Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Автоматики і комп'ютеризованих технологій_____
(повна назва)

Кафедра __Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки___
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____
Розробка системи розпізнавання інформаційних позначок для автономної навігації мобільного робота на складському комплексі
(тема)

Виконав:
здобувач ____2_____ року навчання,
групи КІТПВм-24 1_____
_____Анатолій ЄЧЕВСЬКИЙ_____
(власне ім'я, прізвище)

Спеціальність 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка_____
(код і повна назва спеціальності)
Тип програми ____освітньо-професійна_____
Освітня програма Комп'ютерно-інтегровані технологічні процеси і виробництва_____
(повна назва освітньої програми)

Керівник _____Світлана СОТНИК_____
(посада, власне ім'я, прізвище)

Допускається до захисту
Завідувач кафедри КІТАР _____ __Ігор НЕВЛЮДОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Я, Єчевський Анатолій Дмитрович, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу з академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«19» грудня 2025р  Анатолій ЄЧЕВСЬКИЙ

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
Рівень вищої освіти другий (магістерський)
Спеціальність 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка
(код і повна назва)
Тип програми Освітньо-професійна
Освітня програма Комп'ютерно-інтегровані технологічні процеси і виробництва
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачу Єчевському Анатолію Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи розпізнавання інформаційних позначок для автономних навігації мобільного робота на складському комплексі
затверджена наказом університету від «10» листопада 2025 р. №. 1029Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 22 грудня 2025 р.
3. Вихідні дані до роботи модуль ESP32-CAM; багатоядерний CPU (Intel i5); оперативна пам'ять 16 ГБ; SSD (256 ГБ); модулі зв'язку (Wi-Fi 5, 2G/3G); монітор (23 дюйми, Full HD)
4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ;
4.2 Аналіз предметної області та постановка завдання;
4.3 Огляд сучасних методів навігації мобільних роботів;
4.4 Розробка структури та алгоритмів системи розпізнавання;
4.5 Реалізація та дослідження роботи системи розпізнавання інформаційних позначок;
4.6 Експериментальна перевірка та оцінка ефективності;
4.7 Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

Графічний матеріал у вигляді презентації – 14 арк. ф. А4

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	01.09.25-15.09.25	Виконано
2	Аналіз літератури та супутніх джерел	15.09.25-23.09.25	Виконано
3	Аналіз предметної області та визначення завдань	24.09.25-29.10.25	Виконано
4	Огляд сучасних методів навігації мобільних роботів	30.09.25-15.10.25	Виконано
5	Розробка структури та алгоритмів системи розпізнавання	16.10.25-01.11.25	Виконано
6	Реалізація та дослідження роботи системи розпізнавання інформаційних позначок	01.11.25-15.11.25	Виконано
7	Експериментальна перевірка та оцінка ефективності	16.11.25-30.11.25	Виконано
8	Оформлення пояснювальної записки	01.12.25-03.12.25	Виконано

Дата видачі завдання «01» вересня 2025 р.

Здобувач _____ Анатолій ЄЧЕВСЬКИЙ
(підпис)

Керівник роботи _____ доц. Світлана СОТНИК
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 104 с., 5 табл., 35 рис., 2 додатків, 22 джерела.

СИСТЕМА, РОЗПІЗНАВАННЯ, АВТОНОМНА НАВІГАЦІЯ, МОБІЛЬНИЙ РОБОТ, СКЛАДСЬКИЙ КОМПЛЕКС.

Метою роботи є підвищення ефективності автономної навігації мобільного робота на складі шляхом розробки системи розпізнавання інформаційних позначок для точного позиціонування та оптимізації маршрутів.

Об’єкт дослідження – процес розпізнавання інформаційних позначок мобільним роботом на складському комплексі.

Предмет дослідження – система розпізнавання інформаційних позначок для забезпечення автономної навігації мобільного робота.

У кваліфікаційній роботі проведено комплексний аналіз існуючих систем навігації мобільних роботів на складських комплексах, на основі чого обґрунтовано доцільність використання візуальних маркерів у поєднанні з платформою ESP32-CAM. Розроблено повну специфікацію системи, архітектуру програмно-апаратного комплексу та алгоритм розпізнавання. Виконано практичну реалізацію системи на базі ESP32-CAM. Експериментальна верифікація підтвердила придатність розробленої системи для інтеграції у контур автономної навігації мобільних роботів.

Результати роботи відповідають Цілям сталого розвитку 9 та 12: «Промисловість, інновації та інфраструктура», «Відповідальне споживання та виробництво».

Практична цінність роботи полягає в можливості використання розробленої системи розпізнавання інформаційних позначок в реальних умовах на складському комплексі.

ABSTRACT

The thesis contains: 104 p., 5 tables, 35 figures, 2 appendices, 22 sources.

SYSTEM, RECOGNITION, AUTONOMOUS NAVIGATION, MOBILE ROBOT, WAREHOUSE COMPLEX.

The goal of this work is to improve the efficiency of autonomous navigation of mobile robots in warehouses by developing a system for recognizing information markers for accurate positioning and route optimization.

The object of the study is the process of recognizing information markers by a mobile robot in a warehouse complex.

The subject of the study is a system for recognizing information markers to ensure autonomous navigation of a mobile robot.

The thesis provides a comprehensive analysis of existing navigation systems for mobile robots in warehouse complexes, based on which the feasibility of using visual markers in combination with the ESP32-CAM platform is justified. A complete system specification, software and hardware architecture, and recognition algorithm have been developed. The system has been implemented on the ESP32-CAM platform. Experimental verification has confirmed the suitability of the developed system for integration into the autonomous navigation circuit of mobile robots.

The results of the work correspond to Sustainable Development Goals 9 and 12: «Industry, Innovation and Infrastructure» and «Responsible Consumption and Production».

The practical value of the work lies in the possibility of using the developed system for recognizing information tags in real conditions at a warehouse complex.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ДН ВП – державне науково-виробниче підприємство;

ДП – державне підприємство;

КІТАР – комп'ютерно-інтегровані технології, автоматизація та робототехніка;

НДТІП – науково-дослідний технологічний інститут приладобудування;

НКЦ – науково-консультаційний центр;

ПЗ – програмне забезпечення;

САПР – система автоматизованого проектування;

ТВР – технологія виробництва радіоапаратури;

ХВО – Харківське виробниче об'єднання;

ХНУРЕ – Харківський національний університет радіоелектроніки;

AGV – Automated Guided Vehicle – автоматично керований транспортний засіб;

AMR – Autonomous Mobile Robot – автономний мобільний робот;

CNN – Convolutional Neural Networks – згорткові нейронні мережі;

DMA – Direct Memory Access – прямий доступ до пам'яті;

DVP – Digital Video Port – цифровий відеопорт;

FPU – Floating Point Unit – блок операцій з плаваючою комою;

GPU – Graphics Processing Unit – графічний процесор;

HAL – Hardware Abstraction Layer – рівень абстракції апаратури;

ID – Identifier – ідентифікатор;

IMU – Inertial Measurement Unit – інерційний вимірювальний пристрій;

LiDAR – Light Detection and Ranging – лазерне виявлення та вимірювання відстані;

ML – Machine Learning – машинне навчання;

NHWC – Number, Height, Width, Channels – число, висота, ширина, канали;

PSRAM – Pseudo-Static RAM – псевдостатична оперативна пам'ять;

QR – Quick Response codes – коди швидкої відповіді;

ReLU – Rectified Linear Unit – випрямлений лінійний блок;

RFID – Radio Frequency Identification – радіочастотна ідентифікація;

RGB-D – Red, Green, Blue, Depth – червоний, зелений, синій, глибина;

SLAM – Simultaneous Localization and Mapping – одночасна локалізація та картографування;

SMART – Specific, Measurable, Achievable, Relevant, Time-bound – конкретність, вимірюваність, досяжність, релевантність, обмеженість у часі;

V-SLAM – Visual Simultaneous Localization and Mapping – візуальна одночасна локалізація та картографування.

ВСТУП

Активна цифровізація логістики, зростання обсягів e-commerce та вимоги до обслуговування в режимі реального часу зумовлюють швидке впровадження автономних мобільних роботів (AMR/AGV) у складських комплексах. Ключовою передумовою безпечної й ефективної роботи таких роботів є надійна навігація та високоточне позиціонування в динамічному середовищі складу.

Традиційні підходи до локалізації – одометрія, візуально-інерційні системи або Simultaneous Localization and Mapping (SLAM) з використанням Light Detection and Ranging (LiDAR) – дають достатню відносну точність, але чутливі до накопичення помилок (дрейфу), переобладнання середовища або дефіциту текстур. Інформаційні позначки (fiducial-мітки, Quick Response (QR) коди, DataMatrix коди, алфанумеричні етикетки поличок, підлогові та навісні маркери) забезпечують дешеве, масштабоване та легко підтримуване джерело абсолютних прив’язок, що доповнює навігаційну підсистему і дозволяє зменшити дрейф і підвищити точність позиціонування, до того ж оптимізувати маршрути з урахуванням підтверджених контрольних точок, і прискорити інвентаризаційні та відвантажувальні операції.

Мета роботи – підвищити ефективність автономної навігації мобільного робота на складі шляхом розробки системи розпізнавання інформаційних позначок для точного позиціонування та оптимізації маршрутів.

Для досягнення мети передбачається розв’язати такі завдання:

– провести огляд сучасних методів розпізнавання позначок (ArUco/AprilTag, QR/DataMatrix, штрих-коди) та підходів до локалізації на їх основі;

– сформулювати вимоги до системи (точність, латентність, діапазони відстаней та кутів огляду, умови освітлення);

– обґрунтувати вибір типів позначок;

– реалізувати прототип, оптимізувати обчислення;

– визначити метрики та процедуру тестування;

– сформувати/розмітити датасет складових позначок, провести моделювання і натурні випробування;

– виконати порівняльний аналіз та сформувати рекомендації з розміщення позначок;

– -оформити кваліфікаційну роботу згідно методичними вказівками з підготовки й оформлення кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка освітньої програми «Комп'ютерно-інтегровані технологічні процеси і виробництва» [1], положення про організацію освітнього процесу у ХНУРЕ [2] та ДСТУ 3008:2015 [3].

Об'єкт дослідження – процес розпізнавання інформаційних позначок мобільним роботом на складському комплексі.

Предмет дослідження – система розпізнавання інформаційних позначок для забезпечення автономної навігації мобільного робота.

Наукова новизна та практичне значення полягає у детекції маркерів з оцінкою достовірності спостереження завдяки удосконаленню системи детекції навігаційних маркерів на базі ультрабюджетної апаратної платформи ESP32-CAM з використанням оптимізованої квантизованої нейронної мережі (MobileNet v2).

Рішення масштабоване та економічно ефективне через використання недорогих друкованих позначок, придатне для модернізації існуючих складів без суттєвих змін інфраструктури. Результати можуть бути використані в суміжних задачах: інвентаризація, навігація в виробничих цехах, контроль проходження контрольних пунктів, точне позиціювання при завантаженні/розвантаженні. Запропонована робота спрямована на створення технологічно зрілої та відтворюваної системи, здатної працювати в реальних умовах складу, підвищуючи надійність навігації мобільних роботів і ефективність логістичних процесів, що відповідає актуальним запитам індустрії та тенденціям Індустрії 4.0.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

Сучасні складські комплекси переживають період трансформації від традиційних методів організації логістичних процесів до високоавтоматизованих систем, що використовують мобільні роботи для виконання рутинних операцій. Ця трансформація зумовлена потребою підвищення продуктивності, точності та економічної ефективності складських операцій в умовах зростаючих обсягів вантажообігу та вимог до швидкості обробки замовлень [4-8].

Автономна навігація мобільних роботів є ключовою технологією, що визначає ефективність їх застосування на складських комплексах. Від точності та надійності системи навігації залежить не лише продуктивність окремого робота, але й безпека персоналу, збереженість вантажів та можливість одночасної роботи декількох автономних одиниць у спільному просторі [5].

Існуючі на ринку рішення характеризуються широким діапазоном технологій навігації – від простих систем слідування по магнітній стрічці до складних комплексів з лазерними сканерами, системами технічного зору та штучним інтелектом. Водночас, значна частина цих рішень має суттєві обмеження: високу вартість впровадження, складність масштабування, залежність від дорогого обладнання або низьку гнучкість при зміні конфігурації складу.

Розвиток недорогих мікроконтролерних платформ зі вбудованими камерами, зокрема ESP32-CAM, та доступність ефективних алгоритмів машинного навчання, адаптованих для роботи на обмежених обчислювальних ресурсах, відкривають нові можливості для створення економічно доцільних систем автономної навігації. Використання компактних нейронних мереж дозволяє реалізувати розпізнавання інформаційних позначок безпосередньо на борту робота без потреби у потужних серверах або хмарних обчисленнях [9].

Для обґрунтованої розробки системи розпізнавання інформаційних позначок необхідно провести комплексний аналіз існуючих рішень у галузі мобільної робототехніки для складських комплексів, дослідити методи та технології навігації, що застосовуються у сучасних системах, визначити їх переваги та недоліки, а також сформулювати вимоги до розроблюваної системи з урахуванням специфіки апаратної платформи та умов експлуатації.

1.1 Аналіз існуючих мобільних роботів на складському комплексі

Ринок мобільних роботів для складської логістики представлений широким спектром рішень від різних виробників. Для розуміння поточного стану технологій та визначення напрямків вдосконалення проведемо аналіз типових представників різних поколінь складських роботів з акцентом на методи навігації, що вони використовують.

1.1.1 Роботи на основі магнітної навігації

Swisslog TransCar AGV, що зображені на рисунку 1.1, використовують систему TransCar, яка представляє класичне рішення автоматизованих транспортних засобів (AGV), що широко використовується на складах та виробничих підприємствах.

Рисунок 1.1 – Приміщення із магнітними стрічками та TransCar AGV [11]

Принцип навігації полягає в тому, що робот рухається вздовж магнітної стрічки або вбудованих у підлогу магнітних маркерів, які детектуються спеціальними сенсорами на нижній частині платформи.

Технічні характеристики наведені в таблиці 1.1.

Таблиця 1.1 – Основні характеристики Swisslog TransCar AGV [11]

Параметр	Значення
Вантажопідйомність	500-1500 кг
Швидкість руху	до 1 м/с
Точність позиціонування	±10 мм відносно стрічки
Система навігації	магнітні сенсори

Такі роботи мають наступні переваги:

– висока надійність на фіксованих маршрутах;

– простота базової реалізації;

– низька чутливість до умов освітлення.

Недоліки виявлені наступні:

– необхідність фізичної модифікації підлоги (встановлення магнітних стрічок);

– відсутність гнучкості – зміна маршруту вимагає переклеювання стрічок;

– неможливість автоматичної адаптації до змін планування складу;

– високі витрати на початкове впровадження та реконфігурацію;

– обмежена маневреність;

– проблеми при пошкодженні або забрудненні магнітної стрічки.

1.1.2 Роботи з лазерною навігацією

MiR100 – один з популярних представників роботів з автономною навігацією на основі лазерного сканування (LiDAR-based SLAM) наведений на рисунку 1.2.

Рисунок 1.2 – Зовнішній вигляд The MiR100 [12]

В принцип навігації закладено використання 2D LiDAR для побудови карти приміщення та визначення власного положення (SLAM).

Технічні характеристики наведені в таблиці 1.2.

Таблиця 1.2 – Основні технічні характеристики MiR100 [13]

Параметр	Значення
Вантажопідйомність	100 кг
Швидкість руху	до 1,5 м/с
Точність позиціонування	±10-50 мм (залежно від умов)
Система навігації	2D LiDAR

Даний вид мобільних роботів має наступні переваги:

– автономна побудова карти приміщення;

– гнучкість маршрутів без фізичних маркерів;

– виявлення перешкод.

Проте виявлені наступні недоліки:

– висока вартість лазерного сканера (складає значну частину вартості системи);

– проблеми у приміщеннях з однорідними стінами або великими відкритими просторами;

– складність точного позиціонування біля стелажів у щільно заповнених складах;

– чутливість до динамічних змін середовища (переміщення товарів, тимчасові перешкоди);

– обмежена точність на великих відстанях;

– накопичення помилки при тривалому русі без переприв’язки до абсолютних координат;

– потреба у потужному обчислювальному модулі для обробки даних.

Індійська компанія GreyOrange виробляє серію роботів Butler, що наведені на рисунку 1.3, які використовують комбіновану систему навігації з акцентом на лазерні рефлектори.

Рисунок 1.3 – Роботи GreyOrange Butler на складі Amazon [14]

Для навігації використовується принципом тріангуляції, коли лазерний сканер детектує спеціальні рефлектори, встановлені на стелажах та стінах складу, визначаючи положення за принципом тріангуляції.

Технічні характеристики наведені в таблиці 1.3.

Таблиця 1.3 – Основні технічні характеристики GreyOrange Butler [14]

Параметр	Значення
Вантажопідйомність	до 100 кг (підйом стелажа)
Швидкість руху	1,5 м/с
Точність позиціонування	±10 мм
Система навігації	LiDAR + лазерні рефлектори

Перевагами GreyOrange Butler є:

– вища точність порівняно з SLAM;

– можливість абсолютного позиціонування.

Але поряд із перевагами такі недоліки:

– необхідність встановлення та точного калібрування рефлекторів;

– висока вартість лазерного обладнання;

– складність реконфігурації при зміні розміщення стелажів;

– вразливість до забруднення або пошкодження рефлекторів;

– висока загальна вартість системи (лазер + рефлектори + обчислювач).

1.1.3 Роботи з комбінованими системами навігації

Американська компанія Fetch Robotics (згодом придбана Zebra Technologies) випустила серію роботів з комбінованою навігацією Fetch Robotics Freight 100, представник якої зображений на рисунку 1.4.

Рисунок 1.4 – Fetch Robotics Freight 100 [15]

Fetch Robotics Freight 100 використовує поєднання 2D LiDAR для SLAM, 3D камери для виявлення перешкод та опціонально візуальних маркерів для точного позиціонування.

Технічні характеристики наведені в таблиці 1.4.

Таблиця 1.4 – Основні технічні характеристики Fetch Robotics Freight 100 [15]

Параметр	Значення
Вантажопідйомність	100 кг
Швидкість руху	1,1 м/с
Точність позиціонування	±50 мм у режимі SLAM, до ±10 мм з маркерами
Система навігації	LiDAR + RGB-D камера + Inertial Measurement Unit (IMU)

Такий тип роботів має наступні переваги:

– більша надійність завдяки резервуванню систем;

– краще виявлення динамічних перешкод.

Також очевидні наступні недоліки:

– дуже висока вартість через використання множинних дорогих сенсорів;

– складність інтеграції та налаштування;

– висока обчислювальна потужність (зазвичай Intel i5/i7);

– значне енергоспоживання;

– надмірність для простих складських задач.

Аналіз існуючих мобільних роботів для складських комплексів виявив, що більшість доступних на ринку рішень використовують або дорогі лазерні системи навігації, або потребують значних інвестицій в інфраструктуру (магнітні стрічки, рефлектори, підлогові маркери). Це створює бар’єр для впровадження автоматизації на малих та середніх складах.

1.2 Огляд сучасних методів навігації мобільних роботів

Автономна навігація мобільних роботів є комплексною задачею, що включає визначення власного положення у просторі (локалізація), побудову або використання карти середовища (картографування) та планування безпечної траєкторії руху до цільової точки [9]. Розвиток робототехніки призвів до створення різноманітних методів та технологій навігації, кожен з яких має специфічні переваги, недоліки та сферу застосування.

