

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
СИСТЕМИ ВИЗНАЧЕННЯ ОБ'ЄКТІВ ДЛЯ НАЗЕМНОГО ДРОНУ
RESEARCH AND SOFTWARE IMPLEMENTATION
OF AN OBJECT DETECTION SYSTEM FOR A GROUND DRONE

Виконав: студент 2 курсу, групи КНТ-113м
Спеціальності 121 Інженерія програмного
забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

ДОВГАЛЬ Д.Ф.

(ПРІЗВИЩЕ та ініціали)

Керівник ПАРХОМЕНКО А.В.

(ПРІЗВИЩЕ та ініціали)

Рецензент ЗЕЛЕНЬОВА І.Я.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет комп'ютерних наук та технологій
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ДОВГАЛЯ Дмитра Федоровича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація системи визначення об'єктів для наземного дрону Research and Software Implementation of an Object Detection System for a Ground Drone

керівник проєкту (роботи) к.т.н., доцент, ПАРХОМЕНКО Анжеліка Володимирівна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “07” жовтня 2024 року № 398

2. Строк подання студентом проєкту (роботи) 4 грудня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Дослідження технологій реалізації комп'ютерного зору для мобільних платформ з обмеженими ресурсами. 3. Проєктування програмного забезпечення. 4. Реалізація програмного забезпечення. 5. Тестування та експериментальне дослідження розробленої системи.

5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	ПАРХОМЕНКО А.В., доцент		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання «07» жовтня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Дослідження технологій реалізації комп'ютерного зору для мобільних платформ з обмеженими ресурсами	2 тиждень	Розділ 2
4	Проектування програмного забезпечення	2 тиждень	Розділ 3
5	Реалізація програмного забезпечення	3-7 тиждень	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	7-10 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї.	11 тиждень	Додатки
8	Нормоконтроль та рецензування.	12 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

(підпис) Дмитро ДОВГАЛЬ
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Анжеліка ПАРХОМЕНКО
(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
90 с., 7 табл., 38 рис., 2 дод., 52 джерела.

ІДЕНТИФІКАЦІЯ ОБ'ЄКТІВ, ESP32-CAM, МАШИННЕ НАВЧАННЯ, КОМП'ЮТЕРНИЙ ЗІР, НАЗЕМНИЙ ДРОН, PYTHON, OPENCV.

Об'єкт дослідження – ідентифікація об'єктів за допомогою програмно-апаратного забезпечення наземного дрону.

Предмет дослідження – методи та алгоритми обробки зображень і машинного навчання для ідентифікації об'єктів в умовах обмежених обчислювальних ресурсів мобільних платформ.

Мета роботи – розробити програмне забезпечення прототипу системи визначення об'єктів, що дозволяє ідентифікувати об'єкти у реальному часі з використанням алгоритмів комп'ютерного зору для мобільних платформ із обмеженими ресурсами.

Матеріали, методи та технічні засоби: плата розробки ESP32-Cam, мобільна платформа (наземний дрон), базовий комплект сенсорів для забезпечення стабільності руху платформи, алгоритми комп'ютерного зору, Python, бібліотека OpenCV, методи машинного навчання для розпізнавання об'єктів.

Результати. Розроблено та протестовано програмне забезпечення прототипу системи визначення об'єктів на ESP32-Cam для наземного дрону.

Висновки. Результати досліджень підтвердили можливість застосування розробленого програмного забезпечення для задач ідентифікації об'єктів, але потрібно вжити додаткових заходів для підвищення точності та швидкості обробки зображень в умовах обмежених обчислювальних ресурсів.

Галузь використання – охорона та безпека, екологічний моніторинг.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 90 pages, 7 tables, 38 figures, 2 appendixes, 52 sources.

OBJECT IDENTIFICATION, ESP32-CAM, MACHINE LEARNING, COMPUTER VISION, GROUND DRONE, PYTHON, OPENCV.

The object of the research is object identification using the software and hardware of a ground drone.

The subject of the study is methods and algorithms of image processing and machine learning for object identification in conditions of limited computing resources of mobile platforms.

The aim of the study is to develop software for a prototype object detection system that allows identifying objects in real time using computer vision algorithms for mobile platforms with limited resources.

Materials, methods and equipment: ESP32-Cam development board, mobile platform (ground drone), basic set of sensors to ensure the stability of the platform, computer vision algorithms, Python, OpenCV library, machine learning methods for object recognition.

Results. The software of the prototype object detection system on ESP32-Cam for a ground drone was developed and tested.

Conclusions. The research results confirmed the possibility of using the developed software for object identification tasks, but additional measures should be taken to improve the accuracy and speed of image processing in conditions of limited computing resources.

Application areas: safety and security, environmental monitoring.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Дослідження програмних компонентів систем для рухомих мобільних платформ	10
1.2 Огляд існуючих програмних рішень на системному рівні для наземних дронів.....	12
1.3 Огляд існуючих систем симуляцій для наземних дронів	14
1.4 Огляд методів комп'ютерного зору для застосування в системах наземних дронів.....	15
1.5 Постановка задач дипломної роботи	18
2 Дослідження технологій реалізації комп'ютерного зору для мобільних платформ з обмеженими ресурсами.....	20
2.1 Дослідження мобільних платформ з обмеженими ресурсами	20
2.2 Дослідження підходів обробки зображень для мобільних платформ ...	23
2.3 Дослідження інтеграції технологій машинного навчання для мобільних платформ з обмеженими ресурсами.....	27
2.4 Висновки за розділом 2	31
3 Проєктування програмного забезпечення	32
3.1 Аналіз вимог до програмного забезпечення	32
3.2 Вибір технологій та інструментів розробки програмного забезпечення	35
3.3 Архітектура програмного забезпечення	37
3.4 Розробка архітектури модулів комп'ютерного зору та їх інтеграції	41
3.5 Висновки за розділом 3	43
4 Реалізація програмного забезпечення	44
4.1 Програмування модуля ESP32-Cam.....	44
4.2 Реалізація дистанційного керування наземним дроном	50

4.3 Програмна реалізація алгоритму на основі моделі комп'ютерного зору YOLO для ESP32-Cam	51
4.4 Реалізація вебзастосунку для відображення відеопотоку.....	56
4.5 Висновки за розділом 4	57
5 Тестування та експериментальне дослідження розробленої системи	58
5.1 Тестування системи	58
5.2 Експериментальне дослідження системи	61
5.3 Напрямки подальшого розвитку системи.....	63
5.4 Висновки за розділом 5	64
Висновки	66
Перелік джерел посилань	68
Додаток А Текст програми.....	73
Додаток Б Слайди презентації	81

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ESP-IDF	– Espressif IoT Development Framework;
HOG	– Histogram of Oriented Gradients;
OpenCV	– Open Source Computer Vision Library;
PCL	– Point Cloud Library;
R-CNN	– Region-based Convolutional Neural Networks;
RTOS	– Real-Time Operating System;
ROS	– Robot Operating System;
SSD	– Single Shot Multibox Detector;
SLAM	– Simultaneous Localization and Mapping;
UML	– Unified Modeling Language;
YOLO	– You Only Look Once.

ВСТУП

На сьогоднішній день розробка програмного забезпечення для систем визначення об'єктів для мобільних платформ із обмеженими ресурсами зумовлена зростаючим попитом на автономні роботизовані системи в різних галузях, таких як безпека, логістика, сільське господарство та ін. Наземні дрони, оснащені камерами та сенсорами, все частіше використовуються для виконання складних завдань, зокрема для виявлення перешкод, картографування та аналізу навколишнього середовища. Оскільки ці платформи часто мають обмежені ресурси, особливо в плані обчислювальної потужності та енергоспоживання, важливо розробляти програмні рішення, що дозволяють досягти достатнього рівня визначення та розпізнавання об'єктів при мінімальних витратах.

Використання алгоритмів комп'ютерного зору разом з оптимізованими програмними рішеннями забезпечує швидку та точну обробку даних у реальному часі. Такий підхід дозволяє мобільним платформам з обмеженими апаратними ресурсами виконувати складні завдання, такі як виявлення об'єктів, розпізнавання перешкод та навігація. Оптимізація програмного забезпечення для малопотужних пристроїв дає змогу ефективно обробляти зображення, знижуючи навантаження на процесор та мінімізуючи енергоспоживання. Це дозволяє автономним системам працювати в складних умовах, підвищуючи точність та швидкість при виконанні завдань, таких як моніторинг середовища та адаптація до змін у навколишньому середовищі. Тому розробка програмного забезпечення системи визначення об'єктів є актуальною задачею.

Мета даної дипломної кваліфікаційної роботи – розробити програмне забезпечення прототипу системи визначення об'єктів, що дозволяє ідентифікувати об'єкти у реальному часі з використанням алгоритмів комп'ютерного зору для мобільних платформ із обмеженими ресурсами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження програмних компонентів систем для рухомих мобільних платформ

Розглянемо основні компоненти та архітектуру програмного забезпечення для рухомих мобільних платформ.

Архітектура програмного забезпечення для рухомих мобільних платформ (наприклад, роботів-колісників, гусеничних або автономних транспортних засобів) зазвичай побудована за модульним принципом і складається з наступних основних компонентів [1],[2]:

- рівень апаратного забезпечення, що забезпечує інтерфейс із сенсорами, приводами та іншими фізичними компонентами платформи (контролери, сенсори, модулі живлення, мотори-колеса, серводвигуни тощо);

- системний рівень або middleware – відповідає за інтеграцію апаратного забезпечення та програмного забезпечення, забезпечуючи стандартизовані інтерфейси та обмін даними (операційна система реального часу FreeRTOS [3], Zephyr [4], RTEMS [5], робототехнічні фреймворки ROS/ROS2 [6]);

- рівень контролю, що відповідає за прийняття рішень та керування платформою (PID-регулятори для руху [7], програмування руху по траєкторії та уникнення перешкод);

- рівень локалізації та навігації, що забезпечує визначення місця знаходження платформи та планування траєкторій (алгоритми локалізації, Simultaneous Localization and Mapping, SLAM [8] та навігації, такі як алгоритми A* [9] та Dijkstra для планування шляху [10]);

- рівень обробки даних від сенсорів – обробка зображень за допомогою програмних рішень для комп'ютерного зору, фільтрація даних за допомогою фільтра Кальмана [11], розпізнавання об'єктів;

– рівень взаємодії – побудова інтерфейсів для моніторингу та управління платформою (веб або мобільні застосунки, використання інтерфейсів віддаленого доступу, обмін даними через протоколи MQTT [12]).

Загальну архітектуру програмного забезпечення для рухомих мобільних платформ наведено на рис. 1.1.

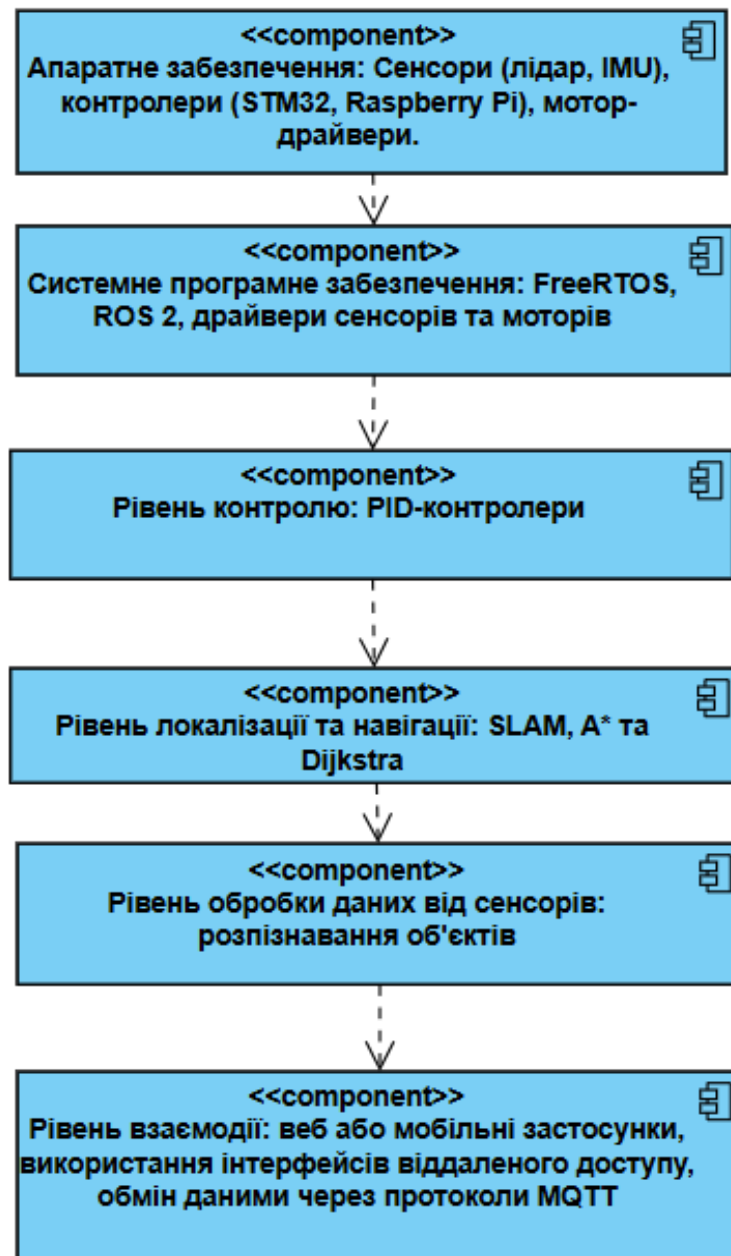


Рисунок 1.1 – Загальна архітектура програмного забезпечення для рухомих мобільних платформ

Наведена архітектура може бути адаптована під конкретну мобільну платформу (наземний дрон, квадрокоптер тощо).

Виділимо основні компоненти програмного забезпечення для рухомих мобільних платформ, таких як наземні дрони, що необхідні для реалізації задач автономної навігації, виявлення та ідентифікації об'єктів, обробки даних у реальному часі [1],[2]:

- система керування рухом – відповідає за управління моторними функціями платформи, включаючи регулювання швидкості, напрямку та маневрування, а саме планування траєкторії – розрахунок оптимального маршруту на основі цілей або отриманих даних та регулювання руху – контроль виконання маневрів із використанням зворотного зв'язку від датчиків;

- модуль обробки зображень: використовується для збору та аналізу візуальної інформації з камер, встановлених на платформі з метою визначення об'єктів та виявлення можливих перешкод;

- модуль інтеграції сенсорів, що включає обробку даних із різних датчиків, таких як GPS, LiDAR, ультразвукові та інерційні модулі з метою моніторингу та побудови карти навколишнього середовища;

- модуль взаємодії з користувачем, що дозволяє оператору віддалено керувати платформою або налаштовувати автономні режими та реалізується через вебінтерфейси, мобільні застосунки або спеціалізовані пульти.

Існують також програмні системи, які забезпечують підтримку різноманітних функцій для наземних дронів.

1.2 Огляд існуючих програмних рішень на системному рівні для наземних дронів

На сьогодні програмні рішення FreeRTOS та ROS є одними з найпоширеніших інструментів для розробки програмного забезпечення на системному рівні для рухомих мобільних платформ [3], [6].

FreeRTOS – це операційна система реального часу (Real-Time Operating System, RTOS), яка використовується для вбудованих систем, зокрема для мобільних роботів, таких як дрони [3]. Вона забезпечує багатозадачність, управління часом реального виконання та ресурси для обмежених апаратних платформ.

Можна виділити наступні переваги FreeRTOS:

- FreeRTOS є дуже легким і малоресурсоємним, що дозволяє працювати на платформах з обмеженими ресурсами, таких як ESP32, STM32 і інші мікро контролери;
- підтримка різних архітектур, даючи змогу адаптувати систему під конкретні вимоги;
- широка підтримка спільноти: FreeRTOS має велику спільноту розробників, що дозволяє отримати підтримку і готові рішення для багатьох застосувань.

FreeRTOS відноситься до системи керування ресурсами на рівні операційної системи реального часу. Це компонент, що забезпечує основні функції управління завданнями та часом реального виконання.

ROS (Robot Operating System) – це відкрита платформа, яка широко використовується у сфері робототехніки, включаючи наземні дрони [6]. Вона забезпечує інфраструктуру для створення програмного забезпечення, інтеграції датчиків та обробки даних.

ROS2 є операційною системою роботів, яка забезпечує інфраструктуру для управління та обробки різних компонентів роботизованих платформ, таких як датчики, виконавчі механізми та алгоритми для автономної навігації.

Наведемо основні можливості ROS:

- інтеграція сенсорних даних, включаючи LiDAR, GPS, ультразвукові сенсори тощо;
- підтримка бібліотек для комп'ютерного зору (OpenCV);
- можливість реалізації модулів руху та алгоритмів SLAM.

ROS2 забезпечує зв'язок між різними частинами робота через стандартизовані інтерфейси, що дозволяє інтегрувати різні апаратні і програмні компоненти в єдину систему. Це включає управління ресурсами, зв'язок між модулями, а також підтримку паралельного виконання задач і розподілених обчислень. Одночасно ROS2 виступає як платформа для створення та виконання програмного забезпечення для роботів. Вона включає в себе багатозадачність, планування руху, обробку сенсорних даних, управління мережею та інші важливі аспекти роботи роботизованих систем

Отже, FreeRTOS використовується в системах з обмеженими ресурсами для забезпечення багатозадачності та керування обчислювальними ресурсами, в той час як ROS (особливо його версія ROS2) надає широку інфраструктуру для розробки і інтеграції різних компонентів роботизованих платформ, таких як сенсори, виконавчі механізми і алгоритми навігації

1.3 Огляд існуючих систем симуляцій для наземних дронів

Розглянемо наступні системи симуляції роботів Webots та Gazebo, що дозволяють розробляти програмні рішення для мобільних платформ, таких як наземні дрони.

Webots – це система симуляції роботів, що дозволяє розробляти, тестувати та моделювати роботів для різних платформ [13], що широко використовується для навчання та досліджень у галузі робототехніки, включаючи роботів, які використовуються на мобільних платформах, таких як наземні дрони.

Webots належить до симуляційного рівня в архітектурі робототехнічних систем і представляє собою інструмент для моделювання та тестування роботів, що дозволяє перевіряти алгоритми без необхідності фізичних експериментів.

Можна виділити наступні переваги Webots:

- моделювання фізики та сенсорів: Webots має вбудовані моделі для фізичних законів і сенсорних даних, що дозволяє точно відтворювати реальні умови;
- підтримка багатьох типів роботів: мобільних роботів, маніпуляторів, автономних систем;
- платформа дозволяє інтегрувати моделі з ROS та іншими програмними середовищами.

