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

Пояснювальна записка

до бакалаврської дипломної роботи

на тему: «Мікроконтролерна система керування рухомим об'єктом в двовимірному просторі»

Виконав: студент 4 курсу, групи 1СП-19б
спеціальності 123 — «Комп'ютерна інженерія»
освітня програма — «Системне програмування»
Фурман М.А.

Керівник к.т.н., доц. каф. ОТ:
Крупельницький Л.В.

Рецензент к.т.н., доц каф. ПЗ:
Войтко В.В.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф Азаров О.Д.
"09" червня 2023 р.

м. Вінниця – 2023 рік

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність — комп'ютерна інженерія

Освітня програма – системне програмування

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., професор Азаров О.Д.
"05" лютого 2023 року

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

1 Тема роботи: «Мікроконтролерна 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 "20" березня 2023 року № 67

2 Строк подання студентом роботи: 16.05.2023

3 Вихідні дані до роботи: тип об'єкту — рухомий транспортний засіб; двовимірний простір з розміткою на дорозі; спосіб керування — автоматичний; 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, 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	08.02.2023 дата	01.06.2023 дата
		підпис	підпис

7 Дата видачі завдання: 08.02.2023

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	8.03.2023	виконано
2	Аналіз історії розвитку безпілотних об'єктів	12.03.2023 – 17.03.2023	виконано
3	Пошук та аналіз методів управління самокерованими об'єктами	18.03.2023 – 20.03.2023	виконано
4	Огляд сучасних систем автоматичного керування рухомим об'єктом	22.03.2023 – 24.04.2023	виконано
5	Вибір компонентів для створення системи керування рухомим об'єктом	25.04.2023 – 07.05.2023	виконано
6	Створення програмного забезпечення та налаштування зв'язку між компонентами апаратної складової	08.05.2023 – 22.05.2023	виконано
7	Тестування працездатності системи	23.05.2023 – 24.05.2023	виконано
8	Оформлення пояснювальної записки	25.05.2023 – 31.05.2023	виконано
9	Перевірка якості виконання бакалаврської роботи	1.06.2023 – 14.06.2023	виконано

Студент _____ Фурман М.А.

Керівник роботи _____ Крупельницький Л.В.

АНОТАЦІЯ

УДК 004.9

Фурмана М.А. Мікроконтролерна система керування рухомим об’єктом в двовимірному просторі. Бакалаврський дипломна робота зі спеціальності 123 — Комп’ютерна Інженерія, Вінниця: ВНТУ, 2023. Пояснювальна записка містить 86 сторінок, 40 рисунків, 3 таблиць та 20 посилань.

У бакалаврській роботі розглянуто методи управління самокерованими рухомими об'єктами, проаналізовано алгоритми комп’ютерного зору та обробки зображень у реальному часі.

На основі існуючих розробок, спроектовано систему керування рухомим об’єктом, що здатна самостійно виявляти перешкоди та планувати маршрут на основі зібраних даних з камери.

Ключові слова: комп'ютерний зір, обробка зображень, Arduino, Raspberry Pi, OpenCV.

ABSTRACT

УДК 004.9

M.A. Furman Microcontroller system for controlling a moving object in two-dimensional space. Bachelor thesis on specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2023. The explanatory note contains 86 pages, 40 figures, 3 tables and 20 references.

In the bachelor's thesis, the methods of controlling self-guided moving objects were considered, and the algorithms of computer vision and real-time image processing were analyzed.

On the basis of existing developments, a moving object control system was designed, capable of independently detecting obstacles and planning a route based on the data collected from the camera.

Keywords: computer vision, image processing, Arduino, Raspberry Pi, OpenCV.

ЗМІСТ

ВСТУП

Актуальність проекту полягає в необхідності створення рухомих об'єктів для впровадження розумних технологій та розвитку автономних систем. Системи на основі мікроконтролерів 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 мікроконтролерних систем керування рухомими об'єктами сприяє автоматизації рутинних завдань, зменшенню людського втручання, підвищенню продуктивності та ефективності роботи. Такі системи можуть оптимізувати використання ресурсів, знижувати витрати енергії та матеріалів, а також покращувати якість та безпеку виконання завдань.

Системи керування рухомими об'єктами відкривають нові можливості для досліджень і розвитку інноваційних технологій. В свою чергу, це призводить до створення платформи для впровадження штучного інтелекту, машинного навчання та аналізу даних, що розширює можливості систем керування та підвищує їх адаптивність до змінних умов.

Об'єктом дослідження є процеси керування рухомими об’єктами.

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

Мета дослідження: створення мікроконтролерної системи керування рухомим об’єктом, що здатна виявляти перешкоди та планувати маршрут на основі зібраних даних.

Задачі для здійснення мети дослідження:

— проаналізувати сучасні системи автоматичного керування;

— здійснити огляд методів управління самокерованими об’єктами;

— 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;

— розробити систему керування рухомим об’єктом.

Практичне значення результатів спроектованої системи керування рухомим об’єктом полягає в актуальності досліджень пов’язаних з комп’ютерним зором самокерованих систем за допомогою сучасних алгоритмів обробки зображень.

Апробація результатів проекту здійснена під час виступу на LII науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії 2023 «Методи розпізнавання рухомих об’єктів», Максим Андрійович Фурман, Леонід Віталійович Крупельницький.

Публікація за темою проекту — Фурман, М.; Крупельницький, Л.. Методи розпізнавання рухомих об'єктів. НТКП ВНТУ. Факультет інформаційних технологій та комп'ютерної інженерії, Ukraine, mar. 2023. Available at: <https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/view/17523>. Date accessed: 26 May. 2023. [1]

1 ОГЛЯД І АНАЛІЗ ІСНУЮЧИХ СИСТЕМ КЕРУВАННЯ РУХОМИМИ ОБ'ЄКТАМИ

Безпілотний мобільний об’єкт відноситься до пристрою або системи, які можуть рухатися або працювати без прямого втручання людини. Він призначений для виконання завдань, навігації в середовищах або виконання певних функцій автономно або з дистанційним керуванням.

Безпілотні мобільні об’єкти можуть бути різних форм, включаючи транспортні засоби, роботів, дрони або навіть космічні кораблі. Ці об’єкти оснащені датчиками, виконавчими механізмами та інтелектуальними системами, які дозволяють їм сприймати навколишнє середовище, приймати рішення та виконувати дії без постійного керування людиною чи фізичної присутності. Це дозволяє не обмежуватись сферою їх використання та забезпечує більший рівень безпеки, економію ресурсів і уникнення людського фактору втручання у процес виконання тих чи інших задач [2].

1.1 Історія розвитку керування рухомими об’єктами

Історія безпілотних мобільних об’єктів охоплює низку великих сфер і демонструє винахідливість людей у створенні інтелектуальних, автономних систем. Від ранніх експериментів до сучасних безпілотних технологій ці об’єкти значно еволюціонували з часом.

Походження безпілотних мобільних об’єктів можна простежити до давніх часів, коли люди винайшли механізми, які могли рухатися без прямого втручання людини. Наприклад, стародавні греки створили автомати (див. рис. 1.1), механічні пристрої, здатні виконувати прості завдання. Ці перші прототипи заклали основу для майбутніх розробок безпілотних систем.

Сучасна ера безпілотних мобільних об’єктів почала формуватися в середині 20 століття з появою робототехніки та автоматизації. Під час Другої світової війни дистанційно керовані апарати, такі як безпілотні літальні апарати (БПЛА) (див. рис. 1.2) і безпілотні наземні апарати (БПНА),

використовувалися у військових цілях, зокрема для спостереження та розвідки [3].

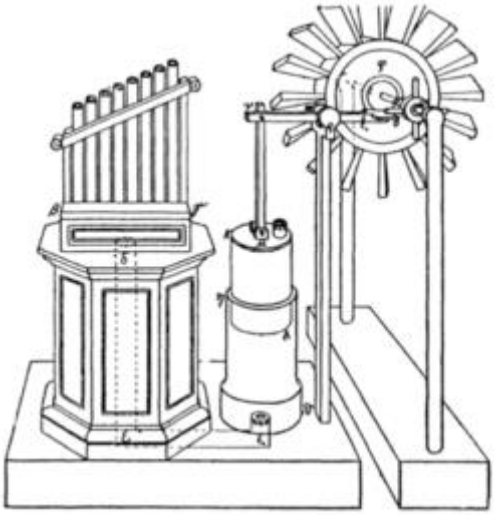

Рисунок 1.1 — Грецький автомат

Рисунок 1.2 — Безпілотний літальний апарат Queen Bee

Після війни акцент на безпілотних мобільних об’єктах перемістився в бік промислового та наукового застосування [4]. Роботизовані руки були розроблені для конвеєрних ліній, що дозволяє автоматизувати та підвищити ефективність виробничих процесів. Також безпілотні підводні апарати (UUV) були розроблені для розвідки та дослідження морського середовища, як на рисунку 1.3.

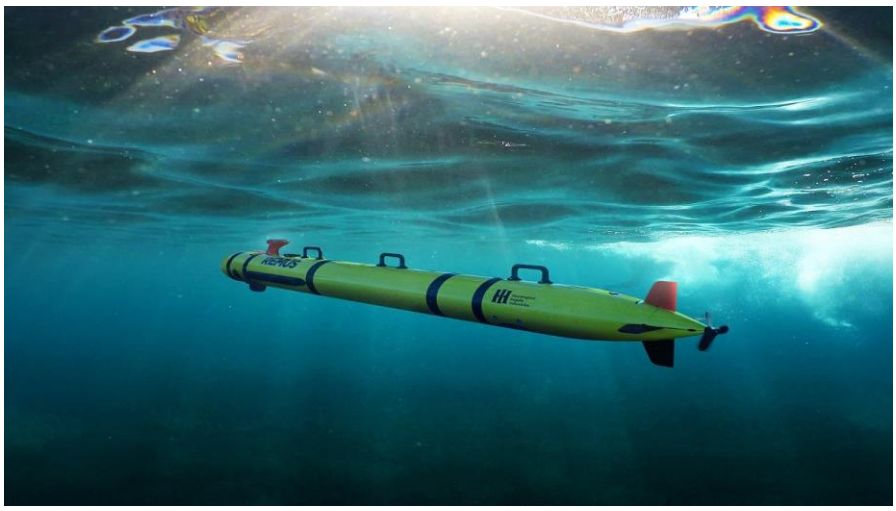

Рисунок 1.3 — Безпілотний підводний апарат REMUS 300

У другій половині 20-го століття прогрес в електроніці, обчислювальній техніці та телекомунікаціях сприяв подальшому розвитку безпілотних мобільних об’єктів. Мініатюризація електронних компонентів і зростання потужності мікропроцесорів зробили можливим створення менших, більш потужних безпілотних систем.

У сфері освоєння космосу безпілотні мобільні об'єкти зіграли вирішальну роль. Місія Радянського Союзу «Луна-2» у 1959 році стала першим космічним кораблем, який досяг Місяця, знаменуючи важливу віху в безпілотному дослідженні космосу. Відтоді безпілотні космічні зонди, такі як космічний апарат «Вояджер» (див. рис. 1.4) і марсоходи, надали цінні дані та ідеї про нашу Сонячну систему та за її межами.

У 21 столітті безпілотні мобільні об'єкти стали більш поширеними та різноманітними. Безпілотні літальні апарати, широко відомі як дрони, набули широкої популярності для різних застосувань, включаючи аерофотозйомку, спостереження, служби доставки та навіть розваги. Розвиток технологій безпілотних літальних апаратів зробив їх більш доступними для широкого загалу для цивільного використання.

Рисунок 1.4 — Космічний апарат «Вояджер»

У сфері транспорту розробка безпілотних автомобілів зробила революцію в автомобільній промисловості (див. рис. 1.5). Ці безпілотні транспортні засоби використовують вдосконалені датчики, камери та алгоритми штучного інтелекту для навігації дорогами та взаємодії з оточенням. Безпілотні автомобілі обіцяють покращити безпеку на дорогах, зменшити затори та підвищити ефективність транспорту [5].

Рисунок 1.5 — Безпілотний автомобіль Waymo

Безпілотні мобільні об’єкти також знайшли застосування в таких сферах, як сільське господарство, будівництво, охорона здоров’я та реагування на стихійні лиха. Сільськогосподарські дрони використовуються для моніторингу посівів і обприскування, а безпілотні будівельні машини допомагають на небезпечних або віддалених будівельних майданчиках. Безпілотні літальні та підводні апарати допомагають у пошуково-рятувальних операціях і екологічному моніторингу. Постійний прогрес у сфері штучного інтелекту, машинного навчання та сенсорних технологій продовжить розширювати межі можливостей цих об’єктів.

Підсумовуючи, історія безпілотних мобільних об’єктів відображає прагнення людства до автоматизації та автономності. Від стародавніх автоматів до сучасних дронів і безпілотних автомобілів ці об’єкти еволюціонували разом із технологічним прогресом. Оскільки постійно продовжується впровадження інновацій та вдосконалення таких систем, безпілотні мобільні об’єкти відіграватимуть дедалі важливішу роль у формуванні нашого майбутнього, надаючи нові можливості та прогрес у багатьох секторах діяльності суспільства.

1.2 Класифікація методів управління самокерованими об’єктами

Управління безпілотними об’єктами передбачає впровадження різних методів для забезпечення їх точної навігації, поведінки та продуктивності. Ці методи дозволяють операторам або автономним системам здійснювати контроль над безпілотними об’єктами, направляючи їх по бажаних шляхах або діях (див. рис. 1.6).

Одним із методів є ручне дистанційне керування, коли людина-оператор використовує пристрій дистанційного керування для безпосереднього керування рухами та діями безпілотного об’єкта [6]. Цей метод зазвичай використовується для дистанційно керованих літаків, дронів або підводних дистанційно керованих апаратів (ROV). Оператор може керувати швидкістю,

напрямком, висотою об’єкта та виконувати певні завдання, маніпулюючи засобами керування.

