НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Приладобудівний факультет
Кафедра комп'ютерно-інтегрованих оптичних та навігаційних систем

До захисту допущено:
Завідувач кафедри
_____ Бурау Н.І.
«__» _____ 2025 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютерно-інтегровані системи та технології в приладобудуванні»
спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»
на тему: «Система автоматичного керування транспортними засобами»

Виконав:
Студент IV курсу, групи ПО-11
Конфалія Олексій Олександрович _____
Керівник:
Старший викладач, к.т.н.
Мамута Марина Сергіївна _____
Рецензент:
Доцент к.т.н.
Барандич Катерина Сергіївна _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2025 рік

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Приладобудівний факультет

Кафедра комп'ютерно-інтегрованих оптичних та навігаційних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»

Освітньо-професійна програма «Комп'ютерно-інтегровані системи та технології в приладобудуванні»

ЗАТВЕРДЖУЮ

Завідувач кафедри КІОНС

_____ Надія БУРАУ

«___»_____2025_ р.

ЗАВДАННЯ

на дипломну роботу студента

Конфалії Олексія Олександровича

1. Тема проекту «Система автоматичного керування транспортними засобами» затверджена наказом по університету від «__»_.___ 2025 р. №_____

2. Термін подання студентом дипломної роботи: 9.06.2025 р.

3. Вихідні дані до роботи: Камера ESP32-CAM з передачею зображення на веб-сервер, Bluetooth-модуль HC-06, плата Arduino Uno з драйвером двигунів DRV8833 , мова програмування python, виявлення найяскравішої точки

4. Зміст пояснювальної записки:

Вступ.

Розділ 1. Класифікація і аналіз систем відстеження.

Розділ 2. Метод виявлення яскравої точки зображення та його інтеграція в систему відстеження.

Розділ 3. Схема керування автономною платформою

5. Перелік графічного матеріалу, включеного до презентації: презентація, що включає схеми та графіки

Дата видачі завдання: 22.05.2025 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Огляд літературних джерел за темою ДР	30.04.2025	
2	Вибір методу відстеження об'єктів.	05.05.2025	
3	Налаштування плати Arduino Uno з усіма модулями, інтеграція в єдину систему	12.05.2025	
4	Розробка програмного забезпечення	19.05.2025	
5	Тестування системи	26.05.2025	
6	Оформлення ПЗ	02.06.2025	
7	Попередній захист	09.06.2025	

Студент Олексій Конфалія
Керівник Марина Мамута

Анотація

У дипломній роботі розроблено систему автоматичного керування мобільною платформою з використанням методів комп’ютерного зору. Основною метою є реалізація оптичного відстеження яскравої точки (лазерного маркера) для автономної навігації транспортного засобу. Проведено аналіз існуючих систем відстеження, зокрема GPS, RFID, LiDAR та інерціальних, а також досліджено алгоритми виявлення та трекінгу об’єктів, такі як SIFT, YOLO, фільтр Калмана, метод Лукаса-Канаде.

У практичній частині використано камеру ESP32-CAM для захоплення відеопотоку, Arduino Uno для керування двигунами за допомогою драйвера DRV8833, та модуль HC-06 для бездротової передачі керувальних команд. Система реалізована у вигляді взаємодії програмного забезпечення ПК з мікроконтролерами, що дозволяє досягти автономного руху відповідно до координат яскравого орієнтира.

Результати тестування підтверджують працездатність системи, здатність реагувати на зміну положення маркера та стабільно виконувати навігаційні задачі. Отримана система може бути використана як основа для побудови роботизованих платформ у навчальних, дослідницьких або спеціалізованих прикладних задачах.

Ключові слова: комп’ютерний зір, ESP32-CAM, Arduino, автономна платформа, трекінг, лазерний маркер.

Abstract

The thesis presents the development of an automatic control system for a mobile platform using computer vision techniques. The main goal is to implement optical tracking of a bright spot (laser marker) for autonomous vehicle navigation. A comprehensive analysis of existing tracking systems was carried out, including GPS, RFID, LiDAR, and inertial systems, as well as object detection and tracking algorithms such as SIFT, YOLO, Kalman filter, and Lucas-Kanade method.

The practical part involves using the ESP32-CAM camera for video streaming, Arduino Uno for motor control via the DRV8833 driver, and the HC-06 module for wireless command transmission. The system is implemented through interaction between PC software and microcontrollers, enabling autonomous movement based on the coordinates of a laser pointer.

Testing results confirm the system's operability, responsiveness to marker movement, and its ability to perform navigation tasks reliably. The developed system can serve as a foundation for robotic platforms in educational, research, or specialized applications.

Keywords: computer vision, ESP32-CAM, Arduino, autonomous platform, tracking, laser marker.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АЦП – Аналого-цифровий перетворювач

ШІМ – Широтно-імпульсна модуляція

BLE – Bluetooth Low Energy (Bluetooth з низьким енергоспоживанням)

BRIEF – Binary Robust Independent Elementary Features (Бінарні надійні незалежні елементарні ознаки)

CCD – Charge-Coupled Device (Пристрій із зарядовим зв’язком)

CFA – Color Filter Array (Масив кольорових фільтрів)

CMOS – Complementary Metal-Oxide-Semiconductor (Комплементарна метал-оксид-напівпровідникова технологія)

DVP – Digital Video Port (Порт цифрового відео)

FAST – Features from Accelerated Segment Test (Ознаки з прискореного сегментного тесту)

FMCW – Frequency-Modulated Continuous Wave (Частотно-модульований безперервний сигнал)

GNSS – Global Navigation Satellite System (Глобальна навігаційна супутникова система)

GPIO – General-Purpose Input/Output (Універсальні порти введення/виведення)

GPS – Global Positioning System (Глобальна система позиціонування)

HTTP – Hypertext Transfer Protocol (Протокол передавання гіпертексту)

I²C – Inter-Integrated Circuit (Інтерфейс між інтегрованими схемами)

IMU – Inertial Measurement Unit (Інерціальна вимірювальна система)

JPEG – Joint Photographic Experts Group (Група експертів з фотозображень)

LiDAR – Light Detection and Ranging (Лазерне виявлення та вимірювання відстані)

MEMS – Micro-Electro-Mechanical Systems (Мікроелектромеханічні системи)

MJPEG – Motion JPEG (Рухомі JPEG-зображення)

NMS – Network Management System (Система керування мережею)

ORB – Oriented FAST and Rotated BRIEF (Орієнтовані FAST та обернені BRIEF)

PCLK – peripheral clock (Тактова частота периферійного пристрою)

PSRAM – Pseudo-Static RAM (Псевдостатична оперативна пам’ять)

PWDN – Power Down (Режим зниженого енергоспоживання)

RAM – Random Access Memory (Оперативна пам’ять)

RFID – Radio Frequency Identification (Ідентифікація радіочастотами)

RTK – Real-Time Kinematic (Кінематичне позиціонування в реальному часі)

SCCB – Serial Camera Control Bus (Серійна шина керування камерою)

SD – Secure Digital (Карта пам’яті)

SIFT – Scale-Invariant Feature Transform (Незмінні до масштабу ознаки)

SURF – Speeded-Up Robust Features (Прискорені надійні ознаки)

TCP – Transmission Control Protocol (Протокол керування передачею)

UART – Universal Asynchronous Receiver/Transmitter (Універсальний асинхронний приймач/передавач)

Wi-Fi – Wireless Fidelity (Бездротова передача даних)

XCLK – External Clock (Зовнішній тактовий сигнал)

YOLO – You Look Only Once (Ти лише раз дивишся)

ВСТУП

Сучасний розвиток технологій у галузі автоматизації та штучного інтелекту призводить до значних змін у транспортній сфері. Зокрема, зростає інтерес до створення автономних мобільних платформ, здатних самостійно орієнтуватися в просторі та приймати рішення без участі оператора. Такі системи широко застосовуються в логістиці, агропромисловості, військовій справі, розумних містах, а також у сфері наукових досліджень і освітніх проєктів.

Одним із ключових напрямів автоматизації транспортних засобів є розробка систем оптичного відстеження — технологій, що використовують камери та алгоритми обробки зображень для визначення положення та траєкторії об’єктів у навколишньому середовищі. На відміну від традиційних методів навігації (GPS, інерціальних датчиків, LiDAR), системи комп’ютерного зору дозволяють реагувати на візуальні сигнали та відслідковувати об’єкти навіть у складних або змінних умовах.

Метою даної дипломної роботи є розробка та практична реалізація системи автоматичного керування рухом транспортного засобу на основі оптичного виявлення яскравої точки — лазерного маркера. Такий підхід дозволяє організувати інтуїтивно зрозумілу модель управління, де оператор може задавати ціль, просто направляючи лазер на поверхню, а система автоматично слідуватиме за ним.

У рамках роботи було проаналізовано основні типи систем відстеження, розглянуто можливості алгоритмів комп’ютерного зору (включаючи методи SIFT, YOLO, Калманівський фільтр, Лукаса-Канаде), обґрунтовано вибір обладнання (ESP32-CAM, Arduino Uno, DRV8833, HC-06), а також розроблено програмну частину на мові Python для аналізу відеопотоку та передавання команд. Окрему увагу приділено питанням точності позиціонування,

стабілізації руху, оптимізації роботи сенсорів та тестуванню системи в реальних умовах.

Актуальність даної теми зумовлена глобальним трендом на автоматизацію та роботизацію транспортних систем, а також потребою в бюджетних та доступних рішеннях для прототипування інтелектуальних автономних платформ.

РОЗДІЛ 1. КЛАСИФІКАЦІЯ І АНАЛІЗ СИСТЕМ ВІДСТЕЖЕННЯ.

1.1 Загальний огляд систем відстеження

Системи відстеження — це не просто набір сенсорів і електроніки. Вони відіграють роль центрального елемента в сучасних автоматизованих платформах, забезпечуючи здатність не тільки фіксувати наявність об’єкта, а й визначати його координати, швидкість, напрям руху й динаміку зміни положення. Саме завдяки таким системам автономні пристрої здатні коректно реагувати на зміну умов середовища, приймати керуючі рішення в реальному часі та адаптувати свою поведінку.

Застосування систем відстеження охоплює широкий спектр галузей: транспорт, автономна навігація, промислові лінії, медицина, військова техніка, доповнена реальність, розумне місто. У кожному випадку центральним залишається завдання — визначити поточне положення об’єкта, його переміщення та взаємодію з іншими об’єктами середовища.

Зокрема, у сфері доповненої реальності системи відстеження відіграють ключову роль у забезпеченні точного позиціонування та навігації. За даними дослідження MarketsandMarkets [1], світовий ринок навігації з доповненою реальністю оцінювався в 1,17 мільярда доларів США у 2024 році та, за прогнозами, досягне 6,33 мільярда доларів США до 2029 року, демонструючи середньорічний темп зростання 40,3% протягом прогнозованого періоду.

Робота системи відстеження передбачає чотири основні етапи. Перший — виявлення об’єкта або події за допомогою сенсорів. Другий — визначення координат, положення, кута або швидкості. Третій — аналіз та інтерпретація інформації на рівні алгоритмів, моделей або фільтрів. І четвертий —

формування реакції та послідовності дій: сигнал, логування або автоматичне коригування дій системи. Попри те, що сам процес відстеження здається простим, насправді в системі працює складна багаторівнева логіка, яка: аналізує, чи можна довіряти поточному джерелу даних або якщо сигнал слабкий або ненадійний, вирішує, чи потрібно підключити інші типи даних.

Оптичні технології, зокрема комп'ютерний зір, стали ключовими в останні роки. Використання нейронних мереж дозволило суттєво підвищити точність розпізнавання об'єктів і передбачення їх траєкторій руху. У той же час вбудовані системи з обмеженими ресурсами продовжують використовувати простіші підходи — фільтрацію за кольором, сегментацію за формою або аналіз контрасту [2].

Супутникова навігація залишається основним джерелом саме глобального позиціонування. Технології RTK дозволяють знизити похибку позиціонування супутникової навігації до 2 см [3]. Вони лежать в основі систем точного землеробства, автономного транспорту й дронів. Попри це, у закритих або екранованих приміщеннях GPS недоступний, що створює потребу в альтернативних локальних рішеннях. Серед таких альтернатив — технології на базі радіосигналів і звуку. RFID, BLE або ультразвук дають можливість точно визначати розташування в межах складських комплексів, медичних установ або критично важливих зон. Наприклад, BLE-маяки забезпечують точність від 2 до 5 метрів при середньому енергоспоживанні 30–50 мкВт [4]. Ультразвукові системи — точніші, але мають обмеження по дальності та прямій видимості.

Інерціальні сенсори стали стандартом для стабілізації положення. Хоча їхня похибка накопичується з часом, вони забезпечують безперервність навігаційного рішення навіть при втраті зовнішніх джерел. При правильному налаштуванні IMU забезпечує точність до 0.05° у динамічних сценаріях [5].

Нажаль, ідеальних технологій не існує, тому використовується комбінований підхід — мультисенсорна інтеграція. Метод злиття даних

датчиків дозволяє підвищити надійність і точність завдяки об’єднанню даних з кількох джерел. Статистичні методи, фільтри Калмана, алгоритми оптимізації та глибокі нейронні мережі — усе це сьогодні є частиною єдиної архітектури керування.

Системи відстеження стали невід'ємною складовою більшості кіберфізичних платформ. У 2024 році глобальний ринок систем внутрішнього позиціонування та навігації оцінювався в понад $35 мільярдів, із прогнозованим середньорічним темпом зростання близько 20%. Це свідчить про стрімке зростання інтересу до технологій, що забезпечують точне позиціонування та навігацію в реальному часі. З огляду на розвиток Інтернету речей та автономних технологій, інтеграція систем відстеження є не просто бажаною, а необхідною умовою ефективної роботи майбутніх платформ [6].

1.1.1 GPS-системи відстеження у транспортних засобах

GPS — знайома багатьом технологія, особливо тим, хто хоч раз в житті користувався навігацією в смартфоні. Але якщо подивитися глибше, то за цим простим словом стоїть складна глобальна система. Це мережа супутників, які постійно обертаються навколо Землі, передаючи сигнали в усі частини планети. Ці сигнали — не просто умовні імпульси. Вони містять точні дані про координати, час і положення супутника у момент передачі. Саме завдяки цій інформації транспортні засоби можуть орієнтуватися у просторі й знати своє розташування з високою точністю.

В основі роботи GPS лежить цілком класична геометрія, а точніше — метод тріангуляції. Кожен супутник, який передає сигнал, надсилає приймачу не лише координати свого положення, а й момент часу, коли саме цей сигнал було передано.

Далі в дію вступає аналіз затримки. Якщо приймач знає, коли сигнал надійшов і коли він був відправлений, він може розрахувати відстань до супутника. Чим більша затримка, тим далі об’єкт. Це схоже на вимірювання

відлуння, тільки замість звуку — радіохвилі, замість лісу — орбіта, а замість секундоміра — атомний годинник.

Щойно пристрій отримає сигнали від кількох супутників, він починає шукати перетин сфер. У кожного супутника є умовна сфера радіуса, що відповідає відстані до приймача. І вже на перетині цих сфер знаходиться точка — тобто реальне місце розташування.

Для тривимірного позиціонування, з урахуванням висоти, потрібно щонайменше чотири супутники. Але чим більше — тим точніше. П’ять, шість, вісім — і система може компенсувати не лише геометричні неточності, а й похибки годинника.

Кожен супутник надсилає сигнал із точно зафіксованим часом відправлення. Приймач, наприклад GPS-пристрій, приймає цей сигнал і фіксує час його отримання. Визначивши, скільки часу пройшло між відправленням і прийманням сигналу, та помноживши цей час на швидкість світла, можна дізнатися, яку відстань пройшов сигнал — тобто, наскільки далеко перебуває супутник.

Щоб визначити місцезнаходження приймача в просторі, використовуються дані щонайменше від чотирьох супутників. Для кожного з них відома його точна позиція в космосі. Також розраховується відстань від приймача до кожного супутника.

Знаючи ці чотири відстані та координати супутників, можна визначити координати самого приймача — його положення в просторі. Це подібно до того, як можна знайти своє місцезнаходження, якщо знати, наскільки далеко ти знаходишся від кількох відомих орієнтирів.

Зобразити це можна як на рис.1.1.

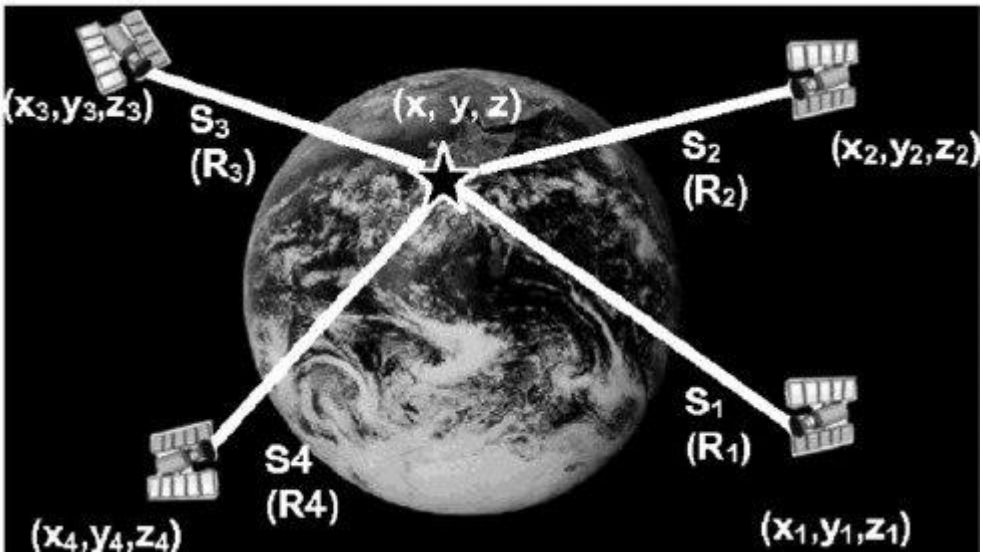

Рис. 1.1 Демонстрація визначення положення GPS на основі трилатерації [7]

При цьому залишається одна проблема.GPS — річ чутлива. Його сигнал можна втратити в місті серед хмарочосів або в тунелі. Тому точність коливається: від кількох метрів до кількох сантиметрів, якщо система підтримує додаткові технології.

Розглядаючи точність GPS неможливо не згадати про вплив теорії відносності на годинники на супутниках, які йдуть швидше, ніж на Землі. Рух супутника приблизно дорівнює 14 000 км/год, що сповільнює час на приблизно 7,2 μс/добу. Супутники знаходяться на висоті 20 200 км, де гравітаційний потенціал відрізняється від земного, що прискорює час на приблизно 45,9 μс/добу. Тому сумарна поправка дорівнює +38 μс/добу. За добу нескомпенсована різниця призвела б до похибки.

Саме через це годинники супутників настроюють на частоту 10,229999995453 MHz замість номінальної 10,23 MHz. Це компенсує релятивістський ефект ще до виходу сигналу.