Gazebo – це система симуляції роботів, що використовується для тестування роботів у фізичних середовищах [14]. Він інтегрується з ROS і дозволяє проводити віртуальні експерименти для мобільних платформ, таких як наземні роботи та дрони.

Програмне рішення Gazebo належить до симуляційного рівня. Це інструмент, який моделює фізичне середовище та взаємодію роботів, що дозволяє розробникам тестувати і перевіряти алгоритми перед їх впровадженням на реальні платформи.

Можна виділити наступні переваги Gazebo:

- Gazebo використовує фізичні моделі, що дозволяє точно симулювати рух, взаємодію з об'єктами і навколишнім середовищем;
- підтримка великих та складних сцен з великою кількістю агентів і об'єктів;
- інтеграція з ROS, що дозволяє проводити тестування алгоритмів навігації та інших функцій роботів.

1.4 Огляд методів комп'ютерного зору для застосування в системах наземних дронів

Огляд програмних рішень таких як ROS та систем симуляції роботів Webots та Gazebo щодо реалізації, інтеграції та використання методів комп'ютерного зору показав наступне.

ROS2 є ключовою платформою для інтеграції методів комп'ютерного зору в робототехнічні системи, дозволяючи створювати, тестувати та реалізовувати алгоритми автономної навігації, розпізнавання об'єктів і аналізу середовища.

Наявні можливості використання та інтеграції методів комп'ютерного зору:

- можливість інтеграції з OpenCV [15] для обробки відео та зображень у реальному часі;
- використання алгоритмів відстеження об'єктів, таких як фільтр Кальмана і оптичний потік, для забезпечення точного навігаційного контролю [16];
- використання бібліотек Point Cloud Library (PCL) для роботи з тривимірними картами та даними з LiDAR або стерео камер [17], [18].

Webots це платформою для симуляції роботів, що дозволяє тестувати алгоритми комп'ютерного зору в змодельованому середовищі з високим рівнем деталізації.

Наявні можливості використання та інтеграції методів комп'ютерного зору:

- симуляція камер і сенсорів для збирання даних;
- реалізація алгоритмів виявлення об'єктів із використанням моделей нейронних мереж;
- тестування методів сегментації зображень та відстеження траєкторій об'єктів для підтримки навігації дронів у складних середовищах [17], [19].

Gazebo використовується для високоточних симуляцій робототехнічних систем, що інтегрують методи комп'ютерного зору з реалістичною фізикою.

Наявні можливості використання та інтеграції методів комп'ютерного зору:

- використання симуляцій камер із високою роздільною здатністю для аналізу об'єктів;

– інтеграція з ROS2 для відпрацювання методів SLAM, таких як тривимірна реконструкція середовища [20];

– тестування алгоритмів глибокого навчання для одночасного виявлення та класифікації об'єктів у складних умовах [21], [22].

Порівняльну характеристику застосування комп'ютерного зору в ROS2, Webots і Gazebo наведено в табл.1.1.

Таблиця 1.1 – Порівняльна характеристика застосування комп'ютерного зору в ROS2, Webots і Gazebo [15] - [22]

Критерій	ROS2	Webots	Gazebo
1	2	3	4
Методи розпізнавання	виявлення об'єктів через інтеграцію з OpenCV; алгоритми глибокого навчання для класифікації	моделювання камер для імітації розпізнавання об'єктів та застосування сегментації зображень	інтеграція алгоритмів нейронних мереж для виявлення об'єктів та використання симульованих LiDAR та камер
Методи відстеження	фільтр Кальмана для прогнозування; оптичний потік для аналізу руху	моделювання відстеження траєкторії об'єктів у віртуальному середовищі	реалістична симуляція динаміки об'єктів для тестування алгоритмів
Методи 3D реконструкції	інтеграція SLAM для побудови карти навколишнього середовища	стереозір і моделювання тривимірної геометрії об'єктів	SLAM з використанням симульованих даних сенсорів

Кінець табл.1.1

1	2	3	4
Сценарії тестування	виявлення та класифікація у реальних умовах	тестування алгоритмів у контрольованих віртуальних середовищах	тестування в реалістичних середовищах із моделюванням фізики
Інтеграція з інструментами	TensorFlow, PyTorch, та OpenCV	взаємодія з ROS2 для експорту алгоритмів у реальні системи	підтримка модулів ROS2 для реального впровадження алгоритмів

З табл.1.1 можна зробити висновок, що ROS2 найкраще підходить для розробки та впровадження алгоритмів комп'ютерного зору у реальних системах. Система симуляції Webots орієнтована на початкове тестування та візуалізацію алгоритмів у контрольованих умовах. А Gazebo надає потужні інструменти для моделювання складних сценаріїв із високим рівнем реалізму, що ідеально підходить для перевірки алгоритмів перед їх використанням у реальних умовах.

1.5 Постановка задач дипломної роботи

В розділі було виконано дослідження програмних компонентів систем для рухомих мобільних платформ, наведена загальна архітектура програмного забезпечення для рухомих мобільних платформ, проведено огляд існуючих програмних рішень на системному рівні для наземних дронів, існуючих систем симуляцій для наземних дронів, методів комп'ютерного зору для застосування в системах наземних дронів.

Для реалізації програмного забезпечення прототипу системи визначення об'єктів необхідно провести огляд та обґрунтування технологій та

інструментів розробки: використання наявного програмного забезпечення на системному рівні для рухомих мобільних платформ або розробка власного програмного забезпечення. Таке рішення повинно враховувати доступні обчислювальні ресурси та можливість масштабування системи.

Мета даної дипломної кваліфікаційної роботи – розробити програмне забезпечення прототипу системи визначення об'єктів, що дозволяє ідентифікувати об'єкти у реальному часі з використанням алгоритмів комп'ютерного зору для мобільних платформ із обмеженими ресурсами.

Для досягнення поставленої мети в дипломній кваліфікаційній роботі потрібно вирішити наступні задачі:

- дослідити мобільні платформи з обмеженими ресурсами;
- дослідити підходи обробки зображень для системи визначення об'єктів для мобільних платформ з обмеженими ресурсами;
- дослідити можливість інтеграції технологій машинного навчання для мобільних платформ з обмеженими ресурсами;
- обґрунтувати вибір технологій і інструментів розробки програмного забезпечення;
- розробити архітектуру програмного забезпечення для системи визначення об'єктів;
- розробити архітектуру модулів комп'ютерного зору та виконати їх інтеграцію в систему визначення об'єктів;
- виконати реалізацію дистанційного керування наземним дроном;
- виконати програмну реалізацію алгоритму на основі моделі комп'ютерного зору для обраної мобільної платформи;
- розробити прототип системи визначення об'єктів;
- провести тестування та експериментальне дослідження розробленого програмного забезпечення для системи визначення об'єктів.

2 ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ РЕАЛІЗАЦІЇ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ МОБІЛЬНИХ ПЛАТФОРМ З ОБМЕЖЕНИМИ РЕСУРСАМИ

2.1 Дослідження мобільних платформ з обмеженими ресурсами

Мобільні платформи з обмеженими ресурсами є основою для побудови компактних та доступних систем комп'ютерного зору. Проведемо дослідження найпоширеніших платформ: ESP32-Cam [23], [24], Raspberry Pi 4 [25], Arduino Nano RP2040 [26] і NVIDIA Jetson Nano [27].

Порівняння характеристик обраних платформ з обмеженими ресурсами за функціональними можливостями наведено у табл. 2.1.

Таблиця 2.1 – Порівняльна характеристика обраних платформ з обмеженими ресурсами за функціональними можливостями [23] - [27]

Платформа	Процесор	Пам'ять	Підключення	Наявність камери	Ціна
ESP32-Cam	Tensilica LX6	520 КБ ОЗУ	Wi-Fi, Bluetooth	OV2640 (1600×1200)	\$10-15
Raspberry Pi 4	ARM Cortex-A72	2/4/8 ГБ LPDDR4	Wi-Fi, Bluetooth, GPIO	Raspberry Pi Camera (до 12 Мп)	\$35-70
Arduino Nano RP2040	Dual Cortex-M0+	264 КБ SRAM	USB, GPIO	Через додаткові модулі	\$10-20
NVIDIA Jetson Nano	Cortex-A57, 128 CUDA	4 ГБ LPDDR4	Ethernet, GPIO	MIPI CSI-2	\$99

З таблиці 2.1 можна зробити висновок, що у платформи Arduino Nano RP2040 відсутня камера, підключення камери можливе через додаткові

модулі тому, для подальшого дослідження будемо розглядати платформи ESP32-Cam, Raspberry Pi 4 та NVIDIA Jetson Nano.

Порівняння характеристик обраних платформ з обмеженими ресурсами для реалізації комп'ютерного зору наведено у табл. 2.2.

Таблиця 2.2 – Порівняльна характеристика обраних платформ з обмеженими ресурсами для реалізації комп'ютерного зору [23] - [25], [27]

Платформа	Підтримка відеопотоку	Споживання енергії	Підтримка TensorFlow, PyTorch, OpenCV	Області застосування
ESP32-Cam	160×120 - 1600×1200, до 30 FPS	0.6-0.9 Вт	TensorFlow Lite Micro, OpenCV (обмежена)	ІоТ-системи, спостереження, портативні системи моніторингу
Raspberry Pi 4	640×480 - 4K, до 60 FPS	5-8 Вт	TensorFlow, PyTorch, OpenCV	Робототехніка, домашня автоматизація, мультимедійні системи
NVIDIA Jetson Nano	1080p - 4K, до 30 FPS	10-15 Вт		Комп'ютерний зір, глибоке навчання, складні системи робототехніки

Згідно критерію підтримка TensorFlow, PyTorch, OpenCV з табл. 2.2 можна зробити наступний висновок:

- у платформи ESP32-Cam підтримка обмежена: TensorFlow Lite Micro дозволяє працювати лише зі спрощеними моделями, а OpenCV застосовується для базової обробки кадрів. Це зумовлено низькою обчислювальною потужністю та обмеженим обсягом пам'яті;

- платформа Raspberry Pi 4 має повноцінну підтримку TensorFlow, PyTorch і OpenCV, що дозволяє реалізовувати складні алгоритми комп'ютерного зору, включаючи навчання та інференс;

- NVIDIA Jetson Nano є оптимальною платформою для TensorFlow, PyTorch і OpenCV завдяки високій обчислювальній потужності та апаратній підтримці GPU, що забезпечує прискорення нейронних мереж.

Якщо враховувати параметр ціна, то ESP32-Cam є найдешевшою платформою з вартістю \$10–15, що робить її привабливим вибором для низькобюджетних проєктів. Raspberry Pi 4 має середню цінову категорію (\$35–70), пропонуючи гнучкість і можливість реалізації складніших проєктів. NVIDIA Jetson Nano належить до вищого цінового сегмента (\$99), але її висока потужність виправдовує витрати для проєктів із високими вимогами до обчислень.

Таким чином, вибір платформи залежить від бюджету проєкту та вимог до підтримки алгоритмів машинного навчання. ESP32-Cam підходить для бюджетних проєктів, де потрібна мінімальна обробка зображення та віддалене керування. Raspberry Pi підійде для складніших завдань, які вимагають високої обчислювальної потужності та можливостей роботи з великими камерами. Jetson Nano підходить для проєктів з машинним навчанням, де потрібна висока продуктивність та обробка відео в реальному часі.

Для проведення дослідження було вирішено обрати платформу ESP32-Cam, оскільки необхідно реалізувати систему визначення об'єктів із обмеженим використанням апаратних ресурсів.

ESP32-Cam забезпечує необхідну функціональність завдяки компактному дизайну, енергоефективності та достатній обчислювальній

потужності для виконання базових задач комп'ютерного зору. Крім того, її низька вартість та доступність роблять платформу оптимальним вибором для розробки систем із обмеженим бюджетом. Платформа підтримує обробку відеопотоку за допомогою вбудованого модуля камери OV2640 та передачу даних через Wi-Fi. Це дозволяє інтегрувати її з іншими системами, передавати відеопотік на сервер для подальшої обробки та реалізовувати задачі, які потребують мінімального енергоспоживання.

2.2 Дослідження підходів обробки зображень для мобільних платформ

Використання підходів обробки зображень для системи визначення об'єктів для мобільних платформ є важливим компонентом, особливо з урахуванням обмежених ресурсів мобільних платформ.

Основними підходами є підготовка даних, попередня обробка для поліпшення якості зображень, а також використання алгоритмів виявлення об'єктів.

Підготовка даних для моделі машинного навчання складається з підготовки розміру кадрів та нормалізації зображень [28]. Для роботи більшості моделей машинного навчання необхідно, щоб вхідні зображення мали фіксований розмір, це досягається за рахунок масштабування зображення, що дозволяє адаптувати кадри до потрібних розмірів без втрати основної інформації. Нормалізація зображень передбачає приведення значень пікселів до діапазону $[0, 1]$. Це дозволяє уникнути великих розбіжностей між значеннями пікселів та забезпечує стабільну роботу моделі машинного навчання.

Попередня обробка для поліпшення якості зображень включає в себе використання фільтрування шуму та корекцію освітлення.

Фільтрування шуму – зменшення шуму, який може погіршити точність розпізнавання.

Можна виділити наступні фільтри [29]:

- фільтр Гаусса, що застосовується для згладжування зображень і усунення дрібного шуму та працює на основі розподілу Гауса, який визначає значення для кожного пікселя залежно від його сусідів;
- медіанний фільтр, що використовується для видалення шуму зображень, зберігаючи при цьому різкі краї. Особливо ефективний для зображень із точковим шумом.

Корекція освітлення це вирівнювання яскравості та контрастності зображення, і виконується за допомогою вирівнювання гістограм – техніки, що покращує контрастність зображення, рівномірно розподіляючи інтенсивності пікселів, що дозволяє поліпшити якість.

Розглянемо найпоширені алгоритми виявлення об'єктів на зображеннях, які можна реалізувати під мобільні платформи: каскади Хаара [30],[31], Histogram of Oriented Gradients (HOG)[32], You Only Look Once (YOLO) [33], Single Shot Multibox Detector (SSD) [34] та Region-based Convolutional Neural Networks (Faster R-CNN) [35].

Каскади Хаара – це класичний метод для розпізнавання об'єктів [30], [31] який використовує особливості (Haar-like features) для виявлення об'єктів, таких як обличчя, автомобілі тощо. Алгоритм застосовує каскад простих класифікаторів, які послідовно фільтрують кадр, швидко відкидаючи непридатні частини зображення.

HOG – це метод, що дозволяє виділити об'єкти, аналізуючи гістограми орієнтованих градієнтів у локальних ділянках зображення [32]. Цей метод особливо ефективний для виявлення людей у кадрі, використовуючи визначення по фігурі.

YOLO є одним із найпопулярніших для виявлення об'єктів [33]. На відміну від традиційних методів, таких як каскади Хаара, YOLO аналізує зображення в цілому, розбиваючи його на сітку та визначаючи кілька об'єктів одночасно. Цей підхід дозволяє дуже швидко обробляти великі зображення і працювати в реальному часі.

SSD – це ще один швидкий алгоритм для виявлення об'єктів у реальному часі [34]. Він подібний до YOLO, але використовує різні розміри «якорів» для виявлення об'єктів. SSD працює швидше за більшість методів на середніх та низьких роздільних зображеннях.

Faster R-CNN — це більш складна модель для виявлення об'єктів, яка використовує регіональні пропозиції для пошуку потенційних об'єктів [35]. Порівняно з іншими методами, вона має високу точність, але значно більші вимоги до ресурсів.

Порівняння алгоритмів виявлення об'єктів на зображеннях наведено у табл. 2.3.

Таблиця 2.3 – Порівняння алгоритмів виявлення об'єктів на зображеннях [30] – [35]

Алгоритм	Переваги	Недоліки	Підтримка ESP32-CAM	Призначення
1	2	3	4	5
Каскади Хаара	швидкий, можна виділити низькі вимоги до ресурсів	низька точність для складних об'єктів	обмежена	просте виявлення об'єктів, наприклад обличчя людини
HOG	висока точність для виявлення людей	потребує більше ресурсів	обмежена	виявлення людей

Кінець табл. 2.3

1	2	3	4	5
YOLO	висока швидкість і точність	погіршення точності для малих об'єктів	підтримується	виявлення об'єктів у реальному часі
SSD	швидкість обробки на середніх зображеннях	менш точний, ніж YOLO, на великих зображеннях	підтримується	виявлення кількох об'єктів одночасно
Faster R-CNN	дуже точний	потребує великих ресурсів	обмежена	виявлення складних сцен

Було вирішено обрати алгоритм YOLO для реалізації системи визначення об'єктів на платформі ESP32-CAM завдяки швидкості обробки в реальному часі.

Однією з основних переваг YOLO є його здатність виконувати виявлення об'єктів за один прохід через мережу, що забезпечує високу швидкість і дозволяє працювати в реальному часі. Це критично важливо для ESP32-Cam, оскільки платформа має обмежені обчислювальні ресурси, але для її використання необхідно мати можливість обробляти відеопотоки без значних затримок. Також на відміну від інших моделей, таких як Faster R-CNN, які використовують попередньо генеровані регіональні пропозиції для виявлення об'єктів, YOLO здійснює обробку зображення одночасно, не виконуючи складних попередніх етапів, що дозволяє прискорити процес виявлення, що є важливим для роботи на ESP32-Cam.

Попри обмеження апаратних ресурсів (520 КБ оперативної пам'яті, 4 МБ флеш-пам'яті), ESP32-Cam може використовувати оптимізовані версії моделей YOLO, такі як Tiny YOLO або YOLOv4-tiny, які мають зменшену

кількість параметрів і потребують менше пам'яті та обчислювальних потужностей.

Одна з причин вибору YOLO для ESP32-Cam полягає у можливості використання TensorFlow Lite Micro, який підтримує спрощені моделі, зокрема YOLO, що дозволяє проводити інференс на мікроконтролерах з обмеженими ресурсами.

Також YOLO здатна одночасно визначати кілька об'єктів в одному кадрі, що важливо для багатьох реальних застосувань, таких як спостереження за кількома об'єктами одночасно. Це дозволяє використовувати ESP32-Cam для задач, де необхідно одночасно ідентифікувати різні типи об'єктів у відеопотоці.

YOLO має великий набір інструментів і бібліотек для реалізації на різних платформах, зокрема підтримує такі бібліотеки, як OpenCV, що дозволяє інтегрувати його з платформами, такими як ESP32-Cam.

Розглянемо технології машинного навчання для мобільних платформ з обмеженими ресурсами.

2.3 Дослідження інтеграції технологій машинного навчання для мобільних платформ з обмеженими ресурсами

На етапі дослідження було розглянуто наступні технології та фреймворки для вирішення поставлених завдань: ESP-WHO (ESP-IDF), TensorFlow Lite, OpenCV.

