Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Інтелектуальний програмно-апаратний модуль супроводу слабозорих на зупинках громадського транспорту
(тема)

Виконав:

здобувач 4 року навчання,

групи КІУКІ-21-6

Олександр НЕЧІТАЙЛО
(власне ім'я, прізвище)

Спеціальність

123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма

Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник: ас. Антон ГАВРАШЕНКО
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Нечітайлу Олександру Віталійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Інтелектуальний програмно-апаратний модуль супроводу слабозорих на зупинках громадського транспорту

затверджена наказом по університету від “ 26 ” травня 2025 р. № 424 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 17 червня 2025 р.

3. Вхідні дані до роботи 1) відеопотік з USB-камери Logitech (720p, 30 fps); 2) модель глибокого навчання YOLOv8n, натренована на зображеннях транспорту; 3) розмітка об'єктів у форматі YOLO (файл data.yaml); 4) оптичне розпізнавання тексту за допомогою Tesseract OCR; 5) периферійні пристрої: навушники, звукова карта, power bank; 6) ОС Raspberry Pi OS Lite, автоматичний запуск скрипту; 7) умови експлуатації: зовнішнє середовище, денне/вечірнє освітлення.

4. Перелік питань, що потрібно опрацювати у роботі
1) аналіз проблем орієнтації слабозорих у міському середовищі;
2) огляд існуючих систем розпізнавання транспорту та OCR;
3) вибір апаратної платформи та її компонентів;
4) навчання моделі YOLOv8n для детекції транспорту;
5) розробка алгоритмів попередньої обробки для OCR;
6) інтеграція модулів детекції, розпізнавання й озвучення;
7) тестування системи у реальних та імітованих умовах.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій Слайд-презентація – 34 слайди

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Вивчення проблемної області та постановка	27.05.25-30.05.25	
	задачі		
2	Розробка технічного завдання	31.05.25-02.06.25	
3	Формування та навчання моделі	03.06.25-05.06.25	
4	Розробка програмної частини	06.06.25-09.06.25	
5	Тестування системи	10.06.25	
6	Оформлення матеріалів кваліфікаційної роботи	11.06.25	
7	Подання кваліфікаційної роботи керівникові та	12.06.25-13.06.25	
	її попередній захист		
8	Подання кваліфікаційної роботи на рецензу-	14.06.25-16.06.25	
	вання		

Дата видачі завдання " 26 " травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Антон ГАВРАШЕНКО
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 110 с., 27 рис., 3 табл., 2 дод., 40 джерел.

RASPBERRY PI, ГЛИБИННЕ НАВЧАННЯ, YOLOV8, TESSERACT, OPENCV, КОМП'ЮТЕРНИЙ ЗІР, OCR, PYTTSX3, ІНТЕЛЕКТУАЛЬНА СИСТЕМА, СЛАБОЗОРІ.

Метою кваліфікаційної роботи є розробка програмно-апаратного модулю для допомоги особам з вадами зору у самостійному орієнтуванні на зупинках громадського транспорту, який забезпечує автоматичне розпізнавання типу транспорту та маршрутного номеру, а також генерацію голосових підказок для користувача.

У ході виконання кваліфікаційної роботи проаналізовано проблеми орієнтації слабозорих осіб у міському середовищі, розглянуто сучасні підходи до комп'ютерного зору, методи глибинного навчання та технології оптичного розпізнавання символів. Реалізовано систему на основі одноплатного комп'ютера Raspberry Pi 5 із використанням попередньо навченої моделі YOLOv8 для виявлення транспорту та Tesseract OCR для розпізнавання номера маршруту. Озвучення інформації здійснюється за допомогою бібліотеки pyttsx3. Система працює в автономному режимі без підключення до Інтернету, орієнтована на носиме використання та протестована в реальних умовах. Отримані результати підтверджують ефективність розробленого рішення та демонструють його потенціал до подальшого розвитку.

ABSTRACT

Bachelor’s thesis: 110 pages, 27 figures, 3 tables, 2 appendices, 40 sources.

RASPBERRY PI, DEEP LEARNING, YOLOV8, TESSERACT, OPENCV, COMPUTER VISION, OCR, PYTTSX3, INTELLIGENT SYSTEM, VISUALLY IMPAIRED.

The major goal of the qualification thesis is to develop a hardware-software module to assist visually impaired individuals in independently navigating public transport stops. The module provides automatic recognition of the type of transport and its route number, as well as generating voice prompts for the user.

During the implementation of the qualification work, the problems of orientation for visually impaired people in urban environments were analyzed, and modern approaches to computer vision, deep learning methods, and optical character recognition technologies were reviewed. A system was implemented based on the Raspberry Pi 5 single-board computer, using a pre-trained YOLOv8 model for vehicle detection and Tesseract OCR for route number recognition. Voice output is provided by the pyttsx3 library. The system operates autonomously without an Internet connection, is intended for wearable use, and was tested under real-world conditions. The results confirm the effectiveness of the developed solution and demonstrate its potential for further development.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ПЗ – програмне забезпечення

FPS – кількість кадрів за секунду (англ., Frames Per Second)

GPU – графічний процесор (англ., Graphics Processing Unit)

OCR – оптичне розпізнавання символів (англ., Optical Character Recognition)

OpenCV – бібліотека комп’ютерного зору з відкритим кодом (англ., Open Source Computer Vision Library)

OS – операційна система (англ., Operating System)

PD – подача живлення по кабелю (англ., Power Delivery)

RAM – оперативна пам’ять (англ., Random Access Memory)

USB – універсальна послідовна шина (англ., Universal Serial Bus)

YOLO – алгоритм швидкої детекції об’єктів (англ., You Only Look Once)

ВСТУП

Сьогодні, нажаль, достатньо гостро стоїть питання реабілітації громадян України з вадами зору. Наслідки військових дій призводять до ураження органів зору як військових, так і мирного населення України осколками, пилом, газом тощо. Оскільки забезпечення інклюзивного середовища в Україні наразі не є пріоритетною задачею через активне вливання коштів в оборонний сектор, держава не може облаштовувати вулиці міст засобами допомоги в орієнтації місцевістю для осіб з вадами зору. Особливо важливим елементом інфраструктури для незрячих є зупинки громадського транспорту, оскілки такі особи не мають права на водіння автомобілю. Не дивлячись на це, такі особи можуть користатися засобами індивідуального супроводу, наприклад, псами-поводирями, та це не є ідеальним рішенням через те, що собака не може озвучити людською мовою тип транспорту та його номер. Тому одним з чудових рішень проблеми орієнтації слабозорих на зупинках громадського транспорту є засоби комп’ютерного зору. З використанням технологій комп’ютерного зору та штучного інтелекту можна розпізнавати громадський транспорт, його тип та номер маршруту.

Засіб комп’ютерного зору зчитуючи дані з камери надає зображення на аналіз навченій моделі штучного інтелекту, що в комбінації з методами озвучування майже повістю зорієнтує слабозору особу на зупинці громадського транспорту шляхом озвучування побаченого. Такий модуль може значно підвищити рівень інклюзії в країні та допоможе в реалізації прав осіб з вадами зору на безбар’єрне пересування по вулицям міста на громадському транспорті.

1 ОГЛЯД ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Проблеми орієнтації слабозорих осіб у міському просторі

Сліпота – найбільш виражений ступінь втрати зору, коли неможливе або сильно обмежене зорове сприйняття навколишнього світу внаслідок глибокої втрати гостроти центрального зору, або звуження поля зору, або порушення інших зорових функцій.

За ступенем зображення залишкового зору відрізняють два типи: абсолютну сліпоту (повністю відсутні зорові відчуття) та практичну сліпоту (можливість сприймати світло або бачення форм) [2].

Розглядаючи проблеми орієнтації слабозорих осіб у міському просторі слід визначити, які взагалі можливості втрачає людина разом з втратою зору. Перш за все – орієнтацію в просторі. Виконання простих повсякденних справ, таких як: походи в магазин за продуктами, на навчання, до державних установ для оформлення та отримання документів, до закладів харчування, на зустрічі з друзями, наприклад, у парку для таких осіб стає справжнім випробуванням. Людина зіштовхується не лише з тими проблемами, що можуть її настигнути у місці до якого вона прямує (підписання документів, орієнтація в продуктових магазинах тощо), а й з проблемою добратися до цих місць.

Великі міста зазвичай налічують розгалужену та обширну систему міських доріг для автотранспорту (перехрестя, мости, тунелі, траси). Для переходу усіх цих дорожніх елементів існують різні рішення, такі як світлофори, підземні/надземні переходи, пішоході переходи. Слід зазначити, що у більшості своїй такі засоби орієнтовані саме на візуальне визначення місця переходу дороги.

Одна з достатньо навантажених вулиць міста Харкова – просп. Науки налічує 6 перехресть на кілометр, з яких лише 3 регульовані, але не мають

звукового сигналу переходу (рисунок 1.1).

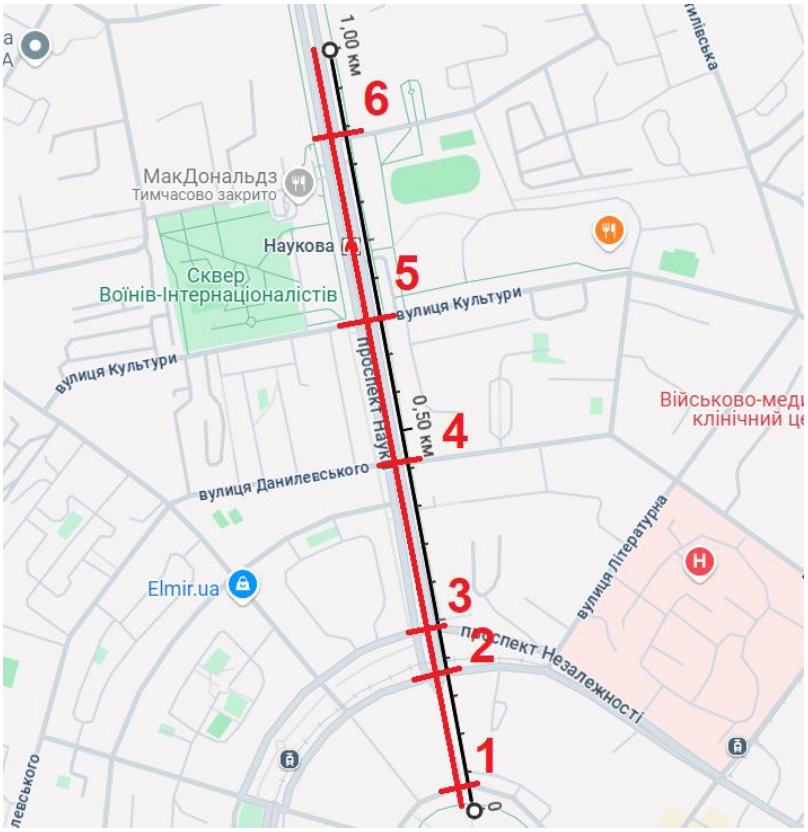

Рисунок 1.1 – Позначення кількості перехресть на 1 км на проспекті Науки

Такі обмеження в мобільності незрячих або слабозорих осіб не лише приносять труднощі в виконанні повсякденних справ, а й приносять відчуття неповноцінності та несамостійності таким особам, що негативно позначається на їх психологічному стані.

З результатів досліджень проведених за останні роки в Україні відомо, що в 2023 році нараховувалося 19 тис. громадян вадами зору. При тому у 2021 році це число становило 17 тис., що свідчить про зростання кількості осіб з вадами зору [4].

Коли в країні з’являється соціальна проблема, наприклад слабозорість, це породжує необхідність запуску процесів подолання даної проблеми. Стосовно питань інвалідності в суспільстві є декілька основних понять: абілітація, реабілітація та реінтеграція. Згідно з Законом України Про реабілітацію у сфері охорони здоров’я, абілітація – комплекс заходів, що

допомагають особі з вродженими та/або такими, що виникли у ранньому віці, обмеженнями повсякденного функціонування, досягти оптимального рівня функціонування у її середовищі [3]. Відповідно, реабілітація – це комплекс заходів, що допомагають особі у якої виникли обмеження повсякденного функціонування у зрілому віці.

У тому ж Законі Про реабілітацію у сфері охорони здоров'я, вказано, що існують допоміжні засоби реабілітації – будь-які зовнішні вироби (включаючи пристрої, обладнання, прилади чи програмне забезпечення), спеціально виготовлені або загальнодоступні, основною метою яких є підтримка або поліпшення функціонування та незалежності особи та сприяння її добробуту [3].

Реінтеграція ж має схоже тлумачення до реабілітації, воно також значить відновлення функціонування людини, але більше не індивідуального функціонування, а функціонування в суспільстві. Тобто, чим менше людина відчуває себе на такою як всі, тим краще проведена реінтеграція такої особи, наприклад, якщо слабозора особа користуючись зупинкою громадського транспорту не потребує питати який вид громадського транспорту наближається та який його номер.

Отже, спираючись на вищесказане можна зробити висновок, що слабозорим особам достатньо проблематично перебувати та функціонувати в сучасних міських умовах. Звертаючи увагу на кількість осіб з порушенням зору та тенденцію зростання їх кількості треба зробити все можливе, щоб такі особи мали змогу вільно переміщатися містом та реалізовувати свої права. Одним з рішень проблем абілітації та реабілітації є виготовлення допоміжного засобу, який допоможе може допомогти особам в підтримці їх функціонування та незалежності. Дивлячись на розвиток технологій Computer Vision які як раз розроблені для того, щоб комп'ютер міг бачити та аналізувати навколишнє середовище цей засіб буде технологічним і допоможе реабілітувати та реінтегрувати слабозорих осіб.

1.2 Аналіз існуючих рішень для допомоги слабозорим

За даними ВООЗ на 2023 рік близько 2.2 мільярди людей мають проблеми з зором [5]. Це число каже про те, що проблема сліпоти та слабозорості є світовою та важливою. На щастя, людство активно бориться з цією проблемою та винаходить різні підходи до покращення та полегшення життя людей з вадами зору. Активно допомагати незрячим та слабозорми в реабілітації люди почали в 20 віці і першим і найвідомішими засобом орієнтації в просторі для незрячих стала біла тростина.

Принцип роботи такої тростини достатньо простий: при переміщення людина рухає лише кистю руки в якій тримає тростину, наконечник тростини переміщується по дузі, крайні точки якої виступають за ширину корпуса на 5 см. з кожної сторони, нога повинна наступати туди, де перевірив простір наконечник. Таким чином, людина може приблизно розуміти куди їй ступати щоб не зачепити ногу та не впасти. Проте такий засіб допомагає орієнтуватися у просторі на дуже малій відстані [1].

Іншим достатньо розповсюдженим засобом є супровід псами-поводирями. Цей спосіб орієнтації є кращим за тростину, оскілки там де йде собака може йти і людина. Додатково такий помічним може допомогти перейти дорогу або швидше знайти дорогу додому або до магазину. Проте, пес-поводир це також жива істота, яку спочатку треба надресирувати для допомоги сліпим, потім власнику собаки необхідно піклуватися про неї. Якщо тварина захворіє, її треба лікувати та на час лікування людина буде без помічника. Такий спосіб зручніший за тростину, але не стабільний.

З розвитком технологій почали виникати нові рішення та засоби для покращення життя людей, підвищення ефективності на підприємництвах та в наукових дослідження. З виникненням нових технологій не забули і про слабозорих, і вчені та інженери почали проміняти нові технології в сфері орієнтації та реабілітації людей з вадами зору.

Таким чином, почали виникати модифікації вже існуючих засобів, як

наприклад WeWALK Smart Cane 2 – розумна тростина від компанії WeWALK.

Даний засіб надає можливість знати не тільки про пригороди на землі, а й над головою за допомогою спеціальних ультразвукових засобів. Також є можливість задати питання голосовому помічнику, щоб той надав інформацію та шлях до необхідних користувачу місць. Є інтеграція з розкладами громадського транспорту. Отже, це та ж біла тростина з приємними доповненнями, проте, вона потребує тісної інтеграції з мобільними телефоном та підключення до інтернету – у разі розрядження телефону або роз'єднання з мережею з додаткових функцій тростини залишиться лише детекція пригород над головою. Функції голосового помічника в тростині є платними. Озвучування лише на англійській мові [39].

Слід спам'ятати про такий засіб як Sonic Pathfinder. Цей засіб є вторинним для орієнтації у просторі і використовується в дуеті з білою тростиною або псем-поводирем. Даний пристрій надягається на голову та шляхом генерації ультразвукових хвиль аналізує середовище на рівні голови користувача, та якщо знаходить пригороду, повідомляє про це вібраційним сигналом. Даний пристрій є прабатьком подібної технології в вже спом'янутій WeWALK Smart Cane 2 та інших схожих розробках. Недоліком технології є те, що вона лише вторинна [36].

Спираючись на те, що в наші часи майже кожна людина має смартфон з доступом в інтернет та вбудованими технологіями GPS, було розроблено низку мобільних застосунків для супроводу слабозорих, наприклад, BlindSquare. Даний застосунок зчитує місце перебування користувача по геолокації смартфону, та по запиту звуковому або невербальному (потрясши телефон озвучується місцезнаходження користувача) про місце в яке треба добратися будує маршрут та озвучує його. Підтримує велику кількість мов. Просте і зручне рішення, але також є більш вторинним, потребує підключення до інтернету та не дає одночасно користуватися іншими функціями смартфону, а також використовує його ресурси, що може

призвести до швидкого розрядження пристрою [40].

З розвитком технологій Computer vision ці технології почали ефективно проміняти і й в допомозі слабозорим та нові більш ефективні рішення почали появлятися на ринку.

Наприклад, OrCam MyEye 3 Pro – спеціальний модуль, який чіпляється з боку окулярів та аналізує те, що бачить. Пристрій озвучує текст, обличчя, продукти в магазині, працює офлайн та має можливість підключення до навушників для більш точного інформування користувача. Проте, цей пристрій ніяк не допомагає в навігації та має велику ціну (від 4000$ до 5000$) і як і всі інші засоби є вторинним доповненням до тростини або собаки-поводиря [24].

Слід зазначити, що всі вищеперераховані засоби відносяться до індивідуальних, які користувач сам собі обирає та використовує як хоче. Існують також засоби, які встановлює держава на вулицях міст або приватні підприємства на своїх територіях, які задовільнять потреби осіб з вадами зору.

Наприклад, тактильні плитки на тротуарах допомагають розуміти де кінець пішохідної дороги, в якому напрямку вона йде и де повертає. Також, деякі підприємства, в основному медичні, на своїх територіях вішають таблички з шрифтом Брайля, що надає можливість слабозорим особам розуміти, в який кабінет вони заходять або в яку аптеку. Нажаль, тактильних тротуарів в Україні дуже мало і не кожна аптека та лікарня встановлює на своїй базі таблички з шрифтом Брайля.

Також існують озвучені системи навігації: звукові сигнали на зеленому світлі світлофору, інфозвукові маяки які знаходяться біля входів, сходів, ліфтів тощо. Озвучування в зупинок в громадському транспорті. Такі системи вже більш розвинуті в Україні, проте, не у всіх містах.

Для більш зручного та ознайомлення їх можна побачити в таблиці 1.1. У зв’язку з тематикою роботи до таблиці заносено саме технічні рішення, оскільки сучасні технології дозволяють значно краще допомагати слабозорим

в адаптації навколишнім середовищем: нові рішення комп'ютерного зору, або ж модифікації класичних рішень.

Таблиця 1.1 – Засоби допомоги та супроводу незрячих: переваги та недоліки кожного з рішень

№	Рішення	Переваги	Недоліки
1	WeWALK Smart Cane 2	Детекція перешкод зверху; Голосовий AI помічник; Навігація; Біла тростина і супровідні функції в одному пристрої.	Для повного функціонування необхідно мати доступ в мережу;
2	Sonic Pathfinder 	Детекція перешкод зверху; Поштовх до розвитку майбутніх технологій.	Навантаження на голову; Вторинний.
3	BlindSquare 	Не потребує додаткових пристроїв; Навігація; Голосовий супровід.	Потребує доступ в мережу; Не дає використовувати інші застосунки смартфону.
4	OrCam MyEye 3 Pro 	Працює без доступу в інтернет; Розпізнає велику кількість об'єктів; Має голосового AI асистента.	Ціна 4000$ Не допомагає в навігації.

Отже, при вирішенні проблеми орієнтації слабозорих на зупинках громадського транспорту слід взяти усі кращі характеристики та не допустити наявності негативних сторін.

1.3 Аналіз структури задачі: умови використання, інформаційні потоки

Ознайомившись з усіма підходами до вирішення проблем орієнтації слабозорих осіб в просторі описаними у попередньому підпункті, можна помітити, що деякі з них зачепили питання пересування громадським транспортом, але наведені рішення працюють лише з доступом у мережу та підключенням до смартфону. Тому доцільним є продумати інший підхід, який буде вирішувати типові проблеми з якими людина з вадами зору може зітнутися під час користування зупинками громадського транспорту.