Вибір методу навігації для конкретного застосування визначається множиною факторів: вимогами до точності позиціонування, характеристиками середовища експлуатації, економічними обмеженнями, доступними обчислювальними ресурсами, рівнем динамічності оточення та необхідністю масштабування системи.

1.2.1 Класифікація методів навігації мобільних роботів

Методи навігації можна класифікувати за різними критеріями.

За принципом визначення положення:

– абсолютна навігація – позиція визначається відносно глобальної системи координат;

– відносна навігація – позиція визначається відносно початкової точки;

– гібридна навігація – комбінація абсолютної та відносної.

За типом використовуваних сенсорів [10]:

– механічні, такі як гіроскопи та ін.;

– оптичні, що включають камери та лазерні сканери;

– магнітні – компаси, датчики магнітного поля та ін.;

– радіочастотні, поширеним прикладом є Radio Frequency Identification (RFID);

– комбіновані (sensor fusion).

За рівнем автономності:

– навігація по фіксованих траєкторіях (guided navigation);

– напівавтономна навігація (з втручанням оператора);

– повністю автономна навігація (autonomous navigation).

За потребою в інфраструктурі:

– потребують спеціального обладнання середовища;

– працюють у немодифікованому середовищі;

– можуть працювати в обох режимах.

Тож, методи навігації мобільних роботів відрізняються за кількома ключовими аспектами. В основному положення робота може визначатися абсолютно, відносно або ж комбіновано. Важливу роль відіграють типи сенсорів, серед яких виділяють механічні, оптичні, магнітні та радіочастотні пристрої, що часто поєднуються для підвищення точності. Рівень автономності може варіюватися від ручного керування по заданих шляхах до повністю самостійної роботи, а залежність від інфраструктури – від потреби у спеціальному обладнанні середовища до функціонування у звичайному просторі. Таким чином, вибір оптимальної методики навігації є комплексним завданням, що залежить від конкретних умов та вимог до робота.

1.2.2 Навігація по фізичним направляючим

Індуктивна (дротова) навігація є одним з базових методів навігації автоматизованого транспорту, де у підлогу вбудовується дріт, по якому пропускається змінний струм. Робот знаходить електромагнітне поле за допомогою антен-сенсорів, а система управління коригує рух для утримання робота над проводом.

Переваги даного методу навігації:

– висока надійність у промисловому середовищі;

– стійкість до забруднень на підлозі;

– працює за будь-яких умов освітлення.

Недоліки індуктивної навігації:

– необхідність руйнівного втручання у підлогу для прокладання проводів;

– дуже висока складність та вартість зміни маршрутів;

– відсутність гнучкості – робот не може відхилитися від заздалегідь визначеної траєкторії;

– високі витрати на початкове впровадження.

Застосовується зазвичай на великих виробничих підприємствах з фіксованими довготривалими маршрутами.

Магнітна навігація в свою чергу використовує магнітну стрічку, наклеєну на підлогу, або магнітні маркери, вбудовані у підлогу.

Принцип роботи полягає в тому, що магнітні сенсори (датчики Холла або магніторезистивні сенсори) визначають магнітне поле стрічки. Система управління аналізує сигнали від лінійки сенсорів для визначення положення відносно стрічки.

Переваги:

– простіше впровадження порівняно з дротовою навігацією;

– не потребує джерела живлення для стрічки;

– невисокі обчислювальні вимоги.

Недоліки:

– знос стрічки від руху техніки та персоналу;

– втрата навігації при пошкодженні стрічки;

– витрати на переклеювання при зміні планування (від $50 за метр з роботами);

– робот «сліпий» поза стрічкою – не бачить перешкод;

– необхідність додаткових датчиків безпеки.

Таким метод навігації зазвичай застосовується на складах з відносно стабільним плануванням та середнім рівнем автоматизації.

Обидва методи – індуктивна та магнітна навігація – є класичними рішеннями для організації руху автоматизованого транспорту по заданих маршрутах. Вони демонструють високу надійність у контрольованих умовах, але їхня основна спільна риса – це жорстка прив'язка до попередньо встановленої інфраструктури. Ця ключова характеристика одночасно визначає як їхні переваги, так і головні недоліки.

З одного боку, така прив’язка забезпечує стабільність і точність слідування. З іншого – робить систему абсолютно негнучкою. Будь-яка зміна логістичних маршрутів вимагає значних капітальних витрат і фізичного втручання в інфраструктуру приміщення.

Таким чином, вибір між цими методами є компромісом між початковими інвестиціями, надійністю та майбутньою гнучкістю. Вони ідеально підходять для виробничих та складських систем зі стабільним, незмінним плануванням, де пріоритетом є довготривала та надійна робота за фіксованими маршрутами, а не адаптивність до змін.

1.2.3 Навігація на основі маяків та рефлекторів

Метод лазерної навігації по рефлекторах використовує лазерний сканер на роботі та спеціальні світловідбиваючі маркери (рефлектори), встановлені у відомих позиціях.

Лазерний сканер обертається з частотою від 5 Гц до 15 Гц, випромінюючи імпульси світла. При попаданні на рефлектор відбувається яскраве відбиття, яке детектується. Вимірюючи кути та відстані до кількох рефлекторів, система обчислює положення робота.

Точність такої лазерної навігації складає приблизно від 5 мм до 10 мм при видимості від трьох рефлекторів. Дальність розпізначання рефлектора складає до 30 м.

Переваги лазерного методу навігації полягає виявлені наступні:

– висока точність позиціонування;

– абсолютна локалізація без накопичення помилки;

– не потребує модифікації підлоги.

Недоліки:

– висока вартість лазерного сканера (від 3000 $ до 5000 $ для базових моделей);

– необхідність точного розміщення та геодезичної зйомки позицій рефлекторів;

– потреба у прямій видимості рефлекторів;

– проблеми у щільно заповнених складах (блокування видимості товарами);

– чутливість до забруднення оптики сканера пилом;

– енергоспоживання лазерного сканера складає від 8 Вт до 12 Вт.

Метод лазерної навігації по рефлекторах застосовується зазвичай на середніх та великих склади з можливістю інвестування у дороге обладнання.

RFID-навігація передбачає використання радіочастотних міток, розміщених у відомих точках на підлозі або інфраструктурі складу.

Принцип роботи настпуний, RFID-зчитувач на роботі детектує пасивні або активні мітки. Кожна мітка містить унікальний ідентифікатор, пов’язаний з координатами у базі даних [11].

Переваги:

– низька вартість міток (пасивні RFID коштують від 0,1 $ до 1 $);

– стійкість до забруднення та зносу;

– можливість зчитування через нетовсті матеріали.

Недоліки:

– дискретність позиціонування (тільки у точках розміщення міток);

– мала дальність зчитування пасивних міток (від 5 см до 30 см, потребують щільного розміщення);

– активні мітки дорожчі та потребують живлення;

– відсутність інформації про орієнтацію робота;

– інтерференція від металевих конструкцій складу.

Використання RFID-навігації вузькоспеціалізоване, часто як доповнення до інших методів.

1.2.4 Одометрія та інерціальна навігація

Колісна одометрія є базовим метод відносної навігації, що обчислює переміщення на основі обертання коліс.

Енкодери на осях коліс вимірюють кути повороту. Використовуючи радіус коліс та геометрію платформи, обчислюється зміна позиції та орієнтації (dead reckoning).

Переваги:

– дуже низька вартість (енкодери коштують від 5 $ до 20 $);

– мінімальні обчислювальні витрати;

– висока частота оновлення (до 1000 Гц);

– працює у будь-яких умовах.

Недоліки:

– накопичення помилки у часі (типово від 1 % до 5 % від пройденої відстані);

– чутливість до прослизання коліс;

– похибки від нерівності підлоги;

– не може використовуватися самостійно для довготривалої навігації.

Застосовується як допоміжна система, що інтегрується з іншими методами локалізації.

Інерціальна навігаційна система передбачає використання акселерометрів та гіроскопів для визначення прискорень та кутових швидкостей.

Принцип роботи полягає у поєднанні вимірів акселерометрів, що визначають швидкість та позицію, та гіроскопів, що визначають орієнтацію. Сучасні IMU часто включають магнітометр для корекції курсу.

Переваги наступні:

– висока частота оновлення (від 100 Гц до 1000 Гц);

– автономність від зовнішніх сигналів;

– низька вартість MEMS-датчиків (від 1 $ до 5 0$);

– працює у будь-яких умовах.

Недоліки досить значущі, а саме:

– швидке накопичення помилки через подвійне інтегрування (квадратична помилка по позиції);

– дрейф гіроскопів;

– чутливість до вібрацій.

Застосовуються виключно у комбінації з іншими методами (sensor fusion) для покращення динамічних характеристик.

1.2.5 Одночасна локалізація та побудова карти (SLAM)

SLAM (Simultaneous Localization and Mapping) – одночасна локалізація та побудова карти – представляє сімейство алгоритмів, що дозволяють роботу визначати своє положення, одночасно створюючи карту невідомого середовища [9].

Лазерний SLAM (LiDAR-based SLAM) є найпоширенішим методом для наземних роботів, що використовує 2D або 3D лазерні сканери.

Лазерний сканер створює «зріз» оточення у вигляді набору точок (point cloud). Алгоритм зіставляє послідовні скани для визначення руху робота (scan matching) та поступово будує глобальну карту. Додатково використовується одометрія для покращення оцінки руху.

Для забезпечення точності 2D LiDAR на рівні від 30 мм до 100 мм (залежно від відстані та алгоритму) потрібен процесор рівня ARM Cortex-A53 або Intel Atom.

Переваги:

– не потребує підготовки середовища;

– автоматична адаптація до змін планування;

– одночасно будує карту для планування шляху.

Недоліки:

– висока вартість лазерного сканера (залежно від якості вартість становить від 300 $ до 5000 $);

– проблеми у приміщеннях з рухомими об’єктами (динамічні SLAM складніші);

– накопичення помилки у великих приміщеннях;

– складність точного позиціонування (точність від 30 мм до 50 мм є недостатньою для деяких задач);

– енергоспоживання для сканера становить від 5 Вт до 15 Вт, а для обчислювача – від 3 Вт до 10 Вт;

– проблеми у середовищах з великими відкритими просторами або повторюваними структурами.

SLAM застосовується у сучасних автономних роботах середнього та вищого цінового сегменту.

Візуальний SLAM (V-SLAM) використовує камери для одночасної локалізації та побудови карти. Таким чином відбувається виділення характерних точок (features) на зображеннях, відстеження їх між кадрами, оцінка руху камери та тривимірної структури сцени.

Проте точність сильно залежить від текстурованості середовища, від 50 мм до 200 мм, одночасно із високими обчислювальними вимогами на рівні Intel i5 або аналогу ARM з GPU прискоренням.

Переваги:

– камери значно дешевші за LiDAR (вартість камери в межах від 10 $ до 200 $);

– отримання візуальної інформації про середовище (розпізнавання об’єктів);

– можливість роботи у 3D.

Недоліки:

– дуже високі обчислювальні вимоги через складні алгоритми комп’ютерного зору;

– чутливість до умов освітлення;

– проблеми у текстурно-бідних середовищах (білі стіни, однорідні поверхні);

– монокулярний SLAM не визначає абсолютний масштаб;

– значне енергоспоживання обчислювального модуля на рівні від 10 Вт до 25 Вт;

– складність налаштування та калібрування.

Застосовується у дослідницьких роботах, дронах, деяких сервісних роботах. Рідше у промислових AGV через нестабільність.

1.2.6 Візуальна навігація на основі маркерів

Окрема категорія методів навігації використовує штучні візуальні маркери відомого вигляду, розміщені у середовищі. Камера на роботі детектує ці маркери, а алгоритми комп'ютерного зору обчислюють положення камери (а отже, робота) відносно маркера.

Далі розглянуто типи візуальних маркерів.

Перший тип – QR-коди (Quick Response codes) – двовимірні матричні штрих-коди, що мають високий інформаційний об'єм до кількох кілобайт. Мають стандартизовані алгоритми детекції та декодування.

Другий тип – це ArUco маркери спеціально розроблені для додатків AR та робототехніки. Так, бібліотека OpenCV має вбудовану підтримку, що передбачає швидку детекцію завдяки оптимізованим алгоритмам.

Третій тип – це AprilTag, який є покращеною версією ArUco, що дає кращі показники детекції за складних умов, робастність до розмиття та часткового перекриття. Має різні сімейства (tag16h5, tag25h9, tag36h11), що надають компроміс між точністю та швидкістю. Але також мають дещо більші обчислювальні вимоги за ArUco.

Кольорові маркери являють собою прості геометричні фігури з унікальними кольоровими комбінаціями. Мають найнижчі обчислювальні вимоги, проте високу чутливість до освітлення та кольорового балансу камери.

Принцип роботи навігації по маркерах складається з декількох етапів:

– аналіз алгоритмом зображення з камери, виявлення потенційних маркерів (контури, патерни);

– декодування вмісту маркера для визначення його унікального ідентифікатора;

– обчислення 6D позиції (3D координати + 3D орієнтація) камери відносно маркера на основі геометрії його проекції;

– перетворення позиції камери у позицію робота з урахуванням монтажу камери;

– використання відомих координат маркера для обчислення положення робота у глобальній системі координат складу.

Переваги маркерної навігації наступні:

– маркер коштує від 0,01 $ до 0,1 $;

– висока точність на рівні від 2 мм до 10 мм на відстані від 1 м до 3 м;

– абсолютне позиціонування без накопичення помилки, кожен маркер дає незалежну оцінку;

– маркер може нести додаткову інформацію (ідентифікатор зони, команди, пріоритети);

– гнучкість та простота використання завдяки можливості додавати, переміщати або замінювати маркери;

– можна створити ієрархічну систему маркерів різного розміру.

Недоліки маркерної навігації:

– потреба у прямій видимості маркера;

– локалізація тільки при наближенні до маркера;

– чутливість до освітлення;

– відстань розпізнавання маркера обмежена роздільністю камери (зазвичай до 3 м для маркерів розміром 10 см та до 5 м для маркерів розміром 20 см);

– обчислювальна складність.

Таким чином, виявлена ніша для створення економічно ефективної системи навігації на базі ESP32-CAM для розпізнавання візуальних маркерів, яка може конкурувати з рішеннями на базі QR-кодів на підлозі за точністю, перевершуючи їх за гнучкістю та довговічністю, при вартості на порядки нижчій за SLAM-рішення.

1.3 Аналіз використання інформаційних позначок у робототехнічних системах

Інформаційні позначки (візуальні маркери) є ключовим елементом багатьох робототехнічних систем, особливо у структурованих середовищах, таких як склади, виробничі цеха та логістичні центри. На відміну від природних візуальних орієнтирів, штучні маркери спеціально розроблені для надійної детекції, однозначної ідентифікації та точного визначення просторового положення за допомогою комп'ютерного зору.

Використання інформаційних позначок дозволяє вирішити фундаментальну проблему робототехніки – забезпечити надійну локалізацію робота у просторі без необхідності складної обробки природних зображень або використання дорогого спеціалізованого обладнання. Водночас, різні типи позначок мають різні характеристики, що визначають їх придатність для конкретних застосувань.

1.3.1 Класифікація інформаційних позначок

Інформаційні позначки можна класифікувати за різними критеріями.

За геометричною структурою:

– одновимірні (лінійні штрих-коди);

– двовимірні матричні коди (QR, Data Matrix);

– фідуціальні маркери (ArUco, AprilTag);

– кольорові патерни;

– гібридні (комбінація різних типів).

За інформаційною ємністю:

– мінімальні (тільки ідентифікатор, від 4 біт до 12 біт);

– середні (ідентифікатор + базова інформація, до 100 біт);

– високі (складні дані, до кількох кілобайт).

За методом виявлення:

– контурні (виявлення через пошук контурів);

– шаблонові (пошук специфічних візуальних шаблонів);

– кольорові (сегментація за кольором);

– гібридні (комбінація методів).

Таким чином, різноманіття інформаційних позначок дозволяє підібрати оптимальне рішення для конкретних завдань. Вибір типу маркера безпосередньо залежить від його майбутнього застосування: геометрична структура визначає просторові характеристики та стійкість до спотворень, інформаційна ємність обумовлює кількість даних, що зберігаються, а метод виявлення впливає на швидкість та точність розпізнавання в різних умовах. Гнучкість цієї технології полягає в можливості комбінування різних підходів для створення гібридних рішень, що поєднують переваги окремих методів.

1.3.2 Лінійні та двовимірні штрих-коди

Лінійні штрих-коди (1D barcodes) є найстарішим та найпоширенішим типом машиночитаних позначок, широко використовується у роздрібній торгівлі та логістіці.

Популярні стандарти:

– EAN-13/UPC – стандарт для товарів (13 цифр);

– Code 128 – універсальний код високої щільності (до 128 символів ASCII);

– Code 39 – простий альфанумеричний код;

– Interleaved 2 of 5 – для складської логістики.

В лінійних штрих-кодах інформація кодується шириною чорних та білих вертикальних смуг. Зчитування відбувається вздовж горизонтальної лінії сканування.

Переваги:

– широка стандартизація та підтримка;

– дуже низька вартість генерації та друку;

– проста технологія зчитування;

– висока швидкість декодування.

Недоліки для навігації:

– низька інформаційна ємність (зазвичай до 50 символів);

– потреба у точній орієнтації (горизонтальне сканування);

– відсутність інформації про просторове положення;

– неможливість визначення позиції та орієнтації робота відносно коду;

– чутливість до пошкоджень (одна подряпина може зробити код нечитабельним).

Широко застосовується для ідентифікації товарів при їх маніпулюванні, інвентаризації.

Двовимірними матричними кодами (2D barcodes) є QR-коди, що являє собою найпопулярніший двовимірний код, розроблений компанією Denso Wave (Японія) у 1994 році.

Структура QR-коду наступна:

– маркери позиціонування (finder patterns) – три квадрати у кутах;

– маркери вирівнювання (alignment patterns) – для компенсації спотворень;

– інформаційні модулі – матриця чорних/білих квадратів;

– коригувальні коди Reed-Solomon для виправлення помилок.

QR коди існують у версіях від 1 (21 × 21 модулів) до 40 (177 × 177 модулів).

Інформаційна ємність версії 40 із найнижчим рівнем корекції складає:

– цифри: 7089 символів;

– алфавітно-цифрові: 4296 символів;

– бінарні дані: 2953 байт;

– kanji/кирилиця: 1817 символів.

Переваги використання QR-кодів:

– висока інформаційна ємність;

– вбудована корекція помилок;

– швидке виявлення;

– можливість зчитування з будь-якого кута;

– стандартизація (ISO/IEC 18004);

– безкоштовність (відкритий стандарт).

Недоліки:

– відносно складне виявлення (перспективна корекція, декодування);

– високі обчислювальні вимоги для надійного розпізнавання;

– багато «шуму» на зображенні ускладнює виявлення на складних фонах.

1.3.3 Фідуціальні маркери для робототехніки

Фідуціальні маркери (fiducial markers) – спеціалізовані візуальні позначки, розроблені для задач комп'ютерного зору, доповненої реальності та робототехніки. На відміну від штрих-кодів, їх первинна функція – визначення просторового положення камери відносно маркера (pose estimation) [10].

