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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему «**СИСТЕМА РЕЄСТРАЦІЇ ТА РОЗПІЗНАВАННЯ НАПРЯМУ РУХУ ЗА ДОПОМОГОЮ КОМП'ЮТЕРНОГО ЗОРУ**»

Виконав: студент 2 курсу, групи 1КІ-21м
спеціальності 123 Комп'ютерна інженерія
_____Ткачук В.М
Керівник к.т.н., доц., доц. каф. ОТ
_____ Колесник І.С.
Опонент к.т.н., доц., зав.каф. МБІС
Карпінець В.В.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
«11» 12 2023 р.

Вінниця 2023

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень магістр
Спеціальність 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри
обчислювальної техніки
проф., д.т.н. О. Д. Азаров
« 26 » вересня 2023 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Ткачуку Валерію Миколайовичу

1 Тема роботи **«Система реєстрації та розпізнавання напряму руху за допомогою комп'ютерного зору»**, керівник роботи Колесник Ірина Сергіївна, к.т.н., доцент,затверджені наказом вищого навчального закладу від 18.09.223 року № 247.

2 Строк подання студентом роботи 9.12.2023 р.

3 Вихідні дані до роботи: методи засновані на класичних алгоритмах машинного навчання, мова програмування C++, бібліотека комп'ютерного зору з відкритим вихідним кодом OpenCV та кросплатформове середовище розробки Code::Blocks.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, огляд та аналіз систем комп'ютерного зору, проектування апаратного забезпечення системи, проектування програмного забезпечення системи, висновки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):

схема електричних з'єднань, блок-схема загального алгоритму роботи програми, програмний код основного алгоритму роботи, програмний код модифікації №1 програми, програмний код модифікації №2 програми

6 Консультанти розділів роботи представлено в табл. 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3,4	Колесник І.С., к.т.н., доц. каф. ОТ		
5	Небава М.І., к.е.н., проф. каф. ЕП і ВМ		

7 Дата видачі завдання __.__._____ р.

8 Календарний план наведено в табл. 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів виконання магістерської роботи	Строк виконання етапів роботи	Прим.
1	Постановка мети та задач роботи	21.10.23	вик.
2	Основні функціональні аспекти комп'ютерного зору	25.10-30.10.23	вик.
3	Проектування апаратного забезпечення системи	31.10-08.11.23	вик
4	Складання структурної схеми пристрою, що розробляється	09.11-15.11.23	вик
5	Складання електричної схеми пристрою, що розробляється	16.11-.20.1123	вик
6	Використання інструментів розробки	21.11-25.11.23	вик
7	Програмна реалізація рішень	26.11-31.11.23	вик
8	Можливі варіанти модернізації програми	01.12-04.12.23	вик
9	Розрахунок економічної частини роботи	01.12-04.12.23	вик
10	Оформлення пояснювальної записки та ілюстративного матеріалу	05.12.23	вик
11	Аналіз виконання роботи, висновки, додатки		
12	Перевірка якості виконання магістерської роботи та усунення недоліків		

Студент Ткачук В. М.

Керівник роботи Колесник І.С.

АНОТАЦІЯ

УДК 004.9

Ткачук В. М.

Система реєстрації та розпізнавання напряму руху за допомогою комп’ютерного зору. Магістерська кваліфікаційна робота зі спеціальності 123 — комп’ютерна інженерія, освітня програма — комп’ютерна інженерія. Вінниця: ВНТУ, 2023, 113 с.

На укр.мові. Бібліогр.: 27 назв, рис. 45, табл. 7.

Дана магістерська кваліфікаційна робота присвячена створенню системи реєстрації та розпізнавання напряму руху за допомогою комп’ютерного зору.

Одним із найбільш актуальних напрямів розвитку штучного інтелекту в останні роки став комп'ютерний зір.

Завдяки накопиченому досвіду та знанням легко розпізнаємо об'єкти та відрізняємо їх один від одного, орієнтуємось у просторі.

Області застосування комп'ютерного зору досить великі: від промислових засобів моніторингу технічних процесів до автономних систем керування, що приймають рішення на основі аналізу отриманої відеоінформації.

Одним із найперспективніших напрямів є використання систем комп'ютерного зору в автоматизованих логістичних системах. Яскравим прикладом цього є широке використання великими сучасними компаніями транспортних роботів, які забезпечують переміщення вантажів у промисловому середовищі без участі оператора.

Ключові слова: комп’ютерний зір, комп'ютер, структурна схема, електрична схема, пристрій, програмне забезпечення, апаратне забезпечення.

ANNOTATION

Tkachyk V.M.

System of registration and recognition of the direction of movement using computer vision. Master's qualification route in the specialty 123 — computer engineering, educational program computer engineering. Vinnitsa, VTNU, 2023, 113 p.

In the Ukr. leng. Libr. name 27, figure 45, table 7.

This master's thesis is devoted to the creation of a system of registration and recognition of the direction of movement using computer vision.

Computer vision has become one of the most relevant areas of artificial intelligence development in recent years.

Thanks to the accumulated experience and knowledge, we easily recognize objects and distinguish them from each other, orient ourselves in space.

The fields of application of computer vision are quite large: from industrial means of monitoring technical processes to autonomous control systems that make decisions based on the analysis of the received video information.

One of the most promising directions is the use of computer vision systems in automated logistics systems. A clear example of this is the widespread use by large modern companies of transport robots, which ensure the movement of goods in an industrial environment without the participation of an operator.

Keywords: computer vision, computer, structural diagram, electrical diagram, device, software, hardware.

ЗМІСТ

ВСТУП

Розвиток комп'ютерних технологій йде величезними темпами. Особливо це стосується сфер штучного інтелекту та робототехніки. Багато речей, які лише кілька десятиліть тому можна було зустріти лише у науково-фантастичній літературі, сьогодні стали невід'ємною частиною нашого життя. Для сучасної людини немає нічого дивного в тому, що на передових автоматизованих заводах замість робітників працюють роботи, дорогами міст їздять безпілотні автомобілі, а штучні нейронні мережі створюють шедеври, які не поступаються роботам професійних художників [1-4].

Одним із найбільш актуальних напрямів розвитку штучного інтелекту в останні роки став комп'ютерний зір. Здатність бачити — найважливіша властивість людини. Саме за допомогою зору ми отримуємо найбільшу кількість інформації про навколишній світ. Поглянувши на якийсь предмет, ми, майже не замислюючись, можемо сказати, що це таке, якими є його приблизні розміри, форма, колір. Завдяки накопиченому досвіду та знанням ми легко розпізнаємо об'єкти та відрізняємо їх один від одного, орієнтуємось у просторі. Згодом виникла ідея наділити зором і штучний інтелект. Однак незважаючи на те, що сучасні комп'ютери вже давно перевершили людину у сфері обчислень та обробки інформації, аналіз зображень представляв для них досить важке завдання. Розвиток машинного навчання та штучних нейронних мереж значно прискорив процес набуття роботами здатності бачити. На сьогоднішній день штучний інтелект здатний розпізнавати номерні знаки автомобілів, читати штрих-коди на товарах у супермаркеті, аналізувати записи з камер відеоспостереження та здійснювати пошук осіб на фотографіях та відеозаписах. Таким чином, комп'ютерний зір є набором методів, що дозволяють комп'ютерам витягувати інформацію з зображень [5-6].

Області застосування комп'ютерного зору досить великі: від промислових засобів моніторингу технічних процесів до автономних систем керування, що приймають рішення на основі аналізу отриманої відеоінформації. Одним із

найперспективніших напрямів є використання систем комп'ютерного зору в автоматизованих логістичних системах. Яскравим прикладом цього є широке використання великими сучасними компаніями транспортних роботів, які забезпечують переміщення вантажів у промисловому середовищі без участі оператора. Дані машини використовуються на підприємствах для транспортування сировини — зі складу до цеху, заготовок — між виробничими етапами, готової продукції — з виробництва на склад та зі складу на відвантаження. Застосування таких транспортних засобів дозволяє зменшити витрати на перевезення, пов'язані з людським фактором та втратою часу, підвищити безпеку на підприємстві, прискорити виробничі процеси. Складання маршруту для даних роботів здійснюється за допомогою напрямних ліній та маркерів. Їх виявлення та використання як суб'єктів управління і є головним завданням систем комп'ютерного зору [2,7-16].

Виходячи із розглянутого, використання систем комп'ютерного зору при розробці автоматично керованих засобів та автономних мобільних роботів у сфері складської логістики є **актуальною** задачею.

Метою дослідження є вдосконалення системи реєстрації та розпізнавання напряму руху за допомогою комп’ютерного зору.

Задачі дослідження:

— виконати огляд та аналіз використання систем технічного зору у розробці автоматично керованих транспортних засобів та автономних мобільних роботів у сфері складської логістики;

— розробити структурну схему, схему електричних з'єднань і підібрано електронну компонентну базу системи, що розробляється;

— розробити програму для розпізнавання напрямної лінії за допомогою комп'ютерного зору;

— запропонувати можливі модифікації основної програми, а також розглянуто їх основні переваги та недоліки;

— провести економічний розрахунок системи, що розробляється.

Наукова новизна отриманих результатів магістерської роботи полягає в удосконаленні методу розпізнавання напрямної лінії та реалізації апаратних і програмних засобів системи для розпізнавання напрямної лінії з допомогою комп'ютерного зору.

Об’єкт дослідження — процеси, що протікають в системах реєстрації та розпізнавання напряму руху за допомогою комп’ютерного зору.

Предмет дослідження — методи та засоби побудови системах реєстрації та розпізнавання напряму руху за допомогою комп’ютерного зору.

Апробацію результатів наукової роботи було проведено на науковій конференції:

«Молодь в науці: дослідження, проблеми, перспективи (МН–2024)», доповідь на тему “Система реєстрації та розпізнавання напряму руху за допомогою комп’ютерного зору ”

1 ОГЛЯД ТА АНАЛІЗ СИСТЕМ КОМП’ЮТЕРНОГО ЗОРУ

1.1 Основні функціональні аспекти комп’ютерного зору

Технічний (машинний) зір — це сфера застосування комп'ютерного зору у промисловості чи виробництві. У той час, як комп'ютерний зір є загальним набором методів, що дозволяють комп'ютерам бачити [1, 17].

Хоча методи технічного зору були спочатку розроблені для видимої частини спектру, ті ж технології обробки можуть бути застосовані до зображень, отриманих в інших частинах спектру, таким як інфрачервоне світло або рентгенівське випромінювання.

Застосування технічного зору цих галузях допомагає замінити людей, які працюють на складальних лініях, які займаються оглядом продукції, роблячи висновки про якість виконання. Системи комп’ютерного зору використовують цифрові камери, на даний момент все частіше інтелектуальні камери, а також програмне забезпечення, що додається до них, яке займається обробкою отриманого зображення. У більшості випадків, у системах комп’ютерного зору використовуються послідовні поєднання різних методів обробки для виконання повного та точного аналізу. Наприклад, система, яка зчитує штрихкод, також займається перевіркою поверхні на наявність подряпин або інших пошкоджень і може виміряти довжину і ширину оброблюваних компонентів.

Системи комп’ютерного зору заточені на виконання завдань вузького та спеціалізованого напряму, таких як підрахунок різних об'єктів на стрічці конвеєрі, читання штрих-кодів, серійних номерів або пошук різних поверхневих дефектів. Вигода використання системи візуального контролю на основі технічного зору полягає у можливості збільшення виробничого обороту, за рахунок 24-годинного режиму роботи та підвищеної точності вимірів, що повторюються. Крім того, перевага використання, даного роду систем перед людьми полягає у відсутності різних людських факторів, таких як стомлюваність, неуважність. Але люди мають більш чуйне сприйняття і більшу адаптацію до розпізнавання нових дефектів.