Іншим методом є навігація за маршрутними точками, коли безпілотні об’єкти запрограмовані на слідування заздалегідь визначеним маршрутам або маршрутним точкам [7]. Оператор або автономна система вводить ряд координат або маршрутних точок, якими має пройти безпілотний об’єкт. Бортова навігаційна система об’єкта використовує датчики, такі як GPS або інерціальні навігаційні системи, щоб відстежувати його положення та коригувати його рух для точного досягнення кожної маршрутної точки.

Для розрахунку оптимальних шляхів або траєкторій для безпілотних об’єктів використовуються методи планування шляху та формування траєкторії [8]. Алгоритми та математичні моделі використовуються для визначення найбільш ефективних або бажаних шляхів, враховуючи такі фактори, як уникнення перешкод, часові обмеження, споживання енергії або конкретні цілі місії. Ці методи часто використовуються в робототехніці, автономних транспортних засобах або космічних місіях для планування безпечних і ефективних маршрутів.

Системи управління зі зворотним зв'язком відіграють вирішальну роль в управлінні безпілотними об'єктами в режимі реального часу [9]. Ці системи використовують датчики для вимірювання стану чи продуктивності об’єкта, порівняння його з бажаними значеннями чи еталонними сигналами та застосування керуючих дій для мінімізації помилки чи відхилення. Це дозволяє безперервно коригувати та стабілізувати поведінку безпілотного об’єкта. Управління зі зворотним зв’язком зазвичай використовується в автономних транспортних засобах, роботизованих системах і промисловій автоматизації.

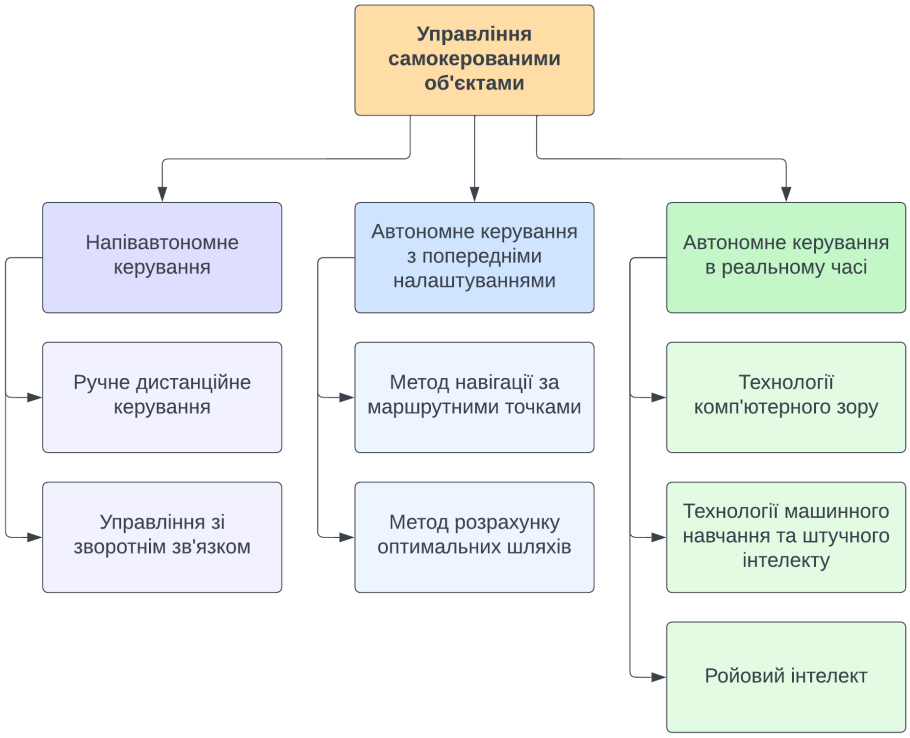

Рисунок 1.6 — Класифікація методів управління самокерованими об’єктами

Технології машинного навчання та штучного інтелекту все частіше використовуються для керування безпілотними об’єктами [10]. Ці методи дозволяють об’єкту навчатися на даних і минулому досвіді, дозволяючи йому адаптуватися та самостійно приймати обґрунтовані рішення. Алгоритми машинного навчання можна навчити розпізнавати шаблони, прогнозувати поведінку та оптимізувати дії керування на основі великих наборів даних. Цей підхід часто використовується в автономних системах, таких як самокеровані автомобілі або інтелектуальні роботи.

Технології комп’ютерного зору та обробки зображень використовуються для аналізу візуальної інформації та забезпечення зворотного зв’язку для керування безпілотними об’єктами [11]. Обробляючи візуальні дані в реальному часі від камер або інших датчиків, система може розпізнавати об’єкти, виявляти особливості, оцінювати відстані або відстежувати рух. Це дозволяє об’єкту реагувати на навколишнє середовище та відповідно регулювати свої рухи.

Новітні технології, такі як ройовий інтелект, дозволяють координувати та контролювати декілька безпілотних об’єктів як колективну групу [12]. Натхненні такими природними системами, як колонії комах, алгоритми ройового інтелекту дозволяють об’єктам взаємодіяти, спілкуватися та координувати свої дії для досягнення спільних цілей. Цей підхід застосовується в таких галузях, як ройова робототехніка, де групи роботів працюють разом, щоб виконувати складні завдання.

Таким чином, керування безпілотними об’єктами включає низку методів і технік, адаптованих до конкретних застосувань і цілей. Ці методи охоплюють ручне дистанційне керування, навігацію за маршрутними точками, планування шляху, системи контролю зі зворотним зв’язком, машинне навчання, комп’ютерне зір і ройовий інтелект. Використовуючи ці методи, оператори або автономні системи можуть ефективно керувати безпілотними об’єктами, дозволяючи їм точно виконувати завдання, адаптуватися до мінливих умов і досягати бажаних результатів.

1.3 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.

Roomba Combo від iRobot є популярним прикладом системи на основі мікроконтролера, яка використовується для керування рухомим об’єктом у двовимірному просторі, зокрема в контексті робота-пилососа, як на рисунку 1.7.

Рисунок 1.7 — Автоматичний пилосос Roomba Combo

З інженерної точки зору, Roomba Combo — це компактний робот-пилосос круглої форми. Верхня поверхня робота пилососа має центральну панель керування з ручного увімкнення та індикаторами стану [13]. Знизу пристрій оснащено кількома ключовими компонентами, зокрема:

— обертові щітки з щетиною та гумові екстрактори, які ефективно знімають бруд, пил і сміття з різних поверхонь підлоги;

— два основних колеса, які забезпечують тягу та мобільність, дозволяючи йому рухатися вперед, назад і маневрувати в просторі;

— ряд датчиків для навігації, виявлення перешкод і аналізу підлоги;

— сміттєвий бак для збору сміття, що є доступним для спорожнення та чищення;

— живлення від акумуляторної батареї та док-станція для заряджання, яка служить місцем для зберігання пристрою та точкою зарядки.

Ця комбінація функцій дозволяє Roomba Combo автономно змінювати своє положення, прибирати різні поверхні підлоги, виявляти та уникати перешкоди.

Компактний дизайн, набір датчиків і розширені функції роблять пристрій універсальним і надійним роботом-пилососом. Зважаючи на ціну пристрою, що сягає 11000 гривень, прилад входить у середній ціновий сегмент.

Adept Lynx AGV від Omron — це автономний керований транспортний засіб, розроблений для виконання завдань із транспортування матеріалів у промислових умовах [14]. Цей AGV має компактну та міцну конструкцію, оснащену передовими системами на основі мікроконтролерів, що зображено на рисунку 1.8.

Рисунок 1.8 —Автоматичний вантажний транспортер Adept Lynx AGV

Основні функціональні особливості AGV Adept Lynx включають:

— лазерні сканери, встановлені на AGV, постійно сканують навколишнє середовище, виявляючи перешкоди та визначаючи орієнтири для навігації;

— складні алгоритми аналізують нанесене на карту середовище, обчислюють оптимальні шляхи та динамічно коригують траєкторії AGV, щоб уникнути перешкоди і забезпечити ефективний рух;

— настроювані та легко регульовані вантажні деки розроблені для роботи з різними корисними навантаженнями та з можливістю вибору різної

вантажопідйомності для розміщення різних розмірів і типів матеріалів або продукції;

— спілкування з іншими машинами, конвеєрами або системами управління складом, забезпечуючи ефективну координацію та синхронізацію операцій з обробки матеріалів;

— численні механізми безпеки, такі як датчики для виявлення людей або об’єктів поблизу, кнопки аварійної зупинки та попереджувальні сигнали для забезпечення безпечної роботи за наявності персоналу чи перешкод.

AGV Adept Lynx — це добре сконструйований пристрій, який забезпечує надійні та ефективні можливості транспортування матеріалів. Гнучкість AGV, вантажопідйомність і можливості інтеграції робить його універсальними та адаптованими до різноманітних вимог. Але через зависоку ціну, яка сягає 1000000 гривень, такий пристрій не є доступним для звичайних користувачів.

Самобалансуючий електричний самокат Segway Ninebot S від Segway-Ninebot є інноваційним персональним транспортним засобом, який поєднує в собі передові технології для плавної та інтуїтивно зрозумілої їзди [15]. Цей рухомий пристрій має компактну конструкцію з міцною рамою та центрально розташованою рульовою колонкою, як зображено на рисунку 1.9.

Рисунок 1.9 —Транспортний засіб Segway Ninebot S

Основні функціональні особливості Segway Ninebot S:

— складні датчики, такі як гіроскопи та акселерометри, для постійного вимірювання центру ваги водія та автоматичного регулювання положення пристрою;

— система електроприводу, яка включає потужний двигун і акумуляторну батарею, що забезпечує ефективну тягу, дозволяючи скутеру розвивати швидкість до 16 та 20 км/год і справлятися з ухилами до 15 градусів;

— ергономічне кермо, що при нахилі у потрібному напрямку реагує на рухи водія, забезпечуючи плавний та точний рух;

— вбудований світлодіодний дисплей, який надає інформацію в реальному часі, демонструючи швидкість, рівень заряду акумулятора та режими їзди;

— передні та задні світлодіодні ліхтарі для видимості в умовах слабкого освітлення;

— подвійна гальмівна система з рекуперативним гальмуванням і механічним ножним гальмом, що забезпечує надійну зупинку та рекуперацію енергії під час уповільнення.

Segway Ninebot S — це добре сконструйований самобалансуючий електричний транспортний засіб, який поєднує в собі передові технології, інтуїтивно зрозуміле керування та функції безпеки. Слабким місцем цього пристрою можна вважати безпеку, адже при використанні користувачем важливо тримати баланс та слідкувати за рухом на дорозі.

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							
Roomba Combo	4	4	4	3	5	4	24
Adept Lynx AGV	5	5	5	4	3	2	24
Segway Ninebot S	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 розділ інженерії та математики, який займається розробкою та аналізом систем, які можуть працювати та регулювати себе без прямого втручання людини. Він охоплює широкий спектр застосувань, від простих побутових приладів до складних автономних транспортних засобів. Основною метою автоматичного керування є розробка алгоритмів, які дозволяють системам досягти бажаної продуктивності, стабільності та надійності [16].

В основі теорії автоматичного керування лежить концепція зворотного зв'язку (див. рис. 2.1), в якій безперервно відстежують вихід або поведінку системи та порівнюють їх із бажаним еталонним або заданим значенням. На основі цього порівняння система управління генерує відповідні коригувальні дії для підтримки або коригування поведінки системи. Ця петля зворотного зв'язку дозволяє системам управління адаптуватися та реагувати на зміни в системі або її середовищі, забезпечуючи досягнення бажаних цілей.

Розробка системи автоматичного керування зазвичай включає кілька ключових етапів. По-перше, чітко визначені вимоги та цілі системи. Це включає визначення бажаних критеріїв ефективності, таких як точність, швидкість, стабільність і придушення перешкод. Далі система аналізується, щоб зрозуміти її динаміку та поведінку. Математичні моделі, такі як диференціальні рівняння або функції передачі, виводяться для опису реакції системи на входи та збурення.

Після того, як система добре зрозуміла, вибирається стратегія контролю. Можуть бути застосовані різні методи керування, починаючи від простих пропорційно-інтегрально-похідних (PID) регуляторів до більш просунутих методів, таких як прогнозне керування моделлю або адаптивне керування. Вибір стратегії керування залежить від таких факторів, як складність системи, бажана продуктивність і доступні ресурси.

Наступним кроком є розробка алгоритму керування, що полягає в розробці математичних рівнянь і алгоритмів, які генеруватимуть керуючі сигнали на основі зворотного зв’язку системи. Алгоритм керування має бути стійким, ефективним і здатним досягати бажаних цілей за різних робочих умов.

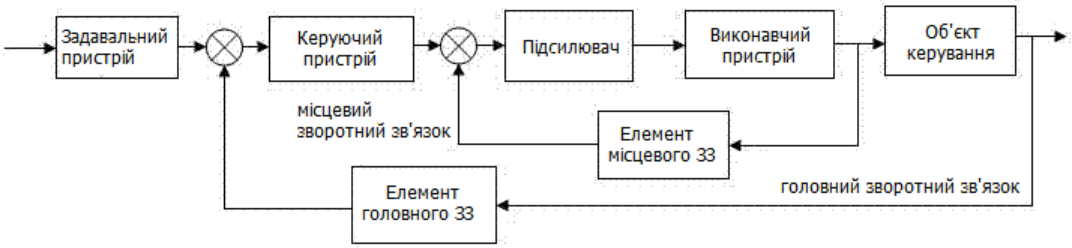

Рисунок 2.1 — Типова схема системи автоматичного керування

Після розробки алгоритму керування його необхідно реалізувати у відповідній апаратній чи програмній платформі, включаючи як програмування мікроконтролерів, так і розробку керуючого програмного забезпечення або налаштування програмованих логічних контролерів (ПЛК). Етап реалізації також включає налаштування датчиків і виконавчих механізмів для взаємодії з системою та забезпечення необхідного зворотного зв’язку та сигналів керування.

Закінчивши етап впровадження системи керування, її тестують і налаштовують, щоб переконатися, що вона відповідає бажаним критеріям ефективності. Важливо взяти до уваги оцінку реакції системи на різні вхідні сигнали та збурення, коригування параметрів керування та оптимізацію поведінки системи. Це може вимагати повторного вдосконалення та коригування для досягнення бажаного рівня продуктивності.