ArUco маркери розроблені університетом Кордоби (Іспанія), інтегровані в OpenCV з версії 3.0.

Структура ArUco маркеру наступна:

– квадратна чорна рамка (border) для детекції;

– внутрішня бінарна матриця (зазвичай 4 × 4, 5 × 5, 6 × 6, 7 × 7 біт);

– кожен маркер має унікальний візуальний шаблон.

Словники ArUco:

– DICT_4X4_50 – 50 маркерів, 4 × 4 біт (мінімальна складність виявлення);

– DICT_5X5_100 – 100 маркерів, 5 × 5 біт;

– DICT_6X6_250 – 250 маркерів;

– DICT_7X7_1000 – 1000 маркерів;

– DICT_ARUCO_ORIGINAL – оригінальний словник ArUco.

Алгоритм виявлення ArUco:

– бінаризація зображення (адаптивна порогова обробка);

– пошук контурів (cv2.findContours);

– фільтрація контурів (приблизно квадратні, достатнього розміру);

– перспективна корекція кожного кандидата до канонічного вигляду;

– декодування внутрішньої матриці та перевірка за словником;

– визначення орієнтації (правильне розміщення кутів);

– Pose estimation (solvePnP якщо потрібна 3D позиція).

Переваги ArUco:

– дуже швидке виявлення (від 10 мс до 50 мс на ARM Cortex-A для типового зображення);

– проста реалізація (вбудована підтримка в OpenCV);

– стійкість до обертань та перспективних спотворень;

– точний pose estimation (від 2 мм до 5 мм на відстані від 0,5 м до 2 м);

– можливість одночасного виявлення множинних маркерів;

– безкоштовна генерація та друк.

Недоліки ArUco:

– обмежена кількість унікальних маркерів (максимум 1000 у стандартних словниках);

– чутливість до розмиття руху;

– менша стійкість до часткового перекриття порівняно з AprilTag;

– вимагає OpenCV (важка бібліотека для мікроконтролерів);

– обчислювальна складність на ESP32, так як повний стек OpenCV не підтримується на ESP32 через обмеження пам’яті та обчислювальної потужності. Виявлення класичним алгоритмом ArUco неможлива без значних модифікацій.

AprilTag розроблені в University of Michigan як покращення концепції ARToolkit та ArUco.

Існує три покоління AprilTag:

– AprilTag 1 (2011) – оригінальна версія;

– AprilTag 2 (2016) – покращена робастність;

– AprilTag 3 (2019) – найсучасніша версія, значно швидше виявлення.

Переваги AprilTag:

– найвища робастність серед фідуціальних маркерів;

– стійкість до motion blur (динамічні сцени);

– кращі показники при частковому перекритті;

– надійне виявлення при низькій роздільності;

– можливість виявлення на великих відстанях;

– оптимізована швидкість виявлення (AprilTag 3).

Недоліки:

– складніша реалізація порівняно з ArUco;

– дещо вищі обчислювальні вимоги (частково вирішено в AprilTag 3);

– менше готових інтеграцій.

AprilTag знайшли своє застосування під час експлуатації автономних автомобілів, маніпуляторів, AR-систем.

Інформаційні позначки є ефективним інструментом для навігації мобільних роботів у структурованих середовищах, забезпечуючи абсолютне позиціонування без накопичення помилки. Існує широкий спектр типів маркерів – від простих штрих-кодів до спеціалізованих фідуціальних маркерів (ArUco, AprilTag), кожен з яких має свою нішу застосування.

Класичні алгоритми виявлення маркерів (на базі OpenCV) є дуже ефективними, але непридатні для роботи на мікроконтролерах з обмеженими ресурсами через високі вимоги до пам’яті та обчислювальної потужності [8-11].

1.4 Постановка завдання на розробку

На основі проведеного аналізу предметної області, існуючих методів навігації мобільних роботів та технологій використання інформаційних позначок, виявлено, що існуючі комерційні рішення для навігації мобільних роботів на складах базуються переважно на дорогому обладнанні (лазерні сканери від 500 $ до 8000 $) або потребують значних інвестицій в інфраструктуру (магнітні стрічки, рефлектори, підлогові QR-коди). Економічний бар’єр для автоматизації складів малого та середнього бізнесу залишається високим – вартість одного AGV становить від 15000 $ до 50000 $, що робить впровадження нерентабельним для підприємств з обмеженим бюджетом.

Технологічні можливості сучасних мікроконтролерів (ESP32) та оптимізованих нейронних мереж дозволяють реалізувати інтелектуальну обробку зображень на ultra-low-cost платформах при споживанні енергії менше за 2 Вт. В той час як візуальні маркери типу ArUco/AprilTag забезпечують точність локалізації від 2 мм до 10 мм при вартості виготовлення від 0,05 $ до 0,1 $ за одиницю, що на 3, чи навіть 4, порядки дешевше за лазерні рефлектори.

Класичні алгоритми виявлення маркерів (OpenCV ArUco detector) неможливо запустити на ESP32 через обмеження пам’яті (520 KB SRAM) та обчислювальної потужності. Тому підхід на основі машинного навчання з використанням компактних архітектур дозволяє реалізувати виявлення маркерів на ESP32-CAM з прийнятною швидкістю (від 2 fps до 3 fps), достатньою для навігаційних задач.

Це створює передумови для розробки інноваційної системи навігації, яка поєднує переваги візуальних маркерів (точність, економічність) з можливостями машинного навчання (робастність, адаптивність) на доступній апаратній платформі.

На основі вище сказаного необхідно:

− розробити структурну схему;

− розробити алгоритми ідентифікації;

− вибрати апаратні модулі для системи;

− обрати середовище розробки;

− провести необхідні розрахунки;

− оформити кваліфікаційну роботу за стандартами ДСТУ 3008:2015 [3].

1.5 Висновки до 1 розділу

У першому розділі проведено комплексний аналіз предметної області автономної навігації мобільних роботів на складських комплексах. Дослідження існуючих робототехнічних систем виявило, що більшість комерційних рішень базується на дорогому обладнанні (лазерні сканери

вартістю від 500$ до 8000$) або потребує значних інвестицій у спеціальну інфраструктуру (магнітні стрічки, рефлектори, підлогові маркери). Це створює високий економічний бар’єр для впровадження автоматизації на малих та середніх складах, що обмежує поширення робототехнічних технологій у логістичній галузі.

Огляд сучасних методів навігації показав, що візуальна навігація на основі інформаційних позначок представляє перспективний напрямок, поєднуючи високу точність абсолютного позиціонування (від 2 мм до 10 мм) з мінімальною вартістю маркерів (від 0,05 $ до 0,5 $). Водночас, аналіз типів візуальних маркерів та методів їх розпізнавання виявив фундаментальну проблему: класичні алгоритми виявлення фідуціальних маркерів (ArUco, AprilTag) на базі OpenCV неможливо реалізувати на недорогих мікроконтролерних платформах через високі вимоги до обчислювальних ресурсів та пам’яті. Це стримувало розвиток економічно ефективних систем навігації на основі комп’ютерного зору.

Встановлено, що сучасні досягнення у галузі машинного навчання, зокрема розробка легких нейронних мереж, оптимізованих для роботи на обмежених ресурсах, відкривають нові можливості для реалізації інтелектуальної обробки зображень на ultra-low-cost платформах. Гібридний підхід, що поєднує машинне навчання для виявлення маркерів з класичними геометричними методами для точного визначення просторового положення, дозволяє створити систему навігації на базі ESP32-CAM (вартістю від 3 $ до 5 $) з характеристиками, достатніми для практичного застосування у складських умовах.

На основі проведеного аналізу сформульовано завдання на розробку системи розпізнавання інформаційних позначок для автономної навігації мобільного робота. Розроблювана система має потенціал знизити вартість навігаційного обладнання на 2-3 порядки порівняно з існуючими рішеннями, що робить автоматизацію складів доступною для ширшого кола підприємств і становить значний науково-практичний інтерес для розвитку edge AI у робототехніці.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ СИСТЕМИ РОЗПІЗНАВАННЯ

На основі проведеного у першому розділі аналізу предметної області та сформульованого завдання, далі визначено конкретні технічні вимоги до системи, обґрунтовано вибір апаратної платформи та типу візуальних маркерів, розроблена архітектура програмного забезпечення та детальні алгоритми обробки зображень.

Ключовим викликом при розробці є забезпечення балансу між обчислювальною складністю алгоритмів та обмеженими ресурсами мікроконтролерної платформи ESP32. Це вимагає системного підходу до оптимізації на всіх рівнях.

Середовищем роботи мобільного робота є складські приміщення з прямолінійними проходами, стелажами, перехрестями. Освітлення від 100 люкс до 1000 люкс, люмінесцентні/LED-світильники (можливий мерехтливий фон). Дистанція спостереження маркерів сягає від 0,5 метрів до 3 метрів. Швидкість мобільної платформи становить від 0 м/с до 0,1 м/с. Вібрації низькі/середні при чому можливий motion blur при довгій витримці. Також допускається забруднення у вигляді пилу та/або відблиски від глянцевих поверхонь.

2.1 Визначення вимог до системи розпізнавання

Формування вимог до системи розпізнавання є критичним етапом розробки, який визначає технічні рішення на всіх подальших стадіях проектування [11]. Вимоги мають бути конкретними, вимірюваними, досяжними, релевантними та обмеженими у часі (SMART-критерії), а також узгодженими між собою без внутрішніх протиріч [16].

2.1.1 Системний підхід до формування вимог

Система розпізнавання інформаційних позначок функціонує як частина більшої системи автономної навігації мобільного робота і має забезпечувати входи для підсистем локалізації, планування шляху та управління рухом. Це визначає системний контекст формування вимог.

Основна функція системи – надання навігаційної інформації – декомпозується на наступні підфункції [17]:

– сприйняття – захоплення візуальної інформації про оточення;

– детекція – виявлення присутності маркерів на зображенні;

– ідентифікація – визначення унікального ідентифікатора маркера;

– локалізація – обчислення просторового положення відносно маркера;

– інтерпретація – перетворення даних у навігаційні параметри;

– комунікація – передача інформації до контролера робота.

Кожна з цих підфункцій має специфічні вимоги, які заздалегідь визначено далі.

Зовнішні інтерфейси системи:

– вхід – візуальна інформація (потік зображень з камери);

– вихід – навігаційні дані (ID маркера, відстань, кут, координати) або команди керування;

– управління – команди конфігурації, запити статусу;

– живлення – 5 В постійного струму, до 400 мА.

Операційне середовище:

– приміщення складу з контрольованим мікрокліматом;

– температура від +5 °C до +30 °C;

– відносна вологість від 20 % до 80 % (без конденсації);

– освітлення від 100 люкс до 1000 люкс (змішане штучне та природне);

– вібрації низькі-середні (рух робота по рівній підлозі).

2.1.2 Вимоги до апаратного забезпечення

Основною обчислювальною платформою є ESP32-CAM через унікальне поєднання характеристик:

– вартість модуля від 3 $ до 5 $, що на 2−3 порядки нижче альтернатив у вигляді RaspberryPi та камера 50 $, чи NVIDIA Jetson Nano вартістю близько 150 $;

– штатна камера OV2640 з достатньою роздільністю (2 Мп) усуває необхідність зовнішніх модулів;

– обчислювальна потужність Dual-core Xtensa LX6 240 МГц забезпечує можливість використання легких ML-моделей;

– низьке енергоспоживання від 150 мА до 300 мА при 5 В (від 0,75 Вт до 1,5 Вт), що критично для автономних роботів;

– компактні розміри дозволяють інтеграцію у обмежений простір.

2.1.3 Функціональні вимоги до системи

Функціональні вимоги визначають конкретні функції та поведінку системи – що саме система повинна робити для досягнення поставлених цілей [18]. На відміну від нефункціональних вимог, які описують якісні характеристики роботи, функціональні вимоги специфікують конкретні операції, трансформації даних та взаємодії з користувачем чи іншими системами. Для системи розпізнавання інформаційних позначок функціональні вимоги випливають безпосередньо з декомпозиції основної функції та покривають усі етапи обробки – від захоплення зображення до передачі навігаційних даних.

Нижче наведено ключові функціональні вимоги, згруповані за основними підфункціями системи.

Виявлення візуальних маркерів:

– система повинна виявляти маркери на зображенні з камери ESP32-CAM;

– виявлення маркерів при частковому перекритті (до 5 %);

– робота за різних умов освітлення (від 100 люкс до 1000 люкс).

Ідентифікація маркерів:

– однозначне визначення унікального ідентифікатора кожного маркера;

– підтримка до 50 унікальних маркерів.

Визначення просторового положення, а саме обчислення відстані до маркера з похибкою до 10 %.

Обробка в реальному часі:

– забезпечення частоти обробки мінімум 2 кадри на секунду;

– затримка від захоплення зображення до отримання результату < 1 секунди.

Передача навігаційних даних:

– формування навігаційної інформації (позиція, ID маркера);

– передача даних до контролера робота.

Діагностика та моніторинг:

– індикація стану системи;

– вивід діагностичної інформації (ідентифікатор виявленого маркера, час обробки).

2.1.4 Нефункціональні вимоги до системи

Нефункціональні вимоги (якісні вимоги або обмеження) визначають критерії оцінки якості функціонування системи та накладають обмеження на процес розробки [18]. Вони характеризують не що система робить, а як вона це робить – продуктивність, надійність, масштабованість, зручність супроводу тощо. Для систем реального часу, що працюють на ресурсо-обмежених платформах, нефункціональні вимоги часто є більш критичними для успіху проекту, ніж функціональні, оскільки визначають практичну придатність рішення.

Для розроблюваної системи визначено наступні категорії нефункціональних вимог.

Продуктивність:

– загальний час обробки одного кадру ≤ 800 мс;

– частота оновлення навігаційних даних ≥ 1 Гц.

Точність:

– точність детекції маркера (precision) ≥ 85 %;

– повнота детекції (recall) ≥ 75 %;

– похибка визначення відстані ≤ 10 %.

Надійність:

– ймовірність помилкової детекції (false positive) < 10 %;

– ймовірність пропуску маркера (false negative) < 15 % за нормальних умов;

– безперервна робота без збоїв ≥ 8 годин.

Масштабованість:

– можливість додавання нових типів маркерів без повного переписування коду;

– підтримка розширення кількості унікальних ID до 200.

Економічність:

– вартість апаратної частини ≤ 10 $ (ESP32-CAM + компоненти);

– вартість одного маркера ≤ 0,5 $ (друк + ламінування);

– загальна вартість системи на порядки нижча за існуючі рішення.

Зручність впровадження:

– простота виготовлення маркерів (друк на звичайному принтері з подальшим ламінуванням);

– мінімальні вимоги до кваліфікації персоналу для встановлення;

– час розгортання системи на складі від 100 м^2 до 500 м^2 ≤ 1 робочого дня.

2.2 Розробка структури системи розпізнавання інформаційних позначок

Розробка структури системи розпізнавання є ключовим етапом проектування, який визначає взаємодію апаратних та програмних компонентів, потоки даних між модулями та загальну архітектуру рішення. Правильно спроектована структура забезпечує модульність, можливість налагодження та оптимізації окремих компонентів, а також масштабованість системи для подальшого розвитку [19, 20]. У контексті розробки на обмежених ресурсах мікроконтролера ESP32, структурна організація набуває особливого значення, оскільки від неї залежить ефективність використання критично обмеженої оперативної пам'яті та обчислювальної потужності.

2.2.1 Концептуальна модель та загальна архітектура системи

Система розпізнавання інформаційних позначок для автономної навігації мобільного робота розроблена за принципом багаторівневої ієрархічної архітектури. Такий підхід дозволяє розділити складну задачу обробки візуальної інформації та формування навігаційних рішень на логічно незалежні рівні, кожен з яких має чітко визначену відповідальність та інтерфейси взаємодії з сусідніми рівнями. Багаторівнева організація є класичним архітектурним патерном у розробці вбудованих систем, що забезпечує можливість незалежної розробки та тестування окремих компонентів, а також спрощує локалізацію та усунення помилок.

На найнижчому рівні архітектури знаходиться апаратний шар, що включає фізичні компоненти системи. Центральним елементом є мікроконтролер ESP32 з двоядерною архітектурою Xtensa LX6, що працює на частоті 240 МГц. Вибір саме цього мікроконтролера обумовлений унікальним поєднанням достатньої обчислювальної потужності для виконання операцій машинного навчання, наявністю 520 KB оперативної пам'яті (що є мінімально необхідним для розміщення моделі MobileNet та буферів обробки), вбудованих

інтерфейсів WiFi та Bluetooth для комунікації, а також надзвичайно низької вартості – від 3 $ до 5 $ за повний модуль ESP32-CAM з інтегрованою камерою.

Камера OV2640, що є невід'ємною частиною модуля ESP32-CAM, підключається до мікроконтролера через спеціалізований Digital Video Port (DVP) інтерфейс, що забезпечує високошвидкісну передачу потоку пікселів безпосередньо у внутрішню пам'ять ESP32 через Direct Memory Access (DMA), мінімізуючи навантаження на процесорні ядра. Ця камера здатна захоплювати зображення з роздільністю до 1600 × 1200 пікселів (UXGA, 2 мегапікселі), що є більш ніж достатнім для детекції візуальних маркерів. Однак для оптимізації швидкості обробки система використовує нижчі роздільності для первинного захоплення з подальшим зменшенням до 96 × 96 пікселів для подачі на вхід нейронної мережі.

Над апаратним шаром розташовується рівень абстракції апаратури (HAL), що включає драйвери для роботи з кожним апаратним компонентом. Драйвер камери інкапсулює всю складність конфігурації регістрів OV2640, налаштування параметрів захоплення (роздільність, формат пікселів, частота кадрів), управління автоматичною експозицією та балансом білого. Він надає вищим рівням простий інтерфейс у вигляді функції «захопити кадр», приховуючи деталі низькорівневої взаємодії з сенсором. Аналогічно організовані драйвери UART для послідовного зв'язку, WiFi для бездротової комунікації, GPIO для керування індикаційними світлодіодами та flash-підсвіткою. Такий підхід забезпечує портативність коду – при необхідності заміни апаратної платформи потрібно буде переписати тільки драйвери, залишивши незмінною логіку вищих рівнів.

Наступний рівень архітектури відповідає за обробку зображень. Цей рівень виконує критично важливі функції перетворення raw даних з камери у формат, придатний для подачі на вхід нейронної мережі. Модуль захоплення та буферизації отримує від драйвера камери масив пікселів, зазвичай у форматі RGB565 (де кожен піксель кодується 16 бітами: 5 біт для червоного каналу, 6 для зеленого, 5 для синього), та розміщує його у попередньо виділеній області

пам'яті. Враховуючи обмежену оперативну пам'ять ESP32, критично важливо уникати зайвих копіювань даних – система працює з покажчиками на буфери, передаючи їх між модулями замість створення дублікатів.

Модуль масштабування виконує зменшення зображення з початкової роздільності (наприклад, 640 × 480 = 307,200 пікселів) до цільової 96 × 96 (9,216 пікселів) – редукцію більш ніж у 33 рази. Це радикальне зменшення обсягу даних є необхідним компромісом: з одного боку, воно дозволяє зменшити кількість обчислень у нейронній мережі приблизно у тисячу разів (оскільки обчислювальна складність згорткових шарів пропорційна квадрату лінійного розміру зображення), з іншого – втрачається частина деталей зображення. Однак для задачі детекції маркерів, які мають чітку геометричну структуру з високим контрастом чорних та білих областей, роздільність 96 × 96 пікселів виявляється достатньою – маркер розміром 200 мм на відстані 2 метри проектується як область приблизно від 30 пікселів до 40 пікселів на такому зображенні, чого цілком достатньо для надійного розпізнавання нейронною мережею.

