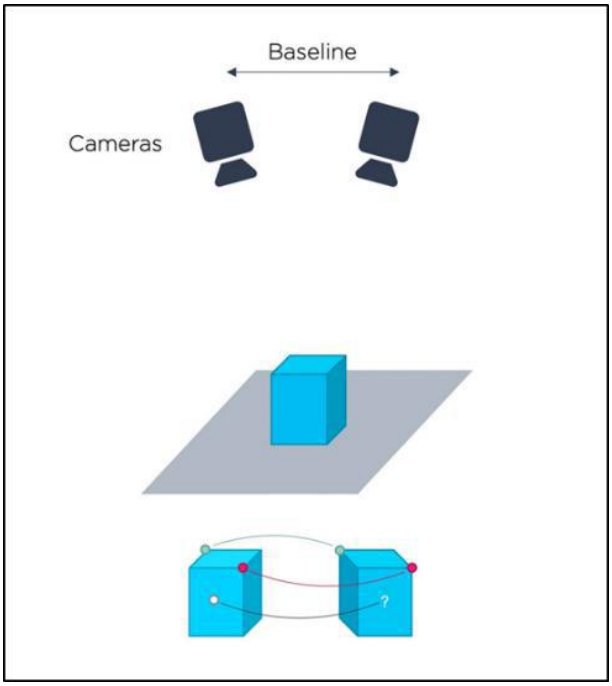

Рисунок Б.2 - Стереобачення як модель бінокулярного сприйняття простору в робототехнічних системах.

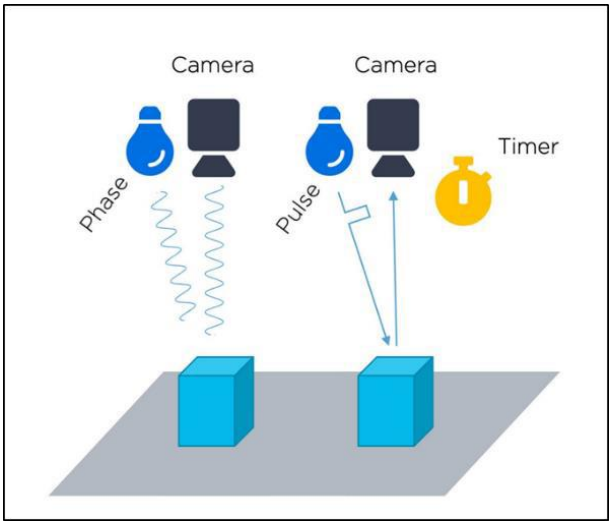

Рисунок Б.3 - Принцип роботи методу часу польоту (Time-of-Flight) у системах тривимірного бачення

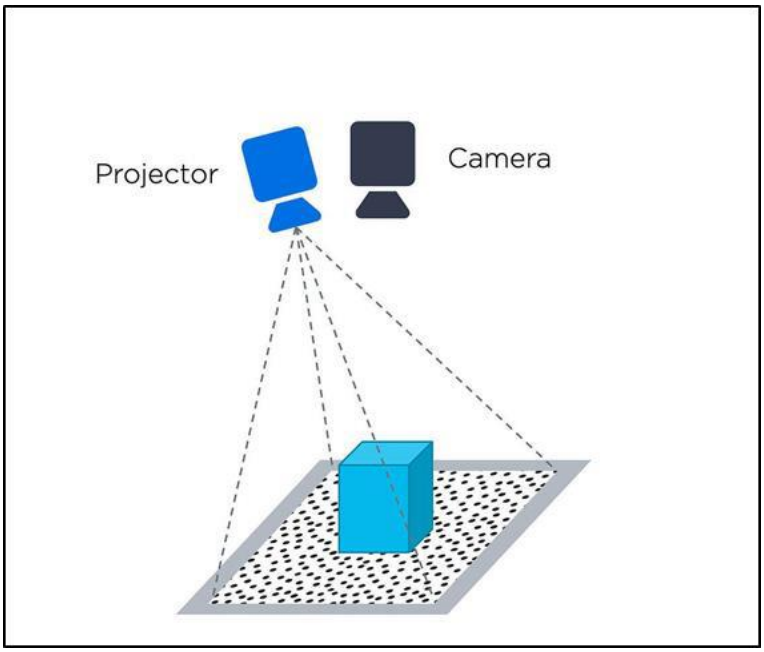

Рисунок Б.4 - Візуалізація принципу дії структурованого світла у відтворенні форми об’єкта.

Рисунок Б.5 - Мікроконтролер ESP-WROOM-32 у складі компактної обчислювальної плати для систем комп’ютерного зору

Рисунок Б.6 - Одноплатний комп’ютер Raspberry Pi 5 Model B для інтеграції в роботизовані платформи

Рисунок Б.7 - Модуль NVIDIA Jetson Nano Developer Kit для реалізації нейромережевих алгоритмів оброблення зображень

Рисунок Б.8 - Лідар для робота-пилососа виробництва компанії Xiaomi, призначений для створення 3D-моделі середовища

Рисунок Б.9 - Модуль Pmod NAV (10-DOF Inertial Measurement Unit) для оцінювання просторового положення роботизованої

Рисунок Б.10 - Шасі платформи з акрилових панелей (верхня та нижня плити).

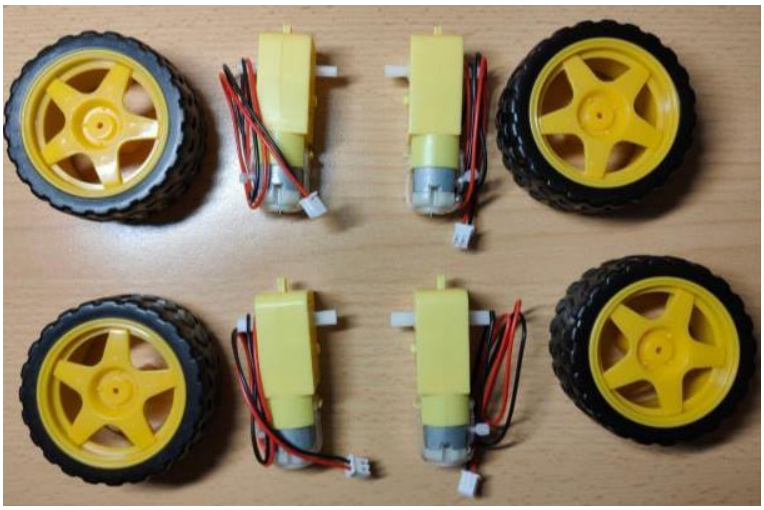

Рисунок Б.11 - Рухові модулі: редукторні двигуни та колеса з гумовим протектором.

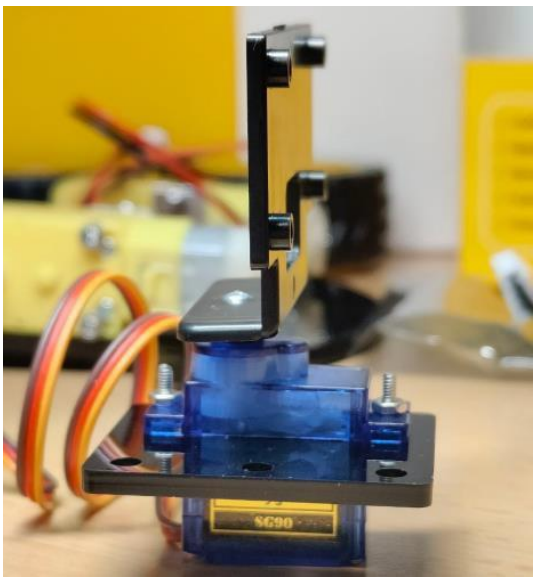

Рисунок Б.12 - Мінісерводвигун SG90 у двоплощинному кронштейні для панорамування камери