Таким чином, теорія автоматичного керування забезпечує основу для проектування та аналізу систем, які можуть працювати автономно. Адже з її допомогою можна вивчати динаміку системи, обирати доцільні стратегії керування, розробляти відповідні алгоритми та впроваджувати їх у апаратне

чи програмне забезпечення, створюючи точну та ефективну конфігурацію системи.

2.1 Модель зв’язку між компонентами системи

У проектуванні структури будь-якої системи необхідно проаналізувати та обрати модель зв’язку між компонентами, яка забезпечить ефективну та якісну взаємодію. Тому для пристрою системи керування рухомим об’єктом було обрано технологію master/slave (див. рис. 2.2). Впроваджуючи головний пристрій для нагляду та координації дій підлеглих пристроїв, ця архітектура пропонує низку важливих переваг [17].

По-перше, ця модель зв’язку забезпечує централізоване управління, дозволяючи ефективно приймати рішення, координувати та синхронізувати рухи об’єкта.

По-друге, полегшується розподіл робочого навантаження, даючи змогу паралельно обробляти дані та оптимально використовувати ресурси системи. Масштабованість і модульність дозволяють легко інтегрувати додаткові підлеглі пристрої, пристосовуючись до мінливих вимог і сприяючи поступовому розвитку системи.

Додатково у даній технології присутня відмовостійкість і резервування, які підвищують надійність системи, оскільки головний пристрій може виявляти та компенсувати збої або несправності підлеглих пристроїв.

Для того, щоб скористатися цією моделлю зв’язку, було обрано бібліотеку WiringPi, що має необхідний функціонал для використання.

WiringPi — це бібліотека для мови програмування C, яка дозволяє керувати вводом/виводом (GPIO) на одноплатних комп'ютерах, таких як Raspberry Pi та надає простий і зрозумілий інтерфейс для взаємодії з GPIO-портами пристрою [18].

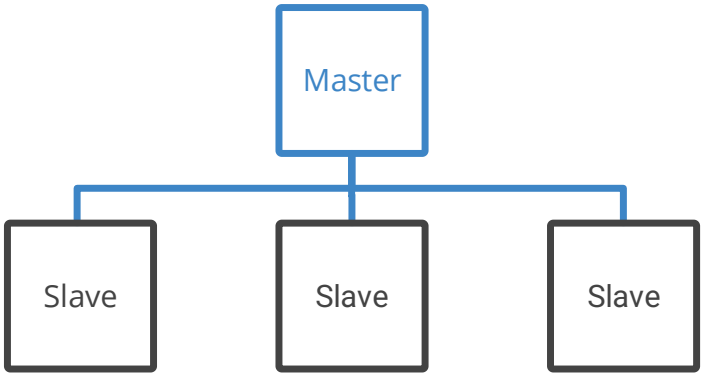

Рисунок 2.2 — Структурна схема технології master/slave

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.

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

— простий інтерфейс, що полегшує роботу з GPIO-портами та комунікацію між пристроями;

— розширений набір функцій для взаємодії з GPIO-портами та шинами, такими як I2C і SPI, що дозволяє зручно реалізувати master-slave комунікацію;

— широкий спектр пристроїв, що спрощує роботу з різними сенсорами. актуаторами та іншими пристроями, що використовуються у системі master-slave.

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

2.2 Другорядний пристрій керування

В якості другорядного пристрою необхідно обрати мікроконтролер, що буде підпорядковуватись керуючому пристрою та контролюватиме частину модулів системи.

Сімейство мікроконтролерів Arduino складається з різних моделей плат, які мають вбудований мікроконтролер і призначені для розробки електронних проектів. Кожна модель має свої особливості та характеристики, але всі вони базуються на одному з основних мікроконтролерів AVR або ARM. Найпопулярніші моделі Arduino включають Arduino Uno, Arduino Mega, Arduino Nano та Arduino Due.

Arduino Uno — широко використовувана плата, популярна серед новачків і любителів. Він забезпечує хороший баланс функцій і простоти, що робить його придатним для широкого кола проектів. Uno має достатню кількість цифрових і аналогових входів/виходів для більшості базових додатків, а його помірна обчислювальна потужність і об'єм пам'яті можуть виконувати прості завдання та керувати різними електронними компонентами.

Arduino Mega розроблений для проектів, які потребують більшої кількості вхідних/вихідних контактів і більше пам'яті (див. рис. 2.3). Завдяки великій кількості контактів і збільшеному об'єму пам'яті Mega може працювати зі складнішими програмами та підключатися до більшої кількості зовнішніх пристроїв. Він часто використовується в проектах, пов'язаних із робототехнікою, автоматизацією та реєстрацією даних, де потрібно контролювати кілька датчиків і виконавчих механізмів.

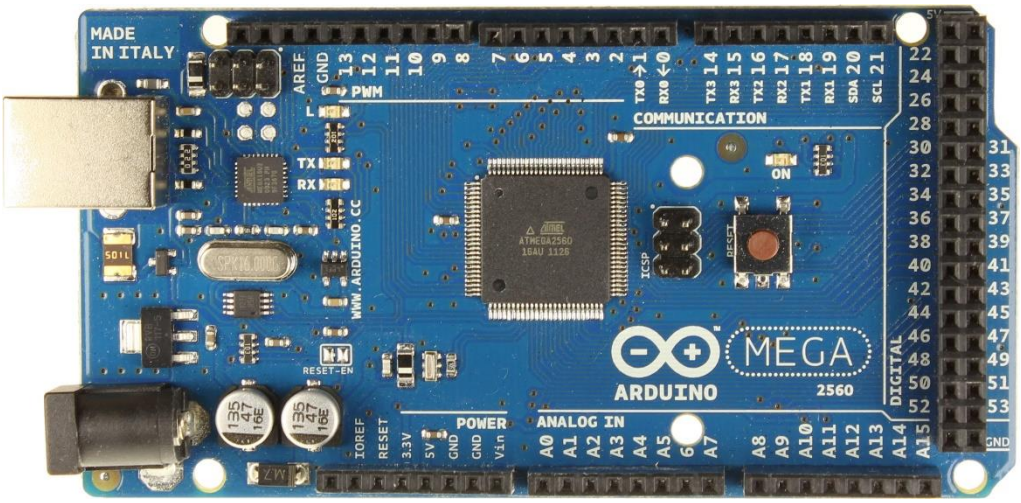

Рисунок 2.3 — Компоновка Arduino Mega

Arduino Nano — це компактна та економічна плата, яка добре підходить для проектів із обмеженим простором або коли потрібен менший форм-фактор (див. рис. 2.4). Незважаючи на невеликий розмір, Nano зберігає багато функцій, властивих більшим платам Arduino. Він зазвичай використовується в носимих пристроях, невеликих проектах IoT і вбудованих системах.

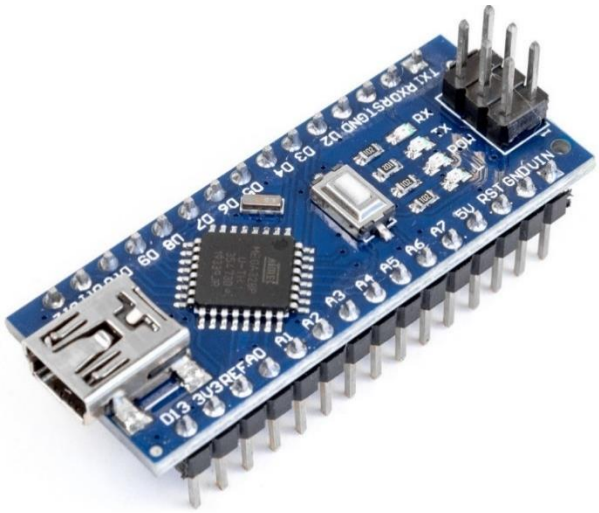

Рисунок 2.4 — Компоновка Arduino Nano

Arduino Due вирізняється з-поміж інших моделей тим, що має 32-розрядний процесор ARM Cortex-M3, що забезпечує значно більшу обчислювальну потужність і об’єм пам’яті. Це робить його ідеальним для додатків, які вимагають високопродуктивних обчислень, таких як обробка аудіо, передова робототехніка та складна маніпуляція даними. Due також пропонує додаткові функції, такі як цифро-аналогові перетворювачі (DAC) і більш просунуті протоколи зв’язку.

Для програмування та завантаження коду на платформу Arduino, було використано Arduino IDE, що показано на рисунку 2.5.

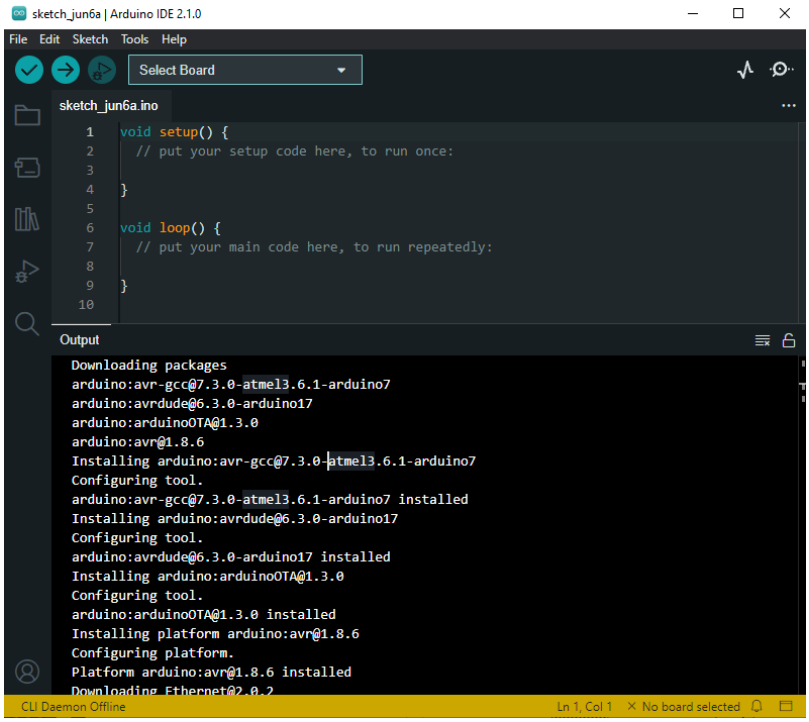

Рисунок 2.5 — Інтерфейс середовища розробки Arduino IDE

Arduino IDE забезпечує зручне середовище розробки, яке дозволяє створювати програми з легкістю, без попереднього досвіду у програмуванні мікроконтролерів завдяки інтуїтивному інтерфейсу та простому функціоналу.

Основні особливості Arduino IDE 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 зробити висновок, що Arduino IDE значно спрощує написання коду для мікроконтролерів Arduino та надає низку корисних особливостей під час користування.

2.3 Головний пристрій керування

В якості головного пристрою необхідно обрати одноплатний комп'ютер, що буде контролювати усі підпорядковані компоненти системи, раціонально розподіляти ресурси та надаватиме зручний інтерфейс для конфігурування.

Одноплатні комп'ютери Raspberry Pi — це компактні та доступні пристрої, які пропонують широкий набір функцій, дозволяючи користувачам підключати та керувати зовнішніми електронними компонентами, такими як датчики, двигуни та світлодіоди. Це робить Raspberry Pi універсальним для різноманітних проектів, включаючи домашню автоматизацію, робототехніку та програми Інтернету речей (IoT).

Крім того, плати Raspberry Pi оснащені портами USB, підключенням до Ethernet і виходами HDMI, що дозволяє легко підключати периферійні пристрої та дисплеї. Іншою особливістю комп'ютерів Raspberry Pi є їх сумісність з різними операційними системами, включаючи такі дистрибутиви Linux, як Raspbian (тепер відома як Raspberry Pi OS) і Ubuntu. Ця гнучкість дозволяє користувачам вибрати ОС, яка відповідає їхнім потребам і забезпечує знайоме обчислювальне середовище. Крім того, моделі Raspberry Pi мають різні рівні обчислювальної потужності та пам'яті, що задовольняють різні вимоги.

Наприклад, Raspberry Pi 4 Model B, випущений у 2019 році, пропонує значне підвищення продуктивності завдяки чотирьохядерному процесору ARM Cortex-A72, до 8 ГБ оперативної пам'яті та підтримці відтворення відео 4K (див. рис. 2.6). Деякі помітні приклади моделей Raspberry Pi включають Raspberry Pi Zero, який неймовірно маленький і економічно ефективний, що робить його придатним для проектів з обмеженим простором або бюджетом.

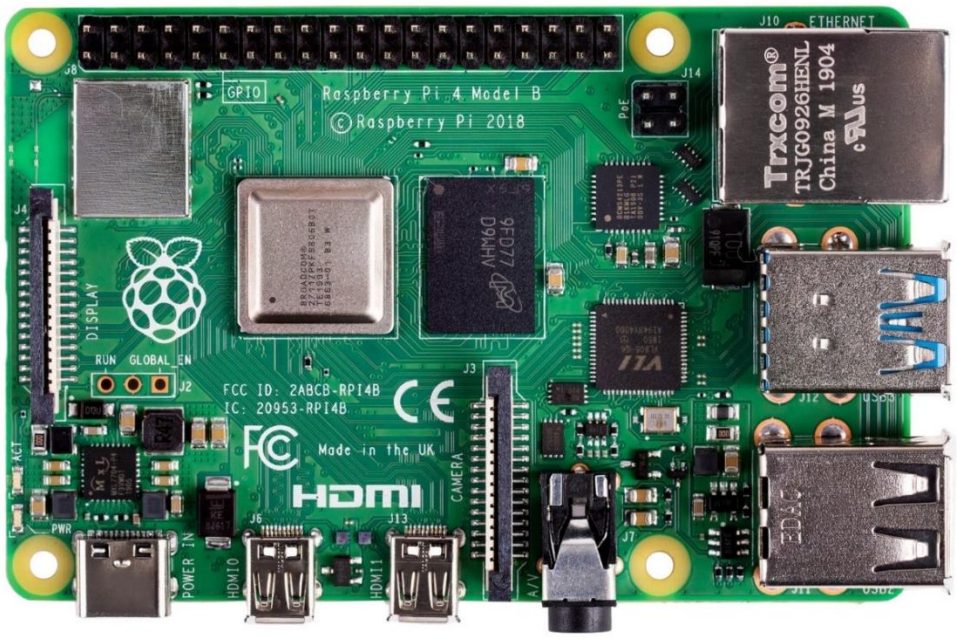

Рисунок 2.6 — Компоновка Raspberry Pi 4 Model B

Raspberry Pi 3 Model B є популярним вибором завдяки вбудованим можливостям Wi-Fi і Bluetooth, які пропонують легке бездротове підключення.