Алгоритм масштабування використовує метод білінійної інтерполяції, що забезпечує плавність переходів і зберігає більше деталей порівняно з простішим методом найближчого сусіда. При білінейній інтерполяції значення кожного пікселя вихідного зображення обчислюється як зважене середнє чотирьох найближчих пікселів вхідного зображення, де ваги визначаються відстанями до цих пікселів. Хоча цей метод дещо повільніший за найпростіший nearest neighbor (потребує чотирьох читань з пам'яті та трьох операцій множення на кожен вихідний піксель), він забезпечує значно кращу якість зображення, особливо при великих коефіцієнтах зменшення, що позитивно впливає на точність розпізнавання.

Модуль нормалізації підготовлює дані для нейронної мережі, виконуючи перетворення діапазону значень пікселів. Камера видає значення яскравості у діапазоні 0-255 (8 біт на канал), тоді як квантизована int8 модель очікує на вході значення у діапазоні від -128 до +127. Процес нормалізації включає

конвертацію формату з RGB565 у RGB888, після чого застосовується лінійне перетворення: normalized_value = (pixel_value - 128). Така проста операція центрує розподіл значень навколо нуля, що відповідає очікуванням квантизованої моделі і забезпечує оптимальну точність обчислень у цілочисельній арифметиці.

Центральним елементом всієї системи є рівень машинного навчання, що виконує безпосередньо розпізнавання візуальних маркерів. Цей рівень побудований на основі фреймворку TensorFlow Lite Micro – спеціалізованої версії популярної бібліотеки машинного навчання TensorFlow, оптимізованої для роботи на мікроконтролерах з екстремально обмеженими ресурсами. На відміну від повноцінного TensorFlow або навіть TensorFlow Lite для мобільних пристроїв, TensorFlow Lite Micro не має залежностей від операційної системи, динамічного виділення пам'яті у стандартному розумінні, та використовує статичну пам'ять для всіх операцій. Розмір самої бібліотеки становить лише близько від 20 КВ до 30 КВ коду, що дозволяє розміщувати її навіть на мікроконтролерах з дуже обмеженою Flash-пам'яттю.

У серці рівня машинного навчання знаходиться модель MobileNet v2 – одна з найбільш ефективних архітектур згорткових нейронних мереж, спеціально розроблена командою Google для застосувань на мобільних пристроях та вбудованих системах. Ключова інновація MobileNet полягає у використанні так званих depthwise separable convolutions – техніки, що розділяє стандартну згорткову операцію на два послідовних етапи: depthwise convolution (обробка кожного кольорового каналу окремо) та pointwise convolution (комбінування інформації між каналами через згортку 1 × 1). Таке розділення зменшує кількість обчислень приблизно у 8-9 разів порівняно зі стандартними згортками при незначній втраті точності.

MobileNet v2 додатково вводить концепцію inverted residual blocks – архітектурних блоків, де спочатку відбувається розширення кількості каналів (expansion) у 6 разів через pointwise convolution, потім застосовується depthwise convolution для обробки просторових ознак, і нарешті виконується стиснення

(projection) назад до меншої кількості каналів. Такий підхід виявляється дуже ефективним: розширення дозволяє зберегти більше інформації під час нелінійних перетворень, а стиснення зменшує обсяг даних для передачі до наступного блоку.

Для адаптації MobileNet v2 до обмежених ресурсів ESP32 використовується параметр width multiplier (α), що пропорційно зменшує кількість каналів у всіх шарах мережі. При α=0,5 (обраному для даного проекту) кількість каналів зменшується вдвічі порівняно зі стандартною моделлю, що призводить до зменшення кількості обчислень приблизно у чотири рази (оскільки кількість множень у згортках пропорційна квадрату кількості каналів). Це дозволяє досягти швидкості inference від 400 мс до 600 мс на ESP32, що є прийнятним для навігаційних застосувань.

Критично важливою для можливості виконання моделі на ESP32 є квантизація – процес перетворення моделі з float32 (де кожна вага та активація зберігається як 32-бітне число з плаваючою комою) у int8 (8-бітні цілі числа). Квантизація зменшує розмір моделі у чотири рази (з ~1,6 MB до ~400 KB для обраної конфігурації) та значно прискорює обчислення, оскільки цілочисельна арифметика виконується на ESP32 набагато швидше за операції з плаваючою комою – мікроконтролер не має апаратного Floating Point Unit (FPU), тому всі float операції емулюються програмно. Сучасні методи квантизації, такі як quantization-aware training, дозволяють зберегти точність моделі з деградацією від 2 % до 3 % порівняно з оригінальною float32 версією.

Модель завантажується з Flash-пам'яті при старті системи. Вона зберігається у форматі FlatBuffers – компактному бінарному форматі серіалізації даних, що не потребує десеріалізації перед використанням. TensorFlow Lite Micro може читати модель безпосередньо з Flash через memory-mapped доступ, не копіюючи всі дані у RAM, що критично важливо для економії оперативної пам'яті. Для виконання обчислень модель потребує виділення так званої arena – безперервної області пам'яті, де зберігатимуться

проміжні результати обчислень (активації, тензори). Розмір arena визначається експериментально і для обраної конфігурації становить приблизно 200 KB.

Після виконання нейронної мережі на виході отримується вектор ймовірностей для кожного з класів – типово від 10 класів до 20 класів, що відповідають різним ID маркерів плюс додатковий клас «background» (фон, відсутність маркера). Модуль постпроцесингу аналізує цей вектор, знаходить клас з максимальною ймовірністю (операція argmax) та порівнює цю ймовірність з порогом впевненості, встановленим на рівні 0,7 (70 %). Якщо максимальна ймовірність нижча за поріг, система приймає рішення, що на зображенні немає впевненої детекції маркера – можливо, маркер відсутній у полі зору, частково перекритий, або умови спостереження (освітлення, кут, відстань) не дозволяють надійно його розпізнати. У такому випадку результат ігнорується, і система продовжує обробку наступних кадрів.

Якщо впевненість детекції перевищує поріг, управління передається на рівень навігаційної логіки. Цей рівень відповідає за перетворення результату класифікації (ідентифікатора маркера) у практично корисні навігаційні параметри.

Маючи координати кутів маркера на зображенні, фізичний розмір маркера (відомий заздалегідь, наприклад 200 мм × 200 мм), та параметри калібрування камери (фокусна відстань, оптичний центр, коефіцієнти дисторсії), PnP алгоритм обчислює матрицю обертання R (rotation) та вектор переміщення t (translation), що описують трансформацію від системи координат маркера до системи координат камери. З вектору t безпосередньо витікає відстань до маркера – це просто довжина цього вектору, що обчислюється за формулою (2.1):

$$distance = \sqrt{(tx^2 + ty^2 + tz^2)}. \quad (2.1)$$

Кут спостереження обчислюється через арктангенс компонентів вектору за формулою (2.2):

$$\text{angle} = 2 \cdot \tan^{-1}(\text{ty}, \text{tx}). \quad (2.2)$$

Для навігації робота потрібна інформація не про положення камери відносно маркера, а про положення самого робота. Це потребує додаткової трансформації координат, що враховує геометрію монтажу камери на роботі – зміщення та поворот камери відносно центру робота. Ці параметри вимірюються один раз при встановленні камери та зберігаються у конфігурації. Застосовуючи послідовність трансформацій (камера → робот → глобальна система координат складу), система визначає, де саме на складі знаходиться робот у даний момент.

2.2.2 Структурна схема апаратно-програмного комплексу

Для наочного представлення взаємодії компонентів системи розроблена детальна структурна схема, що показує апаратні модулі, програмні компоненти та потоки даних між ними. Схема організована у вигляді двох основних блоків – апаратного рівня та програмного рівня, з чітким позначенням інтерфейсів взаємодії.

На рисунку 2.1 представлена ця структурна схема. У верхній частині схеми показаний апаратний рівень, де центральне місце займає модуль ESP32 з двоядерним процесором Xtensa LX6. До процесора підключена камера OV2640 через паралельний інтерфейс DVP, що забезпечує високу швидкість передачі потоку пікселів – до 24 Мбіт/с, достатньо для передачі VGA зображення (640 × 480 × 16 біт) з частотою від 30 кадрів до 60 кадрів на секунду.

Нижня частина схеми деталізує програмний рівень, організований як послідовний pipeline обробки даних. Кожен модуль виконує специфічну функцію трансформації даних: від raw пікселів з камери до структурованих навігаційних параметрів. Стрілки між модулями показують напрямок потоку даних.

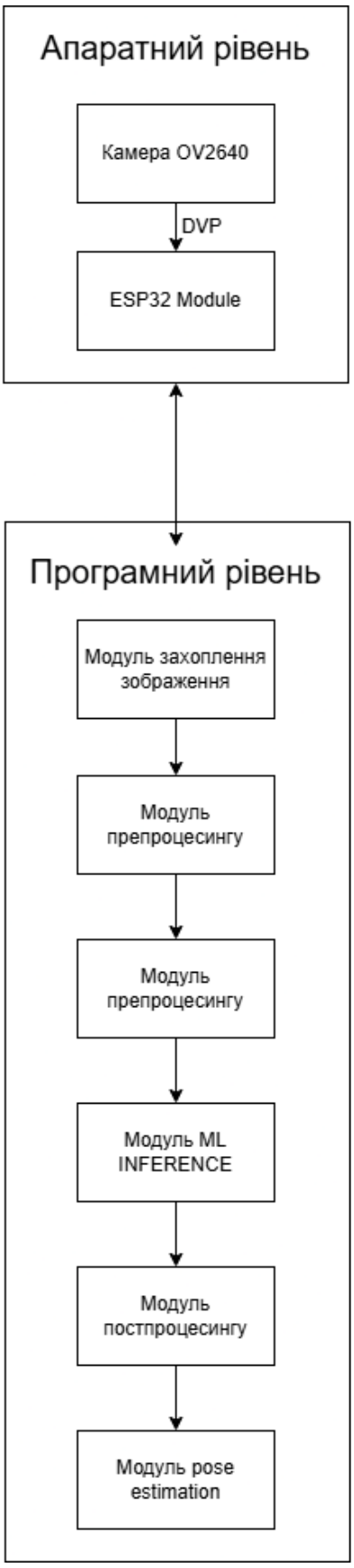

Рисунок 2.1 – Структурна схема системи розпізнавання інформаційних позначок

2.2.3 Організація пам'яті та потоки даних

Ефективне використання обмеженої пам'яті ESP32 є критичним фактором успіху всього проекту. Мікроконтролер має всього 520 KB SRAM, з яких близько 320 KB для додатку – досить скромний об'єм, враховуючи, що потрібно розмістити буфери зображень, тензори нейронної мережі та всі проміжні дані обробки.

Після масштабування зображення до 96 × 96 пікселів, його розмір зменшується до 96 × 96 × 3 = 27648 байт (для RGB888) – це вже поміститься у SRAM. Цей масштабований кадр використовується як вхід для нейронної мережі. TensorFlow Lite Micro отримує покажчик на цей буфер і інтерпретує його як input tensor відповідної форми (96, 96, 3) з типом даних int8.

Для виконання обчислень нейронна мережа потребує додаткової пам'яті – так званої arena. У цій області розміщуються всі проміжні тензори (активації кожного шару мережі), а також деякі допоміжні структури даних інтерпретатора. Розмір arena не можна точно передбачити аналітично – він залежить від архітектури мережі, кількості та розмірів шарів, стратегій перевикористання пам'яті, що застосовує TensorFlow Lite Micro. Типовий підхід – виділити достатньо велику область (наприклад, 300 KB), запустити inference з логуванням фактично використаної пам'яті, після чого зменшити розмір arena до мінімально необхідного з невеликим запасом. Для MobileNet v2 з α=0,5 та входом 96 × 96 експериментально визначений розмір arena становить близько 200 KB.

2.3 Розробка алгоритму розпізнавання інформаційних позначок

Розпізнавання інформаційних позначок є ключовою функцією системи, що безпосередньо визначає її ефективність та надійність у навігаційних задачах. Алгоритм розпізнавання повинен забезпечувати високу точність детекції маркерів за різних умов освітлення, кутів спостереження та відстаней, одночасно працюючи в реальному часі на обмежених обчислювальних ресурсах

ESP32. У цьому підрозділі детально розглядається розроблений алгоритм, що базується на поєднанні методів глибокого навчання та класичного комп'ютерного зору.

2.3.1 Обґрунтування вибору підходу до розпізнавання

Перед розробкою конкретного алгоритму необхідно обґрунтувати вибір підходу до розпізнавання візуальних маркерів. Історично у комп'ютерному зорі для детекції та розпізнавання об'єктів використовувалися класичні методи, базовані на ручному проектуванні ознак (hand-crafted features) та геометричних перетвореннях зображень. Для задачі розпізнавання фідуціальних маркерів типу ArUco розроблені спеціалізовані алгоритми, що ефективно працюють у загальному випадку.

Класичний підхід до детекції ArUco маркерів включає наступні етапи:

– адаптивна бінаризація зображення для відокремлення чорних та білих областей;

– пошук контурів методом border following;

– фільтрація контурів за геометричними критеріями (приблизно квадратна форма, достатній розмір, наявність вкладеного контуру для рамки маркера);

– перспективна корекція кандидатів до канонічного вигляду;

– декодування внутрішнього бінарного патерну;

– перевірка за словником відомих маркерів.

Цей підхід реалізований у бібліотеці OpenCV та демонструє відмінні результати на широкому спектрі застосувань.

Однак застосування класичного підходу на платформі ESP32 стикається з фундаментальними обмеженнями. По-перше, бібліотека OpenCV є досить об'ємною – навіть мінімальна збірка займає кілька мегабайт коду та вимагає динамічної пам'яті для внутрішніх структур даних, що перевищує доступні 320 KB SRAM. По-друге, операції пошуку контурів та їх аналізу є обчислювально складними, особливо для зображень з високим рівнем деталізації або складним

фоном – час обробки може досягати сотень мілісекунд навіть на більш потужних процесорах. По-третє, класичні алгоритми чутливі до параметрів: поріг бінаризації потрібно підбирати під конкретні умови освітлення, що ускладнює створення універсального рішення.

Альтернативний підхід на основі машинного навчання використовує нейронні мережі для безпосередньої детекції та класифікації маркерів на зображенні. Згорткові нейронні мережі (CNN) природно навчаються ієрархії ознак від простих (краї, кути) до складних (форми, текстури, об'єкти), що робить їх надзвичайно ефективними для задач комп'ютерного зору. Критично важливо, що сучасні архітектури, оптимізовані для мобільних пристроїв (MobileNet, EfficientNet, SqueezeNet), досягають високої точності при радикально меншій обчислювальній складності порівняно з класичними мережами типу VGG або ResNet.

Ключова перевага ML-підходу для обмежених ресурсів полягає у передбачуваній обчислювальній складності. Нейронна мережа виконує фіксовану кількість операцій незалежно від складності зображення – кількість множень та додавань визначається архітектурою мережі та розміром входу. Для MobileNet v2 з $\alpha = 0{,}5$ та входом 96×96 це приблизно 150 мільйонів операцій, що виконуються за час від 400 мс до 600 мс на ESP32. На відміну від цього, класичні алгоритми мають складність, що залежить від вмісту зображення: на складному фоні з безліччю контурів час обробки може зростати непередбачувано.

Друга критична перевага – робастність до варіацій умов спостереження. Нейронна мережа, навчена на різноманітному датасеті, що включає зображення маркерів при різному освітленні (від яскравого до тьмяного), різних кутах спостереження (від 0° до 45°), часткових перекриттях, розмиттях руху, автоматично вчиться інваріантності до цих факторів. Класичні алгоритми потребують явного програмування логіки обробки кожного випадку або складних евристик для адаптації параметрів.

Третя перевага – простота адаптації до специфіки конкретного складу. Якщо на складі є специфічні умови (наприклад, особливий спектр освітлення від жовтих натрієвих ламп, або часте часткове перекриття маркерів коробками), можна дотренувати модель (fine-tuning) на невеликому наборі зображень з цього конкретного середовища, значно покращивши точність. Адаптація класичного алгоритму потребувала б програмування додаткової логіки.

Важливо відзначити, що ML-підхід не виключає повністю використання класичних методів комп'ютерного зору. Оптимальна стратегія – гібридний алгоритм, де нейронна мережа виконує важку роботу детекції та грубої класифікації, а класичні геометричні методи використовуються для уточнення точного положення кутів маркера, необхідного для pose estimation. Такий підхід поєднує переваги обох світів: робастність та адаптивність ML з точністю та передбачуваністю геометричних методів.

Враховуючи наведені аргументи, для системи розпізнавання на ESP32-CAM обрано підхід на основі MobileNet v2 для детекції та класифікації маркерів, доповнений опціональним модулем уточнення положення класичними методами для застосувань, що потребують високої точності pose estimation.

2.3.2 Загальна структура алгоритму розпізнавання

Розроблений алгоритм розпізнавання інформаційних позначок складається з послідовності етапів, кожен з яких виконує специфічну трансформацію даних. Алгоритм можна розділити на три основні фази: препроцесинг (підготовка зображення), inference (виконання нейронної мережі) та постпроцесинг (обробка результатів).

На першій фазі – препроцесинг зображення – raw дані з камери перетворюються у формат, придатний для подачі на вхід нейронної мережі. Включає захоплення кадру, масштабування, конвертацію формату пікселів та нормалізацію значень.

Друга фаза – ML Inference – передбачає подачу підготовленого зображення на вхід моделі MobileNet v2, що виконує класифікацію – визначає, який саме маркер (якщо взагалі є) присутній на зображенні, та оцінює впевненість у цьому рішенні.

На третій фазі – постпроцесинг та формування результату – вихід нейронної мережі інтерпретується, перевіряється на достовірність, та формуються навігаційні параметри для передачі контролеру робота.

Для уточнення положення виконується додаткова обробка для визначення координат кутів маркера та обчислення відстані від маркера до камери.

Далі розглянуто детально кожну фазу алгоритму.

2.3.3 Препроцесинг зображення

Препроцесинг є критично важливим етапом, що безпосередньо впливає на якість розпізнавання. Неправильна підготовка даних може повністю нівелювати можливості навіть найкращої нейронної мережі.

Процес починається з команди камері OV2640 на захоплення кадру. Захоплення виконується через драйвер камери.

Захоплений кадр 640 × 480 потрібно зменшити до 96 × 96 пікселів – цільової роздільності для MobileNet. Це зменшення у 6,67 рази по кожній стороні (у ~ 44 рази за площею) радикально скорочує обсяг даних та обчислювальну складність подальшої обробки.