Зараз активно шукають, як зробити GPS ще надійнішим. Наприклад, об’єднують його з іншими сенсорами — гіроскопами, лідарами, навіть камерами [8].

1.1.2 RFID-системи відстеження у транспортних засобах

RFID — це технологія радіочастотної ідентифікації, яка працює за простим і водночас ефективним принципом. У системі зазвичай є мітка або тег, що містить унікальний код, і зчитувач, який фіксує цей код. Передача даних відбувається за допомогою радіохвиль, тобто без потреби у фізичному контакті або прямій видимості між пристроями.

У класичному сценарії RFID-мітка закріплюється на об’єкті — вантажі, транспортному засобі чи упаковці. Коли об'єкт потрапляє в зону дії електромагнітного поля, яке створює зчитувач, мітка миттєво активується та передає закодовану інформацію. Далі система зчитує ці дані, обробляє їх і зберігає або передає для подальших дій. Таким чином, ідентифікація відбувається автоматично та безперервно.

Щоб зрозуміти це на прикладі, уявімо собі вантажівку з RFID-міткою, що в’їжджає на територію складу. На в’їзді стоїть зчитувач. Як тільки транспорт наближається, система без будь-яких додаткових дій фіксує його прихід, ідентифікує номер, час і місце події.

Мітки бувають різних типів. Пасивні RFID-мітки не мають батареї — починають виконувати дії лише під впливом поля зчитувача. Це найдешевший і найпоширеніший варіант, особливо у сфері логістики. Активні мітки оснащені власним джерелом живлення, тож можуть передавати сигнал на десятки або навіть сотні метрів. Напівпасивні мітки мають батарею, але працюють у відповідь на сигнал зчитувача — це проміжний варіант з розширеними функціями.

Типова система RFID складається зі зчитувача, антени та міток. Усі елементи з'єднуються із центральною обчислювальною системою, де відбувається аналіз інформації. Залежно від задачі — від реєстрації подій до формування звітів — система адаптує свої дії. елементи RFID системи та зв’язки між ними зображено на рис 1.2.

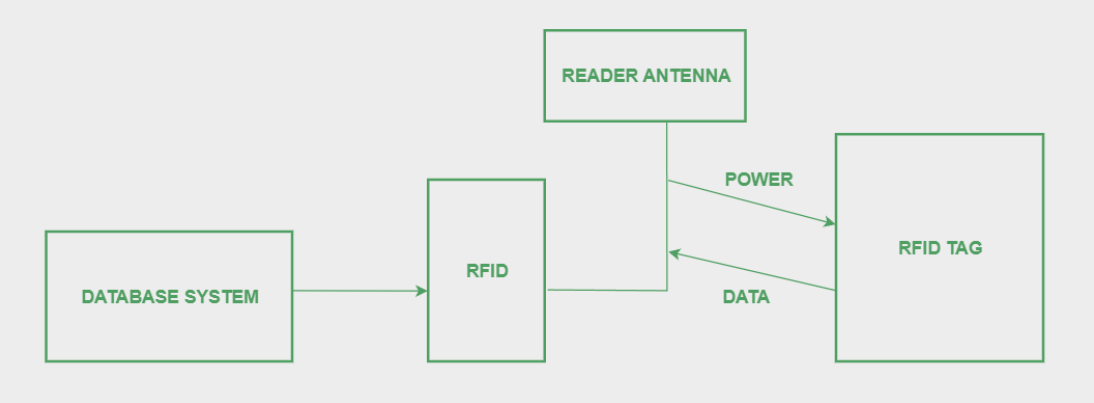

Рис. 1.2 Демонстрація роботи RFID-системи [9]

Головна перевага RFID полягає в тому, що система працює автоматично, без потреби в людському втручанні. Замість ручного введення даних про прибуття, система все фіксує самостійно — за частки секунди. Це економить час, підвищує точність і зменшує ризик помилок.

Технологія має чимало сильних сторін. Зчитування мітки відбувається практично миттєво, навіть без зупинки об’єкта. Вся система працює безконтактно — немає потреби натискати кнопки чи підносити картку до сканера. RFID-мітки стійкі до забруднень — вони функціонують навіть якщо покриті пилом або розміщені під матеріалами на кшталт брезенту. А ще — можна одночасно зчитувати десятки, а іноді й сотні міток, що особливо важливо в умовах складу або логістичного терміналу.

Втім, як і будь-яка технологія, RFID має свої обмеження. Радіохвилі можуть частково гаситися або спотворюватися під дією металу чи рідин, що призводить до втрати сигналу. В окремих випадках зони зчитування перекриваються, і система може реєструвати фантомні об’єкти. Велика кількість міток поблизу також створює перевантаження, що ускладнює точне розпізнавання. І, зрештою, якщо не застосовувати шифрування, сторонні особи можуть перехопити сигнал і зчитати дані без дозволу.

У сфері транспорту RFID знайшов широке застосування. Його використовують на автостоянках і пропускних пунктах для організації безконтактного доступу. У логістиці система допомагає відстежувати переміщення вантажів, у громадському транспорті — автоматизує облік

пасажирів і спрощує оплату. На залізничних станціях RFID забезпечує ідентифікацію вагонів при формуванні составів.

І це лише частина можливостей. З кожним роком RFID інтегрується у все більше галузей. Його використовують у міському просторі для смарт-зупинок, в електромобілях — для безконтактної оплати заряджання, у контрольованих перевезеннях — для моніторингу температури й умов доставки [9].

1.1.3 Інерціальні системи IMU

Інерціальні навігаційні системи займають важливе місце в структурі сучасних засобів визначення положення та орієнтації об’єктів у просторі. З огляду на зростання автономності роботизованих платформ, безпілотних транспортних засобів та авіаційних комплексів, зростає потреба у високоточної навігації в умовах обмеженого доступу до зовнішніх джерел координат. Цей розділ присвячено аналізу принципів роботи IMU, їх математичному опису та порівнянню основних методів реалізації.

Класична інерціальна система складається з трьох осей акселерометрів та трьох осей гіроскопів. У деяких випадках додається магнітометр, утворюючи 9-осьову систему. На рис. 1.3 наведено приклад інерційного вимірювального пристрою:

Рис. 1.3 Інерційний вимірювальний пристрій MS-IMU3025 [10]

Відповідно до типу реалізації, виділяють наступні класи IMU:

— Механічні — засновані на маятникових або гіроскопічних ефектах, мають високу точність, але габаритні та дорогі.
— Лазерні гіроскопи — використовують ефект Саньяка, застосовуються в авіації та космічній техніці.
— Волоконно-оптичні гіроскопи — забезпечують високоточне визначення кута повороту на базі інтерферометрії.
— Мікроелектромеханічні системи — легкі, мініатюрні, недорогі; широко використовуються в мобільних пристроях, дронах і портативних сенсорах.

Найбільше поширення у споживчій та робототехнічній галузях отримали MEMS IMU, з огляду на баланс між вартістю, габаритами та споживанням енергії.

Фізично IMU вимірює:

— лінійне прискорення в системі координат сенсора;
— кутову швидкість навколо трьох осей.

На основі цих величин обчислюється положення та орієнтація в просторі.

Фізично інерціальна вимірювальна система вимірює два ключові параметри: лінійне прискорення у власній системі координат та кутову швидкість, яка описує обертальний рух навколо трьох взаємно перпендикулярних осей. Ці величини є основою для побудови навігаційних розрахунків положення об'єкта у просторі та його орієнтації.

Для представлення орієнтації у просторі використовують кілька математичних моделей. Одним з найпоширеніших способів є матриця напрямних косинусів — 3×3 матриця, елементи якої визначають косинуси кутів між осями інерціальної та локальної систем координат. Альтернативно застосовуються куатерніони, що дозволяють уникнути проблем сингулярностей, характерних для системи кутів Ейлера, які, своєю чергою, зручні для інтерпретації й візуалізації орієнтації.

Щоб визначити швидкість і положення об'єкта у просторі, використовують дані з акселерометрів — датчиків, які вимірюють прискорення. Для цього спочатку обчислюють швидкість, накопичуючи (інтегруючи) значення прискорення з часом. Потім, використовуючи обчислену швидкість, визначають положення об'єкта — знову шляхом накопичення (інтегрування) вже швидкості.

У реальних пристроях ці обчислення виконуються цифровим способом, тобто дискретно, з урахуванням обмежень частоти вимірювань та впливу шумів у даних, які можуть спотворювати результат.

Кутова швидкість використовується для обчислення орієнтації з використанням кінематичних рівнянь обертання. У цьому контексті особливо важливо враховувати ефект накопичення похибок, який виникає внаслідок інтегрування навіть незначних шумів. Для зменшення впливу таких похибок у сучасних IMU активно використовують фільтр Калмана — оптимальний рекурсивний алгоритм оцінювання, що дозволяє поєднувати інформацію від IMU з іншими джерелами, такими як GNSS чи магнітометр.

У контексті MEMS-сенсорів особливо гостро стоїть проблема дрейфу гіроскопів, спричиненого температурною нестабільністю, низькою якістю еталонів та шумами АЦП. Щоб мінімізувати ці ефекти, застосовуються методи калібрування, температурної компенсації, а також сенсорне злиття, що дозволяє зменшити вплив помилок кожного окремого сенсора. Типовими додатковими джерелами є супутникові системи, барометри для оцінки висоти та магнітометри для абсолютної орієнтації по магнітному полю Землі [11].

1.1.4 LiDAR-системи

Технологія LiDAR, набула широкого застосування у сфері автономної навігації, картографування, геодезії, робототехніки та розумної інфраструктури. Зі зростанням потреб у високоточному тривимірному скануванні та побудові цифрових моделей місцевості, LiDAR-системи стали незамінним елементом для формування детальної просторової інформації про навколишнє середовище. Мета цього розділу — розглянути основні типи LiDAR-систем, принципи їх роботи, математичні моделі обробки сигналів, а також порівняти їх ефективність у контексті навігаційних і сенсорних систем.

Принцип дії LiDAR ґрунтується на визначенні часу проходження світлового імпульсу до об'єкта та назад до приймача, що дозволяє обчислити відстань до цілі. Графічно принцип дії наведено на рис 1.4.

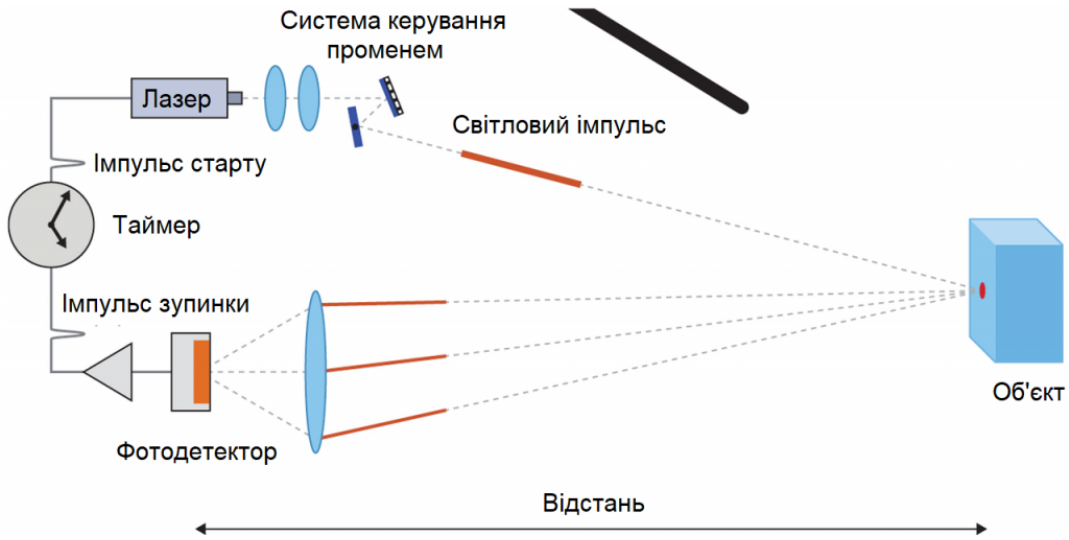

Рис. 1.4 Принцип роботи оптичного далекоміра типу LiDAR [12]

Залежно від архітектури передавання й обробки сигналу, LiDAR-системи реалізуються у різних конструктивних варіантах. Однією з найбільш розповсюджених є пульсуюча архітектура, де лазер випромінює короткі імпульси, і за допомогою точного хронометрування визначається відстань. Такі системи забезпечують добру дальність вимірювання і часто застосовуються у геодезії та автомобільній навігації.

Альтернативою є безперервно-хвильові системи з частотною модуляцією, які замість окремих імпульсів використовують лазер з лінійною модуляцією частоти. Внаслідок аналізу доплерівського зсуву у зворотному сигналі можна не лише визначати положення, але й оцінювати швидкість руху об’єкта

Механічні системи, які фізично обертають лазерний блок для забезпечення широкого поля огляду, залишаються поширеними через свою простоту реалізації, але мають вразливість до зносу і механічних збоїв. Водночас твердотільні рішення, що ґрунтуються на фазованих антенних решітках або мікромеханічних дзеркалах, виключають рухомі частини й демонструють високу надійність. Їхня головна перевага — мініатюрність і відсутність потреби в механічному обслуговуванні, однак вони поки що обмежені в робочій дальності й кутовому покритті.

Кожен тип LiDAR має свої переваги й компроміси, а вибір залежить від конкретної задачі: мобільне картографування, контроль трафіку або автономна навігація у складному середовищі.

У LiDAR-системах відстань до об'єкта визначається за часом, за який лазерний імпульс доходить до об'єкта і повертається назад. Оскільки сигнал проходить шлях туди й назад, цей час ділять навпіл і множать на швидкість світла — так отримують відстань.

Коли лазерний сигнал відбивається від об’єктів, система фіксує не лише час, а й напрямок, у якому був надісланий імпульс. Завдяки цьому кожна точка, від якої відбився сигнал, визначається у тривимірному просторі.

Для побудови повної 3D-карти враховуються кути, під якими здійснюється сканування:

— горизонтальний кут — вказує напрямок обертання навколо вертикальної осі;

— вертикальний кут — вказує нахил вгору або вниз.

Таким чином, кожна точка в просторі визначається трьома параметрами: відстанню до неї та двома кутами — і вся сцена поступово “малюється” з безлічі таких точок.

Для покращення якості сканування використовуються фільтри просторового згладжування, кластеризація точок, алгоритми виявлення площин та методи розпізнавання об'єктів. Інтеграція з GPS та IMU дозволяє реєструвати дані у глобальних системах координат.

LiDAR-системи є високотехнологічними сенсорними пристроями, що дозволяють з високою просторовою точністю формувати тривимірне уявлення про навколишнє середовище. Завдяки здатності працювати на значних відстанях, незалежно від рівня зовнішнього освітлення, вони стали важливим інструментом для створення детальних тривимірних карт місцевості. Така функціональність є особливо цінною для систем автономної навігації, геодезичних задач і високоточних вимірювальних платформ.

Разом з тим, існують певні експлуатаційні обмеження. Зокрема, робота LiDAR-систем значною мірою залежить від погодних умов — туман, дощ або сильне запилення можуть знижувати точність або повністю блокувати прийом сигналу. Крім того, апаратна складність та високі вимоги до оптичних компонентів, особливо у випадку FMCW або твердотільних реалізацій, призводять до значної вартості обладнання. Окрему складність становлять об’єкти з високою прозорістю або дзеркальні поверхні, які можуть некоректно відбивати лазерне випромінювання, що знижує ефективність зчитування даних.

Попри зазначені обмеження, LiDAR залишається стратегічно важливою технологією для автономного позиціонування. Її інтеграція з іншими

сенсорними системами, зокрема камерами, IMU та GNSS, а також використання сучасних методів аналізу даних, таких як глибинне навчання та семантична класифікація, відкриває нові горизонти у застосуванні та функціональності подібних систем у транспорті, мобільній робототехніці та інтелектуальній інфраструктурі [13].

1.1.5 Ультразвукові системи

Ультразвукові технології відіграють значну роль у задачах позиціонування, контролю відстані, об’ємного зондування та запобігання зіткнень, особливо в умовах обмеженої видимості або вартісних обмежень. Зростання кількості автономних роботизованих систем і транспортних засобів стимулює розвиток акустичних сенсорних рішень як складових інтегрованих навігаційних платформ. Метою цього розділу є аналіз фізичних та математичних засад роботи ультразвукових систем, розгляд основних типів реалізації, а також порівняння їх ефективності та обмежень у практичних застосуваннях.

Принцип роботи ультразвукових сенсорів ґрунтується на емісії акустичного сигналу в діапазоні частот понад 20 кГц та фіксації часу, через який цей сигнал повертається у вигляді відбитої хвилі після взаємодії з об’єктом. На основі цього часу затримки розраховується відстань до перешкоди.

Існують два основні підходи до реалізації систем такого типу: активні та пасивні. Активні ультразвукові сенсори використовують один або кілька п’єзоелектричних перетворювачів для генерації та прийому сигналів. Пасивні системи, у свою чергу, лише фіксують наявні звукові хвилі в середовищі без активної генерації імпульсів. Вони застосовуються переважно в акустичному моніторингу, виявленні джерел шуму та біоакустичних дослідженнях.

За конфігурацією поширення хвиль виділяють монопульсні та багатопроменеві системи. У першому випадку зона виявлення обмежена вузьким сектором, що знижує ризик паразитних відбиттів, але обмежує

покриття. У багатопроменевих системах зони покриття формуються послідовною активацією кількох передавачів і приймачів, що забезпечує побудову просторої акустичної карти. Така архітектура актуальна для тривимірного зондування у складних середовищах.

Щоб визначити відстань до об’єкта найпростішим способом, використовують час, за який звуковий сигнал доходить до об’єкта і повертається назад. Оскільки сигнал проходить шлях в обидва боки — туди й назад — враховується тільки половина цього часу. Для розрахунку також потрібно знати швидкість поширення звуку в середовищі. Наприклад, у повітрі за нормальної температури вона приблизно дорівнює 343 метри на секунду.

Якщо для передачі та приймання сигналу використовуються окремі пристрої, час руху сигналу в кожному напрямку можна враховувати окремо.

З метою підвищення точності в складних акустичних умовах застосовуються адаптивні методи фільтрації сигналу, а також розширені моделі обробки — зокрема аналіз фазового зсуву для визначення відносного переміщення або кореляційні методи для виявлення максимальної енергії імпульсу у відбитому сигналі.

У багатопроменевих конфігураціях часто застосовуються масиви перетворювачів, в яких за допомогою методів формування діаграми спрямованості досягається просторове розділення джерел сигналу. Це дозволяє реконструювати розташування декількох об’єктів у просторі одночасно.

Акустичні системи залишаються ефективним і доступним інструментом для визначення відстаней у короткодіапазонних застосуваннях. Їхні головні переваги включають низьку вартість, простоту конструкції, енергоефективність та незалежність від освітлення. Крім того, вони добре функціонують у задимленому або запиленому середовищі, де оптичні сенсори втрачають ефективність.