Для програмування комп'ютеру Raspberry Pi використано інтегроване середовище розробки Geany (див. рис. 2.7) з відкритим вихідним кодом.

Geany розроблений, щоб забезпечити легке та ефективне середовище кодування, пропонуючи необхідні функції для розробки програмного забезпечення та зручний інтерфейс, який полегшує навігацію та роботу.

Geany підтримує кілька мов програмування, включаючи C, C++, Python, Java, PHP тощо. Редактор забезпечує підсвічування синтаксису, згортання коду, автозавершення та розумні відступи, які допомагають підвищити продуктивність і підвищити читабельність коду.

Однією з істотних переваг Geany є його швидкість і ефективність під час редагування та компіляції коду. Він займає невеликий обсяг пам'яті та не потребує великих системних ресурсів, що робить його придатним для пристроїв низького класу з обмеженими технічними характеристиками. Geany пропонує інтегровані інструменти, настроювані команди збирання, для взаємодії з різними компіляторами та системами збирання.

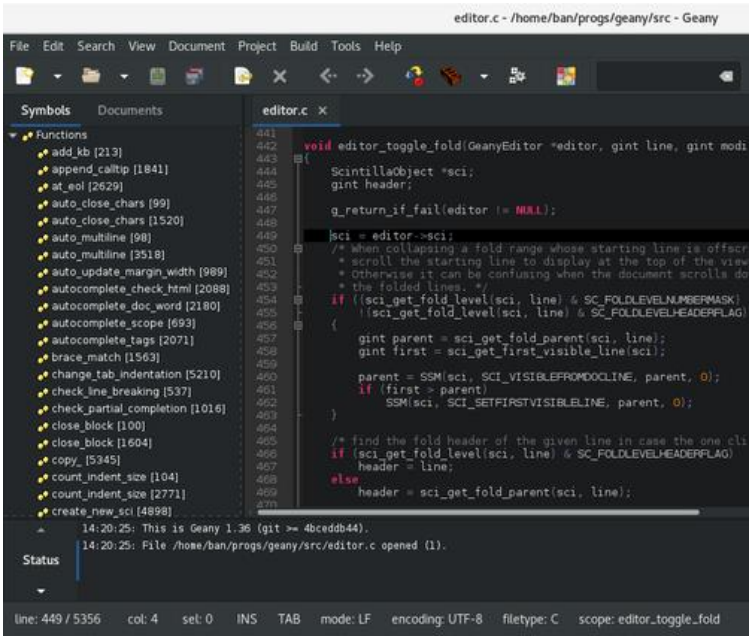

Рисунок 2.7 — Інтерфейс середовища розробки Geany

Geany підтримує кілька мов програмування, включаючи C, C++, Python, Java, PHP тощо. Редактор забезпечує підсвічування синтаксису, згортання коду, автозавершення та розумні відступи, які допомагають підвищити продуктивність і підвищити читабельність коду.

Однією з істотних переваг Geany є його швидкість і ефективність під час редагування та компіляції коду. Він займає невеликий обсяг пам’яті та не потребує великих системних ресурсів, що робить його придатним для пристроїв низького класу з обмеженими технічними характеристиками. Geany пропонує інтегровані інструменти, настроювані команди збирання, для взаємодії з різними компіляторами та системами збирання.

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.

2.4 Модуль відеокамери для введення зображень

Для отримання інформації про простір навколо рухомого об’єкту потрібно обрати модуль відеокамери, що забезпечить систему керування вхідними зображеннями і наддасть можливість використання їх для аналізу та прийняття рішень.

Модулі камери для Raspberry Pi надають можливість знімати зображення та записувати відео безпосередньо з плати Raspberry Pi, додаючи візуальний елемент до проектів. Два найпоширеніші модулі камери для Raspberry Pi — це Raspberry Pi Camera Module V2 і Raspberry Pi High-Quality Camera.

Модуль камери Raspberry Pi Camera Module V2 є популярним вибором для фото- та відеозйомки загального призначення. Він оснащений 8-мегапіксельним датчиком зображення Sony IMX219 і підтримує максимальну роздільну здатність 3280 x 2464 пікселів для фотографій і запису відео 1080p зі швидкістю 30 кадрів на секунду. Модуль камери підключається до плати Raspberry Pi за допомогою стрічкового кабелю та використовує роз’єм CSI (Camera Serial Interface). Це компактний модуль, який легко монтується та інтегрується в різні проекти.

Raspberry Pi High-Quality Camera — це вдосконалений варіант модуля камери, який пропонує більшу гнучкість і якість зображення (див. рис. 2.8). Він має 12,3-мегапіксельний датчик зображення Sony IMX477, який дозволяє знімати зображення та відео з вищою роздільною здатністю. Модуль підтримує змінні об’єктиви з байонетом C або CS, забезпечуючи можливість змінювати фокусну відстань. Високоякісна камера підключається до Raspberry Pi за допомогою стрічкового кабелю. Модуль підтримує ручне налаштування фокусу та захоплення зображень у форматі RAW, що забезпечує розширені можливості постобробки.

Рисунок 2.8 — Модуль Raspberry Pi High-Quality Camera

Камера Raspberry Pi High-Quality Camera підходить для додатків, які вимагають вищої якості зображення, наприклад для фотографії, професійної обробки зображень і машинного зору. Для порівняння, Raspberry Pi Camera Module V2 є більш компактним і доступним варіантом, який пропонує гідну якість зображення та відео. Це універсальний вибір для проектів загального призначення, які потребують візуального введення. З іншого боку, Raspberry Pi High-Quality Camera забезпечує вищу роздільну здатність і можливість використовувати змінні об’єктиви, що робить її більш придатною для проектів, які вимагають професійної якості зображення та налаштування.

Обидва модулі камери підтримуються екосистемою Raspberry Pi і можуть бути легко інтегровані в проекти за допомогою офіційної бібліотеки програмного забезпечення камери Raspberry Pi.

Для обробки зображень з відеокамери Raspberry Pi Camera, вирішено використовувати бібліотеку OpenCV.

OpenCV (Open Source Computer Vision Library) — бібліотека комп’ютерного зору та обробки зображень із відкритим кодом [19]. Вона надає широкий спектр функцій і алгоритмів, які дозволяють розробникам працювати із зображеннями та відео, виконувати різноманітні завдання, такі як виявлення

об’єктів, розпізнавання зображень і маніпулювання зображеннями. Приклад обробки зображень показано на рисунку 2.9.

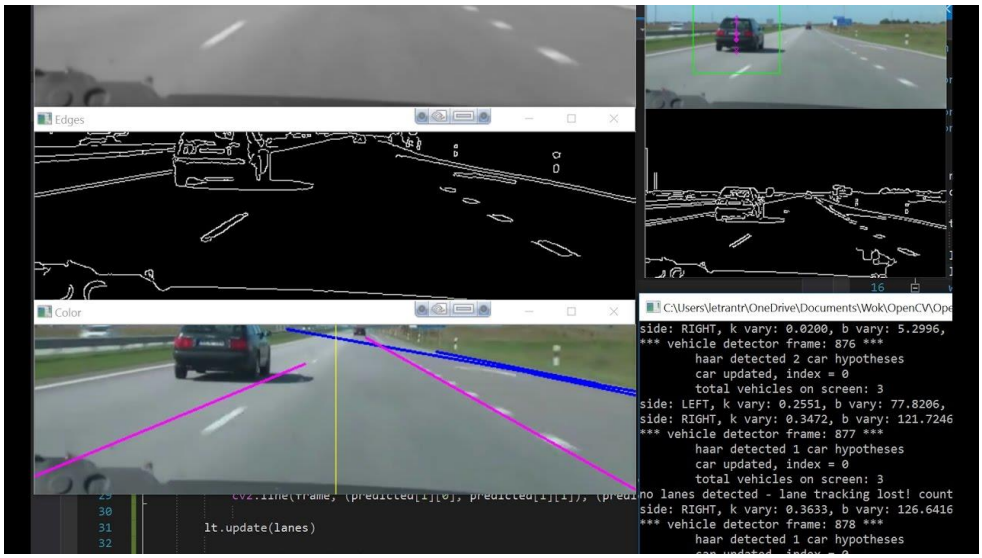

Рисунок 2.9 — Приклад обробки зображень бібліотекою OpenCV

Коли мова заходить про керування відеокамерою Raspberry Pi за допомогою OpenCV, бібліотека забезпечує зручний спосіб 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 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 комбінувати 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.

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

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 10000 mAh, 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.

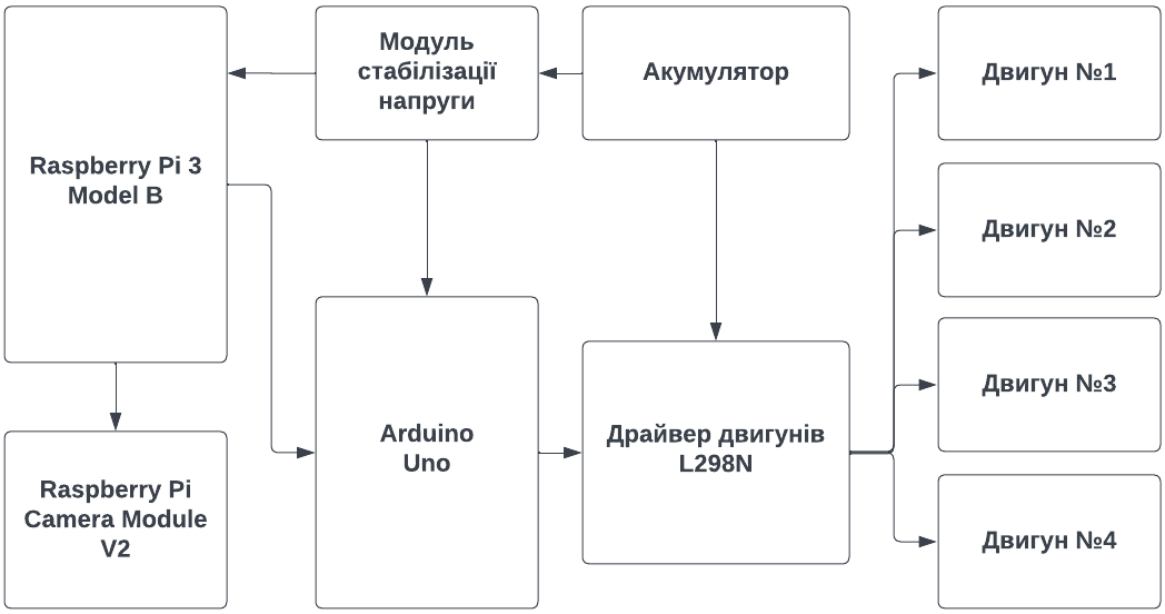

Рисунок 3.1 — Структурна схема системи керування

3.1 Вибір компонентів 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.

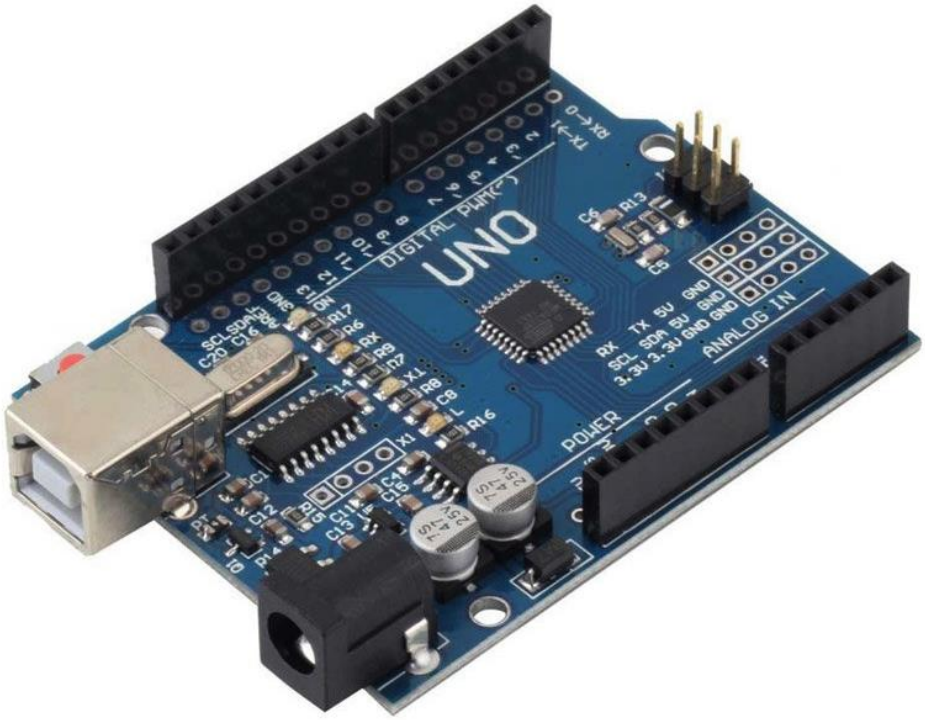

Рисунок 3.2 — Компоновка Arduino Uno

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 від 3.3 В до 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.