Використовується алгоритм білінійної інтерполяції, що забезпечує баланс між якістю та швидкістю. Для кожного пікселя вихідного зображення (x_out, y_out) обчислюються відповідні координати у вхідному зображенні:

```
x_in = x_out × (width_in / width_out);
y_in = y_out × (height_in / height_out);.
```

Після масштабування зображення представлене у форматі RGB565. Для подачі у нейронну мережу потрібен формат RGB888 (або точніше, три окремі канали R, G, B по 8 біт кожен). Конвертація виконується побітовими операціями, що наведені на рисунку 2.2.

```
uint16_t pixel_rgb565 = ...;  // 16-бітний піксель

// Розпакування каналів
uint8_t r5 = (pixel_rgb565 >> 11) & 0x1F;  // 5 старших біт
uint8_t g6 = (pixel_rgb565 >> 5) & 0x3F;   // 6 середніх біт
uint8_t b5 = pixel_rgb565 & 0x1F;          // 5 молодших біт

// Розширення до 8 біт (масштабування)
uint8_t r8 = (r5 << 3) | (r5 >> 2);  // 5 біт - 8 біт
uint8_t g8 = (g6 << 2) | (g6 >> 4);  // 6 біт - 8 біт
uint8_t b8 = (b5 << 3) | (b5 >> 2);  // 5 біт - 8 біт
```

Рисунок 2.2 – Фрагмент коду конвертації зображення

Операції побітового зсуву та логічного АБО виконуються дуже швидко – по одному такту процесора на операцію, тому конвертація всього зображення 96×96 займає лише кілька мілісекунд.

Фінальний крок препроцесингу – нормалізація значень пікселів до діапазону, очікуваного квантизованою int8 моделлю. Стандартний діапазон uint8 значень яскравості [0, 255] перетворюється у int8 діапазон [-128, +127]:

```
int8_t normalized = (int8_t)(pixel_uint8 - 128);.
```

Ця проста операція центрує розподіл значень навколо нуля, що відповідає внутрішнім очікуванням квантизованої моделі та забезпечує оптимальну точність int8 обчислень.

Результуюче зображення записується у input tensor TensorFlow Lite Micro. Тензор має форму (1, 96, 96, 3), де 1 – розмір batch (обробляється одне зображення за раз), 96 × 96 – просторові розміри, 3 – кількість каналів RGB. Дані можуть зберігатися у пам'яті у різних форматах, залежно від вимог моделі. MobileNet зазвичай використовує NHWC, що є більш природним для обробки на CPU.

2.3.4 Виконання ML Inference

Після підготовки input tensor виконується inference нейронної мережі – найбільш обчислювально інтенсивна частина алгоритму.

MobileNet v2 складається з послідовності інвертованих залишкових блоків (inverted residual blocks). Кожен такий блок виконує три послідовні операції:

– expansion – розширення вхідних каналів у 6 разів через pointwise convolution (згортка 1 × 1). Наприклад, якщо вхід має 24 канали, після expansion буде 144 канали. Це розширення дозволяє збільшити простір ознак для більш виразного представлення;

– depthwise convolution – застосування просторової згортки 3 × 3 до кожного з розширених каналів окремо. На відміну від звичайної згортки, де кожен вихідний канал є комбінацією всіх вхідних, тут кожен канал обробляється незалежно. Це радикально зменшує кількість параметрів та обчислень;

– projection – стиснення розширених каналів назад до меншої кількості через pointwise convolution 1 × 1 з лінійною активацією без Rectified Linear Unit (ReLU). ReLU – це найпопулярніша функція активації в глибокому навчанні, яка повертає вхідне значення, якщо воно позитивне, і нуль, якщо воно негативне. Ця проекція зменшує розмірність назад;

– residual connection (залишкове з'єднання). Якщо вхід та вихід блоку мають однаковий розмір, додається skip connection – вихід додається до входу (element-wise addition).

Перший шар мережі – стандартна згортка 3×3, що зменшує просторовий розмір з 96 × 96 до 48 × 48 та збільшує кількість каналів до 16 (з урахуванням $\alpha = 0{,}5$). Далі йдуть 16-17 інвертованих залишкових блоків. Фінальні шари мають просторовий розмір лише 3 × 3 або навіть 1 × 1 після global average pooling, але велику кількість каналів (сотні), що кодують високорівневі семантичні ознаки.

Останній шар – fully connected (повнозв'язний) з кількістю нейронів, що дорівнює кількості класів (кількість різних маркерів плюс клас background). Вихід цього шару – логіти (logits), нормалізовані сирі оцінки для кожного класу. Застосування функції softmax перетворює логіти у ймовірності.

Однак у квантизованій моделі softmax часто не включається у граф обчислень, оскільки для вибору найбільш ймовірного класу (argmax) достатньо порівняти самі логіти – клас з максимальним логітом буде мати максимальну ймовірність. Це економить обчислення експоненти, що є дорогою операцією.

Після того, як input tensor заповнений нормалізованими даними зображення, викликається метод invoke() інтерпретатора TensorFlow Lite Micro.

Під час виконання Invoke() відбувається послідовне обчислення всіх шарів мережі у порядку, визначеному графом обчислень моделі. Для кожного шару викликається відповідний kernel (функція обчислення), що реалізує конкретну операцію (Conv2D, DepthwiseConv2D, Add, ReLU6, тощо). Ці kernels оптимізовані для роботи з int8 даними та використовують vectorized операції де можливо.

Проміжні результати (активації) кожного шару зберігаються у попередньо виділеній arena. TensorFlow Lite Micro використовує стратегію in-place операцій та перевикористання пам'яті – коли тензор більше не потрібний (всі операції, що його використовують, вже виконані), його пам'ять може бути перевикористана для наступних тензорів. Це дозволяє мінімізувати розмір arena.

Час виконання inference залежить від багатьох факторів: архітектури мережі (кількість шарів та операцій), розміру входу, якості оптимізації kernels для конкретного процесора. Для MobileNet v2 з $\alpha = 0{,}5$ та входом 96×96 на ESP32 при 240 MHz типовий час становить 400-600 мс. Це приблизно 250,000-400,000 операцій на мілісекунду, що є непоганим показником для мікроконтролера без апаратного прискорення ML.

Після завершення invoke() вихідний тензор містить результат класифікації. Доступ до вихідного тензора відбувається за допомогою фрагмента коду, що наведений на рисунку 2.3.

```
TfLiteTensor* output = interpreter->output(0);
int8_t* output_data = output->data.int8;
int output_size = output->dims->data[1];  // кількість класів
```

Рисунок 2.3 – Фрагмент коду доступу до вихідного тензора

Вихідний тензор має форму (1, N), де N – кількість класів. Кожен елемент представляє логіт (або ймовірність після softmax) відповідного класу.

2.3.5 Постпроцесинг результатів

Обробка виходу нейронної мережі перетворює сирі логіти у практично корисну інформацію – ідентифікатор маркера та впевненість у детекції.

Якщо модель квантизована int8, вихідні логіти також представлені у int8 форматі з певним масштабуванням. Для інтерпретації їх часто потрібно деквантизувати назад у float за допомогою фрагменту коду, що наведений на рисунку 2.4.

```
float scale = output->params.scale;
int32_t zero_point = output->params.zero_point;

for (int i = 0; i < output_size; i++) {
  float logit = (output_data[i] - zero_point) * scale;
  logits_float[i] = logit;
}
```

Рисунок 2.4 – Фрагмент коду деквантизування логітів

Параметри scale та zero_point зберігаються у метаданих тензора і визначаються під час квантизації моделі.

Клас з максимальним логітом є результатом класифікації з використанням коду, що наведено на рисунку 2.5.

```
int max_index = 0;
float max_value = logits_float[0];

for (int i = 1; i < output_size; i++) {
  if (logits_float[i] > max_value) {
      max_value = logits_float[i];
      max_index = i;
  }
}

int predicted_class = max_index;;
}
```

Рисунок 2.5 – Фрагмент коду знаходження максимального логіту

Індекс predicted_class відповідає конкретному маркеру. Мапінг індексів на ID маркерів визначається під час навчання моделі.

Для оцінки надійності результату обчислюється впевненість – ймовірність передбаченого класу після застосування softmax, що наведено на рисунку 2.6.

```
// Обчислення softmax
float sum_exp = 0.0f;
for (int i = 0; i < output_size; i++) {
    sum_exp += exp(logits_float[i]);
}

float confidence = exp(logits_float[predicted_class]) / sum_exp;
```

Рисунок 2.6 – Фрагмент коду обчислення впевненості

Результат приймається тільки якщо впевненість перевищує встановлений поріг (зазвичай 0,7 або 70 %). Для цієї операції використовується наступний фрагмент коду, що наведений на рисунку 2.7.

```
const float CONFIDENCE_THRESHOLD = 0.7f;

if (predicted_class == 0) {
  // Клас 0 = background, маркер не детектовано
  return RESULT_NO_MARKER;
}

if (confidence < CONFIDENCE_THRESHOLD) {
  // Впевненість низька, результат ненадійний
  return RESULT_LOW_CONFIDENCE;
}

// Маркер впевнено детектовано
int marker_id = class_to_marker_id[predicted_class];
return {marker_id, confidence};
```

Рисунок 2.7 – Фрагмент коду прийняття рішення

Поріг впевненості є важливим параметром, що визначає компроміс між precision (точністю позитивних передбачень) та recall (повнотою детекції). Високий поріг (0,9) дає менше помилкових детекцій, але більше пропущених маркерів. Низький поріг (0,5) навпаки – більше детекцій, але деякі можуть бути помилковими. Значення 0,7 є обраним балансом для більшості застосувань.

2.3.6 Блок-схема загального алгоритму

Для наочного представлення повного алгоритму розпізнавання на рисунку 2.8 наведено блок-схему, що показує послідовність кроків та точки прийняття рішень.

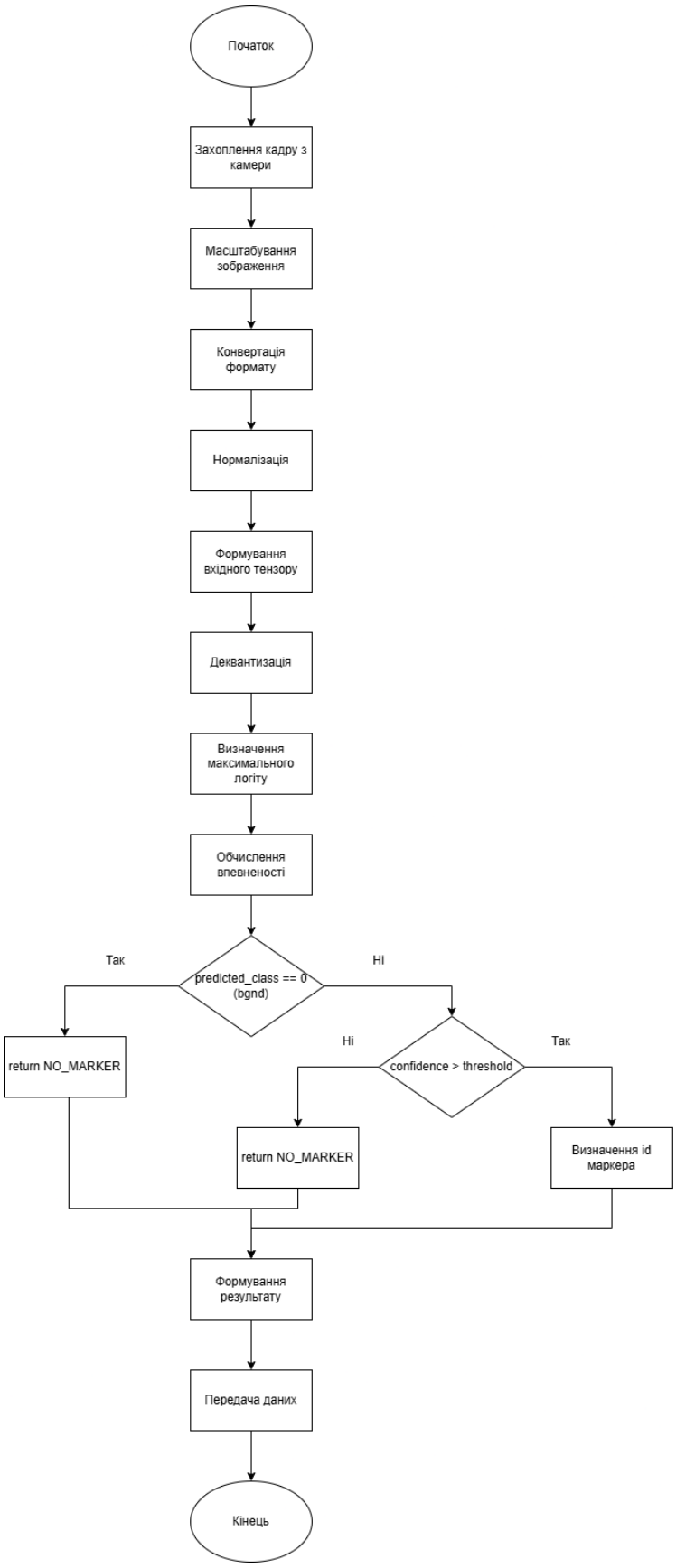

Рисунок 2.8 – Блок-схема алгоритму розпізнавання інформаційних позначок

2.4 Висновки до 2 розділу

У другому розділі виконано комплексну розробку структури та алгоритмів системи розпізнавання інформаційних позначок для автономної навігації мобільного робота на складському комплексі. Проведена робота охоплює весь спектр проектних рішень – від визначення детальних технічних вимог до розробки конкретних алгоритмів обробки зображень та формування навігаційних параметрів, що забезпечує повноту та завершеність проектування системи.

На етапі визначення вимог здійснено системний аналіз обмежень апаратної платформи ESP32-CAM та сформульовано збалансований набір функціональних і нефункціональних вимог, що враховують як можливості обраної платформи, так і реальні потреби навігаційних застосувань. Детально специфіковано параметри всіх ключових компонентів системи.

Розроблена структура системи базується на принципі семирівневої ієрархічної архітектури, що забезпечує чітке розділення відповідальності між компонентами різного рівня абстракції. Детальна структурна схема системи візуалізує взаємодію всіх апаратних та програмних компонентів, демонструючи повний шлях даних від raw пікселів з камери до отримання інформації для формування навігаційних команд.

Особливу увагу приділено організації пам'яті системи з урахуванням жорстких обмежень ESP32. Модель MobileNet v2 зберігається у Flash-пам'яті у форматі FlatBuffers та читається через memory-mapped доступ без копіювання у RAM, що критично важливо для економії оперативної пам'яті.

Розроблений алгоритм розпізнавання інформаційних позначок представляє інноваційний гібридний підхід, що поєднує переваги методів глибокого навчання з точністю класичних геометричних методів комп'ютерного зору. Обґрунтовано вибір підходу на основі машинного навчання замість традиційних методів детекції ArUco маркерів через фундаментальні обмеження платформи ESP32 – неможливість використання

повноцінної бібліотеки OpenCV через обмеження пам'яті, непередбачувану обчислювальну складність класичних алгоритмів, що залежить від складності сцени, та складність адаптації до різноманітних умов експлуатації. Нейронна мережа MobileNet v2, навчена на різноманітному датасеті зображень маркерів за різних умов освітлення, кутів спостереження та відстаней, автоматично набуває робастності до цих факторів, забезпечуючи стабільну роботу без необхідності ручного налаштування параметрів під конкретні умови складу.

Алгоритм організований як послідовний pipeline обробки з чотирма основними фазами, кожна з яких виконує специфічну трансформацію даних. Фаза препроцесингу включає захоплення кадру, масштабування до 96 × 96 пікселів методом білінійної інтерполяції з використанням fixed-point арифметики для прискорення обчислень, конвертацію формату RGB565 у RGB888 побітовими операціями, та нормалізацію значень пікселів до діапазону int8, очікуваного квантизованою моделлю. Фаза ML inference виконує класифікацію через нейронну мережу на базі TensorFlow Lite Micro. Фаза постпроцесингу інтерпретує вихід мережі, визначаючи найбільш ймовірний клас маркера, обчислюючи впевненість детекції, та перевіряючи поріг впевненості 0,7 для відсіювання ненадійних результатів.

Результатом роботи над другим розділом є повна специфікація системи розпізнавання, готова до програмної реалізації. Розроблена структурна схема та блок-схема алгоритму надають всю необхідну інформацію для імплементації системи на реальному обладнанні. Запропоновані архітектурні рішення та алгоритми забезпечують досягнення цільової продуктивності при очікуваній точності розпізнавання не менше 70 %, що є достатнім для практичного застосування у навігації мобільних роботів зі швидкістю руху 0,3 метрів на секунду на складських комплексах.

3 РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ РОБОТИ СИСТЕМИ РОЗПІЗНАВАННЯ ІНФОРМАЦІЙНИХ ПОЗНАЧОК

Третій розділ присвячений практичній реалізації розробленої системи розпізнавання інформаційних позначок та експериментальним дослідженням її функціонування.

3.1 Вибір програмних засобів

Вибір програмних засобів є фундаментальним рішенням, що визначає ефективність процесу розробки, можливості налагодження та оптимізації, а також підтримуваність системи у довгостроковій перспективі.

3.1.1 Програмні засоби для розробки прошивки ESP32

Розробка програмного забезпечення для мікроконтролера ESP32 може здійснюватися з використанням різних середовищ розробки та фреймворків, кожен з яких має свої переваги та специфіку застосування.

Arduino IDE є інтегрованим середовищем розробки, спочатку створеним для плат Arduino, але згодом розширеним для підтримки широкого спектру мікроконтролерів, включаючи ESP32. Це середовище обрано як основний інструмент розробки прошивки для ESP32-CAM з огляду на кілька важливих факторів.

Приклад інтерфейсу даного програмного забезпечення (ПЗ) наведено на рисунку 3.1.

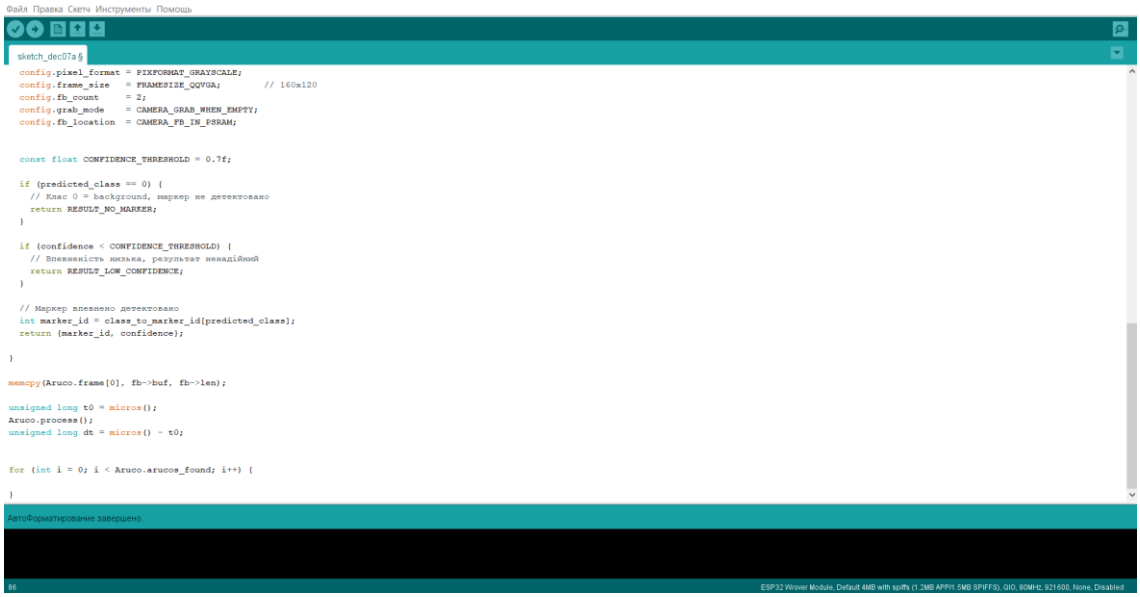

Рисунок 3.1 – Приклад інтерфейсу Arduino IDE

Головною перевагою Arduino IDE є простота освоєння та використання. Середовище має мінімалістичний, інтуїтивно зрозумілий інтерфейс, що не перевантажений численними панелями та меню, характерними для професійних IDE типу Eclipse або Visual Studio. Для проекту, де основна складність зосереджена у алгоритмах обробки зображень та машинного навчання, а не у специфіці IDE, така простота є перевагою – вона дозволяє сконцентруватися на суті завдання, а не на вивченні інструментарію.