ESP-WHO (ESP-IDF) – це спеціалізований фреймворк для комп'ютерного зору, розроблений компанією Espressif для використання на платформі ESP32 [36], [37]. Він є частиною екосистеми ESP-IDF (Espressif IoT Development Framework) і забезпечує інтеграцію з модулями камер, такими як OV2640, для реалізації базових задач обробки зображень.

Основні можливості ESP-WHO:

- розпізнавання облич: підтримка алгоритмів виявлення та ідентифікації облич у реальному часі;
- захоплення відеопотоку: інтеграція з модулями камер для роботи з відео;
- оптимізація для ESP32: фреймворк розроблений спеціально для роботи в умовах обмежених ресурсів ESP32.

TensorFlow Lite – це спрощена версія фреймворку TensorFlow, оптимізована для роботи на пристроях з обмеженими ресурсами, таких як мікроконтролери та мобільні пристрої [38]. Для мікроконтролерів, включаючи ESP32, використовується версія TensorFlow Lite for Microcontrollers.

Основні можливості TensorFlow Lite:

- TensorFlow Lite дозволяє запускати невеликі моделі, такі як Tiny YOLO або моделі для класифікації об'єктів;
- оптимізація для низького енергоспоживання: підходить для пристроїв із мінімальними обчислювальними ресурсами;
- можливість перенесення моделей, створених у TensorFlow, до середовища TensorFlow Lite.

OpenCV (Open Source Computer Vision Library) – це популярна бібліотека для комп'ютерного зору, яка підтримує широкий спектр операцій із зображеннями та відео [15], [39]. Вона широко використовується для обробки відеопотоків, розпізнавання об'єктів і роботи з моделями машинного навчання.

Основні можливості бібліотеки OpenCV наступні:

- обробка зображень та відео: зміна розміру, фільтрація шуму, вирівнювання яскравості та ін.;
- підтримка мови програмування Python;
- OpenCV підтримує завантаження та виконання моделей, створених у TensorFlow, PyTorch або інших фреймворках.

Порівняння технологій та фреймворків для машинного навчання для мобільних платформ з обмеженими ресурсами наведено у табл. 2.4.

Таблиця 2.4 – Порівняння технологій та фреймворків для машинного навчання для мобільних платформ з обмеженими ресурсами [36] – [39]

Технологія (фреймворк)	Переваги	Недоліки	Підтримка ESP32-CAM	Призначення та застосування
1	2	3	4	5
ESP-WHO (ESP-IDF)	спеціалізоване для ESP32, підтримка розпізнавання облич	обмежений набір функцій для складних задач; важко інтегрувати сторонні моделі машинного навчання	обмежена підтримка для складних задач	розпізнавання облич, базові задачі комп'ютерного зору
TensorFlow Lite	оптимізована для пристроїв з низьким енергоспожив анням, використання компактних моделей	обмежена пам'ять ESP32 не дозволяє завантажити навіть мінімальні моделі TensorFlow, а також низька продуктивність при обробці відео	проблеми з інтеграцією на ESP32- CAM	підтримка моделей машинного навчання на мікроконтроле рах

Кінець табл. 2.4

1	2	3	4	5
OpenCV	універсальний фреймворк для комп'ютерного зору, багатий набір функцій для обробки зображень	потрібне потужніше обладнання для обробки відео в реальному часі, що потребує зовнішнього сервера	підтримка для базових задач	обробка зображень та відео, виявлення об'єктів у реальному часі

Порівняння технологій та фреймворків для машинного навчання для мобільних платформ з обмеженими ресурсами дозволяє зробити наступні висновки.

ESP-WHO добре підходить для створення компактних систем для виявлення та розпізнавання облич. Проте, його обмежена функціональність не дозволяє реалізовувати складніші задачі обробки зображень або інтеграцію сучасних алгоритмів машинного навчання.

TensorFlow Lite є гарним рішенням для задач машинного навчання на пристроях з обмеженими ресурсами. Проте, для ESP32-CAM його використання обмежене через недостатній обсяг пам'яті та низьку обчислювальну потужність.

OpenCV є універсальним інструментом для реалізації систем комп'ютерного зору. У поєднанні з платформою ESP32-CAM і Python бібліотека дозволяє здійснювати базову обробку відеопотоків і виконувати розширені операції на зовнішньому сервері, що робить її важливим компонентом у створенні масштабованих систем.

Для реалізації системи з розпізнавання об'єктів за допомогою ESP32-CAM було вирішено використовувати OpenCV для обробки відеопотоку з ESP32-CAM в реальному часі.

2.4 Висновки за розділом 2

У розділі проведено аналіз мобільних платформ із обмеженими ресурсами, методів обробки зображень і технологій машинного навчання. Розглянуто платформи ESP32-CAM, Raspberry Pi 4 і NVIDIA Jetson Nano. На основі порівняльного аналізу обрано ESP32-Cam як оптимальну платформу для розробки системи визначення об'єктів із обмеженим використанням апаратних ресурсів, враховуючи її низьку вартість, компактність і енергоефективність.

Досліджено методи обробки зображень, такі як фільтрація шуму, корекція освітлення та масштабування кадрів, для забезпечення якості вхідних зображень. Серед алгоритмів виділення об'єктів обрано YOLO (Tiny YOLO) завдяки його швидкості та здатності працювати в реальному часі.

Проаналізовано технології ESP-WHO, TensorFlow Lite і OpenCV. Визначено, що OpenCV забезпечує гнучкість і функціональність для реалізації базових завдань комп'ютерного зору, тоді як TensorFlow Lite і ESP-WHO мають обмеження через обчислювальні ресурси ESP32-Cam. OpenCV підтримує YOLO, що дозволяє інтегрувати його з платформами, такими як ESP32-Cam.

Для реалізації системи доцільно використовувати гібридний підхід, де частина обробки (включаючи натреновану модель YOLO) виконується на зовнішньому сервері, що дозволить вирішити питання обмежень апаратної платформи, забезпечуючи обробку відео в реальному часі та збереження точності розпізнавання.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз вимог до програмного забезпечення

Розроблене програмне забезпечення повинно забезпечувати наступні функціональні можливості:

- захоплення відео потоку: отримання зображень у режимі реального часу з камери (ESP32-Cam);
- передача відеопотоку через Wi-Fi;
- виявлення та ідентифікацію об'єктів на зображенні у реальному часі;
- обробка та аналіз даних: класифікація виявлених об'єктів відповідно до заданих класів (люди, транспорт, перешкоди тощо);
- зберігання та передача даних про виявлені об'єкти для подальшої обробки;
- відображення результатів розпізнавання;
- дистанційне керування мобільною платформою (наземним дроном);
- можливість додавання нових алгоритмів або класів об'єктів у майбутньому;
- підтримка роботи з іншими камерами або апаратними платформами.

Також визначимо наступні нефункціональні вимоги: обмежене використання пам'яті (до 520 КБ оперативної пам'яті ESP32-CAM) та робота в умовах низької пропускної здатності Wi-Fi.

Було розроблено діаграми варіантів використання(прецедентів) із використанням мови Unified Modeling Language (UML) на основі визначених функціональних вимог.

UML – це графічна мова моделювання, призначена для визначення, візуалізації та документування моделей об'єктно-орієнтованих програмних систем. Вона широко використовується для проєктування структури та поведінки об'єктно-орієнтованих програм, зокрема для прототипу системи визначення об'єктів на мобільних платформах [40].

Діаграми варіантів використання це інструмент для візуалізації взаємодії між користувачами (акторами) та програмною системою. Вони демонструють функціональні вимоги та поведінку системи з точки зору зовнішніх користувачів.

Для системи визначення об'єктів на мобільних платформах можна виділити наступних акторів: оператор мобільної платформи(наземного дрону), модуль ESP32-Cam, сервер обробки даних.

Діаграма варіантів використання для оператора мобільної платформи(наземного дрону) зображена на рис.3.1.

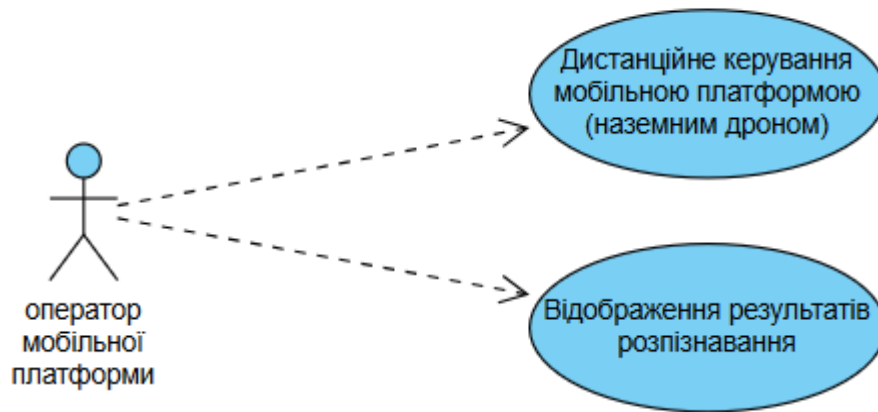


Рисунок 3.1 – Діаграма варіантів використання для оператора мобільної платформи(наземного дрону)

Діаграма варіантів використання для модулю ESP32-Cam зображена на рис. 3.2.

Проектування діаграм варіантів використання було виконано за допомогою онлайн сервісу побудови діаграм Visual Paradigm [41].

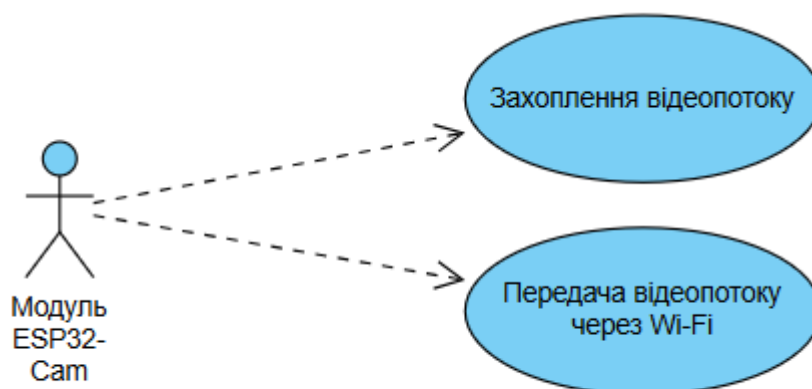


Рисунок 3.2 – Діаграма варіантів використання для модулю ESP32-Cam

Діаграма варіантів використання для серверу обробки даних зображена на рис. 3.3.

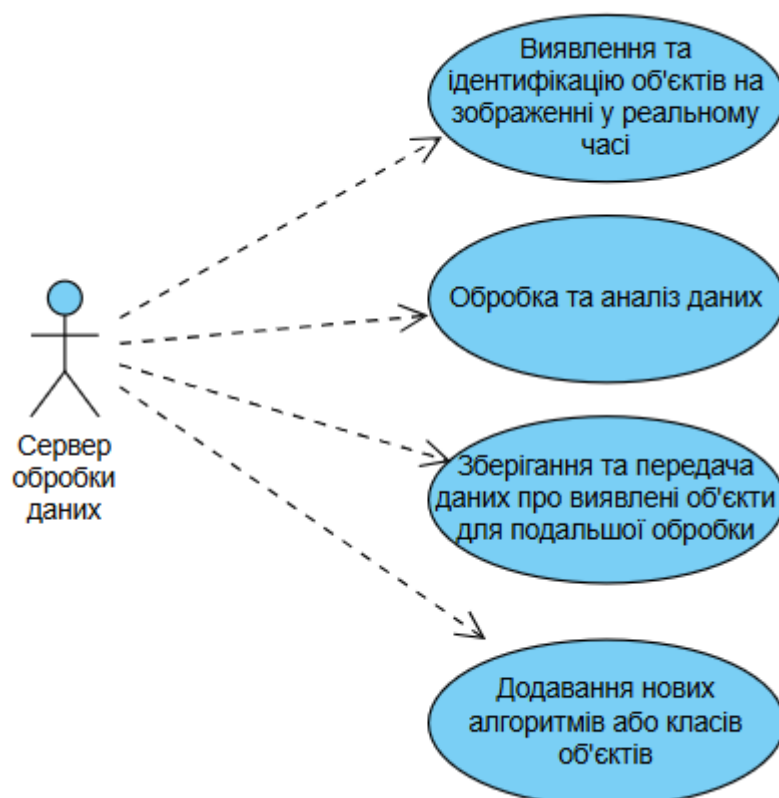


Рисунок 3.3 – Діаграма варіантів використання для серверу обробки даних

3.2 Вибір технологій та інструментів розробки програмного забезпечення

Для розробки програмного забезпечення для системи визначення об'єктів для мобільних платформ на базі ESP32-CAM необхідно обрати технології та інструментів для реалізації основних функцій системи, враховуючи обмеження апаратних ресурсів.

Було вирішено обрати Python як основної мови програмування для обробки відеопотоку за допомогою OpenCV та C/C++ для програмування модуля ESP32-CAM через Arduino IDE.

Мова програмування Python є найпоширеніших інструментів для розробки систем комп'ютерного зору завдяки [42], [43]:

- наявності великої кількості бібліотек, зокрема: OpenCV для обробки зображень і відео та TensorFlow і PyTorch для інтеграції моделей глибокого навчання для алгоритмів обробки даних, відстеження та розпізнавання об'єктів;

- швидкість розробки прототипів систем: Python є високорівневою мовою, що дозволяє швидко створювати прототипи та тестувати алгоритми, наприклад, бібліотека OpenCV надає простий у використанні API для роботи з відеопотоками;

- кросплатформність: Python дозволяє запускати програмне забезпечення на різних операційних системах (Windows, Linux, macOS), що забезпечує гнучкість у виборі апаратної платформи для сервера обробки даних;

- підтримка моделей машинного навчання: TensorFlow і PyTorch Python забезпечує легку інтеграцію натренованих моделей YOLO для виявлення об'єктів.

OpenCV– бібліотека для комп'ютерного зору, що підтримує обробку відео в реальному часі та дозволяє [15], [39]:

– виконувати захоплення відеопотоку за допомогою `cv2.VideoCapture()`;

– виконувати обробку кадрів, таких як зміна розміру, нормалізація та підготовка до інференсу YOLO;

– відображати результати на екрані з графічними елементами (рамки, підписи).

Для програмування модуля ESP32-CAM було обрано мову програмування C/C++ у середовищі розробки Arduino IDE через наступні переваги [44], [45]:

– оптимізація роботи з апаратними ресурсами ESP32-CAM має обмежені апаратні ресурси (520 КБ оперативної пам'яті), що вимагає низькорівневого доступу до функцій апаратури, який забезпечує мова C;

– використання C/C++ дозволяє мінімізувати використання пам'яті та підвищити швидкість виконання коду;

– підтримка бібліотек для ESP32 Arduino IDE надає широкий набір бібліотек, таких як WiFi, для передачі даних через мережу, та бібліотеки для роботи з камерою (ESP32 Camera Web Server);

– простота інтеграції Arduino IDE забезпечує зручну компіляцію та завантаження коду на ESP32-CAM, що дозволяє швидко тестувати програму на реальному пристрої;

– дозволяє налаштувати параметри відеопотоку (роздільна здатність, FPS) відповідно до обмежень апаратної платформи та вимог системи.

Вибір Python з бібліотекою OpenCV як основної технології для обробки відеопотоку забезпечує гнучкість розробки програмного забезпечення. Використання C/C++ для програмування ESP32-CAM через Arduino IDE дозволяє оптимально використовувати апаратні ресурси пристрою.

3.3 Архітектура програмного забезпечення

Архітектура програмного забезпечення для системи визначення об'єктів для наземного дрону представлена у вигляді діаграми компонент, що дозволяє відобразити структуру системи на рівні компонентів, що дозволяє описати основні частини системи та їх взаємодію через інтерфейси.

Основні компоненти архітектури наступні:

- модуль ESP32-Cam для захоплення відео;
- оператор мобільної платформи (наземного дрону);
- сервер для дистанційного керування платформою (застосунок для обробки потоків відео та виконання алгоритмів розпізнавання).

Діаграма компонент розробленого програмного забезпечення наведена на рис. 3.4.

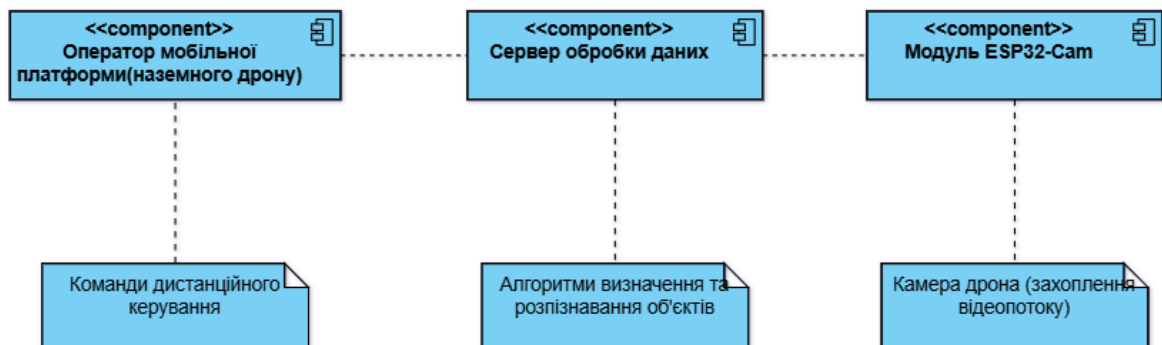


Рисунок 3.4 – Діаграма компонент

Опис функцій модуля ESP32-Cam:

- захоплення відеопотоку через камеру дрона;
- передача відеопотоку по Wi-Fi для подальшої обробки;
- можливість працювати в режимі реального часу, що важливо для задач, пов'язаних з обробкою зображень та прийняттям рішень на основі відео.

ESP32-Cam підключається до серверу обробки даних через Wi-Fi для передавання відео та отримання команд для управління дрону. Камера дрона передає відео на ESP32-Cam. Модуль ESP32-Cam передає відео на сервер обробки даних.

Опис функцій серверу обробки даних:

- обробка відеопотоку, отриманого від модуля ESP32-Cam;
- виконання алгоритмів комп'ютерного зору та машинного навчання для визначення об'єктів на відео;
- збір і зберігання результатів обробки для подальшого аналізу;
- підготовка та передача результатів оператору для відображення.

Сервер підключається до модуля ESP32-Cam через Wi-Fi та отримує дані для обробки. Після обробки він передає результати на відображення оператору.

Опис функцій оператору мобільної платформи (наземного дрону):

- керування рухом дрону за допомогою дистанційного керування через вебінтерфейс застосунку;
- отримання результатів обробки відеопотоку для аналізу в реальному часі.

