Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

МІКРОКОНТРОЛЕРНА НЕЙРОМЕРЕЖЕВА СИСТЕМА КЕРУВАННЯ РУХОМИМ ОБ'ЄКТОМ В ДВОВИМІРНОМУ ПРОСТОРІ

08-54.МКР.020.00.000 ПЗ

Виконав студент 2 курсу, групи 1КІ-23м
спеціальності 123 — «Комп'ютерна інженерія»
_____ Фурман М.А.
Керівник к.т.н., доц. каф. ОТ
_____ Крупельницький Л. В.
«10» грудня 2024 р.
Опонент доц. каф. ПЗ
_____ Ткаченко О. М.
«10» грудня 2024 р.

Допущено до захисту
Завідувач кафедри ОТ
_____ д.т.н., проф. Азаров О.Д.
«__» _____ 2024 р.

ВНТУ 2024

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Галузь знань — Інформаційні технології
Освітньо-кваліфікаційний рівень — магістр
Спеціальність — 123 Комп'ютерна інженерія
Освітня програма – Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., професор Азаров О.Д.
"18" вересня 2024 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Фурману Максиму Андрійовичу

1 Тема роботи: «Мікроконтролерна нейромережева система керування рухомим об'єктом в двовимірному просторі», керівник роботи Крупельницький Леонід Віталійович к.т.н., доцент кафедри ОТ, затверджені наказом вищого навчального закладу від "17" вересня 2024 року № 310.

2 Строк подання студентом роботи: 15.12.2024

3 Вихідні дані до роботи: тип об'єкту — рухомий транспортний засіб; двовимірний простір з розміткою на дорозі; спосіб керування — автоматичний з використанням нейромережі; пуск — через безпровідний засіб зв'язку.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз сучасних систем автоматичного керування рухомим об'єктом, вибір складових системи, розробка алгоритмів роботи, розробка програмного забезпечення, компонування модулів системи, тестування працездатності системи.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) — структурна схема системи керування, структурна схема розпізнавання об'єктів, блок-схема алгоритму руху, схема з'єднань компонентів системи, блок-схема основного алгоритму роботи системи, презентація.

6 Консультанти розділів роботи представлено в таблиці 1

Таблиця 1 – Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1–4	Крупельницький Л.В., к. т. н., доцент каф. ОТ		
5	Небава М. І., доцент, к.е.н., професор каф. ЕПВМ		

7 Дата видачі завдання: 18.09.2024

8 Календарний план роботи зазначений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломної роботи	Строк виконання	Примітка
1	Постановка задачі роботи	19.09.2024	виконано
2	Пошук та аналіз методів управління самокерованими об’єктами	20.09.2024	виконано
3	Пошук та аналіз методів розпізнавання об’єктів	21.09.2024 – 22.09.2024	виконано
4	Аналіз аналогів нейромережевих систем керування рухомим об’єктом	23.09.2024 – 25.09.2024	виконано
5	Вибір компонентів для створення нейромережевої системи керування рухомим об’єктом	26.09.2024 – 05.10.2024	виконано
6	Створення програмного забезпечення та налаштування зв’язку між компонентами апаратної складової	06.10.2024 – 26.10.2024	виконано
7	Тестування працездатності системи	27.10.2024 – 10.11.2024	виконано
8	Розрахунок економічної частини	11.11.2024	виконано
9	Оформлення пояснювальної записки	12.11.2024	виконано
10	Перевірка якості виконання магістерської роботи та усунення недоліків	13.11.2024 – 14.11.2024	виконано
11	Підписи супроводжувальних документів у нормоконтролера, керівника, опонента	15.11.2024 – 16.11.2024	виконано
12	Перевірка на антиплагіат та ШІ	17.11.2024	виконано
13	Попередній захист роботи	18.11.2024	виконано

Студент _____ Фурман М.А.

Керівник роботи _____ Крупельницький Л.В.

АНОТАЦІЯ

УДК 004.9

Фурман М.А. Мікроконтролерна нейромережева система керування рухомим об’єктом в двовимірному просторі. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп’ютерна Інженерія, Вінниця: ВНТУ, 2024 — 140 с. На укр. мові. Бібліогр.: 25 назв; рис.: 65; табл. 15.

У магістерській кваліфікаційній роботі проаналізовано методи управління самокерованими рухомими об'єктами, розглянуто алгоритми комп’ютерного зору та обробки зображень у реальному часі, а також досліджено роботу нейромережі.

Базуючись на існуючих розробках, створено мікроконтролерну нейромережеву систему керування рухомим об’єктом, що здатна з високою точністю розпізнавати перешкоди та планувати маршрут на основі зібраних даних.

Ключові слова: нейромережа, розпізнавання об'єктів, обробка зображень, Arduino, Raspberry Pi, OpenCV.

ABSTRACT

УДК 004.9

M.A. Furman Microcontroller neural network control system of a moving object in two-dimensional space. Master's thesis on specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2024 — 140 p. In Ukrainian speech Bibliography: 25 titles; Fig.: 65; table 15.

In the master's thesis, the methods of controlling self-guided moving objects were analyzed, the algorithms of computer vision and real-time image processing were considered, and the work of a neural network was investigated.

Based on existing developments, a microcontroller-based neural network control system for a moving object has been created, which is able to recognize obstacles with high accuracy and plan a route based on the collected data.

Keywords: neural network, object recognition, image processing, Arduino, Raspberry Pi, OpenCV.

ЗМІСТ

ВСТУП

Актуальність роботи полягає в необхідності створення рухомих об'єктів та у впровадженні нейромережевих технологій для розвитку автономних систем. Системи на основі мікроконтролерів із використанням нейромереж здатні працювати в реальному часі, адаптуватись до змінних умов і взаємодіяти з оточуючим середовищем. Вони використовуються в автопілотах автомобілів, безпілотних літальних апаратах, робототехніці, промислових процесах, медичному обладнанні та багатьох інших сферах.

Створення нейромережевих систем керування рухомими об'єктами вирішує проблему автоматизації для виконання повторюваних та подібних завдань, зменшує фактор людського втручання та підвищує ймовірність на отримання успішного результату. Такі системи можуть полегшити життя користувача за рахунок оптимізації усіх процесів для раціонального розподілу ресурсів та надають гарантії безпеки під час експлуатації.

Системи керування рухомими об'єктами сприяють розвитку інновацій, створюючи платформу для впровадження штучного інтелекту, машинного навчання та аналізу даних, що підвищує їх адаптивність і функціональність.

Об'єктом дослідження є процеси керування рухомими об’єктами за допомогою нейромережевих алгоритмів

Предметом дослідження є мікроконтролерна нейромережева система керування рухомим об’єктом в двовимірному просторі.

Мета дослідження: створення мікроконтролерної нейромережевої системи керування рухомим об’єктом**,** що здатна з високою точністю розпізнавати перешкоди, аналізувати розмітку на дорозі та здійснювати рух.

Для досягнення мети необхідно необхідно вирішити такі задачі:

— проаналізувати аналоги нейромережевих систем автоматичного керування;

— здійснити огляд методів управління самокерованими об’єктами;

— здійснити огляд моделей нейромерж;

— дослідити роботу підпорядкованого пристрою на основі мікроконтролера;

— дослідити роботу керуючого пристрою на основі одноплатного комп’ютера;

— дослідити роботу відеокамери для введення зображень;

— створити алгоритми обробки зображень за допомогою бібліотеки OpenCV;

— натренувати нейромережу для розпізнавання зовнішніх об’єктів.

— розробити систему керування рухомим об’єктом.

Новизна: збільшено точність керування рухомим об’єктом за рахунок удосконалення каскадної моделі нейромережі для класифікації зображень та алгоритму керування рухом.

Практична цінність роботи полягає у застосуванні вдосконалених нейромережевих алгоритмів обробки зображень та алгоритму руху для керування рухомим об’єктом у двовимірному просторі.

Апробація результатів роботи здійснена під час виступу на конференції Молодь в науці: дослідження, проблеми, перспективи (МН-2024) факультету інформаційних технологій та комп'ютерної інженерії 2024 «Розпізнавання об'єктів із використанням нейронних мереж», Максим Андрійович Фурман, Леонід Віталійович Крупельницький.

Публікація за темою роботи — Фурман, М.; Крупельницький, Л.. Розпізнавання об'єктів із використанням нейронних мереж. Молодь в науці: дослідження, проблеми, перспективи, Ukraine, may. 2024. Available at: <https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21246>. Date accessed: 11 Oct. 2024. [1]

1 ОГЛЯД І АНАЛІЗ ІСНУЮЧИХ НЕЙРОМЕРЕЖЕВИХ СИСТЕМ КЕРУВАННЯ РУХОМИМИ ОБ'ЄКТАМИ

Безпілотний рухомий об’єкт — це пристрій або система, що здатна до пересування чи виконання завдань без безпосередньої участі людини. Такі об’єкти створені для автономного виконання функцій, навігації у різних середовищах або дистанційного керування [2]. До них належать різноманітні транспортні засоби, роботи, дрони та навіть космічні апарати.

Безпілотні об’єкти оснащені сенсорами, виконавчими механізмами та інтелектуальними системами, що забезпечують сприйняття оточення, аналіз отриманої інформації, прийняття рішень і виконання потрібних дій без необхідності постійного людського контролю чи фізичної присутності. Завдяки цьому такі об’єкти можуть ефективно виконувати свої завдання в різних умовах, знижуючи ризики, економлячи ресурси та мінімізуючи вплив людського фактору на процеси, де важливі точність і оперативність.

1.1 Історія розвитку нейромережевих систем керування

У 1950-80-х роках перші спроби автоматизації керування транспортом зосереджувалися на розробці простих, жорстко запрограмованих алгоритмів. Ці алгоритми могли забезпечувати певний рівень стабільності в ідеальних умовах, але вони не були стійкими до змін навколишнього середовища [3]. Ранні автопілоти в авіації використовувалися для підтримки рівного польоту, але потребували постійного нагляду та контролю з боку людини, адже вони не могли адаптуватися до змін під час специфічних погодніх умов або під час нестабільної повітряної обстановки, коли потоки вітру можуть значно змінити початковий маршрут або призвести до аварійної ситуації.

Наприклад, автопілот, встановлений на комерційних літаках, міг лише підтримувати висоту та курс, але на злеті чи посадці його потрібно було вимикати в цілях безпеки (див. рис. 1.1). Ідея більшої автоматизації була віддаленою, оскільки обчислювальних ресурсів не вистачало для складних

завдань, а дослідження в сфері штучного інтелекту лише починали розвиватися. Світ потребував змін у підходах до створення автоматизованих систем і наступним кроком мало стати революційне рішення, що могло б бути стійким у різних ситуаціях під час експлуатації.

Рисунок 1.1 — Система автопілоту Boeing C-1 Autopilot

З розвитком комп'ютерної техніки в 1990-х роках з'явилися перші багатошарові нейронні мережі, які дозволили розпізнавати візуальні та сенсорні дані на більш високому рівні, хоч і не гарантували повної автономії. Хоча ці моделі були обмежені через недостатню глибину шарів і обчислювальну потужність, їх почали використовувати в експериментах із автоматизації, активно проводити дослідження та створювати перші прототипи. Ці нейронні мережі могли аналізувати основні ознаки об’єктів, виявляти прості закономірності та навіть робити перші спроби керування, проте обмеженість технологіями того часу, не чіткі вхідні дані з сенсорів та фальшиві вихідні дані значно сповільнювали прогрес на цьому етапі.

Проект NavLab від Університету Карнегі-Меллон (CMU) став важливою та відомою спробою автоматизувати керування транспортним засобом. Навчальні автомобілі, обладнані простими нейронними мережами, могли

розпізнавати об’єкти на дорозі, але швидкість обробки інформації залишалася повільною, тому рух такої системи займав багато часу та не гарантував актуальність передавання даних між ітераціями (див. рис. 1.2.). Автомобіль міг рухатися по заданому курсу, але за умовами дороги все одно потрібен був контроль з боку оператора, щоб дотримуватися певної обережності [4]. Цей проект став важливим кроком до появи систем, здатних самостійно приймати рішення на дорозі.

Рисунок 1.2 — Навчальний автомобіль NavLab 1

На початку 2000-х років виникло багато досліджень спрямованих на аналіз концепції багатошарових конволюційних нейронних мереж (CNN), які певним чином здатні розпізнавати складні образи та деталі, раціональніше підходити до обробки кожного елементу даних. Важливою складовою подібних нейромереж був спосіб аналізу зображення в реальному часі, що є доволі важливим аспектом для створення автопілотів і систем автономного керування. Завдяки CNN автомобілі почали розпізнавати не лише статичні об'єкти, а й передбачати траєкторії руху пішоходів, інших транспортних засобів та реагувати на змінні дорожні умови, зчитувати інформацію з дорожніх знаків, реагувати на зміну кольору світлофору тощо.

Ближче до 2010-х років компанія Tesla почала використовувати такі нейромережі в своїй системі автономного керування, але вже з комерційною метою, аби зацікавити користувачів в ідеї напівавтономного керування з подальшим розвитком в абсолютно автономне (див. рис. 1.3.). Камери, встановлені на автомобілях, разом із глибокими нейромережами обробляли дорожні зображення, розпізнаючи лінії розмітки, знаки, світлофори та інші автомобілі на доволі високому рівні, мали якісний функціонал та інтерфейс для водіїв [5]. Це дозволило автомобілям Tesla з мінімальним втручанням водія утримуватися на смузі, змінювати її при необхідності, розпізнавати навколишні статичні та динамічні об'єкти і навіть виконувати обгони при певних умовах, коли це безпечно та відсоток успіху такого маневру достатньо високий.

Рисунок 1.3 — Автомобіль з автопілотом Tesla Model S

Згодом автоматизовані системи почали використовувати у форматі дронів або, інакше кажучи, у вигляді безпілотних повітряних літальних систем у військовій промисловості на базі нейромережевих систем керування для навігації на певній території або для виконання специфічних завдань, таких як розвідка чи атака у вигляді пуску ракет. Здатність нейромереж до самонавчання надала змогу дронам орієнтуватися в складних умовах, таких як

космічний простір, глибоководні середовища або бойові зони і на даний момент важко уявити сферу застосування в якій подібні системи не мали б застосування чи бодай спроб впровадження.

Наприклад, безпілотні літальні апарати Predator, розроблені для військових потреб, оснащені штучним інтелектом для аналізу даних із камер і сенсорів, що дозволяє зменшити навантаження на операторів і підвищити точність у визначенні об'єктів, а також надає змогу віддалено збирати дані на серверах чи оглядати місцевість у реальному часі, надавати команди чи змінювати маршрут тощо (див. рис. 1.4.). У наукових експедиціях підводні дрони використовуються для дослідження глибоких океанів або порожнин важкодоступних та небезпечних для занурення людей. Здатність нейромереж адаптуватися до динамічних змін навколишнього середовища допомагає їм орієнтуватися в умовах з низькою видимістю, високим тиском та низкою спеціальних умов, які відповідають вимогам конкретної системи чи пристрою

Рисунок 1.4 — Безпілотний літальний апарат MQ-1 Predator

Сучасні нейромережеві системи стали набагато більш адаптивними, стійкими до змін, ергономічними та легшими для освоєння. Завдяки швидким

обчислювальним процесорам та хмарним обчисленням тренування нейромереж стало набагато простішим та масштабнішим, адже системи тепер аналізують надзвичайно великі набори даних та навчаються значно якісніше. Для прикладу космічні дрони, такі як розвідувальні апарати для дослідження астероїдів, здатні використовувати НСК для автоматичної навігації в умовах низької гравітації та непередбачуваного рельєфу. Вони не лише аналізують зображення та поверхневі дані, але й адаптують свої траєкторії для обходу перешкод на основі отриманих у реальному часі даних, що дозволяє зменшити ризики пошкодження апаратів, які виконують складні наукові місії далеко від Землі і це б не стало б можливим без симуляції таких експедицій та тренування нейромереж на достатній кількості сценаріїв.

Однією з відомих моделей космічних дронів є NASA Astrobee. Це серія роботизованих дронів, розроблених NASA для роботи на борту Міжнародної космічної станції (див. рис. 1.5.). Astrobee призначений для автономного виконання різноманітних завдань, як от інспекція, моніторинг станції та допомога астронавтам у повсякденних завданнях.

Рисунок 1.5 — Космічний дрон NASA Astrobee

У майбутньому повна автономія транспорту, заснована на НСК, забезпечить безпечніші, ефективніші та інтегровані транспортні мережі.

Автономні автомобілі, дрони, літаки та підводні апарати зможуть обмінюватися даними в реальному часі для синхронізації дій, що може стати новим рівнем ройового інтелекту. Розумні міські мережі поєднають ці технології, оптимізуючи рух, уникатимуть заторів, знижуватимуть енергоспоживання та підвищуватимуть ефективність логістики, гарантуючи безпеку і адаптацію до змінних умов. Це статистично знизить ймовірність аварій, катастроф, а можливо зменшить їх до астрономічного мінімуму уже в найближчі десятиліття.

1.2 Класифікація методів управління самокерованими об’єктами

Керування безпілотними апаратами вимагає запровадження складних методів для забезпечення точної навігації, прогнозованої поведінки та високої стійкості до зовнішніх чинників. Такі методи дозволяють операторам або автономним системам контролювати безпілотні апарати, скеровуючи їх уздовж заданих траєкторій та дозволяючи здійснювати певний набір інструкцій для досягнення певної цілі. За способом застосування керування поділяється на три основні групи: напівавтономне, автономне з вказанням попередніх налаштувань та автономне в режимі реального часу (див. рис. 1.6).

1.2.1 Напіватономне керування

Один із основних підходів — це дистанційне ручне керування, при якому оператор використовує спеціальний пристрій для безпосереднього впливу на рух і дії безпілотного апарата [6]. Цей метод зазвичай застосовується для керованих дистанційно літальних апаратів, дронів або підводних дистанційно керованих апаратів (ROV). Оператор контролює швидкість, напрямок, висоту апарата та виконує необхідні завдання шляхом маніпуляцій засобами управління.

Системи управління зі зворотним зв'язком частіше знаходять своє застосування у керуванні безпілотними апаратами в режимі реального часу. Такі системи використовують датчики для моніторингу поточного стану чи

продуктивності апарата, порівнюють їх із еталонними, заздалегідь визначеними значеннями, або опорними сигналами та вносять коригувальні дії, мінімізуючи таке відхилення, застосовують певний набір функцій для локалізації об’єкту у просторі. Таким чином забезпечується постійне регулювання та стабілізація поведінки безпілотного апарата на кожній ітерації руху з урахуванням зовнішніх умов.

1.2.2 Автономне керування з попередніми налаштуваннями

Іншим методом є навігація за маршрутними точками. У цьому випадку безпілотні апарати запрограмовані для слідування наперед визначеним маршрутам або точкам навігації [7]. Оператор чи автономна система вводять серію координат чи контрольних точок, які апарат має послідовно досягати. Бортова навігаційна система апарата, спираючись на GPS або інерційні навігаційні системи, відслідковує його положення та коригує рух для послідовного досягнення кожної точки.

Для розробки оптимальних маршрутів або траєкторій використовуються методи планування шляху та формування траєкторії [8]. Алгоритми й математичні моделі дозволяють визначати найбільш ефективні або бажані траєкторії, враховуючи такі чинники, як уникнення перешкод, часові обмеження, споживання енергії або цільові параметри місії [9]. Подібні методи широко застосовуються в робототехніці, в автономному транспорті та для важливих космічних місій, забезпечуючи прогнозоване та резистентне планування маршрутів.

1.2.3 Автономне керування в реальному часі

Технології машинного навчання та штучного інтелекту змінюють підхід до керування безпілотними об'єктами, замінюючи побудову систем керування на основі чітко заданих алгоритмів, навчаючи її самостійно приймати рішення на великому наборі вхідних даних [10]. Завдяки цим методам безпілотні

об'єкти здатні виконувати складні завдання з більшою точністю на основі накопичених даних, що забезпечує більш самостійне керування без втручання користувача у процес. Алгоритми машинного навчання базуються на розпізнаванні певних зразків об’єктів навколо, на прогнозуванні поведінки системи згідно отриманих даних та в результаті здійснюють певні дії, які формуються чітко визначеними вимогами.

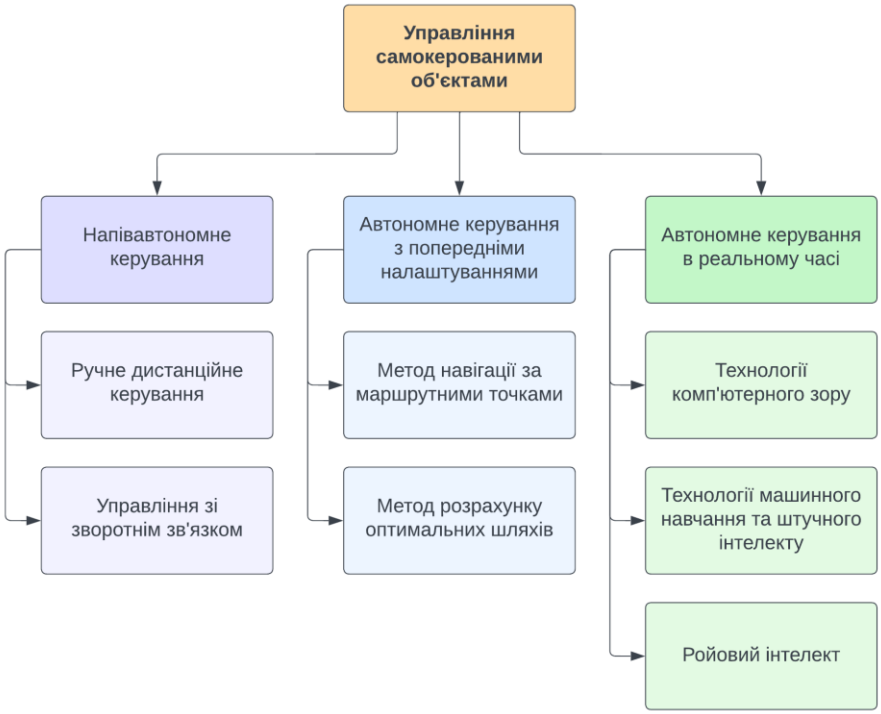

Рисунок 1.6 — Класифікація методів управління самокерованими об’єктами

Технології комп’ютерного зору та обробки зображень забезпечують можливість комплексного аналізу візуальної інформації, тобто даними зафіксованими камерами або іншими сенсорами для сприйняття навколишнього світу. Вони часто слугують фундаментом для інтерактивного зворотного зв’язку в керуванні безпілотними системами, адже дозволяють наочно побачити що знаходиться попереду системи [11]. Обробка візуальних даних у реальному часі, що надходять із камер та інших сенсорних пристроїв, надає змогу системі ідентифікувати об’єкти, виділяти характерні риси,

оцінювати просторові відстані та відстежувати динаміку руху. Такий метод може значно доповнити вже існуючу систему, в якій налаштований певний функціонал та правила руху і таким чином може підвищити точність чи навіть стати повноцінним джерелом отримання інформації.

Сучасні досягнення і найменш досліджені, зокрема технології ройового інтелекту, спрямовані на скоординоване управління множинними безпілотними об’єктами, які функціонують як єдина група [12]. Натхнені моделями поведінки природних систем, наприклад, комашиних колоній, алгоритми ройового інтелекту дозволяють цим об’єктам обмінюватися інформацією, взаємодіяти та погоджувати свої дії задля досягнення спільних цілей, перегруповуватись або створювати певну ієрархію взаємодії між собою. Такий підхід активно застосовується в так званій “ройовій робототехніці”, де групи роботів об’єднують зусилля та за допомогою налагодженої комунікації виконують складне завдання з більшим коефіцієнтом корисної дії. Адже кількість одночасно використовуваних елементів такої системи дозволяє ділити таку задачу на менші підзадачі та зменшує час очікування на результат.

Отже, керування безпілотними об’єктами включає різноманітні методології, які адаптовані для специфічних застосувань та цілей і кожен зі згаданих методів знаходить своє застосування в специфічних ситуаціях. Серед них — ручне дистанційне управління, навігація за маршрутними точками, планування траєкторії, системи зворотного контролю, машинне навчання, комп’ютерний зір і ройовий інтелект. Подібні технології дозволяють як операторам, так і автономним системам здійснювати ефективне управління безпілотними об’єктами, які в свою чергу виконують визначені завдання, адаптуються до змінних умов та досягають очікуваних результатів.

1.3 Огляд методів розпізнавання об'єктів у просторі

Розпізнавання об'єктів у просторі – це підхід до вирішення задач в області комп'ютерного зору, що полягає у визначенні та класифікації ознак об'єктів на зображеннях або у форматі відео. Існує перелік методів, які можна

розділити на традиційні та на методи глибокого навчання (див. рис. 1.7).

Традиційні методи розпізнавання об'єктів у просторі побудовані на використанні вручну створених функцій та алгоритмів для аналізу зображень та виділення характеристик об'єктів. Найбільш відомим є метод гістограм орієнтованих градієнтів або коротко HOG. Цей метод використовує градієнти пікселів на зображенні для створення дескрипторів, які визначають напрям та інтенсивність границь об'єктів, що дозволяє розпізнавати людей, транспортні засоби, але може бути менш ефективним у визначенні об'єктів з складними текстурами та формами.

Наприклад, для розпізнавання пішоходів на зображенні метод HOG розділяє зображення на невеликі блоки, у кожному з яких обчислюються градієнти для визначення напрямів змін яскравості. Ці дані зводяться до гістограми, яка узагальнює орієнтацію градієнтів у межах блоку. Усі гістограми об'єднуються в один дескриптор, який характеризує об'єкт у межах зображення.

Ще одним традиційним методом є метод Віоли — Джонса (Viola–Jones object detection), що полягає у використанні каскадних класифікаторів, які включають в себе послідовність класифікаторів для поетапного відсіювання областей зображення, що не містять об'єкти інтересу. Даний метод може значно прискорити процес розпізнавання, особливо для реального часу, але також може виявитися менш точним в порівнянні з глибоким навчанням у складних умовах або з неоднорідними зображеннями.

Наприклад, у задачі розпізнавання обличчя каскад Хаара побудований так що на початковому рівні класифікатора перевіряються прості ознаки присутності темних ділянок очей над світлими ділянками щік чи форми обличчя. У разі, якщо область проходить цей фільтр, на наступному рівні перевіряються складніші характеристики розташування країв рота чи носа. Кожен наступний етап відкидає ділянки, які не відповідають рисам чи заданим параметрам і продовжує аналіз тільки на можливих ділянках обличчя. Такий

підхід використовується в бібліотеках програмного забезпечення, таких як OpenCV, для створення ефективних детекторів обличчя.

Варто сказати, що традиційні методи розпізнавання об'єктів застосовуються в різноманітних завданнях в яких важливо врахування точно визначених правил та характеристик об'єктів. Проте такі методи можуть виявитися менш підходящими у порівнянні з глибоким навчанням в умовах, де даних не достатньою або сценарії є специфічними.

Метод SIFT (Scale-Invariant Feature Transform) є одним із найпопулярніших алгоритмів для виявлення та опису ключових точок на зображеннях. Він був розроблений Девідом Лоу у 1999 році та забезпечує інваріантність до масштабу, повороту та частково до змін освітлення й шуму. Алгоритм працює у кілька етапів: спочатку зображення проходить через гауссове згладжування для створення масштабного простору, де виявляються екстремуми у просторі Лапласа-Гаусса. Це дозволяє знайти потенційні ключові точки, які є добре визначеними і стійкими до змін. Далі відбувається їх точне визначення, усуваються точки, що є нестабільними або знаходяться на краях.

Після виявлення ключових точок алгоритм створює дескриптори для кожної з них. Ці дескриптори базуються на аналізі локальної області навколо ключової точки, де враховується орієнтація градієнтів. Це забезпечує стійкість до повороту зображення. Дескриптори є векторними представленнями, які можна легко порівнювати для пошуку відповідностей між зображеннями. Завдяки цьому метод SIFT широко використовується в задачах комп’ютерного зору, таких як об’єднання зображень, побудова панорам, розпізнавання об’єктів і тривимірна реконструкція.

Методи глибокого навчання для розпізнавання об'єктів у просторі здобули величезну популярність завдяки їх здатності автоматично вивчати репрезентації об'єктів з великої кількості даних. Однією з ключових архітектур для цього є згорткові нейронні мережі (CNN). Згорткові нейронні мережі (CNN) ефективно виявляють просторові та ієрархічні залежності в

зображеннях. Вони складаються зі спеціальних шарів, таких як згорткові шари, які використовують фільтри для виділення різних особливостей на різних рівнях абстракції, та пулінгові шари, які зменшують розмірність вихідних карт активації. Дана архітектура дозволяє автоматично визначати та використовувати значущі ознаки для класифікації та локалізації об'єктів.