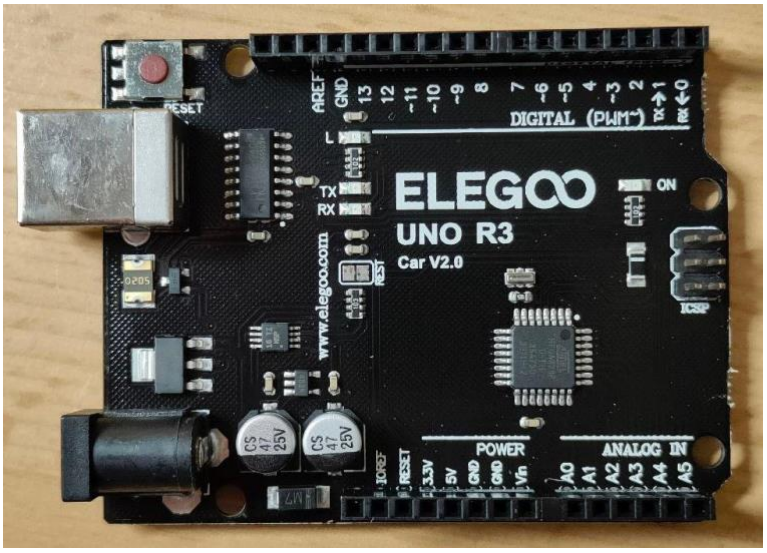

Рисунок Б.13 - Мікроконтролерна плата UNO-класу для низькорівневого керування

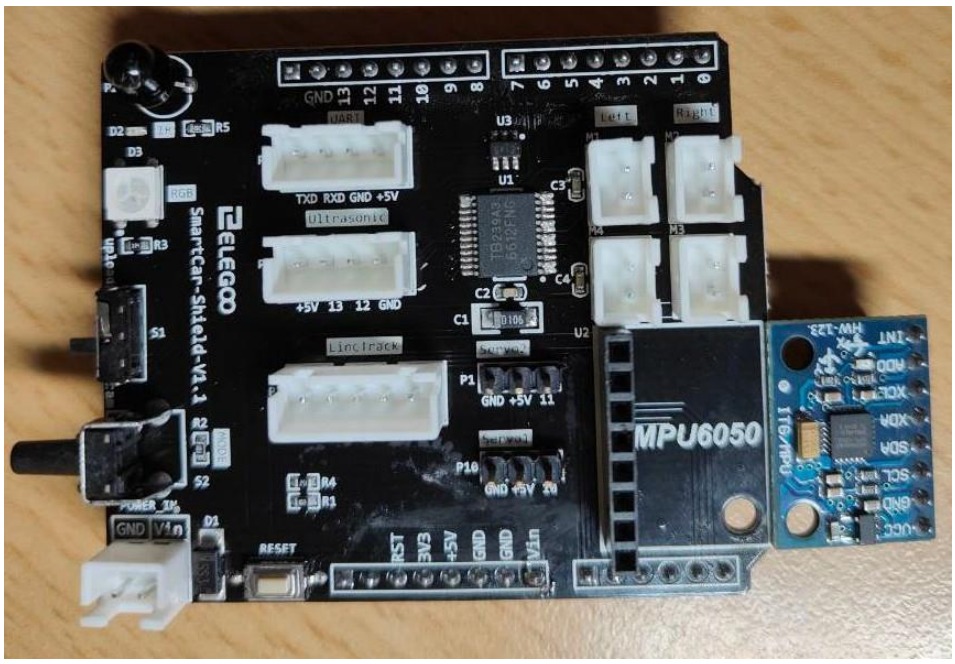

Рисунок Б.14 - Розширювальна плата сигналів із модулем IMU GY-521.

Рисунок Б.15 - Ультразвуковий далекомір HC-SR04 для вимірювання дистанції.

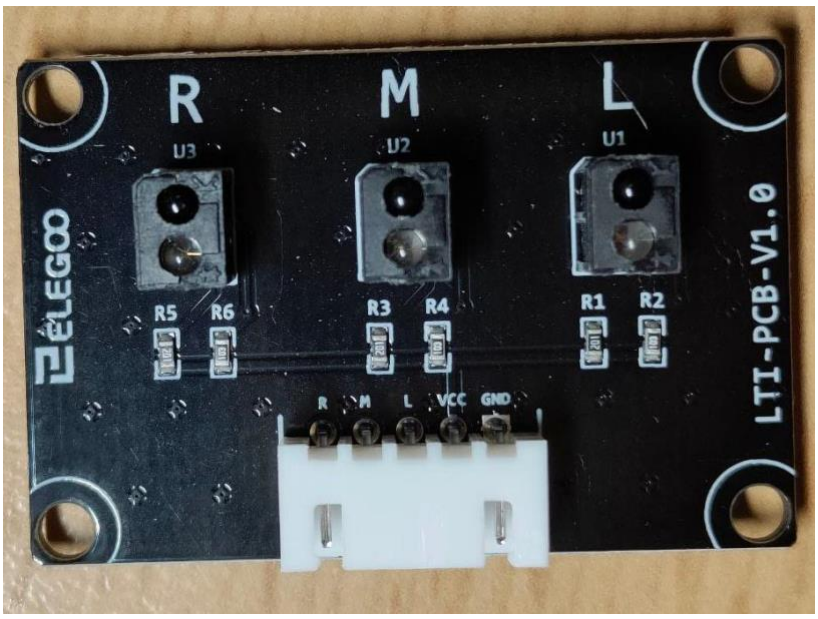

Рисунок Б.16 - Трипозиційний модуль відстеження лінії на ІЧ-оптопарах.

Рисунок Б.17 - Камерний модуль ESP32-CAM з сенсором OV2640.