Машини не бачать так само, як це роблять люди. Камери не рівнозначні системі людського зору, оскільки кожна людина може спиратися на власні здогади

і припущення, а системи технічного зору «бачать» шляхом вивчення отриманого зображення. Зображення розбивається окремі пікселі, попередньо відбувається їх обробка і система намагається зробити висновки за допомогою бази знань або допомогою розпізнавання образів. Хоча деякі алгоритми та методи технічного зору були створені, щоб відповідати зоровому сприйняттю людини, велика кількість методів були створені саме для обробки зображень, що одержуються, і визначення їх властивостей.

Системи технічного та комп'ютерного зору можуть обробляти зображень на рівних з людьми, але системи обробки зображень зазвичай проектуються, щоб виконати поодинокі, періодично повторювані завдання, і, незважаючи на значний прогрес у цій галузі, жодна система технічного чи комп'ютерного зору на даний момент ще не може змагатися на рівних з деякими можливостями людського зору, такими як терпимість до зміни освітлення та погіршення зображення, зміни частин тощо.

1.2 Приклади використання комп’ютерного зору різних галузях

На сьогоднішній день велику популярність набувають системи комп’ютерного зору як пристрої визначення положення координат корисного вантажу при роботі з маніпуляційними роботами.

1.2.1 Автоматизація сортування та відбраковування

Методика технічного зору дає можливість змінити людину у діях сортування та відбракування продукту (рисунок 1.1). Вилучивши «людський фактор», гарантується значне удосконалення якості продукту, збільшення продуктивності та скорочення браку [18].

Технологія полягає в отриманні фотографії предмета виробу (об'єкта, заготовки, деталі) завдяки CCD/CMOS камері в комбінації з оптичними компонентами та приладами освітлення. За допомогою спеціального програмного забезпечення виконується комп'ютерний аналіз і обробка зображення, надалі автоматично відбувається визначення приналежності продукту будь-якому класу,

придатності/непридатності товару чи створюється звіт про результати вивчення вироби у вигляді зручному для сприйняття людиною.

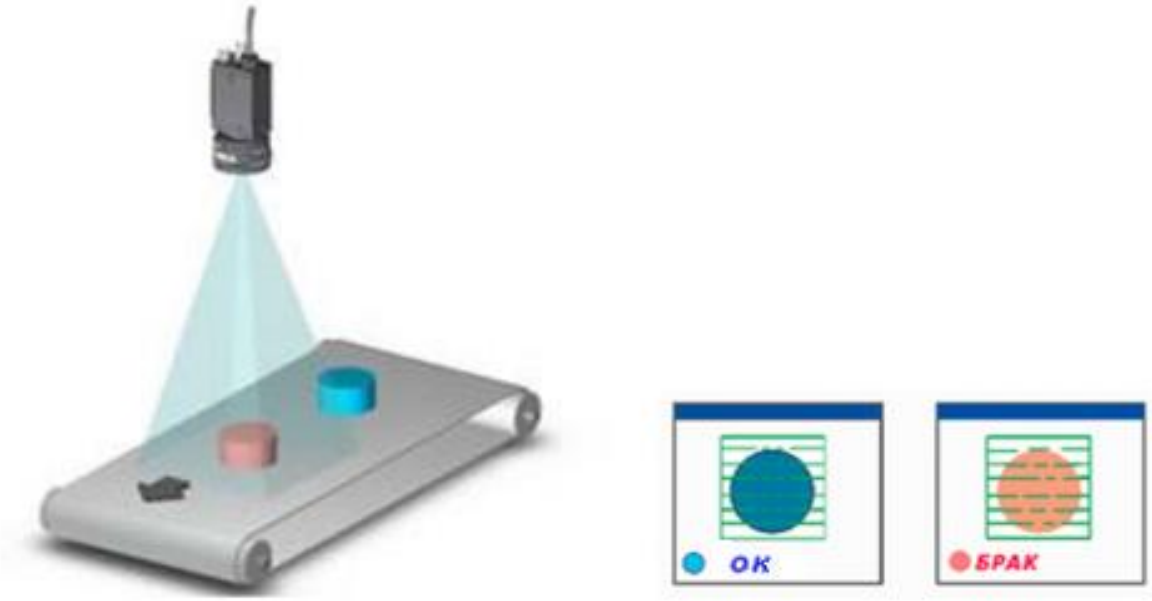

Рисунок 1.1 — Автоматичне сортування на конвеєрі за допомогою відеокамери

Під час відбракування за допомогою промислових відеокамер автоматично формуються такі ознаки: цілісність упаковки продуктів, комплектація упаковки продукту, наявність необхідних написів і маркування, точність наклейки етикеток.

У результаті сортування формуються такі ознаки: відношення товару до якогось класу відповідно встановленим якостям (форма, колір, геометричні розміри, присутність маркувань, написів та ін.)

Переваги:

— гарантується стовідсотковий контроль за правильністю збирання кожної одиниці продукції;

— відпадає «людський фактор» у діях сортування та відбраковування;

— технічний зір управляє виконавчими механізмами сортування та відбракування в режимі реального часу;

— впровадження в наявні виробничі лінії, автоматичні системи управління, технологічні процеси та ін.

Додаткові можливості: можливість здійснити за підтримкою однієї системи контроль якості за декількома ознаками: відсутність зовнішніх пошкоджень, наявність маркування та точності упаковки, перевірка складання виробу, наявність маркування та точності упаковки.

Область застосування: сортування у фармацевтичній промисловості, сортування у харчовій промисловості та ін.

1.2.2 Зчитування та розпізнавання міток та маркувань

Системи комп’ютерного зору та відеодатчики вважаються одними з найбільш результативних інструментів у завданнях автоматизації та ідентифікації обліку продукції у виробничому процесі (рисунок 1.2).

Рисунок 1.2 — Автоматичне зчитування 2D-коду та цифрового маркування

Залежно від спеціалізації якої-небудь області, перед або вже після будь-якої технологічної процедури, за допомогою сучасних промислових відеокамер проводиться конкретна приналежність продукції до типу виробу, з занесенням в базу даних інформації, що зчитується [19].

Технологія: системи комп’ютерного зору при ідентифікації здатна зчитувати та розрізняти цифробуквенні та символьні дані різних розмірів, логотипи, 1-d та 2-d штрих-коди, товарні знаки або будь-яку комбінацію даних компонентів.

Як пристрої введення ідентифікаційних даних в системах цього класу застосовуються лазерні сканери та ПЗЗ-камери. Спеціальне програмне

забезпечення пристосовується під вигляд та параметри ідентифікаційного напису, а також під якість поверхні виробу замовника.

Переваги:

— стовідсоткова надійність ідентифікації;

— майже відсутні обмеження відповідно до виду маркування, що розпізнається, і методу його нанесення;

— відсутні обмеження на конфігурацію шрифтів, що застосовуються.

— можливість визначення частково забруднених чи пошкоджених написів;

— проста переналагодження дає можливість розрізняти маркування іншого виду та конфігурації;

— можливість оновлення програмного та апаратного забезпечення;

— системи дають можливість гарантувати забезпечення достовірних відомостей для систем планування та обліку виробництва верхнього рівня.

Додаткові можливості:

— організація синхронного контролю зчитування маркування та якості продукту;

— можливість застосовувати системи комп’ютерного зору та відеодатчики з метою верифікації маркування.

Область застосування: контроль кодування / маркування / етикетки, контроль упаковки в харчовій індустрії, визначення стандартизованого коду на етикетках, пластикових картках, / контроль дати виробництва / терміну придатності у фармацевтиці та ін.

1.3 Готові рішення

В даний час робототехніка отримала застосування практично у всіх сферах діяльності. Потреба у використанні роботизованих систем обумовленанеобхідністю підвищення ефективності виробничих процесів та позбавлення людей важкої та монотонної ручної праці. Домогтися цього дозволяє автоматизація виробництва, тобто. передача функцій управління підприємством та контролю за виготовленням продукції від людини до машини. Приватним

проявом автоматизації є роботизація. Вона є витіснення людей з виробничого процесу із заміною їх на роботів. Автоматизація та роботизація вже кілька десятиліть лежать в основі розвитку промисловості провідних країн. Найбільші підприємства та компанії інвестують величезні кошти у розробку нових технологій та оснащують свої заводи та підприємства сучасними автоматизованими пристроями та роботизованими комплексами, використання яких дозволяє значно збільшити продуктивність та ефективність, скоротити витрати на персонал, оптимізувати логістичні та виробничі процеси, а також підвищити безпеку праці та усунути помилки, пов'язані з людським фактором.

Яскравим прикладом цієї тенденції є активне впровадження автоматизованих систем для роботи на складах та у логістичних центрах. Серед них найбільше застосування отримали автоматично керовані транспортні засоби або AGV (Automated Guided Vehicles) і автономні мобільні роботи або AMR (Autonomous Mobile Robots). Головна відмінність між ними полягає у способі навігації. AGV пересуваються за допомогою навігаційних ліній і маркерів і завжди слідують за заздалегідь встановленими маршрутами, в той час як AMR мають більшу адаптивність і свободу пересування і орієнтується за допомогою датчиків, які допомагають їм будувати карту навколишнього простору, визначати найкоротший маршрут та коригувати його у разі виникнення перешкод. Існує три основні види AGV та AMR — це транспортні візки, вилкові навантажувачі та буксируючі пристрої. Транспортні візки є роботизованими мобільними платформами, які здійснюють транспортування піддонів з деталями на виробничих об'єктах. Вилкові навантажувачі також виконують функції по переміщенню об'єктів, однак вони можуть бути налаштовані для роботи з важкими вантажами, вагою до 4 тонн, або операцій з вертикального навантаження на висоту до 8 метрів. Буксирувальники також призначені для транспортування і здатні тягнути за собою один або кілька візків з товарами або деталями, а їхня вантажопідйомність може досягати 8 тонн [7].

Однією з провідних компаній у галузі автоматизації та роботизації логістичних центрів є Amazon Robotics, дочірнє підприємство корпорації Amazon.

На сьогоднішній день на складах компанії працює понад 100 000 роботизованих систем. Найвідомішою та найпоширенішою моделлю є помаранчевий робот Kiva, який призначений для переміщення великих візків із вантажем у складських приміщеннях (рисунок 1.3) [20].