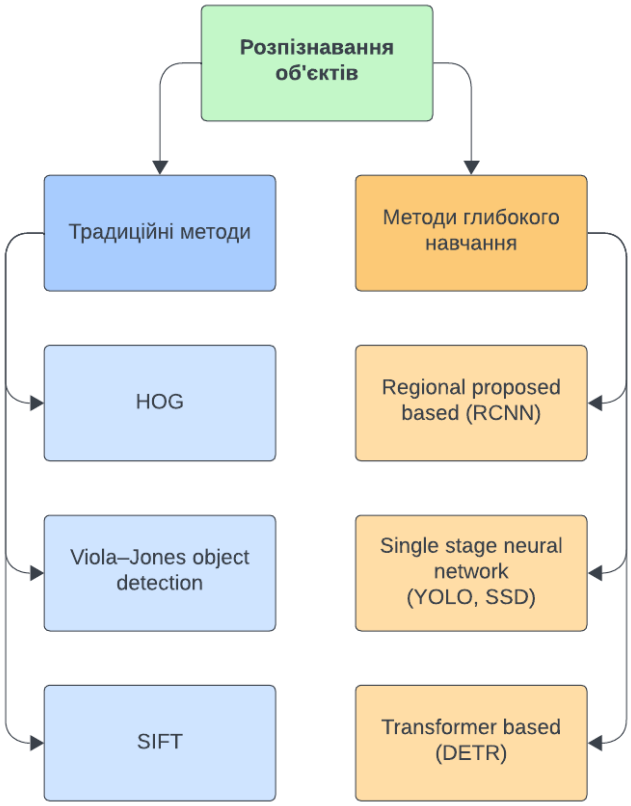

Рисунок 1.7 — Класифікація методів розпізнавання об’єктів

Методи сімейства R-CNN, такі як R-CNN, Fast R-CNN і Faster R-CNN, базуються на використанні регіональних пропозицій для детекції об’єктів. Основна ідея полягає у виділенні потенційних областей, які можуть містити об’єкти, і подальшому аналізі цих областей за допомогою згорткових нейронних мереж (CNN). Початковий алгоритм R-CNN спочатку генерує близько 2000 регіонів, які потім окремо подаються в CNN, що робить його повільним. У вдосконалених версіях Fast R-CNN і Faster R-CNN значно оптимізовано процес обробки, зменшено час виконання за рахунок спільного

використання ознак і введення мережі для регіональних пропозицій (Region Proposal Network, RPN).

Ці підходи є потужними для задач з високою точністю розпізнавання об’єктів, оскільки кожен регіон аналізується детально. Проте вони вимагають великих обчислювальних ресурсів і не завжди підходять для реального часу через їхню багатостадійність. У задачах, де важлива висока точність, наприклад у медичній діагностиці чи аналізі супутникових зображень, R-CNN часто використовуються завдяки своїй здатності враховувати деталі.

Методи на кшталт YOLO (You Only Look Once) і SSD (Single Shot Multibox Detector) пропонують альтернативу багатостадійним підходам, виконуючи детекцію об’єктів за один прохід через модель. У YOLO вся детекція виконується одночасно для всього зображення, поділеного на сітку, де кожна клітинка відповідає за прогнозування об’єктів, які в ній знаходяться. Цей підхід забезпечує швидкість і придатність для задач у реальному часі, таких як системи відеоспостереження чи автономне керування.

SSD, у свою чергу, використовує багаторівневу структуру, прогнозуючи об’єкти на різних масштабах, що дозволяє краще обробляти об’єкти різного розміру. Обидва методи характеризуються швидкістю і простотою, однак часто поступаються багатостадійним моделям у точності, особливо при роботі з невеликими або складними об’єктами. Їх застосування оптимальне для завдань, де час обробки є критичним фактором.

DETR (DEtection TRansformer) представляє новий підхід до детекції об’єктів, базуючись на архітектурі трансформерів. У цьому методі відмовляються від традиційних операцій з регіональними пропозиціями або багатошаровими ознаками, натомість використовуються механізми уваги для прямого прогнозування позицій і категорій об’єктів. Це дозволяє спрощувати процес навчання, оскільки немає необхідності у використанні додаткових компонентів, таких як NMS (non-maximum suppression) чи спеціальних сіток для регіональних пропозицій.

Основна перевага DETR полягає в його здатності працювати з складними сценами завдяки глобальному контексту уваги, що робить його дуже точним. Однак трансформери є обчислювально важкими, тому DETR часто має більший час навчання і вимоги до ресурсів порівняно з традиційними підходами. Цей метод підходить для задач, де важлива здатність обробляти складні зображення, наприклад, в автоматизованій аналітиці чи при побудові систем доповненої реальності.

Окрім того, відсутність інтерпретованості глибоких моделей може робити їх менш придатними для деяких застосувань, де важлива зрозумілість прийнятих рішень чи їхня релевантність. Методи глибокого навчання з використанням згорткових нейронних мереж, демонструють високі досягнення в розпізнаванні об'єктів у просторі та виявляють помірну ефективність завдяки здатності автоматично вивчати репрезентації об'єктів із великої кількості даних.

1.4 Аналіз сучасних систем атоматичного керування рухомим об’єктом

Для того, щоб зрозуміти та проаналізувати переваги чи недоліки нової розробки у порівнянні з уже існуючими на ринку, необхідно визначити їх основні функціональні можливості та складові характеристик. Наведемо деякі схожі існуючі аналоги.

Прикладом рухомого об'єкта, який керується нейромережею, є дрон DJI Tello EDU, створений для освітніх цілей та практичного навчання в галузі штучного інтелекту і програмування [13]. DJI Tello EDU має легку і компактну конструкцію, що надає можливість маневрувати та навчатися програмуванню польоту, як зображено на рисунку 1.8.

Рисунок 1.8 — Навчальний дрон DJI Tello EDU

Основні функціональні особливості DJI Tello EDU:

— вбудована камера для збору зображень, які можуть використовуватися для комп'ютерного зору та розпізнавання об'єктів під час польоту;

— підтримка нейромережевих моделей, завантажених з TensorFlow, що дозволяє здійснювати автономне керування, виконувати завдання розпізнавання об'єктів та відстеження цілей;

— високопродуктивний 14-ядерний процесор Intel Movidius, оптимізований для роботи з AI-завданнями та обробки даних в режимі реального часу; — SDK і API для Python, Scratch і Swift, які відкривають можливість програмування складних сценаріїв для автономних польотів, взаємодії з іншими дронами та побудови моделей нейромереж;

— функція стабілізації для плавного керування та стійкості під час виконання програмованих польотів;

— час польоту до 13 хвилин на одному заряді, що дозволяє проводити достатню кількість експериментів і тестів з різними моделями керування.

DJI Tello EDU є надійним та функціональним інструментом для вивчення основ нейромереж і штучного інтелекту, який дозволяє користувачам на практиці реалізовувати алгоритми машинного навчання та автономного керування в умовах реального часу.

Компактний дизайн, набір датчиків і розширені функції роблять пристрій універсальним і надійним. Зважаючи на ціну пристрою, що сягає 8500 гривень, прилад входить у середній ціновий сегмент.

Робот-собака Spot від Boston Dynamics — це рухомий об'єкт, керований нейромережею, який розроблено для виконання різних завдань у складних умовах [14]. Цей робот має чотириногу конструкцію, яка дозволяє йому легко адаптуватися до різних поверхонь і стабільно пересуватися по нерівній місцевості, як зображено на рисунку 1.9.

Рисунок 1.9 — Робот-собака Spot

Основні функціональні особливості Spot:

— система комп’ютерного зору, оснащена камерами та сенсорами, що дають можливість точно орієнтуватися у просторі й уникати перешкод;

— нейромережева система керування рухом, яка аналізує середовище і приймає рішення в режимі реального часу, дозволяючи роботу адаптувати рухи для стабільного пересування;

— потужні електричні приводи та стабілізаційна система, що дозволяють Spot долати складні перешкоди, такі як сходи, ухили та нерівні поверхні;

— підтримка режиму автономного патрулювання, що дозволяє роботу слідувати за заданим маршрутом і виконувати обстеження території або об'єктів;

— можливість інтеграції з іншими пристроями та платформами, що дозволяє використовувати Spot для збору даних, наприклад, на промислових об’єктах;

— акумуляторна батарея, що забезпечує до 90 хвилин автономної роботи, після чого робот автоматично прямує до зарядної станції.

Spot є потужним автономним роботом, що використовує передові технології комп'ютерного зору та нейромереж для орієнтації в складному середовищі. Його слабким місцем є відносно обмежений час автономної роботи, а також необхідність у періодичному обслуговуванні та калібруванні для безперебійної роботи, а через зависоку ціну, яка сягає 3000000 гривень, такий пристрій не є доступним для звичайних користувачів.

AWS DeepRacer — це автономний гоночний автомобіль, розроблений Amazon Web Services для навчання та практичного застосування машинного навчання, зокрема методів підкріплення [15]. Цей пристрій має компактну і маневрову конструкцію, оснащену системою камер і потужним апаратним забезпеченням, що дозволяє ефективно тренувати та тестувати алгоритми автономного керування, як зображено на рисунку 1.10.

Рисунок 1.10 — Автономний гоночний автомобіль AWS DeepRacer

Основні функціональні особливості:

— інтегрована камера, яка забезпечує реальне зображення траси для аналізу та прийняття рішень під час їзди;

— високопродуктивний процесор Intel Atom, що підтримує складні обчислення для машинного навчання в режимі реального часу;

— підтримка AWS RoboMaker і SageMaker, що дозволяє користувачам налаштовувати, тестувати та вдосконалювати моделі керування за допомогою симуляцій в хмарі;

— технологія підкріплювального навчання (reinforcement learning), яка допомагає автомобілю самостійно вчитися уникати перешкод, залишатися на траєкторії та максимально швидко проходити трасу;

— модуль Wi-Fi для віддаленого контролю та завантаження моделей керування;

— акумуляторна батарея, яка забезпечує до 2 годин автономної роботи, що дозволяє проводити достатню кількість тестів на трасі.

AWS DeepRacer є якісним та надійним інструментом для навчання та тестування функцій автономного водіння, адже в ньому органічно поєднано можливості машинного навчання та гнучкі параметри налаштування, а з урахуванням можливостей тестування цей пристрій може стати основою для досліджень. Його основним обмеженням є необхідність належної підготовки моделей і технічного супроводу для досягнення ефективного результату на треках.

Порівняємо аналоги з розробленим приладом за такими параметрами:

— мобільність;

— автономність;

— навігація;

— безпека;

— розміри;

— ціна.

Порівняння основних характеристик з існуючими аналогами наведено на таблиці 1.1.

Таблиця 1.1 — Порівняння основних характеристик з існуючими аналогами

Параметри / Аналоги	Мобільність	Автономність	Навігація	Безпека	Розміри	Ціна	Результат
Бали							
DJI Tello EDU	4	4	4	3	5	4	24
Spot	5	5	5	4	3	2	24
AWS DeepRacer	5	4	3	2	5	4	23
Власна розробка	5	5	4	5	5	5	29

В результаті аналізу було виявлено, що розроблений прилад набрав найбільшу кількість балів по наведеним вище характеристикам. Основними перевагами для цього стали безпека у використанні та достатньо низька ціна.

2 ВИБІР ТА АНАЛІЗ КОМПОНЕНТІВ СТРУКТУРИ СИСТЕМИ КЕРУВАННЯ РУХОМИМ ОБ'ЄКТОМ

Теорія автоматичного керування — це міждисциплінарний розділ науки, що поєднує інженерію та математику для створення та аналізу систем, здатних функціонувати й саморегулюватися без безпосереднього втручання людини. Її сфера застосувань надзвичайно широка: від побутової техніки до складних автономних транспортних засобів. Головною метою автоматичного керування є розробка алгоритмів, що забезпечують системам досягнення необхідного рівня продуктивності, стабільності та надійності [16].

Центральною ідеєю теорії є концепція зворотного зв'язку (рис. 2.1), яка передбачає постійний моніторинг вихідного сигналу системи та його порівняння із заданим референтним значенням. Відхилення, що виникають, стають основою для коригувальних дій, які спрямовані на утримання системи в бажаному стані. Така структура дозволяє системам адаптуватися до змін у зовнішньому середовищі або внутрішніх параметрах, забезпечуючи певну стабільність.

Розробка систем автоматичного керування передбачає кілька етапів. Спершу формулюються вимоги та критерії ефективності, які можуть включати точність, швидкість реагування, стійкість до зовнішніх збурень. Наступним не менш важливим кроком є моделювання динаміки системи: створення математичних моделей, таких як диференціальні рівняння чи функції передачі, що описують її реакцію на зовнішні чинники.

Обравши стратегію керування, інженери впроваджують методи від найпростіших, як-от PID-регулятори, до сучасних підходів, зокрема прогнозного керування моделлю чи адаптивних методів. Вибір методу визначається складністю об'єкта, технічними вимогами та доступними ресурсами.

На завершальному етапі розробляється алгоритм керування, який перетворює теоретичні принципи в конкретні математичні формули й

інструкції для генерування керуючих сигналів. Алгоритм повинен бути стійким, ефективним та адаптивним до змінних умов експлуатації, забезпечуючи досягнення поставлених цілей.

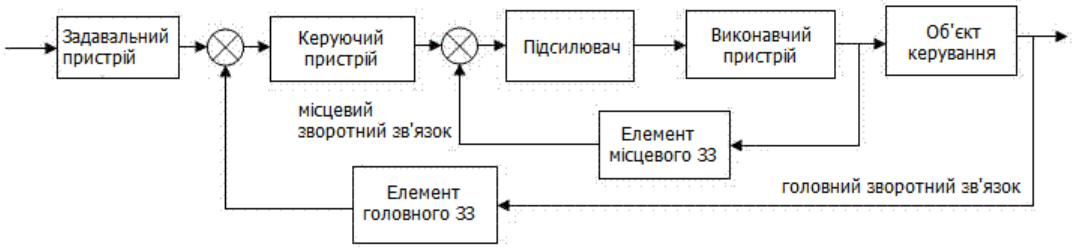

Рисунок 2.1 — Типова схема системи автоматичного керування

Після розробки алгоритму керування наступним етапом є його інтеграція у відповідну апаратну чи програмну платформу. У цьому кроці відбувається програмування мікроконтролерів, створення керуючого програмного забезпечення або конфігурація програмованих логічних контролерів (ПЛК). Реалізація також охоплює налаштування датчиків і виконавчих механізмів, що забезпечують необхідний зворотний зв’язок і точне управління системою.

Після впровадження системи керування проводиться її тестування та налаштування для перевірки відповідності заданим критеріям ефективності, які встановлює розробник. Зазвичай такий процес включає аналіз реакції системи на різні вхідні сигнали та збурення, оптимізацію параметрів керування та коригування алгоритму для досягнення оптимальної продуктивності, яка задовольняє умови. Також у разі необхідності виконується додаткове вдосконалення системи.

Таким чином, теорія автоматичного керування формує основу для створення автономних систем, дозволяючи детально досліджувати їхню динаміку, допомагає обрати ефективні стратегії керування, розробити відповідні алгоритми та успішно впроваджувати їх у реальні сценарії

експулатації. Такі підходи забезпечують точність, а найголовоніше адаптивність і високу продуктивність системи.

2.1 Модель зв’язку між компонентами системи

Вибір архітектури master/slave для систем керування рухомими об’єктами обумовлений її здатністю ефективно поєднувати централізоване управління з перевагами розподіленої обробки даних, адже цей підхід забезпечує не лише стабільну й продуктивну роботу системи, але й відкриває широкі можливості для її адаптації та подальшої модернізації.

Однією з ключових переваг цієї архітектури є її гнучкість, адже головний пристрій може динамічно перерозподіляти завдання між підлеглими елементами в реальному часі [17]. Таке рішення надає змогу системі оперативно реагувати на зміну зовнішніх умов, таких як от зміна траєкторії руху, виявлення перешкод чи зміна експлуатаційних параметрів.

Додатковою перевагою є масштабованість, завдяки якій виникає певний модульний підхід в якому нові функції або пристрої можна інтегрувати в систему без значних змін у її архітектурі, що призводить до зниження витрат на доопрацювання та спрощує процес розширення функціональності.

Архітектура також підтримує розподілену обробку даних, яка дозволяє підлеглим пристроям виконувати локальні обчислення (див. рис. 2.2). Це значно знижує навантаження на головний пристрій і підвищує швидкість реакції системи — критично важливий фактор у керуванні рухомими об’єктами, де кожна затримка може вплинути на точність і стабільність.

Крім того, безпека системи покращується завдяки можливості головного пристрою моніторити стан підлеглих, виявляти аномалії та активувати механізми резервування. Наприклад, у разі виходу з ладу одного підлеглого пристрою завдання можуть бути перерозподілені між іншими, зберігаючи функціональність системи..

Ще однією суттєвою перевагою є уніфікованість інтерфейсів, що спрощує інтеграцію зовнішніх компонентів і налаштування нових елементів системи для зосередження на основних задачах, мінімізуючи складність підключення додаткових пристроїв.

Однак успішна реалізація цієї архітектури потребує ретельного проектування протоколів зв'язку, врахування затримок передачі даних, уникнення перевантаження головного пристрою та дублювання ключових функцій для підвищення надійності. Тому має виникати певний баланс між централізацією та розподілом функцій і це може бути складним, але і критично важливим для запобігання неочікуваних та неочевидних помилок.

Для реалізації моделі master/slave було обрано бібліотеку WiringPi, яка надає необхідний функціонал. WiringPi — це бібліотека для мови C, яка забезпечує роботу з GPIO-портами на одноплатних комп'ютерах, таких як Raspberry Pi. Вона пропонує зручний і зрозумілий інтерфейс для управління введенням/виведенням, що робить її оптимальним вибором для побудови системи керування [18].

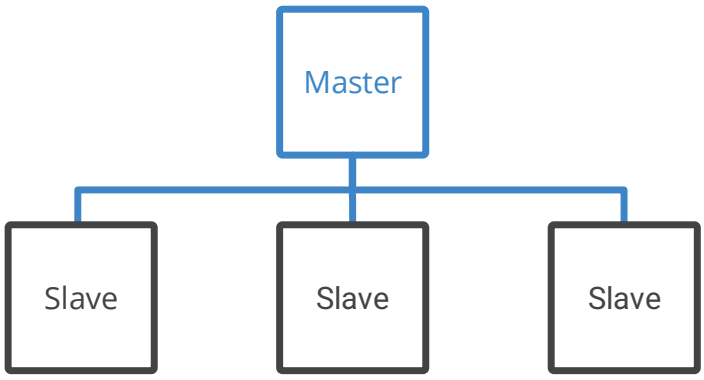

Рисунок 2.2 — Структурна схема технології master/slave

WiringPi забезпечує функції для налаштування пінів, запису та зчитування значень, роботи зі змінними операційними режимами (наприклад, вхід, вихід, PWM), а також для взаємодії з пристроями через шину I2C, SPI та UART [19].

Використання WiringPi для master/slave комунікації має свої переваги. Основні з них:

— інтуїтивно зрозумілий інтерфейс, який значно спрощує роботу з GPIO-портами та налаштування комунікації між пристроями.;

— розширений функціонал для взаємодії з GPIO-портами та периферійними шинами (I2C, SPI), що дозволяє зручно реалізувати механізми master/slave;

— сумісність із широким спектром пристроїв, що полегшує інтеграцію різних сенсорів, актуаторів та інших компонентів, які є невід’ємною частиною системи..

Як результат, обрана модель комунікації між складовими системи забезпечить низку переваг у розробці ієрархії структури та дозволить створити базу для подальших удосконалень пристрою.

2.2 Головний пристрій керування

Для головного пристрою слід обрати одноплатний комп'ютер, який буде відповідати за контроль усіх підпорядкованих елементів системи, оптимальне розподілення ресурсів і надаватиме зручний інтерфейс для конфігурації.

Одноплатні комп'ютери Raspberry Pi є компактними і доступними пристроями, що забезпечують достатній набір функцій для підключення та управління зовнішніми електронними компонентами, датчиками, двигунами чи світлодіодами тощо. Зокрема Raspberry Pi є універсальним для різноманітних застосувань, зокрема для проектів у сферах домашньої автоматизації, робототехніки та Інтернету речей (IoT).

Також плати Raspberry Pi оснащені портами USB, Ethernet, а також виходами HDMI, що дозволяє підключати різноманітні периферійні пристрої та дисплеї. Однією з важливих переваг комп'ютерів Raspberry Pi є їх сумісність з різними операційними системами, такими як Raspbian (тепер Raspberry Pi OS) і Ubuntu, що дозволяє користувачам обирати найбільш підходящий дистрибутив для своїх потреб і працювати у знайомому та

зручному середовищі. Додатково, різні моделі Raspberry Pi мають різні рівні обчислювальної потужності і пам'яті і в залженості від обмежень можна обрати потрібні параметри.

Наприклад, Raspberry Pi 4 Model B, випущений у 2019 році, забезпечує суттєве підвищення продуктивності завдяки чотирьохядерному процесору ARM Cortex-A72, до 8 ГБ оперативної пам'яті та підтримці 4K відео. Однією з особливостей є висока продуктивність, що робить його ідеальним для більш вимогливих завдань (див. рис. 2.3). Серед інших помітних моделей Raspberry Pi варто зазначити Raspberry Pi Zero, який є дуже компактним і економічно ефективним варіантом, тому для легкого та простого проекту це є ідеальним варіантом.

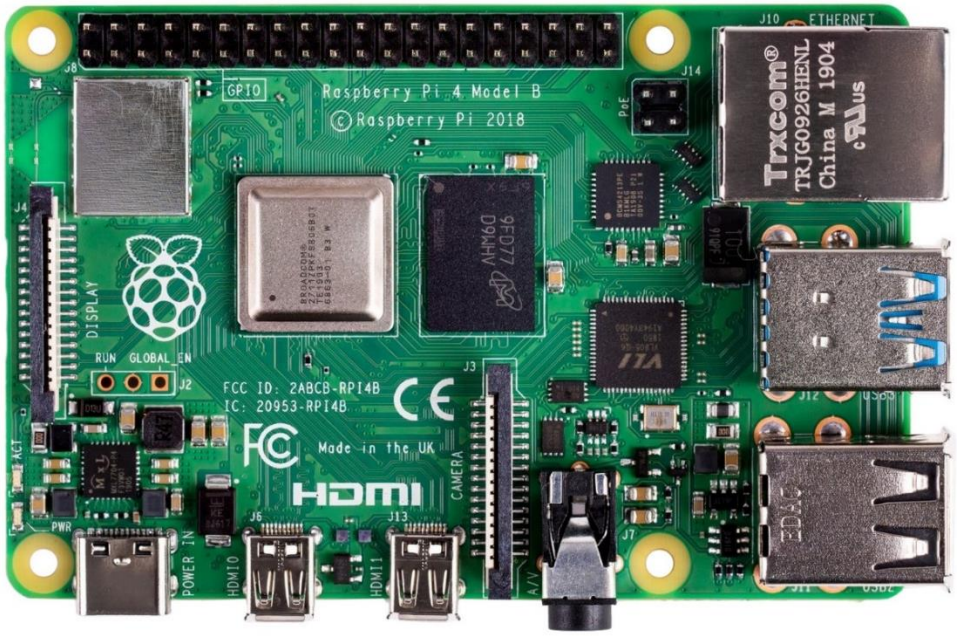

Рисунок 2.3 — Компоновка Raspberry Pi 4 Model B

Raspberry Pi 3 Model B є популярним вибором завдяки вбудованим можливостям Wi-Fi та Bluetooth та мають достатній функціонал, який також є в інших моделях.

Для програмування комп'ютера Raspberry Pi використовується інтегроване середовище розробки Geany (див. рис. 2.4), що має відкритий вихідний код та є безкоштовним. Geany розроблений для того, щоб

забезпечити легке та ефективне середовище для кодування з обмеженим набором функцій для розробки програмного забезпечення та зручний інтерфейс, який полегшує навігацію по проекту.

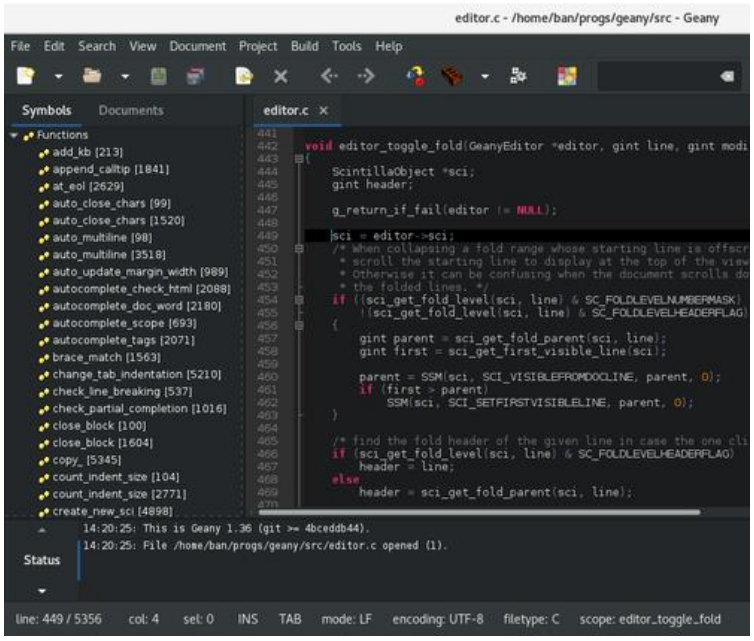

Рисунок 2.4 — Інтерфейс середовища розробки Geany

Geany підтримує кілька мов програмування, таких як C, C++, Python, Java, PHP та інші. Редактор забезпечує підсвічування синтаксису, згортання коду, автозавершення та розумні відступи, що сприяє підвищенню продуктивності та поліпшенню читабельності коду.

Однією з важливих переваг Geany є його швидкість і ефективність при редагуванні та компіляції коду. Він займає невеликий обсяг пам'яті та не вимагає великих системних ресурсів, що робить його ідеальним для пристроїв з обмеженими технічними характеристиками. Додатково Geany пропонує прості інструменти і настроювані команди збирання для роботи з різними компіляторами.

Крім того, Geany забезпечує потужні функції пошуку та заміни, функції керування проектами, можливість працювати з віддаленими файлами через FTP та має сумісність з операційними системами Windows, macOS і Linux.

Враховуючи зазначені переваги, Geany є доцільним варіантом для створення програм на базі одноплатного комп’ютера Raspberry Pi.

Для віддаленого підключення до Raspberry Pi використано VNC Viewer [20]. VNC Viewer є програмою для віддаленого доступу, яка дозволяє підключатися до Raspberry Pi і керувати з комп'ютера за допомогою протоколу VNC (Virtual Network Computing), як зображено на рисунку 2.5.

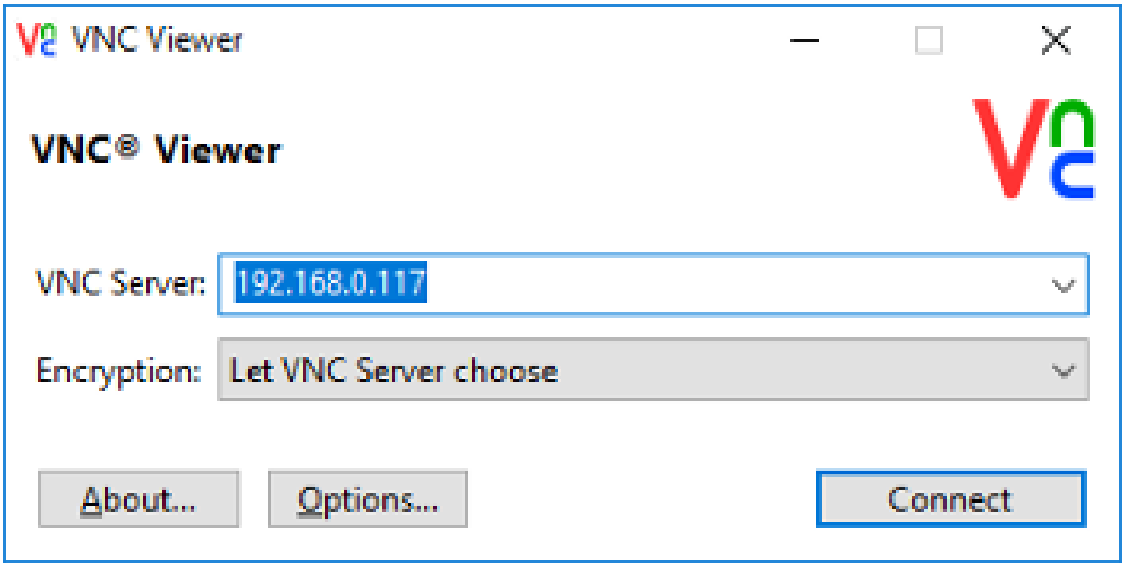

Рисунок 2.5 — Інтерфейс VNC Viewer

Ось деякі переваги використання VNC Viewer для доступу до Raspberry Pi через комп'ютер:

— дозволяє підключатися до Raspberry Pi з будь-якого комп'ютера, що знаходиться в тій же мережі або навіть через Інтернет та отримати доступ до Raspberry Pi навіть з віддаленої локації.

— доступний на різних платформах, включаючи Windows, macOS, Linux і мобільні пристрої на iOS та Android.

— налаштування VNC Viewer для підключення до Raspberry Pi є досить простим процесом. Після налаштування можна швидко встановити з'єднання і почати працювати з Raspberry Pi через комп'ютер.

— надає різноманітні функції, такі як можливість передавати файли між Raspberry Pi та комп'ютером, масштабування та розміщення екрану,

можливість керувати клавіатурою та мишею, а також підтримку шифрування для безпеки.

Отже VNC Viewer є вдалим вибором у якості інструменту для запуску програми на головному пристрою

2.3 Підпорядкований пристрій керування

Для реалізації функцій підпорядкованого пристрою у системі було обрано мікроконтролер, який відповідатиме керуючому пристрою та здійснюватиме контроль над певними модулями системи.

Сімейство мікроконтролерів Arduino пропонує широкий вибір плат, що поєднують вбудований мікроконтролер та необхідну периферію для створення електронних проєктів. Моделі цього сімейства базуються на мікроконтролерах AVR або ARM і адаптовані до різних застосувань. Серед найпопулярніших моделей — Arduino Uno, Arduino Mega, Arduino Nano та Arduino Due.

Arduino Uno є однією з найбільш поширених моделей, яка користується популярністю серед початківців, проте часто використовується і професіоналами. Завдяки збалансованому поєднанню простоти й функціональності, ця плата підходить для багатьох базових та стартових проєктів. Arduino Uno забезпечує достатню кількість цифрових і аналогових входів та виходів для роботи з більшістю простих електронних компонентів, а її обчислювальна потужність і обсяг пам’яті дозволяють виконувати нескладні та маленькі завдання.