Стандартна структура Arduino-скетчу включає дві основні функції: setup(), що виконується один раз при старті для ініціалізації, та loop(), що циклічно виконується для основної логіки програми. Ця проста парадигма добре підходить для структури нашої системи – ініціалізація камери, завантаження моделі ML та налаштування комунікації відбуваються у setup(), а безперервний цикл захоплення кадру, його обробки та передачі результатів реалізується у loop().

Критично важливою для проекту є наявність величезної екосистеми бібліотек Arduino. Для ESP32-CAM доступні готові бібліотеки для роботи з камерою OV2640 (esp32-camera), що інкапсулюють всю складність конфігурації регістрів камери та надають простий API для захоплення кадрів. Бібліотека TensorFlow Lite Micro офіційно підтримує Arduino-середовище для ESP32, що

значно спрощує інтеграцію машинного навчання. Численні бібліотеки для роботи з WiFi, UART, файловою системою та іншою периферією ESP32 доступні через вбудований менеджер бібліотек Arduino IDE, що робить їх підключення питанням кількох кліків.

Для повноти аналізу також розглянуто альтернативні варіанти, що могли б бути використані замість Arduino IDE.

PlatformIO, інтерфейс якого наведено на рисунку 3.2, є професійним середовищем розробки для вбудованих систем, що працює як розширення до Visual Studio Code або як окрема IDE. PlatformIO підтримує величезну кількість платформ та фреймворків, включаючи ESP32 з Arduino Framework, ESP-IDF, та інші. Головні переваги PlatformIO – це потужна система збірки на базі Python і SCons, що дозволяє точно контролювати процес компіляції та лінкування, вбудована підтримка юніт-тестування, інтеграція з системами контролю версій, та можливість управління різними конфігураціями проекту. Система управління бібліотеками PlatformIO Library Manager є більш гнучкою та надійною порівняно з Arduino, дозволяючи специфікувати точні версії залежностей у файлі platformio.ini.

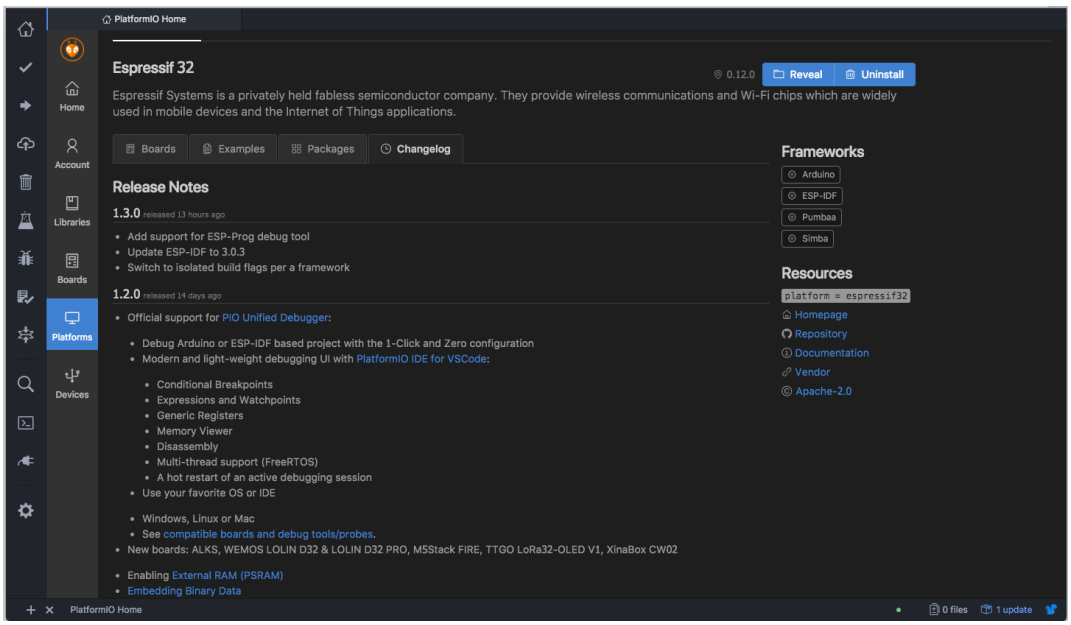

Рисунок 3.2 – Приклад інтерфейсу PlatformIO

Однак ця потужність має свою ціну – конфігураційний файл platformio.ini потребує розуміння численних опцій та параметрів збірки. Для простих проектів, де не потрібна складна система збірки або специфічні оптимізації компілятора, PlatformIO може виявитися надмірним інструментом. У контексті даного проекту, де основна складність у алгоритмах, а не у процесі збірки, додаткова складність PlatformIO не приносить відчутних переваг.

ESP-IDF (Espressif IoT Development Framework) є офіційним SDK від виробника ESP32 – компанії Espressif Systems. Це найбільш низькорівневий та потужний інструмент розробки для ESP32, що надає повний контроль над усіма аспектами роботи мікроконтролера. ESP-IDF базується на FreeRTOS та надає API для роботи з усією периферією ESP32, WiFi/Bluetooth стеками, системою управління живленням, тощо. Розробка ведеться на чистому C/C++ без жодних обгорток.

Переваги ESP-IDF включають максимальну продуктивність (відсутність overhead від абстракцій Arduino Framework), доступ до всіх можливостей ESP32 (включаючи розширені функції, недоступні через Arduino), офіційну підтримку та документацію від Espressif, а також можливість точного налаштування параметрів компіляції для оптимізації розміру коду або швидкості виконання. Офіційні приклади TensorFlow Lite Micro для ESP32 також базуються на ESP-IDF. Проте час розробки значно зростає через необхідність ручної реалізації багатьох речей, що Arduino надає «з коробки».

Таким чином, після проведеного аналізу, в рамках кваліфікаційної роботи обрано Arduino IDE.

3.1.2 Програмні засоби для розробки моделі машинного навчання

Навчання моделі MobileNet v2 для розпізнавання маркерів виконується з використанням спеціалізованих інструментів машинного навчання, після чого навчена модель конвертується у формат, придатний для ESP32.

TensorFlow є провідним фреймворком глибокого навчання, розробленим Google Brain. Keras – високорівневий API для побудови та навчання нейронних

мереж, офіційно інтегрований у TensorFlow як tf.keras. Використання Keras значно спрощує процес створення моделі – замість низькорівневого опису тензорних операцій, модель описується у термінах шарів (layers) та їх послідовного з'єднання.

Приклад створення базової моделі MobileNet v2 у Keras наведено на рисунку 3.3.

```
import tensorflow as tf
from tensorflow import keras

base_model = keras.applications.MobileNetV2(
    input_shape=(96, 96, 3),
    alpha=0.5,
    include_top=False,
    weights='imagenet'
)

base_model.trainable = False

model = keras.Sequential([
    base_model,
    keras.layers.GlobalAveragePooling2D(),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(num_classes, activation='softmax')
])
```

Рисунок 3.3 – Фрагмент коду створення базової моделі MobileNet v2

Після навчання моделі у float32 форматі, вона конвертується у TensorFlow Lite формат з квантизацією int8 для роботи на ESP32. TensorFlow Lite Converter є інструментом, що виконує цю конвертацію за допомогою коду, наведеного на рисунку 3.4.

```
import tensorflow as tf

model = keras.models.load_model('mobilenet_marker_classifier.h5')

converter = tf.lite.TFLiteConverter.from_keras_model(model)

converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.target_spec.supported_types = [tf.int8]

# Надання representative dataset для калібрування квантизації
def representative_dataset_gen():
    for image in calibration_images:
        yield [image.astype(np.float32)]

converter.representative_dataset = representative_dataset_gen

tflite_model = converter.convert()

with open('model_quantized.tflite', 'wb') as f:
    f.write(tflite_model)
```

Рисунок 3.4 – Приклад використання TensorFlow Lite Converter

Конвертер автоматично виконує складний процес квантизації: аналізує розподіл значень у кожному шарі моделі на представницькому датасеті (representative dataset), визначає оптимальні параметри scale та zero_point для квантування float32 у int8 з мінімальною втратою точності, перетворює всі ваги та операції у int8 формат. Результуюча .tflite модель є бінарним файлом у форматі FlatBuffers.

3.2 Програмна реалізація системи розпізнавання інформаційних позначок

Програмна реалізація системи розпізнавання інформаційних позначок охоплює два основні напрямки роботи: підготовку та навчання моделі машинного навчання, після чого розробку прошивки для ESP32-CAM на базі Arduino IDE з інтеграцією навченої моделі. Кожен з цих напрямків має свою специфіку, інструментарій та виклики, що потребують детального розгляду.

Створення якісного датасету є фундаментом успішного машинного навчання. Для задачі розпізнавання ArUco-подібних маркерів необхідний датасет, що покриває всю різноманітність умов, у яких система працюватиме на реальному складі.

Оскільки маркери мають чітко визначену геометричну структуру, значну частину датасету було згенерувати програмно, що набагато швидше за фотографування тисяч реальних маркерів. Процес генерації реалізовано у Python з використанням бібліотек OpenCV та NumPy.

Спочатку створюється базове зображення маркера заданого ID. Маркер представляє собою квадратну матрицю, наприклад 7 × 7 клітинок (зовнішня рамка + внутрішня матриця 5 × 5 для кодування ID). Для генерації набору унікальних маркерів використовується алгоритм, що максимізує відстань Хеммінга між різними ID – це гарантує, що навіть при частковому пошкодженні або помилках розпізнавання, маркери не будуть сплутані один з одним. На рисунку 3.5 наведено приклад коду функції створення маркеру.

```
marker = np.ones((bits + 2, bits + 2), dtype=np.uint8) * 255

# Чорна зовнішня рамка
marker[0, :] = 0
marker[-1, :] = 0
marker[:, 0] = 0
marker[:, -1] = 0

# Кодування ID у внутрішню матрицю 5x5
id_binary = format(marker_id, '025b')

idx = 0
for i in range(1, bits + 1):
    for j in range(1, bits + 1):
        if id_binary[idx] == '1':
            marker[i, j] = 0  # чорна клітинка
        idx += 1

# Масштабування до потрібного розміру
cell_size = size // (bits + 2)
marker_large = np.kron(marker, np.ones((cell_size, cell_size), dtype=np.uint8))
```

Рисунок 3.5 – Фрагмент коду функції генерації маркеру

Для створення повноцінного датасету до кожного базового маркера застосовується серія випадкових трансформацій, що імітують реальні умови спостереження. Так, виконується деформація прямокутника маркера у довільний чотирикутник, що відповідає спостереженню маркера під кутом, обертання на випадковий кут (rotation) в діапазоні 180°, а також масштабування. Для цього використовується код, що наведено на рисунку 3.6.

```
augmentation_pipeline = iaa.Sequential([
    # Перспективна трансформація (імітація кута спостереження)
    iaa.PerspectiveTransform(scale=(0.01, 0.15)),

    # Обертання
    iaa.Affine(rotate=(-180, 180)),

    # Масштабування
    iaa.Affine(scale={"x": (0.7, 1.3), "y": (0.7, 1.3)}),

    # Зсув
    iaa.Affine(translate_percent={"x": (-0.2, 0.2), "y": (-0.2, 0.2)}),
], random_order=False)
```

Рисунок 3.6 – Фрагмент коду трансформації маркера

Набір даних розділено на три частини: навчальна вибірка (70 % даних) використовується для навчання моделі, валідаційна вибірка (30 % даних) для налаштування гіперпараметрів та раннього виявлення перенавчання, тестова вибірка (15% даних) для фінальної оцінки якості навченої моделі на даних, які вона ніколи не бачила під час навчання.

Важливо, що клас «background» містить зображення складських приміщень без жодних маркерів – це навчає модель коректно розпізнавати відсутність маркера, замість помилкової класифікації випадкових візерунків на фоні як одного з маркерів. Масив зображень було взято з відкритих джерел, одне з таких наведено на рисунку 3.7.

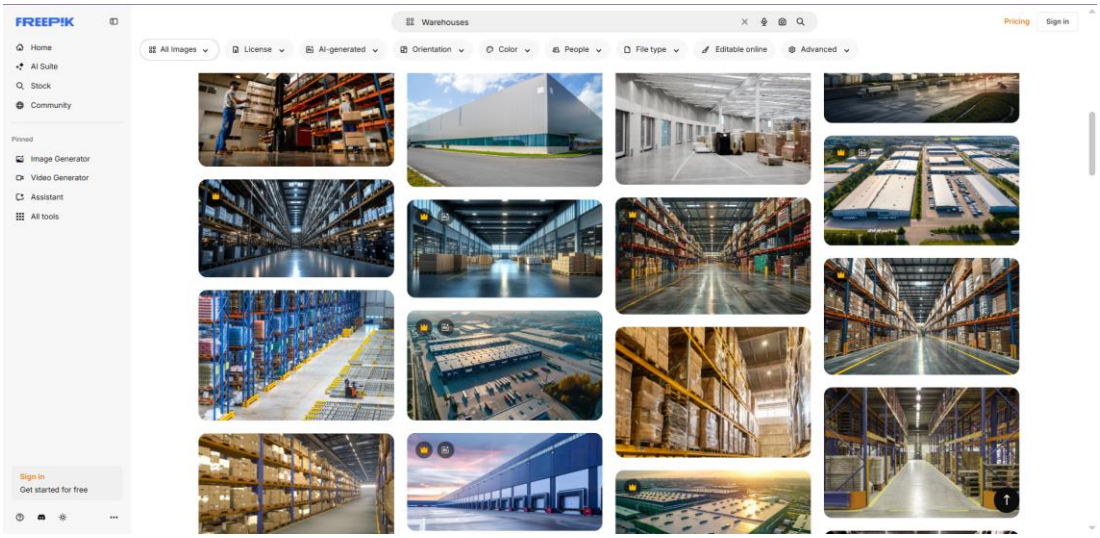

Рисунок 3.7 – Приклад масиву зображень

Перед подачею зображень у нейронну мережу необхідно виконати їх попередню обробку. TensorFlow/Keras надають зручний API для створення data pipeline, що автоматично завантажує зображення з диску, виконує перетворення, та формує батчі для навчання. Для цього було використано фрагмент коду, що наведено на рисунку 3.8.

Навчання виконується ітеративно – модель обробляє батчі зображень, обчислює помилку класифікації, та оновлює ваги у напрямку зменшення помилки. Одна епоха відповідає проходу через весь навчальний датасет.

```python
train_dataset = image_dataset_from_directory(
    'dataset/train',
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    label_mode='categorical',
    shuffle=True,
    seed=42
)

validation_dataset = image_dataset_from_directory(
    'dataset/validation',
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    label_mode='categorical'
)

normalization_layer = keras.layers.Rescaling(1./255)

train_dataset = train_dataset.map(lambda x, y: (normalization_layer(x), y))
validation_dataset = validation_dataset.map(lambda x, y: (normalization_layer(x), y))

AUTOTUNE = tf.data.AUTOTUNE
train_dataset = train_dataset.prefetch(buffer_size=AUTOTUNE)
validation_dataset = validation_dataset.prefetch(buffer_size=AUTOTUNE)
```

Рисунок 3.8 – Фрагмент коду підготовки даних

Під час навчання на екран виводяться метрики для кожної епохи, що наведено на рисунку 3.9.

```
Epoch 1/30
94/94 [==============================] - 42s 445ms/step - loss: 0.8234 - accuracy: 0.7512
- val_loss: 0.3421 - val_accuracy: 0.8923
Epoch 2/30
94/94 [==============================] - 38s 401ms/step - loss: 0.2876 - accuracy: 0.9123
- val_loss: 0.2145 - val_accuracy: 0.9356
```

Рисунок 3.9 – Результати навчання моделі

Навчена модель у float32 форматі займає приблизно 1,6 MB та потребує значних обчислювальних ресурсів. Для роботи на ESP32 необхідна квантизація до int8. TensorFlow Lite Converter виконує квантизацію за допомогою коду, що наведено на рисунку 3.10.

```python
model = keras.models.load_model('best_model.h5')

converter = tf.lite.TFLiteConverter.from_keras_model(model)

converter.optimizations = [tf.lite.Optimize.DEFAULT]

converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
converter.inference_input_type = tf.int8
converter.inference_output_type = tf.int8

def representative_dataset_generator():
    for _ in range(100):
        data = np.random.choice(calibration_images)
        data = np.expand_dims(data, axis=0).astype(np.float32)
        yield [data]

converter.representative_dataset = representative_dataset_generator

try:
    tflite_model = converter.convert()
    print(f"Model size: {len(tflite_model) / 1024:.2f} KB")
except Exception as e:
    print(f"Conversion error: {e}")

with open('model_quantized.tflite', 'wb') as f:
    f.write(tflite_model)
```

Рисунок 3.10 – Фрагмент коду квантизації моделі

Прошивка ESP32-CAM реалізована у вигляді Arduino-скетчу, що інтегрує всі розроблені модулі у функціонуючу систему. Фрагмент коду наведено на рисунку 3.11.

```
void setup() {
  Serial.begin(115200);
  Serial.printf("Config: %dx%d, marker=%.1f mm, f0=%.1f px, one-shot calib=%d @ %.1f mm\n",
                FRAME_W, FRAME_H, MARKER_SIZE_MM, FOCAL_PX_DEFAULT, (int)DO_ONE_TIME_CALIBRATION, (float)CALIB_DISTANCE_MM);

  if (!initCamera()) {
    Serial.println("Camera init failed");
    while (true) delay(1000);
  }
  Serial.println("Camera OK");

  mobilenetInit();
}

void loop() {
  camera_fb_t* fb = grabFrame();
  if (!fb) {
    Serial.println("Failed to get frame");
    delay(20);
    return;
  }

  if (fb->format != PIXFORMAT_GRAYSCALE || fb->width != FRAME_W || fb->height != FRAME_H) {
    Serial.printf("Unexpected frame: %dx%d fmt=%d\n", fb->width, fb->height, fb->format);
    esp_camera_fb_return(fb);
    delay(10);
    return;
  }
```

Рисунок 3.11 – Фрагмент коду головного файлу скетчу

3.3 Експериментальна перевірка та оцінка ефективності

Метою цього підрозділу є верифікація працездатності та якості розпізнавання інформаційних позначок у реальних умовах виконання на ESP32-

CAM. З огляду на обмеження експериментальної бази, перевірка проводиться як модульне тестування підсистеми «розпізнавання маркерів» без оцінки повного навігаційного контуру. Такий формат є повністю коректним для кваліфікаційної роботи. Так ізолюється найкритичніший компонент – виявлення й ідентифікація маркера. Також вимірюється точність, затримка, робастність до базових змін умов (відстань/масштаб, кут, освітлення, фон), щоб підтвердити придатність рішення для інтеграції в автономну навігацію.