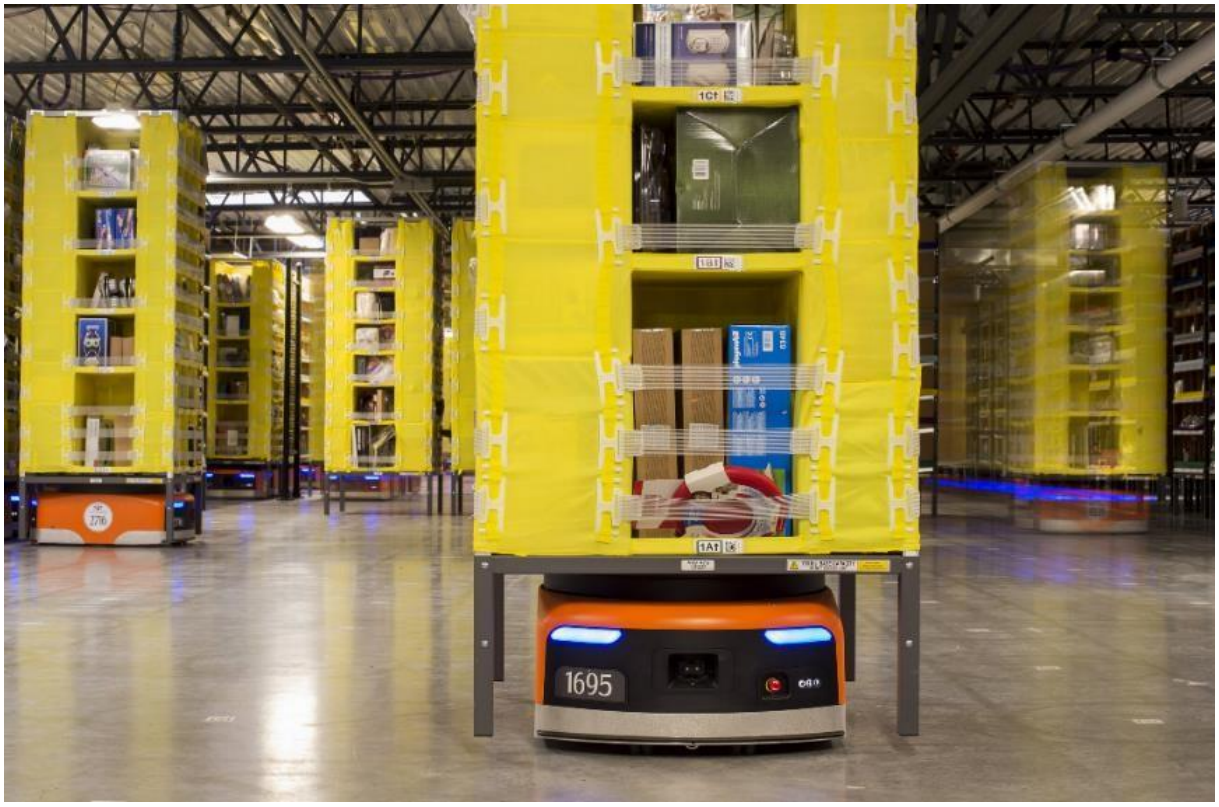

Рисунок 1.3 – Робот Kiva компанії Amazon

Дані AGV оснащені двома камерами із системами комп'ютерного зору: перша розташована на верхній частині корпусу і призначена для зчитування штрих-кодів, наклеєних на мобільні стелажі, для отримання інформації про товари, що зберігаються в них, інша розміщена на дні транспортного засобу, вона дозволяє зчитувати спеціальні штрих - коди на підлозі складу, які відповідають за навігацію робота та формують його шлях з однієї точки до іншої.

Серед розробок варто відзначити універсальні мобільні платформи AGV 1811 від компанії INTEC, призначені для автоматизації внутрішньоцехової логістики (рисунок 1.4). Навігація даних систем заснована на використанні комп'ютерного зору та контрастних ліній, що наносяться на покриття підлоги складських приміщень і формують заздалегідь заданий маршрут руху робота.

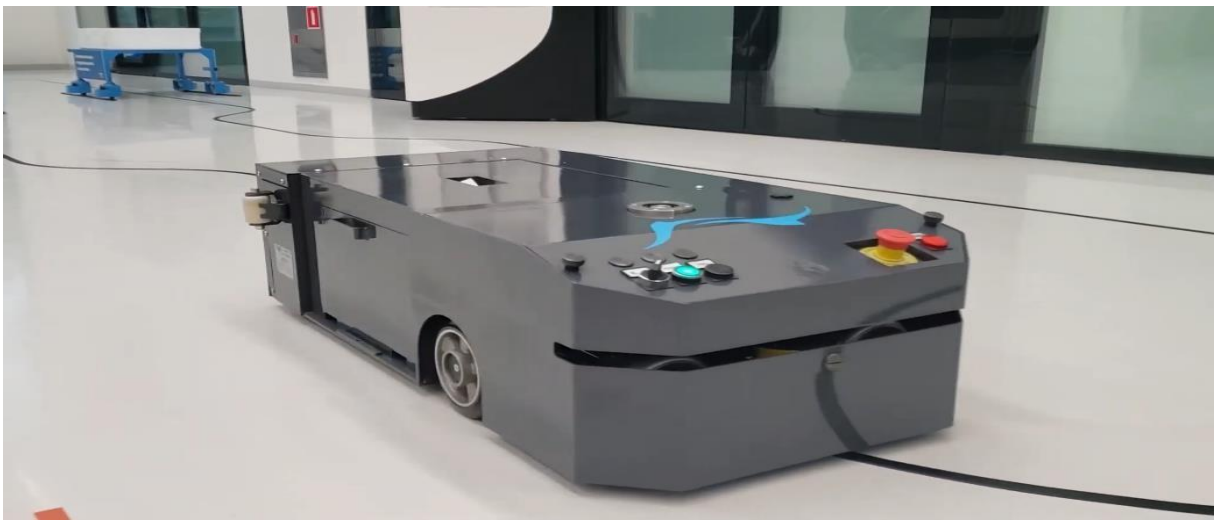

Рисунок 1.4 — AGV 1811 від компанії INTEC

Наступною розробкою є сенсор із вбудованою системою 3D-стереокамер, що дозволяє здійснювати 3D-сприйняття в режимі реального часу та забезпечує 3D-вимірювання та позиціонування у просторі. Таким чином, орієнтація у просторі та технічний зір стають реальністю (рисунок 1.5) [20].

Стерео сенсор KUKA_3D Perception розпізнає неструктуроване оточення в режимі реального часу і точно позиціонує об'єкти з різною кінематикою в просторі. Це перший 3D-сенсор у світі, який наділяє роботів такими можливостями. Результат цього прогресивного рішення: робот може фіксувати своє становище у просторі з точністю до міліметра та вирішувати завдання ще швидше та ефективніше.

Завдяки інтегрованій технології KUKA_3D Perception обробляє зображення безпосередньо в сенсорі і фіксує поточне положення об'єкта з точністю до міліметра. Масиви точок створюються дуже просто. 3D-сенсор може розпізнавати своє оточення як в умовах природного світла, так і за недостатнього освітлення. Навіть за швидких переміщень забезпечується надійна точність. Крім того, в невеликому просторі без труднощів і збоїв можуть працювати кілька сенсорів.

Рисунок 1.5 — Стерео сенсор KUKA _3D Perception

Інтегрована графічна карта дозволяє обробляти зображення глибини у сенсорі. Зовнішні розрахунки не потрібні — ідеально підходить для мобільних робототехнічних систем. Високопродуктивне апаратне забезпечення розроблене для використання у роботизованому середовищі.

Можна легко налаштувати KUKA _3D Perception через веб-браузер за допомогою зручного інтерфейсу. 3D-сенсор також може бути підключений через стандартні інтерфейси, до стандартних бібліотек обробки зображень та інших програмних середовищ, таких як ROS.

Він має два виконання з різними базовими відстанями: 65 мм або 160 мм. Вони дозволяють обчислювати зображення глибини для використання у ближньому чи далекому діапазоні. Подання зображення може бути монохромним або кольоровим.

Ще однією розробкою, що заслуговує уваги є iRVision — це розроблена компанією FANUC система візуального виявлення за технологією plug&play. Вона повністю інтегрована з контролером R-30iB, швидко встановлюється, відрізняється простотою використання та високою гнучкістю. Завдяки системі розпізнавання двох або тривимірних деталей вона може визначати місце розташування довільно

розташованих виробів будь-якої форми і розміру. Вона також може зчитувати штрих-коди, сортувати за кольором, забезпечувати гнучку подачу деталей, високошвидкісне візуальне лінійне відстеження (iRPickTool) і взяття коробів/панелей. Система iRVision відіграє ключову роль у збільшенні продуктивності та забезпечує додаткову економію коштів, оскільки знімає необхідність у технологічному оснащенні (рисунок 1.6) [20].

Одним із рішень, що використовують цю технологію, є FANUC 3D Area Sensor.

Високошвидкісний просторовий бар'єрний датчик FANUC 3D Area Sensor повністю інтегрований з контролером робота та служить для миттєвого створення просторових карток з використанням технології технічного зору. Вона дозволяє роботу ідентифікувати і брати окремі деталі, досягаючи тривалості циклу від 8 до 12 секунд, навіть якщо вони вкриті брудом, іржею, маслом або, якщо мова про пакети, не мають чітких характеристик. Датчик легко настроюється за допомогою комп'ютера.

Рисунок 1.6 — Система iRVision

2 ПРОЕКТУВАННЯ АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

Як основа для реалізації подібних систем найчастіше використовуються одноплатні комп'ютери (Single-Board Computers), мініатюрні самодостатні електронно-обчислювальні пристрої, апаратна частина яких розміщується лише на одній-єдиній електронній платі, за розмірами приблизно порівнянною з сучасною банківською пластиковою карткою. Таким чином, головною перевагою цих девайсів є компактність та мобільність. Крім того, подібні мікрокомп'ютери мають непогану продуктивність, якщо враховувати клас пристрою, і можуть бути використані для вирішення безлічі різних прикладних завдань. Також варто відзначити їхню досить низьку вартість і, як наслідок, популярність. Ще одноплатні комп'ютери мають можливість встановлення на них різних операційних систем, що полегшує їх використання.

Для виявлення напрямної лінії використовується невелика веб-камера, яка здійснює захоплення зображення для подальшої передачі на мікрокомп'ютер. Далі відбувається обробка отриманого кадру, і основі її результатів виробляються управляючі впливи, передані на мікроконтролер, безпосередньо пов'язані з моторами транспортного засобу.

Таким чином, першим етапом розробки система реєстрації та розпізнавання напряму руху за допомогою комп’ютерного зору є вибір найбільш підходящої моделі одноплатного комп'ютера.

2.1 Вибір одноплатного комп'ютера

Найпопулярнішою серією одноплатних комп'ютерів на сьогоднішній день вважається Raspberry Pi. Для вирішення поставлених завдань було зроблено вибір на користь одноплатного комп'ютера Raspberry Pi 3 Model B. Зовнішній вигляд цієї моделі представлений рисунку 2.1.

Рисунок 2.1 — Одноплатний комп’ютер Raspberry Pi 3 Model B

Серед параметрів даного одноплатного комп'ютера варто відзначити потужніший у порівнянні з попередньою моделлю лінійки чотириядерний процесор Cortex-A53, що працює на частоті 1,2 ГГц. Об'єм оперативної пам'яті Raspberry Pi 3 Model B складає 1 Гб. Ця модель містить чотири порти USB 2.0, порт HDMI для підключення монітора, виведення для кабелю Ethernet, а також вбудовані адаптери Bluetooth і Wi-Fi.

Даний одноплатний комп'ютер має відмінну продуктивність, що є необхідною умовою при роботі з алгоритмами комп'ютерного зору, містить достатню кількість USB портів, підтримує використання бездротової передачі даних за допомогою Bluetooth і Wi-Fi.

2.2 Вибір веб-камери

Іншим важливим елементом системи, що розробляється, є веб-камера, яка буде здійснювати захоплення зображення для наступної його передачі на Raspberry Pi. Для вирішення поставлених завдань було зроблено вибір на користь невеликої веб-камери Logitech C270. Зовнішній вигляд цієї моделі представлений на рисунку

2.2.

Рисунок 2.2 — Logitech C270

Logitech C270 здійснює відеозапис у форматі високої чіткості (HD) з роздільною здатністю 720p (1280×720). Фотографії камера може робити з роздільною здатністю 3 Мп. Її діагональне поле зору становить 55º.