Arduino Mega призначена для більш складних проєктів, що потребують великої кількості вхідних/вихідних контактів та збільшеного обсягу пам’яті (див. рис. 2.6). Завдяки цим характеристикам Mega ідеально підходить для розробки систем автоматизації, робототехнічних проєктів чи пристроїв збору даних. Її можливості дозволяють працювати з великою кількістю датчиків і виконавчих механізмів, а також виконувати складніші програми.

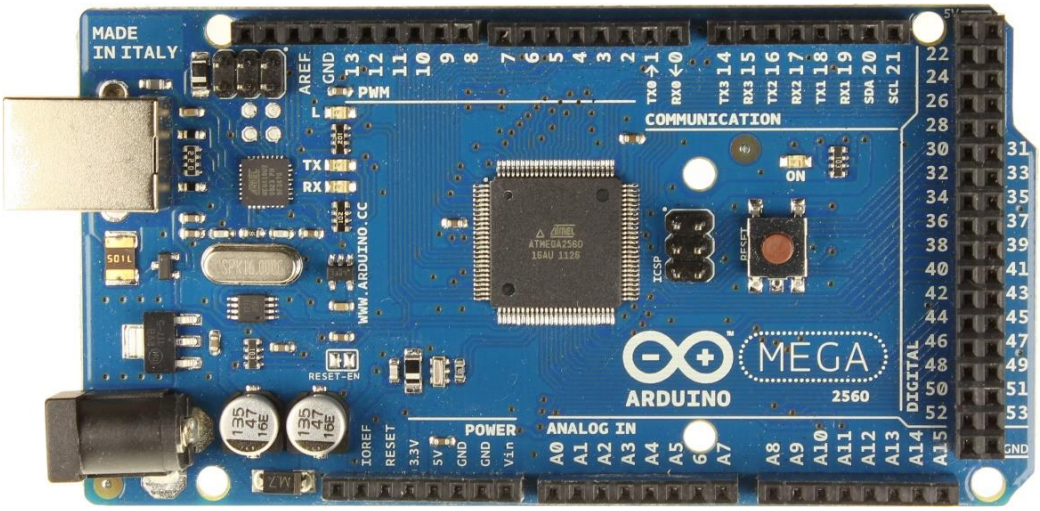

Рисунок 2.6 — Компоновка Arduino Mega

Arduino Nano є компактною та економічно доступною платою, яка оптимально підходить для проєктів із обмеженим простором або тих, що потребують малого форм-фактора (див. рис. 2.7). Незважаючи на свої скромні розміри, Nano зберігає широкий функціонал, притаманний більшим моделям Arduino. Ця плата часто застосовується у носимих пристроях, компактних IoT рішеннях та вбудованих системах, де важливі мініатюрність і ефективність.

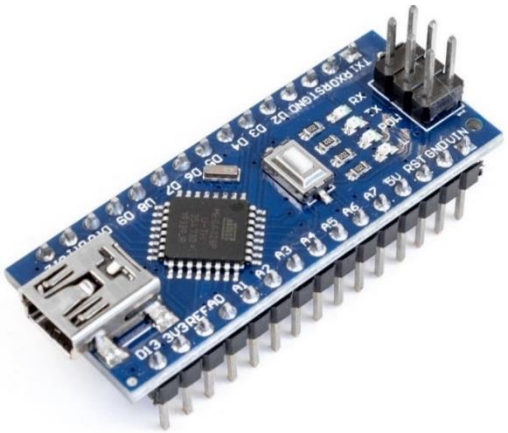

Рисунок 2.7 — Компоновка Arduino Nano

Arduino Due вирізняється серед інших моделей завдяки використанню 32-розрядного процесора ARM Cortex-M3, який забезпечує значно вищу обчислювальну потужність і розширений обсяг пам’яті. Дані характеристики

роблять його ідеальним вибором для застосувань, що потребують надскладних обчислень, зокрема обробки аудіосигналів, складних робототехнічних систем і роботи з великими обсягами даних. Окрім цього, Due оснащений додатковими функціями, такими як цифро-аналогові перетворювачі (DAC) і підтримка вдосконалених протоколів зв'язку, що відкриває можливості для створення складних і багатофункціональних систем.

Для програмування та завантаження коду на платформу Arduino, було використано Arduino IDE, що показано на рисунку 2.8.

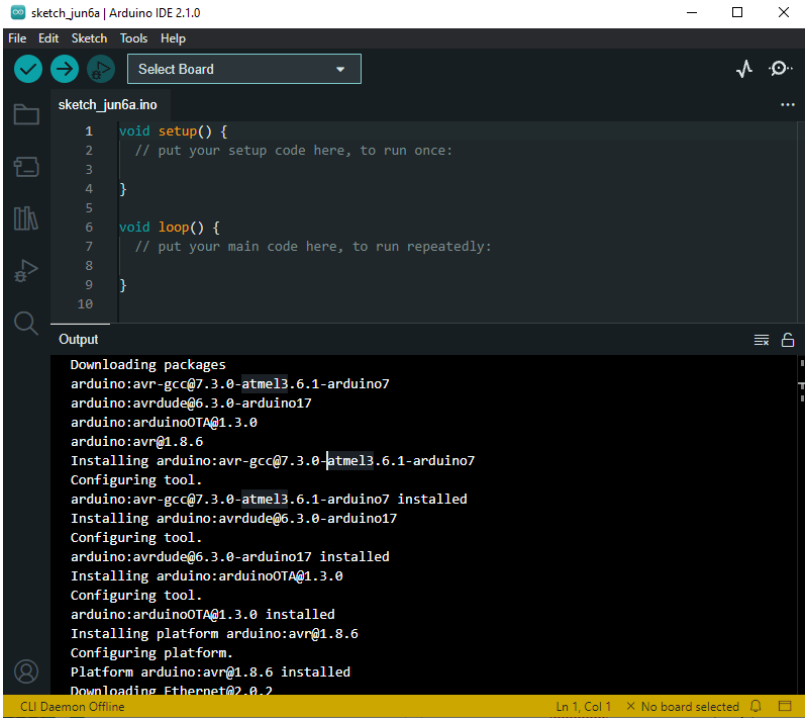

Рисунок 2.8 — Інтерфейс середовища розробки Arduino IDE

Arduino IDE надає зручне середовище розробки, яке дозволяє створювати програми без попереднього досвіду в програмуванні мікроконтролерів завдяки інтуїтивно зрозумілому інтерфейсу, простому функціоналу та завдяки великій кількості документації через підтримку спільноти.

Основні особливості включають:

— підсвічування синтаксису для різних мов програмування, таких як C та C++;

— виявлення помилок під час компіляції;

— завантаження коду на плату Arduino через USB-порт або інше з'єднання;

— можливість переглядати вивід програми або надсилати команди за допомогою монітора порту;

— вбудована бібліотека, що містить різні функції та методи для спрощення розробки, включаючи бібліотеки власної розробки користувачів;

— підтримка більшості моделей Arduino;

— відлагодження програмного коду за допомогою точок зупинки і виведення значень змінних для аналізу.

Можна сказати, що дане середовище значно спрощує написання коду для мікроконтролерів Arduino та надає низку корисних особливостей під час користування та розробки прогармного забезпечення

2.4 Модуль відеокамери для введення зображень

Для отримання інформації про простір навколо рухомого об’єкта необхідно вибрати модуль відеокамери, який забезпечить систему керування вхідними зображеннями та дозволить їх використовувати для аналізу та прийняття рішень. Без камери будь-які самокеровані дії не є можливими, тому цей елемент є ключовим.

Модулі камер для Raspberry Pi дозволяють знімати зображення та записувати відео з самої плати Raspberry Pi, додаючи візуальне джерело іформації на яку можна опиратися безспосердньо в побудованих алгоритмах. Два найпоширеніші модулі камер для Raspberry Pi — це Raspberry Pi Camera Module V2 і Raspberry Pi High-Quality Camera.

Raspberry Pi Camera Module V2 є популярним вибором для фото- та відеозйомки загального призначення. Модуль оснащено 8-мегапіксельним датчиком зображення Sony IMX219 та підтримує максимальну роздільну

здатність 3280 x 2464 пікселів для фотографій та запису відео 1080p зі швидкістю 30 кадрів на секунду (див. рис. 2.9). Модуль підключається до плати Raspberry Pi через роз’єм CSI (Camera Serial Interface) за допомогою стрічкового кабелю.

Raspberry Pi High-Quality Camera є вдосконаленим варіантом, що пропонує кращу якість зображення та більше налаштувань . Модуль оснащений 12,3-мегапіксельним датчиком Sony IMX477, який дозволяє знімати зображення та відео з вищою роздільною здатністю. Такий модуль підтримує змінні об’єктиви з байонетом C або CS, що дозволяє змінювати фокусну відстань. Raspberry Pi High-Quality Camera підключається до Raspberry Pi через стрічковий кабель і підтримує ручне налаштування фокусу та захоплення зображень у форматі RAW.

Рисунок 2.9 — Модуль Raspberry Pi High-Quality Camera

Raspberry Pi High-Quality Camera підходить для застосувань де потрібна висока якість зображення, таких я фотографування, професійна обробка зображень або машинний зір. Камера забезпечує вищу роздільну здатність та підтримує змінні об’єктиви і таким чином можна налаштовувати фокусну відстань для досягнення оптимальних результатів, а це в свою чергу

робить її оптимальним варіантом для сценаріїв де потрібно отримати високу якість зображення..

Для обробки зображень з відеокамери Raspberry Pi Camera було вирішено використовувати бібліотеку OpenCV. OpenCV (Open Source Computer Vision Library) — це бібліотека комп'ютерного зору та обробки зображень з відкритим кодом. Бібліотека надає розробникам широкий набір функцій і алгоритмів для роботи з зображеннями та відео, щоб вирішувати завдання виявлення об'єктів, розпізнавання зображень і допомагає маніпулювати зображеннями. Приклад обробки зображень можна побачити на рисунку 2.10.

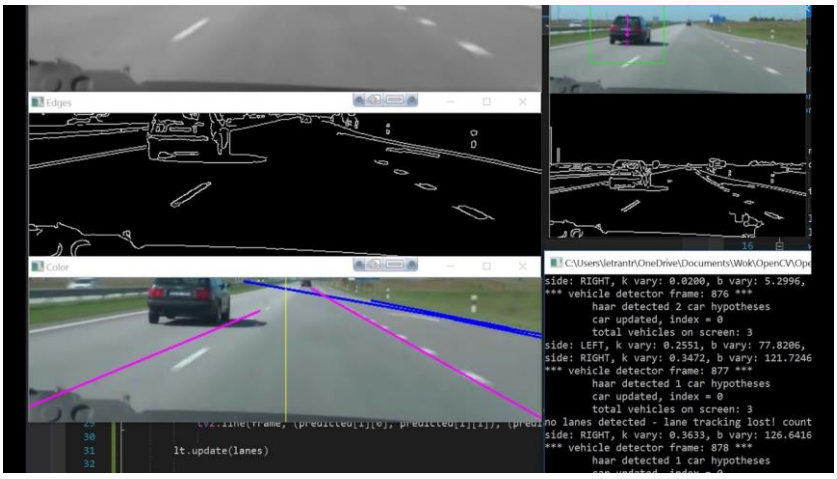

Рисунок 2.10 — Приклад обробки зображень бібліотекою OpenCV

Коли йдеться про керування відеокамерою Raspberry Pi за допомогою OpenCV, бібліотека забезпечує зручний інтерфейс для захоплення та обробки відеокадрів або зображень для подальшого використання. OpenCV підтримує доступ до модуля камери Raspberry Pi через API Video4Linux (V4L), що дозволяє легко записувати потокове відео в реальному часі або захоплювати окремі кадри.

Щоб використовувати OpenCV із віодеокамерою Raspberry Pi, спочатку потрібно встановити OpenCV на комп'ютер Raspberry Pi. Після встановлення

стане можливим імпортувати бібліотеку OpenCV у свій сценарій Python чи C++ і використовувати її функції для взаємодії з камерою.

OpenCV може керувати виводом камери в режимі реального часу, застосовувати фільтри, виконувати виявлення або розпізнавання об’єктів тощо. Також включена можливість комбінування OpenCV з іншими бібліотеками та фреймворками для створення складних програм комп’ютерного зору, що потребують більшого набору функцій для аналізу зображень.

Бібліотека OpenCV широко використовується для великої кількості задач по обробці візуальних даних, а наявні в ній функції задовольняють вимоги до системи керування рухомим об’єктом.

2.5 Архітектура каскадної моделі Хаара

Каскадна модель Хаара — це алгоритм, створений для розпізнавання об'єктів у зображеннях або відео. Його основою є каскад класифікаторів, які виконують аналіз зображення за допомогою особливих ознак (Haar-like features), розроблених на основі ідей, запозичених з роботи людського зору. Архітектура моделі базується на багатошаровій структурі, де кожен рівень (каскад) виконує специфічну задачу — від початкового грубого відсіювання непотрібних областей до точного визначення об'єкта.

Головною перевагою моделі є її швидкодія. Алгоритм реалізований так, щоб швидко відкидати регіони, які з високою ймовірністю не містять об'єкта, що шукається. Це досягається завдяки використанню простих порогових функцій, побудованих на основі сум інтегрального зображення (див. рис. 2.11). Завдяки цьому алгоритм є ефективним навіть на пристроях із обмеженими обчислювальними ресурсами.

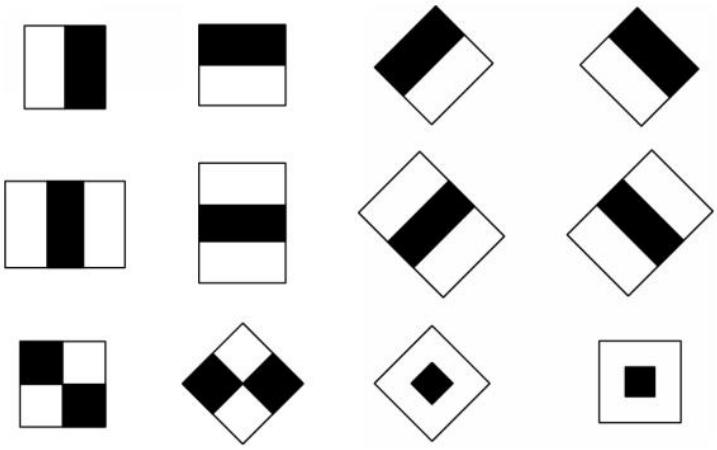

Рисунок 2.11 — Приклад розташування пікселів, які розпізнає модель

Для розпізнавання об'єктів, зокрема на дорозі (наприклад, транспортних засобів, пішоходів чи дорожніх знаків тощо), каскадна модель дозволяє швидко ідентифікувати важливі області за ознаками контрасту, форми, за порядком розташування пікселів або базуючись на сотні інших особливостей, якими може бути наповнений отриманий кадр (див. рис. 2.12). Однак точність залежить від якості підготовки даних і навчання, адже їхня кількість підвищує шанс на успіх. Використання каскадної архітектури також знижує кількість хибних позитивних результатів, оскільки кожен наступний рівень класифікатора уточнює аналіз.

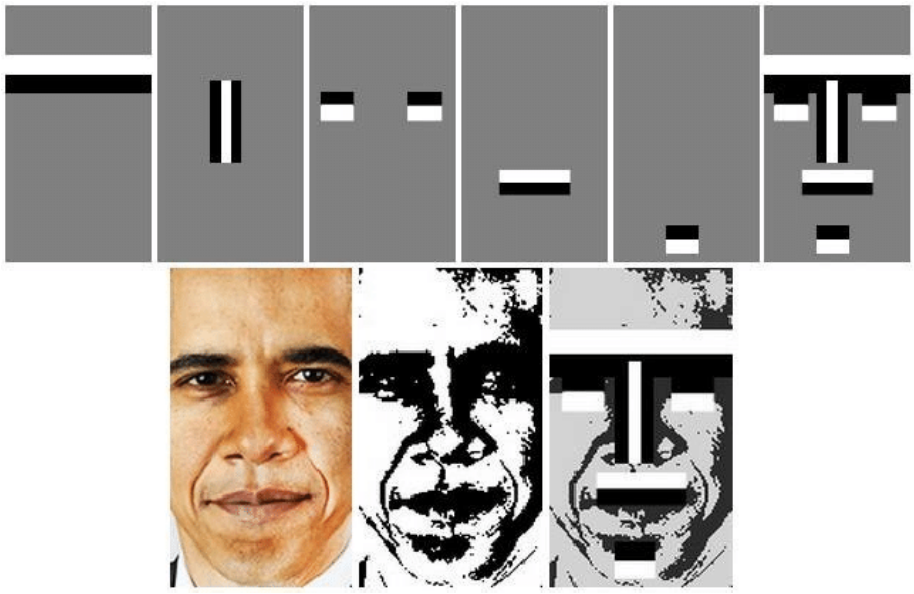

Рисунок 2.12 — Приклад розпізнавання об’єкту за ознаками

Процес розпізнавання проходить кілька ключових етапів (див. рис. 2.13). Спочатку відбувається підготовка зображення, що включає його перетворення в градації сірого, оскільки каскадна модель працює тільки з інтенсивністю пікселів, а кольорова інформація (RGB) є надлишковою для його роботи. Готове сіре зображення піддається обчисленню інтегрального зображення. Інтегральне зображення представляє кожну точку як суму значень пікселів у прямокутнику від початку координат до цієї точки. Це дозволяє значно пришвидшити обчислення ознак, оскільки для визначення сум пікселів у будь-якому прямокутнику потрібно лише чотири операції з інтегрального зображення.

Далі розпочинається етап пошуку Хаар подібних ознак. Такі ознаки представляють собою прямокутні маски, які аналізують контраст між областями зображення. Наприклад, для розпізнавання обличчя використовуються ознаки, які перевіряють контраст між областями навколо очей і щік або між носом і щоками. Різні форми цих масок (прямокутники горизонтальної, вертикальної, діагональної орієнтації на фреймі) використовуються для різних типів об’єктів в залежності від того, наскільки комплексним і точно визначеним є бажаний об’єкт. Кожна маска застосовується до всіх можливих положень і масштабів у межах зображення, щоб якомога точніше визначити об’єкт.

На наступному етапі ці ознаки проходять через каскад бінарних класифікаторів, які складаються з багатьох простих рішень, званих слабкими класифікаторами. Кожен слабкий класифікатор аналізує одну Haar-like ознаку і повертає рішення: чи є ймовірність наявності об’єкта у вікні. Каскад працює за принципом жорсткої послідовності: якщо вікно "провалюється" на якомусь із рівнів каскаду, подальша обробка цього вікна припиняється, що значно пришвидшує процес розпізнавання. Лише ті вікна, які проходять усі рівні каскаду, вважаються такими, що містять об'єкт як зображено на рисунку 2.13.

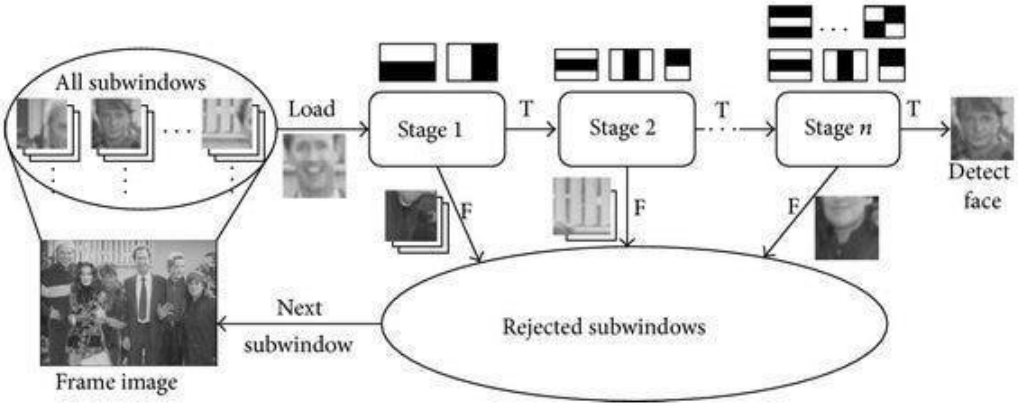

Рисунок 2.13 — Етапи розпізнавання об’єкту каскадною моделлю

Масштабування зображення є також важливим, оскільки об’єкти можуть бути різного розміру, зображення аналізується в багатьох масштабах. Часто це вирішується зменшенням вікна ознак або за допомогою простого обрізання кадру для зміщення фокусу та для охоплення бажаної зони. Таким чином, алгоритм може виявляти об'єкти незалежно від їхнього розміру на зображенні.

Результати всіх вікон, що пройшли через каскад, об'єднуються і якщо кілька вікон вказують на один і той самий об'єкт, застосовується метод групування вікон (non-maximum suppression), який залишає лише одне вікно із найвищою ймовірністю та запобігає виникненню дублювання детекцій як таких.

Заключним етапом є візуалізація або подальша обробка результатів. Координати вікон, що містять об'єкти, передаються в систему для відображення (наприклад, обведення рамками виявлених об'єктів чи інший спосіб їх виділення) або для використання в інших задачах, таких як трекінг чи класифікація.

Для використання вищезгаданої моделі, використано Cascade Trainer GUI. Cascade Trainer GUI є графічним інтерфейсом для навчання каскадних класифікаторів за допомогою бібліотеки OpenCV (див. рис. 2.14). Його основна мета — полегшити процес створення моделей для виявлення об’єктів на основі методу каскадів Хаара або бінарних моделей LBP (Local Binary Patterns). Цей інструмент, створений для автоматизації складних аспектів

навчання, забезпечує доступність методології як для дослідників, так і для практиків.

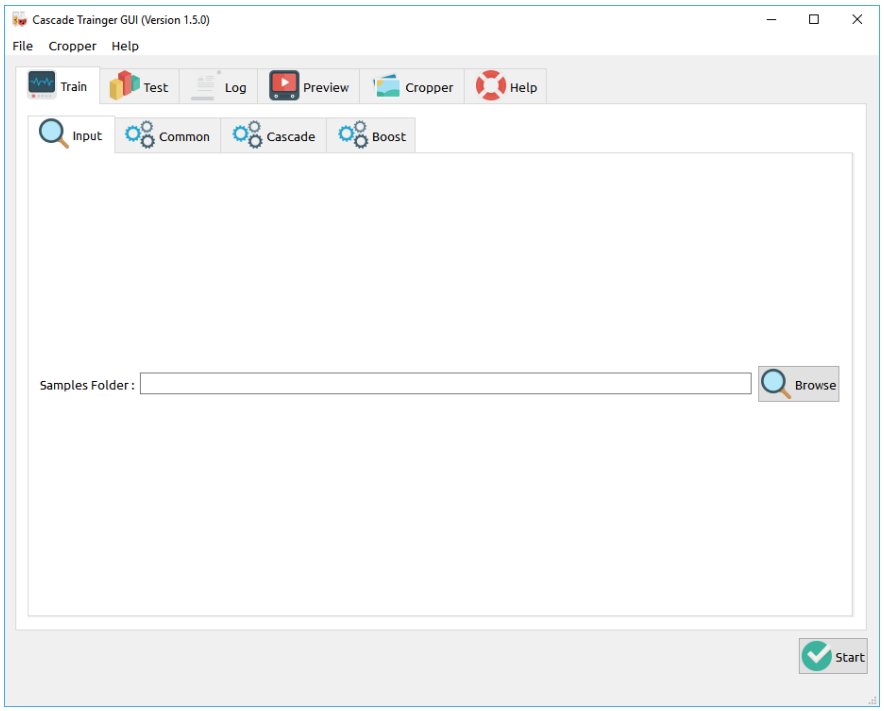

Рисунок 2.14 — Інтерфейс Cascade Trainer GUI

Однією з головних переваг Cascade Trainer GUI є спрощення конфігурації параметрів навчання, які в іншому випадку потребували б ручного налаштування через термінал з багатьма інструкціями щодо параметрів. Інтерфейс дозволяє користувачу легко налаштовувати такі параметри, як кількість позитивних і негативних зображень, кількість етапів навчання та порогові значення. Такий процес є прозорим та очевидним, адже можна одразу відслідкувати усе, що відбувається в програмі.

Наступною перевагою є інтеграція інструментів для підготовки даних. Cascade Trainer GUI дозволяє обробляти позитивні та негативні зображення прямо в програмі, виконуючи такі дії, як обрізання, нормалізація або генерація множинних варіацій для підвищення стійкості моделі. Завдяки цьому процес збору та попередньої обробки даних стає значно швидшим та менш трудомістким, адже не потрібно надто багато часу власноруч обробляти вхідні дані.

2.6 Алгоритми роботи системи

В минулих підпунктах розділу було визначено компоненти системи керування рухомим об’єктом, тому переходимо до розробки алгоритмів роботи, що описуватимуть послідовність дій системи під час обробки зображень та руху.

Управління рухомим об’єктом мікроконтролерною системою здійснюється у двовимірному просторі, нехтуючи параметром висоти. Для виконання цього завдання, розділимо його на декілька логічних блоків:

— основний алгоритм роботи системи;

— отримання та обробка зображень з відеокамери;

— розпізнавання зовнішніх об’єктів на отриманому зображенні;

— процес прийняття рішення та рух системи.

Пуск системи здійснюється за рахунок підключення джерела живлення до головного пристрою, підключення через VNC Viewer, та запуску основного алгоритму, який починає збирати дані з відеокамери, аналізуючи своє положення на дорозі з розміткою, але не здійснюючи переміщення зі стартового положення.

Далі необхідно подати живлення на підпорядкований пристрій, що керуватиме моторами, щоб почати рух системи. Через комунікацію між двома пристроями керування буде здійснюватися передавання перехоплених кадрів з відеокамери. Після обробки зображень, головний пристрій надає результат для прийняття рішення підпорядкованому пристрою, як повинна змінювати положення система відносно поточного розташування у просторі (див. рис. 2.15). Для отримання та обробки зображень використовуємо функції бібліотеки OpenCV.

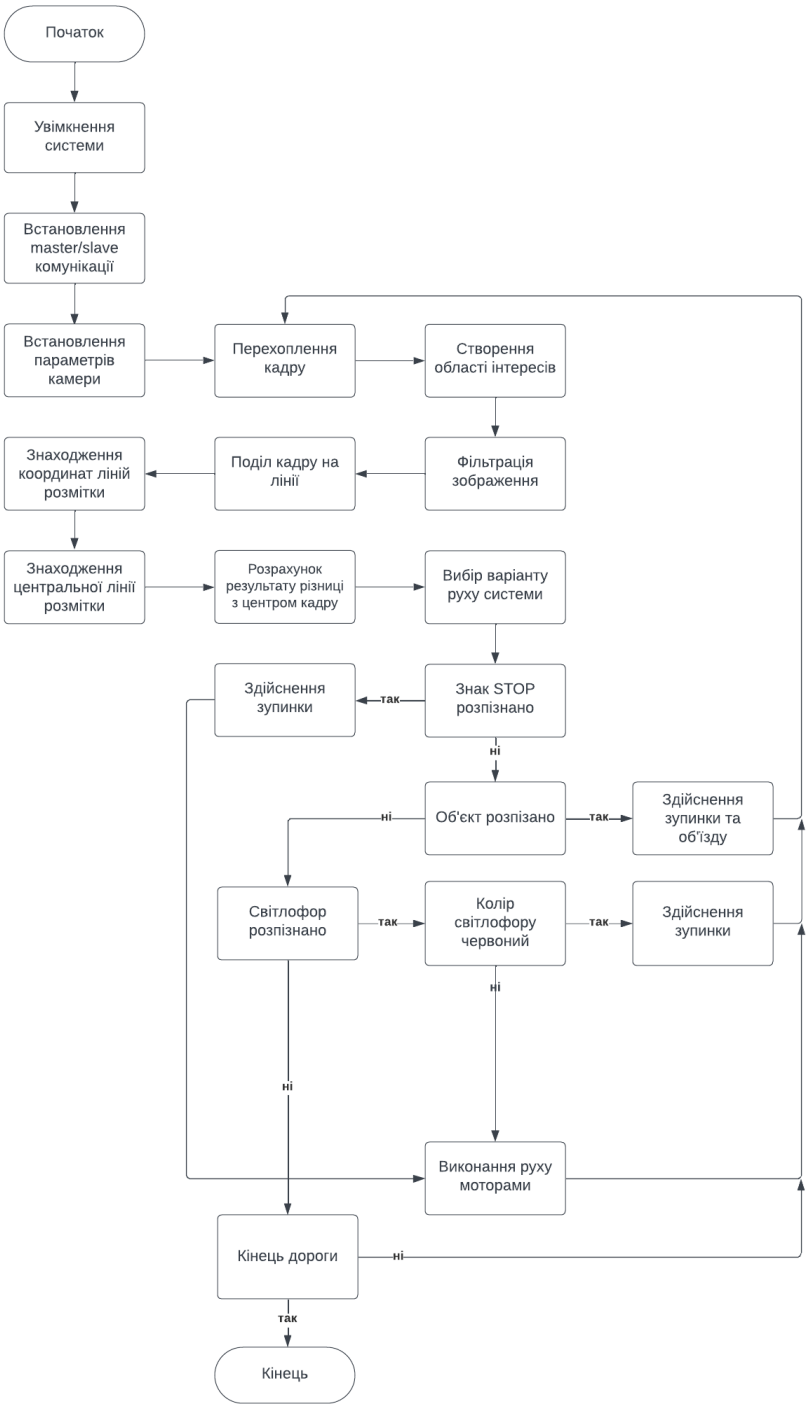

Рисунок 2.15 — Блок-схема основного алгоритму роботи системи

Опишемо послідовність виконання основного алгоритму:

— увімкнення системи;

— встановлення master/slave комунікації;

— встановлення основних параметрів камери;

— перехоплення кадру;

— створення області інтересів та трансформація перспективи зображення для фокусування на дорозі з розміткою;

— фільтрація зображення для зміни інтенсивності пікселів та розпізнавання границь навколо розмітки на кадрі;

— поділ кадру на лінії шириною в один піксель;

— знаходження координат ліній розмітки за допомогою пікселів, що мають більшу інтенсивність;

— визначення центральної лінії розмітки;

— розрахунок результату різниці між центром лінії розмітки та центром кадру;

— вибір варіанту руху системи, що залежить від результату обчислень;

— якщо знак STOP розпізнано – здійснюється зупинка системи на короткий період часу.