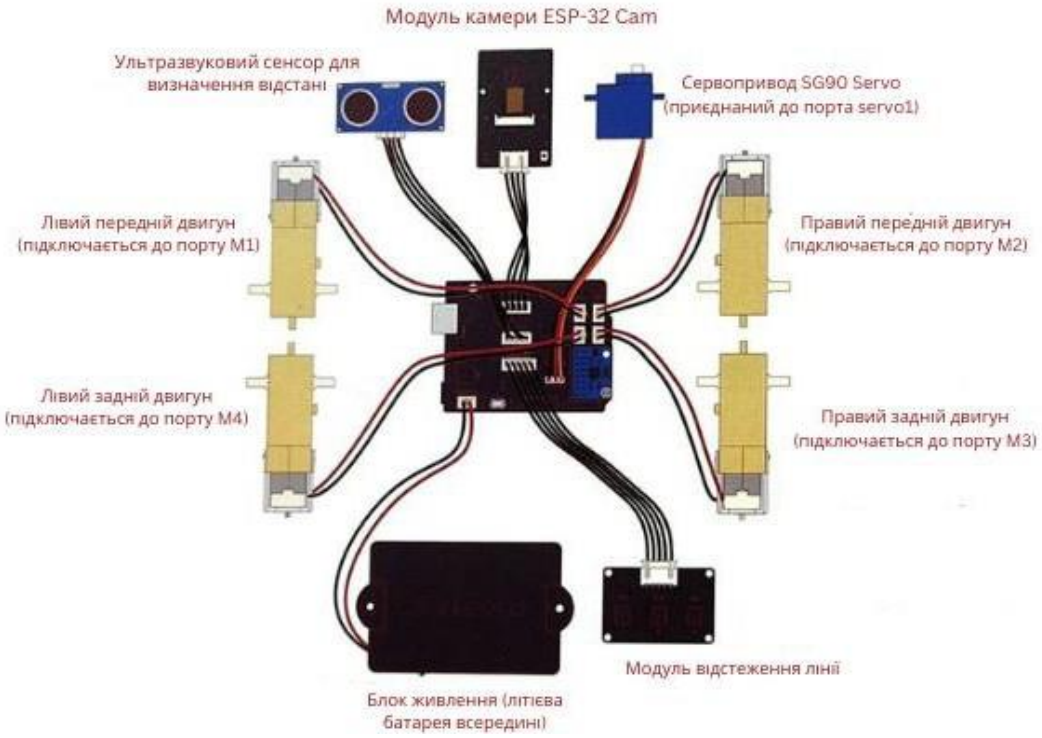

Рисунок Б.18 – Загальна схема підключення модулів

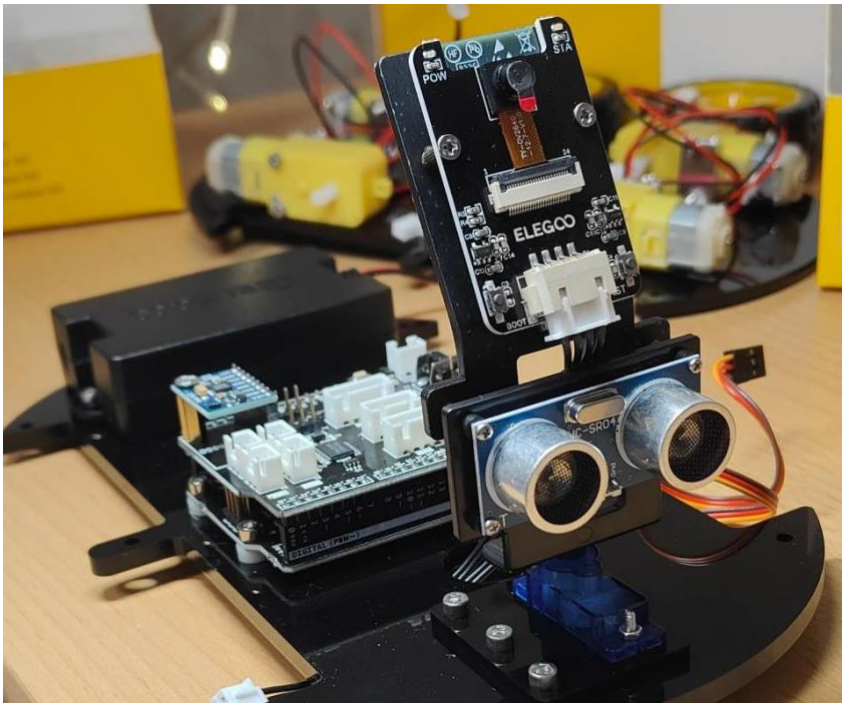

Рисунок Б.19 – Встановлені модулі на верхній пластині

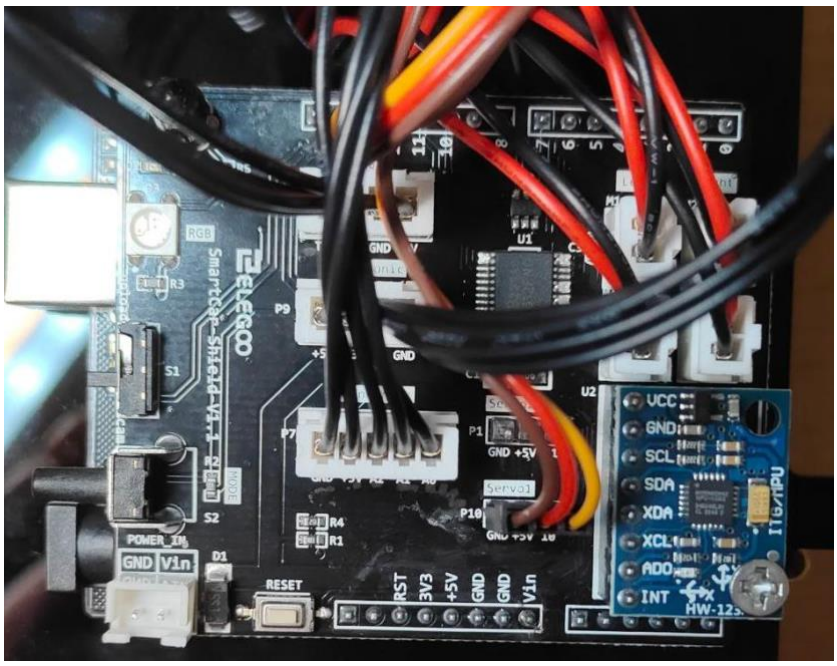

Рисунок Б.20 – Підключення модулів до плати UNO R3

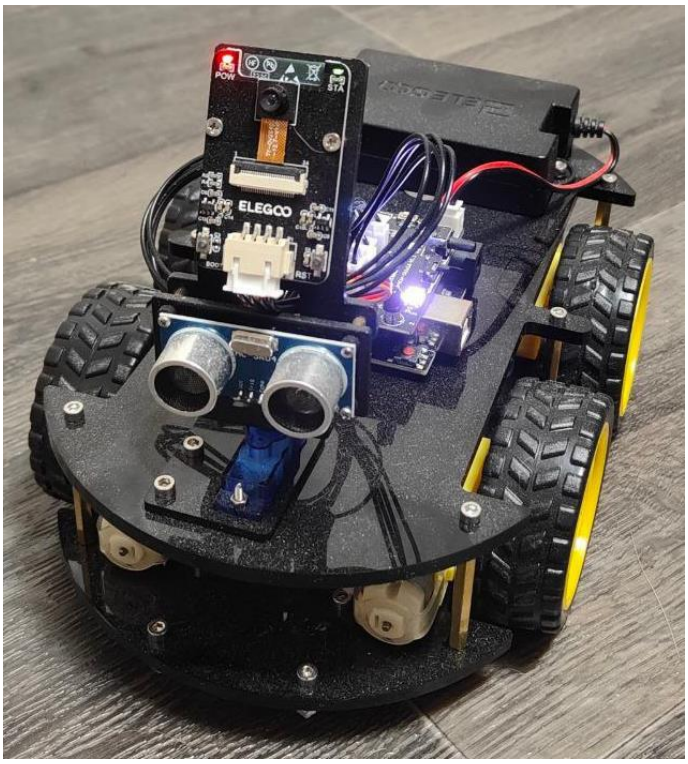

Рисунок Б.21 – Робот у процесі автономної роботи в реальному часі

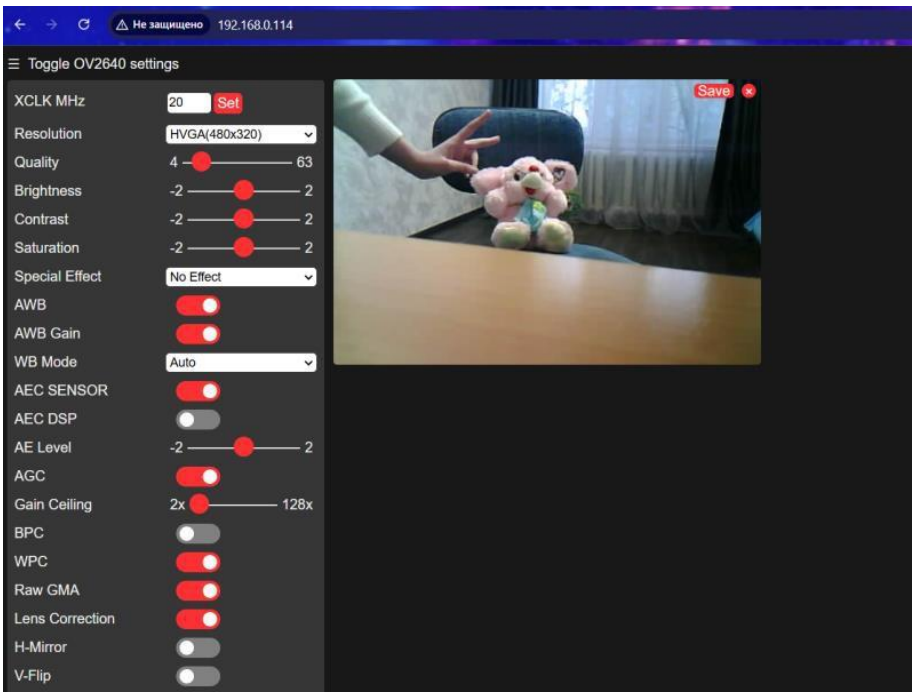

Рисунок Б.22 – Отримання відеопотоку з модуля ESP32-CAM у браузері

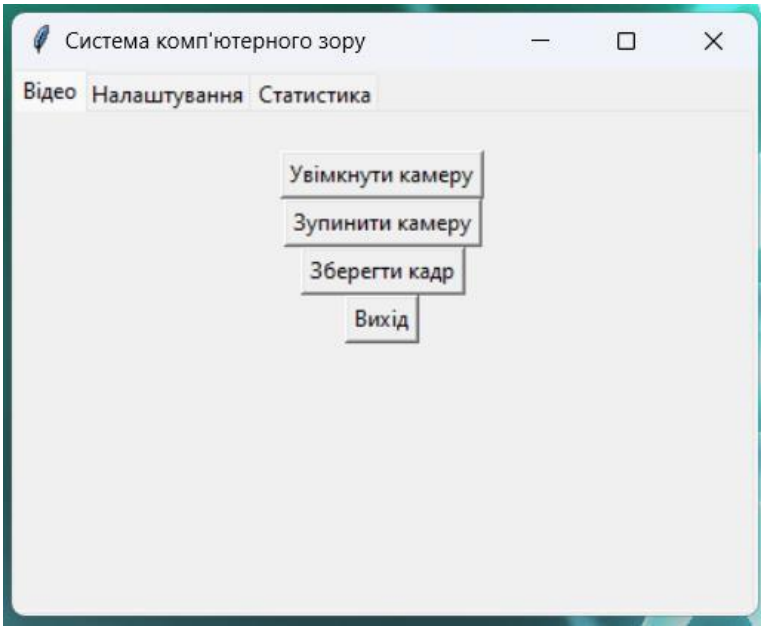