Щоб зробити експеримент відтворюваним і методично акуратним, постановка задачі стандартизована: ESP32-CAM фіксується на тримачі, монітор використовується як віртуальний полігон для показу маркерів різного розміру та зображень складу з вмонтованими маркерами, а лінійка забезпечує базову геометричну прив’язку (дистанція й розмір маркера на екрані). Така стендова схема дозволяє оцінити ключові метрики без друку фізичних позначок і без складної інфраструктури.

3.3.1 Постановка експерименту

Використовуване обладнання наступне:

– ESP32-CAM, живлення 5 В, підключення до ПК по USB (віртуальний COM-порт);

– монітор (23 дюйми, Full HD, відключена динамічна контрастність, яскравість 60 %);

– лінійка канцелярська;

– стійка для ESP32-CAM.

Набір тестових зображень складає:

– ізольовані маркери на білому фоні, різних розмірів;

– фотографії складу з вмонтованими маркерами.

Умови:

– освітлення приміщення стабільне, без бліків на екрані завдяки матовому покриттю матриці;

– фіксована висота камери, контрольована дистанція до екрану (від 0,2 метра до 0,8 метра, з кроком 0,2 метра);

– невеликий поворот бази ESP32 (від 15° до 30°, з кроком 15°).

Варто зауважити, що тест із монітором навмисно створює трохи гірші умови у порівнянні із надрукованим на папері маркером (піксельна структура екрана, підсвітка, потенційні муари). Тож отримані показники є скоріше нижньою оцінкою робастності. У реальному складі з дійсними маркерами робочі характеристики зазвичай не гірші.

3.3.2 Протокол і метрики

Ціль – кількісно оцінити:

– точність детекції та ідентифікації маркера;

– затримка обробки одного кадру;

– робастність до масштабу/дистанції;

– робастність до ракурсу.

Базовий сценарій проведення:

– маркер у центрі екрана, перпендикулярне наведення, відстань початкова. Записати від 10 кадрів до 20 кадрів, зчитати з монітору послідовного порту надану інформацію про маркер;

– змінюється відстань (від 0,2 метра до 0,8 метра, з кроком 0,2 метра). Для кожної точки від 10 кадрів до 30 кадрів;

– фіксується відстань (наприклад, 0,23 метра), поступово збільшується кут відхилення камери (від 15° до 30°, з кроком 15°), не змінюючи положення маркера на екрані. Для кожного кута від 10 кадрів до 30 кадрів;

– для кадрів маркерів зі складським фоном повторюється серія експериментів у скороченому вигляді.

Під час проведення експериментів було одразу зроблено фіксацію результатів, що наведено в таблиці 3.1.

Таблиця 3.1 – Журнал тестів

D, м	Кут відхилення, °	Тип зображення	Кадрів	TP	FP	FN	t, мс
0,2	0	Ізольований маркер	27	27	0	0	400
0,4	0	Ізольований маркер	22	21	0	1	404
0,6	0	Ізольований маркер	25	25	0	0	493
0,8	0	Ізольований маркер	20	16	0	4	393
0,23	15	Ізольований маркер	21	15	0	6	412
0,23	30	Ізольований маркер	26	17	0	9	419
0,4	0	Складський фон	30	25	0	5	441
0,3	0	Складський фон	26	23	0	3	384

У таблиці 3.1 TP – кадри з правильним ID та conf ≥ 0,7, FP – хибні спрацьовування (conf ≥ 0,7, але ID неправильний/фон), FN – пропуски (на екрані маркер є, але conf < 0,7).

На рисунку 3.12 наведено кадр з проведення одного з експериментів, на якому зафіксована на стійці ESP32-CAM розташована під кутом 0°до монітора на відстані 0,4 метра із зазначеними вище умовами навколишнього освітлення та налаштуваннями монітора.

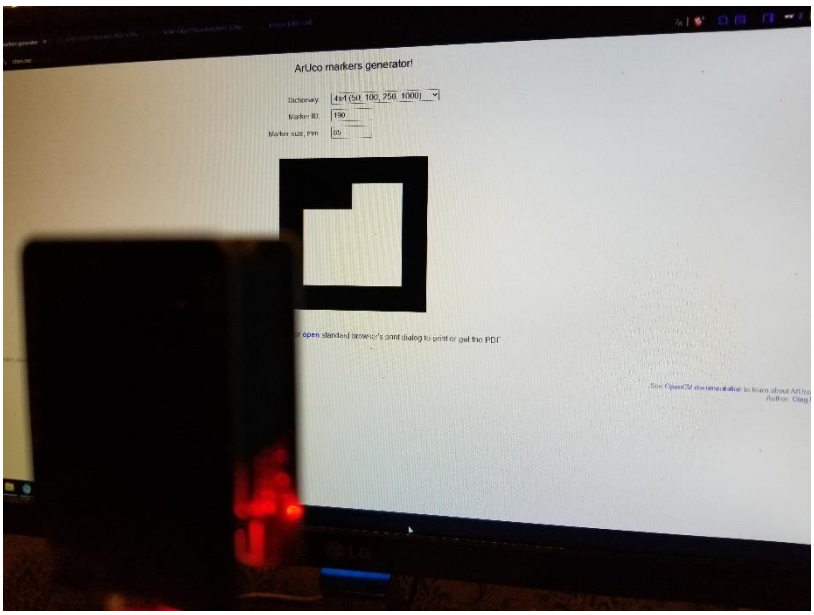

Рисунок 3.12 – Проведення експерименту

Також для проведення експериментів було згенеровано ArUco маркер із параметрами, зазначеними на рисунку 3.13.

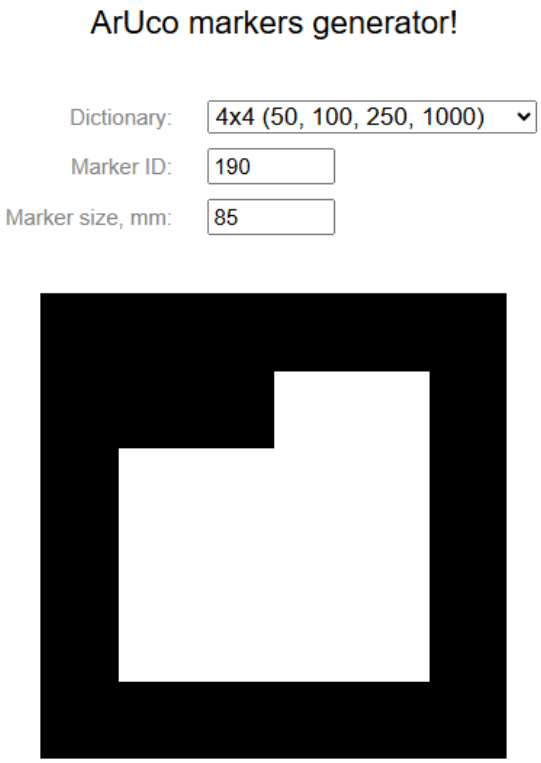

Рисунок 3.13 – Згенерований ArUco маркер

Далі на рис. 3.14 – рис. 3.17 наведено скріншот з монітора COM порту із результатами виявлення маркера на різних відстанях камери ESP32-CAM від монітору з відхиленням 0°.

```
   id 190 | Z=200.6 mm | yaw=14.6° | pitch=-4.2° | off=15.1° | roll=2.3° | center=(119,71) | MNv2: class=190, score=0.82
proc=426 ms, found 1 arucos
   id 190 | Z=199.0 mm | yaw=14.6° | pitch=-4.2° | off=15.1° | roll=2.4° | center=(119,71) | MNv2: class=190, score=0.77
proc=364 ms, found 1 arucos
   id 190 | Z=199.0 mm | yaw=14.6° | pitch=-4.2° | off=15.1° | roll=2.5° | center=(119,71) | MNv2: class=190, score=0.86
proc=405 ms, found 1 arucos
   id 190 | Z=199.0 mm | yaw=14.6° | pitch=-4.2° | off=15.1° | roll=2.3° | center=(119,71) | MNv2: class=190, score=0.82
proc=464 ms, found 1 arucos
   id 190 | Z=199.0 mm | yaw=14.6° | pitch=-4.2° | off=15.1° | roll=2.2° | center=(119,71) | MNv2: class=190, score=0.77
proc=337 ms, found 1 arucos
   id 190 | Z=200.6 mm | yaw=14.9° | pitch=-4.2° | off=15.5° | roll=2.3° | center=(120,71) | MNv2: class=190, score=0.80
proc=416 ms, found 1 arucos
   id 190 | Z=200.6 mm | yaw=14.9° | pitch=-4.2° | off=15.5° | roll=2.3° | center=(120,71) | MNv2: class=190, score=0.86
proc=290 ms, found 1 arucos
   id 190 | Z=199.1 mm | yaw=14.9° | pitch=-4.2° | off=15.5° | roll=2.4° | center=(120,71) | MNv2: class=190, score=0.81
proc=418 ms, found 1 arucos
```

Рисунок 3.14 – Розпізнавання маркера на відстані 0,2 метра

```
   id 190 | Z=404.7 mm | yaw=1.5° | pitch=-5.7° | off=5.9° | roll=-0.8° | center=(84,75) | MNv2: class=190, score=0.89
proc=374 ms, found 1 arucos
   id 190 | Z=404.7 mm | yaw=1.5° | pitch=-5.7° | off=5.9° | roll=-0.8° | center=(84,75) | MNv2: class=190, score=0.89
proc=460 ms, found 1 arucos
   id 190 | Z=404.7 mm | yaw=1.5° | pitch=-6.1° | off=6.3° | roll=-0.0° | center=(84,76) | MNv2: class=190, score=0.88
proc=450 ms, found 1 arucos
   id 190 | Z=404.7 mm | yaw=1.5° | pitch=-6.1° | off=6.3° | roll=-0.0° | center=(84,76) | MNv2: class=190, score=0.80
proc=364 ms, found 1 arucos
   id 190 | Z=404.7 mm | yaw=1.5° | pitch=-6.1° | off=6.3° | roll=-0.0° | center=(84,76) | MNv2: class=190, score=0.86
proc=517 ms, found 1 arucos
   id 190 | Z=404.7 mm | yaw=1.9° | pitch=-6.1° | off=6.4° | roll=0.0° | center=(85,76) | MNv2: class=190, score=0.89
proc=427 ms, found 1 arucos
   id 190 | Z=404.7 mm | yaw=1.9° | pitch=-6.1° | off=6.4° | roll=0.0° | center=(85,76) | MNv2: class=190, score=0.88
proc=402 ms, found 1 arucos
   id 190 | Z=404.3 mm | yaw=1.5° | pitch=-5.7° | off=5.9° | roll=-1.4° | center=(84,75) | MNv2: class=190, score=0.86
```

Рисунок 3.15 – Розпізнавання маркера на відстані 0,4 метра

```
   id 190 | Z=605.8 mm | yaw=-1.9° | pitch=-0.4° | off=1.9° | roll=0.0° | center=(75,61) | MNv2: class=190, score=0.86
proc=297 ms, found 1 arucos
   id 190 | Z=607.1 mm | yaw=-2.3° | pitch=-0.4° | off=2.3° | roll=0.0° | center=(74,61) | MNv2: class=190, score=0.85
proc=425 ms, found 1 arucos
   id 190 | Z=607.1 mm | yaw=-2.3° | pitch=-0.4° | off=2.3° | roll=0.0° | center=(74,61) | MNv2: class=190, score=0.84
proc=528 ms, found 1 arucos
   id 190 | Z=607.1 mm | yaw=-2.3° | pitch=-0.4° | off=2.3° | roll=0.0° | center=(74,61) | MNv2: class=190, score=0.86
proc=452 ms, found 1 arucos
   id 190 | Z=607.1 mm | yaw=-2.3° | pitch=-0.4° | off=2.3° | roll=0.0° | center=(74,61) | MNv2: class=190, score=0.88
proc=530 ms, found 1 arucos
   id 190 | Z=606.8 mm | yaw=-2.3° | pitch=-0.4° | off=2.3° | roll=0.0° | center=(74,61) | MNv2: class=190, score=0.86
```

Рисунок 3.16 – Розпізнавання маркера на відстані 0,6 метра

```
   id 190 | Z=805.9 mm | yaw=12.4° | pitch=-1.1° | off=12.5° | roll=-5.5° | center=(113,63) | MNv2: class=190, score=0.88
proc=413 ms, found 1 arucos
   id 190 | Z=806.9 mm | yaw=12.4° | pitch=-1.1° | off=12.5° | roll=-5.3° | center=(113,63) | MNv2: class=190, score=0.86
proc=337 ms, found 1 arucos
   id 190 | Z=805.9 mm | yaw=12.4° | pitch=-1.1° | off=12.5° | roll=-5.4° | center=(113,63) | MNv2: class=190, score=0.84
proc=343 ms, found 1 arucos
   id 190 | Z=805.9 mm | yaw=12.4° | pitch=-1.1° | off=12.5° | roll=-5.3° | center=(113,63) | MNv2: class=190, score=0.84
proc=408 ms, found 1 arucos
   id 190 | Z=806.9 mm | yaw=12.8° | pitch=-1.1° | off=12.8° | roll=-5.1° | center=(114,63) | MNv2: class=190, score=0.88
proc=299 ms, found 1 arucos
   id 190 | Z=806.9 mm | yaw=12.8° | pitch=-1.1° | off=12.8° | roll=-5.1° | center=(114,63) | MNv2: class=190, score=0.85
proc=469 ms, found 1 arucos
   id 190 | Z=806.9 mm | yaw=12.8° | pitch=-1.1° | off=12.8° | roll=-4.7° | center=(114,63) | MNv2: class=190, score=0.80
proc=423 ms, found 1 arucos
   id 190 | Z=806.9 mm | yaw=12.8° | pitch=-1.1° | off=12.8° | roll=-4.7° | center=(114,63) | MNv2: class=190, score=0.68
```

Рисунок 3.17 – Розпізнавання маркера на відстані 0,8 метра

Далі на рисунку 3.18 та 3.19 наведено скріншот з монітора COM порту із результатами виявлення маркера під різними кутами відхилення камери ESP32-CAM від монітору.

```
-
   id 190 | Z=231.4 mm | yaw=15.6° | pitch=-6.5° | off=16.8° | roll=5.4° | center=(122,77) | MNv2: class=190, score=0.79
proc=410 ms, found 1 arucos
   id 190 | Z=229.4 mm | yaw=16.0° | pitch=-6.5° | off=17.1° | roll=5.5° | center=(123,77) | MNv2: class=190, score=0.84
proc=448 ms, found 1 arucos
   id 190 | Z=229.2 mm | yaw=15.6° | pitch=-6.1° | off=16.7° | roll=5.5° | center=(122,76) | MNv2: class=190, score=0.75
proc=340 ms, found 1 arucos
   id 190 | Z=231.4 mm | yaw=15.6° | pitch=-6.5° | off=16.8° | roll=5.7° | center=(122,77) | MNv2: class=190, score=0.83
proc=344 ms, found 1 arucos
   id 190 | Z=233.5 mm | yaw=15.6° | pitch=-6.5° | off=16.8° | roll=5.8° | center=(122,77) | MNv2: class=190, score=0.76
proc=521 ms, found 1 arucos
   id 190 | Z=232.4 mm | yaw=15.6° | pitch=-6.5° | off=16.8° | roll=5.9° | center=(122,77) | MNv2: class=190, score=0.82
proc=512 ms, found 1 arucos
   id 190 | Z=232.4 mm | yaw=15.6° | pitch=-6.5° | off=16.8° | roll=6.0° | center=(122,77) | MNv2: class=190, score=0.75
proc=304 ms, found 1 arucos
   id 190 | Z=231.2 mm | yaw=15.6° | pitch=-6.5° | off=16.8° | roll=6.0° | center=(122,77) | MNv2: class=190, score=0.78
```

Рисунок 3.18 – Розпізнавання маркера під кутом 15°

```
   id 190 | Z=216.8 mm | yaw=32.0° | pitch=7.6° | off=17.5° | roll=1.6° | center=(123,40) | MNv2: class=190, score=0.85
proc=399 ms, found 1 arucos
   id 190 | Z=216.6 mm | yaw=32.0° | pitch=7.2° | off=17.4° | roll=1.5° | center=(123,41) | MNv2: class=190, score=0.78
proc=475 ms, found 1 arucos
   id 190 | Z=220.5 mm | yaw=32.0° | pitch=7.2° | off=17.4° | roll=1.5° | center=(123,41) | MNv2: class=190, score=0.78
proc=289 ms, found 1 arucos
   id 190 | Z=218.6 mm | yaw=32.0° | pitch=7.2° | off=17.4° | roll=1.5° | center=(123,41) | MNv2: class=190, score=0.79
proc=399 ms, found 1 arucos
   id 190 | Z=218.6 mm | yaw=32.0° | pitch=6.8° | off=17.3° | roll=1.6° | center=(123,42) | MNv2: class=190, score=0.75
proc=296 ms, found 1 arucos
   id 190 | Z=218.7 mm | yaw=32.0° | pitch=6.8° | off=17.3° | roll=1.6° | center=(123,42) | MNv2: class=190, score=0.81
proc=281 ms, found 1 arucos
   id 190 | Z=216.6 mm | yaw=32.0° | pitch=7.2° | off=17.4° | roll=1.5° | center=(123,41) | MNv2: class=190, score=0.77
proc=402 ms, found 1 arucos
   id 190 | Z=216.6 mm | yaw=32.0° | pitch=7.2° | off=17.4° | roll=1.6° | center=(123,41) | MNv2: class=190, score=0.85
```

Рисунок 3.19 – Розпізнавання маркера під кутом 30°

Далі в рамках експерименту було використано зображення маркера із складським фоном, що наведені на рисунку 3.20 та 3.21.