Однак обмеження також суттєві: вплив температури, вітру та вологості на швидкість звуку, обмежена дальність дії, схильність до інтерференції,

складність точного просторового позиціонування. Висока залежність від якості калібрування також ускладнює їх застосування в динамічних умовах.
Незважаючи на обмеження, ультразвукові системи широко застосовуються в побутових та промислових системах — від паркувальних сенсорів до навігації автономних роботів. Подальший розвиток технологій формування променів та мікроелектромеханічних масивів може значно розширити функціональність акустичних платформ у майбутньому [14].

1.2. Огляд алгоритмів виявлення та трекінгу для реалізації оптичної навігації

Системи оптичного відстеження є ключовим елементом сучасних комп’ютерно-інтегрованих платформ, що забезпечують здатність виявляти, ідентифікувати, супроводжувати й прогнозувати переміщення об’єктів у полі зору в реальному часі. Вони активно застосовуються у сфері автономного транспорту, робототехніки, відеоспостереження, доповненої реальності та інтелектуального аналізу сцен. У межах цього розділу буде розглянуто та проаналізовано найбільш розповсюджені методи виявлення й супроводу об’єктів, зокрема: метод Хаара, детектори ключових точок і дескриптори ознак (SIFT, SURF, ORB), метод глибокого навчання YOLO, алгоритм Калмана та метод Лукаса–Канаде.

Функціональна структура типової системи оптичного відстеження передбачає кілька етапів. На першому — виконується фіксація зображення за допомогою камери або системи камер. Далі зображення нормалізується, обробляється з метою видалення шуму та артефактів, після чого виділяються об’єкти або області інтересу. На основі обраної моделі виконується супровід кожного з об’єктів у часовій послідовності кадрів. У завершальному блоці формується рішення, наприклад — подача керуючого сигналу або сповіщення користувача.

Залежно від підходу до детекції об’єктів, виділяють кілька категорій алгоритмів. Методи на основі попередньої обробки передбачають виконання виявлення об’єктів на кожному кадрі незалежно. У свою чергу, алгоритми супроводу використовують інформацію про зовнішній вигляд об’єкта з попередніх кадрів і здійснюють пошук відповідностей у поточному зображенні. Третій клас — гібридні алгоритми, які об’єднують переваги обох підходів. До таких належать, зокрема, DeepSORT, ByteTrack, CenterTrack тощо.

В основі роботи алгоритмів супроводу лежить принцип прогнозування траєкторії об’єкта. Система відстеження не лише фіксує положення, але й аналізує динаміку руху — швидкість, напрямок, розміри. У випадку тимчасової втрати видимості (наприклад, при частковому перекритті іншими об’єктами), алгоритм повинен оцінити ймовірність того, що повторно виявлений об’єкт є тим самим, що був раніше, або іншим. Саме для таких задач використовуються математичні моделі типу фільтра Калмана або методів асоціації треків.

До типових проблем, з якими стикаються подібні системи, належать: зміни освітлення, часткові оклюзії, велика кількість об’єктів у сцені, шум у зображенні, затримки обробки при високій складності моделей. З метою підвищення надійності, сучасні системи часто доповнюються альтернативними джерелами інформації — лазерним сканером (LiDAR), радаром, даними супутникової навігації або цифровими картами. Завдяки такому мультисенсорному підходу (сенсорне злиття) система здатна сформувати повну просторову картину, забезпечуючи точну орієнтацію й ефективне прийняття рішень у складних умовах навколишнього середовища.

У наступних підрозділах буде детально розглянуто ключові методи реалізації систем оптичного відстеження, що застосовуються в рамках даної роботи: каскадний класифікатор Хаара, детектори та дескриптори ознак, архітектуру YOLO, фільтр Калмана та метод оптичного потоку Лукаса–Канаде[15].

1.2.1 Метод відстеження об’єктів Хаара

Це метод, який базується на використанні простих візуальних шаблонів, відомих як фільтри Хаара. Основна ідея полягає в тому, щоб знаходити контрастні області зображення (наприклад, світла ділянка поруч із темною), які характерні для певних об'єктів — зокрема, рис обличчя: очей, носа, рота.

Ключовий елемент — інтегральне зображення. Воно дозволяє обчислювати суму значень пікселів у прямокутній області зображення всього за кілька операцій. Це радикально прискорює обчислення, дозволяючи використовувати сотні фільтрів у реальному часі. Детальніше про використання методу Хаара можна побачити на рис. 1.5:

Рис. 1.5 Демонстрація каскадного класифікатора Хаара для виявлення обличчя на фото [16]

Для класифікації зображень метод використовує алгоритм AdaBoost, який формує ансамбль слабких класифікаторів на основі простих фільтрів Хаара і об'єднує їх у сильний. Ці класифікатори організовуються у каскад: на кожному етапі приймається рішення, чи може об'єкт бути цільовим. Якщо хоча б один класифікатор на рівні дає негативну відповідь — об'єкт відкидається [16].

1.2.2 Детектори ключових точок і дескриптори ознак SIFT, SURF, ORB

Ці методи працюють з виявленням унікальних локальних ознак на зображенні та створенням їх описів (дескрипторів), які дозволяють зіставити одні й ті ж об'єкти на різних кадрах, незалежно від масштабу, повороту, часткового перекриття або зміни освітлення. Ілюстрацію роботи алгоритмів

SIFT, SURF та ORB які виявляють та зіставляють ключові точки між двома зображеннями наведено на рис.1.6

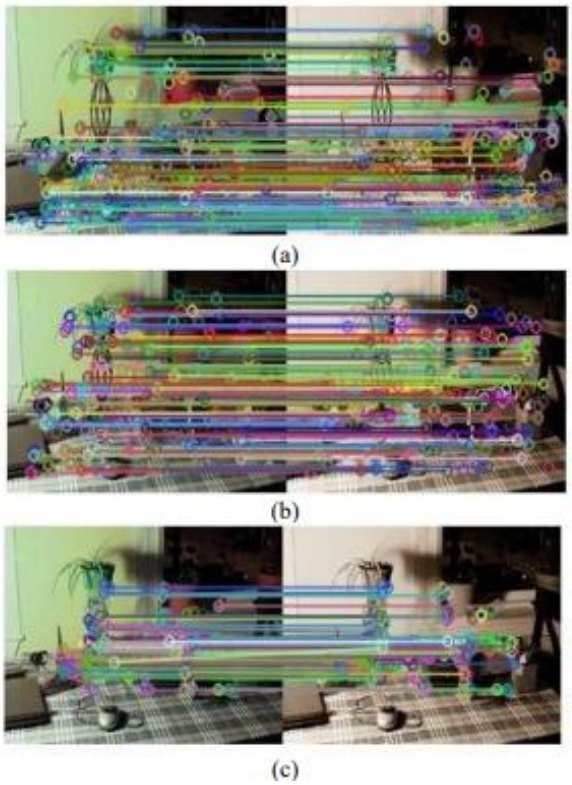

Рис. 1.6 Демонстрація роботи алгоритмів SIFT(a), SURF(b) та ORB(c) [17]

SIFT — це класичний метод, що формує простір масштабів через гауссове згладжування — процес згладження зображення фільтром Гауса, який зменшує шум і дрібні деталі, щоб дозволити виявляти лише суттєві структури, особливо екстремуми в багатомасштабному просторі та обчислення різниці гаусів (DoG) для виявлення екстремумів у просторі зображень. Кожна ключова точка характеризується координатами, масштабом і орієнтацією, після чого формується дескриптор — гістограма градієнтів яскравості у навколишньому регіоні.

SURF — прискорена версія SIFT, що використовує апроксимацію гаусового фільтра з допомогою інтегрального зображення. Ключові точки в SURF виявляються за допомогою гессіанової матриці (детермінант гессіана), що підвищує точність і швидкість.

ORB — це швидкий і відкритий аналог SIFT/SURF. Він поєднує детектор кутів FAST із бінарним дескриптором BRIEF, доповненим обчисленням орієнтації.

Методи SIFT, SURF і ORB відіграють ключову роль у задачах виявлення та зіставлення локальних ознак на зображеннях. Їх застосування дозволяє досягти високої стійкості до масштабних змін, обертання, часткових оклюзій і варіацій освітлення, що є критичним для надійної роботи комп’ютерного зору в реальному середовищі. Кожен із розглянутих алгоритмів має свої переваги: SIFT — високу точність при складних умовах, SURF — пришвидшену обробку при збереженні точності, а ORB — ефективність та відкритість, що робить його придатним для реалізації на обмежених обчислювальних ресурсах. Завдяки цьому, дані методи широко використовуються у завданнях трекінгу, побудови карт, 3D-відновлення, SLAM і робототехніці [17].

1.2.3 Метод YOLO

Суть назви — абсолютно технічна: вона дійсно «дивиться» на зображення лише один раз. Тобто, замість того, щоб спочатку шукати області, де можливо є об’єкти, а потім перевіряти кожну з них окремо — YOLO бере все зображення одразу. Модель обробляє зображення повністю в одному проході, при цьому кожна його частина (комірка сітки) незалежно передбачає параметри потенційного об'єкта.

Уявімо, що зображення поділено на сітку 13×13 — класичний приклад у YOLOv3. Кожна така комірка відповідає за передбачення кількох прямокутних рамок, і для кожної рамки модель повертає наступне: координати центру (x, y), ширину й висоту (w, h), коефіцієнт об'єктності та розподіл імовірностей належності до кожного з класів.

Таким чином, обчислення відбуваються паралельно, і на виході формується багатовимірний тензор, що описує весь вміст сцени. Потім застосовується процедура NMS, яка усуває надмірні або перекривані рамки,

залишаючи найбільш вірогідні детекції. У підсумку отримуємо фінальну інтерпретацію: де знаходиться кожен об’єкт, до якого класу він належить і наскільки впевнена модель у своїй оцінці.

Цей підхід дозволяє досягти балансу між швидкістю і точністю, особливо в задачах реального часу, таких як відеоаналітика або автономна навігація.

Для кожної клітинки модель передбачає:

— координати центру прямокутної рамки (x, y),

— ширину та висоту об’єкта (w, h),

— об’єктність (чи є тут об’єкт, чи лише фон),

— розподіл ймовірностей по всіх класах (наприклад, «це або людина, або велосипед»).

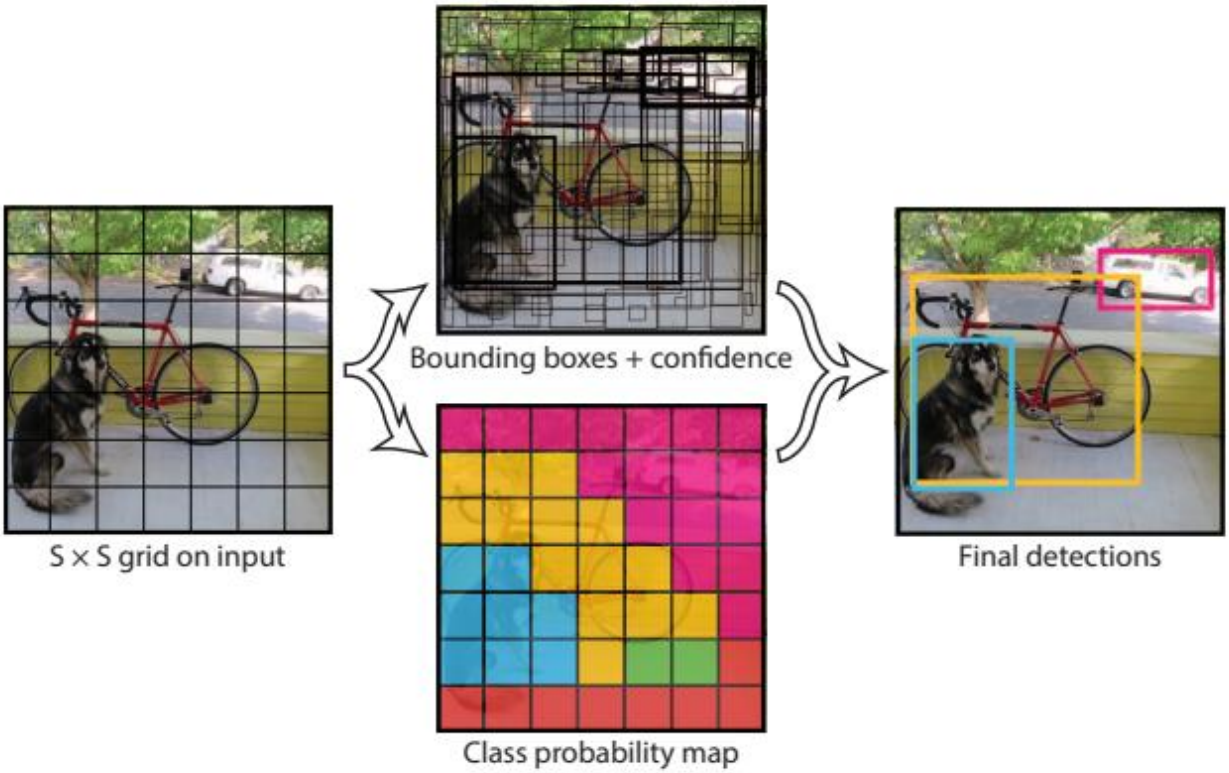

Рис. 1.7 Ілюстрація використання методу YOLO [18]

Всі ці передбачення зливаються в єдиний тензор, з якого потім, за допомогою NMS, видаляють надлишкові рамки. Тобто, якщо дві рамки перекриваються, а ймовірність у однієї вища — друга відсіюється.

Архітектурно YOLO — це згорткова нейромережа. Вона проходить зображення через серії фільтрів, які витягують ознаки: від простих (лінії, краї) до складних (форми, об’єкти). Потім ці ознаки узагальнюються й проходять кілька етапів згортки, аж доки мережа не сформує фінальний блок передбачень. Таким чином, метод YOLO є однією з найефективніших архітектур для детекції об’єктів в умовах реального часу. Його головною перевагою є здатність обробляти зображення за один прохід, що значно підвищує швидкодію системи без суттєвих втрат точності. Завдяки уніфікованому підходу до виявлення, локалізації та класифікації, YOLO забезпечує компактність, масштабованість і простоту інтеграції в мобільні або вбудовані системи. Саме це робить його особливо придатним для задач комп’ютерного зору в автономній робототехніці, розумному відеоспостереженні, доповненій реальності та інших прикладних сферах, де критичними є як швидкість, так і точність прийняття рішень [18].

1.2.4 Калманівський фільтр

Калманівський фільтр — це метод оцінювання стану динамічної системи, що піддається випадковим збуренням, на основі обмеженої та зашумленої інформації з сенсорів. Його математичне ядро ґрунтується на лінійних рівняннях стану та спостереження.

Спочатку виконується прогноз стану — це оцінка, де об’єкт мав би бути зараз, якщо врахувати його попереднє положення та можливий вплив ззовні (наприклад, прискорення чи інші керуючі дії). Також враховується, що в реальному світі завжди є певна похибка або невизначеність — це називають шумом процесу.

Потім відбувається моделювання спостереження — тобто система порівнює очікуваний стан об'єкта з тим, що реально бачить сенсор. Але і тут є похибка — це шум вимірювання, який теж враховується.

Ці два етапи разом дозволяють системі більш точно оцінити реальний стан об'єкта, навіть якщо окремі вимірювання є неточними або неповними.

Такий підхід лежить в основі, наприклад, фільтра Калмана, що часто використовується в системах навігації та трекінгу.

Цей процес ітеративний: кожен новий вимір коригує попередній прогноз. Основна перевага — здатність згладжувати неточності, утримуючи плавну траєкторію об'єкта навіть при втраті сигналу. У реальних умовах це дає можливість відстежувати об'єкт, навіть якщо сенсор тимчасово не фіксує його положення [19].

1.2.5 Метод Лукаса-Канаде

Метод Лукаса-Канаде є одним із найвідоміших і найпоширеніших підходів для визначення руху об'єктів у відео або послідовності зображень.

Його суть полягає в спостереженні за тим, як змінюються положення пікселів на зображенні з часом. Основне припущення цього методу — яскравість кожної точки сцени залишається майже незмінною під час її переміщення в межах невеликого проміжку часу. Інакше кажучи, якщо певна точка на одному кадрі має певну яскравість, то на наступному кадрі ця ж точка (хоча й переміщена) все ще має приблизно ту саму яскравість. Це називається гіпотезою постійності яскравості.

Для того щоб визначити, як саме перемістилася точка, аналізується не один піксель, а невелика ділянка зображення — наприклад, квадрат 5×5 або 7×7 пікселів. Лукас і Канаде припустили, що всі пікселі в цій області рухаються однаково. Таке припущення дозволяє зібрати більше даних і отримати надійніший результат. Адже, якщо дивитися лише на один піксель, недостатньо інформації, щоб однозначно сказати, у якому напрямку і з якою швидкістю він змістився.

Метод використовує інформацію про те, як яскравість змінюється у просторі (тобто по зображенню) та в часі (між кадрами), щоб побудувати математичну модель. Ця модель дозволяє обчислити вектор — стрілку, яка показує напрямок і швидкість руху області зображення між двома кадрами. Такі

вектори оптичного потоку дозволяють побачити, як рухаються об'єкти на відео, навіть якщо вони не мають чітких контурів або їх форма трохи змінюється.

Завдяки своїй простоті та ефективності метод Лукаса-Канаде широко застосовується в різних завданнях комп’ютерного зору: від стеження за рухом транспортних засобів, жестів рук, до автономної навігації роботів і дронів. Його головна перевага — здатність швидко й точно визначати локальний рух навіть у реальному часі [20].

1.3 Висновок до розділу 1

У розділі було проаналізовано сучасні системи відстеження об’єктів, зокрема GPS, RFID, IMU, LiDAR та ультразвукові сенсори. Кожна з них має свої переваги й обмеження: глобальні системи дають координати, але втрачають точність у закритих просторах; локальні сенсори ефективні на малих відстанях; інерціальні модулі забезпечують безперервність, проте накопичують похибки.
Особливу увагу приділено оптичним методам — алгоритмам Хаара, SIFT, YOLO, фільтру Калмана та методу Лукаса-Канаде. Вони дозволяють виявляти, розпізнавати та прогнозувати рух об’єктів у реальному часі. Оптичний підхід виявився найбільш придатним для реалізації недорогої та ефективної системи на мобільній платформі.

РОЗДІЛ 2. МЕТОД ВИЯВЛЕННЯ ЯСКРАВОЇ ТОЧКИ ЗОБРАЖЕННЯ ТА ЙОГО ІНТЕГРАЦІЯ В СИСТЕМУ ВІДСТЕЖЕННЯ.

2.1 Принципи роботи цифрової камери та мозаїка Байера

У цифровій камері об’єктив фокусує світло від сцени на площину світлочутливого сенсора, утворюючи зменшене зображення. Це виконується за принципом тонкої лінзи: відстань до об’єкта, відстань до сенсора та фокусна відстань пов’язані формулою тонкої лінзи [21]:

$$\frac{1}{f} = \frac{1}{d_o} + \frac{1}{d_i} \quad (2.1)$$

де:

f – фокусна відстань,

d_o – відстань до об’єкта,

d_i – відстань від лінзи до сенсора

За правильного фокусування на сенсорі формується чітке зображення об’єкта. Розмір проєкції залежить від фокусної відстані і відстані до об’єкта: приблизно, лінійний розмір проєкції H_i пов’язаний з розміром об’єкта як $H_i \approx \frac{f}{d_o} H_o$(2.2). Об’єктив також визначає кут огляду камери – чим коротша фокусна відстань, тим більший кут огляду.