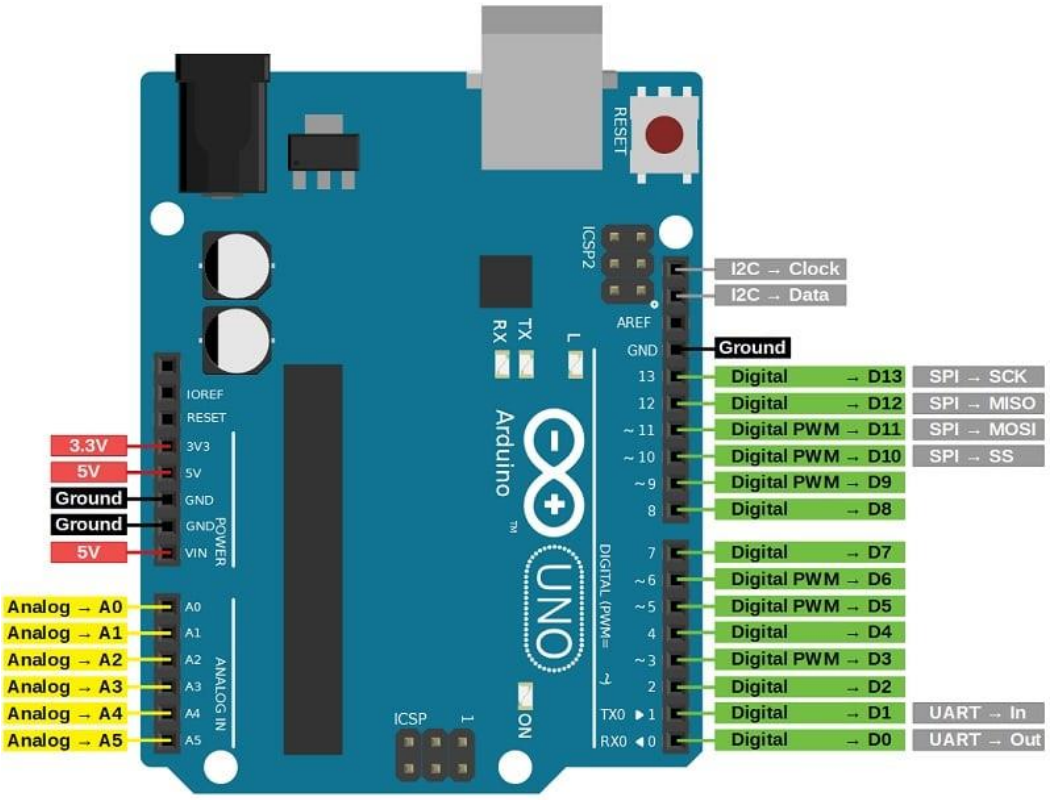

Рисунок 3.3 — Призначення виводів Arduino Uno

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.

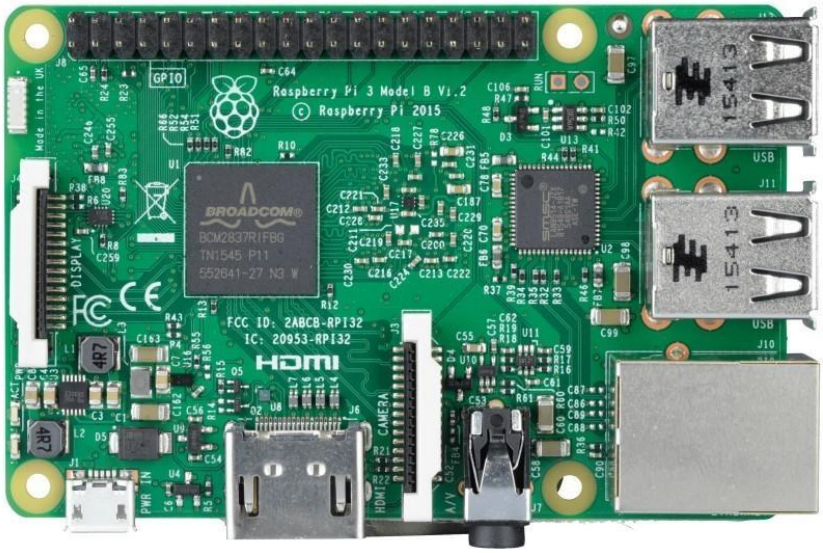

Рисунок 3.4 — Компоновка Raspberry Pi 3 Model B

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.

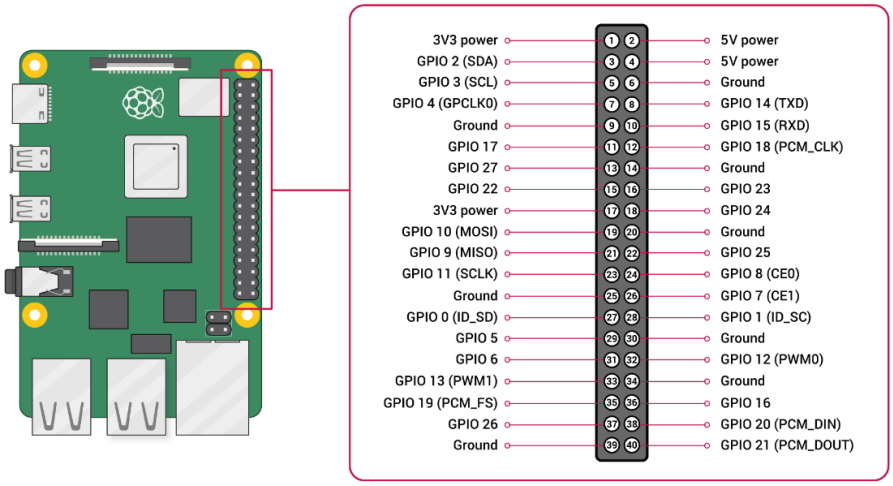

Рисунок 3.5 — Найменування виводів Raspberry Pi 3 Model B

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.

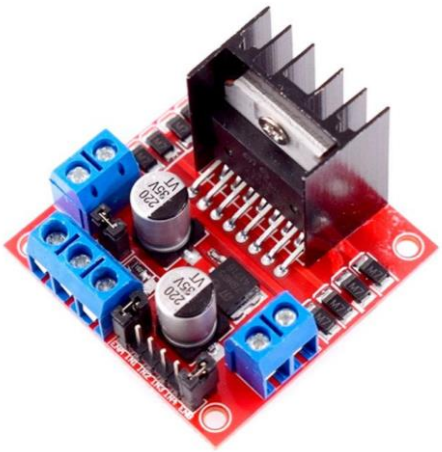

Рисунок 3.6 — Компоновка L298N

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.

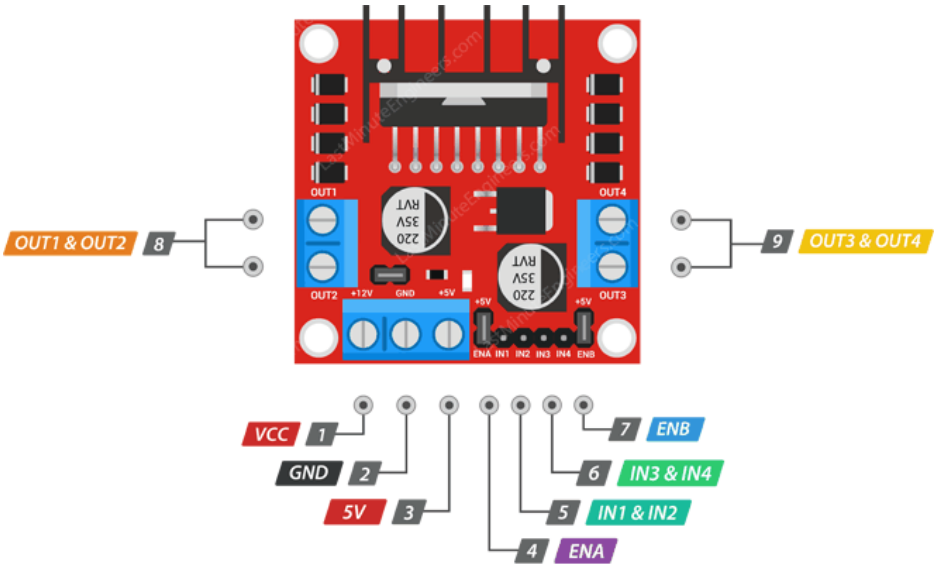

Рисунок 3.7 — Найменування виводів L298N

В якості джерела живлення було вибрано Xiaomi Mi Power Bank 3 з напругою 9В та ємністю в 10000 mAh (див. рис. 3.8).

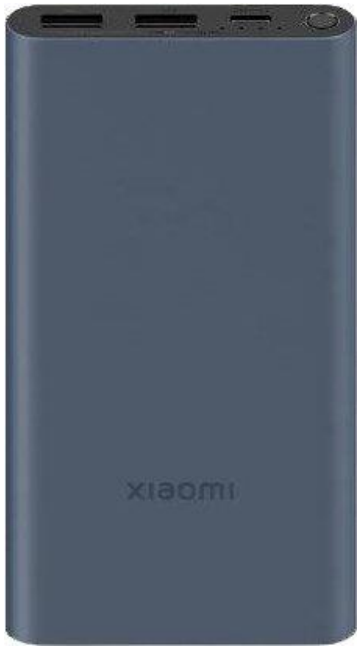

Рисунок 3.8 — Джерело живлення Xiaomi Mi Power Bank 3

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

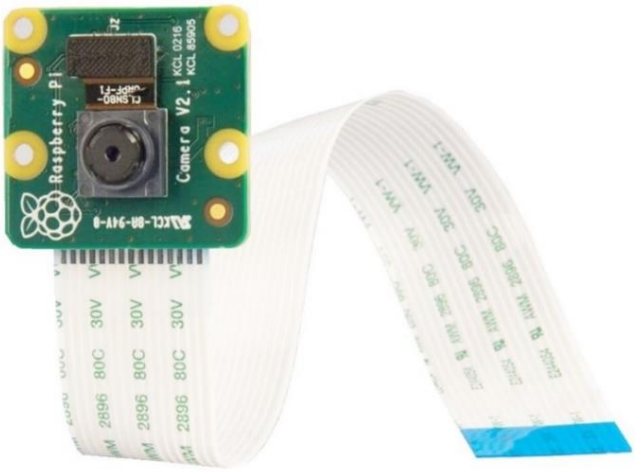

Рисунок 3.9 —Компоновка Raspberry Pi Camera Module V2

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.

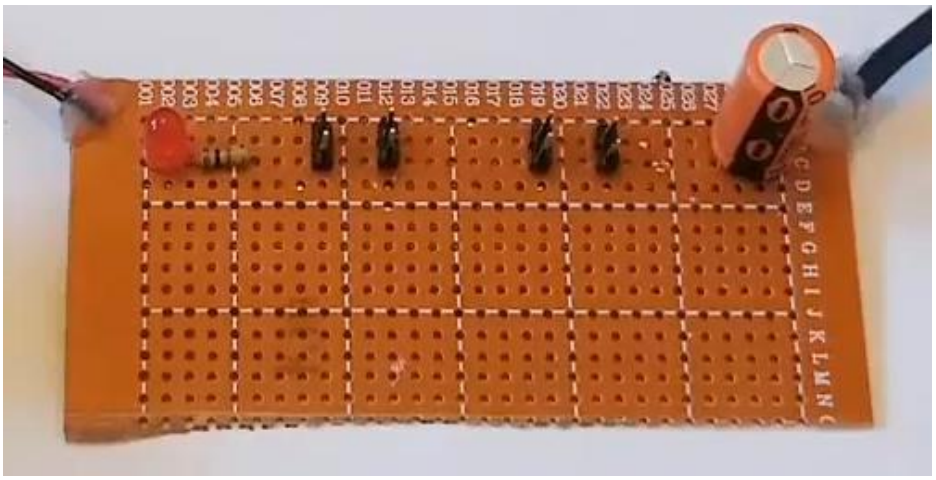

Рисунок 3.10 — Компоновка модуля стабілізації напруги

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 Розробка алгоритмів роботи

В минулому розділі 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;

— 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, які після обробки в головному пристрої надають результат для прийняття рішення підпорядкованому пристрою, як повинна змінювати положення системи відносно поточного розташування у просторі. Для отримання та обробки зображень використовуємо функції бібліотеки OpenCV.

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;

— виконання руху моторами.

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 3.12.

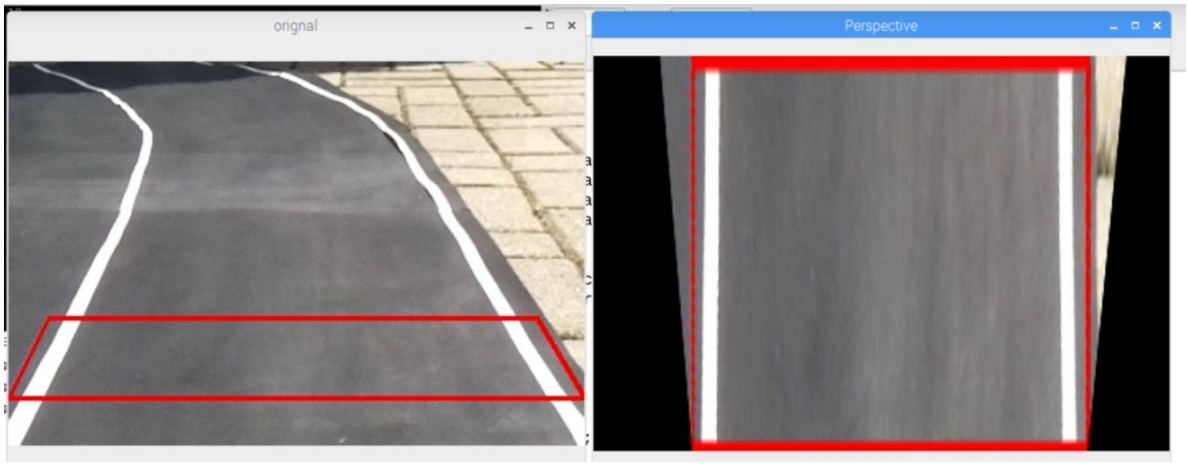

Рисунок 3.12 — Зображення перехопленого кадру ліворуч та кадру з трансформованою перспективою праворуч

Для кращого розпізнавання розмітки необхідно здійснити фільтрацію зображення, піднявши інтенсивність розмітки порівняно з дорогою та виділивши контури розмітки, як на рисунку 3.13.