— якщо знак STOP не розпізнано – відбувається спроба розпізнати об’єкт а дорозі.

— якщо об’єкт розпізнано – відбувається зупинка на короткий період часу та здійснюється об’їзд цього об’єкту.

— якщо знак STOP та об’єкт н дорозі не розпізнано – відбувається спроба розпізнати світлофор.

— якщо світлофор розпізнано і він перебуває в режимі червоного кольору – відбувається зупинка на період часу, поки колір не буде змінено на зелений.

— в інших випадках виконується рух моторами.

2.6.1 Алгоритм обробки зображень

Отримання та обробка зображень з відеокамери починається з перехоплення кадру. Далі необхідно створити область інтересів, яка буде обмежена чотирма лініями. В цій області буде знаходитись розмітка дороги по

якій рухатиметься система. Щоб це зробити, необхідно трансформувати перспективу, як на рисунку 2.16.

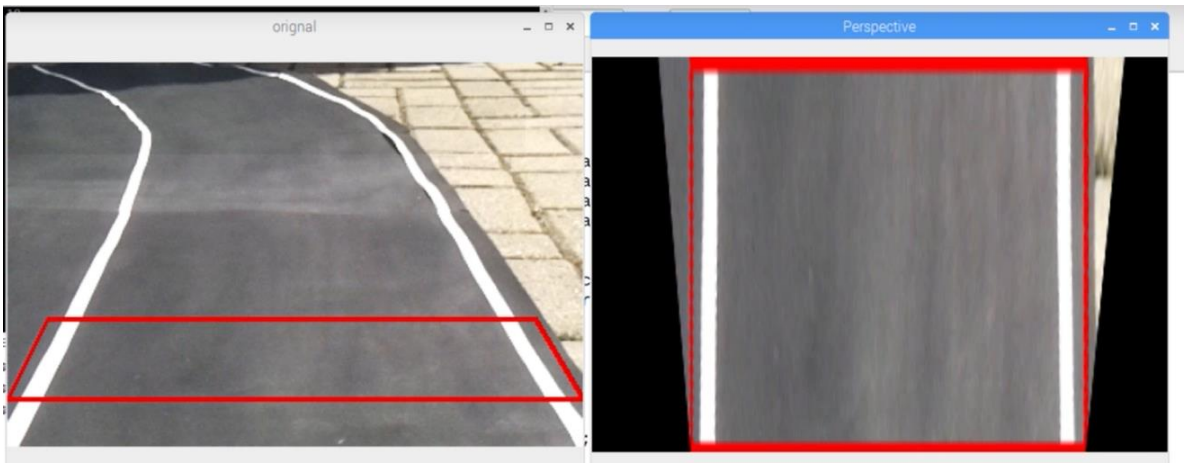

Рисунок 2.16 — Зображення перехопленого кадру ліворуч та кадру з трансформованою перспективою праворуч

Для кращого розпізнавання розмітки необхідно здійснити фільтрацію зображення, піднявши інтенсивність розмітки порівняно з дорогою та виділивши контури розмітки, як на рисунку 2.17.

Рисунок 2.17 — Зображення кадру після фільтрації

Для знаходження X координати лівої та правої ліній розмітки, ділимо кадр на лінії шириною в один піксель. Далі ділимо кадр на дві частини, переконавшись, що в кожну з них потрапила одна лінія розмітки. У кожній з цих частин, перебираючи усі лінії знаходимо піксель, який матиме максимальне значення інтенсивності (він є X координатою лінії розмітки).

Для визначення центру лінії між правою та лівою сторонами розмітки, необхідно використати їх X координати у формулі:

$$\text{LaneCenter} = \frac{(\text{RightLanePos} - \text{LeftLanePos})}{2} + \text{LeftLanePos} \qquad (2.1)$$

де LaneCenter — X координата центральної лінії розмітки;

RightLanePos — X координата правої лінії розмітки;

LeftLanePos — X координата лівої лінії розмітки.

Використовуючи обчислені позиції сторін розмітки, наносимо на кадр прямі лінії для демонстрації точності розпізнавання, як на рисунку 2.18.

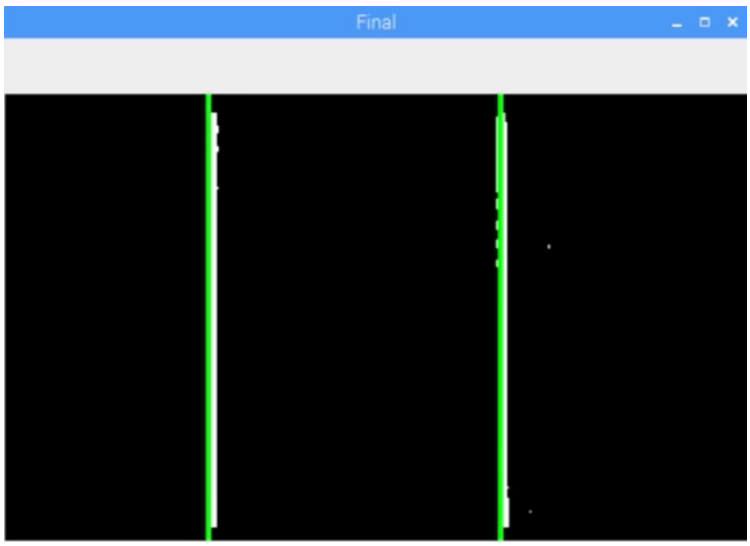

Рисунок 2.18 — Зображення кадру після знаходження координат ліній розмітки

Для розрахунку результату, що буде використаний для поворотів моторів, необхідно визначити різницю між обчисленою позицією центральної лінії та позицією центру кадру використовуємо формулу:

$$Result = LaneCenter - FrameCenter \quad (2.2)$$

де LaneCenter — X координата центральної лінії розмітки;

FrameCenter — X координата центра кадру.

Тепер наносимо обчислену центральну розмітку на зображення кадру та закінчуємо його обробку, як на рисунку 2.19.

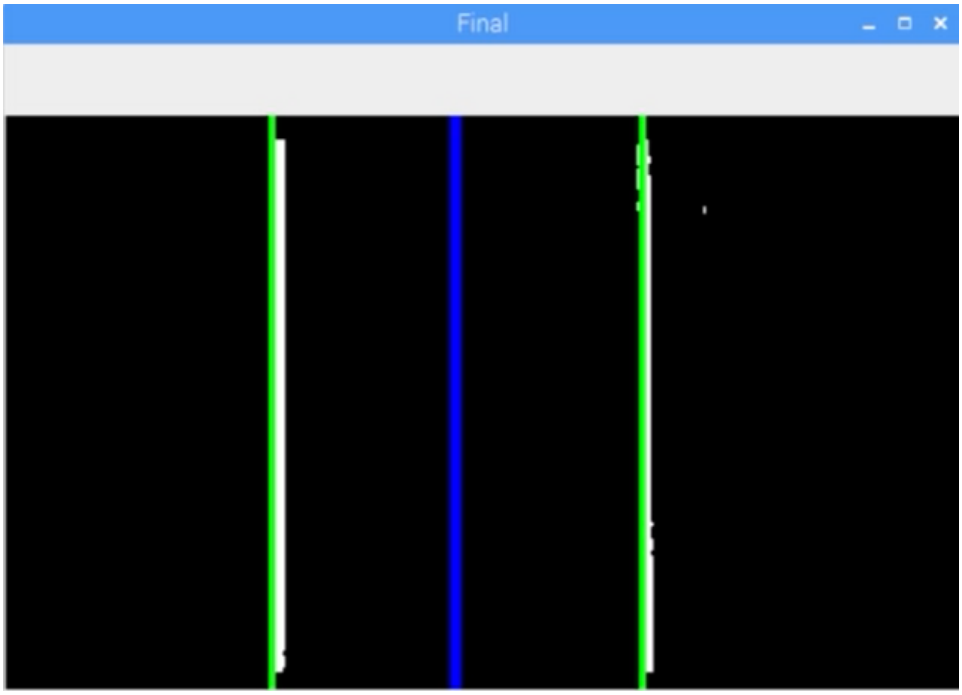

Рисунок 2.19 — Зображення кадру після знаходження центру розмітки

2.6.2 Алгоритм розпізнавання об'єктів

Для початку розпізнавання будь-яких об’єктів, необхідно завантажити каскадний файл у форматі .xml, що був натренований на позитивних та негативних прикладах об’єкта у полі зору камери. Про деталі отримання таких каскадів йдеться в третьому розділі роботи. Для прикладу об’єкту, який

розпізнається, візьмемо знак STOP, зважаючи на те, що підхід є аналогічним для розпізнавання будь-яких інших об’єктів.

Далі визначаємо область, яку потрібно аналізувати, а саме прямокутник у верхній частині кадру. Після цього обраний регіон перетворюється в градації сірого, щоб спростити подальший аналіз, а яскравість зображення вирівнюється для покращення якості обробки.

Наступним кроком в обраній області виконується пошук об’єктів, що відповідають шаблону, за допомогою алгоритму, який працює з попередньо завантаженим каскадом. Якщо було знайдено об'єкти, то для кожного з них обчислюються координати, які визначають прямокутник, що охоплює кожен знайдений знак. Цей прямокутник відображається на зображенні разом із текстовою позначкою, яка вказує, що це знак "STOP", як зображено на рисунку 2.20.

Рисунок 2.20 — Розпізнаний знак STOP на кадрі

Після цього оцінюється відстань до знаку, базуючись на ширині прямокутника, і ця інформація також відображається на зображенні у вигляді тексту. Для того щоб обчислити відстань, необхідно виміряти ширину прямокутника в пікселях за такою формулою:

$$\mathrm{RectWidth} = \mathrm{P2.x} - \mathrm{P1.x} \quad (2.3)$$

де RectWidth — це ширина прямокутника в пікселях, що охоплює знайдений об'єкт на зображенні;

P2.x — це X-координата правого нижнього кута;

P1.x — це X-координата лівого верхнього кута.

Експериментальним шляхом спочатку знаходимо ширину прямокутника на відстані 15 сантиметрів, що дорівнює 82 пікселів, а далі на відстані 30 сантиметрів знаходимо ширину прямокутника, що дорівнює 68 пікселів.

Тепер створюємо систему лінійних рівнянь, яка використовується для визначення відстані до об'єкта на основі розміру його зображення. Рівняння виглядають так:

$$15 = 82\mathrm{m} + \mathrm{c} \quad (2.4)$$

$$30 = 68\mathrm{m} + \mathrm{c} \quad (2.5)$$

де m — це масштабний коефіцієнт, який визначає, як ширина об'єкта на зображенні впливає на реальну відстань до нього;

c — це константа, що визначає зсув, коригуючи масштаб за допомогою калібрування;

Тепер можемо знайти значення m і c за допомогою методу заміни змінних. Спочатку виразимо c з першого рівняння. Для першого рівняння маємо:

$$\mathrm{c} = 15 - 82\mathrm{m}$$

Тепер підставимо це вираження для c у друге рівняння:

$$30 = 68\mathrm{m} + (15 - 82\mathrm{m})$$

$$30 = -14\mathrm{m} + 15$$

$$15 = -14\mathrm{m}$$

$$\mathrm{m} = -\frac{15}{14} \approx -1.07$$

Отримано масштабний коефіцієнт m = -1.07. Тепер підставимо знайдене значення m = -1.07 у рівняння для обчислення c:

$$\mathrm{c} = 15 - 82(-1.07)$$

$$\mathrm{c} = 15 + 87.74 = 102.74$$

Отже, значення константи c ≈ 102.74. Тепер відстань до об'єкта обчислюється за емпірично виведеною формулою:

$$\mathrm{DistanceStop} = -1.07 \cdot (\mathrm{P2.x} - \mathrm{P1.x}) + 102.597 \qquad (2.6)$$

де DistanceStop — це відстань до об’єкта в сантиметрах;

$\mathrm{P2.x} - \mathrm{P1.x}$ — це ширина прямокутника, що охоплює знайдений об'єкт на зображенні в пікселях;

Коефіцієнт 1.07 — це масштабний фактор, який визначає залежність відстані до об'єкта від його видимої ширини на зображенні;

Константа 102.597 — це зсув, який компенсує масштаб і позицію, забезпечуючи, що результат формули відповідає реальній відстані.

Далі встановлюємо бажані границі, при яких система має зупинитися. У цьому випадку було обрано проміжок між 5 та 20 сантиметрів, щоб система зупинялася не надто близько і не надто далеко від знаку. Після закінчення розпізнавання та короткої зупинки — рух продовжується.

2.6.3 Алгоритм руху системи

Після отримання результату обробки, необхідно обрати варіант руху системи. Можливі випадки розміщено на блок-схемі, яку продемонстровано на рисунку 2.21.

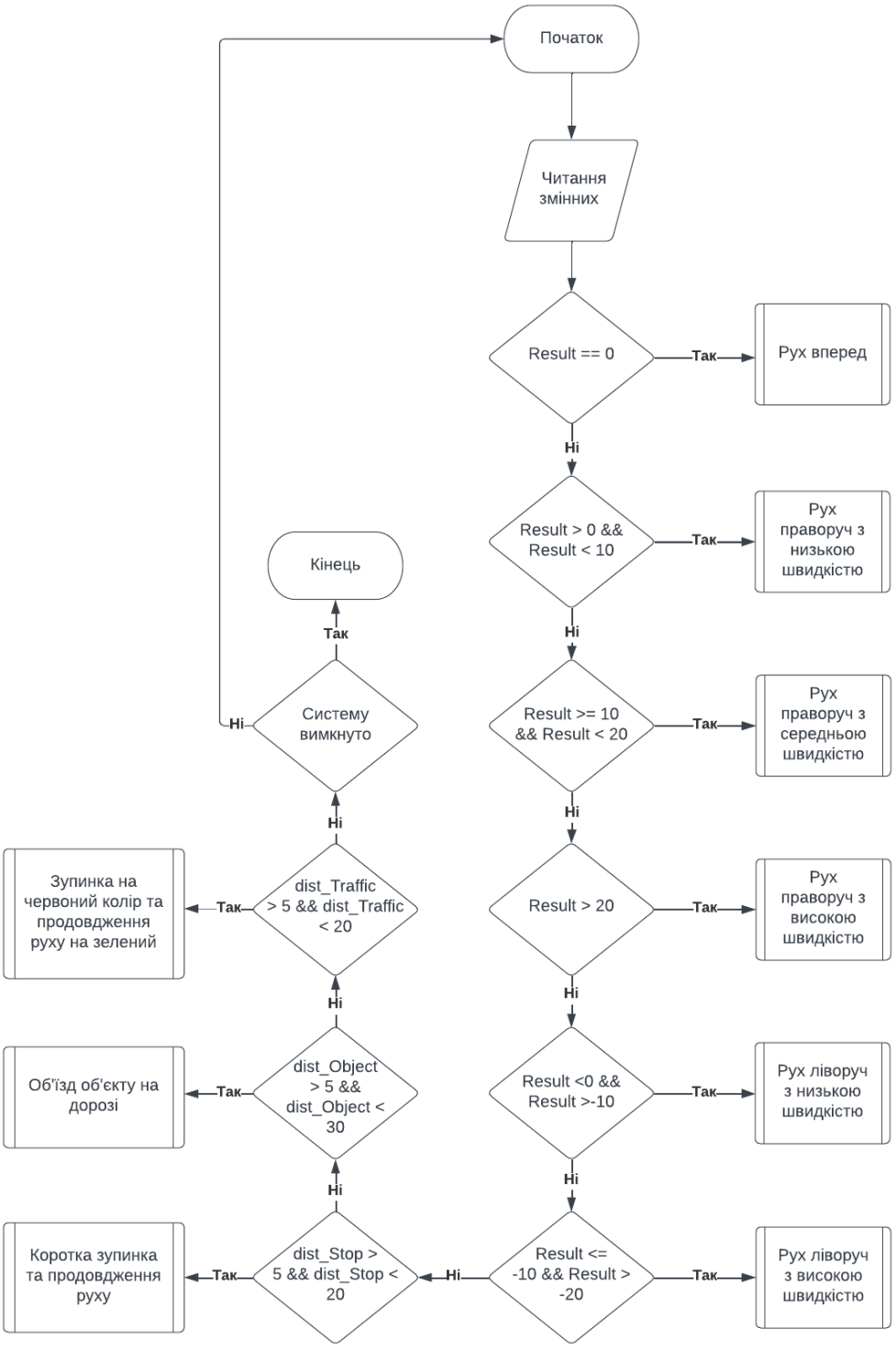

Рисунок 2.21 — Блок-схема алгоритму руху системи

Розробка алгоритмів завершена і можна переходити до її програмної реалізації.

3 РОЗРОБКА СИСТЕМИ КЕРУВАННЯ РУХОМИМ ОБ'ЄКТОМ

Посилаючись на минулий розділ, в якості головного пристрою керування було обрано платформу на основі одноплатного комп'ютеру Raspberry Pi, а в якості підпорядкованого — платформу мікроконтролеру Arduino (див. рис. 3.1). Для обробки зображень використаємо модуль відеокамери Raspberry Pi Camera, що дозволить обробляти зображення з навколишнього середовища та передавати їх в автоматизовану систему для подальшого прийняття рішень стосовно руху об'єкту.

Для створення руху системи в двовимірному просторі, необхідно використати двигуни постійного струму. Далі обираємо драйвер двигунів, що матиме достатню напругу для їх живлення. У якості джерела живлення усієї системи використаємо акумулятор ємністю в 10 Ah, що покриває усі витрати електроенергії системою. Враховуючи, що двигуни можуть спричиняти перепади напруги, створимо модуль стабілізації, що включає конденсатор, який накопичуватиме заряд і не допустить вимикання головного пристрою керування.

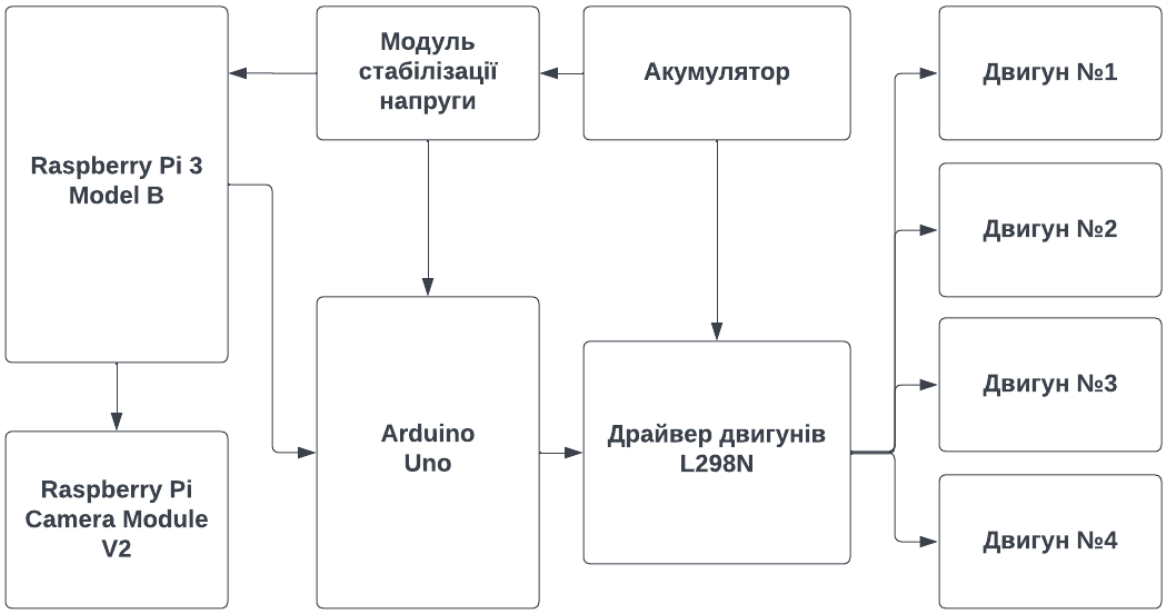

Рисунок 3.1 — Структурна схема системи керування

3.1 Вибір складових системи

Проаналізувавши усі варіанти мікроконтролерів Arduino було зроблено висновок, що у нашому випадку Arduino Uno (див. рис. 3.2) ідеально підходить у якості підпорядкованого пристрою керування для розробки невеликих за масштабом систем.

Рисунок 3.2 — Компоновка Arduino Uno

Основні параметри:

— робоча напруга – 5 В;

— вхідна напруга (рекомендовано) – від 7 до 12 В;

— вхідна напруга (максимальна) – від 6 до 20 В;

— цифрові входи/виходи – 14 (6 з яких можуть використовуватися як виходи ШІМ);

— аналогові входи – 6;

— постійний струм через вхід/вихід – 40 мА;

— постійний струм для виведення – 3.3 В при 50 мА;

— флеш-пам'ять – 32 Кб, з яких 0,5 Кб використовуються для завантажувача;

— ОЗУ – 2 Кб ;

— EEPROM – 1 Кб ;

— тактова частота – 16 Мгц.

Позначення виводів мікроконтролера Arduino Uno показано на рисунку 3.3.

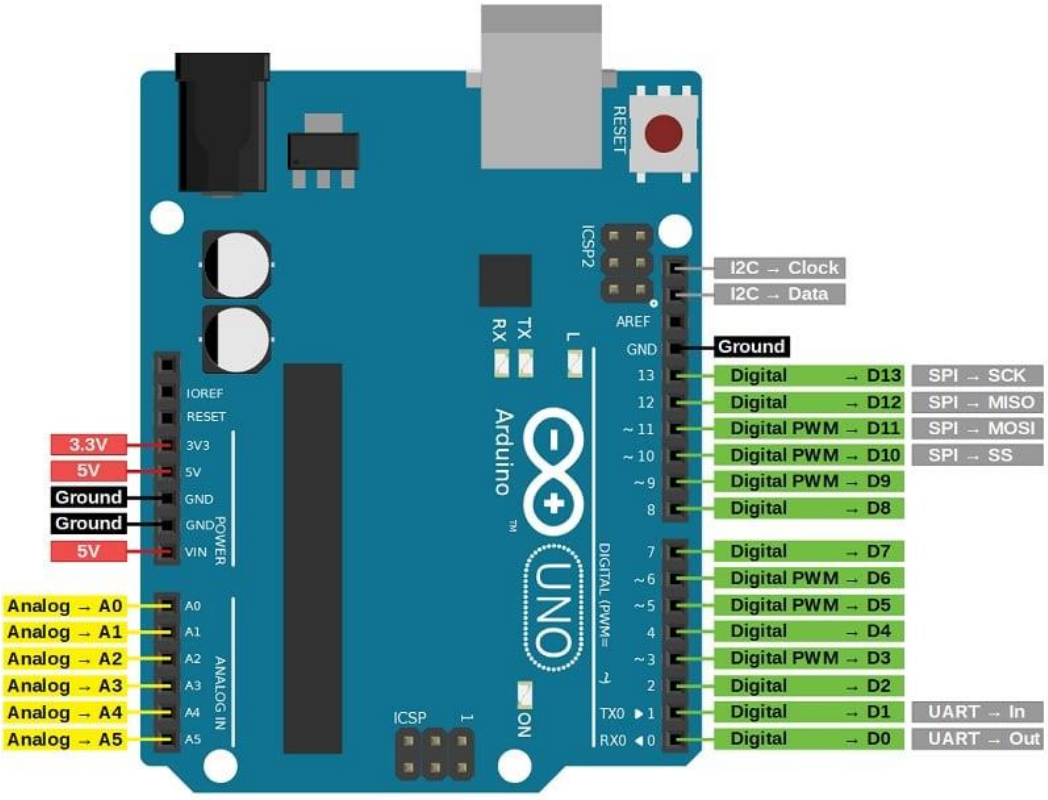

Рисунок 3.3 — Призначення виводів Arduino Uno

Відмінними особливостями мікроконтролера Arduino Uno є :

— 8–розрядний високопродуктивний AVR мікроконтролер з малим споживанням;

— прогресивна RISC архітектура;

— 130 високопродуктивних команд, більшість з яких виконується за один такт;

— наявність 32 8–розрядних робочих регістра загального призначення та регістрів керування периферією;

— повністю статична робота;

— продуктивність становить 16 MIPS при тактовій частоті 16 МГц;

— вбудований двоцикловий перемножувач з окремим вбудованим генератором;

Для системи керування рухомим об’єктом у якості головного пристрою керування найбільше підходить Raspberry Pi 3 Model B (див. рис. 3.4) за рахунок нижчої ціни з подібними функціональними можливостями порівняно з Raspberry Pi 4 Model B.

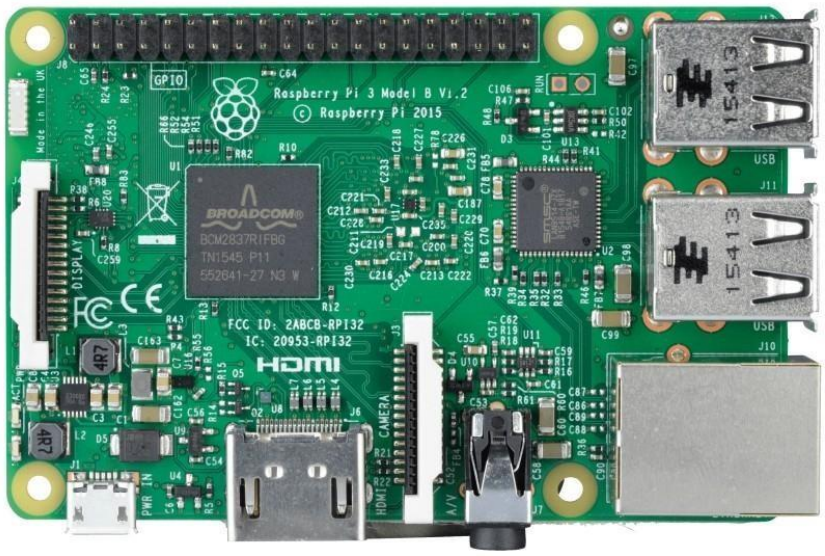

Рисунок 3.4 — Компоновка Raspberry Pi 3 Model B

Основні параметри:

— процесор оснащений Broadcom BCM2837B0 SoC (System-on-a-Chip) із чотирьохядерним процесором ARM Cortex-A53 1,2 ГГц;

— 1 ГБ оперативної пам'яті LPDDR2;

— microSD для зберігання операційної системи, програм і даних;

— вбудований Wi-Fi (802.11n) і Bluetooth 4.2;

— чотири порти USB 2.0;

— порт HDMI, який підтримує відеовихід Full HD (1080p);

— аудіороз'єм 3,5 мм для підключення навушників або колонок;

— 40-контактний роз’єм GPIO (введення/виведення загального призначення);

— живлення через micro USB зі входом 5 В постійного струму.

Позначення виводів мікроконтролера Raspberry Pi Model B показано на рисунку 3.5.

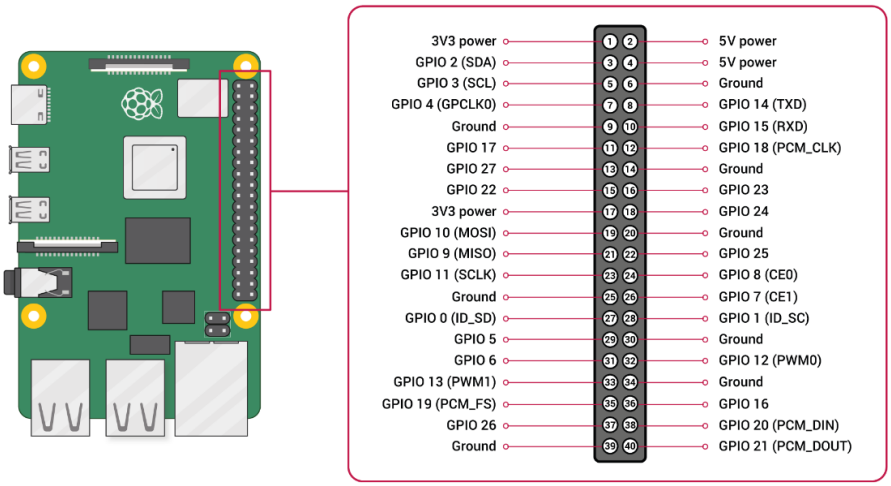

Рисунок 3.5 — Найменування виводів Raspberry Pi 3 Model B

Для керування моторами скористаємося драйвером моторів L298N (див. рис. 3.6). Цей драйвер призначений для керування швидкістю та напрямком двигунів постійного струму та крокових двигунів.

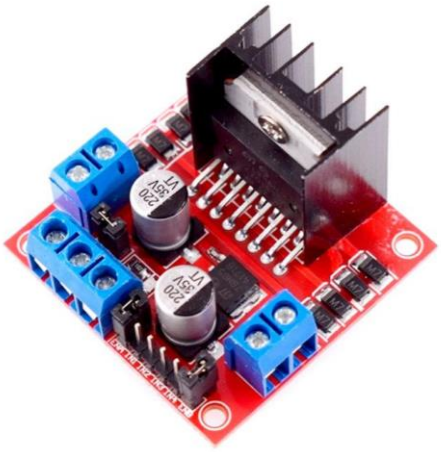

Рисунок 3.6 — Компоновка L298N

До основних параметрів L298N відносяться:

— максимальний струм до 2А на канал;

— діапазон напруг живлення від 5 до 35 вольт;

— окремі керуючі входи для кожного каналу двигуна, що дозволяє працювати з різними логічними рівнями (TTL, CMOS тощо);

— містить такі вбудовані функції захисту, як захист від перегріву, захисту від перевантаження по струму та блокування від низької напруги;

— містить радіатор для розсіювання тепла, що утворюється під час роботи, особливо при вищих струмах.

Складові даного модуля продемонстровано на рисунку 3.7.