Оператор взаємодіє з сервером обробки даних для отримання результатів виявлення об'єктів. Оператор також надсилає команди для дистанційного керування дроном через вебзастосунок.

Опис взаємодії між компонентами наступний:

- модуль ESP32-Cam отримує відеопотік від камери дрона і передає його на сервер обробки даних через Wi-Fi;
- сервер обробки даних виконує алгоритми розпізнавання об'єктів на основі отриманого відеопотоку. Після обробки відео сервер передає результати виявлення об'єктів (класи об'єктів) до оператора мобільної платформи для відображення;

– оператор мобільної платформи також може надсилати команди управління дроном через вебінтерфейс застосунку, які передаються до модуля ESP32-Cam для виконання відповідних дій на платформі.

Діаграма розгортання використовуються для відображення фізичної архітектури системи та її компонентів, а також для показу, як ці компоненти взаємодіють на фізичному рівні. Діаграма розгортання для системи визначення об'єктів для наземного дрону показує, як програмні компоненти взаємодіють з фізичними пристроями (сервери, камери, ESP32-Cam тощо).

Діаграма розгортання розробленого програмного забезпечення наведена на рис. 3.5.

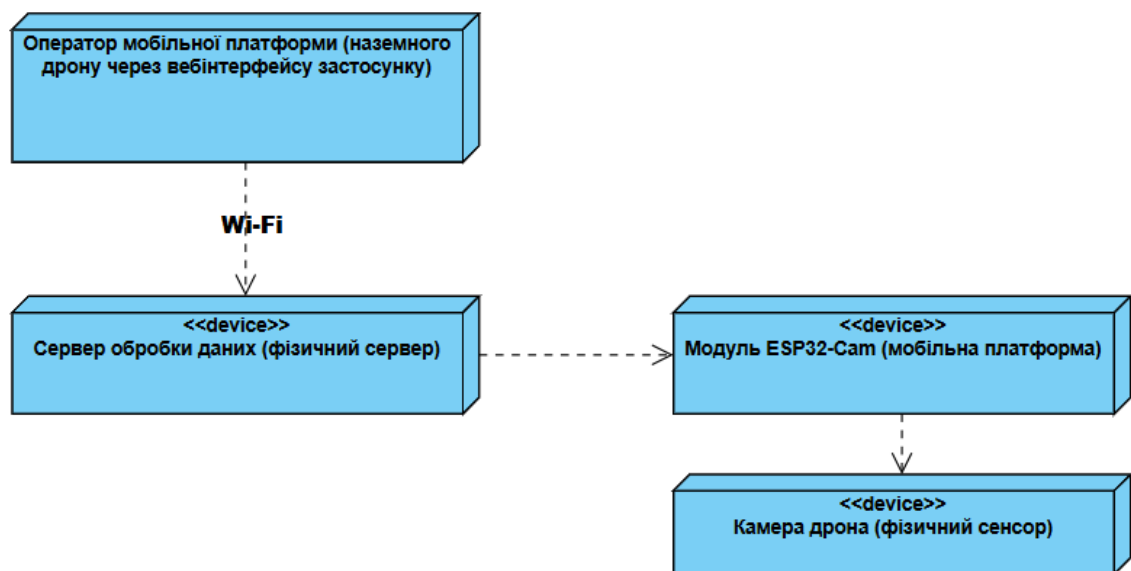


Рисунок 3.5 – Діаграма розгортання

Опис діаграми розгортання:

– оператор мобільної платформи: використовує вебінтерфейс застосунку для отримання результатів від системи і керування дроном; взаємодіє з сервером обробки даних через Wi-Fi;

– модуль ESP32-Cam – фізичний пристрій, що містить чіп ESP32, камеру та модуль Wi-Fi, що захоплює відео з камери дрона і передає його на

сервер обробки даних через Wi-Fi та отримує команди дистанційного керування від оператора через сервер;

– сервер обробки даних – фізичний сервер для обробки відео потоку, що виконує алгоритми розпізнавання об'єктів, обробляє дані та передає їх оператору; має підключення до інтернету для обміну даними з ESP32-Cam та оператором.

Діаграми класів представляють собою діаграми на яких буде показано класи системи та зв'язки між ними (рис. 3.6).

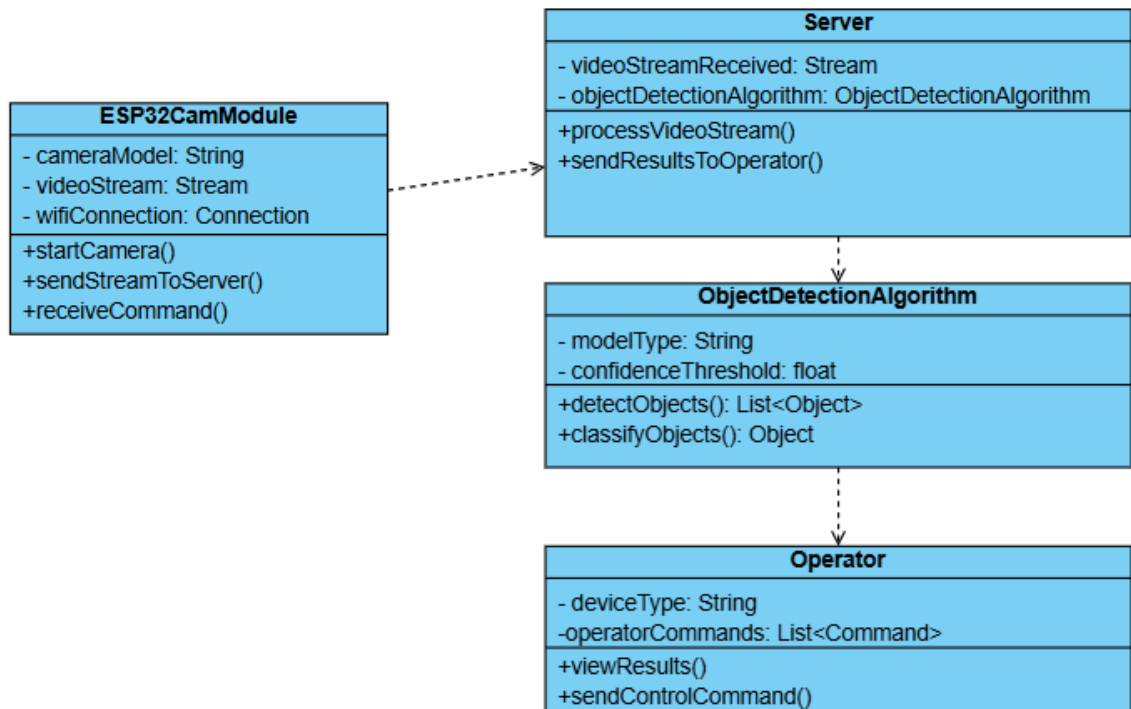


Рисунок 3.6 – Діаграма класів

Клас **ESP32CamModule** відповідає за захоплення відео з камери дрона та передачу його на сервер. Він також отримує команди від оператора для управління рухом дрону.

Клас **Server** обробляє відеопотік, отриманий від модуля ESP32-Cam. Він використовує алгоритм для розпізнавання об'єктів, а після обробки передає результати оператору.

Клас `ObjectDetectionAlgorithm` реалізує алгоритми визначення та розпізнавання об'єктів на відео. Може містити різні моделі для розпізнавання об'єктів, наприклад, YOLO або інші, а також методи для класифікації виявлених об'єктів.

Клас `Operator` представляє оператора, який взаємодіє з системою. Він отримує результати розпізнавання об'єктів і може надіслати команди для керування дроном.

3.4 Розробка архітектури модулів комп'ютерного зору та їх інтеграції

Основою системи є інтеграція алгоритму YOLO із бібліотекою `OpenCV`, яка дозволяє обробляти відеопотік, виявляти об'єкти та візуалізувати результати на екрані. Діаграма компонент інтеграції YOLO з `OpenCV` наведена на рисунку 3.7.

Система складається з наступних компонентів:

- модуль захоплення відеопотоку: відповідає за отримання відеопотоку з камери ESP32-CAM або іншого джерела; модуль реалізовано за допомогою функцій `OpenCV` (`cv2.VideoCapture`) для читання кадрів із камери;

- модуль попередньої обробки відео: підготовка кожного кадру до обробки: масштабування, нормалізація, перетворення у формат, який приймає модель YOLO та включає зміну розміру кадру та перетворення його до тензорного представлення;

- модуль визначення об'єктів у кадрі за допомогою натренованої моделі YOLO: виконує обчислення за допомогою попередньо натренованої моделі YOLO; вхідні дані (підготовлений кадр) передаються до моделі, яка повертає координати виявлених об'єктів, їх класи та рівень впевненості.;

- модуль постобробки результатів: фільтрує результати визначення об'єктів на основі рівня впевненості (>0.5) та формує прямокутники навколо

об'єктів та підписує їхні класи (реалізовано за допомогою функцій OpenCV `cv2.rectangle`, `cv2.putText`);

– модуль візуалізації: накладає результати визначення об'єктів на відеокадр: відображає кадри з результатами на екрані у реальному часі (реалізовано через функцію OpenCV `cv2.imshow()`).

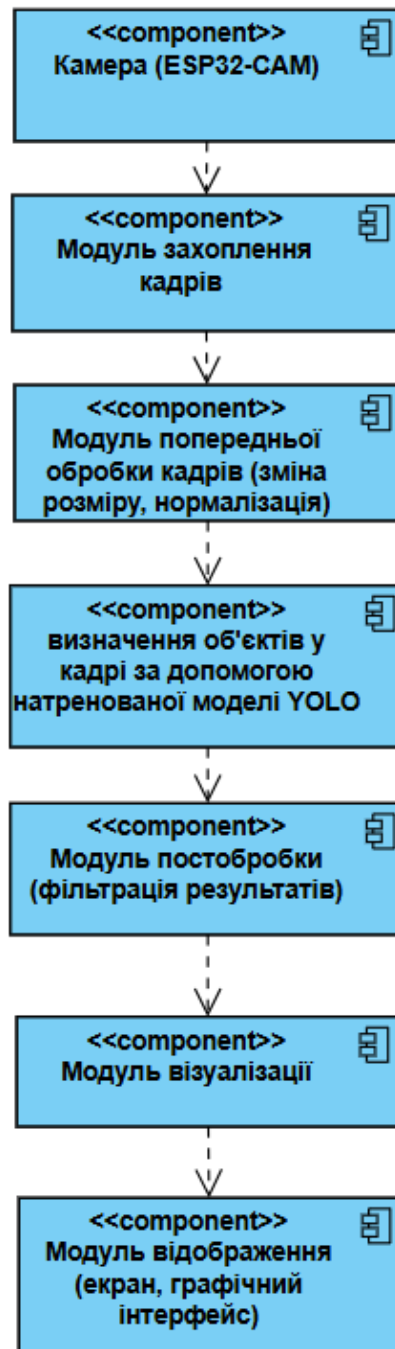


Рисунок 3.7 – Діаграма компонент інтеграції YOLO з OpenCV

3.5 Висновки за розділом 3

В даному розділі було проведено аналіз вимог до програмного забезпечення та розроблено діаграми варіантів використання (прецедентів) за допомогою мови UML на основі визначених функціональних вимог. Обґрунтовано вибір мови програмування Python для основної частини програмного забезпечення з використанням бібліотеки OpenCV. Для програмування модуля ESP32-Cam було вирішено використовувати C/C++ в середовищі Arduino IDE. Також спроектовано архітектуру програмного забезпечення системи визначення об'єктів для наземного дрону, що представлена у вигляді діаграми компонент. Було розроблено діаграму розгортання та діаграму класів системи, а також описано архітектуру інтеграції YOLO з OpenCV для обробки відео в реальному часі.

4 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

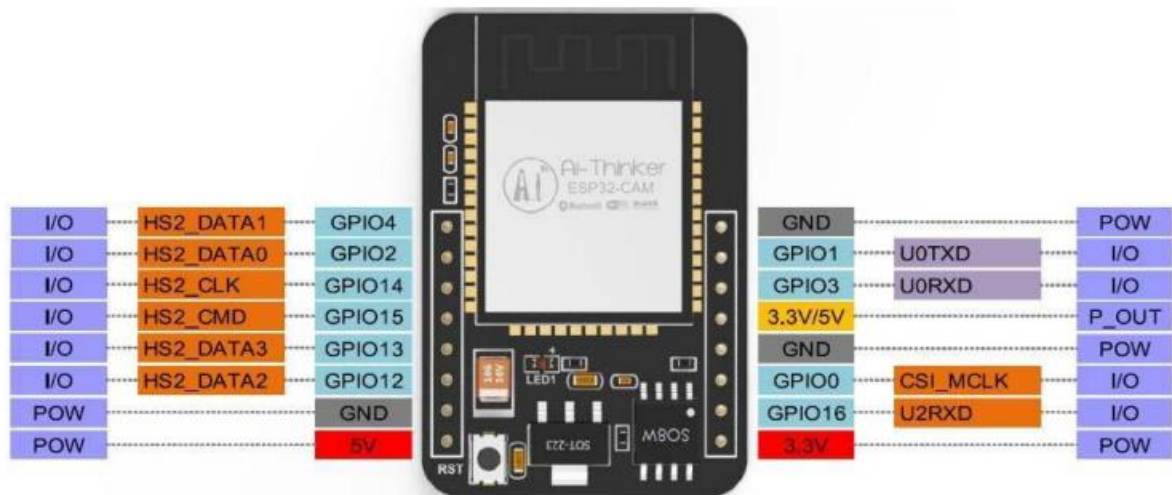
4.1 Програмування модуля ESP32-Cam

Модуль ESP32-Cam – модуль, що містить потужний мікроконтролер на базі чіпа ESP32, включає в себе камеру, бездротову мережу Wi-Fi та Bluetooth. Це один з найпопулярніших модулів для створення бездротових систем, включаючи дистанційне керування дронами. Можна визначити наступні переваги цього модуля – це його компактні розміри, низька вартість та висока функціональність.

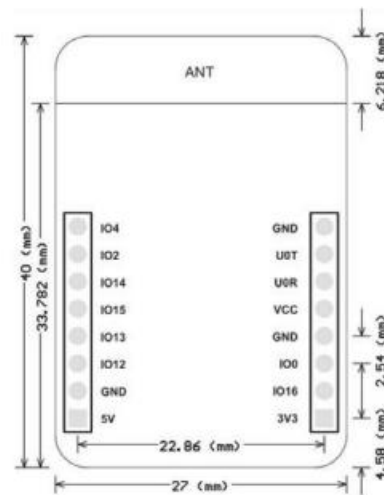
В роботі розроблено програму для модуля ESP32-Cam з метою реалізації функції дистанційного керування наземним дроном для моніторингу навколишнього середовища у режимі реального часу, що дозволить визначити об'єкти в радіусі спостереження.

Модуль ESP32-Cam дозволяє здійснювати передачу відео або зображень через Wi-Fi. Модуль оснащений камерою OV2640 з роздільною здатністю 2 МП, що дозволяє отримувати зображення високої якості, та вбудований Wi-Fi модуль для підключення до бездротових мереж та передачі даних на відстані.

Модуль ESP32-Cam містить чіп ESP32-S та GPIO [23] для підключення периферійних пристроїв, слот для картки microSD, який можна використовувати для зберігання зображень, зроблених за допомогою камери (рис. 4.1). Розпиновка ESP32-Cam (AI-Thinker module) має три контакти GND і два контакти живлення: 3,3В або 5В. GPIO1 і GPIO3 є послідовними контактами. Крім того, GPIO0 також відіграє важливу роль, оскільки він визначає, чи перебуває ESP32 у режимі флеш-пам'яті. Коли GPIO 0 підключено до GND, ESP32 блимає.



Schematic Diagram



Dimension Diagram

Рисунок 4.1 – Схематична діаграма модуля ESP32-Cam [46]

Виділимо основні характеристики модуля ESP32-Cam [47]:

- чіп ESP32 з двоядерним процесором;
- камера OV2640;
- вбудований Wi-Fi і Bluetooth;
- підтримка зовнішньої SD-карти;
- цифрові входи/виходи для підключення датчиків або інших периферійних пристроїв.

Для програмування модуля ESP32-Cam використовують середовище розробки Arduino IDE, що підтримує цей модуль завдяки бібліотекам та платам ESP32.

Для того, щоб почати програмувати ESP32-Cam, необхідно налаштувати Arduino IDE, для цього необхідно:

- завантажити та встановити Arduino IDE;
- додати підтримку плат ESP32 у Arduino IDE через менеджер плат;
- вибрати тип плати "ESP32 Wrover Module" або відповідну плату з серії ESP32.

Далі необхідно відкрити відповідний текст програми, для цього потрібно вибрати теку DIY_Surveillance_camera_with_ESP32-Camera (рис. 4.2).



Рисунок 4.2 – Відкриття відповідного тексту програми

Далі потрібно переконатися, що Wi-Fi, який потрібно підключити, має діапазон 2,4G, а не 5G. Потім, від'єднайте дроти 5 В та GND, підключені до плати розширення, інакше це вплине на запис тексту програми (рис. 4.3).