Рисунок 3.13 — 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:

$$\text{LaneCenter} = \frac{(\text{RightLanePos} - \text{LeftLanePos})}{2} + \text{LeftLanePos} \qquad (3.1)$$

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 3.14.

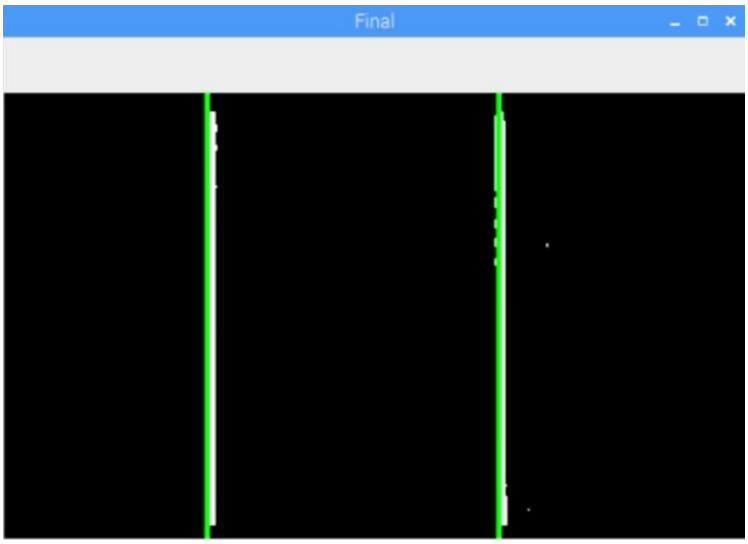

Рисунок 3.14 — Зображення кадру після знаходження координат ліній розмітки

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

$$Result = LaneCenter - FrameCenter \qquad (3.2)$$

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 3.15.

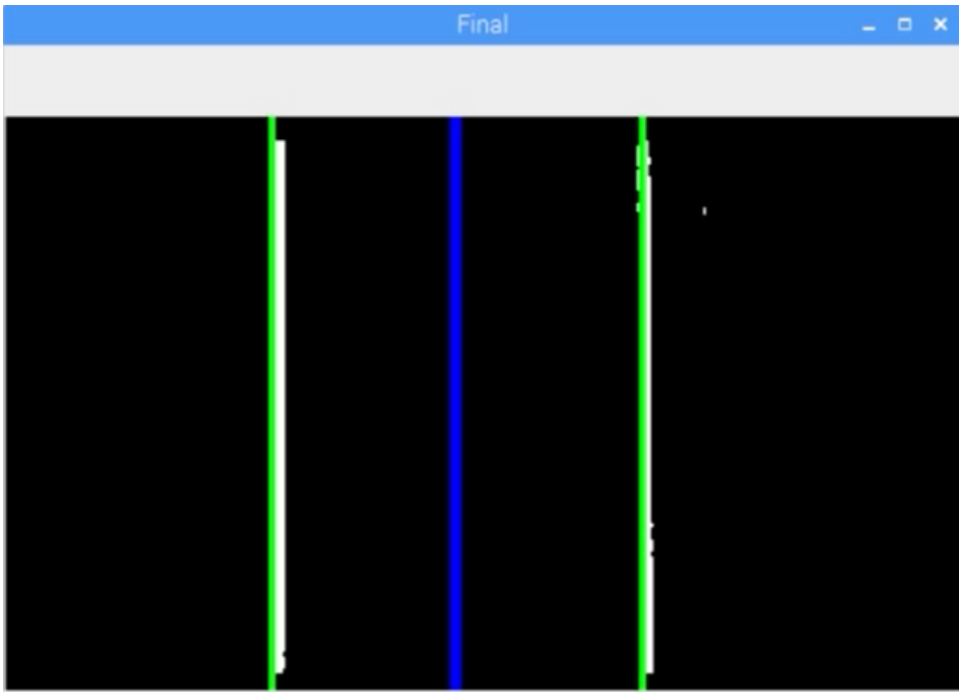

Рисунок 3.15 — Зображення кадру після знаходження центру розмітки

Після отримання результату обробки, необхідно обрати варіант руху системи. Можливі випадки розміщено на блок-схемі, яку продемонстровано на рисунку 3.16.

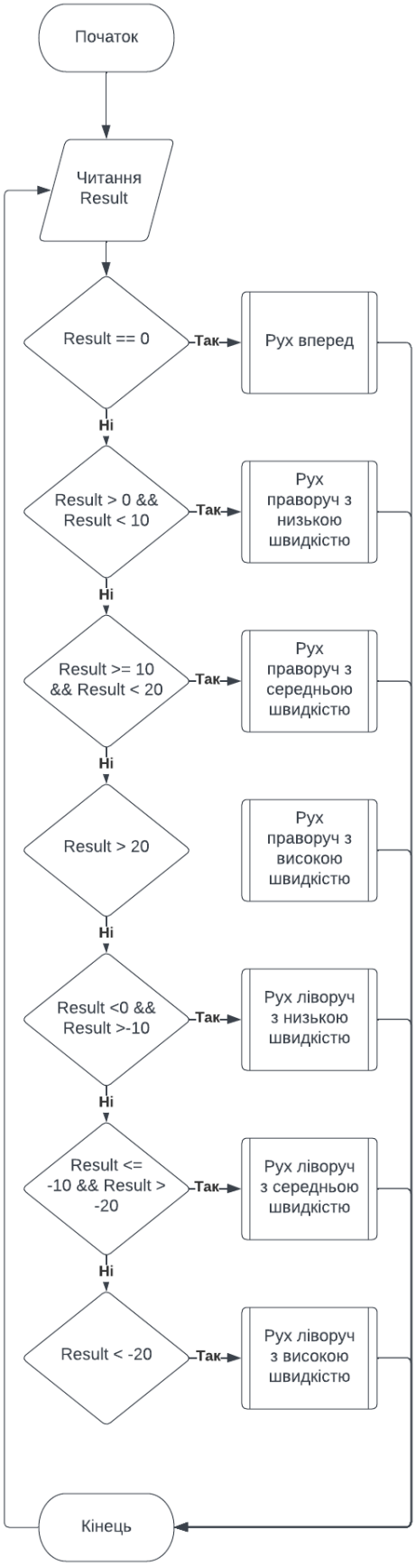

Рисунок 3.16 — Блок-схема алгоритму руху системи

На рисунку 3.17 зображено схему алгоритму роботи системи.

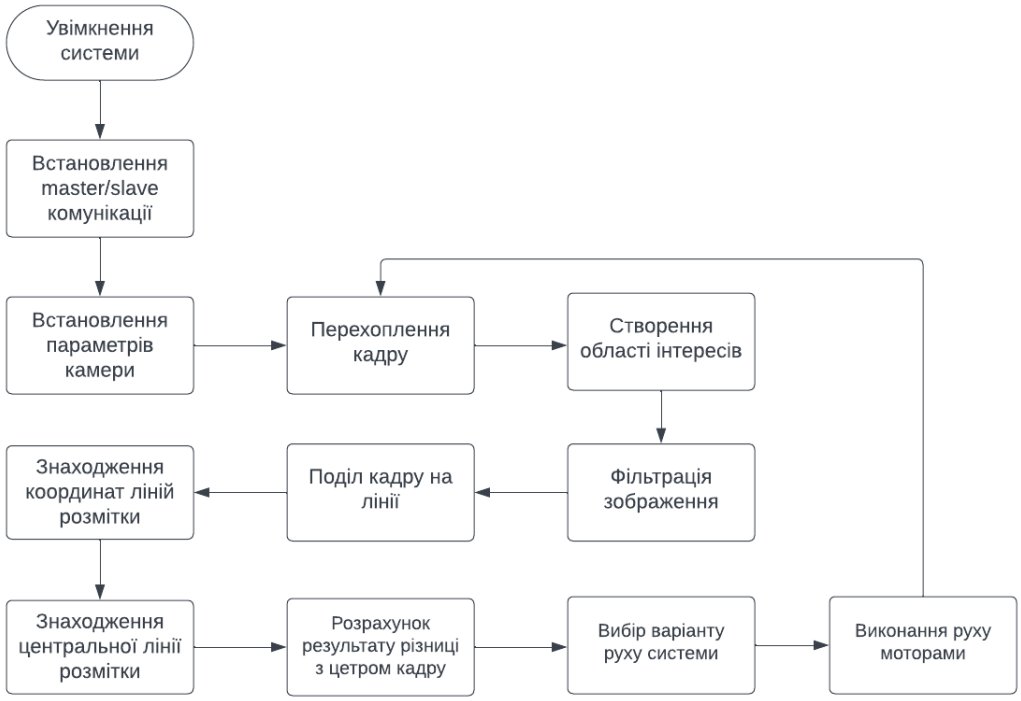

Рисунок 3.17 — Блок-схема основного алгоритму роботи системи

Розробка алгоритму завершена і можна переходити до її програмної реалізації.