Для вирішення даного питання, треба розуміти, як взагалі слабозора особа може користатися зупинками громадського транспорту, які проблеми можуть виникнути, які з них вирішуються технічним підходом та яким саме.

Перш за все, людина планує свій шлях: на яку зупинку та який маршрут їй потрібен. Це легко досягається шляхом голосового пошуку та отримання звукової відповіді в пошуковій системі Google. Далі людина виходить з житла та починає свій шлях до зупинки громадського транспорту. На цьому етапі вже існують засоби для допомоги в навігації: біла тростина або її технічно оснащена версія, пес-поводир, маршрут може озвучувати застосунок Google Maps або BlindSquare. Дійшовши до зупинки громадського транспорту незряча особа очікує необхідний транспорт, проте, тут і виникає проблема. У більшості випадків транспорт який приїзжає не озвучує свій номер. Людина може спитати інших про номер транспорту, але що робити якщо нікого немає на зупинці? Спитати водія може бути проблематично у зв’язку з гучним двигуном. Тому в такій ситуації необхідне додаткове рішення, яке б не займало смартфон у разі використання на ньому навігаційних карт та мого б незалежно працювати у разі розрядження

смартфону.

Пристрій, який може вирішити питання орієнтації на зупинках громадського транспорту може мати наступний вигляд: основний модуль з прикріпленим до нього акумулятором, системою охолодження, кнопкою ввімкнення з спеціальним механізмом для закріплення на поясі має розмір 8.7 см в довжину і 5.5 см в ширину, від нього йде два проводи: один з камерою, інший з навушником.

З використанням спеціального модуля для супроводу на зупинках громадського транспорту алгоритм дій слабозорої особи змінитеся. Перед виходом людина закріплює модуль на поясі, чіпляє камеру в районі грудей та вставляє навушник в вухо. Після цього доходить до зупинки громадського транспорту та вмикає модуль шляхом натиснення на відповідну кнопку на модулі. Після цього модуль запускаєтеся та аналізує видимість для того, щоб у подальшій роботі застосовувати відповідні фільтри для покращення видимості об’єктів. Тобто, якщо на вулиці вечір, модуль це розуміє та накладає фільтр висвітлення зображення. Після цього користувач отримує звукове повідомлення про готовність роботи модуля та займає відповідну позицію: підходить до бордюру поруч з дорогою та дивиться в напряму набли транспорту (рисунок 1.2).

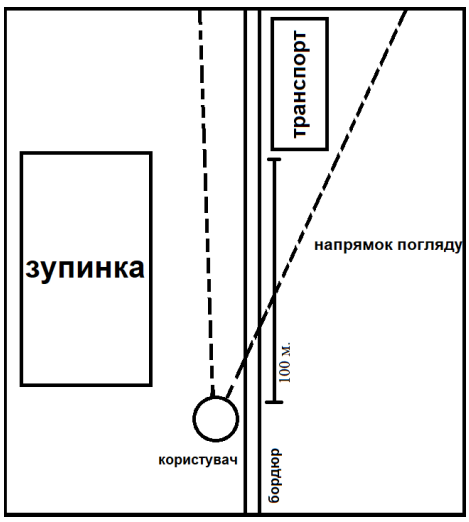

Рисунок 1.2 – Схематична проекція положення користувача на зупинці громадського транспорту

Камера один раз на певний проміжок часу захоплює кадр та надсилає на обробку. Коли один з видів громадського транспорту (автобус, тролейбус, трамвай) попадає в поле зору програмне забезпечення на основі натренованої моделі ШІ. Проаналізувавши вид транспорту мережа розпізнаю номер на ньому. Після цього побачене модулем озвучується користувачу.

Такий засіб не є незалежним від інтернету та смартфону, і залежить лише від ємкості акумулятора та кількості заряду. Механізм обробки зображень в різних умовах видимості присутній. Єдиним обмеженням може бути використання приладу в умовах підвищеної вологості, конкретно в умовах дощу.

1.4 Системи розпізнавання транспорту: аналіз підходів

Існують різні способи збору даних про оточуюче середовище та їх переведення у цифровий формат (рисунок 1.3).

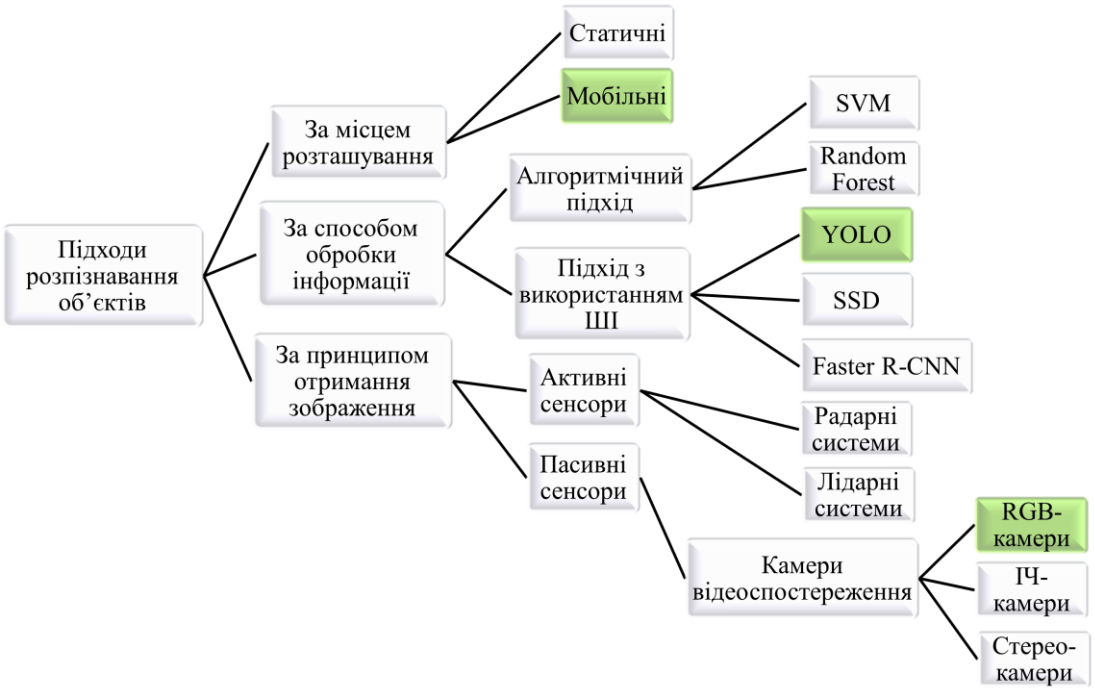

Рисунок 1.3 – Діаграма підходів класифікації та розпізнавання об'єктів

Радар (Radio Detection and Ranging) – це активний сенсор, який випромінює радіохвилі та аналізує їх відбиття від об'єктів для визначення їх положення, швидкості та інших характеристик. У контексті комп'ютерного зору, радар забезпечує:

- дальність виявлення: здатність виявляти об'єкти на значній відстані;
- стійкість до погодних умов: ефективна робота в умовах дощу, туману та снігу;
- вимірювання швидкості: здатність визначати швидкість об'єктів за допомогою ефекту Доплера.

Однак, радар має обмежену роздільну здатність і не забезпечує детальну інформацію про форму об'єктів. Це обмеження можна подолати шляхом поєднання даних радара з іншими сенсорами, такими як лідар, для покращення точності виявлення об'єктів [38].

Лідар (Light Detection and Ranging) – це активний оптичний сенсор, який використовує лазерні імпульси для створення високоточної тривимірної карти навколишнього середовища. Основні переваги лідара включають:

- висока роздільна здатність: здатність точно відображати форму та розміри об'єктів;
- точне визначення відстані: забезпечує точні вимірювання відстані до об'єктів.

Проте, ефективність лідара може знижуватися в несприятливих погодних умовах, таких як туман або сильний дощ. Для підвищення надійності систем виявлення об'єктів, лідара часто поєднують з радаром, що дозволяє компенсувати недоліки кожного з сенсорів [33].

Звичайна ж RGB-камера формує зображення завдяки збиранню видимого світла. Розпізнає кольори, форми, текстури об’єктів. Камери мають перевагу в простоті використання, вони достатньо доступні та мають велику кількість попередньо підготовлених алгоритмів обробки. Але, їхня ефективність знижується при слабкому освітленні, у тумані або вночі.

Інфрачервоні камери (тепловізори) працюють в ІЧ-діапазоні та

виявляють теплове випромінювання об'єктів. Це дозволяє бачити у повній темряві, через дим або легкий туман. Та дана технологія достатньо вузьконаправлена і необхідна саме в нічних умовах.

Стерео-камери складаються з двох або більше звичайних камер, що дозволяє отримувати інформацію про глибину за допомогою принципу триангуляції. Тринґуляція ж це визначення відстані до об’єктів шляхом порівняння зображень з двох точок зору. Також вузьконаправлена технологія саме для визначення відстаней.

Через наявні недоліки та велику коштовність визначених вище технологій для реалізації роботи даного методу розпізнавання громадського транспорту як пристрій для передачі вхідних даних обрано саме RGB-камеру через її доступність та простоту.

Щодо питання аналізу та класифікації транспортних засобів, класичні методи виявлення об'єктів базувалися на використанні вручну заданих ознак, таких як контури, кольори, форми та розміри. Для цього застосовували фільтри, наприклад оператори Sobel чи Canny, аналіз гістограм кольору, а також виявлення геометричних характеристик об’єктів. Такі підходи працювали задовільно лише в контрольованих середовищах і були дуже чутливими до шуму, зміни освітлення та часткових перекриттів. У реальних умовах, особливо на вулиці, ці методи виявились неефективними, оскільки не могли забезпечити достатню точність і надійність [31].

З розвитком машинного навчання з’явилися більш гнучкі підходи, які поєднували вручну витягнуті ознаки з алгоритмами класифікації. Найчастіше застосовувалися алгоритми на кшталт SVM (метод опорних векторів) та Random Forest. Вони покращили точність порівняно з класичними методами та були здатні працювати в різноманітних середовищах, однак усе ще вимагали якісної ручної підготовки ознак, що обмежувало їх застосування в складних динамічних умовах [15].

Справжній прорив у галузі виявлення об’єктів стався з появою глибокого навчання. Алгоритми на базі згорткових нейронних мереж (CNN)

навчилися автоматично витягувати релевантні ознаки з великих обсягів даних. Найвідомішими та найефективнішими моделями сьогодні є YOLO (You Only Look Once), SSD (Single Shot Multibox Detector) та Faster R-CNN. Зокрема, YOLOv8 продемонстрував чудовий баланс між швидкістю та точністю, досягаючи показника mAP@50 0.62 із затримкою лише 1.3 мс, що робить його ідеальним вибором для застосувань у реальному часі [19]. SSD також дозволяє виконувати виявлення швидко, але дещо поступається за точністю, тоді як Faster R-CNN забезпечує найвищу точність, проте повільніший через двоетапний процес обробки [11].

Таким чином, сучасні системи виявлення об'єктів спираються передусім на глибоке навчання, яке об'єднує витяг ознак і класифікацію в єдину модель, забезпечуючи надійну роботу в реальному середовищі навіть за складних умов, тому саме YOLO було обрано як модель для навчання.

Щодо класифікації громадського транспорту, основні наземні засоби переміщення це автобуси, тролейбуси та трамваї. Відрізнити їх достатньо просто. При розмітці датасету достатньо визначати контури автобусу від колес до даху, у тролейбуса від колес до кінця струмоприймачів, а у трамваїв від колес до кінця пантографу. Модель зможе відрізнити тролейбус від автобуса тим, що автобус не має струмоприймачів, а тролейбус від трамвая ти, що трамвай не має колес та має іншу форму корпусу та пантографу.

1.5 Мета та задачі дослідження

Отже, спираючись на все вище сказане, на те, що зважаючи на підвищення кількості слабозорих осіб в Україні з початком війни, проаналізувавши підходи до вирішення проблем з навігацією та орієнтацією і виявивши те, що дуже мало рішень звертають увагу на проблеми орієнтації на зупинках громадського транспорту. Тому метою кваліфікаційної роботи є розробка програмно-апаратного модулю для допомоги особам з вадами зору у самостійному орієнтуванні на зупинках громадського транспорту, який

забезпечує автоматичне розпізнавання типу транспорту та маршрутного номеру, а також генерацію голосових підказок для користувача.

Слід зазначити, що при досягненні поставленої мети існує низка обмежень, зумовлених як технічними, так і організаційними чинниками:

- складність реалізації вологостійкості апаратного модуля;
- обмеження в часі автономної роботи апаратного модуля;
- складність в організації збору великої кількості графічного матеріалу для датасету;
- вартість компонентів, яка обмежує можливість розгортання системи у великому масштаб.

Для реалізації системи необхідні такі типи вхідних даних:

- зображення транспортних засобів, зняті з персонального пристрою (кадри у форматі RGB);
- розмічені датасети, які містять координати об’єктів (bounding boxes) для типів транспорту (автобус, тролейбус, трамвай) та номерів маршрутів;
- аугментовані дані із симульованими погодними ефектами: туман, дощ, ніч, сутінки, тощо;
- аудіосигнали, які транслюються через динамік у відповідь на розпізнавання.

Отже, для досягнення поставленої мети необхідно виконати низку задач:

- збір та розмітка даних для глибинного навчання моделі YOLO розпізнаванню різних типів транспортних засобів (автобус, тролейбус, трамвай) та їх маршрутні номери;
- аугментація датасету шляхом моделювання умов низької видимості, створюючи штучні погодні умов (туман, дощ, ніч тощо) в різний час доби (денне світло, сутінки, темрява);
- розробка програмного модуля ідентифікації транспорту із застосуванням методів комп’ютерного зору;
- розробка програмного модуля голосового супроводу на основі

методів синтезування мовлення для відтворюння в режимі реального часу через аудіоінтерфейс розробленого пристрою;

- побудова апаратного модуля для збору, аналізу та постпроцесінгу даних;
- тестування та адаптація системи до реальних умов експлуатації.

Подальший розвиток проєкту полягає у розширенні функціоналу для роботи з розумними зупинками (IoT-інфраструктура), інтеграції з мобільним додатком для персоналізації налаштувань, використанні технологій штучного інтелекту для адаптації системи до різних користувачів [4].

2 ОБГРУНТУВАННЯ ВИБОРУ МЕТОДІВ ТА ТЕХНОЛОГІЙ

2.1 Аналіз технологій комп’ютерного зору для детекції транспортних засобів

Спираючись на все описане в першому розділі даної роботи можливим рішенням проблеми орієнтації слабозорих осіб на зупинках громадського транспорту буде пристрій, який складається з апаратної частити: одноплатний комп’ютер, камера, навушники та акумулятор; та програмної: програма збору зображень, їх обробки, аналізу та озвучування побаченого.

Оскільки основною технологією для детекції транспортних засобів було обрано модель YOLOv8 від Ultralytics та обґрунтовано в підрозділі 1.4, слід проаналізувати та визначити механізм роботи даної моделі: формат датасету, процес навчання моделі та критерії визначення її готовності.

Датасет – це набори пар вхід-вихід, які відповідають, якщо дійсні, інтенсіональному визначенню завдання. Інтенсіональне ж визначення описує зв'язок між входом і виходом (наприклад, вихід в автоматичному розпізнаванні мовлення є транскрипцією аудіосигналу на вході) [37]. Тобто, датасет слугує набором даних є представлення вхідних даних (зображення) вихідними даними (координати знаходження цільового об’єкту на зображенні). Для вирішення задач детекції громадського транспорту та його номерів можна виділити датасети за умовами середовища, типом задач, за наявності глибинної інформації та джерелом даних [18]. Оскільки розробляється рішення для слабозорих на зупинках громадського транспорту, то датасет буде складатися з зображень транспорту міста Харкова, з синтетичними перешкодами та без, окремого датасету для OCR не потрібно, оскільки буде використовуватися існуюче рішенняю. Кмера буде звичайна без глибини та весь пристрій буде мобільним та висіти на користувачі (рисунок 2.1).

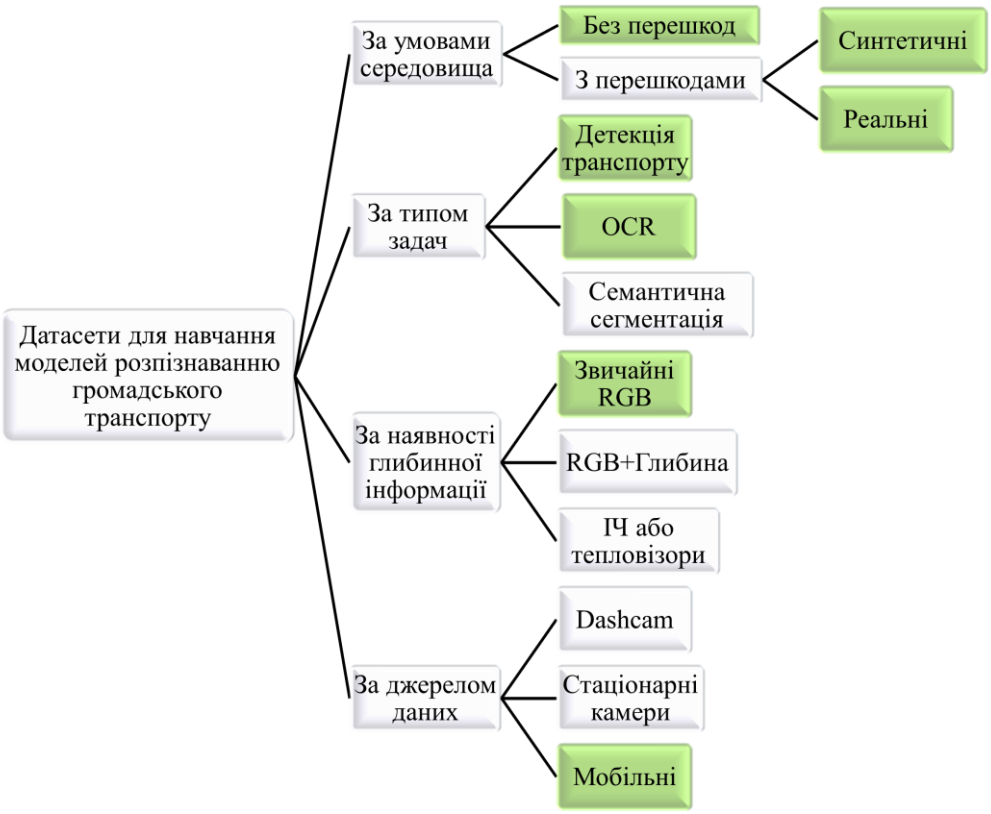

Рисунок 2.1 – Класова діаграма різновидів датасетів

Простими словами, датасет представляє собою набір зображень з позначеними на ньому об’єктами (рисунок 2.2).

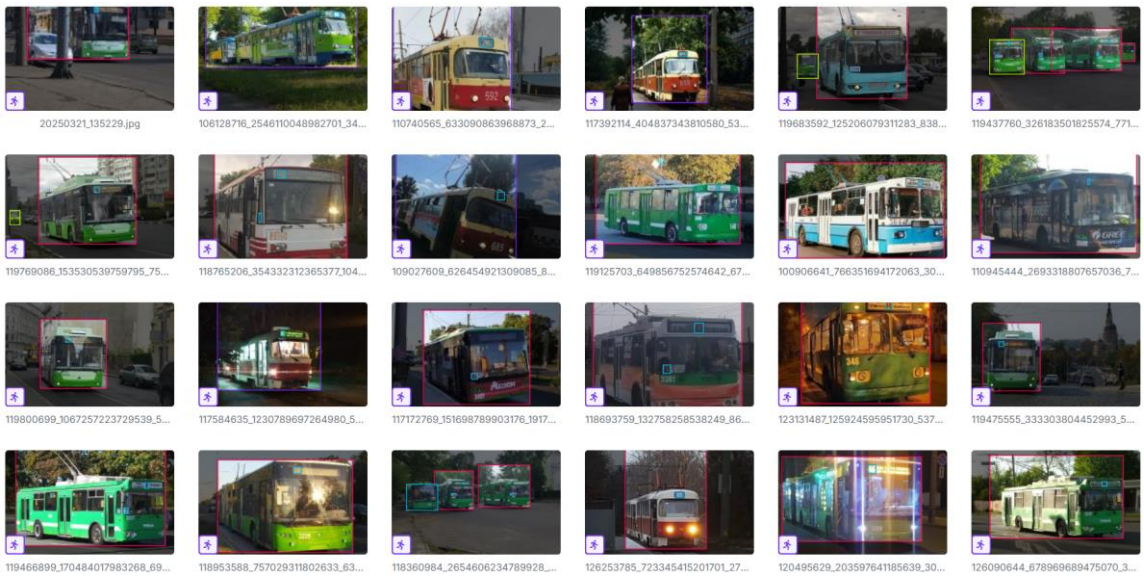

Рисунок 2.2 – Розмічений датасет з громадським транспортом на сервісі Roboflow

Кожна модель має свій спеціальний формат датасетів і YOLO не виключення. Формат Ulrtalytics YOLO – це формат конфігурації набору даних, який дозволяє визначити кореневий каталог набору даних, відносні шляхи до каталогів зображень для навчання/оцінки/тестування або *.txt файли, що містять шляхи до зображень та словник імен класів [21].