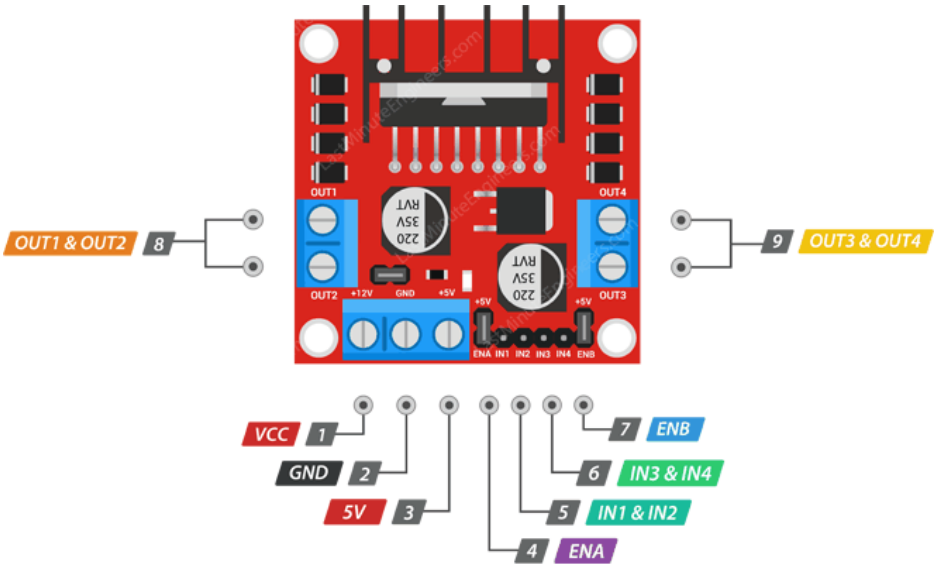

Рисунок 3.7 — Найменування виводів L298N

В якості джерела живлення було вибрано Xiaomi Mi Power Bank 3 з напругою 9В та ємністю в 10 Ah (див. рис. 3.8).

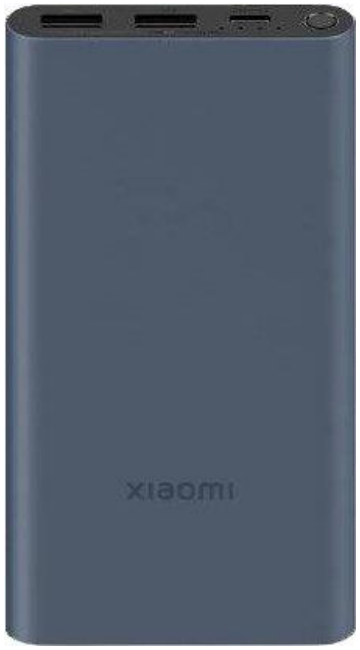

Рисунок 3.8 — Джерело живлення Xiaomi Mi Power Bank 3

Для отримання зображень навколишнього світу з високою якістю обрано модуль камери Raspberry Pi Camera Module V2 (див. рис. 3.9).

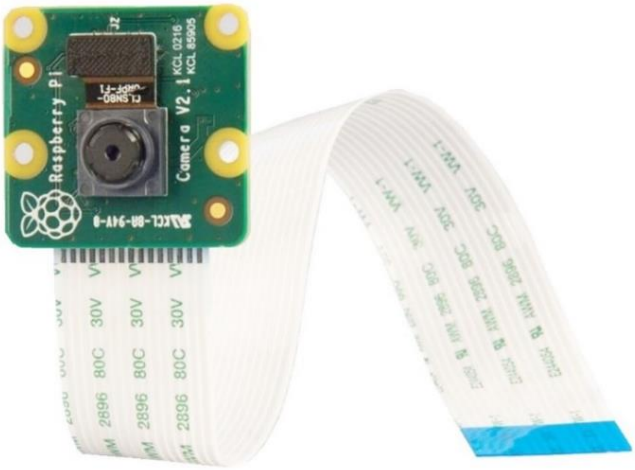

Рисунок 3.9 —Компоновка Raspberry Pi Camera Module V2

Основні характеристики Raspberry Pi Camera Module V2:

— сенсор OmniVision OV5647, який має розмір 1/4-дюйма та роздільну здатність 5 мегапікселів. Це дозволяє отримувати якісні зображення з високою деталізацією;

— здатність записувати відео з роздільною здатністю до 1080p при 30 кадрах в секунду або 720p при 60 кадрах в секунду, що дозволяє отримувати плавні і чіткі зображення в режимі реального часу;

— підключення до Raspberry Pi за допомогою гнізда CSI (Camera Serial Interface), що забезпечує швидку передачу даних і спрощує процес підключення;

— регульований фокус, що дозволяє сконфігурувати чіткість зображення відповідно до потреб користувача;

— сумісність з різними моделями Raspberry Pi, забезпечуючи просту інтеграцію з даними платформами.

Модуль стабілізації напруги включає у собі друковану плату з конденсатором ємністю в 1000 uF, резистором в 1 кОм, діодом та шпильками для підключення інших компонентів (див. рис. 3.10). Конденсатор вибрано з ємністю в 1000 uF.

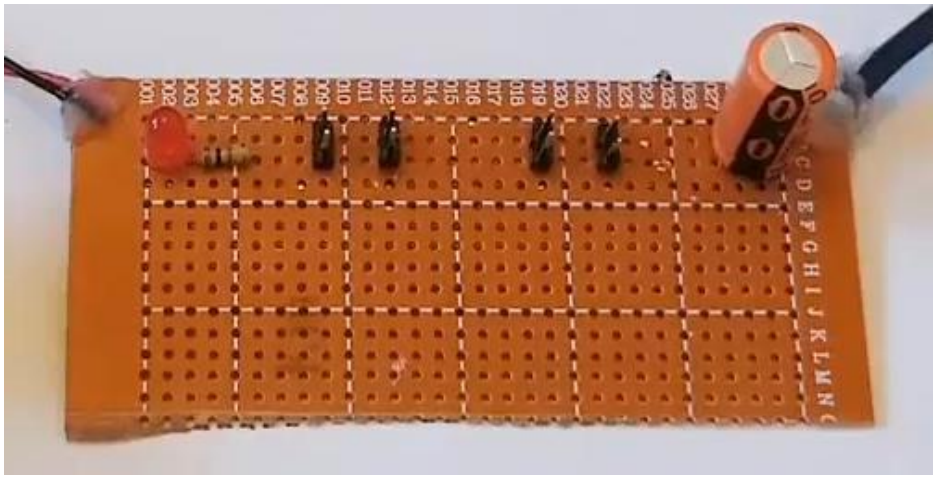

Рисунок 3.10 — Компоновка модуля стабілізації напруги

Після визначення основних складових структури, необхідно обрати корпус, в якому вони будуть розташовані. Використаємо комплект шасі, в якому вбудовано чотири двигуни з колесами та розміщені дві поверхні з PBS пластику, як на рисунку 3.11.

Рисунок 3.11 — Шасі з вбудованими двигунами

Як результат, було обрано усі компоненти системи керування рухомим об’єктом.

3.2. Тренування нейромережі

Для розпізнавання об’єктів на дорозі, спочатку необхідно натренувати нейромережу на позитивних та негативних прикладах. Для цього, створюємо дві директорії з назвами “n” та “p” для негативних та позитивних прикладів зображень об’єкту відповідно. Далі починаємо наповнювати директорії подібними кадрами з камери, як зображено на рисунку 3.12 та 3.13

Рисунок 3.12 — Позитивний приклад об’єкту

Рисунок 3.13 — Негативний приклад об’єкту

Для нейромережі важливо отримати якомога більше кадрів з різних положень та з різної відстані, адже від цього залежить точність розпізнавання та зменшується шанс отримання хибних результатів. Після цього використаємо Cascade Trainer GUI у режимі “Cropper” для усіх позитивних семплів, вирізаючи об’єкти з кадрів для того, щоб збільшити якість вхідних даних на яких буде тренуватися нейромережа (див. рис. 3.14.)

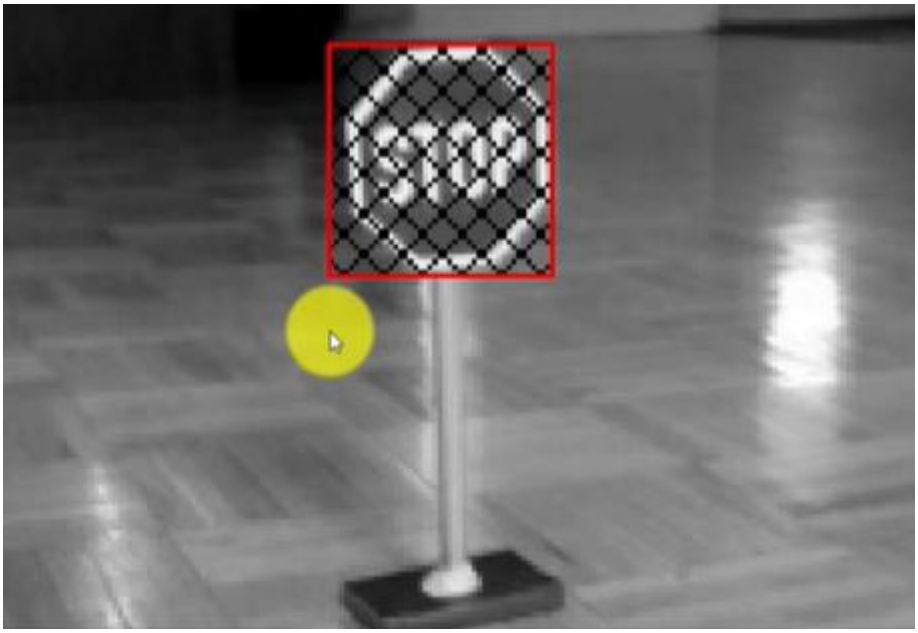

Рисунок 3.14 — Приклад обрізання позитивного семплу

Наступним кроком є завантаження негативних та обрізаних позитивних прикладів об’єкту в Cascade Trainer GUI в режимі “Train”. Також необхідно зазначити параметри тренування такі як кількість ітерацій, розміри прикладів та обрати модель Хаара, як алгоритм розпізнавання (див. рис. 3.15.). Далі починаємо процес тренування та очікуємо на створення XML файлу з каскадом, як результат обробки. У розділі “Log” можна побачити процес обробки з детальною інформацією про кроки тренування, як зображено на рисунку 3.16.

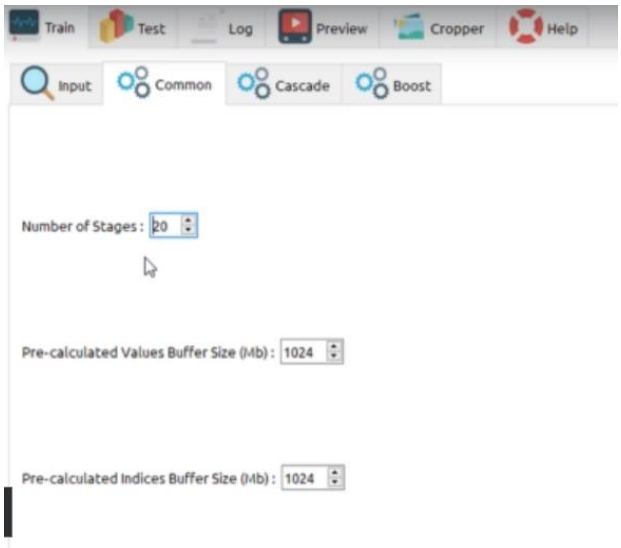

Рисунок 3.15 — Параметри тренування нейромережі

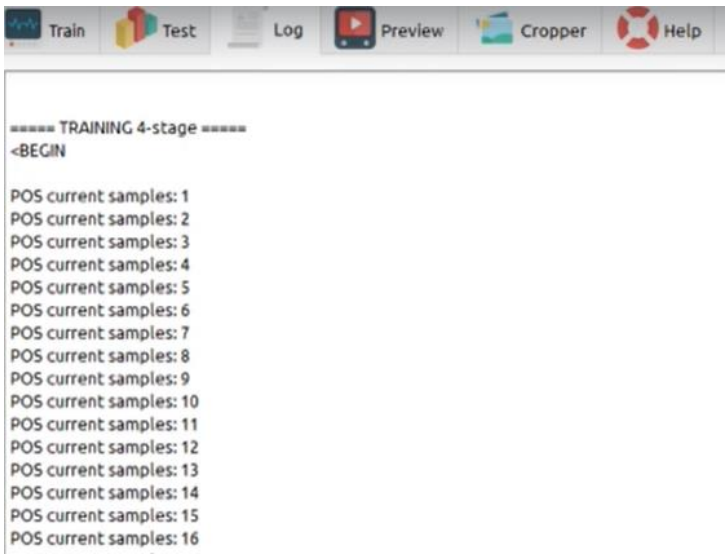

Рисунок 3.16 — Процес обробки семплів

Після отримання каскаду, його можна попередньо протестувати в режимі “Test”. Для цього додаємо шляхи до XML файлу та до позитивних прикладів об’єкту. Як результат буде згенеровано зображення обведені червоним квадратиком у разі ймовірного розпізнавання об’єкту на кадрі. Візуально можна проаналізувати чи квадратики дійсно охоплюють очікувану зону на кадрі, як зображено на рисунку 3.17.

Рисунок 3.17 — Успішне розпізнавання об’єкту

3.3 Розробка програмного забезпечення

Керуюча програма реалізована на базі одноплатного комп’ютера Raspberry Pi 3 Model B. При створенні програми використано такий перелік бібліотек:

— raspicam_cv;

— opencv2;

— iostream;

— chrono;

— ctime;

— wiringPi.

Бібліотека raspicam_cv використовується для встановлення початкових параметрів відеокамери в функції Setup() та для перехоплення кадрів в функції Capture(). Використання бібліотеки показано на лістингу 3.1.

Лістинг 3.1— Використання функцій бібліотеки opencv2

```
// Встановлення основних параметрів камери
void Setup(int argc, char** argv, RaspiCam_Cv& Camera)
{
   Camera.set(CAP_PROP_FRAME_WIDTH, ("-w", argc, argv, 400));
   Camera.set(CAP_PROP_FRAME_HEIGHT, ("-h", argc, argv, 240));
   Camera.set(CAP_PROP_BRIGHTNESS, ("-br", argc, argv, 50));
   Camera.set(CAP_PROP_CONTRAST, ("-co", argc, argv, 50));
   Camera.set(CAP_PROP_SATURATION, ("-sa", argc, argv, 50));
   Camera.set(CAP_PROP_GAIN, ("-g", argc, argv, 50));
   Camera.set(CAP_PROP_FPS, ("-fps", argc, argv, 0));
}
void Capture() // Перехоплення кадру з камери
{
   Camera.grab(); // Захоплення кадру
   Camera.retrieve(frame); // Декодування та повернення фрейму
  // Зміна кольорової схеми
  cvtColor(frame, frame, COLOR_BGR2RGB);
}
```

Бібліотека opencv2 використовується для створення області інтересів, фільтрації зображення, пошуку ліній та центру розмітки дороги, що перехоплюється відеокамерою у вигляді кадрів. Функціонал продемонстровано на лістингу 3.2.

Лістинг 3.2 — Використання функцій бібліотеки opencv2

```
void Perspective(){
```

```
// Створення області інтересів
    // Лінії, що утворюють область інтересів
    line(frame, Source[0], Source[1], Scalar(0, 0, 255), 2);
    line(frame, Source[1], Source[3], Scalar(0, 0, 255), 2);
    line(frame, Source[3], Source[2], Scalar(0, 0, 255), 2);
    line(frame, Source[2], Source[0], Scalar(0, 0, 255), 2);
    Matrix = getPerspectiveTransform(Source, Destination); // Створення трансформації перспективи відносно обраних точок за зображені
    warpPerspective(frame, framePers, Matrix, Size(400, 240)); // Заміна оригінального фрейму транфсформованою перспективою
}
void Threshold()
{
    cvtColor(framePers, frameGray, COLOR_RGB2GRAY); // Зміна кольорової схеми
    inRange(frameGray, 230, 255, frameThresh); // Створення порогу кольорів на фреймі (щоб було видно тільки чорний та білий кольори)
    Canny(frameGray, frameEdge, 900, 900, 3, false); // Створення границь навколо об'єктів на фреймі
    add(frameThresh, frameEdge, frameFinal);
    cvtColor(frameFinal, frameFinal, COLOR_GRAY2RGB);
    cvtColor(frameFinal, frameFinalDuplicate, COLOR_RGB2BGR);
}
void Histrogram() // Поділ області інтересів на лінії
{
    histrogramLane.resize(400);
    histrogramLane.clear();
    for (int i = 0; i < 400; i++)       //frame.size().width = 400
    {
```

```
            ROILane = frameFinalDuplicate(Rect(i, 140, 1, 100));
            divide(255, ROILane, ROILane);
            histrogramLane.push_back((int)(sum(ROILane)[0]));
        }
    void LaneFinder() // Знаходження сторін розмітки
    {
        vector<int>::iterator LeftPtr;
        LeftPtr = max_element(histrogramLane.begin(), histrogramLane.begin() +
150);
        LeftLanePos = distance(histrogramLane.begin(), LeftPtr);
        vector<int>::iterator RightPtr;
        RightPtr = max_element(histrogramLane.begin() + 250,
histrogramLane.end());
        RightLanePos = distance(histrogramLane.begin(), RightPtr);
        line(frameFinal, Point2f(LeftLanePos, 0), Point2f(LeftLanePos, 240),
Scalar(0, 255, 0), 2);
        line(frameFinal, Point2f(RightLanePos, 0), Point2f(RightLanePos, 240),
Scalar(0, 255, 0), 2);
    }
    void LaneCenter() // Обчислення центру розмітки
    {
        laneCenter = (RightLanePos - LeftLanePos) / 2 + LeftLanePos;
        frameCenter = 188;
        line(frameFinal, Point2f(laneCenter, 0), Point2f(laneCenter, 240),
Scalar(0, 255, 0), 3);
        line(frameFinal, Point2f(frameCenter, 0), Point2f(frameCenter, 240),
Scalar(255, 0, 0), 3);
        Result = laneCenter - frameCenter;
    }
```

Також бібліотека opencv2 містить функціонал для створення класифікаторів, що дозволяються розпізнавати об’єкти на основі вхідних каскадних файлів. Приклад такого класифікатора продемонстровано на лістингу 3.3

Лістинг 3.3 — Класифікатор розпізнавання знаку STOP

```
void Stop_detection()
{
  if(!Stop_Cascade.load("//home//pi//Desktop//MACHINE LEARNING//Stop_cascade.xml"))
    {
      printf("Unable to open stop cascade file");
    }
    RoI_Stop = frame_Stop(Rect(200,0,200,140));
    cvtColor(RoI_Stop, gray_Stop, COLOR_RGB2GRAY);
    equalizeHist(gray_Stop, gray_Stop);
    Stop_Cascade.detectMultiScale(gray_Stop, Stop);
    for(int i=0; i<Stop.size(); i++)
    {
      Point P1(Stop[i].x, Stop[i].y);
      Point P2(Stop[i].x + Stop[i].width, Stop[i].y + Stop[i].height);
      rectangle(RoI_Stop, P1, P2, Scalar(0, 0, 255), 2);
      putText(RoI_Stop, "Stop Sign", P1, FONT_HERSHEY_PLAIN, 1, Scalar(0, 0, 255, 255), 2);
      dist_Stop = (-1.07)*(P2.x-P1.x) + 102.597;
    ss.str(" ");
    ss.clear();
    ss<<"D = "<<dist_Stop<<"cm";
    putText(RoI_Stop, ss.str(), Point2f(1,130), 0,1, Scalar(0,0,255), 2);
    }
```

Бібліотека iostream надає змогу виводити корисну інформацію у консоль для розуміння процесу перебігу алгоритму та для відлагодження програми у разі виникнення помилок під час виконання чи компіляції. Використання бібліотеки показано на лістингу 3.4.

Лістинг 3.4 — Використання функції виводу бібліотеки iostream

```
cout << "Connecting to camera" << endl;
if (!Camera.open())
{
   cout << "Failed to Connect" << endl;
}
cout << "Camera Id = " << Camera.getId() << endl;
```

Бібліотеки chrono та ctime потрібні для обчислення кількості оброблених кадрів відеокамерою за секунду. Використання бібліотеки показано на лістингу 3.5.

Лістинг 3.5 — Обчислення кадрів за секунду

```
auto start = std::chrono::system_clock::now();
auto end = std::chrono::system_clock::now();
std::chrono::duration<double> elapsed_seconds = end - start;
float t = elapsed_seconds.count();
int FPS = 1 / t; // Обчислення кількості кадрів за секунду
```

Бібліотека wiringPi потрібна для встановлення зв’язку між головним на підпорядкованим пристроями. При використанні цієї бібліотеки стає можливим в керуючій програмі налаштовувати подачу сигналів на мотори залежно від результату обробки кадру камерою. Використання бібліотеки продемонстровано на лістингу 3.6.

Лістинг 3.6 — Встановлення зв’язку між пристроями та керування підпорядкованим Arduino Uno

```
wiringPiSetup();
pinMode(21, OUTPUT);
pinMode(22, OUTPUT);
pinMode(23, OUTPUT);
pinMode(24, OUTPUT);
      if (Result == 0)
      {
         digitalWrite(21, 0);
         digitalWrite(22, 0);    //decimal = 0
         digitalWrite(23, 0);
         digitalWrite(24, 0);
         cout << "Forward" << endl;
      }
      else if (Result > 0 && Result < 10)
      {
         digitalWrite(21, 1);
         digitalWrite(22, 0);    //decimal = 1
         digitalWrite(23, 0);
         digitalWrite(24, 0);
         cout << "Right1" << endl;
      }
      else if (Result >= 10 && Result < 20)
      {
         digitalWrite(21, 0);
         digitalWrite(22, 1);    //decimal = 2
         digitalWrite(23, 0);
         digitalWrite(24, 0);
         cout << "Right2" << endl;
      }
      else if (Result > 20)
```

```
{
   digitalWrite(21, 1);
   digitalWrite(22, 1);    //decimal = 3
   digitalWrite(23, 0);
   digitalWrite(24, 0);
   cout << "Right3" << endl;
}
else if (Result <0 && Result >-10)
{
   digitalWrite(21, 0);
   digitalWrite(22, 0);    //decimal = 4
   digitalWrite(23, 1);
   digitalWrite(24, 0);
   cout << "Left1" << endl;
}
else if (Result <= -10 && Result > -20)
{
   digitalWrite(21, 1);
   digitalWrite(22, 0);    //decimal = 5
   digitalWrite(23, 1);
   digitalWrite(24, 0);
   cout << "Left2" << endl;
}
else if (Result < -20)
{
   digitalWrite(21, 0);
   digitalWrite(22, 1);    //decimal = 6
   digitalWrite(23, 1);
   digitalWrite(24, 0);
   cout << "Left3" << endl;
}
```

Тепер необхідно створити підпорядковану програму для безпосереднього керування моторами, що спричиняють рух.

Для цього спочатку напишемо функцію, що встановлює початкові параметри для виводів Arduino Uno, як показано на лістингу 3.6.

Лістинг 3.6 — Встановлення початкових параметрів

```
void setup() {
pinMode(EnableL, OUTPUT);
pinMode(HighL, OUTPUT);
pinMode(LowL, OUTPUT);
pinMode(EnableR, OUTPUT);
pinMode(HighR, OUTPUT);
pinMode(LowR, OUTPUT);
pinMode(D0, INPUT_PULLUP);
pinMode(D1, INPUT_PULLUP);
pinMode(D2, INPUT_PULLUP);
pinMode(D3, INPUT_PULLUP);
}
```

Далі створюємо функцію отримання даних з портів, що підключені до Raspberry Pi та надсилають їх до мікроконтролера, як на лістингу 3.7.

Лістинг 3.7 — Отримання даних з портів Raspberry Pi

```
void Data()
{
a = digitalRead(D0);
b = digitalRead(D1);
c = digitalRead(D2);
d = digitalRead(D3);
data = 8*d+4*c+2*b+a;
}
```

Для створення руху, необхідно описати функції руху вперед, назад, ліворуч та праворуч. Слід зауважити, що повороти можуть здійснюватися з різною швидкістю в залежності від вхідних даних. На лістингу 3.8 наведено приклад однієї з функції повороту праворуч.

Лістинг 3.8 — Приклад одного з випадків повороту праворуч

```
void Right1()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,255);
  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,160);
}
```

На цьому розробка програмного забезпечення головного і підпорядкованого пристроїв завершена.

3.4 Компонування модулів системи

Для того, щоб встановити зв’язок між компонентами, необхідно визначитись з портами підключень для кожного з них. У таблицях 3.1, 3.2 продемонстровано з’єднання основних компонентів структури.

Таблиця 3.1 — Порти підключення Arduino з L298N

Модуль	Порти підключення					
Arduino	D5	D6	D7	D8	D9	D10
L298N	ENB	IN4	IN3	IN2	IN1	ENA

Таблиця 3.2 — Порти підключення Arduino з Raspberry Pi

Модуль	Порти підключення			
Arduino	D3	D2	D1	D0
Raspberry Pi	GPIO19	GPIO13	GPIO6	GPIO5

Також розглянемо схему з’єднань усіх компонентів системи керування рухомим об’єктом, що зображена на рисунку 3.18.

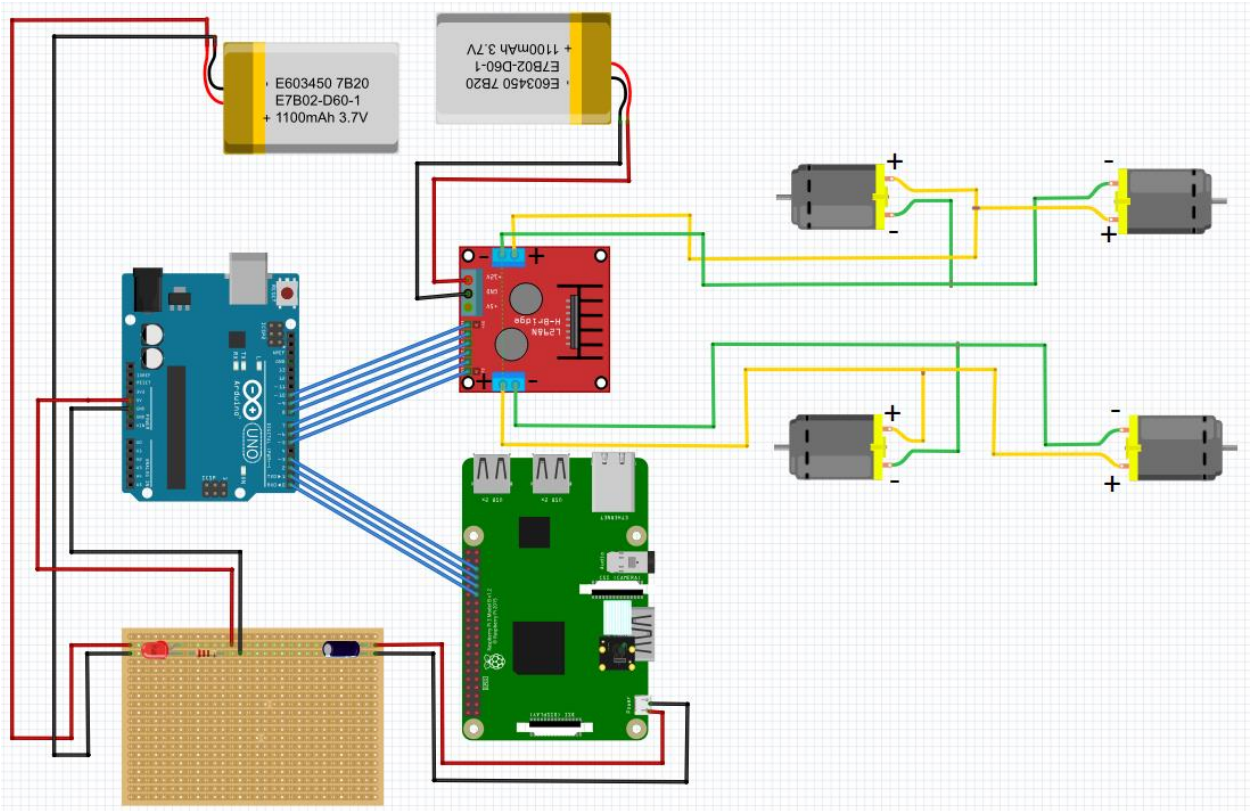

Рисунок 3.18 — Схема з’єднань компонентів системи

Після підключення модулів системи, розробку апаратної частини можна вважати закінченою.

4 ТЕСТУВАННЯ І ВИПРОБУВАННЯ СИСТЕМИ КЕРУВАННЯ РУХОМИМ ОБ'ЄКТОМ

Одним із фундаментальних аспектів тестування є визначення тестових сценаріїв, які охоплюють широкий діапазон можливих вхідних умов. Розробляючи репрезентативні сценарії тестування, можна оцінити, наскільки добре система реагує на різні вхідні дані, з якими вона може зіткнутися під час фактичної роботи.

Обробка помилок є критичним аспектом тестування в системі керування рухомими об'єктами. Це передбачає оцінку здатності системи обробляти несподівані або помилкові вхідні дані. Моделюючи ситуації, коли вхідні дані пошкоджені, неповні або із затримкою, можна оцінити, як система реагує та відновлюється після таких помилок.

Тестування продуктивності має вирішальне значення для оцінки продуктивності системи за різних навантажень і умов. Це тестування передбачає оцінку часу відгуку системи, затримки та точності руху в сценаріях реального часу. Піддаючи систему різноманітним тестам продуктивності, можна виявити будь-які вузькі місця або області для оптимізації, щоб забезпечити оптимальну продуктивність під час роботи. Протягом усього процесу тестування важливо реєструвати та аналізувати дані.