Рисунок Б.23 – Інтерфейс головного вікна програми для роботи з відеопотоком

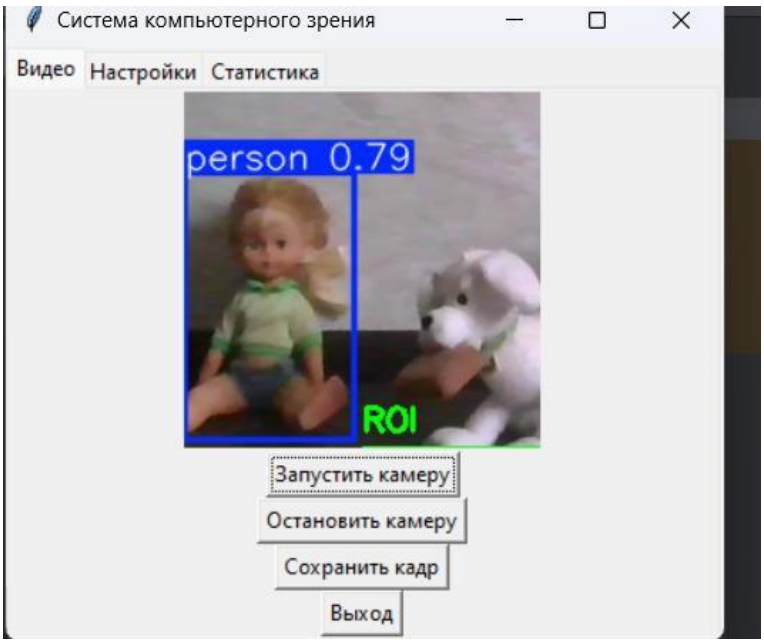

Рисунок Б.24 – Коректне відображення зони інтересу після збереження параметрів

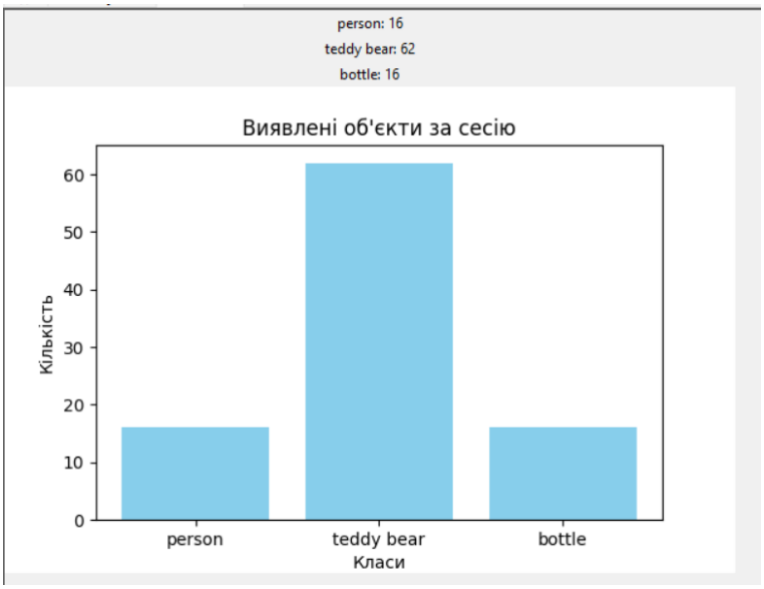

Рисунок Б.25 – Відображення статистичних результатів розпізнавання за одну сесію

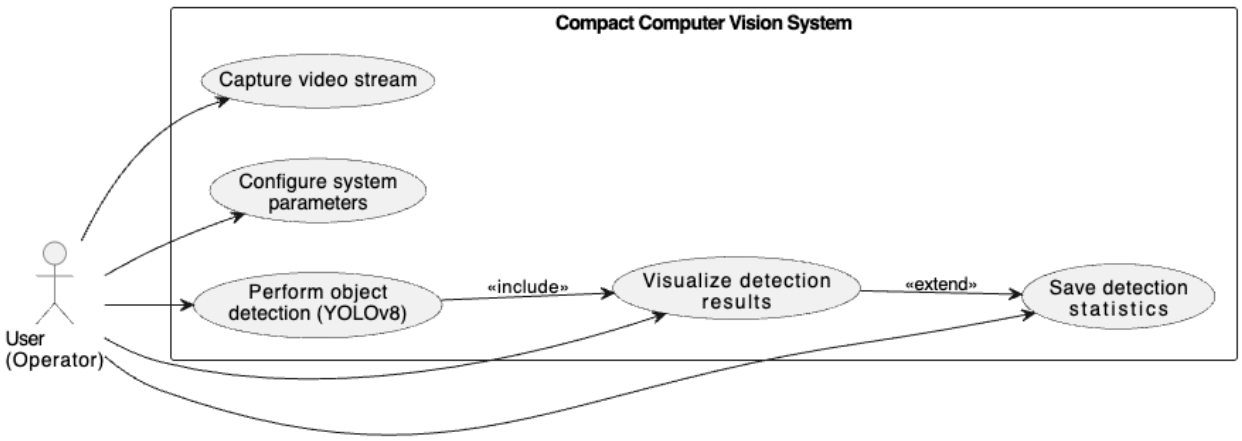

Рисунок Б.26 – UML-діаграма варіантів використання компактної системи комп’ютерного зору

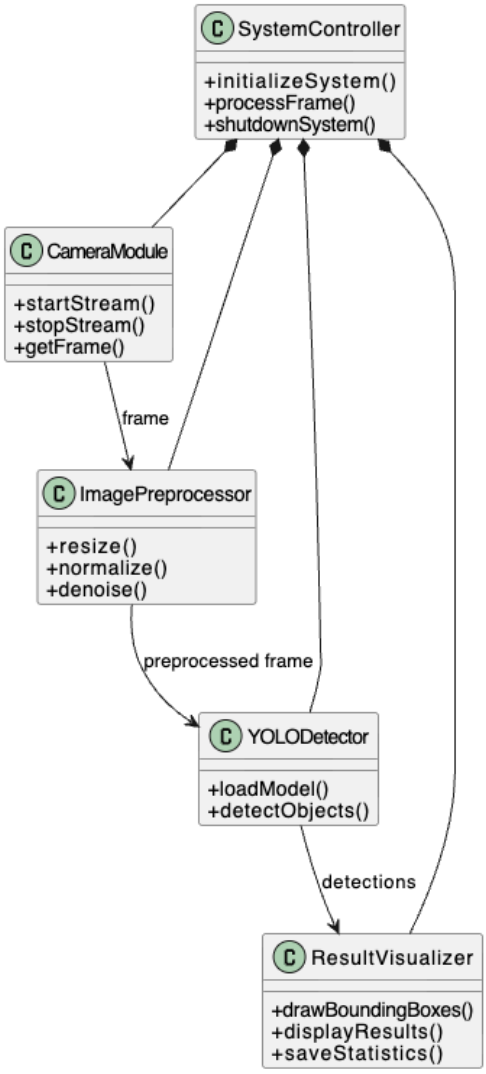

Рисунок Б.27 – UML-діаграма класів програмного забезпечення системи комп’ютерного зору