Для розуміння всього процесу підготовки датасету на сайті Ultralytics наведено відповідну інструкцію з наступними кроками:

- зібрати зображення, які можна завантажити з викритих джерел, публічних датасетів або зробити самостійно;

- анотувати зображення з використанням обмежувальних рамок, сегментів або ключових точок, залежно від завдання;

- експортувати анотації в формат YOLO. Даний формат має вигляд .txt файлу, де кожен рядок відповідає об’єкту на зображенні. Представлення об’єкту має наступний вигляд: class x_center y_center width height;

- організувати дата сет по директоріям, де є основна директорія, наприклад – “dataset”, в ній директорії “train/” та “val/”, в кожній з яких директорії “images/” та “labels/” з зображеннями та анотаціями .txt (зображення та його анотації повинні мати однакову назву);

- створити конфігураційний файл “data.yaml”, де описується організація датасету та його класів [8].

Таким чином, якщо всі кроки виконано вірно, буде отримано підготовлений датасед для навчання моделі.

Щодо процесу навчання моделі, підготовлений датасет передається до скрипту навчання, наприклад, на мові програмування Python (лістинг 2.1).

Лістинг 2.1 – Приклад вмісту скрипту для навчання моделі YOLOv8n

```
from ultralytics import YOLO

# завантаження моделі
model = YOLO("yolo8n.pt")  # завантаження пустої не натренованої моделі

# команда тренування моделі
```

```
results = model.train(data="data.yaml", epochs=100, imgsz=640) #
в параметрах вказується конфігураційний файл, кількість епох
тренування та розмір
```

Якщо детальніше зануритися в процеси навчання моделі, то необхідно розглянути процес глибинного навчання. Глибинне навчання базується на шарах:

- вхідний шар, де передаються сирі дані, наприклад зображення у представлення матриці пікселей;

- приховані шари, в яких відбувається обчислення: витягуються ознаки, знаходяться шаблони;

- вихідний шар, який видає найймовірніший результат, наприклад, клас об’єкту.

Кожен шар складається з нейронів, які приймають кілька чисел на вході (вектор), множать їх на ваги (коефіцієнти), додають зміщення (bias), пропускають результат через активаційну функцію (наприклад, ReLU, Sigmoid).

Визначається нейрон наступною формулою:

$$y = \sigma(\sum_{i=1}^{n} \omega_i x_i + b) \quad (2.1)$$

,

де y – вихід нейрона; σ – активаційна функція; w_i – вага; x_i – вхід; b – зміщення.

Загальний алгоритм обробки даних має основні етапи:

- дані передаються зверху вниз;

- обчислюються втрати, тобто, наскільки припущення відрізняється від правди, наприклад, функцією кросс-ентропії:

$$Loss = -\sum y_{\text{істина}} \cdot \log\left(y_{\text{припущення}}\right) \quad (2.2)$$

;

- спрацьовує зворотнє поширення з використанням ланцюгового правила похідних:

$$\frac{\partial L}{\partial \omega} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial \omega} \quad ; \tag{2.3}$$

- відбувається оновлення вагів:

$$\omega_{\text{новий}} = \omega_{\text{старий}} - \alpha \cdot \frac{\partial L}{\partial \omega} \quad . \tag{2.4}$$

Приблизно такий процес відбувається при навчанні моделі. Та постає питання: як зрозуміти, що модель навчилася. Основною метрикою для визначення є mAP.

Модель вже має завантажені датасети, та розуміє де рамки об’єкту та який його клас завдяки анотаціям. Також, у процесі навчання модель має передбачені моделі, які містять координати прямокутника, клас та впевненість. Для кожного передбачення обчислюється IoU (Intersection over Union) за формулою IoU = площа перетину/площа об’єднання, та за отриманими значеннями оцінює на скільки модель гарно вгадала місце об’єкта. Якщо IoU >= 0.5, то це True Positive (TP), інакше - False Positive (FP). Якщо модель взагалі нічого не знайшла, але об’єкт був у датасеті, то це False Negative (FN).

Модель передбачає багато рамок з різними значеннями довіри. Відбувається сортування їх від найбільшої довіри до найменшої і для кожного порогу визначається Precision = TP / (TP + FP) – точність та Recall = TP / (TP + FN) – повнота. Потім будується крива Precision vs Recall: для кожного рівня confidence є значення (P, R). Крива змінюється в залежності від порогу.

Далі обчислюється площина під кривою (P, R), це є AP (Average Precision). Наприклад, якщо площа під кривою Precision-Recall = 0.85 – це AP = 85%. Далі і отримується той самий mAP (mean Average Precision): обчислюєтеся AP для кожного класу, після чого знаходиться середнє арифметичне – це і є mAP. Чим ближче значення mAP до 1, тим краще навчена модель [12].

Отже, дана технологія притаманна виявленню будь яких об’єктів на зображеннях. Це стосується і громадського транспорту – така технологія чудово допоможе в задачі детекції автобусів, тролейбусів та трамваїв.

2.2 OCR-розпізнавання номера маршруту

У зв’язку з актуалізацією Computer vision у сучасних молодих проектах та стартапах та з розвитком технологій штучного інтелекту, на сьогоднішній день з’явилося багато OCR-систем, серед яких: ABBYY FineReader, Google Cloud Vision API, Amazon Textract, PaddleOCR, EasyOCR та Tesseract (рисунок 2.3).

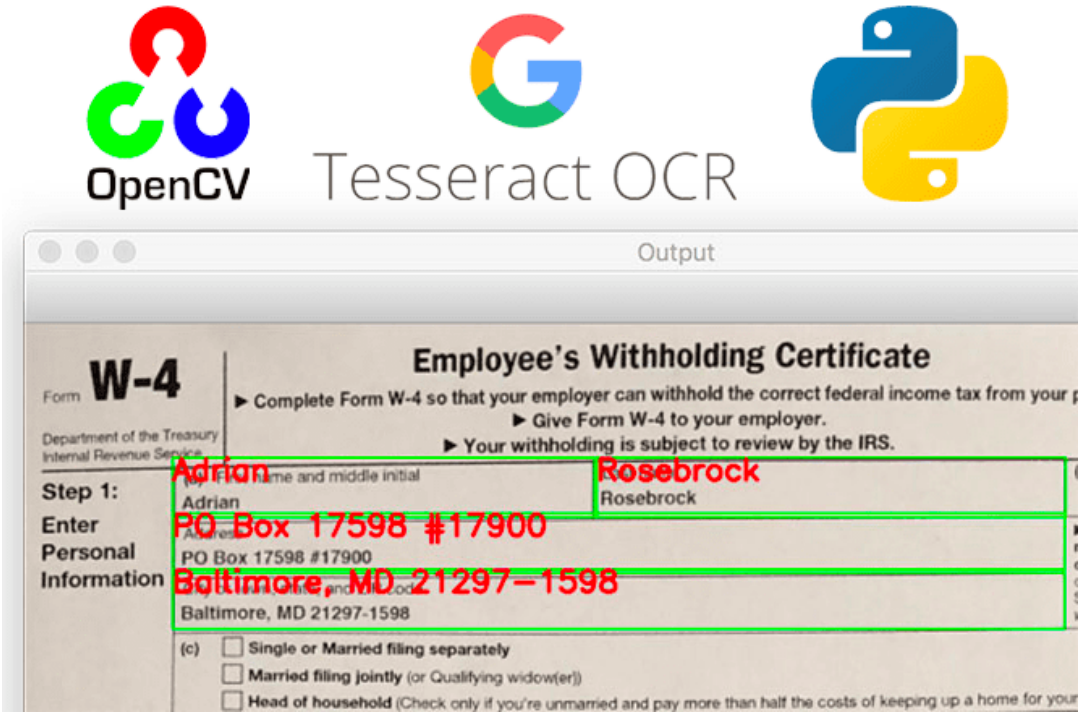

Рисунок 2.3 – Приклад використання Tesseract OCR з Python та OpenCV

FineReader комерційний продукт, має велику точність розпізнавання тексту і підтримує велику кількість мов, проте має обмеження щодо відкритості коду та безкоштовного використання. Google Cloud Vision API та Amazon Textract є гарними хмарними рішеннями, проте не безкоштовні, не

потребують ресурсів пристрою, але потребують стабільного інтернет-з’єднання, що не відповідає меті досліджуваного проєкту. Більш доступними же є PaddleOCR, EasyOCR та Tesseract.

EasyOCR є більш молодою бібліотекою та базується на PyTorch. Використовує технологію CRNN (Convolutional Recurrent Neural Networks).

CRNN На вхід подається послідовність даних, зокрема зображення тексту. Згорткові шари витягують просторові ознаки, як-от контури та форми символів. Далі ці ознаки передаються у рекурентні шари, які аналізують послідовність та зберігають контекст попередніх фрагментів. Передача від згорткових до рекурентних шарів відбувається через семплювання (зменшення розмірності), що зберігає важливі характеристики зображення. На виході модель формує послідовність символів через повнозв’язаний шар, що генерує остаточне передбачення, наприклад — розпізнаний текст [7].

EasyOCR добре розпізнає текст і в звичайних і в складних умовах, проте, потребує багато ресурсів та менш надійна в вбудованих системах без GPU, а апаратна база роботи не передбачає використання графічних процесорів [10].

PaddleOCR – популярний фреймворк на базі PaddlePaddle від Baidu. Показує велику ефективність в розпізнаванні тексту та так же як і EasyOCR використовує технологію CRNN. Для детекції використовує DBNet. Проте, дане рішення орієнтоване більше на хмарні рішення та погано оптимізований для пристроїв на ARM-архітектурі. Апаратна база досліджуваного модуля має саме таку архітектуру, тому використання даного фреймворку не підходить [25].

Найкращим та найнадійнішим рішенням в контексті досліджуваного апаратно-програмного модуля є Tesseract-OCR. Дана технологія підтримується та покращується вже достатньо довгий час (з 2006 року). Підтримує багато мов, починаючи з версії 4.0 використовує LSTM (Long Short-Term Memory) нейронні мережі для покращення точності розпізнавання.

Ключовою особливістю LSTM є наявність спеціальних компонентів, зокрема комірки пам'яті та трьох типів “воріт” (вхідних, вихідних та воріт забування), які регулюють потік інформації та дозволяють зберігати важливі дані протягом тривалих періодів часу. Це забезпечує ефективне навчання на послідовностях великої довжини, що особливо корисно в задачах обробки природної мови, розпізнавання мови та інших послідовнісних даних [34].

Однією з головних переваг Tesseract є локальне функціонування без потреби у GPU або зовнішніх сервісах, що критично важливо для вбудованих пристроїв і систем, які працюють офлайн. Tesseract також дозволяє гнучко налаштовувати розпізнавання (наприклад, whitelist символів, режим сегментації, мову), що особливо ефективно у задачах, де потрібно розпізнавати лише цифри маршруту [32].

Отже, через свою невибагливість до хмарних технологій та графічних процесорів, серед всіх актуальних на даний момент систем розпізнавання символів Tesseract-OCR краще сього підходить для досягнення мети роботи.

2.3 Застосування бібліотеки OpenCV для обробки відео

OpenCV або ж Open Source Computer Vision Library – бібліотека з відкритим кодом, призначена для обробки зображень та відео. Може забезпечувати ефективні засоби захоплення, аналізу та трансформації кадрів у режимі реального часу [6][23].

Перш за все, для того щоб почати процес обробки необхідно отримати те, що необхідно обробити. За це в бібліотеці відповідає клас cv2.VideoCapture, в який можна передати індекс камери встановленої на пристрої або шлях до відео файлу. Метод read() повертає поточний кадр у вигляді матриці зображення. Для малопотужних пристроях може потребуватися можливість зменшити частоту захоплення кадрів і роздільну здатність, і для цього існує метод set() [23].

Після захоплення кадру його вже можна обробляти та застосовувати

певні перетворення. Оскільки кадр представлено як масив пікселів, то до усі методи форматування легко приміняться до зображення шляхом зміни відтінку кожного пікселя. Так, наприклад метод cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) перетворює всі відтінки зображення на сірі (рисунок 2.4) [28], що часто використовується у роботі з системами глибинного навчання та OCR, на які направлене дослідження. Також особливо корисними та використовуваними для покращення якості зображень номерів маршрутів та не тільки є методи згладжування (cv2.GaussianBlur), порогової фільтрація (cv2.threshold) та бінаризації з адаптивним порогом (cv2.adaptiveThreshold) [17].

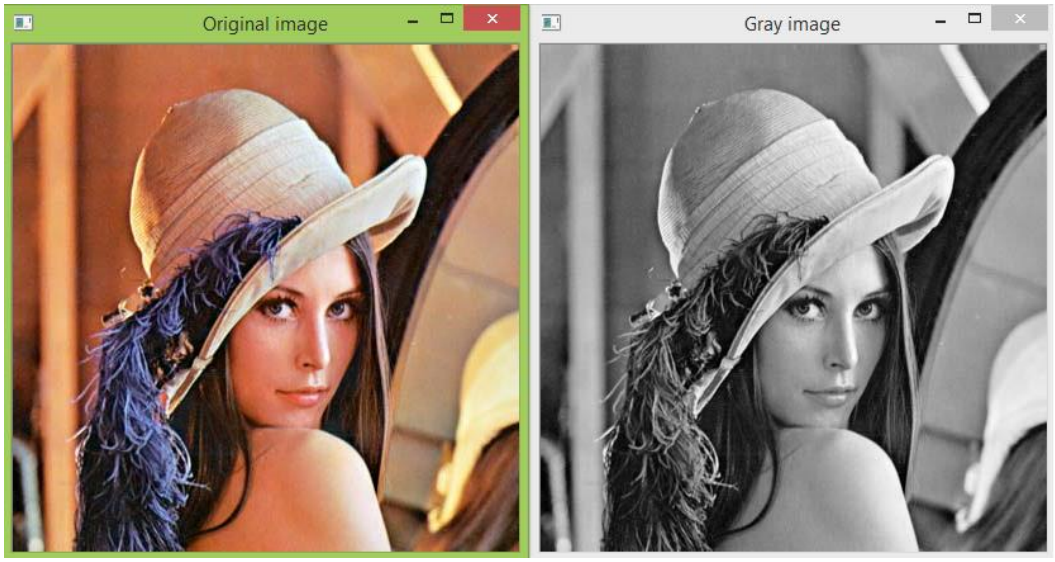

Рисунок 2.4 – Приклад застосування методу cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

Важливими функціями обробки зображення також є масштабування та обрізання кадрів. З використанням команди cv2.resize можна привести зображення до стандартного розміру, що знижує навантаження на системи глибинного навчання. Функція frame[y1:y2, x1:x2] забезпечує виділення лише релевантної області кадру для подальшої роботи з ним, наприклад, область з номером маршруту.

З використанням функцій фільтрації зображення також можна

враховувати умови зовнішнього середовища. Наприклад, при виявленні поганих умов (туман, ніч, дощ) можна застосовувати алгоритми покращення контрасту, такі як CLAHE (Contrast Limited Adaptive Histogram Equalization), або додаткові фільтри згладжування, щоб зменшити шум. Застосування таких методів істотно підвищує якість вхідних зображень і, відповідно, точність роботи OCR та моделей глибинного навчання [35].

Отже, OpenCV може забезпечити усі функції необхідні для ефективного захоплення та обробки відеопотоку. Конкретно для досліджуваного проєкту можливість бібліотеки захоплювати та обробляти кадри для подальшої роботи з ними моделі YOLO та вирізання та переведення у сірі відтінки фрагментів з номерами маршрутів робить її незамінним компонентом майбутнього модуля детекції громадського транспорту та номерів маршрутів.

2.4 Вибір технології генерації голосових повідомлень

Існує велика кількість програмних підходів для озвучування голосових повідомлень на сьогодні. Дана функція є зручною не тільки для повсякденних справ як, наприклад, голосові запити в інтернеті, а й ключовою для засобів допомоги слабозорим. Розповсюдженими є такі технології як gTTS (Google Text-to-Speech), TTS (Coqui.ai), SpeechDispatcher, pyttsx3.

Найпопулярнішою технологією переведення тексту в голос озвучування якої чули майже всі користувачі сучасних технологій – це gTTS. Дана бібліотека використовує API Google для синтезу мовлення, має високу якість голосу, підтримує багато мов, проте, не може працювати локально та потребує постійного підключення до мережі [14].

TTS від Coqui.ai відносно молода сучасна бібліотека глибокого навчання, яка допомагає створювати нейромережеві моделі голосу. З її використанням можна досягти майже природний голос, але навчання моделі може зайняти багато часу та потребувати багато ресурсів, а рішення яке

розглядається в даній роботі не потребує наближеного до реального голосу.

Найзручнішою бібліотекою для локального незалежного озвучування є pyttsx3. Це офлайн-бібліотека, яка використовує рушії мовлення системи на якій використовується. Наприклад, SAPI5 на Windows, NSSpeechSynthesizer на macOS, espeak на Linux [26]. Даний підхід дозволяє технології працювати без підключення до мережі та без завантаження додаткових рушіїв мовлення. Має можливість налаштування темпу, гучності мовлення та надає можливість створювати асинхронне озвучування, що робить її ідеальною для використання на малопотужних комп'ютерах та для озвучування побаченого в реальному часі [16].

Оцінка затримки в озвучуванні є дуже важливим аспектом для систем озвучування побаченого в системах реального часу, де побачене повинне озвучуватися одразу після детекції. Наприклад, gTTS має суттєву затримку через необхідність виконання запиту на сервер Google. При стабільному зв'язку ця затримка може становити 1-2 секунди [13]. Аналізуючи рух мобільних об'єктів така затримка може буди занадто великою.

Натомість бібліотека pyttsx3, яка працює повністю офлайн, забезпечує мінімальну затримку. Озвучення запускається локальним рушієм (наприклад, espeak або SAPI5), і аудіопотік формується безпосередньо в оперативній пам'яті, без участі зовнішніх серверів. Як показує практика, повний цикл генерації мовлення та початку відтворення триває лише кілька сотень мілісекунд, що дає змогу практично миттєво реагувати на зміну ситуації [27]. Додатково, pyttsx3 добре працює з багатопотоковою обробкою: синтез мовлення можна запускати в окремому потоці Python, що дозволяє паралельно виконувати комп'ютерний зір, детекцію об'єктів, OCR та озвучення без затримок у логіці програми.

Отже, в питаннях затримки pyttsx3 також демонструє найкращі показники затримки й сумісності для систем реального часу, що робить його ідеальним варіантом для вибору як технології генерації звукових повідомлень в досліджуваній системі.

2.5 Обґрунтування вибору апаратної платформи

На сьогоднішній день використання портативних компактних комп’ютерних систем є розповсюдженим явищем. Така тенденція з’явилася з появою на ринку подібних рішень, таких як NVIDIA Jetson Nano, Google Coral Dev Board, Odroid XU4, Raspberry Pi та інші.

NVIDIA Jetson Nano є гарним пристроєм для реалізації проєктів на основі технологій штучного інтелекту розроблений NVIDIA. Оснащений 4-ядерним процесором ARM Cortex-A57 та 128-ядерним GPU архітектури Maxwell, що дозволяє ефективно виконувати інференс нейронних мереж без потреби у зовнішньому прискорювачі [20]. Перевагою пристрою є наявність графічного процесору, що значно підвищує задачі обробки зображень та роботи штучного інтелекту при тому маючи невеликий розмір (69.6 × 45 мм). Але через наявність графічного процесору та необхідності додаткового охолодження робить рішення коштовним та складним для перенесення та використання у вуличних умовах.

Однопалатний комп’ютер Google Coral Dev Board також виконує актуальні задачі пов’язані з машинним навчанням. Підходить для створення прототипів вбудованих систем яким потрібен локальний інференс моделей глибокого навчання з мінімальними затримками та низьким енергоспоживанням. Основу пристрою становить System-on-Module (SoM), який включає в себе процесор NXP i.MX 8M (4 ядра Cortex-A53 та Cortex-M4F), 1 або 4 ГБ LPDDR4 RAM, 8 ГБ eMMC-пам’яті, а також модулі Wi-Fi 2x2 MIMO та Bluetooth 4.2. Головною особливістю є інтегрований Edge TPU — спеціалізований співпроцесор від Google, здатний виконувати до 4 трильйонів операцій на секунду (TOPS) при споживанні лише 2 Вт енергії. Це дозволяє запускати оптимізовані моделі TensorFlow Lite, такі як MobileNet v2, з продуктивністю до 400 кадрів на секунду, що робить його ідеальним для задач комп’ютерного зору в реальному часі [9]. Проте, Edge TPU підтримує лише моделі TensorFlow Lite. Базова версія має лише 1 ГБ

RAM, що може бути недостатньо для деяких складних задач; хоча доступна версія з 4 ГБ RAM, вона менш поширена.