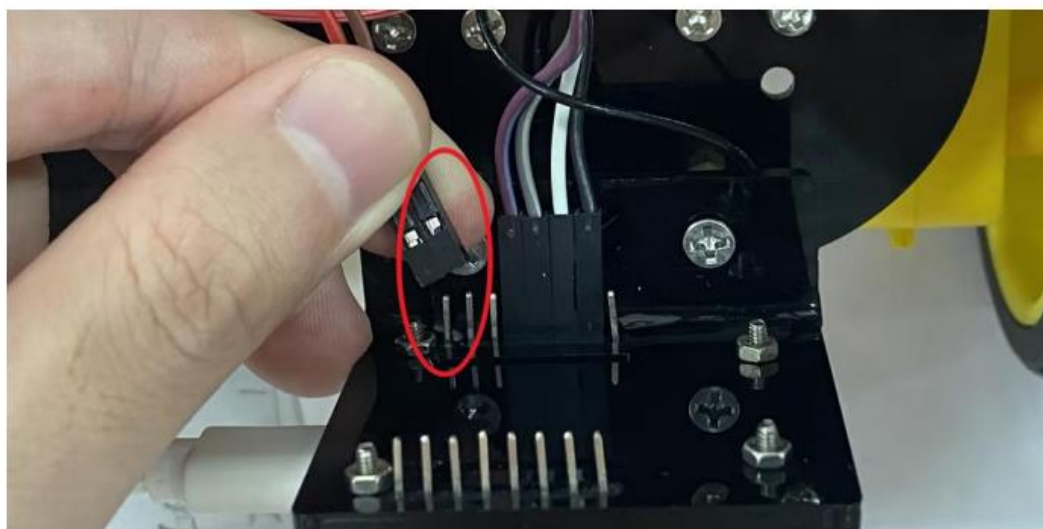
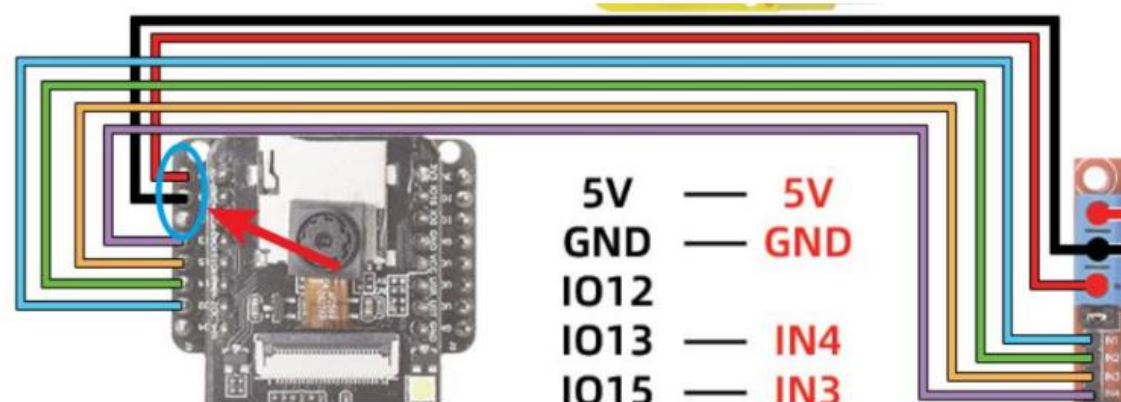


Рисунок 4.3 – Від'єднання дротів 5 В та GND

Тепер підключіть плату розробки ESP32-CAM до комп'ютера через порт Type-C плати розширення. Перейдіть до Інструменти => Плата => ESP32Arduino в Arduino IDE і виберіть модуль ESP32 Wrover (рис. 4.4).

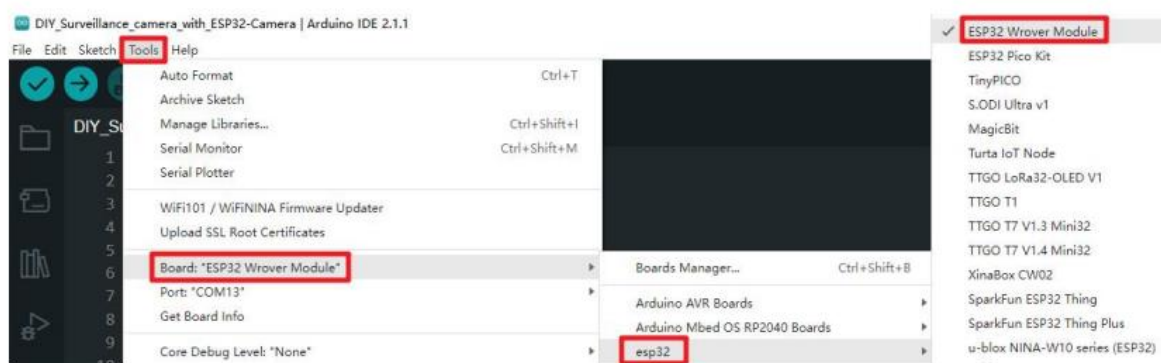


Рисунок 4.4 – Вибір модуля ESP32 Wrover

Далі перейдіть до Інструменти => Порт і виберіть COM-порт (рис. 4.5), до якого підключено ESP32 (вибраний номер COM може не бути однаковим для всіх, він повинен іншим).

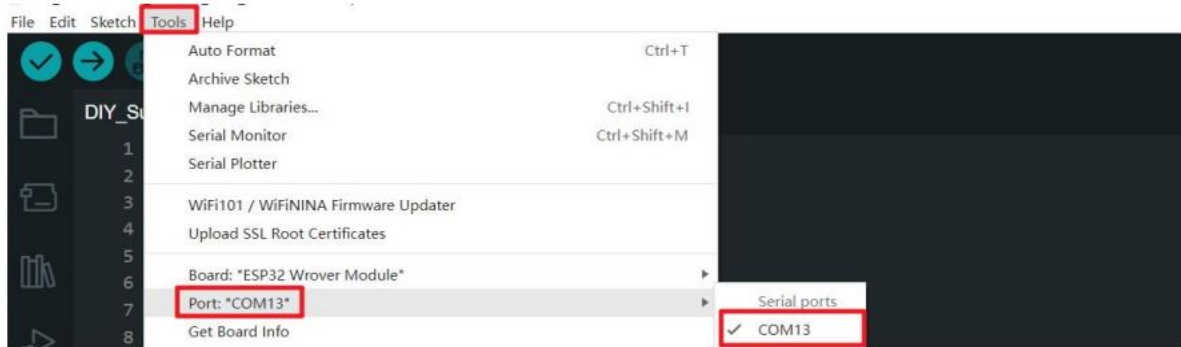


Рисунок 4.5 – Вибір COM-порту

Далі у Інструменти => Схема розділу вибрати «Величезна програма (3 МБ або OTA)» (рис. 4.6).

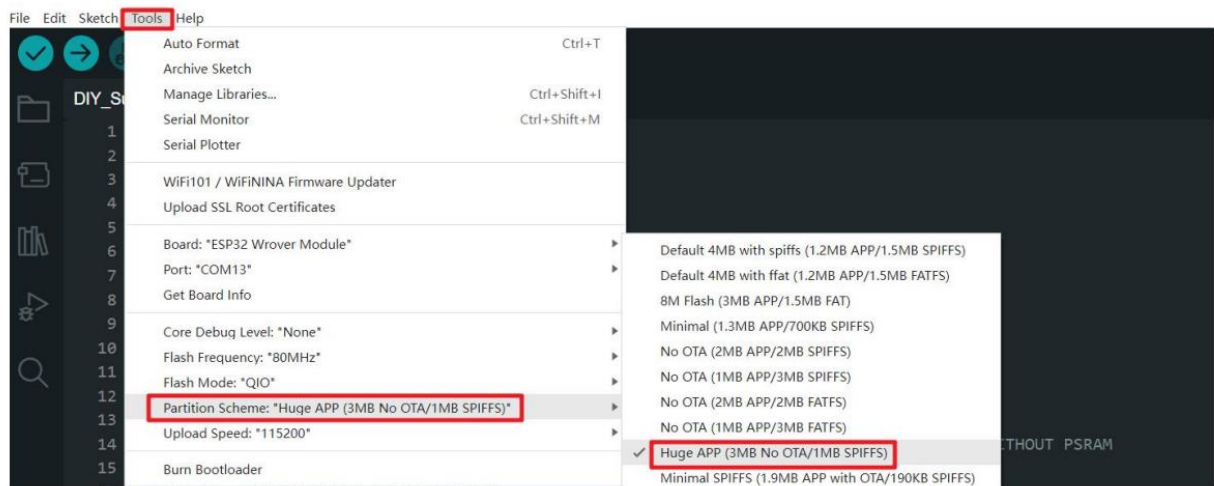


Рисунок 4.6 – Вибір програми

Потім натиснути кнопку «Завантажити» (рис. 4.7), щоб записати текст програми (якщо запис не вдасться, перевірте, чи правильний послідовний порт).

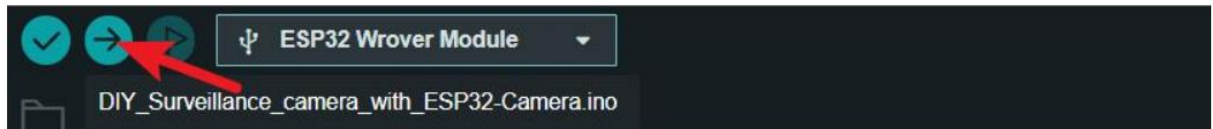


Рисунок 4.7 – Завантаження програми

Після успішного завантаження з'являється запит «Завантаження завершено» (рис. 4.8).

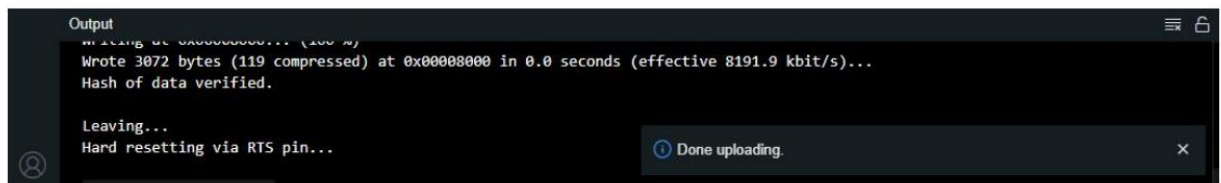


Рисунок 4.8 – Запит «Завантаження завершено»

Наступний етап це перегляд IP-адреси.

Після успішного завантаження коду ESP32-CAM перезавантажиться. Для цього необхідно відкрити монітор послідовного порту у верхньому правому куті на швидкості 115200 бод і натисніть кнопку скидання на платі розширення ESP32-CAM (рис. 4.9).



Рисунок 4.9 – Натиснення кнопки скидання

Коли ESP32-CAM успішно підключено до Wi-Fi, можна побачити IP адресу (рис. 4.10).

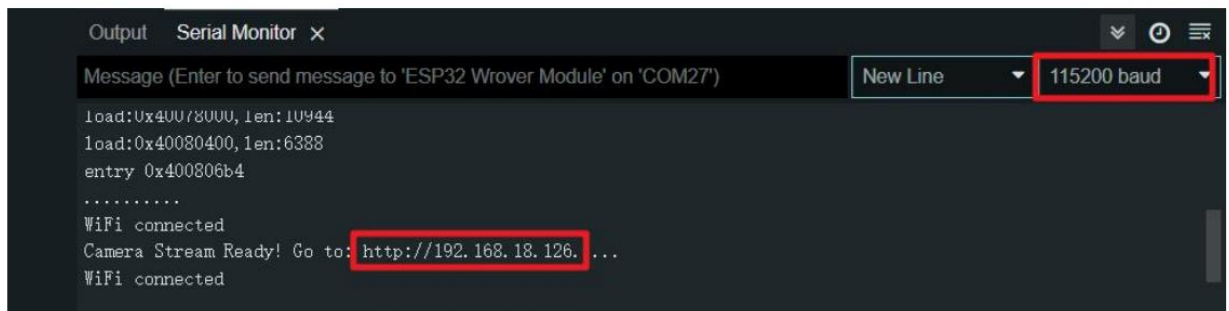


Рисунок 4.10 – Перегляд IP адреси

Далі виконати перегляд запису камери: для цього необхідно відкрити браузер за допомогою пристрою, підключеного до тієї ж локальної мережі WiFi, ввести IP-адресу камери в адресний рядок і перейти за посиланням. Далі можна побачити екран камери. Зверніть увагу, що це функція окремої камери, тому екран буде горизонтальним.

У випадку, якщо екран камери відсутній, то необхідно перевірити, чи правильно вказано ім'я та пароль WiFi; чи використовується мережа 2.4G (а не 5G) та чи достатній рівень сигналу; далі перезавантажити ESP32 Cam, щоб забезпечити стабільне живлення.

4.2 Реалізація дистанційного керування наземним дроном

Для дистанційного керування дроном використовується ESP32 для з'єднання з пультом керування або мобільним застосунком. Задача полягає в тому, щоб реалізувати двостороннє з'єднання між пультом керування та дроном для отримання даних (наприклад, відео або стану системи) і передачі команд.

Підключення до сервера для керування дроном. Для реалізації підключення дрон використовує серво-приводи для руху, які можна контролювати через PWM-сигнали, а для передачі команд можна використовувати WebSocket або TCP-з'єднання. Керування дроном через ESP32 може бути реалізоване на мобільному застосунку або через веб-

інтерфейс. Для реалізації цього підходу можна використовувати бібліотеки для ESP32, такі як WiFi.h для підключення до Wi-Fi, а також бібліотеки для роботи з серво-приводами, такі як Servo.h або ESP32Servo.h.

Для керування рухом наземного дрону можна використати такі алгоритми: визначення напрямку руху через кнопки на мобільному застосунку або через веб-інтерфейс (вперед, назад, вліво, вправо) або застосування PID-регулювання для забезпечення точного контролю за рухом.

Відображення відео в реальному часі можна реалізувати наступним чином: відео, що надходить з ESP32-Cam, можна транслювати на мобільний пристрій через HTTP або WebSocket, далі можна створити інтерфейс для перегляду цього відео у веб браузері.

4.3 Програмна реалізація алгоритму на основі моделі комп'ютерного зору YOLO для ESP32-Cam

Визначення та розпізнавання об'єктів на відеопотоці в режимі реального часу за допомогою моделі комп'ютерного зору YOLO, реалізовано з використанням модуля ESP32-Cam, з подальшим відображенням результатів на екрані.

Алгоритм виявляє об'єкти на зображенні, класифікує їх за категоріями та відображає результати у вигляді прямокутників навколо знайдених об'єктів з відповідними підписами.

Нижче наведено вербальний опис алгоритму.

Крок 1. Імпорт та підключення бібліотек до відеопотоку.

```
import cv2
```

```
import numpy as np
```

cv2 – бібліотека OpenCV використовується для роботи із зображеннями та відео.

numpy – потрібна для роботи з числовими масивами, що важливо для обробки даних нейронною мережею.

```
url = "http://192.168.0.102:81/stream"
```

Далі задається URL для підключення до відеопотоку ESP32-Cam.

Крок 2. Завантаження YOLO моделі.

Завантажується заздалегідь навчена модель YOLO, що включає ваги yolov3.weights та конфігураційний файл yolov3.cfg. Якщо завантаження відбувається успішно, виводиться повідомлення, інакше помилка виводиться на екран.

```
net = cv2.dnn.readNet(r"C:\Users\Dovgal
Dima\Desktop\esp32cam_video_stream_on_web_server\CameraWebServer\yolov
3.weights",
r"C:\Users\Dovgal
Dima\Desktop\esp32cam_video_stream_on_web_server\CameraWebServer\yolov
3.cfg")
```

Крок 3. Отримання імен шарів.

Отримуємо імена всіх шарів мережі та визначаємо вихідні шари, які будуть використовуватись для обробки результатів.

```
layer_names = net.getLayerNames()
output_layers = [layer_names[i] - 1] for i in
net.getUnconnectedOutLayers()]
```

Крок 4. Завантаження класів об'єктів.

Завантажується список класів об'єктів із coco.names. Кожен клас відповідає типу об'єктів, які можуть розпізнати модель (наприклад, людина, автомобіль тощо).

```
with open(r"C:\Users\Dovgal
Dima\Desktop\esp32cam_video_stream_on_web_server\CameraWebServer\coco.n
ames", "r") as f:
classes = [line.strip() for line in f.readlines()]
```

Крок 5. Відкриття відеопотоку.

Встановлюється з'єднання з потоком відео за заданим URL. Перевіряється успішність відкриття потоку.

```
cap = cv2.VideoCapture(url)
```

Крок 6. Читання та обробка кадрів.

Поки потік відкритий, кадри читаються із потоку. Якщо кадру не вдалося отримати, цикл завершиться.

```
while cap.isOpened():
    ret, frame = cap.read()
```

Крок 7. Передобробка кадру

Кадр перетворюється на формат, що підходить для входу нейронної мережі. Масштабується до 416x416 пікселів та нормалізується.

```
blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True,
crop=False)
```

Крок 8. Прогнозування об'єктів.

Вхідні дані подаються до мережі, яка повертає прогнозування кожного кадру.

```
net.setInput(blob)
outs = net.forward(output_layers)
```

Крок 9. Обробка прогнозування.

Обробляються вихідні дані та обчислюються координати та розміри виявлених об'єктів.

```
for out in outs:
    for detection in out:
```

Крок 10. Фільтрування та відображення результатів.

Використовується Non-Maximum Suppression для видалення зайвих прямокутників.

```
indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
for i in indexes:
    x, y, w, h = boxes[i]
    label = str(classes[class_ids[i]])
    cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
```

```
cv2.putText(frame, f"{label} {int(confidence * 100)}%", (x, y - 10), font,
1, color, 2)
```

На кожному кадрі малюються прямокутники навколо об'єктів та додаються підписи з іменами класів та впевненістю передбачення.

Крок 11. Виведення кадрів на екран.

Оброблений кадр виводиться на екран. Завершити цикл можна, натиснувши 'q'.

```
cv2.imshow("ESP32-CAM Object Detection", frame)
```

Крок 12. Звільнення обчислювальних ресурсів.

Після завершення роботи звільняються ресурси, закривається потік та вікна відображення.

```
cap.release()
```

```
cv2.destroyAllWindows()
```

Схему алгоритму наведено на рис. 4.11.

В результаті розроблено алгоритм, що складається з таких основних кроків: завантаження моделі, обробка відеопотоку та відображення результатів. На основі цього алгоритму реалізовано систему розпізнавання об'єктів у режимі реального часу на відеопотоці з використанням моделі YOLO.

Текст програмної реалізації алгоритму розпізнавання об'єктів в відеопотоці ESP32Cam наведено у додатку А.