Оскільки камера розташовуватиметься відносно близько до підлоги, невеликий кут огляду ніяк не позначиться на роботі системи. Крім того, для виявлення напрямної лінії не потрібно великої роздільної здатності пристрою, наведені вище характеристики чудово підходять для виконання поставленого завдання. Також варто відзначити низьку вартість цієї моделі, це відмінний варіант за співвідношенням ціна-якість.

2.3 Складання структурної схеми пристрою, що розробляється

Першим етапом розробки схемотехніки електронного устрою є графічне уявлення сукупності основних ланок об'єкта і зв'язків з-поміж них, тобто складання його структурної схеми.

Структурна схема розроблюваної системи представлена рисунку 2.3.

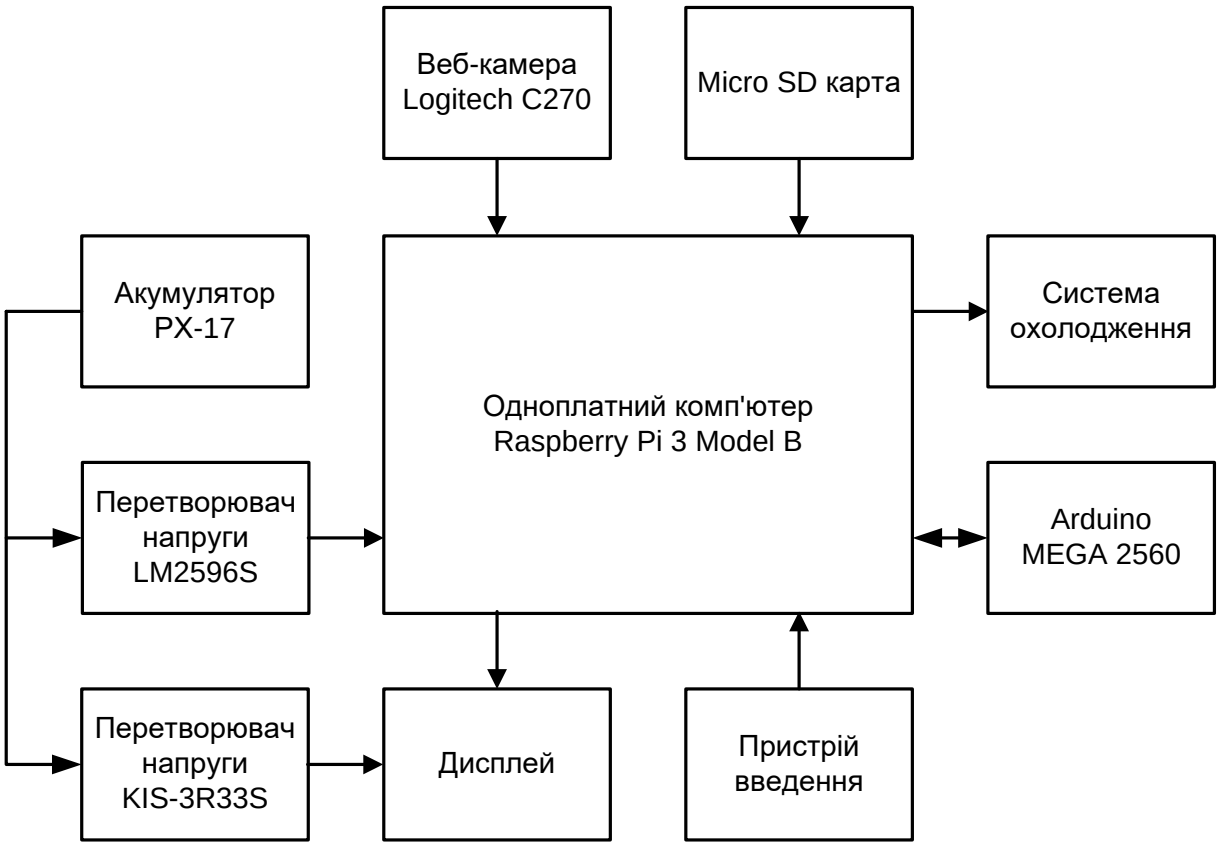

Рисунок 2.3 — Структурна схема системи, що розробляється

Як видно, система, що розробляється, складається з десяти основних елементів:

— комп'ютер Raspberry Pi 3 Model B;

— веб-камера Logitech C270;

— акумулятора PX-17;

— перетворювач напруги LM2596S;

— перетворювач напруги KIS-3R33S;

— дисплей;

— microSD карта;

— Arduino MEGA 2560;

— пристрої введення (клавіатура та комп'ютерна миша);

— вентилятор, що здійснює охолодження мікрокомп'ютера.

Як джерело живлення для даної системи використовується літієвий акумулятор PX-17 фірми PowerX з величиною вихідної напруги 11,1 В та ємністю

3000 мА*год (рис.2.4.)

Рисунок 2.4 — Акумулятор літій-іонний (Li-Ion) 11.1 В 3000 мА*год

Варто відзначити, що не всі елементи системи, що розробляється, можуть бути безпосередньо підключені до даного акумулятора. Наприклад, для живлення одноплатного комп'ютера та дисплея необхідна напруга живлення 5 В, однак ці пристрої не оснащені вбудованими перетворювачами напруги.

Щоб усунути цю проблему, було прийнято рішення використовувати для живлення від акумулятора мікрокомп'ютера регульований імпульсний понижувальний перетворювач напруги LM2596S (рисунок 2.5, зліва), а для живлення дисплея — нерегульований перетворювач напруги KIS-3R33S (рисунок 2.5, праворуч).

Рисунок 2.6 — Перетворювачі напруги LM2596S та KIS-3R33S

Як пристрій для відображення даних був використаний LCD дисплей Waveshare з діагоналлю 4,3 дюйми та роздільною здатністю 480x272 пікселя, розроблений спеціально для роботи з Raspberry Pi (рисунок 2.7).

Рисунок 2.7 — LCD дисплей Waveshare для виведення інформації

Карта пам'яті або флеш-карта — компактний електронний носій інформації, що використовується для зберігання цифрової інформації. Сучасні карти пам'яті виготовляються на основі флеш-пам'яті, хоча принципово можуть використовуватися й інші технології. Карти пам'яті широко використовуються в електронних пристроях, включаючи цифрові фотоапарати, стільникові телефони, ноутбуки, MP3-плеєри та ігрові консолі. Карти пам'яті є компактними, перезаписуваними, і, крім того, вони можуть зберігати дані без споживання енергії (енергонезалежність). В якості такої карти застосовано MicroSD карта пам'яті SanDisk Extreme Pro microSDXC UHS I 64GB Class A2 V30, яку показано на рисунку 2.8.

Рисунок 2.8 — Карта пам'яті SanDisk Extreme Pro microSDXC UHS I 64GB

Принцип роботи системи полягає в тому, що одноплатний комп'ютер Raspberry Pi отримуватиме зображення з веб-камери, оброблятиме отриману інформацію і передаватиме її на мікроконтролер Arduino, пов'язаний з ним за допомогою послідовного зв'язку через USB кабель і керуючий моторами мобільного робота.

Arduino Mega 2560 (рис. 2.9) є вибором, який має свої переваги. Існує кілька ключових причин, чому обрано Arduino Mega 2560 як керуючу платформу.

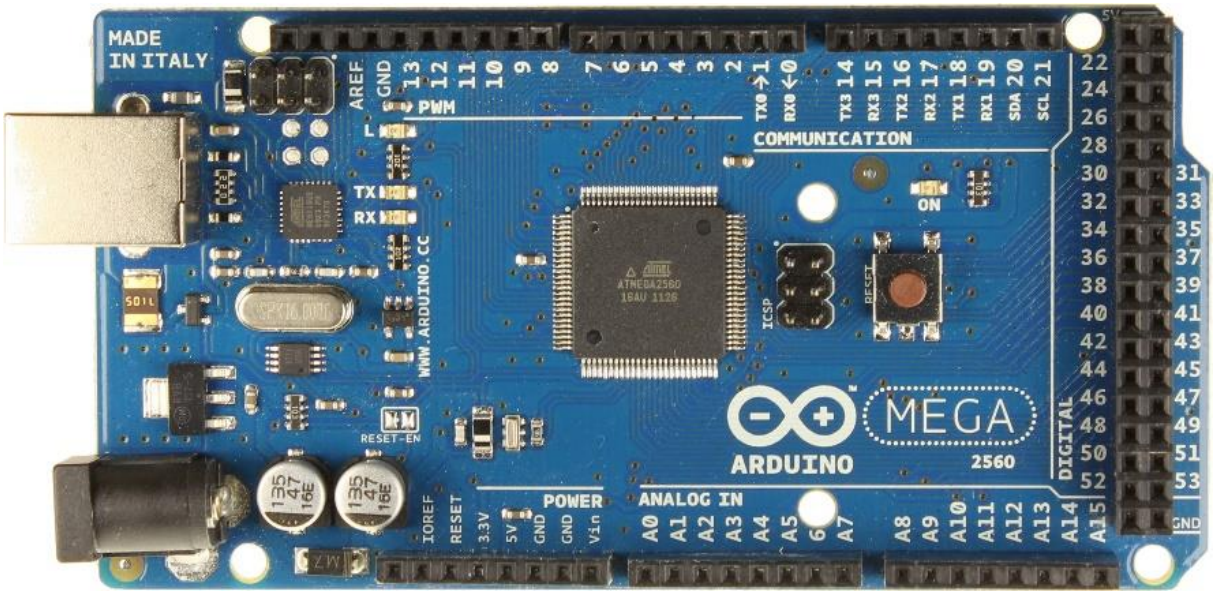

Рисунок 2.9 — Платформа Arduino Mega

Arduino Mega 2560 має значно більшу кількість портів введення-виведення порівняно з іншими моделями Arduino. Це дозволяє підключати більше сенсорів, пристроїв та периферійних пристроїв без необхідності використовувати додаткові проміжні плати або розширювачі.

Arduino Mega 2560 має потужний мікроконтролер ATmega2560 з великою кількістю вбудованої пам'яті програм та EEPROM. Це дає можливість розробляти складніші програми та зберігати більше даних без необхідності використання зовнішніх засобів збереження.

Arduino Mega 2560 підтримує багато різних типів комунікаційних інтерфейсів, таких як UART, I2C, SPI, що дозволяє легко взаємодіяти з іншими пристроями та модулями. Ця гнучкість забезпечує широкі можливості для з'єднання та обміну даними з різними пристроями.

Arduino Mega 2560 має широке співтовариство розробників та підтримку онлайн-ресурсів. Це означає, що можна швидко знайти документацію, приклади коду, підказки та рішення проблем, що допомагає зберегти час та спрощує процес розробки.

Загалом, вибір Arduino Mega 2560 має широкі можливості, потужний мікроконтролер, розширені комунікаційні властивості та підтримка співтовариства зробили його ідеальним вибором у розробці.

В якості пристроїв введення обрано Raspberry Pi Foundation USB-клавіатуру і мишу, які показано на рисунку 2.10, які мають стандартне призначення, а тому їх опис упущено.