Сформоване оптикою зображення реєструється світлочутливим сенсором (матрицею) камери. Сьогодні використовуються дві основні технології сенсорів: ПЗС (CCD) та КМОН (CMOS). Обидва типи складаються з масиву мікроскопічних світлочутливих елементів – пікселів, що перетворюють фотони світла на електричний заряд. У CCD-матрицях заряд від кожного фотодіода послідовно зчитується та передається до виходу, тоді як у CMOS-матрицях біля

кожного фотодіода розміщені транзистори, що дозволяють зчитувати сигнал з кожного пікселя індивідуально [22].

КМОН-сенсори зазвичай інтегрують в кожен піксель підсилювач та іншу електроніку, тому сигнал можна паралельно зчитувати з багатьох точок матриці. Над фотодіодами часто розташовані мікролінзи, які концентрують світло точно на фоточутливу область пікселя, підвищуючи ефективність збору світла. На рис. 2.1 показано схематичну будову пікселя активної CMOS-матриці з мікролінзою та кольоровим фільтром:

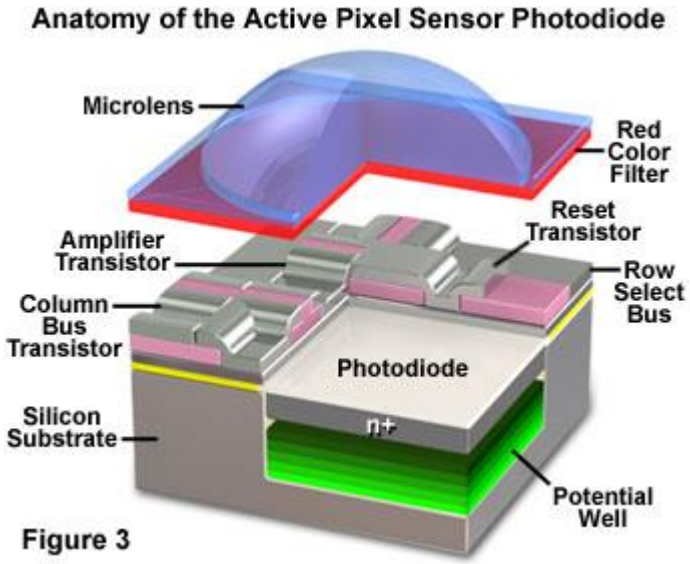

Рис. 2.1 Анатомія активного фотодіода піксельного сенсора [22]

Кожен піксель містить фотодіод, який під дією світла генерує електрони. Фотони, що потрапляють на фотодіод, вибивають з його матеріалу електрони (внутрішній фотоелектричний ефект). Згенеровані електрони збираються в потенціальній ямі (конденсуються на затворі) протягом часу експозиції. Кількість накопиченого заряду пропорційна інтенсивності світла та часу експозиції. Квантова ефективність сенсора визначає, яка частка фотонів перетворюється на електрони. Типове значення квантової ефективності кремнієвих сенсорів видимого діапазону становить десятки відсотків. Наприклад, для ПЗС-сенсора приблизно 2 фотони генерують 1 електрон (ефективність 50%).

Накопичений заряд пікселя потім зчитується і перетворюється на напругу за допомогою підсилювача. Скалювання вибирається так, щоб максимальний корисний заряд (ємність потенціальної ями) відповідав повному діапазону аналогово-цифрового перетворювача. У CCD-матрицях повне заповнення ями становить порядку 5×10^4 електронів, що відповідає вихідній напрузі близько 1 В, тобто кожен електрон спричиняє зміну напруги приблизно на 20μВ [23].

Для отримання правильної експозиції (сигналу, що не перевищує ємність пікселів і перевищує шум) необхідно контролювати час накопичення заряду – час експозиції. У традиційних фотографічних камерах це робить механічний затвор, що відкриває сенсор на визначений час. У сучасних цифрових камерах широко використовується електронний затвор – можливість керувати часом інтеграції сигналу електрично. Наприклад, у ПЗС-матрицях можна скинути заряд з усіх пікселів і через заданий проміжок часу зчитати його – таким чином реалізується функція витримки. CMOS-сенсори також дозволяють вибірково керувати зчитуванням. Типові значення витримки для відеокамер відповідають частоті кадрів: ~1/50 с або 1/60 с для європейських стандартів. Якщо за цей час на сенсор потрапляє недостатньо світла, камера може збільшити експозицію (аж до кількох кадрів) або підсилити сигнал [22].

Світлочутливі сенсори по суті реагують лише на інтенсивність світла, а не на його колір. Фотодіод однаково генерує електрони від фотонів різної довжини хвилі (в межах чутливого спектру кремнію ~400–1000 нм). Тому монохромний сенсор (без фільтрів) дає "чорно-біле" зображення: яскравість без розрізнення кольорів. Для отримання кольорового зображення потрібно розділити падаюче світло на складові за кольорами (довжинами хвиль) і виміряти інтенсивність кожної складової. Один підхід – використати три окремі сенсори для трьох базових кольорів (наприклад, червоного, зеленого, синього). У таких системах оптична призма в об’єктиві розділяє вхідне світло на три пучки, спрямовуючи кожен на свою матрицю: один сенсор реєструє тільки червоне зображення, другий – зелене, третій – синє [22].

Подібна схема реалізована в так званих 3CCD-камерах (або 3CMOS), що історично застосовувалися у професійних відеокамерах. Потім три одноколірні зображення об’єднуються піксель-до-пікселя в одне повноколірне зображення. Перевага трисенсорної схеми – висока кольорова точність і роздільна здатність без інтерполяції, оскільки кожна координата сцени має виміряне значення R, G, B. Однак недоліки – складність і висока ціна: потрібен спеціальний блок призм та фільтрів, три окремі матриці, ідеальне розташування їх між собою, а також об’єктив, здатний покрити три окремі сенсори одночасно. Трисенсорні камери забезпечують чудову якість зображення, проте громіздкі та дорогі [23].

Більш поширеним у споживчих цифрових камерах є одно-сенсорний підхід: використовується одна матриця, але над кожним її пікселем розташовано мікроскопічний кольоровий фільтр, який пропускає світло лише певного діапазону хвиль (кольору). Таким чином, кожен окремий піксель реєструє лише один з трьох основних кольорів (R, G або B) у відповідній точці зображення [22].

Наприклад, якщо піксель накрито зеленим фільтром, він генерує заряд пропорційно інтенсивності лише зеленої компоненти світла; червоні та сині фотони цей фільтр не пропустить. Інший сусідній піксель може мати червоний фільтр і реєструвати тільки червоне світло тощо. Щоб отримати повноколірне зображення з такого сенсора, необхідно знати, який фільтр стоїть над кожним пікселем, і обчислити (інтерполювати) відсутні дві колірні компоненти на основі сусідніх пікселів. Схема розташування кольорових фільтрів по всій матриці називається кольоровою мозаїкою або кольоровим фільтрувальним масивом (Color Filter Array, CFA). Найпоширеніший шаблон CFA – це мозаїка Байера, розроблена в 1970-х роках інженером Брюсом Байєром(Bryce E. Bayer) [22].

Особливість шаблону Байера – нерівномірне співвідношення кольорів: 50% пікселів – із зеленим фільтром, 25% – з червоним, 25% – з синім. На практиці це означає, що в двовимірному масиві фільтри утворюють чергування

рядків: один рядок чергує зелений і червоний (G, R, G, R, ...), наступний – синій і зелений (B, G, B, G, ...) і так далі, як показано на рис. 2.2.

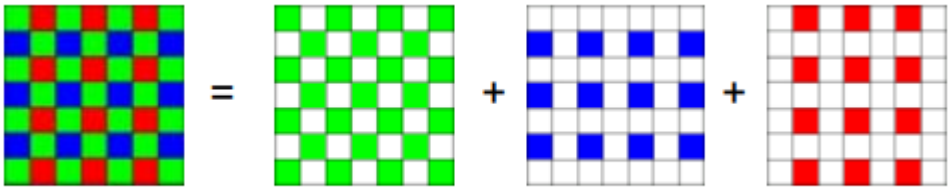

Рис. 2.2 Розподіл кольорів в мозаїці Байера [22]

В мозаїці Байера зелений компонент має подвійну щільність семплювання, оскільки людське око найбільш чутливе саме до зеленого світла. В результаті сенсор отримує більше інформації про яскравість (люмінанс) сцени, яку головним чином визначає зелений канал, покращуючи суб'єктивну різкість зображення. Іншими словами, мозаїка Байера оптимізована під особливості зору людини [22].

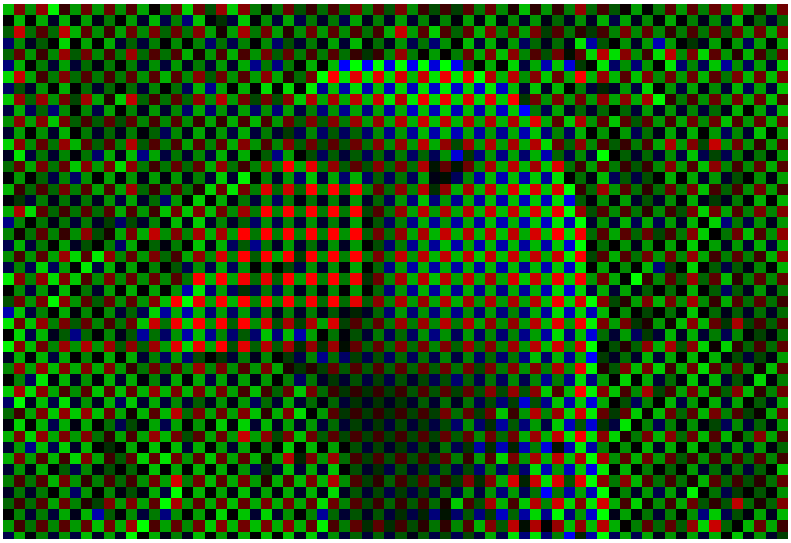

Рис. 2.3 Візуалізація RAW-даних із камери [24]

Як показано на рис. 2.3, одночіповий сенсор з мозаїкою Байера дає на виході неповні дані про зображення: у кожній точці (пікселі) відомо лише інтенсивність одного кольору (того, для якого там стоїть фільтр). Для відновлення повноколірного зображення необхідно обчислити (інтерполювати) відсутні дві колірні компоненти в кожному пікселі на основі даних сусідніх пікселів. Цей процес називається демозаїка або кольорова інтерполяція. У

найпростішому випадку можна використати білінійну інтерполяцію: припустити, що колірні значення змінюються плавно по зображенню, і бракувальне значення можна отримати як середнє з найближчих сусідів того ж кольору [22].

Розглянемо, приміром, піксель під зеленим фільтром (G-піксель). Він має достовірно виміряне значення G (зелений канал). Щоб отримати для нього значення червоного R та синього B, беремо сусідні пікселі з відповідними фільтрами. Для зеленофільтрового пікселя його R можна оцінити за двома найближчими червоними сусідами – наприклад, ліворуч і праворуч від нього, як зображено на рис. 2.4

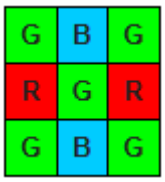

Рис. 2.4 Розподіл сусідніх кольорів для зеленого пікселю [25]

Значення R інтерполюється як середнє цих двох сусідніх R

$$R_{\text{interp}}(i,j) = \frac{R(i,j-1)+R(i,j+1)}{2} \tag{2.3}$$

де (i,j) – координати зеленого пікселя, а $(i,j-1)$ та $(i,j+1)$ – сусідні пікселі з червоним фільтром ліворуч і праворуч від нього.

Аналогічно до формули (2.3), для того ж пікселя обчислюємо синій компонент B – усереднюємо значення найближчих синіх пікселів зверху і знизу:

$$B_{\text{interp}}(i,j) = \frac{R(i,j-1)+R(i,j+1)}{2} \tag{2.4}$$

Таким чином, для кожного зеленого пікселя за допомогою формул(2.3) і (2.4) отримано оцінки R та B.

Для пікселя під червоним фільтром (R-піксель) відомо R, а відсутні G та B беремо з сусідів. Найближчі сусіди червоного пікселя, які несуть зелену компоненту – це безпосередньо прилеглі зверху, знизу, зліва і справа (на мозаїці Байера всі 4 сусіди R-пікселя будуть зеленими), як показано на рис. 2.5

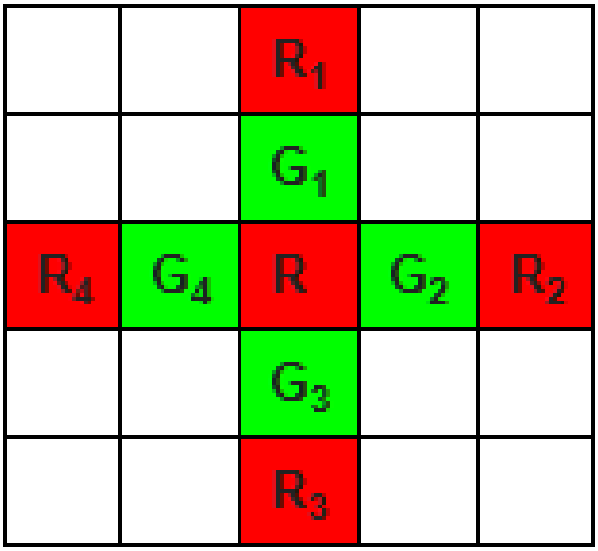

Рис. 2.5 Інтерполяція зеленого кольору для червоного пікселя [25]

Можна взяти середнє цих чотирьох значень G, або ж – для кращої точності – врахувати лише 2 з них по певній осі. Один із підходів – взяти середнє двох зелених пікселів по вертикалі або по горизонталі, залежно від характеру локальної структури зображення. Наприклад, якщо різниця між верхнім і нижнім сусіднім червоним R_1 та R_3 менша, ніж різниця між лівим і правим червоним R_2 та R_4 , то інтенсивність змін у локальному фрагменті скоріш за все орієнтована горизонтально, і для інтерполяції зеленого компонента доцільніше взяти вертикальних сусідів G (тобто усереднити G_1 та G_3). І навпаки, якщо по горизонталі сусіди більш схожі (менший градієнт), беремо горизонтальне середнє (G_2 та G_4). Якщо ж різниця між сусідами в обох напрямках мала (піксель лежить у відносно однорідній області), можна усереднити всіх чотирьох сусідів. Такий адаптивний підхід враховує кореляцію кольорових компонент по напряму і дозволяє зменшити артефакти на краях зображення [25].

Формально адаптивну інтерполяцію зеленого каналу, аналогічно формулам (2.3) і (2.4), на червоному пікселі можна описати такою умовною формулою

$$G_{\text{interp}}(i,j) =$$

$$= \begin{cases} \frac{G(i-1,j)+G(i+1,j)}{2}, \text{якщо } |\,\mathrm{R(i-1,j)-R(i+1,j)}\,| < |\,\mathrm{R(i,j-1)-R(i,j+1)}\,| \\ \frac{G(i,j-1)+G(i,j+1)}{2}, \text{інашке} \end{cases} \quad (2.5)$$

Аналогічні міркування застосовуються для оцінки зеленого на синіх пікселях. Для пікселя під синім фільтром (B-піксель) відомо B; червоний компонент для нього можна отримати, взявши середнє значення з чотирьох найближчих червоних пікселів по діагоналях

$$R_{interp}(i,j) = R(i-1,j-1) + R(i-1,j+1) + R(i+1,j-1) + R(i+1,j+1) \quad (2.6)$$

Своєю чергою, зелений для цього B-пікселя інтерполюють адаптивно за сусідніми зеленими (як описано вище для R-пікселя, тільки замість сусідніх R використовуються сусідні B для вибору напрямку інтерполяції G) [25].

2.2 Технічна характеристика камери ESP32-CAM (OV2640)

Камера ESP32-CAM є інтегрованим модулем, що поєднує функціональність бездротового мікроконтролера (на базі чипа ESP32) та цифрової камери (матриці OV2640). Камера OV2640 використовує CMOS-матрицю, яка складається з мільйонів світлочутливих осередків (пікселів). Кожен осередок реагує на інтенсивність падаючого світла, генеруючи пропорційний електричний заряд.

Оскільки самі фотодіоди не можуть розрізняти кольори, над CMOS-сенсором розташована кольорова мозаїка Байера, яка розташовує фільтри червоного (R), зеленого (G) та синього (B) над пікселями.

Матриця OV2640 — це 1/4-дюймовий CMOS-сенсор, який підтримує зображення з максимальною роздільною здатністю до 1600x1200 пікселів. Основний процес включає такі етапи, як показано на рис 2.6:

Рис. 2.6 Блок схема алгоритму роботи камери

Для керування налаштуваннями камери використовується двопровідна шина SCCB (аналог I²C). Через SCCB передаються команди конфігурації регістрів OV2640: вибір роздільності, формату, частоти кадрів, експозиції тощо. Відеодані з сенсора передаються через паралельний інтерфейс DVP, який складається з 8-бітної шини даних D0–D7 і сигналів синхронізації: тактовий сигнал пікселів PCLK, горизонтальна синхронізація HREF та вертикальна синхронізація VSYNC.

ESP32 отримує зображення саме по цьому інтерфейсу: кожен кадр передається як послідовність байтів пікселів, синхронізована імпульсами VSYNC/HREF. Для роботи сенсора потрібен зовнішній тактовий сигнал (XCLK,

20 МГц), який подається з мікроконтролера (ESP32). Також є вивід керування живленням камери PWDN, який дозволяє вимикати живлення матриці для економії енергії.

На модулі ESP32-CAM виведено ряд контактів (GPIO ESP32), багато з яких зайняті під роботу камери та карти пам’яті. На рис. 2.7 показано схему розташування пінів цього модуля.