Odroid XU4 – це потужний одноплатний комп'ютер від Hardkernel, який оснащений 8-ядерним процесором Samsung Exynos5422 з архітектурою big.LITTLE (4 ядра Cortex-A15 та 4 ядра Cortex-A7). Це дозволяє ефективно розподіляти навантаження між ядрами для оптимальної продуктивності та енергоспоживання. Має високу обчислювальну потужність, підтримку eMMC-накопичувачів для швидкого зберігання, має Gigabit Ethernet та USB 3.0. Проте складний в налаштуванні, має обмежену спільноту користувачів, меншу кількість доступних дистрибутивів [22].

Raspberry Pi 5 базується на чотириядерному 64-бітному процесорі Arm Cortex-A76 із тактовою частотою 2.4 ГГц, що забезпечує у 2–3 рази вищу продуктивність порівняно з попередником Raspberry Pi 4. Графічний процесор VideoCore VII підтримує виведення 4K HDR з частотою 60 кадрів за секунду на двох micro-HDMI портах.

Оперативна пам'ять представлена конфігураціями від 4 до 8 ГБ типу LPDDR4X-4267 SDRAM, що дозволяє ефективно обробляти відео та запускати моделі глибинного навчання без суттєвих затримок. Суттєвою інновацією Raspberry Pi 5 є чип RP1, розроблений безпосередньо Raspberry Pi Ltd. Він забезпечує пришвидшене введення/виведення даних, зокрема для USB, GPIO та дисплейних інтерфейсів [29].

Однією з ключових переваг Raspberry Pi 5 є його енергоефективність. Навіть при збільшеній продуктивності пристрій зберігає помірне енергоспоживання, що робить його придатним для використання в автономному режимі, з живленням від акумуляторів або портативних джерел енергії (таблиця 2.1).

Таблиця 2.1 – Переваги та недоліки апаратних платформ для реалізації модуля допомоги слабозорим на зупинках громадського транспорту

№	Рішення	Переваги	Недоліки
1	NVIDIA Jetson Nano 	Наявність графічного процесора призначеного до вирішення задач ML.	Висока ціна; Складність в перенесенні та вуличному використанні.
2	Google Coral Dev Board 	Наявність графічного процесора призначеного до вирішення задач машинного навчання.	Висока ціна; Мала варіативність в виборі моделі машинного навчання.
3	Odroid XU4 	Висока обчислювальна потужність; Підтримка eMMC-накопичувачів.	Обмежена спільнота користувачів; Невелика кількість доступних дистрибутивівв.
4	Raspberry Pi 5 	Велика спільнота користувачів; Універсальність; Велика кількість комплектацій; Можливість додавання плат розширення; Багато оперативної пам’яті.	Відсутність графічного процесора; Відсутність порту для навушників.

У поєднанні з активною спільнотою користувачів, великою кількістю документації та підтримкою популярних фреймворків для комп’ютерного зору й машинного навчання (OpenCV, PyTorch, TensorFlow), Raspberry Pi 5 забезпечує повноцінну програмно-апаратну екосистему [30].

Для вибору апаратної платформи можна зіставити порівняльну таблицю (Таблиця 2.1) та проаналізувати усі варіанти та вибрати найкращу апаратну платформу.

Отже, проаналізувавши Таблицю 2.1 та усе вищенаписане, можна прийти висновку, що Raspberry Pi 5 ідеально підійде для виконання цілей роботи: має відносно невелику ціну, велику кількість варіантів встановлення охолодження та корпусів, має багато портів для підключення периферії, дозволяє встановити будь яку ОС та модель машинного навчання. Даний краще сього підходить для реалізації апаратної частини модуля для допомоги слабозорим на зупинках громадського транспорту.

2.6 Протоколи та інтерфейси апаратної платформи на основі однопалатного комп’ютера Raspberry Pi 5

Оскільки основною апаратною платформою для реалізації проєкту обрано Raspberry Pi 5, то слід більш детально розглянути модулі даного однопалатного комп’ютера.

Raspberry Pi 5 має компактні розміри (85.6 × 56.5 мм) та працює на основі 64-бітного процесора Arm Cortex-A76, що дозволяє реалізовувати обчислювально-інтенсивні завдання, включаючи комп’ютерний зір та синтез мовлення.

На платі Raspberry Pi 5 передбачено кілька типів інтерфейсів для введення та виведення даних. До них належать два порти USB 3.0, два USB 2.0, два micro-HDMI (до 4K@60 Гц), інтерфейс PCIe 2.0 для зовнішніх накопичувачів, а також стандартний 40-контактний GPIO-роз’єм, сумісний із попередніми версіями Raspberry Pi. GPIO (General Purpose Input/Output)

використовується для підключення різноманітних датчиків і модулів та підтримує такі протоколи, як I²C, SPI та UART. Це надає гнучкість для майбутнього розширення системи або підключення додаткових сенсорів.

У межах проєкту планується використання USB-камери для захоплення відеопотоку. Підключення відбувається через порт USB 3.0, що забезпечує високу пропускну здатність для обробки зображень у реальному часі. USB-інтерфейс підтримується на апаратному рівні та має драйверну сумісність з більшістю камер, що працюють за стандартом UVC (USB Video Class).

Для виведення звуку буде застосовано зовнішню USB-звукову карту, до якої підключатимуться навушники. Raspberry Pi 5 не має окремого 3.5-мм аудіовиходу, тому для проєктів, які потребують аудіовиводу, рекомендовано використовувати зовнішні звукові рішення. USB-аудіоадаптери повністю підтримуються ОС Raspberry Pi OS без потреби у встановленні додаткових драйверів. Це дозволяє реалізувати локальне озвучення повідомлень про транспорт за допомогою бібліотеки pyttsx3 з мінімальною затримкою.

Живлення Raspberry Pi 5 здійснюється через USB-C порт із підтримкою стандарту PD (Power Delivery). Згідно з офіційною документацією, рекомендоване живлення має становити 5 В при струмі не менше 5 А (25 Вт). У зв’язку з цим, для мобільного використання буде застосовано павербанк із підтримкою 5 В / 5 А або PD-профілю 9 В / 3 А, що автоматично знижується до потрібного рівня. Надійне живлення критично важливе для стабільної роботи системи, особливо при підключенні декількох USB-пристроїв [29].

Схема фізичного з’єднання включає USB-камеру, підключену до одного з USB 3.0 портів, USB-звукову карту з навушниками – до USB 2.0, а також підключення живлення через порт USB-C від павербанка. Завдяки невеликим розмірам і наявності всіх необхідних інтерфейсів Raspberry Pi 5 не потребує додаткових контролерів або плат розширення, що спрощує конструкцію й забезпечує високу надійність у польових умовах.

Схему комп’ютера можна побачити на відповідній схемі (рисунок 2.5).

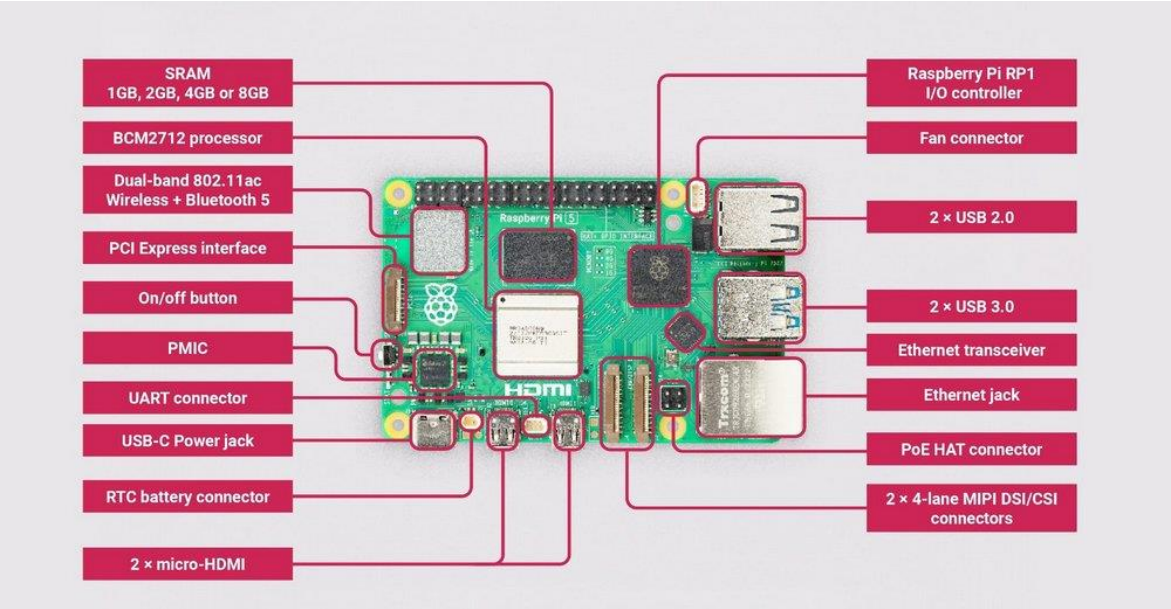

Рисунок 2.5 – Схема розташування елементів на платі Raspberry Pi 5

Отже, проаналізувавши всі можливі технічні підходи до вирішення задачі орієнтації слабозорих осіб на зупинках громадського транспорту, можна сформувати програмно-апаратний стек для узагальненого розуміння роботи програмно-апаратного модуля. Апаратною базою для рішення буде одноплатний комп’ютер Raspberry PI 5 до якого буде під’єднано камеру для зібрання зображень навколишнього середовища, навушники для виводу інформації, акумулятор для портативності та живлення. Програмною ж базою буде ОС PiOS без графічного інтерфейсу, мова програмування Python, модель YOLOv8n для розпізнавання транспорту на зображенні та Tesseract OCR для розпізнавання номерів маршрутів. Результат озвучує pyttsx3. Для зручного ознайомлення з рішенням представлено абстрактне схематичне рішення (рисунок 2.6).

Структура роботи рішення буде представляти цикл обробки зображень, де апаратна та програмна частини обмінюються відповідною інформацією.

Спочатку елементи периферії (конкретно камера) отримує зображення. Підключення до плати дозволяє передати зображення до програми, яка базується на основі операційної системи PiOS. Програма аналізує зображення та виводить результат на навушник, після чого камера знову починає

захоплення зображення і цикл повторюється.

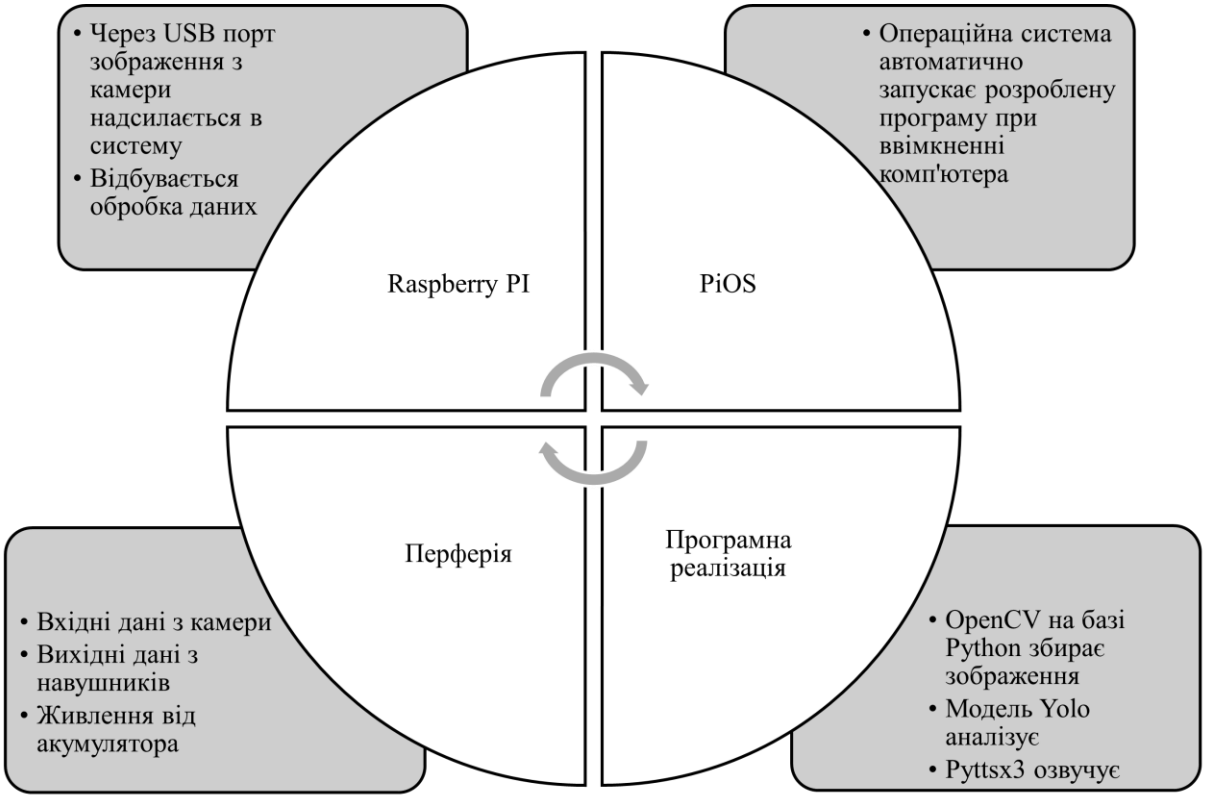

Рисунок 2.6 – Абстрактне уявлення циклу роботи програмно-апаратного рішення де зліва hardware елементи, а справа – software

Отже, наведена на рисунку 2.6 схема допоможе зорієнтуватися у подальшій розробці досліджуваного рішення та в узагальненні роботи рішення у наступному розділі.

3 ФОРМАЛЬНА ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ РІШЕННЯ

3.1 Загальна архітектура програмно-апаратного комплексу

Як було обумовлено в усіх попередніх розділах, рішення проблеми орієнтації слабозорих на зупинках громадського транспорту повинно складатися з двох підходів: апаратного та програмного. Архітектура рішення передбачає тісну інтеграцію апаратної частини (Raspberry Pi, камера, аудіоінтерфейс) та програмних модулів, які забезпечують обробку зображень, детекцію об’єктів, розпізнавання номерів маршрутів і голосове озвучення результатів користувачу.

Для формалізації логіки функціонування системи використано IDEF0-схему, яка наочно відображає структуру процесу та взаємодію між його складовими. Верхній рівень моделі демонструє основну функцію: виявлення та озвучення наближення громадського транспорту, яка реалізується через послідовну роботу кількох підсистем (рисунок 3.1).

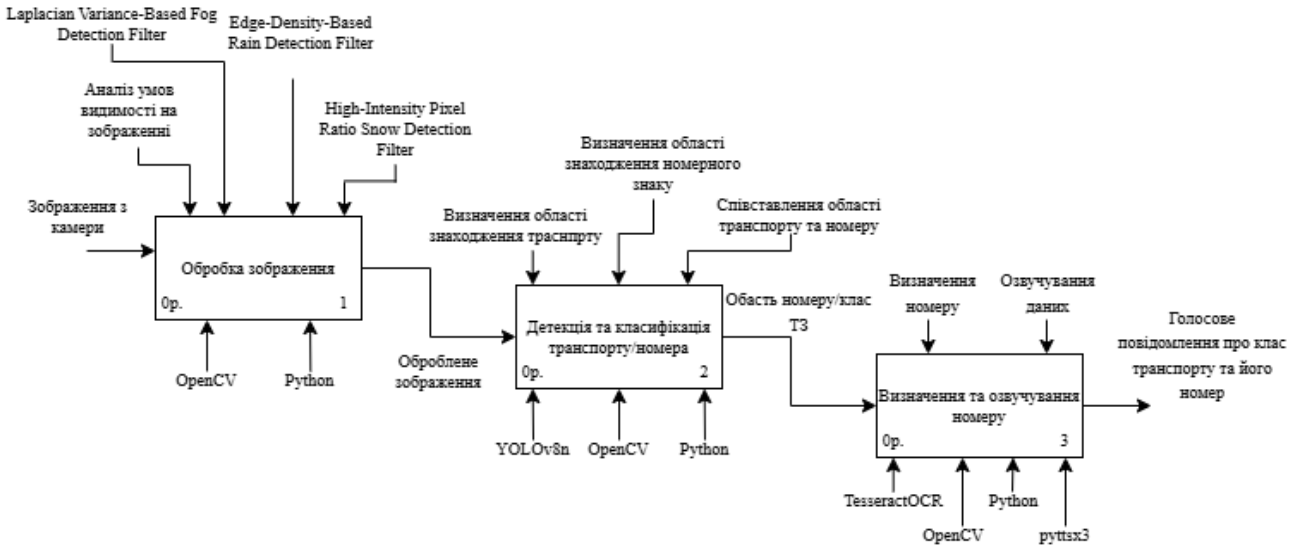

Рисунок 3.1 – IDEF0-схема загальної архітектури програмно-апаратного комплексу

На вхід системи подається відеопотік з камери, а також чинники зовнішнього середовища (освітлення, погодні умови). Ці дані обробляються блоком комп’ютерного зору (YOLOv8), який здійснює детекцію об’єктів транспорту. Далі блок обробки виділяє з кадру область, де очікується номер маршруту, і передає її в модуль оптичного розпізнавання символів (OCR, Tesseract). Якщо номер маршруту успішно розпізнано – система формує текстове повідомлення і передає його у модуль озвучення (pyttsx3), що генерує голосовий сигнал. У разі невдачі розпізнавання озвучується лише тип транспорту (наприклад, “автобус”).

Додатковим елементом архітектури є модуль фільтрації умов середовища, який застосовує попередню обробку кадрів (наприклад, покращення контрасту при поганому освітленні), що підвищує точність подальших етапів обробки.

3.2 Структурні схеми функціонування модулів

3.2.1 Алгоритм виявлення транспортного засобу

Для виявлення транспортних засобів було обрано модель YOLOv8n. Перед наданням моделі зображення слід врахувати спосіб прийому зображення та самого зображення та видимості об’єктів на ньому. Саме тому в модуль детекції транспортних засобів додано можливість попереднього аналізу погодних умов та присвоєння фільтрів в залежності від виявлених умов.

Таким чином, в модулі враховано набір сценаріїв можливих погодних умов: дощ, сніг, туман та темрява.

Тобто, при запуску модуля, перш за все відбувається аналіз середовища і потім починається цикл аналізу зображень. Більш детально зрозуміти процес роботи модуля виявлення транспортних засобів можна передивившись блок-схему алгоритму (рисунок 3.2).

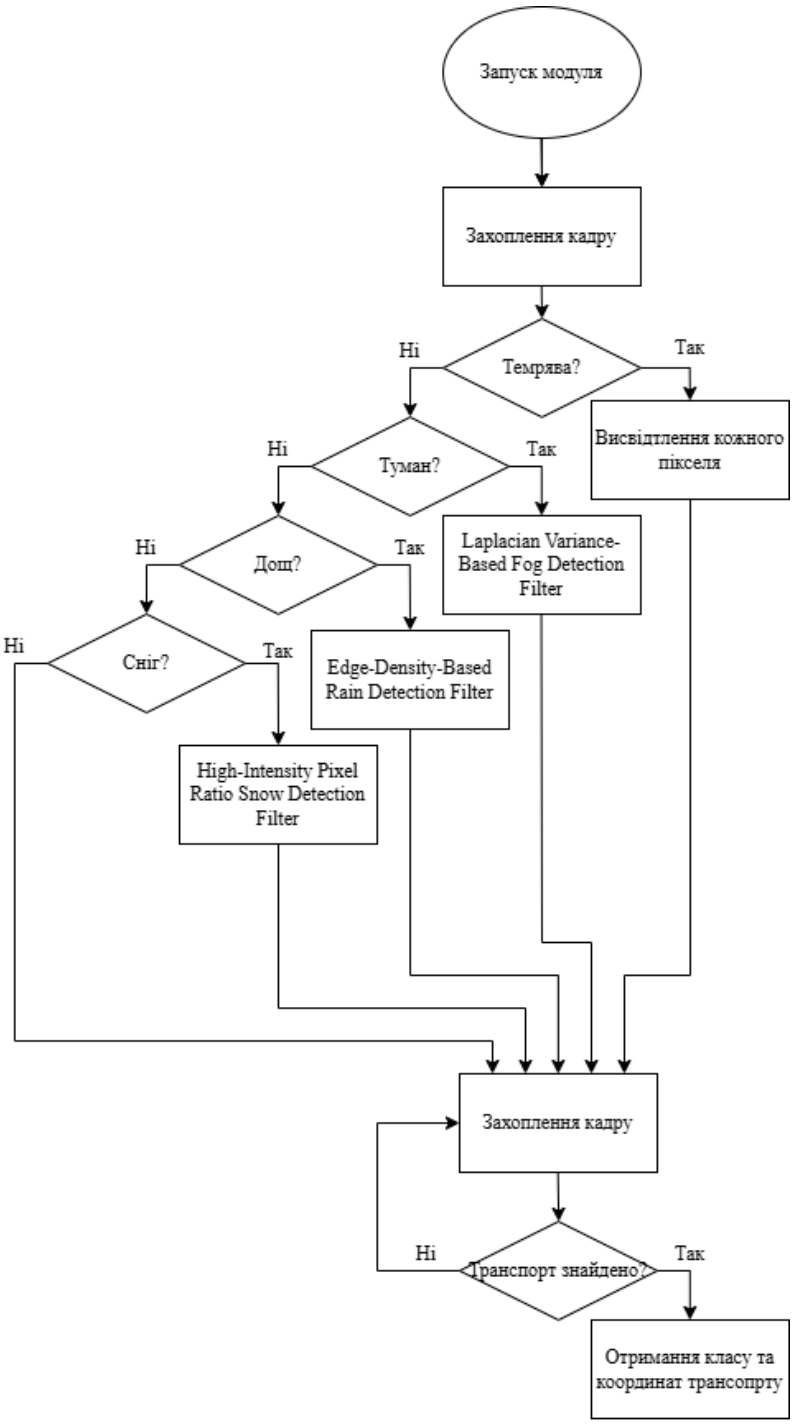

Рисунок 3.2 – Блок-схема роботи модуля виявлення транспорту

Отже, ознайомившись з блок-схемою на рисунку 3.2 можна зрозуміти алгоритм роботи модуля виявлення транспорту. Результатом його роботи є набір вихід з набором координат, а саме праву верхню точку та ліву нижню області знайденого транспорту та клас, які надає YOLOv8n. Далі ці данні координати області використовуються для визначення номеру виявленого транспорту.