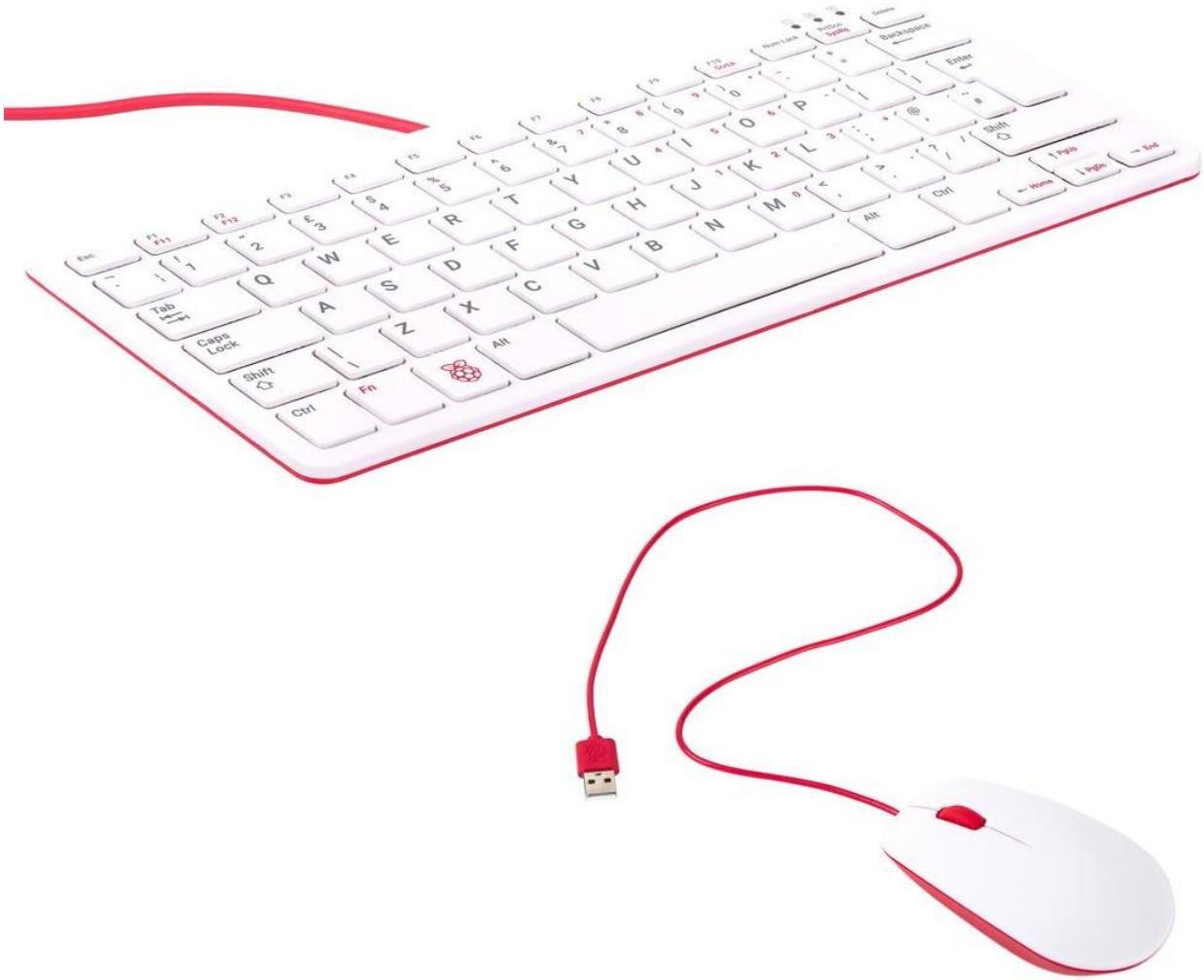

Рисунок 2.10 — Raspberry Pi Foundation USB-клавіатура і миша

Як система охолодження для Raspberry Pi буде виступати невеликий вентилятор, розташований над мікрокомп'ютером. Його використання в цій системі вкрай бажано, тому що подібні одноплатні обчислювальні пристрої мають властивість досить швидко нагріватися, що при високих температурах навколишнього середовища нерідко призводить до виходу їх з ладу та неможливості їхнього подальшого використання.

2.4 Складання електричної схеми пристрою, що розробляється

2.4.1 Підключення Raspberry Pi до Arduino

Для того, щоб працювати з інтерфейсом GPIO Raspberry Pi 3 Model B, який призначений для забезпечення зв'язку одноплатного комп'ютера з додатковими компонентами (модулями), що підключаються до нього, необхідно детально розібратися в пристрої та призначенні його основних цифрових входів і виходів (пінів). Конфігурація контактів або, інакше кажучи, розпінання даного мікрокомп'ютера представлена рисунку 2.11.

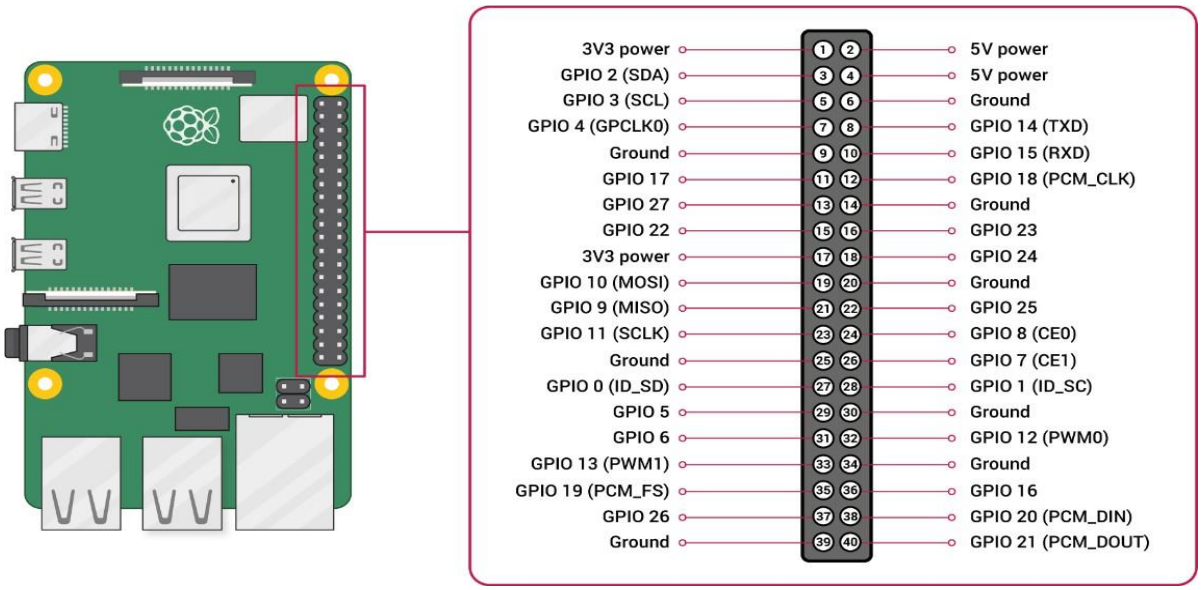

Рисунок 2.11 — Розпинання Raspberry Pi 3 Model B

Число пінів GPIO Raspberry Pi 3 Model B дорівнює 40. Всі вони пронумеровані і діляться на три основні групи:

— живлення (на схемах маркуються Power);

— заземлюючі (GND, Ground);

— порти (часто позначаються як GPIO).

Перші необхідні подачі різних напруг — 3.3 і 5 В. Другі забезпечують безпеку роботи плати, відводячи електрику. Треті ж виступають як інтерфейси, здатні приймати і відправляти сигнали. Саме до них користувач підключає додаткові модулі та периферійні пристрої.

Також варто відзначити, що закладена в центральний процесор логічна нумерація пінів відрізняється від наведеної раніше на фізичній схемі. Так, що «штирьки», що не виконують функцій введення-виведення одноплатного комп'ютера номерів не мають, що слід враховувати при написанні коду, оскільки програмне забезпечення орієнтується саме на логічні номери.

Більшість периферії підключається по існуючих зовнішніх інтерфейсів, а саме — USB.

Проте як пристрій, що управляє швидкістю і напрямом обертання моторів мобільного робота на підставі даних, що надійшли з веб-камери і пройшли обробку в одноплатному комп'ютері Raspberry Pi, виступає мікроконтролер Arduino MEGA 2560. Його зв'язок з мікрокомп'ютером реалізований за допомогою UART (Universal Asynchronous Receiver), універсального асинхронного приймача, який є логічною схемою, призначеною для організації обміну інформацією з іншими цифровими пристроями. Фізичне з'єднання двох девайсів здійснюється за допомогою кабелю USB. Програмне ж передбачає написання спеціального коду для кожного з пристроїв, а також підключення додаткових ресурсів, зокрема бібліотеки WiringPi, що забезпечує доступ до GPIO-контактів та UART інтерфейсу Raspberry Pi для програм, написаних мовою C і C++.

Першим етапом у процесі встановлення зв'язку між компонентами системи, що розробляється, є підключення бібліотеки WiringPi. Однак перш ніж перейти безпосередньо до її встановлення, потрібно завантажити всі необхідні для цього файли, скориставшись інтернет-ресурсами, розмістити їх на переносному USB-флеш-накопичувачі, після чого підключити його до одного з USB портів Raspberry Pi і перенести дані на комп'ютер. Так як операційною системою даного пристрою є Linux, наступним кроком стане введення в консоль мікрокомп'ютера набору певних команд, наведених нижче:

```
cd
tar xfz wiringPi-98bcb20.tar.gz
cd wiringPi-98bcb20
```

./build

Після цього автоматично запуститься процес встановлення завантаженої бібліотеки. Варто зазначити, що цифри та літери після запису wiringPi (в даному випадку це 98bcb20) є ідентифікатором версії бібліотеки..

Другим етапом є визначення номера послідовного порту, якого підключений мікроконтролер Arduino. Для цього можна скористатися наведеною консольною командою: ls /dev/tty*.

У результаті виконання на екрані висвітиться список всіх доступних послідовних пристроїв (рисунок 2.12). Після цього потрібно відключити USB провід, що з'єднує Raspberry Pi та Arduino, і знову ввести цю команду. Найменування в оновленому списку відсутнє і буде шуканим номером порту.

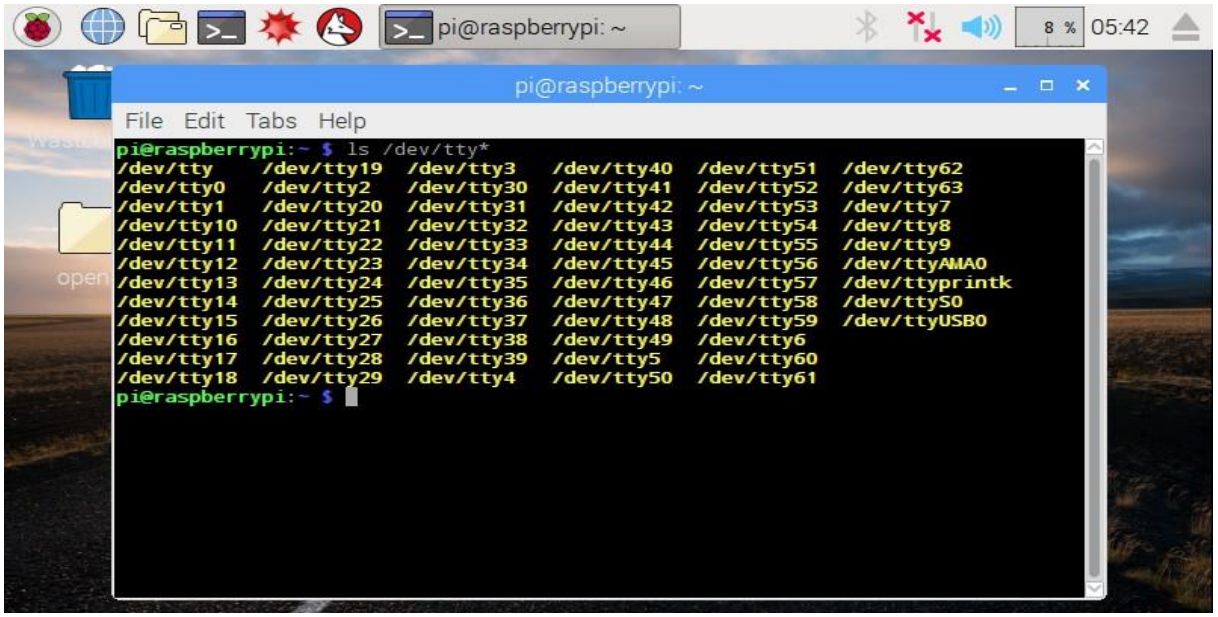

Рисунок 2.12 — Результат виконання команди ls /dev/tty*

Третім і заключним етапом налаштування з'єднання є написання невеликих тестових програм для кожного пристрою для перевірки процесу обміну даними.