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;
    }
```

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 3.3.

Лістинг 3.3. 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 3.4.

Лістинг 3.4. 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 3.5.

Лістинг 3.5. 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.

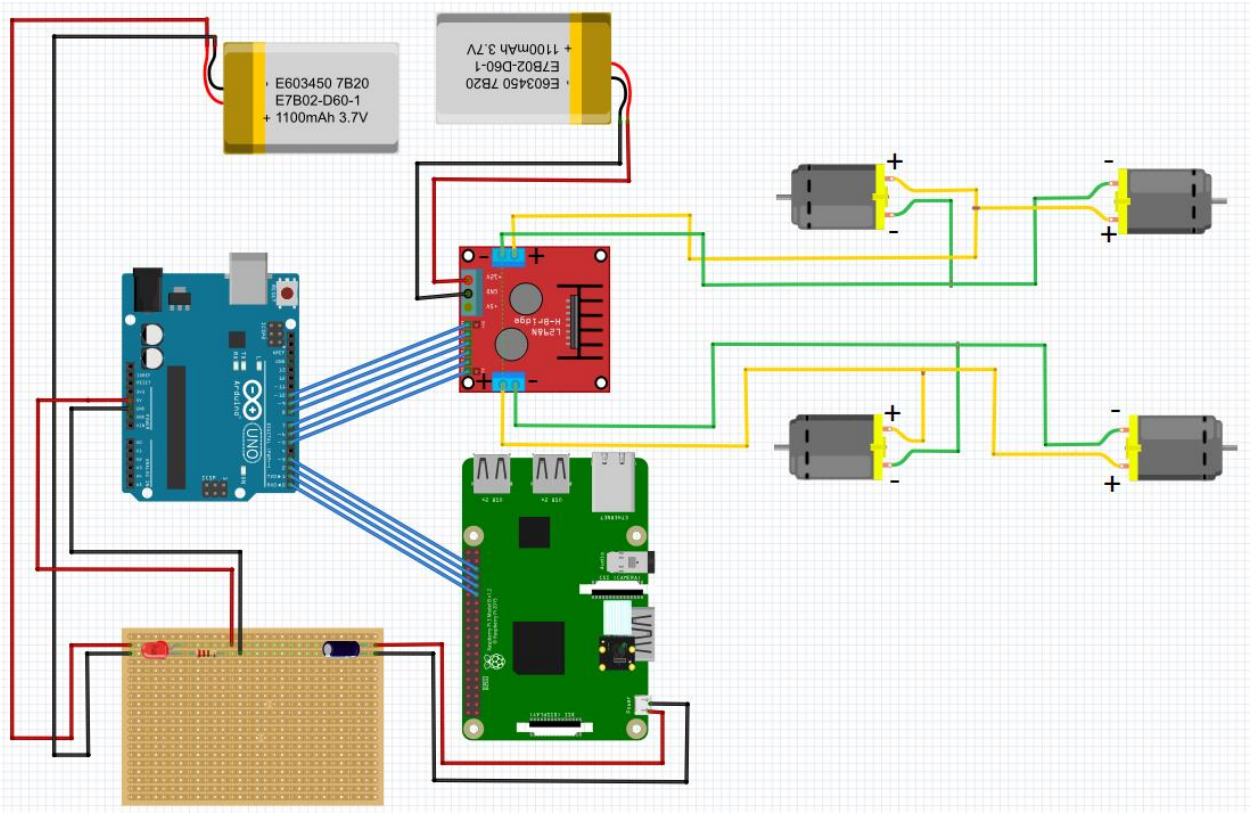

Рисунок 3.18 — Схема з’єднань компонентів системи

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.

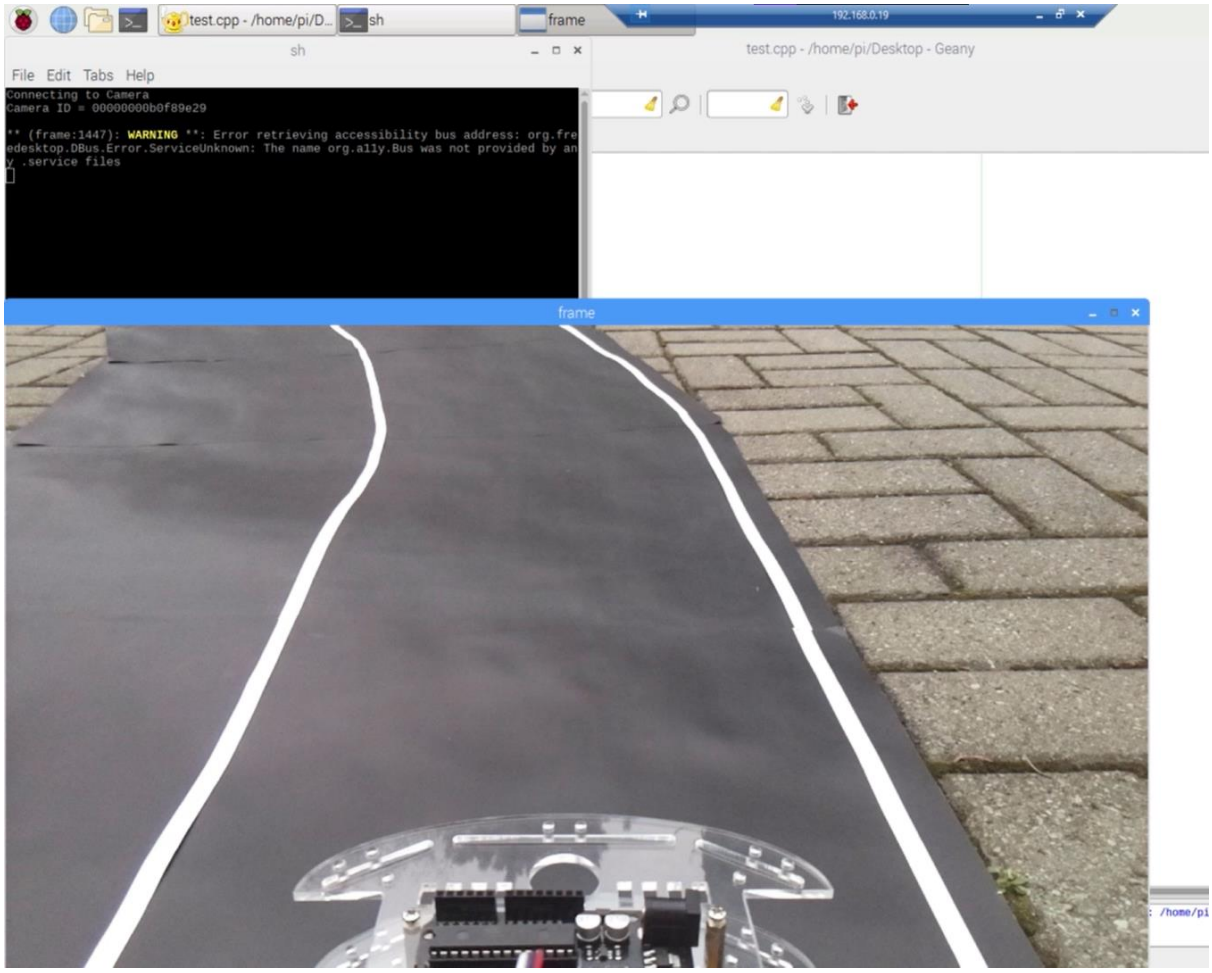

Рисунок 4.1 — Результат успішного підключення камери до системи

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

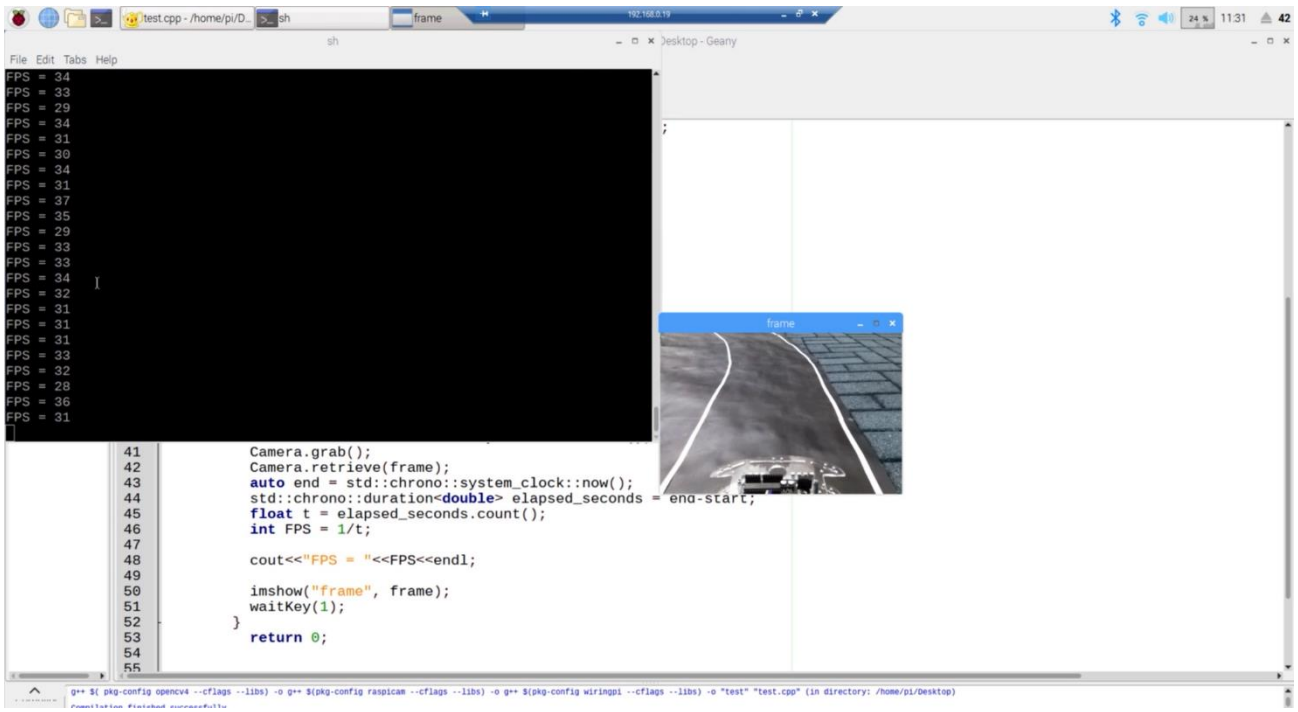

Рисунок 4.2 — Результат обрахунку кількості кадрів в секунду

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.

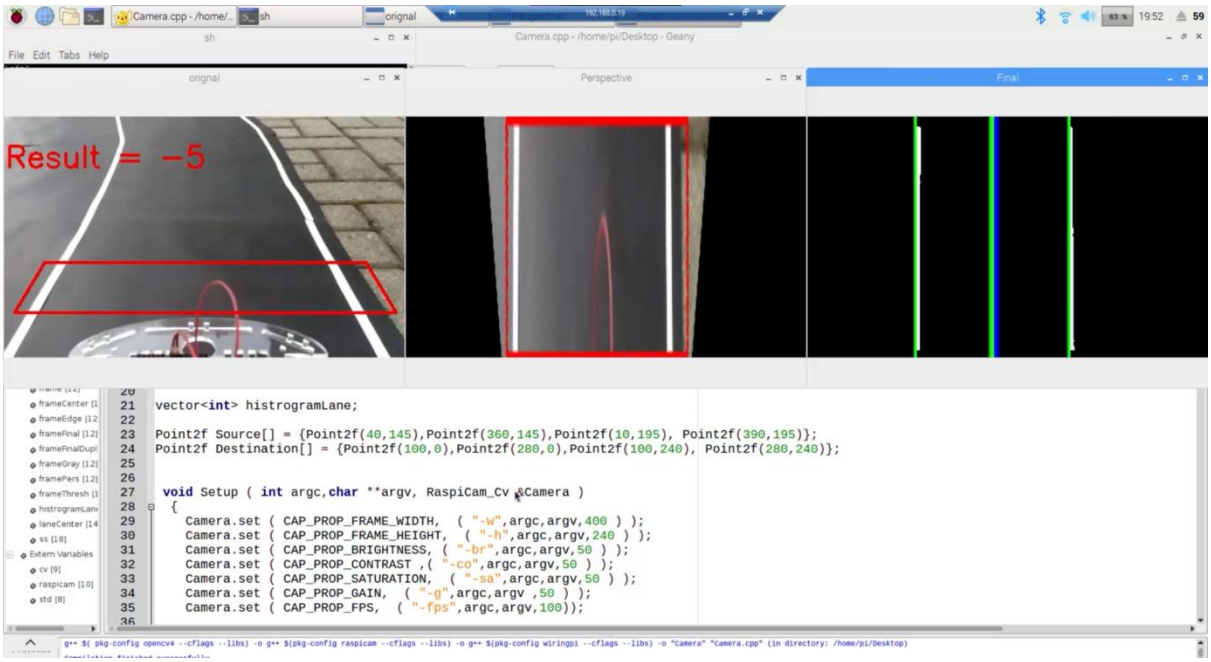

Рисунок 4.3 — Початок руху системи

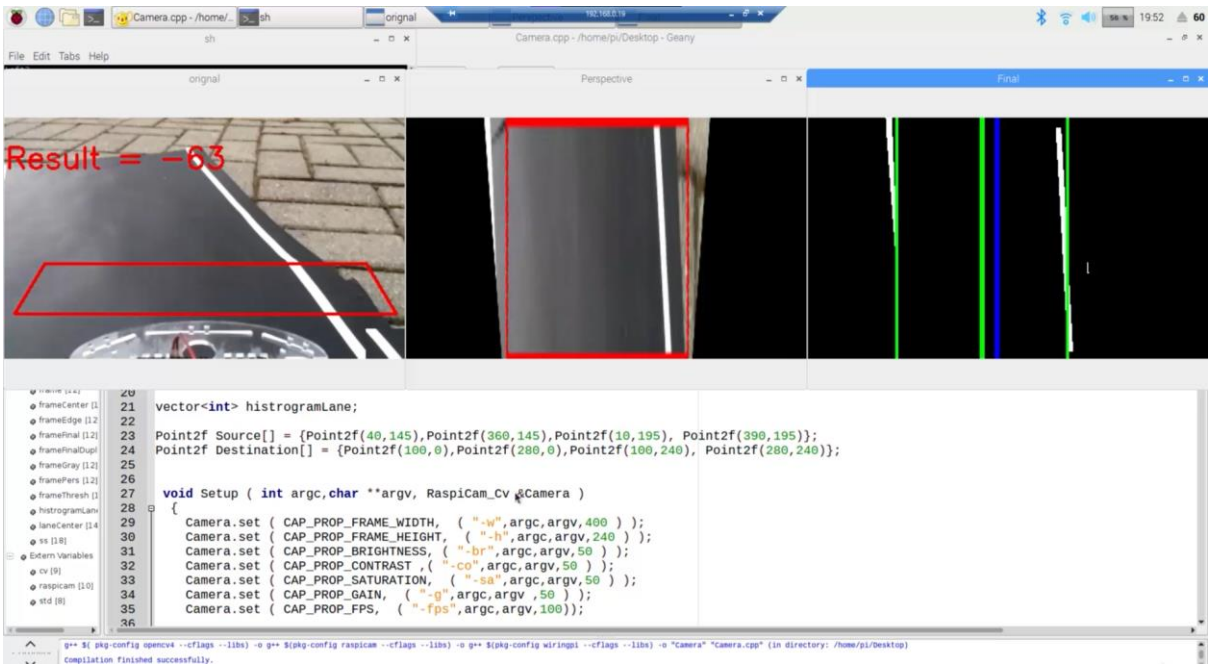

Рисунок 4.4 — Продовження руху системи

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

ВИСНОВКИ

В результаті виконаної роботи було розроблено систему керування рухомим об’єктом в двовимірному просторі.

В першому розділі розглянута історія розвитку безпілотних рухомих об’єктів. Від стародавніх автоматів до сучасних дронів і безпілотних автомобілів ці об’єкти еволюціонували разом із технологічним прогресом.

У другому розділі описано теорію автоматичного керування. Розглянута модель зв’язку між компонентами системи. Проаналізовані можливі варіанти вибору головних та підпорядкованих пристроїв керування, модулів відеокамер для перехоплення зображень.

Третій розділ присвячено розробці алгоритму системи керування рухомим об’єктом. Було створено програми для головного та підпорядкованого пристрою з можливостями обробки зображень та руху системи. За допомогою цих можливостей, система здатна розпізнавати розмітку на дорозі та приймати рішення стосовно руху.

В четвертому розділі було проведено тестування працездатності розробленої системи. Система керування рухомим об’єктом здійснила вдалі спроби розпізнавання розмітки та здійснила коректний рух по траєкторії дороги.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Фурман, М.; Крупельницький, Л.. Методи розпізнавання рухомих об'єктів. НТКП ВНТУ. Факультет інформаційних технологій та комп'ютерної інженерії, Ukraine, mar. 2023. Available at: <https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki 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. A Brief History of Drones - Imperial War Museums [Електронний ресурс] // Imperial War Museums – Режим доступу: https://www.iwm.org.uk/history/a-brief-history-of-drones

4. REMUS 300 Unmanned Underwater Vehicle (UUV) [Електронний ресурс] // Naval Technology – Режим доступу: https://www.naval-technology.com/projects/remus-300-unmanned-underwater-vehicle-uuv/

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-

ДОДАТОК А

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

«__» _____ 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської дипломної роботи

Мікроконтролерна система керування рухомим об’єктом в двовимірному просторі

08-23.БДР.018.00.000 ТЗ

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

_____ Крупельницький Л.В

Виконав: студент гр. 1СП-19б

_____Фурман М.А.

2023

1 Підстава для виконання бакалаврської дипломної роботи (БДР):

— актуальність роботи полягає в необхідності створення рухомих об'єктів для впровадження розумних технологій та розвитку автономних систем;

— наказ про затвердження теми БДР.

2 Мета і призначення БДР:

— Метою роботи є створення мікроконтролерної системи керування рухомим об’єктом, що здатна виявляти перешкоди та планувати маршрут на основі зібраних даних;

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

3 Вихідні дані для виконання БДР:

— 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 БДР:

— аналіз історії розвитку безпілотних об’єктів;

— пошук та аналіз методів управління самокерованими об’єктами;

— огляд сучасних систем автоматичного 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 БДР та очікувані результати:

Етапи роботи та очікувані результати приведено в Таблиці А.1

Таблиця А.1 — Етапи БКР

№ з/п	Назва етапів дипломної роботи	Термін виконання		Очікувані результати
		початок	закінчення	
1	Постановка задачі роботи	8.03.2023	11.03.2023	Задачі дослідження
2	Аналіз історії розвитку безпілотних об’єктів	12.03.2023	17.03.2023	Аналітичний огляд джерел