3.2.2 Розпізнавання номера маршруту

Для реалізації модуля розпізнавання номеру маршруту громадського транспорту було обрано технологію оптичного розпізнавання символів TesseractOCR. У процесі проєктування архітектури цього модуля було встановлено, що передача повного кадру з відеопотоку безпосередньо до TesseractOCR є неефективною, оскільки це суттєво збільшує обчислювальні витрати та може знижувати точність розпізнавання через наявність зайвої інформації на зображенні.

Аналіз візуальних особливостей маршрутних номерів показав, що вони, як правило, розміщені на контрастному (чорному або білому) прямокутному фоні, часто з вбудованим підсвічуванням. Це робить їх візуально помітними для людини та дозволяє сформувати припущення, що модель об'єктного детектування YOLOv8 також здатна ефективно ідентифікувати такі зони при наявності якісної анотації під час навчання.

Відповідно, було прийнято рішення додатково навчити модель YOLOv8 не лише виявляти і класифікувати типи громадського транспорту, а й локалізувати зону, де розміщено номер маршруту.

До виділеної за допомогою YOLOv8 області застосовується функціонал OCR через Tesseract. Проте слід враховувати, що алгоритми Tesseract працюють оптимальніше при обробці чорно-білих або сірих зображень, оскільки кольорова інформація може вносити шум. У зв’язку з цим, перед передачею вхідного зображення на обробку, до виділеної області застосовуються попередні фільтри за допомогою бібліотеки OpenCV, які зменшують кількість кольорових компонентів та підвищують контрастність текстових символів.

Алгоритм роботи модуля розпізнавання номеру маршруту з урахуванням вищезазначених етапів подано у вигляді блок-схеми на рисунку 3.3.

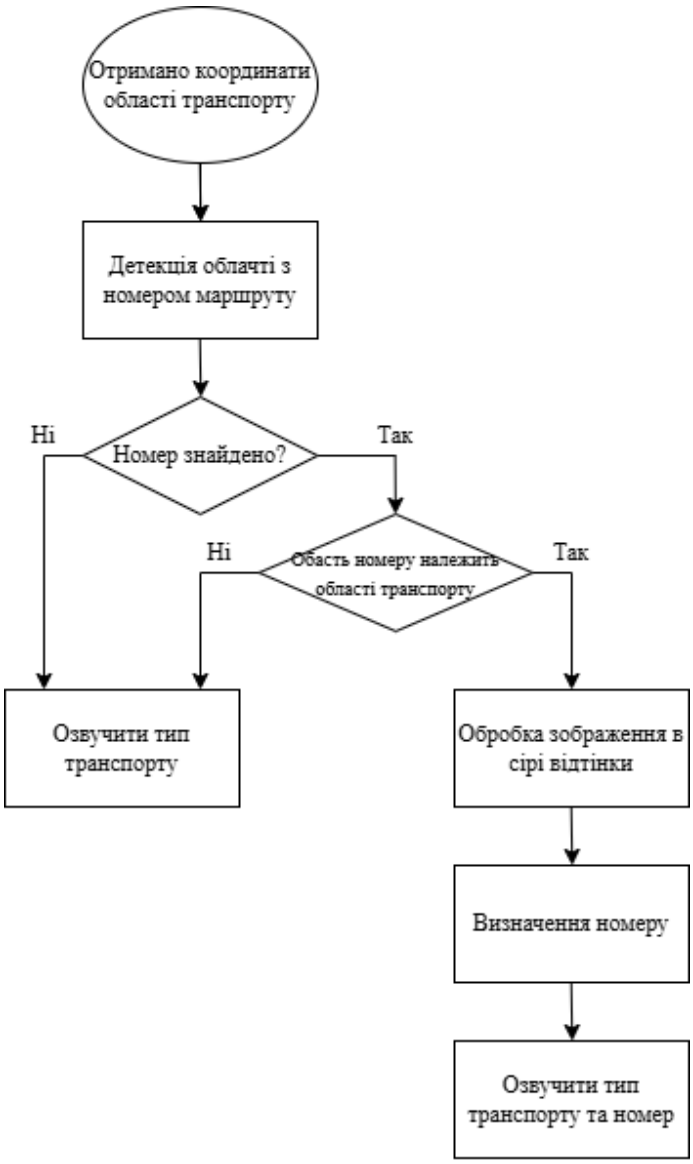

Рисунок 3.3 – Блок-схема роботи модуля виявлення номеру маршруту

Отже, як видно з рисунку 3.3, алгоритм роботи модуля виявлення номеру маршруту передбачає виконання спроби розпізнавання текстової інформації (номеру маршруту) лише у випадку, якщо область з номером розташована всередині виявленої області громадського транспорту. Такий підхід дозволяє уникнути хибного розпізнавання сторонніх символів або об'єктів, що можуть бути присутніми на зображенні поза межами транспортного засобу.

Таким чином, модуль забезпечує можливість ідентифікації типу транспорту (наприклад, автобус, трамвай або тролейбус) незалежно від наявності чи якості зображення маршрутного номеру, що є особливо

важливим у випадках обмеженої видимості, руху транспорту або часткового перекриття номера.

Такий підхід підвищує надійність системи та дозволяє формувати повідомлення про наближення відповідного класу громадського транспорту навіть у тих ситуаціях, коли номер маршруту не може бути розпізнаний.

3.2.3 Озвучення результату

Механізм озвучування результатів розпізнавання передбачає три основні сценарії взаємодії з користувачем:

- відсутність озвучування у разі, якщо об’єкти громадського транспорту не були виявлені;
- озвучування лише типу (класу) транспорту, що наближається, без зазначення маршрутного номеру;
- озвучування як типу транспорту, так і його маршрутного номеру, у випадку успішного розпізнавання обох параметрів.

З метою забезпечення максимальної ефективності роботи системи в реальному часі, функціонал озвучування реалізовано в асинхронному режимі. Це означає, що виведення голосового повідомлення відбувається у фоновому (окремому) потоці, не блокуючи основний цикл обробки кадрів. Такий підхід дозволяє уникнути затримок у роботі системи та забезпечити її безперервне функціонування незалежно від тривалості озвучування.

Оскільки в описі модуля визначення номеру маршруту, а саме в блок-схемі на рисунку 3.3 вже наведено сценарії озвучування, а озвучування результату детекції об’єктів є основним вихідним показником рішення, то для даного підрозділу є доцільним навести загальну блок-схему та відобразити весь алгоритм для озвучування результату (рисунок 3.4).

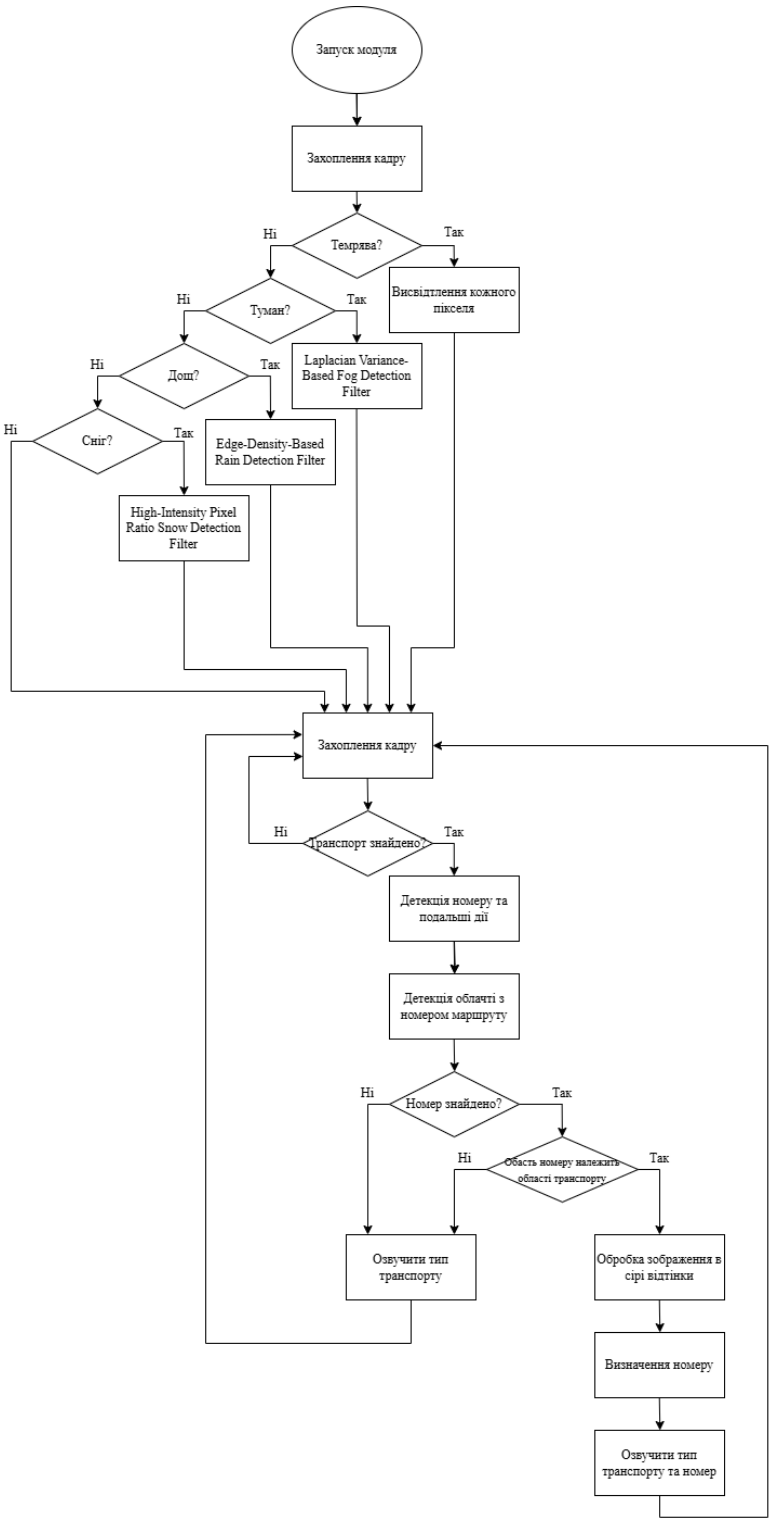

Рисунок 3.4 – Загальна блок-схема роботи рішення

Виходячи з рисунку 3.4 можна побачити загальне функціонування програмно-апаратного модуля для допомого слабозорим з орієнтування на зупинках громадського транспорту.

3.3 Програмна реалізація модулів

3.3.1 Детекція транспорту: YOLOv8n

Для навчання моделі YOLOv8n необхідно виконати декілька дій: підготувати датасет у файловій системі, конфігураційний файл, написати невеликий модуль для тренування та задіяти модель в основному модулі.

Щоб правильно розподілити файли датасету у файловій системі необхідно помістити їх в спільну директорію розподілити на зображення датасету в директорію images/train та на файли розмітки в labels/train. При розробці рішення загальну директорію було вирішено назвати “data”. Графічне відображення файлової структури можна побачити на рисунку 3.5.

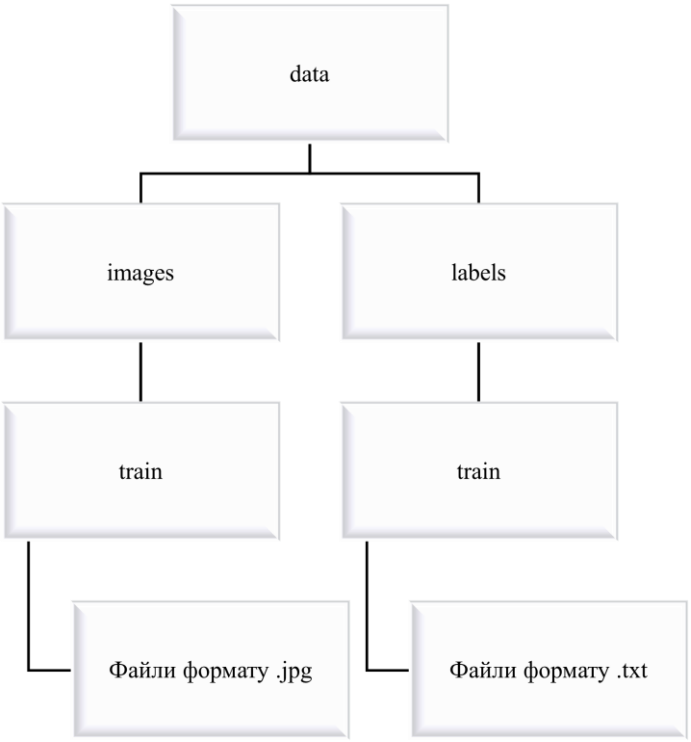

Рисунок 3.5 – Файлова структура файлів датасету

Таке розподілення файлів є необхідною умовою для навчання моделей YOLO, при якому використовується конфігураційний файл, де і указуються шлях до матеріалів датасету.

Конфігураційний файл мусить мати розширення .yaml, мати шляхи до зображень та розмітки датасету, та містити опис класів: ідентифікатор та назву (ідентифікатор має відповідати відповідним класам в файлах розмітки). Конкретно для досліджуваного рішення було створено конфігураційний файл з шляхами до правильно розподілених до цього файлів датасету, та опис класів та їх ідентифікаторів: 0 – номер маршруту; 1 – автобус; 2 – трамвай; 3 – тролейбус (лістинг 3.1).

Лістинг 3.1 – Зміст конфігураційного файлу conf.yaml

```
path: I:\PTDM\PTDM\data
train: images/train
val: images/train

names:
  0: number
  1: bus
  2: tram
  3: trolleybus
```

Після підготовки конфігураційного файлу його можна використовувати для навчання моделі. Навчання ж моделі можна проводити як через консоль, так і написавши скрипт. В даній роботі було обрано другий спосіб через особливість роботи бібліотеки ultralytics.

Для запуску навчання достатньо встановити та імпортувати бібліотеку ultralytics, обрати модель та визвати команду для навчання (лістинг 3.2).

Лістинг 3.2 – Найпростіша реалізація навчання моделі

```
from ultralytics import YOLO

def main():
    model = YOLO("yolov8n.yaml")
    result = model.train(data="config.yaml", epochs=200)

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

Але така реалізація має значні недоліки. Наприклад, при перших

спробах навчання моделі в рамках даної роботи, перший тестовий датасет мав приблизно 300 зображень та навчання запускалося в 200 епох. Приблизний час обробки всіх зображень був близько 1 хвилини. З такою продуктивністю обробка всіх зображень на процесорі Intel Core I7-1165G7 займала 3 години. З цього виходить, що для підвищення ефективності рішення збільшенням зображень в датасеті буде підвищуватися кількість часу навчання моделі. Це твердження було перевірено з фінальним сформованим датасетом на 700 зображень. Навчання моделі таким датасетом з використанням ресурсів ЦП зайняло би приблизно 30 годин.

Отже, було прийнято рішення знайти більш ефективний спосіб навчання моделі. І найкращим рішенням було обрано навчання моделі з використанням ресурсів графічного процесору. Відомо, що ГП краще працює з великою кількістю даних, з обробкою зображень та паралельними розрахунками. Для задіяння ГП в процесі навчання необхідно було встановити CUDA на систему на якій проводилося навчання моделі.

CUDA (Compute Unified Device Architecture) – це паралельна обчислювальна платформа та програмна модель, розроблена компанією NVIDIA, яка дозволяє використовувати графічні процесори для загального (не графічного) програмування. Як вже було зазначено, CUDA надає розробникам можливість використовувати потужність GPU для прискорення обчислювально інтенсивних задач, таких як обробка зображень, машинне навчання, фізичне моделювання тощо.

Для застосування CUDA в Python-скрипті необхідно імпортувати бібліотеку torch. Torch (PyTorch) – це відкрита бібліотека для глибокого навчання (deep learning), яка надає гнучке та зручне середовище для побудови, навчання і тестування нейронних мереж.

Таким чином код реалізації навчання моделі з використанням CUDA дещо зміниться: буде додано рядки для перевірки підключення CUDA та наявності ГП, та буде додано параметр ідентифікатора пристрою в команду train (лістинг 3.3).

Лістинг 3.3 – Фінальна реалізація навчання моделі

```
from ultralytics import YOLO
import torch

def main():
    print(torch.cuda.is_available())
    print(torch.cuda.get_device_name(0))

    model = YOLO("yolov8n.yaml")
    result = model.train(data="config.yaml", epochs=200,
device=0)

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

Із застосуванням CUDA ефективність навчання значно зросла. Проходження 200 епох з 700 зображеннями датасету зайняло 30 хвилин.

Під час навчання можна передивлятися результати навчання за кожну епоху. А після проходження всіх епох формується загальний звіт, на якому зазначено кількість зображень, кількість знайдених об’єктів та якість класифікації mAP50 (рисунок 3.6).

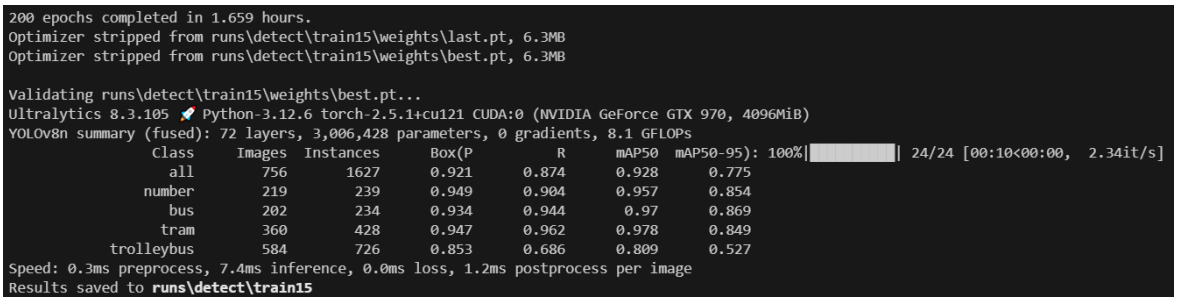

Рисунок 3.6 – Результат навчання фінальної моделі

Окрім наведеного вище звіту формується директорія runs\detect\train(n), де n – номер тренування за датасетом. В цій дерикторії і формується два варіанти моделі: остання та найкраща. Звісно для подальшого використання береться найкраща модель.

Також, результативна директорія містить різні аналітичні графічні дані. Наприклад, графіки помилок, загублених об’єктів та успішності моделі (рисунок 3.7).