Лістинг програми для Raspberry Pi мовою C++ наведено нижче:

```
#include <iostream>
#include <wiringPi.h>
#include <wiringSerial.h> using
namespace std;
```

```
int main()
int serialPort = serialOpen("/dev/ttyUSB0", 9600); if
(serialPort == -1)
{
cout << "Failed open port" << endl; return 0;
}
while (true)
{
serialPutchar(serialPort, 'f'); cout << "f" <<
endl;
}
serialClose(serialPort);return 0;
}
```

Розберемо цю програму докладніше. Насамперед у ній відбувається підключення заголовних файлів, необхідні роботи коду. Бібліотека iostream є частиною стандартної бібліотеки C++ та відповідає за введення інформації з клавіатури та виведення даних у консоль. Бібліотека wiringPi дозволяє звертатися до GPIO інтерфейсу Raspberry Pi, а бібліотека wiringSerial надає інструменти для роботи з послідовним портом мікрокомп'ютера.

Далі всередині основної функції main відбувається ініціалізація послідовного порту, до якого підключений мікроконтролер Arduino, допомогою команди serialOpen(), що приймає як аргументи номер порту, визначений раніше, і швидкість передачі даних. У цій програмі було вирішено використовувати стандартну швидкість передачі даних 9600 бод. Це число визначає кількість двійкових бітів, що відправляються по послідовній лінії за секунду. Контроль за станом порту здійснюватиметься за допомогою змінної serialPort типу int [21].

Нижче відбувається перевірка успішності відкриття послідовного порту за допомогою конструкції if. Якщо змінна serialPort містить значення –1, це означає, що зв'язок між пристроями не була встановлена, внаслідок чого консоль виведеться повідомлення про помилку, а програма завершить свою роботу.

Якщо ж підключення Raspberry Pi до Arduino пройшло успішно, програма потрапляє в нескінченний цикл і починає відправляти в мікроконтролер і виводити в консоль символ «f», що управляє, що означає, що лінія розпізнана, а мобільний робот повинен рухатися прямо. Відповідно позначено й інші команди: повернути ліворуч — "l", повернути праворуч — "r". Варто зазначити, що використання єдиного символу f замість цілого слова forward підвищує швидкодію пристрою.

Вихід із нескінченного циклу означає зупинку мобільного робота, закриття послідовного порту за допомогою команди serialClose() та завершення роботи всієї програми внаслідок якоїсь події. Це, наприклад, може бути втрата напрямної лінії, натискання певної кнопки користувачем або поява в зоні видимості червоної камери

"Стоп" прапорець. У цьому тестовому прикладі така умова не передбачено, так як аналіз додаткових змінних ускладнив би і сповільнив процес перевірки з'єднання між Raspberry Pi Arduino, тому вихід з циклу здійснюється примусовим закриттям консолі.

Блок-схема алгоритму передачі з Raspberry Pi на Arduino представлена рисунку 2.11. Далі перейдемо до алгоритму роботи мікроконтролера. Лістинг програми для Arduino наведено нижче:

```
void setup()
{
Serial.begin(9600);
}
void loop()
{
if (Serial.available() > 0)
char getCh = Serial.read(); Serial.println(getCh);
}
```

Рисунок 2.13 — Блок-схема алгоритму передачі

Розберемо цю програму докладніше. Насамперед у блоці setup відбувається

налаштування послідовного порту та встановлення швидкості передачі даних у 9600 бод за допомогою команди Serial.begin(). Далі в блоці loop здійснюється перевірка наявності даних у буфері порту за допомогою перевірки значення, яке повертається функцією Serial.available(). Якщо керуючий символ прийшов на Arduino, він записується в змінну getCh типу char за допомогою Serial.read() команди для подальшого використання. Наприкінці програма виводить отримані дані монітор послідовного порту з допомогою функції Serial.println().

Блок-схема алгоритму прийому даних Arduino с Raspberry Pi представлена рисунку 2.14.

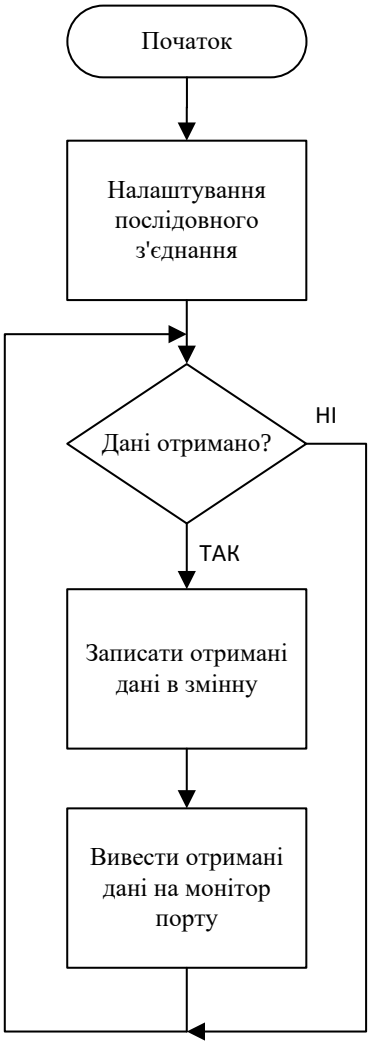

Рисунок 2.14 — Блок-схема алгоритму прийому даних

Після запуску основної програми, представленої в Додатку Б, одноплатний

комп'ютер Raspberry Pi розпізнає лінію, визначить напрямок руху і відправить відповідний символ на Arduino. У свою чергу, мікроконтролер подасть напругу на мотори, і мобільний робот поїде встановленим маршрутом.

2.4.2 Електрична схема пристрою, що розробляється

Схема підключення вентилятора до Raspberry Pi 3 Model B представлена рисунку 2.15.

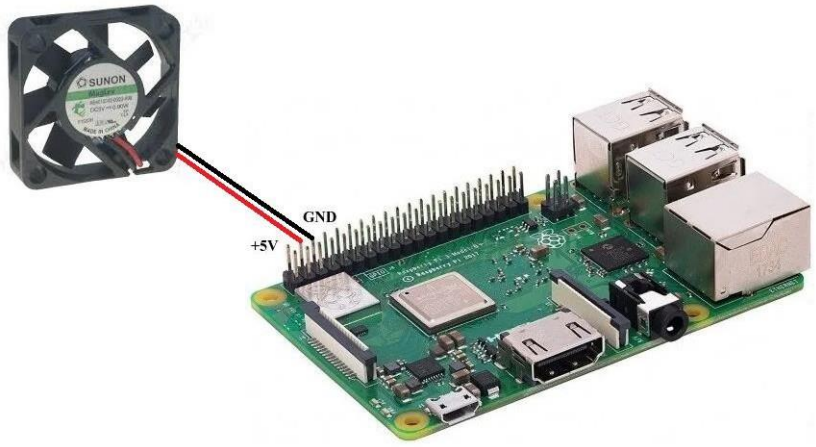

Рисунок 2.15 — Підключення вентилятора до Raspberry Pi 3 Model B

Як видно, вентилятор має лише два дроти: живлення та заземлення, які підключаються до пін Raspberry Pi під номерами 4 та 6 відповідно.

На рисунку 2.16 представлена схема електричних з'єднань розроблюваного пристрою.

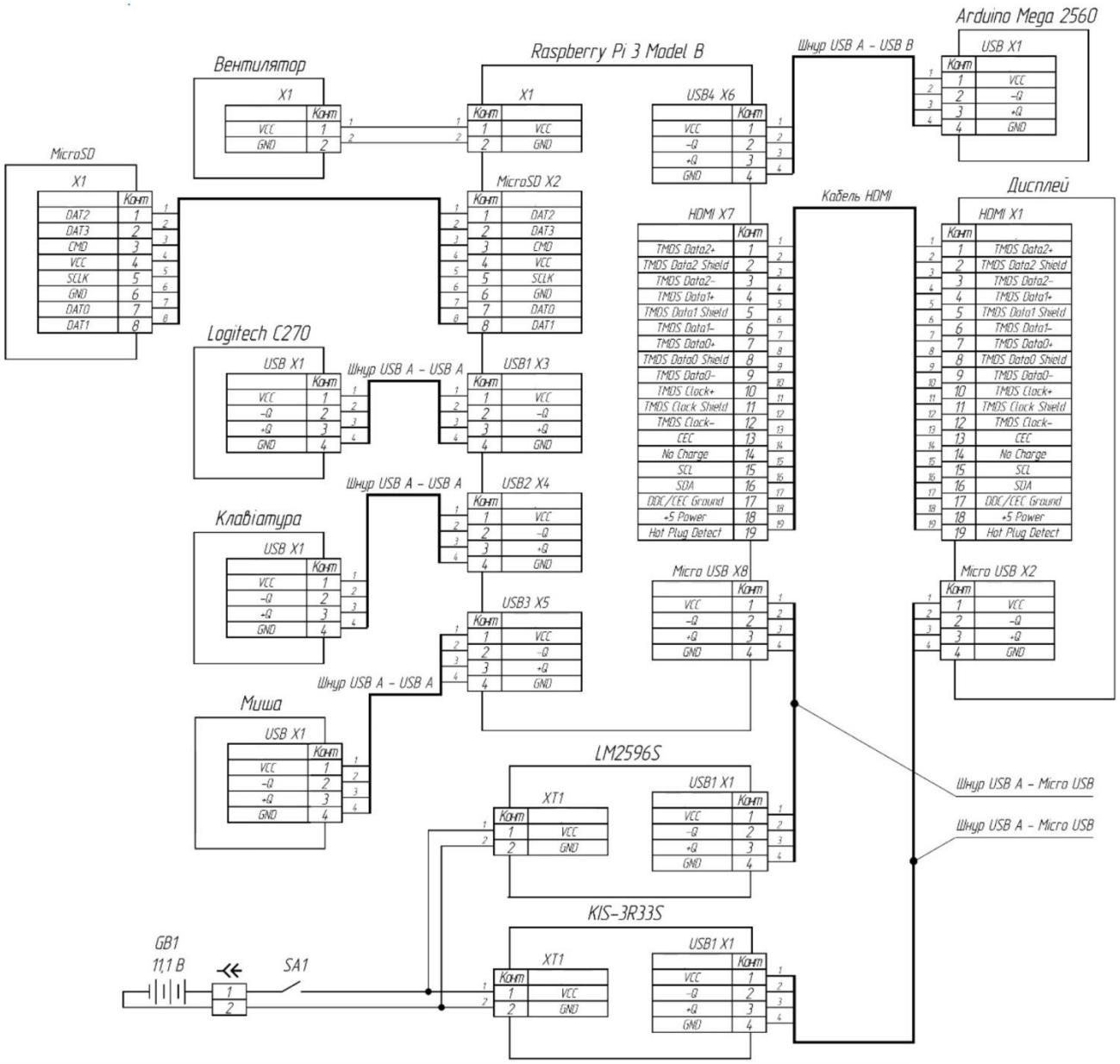

Рисунок 2.16 — Схема електричних з'єднань

3 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Використання інструментів розробки

Для вирішення поставлених завдань було обрано мову програмування високого рівня C++, бібліотека комп'ютерного зору з відкритим вихідним кодом OpenCV та кросплатформове середовище розробки Code::Blocks.