3	Пошук та аналіз методів управління самокерованими об’єктами	18.03.2023	20.03.2023	Розділ 1
4	Огляд сучасних систем автоматичного керування рухомим об’єктом	22.03.2023	24.04.2023	Розділ 1
5	Вибір компонентів для створення системи керування рухомим об’єктом	25.04.2023	07.05.2023	Розділ 2
6	Створення програмного забезпечення та налаштування зв’язку між компонентами апаратної складової	08.05.2023	22.05.2023	Розділ 3
7	Тестування працездатності системи	23.05.2023	24.05.2023	Розділ 4
8	Оформлення пояснювальної записки	25.05.2023	31.05.2023	ПЗ, додатки, презентація

6 Матеріали, що подаються до захисту БДР

Пояснювальна записка БДР, графічні та ілюстративні матеріали, протокол попереднього захисту роботи на кафедрі, відзив наукового керівника, рецензія рецензента, анотації до БДР українською та іноземною мовами, нормоконтроль про відповідність оформлення БДР діючим вимогам.

7 Порядок контролю виконання та захисту БДР

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

8 Вимоги до оформлювання та порядок виконання БКП 8.1 При оформлювання БКП використовуються:

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

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

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

— Методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 «Комп’ютерна інженерія» (освітня програма «Системне програмування»). Кафедра обчислювальної техніки ВНТУ 2022;

— документами на які посилаються у вище вказаних. 8.2 Порядок виконання БКП викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21».

ДОДАТОК Б

Структурна схема системи керування

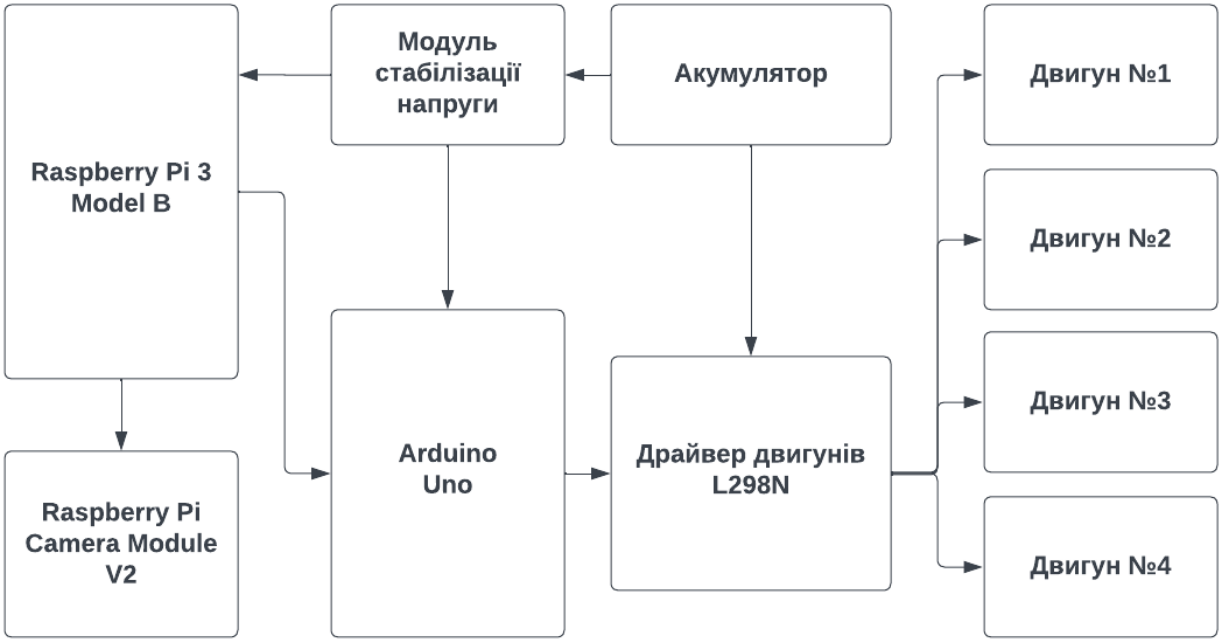

Рисунок Б.1 — Структурна схема системи керування

ДОДАТОК В

Блок-схема алгоритму руху системи

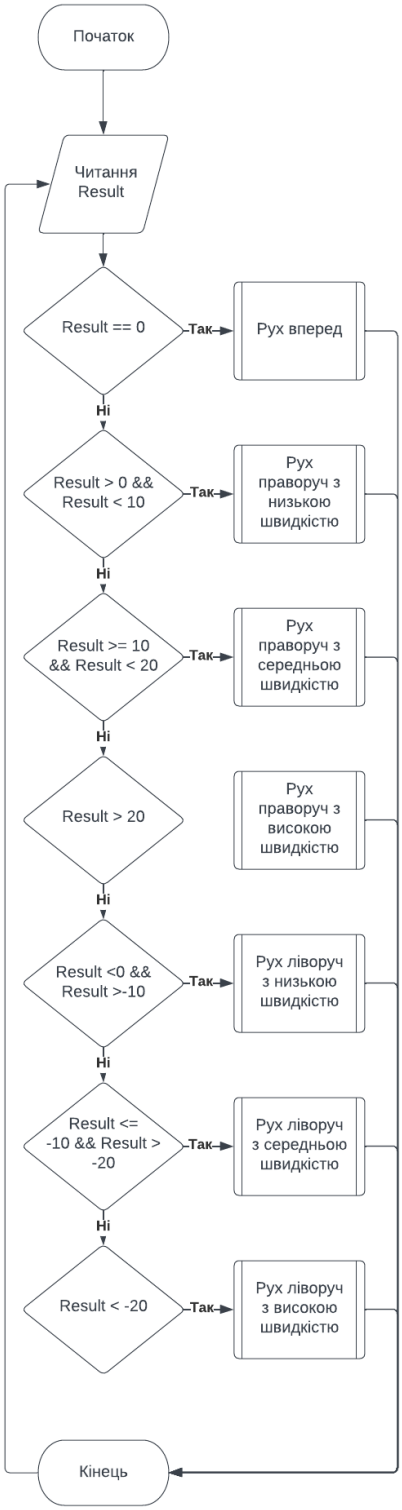

Рисунок В.1 — Блок-схема алгоритму руху системи

ДОДАТОК Г

Блок-схема основного алгоритму роботи системи

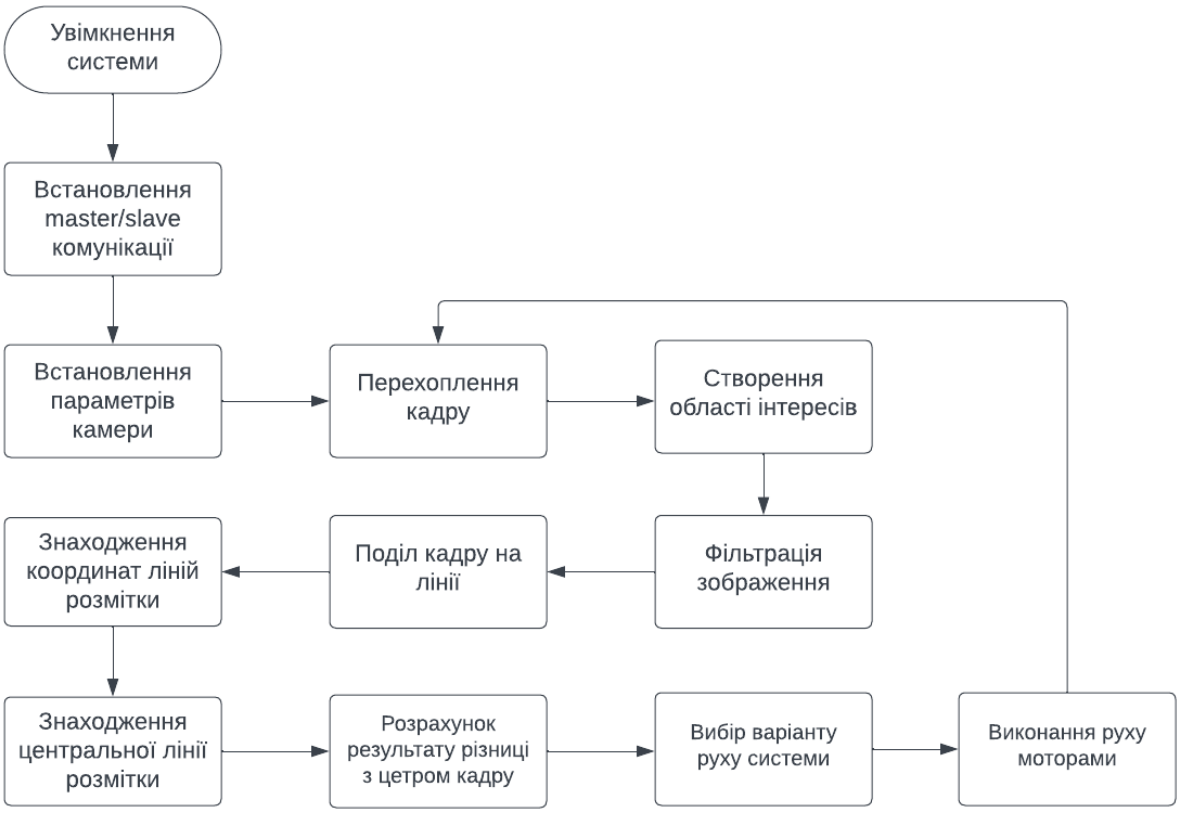

Рисунок Г.1 — Блок-схема основного алгоритму роботи системи

ДОДАТОК Д

Схема з’єднань компонентів системи

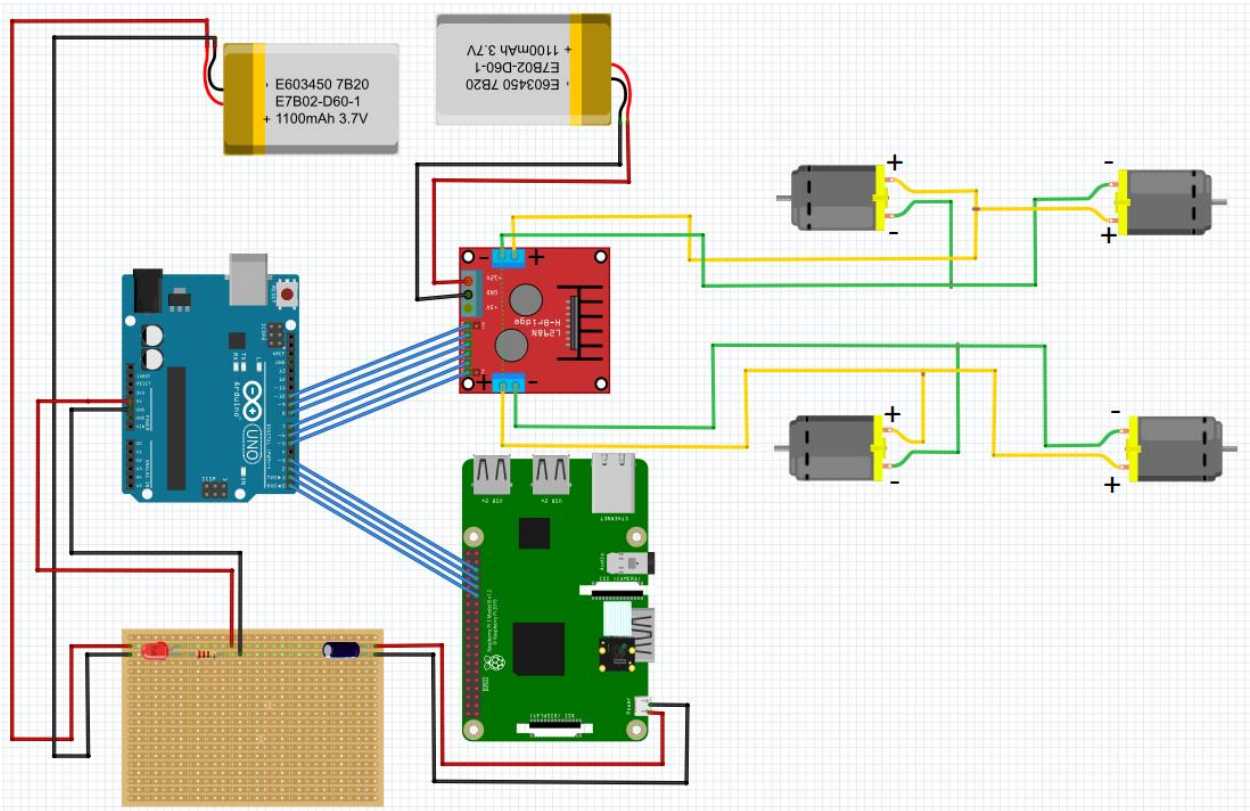

Рисунок Д.1 — Схема з’єднань компонентів системи

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;
Mat ROILane;
int LeftLanePos, RightLanePos, frameCenter, laneCenter, Result;

RaspiCam_Cv Camera;

stringstream ss;

vector<int> histrogramLane;

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) };


void Setup(int argc, char** argv, RaspiCam_Cv& Camera) // Встановлення
основних параметрів камери
{
   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
   cvtColor(frame, frame, COLOR_BGR2RGB); // Зміна кольорової схеми
}

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); // Об'єднання двох трансформацій
   cvtColor(frameFinal, frameFinal, COLOR_GRAY2RGB); // Зміна кольорової
схеми
   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 ліній
{
    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() // Знаходження центру між розміткою на дорозі
{
    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;
}

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();

   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;
}

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 << "bMove 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);

        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; // Обчислення кількості кадрів за секунду
        cout<<"FPS = "<<FPS<<endl;
    }
    return 0;

}
```

ДОДАТОК Ж

Лістинг модуля підпорядкованого пристрою

```
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 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 loop()
{
  Data();
  if(data==0)
    {
      Forward();
    }

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

```
  }

 else if(data==2)
  {
    Right2();
  }

 else if(data==3)
  {
    Right3();
  }

 else if(data==4)
  {
    Left1();
  }

 else if(data==5)
  {
    Left2();
  }

 else if(data==6)
  {
    Left3();
  }
}
```

ДОДАТОК И

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

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

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності
Unicheck

Оригінальність 89,5% Схожість 10,5%

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 Unicheck щодо роботи.

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

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