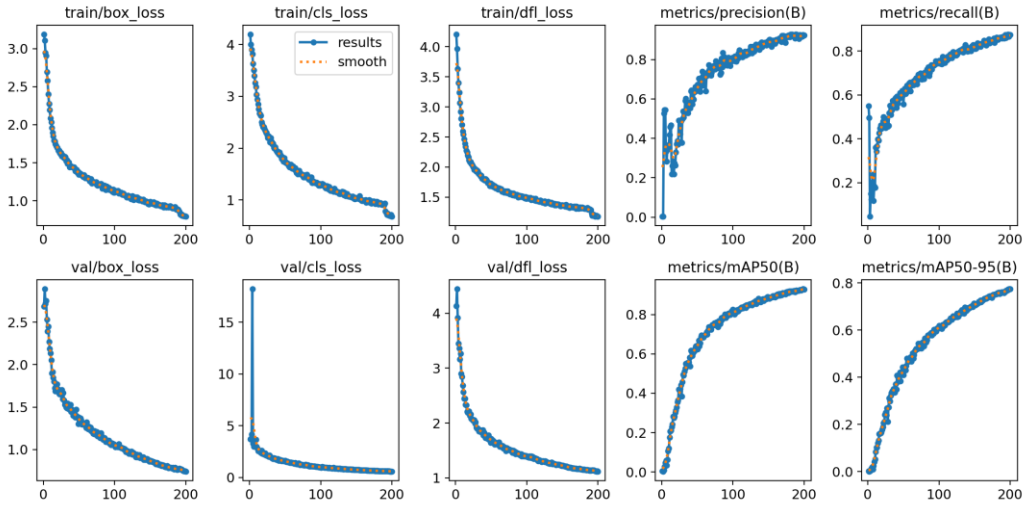

Рисунок 3.7 – Графіки сформовані після останнього тренування моделі

Отже, взявши найкращу модель після тренування за шляхом runs\detect\train15\weights\best.pt її можна використовувати для подальшого аналізу зображень.

В рамках програмної реалізації було вирішено притримуватися принципів ООП, тому модель використовується в класі “ObjectDetector” (лістинг 3.4).

Лістинг 3.4 – Реалізація класу ObjectDetector

```
from ultralytics import YOLO

class ObjectDetector:
    def __init__(self, model_path):
        self.model = YOLO(model_path)

    def detect(self, frame):
        results = self.model(frame)
        return results
```

Клас приймає як параметр шлях до моделі, після чого в методі detect передати кадр моделі.

Саме для аналізу зображень створено інший клас – Analizer. В ньому створюється ObjectDetector. Метод analize запускає процес аналізу зображення, і саме тут задіяна модель YOLO (лістинг 3.5).

Лістинг 3.5 – Фрагмент методу analize, де використовується модель

```
def analize(self, snapshot):
    if snapshot is not None:
        results = self.detector.detect(snapshot)

        transports = []
        numbers = []

        for result in results:
            for box in result.boxes.data:
                x1, y1, x2, y2, conf, cls = box.tolist()
```

Тут важливо звернути увагу на те, які данні повертає модель в results. Взагалі, results – це об'єкт Results, який має:

1. results.boxes – прямокутники (bounding boxes) для знайдених об'єктів:

- .хуху – координати боксів у форматі (x1, y1, x2, y2);
- .conf – впевненість (confidence score);
- .cls – клас об'єкта (номер класу);
- .id – ID об’єкта (якщо відстеження).

2. results.names – словник відповідностей між класами (числами) і їх назвами;

3. results.plot() – візуалізація (малює бокс на зображенні).

В даному випадку використовується лише results.boxes.

Перш за все аналізується показник впевненості. У ході тестувань було зроблено висновок, що показник впевненості 0.65 (65%) є достатнім для того, щоб модель знаходила об’єкти та не плутала їх. Якщо знайдений об’єкт відповідає умові, то він додається в відповідний список транспортів (transports) чи номерів (numbers). Розподілення у вірний список відбувається завдяки показнику cls. Для даного рішення, як вже було сказано, існує 4 класи. Клас 3 – це номери, всі інші – транспорт. Таким чином за номером класу можна визначати умову, для віднесення об’єкту до списку номерів чи транспортів (лістинг 3.6).

Лістинг 3.6 – Фрагмент з відбором по впевненості та призначення в списки

```
…
class_id = int(cls)
box_data = (int(x1),int(y1),int(x2),int(y2))

if class_id in [0, 1, 2]:
    transports.append((class_id, box_data))
elif class_id == 3:
    numbers.append(box_data)
…
```

Такий розподіл по спискам необхідне для того, щоб визначити, до якого транспорту належить номер. Наприклад, якщо на зображенні є автобус №102 та тролейбус №3 модель починає детекцію об’єктів. Оскільки номери менші, то з більшою вірогідністю спочатку буде знайдено транспорти. Припустимо, що першим було знайдено тролейбус, за ним автобус, а вже потім спочатку номер “102” і “3” після нього. Якщо не визначати де знаходиться номер, то в результаті було б озвучено: “Тролейбус номер 102 наближається. Автобус номер 3 наближається”. Щоб запобігти плутанини, було запроваджено механізм перевірки належності номера до транспорту (лістинг 3.7).

Лістинг 3.7 – Механізм визначення належності номера до транспорту

```
for class_id, (tx1, ty1, tx2, ty2) in transports:
     for nx1, ny1, nx2, ny2 in numbers:
          if nx1>tx1 and ny1>ty1 and nx2<tx2 and ny2<ty2:
               …
```

Беруться координати транспорту та номеру відбувається перевірка: якщо координати лівого верхнього кута більші по значенням за координати відповідного кута транспорту та координати правого нижнього кута номера менші за відповідні координати транспорту, то можна зробити припущення, що розглянутий номер належить певному транспорту. Такі умови дозволять також уникнути озвучування номерів, які випадково могли б потрапити у кадр та бути схожими на номери маршрутів.

3.3.2 OCR-модуль: Tesseract

Для розпізнавання модуля в системі повинен бути встановлений tesseract-ocr. Слід зазначити, що при використанні технології на системах Linux можна не вказувати шлях до виконавчого файлу.

Щоб використовувати функції Tesseract в python-застосунках, необхідно інсталювати та імпортувати бібліотеку pytesseract.

Оскільки функції OCR необхідні тільки при виконанні умови, що номер знаходиться на транспорті, було вирішено не розширяти архітектуру програмного рішення та не реалізовувати OCR-модуль як окремий об’єкт. Область де відбувається детекція символів визначається моделлю, яка спирається на розмітку датасету. Про це детальніше буде описано в розділі 3.4. Усі необхідні підготовки та виклик функцій OCR відбуваються в класі Analizer одразу після перевірки умов на знаходження номеру в рамках транспорту (лістинг 3.8).

Лістинг 3.8 – Опис роботи OCR-модуля

```
…
cropped = snapshot[int(ny1):int(ny2), int(nx1):int(nx2)]
gray = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
text = pytesseract.image_to_string(
     gray, config='--psm 7 -c
tessedit_char_whitelist=0123456789'
)
number = ''.join(filter(str.isdigit, text))
if number:
     label = class_names.get(class_id, "unknown")
     return f"{label} {number} is coming"
```

Першим кроком є виділення області інтересу, координати якої відомі завдяки роботі моделі.

Наступним етапом є перетворення обрізаного зображення у градації сірого, це необхідно для зменшення розмірності даних та полегшення процесу розпізнавання, оскільки більшість алгоритмів OCR працюють краще на монохромних зображеннях.

Далі здійснюється розпізнавання тексту за допомогою бібліотеки pytesseract. Тут використовується конфігурація --psm 7, яка вказує Tesseract, що на зображенні міститься лише один рядок тексту. Крім того, параметр tessedit_char_whitelist=0123456789 обмежує розпізнавані символи лише цифрами, що підвищує точність для задач числового розпізнавання.

Після розпізнавання тексту здійснюється фільтрація символів, щоб вилучити лише цифри. Якщо після фільтрації було знайдено хоч одне число (if number:), відбувається ідентифікація класу об'єкта через словник class_names за його ідентифікатором class_id. У випадку відсутності відповідного значення використовується значення “unknown”.

Таким чином дуже швидко і просто можна зчитати значення символів та перевести їх у текстовий вигляд для подальшого озвучування.

3.3.3 Модуль озвучення

Операційна система Linux надає користувачеві високу гнучкість у налаштуванні голосових технологій, однак для забезпечення можливості озвучення тексту необхідно встановити відповідне програмне забезпечення. Одним із найпоширеніших рішень для перетворення тексту у мовлення є бібліотека pyttsx3, яка є платформонезалежною та підтримує декілька бекендів синтезу мовлення. У випадку з Linux, ця бібліотека за замовчуванням використовує синтезатор eSpeak.

eSpeak – це програмне забезпечення для синтезу мовлення з відкритим кодом, яке забезпечує генерацію голосу на основі тексту. Воно працює в термінальному режимі та підтримується більшістю голосових бібліотек, таких як pyttsx3. Без його встановлення озвучування не буде можливим, оскільки саме eSpeak виступає низькорівневим синтезатором, який виконує фактичне перетворення тексту у звук.

Окрім того, для відтворення згенерованого голосу в системі необхідний доступ до аудіопристроїв. У Linux для цього широко використовується пакет

alsa-utils, до складу якого входить утиліта aplay. Саме вона відповідає за відтворення .wav-файлів та інших аудіо у середовищі ALSA (Advanced Linux Sound Architecture). Якщо alsa-utils не встановлений, система не зможе фізично вивести звук на колонки або навушники, що зробить роботу озвучування неефективною, навіть якщо синтезатор працює правильно.

Таким чином, встановлення eSpeak і alsa-utils є критично необхідною умовою для реалізації функціональності голосового озвучування тексту у Linux-системах.

Нижче наведено приклад Python-коду, що реалізує систему озвучування тексту з використанням бібліотеки pyttsx3 у асинхронному режимі (лістинг 3.9).

Лістинг 3.9 – Реалізація модуля озвучування

```
import pyttsx3
import threading
import queue

class Speaker:
    def __init__(self):
        self.engine = pyttsx3.init()
        self.queue = queue.Queue()
        self.thread = threading.Thread(target=self._run,
daemon=True)
        self.thread.start()

    def say(self, text):
        self.queue.put(text)

    def _run(self):
        while True:
            text = self.queue.get()
            if text is None:
                break
            self.engine.say(text)
            self.engine.runAndWait()
```

Даний модуль реалізує клас Speaker, який інкапсулює у собі механізм асинхронного озвучування текстових повідомлень. У конструкторі (__init__) ініціалізується голосовий рушій pyttsx3 та створюється черга повідомлень queue.Queue(), яка дозволяє зберігати кілька текстів, що очікують на

озвучення. Для обробки цієї черги створюється фоновий потік threading.Thread, який працює у фоновому (демонічному) режимі.

Метод say(self, text) дозволяє іншим частинам програми поставити текст у чергу на озвучення. Завдяки черзі, текстові повідомлення можуть надходити з різних потоків або подій без ризику конфлікту.

Метод _run(self) є основною функцією потоку. У безкінечному циклі він очікує надходження тексту з черги (self.queue.get()) і викликає функції engine.say(text) для генерації мовлення, а також engine.runAndWait(), яка блокує потік до завершення озвучування. У разі, якщо в чергу передано значення None, цикл завершується.

Таким чином, реалізація забезпечує немиттєве, але послідовне озвучення текстових повідомлень без блокування основного потоку виконання програми, що особливо корисно у мультимедійних або інтерактивних застосунках.

Виклик методу say відбувається в головному виконавчому файлі main.py після того, яка Analizer повертає результат (лістинг 3.10).

Лістинг 3.10 – Сумісна робота Speaker та Analizer

```
result = analyzer.analize(snapshot)
if result:
     speaker.say(result)
```

Таким чином досягається проста та ефективна взаємодія між модулями для досягнення швидкого та якісного озвучування.

3.3.4 Модуль роботи з камерою та OpenCV

Запропонований клас FrameGrabber реалізує обгортку над функціональністю cv2.VideoCapture із логікою, що дозволяє зчитувати відео з камери або відеофайлу, зберігаючи повні кадри з заданим інтервалом у секундах (лістинг 3.11).

Лістинг 3.11 – Ініціалізація FrameGrabber

```
import cv2
import time

class FrameGrabber:
    def __init__(self, source, interval_sec=1):
        self.cap = cv2.VideoCapture(source)
        self.interval = interval_sec
        self.last_time = time.time()
        self.fps = self.cap.get(cv2.CAP_PROP_FPS)
```

У конструкторі класу ініціалізується відеопотік за допомогою cv2.VideoCapture(source), де source – це або шлях до відеофайлу, або індекс вебкамери (наприклад, 0 для першої камери системи). Також зберігається інтервал у секундах між зніманням кадрів (interval_sec) та час останнього знімання (last_time). Окрім того, зчитується FPS (кадри за секунду) потоку для адаптації обробки.

Метод read() виконує основну функцію – зчитування чергового кадру з потоку. Якщо кадр зчитано успішно (ret == True), виконується перевірка, чи пройшов заданий інтервал часу з моменту останнього знімання. Якщо умова виконується, створюється копія кадру (snapshot = frame.copy()), яка повертається як знімок (лістинг 3.12).

Лістинг 3.12 – Реалізація методу read

```
def read(self):
        ret, frame = self.cap.read()
        if not ret:
            return None, None

        snapshot = None
        current_time = time.time()
        if current_time - self.last_time >= self.interval:
            self.last_time = current_time
            snapshot = frame.copy()

        return frame, snapshot
```

Таким чином, кожен виклик read() повертає:

- frame – поточний кадр із відеопотоку;

- snapshot – кадр-копію, якщо пройшов інтервал часу; інакше – None.

Це дозволяє програмі постійно працювати з відео, але відбирати знімки лише раз на N секунд, що суттєво знижує навантаження на систему зберігання або обчислення.

Метод get_fps() повертає кількість кадрів за секунду, яку повідомляє сам відеофайл або камера. Це може бути використано для логування або динамічного налаштування інтервалів обробки (лістинг 3.13).

Лістинг 3.13 – Реалізація методу get_fps

```
def get_fps(self):
     return self.fps
```

Метод release() закриває відеопотік та звільняє ресурси камери. У випадку неперервної роботи з вебкамерою це критично важливо для коректного завершення сесії та запобігання блокування пристрою (лістинг 3.14).

Лістинг 3.14 – Реалізація методу release

```
def release(self):
     self.cap.release()
```

Працює FrameGrabber в головному циклі виконавчого файлу main.py (лістинг 3.15).

Лістинг 3.15 – Використання FrameGrabber в main.py

```
while True:
     start = time.time()

     frame, snapshot = frame_grabber.read()
          if frame is None:
               break
```

Таким чином налагоджено зручний модуль для захоплення файлів, який одночасно може демонструвати те, що бачить та брати кадри в фіксований час для зручного відлагодження.

3.4 Особливості розмітки та підготовки датасету

3.4.1 Збір даних

Процес розмітки датасету є ключовим етапом при підготовці даних для задач комп’ютерного зору, таких як виявлення об’єктів. У випадку із громадським транспортом – зокрема автобусами, тролейбусами та трамваями – розмітка виявляється достатньо прямолінійною, але при цьому вимагає уважності та чіткого дотримання обраних критеріїв класифікації.

Наприклад, автобуси зазвичай мають масивну прямокутну форму з характерними елементами, такими як двері та вікна у два ряди (рисунок 3.8).

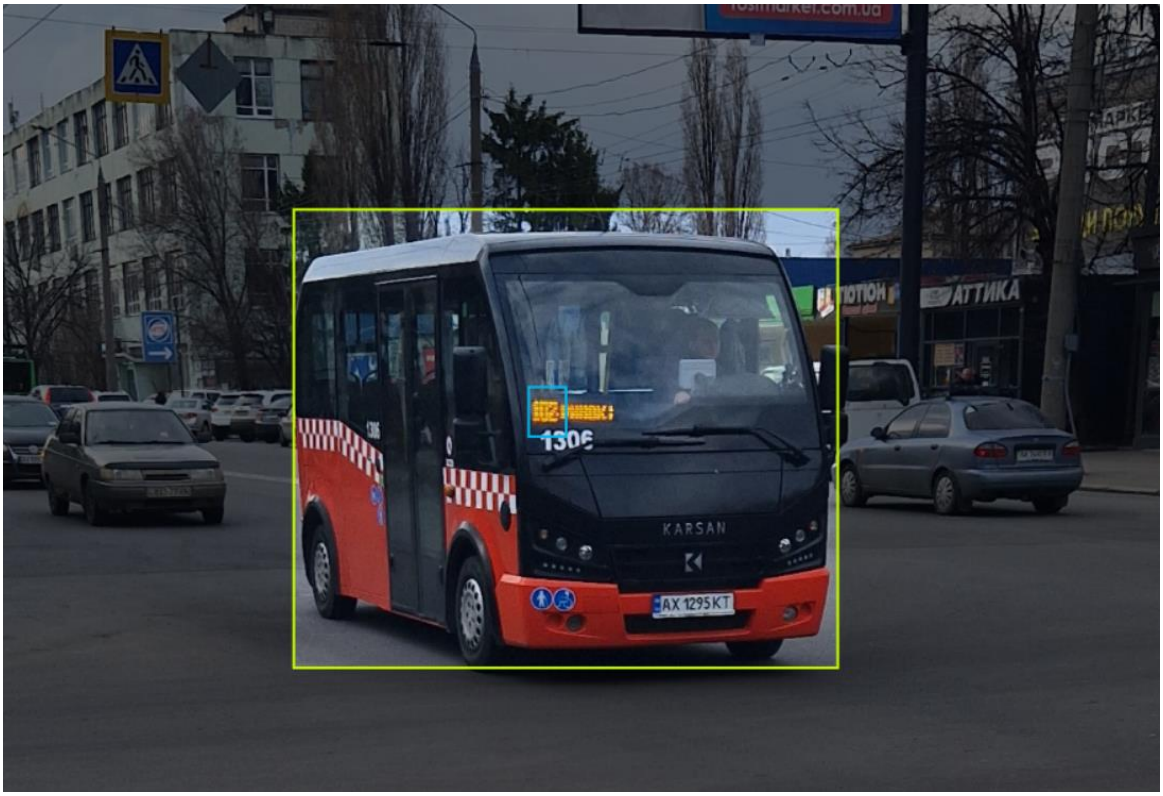

Рисунок 3.8 – Приклад розмітки для автобуса

Тролейбуси подібні до автобусів за формою, однак мають характерні пантографи (токоприймачі) на даху, що з’єднують їх з контактною мережею (рисунок 3.9).

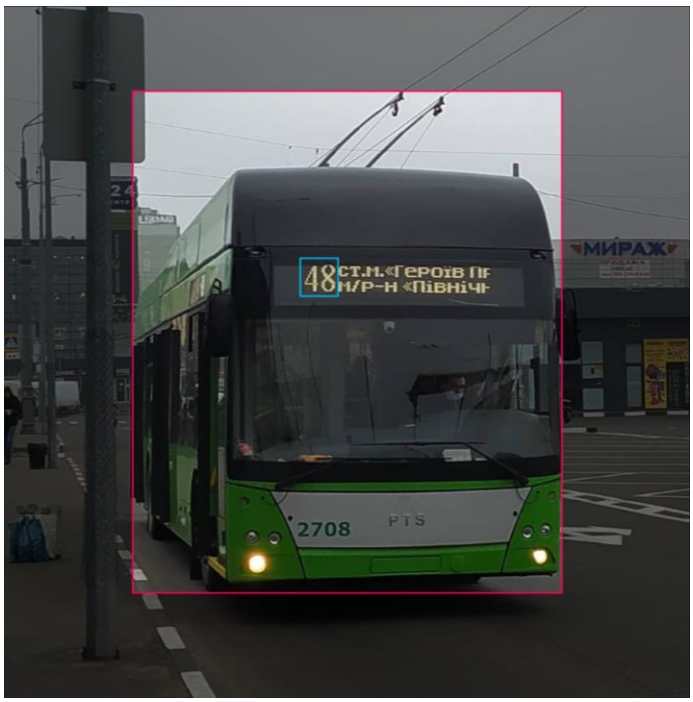

Рисунок 3.9 – Приклад розмітки для тролейбуса

Трамваї, у свою чергу, найчастіше рухаються по рейках, мають видовжену форму і характерні стики між секціями (рисунок 3.10).

Рисунок 3.10 – Приклад розмітки для трамвая

Для розмітки зображень було обрано сервіс Roboflow. Він має весь необхідний інструментарій для розмітки та зручний спосіб експорту зображень з розміткою для різних моделей.

Також варто мати в датасеті зображення, які містять по декілька транспортних засобів. Такі випадки часто можуть виникати на зупинках громадського транспорту, транспорт може перекривати один одного та ін. Якщо транспорт перекрито, то його необхідно виділяти з приблизним продовженням (рисунок 3.11).

Рисунок 3.11 – Приклад розмітки з декількома об’єктами які перекривають один одого

Розмітка номерів відбувається виділенням зони, де відповідно є номер. На попередніх зображеннях можна побачити принцип розмітки.

Матеріали для датасету були взяті з двох джерел (756 зображень): відкриті датасети з сайту Roboflow (326 зображень) та з сторінки в соцмережі Instagram “gor_transport” з зображеннями транспорту міста Харова (430 зображень). На зображеннях анотовано 239 автобуси, 234 трамвая, 428 тролейбусів та 726 номерів.