3.1.1 Мова програмування C++

C++ є високорівневим компілюваним мовою програмування загального призначення, який підходить для створення різних додатків. На сьогоднішній день, згідно з індексом TIOBE, який оцінює популярність мов програмування, на основі підрахунку результатів пошукових запитів, C++ входить у топ 4 найпоширеніших мов у світі разом з Python, C і Java (рисунок 3.1).

Apr 2023	Apr 2022	Change	Programming Language	Ratings	Change
1	**1**		**Python**	**14.51%**	**+0.59%**
2	2		C	14.41%	+1.71%
3	3		Java	13.23%	+2.41%
4	4		C++	12.96%	+4.68%
5	5		C#	8.21%	+1.39%
6	6		Visual Basic	4.40%	-1.00%
7	7		JavaScript	2.10%	-0.31%
8	9	^	SQL	1.68%	-0.61%
9	10	^	PHP	1.36%	-0.28%
10	13	^	Go	1.28%	+0.20%

Рисунок 3.1 — Рейтинг найпопулярніших мов програмування

З основних переваг C++ є його висока продуктивність та швидкість, які досягаються завдяки кільком факторам.

По-перше, C++ входить у групу мов програмування, що компілюються. Програма, написана такою мовою, спочатку перетворюється компілятором на машинний код (компілюється) і записується у виконуваний файл, який потім

безпосередньо взаємодіє з апаратною складовою. Такий підхід дозволяє досягти вищої швидкості виконання програми і, отже, більшої ефективності. Інтерпретовані мови програмування, наприклад Python, працюють за іншим принципом. Написані на них програми пропускають етап компіляції та перетворюються на машинний код прямо під час виконання за допомогою іншої програми — інтерпретатора, що сповільнює весь процес.

По-друге, C++ є мовою програмування із сильною статичною типізацією. Статична типізація передбачає, що перевірка даних заданим типу виконується на етапі компіляції. З цього випливає, що для успішного запуску програми програмісту потрібно ще на етапі написання коду оголошувати типи змінних перед їх використанням. Даний підхід робить синтаксис жорсткішим, проте дозволяє знизити кількість помилок і прискорити виконання програм за рахунок того, що компілятор, знаючи з якими саме типами даних він має справу, може проводити додаткові маніпуляції щодо оптимізації коду. Сильна типізація означає, що мова програмування не дозволяє поєднувати у виразах різні типи даних та не здійснює їх автоматичні приховані перетворення. Це змушує розробників бути більш уважними під час написання коду, але водночас підвищує швидкість виконання програм.

По-третє, C++ успадкував від мови C багаті можливості роботи з пам'яттю, завдяки яким програмісти можуть самостійно вирішувати, в яких саме осередках і як довго зберігати інформацію, і мають можливість керувати низькорівневими ресурсами, такими як регістри процесора, кеш та ін. в C++ відсутнє автоматичне складання сміття. Це дозволяє підвищити продуктивність і прискорити виконання програм, оскільки видалення непотрібних об'єктів не витрачається час, проте відповідальність за ці дії перекладається з машини на розробників, змушених особисто контролювати виділення пам'яті.

Таким чином, на сьогоднішній день C++ є однією з найпродуктивніших і найшвидших мов програмування. Доказом цього є експеримент, проведений португальськими вченими, які протестували 27 найпопулярніших мов програмування шляхом компіляції/виконання програм з використанням сучасних

компіляторів, віртуальних машин, інтерпретаторів та бібліотек та у кожному випадку виміряли такі параметри, як час виконання, споживання пам'яті та енергоспоживання. Результати цього дослідження представлені рисунку 3.2.

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

Рисунок 3.2 — Результати дослідження енергоефективності найпопулярніших мов програмування

Як видно, C++ показує відмінні результати у всіх категоріях і входить до топ-5 найенергоефективніших мов програмування. Це особливо актуально при роботі із зображеннями та 3D, де потрібна висока швидкість роботи та продуктивність. Саме тому C++ нерідко застосовується для створення графічних додатків та різних прикладних програм, а також його часто використовують для створення комп'ютерних ігор з багатою і насиченою візуалізацією.

Іншою значною перевагою C++ є його універсальність. Наявність компіляторів, бібліотек та інструментів розробки майже під усі поширені операційні системи дозволяє легко переносити написані цією мовою програми з однієї платформи на іншу. Наприклад, браузер Chrome, написаний переважно мовою C++, працює і Windows, і Linux, і на macOS.

Також C++ вважається чудовим фундаментом вивчення програмування. Незважаючи на складний синтаксис, ця мова є еталонною, на її прикладі було розроблено багато сучасніших мов програмування, такі як C#, JavaScript, Java, що мають більш просту структуру.

Таким чином, беручи до уваги всі перераховані вище переваги мови C++, було прийнято рішення про використання його як основи для реалізації програмної частини.

3.1.2 Бібліотека комп'ютерного зору OpenCV

На сьогоднішній день існує безліч додатків, призначених для розробників та дослідників в області комп'ютерного зору та штучного інтелекту: AForge.NET, VXL, LTI та ін. Але найпопулярнішою є бібліотека OpenCV [22].

OpenCV (Open Source Computer Vision Library) є бібліотекою комп'ютерного зору з відкритим вихідним кодом. Вона реалізована мовами програмування C/C++, проте поставляється також інших мов, таких як Python, Java, Ruby, Matlab, Lua, тощо. OpenCV підтримує різні операційні системи, до яких належать Windows, Linux, MacOS, iOS, Android та інших. Бібліотека поширюється безкоштовно за ліцензією BSD, тобто. її можна використовувати у проектах з відкритим вихідним кодом, а й у закритих, комерційних проектах [22, 23].

OpenCV містить понад 2500 оптимізованих алгоритмів комп'ютерного зору та машинного навчання, що дозволяють вирішувати найрізноманітніші завдання. Ось деякі з них:

— виявлення та розпізнавання осіб;

— ідентифікація об'єктів;

— класифікація дій людини на відео;

— відстеження об'єктів, що рухаються;

— розпізнавання та вилучення 3D-моделей за двовимірними зображеннями;

— зшивання зображень для отримання більш високої роздільної здатності;

— знаходження схожих зображення у базі даних;

— видалення червоних очей з фотографій, зроблені за допомогою спалаху.

Завдяки своєму простому інтерфейсу та широкому функціоналу OpenCV широко використовується багатьма великими компаніями для розробки своїх додатків, а також вченими в науково-дослідних роботах. Наприклад, бібліотека застосовувалася для зшивки супутникових карт, зменшення шуму на медичних знімках, у безпілотних наземних, повітряних та підводних апаратах [22].

Загальна карта проекту OpenCV представлена рисунку 3.3.

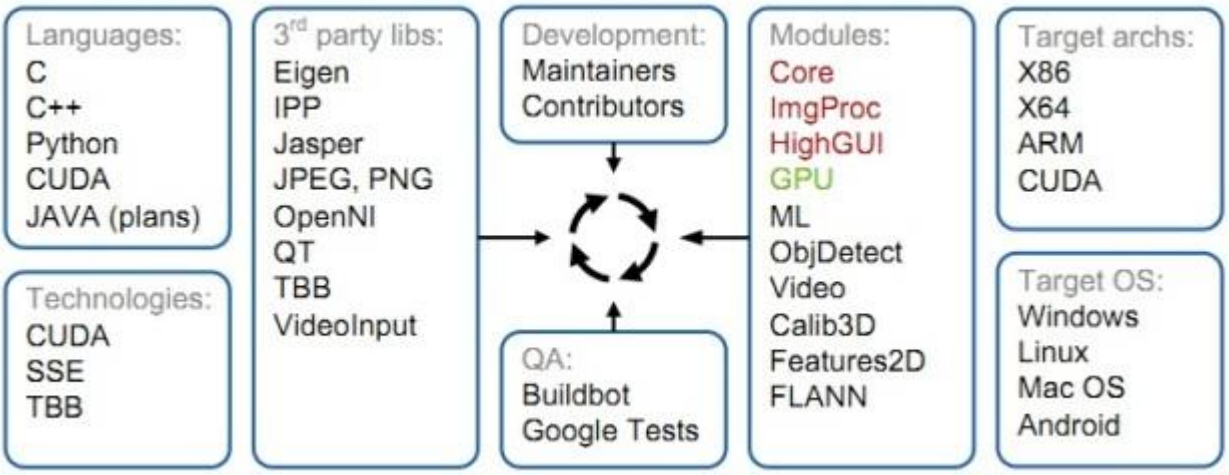

Рисунок 3.3 — Карта проекту OpenCV

Бібліотека OpenCV є структурованою бібліотекою. Вона розділена на чотири основні компоненти:

— Core — модуль, який містить базові структури даних та алгоритми (базові операції над багатовимірними числовими масивами, матрична алгебра, математичні функції, генератори випадкових чисел, базові функції 2D-графіки, запис/відновлення структур даних в/з XML);

— CV — модуль обробки зображень та комп'ютерного зору, який дозволяє проводити базові операції над зображеннями (фільтрація, геометричні перетворення, перетворення колірних просторів), аналіз зображень (вибір відмітних ознак, морфологія, пошук контурів), аналіз руху, стеження за об'єктами, виявлення об'єктів, зокрема осіб;

— HighGUI — модуль для введення/виведення зображень і відео, створення інтерфейсу користувача (захоплення відео з камер і з відеофайлів, читання/запис статичних зображень, функції для організації простого UI);

— ML — модуль, що реалізує алгоритми машинного навчання (метод найближчих сусідів, дерева рішень, бустинг, градієнтний бустинг дерев рішень, випадковий ліс, машина опорних векторів, нейронні мережі та ін.) [22].

Структура бібліотеки OpenCV показана рисунку 3.4.

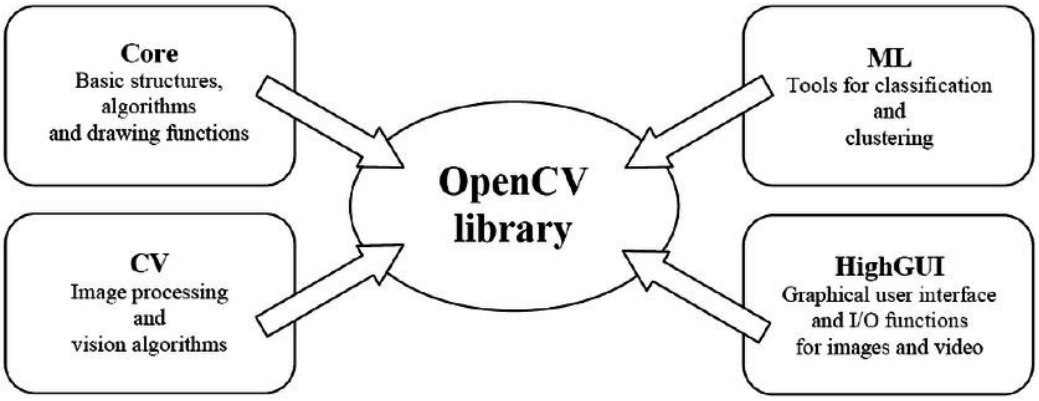

Рисунок 3.4 — Структура бібліотеки OpenCV

Таким чином, простий інтерфейс та величезна кількість корисних функцій роблять OpenCV чудовим інструментом для вирішення задачі реєстрації та розпізнавання напряму руху за допомогою комп’ютерного зору.