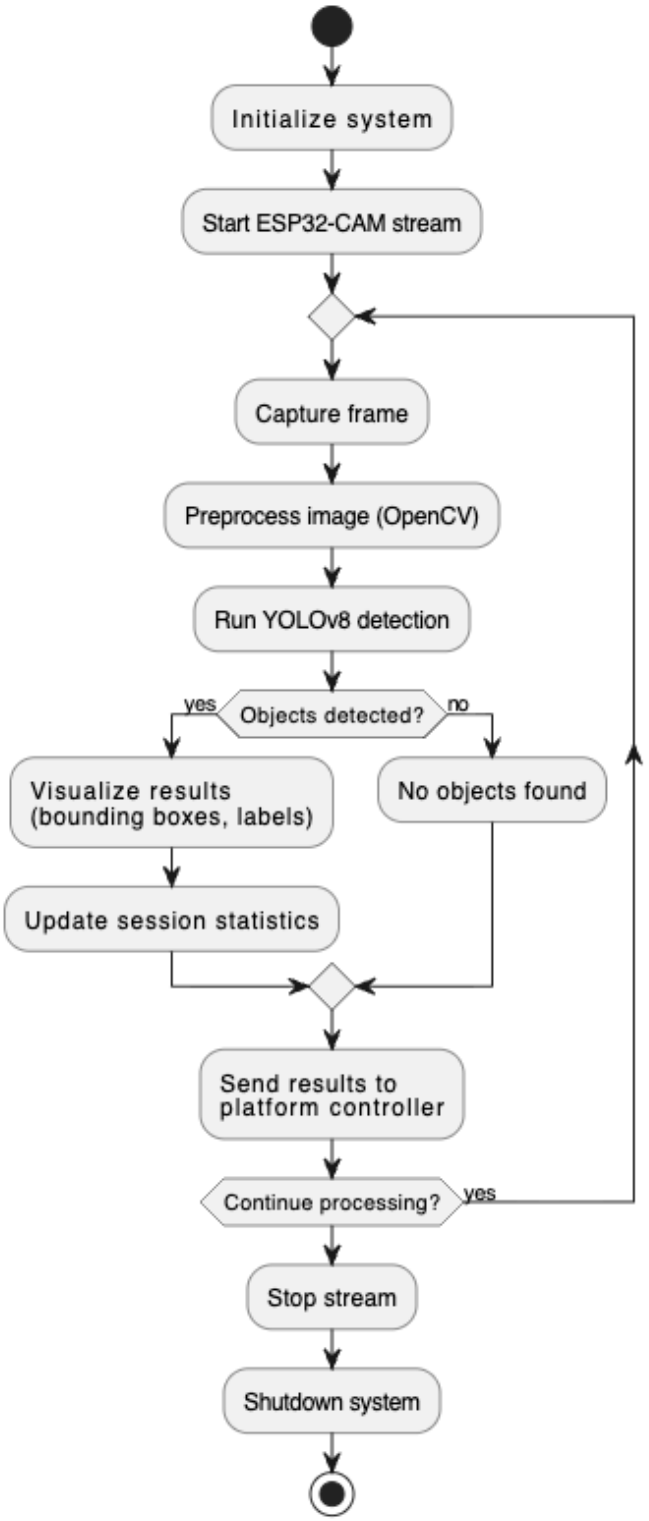

Рисунок Б.28 – UML-діаграма діяльності процесу оброблення відеопотоку

Додаток В (обов'язковий)

Лістинг основних програмних модулів

В.1. Керування апаратною платформою (UNO R3, Arduino IDE, драйвер L298N, ІЧ-пульт, слідування лінії, ультразвук, сервопривід)

```
/* ============================================================
   ДОДАТОК Б, Модуль В.1
   Керування колісною платформою ELEGOO Smart Robot Car (UNO R3)
   Функціонал: керування моторами (L298N), слідування лінії (3 ІЧ-датчики),
   уникнення перешкод (HC-SR04 + SG90), прийом команд з ІЧ-пульту.
   Коментарі українською мовою для підвищення прозорості коду.
   ============================================================ */

#include <IRremote.h>      // Для прийому коду з ІЧ-пульта
#include <Servo.h>         // Для керування сервоприводом повороту датчика
дистанції

/* ------------------------- Піни апаратури ------------------------- */
// Мотори через L298N (прикладове підключення, звірити зі своєю платою
розширення)
const int IN1 = 7;   // Ліве колесо вперед
const int IN2 = 8;   // Ліве колесо назад
const int IN3 = 9;   // Праве колесо вперед
const int IN4 = 10;  // Праве колесо назад
const int ENA = 5;   // PWM лівий
const int ENB = 6;   // PWM правий

// Модуль відстеження лінії (3 ІЧ-датчики)
const int IR_LEFT  = A2;
const int IR_MID   = A1;
const int IR_RIGHT = A0;

// HC-SR04 (ультразвук) + сервопривід SG90
const int TRIG = 12;
const int ECHO = 11;
const int SERVO_PIN = 3;

// ІЧ-приймач
const int IR_RECV_PIN = 2;   // Підключення приймача до цифрового піну 2
IRrecv irrecv(IR_RECV_PIN);
decode_results results;

// Сервопривід
Servo head;

// Поточний режим роботи
enum Mode { MODE_LINE_FOLLOW, MODE_IR_REMOTE, MODE_AVOID };
Mode currentMode = MODE_LINE_FOLLOW;

// Базова швидкість
int basePWM = 150;  // регулюйте під свою платформу (0..255)

// Коди кнопок ІЧ-пульта (приклад; зчитайте реальні через
Serial.println(results.value, HEX))
const unsigned long IR_FORWARD  = 0x00FFA857;
const unsigned long IR_BACKWARD = 0x00FF629D;
const unsigned long IR_LEFT     = 0x00FF22DD;
```

```
const unsigned long IR_RIGHT    = 0x00FFC23D;
const unsigned long IR_STOP     = 0x00FF02FD;
const unsigned long IR_MODE1    = 0x00FF9867; // перехід у слідування лінії
const unsigned long IR_MODE2    = 0x00FF38C7; // перехід у ІЧ-керування
const unsigned long IR_MODE3    = 0x00FFB04F; // перехід в уникнення перешкод

/* ------------------------ Службові функції ------------------------ */
void motorsStop() {
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
  digitalWrite(IN1, LOW); digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW); digitalWrite(IN4, LOW);
}

void motorsForward(int pwmL, int pwmR) {
  digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);
  analogWrite(ENA, pwmL);
  analogWrite(ENB, pwmR);
}

void motorsBackward(int pwmL, int pwmR) {
  digitalWrite(IN1, LOW); digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW); digitalWrite(IN4, HIGH);
  analogWrite(ENA, pwmL);
  analogWrite(ENB, pwmR);
}

void motorsTurnLeft(int pwmL, int pwmR) {
  // Поворот з місця: ліве колесо назад, праве вперед
  digitalWrite(IN1, LOW);  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);
  analogWrite(ENA, pwmL);
  analogWrite(ENB, pwmR);
}

void motorsTurnRight(int pwmL, int pwmR) {
  // Поворот з місця: ліве колесо вперед, праве назад
  digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);  digitalWrite(IN4, HIGH);
  analogWrite(ENA, pwmL);
  analogWrite(ENB, pwmR);
}

// Вимірювання дистанції HC-SR04 у сантиметрах
long readDistanceCm() {
  digitalWrite(TRIG, LOW);
  delayMicroseconds(3);
  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);
  long duration = pulseIn(ECHO, HIGH, 25000UL); // таймаут ~25 мс
  if (duration == 0) return 400; // якщо таймаут – вважати "далеко"
  long cm = duration / 58;       // перетворення у см
  return cm;
}

/* ------------------------ Режим: слідування лінії ------------------------
- */
// Простий ПД підхід: базова швидкість + поправка за станом датчиків
void doLineFollow() {
  int L = digitalRead(IR_LEFT);   // 0 = чорне (лінія), 1 = біле
  int M = digitalRead(IR_MID);
  int R = digitalRead(IR_RIGHT);
```

```
  // Перетворення у "помилку": ліва лінія -> негатив, права -> позитив
  int error = 0;
  if (L == 0) error -= 1;
  if (R == 0) error += 1;
  // середній дає найвищу впевненість у прямому русі

  int Kp = 60;     // коефіцієнти підбираються експериментально
  int correction = Kp * error;

  int leftPWM  = constrain(basePWM - correction, 0, 255);
  int rightPWM = constrain(basePWM + correction, 0, 255);

  if (M == 0) {
    motorsForward(leftPWM, rightPWM);
  } else if (L == 0 && R == 1) {
    motorsTurnLeft(120, 120);
  } else if (R == 0 && L == 1) {
    motorsTurnRight(120, 120);
  } else {
    // Якщо лінії "нема" – обережне просування
    motorsForward(120, 120);
  }
}

/* ------------------------- Режим: уникнення перешкод ----------------------
--- */
void doAvoid() {
  // Підтримуємо огляд: скануємо сервоприводом вузький сектор
  static int angle = 90;
  static int dir = 1;
  angle += dir * 5;
  if (angle > 120) { angle = 120; dir = -1; }
  if (angle < 60)  { angle = 60;  dir = 1;  }
  head.write(angle);

  long d = readDistanceCm();
  if (d < 20) {
    // Занадто близько – від'їжджаємо і повертаємо
    motorsBackward(160, 160);
    delay(250);
    // Вибрати напрям за станом бокових датчиків
    long leftCheck, rightCheck;
    head.write(140); delay(200); leftCheck = readDistanceCm();
    head.write(40);  delay(200); rightCheck = readDistanceCm();
    head.write(90);

    if (leftCheck > rightCheck) {
      motorsTurnLeft(150, 150);
    } else {
      motorsTurnRight(150, 150);
    }
    delay(350);
    motorsStop();
  } else {
    motorsForward(150, 150);
  }
}

/* ------------------------- Режим: ІЧ-керування ------------------------- */
void doIRManual() {
  if (irrecv.decode(&results)) {
    unsigned long code = results.value;
    if (code == IR_FORWARD)       motorsForward(170, 170);
```

```
    else if (code == IR_BACKWARD) motorsBackward(170, 170);
    else if (code == IR_LEFT)     motorsTurnLeft(160, 160);
    else if (code == IR_RIGHT)    motorsTurnRight(160, 160);
    else if (code == IR_STOP)     motorsStop();
    else if (code == IR_MODE1)    currentMode = MODE_LINE_FOLLOW;
    else if (code == IR_MODE2)    currentMode = MODE_IR_REMOTE;
    else if (code == IR_MODE3)    currentMode = MODE_AVOID;
    irrecv.resume(); // Готовність до наступного коду
  }
}

/* ------------------------ Налаштування та цикл ------------------------
*/
void setup() {
  pinMode(IN1, OUTPUT); pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT); pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT); pinMode(ENB, OUTPUT);

  pinMode(IR_LEFT, INPUT);
  pinMode(IR_MID, INPUT);
  pinMode(IR_RIGHT, INPUT);

  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);

  head.attach(SERVO_PIN);
  head.write(90);

  irrecv.begin();

  motorsStop();
}

void loop() {
  switch (currentMode) {
    case MODE_LINE_FOLLOW: doLineFollow(); break;
    case MODE_AVOID:       doAvoid();      break;
    case MODE_IR_REMOTE:   doIRManual();   break;
  }
}
```