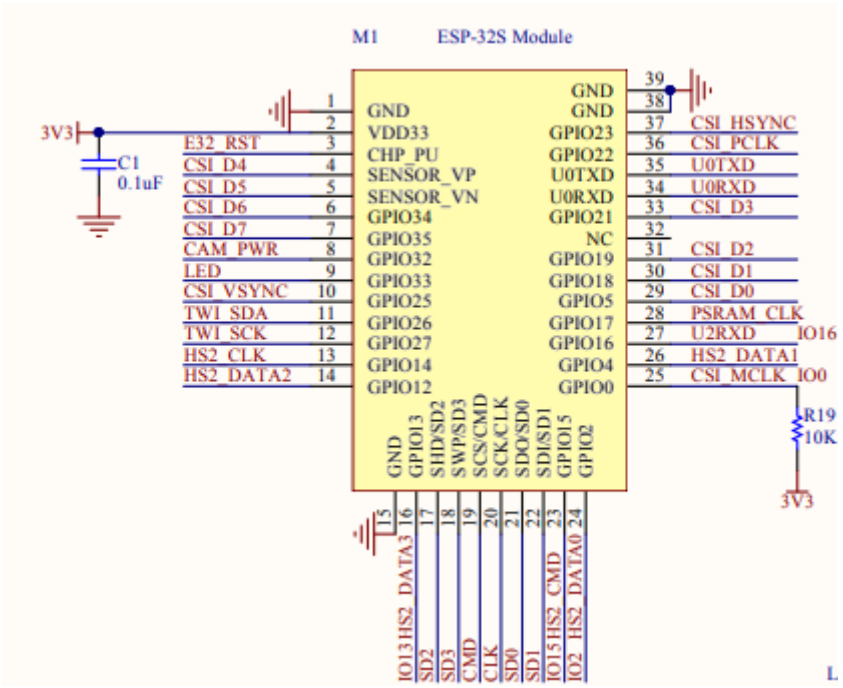

Рис. 2.7 Електрична схема підключення ESP32CAM [26]

Сенсор підключено до ESP32 по виділених лініях. Вісім ліній даних D0–D7 від камери під’єднані до GPIO 5, 18, 19, 21, 36, 39, 34, 35 відповідно. Лінії синхронізації: вхід зовнішнього тактового сигналу XCLK – GPIO 0, вихід такту пікселів PCLK – GPIO 22, вертикальна синхронізація VSYNC – GPIO 25, горизонтальна синхронізація HREF – GPIO 23. Шина керування SCCB використовує SDA на GPIO 26 та SCL на GPIO 27 – через них ESP32 конфігурує регістри OV2640 по протоколу, сумісному з I²C. Контакт PWDN (power down) камери з’єднано з GPIO 32, що дозволяє програмно вимикати живлення матриці. Як згадувалося, контакти U0R і U0T – це UART0 ESP32, відповідно GPIO 3 (RX0) та GPIO 1 (TX0). нормальному режимі вони використовуються для завантаження прошивки та виведення Serial. Після прошивки, за

необхідності, їх можна перепризначити для роботи з периферією, але тоді доступ до стандартного Serial Monitor буде втрачено. Зважайте, що під час старту ESP32 видає службові повідомлення по UART0, тому підключені до GPIO1/3 пристрої можуть отримувати “сміттєві” байти на початку. У більшості випадків ці піни не займаються користувачем в проектах з камерою (окрім як для відлагодження). Вивід GPIO 0 використовується для перемикання режиму завантаження. На модулі він задіяний також апаратно як вхід XCLK для камери, проте під час апаратного ресету стан цього піну визначає режим: якщо притиснутий до GND, запускається завантажувач флеш-пам’яті. В робочому режимі на GPIO0 виводиться 20 МГц сигнал такту для OV2640, тому використовувати його як GPIO не можна. Багато версій ESP32-CAM постачаються з мікросхемою псевдо-статичної RAM (4 МБайт), що підпаяна на платі для розширення пам’яті ESP32. Вона підключена до спеціальних пінів (GPIO16, GPIO17 та ін.) через внутрішній контролер і не має окремих виводів назовні. GPIO16/17 фактично зайняті PSRAM і не можуть використовуватися інакше, якщо PSRAM ввімкнено. Тому хоча фізично модуль має 16 GPIO, реально доступних для користувача лишається дуже мало після врахування камери, SD та PSRAM [26].

Для початку роботи спочатку необхідно налаштувати та запустити модуль камери OV2640. В коді оголошується структура конфігурації, в якій задаються всі параметри роботи камери: номер ШІМ-каналу для генерації XCLK, номер таймера, прив’язки GPIO-пінів до сигналів камери, частота XCLK та формати кадру. Для плати ESP32-CAM ці параметри жорстко визначені і встановлюються відповідно до GPIO, яким підключено лінії D0–D7 сенсора pin_xclk = GPIO0, pin_pclk = GPIO22, pin_vsync = GPIO25, pin_href = GPIO23, pin_sccb_sda = GPIO26, pin_sccb_scl = GPIO27, pin_pwdn=GPIO32, pin_reset=-1. Також вказується:

— Формат пікселів PIXFORMAT_JPEG, тобто камера має віддавати вже стиснене JPEG-зображення.

— Розмір кадру frame_size – за замовчуванням (1600×1200) для максимальної якості, якщо виявлено PSRAM (додаткова зовнішня RAM, використовується для буферизації великих зображень). Інакше можна встановити меншу роздільність (наприклад 800×600) для економії пам’яті.
— Якість JPEG (jpeg_quality) – числом від 0 (краща якість, більший файл) до 63 (гірша якість). У прикладі виставлено ~10–12 для компромісу між якістю та розміром.
— Кількість кадрових буферів (fb_count) – 2 при наявності PSRAM (щоб можна було захоплювати новий кадр, поки попередній відправляється) або 1, якщо пам’яті обмаль.

Після заповнення структури викликається функція, яка здійснює низькорівневу ініціалізацію камери. Вона налаштовує інтерфейси SCCB/DVP, генерує тактовий сигнал XCLK, подає живлення на матрицю (через PWDN) та перевіряє зв’язок із сенсором. У разі успіху камера готова до захоплення зображень, інакше виводиться повідомлення про помилку ініціалізації.

Наступний етап – підключення ESP32 до бездротової мережі Wi-Fi, щоб зробити веб-сервер доступним в локальній мережі. В коді для ESP32CAM оголошено константи ssid і password, які потрібно замінити на назву та пароль вашої мережі Wi-Fi. У функції setup() викликається WiFi.begin(ssid, password), після чого контролер намагається підключитися до точки доступу Після успішного з’єднання модуль отримує локальну IP-адресу від DHCP маршрутизатора.

Передача відео з ESP32-CAM до браузера відбувається за протоколом HTTP у вигляді потоку MJPEG. Модуль OV2640 по команді ESP32 захоплює поточний кадр (у форматі JPEG) та передає його по паралельному інтерфейсу DVP до буфера в оперативній пам’яті ESP32. Після отримання кадру ESP32 формує частину відповіді для клієнта. Якщо це перший кадр у потоці, відправляються HTTP-заголовки: код 200 OK, тип контенту multipart/x-mixed-replace з певним роздільником (boundary). Далі кожен кадр надсилається як

окрема частина цього мультимедійного контенту: перед самим JPEG додається текстовий заголовок з типом Content-Type: image/jpeg і розміром (Content-Length), після чого йдуть байти JPEG-зображення, а за ними – строка --boundary для відділення від наступного кадру. Таким чином, з боку сервера це виглядає як нескінченна відповідь, всередині якої багато вкладених зображень.

Сформовані пакети HTTP відправляються модулем ESP32 через Wi-Fi TCP/IP стек до клієнта. ESP32 виступає TCP-сервером, а браузер – TCP-клієнтом. Мережевий рівень забезпечує доставку пакетів; при гарному сигналі і пропускній здатності Wi-Fi час доставки одного кадру може бути кілька десятків мілісекунд. Веб-браузер, отримавши відповідь з типом multipart/x-mixed-replace, починає інтерпретувати її як потік. При першому JPEG він відображає зображення на сторінці. Коли надходить наступний сегмент (відмічений boundary), браузер замінює попереднє зображення новим без перезавантаження сторінки. Такий механізм дозволяє здійснювати *плавне* оновлення кадру у вікні браузера – користувач бачить відео з мінімальними затримками, хоча насправді це послідовність статичних JPEG. Оновлення відбувається настільки швидко, наскільки дозволяє частота надходження кадрів з камери та мережа (зазвичай 5–10 кадрів/с для 2МП, може бути більше для нижчих роздільностей). Браузер самостійно декодує JPEG кадри засобами вбудованого рендерингу зображень [27]. Повний код для ініціалізації камери наведено в додатку А.

Під час вибору оптичного модуля для реалізації системи відстеження об’єктів були розглянуті й інші популярні камери. Наприклад, Raspberry Pi Camera V2 вирізняється високою роздільною здатністю та чудовою якістю зображення, однак вимагає наявності плати Raspberry Pi та додаткових налаштувань інтерфейсів, що збільшує вартість і складність розробки. Камера OV7670 без FIFO є бюджетним варіантом, однак її використання ускладнене відсутністю буфера кадрів і потребує ручного керування синхронізацією сигналів, що унеможливлює зручну роботу з Python/OpenCV у реальному часі.

Модулі на кшталт OV5640 або Arducam Mini забезпечують вищу якість зображення, але потребують додаткової периферії, програмної підтримки (через ESP-IDF або STM32 HAL), а інколи й потужнішого процесора. Узагальнене порівняння характеристики можна побачити в таблиці 2.1.

Таблиця 2.1 Порівняння камер[28,29,30,31]

Параметр	ESP32-CAM OV2640	Raspberry Pi Camera V2	OV7670(без FIFO)	Arducam Mini
Роздільна здатність	до 1600x1200	8 Мп (3280x2464)	до 640x480	до 2 Мп
Інтерфейси	Wi-Fi, UART	CSI	DCMI	SPI/I2C
Необхідність окремого контролера	Ні	Так (Raspberry Pi)	Так	Так
Простота підключення	Висока (налаштування через Arduino IDE, живлення 5В, інтерфейс UART або Wi-Fi, підтримка WebUI)	Середня (потрібна плата Raspberry Pi та налаштування системи)	Низька (відсутність FIFO, потребує ручного керування синхронізацією)	Середня (потрібна сумісність SPI/I2C та бібліотек)
Безротова передача	Так	Ні	Ні	Можливо
Ціна	~$8	~$25–30	~$3–5	~$10–15

На рис.2.8 представлено приклади модулів камер.

а)

б)

в)

г)

Рис. 2.8 Камери OV2640 ESP32-Cam(а) [27], Raspberry Pi Camera V2(б) [29], Arducam Mini(в) [30], OV7670(г) [32]

Таким чином, ESP32-CAM є оптимальним компромісом між вартістю, якістю зображення, простотою інтеграції та наявністю бездротових комунікацій. Вона дає змогу реалізувати систему з віддаленим моніторингом без необхідності окремого контролера, оскільки ESP32 вже виконує роль процесора.

Причини вибору для роботи. У межах реалізації системи оптичного відстеження на автономній платформі, основними критеріями вибору камери були: підтримка бездротової передачі відео, достатня якість зображення для детектування світлової мітки (лазерної точки), сумісність з Python/OpenCV через IP-потік, компактність та мінімальні енергозатрати. Модуль ESP32-CAM задовольняє всі ці умови. Крім того, широкий досвід спільноти, наявність

прикладів реалізацій, документації та відкритого коду сприяє пришвидшенню розробки системи.

Практичне використання ESP32-CAM дозволяє реалізувати систему виявлення об’єктів у реальному часі, що критично важливо для автономного керування рухомим пристроєм. Додатковою перевагою є вбудована логіка ініціалізації та налаштування мережі, що дозволяє уникнути складних етапів налаштування при кожному ввімкненні системи [28].

2.3 Перетворення зображення у відтінки сірого

Перетворення зображення у відтінки сірого є однією з найпоширеніших і водночас ключових процедур попередньої обробки зображень у комп’ютерному зорі. Це пояснюється тим, що у багатьох випадках завдання обробки не вимагають кольорової інформації, натомість потребують підвищеної чіткості структури або контрасту між об'єктами. Відтак, зменшення кількості кольорових каналів дозволяє спростити обчислення, знизити навантаження на апаратну частину та покращити швидкодію алгоритмів.

У рамках створення мобільної автономної платформи для відстеження яскравої точки (лазерного променя), головним завданням є визначення найбільш контрастного і яскравого пікселя кадру. Для цього зовсім не потрібно мати кольорову інформацію — навпаки, вона лише ускладнює аналіз. Перетворення у відтінки сірого дозволяє отримати одну шкалу яскравості, що значно полегшує пошук локального максимуму, тобто найяскравішої точки кадру. Крім того, зменшення обсягу даних (з трьох каналів до одного) істотно знижує час на обробку кадру, що є критичним параметром у реальному часі, особливо на малопотужних процесорах типу ESP32.

Типове зображення у форматі RGB складається з трьох кольорових каналів: Red (червоний), Green (зелений) та Blue (синій). Кожен піксель зображення представлений як набір з трьох значень (R, G, B), які визначають інтенсивність кожного з каналів у межах від 0 до 255. Таким чином, кожен

піксель вимагає три байти пам’яті. У відтінках сірого піксель представлений лише одним значенням яскравості (інтенсивності), що значно економить ресурси при обробці.

Процес перетворення кольорового зображення в монохромне (градації сірого) виконується шляхом розрахунку середньозваженого значення трьох кольорових каналів. Цей метод є стандартом у більшості комп'ютерних бібліотек, зокрема OpenCV, і реалізується дуже швидко навіть на пристроях з обмеженими обчислювальними ресурсами. Наприклад, у середовищі Python + OpenCV, які в результаті і будуть використовуватися в подальшій розробці дипломної роботи.

Реалізація цього методу відбувається за допомогою команди:

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

Функція cv2.cvtColor() з бібліотеки OpenCV виконує перетворення кольорового зображення з одного колірного простору в інший.

У випадку параметра cv2.COLOR_BGR2GRAY здійснюється перетворення з колірного простору BGR (стандартний формат зображень в OpenCV) у градації сірого (grayscale).

Вхід: трьохканальне зображення frame типу uint8, розміру (H, W, 3), де кожен піксель представлений трійкою значень у порядку (B, G, R), кожне з яких лежить у діапазоні [0, 255].

Вихід: одноканальне зображення типу uint8, розміру (H, W), де кожен піксель представлений єдиним значенням яскравості (інтенсивності світла) [34].

Конверсія в градації сірого виконується шляхом зваженого сумування кольорових каналів з використанням наступної формули:

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \quad (2.7)$$

де:

— Y — яскравість пікселя у градаціях сірого,

— R, G, B — компоненти червоного, зеленого та синього кольору відповідно.

Ці коефіцієнти базуються на стандарті Rec. 601 (Recommendation ITU-R BT.601)[33] — телевізійному стандарті, що описує перетворення RGB у luma-компоненту для відеосигналів. Згідно з фізіологічними особливостями людського зору, найбільше сприймається зелений компонент (коефіцієнт 0.587), далі червоний (0.299), і найменше — синій (0.114).

На практиці, функція cvtColor() виконує операцію для всього зображення як матричне множення, з використанням ядра перетворення (матриці коефіцієнтів) відповідно формулі (2.7):

$$[Y] = [0.114 \;\; 0.587 \;\; 0.229] \begin{bmatrix} B \\ G \\ R \end{bmatrix} \tag{2.8}$$

Або, для всього зображення:

$$I_{gray}(i,j) = 0.299 \cdot \; I_R(i,j) + \; 0.587 \cdot \; I_G(i,j) + 0.114 \cdot I_B(i,j) \tag{2.9}$$

де $I_{gray}(i,j)$ — значення пікселя у сірому зображенні в точці (i,j), а I_R, I_G, I_B — значення каналів RGB відповідного пікселя.

Результат перетворення в монохромне зображення наведено на рис 2.9:

Рис. 2.9 Результат перетворення в монохромне зображення [35]

Використання перетворення зображення у відтінки сірого є логічним, математично обґрунтованим і технологічно ефективним кроком у побудові системи виявлення та відстеження яскравих об'єктів. Такий підхід спрощує реалізацію, підвищує швидкодію та забезпечує необхідну точність [35]. Код реалізації наведено в додатку Б.

2.4 Виявлення найяскравішої точки

Виявлення найяскравішої точки на зображенні є критичним етапом у реалізації системи оптичного відстеження, орієнтованої на ідентифікацію

лазерного променя. Після перетворення вхідного зображення у відтінки сірого, кожен піксель набуває значення яскравості від 0 (чорний) до 255 (білий). Таким чином, найяскравіша точка на зображенні буде мати максимальне значення інтенсивності серед усіх пікселів.

Для виявлення цієї точки у роботі застосовується функція cv2.minMaxLoc(), яка обчислює мінімальні та максимальні значення інтенсивності пікселів у зображенні, а також координати відповідних пікселів.

Відповідно до минулого розділу ця операція визначається наступним чином:

$$\max_{(x,y)} I(x,y) \tag{2.10}$$

де $I(x,y)$ — інтенсивність пікселя з координатами (x,y) у матриці зображення. Функція cv2.minMaxLoc(gray) повертає $max\, I$ і x_{max}, y_{max}. Це значення порівнюється з пороговим значенням, яке у даному випадку вибрано як T = 250, тобто лише пікселі, дуже близькі до білого, розглядаються як потенційні лазерні відблиски. Значення порогу T = 250 було підібрано експериментально на основі тестування системи в різних умовах освітлення. Враховувались наступні фактори:

— Тип лазера — червоний лазер має спектральний пік у діапазоні 630–650 нм, що часто створює яскраву пляму в центрі кадру.

— Освітленість середовища — при денному світлі загальний рівень яскравості зображення зростає, тому надто низький поріг створював хибні спрацьовування.

— Шум сенсора та артефакти компресії — при використанні Wi-Fi-потоку у ESP32-CAM виявляються псевдояскраві точки, які потрібно відфільтрувати.

Обраний метод (cv2.minMaxLoc) забезпечує найкраще співвідношення між простотою реалізації, швидкістю виконання та стабільністю в умовах слабкого освітлення. Для задачі виявлення лазерної точки (яка зазвичай є найяскравішим пікселем на сцені) цей метод є практично оптимальним. Також

важливо, що функція працює безпосередньо з матрицею пікселів без потреби додаткових бінаризацій або згладжувань, що мінімізує затримки — критичний параметр для реального часу в мобільній робототехніці. Приклад відстеження яскравої точки навдено на рис 2.10:

Рис. 2.10 Результат виявлення яскравої точки

Візуальна індикація місцезнаходження точки здійснюється за допомогою функції cv2.circle() та виводу координат — це дозволяє проводити валідацію роботи алгоритму без складних засобів дебагу [36]. Повний код наведено в додатку Б.

2.5 Геометричне позиціонування та стабілізація реакції системи з часовою затримкою

Геометричне позиціонування в системі відстеження об’єктів — це процес визначення просторового положення лазерної точки відносно центра зображення та прийняття рішення щодо дій виконавчих механізмів платформи. У запропонованій роботі реалізовано базовий метод позиціонування, заснований на визначенні горизонтального відхилення найбільш яскравої точки

від центральної вертикальної осі зображення. З технічної точки зору, після виявлення координат (x_{max}, y_{max}) найяскравішого пікселя, виконується порівняння абсциси x_{max} з центральною координатою кадру $x_{center} = width/2$.

Різниця між ними $\Delta x = x_{max} - x_{center}$ є критерієм прийняття рішень щодо обертання або прямолінійного руху платформи:

— Якщо $\Delta x \approx 0$, об'єкт знаходиться по центру — платформа рухається вперед, як показано на рис 2.11;

Рис. 2.11 Виявленні яскравої точки в центрі

— Якщо $\Delta x < 0$, об'єкт лівіше центру — платформа повертає вліво, як показано на рис 2.12;

Рис. 2.12 Виявленні яскравої точки зліва відносно центру

— Якщо $\Delta x > 0$, об'єкт правіше центру — платформа повертає вправо, як показано на рис 2.13.