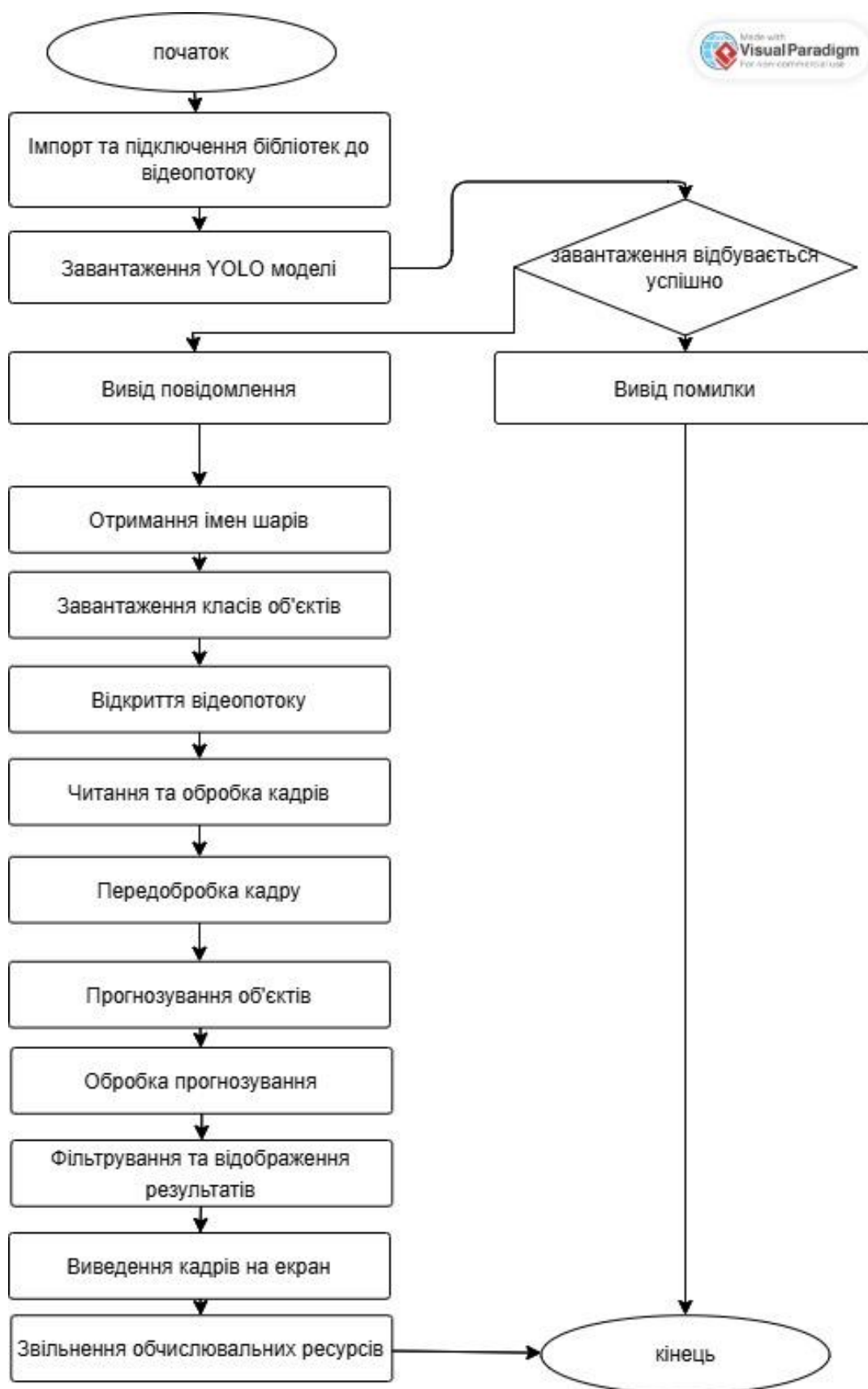


Рисунок 4.11 – Схема розробленого алгоритму

4.4 Реалізація вебзастосунку для відображення відеопотоку

Було розроблено вебзастосунок з використанням фреймворку Django, який відображає відеопотік з камери мікроконтролеру ESP32-CAM на вебсторінці.

Структура вебзастосунку `object_detection` наведена на рисунку 4.12.

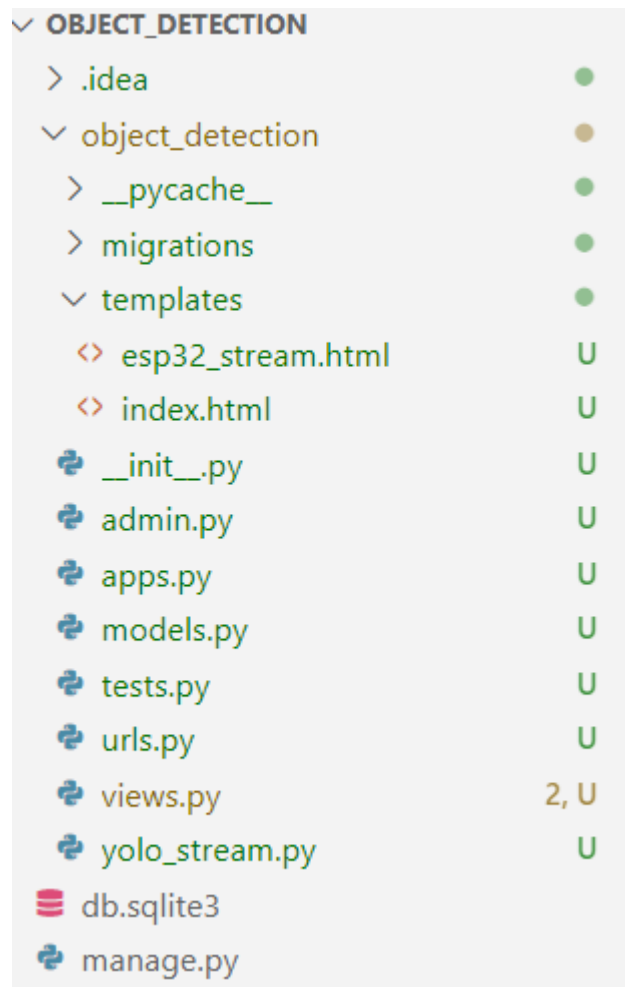


Рисунок 4.12 – Структура вебзастосунку `object_detection`

Проект складається з наступних розроблених компонентів:

- `yolo_stream.py` – алгоритм розпізнавання об’єктів в відеопотоці ESP32Cam;

- `view.py` – визначає логіку відображення відеопотоку з камери або представлення для завантаження HTML сторінки з потоком;

- esp32_stream.html – шаблон HTML, який відображає відеопотік з камери;
- index.html – головна сторінка програми;
- urls.py – маршрути вебзастосунку;
- manage.py – дозволяє виконати запуск вебзастосунку.

У проєкті передбачається використання камери ESP32-CAM, яка передає відеопотік через IP-адресу. У програмі Django необхідно вказати IP-адресу камери, наприклад, <http://192.168.0.105:81/stream>. Програма дозволяє відобразити реальний відеопотік з камери мікроконтролера на вебсторінці. Проєкт демонструє базову роботу з Django, налаштування програми, шаблони та маршрути, а також використання зовнішніх пристроїв (камер) для відображення даних у реальному часі.

4.5 Висновки за розділом 4

В даному розділі було розроблено програму для модуля ESP32-Cam, яка забезпечує дистанційне керування наземним дроном і моніторинг навколишнього середовища в реальному часі. Система дозволяє передавати відеопотік через Wi-Fi, забезпечуючи високу якість зображення завдяки використанню камери OV2640.

Реалізовано підключення між ESP32 та пультом керування або мобільним застосунком для управління рухом дрону. Використано сервоприводи для переміщення та бібліотеки, такі як `WiFi.h` і `Servo.h`. Відображення відеопотоку здійснюється через HTTP або WebSocket.

Розроблено алгоритм, що складається з таких основних кроків: завантаження моделі, обробка відеопотоку та відображення результатів. На основі цього алгоритму реалізовано систему розпізнавання об'єктів у режимі реального часу на відеопотоці з використанням моделі YOLO.

Розроблено вебзастосунок з використанням фреймворку Django, який відображає відеопотік з камери мікроконтролера ESP32-CAM на вебсторінці.

5 ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

Розділ присвячений тестуванню та експериментальному дослідженню розробленого програмного забезпечення для системи визначення об'єктів для мобільних платформ на базі ESP32-CAM.

5.1 Тестування системи

Тестування розробленого програмного забезпечення для системи визначення об'єктів для мобільних платформ на базі ESP32-CAM проводилося з метою перевірки функціональності системи, стабільності роботи, достовірності визначення та розпізнавання об'єктів та оцінки ступеню використання обчислювальних ресурсів.

Було проведено функціональне тестування програмного забезпечення, тестування для обчислення показника достовірності визначення та розпізнавання об'єктів різної складності, тестування на різних наборах даних, тестування для перевірки стабільності результатів.

Функціональне тестування програмного забезпечення включає перевірку роботи функціональних можливостей програми та спрямоване на перевірку відповідності основних функцій системи заявленим вимогам.

Під час тестування було здійснено наступних функцій системи:

- захоплення відеопотоку за допомогою модуля ESP32-CAM;
- передача відеопотоку на сервер для обробки;
- виявлення об'єктів із використанням попередньо натренованої моделі YOLO;
- візуалізація результатів у вигляді прямокутників навколо знайдених об'єктів із відповідними підписами.

У ході тестування перевірялися: коректність роботи всіх етапів; стабільність виконання кожної функції під час тривалого навантаження;

відповідність вихідних даних очікуваним результатам. Результати показали, що всі функції виконуються коректно, система успішно захоплює, передає та обробляє відеопотік, а також забезпечує точне відображення результатів.

В процесі тестування на різних наборах даних проводилась перевірка:

- для високоякісних зображень із стабільним освітленням;
- та для кадрів, отриманих у реальних умовах (низьке освітлення, фоновий шум).

У стабільних умовах програмне забезпечення показує достатньо високий показник достовірності визначення та розпізнавання (94%), а у складних умовах (погане освітлення, шум) цей показник знижується (до 58%).

Тестування повторюваності результатів проводилося для серії тестів із однаковими вхідними даними, щоб перевірити стабільність результатів. Програмне забезпечення демонструє стабільність із незначним відхиленням ($<1\%$) у вихідних даних.

Також проводилося тестування в умовах слабого сигналу Wi-Fi:

- стабільний Wi-Fi (швидкість передачі ≥ 50 Мбіт/с);
- слабкий Wi-Fi (швидкість передачі ≤ 5 Мбіт/с).

За стабільного Wi-Fi розроблене програмне забезпечення працювало без збоїв, тоді як за слабого сигналу спостерігалися затримки у передаванні кадрів (до 300 мс) та періодичні втрати зображення.

Результати проведених тестувань програмного забезпечення для системи визначення об'єктів для мобільних платформ на базі ESP32-CAM наведені у табл. 5.1.

Таблиця 5.1 – Результати тестування програмного забезпечення

Критерій	Опис	Результати
Функціональне тестування	Виконання основних функцій системи	Усі функції виконуються коректно
Тестування на різних наборах даних	Перевірка в стабільних та складних умовах	Достатньо високий показник достовірності визначення та розпізнавання
Повторюваність результатів	Серія тестів із однаковими вхідними даними	Відхилення результатів менш ніж 1%
Працездатність за умови слабого Wi-Fi	Оцінка здатності програмного забезпечення працювати при низькій швидкості передачі даних	Затримки до 300 мс, часткова втрата кадрів

Результати тестування програмного забезпечення демонструють відповідність функціональних можливостей системи заданим вимогам. З таблиці 5.1 можна виділити, що обмеження обчислювальної потужності ESP32-CAM і залежність від стабільного з'єднання Wi-Fi впливають на продуктивність системи в складних умовах.

5.2 Експериментальне дослідження системи

Дослідження оцінки ступеню використання обчислювальних ресурсів проводилося за наступними параметрами:

- час обробки кадру: під час тестування вимірювався середній час обробки одного кадру для двох роздільних здатностей відео потоку (320×240 та 640×480);

- обсяг використаної пам'яті: проведено моніторинг використання оперативної пам'яті та флеш-пам'яті ESP32-CAM;

- ступінь завантаження CPU ESP32-CAM: для оцінки продуктивності проводився моніторинг завантаження процесора для двох роздільних здатностей відео потоку та визначався показник середнього завантаження процесора (320×240 та 640×480);

- енергоспоживання: вимірювання енергоспоживання за допомогою мультиметра під час роботи системи, в результаті експерименту було виявлено, що середнє споживання енергії системи становило 0.65 Вт (при живленні від 5 В).

Для оцінки показника достовірності розпізнавання об'єктів заданого класу розробленою системою було проведено на наборах даних із різними об'єктами:

- для простих об'єктів (люди);
- для складних об'єктів (маленькі предмети, об'єкти з низькою контрастністю).

Для простих об'єктів достовірність розпізнавання склала 94 %, тоді як для складних достовірність знижувалася до 77%, що вказує на обмеження моделі при обробці складних типів даних.

Результати експериментального дослідження програмного забезпечення для системи визначення об'єктів для мобільних платформ на базі ESP32-CAM наведені у табл. 5.2.

Таблиця 5.2 – Результати експериментального дослідження програмного забезпечення

Критерій	Опис	Результати
Час обробки кадру	Вимірювання часу обробки для різних роздільних здатностей	320×240: 107 мс 640×480: 179 мс
Обсяг використаної пам'яті	Моніторинг використання оперативної та флеш-пам'яті ESP32-CAM	Оперативна: 165 КБ, Флеш: 3.1 МБ
Завантаження CPU ESP32-CAM	Оцінка завантаження CPU під час обробки кадрів	320×240: 47.5 % 640×480: 48.54 %
Енергоспоживання	Вимірювання енергоспоживання під час роботи системи	Середнє споживання: 0.65 Вт
Достовірність розпізнавання простих об'єктів	оцінка ідентифікації об'єктів в умовах реальних даних та порівняння результатів із еталонними даними	94 %
Достовірність розпізнавання складних об'єктів	оцінка ідентифікації об'єктів в умовах реальних даних та порівняння результатів із еталонними даними	77 %

Також проводилися додаткові експерименти впливу умов освітлення на оцінку показника достовірності розпізнавання простих та складних

об'єктів, які показали, що зі зниженням рівня освітленості показник достовірності розпізнавання об'єктів знижується.

Результати експериментального дослідження показують, що система має достатню продуктивність за звичайних умов. Обмеження ресурсів ESP32-CAM частково можна вирішити за рахунок оптимізації алгоритмів, але для роботи в складніших умовах необхідне подальше вдосконалення та розширення функціональних можливостей системи.

5.3 Напрямки подальшого розвитку системи

Розроблене програмне забезпечення для системи визначення об'єктів для мобільних платформ на базі ESP32-CAM в результаті виконання роботи вирішує поставлені задачі та показує достатньо добрі результати.

Проведений аналіз результатів та отримані оцінки використання обчислювальних ресурсів та оцінки показника достовірності розпізнавання різних об'єктів у реальних умовах показують, що можливе подальше вдосконалення та розширення функціональних можливостей системи в умовах обмежених обчислювальних ресурсів:

- здійснити оптимізацію алгоритмів та моделей машинного навчання за рахунок квантизації та спрощення архітектури [48],[49];
- використати легкі моделі: Tiny YOLO або MobileNet, які адаптовані до платформ IoT [48],[50];
- виконати інтеграцію функцій відстеження об'єктів у реальному часі з можливістю передбачення траєкторії руху;
- здійснити реалізацію підтримки голосового або текстового сповіщення користувача про ідентифіковані об'єкти;
- реалізувати передачу відеопотоку або результатів розпізнавання у хмарні сервіси для централізованого зберігання, подальшої обробки та виконання обчислень, наприклад за допомогою глибоких моделей навчання з використанням Amazon Web Services;

- додати додаткові сенсори (наприклад, LiDAR, ультразвукових сенсорів) для підвищення точності та стійкості системи в умовах низької освітленості або складного середовища;
- впровадити енергозберігаючі режими роботи ESP32-CAM, які активуються лише при необхідності обробки відеопотоку.

5.4 Висновки за розділом 5

В даному розділі було описано тестування та проведено експериментальне дослідження розробленого програмного забезпечення для системи визначення об'єктів для мобільних платформ на базі ESP32-CAM.

Результати тестування та експериментального дослідження розробленого програмного забезпечення для системи визначення об'єктів для мобільних платформ на базі ESP32-CAM підтверджують, що система здатна виконувати основні функції з високою достовірністю та стабільністю за нормальних умов роботи. Всі основні функції системи, такі як захоплення відеопотоку, передача даних на сервер, виявлення об'єктів та візуалізація результатів, працюють коректно та відповідають заявленим вимогам. Тестування показало, що система здатна розпізнавати об'єкти в умовах стабільного освітлення на достатньому рівні, але при складних умовах, таких як низьке освітлення або наявність шуму, ступінь достовірності розпізнавання знижується.

Також було проведено тестування для оцінки стабільності результатів, яке підтвердило, що система демонструє мінімальні відхилення в результатах при повторному тестуванні (менше ніж 1%). Це свідчить про високий рівень стабільності та надійності програмного забезпечення в умовах однакових вхідних даних.

Тестування працездатності системи в умовах слабкого Wi-Fi показало, що при зниженій швидкості передачі даних (менше 5 Мбіт/с) система стикається із затримками в передачі кадрів і періодичними втратами

зображень, що знижує ефективність розпізнавання об'єктів. Це вказує на залежність роботи системи від стабільного інтернет-з'єднання.

Експериментальне дослідження, яке включало оцінку використання обчислювальних ресурсів, показало, що ESP32-CAM має обмеження в плані обчислювальної потужності. Система виявляється здатною обробляти відеопотік при роздільній здатності 320×240 , однак для роботи з вищими роздільними здатностями (640×480) спостерігається значне збільшення завантаження процесора та часу обробки кадрів. Це свідчить про необхідність оптимізації алгоритмів для більш ефективного використання обмежених ресурсів.

Результати експериментів з енергоспоживання показали, що середнє споживання енергії системи становить 0.65 Вт, що є достатньо низьким для роботи в автономному режимі. Однак для тривалих робочих сесій важливим є зниження енергоспоживання через додаткові заходи оптимізації.

Достовірність розпізнавання об'єктів у реальних умовах залежить від типу об'єкта: для простих об'єктів достовірність є високою (94%), тоді як для складних об'єктів із низькою контрастністю результат знижується (58%). Це свідчить про необхідність подальшої оптимізації моделей для більш ефективної обробки складних типів даних.

Проведене експериментальне дослідження програмного забезпечення підтверджує, що розроблена система має достатній рівень продуктивності для використання в умовах стабільних параметрів, однак для роботи в складніших умовах (погане освітлення, низька швидкість передачі даних) необхідне подальше вдосконалення.

Також було запропоновано подальше вдосконалення та розширення функціональних можливостей системи в умовах обмежених обчислювальних ресурсів та спрямовані на оптимізацію обчислювальних ресурсів, покращення достовірності розпізнавання в складних умовах і забезпечення стабільної роботи при низькому рівні сигналу Wi-Fi.

ВИСНОВКИ

В результаті виконання дипломної кваліфікаційної роботи було розроблено програмне забезпечення прототипу системи визначення об'єктів, що дозволяє ідентифікувати об'єкти у реальному часі з використанням алгоритмів комп'ютерного зору для мобільних платформ із обмеженими ресурсами.

Проведено аналіз мобільних платформ із обмеженими ресурсами, методів обробки зображень і технологій машинного навчання. Розглянуто платформи ESP32-CAM, Raspberry Pi 4 і NVIDIA Jetson Nano. На основі порівняльного аналізу обрано ESP32-Cam як оптимальну платформу для розробки системи визначення об'єктів із обмеженим використанням апаратних ресурсів, враховуючи її низьку вартість, компактність і енергоефективність.

Проведено аналіз вимог до програмного забезпечення та розроблено діаграми варіантів використання (прецедентів) за допомогою мови UML на основі визначених функціональних вимог. Обґрунтовано вибір мови програмування Python для основної частини програмного забезпечення з використанням бібліотеки OpenCV. Для програмування модуля ESP32-Cam було вирішено використовувати C/C++ в середовищі Arduino IDE. Також спроектовано архітектуру програмного забезпечення системи визначення об'єктів для наземного дрону, що представлена у вигляді діаграми компонент.