Повинні бути визначені чіткі межі тестування та впроваджені відповідні запобіжні заходи, щоб запобігти будь-якій шкоді чи пошкодженню, які можуть виникнути під час тестування системи керування рухомими об'єктами. Забезпечення безпеки операторів і навколишнього середовища має першочергове значення. Дотримуючись систематичного та комплексного підходу до тестування, можна підтвердити та перевірити продуктивність, надійність і безпеку системи керування рухомими об'єктами в реальному часі в двовимірному просторі. Тестування допомагає виявити та вирішити будь-які проблеми чи обмеження, забезпечуючи точну та ефективну роботу системи в реальних сценаріях.

4.1 Тестування працездатності системи

Для початку, протестуємо роботу камери при подачі живлення у систему. Важливо зрозуміти чи камера вмикається та починає перехоплювати кадри. Для цього використаємо консоль, що повинна вивести інформацію про успішне підключення камери до системи та ідентифікатор камери, як на рисунку 4.1.

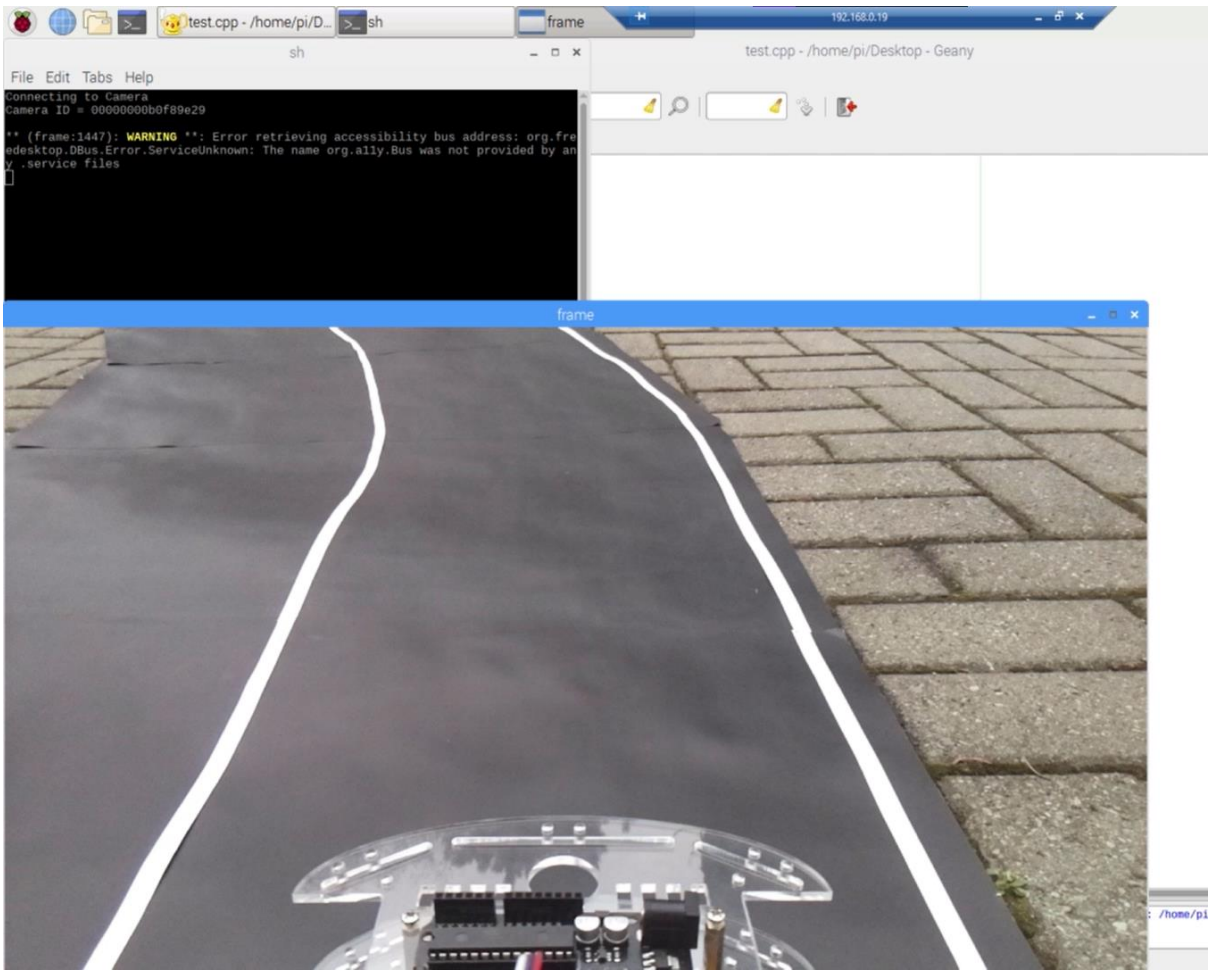

Рисунок 4.1 — Результат успішного підключення камери до системи

Перехоплення кадрів тестуємо також використовуючи консоль, але вже за допомогою розрахунку кількості оброблених кадрів в секунду відеокамерою, як на рисунку 4.2.

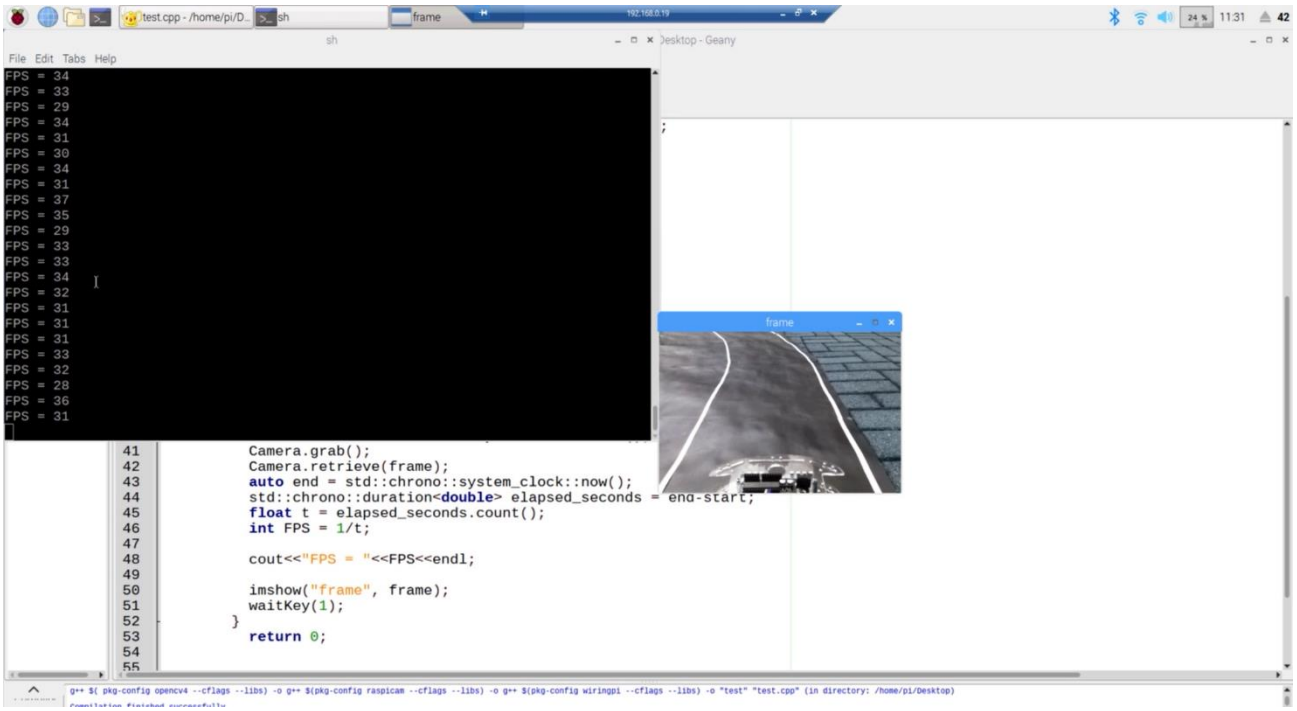

Рисунок 4.2 — Результат обрахунку кількості кадрів в секунду

Тепер подаємо живлення на Arduino Uno для тестування взаємодії з головним пристроєм Raspberry Pi під час master/slave комунікації. Результат руху системи після отримання даних з камери зображено на рисунках 4.3, 4.4.

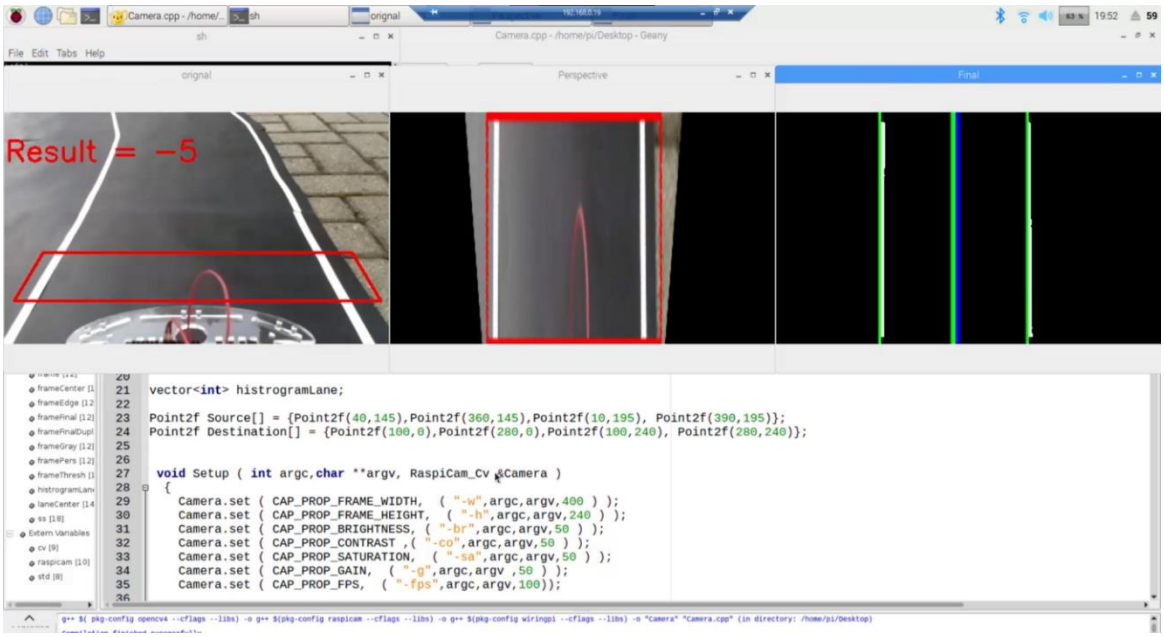

Рисунок 4.3 — Початок руху системи

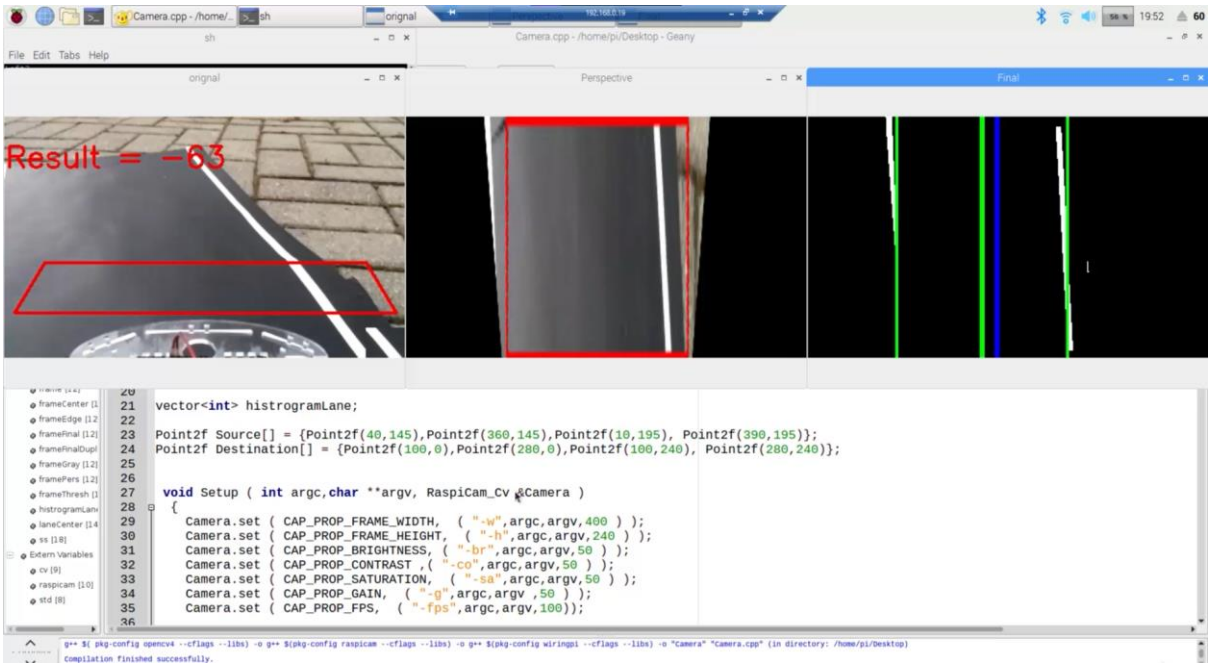

Рисунок 4.4 — Продовження руху системи

Як результат, система керування рухомим об’єктом подолала усі необхідні тести і готова для подальшого використання.

4.2 Тестування розпізнавання об’єктів

Для тестування розпізнавання об’єктів необхідно розмістити знак STOP біля дорожньої розмітки як зображено на рисунку 4.5. Після цього вмикаємо систему та починаємо перехоплювати кадри.

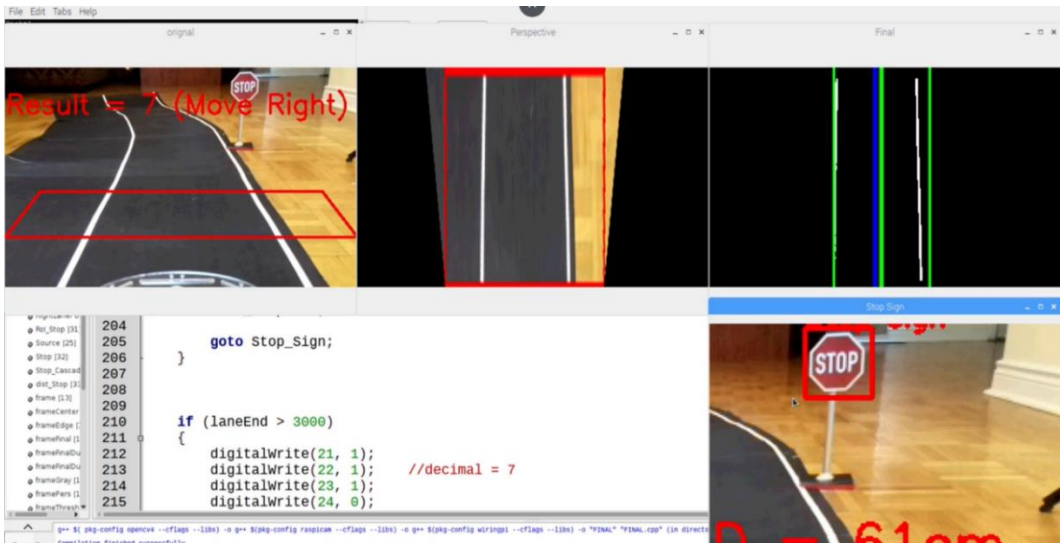

Рисунок 4.5 — Початок розпізнавання знаку STOP

У разі успішного розпізнавання навколо знаку буде намальовано квадратик, який свідчить про знаходження вказаного об’єкту та виведено його назву. Далі система зупиняється на зазначеній відстані від знаку STOP протягом короткого періоду часу та продовжує рух, як зображено на рисунку 4.6.

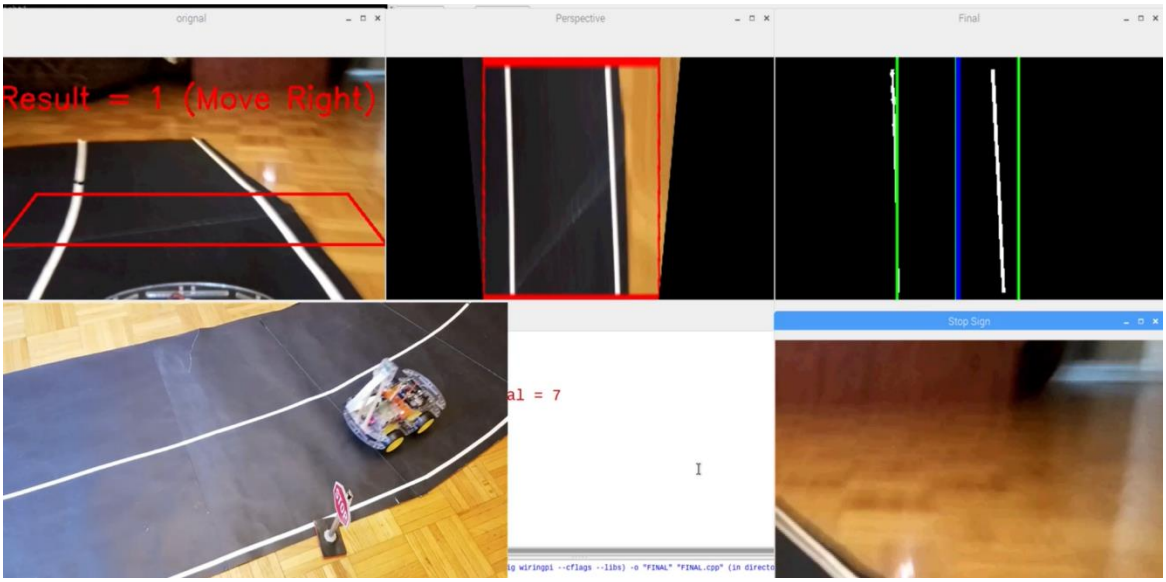

Рисунок 4.6 — Продовження руху після розпізнавання знаку STOP

Для здійснення тесту на об’їзд зовнішнього об’єкту розмістимо іграшковий автомобіль на дорозі, як зображено на рисунку 4.7.

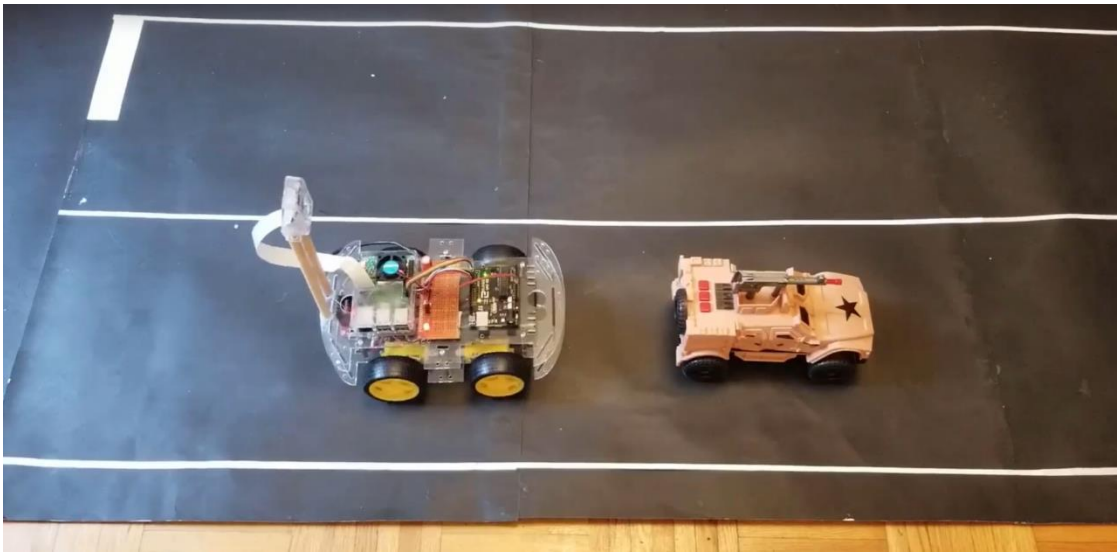

Рисунок 4.7 — Приклад розміщення об’єкту на дорозі

Далі здійснюємо пуск. У разі успішного розпізнавання об’єкту, буде здійснено об’їзд, як зображено на рисунках 4.8, 4.9 та 4.10.

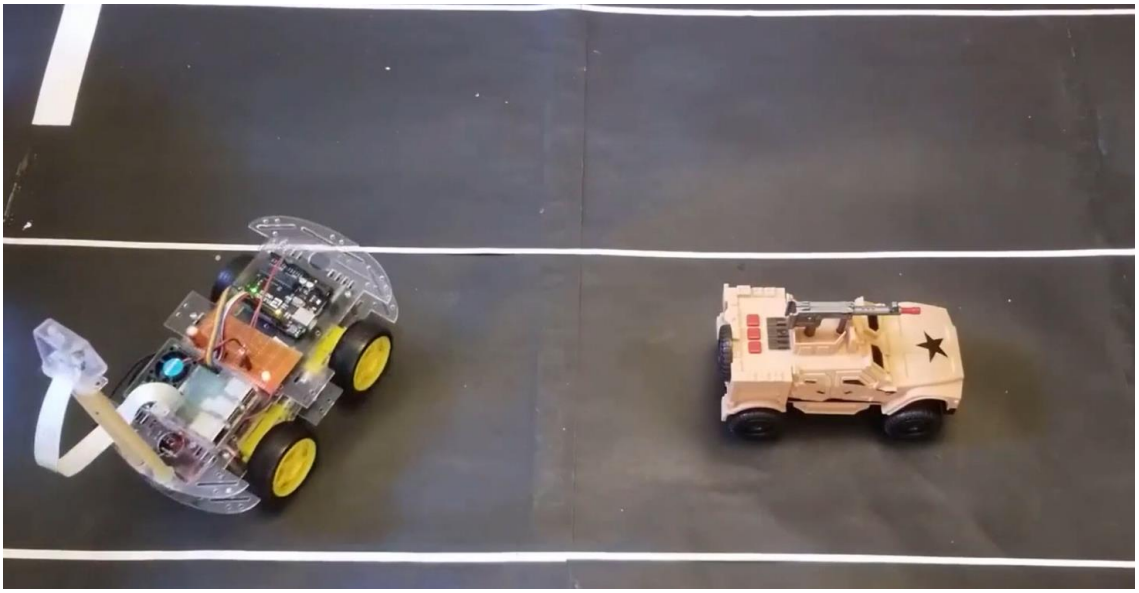

Рисунок 4.8 — Початок об’їзду об’єкту на дорозі

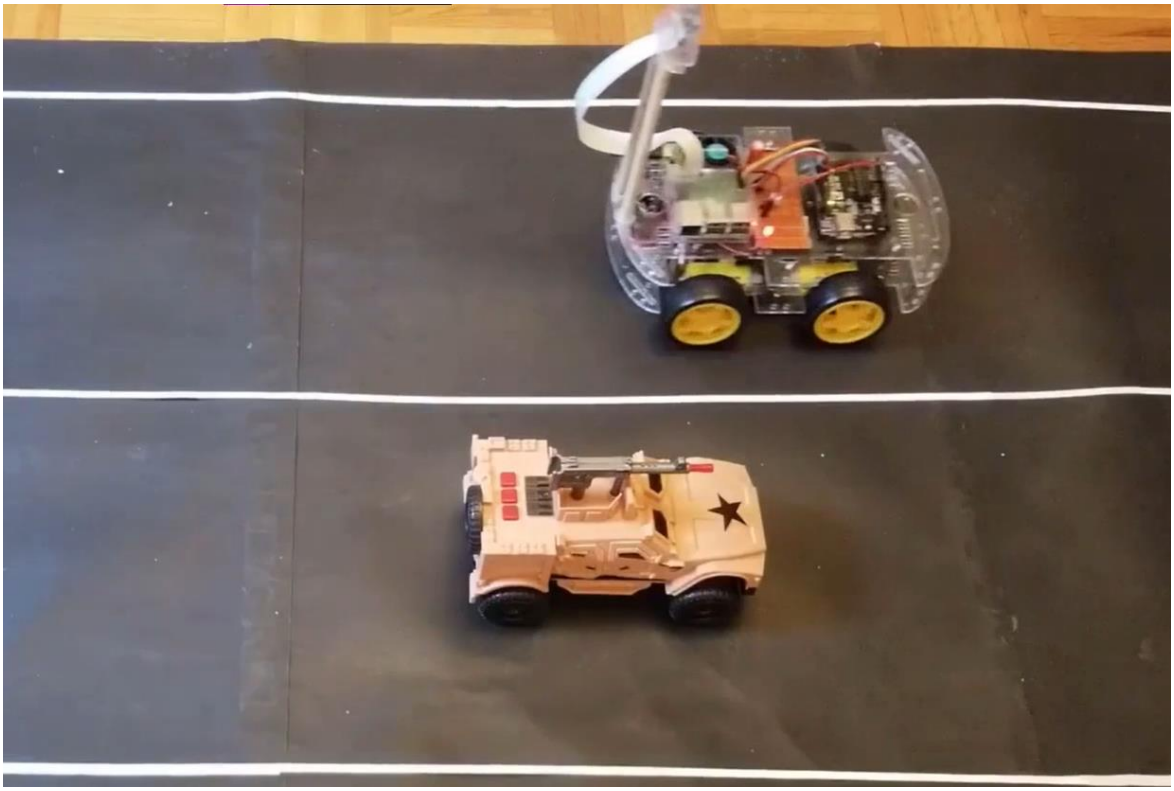

Рисунок 4.9 — Продовження об’їзду об’єкту на дорозі

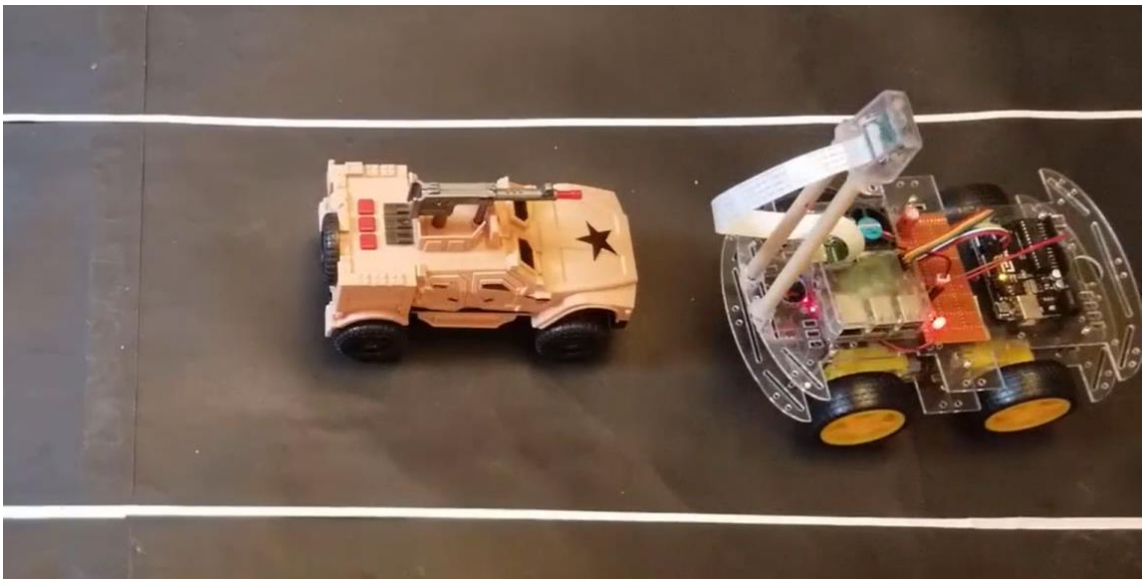

Рисунок 4.10 — Закінчення об’їзду об’єкту на дорозі

Для тестування світлофору, спочатку увімкнемо червоний діод. Після цього здійснюємо пуск системи та чекаємо поки система не розпізнає світлофор та не зупиниться на певній дистанції від нього. Під час успішного розпізнавання система зупинить свій рух, як на рисунку 4.11.

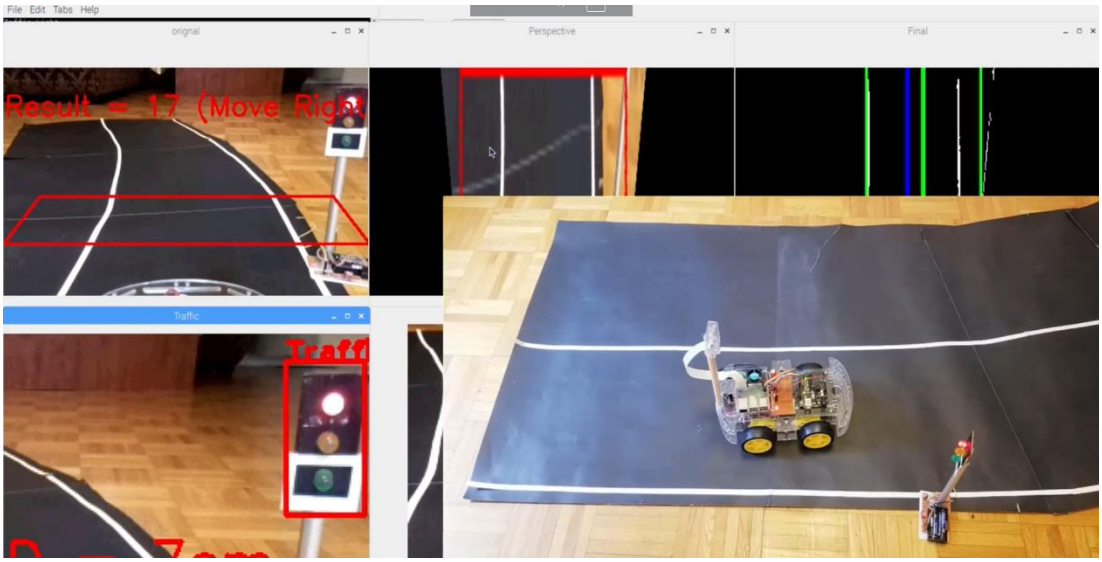

Рисунок 4.11 — Приклад розпізнавання червоного кольору світлофору

Тепер для продовження руху, необхідно вимкнути червоний діод та увімкнути зелений. Після розпізнавання зеленого кольору світлофору, система продовжить рух у нормальному режимі, як на рисунку 4.12.