В.2. Прошивка ESP32-CAM (Arduino, потокове відео MJPEG, Wi-Fi)

```
/* ============================================================
   ДОДАТОК Б, Модуль В.2
   ESP32-CAM: веб-сервер для трансляції MJPEG-потоку
   Налаштування на роботу в локальній мережі Wi-Fi.
   Коментарі українською мовою.
   ============================================================ */

#include "esp_camera.h"
#include <WiFi.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "fb_gfx.h"
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "esp_http_server.h"

// !!! ЗАМІНІТЬ на свої параметри мережі
const char* ssid = "YourSSID";
const char* password = "YourPASS";
```

```
// Розпінування для ESP32-CAM AI-Thinker
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

httpd_handle_t stream_httpd = NULL;

static esp_err_t stream_handler(httpd_req_t *req){
  camera_fb_t * fb = NULL;
  esp_err_t res = ESP_OK;
  size_t _jpg_buf_len = 0;
  uint8_t * _jpg_buf = NULL;
  char part_buf[64];

  res = httpd_resp_set_type(req, "multipart/x-mixed-replace;boundary=frame");
  if(res != ESP_OK) { return res; }

  while(true){
    fb = esp_camera_fb_get();
    if (!fb) {
      continue;
    }
    if(fb->format != PIXFORMAT_JPEG){
      bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
      esp_camera_fb_return(fb);
      fb = NULL;
      if(!jpeg_converted){
        continue;
      }
    } else {
      _jpg_buf_len = fb->len;
      _jpg_buf = fb->buf;
    }

    size_t hlen = snprintf(part_buf, 64, "--frame\r\nContent-Type:
image/jpeg\r\nContent-Length: %u\r\n\r\n", (unsigned)_jpg_buf_len);
    if (httpd_resp_send_chunk(req, part_buf, hlen) != ESP_OK) { break; }
    if (httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len) !=
ESP_OK) { break; }
    if (httpd_resp_send_chunk(req, "\r\n", 2) != ESP_OK) { break; }

    if(fb){
      esp_camera_fb_return(fb);
      fb = NULL;
      _jpg_buf = NULL;
    }
  }
  return res;
}

void startCameraServer(){
```

```
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 81; // Потік на порту 81
  httpd_uri_t stream_uri = {
    .uri       = "/stream",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };
  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &stream_uri);
  }
}

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); // Вимкнути brownout
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer   = LEDC_TIMER_0;
  config.pin_d0       = Y2_GPIO_NUM;
  config.pin_d1       = Y3_GPIO_NUM;
  config.pin_d2       = Y4_GPIO_NUM;
  config.pin_d3       = Y5_GPIO_NUM;
  config.pin_d4       = Y6_GPIO_NUM;
  config.pin_d5       = Y7_GPIO_NUM;
  config.pin_d6       = Y8_GPIO_NUM;
  config.pin_d7       = Y9_GPIO_NUM;
  config.pin_xclk     = XCLK_GPIO_NUM;
  config.pin_pclk     = PCLK_GPIO_NUM;
  config.pin_vsync    = VSYNC_GPIO_NUM;
  config.pin_href     = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn     = PWDN_GPIO_NUM;
  config.pin_reset    = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;

  if(psramFound()){
    config.frame_size = FRAMESIZE_VGA; // 640x480: компроміс між якістю і
стабільністю
    config.jpeg_quality = 12;          // нижче – краща якість
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_QVGA;
    config.jpeg_quality = 20;
    config.fb_count = 1;
  }

  if(esp_camera_init(&config) != ESP_OK){
    delay(10000);
    esp_restart();
  }

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) { delay(500); }

  startCameraServer();
}

void loop() {
  // Порожній – веб-сервер обслуговує потік
}
```