3.4.2 Синтетичне розширення

У контексті навчання моделей глибокого навчання для розпізнавання об'єктів, зокрема транспортних засобів на зображеннях, якість та різноманітність вхідних даних відіграють вирішальну роль. Оскільки моделі машинного навчання чутливі до умов, у яких було знято зображення, одноманітний датасет може призвести до переобучення (overfitting) – коли модель добре працює на навчальних прикладах, але втрачає точність при застосуванні до нових, реальних зображень.

Аугментація зображень – це метод штучного розширення датасету шляхом застосування до зображень різноманітних трансформацій: зміна освітлення, поворотів, шумів, погодних умов тощо. Це дозволяє моделі «побачити» більше варіантів одного і того ж класу об'єкта, підвищити узагальнюючу здатність, і, відповідно, – робастність моделі до нових середовищ.

У роботі було обрано кілька специфічних типів аугментацій, які імітують типові умови функціонування систем відеоспостереження в міському середовищі:

- імітація дощу;
- імітація снігу;
- ефект нічної зйомки;
- імітація руху та розмиття.

Обрані трансформації не є випадковими: вони моделюють реальні виклики, з якими стикається система комп'ютерного зору у міському середовищі. Основна мета – не просто збільшити обсяг даних, а покращити здатність моделі до генералізації в умовах, які не представлені у початковому наборі зображень.

Крім того, ці трансформації не змінюють геометрію об’єктів на зображенні (на відміну, скажімо, від поворотів чи масштабування), що дозволяє зберегти валідність координат bounding box-ів, а отже – не потребує

повторної розмітки або адаптації розміток.

Для покращення узагальнюючої здатності моделі було реалізовано синтетичне розширення зображень. Це дозволяє моделі навчитися розпізнавати об'єкти громадського транспорту в різних погодних умовах та при недостатньому освітленні. Нижче наведено фрагмент скрипту, що виконує аугментацію з використанням бібліотеки Kornia.

Використовується модуль kornia.augmentation, що забезпечує ефективні перетворення у форматі тензорів PyTorch, з можливістю обробки на GPU. Кожна трансформація застосовується до зображення з ймовірністю 100 % (параметр p=1.0), що дозволяє точно контролювати вихід (лістинг 3.16).

Лістинг 3.16 – Визначення погодних аугментацій

```
weather_transforms = [
    ("rain", K.RandomRain(p=1.0)),
    ("snow", K.RandomSnow(p=1.0)),
    ("night", K.ColorJitter(brightness=(0.1, 0.3),
contrast=(0.5, 0.8), p=1.0)),
    ("blur", K.RandomMotionBlur(kernel_size=(7, 11), angle=(-45,
45), direction=(-1.0, 1.0), p=1.0)),
]
```

Далі зображення, представлене у вигляді тензора, піддається вибраній аугментації. Після цього результат перетворюється назад у масив NumPy та масштабуються значення пікселів до діапазону [0, 255], щоб забезпечити коректне збереження у форматі .jpg. Функція cv2.imwrite записує зображення на диск (лістинг 3.17).

Лістинг 3.17 – Обробка та збереження зображень

```
augmented = transform(img_tensor)
augmented = augmented.squeeze(0).permute(1, 2, 0).cpu().numpy()
augmented = (augmented * 255).astype('uint8')
cv2.imwrite(out_img_path, cv2.cvtColor(augmented,
cv2.COLOR_RGB2BGR))
```

Далі відбувається збереження відповідності між зображенням та його розміткою у форматі YOLO. Оскільки аугментації не впливають на

геометричне розташування об’єктів, розмітка може бути перенесена без змін, що спрощує процес аугментації та зберігає валідність навчального набору (лістинг 3.18).

Лістинг 3.18 – Копіювання відповідних розміток

```
new_label_path = os.path.join(OUT_LABEL_DIR,
new_name.rsplit('.', 1)[0] + ".txt")
shutil.copy(label_path, new_label_path)
```

Після модифікації всіх зображень маємо наступні приклади (рисунок 3.12).

а)

б)

в)

г)

Рисунок 3.12 – Умови для аугментації датасету: а) дощ; б) сніг; в) затемення; г) розмиття

Після створення нових модифікованих зображень, модель необхідно донавчити. Це робиться аналогічно до процесу навчання з нуля: лише змінюється шлях до датасету та для навчання вказується вже готова модель (лістинг 3.19).

Лістинг 3.19 – Скрипт для донавчання моделі

```
from ultralytics import YOLO
import torch

def main():
    print(torch.cuda.is_available())
    print(torch.cuda.get_device_name(0))

    model = YOLO("best.pt")#зміна шляху до моделі
    result = model.train(data="config.yaml", epochs=50,
device=0)

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

Слід зазначити, що кількість епох повинна бути значно меншою від начальної кількості. Тому у даному випадку кількість епох було зменшено до 50. Робиться це для того, щоб модель не оцінювала звичайні та аргументовані умови як рівні. Аргументації – це лише додатковий приклад.

Таким чином, після присвоєння аугментацій та донавчання моделі, датасет збільшився на 300 зображень (взято 1/3 з кожного класу), по 75 зображень з певною аугментацією.

Також для підвищення якості роботи рішення в різних погодних умовах було впроваджено механізм розпізнавання погодних умов та накладання відповідних фільтрів для підвищення видимості об’єктів на зображенні у вигляді класу WeatherConditionDetector.

Метод detect призначений для класифікації поточного зображення за умовами зйомки. Спочатку кадр переводиться в колірний простір HSV для виділення яскравості (Value, або канал V). Якщо середня яскравість менше 50 – припускається нічна сцена. В інших випадках перевіряються евристики на туман, дощ, сніг відповідними методами (лістинг 3.20).

Лістинг 3.20 – Оголошення класу WeatherConditionDetector та основного методу detect

```
class WeatherConditionDetector:
    def detect(self, frame):
        hsv = cv2.cvtColor(frame, cv2.COLOR_RGB2HSV)
        brightness = np.mean(hsv[..., 2])

        if brightness < 50:
            return 'night'
        elif self.is_foggy(frame):
            return 'fog'
        elif self.is_rainy(frame):
            return 'rain'
        elif self.is_snowy(frame):
            return 'snow'
        else:
            return 'clear'
```

Туман характеризується низькою контрастністю, тому варіація значень лапласіану використовується як індикатор. Якщо значення менше 100 – сцена вважається туманною (лістинг 3.21).

Лістинг 3.21 – Визначення туманної погоди

```
def is_foggy(self, frame):
    gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
    laplacian_var = cv2.Laplacian(gray, cv2.CV_64F).var()
    return laplacian_var < 100
```

Наявність дощу може викликати численні дрібні відблиски або смуги, які створюють багато контурів. Алгоритм Кенні виявляє контури, і якщо їх кількість перевищує 15 000 – припускається дощ (лістинг 3.22).

Лістинг 3.22 – Виявлення дощу

```
def is_rainy(self, frame):
    gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
    edges = cv2.Canny(gray, 100, 200)
    return np.sum(edges > 0) > 15000
```

Сніг зазвичай проявляється великою кількістю білих пікселів. Порогом вважається понад 10000 пікселів з яскравістю вище 220 (лістинг 3.23).

Лістинг 3.23 – Виявлення снігу

```
def is_snowy(self, frame):
    gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
    white_pixels = np.sum(gray > 220)
    return white_pixels > 10000
```

Залежно від виявленої умови, до зображення застосовується відповідний фільтр (лістинг 3.24).

Лістинг 3.24 – Застосування фільтрів відповідно до умов