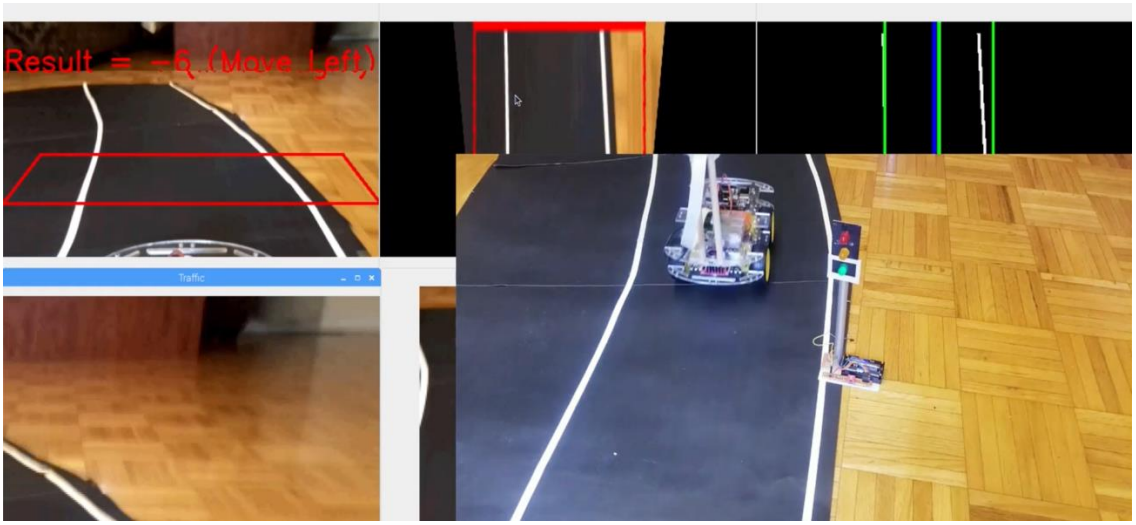

Рисунок 4.12 — Приклад розпізнавання зеленого кольору світлофору

Як результат, система керування рухомим об’єктом успішно розпізнала усі об’єкти на яких її було натреновано та виконала запрограмовані дії, що свідчать про коректну роботу нейромережі.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нового виробу, а саме мікроконтролерної нейромережевої системи керування рухомим об’єктом в двовимірному просторі. Особливістю виробу є те, що за рахунок нейромережевих алгоритмів збільшена точність та функціональність керування. Системи на основі мікроконтролерів із використанням нейромереж здатні працювати в реальному часі, адаптуватись до змінних умов і взаємодіяти з оточуючим середовищем. Вони використовуються в автопілотах автомобілів, безпілотних літальних апаратах, робототехніці, промислових процесах, медичному обладнанні та багатьох інших сферах.

Аналогом розробки є засіб для тестування RL-моделей у гонках на треку AWS DeepRacer - 16 469,61 грн.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 5.1.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Кри-те-	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність в

Продовження табл. 5.1

Ринкові переваги					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження табл. 5.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 5.2

Таблиця 5.2 – Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	3	4	3
Наявність аналогів на ринку	3	4	3
Цінова політика	4	3	3
Технічні та споживчі властивості виробу	3	4	4
Експлуатаційні витрати	4	3	3
Ринок збуту	3	4	4
Конкурентоспроможність	4	3	3
Фахівці з технічної і комерційної реалізації	3	4	4
Фінансування	4	3	4
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	3	3	4
Супровідна документація	3	4	3
Сума	40	42	41
Середньоарифметична сума балів	(40+42+41) / 3 = 41		

За даними таблиці 5.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 5.3.

Таблиця 5.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового виробу є високим, що досягається за рахунок того, що системи на основі мікроконтролерів із використанням нейромереж здатні працювати в реальному часі, адаптуватись до змінних умов і взаємодіяти з оточуючим середовищем. А також, за рахунок нейромережевих алгоритмів збільшена точність та функціональність керування.

5.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t \,, \qquad (5.1)$$

де М – місячний посадовий оклад конкретного розробника (дослідника), грн.;

$T_р$ – число робочих днів в місяці, 22 днів;

t – число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 5.4.

Таблиця 5.4 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	35000	1590,91	33	52500,000
Інженер	32000	1454,55	33	48000,000
Всього				100500,00

Витрати на основну заробітну плату робітників ($З_р$) розраховуються на основі норм часу, які необхідні для виконання даної роботи, розраховуються за формулою:

$$З_р = \sum_{1}^{n} t_i \cdot С_i \cdot К_с , \qquad (5.2)$$

де t_i – норма часу (трудомісткість) на виконання конкретної роботи, годин;

n – число робіт по видах та розрядах;

$К_с$ – коефіцієнт співвідношень, який установлений в даний час Генеральною тарифною угодою між Урядом України і профспілками;

Сі – погодинна тарифна ставка робітника відповідного розряду, який виконує відповідну роботу, грн./год.

$С_i$ визначається за формулою:

$$С_i = \frac{М_м \cdot К_i}{Т_р \cdot Т_{зм}}, \qquad (5.3)$$

де $М_н$ – мінімальна місячна оплата праці, грн., 8000 грн. у 2024 р.

$К_i$ – тарифний коефіцієнт робітника відповідного розряду;

$Т_р$ – число робочих днів в місяці, $Т_р$ = 22 дні;

$Т_{зм}$ – тривалість зміни, $Т_{зм}$ = 8 годин.

Погодинна тарифна ставка згідно чинного законодавства у грудні 2024 року = 48 грн./год. Розрахунки заносимо до табл. 5.5.

Таблиця 5.5 – Витрати на основну заробітну плату робітників

Найменування робіт	Трудомісткість, год.	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн.	Величина оплати на робітника грн.
Заготівельні	3	3	1,35	64,8	194,4
Слюсарно-збиральні	8	3	1,35	64,8	518,4
Налагоджувальні та контрольні	2	5	1,7	81,6	163,2
Всього					876,00

Додаткова заробітна плата прийнято розраховувати як 13% від основної заробітної плати розробників та робітників:

$$З_д = (З_{о.роз} + З_{о.роб}) \cdot 13\% /100\% \quad (5.4)$$

$$З_д = (100500,00 + 876,00) \cdot 13\%/100\% = 13178,88 \text{ (грн.)}$$

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_{о.роз} + З_{о.роб} + З_{д.}) \cdot 22\%/100\% \quad (5.5)$$

$$Н_з = (100500,00 + 876,00 + 13178,88) \cdot 22\%/100\ \% = 25009,35 \text{ (грн.)}$$

Оскільки для розроблювального програмного продукту не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді амортизація обладнання, що використовувалась для розробки розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \quad (5.6)$$

де Ц – балансова вартість обладнання, грн.;

Т – термін корисного використання обладнання згідно податкового законодавства, років

$t_{вик}$ – термін використання під час розробки, місяців

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 20000 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 1,500 міс.

$$A_{обл} = \frac{20000}{2} \times \frac{1{,}5}{12} = 1250 \text{ грн.}$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 5.6.

Таблиця 5.6 – Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн.
Комп'ютер та комп'ютерна периферія	20000	2	1,500	1250,000
Офісне обладнення (меблі)	40000	4	1,500	1250,000
Приміщення	1250000	20	1,500	8197,206
Всього				10697,21

Амортизація обладнання, що використовувалось робітниками, розраховується аналогічно, результати розрахунків зведено в таблицю 5.7 і враховуються при розрахунку виробничої собівартості виробу.

Таблиця 5.7 – Амортизаційні відрахування матеріальних і нематеріальних ресурсів для робітників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання		Амортизаційні відрахування, грн.
			год.	міс.	
Комп'ютер	20000	2	2	0,0114	9,4697
Спеціалізоване обладнання (меблі)	40000	4	2	0,0114	9,4697
Приміщення	1250000	20	13	0,0739	384,7064
Всього					403,6458

Так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів, а також спеціалізованого обладнання менше 20000 грн (Microsoft Windows 10 - 11000 грн., інше використане програмне забезпечення безкоштоване:VNC viewer trial, Raspbian OS, IDE Geany, Cascade Trainer, Arduino IDE, а також паяльний набір з РК-дисплеєм JCD 908S 80W у футлярі 990 грн., як ДШП), то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю, $B_{спец.\ обл..}$ = 11990 грн.

Витрати на комплектуючі, що були використані на виготовлення розраховуються за формулою:

$$K = \sum_{i=1}^{n} H_i \cdot Ц_i \cdot K_i , \qquad (5.7)$$

де H_i - кількість комплектуючих i-го виду, шт;

$Ц_i$ - роздрібна ціна комплектуючих i-го виду, грн;

K_i - коефіцієнт транспортних витрат, K_i=1,1;

n - кількість видів матеріалів.

Проведені розрахунки зводимо до таблиці 5.8 без врахування транспортних витрат.

Таблиця 5.8 – Витрати на комплектуючі

Найменування комплектуючих	Кількість	Ціна за штуку, грн.
Raspberry Pi 3 Model B.	1	1320
Arduino UNO	1	220
Драйвер моторів L298N	1	100
Шлейф для камери Raspberry	1	173
Raspberry Pi Camera Module V2	1	1350
Набір шасі з вбудованими двигунами	1	430
Xiaomi Mi Power Bank 3	1	1000
Конденсатор ємністю 1000 uF	1	100
Резистор опором 1 кОм	1	10
Світлодіоди	4	8
Набір з’єднувальних дротів	1	40
USB Type-C кабелі	2	100
Друкована плата	1	15
Шпильки однорядні	1	10
Всього		***5000,00***

Витрати на комплектуючі, що були використані на розробку з врахуванням транспортних втрат:

$$H = 5000 \cdot 1{,}1 = 5500{,}00 \text{ (грн.)}$$

Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (5.8)$$

де В – вартість 1 кВт-години електроенергії для 1 класу підприємства з ПДВ в 2024 році для Вінницької області за даними Енера-Вінниця, В = 10,42 грн./кВт;

П — встановлена середня потужність обладнання, кВт. П = 0,3 кВт;

Ф — фактична кількість годин роботи обладнання, годин.

$К_п$ — коефіцієнт використання потужності, $К_п$ = 0,8.

$$В_е = 0,8 \cdot 0,3 \cdot 8 \cdot 33 \cdot 10,42 + 0,8 \cdot 0,3 \cdot 4,0 \cdot 10,42 = 660,2112 + 10,003 = 670,21 \text{ (грн.)}$$

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.9)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$$I_в = (100500,00 + 639,48) \cdot 73\% / 100\% = 74004,48 \text{ (грн.)}$$

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні

(загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$H_{нзв} = (З_о + З_р) \cdot \frac{H_{нзв}}{100\%}, \quad (5.10)$$

де $H_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$H_{нзв} = (100500{,}00 + 876{,}00) \cdot 125\% / 100\ \% = 126720 \text{ (грн.)}$$

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$$B_{заг} = 100500{,}00+876{,}00+13178{,}88+25009{,}35+10697{,}21+$$
$$403{,}6458+11990+ +5500{,}00+670{,}21+74004{,}48+126720 = 369549{,}78 \text{ грн.}$$

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються *ЗВ*, визначається за формулою:

$$ЗВ = \frac{B_{заг}}{\eta} \text{ (грн)}, \quad (5.11)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

$$ЗВ = 369549,78 / 0,5 = 739100 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

— вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

— зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

— кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

— визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності

інноваційних проектів передбачається розрахунок таких важливих показників:

— абсолютного економічного ефекту (чистого дисконтованого доходу);

— внутрішньої економічної дохідності (внутрішньої норми дохідності);

— терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об’єднати, враховуючи визначені ситуації з відповідними умовами.

5.4 Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (5.12)$$

де $\pm\Delta Ц_o$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_o = Ц_б \pm \Delta Ц_o$;

$Цб$ – вартість програмного продукту у році до впровадження результатів розробки;

ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$.

p – коефіцієнт, який враховує рентабельність продукту;

ϑ – ставка податку на прибуток, у 2024 році ϑ =18%.

Припустимо, що при прогнозованій ціні 10200 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на 800 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 1500 шт., протягом другого року – на 1300 шт., протягом третього року на 1000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$$\Delta П_1 = (0*800 + (10200 + 800)*1500)* 0{,}8333* 0{,}15) * (1 - 0{,}18) = 1568249{,}937 \text{ грн.}$$

$$\Delta П_2 = (0*800 + (10200 + 800)*(1500+1300)* 0{,}8333* 0{,}15) * (1 - 0{,}18) = 3156999{,}874 \text{ грн.}$$

$$\Delta П_3 = (0*800 + (10200 + 800)*(1500+1300+1000)* 0{,}8333* 0{,}15) * (1 - 0{,}18) = 4284499{,}829 \text{ грн.}$$

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 9009749,64 грн.

5.5 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.13)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т – період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ = 0,05…0,15;

t – період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

ПП =

(1568249,937/(1+0,1)[1])+(3156999,874/(1+0,1)[2])+(4284499,829/(1+0,1)[3]) =

1425681,76 + 2609090,8 + 3219008,14 = 7253780,702 грн.

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (5.14)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}$=2...5, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 * 739100 = \text{грн.}$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід *(NPV, Net Present Value)* для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV, \tag{5.15}$$

$$E_{абс} = 7253780{,}702 - 739100 = 6514681{,}14 \text{ грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої норми дохідності *(IRR, Internal Rate of Return)* вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.16}$$

де $T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{(1 + 6514681{,}14/\ 739100} - 1 = 1{,}141$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{5.17}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d = (0,09…0,14);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0{,}14 + 0{,}05 = 0{,}19.$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \tag{5.18}$$

$$T_{ок} = 1 / 1{,}141 = 0{,}88 \text{ р.}$$

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,88 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку продукту, сума яких складає 739100 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,88 роки.

ВИСНОВКИ

В результаті виконаної роботи було розроблено нейромережеву систему керування рухомим об’єктом в двовимірному просторі, що здатна з високою точністю розпізнавати перешкоди, аналізувати розмітку на дорозі та здійснювати рух.

Проаналізовано аналоги нейромережевих систем автоматичного керування, їх переваги та недоліки в порівнянні з власною розробкою. Здійснено методів управління самокерованими об’єктами, їх категорії та особливості. Розглянуто моделі нейромереж та доцільність їх використання відповідно до поставлених завдань.

Досліджено роботу відеокамери для введення зображень, підпорядкованого пристрою на основі мікроконтролера, керуючого пристрою на основі однопалатного комп’ютера та за допомогою master/slave комунікації встановлено зв’язок між ними. Налаштовано контроль системи за допомогою віддаленого керування через VNC Viewer для здійснення пуску системи.

Створено алгоритми обробки зображень розмітки на дорозі. Натреновано каскадну модель нейромережі для розпізнавання зовнішніх об’єктів та розроблено систему керування рухомим об’єктом за допомогою удосконаленого алгоритму руху.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Фурман, М.; Крупельницький, Л.. Розпізнавання об'єктів із використанням нейронних мереж. Молодь в науці: дослідження, проблеми, перспективи, Ukraine, may. 2024. Available at: <https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21246>. Date accessed: 11 Oct. 2024.2023/paper/view/17523>. Date accessed: 26 May. 2023.

2. Сучасні інформаційні технології та системи в управлінні [Електронний ресурс] : зб. матеріалів I Міжнар. наук.-практ. конф. молодих вчених, аспірантів і студентів ; 19–20 квітня 2018 р. — Київ : КНЕУ, 2018. — 266 с

3. Evolution of neural control structures: some experiments on mobile robots [Електронний ресурс] // Robotics and Autonomous Systems – Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/0921889096810086

4. Vision and navigation for the Carnegie-Mellon Navlab [Електронний ресурс] // Vision and navigation for the Carnegie-Mellon Navlab – Режим доступу: https://www.ri.cmu.edu/pub_files/pub2/thorpe_charles_1988_1/thorpe_charles_1988_1.pdf

5. How Autonomous Vehicles Work [Електронний ресурс] // Let's Talk Autonomous Driving – Режим доступу: https://ltad.com/about/how-autonomous-vehicles-work.html

6. Wireless Remote Control of a Mobile Robot [Електронний ресурс] // ResearchGate – Режим доступу: https://www.researchgate.net/publication/303341093_Wireless_Remote_Control_of_a_Mobile_Robot

7. Surface navigation and mobility intelligence on the Mars Exploration Rovers. In Intelligence for space robotics [Електронний ресурс] // ResearchGate – Режим доступу:

https://www.researchgate.net/publication/268411914_Surface_navigation_and_mobility_intelligence_on_the_Mars_Exploration_Rovers_In_Intelligence_for_space_robotics

8. Practical time-optimal trajectory planning for robots: a convex optimization approach [Електронний ресурс] // ResearchGate – Режим доступу: https://www.researchgate.net/publication/229036395_Practical_time-optimal_trajectory_planning_for_robots_a_convex_optimization_approach

9. Let's Build Robots: Feedback System [Електронний ресурс] // AzoRobotics – Режим доступу: https://www.azorobotics.com/Article.aspx?ArticleID=161

10. Building Trust in Artificial Intelligence, Machine Learning, and Robotic [Електронний ресурс] // ResearchGate – Режим доступу: https://www.researchgate.net/profile/Keng-Siau-2/publication/324006061_Building_Trust_in_Artificial_Intelligence_Machine_Learning_and_Robotics/links/5ab8744baca2722b97cf9d33/Building-Trust-in-Artificial-Intelligence-Machine-Learning-and-Robotics.pdf

11. Технологія комп'ютерного зору: типові помилки під час розробки та впровадження [Електронний ресурс] // Brainberry – Режим доступу: https://brainberry.ua/uk/newsroom/blog/computer-vision-technology-common-mistakes

12. Алгоритми ройового інтелекту та їх застосування [Електронний ресурс] // Факультет математики, природничих наук та технологій ЦДУ ім. В.Винниченка – Режим доступу:

https://phm.cuspu.edu.ua/nauka/konferentsii/fizyka-tekhnolohii-navchannia/99-2017/komp-iuterni-nauky-ta-informatsiini-tekhnolohii/1122-alhorytmy-royovoho-intelektu-ta-yikh-zastosuvannya.html

13. How Robotic Vacuums Work [Електронний ресурс] // HowStuffWorks – Режим доступу:

https://electronics.howstuffworks.com/gadgets/home/robotic-vacuum.htm

14. LD Series Autonomous Mobile Robots [Електронний ресурс] // Omron Automation – Режим доступу: https://automation.omron.com/en/us/products/family/LD

15. Segway Ninebot S Review [Електронний ресурс] // GadgetReview – Режим доступу: https://www.gadgetreview.com/segway-ninebot-s-review

16. Automatic control theory [Електронний ресурс] // Encyclopedia of Mathematics – Режим доступу: https://encyclopediaofmath.org/wiki/Automatic_control_theory

17. Microcontroller Based Master Slave Communication for Electrical Stepper Motor [Електронний ресурс] // International Journal of Science and Research – Режим доступу: https://www.ijsr.net/archive/v5i5/NOV163471.pdf

18. How to use WiringPi Library on Raspberry Pi [Електронний ресурс] // ElectronicWings – Режим доступу: https://www.electronicwings.com/raspberry-pi/how-to-use-wiringpi-library-on-raspberry-pi

19. OpenCV - Open Computer Vision Library [Електронний ресурс] // OpenCV – Режим доступу: https://opencv.org/

20. VNC Viewer [Електронний ресурс] // RealVNC – Режим доступу: https://help.realvnc.com/hc/en-us/articles/360003474552-How-do-I-get-started-with-VNC-Connect-on-Windows-and-Mac-

21. Cascade classifier for face detection [Електронний ресурс] // Cascade classifier for face detection – Режим доступу: https://www.researchgate.net/publication/303325397_Cascade_classifier_for_face_detection

22. OpenCV for Computer Vision Applications [Електронний ресурс] // OpenCV for Computer Vision Applications – Режим доступу: https://www.researchgate.net/publication/301590571_OpenCV_for_Computer_Vision_Applications

23. Working Principle of Arduino and Using it as a Tool for Study and Research [Електронний ресурс] // Working Principle of Arduino and Using it as a Tool for Study and Research – Режим доступу:

https://www.researchgate.net/publication/326316390_Working_Principle_of_Arduino_and_Using_it_as_a_Tool_for_Study_and_Research

24. Building Custom HAAR-Cascade Classifier for face Detection [Електронний ресурс] // Building Custom HAAR-Cascade Classifier for face Detection – Режим доступу: https://www.researchgate.net/publication/338680205_Building_Custom_HAAR-Cascade_Classifier_for_face_Detection

25. Robotic Automation of Software Testing From a Machine Learning Viewpoint // Robotic Automation of Software Testing From a Machine Learning Viewpoint – Режим доступу: https://www.researchgate.net/publication/367815939_Robotic_Automation_of_Software_Testing_From_a_Machine_Learning_Viewpoint

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

_____ проф., д.т.н. О.Д. Азаров

«___» ____ р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

Мікроконтролерна нейромережева система керування рухомим об’єктом в двовимірному просторі

08-54.МКР.020.00.000 ТЗ

Керівник роботи: к.т.н., доц. каф. ОТ:

_____ Крупельницький Л.В

Виконав: студент гр. 1КІ-23м

_____Фурман М.А.

2024

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність роботи полягає в необхідності створення рухомих об'єктів та у впровадженні нейромережевих технологій для розвитку автономних систем;

1.2 Наказ ВНТУ від “17” вересня 2024 року № 310 про затвердження теми МКР

2 Мета і призначення МКР

2.1 Мета роботи: створення мікроконтролерної нейромережевої системи керування рухомим об’єктом**,** що здатна з високою точністю розпізнавати перешкоди, аналізувати розмітку на дорозі та здійснювати рух.

2.2 Призначення розробки — автоматизоване управління рухомим об’єктом в двовимірному просторі.

3 Вихідні дані для виконання МКР

Вихідні дані для виконання МКР:

— функціональне призначення — управління рухомим об’єктом в двовимірному просторі;

— тип об’єкту — рухомий транспортний засіб;

— двовимірний простір з розміткою на дорозі;

— спосіб керування — автоматичний;

— пуск — через безпровідний засіб зв’язку.

4 Технічні вимоги до виконання МКР

Технічні вимоги до виконання МКР:

— аналіз історії розвитку безпілотних нейромережевих об’єктів;

— пошук та аналіз методів управління самокерованими об’єктами;

— пошук та аналіз методів розпізнавання об’єктів;

— огляд сучасних систем керування рухомим об’єктом;

— вибір компонентів для створення системи керування рухомим об’єктом;

— створення програмного забезпечення та налаштування зв’язку між компонентами апаратної складової;

— тестування працездатності системи.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1

Таблиця А.1 — Етапи МКР

№ з/п	Назва етапу	Термін виконання		Очікувані результати
		початок	закінчення	
1	Постановка задачі роботи	19.09.2024	19.09.2024	Задачі дослідження
2	Пошук та аналіз методів управління самокерованими об’єктами	20.09.2024	20.09.2024	Аналітичний огляд джерел
3	Пошук та аналіз методів розпізнавання об’єктів	21.09.2024	22.09.2024	Аналітичний огляд джерел
4	Аналіз аналогів нейромережевих систем керування рухомим об’єктом	23.09.2024	25.09.2024	Розділ 1
5	Вибір компонентів для створення нейромережевої системи керування рухомим об’єктом	26.09.2024	05.10.2024	Розділ 2
6	Створення програмного забезпечення та налаштування зв’язку між компонентами апаратної складової	06.10.2024	26.10.2024	Розділ 3
7	Тестування працездатності системи	27.10.2024	10.11.2024	Розділ 4
8	Розрахунок економічної частини	11.11.2024	11.11.2024	Розділ 5
9	Оформлення пояснювальної записки	12.11.2024	12.11.2024	ПЗ, графічний матеріал і презентація
10	Перевірка якості виконання магістерської роботи та усунення недоліків	13.11.2024	14.11.2024	Оформлені документи
11	Підписи супроводжувальних документів у нормоконтролера, керівника, опонента	15.11.2024	16.11.2024	Оформлені документи
12	Перевірка на антиплагіат та ШІ	17.11.2024	17.11.2024	Оформлені документи

6 Матеріали, що подаються до захисту МКР

Пояснювальна записка МКР, графічні та ілюстративні матеріали, протокол попереднього захисту роботи на кафедрі, відгук наукового керівника, відгук опонента, анотації до МКР українською та іноземною мовами, нормоконтроль про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлювання МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ–03.02.02 П.001.01:21.