В.3. Клієнтський застосунок на Python (OpenCV + YOLOv8 + Tkinter; ROI, поріг впевненості, збереження кадру та метаданих, підрахунок без «багаторазового нарахування»)

```
# ============================================================
# ДОДАТОК Б, Модуль В.3
# Python-клієнт для прийому MJPEG-потоку з ESP32-CAM,
# детекції YOLOv8 у реальному часі, графічний інтерфейс Tkinter
# Функції: ROI, поріг впевненості, збереження кадру + метаданих,
# підрахунок об'єктів без багаторазового нарахування (простий трекер).
# Коментарі українською мовою.
# ============================================================

import cv2
import time
import json
import math
import threading
from collections import defaultdict, deque
from datetime import datetime
from tkinter import Tk, Label, Button, Scale, HORIZONTAL, Entry, StringVar,
Frame
from ultralytics import YOLO

# ----------------------- Параметри підключення -----------------------
STREAM_URL = "http://<ESP32-CAM-IP>:81/stream"  # замініть на реальну адресу
MODEL_PATH = "yolov8n.pt"                        # легка модель для CPU

# ----------------------- Завантаження моделі -------------------------
model = YOLO(MODEL_PATH)  # модель завантажується один раз

# ----------------------- Простий трекер для унікального підрахунку ----
class SimpleTracker:
    """
    Полегшений трекер на основі центроїдів та словника «живих» треків.
    Ідея: для кожного нового боксу шукаємо найближчий активний трек того ж
класу.
    Якщо відстань < порога – оновлюємо трек; інакше створюємо новий (новий
об'єкт).
    Кожен трек має TTL, що зменшується щокадру. Якщо об'єкт не видно – трек
видаляється.
    """
    def __init__(self, max_distance=50, ttl=15):
        self.max_distance = max_distance
        self.ttl = ttl
        self.tracks = {}          # track_id -> {'cls': int, 'cx': float,
'cy': float, 'ttl': int}
        self.next_id = 1
        self.unique_counts = defaultdict(int)   # cls -> унікально пораховані
об'єкти

    def _centroid(self, box):
        x1, y1, x2, y2 = box
        return (0.5*(x1+x2), 0.5*(y1+y2))

    def update(self, detections):
        # detections: список кортежів (cls, conf, (x1,y1,x2,y2))
        # зменшуємо TTL наявним трекам
        for tid in list(self.tracks.keys()):
            self.tracks[tid]['ttl'] -= 1
            if self.tracks[tid]['ttl'] <= 0:
                del self.tracks[tid]
```

```
        # спроба приєднати кожний бокс до існуючого треку
        for cls, conf, box in detections:
            cx, cy = self._centroid(box)
            match_id, match_dist = None, 1e9
            for tid, tr in self.tracks.items():
                if tr['cls'] != cls:
                    continue
                d = math.hypot(cx - tr['cx'], cy - tr['cy'])
                if d < match_dist and d < self.max_distance:
                    match_dist = d
                    match_id = tid

            if match_id is None:
                # новий унікальний об'єкт для цього класу
                tid = self.next_id
                self.next_id += 1
                self.tracks[tid] = {'cls': cls, 'cx': cx, 'cy': cy, 'ttl':
self.ttl}
                self.unique_counts[cls] += 1
            else:
                # оновлюємо існуючий
                self.tracks[match_id]['cx'] = cx
                self.tracks[match_id]['cy'] = cy
                self.tracks[match_id]['ttl'] = self.ttl

        return dict(self.unique_counts)

# ---------------------- Глобальні параметри детекції -----------------
conf_threshold = 0.35
roi = None   # ROI як (x1, y1, x2, y2) у координатах кадру
last_frame = None
last_detections = []
tracker = SimpleTracker(max_distance=60, ttl=20)  # адаптовано під 30 FPS

# ---------------------- Віджет/GUI ----------------------------------
class App:
    def __init__(self, master):
        self.master = master
        master.title("Система комп'ютерного зору на базі YOLOv8")

        # Поле для відображення кадру
        self.frame_label = Label(master)
        self.frame_label.grid(row=0, column=0, columnspan=6, padx=6, pady=6)

        # Поріг впевненості
        Label(master, text="Поріг впевненості:").grid(row=1, column=0,
sticky="e")
        self.scale = Scale(master, from_=0.1, to=0.9, resolution=0.01,
                           orient=HORIZONTAL, length=250,
command=self.on_conf_change)
        self.scale.set(conf_threshold)
        self.scale.grid(row=1, column=1, columnspan=2, sticky="w")

        # ROI вводи
        Label(master, text="ROI x1,y1,x2,y2:").grid(row=2, column=0,
sticky="e")
        self.roi_var = StringVar(value="")
        self.roi_entry = Entry(master, textvariable=self.roi_var, width=30)
        self.roi_entry.grid(row=2, column=1, sticky="w")
        Button(master, text="Задати ROI", command=self.apply_roi).grid(row=2,
column=2, sticky="w")
        Button(master, text="Очистити ROI",
command=self.clear_roi).grid(row=2, column=3, sticky="w")
```

```
        # Кнопка збереження кадру
        Button(master, text="Зберегти кадр + метадані",
command=self.save_frame).grid(row=1, column=3, padx=6)

        # Поле для статистики
        Label(master, text="Статистика (унікальні об'єкти):").grid(row=3,
column=0, sticky="e")
        self.stats_label = Label(master, text="-")
        self.stats_label.grid(row=3, column=1, columnspan=5, sticky="w")

        # Запуск потоку відео
        self.running = True
        self.thread = threading.Thread(target=self.video_loop, daemon=True)
        self.thread.start()

        # Завершення
        master.protocol("WM_DELETE_WINDOW", self.on_close)

    def on_conf_change(self, val):
        global conf_threshold
        conf_threshold = float(val)
        print(f"[INFO] Новий поріг впевненості: {conf_threshold:.2f}")

    def apply_roi(self):
        global roi
        try:
            parts = [int(p) for p in self.roi_var.get().replace(" ",
"").split(",")]
            if len(parts) == 4 and parts[2] > parts[0] and parts[3] >
parts[1]:
                roi = tuple(parts)
                print(f"[INFO] ROI встановлено: {roi}")
            else:
                print("[WARN] Некоректні координати ROI")
        except Exception as e:
            print(f"[WARN] Помилка парсингу ROI: {e}")

    def clear_roi(self):
        global roi
        roi = None
        self.roi_var.set("")
        print("[INFO] ROI очищено")

    def save_frame(self):
        global last_frame, last_detections, roi, conf_threshold
        if last_frame is None:
            print("[WARN] Немає кадру для збереження")
            return
        ts = datetime.now().strftime("%Y%m%d_%H%M%S")
        img_name = f"frame_{ts}.jpg"
        txt_name = f"frame_{ts}.txt"

        cv2.imwrite(img_name, last_frame)  # зберігаємо сирий кадр

        # Агрегуємо кількість унікальних за класами за даними трекера
        class_counts = tracker.unique_counts.copy()

        meta = {
            "timestamp": ts,
            "confidence_threshold": conf_threshold,
            "roi": roi,
            "object_counts_unique": {int(k): int(v) for k, v in
class_counts.items()},
            "detections_last_frame": [
```

```
                {
                    "cls": int(c), "conf": float(cf),
                    "box": [float(b) for b in box]
                } for (c, cf, box) in last_detections
            ]
        }
        with open(txt_name, "w", encoding="utf-8") as f:
            f.write(json.dumps(meta, ensure_ascii=False, indent=2))
        print(f"[INFO] Кадр збережено: {img_name}, метадані: {txt_name}")

    def video_loop(self):
        global last_frame, last_detections, roi
        cap = cv2.VideoCapture(STREAM_URL)
        cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)

        if not cap.isOpened():
            print("[ERROR] Не вдалося відкрити потік. Перевірте STREAM_URL.")
            return

        while self.running:
            ok, frame = cap.read()
            if not ok:
                continue

            # Застосувати ROI, якщо задано
            display = frame.copy()
            frame_for_infer = frame
            if roi is not None:
                x1, y1, x2, y2 = roi
                x1 = max(0, min(x1, frame.shape[1]-1))
                x2 = max(0, min(x2, frame.shape[1]-1))
                y1 = max(0, min(y1, frame.shape[0]-1))
                y2 = max(0, min(y2, frame.shape[0]-1))
                sub = frame[y1:y2, x1:x2]
                frame_for_infer = sub

            # Інференс YOLOv8
            results = model.predict(source=frame_for_infer,
conf=conf_threshold, verbose=False)

            dets = []
            try:
                r = results[0]
                if r.boxes is not None and len(r.boxes) > 0:
                    for b in r.boxes:
                        cls = int(b.cls[0].item())
                        conf = float(b.conf[0].item())
                        x1b, y1b, x2b, y2b = b.xyxy[0].tolist()
                        # Перенесення координат з субкадру у глобальні, якщо
ROI
                        if roi is not None:
                            x1g = x1 + x1b; y1g = y1 + y1b
                            x2g = x1 + x2b; y2g = y1 + y2b
                        else:
                            x1g, y1g, x2g, y2g = x1b, y1b, x2b, y2b
                        dets.append((cls, conf, (x1g, y1g, x2g, y2g)))
            except Exception as e:
                dets = []

            last_detections = dets

            # Оновити трекер та отримати унікальні лічильники
            unique_counts = tracker.update(dets)
```

```
            # Візуалізація
            for cls, conf, (x1b, y1b, x2b, y2b) in dets:
                cv2.rectangle(display, (int(x1b), int(y1b)), (int(x2b),
int(y2b)), (0, 255, 0), 2)
                label = f"{model.names.get(cls, str(cls))} {conf*100:.0f}%"
                cv2.putText(display, label, (int(x1b), int(y1b)-6),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,255,0), 1)

            if roi is not None:
                cv2.rectangle(display, (roi[0], roi[1]), (roi[2], roi[3]),
(255, 255, 0), 2)
                cv2.putText(display, "ROI", (roi[0], roi[1]-6),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255,255,0), 2)

            # Оновлення статистики у GUI
            name_map = {i: n for i, n in model.names.items()}
            stats_text = ", ".join(f"{name_map.get(k, k)}: {v}" for k, v in
sorted(unique_counts.items()))
            if stats_text == "": stats_text = "-"
            self.stats_label.config(text=stats_text)

            # Відобразити у вікні Tk
            # Примітка: для простоти використовуємо imencode->PhotoImage?
Натомість скористаємось OpenCV вікном:
            # Але умова – все в Tkinter. Тож конвертуємо кадр у формат, що
розуміє Tk.
            bgr = display
            rgb = cv2.cvtColor(bgr, cv2.COLOR_BGR2RGB)
            h, w = rgb.shape[:2]
            # масштаб для вікна
            scale = 800.0 / max(h, w)
            if scale < 1.0:
                rgb = cv2.resize(rgb, (int(w*scale), int(h*scale)),
interpolation=cv2.INTER_AREA)
            last_frame = display.copy()

            # Конвертація в TK PhotoImage через PIL (уникнути зайвих
залежностей не вийде)
            try:
                from PIL import Image, ImageTk
                img = ImageTk.PhotoImage(image=Image.fromarray(rgb))
                self.frame_label.configure(image=img)
                self.frame_label.image = img
            except Exception as e:
                # Якщо PIL недоступна – запасний варіант (відкрити вікно
OpenCV)
                cv2.imshow("Preview (fallback OpenCV window)", display)
                cv2.waitKey(1)

        cap.release()
        cv2.destroyAllWindows()

    def on_close(self):
        self.running = False
        self.master.after(200, self.master.destroy)

def main():
    root = Tk()
    app = App(root)
    root.mainloop()

if __name__ == "__main__":
    main()
```