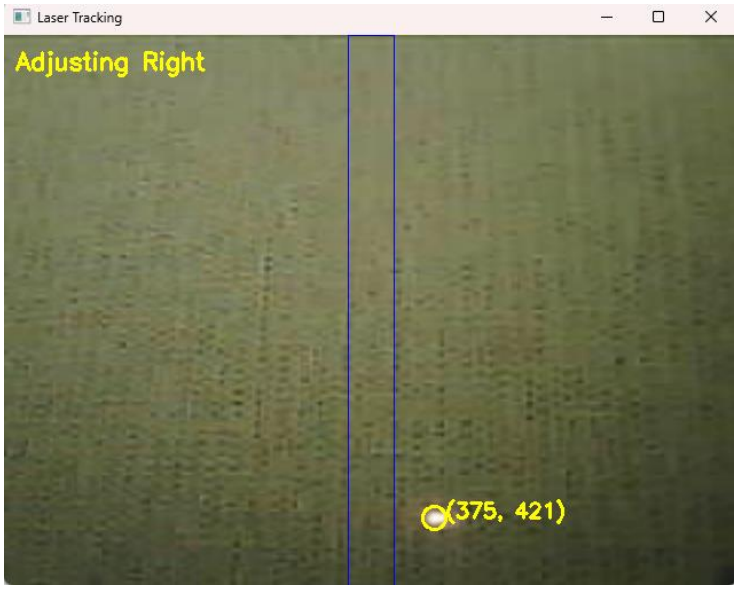

Рис. 2.13 Виявленні яскравої точки справа відносно центру

Для дискретного середовища, в якому кадр має ширину , центр визначається як:

$$x_{center} = \left[\frac{W}{2}\right] \tag{2.11}$$

Далі відповідно до формули (2.11) вводиться величина допустимого відхилення (у пікселях), в межах якої система вважає точку центрованою:

$$|x_{max} - x_{max}| \leq \varepsilon \tag{2.12}$$

Якщо умова не виконується, робот обертається відповідно до знаку відхилення. Наприклад, при , виконується поворот вліво, потім зупинка. Повторна перевірка дозволяє створити цикл самоцентрування.

Введення величини (в коді — tolerance = 20) запобігає надмірно частому переключенню рухів, що могло би виникати через шум чи коливання кадру.

Узагальнено, логіку можна виразити як функцію керування:

$$f(\Delta x) = \begin{cases} \text{поворот ліворуч}, \Delta x < -\varepsilon \\ \text{поворот праворуч}, \Delta x > \varepsilon \\ \text{рух вперед}, |\Delta x| \leq \varepsilon \end{cases} \tag{2.13}$$

Цей принцип дозволяє формалізувати керування платформою у вигляді простого дискретного ПІ-регулятора.

У системах реального часу, особливо в автономних робототехнічних платформах, важливо уникати надмірно чутливої реакції на незначні коливання сенсорних даних. Одним із перевірених способів зменшення коливань є введення часової затримки перед виконанням критичних дій після зміни стану вхідного сигналу — зокрема, після появи або зникнення лазерної точки.

У запропонованій роботі затримка використовується одразу після виявлення найяскравішої точки, яка перевищує встановлений поріг. Реалізована затримка складає 4 секунди та виконує роль фільтра підтвердження сигналу, що дозволяє:

— уникати помилкових спрацьовувань через шум або бліки;

— стабілізувати реакцію системи;

— дати час користувачу точно направити лазер.

Нехай система отримує змінну $s(t)$, яка набуває значення 1 при появі лазерної точки (тобто $max\, I(x,y) \geq T$) і 0 при відсутності. У традиційній системі керування миттєва реакція реалізується як:

$$u(t) = f(s(t)) \tag{2.14}$$

Однак у випадку з часовою затримкою до формули 2.14 вводиться логіка очікування:

$$u(t) = \begin{cases} 0, \text{якщо } s(t) = 1 \text{ і } t\ -\ t_0 < \tau \\ f(s(t)), \text{якщо } s(t) = 1 \text{ і } t\ -\ t_0 \geq \tau \end{cases} \tag{2.15}$$

де:

— t_0 — момент першого виявлення сигналу

— τ — фіксована затримка (у нашому випадку 4 секунди);

— $u(t)$ — керуюча дія (наприклад, рух вперед, поворот, зупинка).

У Python-коді затримка реалізована через модуль time, з використанням змінної stop_start_time, яка фіксує момент появи лазера.

Цей підхід гарантує, що платформа не буде одразу реагувати на одиничне або помилкове виявлення точки, а лише після підтвердження її стійкої присутності. Причини використання часової затримки в цій роботі зумовлена такими причинами:

— Стабілізація системи — зменшується ймовірність осциляцій або "тремтіння" через незначні флуктуації сигналу.
— Зниження впливу шуму — особливо актуально при роботі з компресованим відеопотоком Wi-Fi та JPEG.
— Людино-орієнтований інтерфейс — дозволяє оператору точніше навести лазерну точку, не викликаючи миттєвих неконтрольованих реакцій.
— Сумісність з обмеженими ресурсами — затримка реалізується без використання додаткових фільтрів або складних обчислень.

Використання часової затримки у 4 секунди після фіксації лазерної точки значно покращує стабільність системи, робить її більш надійною в умовах реального середовища та знижує вплив фонових перешкод. Це просте та дієве рішення, яке ідеально підходить для систем з обмеженими обчислювальними ресурсами, таких як ESP32-CAM. Повний код реалізації наведено в додатку Б.

2.6 Висновок до розділу 2

У даному розділі розглянуто принцип роботи цифрової камери, зокрема механізм формування кольорового зображення на основі мозаїки Байера та подальше його перетворення в придатний для обробки формат. Особливу увагу приділено можливостям камери ESP32-CAM на базі сенсора OV2640, що завдяки вбудованому модулю передачі зображень дозволяє використовувати її в компактних роботизованих системах.

Розглянуто процес обробки відеопотоку: від переведення кадру у відтінки сірого до виділення найяскравішої точки, яка використовується як маркер цільового положення. Встановлено, що подібний підхід забезпечує ефективне та швидке виявлення координат лазерного вказівника, з мінімальними обчислювальними затратами.

Завдяки геометричному аналізу положення яскравої точки та введенню часової затримки вдалося підвищити стабільність системи та уникнути

коливань у керуванні. Така обробка стала основою для побудови точного й адаптивного алгоритму оптичного стеження.

РОЗДІЛ 3. СХЕМА КЕРУВАННЯ АВТОНОМНОЮ ПЛАТФОРМОЮ

3.1 Використання плати Arduino для керування виконавчими механізмами

Плата Arduino Uno є мікроконтролерною платформою, заснованою на чіпі ATmega328P, що забезпечує програмоване керування електронними компонентами. У рамках даної роботи її основною функцією є приймання команд з персонального комп’ютера через Bluetooth-модуль (детально розглядається в наступних розділах), обробка отриманих даних та формування сигналів керування для виконавчих механізмів — зокрема, електродвигунів, світлодіодів та інших елементів зворотного зв’язку. Програмне керування здійснюється шляхом генерації цифрових або ШІМ-сигналів (широтно-імпульсна модуляція), які надсилаються на відповідні виводи мікроконтролера. Arduino підтримує 14 цифрових входів/виходів, з яких 6 можуть бути використані як ШІМ-канали, а також має 6 аналогових входів для зчитування аналогових сигналів.

Загальний алгоритм роботи плати Arduino Uno:

Рис. 3.1 Загальний алгоритм роботи плати Arduino Uno

На початку роботи плата Arduino ініціалізує послідовне з'єднання UART через Bluetooth-модуль та переходить у режим очікування вхідних даних. Передача команд здійснюється з комп'ютера, на якому виконується обробка зображень, отриманих у результаті трансляції з камери ESP32-CAM на вбудований веб-сервер. Згідно з результатами аналізу відеопотоку, надсилаються символи-коди керування (наприклад 'L', 'R', 'F' чи 'S'), які інтерпретуються мікроконтролером відповідно до логіки реалізованої прошивки.

Залежно від отриманого символу Arduino формує керуючий сигнал, який надсилається на відповідні цифрові або ШІМ-виводи. Ці сигнали передаються на входи драйвера двигунів DRV8833, що буде описаний у наступному розділі, який у свою чергу активує відповідні двигуни. Результатом є виконання заданої дії — обертання коліс у відповідному напрямку, зупинка або прямолінійний рух. Таким чином, Arduino виступає в ролі інтерпретатора команд в реальному часі, що безпосередньо керує механічною частиною платформи.

Arduino Uno демонструє високу надійність навіть при незначних коливаннях напруги живлення, що є особливо важливим у мобільних застосуваннях, де стабільність джерела живлення не завжди гарантована. Завдяки розвиненій екосистемі бібліотек користувач має змогу підключати периферійні пристрої — як-от драйвери двигунів, Bluetooth-модулі або серводвигуни — без потреби в глибокому розумінні низькорівневого програмування. Це значно скорочує час на розробку та тестування, особливо на етапі прототипування. Наявність великої кількості відкритих проєктів і прикладів дозволяє прискорити процес навчання та пошуку рішень при виникненні типових проблем. Останній вагомий фактор — економічна доступність. Вартість Arduino Uno залишається низькою, що дозволяє реалізовувати функціональні прототипи без значних витрат, а також легко замінювати модуль у випадку фізичного пошкодження або збоїв. Сукупність перелічених чинників робить Arduino Uno найбільш збалансованим рішенням у контексті цієї роботи [37].

3.2 Драйвер двигунів DRV8833: функціонал, підключення, режими

DRV8833 — це двоканальний драйвер двигунів постійного струму на базі H-моста, призначений для ефективного та компактного керування двома двигунами або одним біполярним кроковим двигуном. У межах даної роботи драйвер використовується як виконавчий елемент для керування чотирма мотор-редукторами, згрупованими попарно (ліва та права сторона платформи).

Кожен канал драйвера реалізує повноцінний H-мостовий каскад, що дозволяє керувати напрямом обертання та швидкістю двигуна за допомогою цифрових логічних рівнів та широтно-імпульсної модуляції (ШІМ). Принцип дії полягає у подачі логічних сигналів на відповідні входи IN1–IN2 для одного двигуна та IN3–IN4 для іншого, що зумовлює комутацію внутрішніх MOSFET-транзисторів та утворення відповідної полярності напруги на виводах двигуна.

На рис 3.2 наведено електричну схему підключення моторів до DRV8833

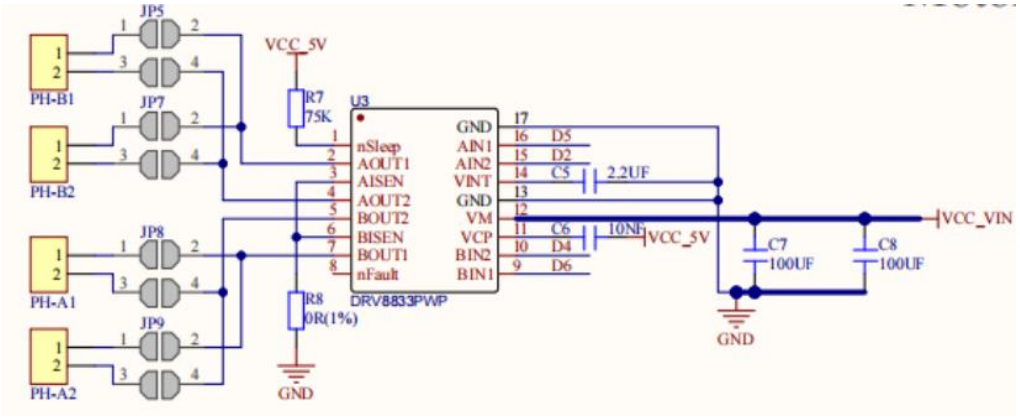

Рис. 3.2 Електрична схема підключення двигунів

На основі електричної схеми видно, що до виводів драйвера DRV8833 підключено два двигуни (по два дроти на кожен). Сигнали керування IN1–IN4 подаються з цифрових пінів D6, D7, D4 та D5 плати Arduino. Конденсатори C5 (2.2 мкФ), C7 та C8 (по 100 мкФ) забезпечують фільтрацію пульсацій живлення, а резистори R7 та R8 забезпечують підтягування та контроль сплячого режиму (nSLEEP). Напруга живлення подається через VCC_5V з плати Arduino або окремого джерела живлення. Важливо зазначити, що виводи AOUT1/AOUT2 і BOUT1/BOUT2 є виходами H-мосту для обох каналів, а BISEN та AISEN можуть бути використані для моніторингу струму через двигуни.

Мікросхема підтримує кілька режимів: прямий рух, реверс, гальмування та вільне обертання (coast mode), які реалізуються комбінаціями сигналів на входах IN1–IN4. Швидкість регулюється шляхом зміни коефіцієнта заповнення ШІМ-сигналу, що дозволяє забезпечити точне керування платформою, особливо при русі на невисоких швидкостях.

У реалізації використовується цифрове керування сигналами DIR та PWM, які подаються з плати Arduino Uno. Наприклад, для руху вперед обидві сторони отримують однаковий напрям сигналу DIR та однакову частоту PWM. Для повороту одна сторона сповільнюється або змінює напрям руху. У системі використовуються стандартні мотор-редуктори. Ці двигуни мають робочу напругу 3–6 В, типовий споживаний струм до 250 мА та передавальне число 1:48. Вони забезпечують адекватну швидкість і крутний момент для

пересування платформи навіть на нерівних поверхнях. Завдяки тому, що максимальний робочий струм двигунів не перевищує допустимі параметри DRV8833 (до 1.2 А на канал, 2 А пікове), сумісність компонентів повністю гарантована.

У межах дослідження було розглянуто декілька альтернатив DRV8833, зокрема драйвери L298N та TB6612FNG. Хоча L298N є поширеним варіантом, його основними недоліками є низький ККД через втрати на транзисторах (падіння напруги до 2 В) та великі розміри. TB6612FNG є більш сучасним аналогом з вищою ефективністю, однак його вартість дещо вища. У порівнянні з ними, DRV8833 забезпечує компактність, низьке тепловиділення, надійність та можливість роботи з низьковольтними двигунами. Це робить його найбільш оптимальним варіантом для мобільних платформ на основі Arduino.

Виконавчі механізми системи — це 4 однакових двигунів постійного струму зображені на рис. 3.3. Ці двигуни є стандартними мотор-редукторами зі зниженим передавальним числом, що забезпечують достатній крутний момент для руху платформи навіть по нерівних поверхнях.

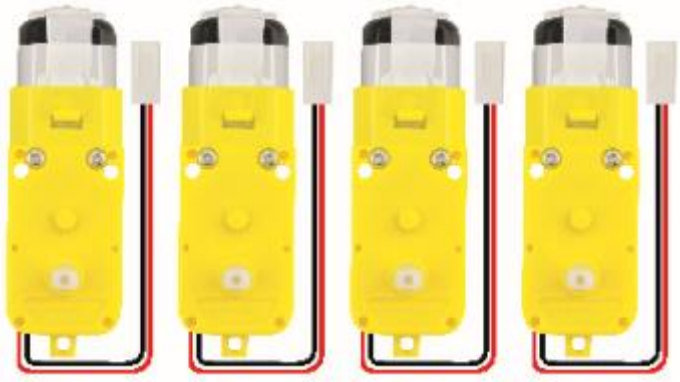

Рис. 3.3 Мотор-редуктор TT

Типові характеристики цих двигунів:

- Робоча напруга: 3–6 В
- Струм споживання: ~120–250 мА (без навантаження)
- Швидкість обертання: до 200 об/хв
- Передавальне число редуктора: 1:48
- Кріплення: стандартне під шасі

Ці мотори добре сумісні з драйвером DRV8833 як по напрузі, так і по допустимому струму.

Така комбінація дозволяє досягти стабільного та контрольованого руху платформи при мінімальному енергоспоживанні[38]. Код для керування двигунами наведено в додатку В.

3.3 Зв’язок між ESP32-CAM та Arduino

У побудові автономних систем з елементами керування важливо забезпечити надійний зв’язок між обчислювальним вузлом, який здійснює аналітику, та мікроконтролером, що відповідає за керування виконавчими механізмами. У запропонованій реалізації обчислення координат та прийняття рішень виконується на комп’ютері, де здійснюється обробка відеопотоку з камери ESP32-CAM, а керуючі команди передаються на Arduino через Bluetooth-модуль HC-06 зобаржений на рис. 3.4, який працює як інтерфейс послідовного зв’язку (Serial over Bluetooth).

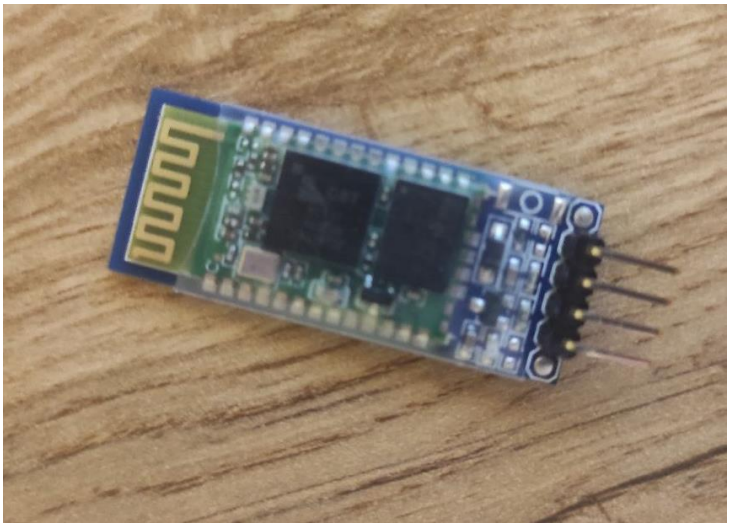

Рис. 3.4 Bluetooth модуль HC-06

HC-06 є популярним Bluetooth-модулем класу 2, який дозволяє реалізувати бездротову передачу даних через інтерфейс UART. Модуль працює на частоті 2.4 ГГц, підтримує стандарт Bluetooth 2.0 + EDR і має простий протокол взаємодії. Він має чотири виводи: VCC, GND, TXD, RXD.

Підключення модуля HC-06 до плати Arduino Uno є одним із найпростіших способів реалізації бездротового зв’язку між мікроконтролером та зовнішніми пристроями, зокрема комп’ютером або смартфоном. Це дає змогу надсилати й приймати дані через стандартний інтерфейс Bluetooth по віртуальному COM-порту.

Живлення модуля здійснюється через пін VCC, який підключається до виводу 5V на Arduino. Попри те, що сам модуль HC-06 працює на логіці 3.3 В, більшість модифікацій модуля вже мають вбудований стабілізатор, що дозволяє подавати 5V без додаткових елементів. Пін GND підключається до загального нульового потенціалу (GND) на Arduino.