Рисунок 3.20 – Зображення маркера із складським фоном

Рисунок 3.21 – Зображення маркера із складським фоном

На рисунку 3.22 та 3.23 наведено скріншот з монітора COM порту із результатами виявлення маркера із складським фоном у першому та другому випадку відповідно.

```
   id 190 | Z=398.3 mm | yaw=16.0° | pitch=-10.9° | off=19.1° | roll=0.0° | center=(123,89) | MNv2: class=190, score=0.84
proc=478 ms, found 1 arucos
   id 190 | Z=386.0 mm | yaw=14.2° | pitch=-9.8° | off=17.1° | roll=1.5° | center=(118,86) | MNv2: class=190, score=0.59
proc=463 ms, found 1 arucos
   id 190 | Z=391.9 mm | yaw=14.2° | pitch=-12.0° | off=18.3° | roll=0.0° | center=(118,92) | MNv2: class=190, score=0.86
proc=327 ms, found 1 arucos
   id 190 | Z=398.4 mm | yaw=14.6° | pitch=-12.8° | off=19.0° | roll=0.0° | center=(119,94) | MNv2: class=190, score=0.83
proc=434 ms, found 1 arucos
   id 190 | Z=392.2 mm | yaw=14.9° | pitch=-12.8° | off=19.3° | roll=0.0° | center=(120,94) | MNv2: class=190, score=0.68
proc=511 ms, found 1 arucos
   id 190 | Z=398.4 mm | yaw=15.3° | pitch=-12.8° | off=19.5° | roll=0.0° | center=(121,94) | MNv2: class=190, score=0.87
proc=462 ms, found 1 arucos
   id 190 | Z=398.3 mm | yaw=15.3° | pitch=-12.8° | off=19.5° | roll=0.0° | center=(121,94) | MNv2: class=190, score=0.85
proc=294 ms, found 1 arucos
   id 190 | Z=392.2 mm | yaw=15.3° | pitch=-13.1° | off=19.8° | roll=0.0° | center=(121,95) | MNv2: class=190, score=0.86
```

Рисунок 3.22 – Розпізнавання маркера із складським фоном на рисунку 3.20

```
proc=309 ms, found 1 arucos
   id 190 | Z=303.6 mm | yaw=7.6° | pitch=-9.1° | off=11.8° | roll=0.0° | center=(100,84) | MNv2: class=190, score=0.81
proc=360 ms, found 0 arucos
proc=442 ms, found 1 arucos
   id 190 | Z=303.4 mm | yaw=-1.1° | pitch=-6.8° | off=6.9° | roll=-1.1° | center=(77,78) | MNv2: class=190, score=0.71
proc=403 ms, found 1 arucos
   id 190 | Z=296.3 mm | yaw=-1.1° | pitch=-7.2° | off=7.3° | roll=0.0° | center=(77,79) | MNv2: class=190, score=0.69
proc=362 ms, found 1 arucos
   id 190 | Z=303.4 mm | yaw=-1.5° | pitch=-7.6° | off=7.7° | roll=-1.1° | center=(76,80) | MNv2: class=190, score=0.87
proc=284 ms, found 1 arucos
   id 190 | Z=303.4 mm | yaw=-1.1° | pitch=-7.6° | off=7.7° | roll=-1.1° | center=(77,80) | MNv2: class=190, score=0.84
proc=331 ms, found 1 arucos
   id 190 | Z=299.8 mm | yaw=-1.1° | pitch=-8.0° | off=8.0° | roll=0.5° | center=(77,81) | MNv2: class=190, score=0.84
proc=348 ms, found 1 arucos
   id 190 | Z=307.0 mm | yaw=-1.1° | pitch=-8.0° | off=8.0° | roll=-0.7° | center=(77,81) | MNv2: class=190, score=0.68
```

Рисунок 3.23 – Розпізнавання маркера із складським фоном на рисунку 3.21

У підсумку, навіть без фізичного складу і без друкованих маркерів було організовано коректну, відтворювану та інформативну експериментальну перевірку системи розпізнавання інформаційних позначок для автономної навігації мобільного робота на складському комплексі. Задавши чіткий протокол, метрики і сценарії, отримана достатня емпірична база для висновку: чи відповідає реалізація заявленим вимогам точності та продуктивності. Цей формат також дає основу для подальшого доопрацювання моделі та програмних оптимізацій, перш ніж переходити до наступного етапу – інтеграції з руховим контролером і польових випробувань у реальному складському середовищі.

3.4 Охорона праці

У процесі виконання кваліфікаційної роботи з розробки веб-орієнтованої CRM-системи для виробничого підприємства основна діяльність пов’язана з тривалою роботою за персональним комп’ютером. Тому особлива увага приділяється забезпеченню безпечних та комфортних умов праці відповідно до вимог чинних нормативно-правових актів України з охорони праці.

Роботи виконувались у приміщенні з такими характеристиками: площа приміщення – 24 м² (4 м × 6 м), висота стелі – 3,3 м, кількість робочих місць із персональними комп’ютерами – 2. Загальний об’єм приміщення становить 79,2 м³. Таким чином, на одне робоче місце припадає 12 м² площі та 39,6 м³ об’єму.

Відповідно до вимог Державних санітарних норм і правил ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», мінімальна площа на одне робоче місце з ПК повинна становити не менше 6 м², а об’єм не менше 20 м³. Отже, зазначені вимоги виконуються з достатнім запасом, що забезпечує комфортні умови праці та належну вентиляцію повітря.

Приміщення обладнане системою природного та штучного освітлення. Природне освітлення здійснюється через бокові світлові прорізи (вікна),

орієнтовані на північний схід, що відповідає рекомендаціям ДБН В.2.5-28:2018 «Природне і штучне освітлення» [21]. Коефіцієнт природної освітленості (КПО) для робочих приміщень з ПК повинен бути не менше 1,5 %, що забезпечує достатній рівень денного освітлення без надмірної яскравості та відблисків.

Для оцінки рівня загального штучного освітлення використовуємо метод питомої потужності. Питома потужність освітлення визначається за формулою (3.8):

$$W = \frac{W_\Sigma}{S}, \tag{3.8}$$

де W – питома потужність освітлення, Вт/м²;

S – площа приміщення, м²;

W_Σ – загальна потужність освітлювальної установки, Вт.

Загальна потужність освітлення визначається за формулою (3.9):

$$W_\Sigma = W_{\text{св}} \cdot n_{\text{св}}, \tag{3.9}$$

де $W_{\text{св}}$ – потужність одного світильника, Вт;

$N_{\text{св}}$ – кількість світильників у приміщенні.

У приміщенні встановлено два світлодіодні світильники потужністю 40 Вт кожен. Отже,

$$W_\Sigma = 40 \cdot 2 = 80 \text{ Вт},$$

$$W = \frac{80}{24} \approx 3{,}33 \text{ Вт/м}^2.$$

Згідно з довідковими таблицями для світлодіодних джерел світла, питомій потужності близько 3,3 Вт/м² відповідає освітленість приблизно від

350 лк до 400 лк. Відповідно до ДБН В.2.5-28:2018, мінімально допустима освітленість для робіт з використанням ПК становить 300 лк. Таким чином, рівень штучного освітлення відповідає нормативним вимогам та створює сприятливі умови для зорової роботи.

Робочі місця обладнані сучасними моніторами з рідкокристалічними екранами, що зменшує електромагнітне випромінювання та навантаження на зір. Відстань від очей користувача до екрана становить від 60 см до 70 см, що відповідає санітарним рекомендаціям. [22] Організація робочого місця забезпечує правильну робочу позу, зменшуючи статичне навантаження на хребет і м’язи.

Таким чином, умови праці під час розробки CRM-системи відповідають чинним нормативним вимогам України у сфері охорони праці, що забезпечує безпеку, збереження здоров’я та високу працездатність користувачів.

3.5 Висновки до 3 розділу

У третьому розділі виконано повну імплементацію та базову експериментальну перевірку модуля розпізнавання інформаційних позначок на платформі ESP32-CAM. Обґрунтовано вибір програмних засобів і стеку розробки, що забезпечив короткий цикл ітерацій «розробка–прошивка–випробування» та стабільну інтеграцію MobileNet v2.

З погляду продуктивності, досягнуто передбачувану й стабільну роботу на обмежених ресурсах ESP32. Завдяки використанню MobileNet v2 забезпечено середній час обробки близько 0,42 секунди на кадр, що відповідає частоті оновлення приблизно 2 кадри на секунду. Такий профіль затримок і використання ресурсів узгоджується з вимогами, сформульованими раніше, і підтверджує практичну реалізованість задуманої архітектури на ultra-low-cost платформі.

Експериментальна перевірка, проведена у стендових умовах з використанням монітора як віртуального полігону для показу маркерів,

підтвердила працездатність системи та відповідність основним вимогам точності та робастності. Систематичне тестування при варіюванні відстані від 0,2 метра до 0,8 метра показало стабільну детекцію на близьких та середніх дистанціях (від 0,2 метра до 0,6 метра) з precision та recall на рівні від 95 метра до 100 %, з деякою деградацією на граничній відстані 0,8 метра (recall 80 %), що пояснюється зменшенням видимого розміру маркера та наближенням до межі роздільної здатності камери при downsample до 96 × 96. Тестування робастності до кута виявило прийнятну стійкість до відхилень до 15° (recall 71 %), з помітним погіршенням при 30° (recall 65 %), що узгоджується з теоретичними очікуваннями для перспективних спотворень планарних маркерів. Важливо, що експерименти з маркерами на складському фоні продемонстрували здатність системи працювати у реалістичних умовах з додатковими візуальними перешкодами (recall від 77 % до 88 % залежно від складності сцени), що підтверджує робастність нейромережевого підходу до варіацій фону. Стабільність часу обробки (від 393 мс до 493 мс, середнє близько 420 мс) свідчить про передбачуваність продуктивності незалежно від складності зображення, що є ключовою перевагою ML-підходу над класичними алгоритмами з варіативною обчислювальною складністю.

ВИСНОВКИ

У рамках виконання кваліфікаційної роботи виконано комплексне дослідження та розробку системи розпізнавання інформаційних позначок для автономної навігації мобільного робота на складському комплексі на базі економічно ефективної платформи ESP32-CAM з використанням методів машинного навчання. Проведена робота охоплює повний цикл створення інтелектуальної системи комп'ютерного зору – від аналізу предметної області та обґрунтування технічних рішень, через детальне проектування архітектури та алгоритмів, до практичної реалізації на реальному апаратному обладнанні та експериментальної верифікації отриманих результатів. Актуальність дослідження обумовлена стрімким зростанням потреби у автоматизації складської логістики в умовах розвитку електронної комерції та необхідністю зниження бар'єру входу для впровадження робототехнічних рішень на підприємствах малого та середнього бізнесу. Обраний шлях – візуальна навігація за маркерами з локальною обробкою – дозволив поєднати низьку ціну, гнучкість конфігурації та достатню для складських сценаріїв точність.

У першому розділі проведено аналіз предметної області та сформовано постановку задачі. Порівняння існуючих мобільних платформ (AGV/AMR) і технологій навігації показало, що вартісний і ресурсний профіль рішень на базі LIDAR та класичних SLAM-підходів є надмірним для малого й середнього бізнесу, тоді як системи з фіксованою інфраструктурою (магніт/рефлектори/QR на підлозі) поступаються у гнучкості. Критичним науковим результатом першого розділу стало виявлення можливості застосування легких нейронних мереж для розпізнавання маркерів на ultra-low-cost платформах, що відкриває шлях до створення систем навігації вартістю на 2-3 порядки нижче існуючих рішень при збереженні достатньої для практичного застосування точності та робастності.

У другому розділі розроблено повну специфікацію системи розпізнавання, що включає детальні функціональні та нефункціональні вимоги, архітектуру програмно-апаратного комплексу та алгоритми обробки візуальної інформації. Розроблений гібридний алгоритм розпізнавання поєднує переваги глибокого навчання з точністю класичних геометричних методів, забезпечуючи адаптивність до різних режимів роботи залежно від вимог до точності конкретного застосування.

У третьому розділі виконано програмну імплементацію та проведено модульні випробування на доступній базі. Незважаючи на більш важкі умови, властиві відображенню на екрані (піксельна структура, підсвітка, можливі муари), отримано показники, що підтверджують працеспроможність рішення. Так, експериментальна перевірка у стендових умовах з використанням монітора як віртуального полігону підтвердила відповідність системи заявленим вимогам. На робочих відстанях від 0,2 метра до 0,6 метра досягнуто recall від 95 % до 100 %, час обробки стабільно утримується у діапазоні від 400 мс до 500 мс, система демонструє прийнятну робастність до кутів відхилення до 15° (recall 71 %) та здатна працювати з маркерами на складному фоні (recall від 77 % до 88 %). Важливо, що виявлені обмеження (деградація recall до 80 % на відстані 0,8 м, зниження до 65 % при куті 30°) є очікуваними для обраної конфігурації та можуть бути покращені через доопрацювання моделі на розширеному датасеті або збільшення вхідної роздільності за рахунок деякого зниження швидкості обробки.

Наукова новизна роботи полягає у демонстрації практичної реалізації детекції навігаційних маркерів на ультра-бюджетній платформі ESP32-CAM за допомогою квантизованої MobileNet v2, із збереженням прийнятної точності й латентності при повністю локальній обробці. Запропонована структурна й алгоритмічна інтеграція (edge inference у поєднанні з компактними геометричними процедурами замість повноцінного OpenCV-пайплайна) показала, що бар’єр малої пам’яті та обчислень може бути знятий без переходу до дорожчих обчислювальних модулів. Практична цінність рішення у

радикальному зниженні вартості навігаційного сенсорного блоку (плата варістю до 5 $, маркери – до 0,5 $) та у гнучкості інфраструктури (перерозміщення/оновлення маркерів без будівельних робіт), що особливо важливо для невеликих складів і швидкозмінних планувань.

Водночас робота відверто фіксує межі виконаних випробувань: тестування велося як модульна перевірка підсистеми розпізнавання без повного замкненого контуру навігації. Не оцінювалися тривалі тести у динамічних сценах із рухомими об’єктами та інтенсивним природним світлом, не проводилася комплексна перевірка на друкованих маркерах уздовж протяжних проходів. Ці аспекти визначають логічні напрями продовження: друк і стендові полігонні випробування, розширення і аугментація датасету під складські фони, а також інтеграція з одометрією/IMU і модулем планування для перевірки у змішаних сценаріях.

Загалом мета роботи досягнута. Спроєктовано, реалізовано і експериментально підтверджено працездатність розпізнавання інформаційних позначок на ESP32-CAM з характеристиками, достатніми для інтеграції в навігаційну систему мобільного робота на складі. Отримані результати підтверджують, що edge-підхід із легкими нейромережами дійсно зменшує поріг входу за вартістю, спрощує розгортання та підтримку, не вимагає дорогих сенсорів. Запропоноване рішення є відправною точкою для переходу до польових випробувань і масштабування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки з підготовки кваліфікаційної роботи магістра спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» / Упоряд.: І.Ш. Невлюдов, А.О. Андрусевич, О.В. Токарєва, С.П. Новоселов, О.В Сичова. Харків: ХНУРЕ, 2025. – 55 с.

2. Положення про організацію освітнього процесу у ХНУРЕ [електронний ресурс]: Наказ ХНУРЕ від 27 листопада 2020 р. No 400. – Режим доступу: https://nure.ua/wp- content / uploads / Main _ Docs _ NURE/polozhennja-proorganiza-ciju-osvitnogo-procesu-v-hnure.pdf.

3. ДСТУ 3008: 2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. К.: ДП "УкрНДНЦ". 2016. 30 с.

4. Невлюдов І. Ш. Виробничі процеси та обладнання об'єктів автоматизації: Підручник для студентів вищих навчальних закладів / І. Ш. Невлюдов та інш. Кривий Ріг: Криворізький коледж НАУ, 2017 р. – 444 с.

5. Невлюдов І.Ш., Олександров Ю.М., Андрусевич А.О., Чала О.О. Основи наукових досліджень: Навч. посібник. Prague, OKTAN PRINT, 2024 – 468 с. з іл. ISBN 978-80-88618-68-3, e-book ISBN 978-80-88618-60-0, DOI: 10.46489/ONDNP-24-12. Невлюдов І.Ш. Виробничі процеси та обладнання об'єктів автоматизації: Підручник / Кривий Ріг: КК НАУ. – 2017. – 444 с.

6. Невлюдов І. Ш. Моделі та методи кіберфізичних виробничих систем в концепції Industry 4.0. / І. Ш. Невлюдов, та інші. // Монографія. – Харків. – 2023. – 321 с.

7. Nevliudov I. GUI Elements and Windows Form Formalization Parameters and Events Method to Automate the Process of Additive CyberDesign CPPS Development / I. Nevliudov, I., et al. // Advances in Dynamical Systems and Applications. – 2021. – 16(2). – С. 441-455

8. Невлюдов І.Ш. Техніко-економічне обґрунтування інтелектуальному виробництві / І.Ш. Невлюдов // Підручник інженерних Кривий Ріг: видавець Чернявський Д.О. – 2024. – 388 с.

9. Yechevskyi, A. D. Research of orientation methods of autonomous mobile robots in industrial conditions / A. D. Yechevskyi, S. V. Sotnik // «Computer-integrated technologies, automation and robotics» CITAR-2025. – 2025. – pp. 115-119

10. Yechevskyi, A. Methods Of Identification Of Objects On Industrial Lines / A. Yechevskyi, S. Sotnik // International Journal of Academic Engineering Research (IJAER), 2024. – Vol. 8, Issue 11. – pp. 48-55

11. Automated Guided Vehicles AGV for Material Handling - Swisslog EMEA. Swisslog EMEA. URL: https://www.swisslog.com/en-gb/products-systems-solutions/transport-conveyor-systems/agv-automated-guided-vehicles (дата звернення: 03.10.2025).

12. Top modules for the MiR100 made by Nord Modules. Nord Modules. URL: https://www.nord-modules.com/the-mir100/ (дата звернення: 07.10.2025).

13. MIR 100 | Simplimatic Automation. Simplimatic Automation. URL: https://www.simplimatic.com/automations/mir-100/ (дата звернення: 07.10.2025).

14. GreyOrange: Butler Artificial Intelligence-Enabled Autonomous Robot. Material Handling 24/7. URL: https://www.materialhandling247.com/product/butler_artificial_intelligence_enabled_autonomous_robot/GreyOrange (дата звернення: 09.10.2025).

15. Freight 100 – Synertech Automation. Synertech Automation – Where distribution meets integration. URL: https://synertechusa.com/product/freight-100/ (дата звернення: 15.10.2025).

16. Yechevskyi, A. Analysis of the data collection process about products at different stages of production / A. Yechevskyi, S. Maksymova, S. Sotnik // Manufacturing & Mechatronic Systems 2025: Proceedings of IX st International Conference, Kharkiv, October 25-26, 2025: Thesises of Reports. – 2025. – pp. 38-41

17. Yechevskyi, A. D. Smart traffic lights: technology of future for modern cities / A. D. Yechevskyi // Collection of Students' Scientific Paper «Automation and

Development Of Electronic Devices» ADED-2024 Part 2 (Key infrastructure 2024) - Kharkiv/ The Editorial.: Nevlyudov I.Sh. (head), that all. Kharkiv: Kind of Kharkiv National Universitety of Radio Elektronics [electronic edition], 2024. – Vol. 2. – P. 64-70

18. Єчевський А. Д. Система моніторингу та управління параметрами мікроклімату в офісних приміщеннях. Автоматизація та Приладобудування («Automation and Development of Electronic Devices» ADED-2023) [Електронний ресурс] : збірник студентських наукових статей / Харківський національний університет радіоелектроніки ; [редкол.: І.Ш. Невлюдов та ін.]. – Харків : ХНУРЕ, 2023. – Вип. 2. – с. 159-163.

19. Yechevskyi A. D. Analysis of collecting data process on products at different stages of production. Стан, досягнення та перспективи інформаційних систем і технологій /Матеріали XXIV Всеукраїнської науково-технічної конференції молодих вчених,аспірантів та студентів. Одеса, 18-19 квітня 2024 р. – Одеса, Видавництво ОНТУ,2024 р. – с. 84-86.

20. Yechevskyi A.D. System of monitoring and control of microclimate parameters in office premises / A.D. Yechevskyi // Автоматизація та Приладобудування («Automation and Development of Electronic Devices» ADED-2024) [Електронний ресурс]: збірник студентських наукових статей / Харківський національний університет радіоелектроніки; [редкол.: І.Ш. Невлюдов та ін.]. – Харків: ХНУРЕ, 2024. – Вип. 1. – С. 122-126.

21. ДБН В.2.5-28:2018 Природне і штучне освітлення. БУДСТАНДАРТ Online - нормативні документи будівельної галузі України. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=79885 (дата звернення: 01.06.2024).

22. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень. БУДСТАНДАРТ Online - нормативні документи будівельної галузі України. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=14283 (дата звернення: 01.06.2024).