Розроблено діаграму розгортання та діаграму класів системи, а також описано архітектуру інтеграції YOLO з OpenCV для обробки відео в реальному часі. Розроблено програму для модуля ESP32-Cam, яка забезпечує дистанційне керування наземним дроном і моніторинг навколишнього середовища в реальному часі. Система дозволяє передавати відеопотік через Wi-Fi, забезпечуючи високу якість зображення завдяки використанню камери OV2640.

Розроблено алгоритм, що складається з таких основних кроків: завантаження моделі, обробка відеопотоку та відображення результатів. На основі цього алгоритму реалізовано систему розпізнавання об'єктів у режимі реального часу на відеопотоці з використанням моделі YOLO. Розроблено вебзастосунок з використанням фреймворку Django, який відображає відеопотік з камери мікроконтролеру ESP32-CAM на вебсторінці.

Було проведено тестування та експериментальне дослідження розробленого програмного забезпечення для системи визначення об'єктів для мобільних платформ на базі ESP32-CAM.

Проведене експериментальне дослідження програмного забезпечення підтверджує, що розроблена система має достатній рівень продуктивності для використання в умовах стабільних параметрів, однак для роботи в складніших умовах (погане освітлення, низька швидкість передачі даних) необхідне подальше вдосконалення.

Наукова новизна роботи полягає в тому, що розроблено алгоритм розпізнавання об'єктів у режимі реального часу на відеопотоці з високою достовірністю розпізнавання за рахунок використання моделі YOLO.

Практична цінність результатів роботи полягає в тому, що розроблене програмне забезпечення може бути використано як система відстеження та розпізнавання об'єктів для моніторингу навколишнього середовища в реальному часі для рухомих мобільних платформ з обмеженими ресурсами.

Майбутні дослідження будуть спрямовані на подальше вдосконалення та розширення функціональних можливостей системи в умовах обмежених обчислювальних ресурсів, на оптимізацію використання обчислювальних ресурсів, покращення достовірності розпізнавання в складних умовах і забезпечення стабільної роботи при низькому рівні сигналу Wi-Fi.

Результати роботи було представлено на міжнародній науково-практичній конференції «Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій» (Запоріжжя, 10-12 грудня 2024 року) [51].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Ahmed A. Software Architectures for Robotics Systems: A Systematic Mapping Study / A. Ahmed, M.Ali Babar // Journal of Systems and Software. – 2016. – Vol. 122. – P. 16–39.
2. Software Architectures for Mobile Robots [Electronic resource]. – Access mode : <https://ar5iv.org/html/2206.03233>.
3. FreeRTOS [Electronic resource]. – Access mode : <https://www.freertos.org/>.
4. Zephyr 4.0 [Electronic resource]. – Access mode : <https://zephyrproject.org/>.
5. The RTEMS Project [Electronic resource]. – Access mode : <https://www.rtems.org/>.
6. ROS – Robot Operating System The Robot Operating [Electronic resource]. – Access mode : <https://www.ros.org/>.
7. Теорія керування для інформатиків : підручник / Ю. В. Крак, А. В. Шатирко. – К. : ВПЦ "Київський університет", 2015. – 175 с.
8. What Is SLAM? [Electronic resource]. – Access mode : <https://ch.mathworks.com/discovery/slam.html>.
9. Алгоритм пошуку A^* [Електрон. ресурс]. – Режим доступу : https://uk.wikipedia.org/wiki/Алгоритм_пошуку_A*.
10. Алгоритм Дейкстри [Електрон. ресурс]. – Режим доступу : https://uk.wikipedia.org/wiki/Алгоритм_Дейкстри.
11. Фільтр Калмана [Електрон. ресурс]. – Режим доступу : https://uk.wikipedia.org/wiki/Фільтр_Калмана.
12. MQTT: The Standard for IoT Messaging [Electronic resource]. – Access mode : <https://mqtt.org/>.
13. Simulating your robots with Webots [Electronic resource]. – Access mode : <https://cyberbotics.com/>.

14. Gazebo (simulator) [Electronic resource]. – Access mode : [https://en.wikipedia.org/wiki/Gazebo_\(simulator\)](https://en.wikipedia.org/wiki/Gazebo_(simulator)).
15. OpenCV [Electronic resource]. – Access mode : <https://opencv.org/>.
16. Соколов, К. А. Використання методів обчислення оптичного потоку для вирішення задач детекції та супроводження малогабаритних цілей / Соколов К. А. // Радіотехнічні проблеми, сигнали, апарати та системи : матеріали X Міжнародної науково-технічної конференції [Київ, 09-11 листопада 2021 р.]. – Київ, 2021. – С. 8-10.
17. IBVS for Drone operations [Electronic resource]. – Access mode : <https://github.com/satwikio/Ground-vehicle-following-drone>.
18. Computer Vision using ROS2 [Electronic resource]. – Access mode : https://github.com/jmguerreroh/ros2_computer_vision.
19. ROS2-Drones-with-AI-and-Computer-Vision [Electronic resource]. – Access mode : <https://github.com/noshluk2/ROS2-Drones-with-AI-and-Computer-Vision>.
20. Реконструкція карти SLAM за допомогою lidar [Електрон. ресурс]. – Режим доступу : <https://ua.gzcyndar.net/info/slam-map-reconstruction-using-lidar-47851081.html>.
21. Drones with ROS2 [Electronic resource]. – Access mode : https://github.com/Robotisim/drones_ROS2.
22. PX4-ROS2-Gazebo Drone Simulation Template [Electronic resource]. – Access mode : <https://github.com/SathanBERNARD/PX4-ROS2-Gazebo-Drone-Simulation-Template/>.
23. ESP32-CAM із камерою OV2640 [Електрон. ресурс]. – Режим доступу : <https://www.mini-tech.com.ua/ua/esp32-cam>.
24. ESP32 Camera Driver [Electronic resource]. – Access mode : <https://github.com/espressif/esp32-camera>.
25. Raspberry Pi [Electronic resource]. – Access mode : <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>.

26. Nano RP2040 Connect [Electronic resource]. – Access mode : <https://docs.arduino.cc/hardware/nano-rp2040-connect/>.
27. Jetson Nano [Electronic resource]. – Access mode : <https://developer.nvidia.com/embedded/jetson-nano>.
28. Getting started with Image Processing Using OpenCV [Electronic resource]. – Access mode: <https://www.analyticsvidhya.com/blog/2023/03/getting-started-with-image-processing-using-opencv/>.
29. Фільтрація зображень засобами OpenCV [Електрон. ресурс]. – Режим доступу : <https://ai-tern.in.ua/Filtration.html>.
30. Cascade Classifier [Electronic resource]. – Access mode : https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html.
31. Using Haar Cascade for Object Detection [Electronic resource]. – Access mode : <https://machinelearningmastery.com/using-haar-cascade-for-object-detection/>.
32. Histogram of Oriented Gradients and Object Detection [Electronic resource]. – Access mode : <https://pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/>.
33. Real-time Object Detection with YOLO and Webcam: Enhancing Your Computer Vision Skills [Electronic resource]. – Access mode : <https://dipankarmedh1.medium.com/real-time-object-detection-with-yolo-and-webcam-enhancing-your-computer-vision-skills-861b97c78993>.
34. SSD object detection: Single Shot MultiBox Detector for real-time processing [Electronic resource]. – Access mode : <https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06>.
35. The Fundamental Guide to Faster R-CNN [Electronic resource]. – Access mode : <https://viso.ai/deep-learning/faster-r-cnn-2/>.
36. AIoT Solution [Electronic resource]. – Access mode : <https://www.espressif.com/en/products/devkits/esp-eye/overview>.

37. Espressif IoT Development Framework [Electronic resource]. – Access mode : <https://github.com/espressif/esp-idf>.
38. TensorFlow Lite – Real-Time Computer Vision on Edge Devices [Electronic resource]. – Access mode : <https://viso.ai/edge-ai/tensorflow-lite/>.
39. OpenCV: Open Source Computer Vision Library [Electronic resource]. – Access mode : <https://github.com/opencv/opencv>.
40. Розділ 2. Основи UML [Електрон. ресурс]. – Режим доступу : <https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-basics.html>.
41. Visual Paradigm [Electronic resource]. – Access mode : <https://www.visual-paradigm.com/>.
42. Python [Electronic resource]. – Access mode : <https://www.python.org/>.
43. The Role of Python in Robotics and Automation: Creating Intelligent Machines [Electronic resource]. – Access mode : <https://moldstud.com/articles/p-the-role-of-python-in-robotics-and-automation-creating-intelligent-machines>.
44. Getting Started With ESP32-CAM [Electronic resource]. – Access mode : <https://lastminuteengineers.com/getting-started-with-esp32-cam/>.
45. Using ESP32-CAM with Arduino IDE Tutorial [Electronic resource]. – Access mode : <https://www.olimex.com/Products/IoT/ESP32/ESP32-CAM/resources/Using-ESP32-CAM-with-Arduino-IDE.pdf>.
46. ESP32-CAM модуль wi-fi з камерою 2mp [Електрон. ресурс]. – Режим доступу : <https://komcat.net.ua/manual/esp32cam.html>.
47. ESP32-CAM and Other Cool Projects on RNT [Electronic resource]. – Access mode : https://www.espressif.com/en/news/ESP32_CAM.
48. Advancements in TinyML: Applications, Limitations, and Impact on IoT Devices [Electronic resource]. – Access mode : <https://www.mdpi.com/2079-9292/13/17/3562>.
49. Tiny-YOLO with 3-bit quantization [Electronic resource]. – Access mode : <https://github.com/Shunli-Wang/Tiny-YOLO-LSQ>.

50. MobileNet – Efficient Deep Learning for Mobile Vision [Electronic resource]. – Access mode : <https://viso.ai/deep-learning/mobilenet-efficient-deep-learning-for-mobile-vision/>.

51. Довгаль Д. Ф. Дослідження та програмна реалізація системи визначення об'єктів для наземного дрону / Д.Ф.Довгаль, Н.О. Миронова, А.В.Пархоменко // Міжнародна науково-практична конференція «Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій», 10-12 грудня 2024 (подано до друку).

52. "Object Detection" - Django проєкт по перехопленні відео потоку та розпізнаванню об'єктів [Electronic resource]. – Access mode : https://github.com/Dmitriy-1986/object_detection/tree/main.

ДОДАТОК А
Текст програми

A.1 Текст файлу yolo_stream.py алгоритму розпізнавання об'єктів в відеопотоці ESP32Cam

```

import cv2
import numpy as np
from django.http import StreamingHttpResponse

# Шляхи к YOLO
WEIGHTS_PATH = r"C:\Users\Dovgal
Dima\Desktop\esp32cam_video_stream_on_web_server\CameraWebServer\yolov
3.weights"
CFG_PATH = r"C:\Users\Dovgal
Dima\Desktop\esp32cam_video_stream_on_web_server\CameraWebServer\yolov
3.cfg"
NAMES_PATH = r"C:\Users\Dovgal
Dima\Desktop\esp32cam_video_stream_on_web_server\CameraWebServer\coco.n
ames"

# Завантаження YOLO моделі
net = cv2.dnn.readNet(WEIGHTS_PATH, CFG_PATH)
with open(NAMES_PATH, "r") as f:
    classes = [line.strip() for line in f.readlines()]
layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]

# Відеопотік ESP32
VIDEO_URL = "http://192.168.88.233:81/stream"

# Генератор кадрів для відеопотоку
def yolo_stream():
    cap = cv2.VideoCapture(VIDEO_URL)
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        # Передобробка кадру
        blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True,
crop=False)
        net.setInput(blob)
        outs = net.forward(output_layers)

        # Обробка виходів

```

```

class_ids, confidences, boxes = [], [], []
height, width, _ = frame.shape

for out in outs:
    for detection in out:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.5:
            center_x, center_y, w, h = (int(detection[0] * width), int(detection[1] *
height),
                                     int(detection[2] * width), int(detection[3] * height))
            x, y = int(center_x - w / 2), int(center_y - h / 2)
            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

for i in indexes:
    x, y, w, h = boxes[i]
    label = str(classes[class_ids[i]])
    confidence = confidences[i]
    color = (0, 255, 0)
    cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
    cv2.putText(frame, f'{label} {int(confidence * 100)}%', (x, y - 10),
cv2.FONT_HERSHEY_PLAIN, 1, color, 2)

# Кодуємо кадр у формат JPEG для надсилання в потік
_, buffer = cv2.imencode('.jpg', frame)
frame = buffer.tobytes()

yield (b'--frame\r\n'
      b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

cap.release()

```

A.2 Текст файлу views.py

```

from django.shortcuts import render
from django.http import StreamingHttpResponse
from .yolo_stream import yolo_stream

# Генератор для передачі відеопотоку

```

```
def video_feed(request):
    return StreamingHttpResponse(yolo_stream(), content_type='multipart/x-mixed-replace; boundary=frame')
```

Представлення для завантаження HTML сторінки з потоком

```
def esp32_stream(request):
    context = {'camera_url': '/object_detection/video_feed/'}
    return render(request, 'esp32_stream.html', context)
```

A.3 Текст файлу esp32_stream.html

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
        content="width=device-width, user-scalable=no, initial-scale=1.0,
maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>ESP32-CAM Stream</title>
    <style>
        body {
            background: rgba(0, 0, 0, 0.91);
            color: darkgrey;
            text-align: center;
        }
        h1, h2 {
            margin: 10px 0;
        }
        img {
            border: 3px solid darkgrey;
            margin-top: 20px;
        }
    </style>
</head>
<body>
    <h1>ESP32-CAM Live Stream</h1>
    <h2>Dovgal D. Project Django</h2>
    <div>
        
    </div>
</body>
```

</html>

A.4 Текст файлу urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('video_feed/', views.video_feed, name='video_feed'), # Подача
    відеопотока
    path('esp32_stream/', views.esp32_stream, name='esp32_stream'), # HTML
    сторінка
]
```

A.5 Текст програми test.cpp експериментального дослідження розробленого програмного забезпечення

```
#include "esp_camera.h"
#include <WiFi.h>
#include <Arduino.h>

#define CAMERA_MODEL_AI_THINKER // Has PSRAM
#include "camera_pins.h"

// =====
// Enter your WiFi credentials
// =====
const char *ssid = "Anonymus-2000";
const char *password = "8Fxh8FwQNpW8L3rFb";

unsigned long frameTime = 0; // Время обработки кадра
unsigned long lastTime = 0;
unsigned long currentTime = 0;

void startCameraServer();
void setupLedFlash(int pin);

// Функція для виведення статистики мікроконтролера
void printSystemStats() {
    // Час роботи
    currentTime = millis();
    Serial.print("=====\\n");
    Serial.print("System Uptime: ");
```

```

Serial.print(currentTime / 1000); // Час роботи в секундах
Serial.println(" seconds");

// Пам'ять
Serial.print("Free Heap Memory: ");
Serial.print(ESP.getFreeHeap());
Serial.print(" bytes | Total Heap Memory: ");
Serial.print(ESP.getHeapSize());
Serial.println(" bytes");

// Статистика завантаження CPU (заснована на часі виконання)
unsigned long freeMem = ESP.getFreeHeap();
float cpuLoad = (100 - (freeMem / (float) ESP.getHeapSize()) * 100);
Serial.print("CPU Load: ");
Serial.print(cpuLoad);
Serial.println("%");

// Wi-Fi
Serial.print("WiFi Status: ");
if (WiFi.status() == WL_CONNECTED) {
    Serial.print("Connected, IP: ");
    Serial.print(WiFi.localIP());
} else {
    Serial.print("Not connected");
}
Serial.println();

// Статистика часу обробки кадру
Serial.print("Frame Processing Time: ");
Serial.print(frameTime); // Час обробки кадру в мілісекундах
Serial.println(" ms");

// Стан камери
sensor_t *s = esp_camera_sensor_get();
Serial.print("Camera Sensor PID: ");
Serial.println(s->id.PID);
Serial.print("Current Frame Resolution: ");
Serial.println(s->status.framesize);

Serial.println("=====");
}

void setup() {
    Serial.begin(115200);
    Serial.setDebugOutput(true);

```

```

Serial.println();

camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sccb_sda = SIOD_GPIO_NUM;
config.pin_sccb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.frame_size = FRAMESIZE_UXGA;
config.pixel_format = PIXFORMAT_JPEG; // for streaming
config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
config.fb_location = CAMERA_FB_IN_PSRAM;
config.jpeg_quality = 12;
config.fb_count = 1;

if (psramFound()) {
    config.jpeg_quality = 10;
    config.fb_count = 2;
    config.grab_mode = CAMERA_GRAB_LATEST;
} else {
    config.frame_size = FRAMESIZE_SVGA;
    config.fb_location = CAMERA_FB_IN_DRAM;
}

// camera init
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
}

```

```

sensor_t *s = esp_camera_sensor_get();
// initial sensors are flipped vertically and colors are a bit saturated
if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1);    // flip it back
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, -2); // lower the saturation
}
// drop down frame size for higher initial frame rate
if (config.pixel_format == PIXFORMAT_JPEG) {
    s->set_framesize(s, FRAMESIZE_QVGA);
}

WiFi.begin(ssid, password);
WiFi.setSleep(false);

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");

startCameraServer();
Serial.print("Camera Ready! Use 'http://");
Serial.print(WiFi.localIP());
Serial.println("' to connect");
}

void loop() {
    // Monitor frame processing time
    unsigned long startFrameTime = millis();
    camera_fb_t *fb = esp_camera_fb_get(); // Захват кадру
    frameTime = millis() - startFrameTime; // Зміна часу обробки кадру

    // Після захоплення кадру - виводимо статистику
    if (fb) {
        esp_camera_fb_return(fb); // Звільняємо буфер кадру
    }
    // Друкуємо системну статистику
    printSystemStats();
    delay(1000); // Інтервал в 1 секунду для виведення статистики
}

```

ДОДАТОК Б
Слайди презентації

Національний університет
«Запорізька політехніка»

Магістерська робота на тему:

Дослідження та програмна реалізація системи визначення об'єктів для наземного дрону

Виконав:
ст.гр. КНТ-113м
Керівник:
доц., к.т.н.