Щоб забезпечити правильну передачу даних, вивід TXD модуля HC-06 (який передає дані від Bluetooth) підключається безпосередньо до RX (D0) Arduino. Натомість пін RXD HC-06 (який приймає дані з Arduino) не можна напряму підключати до TX (D1) Arduino, оскільки Arduino використовує логіку 5V, а модуль — 3.3V. Щоб уникнути пошкодження, потрібно використовувати дільник напруги з двох резисторів, наприклад на 1 кОм і 2 кОм, які понижують напругу з 5V до безпечного рівня приблизно 3.3V[39].

Серед причин вибору модуля HC-06 варто відзначити його простоту у використанні, високу сумісність із Arduino Uno, стабільність з'єднання, а також низьку вартість.

З технічного погляду, між ESP32-CAM та Arduino Uno немає безпосереднього дротового з’єднання. Натомість, камера працює як автономний модуль, підключений до локальної Wi-Fi мережі, передає відеопотік через HTTP-протокол, а аналіз зображення (наприклад, виявлення яскравої точки лазеру) відбувається на стороні комп’ютера. Після обробки відео, Python-скрипт формує керуючі сигнали відповідно до положення точки на зображенні. Ці сигнали надсилаються на модуль HC-06, підключений до Arduino.

Важливо зазначити, що модуль HC-06 працює у режимі slave, тобто не ініціює з’єднання, а лише відповідає на запит. Комп’ютер або ноутбук із

вбудованим Bluetooth-модулем створює з’єднання з HC-06, після чого для нього створюється віртуальний COM-порт (наприклад, COM8), з яким працює Python через бібліотеку pyserial[40].

Така структура дозволяє повністю ізолювати сенсорну частину (ESP32-CAM) від керуючої логіки Arduino, що підвищує гнучкість системи. Це також зменшує складність з’єднань, адже не потребує дротового UART-зв’язку між ESP32-CAM та Arduino. Код Python-скрипта, який здійснює з’єднання та керування:

```
import  serial
ser = serial.Serial(’COM8’, 9600)
ser.write(b’F’)
```

В даному випадку надсилається сигнал про початок руху вперед.

З боку Arduino Uno цей сигнал приймається і обробляється в loop():

```
If (Serial.available() > 0){
        Char command = Serial.read();
        processCommand(command);
}
```

3.4 Загальна робота системи та тестування

У цьому розділі представлено комплексний опис інтегрованої системи, що реалізує автоматичне керування транспортною платформою на основі оптичного трекінгу яскравої точки, яку генерує лазер. Система поєднує в собі модуль комп’ютерного зору, обчислювальний модуль обробки зображень, мікроконтролерну платформу керування, виконавчі механізми руху та канали зв’язку. На рис.3.5 представлено цілісну систему у вигляді рухомої платформи з камерою, двигунами, платами Arduino і DRV8833.

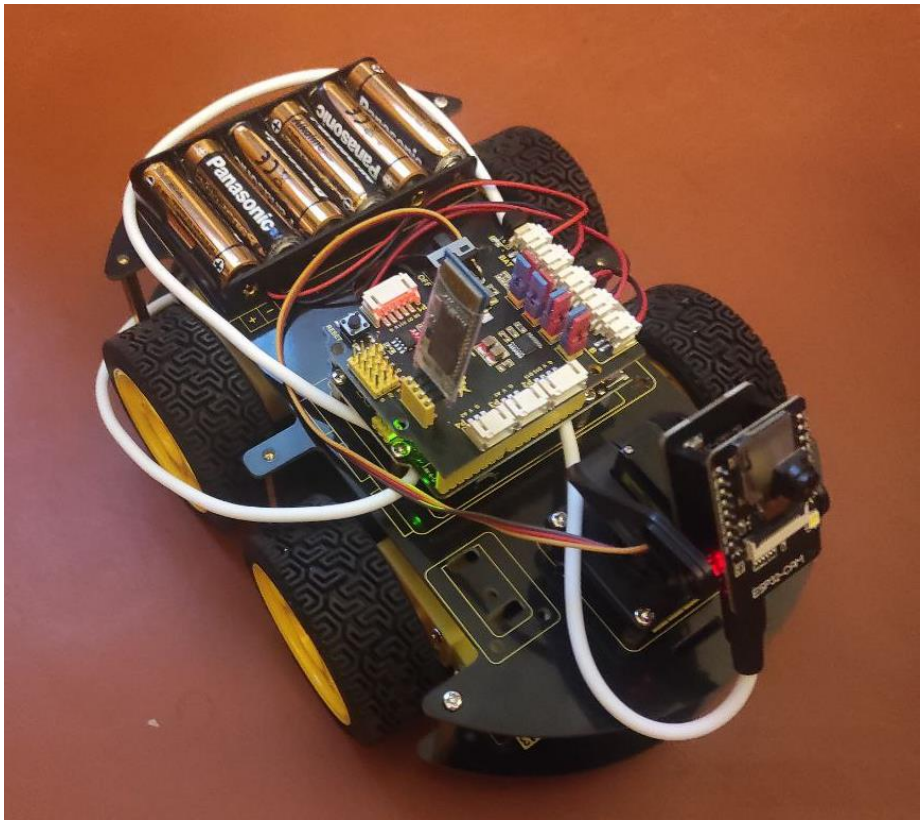

Рис. 3.5 Робоча система автоматичного керування мобільною платформою.

Основу спостереження за простором становить камера ESP32-CAM, що генерує відеопотік у режимі реального часу. Отримане зображення передається на хост-комп’ютер по Wi-Fi з використанням вбудованого веб-сервера, на якому локально виконується аналіз поточного кадру. Обробка кадрів виконується в середовищі Python з використанням бібліотек OpenCV, де відбувається перетворення зображення у відтінки сірого, фільтрація, бінаризація та пошук найяскравішої точки. Згідно з її положенням у кадрі обчислюється напрямок руху, який переводиться у відповідну команду.

Формована команда передається через послідовне з’єднання (UART) за допомогою Bluetooth-модуля HC-06, підключеного до плати Arduino UNO. Мікроконтролер отримує коди команд (наприклад, 'F', 'L', 'R', 'S') та здійснює логічне керування драйвером двигунів DRV8833, що забезпечує рух мобільної платформи у заданому напрямку.

Система реалізує зворотний зв’язок за допомогою безперервного аналізу вхідного відеопотоку. Завдяки наявності часової затримки після виявлення та інтерпретації положення лазерної точки забезпечується стабілізація та усунення коливань у русі платформи.

Сигнальний трафік між модулями є мінімальним, а обчислювальні ресурси розподілені між периферійною обробкою (камера) та центральною (ПК). Така архітектура дозволяє забезпечити адаптивне керування в реальному часі з використанням недорогих компонентів.

Система була зібрана на базі шасі з чотирма колесами (дві пари моторів), що керуються по каналах A та B драйвера DRV8833. Відповідне програмне забезпечення розроблено з урахуванням асинхронної обробки сигналів та таймерів для керування ШІМ. На рис. 3.6 зображено відповідну функціональну схему роботи системи:

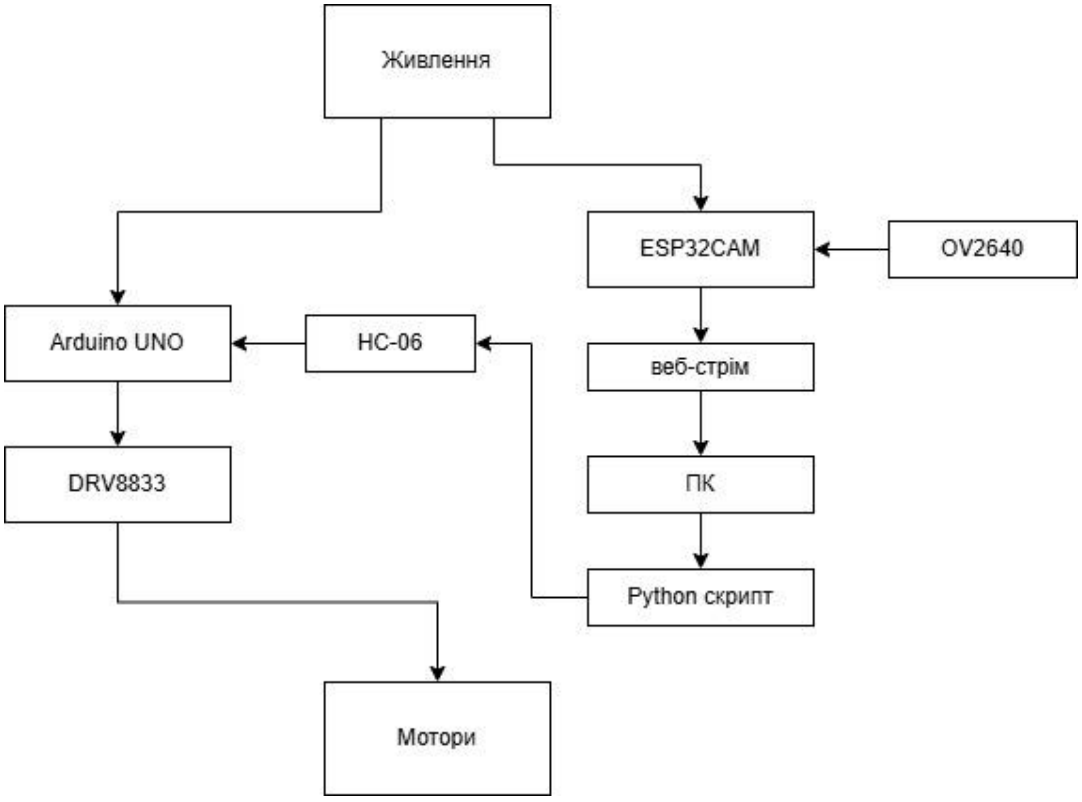

Рис. 3.6 Схема роботи системи автоматичного керування мобільною платформою

Детальний опис:

1. Живлення подається на обидві гілки системи:

— до ESP32-CAM, яка використовує камеру OV2640 для зйомки,
— та до Arduino UNO, яка відповідає за керування виконавчими механізмами.

2. ESP32-CAM формує веб-стрім (відеопотік), який передається на ПК через Wi-Fi.
3. ПК приймає цей потік і передає його на обробку в Python-скрипт, де виконується аналіз зображення (наприклад, виявлення лазерної точки).
4. Згідно з результатом обробки Python-скрипт надсилає керуючі команди через Bluetooth до модуля HC-06, що підключений до Arduino UNO.
5. Arduino UNO приймає ці команди по UART і формує відповідні сигнали керування для драйвера DRV8833.
6. DRV8833 живить і керує двигунами, які відповідають за рух платформи.

Швидкодія розпізнавання точки в більшості залежить від розширення в якому працює камера OV2640. На рис. 3.7 наведено графік залежності затримки (мс) від роздільної здатності (пікселі) камери.

Рис. 3.7 Графік залежності затримки від роздільної здатності камери

З графіка видно, що:

— За низьких роздільних здатностей (160×120, 320×240) система працює з мінімальною затримкою, що критично важливо для режимів реального часу, наприклад, слідування за лазерною точкою.

— Починаючи з 640×480, зростання часу реакції стає майже експоненційним — це пов’язано з лінійним збільшенням кількості пікселів у кадрі, яке викликає більше обчислень у кожному циклі сканування (перетворення в сіре, пошук яскравої точки, оцінка координат).

— При 1024×768 час реакції сягає майже 0.4 секунди, що робить систему непридатною для завдань із високими вимогами до оперативності реагування.

Для оптимізації камери також можна використовувати деякі з вбудованих параметрів камери OV2640.

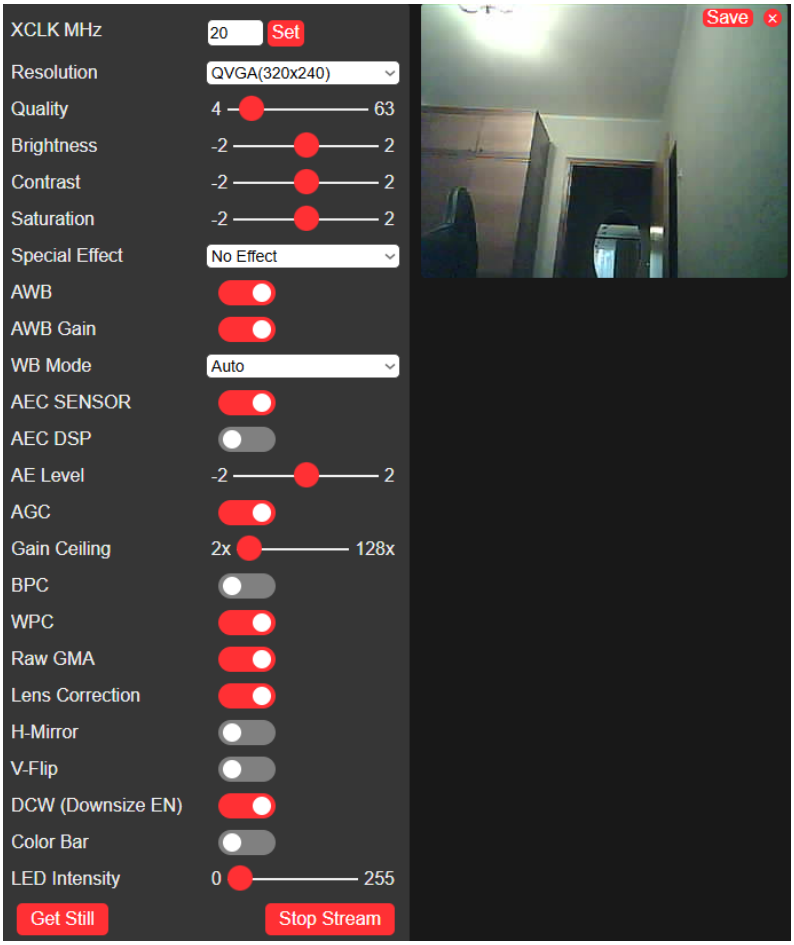

Рис 3.8 Скріншот веб-інтерфейсу OV2640 (stream)

Нижче пояснено вибір ключових параметрів, що оптимізують продуктивність:

- Resolution (роздільна здатність): QVGA (320x240)
  Це оптимальне значення, що забезпечує чіткість для виявлення яскравої точки та достатню швидкодію (~10–15 fps). Більша роздільна здатність (наприклад, VGA, SVGA) значно знижує швидкість потоку.
- Quality: 4. Чим нижча якість, тим менший розмір кадру JPEG, а отже, швидша обробка. Значення 4 — компроміс між швидкістю та візуальною читабельністю.
- Brightness / Contrast / Saturation: -2. Зменшені значення допомагають зменшити шуми фону та покращити контрастність яскравої точки.
- Special Effect: No Effect. Жодних ефектів не застосовується, оскільки обробка відбувається алгоритмічно після зчитування.
- AWB (Auto White Balance): вимкнено, WB Mode: Auto. Для стабільності колірної температури AWB вимикається, що особливо важливо при роботі з лазерною точкою червоного кольору.
- AEC SENSOR: увімкнено, AEC DSP: вимкнено, AE Level: -2. Сенсор автоматично підлаштовує експозицію. Рівень AE вручну зміщено в бік темнішого, щоб зменшити засвічення.
- AGC (Automatic Gain Control): увімкнено, Gain Ceiling: 2x. Автоматичне підсилення допомагає в умовах недостатнього освітлення, але обмежується до 2x, щоб уникнути зайвих шумів.
- Lens Correction: увімкнено. Корекція спотворень об’єктива дозволяє точно локалізувати координати точки, особливо в краях кадру.
- Raw GMA, DCW, Color Bar: вимкнено. Ці параметри не використовуються. Включення їх може впливати на продуктивність або нести зайве навантаження на процесор.
- LED Intensity: 0 .Вбудований світлодіод вимикається, щоб уникнути небажаних засвічень на об’єкті під час аналізу.

3.5 Висновок до розділу 3

У третьому розділі було реалізовано і проаналізовано архітектуру системи керування автономною мобільною платформою, засновану на поєднанні кількох ключових апаратних компонентів. Центральну роль у цій системі виконує плата Arduino Uno, яка забезпечує інтерпретацію команд та безпосереднє керування виконавчими механізмами — двигунами. Для реалізації ефективного керування обертами було використано драйвер двигунів DRV8833, що забезпечив незалежне керування двома каналами з підтримкою ШІМ-модуляції та захистами від перевантажень.

Особливу увагу було приділено організації зв’язку між блоком обробки зображень (ESP32-CAM) та блоком керування рухом (Arduino). Для цього був використаний модуль Bluetooth HC-06, який у парі з COM-портом комп’ютера забезпечив надійний бездротовий обмін даними через стандартний послідовний інтерфейс UART. У рамках підрозділу було детально описано механізм передавання команд, синхронізацію та часову логіку взаємодії, що дозволяє платформі швидко реагувати на об’єкти у полі зору камери.

Загальна схема роботи системи була змодельована, протестована та підтвердила свою працездатність у реальних умовах. Була побудована функціональна схема з урахуванням енергоживлення, логіки взаємодії між компонентами та програмної обробки сигналів. Під час експериментів система демонструвала адекватну поведінку — своєчасну зупинку, повороти та рух відповідно до координат виявленої лазерної точки.

Таким чином, у межах даного розділу вдалося сформувати працездатну структуру автономної системи на базі недорогих мікроконтролерів і двигунів, що повністю задовольняє вимоги до задачі мобільного слідкування за об’єктом. Отримані результати закладають основу для подальшого вдосконалення

системи, зокрема шляхом розширення функціональності, інтеграції додаткових датчиків та покращення алгоритмів керування.

ВИСНОВОК

У результаті виконання дипломної роботи було розроблено та реалізовано функціональну систему оптичного відстеження об’єктів для керування автономною мобільною платформою. Система поєднує в собі можливості комп’ютерного зору, бездротової передачі даних, а також модульного керування виконавчими механізмами з використанням недорогих мікроконтролерів.

На основі проведеного аналізу сучасних методів відстеження об’єктів було обґрунтовано вибір підходу, що базується на виявленні найяскравішої точки в зображенні, отриманому з камери ESP32-CAM. Детально досліджено процеси формування зображення, зокрема демозаїкацію матриці Байєра, перетворення у відтінки сірого, та алгоритмічні принципи локалізації світлових об’єктів. Застосування математичного аналізу координат дозволило реалізувати геометричне позиціонування об’єкта відносно центра кадру та здійснювати динамічне керування рухом платформи.

Особливу увагу приділено апаратній реалізації. Було розглянуто підключення та конфігурацію ключових елементів — камери ESP32-CAM з сенсором OV2640, плати Arduino Uno, драйвера двигунів DRV8833 та Bluetooth-модуля HC-06. Усі компоненти інтегровані в єдину систему, що забезпечує передачу відео на ПК, обробку зображення в реальному часі за допомогою Python-скриптів, і подальше дистанційне керування платформою через бездротовий канал зв’язку.