В.4. Допоміжний модуль Python для читання локальних відеофайлів або USB-камери (для офлайн-тестів, ті самі налаштування детекції, збереження метаданих)

```
# ============================================================
# ДОДАТОК Б, Модуль В.4
# Офлайн/USB тестер: заміна потоку ESP32-CAM на локальне відео або вебкамеру.
# Коментарі українською мовою.
# ============================================================

import cv2
import json
from datetime import datetime
from ultralytics import YOLO

SOURCE = 0                  # 0 для вебкамери, або шлях до .mp4/.avi
MODEL_PATH = "yolov8n.pt"
CONF = 0.35

model = YOLO(MODEL_PATH)

def save_with_meta(frame, detections, prefix="offline"):
    ts = datetime.now().strftime("%Y%m%d_%H%M%S")
    img_name = f"{prefix}_{ts}.jpg"
    txt_name = f"{prefix}_{ts}.txt"
    cv2.imwrite(img_name, frame)
    meta = {
        "timestamp": ts,
        "confidence_threshold": CONF,
        "detections": [
            {"cls": int(c), "conf": float(cf), "box": [float(x) for x in
box]}
            for (c, cf, box) in detections
        ]
    }
    with open(txt_name, "w", encoding="utf-8") as f:
        f.write(json.dumps(meta, ensure_ascii=False, indent=2))
    print(f"[INFO] Збережено: {img_name}, {txt_name}")

def run():
    cap = cv2.VideoCapture(SOURCE)
    if not cap.isOpened():
        print("[ERROR] Не вдалося відкрити джерело")
        return
    while True:
        ok, frame = cap.read()
        if not ok:
            break

        results = model.predict(source=frame, conf=CONF, verbose=False)
        dets = []
        r = results[0]
        if r.boxes is not None and len(r.boxes) > 0:
            for b in r.boxes:
                cls = int(b.cls[0].item())
                conf = float(b.conf[0].item())
                x1, y1, x2, y2 = b.xyxy[0].tolist()
                dets.append((cls, conf, (x1, y1, x2, y2)))
                cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)),
(0,255,0), 2)
                name = model.names.get(cls, str(cls))
                cv2.putText(frame, f"{name} {conf*100:.0f}%", (int(x1),
int(y1)-6),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,255,0), 1)
```

```
        cv2.imshow("Offline/USB detector", frame)
        key = cv2.waitKey(1) & 0xFF
        if key == ord('s'):
            save_with_meta(frame, dets, prefix="offline")
        if key == 27:  # ESC
            break

    cap.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    run()
```

B.5. Мінімальний модуль ESP32-CAM для сумісності з Python/OpenCV без MJPEG (HTTP-знімок за запитом)

```
/* ============================================================
   ДОДАТОК Б, Модуль B.5
   ESP32-CAM: простий HTTP-сервер, який віддає одиночний JPEG-знімок
   за запитом /jpg. Зручно для тестів OpenCV (cv2.VideoCapture з urlopen).
   Коментарі українською мовою.
   ============================================================ */

#include "esp_camera.h"
#include <WiFi.h>
#include "esp_http_server.h"

const char* ssid = "YourSSID";
const char* password = "YourPASS";

// Розпінування AI-Thinker як у модулі B.2 (не дублюємо для стислості)
httpd_handle_t camera_httpd = NULL;

static esp_err_t jpg_handler(httpd_req_t *req){
  camera_fb_t * fb = esp_camera_fb_get();
  if (!fb) {
    httpd_resp_send_500(req);
    return ESP_FAIL;
  }
  httpd_resp_set_type(req, "image/jpeg");
  httpd_resp_set_hdr(req, "Content-Disposition", "inline;
filename=capture.jpg");
  httpd_resp_send(req, (const char *)fb->buf, fb->len);
  esp_camera_fb_return(fb);
  return ESP_OK;
}
void startCameraServer(){
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  httpd_uri_t jpg_uri = { .uri="/jpg", .method=HTTP_GET,
.handler=jpg_handler, .user_ctx=NULL };
  if (httpd_start(&camera_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(camera_httpd, &jpg_uri);
  }
}

void setup() {
  // ... ініціалізація камери (аналогічно B.2), Wi-Fi підключення ...
  startCameraServer();
}
void loop() {}
```

Додаток Г (обов'язковий)

Протокол перевірки кваліфікаційної роботи

Назва роботи: «Проєктування компактної системи комп'ютерного зору для автономної роботизованої платформи»

Тип роботи: магістерська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра АІІТ
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) 0.06 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.
- У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

Бісікало О.В., зав. каф. АІІТ _____ (підпис)
(прізвище, ініціали, посада)

Овчинников К.В., доц. каф. АІІТ _____ (підпис)
(прізвище, ініціали, посада)

Особа, відповідальна за перевірку _____ Маслій Р.В.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Ковтун В. В., д.т.н., доцент кафедри КСУ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Поліщук Я. В.
(підпис) (прізвище, ініціали)