ДОДАТОК Б

Структурна схема системи керування

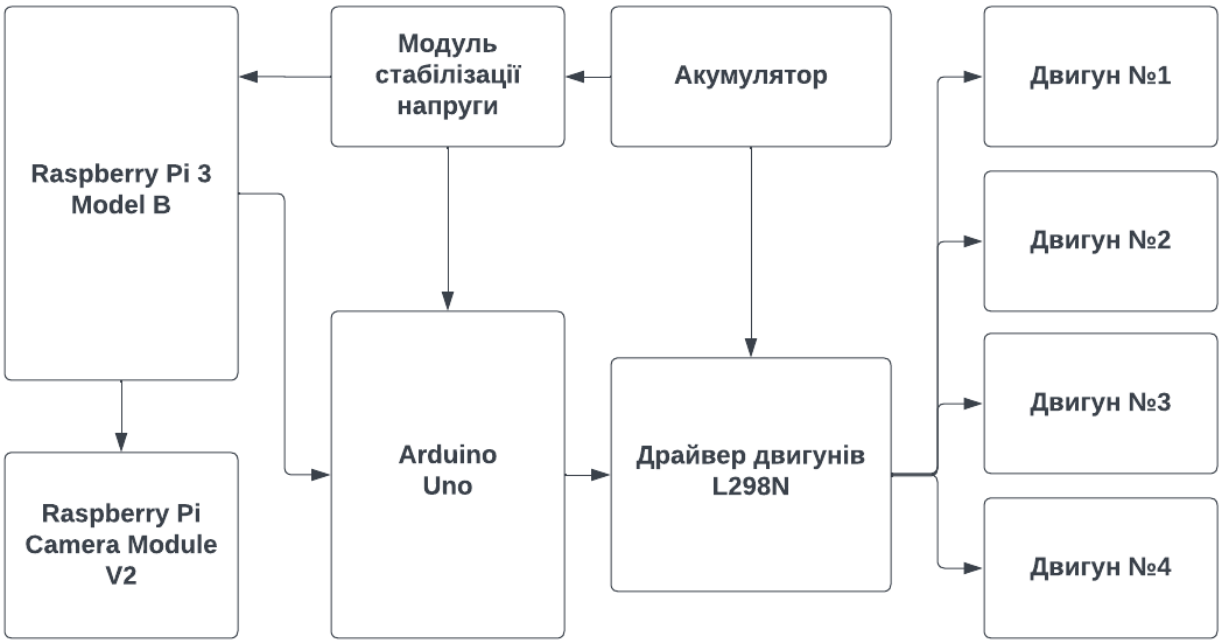

Рисунок Б.1 — Структурна схема системи керування

ДОДАТОК В

Блок-схема алгоритму руху системи

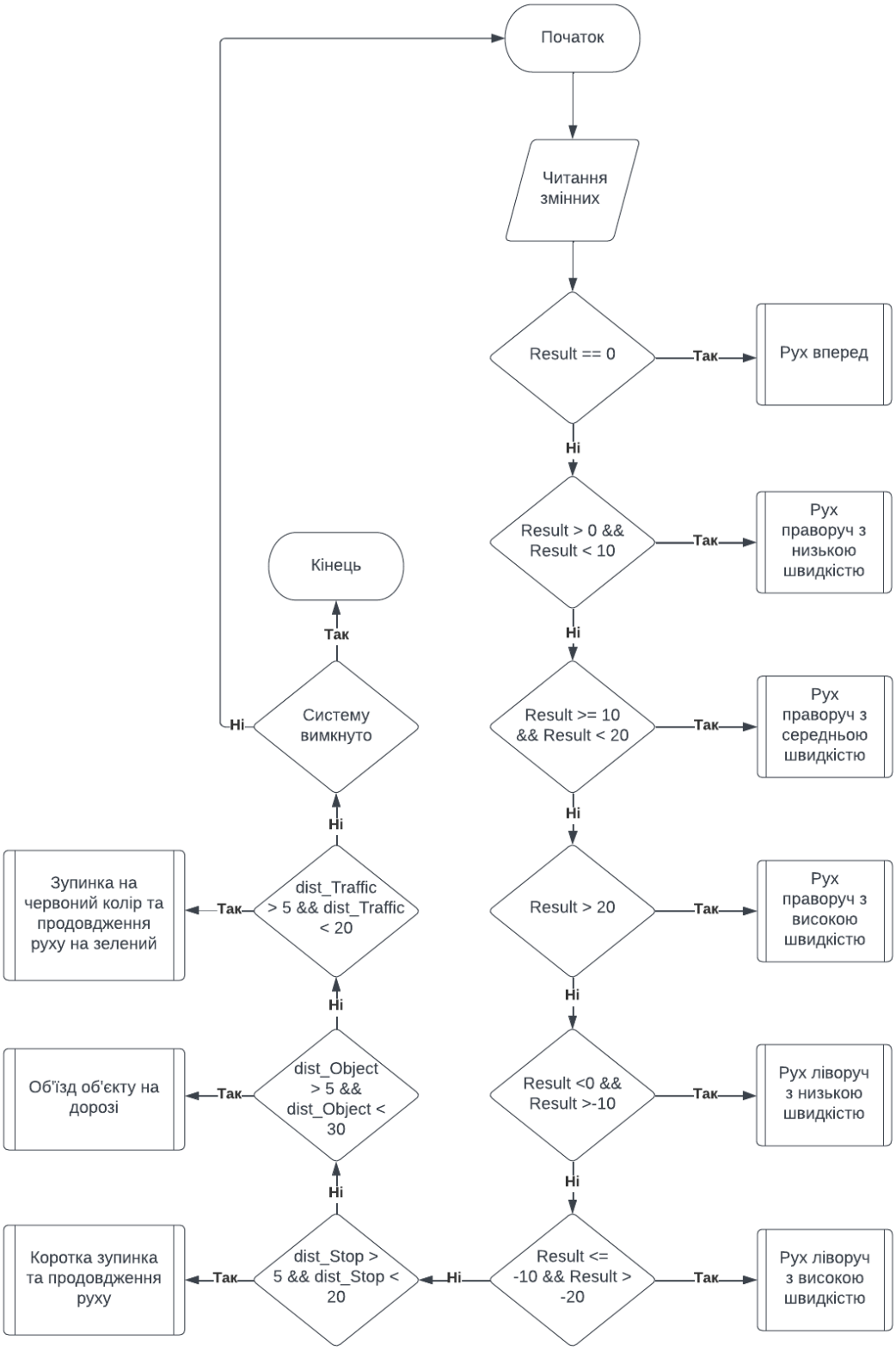

Рисунок В.1 — Блок-схема алгоритму руху системи

ДОДАТОК Г

Блок-схема основного алгоритму роботи системи

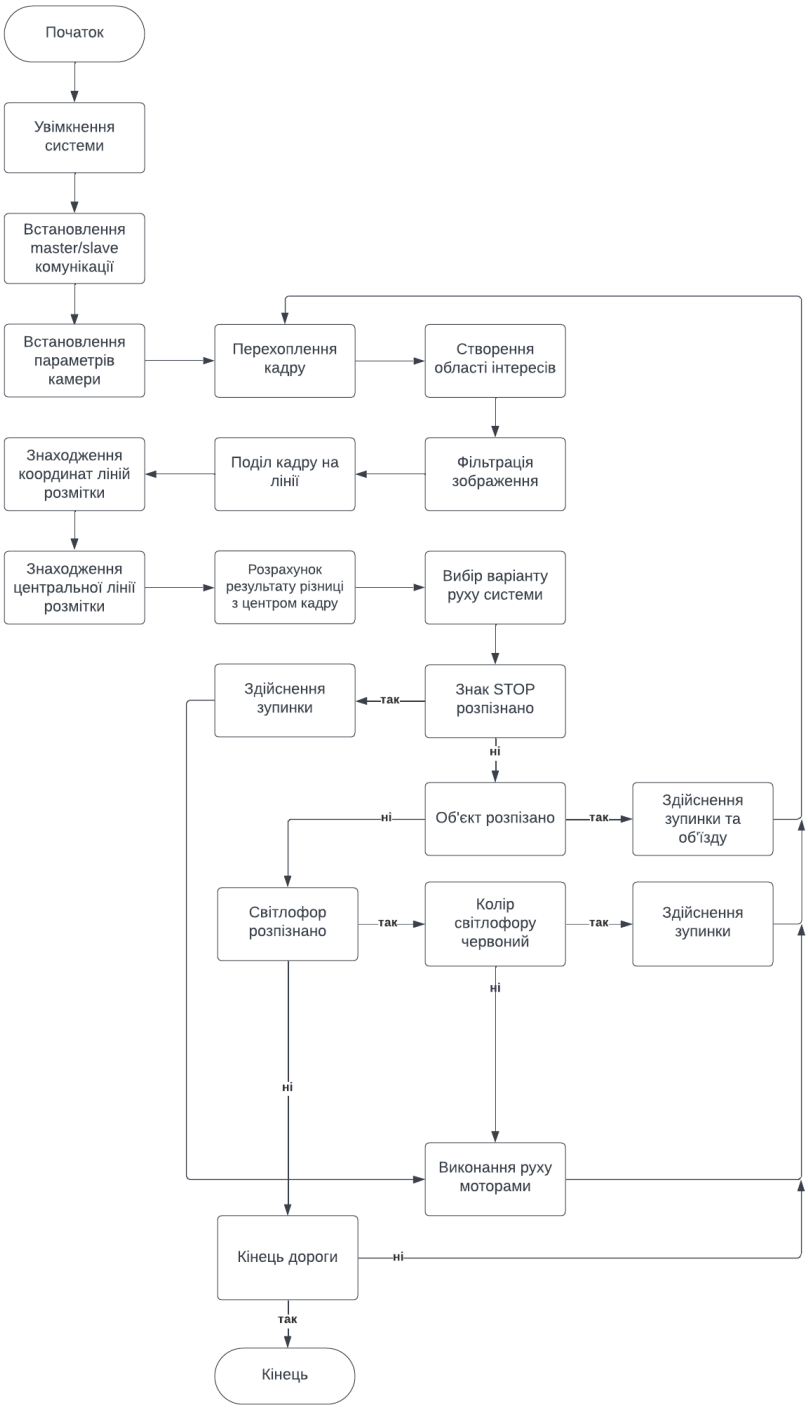

Рисунок Г.1 — Блок-схема основного алгоритму роботи системи

ДОДАТОК Д

Схема з’єднань компонентів системи

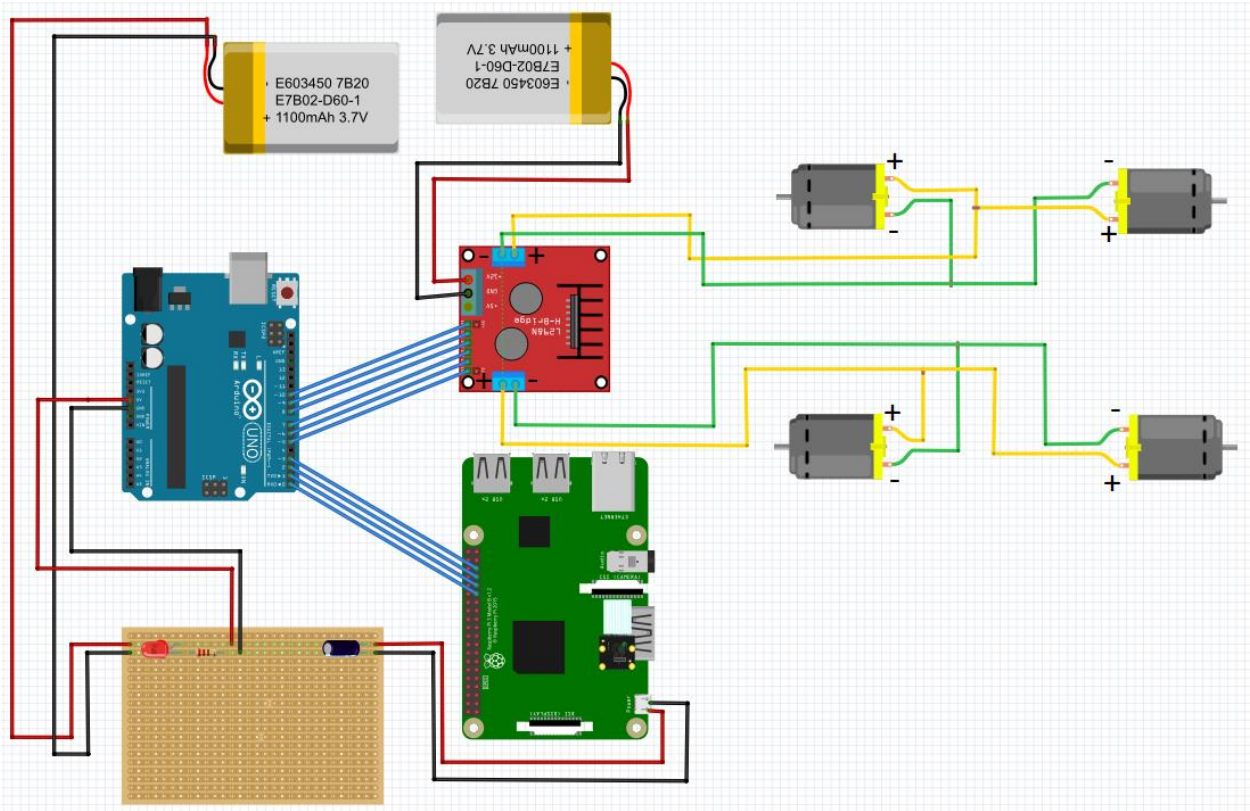

Рисунок Д.1 — Схема з’єднань компонентів системи

ДОДАТОК Е

Лістинг модуля головного пристрою

```
#include <opencv2/opencv.hpp>
#include <raspicam_cv.h>
#include <iostream>
#include <chrono>
#include <ctime>
#include <wiringPi.h>

using namespace std;
using namespace cv;
using namespace raspicam;

Mat frame, Matrix, framePers, frameGray, frameThresh, frameEdge, frameFinal, frameFinalDuplicate, frameFinalDuplicate1;
Mat ROILane, ROILaneEnd;
int LeftLanePos, RightLanePos, frameCenter, laneCenter, Result, laneEnd;

RaspiCam_Cv Camera;

stringstream ss;

vector<int> histrogramLane;
vector<int> histrogramLaneEnd;

Point2f Source[] = {Point2f(40,135),Point2f(360,135),Point2f(0,185), Point2f(400,185)};
Point2f Destination[] = {Point2f(100,0),Point2f(280,0),Point2f(100,240), Point2f(280,240)};

CascadeClassifier Stop_Cascade, Object_Cascade, Traffic_Cascade;
Mat frame_Stop, RoI_Stop, gray_Stop, frame_Object, RoI_Object, gray_Object, frame_Traffic, RoI_Traffic, gray_Traffic;
vector<Rect> Stop, Object, Traffic;
int dist_Stop, dist_Object, dist_Traffic;

 void Setup ( int argc,char **argv, RaspiCam_Cv &Camera )
  {
    Camera.set ( CAP_PROP_FRAME_WIDTH,  ( "-w",argc,argv,400 ) );
    Camera.set ( CAP_PROP_FRAME_HEIGHT,  ( "-h",argc,argv,240 ) );
    Camera.set ( CAP_PROP_BRIGHTNESS, ( "-br",argc,argv,50 ) );
    Camera.set ( CAP_PROP_CONTRAST ,( "-co",argc,argv,50 ) );
    Camera.set ( CAP_PROP_SATURATION,  ( "-sa",argc,argv,50 ) );
```

```
   Camera.set ( CAP_PROP_GAIN,  ( "-g",argc,argv ,50 ) );
   Camera.set ( CAP_PROP_FPS,  ( "-fps",argc,argv,0));

}

void Capture()
{
   Camera.grab();
   Camera.retrieve( frame);
   cvtColor(frame, frame_Stop, COLOR_BGR2RGB);
   cvtColor(frame, frame_Object, COLOR_BGR2RGB);
   cvtColor(frame, frame_Traffic, COLOR_BGR2RGB);
   cvtColor(frame, frame, COLOR_BGR2RGB);

}

void Perspective()
{
line(frame,Source[0], Source[1], Scalar(0,0,255), 2);
line(frame,Source[1], Source[3], Scalar(0,0,255), 2);
line(frame,Source[3], Source[2], Scalar(0,0,255), 2);
line(frame,Source[2], Source[0], Scalar(0,0,255), 2);


Matrix = getPerspectiveTransform(Source, Destination);
warpPerspective(frame, framePers, Matrix, Size(400,240));
}

void Threshold()
{
cvtColor(framePers, frameGray, COLOR_RGB2GRAY);
inRange(frameGray, 230, 255, frameThresh);
Canny(frameGray,frameEdge, 900, 900, 3, false);
add(frameThresh, frameEdge, frameFinal);
cvtColor(frameFinal, frameFinal, COLOR_GRAY2RGB);
cvtColor(frameFinal, frameFinalDuplicate, COLOR_RGB2BGR);
cvtColor(frameFinal, frameFinalDuplicate1, COLOR_RGB2BGR);

}

void Histrogram()
{
   histrogramLane.resize(400);
   histrogramLane.clear();
```

```
   for(int i=0; i<400; i++)
   {
ROILane = frameFinalDuplicate(Rect(i,140,1,100));
divide(255, ROILane, ROILane);
histrogramLane.push_back((int)(sum(ROILane)[0]));
   }

histrogramLaneEnd.resize(400);
       histrogramLaneEnd.clear();
for (int i = 0; i < 400; i++)
{
        ROILaneEnd = frameFinalDuplicate1(Rect(i, 0, 1, 240));
        divide(255, ROILaneEnd, ROILaneEnd);
        histrogramLaneEnd.push_back((int)(sum(ROILaneEnd)[0]));


}
  laneEnd = sum(histrogramLaneEnd)[0];
  cout<<"Lane END = "<<laneEnd<<endl;
}

void LaneFinder()
{
   vector<int>:: iterator LeftPtr;
   LeftPtr = max_element(histrogramLane.begin(), histrogramLane.begin() + 150);
   LeftLanePos = distance(histrogramLane.begin(), LeftPtr);

   vector<int>:: iterator RightPtr;
   RightPtr = max_element(histrogramLane.begin() +250, histrogramLane.end());
   RightLanePos = distance(histrogramLane.begin(), RightPtr);

   line(frameFinal, Point2f(LeftLanePos, 0), Point2f(LeftLanePos, 240), Scalar(0,
255,0), 2);
   line(frameFinal, Point2f(RightLanePos, 0), Point2f(RightLanePos, 240),
Scalar(0,255,0), 2);
}

void LaneCenter()
{
   laneCenter = (RightLanePos-LeftLanePos)/2 +LeftLanePos;
   frameCenter = 188;
```

```
   line(frameFinal, Point2f(laneCenter,0), Point2f(laneCenter,240), Scalar(0,255,0),
3);
   line(frameFinal,          Point2f(frameCenter,0),          Point2f(frameCenter,240),
Scalar(255,0,0), 3);

   Result = laneCenter-frameCenter;
}

void Stop_detection()
{
   if(!Stop_Cascade.load("//home//pi//Desktop//MACHINE
LEARNING//Stop_cascade.xml"))
   {
printf("Unable to open stop cascade file");
   }

   RoI_Stop = frame_Stop(Rect(200,0,200,140));
   cvtColor(RoI_Stop, gray_Stop, COLOR_RGB2GRAY);
   equalizeHist(gray_Stop, gray_Stop);
   Stop_Cascade.detectMultiScale(gray_Stop, Stop);

   for(int i=0; i<Stop.size(); i++)
   {
Point P1(Stop[i].x, Stop[i].y);
Point P2(Stop[i].x + Stop[i].width, Stop[i].y + Stop[i].height);

rectangle(RoI_Stop, P1, P2, Scalar(0, 0, 255), 2);
putText(RoI_Stop, "Stop Sign", P1, FONT_HERSHEY_PLAIN, 1,  Scalar(0, 0,
255, 255), 2);
dist_Stop = (-1.07)*(P2.x-P1.x) + 102.597;

      ss.str(" ");
      ss.clear();
      ss<<"D = "<<dist_Stop<<"cm";
      putText(RoI_Stop, ss.str(), Point2f(1,130), 0,1, Scalar(0,0,255), 2);

   }

}

void Traffic_detection()
{
   if(!Traffic_Cascade.load("//home//pi//Desktop//MACHINE
LEARNING//Trafficc_cascade.xml"))
```

```
   {
printf("Unable to open traffic cascade file");
   }

   RoI_Traffic = frame_Traffic(Rect(200,0,200,140));
   cvtColor(RoI_Traffic, gray_Traffic, COLOR_RGB2GRAY);
   equalizeHist(gray_Traffic, gray_Traffic);
   Traffic_Cascade.detectMultiScale(gray_Traffic, Traffic);

   for(int i=0; i<Traffic.size(); i++)
   {
Point P1(Traffic[i].x, Traffic[i].y);
Point P2(Traffic[i].x + Traffic[i].width, Traffic[i].y + Traffic[i].height);

rectangle(RoI_Traffic, P1, P2, Scalar(0, 0, 255), 2);
putText(RoI_Traffic, "Traffic Light", P1, FONT_HERSHEY_PLAIN, 1,  Scalar(0,
0, 255, 255), 2);
dist_Traffic = (-1.07)*(P2.x-P1.x) + 102.597;

      ss.str(" ");
      ss.clear();
      ss<<"D = "<<P2.x-P1.x<<"cm";
      putText(RoI_Traffic, ss.str(), Point2f(1,130), 0,1, Scalar(0,0,255), 2);

   }

}

void Object_detection()
{
   if(!Object_Cascade.load("//home//pi//Desktop//MACHINE
LEARNING//Object_cascade.xml"))
   {
printf("Unable to open Object cascade file");
   }

   RoI_Object = frame_Object(Rect(100,50,200,190));
   cvtColor(RoI_Object, gray_Object, COLOR_RGB2GRAY);
   equalizeHist(gray_Object, gray_Object);
   Object_Cascade.detectMultiScale(gray_Object, Object);

   for(int i=0; i<Object.size(); i++)
   {
Point P1(Object[i].x, Object[i].y);
```

```
Point P2(Object[i].x + Object[i].width, Object[i].y + Object[i].height);

rectangle(RoI_Object, P1, P2, Scalar(0, 0, 255), 2);
putText(RoI_Object, "Object", P1, FONT_HERSHEY_PLAIN, 1,  Scalar(0, 0, 255,
255), 2);
dist_Object = (-0.48)*(P2.x-P1.x) + 56.6;

      ss.str(" ");
      ss.clear();
      ss<<"D = "<<dist_Object<<"cm";
      putText(RoI_Object, ss.str(), Point2f(1,130), 0,1, Scalar(0,0,255), 2);

   }

}

int main(int argc,char **argv)
{

   wiringPiSetup();
   pinMode(21, OUTPUT);
   pinMode(22, OUTPUT);
   pinMode(23, OUTPUT);
   pinMode(24, OUTPUT);

   Setup(argc, argv, Camera);
   cout<<"Connecting to camera"<<endl;
   if (!Camera.open())
   {

cout<<"Failed to Connect"<<endl;
   }

cout<<"Camera Id = "<<Camera.getId()<<endl;

    while(1)
   {

   auto start = std::chrono::system_clock::now();

   Capture();
   Perspective();
   Threshold();
   Histrogram();
```

```
   LaneFinder();
   LaneCenter();
   Stop_detection();
   Object_detection();
   Traffic_detection();

   if (dist_Stop > 5 && dist_Stop < 20)
   {
digitalWrite(21, 0);
digitalWrite(22, 0);    //decimal = 8
digitalWrite(23, 0);
digitalWrite(24, 1);
cout<<"Stop Sign"<<endl;
dist_Stop = 0;

goto Stop_Sign;
   }

      if (dist_Object > 5 && dist_Object < 30)
   {
digitalWrite(21, 1);
digitalWrite(22, 0);    //decimal = 9
digitalWrite(23, 0);
digitalWrite(24, 1);
cout<<"Object"<<endl;
dist_Object = 0;

goto Object;
   }

    if (laneEnd > 4500)
   {
        digitalWrite(21, 1);
digitalWrite(22, 1);    //decimal = 7
digitalWrite(23, 1);
digitalWrite(24, 0);
cout<<"Lane End"<<endl;
   }

      if (Result == 0)
   {
digitalWrite(21, 0);
digitalWrite(22, 0);    //decimal = 0
digitalWrite(23, 0);
```

```
digitalWrite(24, 0);
cout<<"Forward"<<endl;
   }

        else if (Result >0 && Result <10)
   {
digitalWrite(21, 1);
digitalWrite(22, 0);    //decimal = 1
digitalWrite(23, 0);
digitalWrite(24, 0);
cout<<"Right1"<<endl;
   }

      else if (Result >=10 && Result <20)
   {
digitalWrite(21, 0);
digitalWrite(22, 1);    //decimal = 2
digitalWrite(23, 0);
digitalWrite(24, 0);
cout<<"Right2"<<endl;
   }

      else if (Result >20)
   {
digitalWrite(21, 1);
digitalWrite(22, 1);    //decimal = 3
digitalWrite(23, 0);
digitalWrite(24, 0);
cout<<"Right3"<<endl;
   }

      else if (Result <0 && Result >-10)
   {
digitalWrite(21, 0);
digitalWrite(22, 0);    //decimal = 4
digitalWrite(23, 1);
digitalWrite(24, 0);
cout<<"Left1"<<endl;
   }

      else if (Result <=-10 && Result >-20)
   {
digitalWrite(21, 1);
digitalWrite(22, 0);    //decimal = 5
```

```
digitalWrite(23, 1);
digitalWrite(24, 0);
cout<<"Left2"<<endl;
   }

      else if (Result <-20)
   {
digitalWrite(21, 0);
digitalWrite(22, 1);    //decimal = 6
digitalWrite(23, 1);
digitalWrite(24, 0);
cout<<"Left3"<<endl;
   }

   Stop_Sign:
   Object:

  if (laneEnd > 4500)
   {
     ss.str(" ");
     ss.clear();
     ss<<" Lane End";
     putText(frame, ss.str(), Point2f(1,50), 0,1, Scalar(255,0,0), 2);

    }

   else if (Result == 0)
   {
     ss.str(" ");
     ss.clear();
     ss<<"Result = "<<Result<<" (Move Forward)";
     putText(frame, ss.str(), Point2f(1,50), 0,1, Scalar(0,0,255), 2);

    }

   else if (Result > 0)
   {
     ss.str(" ");
     ss.clear();
     ss<<"Result = "<<Result<<" (Move Right)";
     putText(frame, ss.str(), Point2f(1,50), 0,1, Scalar(0,0,255), 2);

    }
       else if (Result < 0)
```

```
{
  ss.str(" ");
  ss.clear();
  ss<<"Result = "<<Result<<" (Move Left)";
  putText(frame, ss.str(), Point2f(1,50), 0,1, Scalar(0,0,255), 2);

 }

namedWindow("orignal", WINDOW_KEEPRATIO);
moveWindow("orignal", 0, 100);
resizeWindow("orignal", 640, 480);
imshow("orignal", frame);

namedWindow("Perspective", WINDOW_KEEPRATIO);
moveWindow("Perspective", 640, 100);
resizeWindow("Perspective", 640, 480);
imshow("Perspective", framePers);

namedWindow("Final", WINDOW_KEEPRATIO);
moveWindow("Final", 1280, 100);
resizeWindow("Final", 640, 480);
imshow("Final", frameFinal);

namedWindow("Stop Sign", WINDOW_KEEPRATIO);
moveWindow("Stop Sign", 1280, 580);
resizeWindow("Stop Sign", 640, 480);
imshow("Stop Sign", RoI_Stop);

namedWindow("Object", WINDOW_KEEPRATIO);
moveWindow("Object", 640, 580);
resizeWindow("Object", 640, 480);
imshow("Object", RoI_Object);

namedWindow("Traffic", WINDOW_KEEPRATIO);
moveWindow("Traffic", 0, 580);
resizeWindow("Traffic", 640, 480);
imshow("Traffic", RoI_Traffic);

waitKey(1);
auto end = std::chrono::system_clock::now();
std::chrono::duration<double> elapsed_seconds = end-start;

float t = elapsed_seconds.count();
int FPS = 1/t;
```

```
    cout<<"FPS = "<<FPS<<endl;

    }

    return 0;
}
}
```

ДОДАТОК Ж

Лістинг модуля підпорядкованого пристрою

```
int i =0;
unsigned long int j =0;

const int EnableL = 5;
const int HighL = 6;       // LEFT SIDE MOTOR
const int LowL =7;

const int EnableR = 10;
const int HighR = 8;       //RIGHT SIDE MOTOR
const int LowR =9;

const int D0 = 0;       //Raspberry pin 21    LSB
const int D1 = 1;       //Raspberry pin 22
const int D2 = 2;       //Raspberry pin 23
const int D3 = 3;       //Raspberry pin 24    MSB

int a,b,c,d,data;


void setup() {

pinMode(EnableL, OUTPUT);
pinMode(HighL, OUTPUT);
pinMode(LowL, OUTPUT);


pinMode(EnableR, OUTPUT);
pinMode(HighR, OUTPUT);
pinMode(LowR, OUTPUT);

pinMode(D0, INPUT_PULLUP);
pinMode(D1, INPUT_PULLUP);
pinMode(D2, INPUT_PULLUP);
pinMode(D3, INPUT_PULLUP);


}

void Data()
{
  a = digitalRead(D0);
```

```
  b = digitalRead(D1);
  c = digitalRead(D2);
  d = digitalRead(D3);

  data = 8*d+4*c+2*b+a;
}

void Forward()
{
 digitalWrite(HighL, LOW);
 digitalWrite(LowL, HIGH);
 analogWrite(EnableL,255);

 digitalWrite(HighR, LOW);
 digitalWrite(LowR, HIGH);
 analogWrite(EnableR,255);

}


void Backward()
{
 digitalWrite(HighL, HIGH);
 digitalWrite(LowL, LOW);
 analogWrite(EnableL,255);

 digitalWrite(HighR, HIGH);
 digitalWrite(LowR, LOW);
 analogWrite(EnableR,255);

}

void Stop()
{
 digitalWrite(HighL, LOW);
 digitalWrite(LowL, HIGH);
 analogWrite(EnableL,0);

 digitalWrite(HighR, LOW);
 digitalWrite(LowR, HIGH);
 analogWrite(EnableR,0);

}
```

```
void Left1()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,160);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,255);

}

void Left2()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,90);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,255);

}


void Left3()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,50);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,255);

}

void Right1()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,255);

  digitalWrite(HighR, LOW);
```

```
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,160);  //200

}
void Right2()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,255);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,90);   //160

}

void Right3()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,255);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,50);   //100

}

void UTurn()
{
  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);
  delay(400);

  analogWrite(EnableL, 250);
  analogWrite(EnableR, 250);    //forward
  delay(1000);

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);
  delay(400);

  digitalWrite(HighL, HIGH);
  digitalWrite(LowL, LOW);
```

```
  digitalWrite(HighR, LOW);   //   left
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 255);
  analogWrite(EnableR, 255);
  delay(700);

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);
  delay(400);

  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, LOW);   // forward
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 255);
  analogWrite(EnableR, 255);
  delay(900);

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);
  delay(400);

  digitalWrite(HighL, HIGH);
  digitalWrite(LowL, LOW);
  digitalWrite(HighR, LOW);    //left
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 255);
  analogWrite(EnableR, 255);
  delay(700);


  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);
  delay(1000);

  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL, 150);
  analogWrite(EnableR, 150);
  delay(300);
}
```

```
void Object()
{

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);          //stop
  delay(1000);

  digitalWrite(HighL, HIGH);
  digitalWrite(LowL, LOW);
  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);       //left
  analogWrite(EnableL, 250);
  analogWrite(EnableR, 250);
  delay(500);

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);          //stop
  delay(200);

  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);         //forward
  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 255);
  analogWrite(EnableR, 255);
  delay(1000);

  analogWrite(EnableL, 0);         //stop
  analogWrite(EnableR, 0);
  delay(200);

  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, HIGH);       //right
  digitalWrite(LowR, LOW);
  analogWrite(EnableL, 255);
  analogWrite(EnableR, 255);
  delay(500);

  analogWrite(EnableL, 0);            //stop
  analogWrite(EnableR, 0);
  delay(1000);
```

```
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, LOW);       // forward
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 150);
  analogWrite(EnableR, 150);
  delay(500);

   i  = i+1;
}

void Lane_Change()
{

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);          //stop
  delay(1000);

  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, HIGH);
  digitalWrite(LowR, LOW);       //Right
  analogWrite(EnableL, 250);
  analogWrite(EnableR, 250);
  delay(500);

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);          //stop
  delay(200);

  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);         //forward
  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 255);
  analogWrite(EnableR, 255);
  delay(800);

  analogWrite(EnableL, 0);          //stop
  analogWrite(EnableR, 0);
  delay(200);

  digitalWrite(HighL, HIGH);
```

```
  digitalWrite(LowL, LOW);
  digitalWrite(HighR, LOW);        //LEFT
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 255);
  analogWrite(EnableR, 255);
  delay(500);

  analogWrite(EnableL, 0);            //stop
  analogWrite(EnableR, 0);
  delay(1000);


  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, LOW);      // forward
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 150);
  analogWrite(EnableR, 150);
  delay(500);

}

void loop()
{
    if (j > 25000)
    {
      Lane_Change();
      i = 0;
      j = 0;
    }

  Data();
  if(data==0)
   {
     Forward();
     if (i>0)
     {
      j = j+1;
     }
   }

  else if(data==1)
   {
     Right1();
```

```
        if (i>0)
   {
    j = j+1;
   }
 }

else if(data==2)
 {
  Right2();
        if (i>0)
   {
    j = j+1;
   }
 }

else if(data==3)
 {
  Right3();
        if (i>0)
   {
    j = j+1;
   }
 }

else if(data==4)
 {
  Left1();
        if (i>0)
   {
    j = j+1;
   }
 }

else if(data==5)
 {
  Left2();
        if (i>0)
   {
    j = j+1;
   }
 }

else if(data==6)
 {
```

```
    Left3();
        if (i>0)
    {
     j = j+1;
    }
  }

 else if(data==7)
  {
   UTurn();
  }

 else if (data==8)
  {
    analogWrite(EnableL, 0);
    analogWrite(EnableR, 0);
    delay(4000);

    analogWrite(EnableL, 150);
    analogWrite(EnableR, 150);
    delay(1000);
  }

    else if(data==9)
  {
   Object();
  }

       else if(data==10)
  {
    analogWrite(EnableL, 0);
    analogWrite(EnableR, 0);
    delay(2000);
  }


   else if(data>10)
  {
   Stop();
  }

}
```

ДОДАТОК И

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Мікроконтролерна нейромережева система керування рухомим об’єктом в двовимірному просторі

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності
Turnitin

Оригінальність 89% Схожість 11%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Turnitin щодо роботи.

Автор роботи _____ Фурман М.А.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Крупельницький Л.В.
(підпис) (прізвище, ініціали)