У результаті тестування система продемонструвала стабільну роботу та здатність точно реагувати на зміну положення лазерної мітки, що свідчить про досягнення поставленої мети. Важливим аспектом є також універсальність та масштабованість створеної архітектури, що дозволяє надалі розширювати функціональність системи, наприклад, шляхом використання складніших алгоритмів трекінгу або впровадження зворотного зв’язку з сенсорів.

Таким чином, розроблена система є прикладом ефективного застосування доступних апаратних та програмних рішень для створення інтелектуального автоматизованого керування транспортним засобом із використанням методів комп’ютерного зору.

СПИСОК ДЖЕРЕЛ

1. MarketsandMarkets. Augmented Reality Navigation Market [Електронний ресурс]. – MarketsandMarkets. – Режим доступу: https://www.marketsandmarkets.com/Market-Reports/ar-navigation-market-150009689.html (дата звернення: 27.05.2025).
2. Viso.ai. Object Tracking in Computer Vision [Електронний ресурс]. – Viso.ai. – Режим доступу: https://viso.ai/deep-learning/object-tracking/ (дата звернення: 09.06.2025).
3. Geonadir. RTK Explained [Електронний ресурс]. – Geonadir. – Режим доступу: https://geonadir.com/rtk-explained/ (дата звернення: 09.06.2025).
4. Marvelmind Robotics. Indoor Positioning System Terminology [Електронний ресурс]. – Marvelmind. – Режим доступу: https://marvelmind.com/solution/indoor_positioning_system_terminology/ (дата звернення: 09.06.2025).
5. GPS World. Inertial Labs Launches New Generation of INS & IMUs [Електронний ресурс]. – GPS World. – Режим доступу: https://www.gpsworld.com/inertial-labs-launches-new-generation-of-ins-imus/ (дата звернення: 09.06.2025).
6. Verified Market Reports. Indoor Positioning Systems and Navigation Market [Електронний ресурс]. – Verified Market Reports. – Режим доступу: https://www.verifiedmarketreports.com/product/indoor-positioning-systems-and-navigation-market/ (дата звернення: 09.06.2025).
7. Kijewski T. GPS triangulation concept [Електронний ресурс]. ResearchGate. – Режим доступу: https://www.researchgate.net/figure/GPS-triangulation-concept-Kijewski-et-al-2006_fig3_330566199 (дата звернення: 09.06.2025).

8. Amobbs. Technical Manual on IMU Principles [Електронний ресурс]. – Amobbs. – Режим доступу: https://d1.amobbs.com/bbs_upload782111/files_33/ourdev_584835O21W59.pdf (дата звернення: 09.06.2025).
9. GeeksforGeeks. Introduction of Radio Frequency Identification (RFID) [Електронний ресурс]. – GeeksforGeeks. – Режим доступу: https://www.geeksforgeeks.org/introduction-of-radio-frequency-identification-rfid/ (дата звернення: 09.06.2025).
10. Inertial Labs. IMU – Inertial Measurement Units [Електронний ресурс]. – Inertial Labs. – Режим доступу: https://inertiallabs.com/products/imu-inertial-measurement-units/ (дата звернення: 09.06.2025).
11. Borowska M. et al. Kalman Filter-Based Localization of Mobile Robot Using UWB Technology // Sensors [Електронний ресурс]. – 2024. – Vol. 24, Issue 22. – Article №7140. – Режим доступу: https://www.mdpi.com/1424-8220/24/22/7140 (дата звернення: 09.06.2025).
12. Seltok Photonics. Загальний огляд LiDAR для безпілотних автомобілів [Електронний ресурс]. – Seltokphotonics.com. – Режим доступу: https://seltokphotonics.com/info/articles/zagal%60nyy_oglyad_lidar_dlya_bezpilotnykh_avtomobiliv/ (дата звернення: 09.06.2025).
13. de Lima J.P. et al. Detection of Humans in Outdoor Environments Using Thermal and Visual Images // Applied Sciences [Електронний ресурс]. – 2019. – Vol. 9, Issue 19. – Article №4093. – Режим доступу: https://www.mdpi.com/2076-3417/9/19/4093 (дата звернення: 09.06.2025).
14. Alahi A. et al. Tracking the Untrackable: Learning to Track Multiple Cues with Long-Term Dependencies // Sensors [Електронний ресурс]. – 2013. – Vol. 13, Issue 3. – P. 3501–3526. – Режим доступу: https://www.mdpi.com/1424-8220/13/3/3501 (дата звернення: 09.06.2025).
15. Xu Y. et al. Real-Time Pedestrian Detection and Tracking Using Fusion of LiDAR and Camera // Sensors [Електронний ресурс]. – 2020. – Vol. 20, Issue

20. – Article №5772. – Режим доступу: https://www.mdpi.com/1424-8220/20/20/5772 (дата звернення: 09.06.2025).

16. Viola P., Jones M. Rapid Object Detection using a Boosted Cascade of Simple Features [Електронний ресурс] // Carnegie Mellon University. – Режим доступу: https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf (дата звернення: 09.06.2025).
17. Bewley A. et al. Simple Online and Realtime Tracking with a Deep Association Metric [Електронний ресурс] // arXiv. – 2017. – Режим доступу: https://arxiv.org/pdf/1710.02726 (дата звернення: 09.06.2025).
18. Redmon J., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection // CVPR 2016 [Електронний ресурс]. – Режим доступу: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf (дата звернення: 09.06.2025).
19. Burgess Z. How a Kalman Filter Works in Pictures [Електронний ресурс]. – BZARG. – Режим доступу: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/ (дата звернення: 09.06.2025).
20. University of Washington. Motion and Tracking [Електронний ресурс] // CSE 455 Course Slides. – 2025. – Режим доступу: https://courses.cs.washington.edu/courses/cse455/25wi/slides/17_motion_and_tracking.pdf (дата звернення: 09.06.2025).
21. Hecht E. Optics. – 4th ed. – New York : Addison Wesley, 2002. – 800 p
22. Freie Universität Berlin. Computer Vision – Lecture 2 [Електронний ресурс]. – Режим доступу: https://www.inf.fu-berlin.de/lehre/WS02/robotik/Vorlesungen/Vorlesung2/ComputerVision-2.pdf (дата звернення: 09.06.2025).

23. TRTS1004. CCD 센서의 특징 및 구조 분석 [Електронний ресурс]. – Tistory.com. – Режим доступу: https://trts1004.tistory.com/12109537 (дата звернення: 09.06.2025).

24. UkrBezpeka. Матриці у відеокамерах: типи та відмінності [Електронний ресурс]. – Ukrbezpeka.com. – Режим доступу: https://ukrbezpeka.com/matricy-v-kamerah-videonabljudenija/ (дата звернення: 09.06.2025).

25. TRTS1004. Bayer Pattern의 이해 [Електронний ресурс]. – Tistory.com. – Режим доступу: https://trts1004.tistory.com/12109232 (дата звернення: 09.06.2025).

26. Random Nerd Tutorials. ESP32-CAM AI-Thinker Pinout Guide [Електронний ресурс]. – Режим доступу: https://randomnerdtutorials.com/esp32-cam-ai-thinker-pinout/ (дата звернення: 09.06.2025).

27. Alumina PCB. OV2640: The Ultimate Guide on the Image Sensor [Електронний ресурс]. – Режим доступу: https://aluminapcb.com/ov2640-the-ultimate-guide-on-the-image-sensor/ (дата звернення: 09.06.2025).

28. Omnivision. OV2640 CMOS Image Sensor Datasheet [Електронний ресурс]. – UCTRONICS. – Режим доступу: https://www.uctronics.com/download/cam_module/OV2640DS.pdf (дата звернення: 09.06.2025).

29. Raspberry Pi Foundation. Camera Module Documentation [Електронний ресурс]. – Режим доступу: https://www.raspberrypi.com/documentation/accessories/camera.html (дата звернення: 09.06.2025).

30. Omnivision. OV7670 Camera Module Datasheet [Електронний ресурс]. – MIT. – Режим доступу:

https://web.mit.edu/6.111/www/f2016/tools/OV7670_2006.pdf (дата звернення: 09.06.2025).

31. Arducam. Arducam Shield Mini 2MP Plus – JPEG Compression Overview [Електронний ресурс]. – Docs.arducam.com. – Режим доступу: https://docs.arducam.com/Arduino-SPI-camera/Legacy-SPI-camera/Hardware/Arducam-Shield-Mini-2MP-Plus/#54-jpeg-compression (дата звренення: 09.06.2025).
32. DUAITEK. Módulo OV7670 Cámara Arduino Salida 18 Pin [Електронний ресурс]. – Режим доступу: https://www.duaitek.com.ar/modulo-ov7670-camara-arduino-salida-18-pin/p/MLA32487286 (дата звернення: 09.06.2025).
33. ITU. Recommendation ITU-R BT.601-7: Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios [Електронний ресурс]. – International Telecommunication Union. – Режим доступу: https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.601-7-201103-I!!PDF-E.pdf (дата звернення: 09.06.2025).
34. LearnOpenCV. Color Spaces in OpenCV (C++ / Python) [Електронний ресурс]. – Режим доступу: https://learnopencv.com/color-spaces-in-opencv-cpp-python/ (дата звернення: 09.06.2025).
35. Gonzalez R., Woods R. Digital Image Processing. 4th Edition – New York: Pearson, 2018. – 1168 p. (Електронна версія: https://www.cl72.org/090imagePLib/books/Gonzales,Woods-Digital.Image.Processing.4th.Edition.pdf, дата звернення: 09.06.2025).
36. LearnOpenCV. Color Spaces in OpenCV (C++ / Python) [Електронний ресурс]. – Режим доступу: https://learnopencv.com/color-spaces-in-opencv-cpp-python/ (дата звернення: 09.06.2025).
37. Arduino Ukraine. Arduino Uno – технічна документація [Електронний ресурс]. – Режим доступу: https://doc.arduino.ua/ru/hardware/Uno (дата звернення: 09.06.2025).

38. Keyestudio. KS0559F – 4WD BT Multi-purpose Car V2.0 Arduino Code [Електронний ресурс]. – GitHub. – Режим доступу: https://github.com/keyestudio/KS0559-KS0559F-Keyestudio-4WD-BT-Multi-purpose-Car-V2.0-Arduino/blob/master/Arduino/arduino.md (дата звернення: 09.06.2025).
39. ETC. HC-06 Bluetooth Module Datasheet [Електронний ресурс]. – Alldatasheet.com. – Режим доступу: https://www.alldatasheet.com/datasheet-pdf/view/1179032/ETC1/HC-06.html (дата звернення: 09.06.2025).
40. PySerial. Python Serial Port Extension Documentation [Електронний ресурс]. – ReadTheDocs.io. – Режим доступу: https://pyserial.readthedocs.io/en/latest/ (дата звернення: 09.06.2025).

Додаток А

Програмний код для ініціалізації відеопотоку з камери

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

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

#define PART_BOUNDARY "123456789000000000000987654321"
#define CAMERA_MODEL_AI_THINKER

#if defined(CAMERA_MODEL_WROVER_KIT)
  #define PWDN_GPIO_NUM    -1
  #define RESET_GPIO_NUM   -1
  #define XCLK_GPIO_NUM    21
  #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      19
  #define Y4_GPIO_NUM      18
  #define Y3_GPIO_NUM       5
  #define Y2_GPIO_NUM       4
  #define VSYNC_GPIO_NUM   25
  #define HREF_GPIO_NUM    23
  #define PCLK_GPIO_NUM    22

#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     25
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       32
  #define VSYNC_GPIO_NUM    22
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21
```

```
#elif defined(CAMERA_MODEL_M5STACK_WITHOUT_PSRAM)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     25
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       17
  #define VSYNC_GPIO_NUM    22
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_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
#else
  #error "Camera model not selected"
#endif

static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary=" PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n";

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, _STREAM_CONTENT_TYPE);
if(res != ESP_OK){
  return res;
}

while(true){
  fb = esp_camera_fb_get();
  if (!fb) {
    Serial.println("Camera capture failed");
    res = ESP_FAIL;
  } else {
    if(fb->width > 400){
      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){
          Serial.println("JPEG compression failed");
          res = ESP_FAIL;
        }
      } else {
        _jpg_buf_len = fb->len;
        _jpg_buf = fb->buf;
      }
    }
  }
  if(res == ESP_OK){
    size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
```

```
      res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY,
strlen(_STREAM_BOUNDARY));
    }
    if(fb){
      esp_camera_fb_return(fb);
      fb = NULL;
      _jpg_buf = NULL;
    } else if(_jpg_buf){
      free(_jpg_buf);
      _jpg_buf = NULL;
    }
    if(res != ESP_OK){
      break;
    }
    //Serial.printf("MJPG: %uB\n",(uint32_t)(_jpg_buf_len));
  }
  return res;
}

void startCameraServer(){
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 80;
```

```
  httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };

  //Serial.printf("Starting web server on port: '%d'\n", config.server_port);
  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &index_uri);
  }
}

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout
detector

  Serial.begin(115200);
  Serial.setDebugOutput(false);

  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_UXGA;
  config.jpeg_quality = 10;
  config.fb_count = 2;
} else {
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
}

// Camera init
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
```

```
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
  // Wi-Fi connection
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  Serial.print("Camera Stream Ready! Go to: http://");
  Serial.print(WiFi.localIP());

  // Start streaming web server
  startCameraServer();
}

void loop() {
  delay(1);
}
```

Додаток Б

Програмний код для розпізнавання яскравої точки та управління автономною платформою

```
import cv2
import numpy as np
import serial
import time

ser = serial.Serial('COM8', 9600, timeout=1)
time.sleep(2)

url = "http://***.***.**.***:**/stream"
cap = cv2.VideoCapture(url)

if not cap.isOpened():
    print("Не вдалося підключитися до потоку.")
    exit()
else:
    print("Потік відкрито. Натисніть ESC для виходу.")

laser_found = False
stop_start_time = None
center_x = 120
threshold = 250
tolerance = 40

# Логіка пошуку
searching = False
```

```
search_phase = 'turn'
search_start_time = None

# Логіка етапного повороту після 4 секунд
adjusting = False
adjust_start_time = None
adjusting_direction = None

while True:
    ret, frame = cap.read()
    if not ret:
        print("Кадр не отримано")
        break

    frame = cv2.resize(frame, (240, 240))
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(gray)
    current_time = time.time()

    cv2.rectangle(frame, (center_x - tolerance, 0), (center_x + tolerance, frame.shape[0]), (255, 0, 0), 1)

    if max_val < threshold:
        if not searching:
            search_phase = 'turn'
            search_start_time = current_time
            searching = True
            ser.write(b'L')
        else:
```

```
            elapsed = current_time - search_start_time
            if search_phase == 'turn' and elapsed >= 2.0:
                ser.write(b'S')
                search_phase = 'pause'
                search_start_time = current_time
            elif search_phase == 'pause' and elapsed >= 1.0:
                ser.write(b'L')
                search_phase = 'turn'
                search_start_time = current_time

        laser_found = False
        stop_start_time = None
        adjusting = False
        cv2.putText(frame, f"Laser NOT found - Searching ({search_phase})", (10, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
    else:
        searching = False
        distance = max_loc[0] - center_x

        if not laser_found:
            ser.write(b'S')
            stop_start_time = current_time
            laser_found = True
            adjusting = False
            cv2.putText(frame, "Laser found - Stopping 4s", (10, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
        else:
            elapsed = current_time - stop_start_time
            if elapsed < 4:
```

```
            ser.write(b'S')
            cv2.putText(frame, f"Stopping... {int(4 - elapsed)}s left", (10, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
            adjusting = False
        else:
            if abs(distance) <= tolerance:
                ser.write(b'F')
                adjusting = False
                cv2.putText(frame, "Centered - Moving Forward", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
            else:
                # ініціація нового циклу повороту
                if not adjusting:
                    adjusting = True
                    adjust_start_time = current_time
                    adjusting_direction = b'R' if distance > 0 else b'L'
                    ser.write(adjusting_direction)
                else:
                    elapsed_adjust = current_time - adjust_start_time
                    if elapsed_adjust >= 0.5:
                        ser.write(b'S')
                        adjusting = False  # почати новий цикл у наступному кадрі

                cv2.putText(frame, "Turning to center...", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 255), 2)

    if max_val >= threshold:
        cv2.circle(frame, max_loc, 10, (0, 255, 255), 2)
        coord_text = f"({max_loc[0]}, {max_loc[1]})"
```

```
        cv2.putText(frame, coord_text, (max_loc[0] + 10, max_loc[1]),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 255), 2)

    cv2.imshow("Laser Tracking", frame)

    if cv2.waitKey(1) == 27:
        break

ser.write(b'S')
ser.close()
cap.release()
cv2.destroyAllWindows()
```

Додаток В

Програмний код для керування двигунами

```
const int left_ctrl = 2;     //define the direction control pins of group B motor
const int left_pwm = 5;      //define the PWM control pins of group B motor
const int right_ctrl = 4;    //define the direction control pins of group A motor
const int right_pwm = 6;     //define the PWM control pins of group A motor

void setup() {
  Serial.begin(9600);
  pinMode(left_ctrl, OUTPUT);
  pinMode(left_pwm, OUTPUT);
  pinMode(right_ctrl, OUTPUT);
  pinMode(right_pwm, OUTPUT);

  // Зупинити двигуни на початку
  stopMotors();
}

void loop() {
  if (Serial.available() > 0) {
    char command = Serial.read();
    processCommand(command);
  }
}

void processCommand(char command) {
  switch (command) {
```

```
    case 'F':
      goForward();
      break;
    case 'B':
      goBackward();
      break;
    case 'L':
      turnLeft();
      break;
    case 'R':
      turnRight();
      break;
    case 'S':
      stopMotors();
      break;
    // Додайте інші команди за потреби (наприклад, швидкість)
  }
}

void goForward() {
  digitalWrite(left_ctrl, HIGH);
  analogWrite(left_pwm, 155);
  digitalWrite(right_ctrl, HIGH);
  analogWrite(right_pwm, 155);
}

void goBackward() {
  digitalWrite(left_ctrl, LOW);
```

```
  analogWrite(left_pwm, 100);
  digitalWrite(right_ctrl, LOW);
  analogWrite(right_pwm, 100);
}

void turnLeft() {
  digitalWrite(left_ctrl, LOW);
  analogWrite(left_pwm, 100);
  digitalWrite(right_ctrl, HIGH);
  analogWrite(right_pwm, 155);
}

void turnRight() {
  digitalWrite(left_ctrl, HIGH);
  analogWrite(left_pwm, 155);
  digitalWrite(right_ctrl, LOW);
  analogWrite(right_pwm, 100);
}

void stopMotors() {
  digitalWrite(left_ctrl, LOW);
  analogWrite(left_pwm, 0);
  digitalWrite(right_ctrl, LOW);
  analogWrite(right_pwm, 0);
}

// Функція для керування швидкістю (використовує PWM)
void setMotorSpeed(int motor, int speed) {
```

```
  if (motor == 1) {
    analogWrite(left_pwm, speed); // 0-255 (ЗМІНЕНО на left_pwm)
  } else if (motor == 2) {
    analogWrite(right_pwm, speed); // 0-255 (ЗМІНЕНО на right_pwm)
  }
}
```