```
def apply_filter(self, frame, condition):
    if condition == 'fog':
        return cv2.detailEnhance(frame, sigma_s=10,
sigma_r=0.15)
    elif condition == 'rain' or condition == 'snow':
        return cv2.fastNlMeansDenoisingColored(frame, None, 10,
10, 7, 21)
    elif condition == 'night':
        return cv2.convertScaleAbs(frame, alpha=1.5, beta=30)
    return frame
```

Перевірка ж на погодні умови відбувається один раз при активації модуля, після цього застосовуються відповідні фільтри.

3.5 Технічна реалізація апаратної частини

Апаратна реалізація інтелектуального програмно-апаратного комплексу здійснена на основі сучасного одноплатного комп’ютера Raspberry Pi 5, який забезпечує необхідну обчислювальну потужність та енергоефективність для реалізації алгоритмів комп’ютерного зору, розпізнавання тексту та синтезу мовлення в режимі реального часу. Вибір цієї платформи обумовлений її компактністю, високим рівнем інтеграції, підтримкою популярних бібліотек Python (OpenCV, Tesseract, pyttsx3) та активною спільнотою розробників.

Мікрокомп’ютер Raspberry Pi 5 оснащений 4-ядерним процесором Arm Cortex-A76 з тактовою частотою 2.4 ГГц, 8 ГБ оперативної пам’яті типу

LPDDR4X, роз’ємами USB 3.0/2.0, HDMI, Gigabit Ethernet, а також слотом для microSD-карти або NVMe SSD через PCIe (за потреби). У межах реалізованого комплексу використовується пасивно-активна система охолодження, яка включає компактний вентилятор, встановлений на поверхні радіатора, що забезпечує стабільну роботу процесора навіть за підвищеного навантаження під час обробки відеопотоку.

Для захоплення зображень застосовується зовнішня Logitech Web Camera з об’єктивом Carl Zeiss Tessar 2.0/3.7, яка має роздільну здатність 2 МП. Камера підключається до порту USB 3.0 на платі Raspberry Pi, що забезпечує достатню пропускну здатність для передачі відеопотоку без втрат. Вибір саме цієї моделі зумовлений її сумісністю з Linux-системами, стабільною роботою через стандарт UVC (USB Video Class), а також компактними габаритами, які дозволяють використовувати камеру в носимих або мобільних рішеннях.

Аудіовихід реалізовано за допомогою зовнішньої USB-звукової карти, яка входить до комплекту Logitech G Pro X. Оскільки самі повнорозмірні навушники не використовуються в системі, до звукової карти підключаються стандартні дротові навушники-крапельки через вихід 3.5 мм (minijack). USB-звукова карта також підключається до одного з портів USB 3.0 Raspberry Pi 5, що гарантує якісну передачу аудіосигналу з мінімальною затримкою. Система працює в офлайн-режимі, що дозволяє уникнути залежності від Інтернету.

Живлення пристрою здійснюється через USB-C порт, який підтримує стандарт Power Delivery (PD). Враховуючи технічні вимоги Raspberry Pi 5 (до 5В / 5А, що дорівнює 25 Вт) та наявність підключеної камери й аудіоінтерфейсу, обрано конфігурацію живлення від павербанка з підтримкою PD 3.0 та який підтримує вихідні параметри 5В / 5А або 9В / 3А, що автоматично адаптується для Raspberry Pi. Оснащений цифровим дисплеєм заряду, USB-C виходами, мають достатню ємність для автономної роботи системи протягом кількох годин.

На пристрої попередньо встановлено операційну систему Raspberry Pi OS Lite, яка оптимізована для роботи без графічного інтерфейсу, що дозволяє мінімізувати споживання ресурсів. Уся логіка обробки зображень, виконання моделі YOLOv8n, розпізнавання номера маршруту та озвучення результатів реалізована за допомогою Python. Програмне забезпечення розміщено в окремому каталозі на внутрішній SD-карті. Для зручності користування передбачено автоматичний запуск скрипта після увімкнення пристрою за допомогою системного механізму systemd, що гарантує автономність та безперервність функціонування без необхідності втручання користувача.

Таким чином, технічна реалізація апаратної частини проєкту забезпечує повну незалежність системи від зовнішніх серверів чи операторів, дозволяючи працювати в режимі реального часу в умовах обмеженого доступу до інфраструктури (на зупинках, у русі, вночі або під дощем). Компактність, енергоефективність і висока інтеграція компонентів робить цю систему зручною для носимого або мобільного застосування.

3.6 Тестування модуля

З метою перевірки працездатності розробленого програмно-апаратного комплексу було проведено тестування в умовах, максимально наближених до реального середовища експлуатації. Основна мета тестування полягала у визначенні стабільності роботи системи, точності детекції транспортних засобів, розпізнавання номерів маршрутів та якості озвучення інформації користувачу у різних умовах зовнішнього середовища.

Тестування складалося з двох частин: імітаційне в середовищі розробки та в польових умовах на вулиці. На першому етапі, за допомогою програми OBS (Open Broadcaster Software), було згенеровано віртуальний відеопотік (OBS Virtual Camera), що демонстрував зображення різних типів транспорту (автобуси, тролейбуси, трамваї) у стандартних та ускладнених умовах (туман, ніч, дощ). Це дозволило перевірити поведінку моделі у ситуаціях, де реальне

тестування було фізично неможливим через літній сезон (таблиця 3.1).

Таблиця 3.1 – Таблиця результатів тестування в реальних та імітованих умовах

Умови тестування	Кількість тестів, шт.	Точність детектування ТЗ (точність), %	Точність класифікації типу ТЗ, %	OCR правильний, %	Повне озвучення (успіх), шт.
Реальні, денне світло, (100-10000 lx)	10	100%	90%	70%	7
Реальні, сутінки, (50-100 lx)	10	80%	70%	50%	5
Імітація, темрява, (10-50 lx)	10	90%	70%	70%	7
Імітація, туман	10	100%	90%	70%	7
Імітація, дощ	10	100%	100%	80%	6
Імітація, сніг	10	100%	100%	90%	9
Імітація, замилення	10	100%	80%	60%	6
Усього / Середнє	70	95.7%	85.7%	70%	47

Другий етап тестування проводився в місті Харкові протягом одного дня, на зупинках громадського транспорту у звичайну літню погоду. Було обрано кілька локацій, де спостерігався рух автобусів, тролейбусів та трамваїв у денний та вечірній час.

Тож систему протестовано в загальній кількості 70 сценаріїв, із яких 20 – у реальних умовах (денне світло та сутінки), а 50 – в імітаційних (з використанням віртуальної камери OBS для моделювання складних погодних і світлових умов).

У реальних умовах удень система показала максимальну точність: всі транспортні засоби були виявлені, а OCR спрацював у 70% випадків. Це вказує на стабільність і надійність системи в оптимальних умовах освітлення. У вечірній період точність дещо знизилась – детекція знизилась до 80%, а повне озвучення вдалося лише в половині випадків. Це пояснюється зниженням якості зображення, збільшенням тіней і відблисків, що ускладнюють OCR.

В імітаційних умовах система демонструє стабільно високу здатність до виявлення об’єктів (80-100% у всіх групах), що свідчить про хорошу генералізацію YOLOv8n навіть у складних сценах. Однак, OCR показує помітно нижчу точність: лише 50-80% залежно від сценарію.

Для більш зручного аналізу, дані було переведено в графічний вигляд. Можна побачити статистику деткції транспорту (рисунок 3.13).

Кількість хибних класифікацій або недетекцій за весь експеримент становила лише 10 з 70 (14.29%), з них частина – через часткове перекриття транспорту або помилкове визначення типу (наприклад, тролейбус як автобус). Цей показник є прийнятним на етапі прототипу.

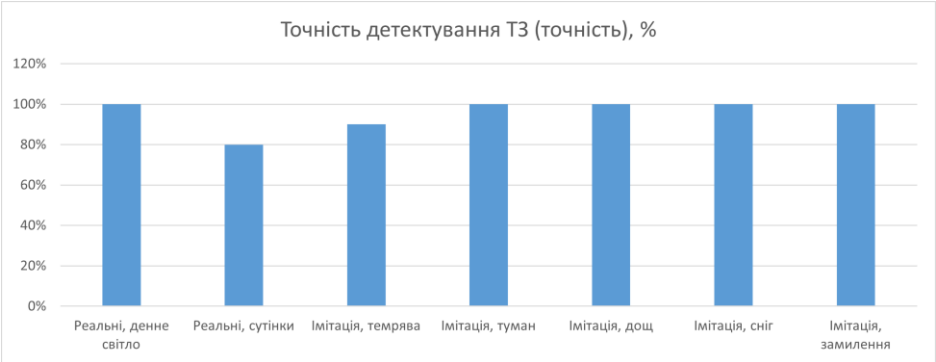

Рисунок 3.13 – Діаграма успішності детекції транспорту

Створено діаграму для перегляду успішності класифікації транспорту (рисунок 3.14).

Рисунок 3.14 – Діаграма успішності класифікації транспорту

Сформовано результативну діаграму для перегляду найуспішніших кейсів (рисунок 3.15).

Рисунок 3.15 – Діаграма з результатами детекії транспорту та номеру

4 ІНСТРУКЦІЯ КОРИСТУВАЧА ТА ДЕМОНСТРАЦІЯ РОБОТИ

4.1 Склад системи та комплектація

Розроблений програмно-апаратний комплекс складається з низки компактних електронних та периферійних пристроїв, які забезпечують повноцінну автономну роботу системи в режимі реального часу. Основними функціональними елементами є мікрокомп’ютер Raspberry Pi 5, камера для відеозахоплення, аудіоінтерфейс для озвучення результатів, навушники, джерело живлення та допоміжне кріплення для носимого формату використання.

Центральним обчислювальним модулем комплексу виступає Raspberry Pi 5, обладнаний чотириядерним процесором ARM Cortex-A76, 8 ГБ оперативної пам’яті та пасивно-активною системою охолодження з вентилятором. На пристрої попередньо встановлено операційну систему Raspberry Pi OS Lite, а програмне забезпечення зберігається на внутрішній microSD-карті. Для коректної роботи в режимі високого навантаження передбачено підключення стабільного джерела живлення потужністю не менше 25 Вт (рисунок 4.1).

Рисунок 4.1 – Raspberry Pi 5 з встановленою системою охолодження

Для захоплення відео використовується зовнішня USB-камера Logitech Web Camera з об’єктивом Carl Zeiss Tessar 2.0/3.7, яка має роздільну здатність 2 МП і підтримує високу чіткість зображення навіть за слабкого освітлення. Камера оснащена якісною лінзою з автофокусом і сертифікатом UVC (USB Video Class), що забезпечує повну сумісність із Linux-системами та стабільну роботу на платформі Raspberry Pi. Пристрій кріпиться на грудях користувача або на верхній частині одягу за допомогою кліпси чи ремінця, орієнтуючись у напрямку зору користувача (рисунок 4.2).

Рисунок 4.2 - USB-камера Logitech Web Camera

Озвучення інформації реалізується за допомогою зовнішньої звукової карти Logitech G Pro X, до якої підключаються звичайні дротові навушники через роз’єм 3.5 мм. Така конфігурація дозволяє забезпечити стабільну якість звуку та мінімальні затримки при синтезі мовлення. Рекомендовано використовувати один навушник, що вільно розміщується у вусі користувача (рисунок 4.3).

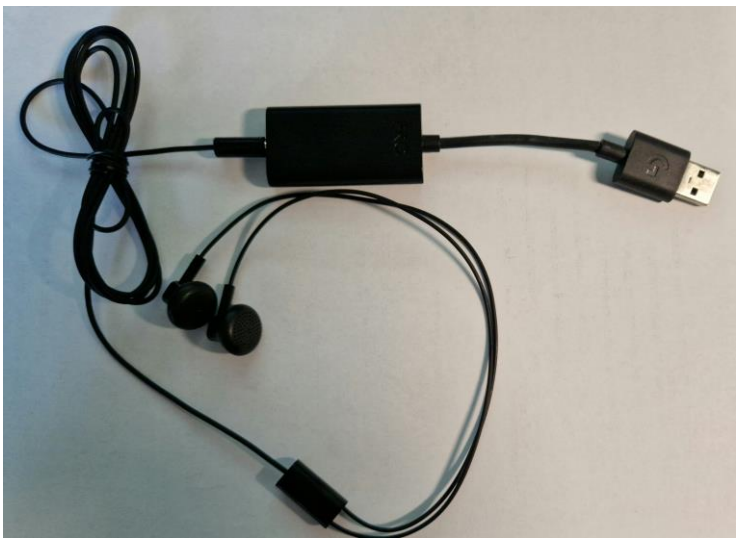

Рисунок 4.3 – Звукова карта Logitech G Pro X з навушником

Комплекс живиться від зовнішнього накопичувача з підтримкою Power Delivery. Ємність – 10000 мА, вихідна потужність – 5В / 3А. Це дозволяє забезпечити безперервну роботу пристрою протягом 4-6 годин. У демонстраційному зразку використовується джерело живлення типу Baseus PD 20W 10000mAh (рисунок 4.4).

Рисунок 4.4 – Джерело живлення

Уся система формує носимий пристрій (рисунок 4.5).

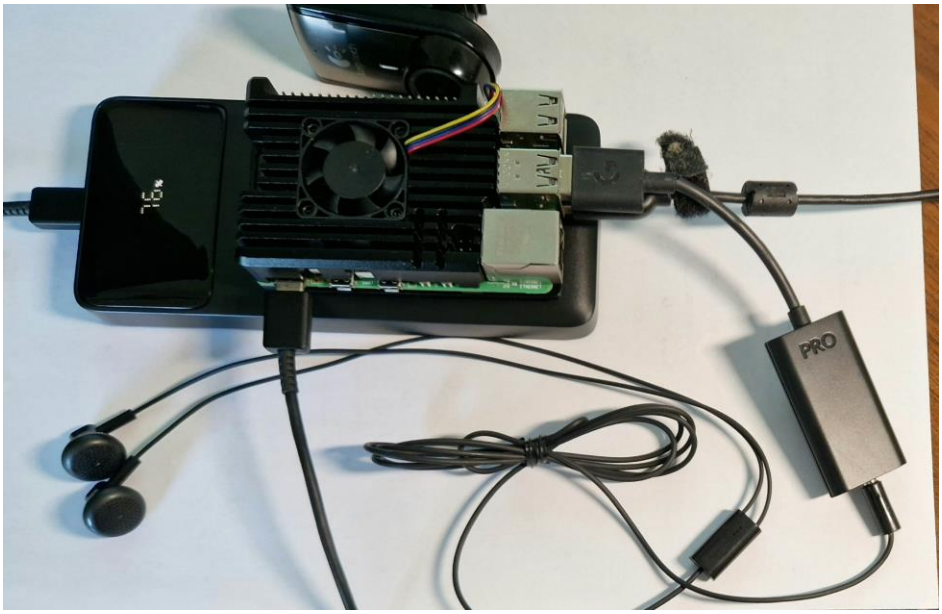

Рисунок 4.5 – Загальний вигляд модуля

Послідовність встановлення апаратного модуля: 1) Raspberry Pi закріплюється на поясі користувача; 2) камера – на грудях; 3) навушник – у вусі; 4) Павербанк – ззаду на поясі. Проводка прокладена по одягу таким чином, щоб не заважати рухам і не обмежувати огляд (рисунок 4.6).

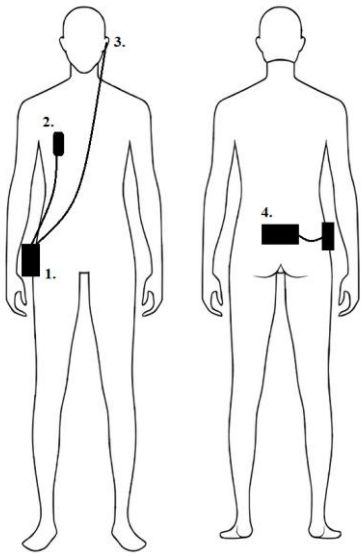

Рисунок 4.6 – Розташування елемнетів на користувачі

Таким чином, оскільки елементи є достатньо компактними та легкими, перенесення установки буде простим та зручним для користувача.

4.2 Підготовка до роботи

Для забезпечення стабільного функціонування програмно-апаратного комплексу перед його використанням необхідно виконати низку підготовчих дій, спрямованих на підключення всіх компонентів, правильне фізичне розміщення елементів системи та перевірку готовності до автономної роботи.

Передусім слід переконатися, що павербанк повністю заряджений. Рекомендовано використовувати зовнішнє джерело живлення з підтримкою технології Power Delivery потужністю не менше 25 Вт, що дозволяє уникнути раптових вимкнень системи або зниження продуктивності при пікових навантаженнях. Заряджання павербанка здійснюється стандартним USB-C кабелем через мережевий адаптер.

Далі здійснюється підключення периферійних пристроїв (якщо не під’єднані) до Raspberry Pi 5. У порт USB 3.0 підключається камера Logitech Web Camera Carl Zeiss Tessar 2.0/3.7, яка автоматично розпізнається операційною системою без необхідності встановлення додаткових драйверів. В інший порт USB 3.0 підключається звукова карта Logitech G Pro X, до якої через аудіороз’єм mini-jack (3.5 мм) під’єднуються звичайні дротові навушники, що дозволяють користувачу почути голосові сповіщення від системи.

Після цього необхідно розмістити обладнання на тілі користувача. Raspberry Pi закріплюється на поясі за допомогою спеціального кріплення, що забезпечує вентиляцію корпусу. Камера встановлюється в області грудей, орієнтовано прямо перед собою – важливо забезпечити стабільне положення без нахилів, що дозволяє отримати правильний кут огляду для системи комп’ютерного зору. Навушник вставляється у вухо з тієї сторони, яка не перекривається одягом або волоссям.

Після підключення всіх елементів слід увімкнути джерело живлення, після чого Raspberry Pi автоматично завантажується, а встановлений програмний модуль запускається у фоновому режимі без участі користувача. Система одразу починає захоплення відео, і при виявленні транспортного засобу спрацьовує механізм озвучення інформації про його тип та номер маршруту.

Таким чином, процес підготовки до роботи займає не більше кількох хвилин та не потребує спеціальної технічної підготовки користувача, що є важливою умовою практичного використання системи людьми з порушеннями зору.

4.3 Сценарій типового використання

Типовим сценарієм експлуатації програмно-апаратного комплексу є ситуація, коли користувач із порушенням зору очікує громадський транспорт на зупинці. Система, закріплена на тілі, працює в автономному режимі. Наступний сценкрій містить таку послідовність дій:

1) Увімкнення
- користувач із вадами зору підходить до зупинки;
- підключає павербанк – система автоматично запускається.

2) Робота системи
- камера на грудях транслює відео;
- кожен кадр аналізується моделями YOLOv8n і Tesseract.

3) Розпізнавання транспорту
- виявляється тип транспорту (автобус, тролейбус, трамвай);
- якщо можливо – визначається номер маршруту.

4) Оповіщення
- якщо номер є: звучить “Автобус, номер 45”;
- якщо номер не розпізнано: звучить лише тип – “Тролейбус”;
- звук передається у навушник.

У разі, якщо очікуваний транспорт не озвучується, слід врахувати можливий вплив змінних умов середовища – раптове зниження освітлення (наприклад, вхід у тінь), сильне зустрічне сонце, туман або опади можуть призвести до зниження якості кадру. У такому випадку рекомендовано перезавантажити систему, короткочасно вимкнувши й знову увімкнувши живлення через power bank. Це призведе до повторного запуску програмного модуля із застосуванням оновлених фільтрів, адаптованих до поточних умов.

4.4 Обмеження розробленої системи

Хоча запропонований програмно-апаратний комплекс демонструє високу ефективність у більшості стандартних ситуацій, на етапі прототипу система має низку функціональних та експлуатаційних обмежень, які необхідно враховувати під час її використання.

Перш за все, пристрій не має захисту від вологи та атмосферних опадів, що унеможливлює його експлуатацію в умовах дощу або снігопаду. Камера, звукова карта та сам Raspberry Pi працюють у відкритому середовищі без герметичного корпусу, тому потрапляння вологи може призвести до короткого замикання або виходу з ладу компонентів. З цієї причини використання системи в умовах опадів не рекомендується, якщо вона не дооснащена водозахисним кожухом чи герметичним контейнером.

Попри це, сценарії роботи системи в умовах дощу, снігу та туману були враховані на етапі імітаційного тестування. За допомогою програмної симуляції складних погодних умов вдалося оцінити стійкість алгоритмів до втрати контрастності, розмиття кадру та шумів. Ці експерименти довели, що основна проблема в таких умовах – погіршення результатів OCR-розпізнавання, тоді як детектор об'єктів (YOLOv8n) продовжує працювати з високою точністю.

Ще одним обмеженням є необхідність стабільного рівня заряду павербанка. У випадках, коли заряд опускається нижче 20–30%, можливе

зниження напруги живлення та нестабільна робота модулів (особливо звукового синтезу). Тому перед використанням системи рекомендується переконатися, що заряд power bank становить не менше 50%, а краще – повний.

Робочий температурний діапазон системи становить орієнтовно від 0 °C до +35 °C. При нижчих температурах можливе охолодження елементів нижче номінальних значень, що порушує стабільність роботи. При високих температурах за умов інтенсивного сонячного освітлення може виникати перегрів, особливо при тривалій роботі без зовнішнього охолодження. У системі передбачено пасивно-активне охолодження Raspberry Pi, однак для експлуатації влітку на відкритому сонці рекомендується проводити періодичне охолодження пристрою або використовувати його у тіні.

Також слід враховувати, що система не має візуального або звукового інтерфейсу контролю стану. Відсутність озвученого повідомлення не завжди означає відсутність транспорту – можливо, об'єкт був закритий, номер нечитабельний, або поточні умови середовища різко змінились. У таких випадках рекомендовано перезавантажити пристрій, тимчасово вимкнувши живлення через павербанк і знову увімкнувши його через кілька секунд.

Попри зазначені обмеження, прототип системи демонструє стабільну роботу в більшості міських умов, для яких і був розроблений, а врахування граничних ситуацій у процесі тестування свідчить про готовність комплексу до подальшого доопрацювання.

4.5 Перспективи подальшого розвитку

Незважаючи на те, що запропонований програмно-апаратний комплекс уже забезпечує базову функціональність системи озвучення громадського транспорту для осіб із вадами зору, він має значний потенціал для подальшого розвитку, масштабування та інтеграції з іншими технологічними рішеннями.

Одним із ключових напрямів удосконалення є розширення та уточнення навчального датасету, зокрема за рахунок збирання нових прикладів у різних погодних умовах, при різному освітленні та з різних ракурсів. Це дозволить покращити генералізацію моделі та зменшити кількість хибних класифікацій. Разом із цим можливо реалізувати механізм повторного донавчання моделі в польових умовах або на сервері за підсумками збору зворотного зв'язку від користувачів, що дозволить адаптувати систему під локальні особливості транспорту та міського середовища.

Іншим важливим напрямом є інтеграція з GPS-модулем, що дозволить не лише визначати місце розташування користувача, а й пов'язувати його з даними про зупинки, графіки руху або маршрути транспорту. Це створить умови для реалізації повноцінної навігаційної системи: наприклад, система зможе озвучити, який саме автобус наближається до потрібної зупинки, або попередити про зміну маршруту.

Також перспективною є інтеграція з мобільними додатками та сервісами типу EasyWay, Moovit, CityBus тощо, що дозволить поєднувати візуальну та голосову інформацію з офіційними даними розкладу, прогнозами прибуття транспорту та іншими інфраструктурними можливостями міста. У такому разі система зможе не тільки "бачити" транспорт, а й знати, що саме має прибути до зупинки.

З технічної точки зору, одним із стратегічних завдань є розробка спеціалізованої одноплатної апаратної платформи, яка поєднувала б функціональність Raspberry Pi, джерело живлення та необхідні периферійні модулі в одному компактному пристрої. Така інтеграція дозволила б значно зменшити розміри системи, покращити її ергономіку, а також оптимізувати енергоспоживання. Перспективним є створення власного одноплатного комп'ютера з апаратним прискоренням обробки нейромережевих моделей (наприклад, за рахунок вбудованого GPU або NPU), що дозволить забезпечити ще вищу швидкодію при одночасному зниженні навантаження

на центральний процесор.

Додаткові можливості для розвитку включають багатомовність озвучення (зокрема реалізацію української, англійської, польської мов у залежності від налаштувань), індивідуальне налаштування режиму повідомлень (лише тип транспорту, тип + номер, голос + вібрація), а також створення мобільного застосунку для супроводу користувача, контролю стану пристрою, оновлення прошивки та персоналізації параметрів.

Таким чином, створена система закладає надійне функціональне підґрунтя для розбудови масштабованого інструменту мобільної орієнтації у міському середовищі, що здатен адаптуватися до потреб слабозорих користувачів і розвиватися разом із сучасними технологічними тенденціями.

ВИСНОВКИ

Проблема орієнтації слабозрих осіб в міських умовах на сьогоднішній день є дуже актуальною проблемою в Українському суспільстві, що підтверджують відповідні дослідження. Проаналізувавши всі сучасні підходи до вирішення проблем навігації та орієнтації незрячих в просторі, можна прийти висновку, що дуже мало з них вирішують таку важливу проблему, як переміщення особи з вадами зору засобами громадського транспорту.

Вирішення даної проблеми потребує спеціалізованого технічного рішення, яке не буде потребувати підключення до мережі у разі її відсутності або перебування за межами міста, та не займати ресурси смартфону, який також може виконувати інші задачі навігації по місту, як наприклад Google Maps. Таким рішення може бути окремий програмно-апаратний модуль, незалежний від мережі та смартфону, та зручний для перенесення та в умовах знаходження на просторі міських вулиць.

Для розробки запропонованого рішення було вирішено низку завдань:

- зібрано та розмічено дані для глибинного навчання моделі YOLO розпізнаванню різних типів транспортних засобів (автобус, тролейбус, трамвай) та їх маршрутних номерів;
- аугментовано датасет шляхом моделювання умов низької видимості, створюючи штучні погодні умов (туман, дощ, ніч тощо) в різний час доби (денне світло, сутінки, темрява);
- розроблено програмний модуль ідентифікації транспорту із застосуванням методів комп’ютерного зору;
- розроблено програмний модуль голосового супроводу на основі методів синтезування мовлення для відтворюння в режимі реального часу через аудіоінтерфейс розробленого пристрою;
- побудовано апаратний модуль для збору, аналізу та постпроцесінгу даних;

- протестовно та адаптовано системи до реальних умов експлуатації.

За результатами тестування зроблено висновок, що система потребує покращення, оскільки результати тестування в умовах темряви та сутінок, а також в умовах замилення показали не достатньо чіткі результати. Також, проаналізувавши різницю між тестуванням в реальних умовах сутінок та імітованій темряві видно, що в реальних умовах показники гірше ніж в імітованих. Це говорить про те, що тестування в усіх імітованих сценаріях може показувати результат не релевантний до реальних умов.

Тому у подальших перспективах розвитку слід розглянути можливості для розширення датасету зображеннями зробленими в реальних умовах обмеженої видимості, та покращення алгоритмів аналізу та обробки зображень для кращої видимості транспортних засобів та їх маршрутних номерів на зображення.

Можна додати варіативну можливість підключення до інтернет-мережі для інтеграції з Google Maps або іншими додатками для відстеження геолокації та розкладів громадського транспорту.

Також, слід розробити більш захищений та зручніший корпус для апаратного модуля, який буде забезпечувати умови вологостійкості, захисту від переохолодження та перегрівання.

Ще одним шляхом покращення рішення може бути відмова від елементів зовнішнього живлення формату power bank, та інтеграція власного акумулятора, що дозволить розмірити джерело живлення в корпусі більш компактно.

В цілому, розроблений інтелектуальний програмно-апаратний модуль забезпечує високу якість розпізнавання громадського транспорту та маршрутних номерів в умовах денного освітлення, не має проблем з портативністю та автономністю та цілком забезпечує можливі потреби салбозорих осіб при використанні зупинок громадського транспорту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Вміння користуватися білою тростиною – одна з головних компетенцій незрячої дитини. URL: https://naurok.com.ua/stattya-vminnya-koristuvatisya-biloyu-trostinoyu---odna-z-golovnih-kompetenciy-nezryacho-ditini-198664.html (даат звернення: 12.05.2025).

2. Волошина О. В. Основи корекційної педагогіки : конспект лекцій. Вінниця : ВДПУ ім. М. Коцюбинського, 2012. – 149 с.

3. Закон України Про реабілітацію у сфері охорони здоров'я. URL: https://zakon.rada.gov.ua/laws/show/1053-20#Text (дата звернення 21.05.2025).

4. Нечітайло О. В., Гаврашенко А. О. Інтелектуальний модуль навігації для реабілітації осіб з вадами зору на зупинках громадського транспорту. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління : тези доп. 15-ї міжнар. наук.-техн. конф. (24–25 квіт. 2025 р., Баку – Харків – Жиліна). – Т. 3. – Харків ; Баку ; Жиліна, 2025. – С. 123.

5. Blindness and vision impairment. URL: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment (дата звернення: 12.05.2025).

6. Bradski G. The OpenCV Library. Dr. Dobb's Journal of Software Tools. – 2000.

7. Convolutional Recurrent Neural Network For Text Recognition. URL: https://www.xenonstack.com/insights/crnn-for-text-recognition (дата звернення: 13.05.2025).

8. Datasets Overview. URL: https://docs.ultralytics.com/datasets/#steps-to-contribute-a-new-dataset (дата звернення: 13.05.2025).

9. Dev Board. URL: https://coral.ai/products/dev-board/ (дата звернення: 14.05.2025).

10. EasyOCR documentation. URL: https://www.jaided.ai/easyocr/ (дата

звернення: 13.05.2025).

11. Faster R-CNN vs YOLO vs SSD – Object Detection. Algorithms. IBM Data and AI, 2022.

12. Goodfellow, Bengio, Courville. Deep Learning. MIT Press. 2016. URL: https://www.deeplearningbook.org/ (дата звернення: 13.05.2025).

13. Google Cloud Text-to-Speech API Overview. URL: https://cloud.google.com/text-to-speech/docs/overview (дата звернення: 14.05.2025).

14. gTTS: Google Text-to-Speech Python Library. URL: https://pypi.org/project/gTTS (дата звернення: 14.05.2025).

15. Islam, M. M., et al. A Classical Approach to Handcrafted Feature Extraction Techniques. arXiv, 2022.

16. Januszewski A. Python Automation Cookbook: 75 Python Automation Ideas for Web Scraping, Data Wrangling, and Processing Excel, Reports, Emails, and More. Packt Publishing, 2020.

17. Joshi, K. OpenCV 4 with Python Blueprints. Packt Publishing, 2019.

18. Juneja A., Kumar V., Singla S. A Systematic Review on Foggy Datasets: Applications and Challenges. Archives of Computational Methods in Engineering, 2021.

19. Keylabs. YOLOv8 vs Faster R-CNN: A Comparative Analysis, 2023.

20. NVIDIA Jetson Nano. URL: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development/ (дата звернення: 14.05.2025).

21. Object Detection Datasets Overview. URL: https://docs.ultralytics.com/datasets/detect/ (дата звернення: 13.05.2025).

22. ODROID-XU4. URL: https://wiki.odroid.com/odroid-xu4/odroid-xu4 (дата звернення: 14.05.2025).

23. OpenCV Documentation – VideoCapture. URL: https://docs.opencv.org/4.x/d8/dfe/classcv_1_1VideoCapture.html (дата

звернення: 13.05.2025).

24. OrCam MyEye 3 Pro. URL: https://www.orcam.com/en-us/orcam-myeye-3-pro?srsltid=AfmBOopPeoIdzkklQ_6zSSKoikzJ5WNeq2oSImTPVByOspmq6PNcxWO9 (дата звернення: 12.05.2025).

25. PaddlePaddle OCR. GitHub Repository. URL: https://github.com/PaddlePaddle/PaddleOCR (дата звернення: 13.05.2025).

26. pyttsx3 documentation. URL: https://pyttsx3.readthedocs.io (дата звернення: 14.05.2025).

27. pyttsx3 – Text-to-Speech conversion library in Python. URL: https://pyttsx3.readthedocs.io (дата звернення: 14.05.2025).

28. Python OpenCV: Converting an image to gray scale. URL: https://techtutorialsx.com/2018/06/02/python-opencv-converting-an-image-to-gray-scale/ (дата звернення: 13.05.2025).

29. Raspberry Pi Ltd. Raspberry Pi 5 Product Brief. URL: https://datasheets.raspberrypi.com/rpi5/ (дата звернення: 14.05.2025).

30. Raspberry Pi 5: Everything You Need to Know. URL: https://www.lifewire.com/raspberry-pi-5-7975988 (дата звернення: 14.05.2025).

31. Schindler, K. Object Detection by Global Contour Shape. ETH Zürich, 2008.

32. Smith, R. (2007). An Overview of the Tesseract OCR Engine. Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), 629–633. URL: https://tesseract-ocr.github.io/docs/tesseracticdar2007.pdf (дата звернення: 13.05.2025)

33. Song J., Zhao L., Skinner K. A. LiRaFusion: Deep Adaptive LiDAR-Radar Fusion for 3D Object Detection, 2024.

34. Staudemeyer, R. C., & Rothstein Morris, E. Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks. arXiv preprint arXiv, 2019.

35. Szeliski R. Computer Vision: Algorithms and Applications. Springer,

2022.

36. The Sonic Pathfinder. URL: https://sonicpathfinder.org/pa/pf_blerb.html (дата звернення: 12.05.2025).

37. Torralba A., Efros A. A. Unbiased Look at Dataset Bias. arXiv, 2020.

38. Wang Y., Deng J., Li Y., Hu J., Liu C., Zhang Y., Ji J., Ouyang W., Zhang Y. Bi-LRFusion: Bi-Directional LiDAR-Radar Fusion for 3D Dynamic Object Detection. arXiv, 2023.

39. What does the brand new smart cane 2 do for you? URL: https://wewalk.io/en/product/ (дата звернення: 12.05.2025).

40. What is BlindSquare? URL: https://www.blindsquare.com/about/ (дата звернення: 12.05.2025).