Дмитро ДОВГАЛЬ
Анжеліка ПАРХОМЕНКО

Рисунок Б.1 – Слайд 1

Актуальність роботи



Наземний дрон-мінер



Наземний дрон розвідник-розмінер



Рисунок Б.2 – Слайд 2

Мета та задачі роботи

Мета роботи - розробити програмне забезпечення прототипу системи визначення об'єктів, що дозволяє ідентифікувати об'єкти у реальному часі з використанням алгоритмів комп'ютерного зору для мобільних платформ із обмеженими ресурсами

Для досягнення поставленої мети в дипломній кваліфікаційній роботі потрібно вирішити наступні **задачі**:

- дослідити мобільні платформи з обмеженими ресурсами;
- дослідити підходи обробки зображень для системи визначення об'єктів для мобільних платформ з обмеженими ресурсами;
- дослідити можливість інтеграції технологій машинного навчання для мобільних платформ з обмеженими ресурсами;
- обґрунтувати вибір технологій і інструментів розробки програмного забезпечення;
- розробити архітектуру програмного забезпечення для системи визначення об'єктів;
- розробити архітектуру модулів комп'ютерного зору та виконати їх інтеграцію в систему визначення об'єктів;
- виконати реалізацію дистанційного керування наземним дроном;
- виконати програмну реалізацію алгоритму на основі моделі комп'ютерного зору для обраної мобільної платформи;
- розробити прототип системи визначення об'єктів;
- провести тестування та експериментальне дослідження розробленого програмного забезпечення для системи визначення об'єктів.

3

Рисунок Б.3 – Слайд 3

Аналіз вимог до програмного забезпечення

Функціональні вимоги:

- захоплення відео потоку: отримання зображень у режимі реального часу з камери (ESP32-Cam);
- передача відеопотоку через Wi-Fi;
- виявлення та ідентифікацію об'єктів на зображенні у реальному часі;
- обробка та аналіз даних: класифікація виявлених об'єктів відповідно до заданих класів (люди, транспорт, перешкоди тощо);
- зберігання та передача даних про виявлені об'єкти для подальшої обробки;
- відображення результатів розпізнавання;
- дистанційне керування мобільною платформою (наземним дроном);
- можливість додавання нових алгоритмів або класів об'єктів у майбутньому;
- підтримка роботи з іншими камерами або апаратними платформами.

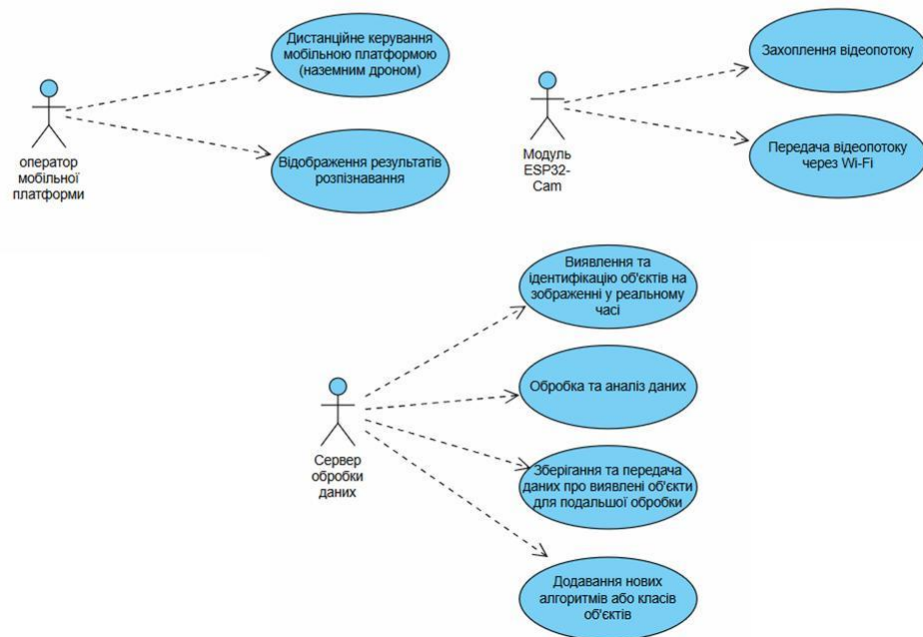
Нефункціональні вимоги:

- обмежене використання пам'яті (до 520 КБ оперативної пам'яті ESP32-CAM);
- робота в умовах низької пропускної здатності Wi-Fi.

4

Рисунок Б.4 – Слайд 4

Діаграми варіантів використання



5

Рисунок Б.5 – Слайд 5

Порівняльна характеристика застосування комп'ютерного зору в ROS2, Webots і Gazebo

Критерій	ROS2	Webots	Gazebo
Методи розпізнавання	виявлення об'єктів через інтеграцію з OpenCV; алгоритми глибокого навчання для класифікації	моделювання камер для імітації розпізнавання об'єктів та застосування сегментації зображень	інтеграція алгоритмів нейронних мереж для виявлення об'єктів та використання симульованих LiDAR та камер
Методи відстеження	фільтр Кальмана для прогнозування; оптичний потік для аналізу руху	моделювання відстеження траєкторії об'єктів у віртуальному середовищі	реалістична симуляція динаміки об'єктів для тестування алгоритмів
Методи 3D реконструкцій	інтеграція SLAM для побудови карти навколишнього середовища	стереозір і моделювання тривимірної геометрії об'єктів	SLAM з використанням симульованих даних сенсорів
Сценарії тестування	виявлення та класифікація у реальних умовах	тестування алгоритмів у контрольованих віртуальних середовищах	тестування в реалістичних середовищах із моделюванням фізики
Інтеграція з інструментами	TensorFlow, PyTorch, та OpenCV	взаємодія з ROS2 для експорту алгоритмів у реальні системи	підтримка модулів ROS2 для реального впровадження алгоритмів

6

Рисунок Б.6 – Слайд 6

Порівняльна характеристика обраних платформ з обмеженими ресурсами за функціональними можливостями

Платформа	Процесор	Пам'ять	Підключення	Наявність камери	Ціна
ES2P3-Cam	Tensilica LX6	520 КБ ОЗУ	Wi-Fi, Bluetooth	OV2640 (1600×1200)	\$10-15
Raspberry Pi 4	ARM Cortex-A72	2/4/8 ГБ LPDDR4	Wi-Fi, Bluetooth, GPIO	Raspberry Pi Camera (до 12 Мп)	\$35-70
Arduino Nano RP2040	Dual Cortex-M0+	264 КБ SRAM	USB, GPIO	Через додаткові модулі	\$10-20
NVIDIA Jetson Nano	Cortex-A57, 128 CUDA	4 ГБ LPDDR4	Ethernet, GPIO	MIPI CSI-2	\$99

7

Рисунок Б.7 – Слайд 7

Порівняльна характеристика обраних платформ з обмеженими ресурсами для реалізації комп'ютерного зору

Платформа	Підтримка відеопотоку	Споживання енергії	Підтримка TensorFlow, PyTorch, OpenCV	Області застосування
ES2P3-Cam	160×120 - 1600×1200, до 30 FPS	0.6-0.9 Вт	TensorFlow Lite Micro, OpenCV (обмежена)	IoT-системи, спостереження, портативні системи моніторингу
Raspberry Pi 4	640×480 – 4K, до 60 FPS	5-8 Вт	TensorFlow, PyTorch, OpenCV	Робототехніка, домашня автоматизація, мультимедійні системи
NVIDIA Jetson Nano	1080p - 4K, до 30 FPS	10-15 Вт	TensorFlow, PyTorch, OpenCV	Комп'ютерний зір, глибоке навчання, складні системи робототехніки

8

Рисунок Б.8 – Слайд 8

Порівняння алгоритмів виявлення об'єктів на зображеннях

Алгоритм	Переваги	Недоліки	Підтримка ESP32-CAM	Призначення
Каскади Хаара	швидкий, можна виділити низькі вимоги до ресурсів	низька точність для складних об'єктів	обмежена	просте виявлення об'єктів, наприклад обличчя людини
HOG	висока точність для виявлення людей	потребує більше ресурсів	обмежена	виявлення людей
YOLO	висока швидкість і точність	погіршення точності для малих об'єктів	підтримується	виявлення об'єктів у реальному часі
SSD	швидкість обробки на середніх зображеннях	менш точний, ніж YOLO, на великих зображеннях	підтримується	виявлення кількох об'єктів одночасно
Faster R-CNN	дуже точний	потребує великих ресурсів	обмежена	виявлення складних сцен

9

Рисунок Б.9 – Слайд 9

Порівняння технологій та фреймворків для машинного навчання для мобільних платформ з обмеженими ресурсами

Технологія (фреймворк)	Переваги	Недоліки	Підтримка ESP32-CAM	Призначення
ESP-WHO (ESP-IDF)	спеціалізоване для ESP32, підтримка розпізнавання облич	обмежений набір функцій для складних задач; важко інтегрувати сторонні моделі машинного навчання	обмежена підтримка для складних задач	розпізнавання облич, базові задачі комп'ютерного зору
TensorFlow Lite	оптимізована для пристроїв з низьким енергоспоживанням, використання компактних моделей	обмежена пам'ять ESP32 не дозволяє завантажити навіть мінімальні моделі TensorFlow, а також низька продуктивність при обробці відео	проблеми з інтеграцією на ESP32-CAM	підтримка моделей машинного навчання на мікроконтролерах
OpenCV	універсальний фреймворк для комп'ютерного зору, багатий набір функцій для обробки зображень	потрібне потужніше обладнання для обробки відео в реальному часі, що потребує зовнішнього сервера	підтримка для базових задач	обробка зображень та відео, виявлення об'єктів у реальному часі

10

Рисунок Б.10 – Слайд 10

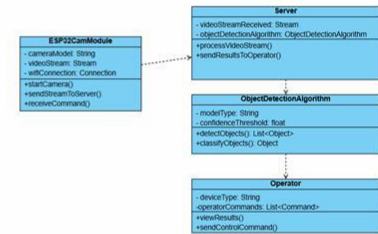
Діаграми компонент, розгортання та класів



Діаграма компонент програмного забезпечення



Діаграма розгортання програмного забезпечення



Діаграма класів програмного забезпечення

11

Рисунок Б.11 – Слайд 11

Архітектура модулів та схема розробленого алгоритму комп'ютерного зору



Архітектура модулів комп'ютерного зору

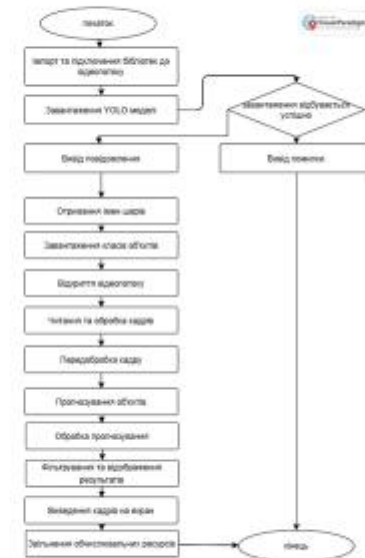


Схема розробленого алгоритму

12

Рисунок Б.12 – Слайд 12

Результати тестування програмного забезпечення

Критерій	Опис	Результати
Функціональне тестування	Виконання основних функцій системи	Усі функції виконуються коректно
Тестування на різних наборах даних	Перевірка в стабільних та складних умовах	Достатньо високий показник достовірності визначення та розпізнавання
Повторюваність результатів	Серія тестів із однаковими вхідними даними	Відхилення результатів менш ніж 1%
Працездатність за умови слабого Wi-Fi	Оцінка здатності програмного забезпечення працювати при низькій швидкості передачі даних	Затримки до 300 мс, часткова втрата кадрів

13

Рисунок Б.13 – Слайд 13

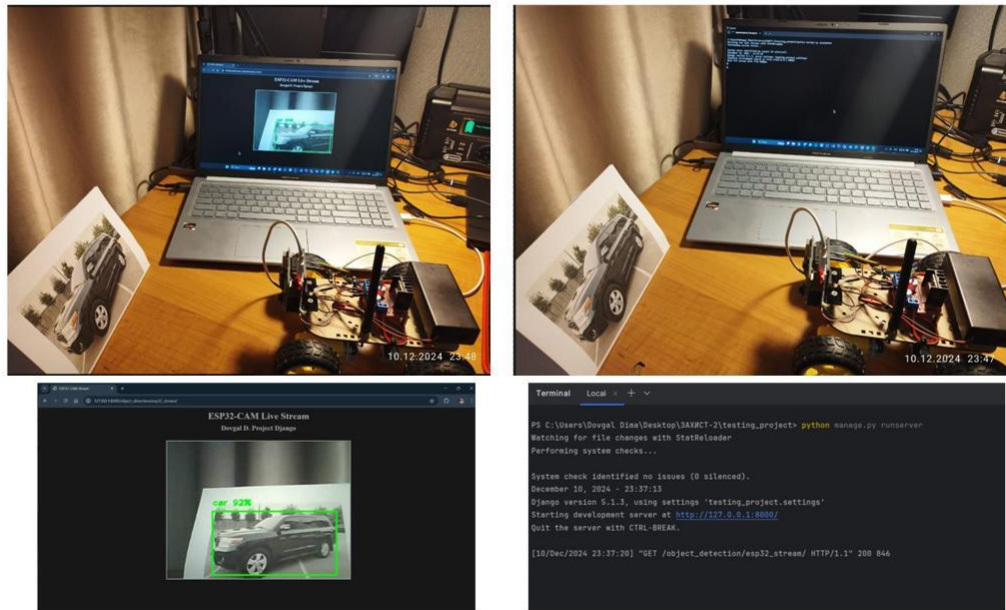
Результати експериментального дослідження програмного забезпечення

Критерій	Опис	Результати
Час обробки кадру	Вимірювання часу обробки для різних роздільних здатностей	320×240: 107 мс 640×480: 179 мс
Обсяг використаної пам'яті	Моніторинг використання оперативної та флеш-пам'яті ESP32-CAM	Оперативна: 165 КБ, Флеш: 3.1 МБ
Завантаження CPU ESP32-CAM	Оцінка завантаження CPU під час обробки кадрів	320×240: 47.5 % 640×480: 48.54 %
Енергоспоживання	Вимірювання енергоспоживання під час роботи системи	Середнє споживання: 0.65 Вт
Достовірність розпізнавання простих об'єктів	оцінка ідентифікації об'єктів в умовах реальних даних та порівняння результатів із еталонними даними	94%
Достовірність розпізнавання складних об'єктів	оцінка ідентифікації об'єктів в умовах реальних даних та порівняння результатів із еталонними даними	77%

14

Рисунок Б.14 – Слайд 14

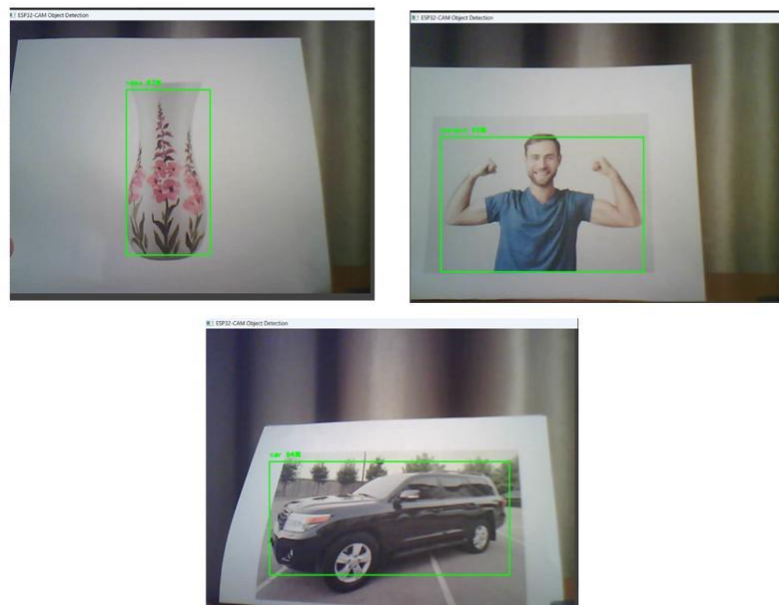
Прототип системи



15

Рисунок Б.15 – Слайд 15

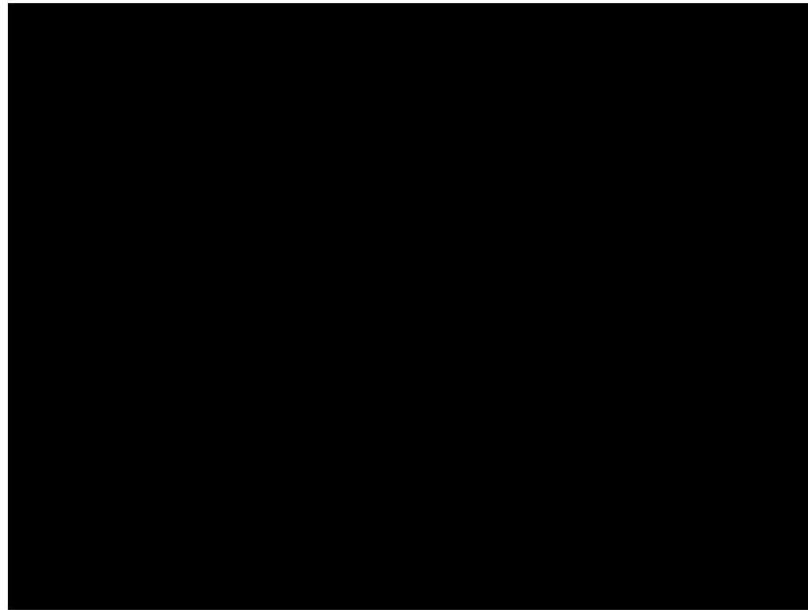
Результати визначення та розпізнавання об'єктів



16

Рисунок Б.16 – Слайд 16

Відеодемонстрація роботи програмного забезпечення



17

Рисунок Б.17 – Слайд 17

Висновки

- В результаті виконання роботи було розроблено програмне забезпечення прототипу системи визначення об'єктів, що дозволяє ідентифікувати об'єкти у реальному часі з використанням алгоритмів комп'ютерного зору для мобільних платформ із обмеженими ресурсами.
- Наукова новизна роботи полягає в тому, що розроблено алгоритм розпізнавання об'єктів у режимі реального часу на відеопотоці з високою достовірністю розпізнавання за рахунок використання моделі YOLO.
- Практична цінність результатів роботи полягає в тому, що розроблене програмне забезпечення може бути використано як система відстеження та розпізнавання об'єктів для моніторингу навколишнього середовища в реальному часі для рухомих мобільних платформ з обмеженими ресурсами.
- Майбутні дослідження будуть спрямовані на подальше вдосконалення та розширення функціональних можливостей системи в умовах обмежених обчислювальних ресурсів, на оптимізацію використання обчислювальних ресурсів, покращення достовірності розпізнавання в складних умовах і забезпечення стабільної роботи при низькому рівні сигналу Wi-Fi.
- Результати роботи було представлено на міжнародній науково-практичній конференції «Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій» (Запоріжжя, 10-12 грудня 2024 року)

18

Рисунок Б.18 – Слайд 18