3.1.3 Середовище розробки Code::Blocks

Code::Blocks є безкоштовним кросплатформовим інтегрованим середовищем розробки (Integrated Development Environment — IDE) з відкритим вихідним кодом для програмування мовою C/C++. Дана IDE має версії найпопулярніших операційних систем, серед яких Windows, Linux, MacOS.

Основною перевагою Code::Blocks є простий, зрозумілий і зручний інтерфейс користувача (рисунок 3.5). Він побудований на основі плаваючих і розтягуються док-панелей, які можна пристиковувати до будь-яких сторін головного вікна програми, просто перетягуючи їх мишкою. Завдяки цій функції можна дуже гнучко налаштовувати різні компонування інтерфейсу для різних розмірів екрану. Якщо кілька моніторів, ця IDE дозволяє відокремити деякі панелі від основного вікна і розмістити їх на сусідніх моніторах.

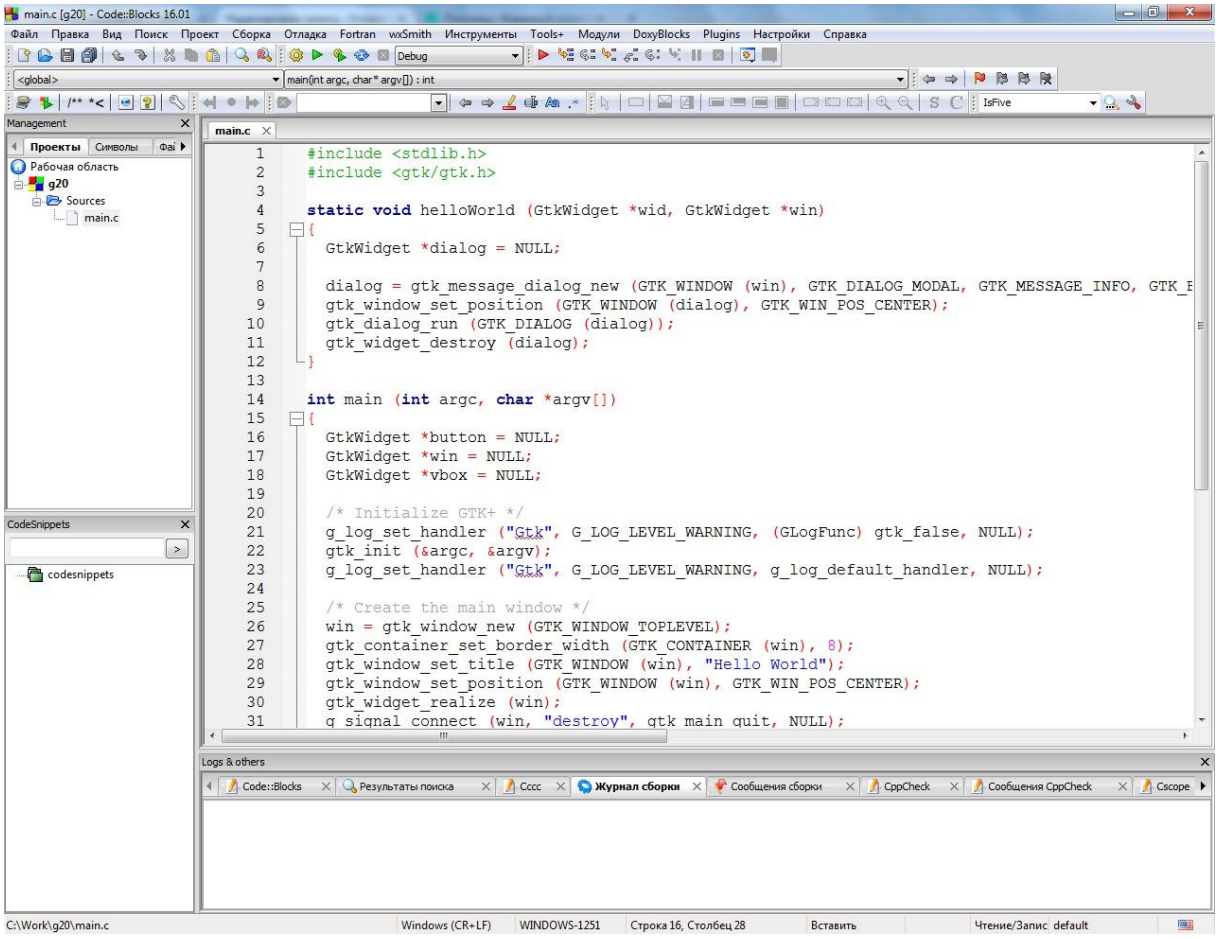

Рисунок 3.5 — Інтерфейс Code::Blocks

Ще одним плюсом Code::Blocks є наявність безлічі готових вбудованих шаблонів проектів, що дуже зручно, так як це заощаджує час користувача, дозволяючи йому пропустити етап налаштування програми та відразу приступити до написання коду (рисунок 3.6) [24].

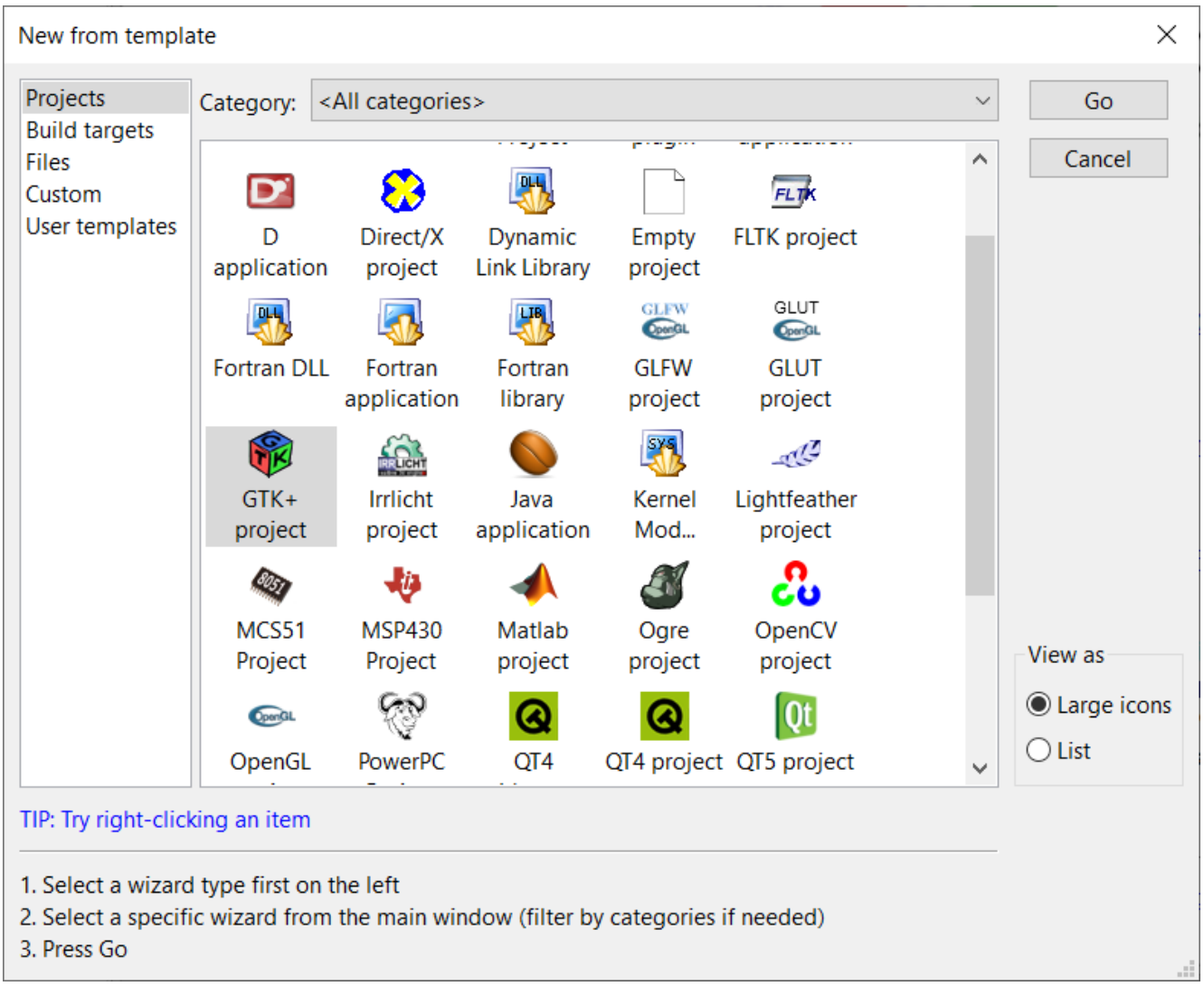

Рисунок 3.6 — Шаблони у Code::Blocks

Редактор Code::Blocks має безліч функцій, що спрощують та прискорюють процес написання коду, щоб створити всі необхідні умови для комфортної роботи програміста. Ось деякі з них:

— виділення синтаксису;

— автодоповнення коду;

— браузер класів;

— згортання ділянок коду;

— розумний відступ;

— швидке перемикання між .c та .h файлами;

— користувацькі комбінації клавіш.

Серед переваг Code::Blocks також можна виділити підтримку великої

кількості компіляторів і можливість легко перемикатися між ними залежно від ситуації; здатність реалізовувати паралельне складання, задіявши при цьому кілька ядер процесора; наявність розвинених засобів підтримки проектів, включаючи workspace для об'єднання кількох проектів в одному просторі; оснащення механізмом імпорту проектів з інших середовищ розробки та експорту файлів у різноманітні формати.

Беручи до уваги всі перераховані вище переваги інтегрованого середовища розробки Code::Blocks, було прийнято рішення про використання її як основного інструменту для написання коду програми розпізнавання напрямної лінії мобільним роботом за допомогою комп'ютерного зору.

Таким чином, у цьому було зроблено вибір мови програмування, бібліотеки комп'ютерного зору та кросплатформового середовища розробки для реалізації програмної частини.

3.2 Алгоритм розпізнавання напрямної лінії

В основі алгоритму реєстрації та розпізнавання напряму руху за допомогою комп’ютерного зору лежить аналіз відеоряду, що надходить з веб-камери на одноплатний комп'ютер і представляє собою набір зображень, які називають кадрами. Для отримання інформації про положення робота щодо лінії, яка надалі буде застосована для корекції траєкторії його руху, використовуються різноманітні методи обробки зображень. Вони дозволяють накладати на кадри фільтри, виділяти ними зони інтересу, проводити геометричні перетворення, маніпулювати пікселями тощо. Саме комбінація цих алгоритмів та функцій дозволяє мобільному роботу «бачити» напрямну лінію та слідувати їй.

3.2.1 Колірна сегментація кадру

Сегментація — це поділ цифрового зображення на кілька сегментів з метою спрощення процесу його аналізу. Для реалізації колірної сегментації кадрів, що надходять із веб-камери на одноплатний комп'ютер, скористаємося інструментами бібліотеки OpenCV для маніпуляції просторами кольорів.

Колірний простір — це модель уявлення кольору, за якою будь-який відтінок може бути представлений у вигляді точки, що має певні колірні координати. Найпопулярнішим і широко використовується колірним простором є модель RGB, в основі якої лежить використання комбінацій трьох основних кольорів — червоного (Red), зеленого (Green) та синього (Blue). Ця модель є адитивною, тобто відтінки виходять за допомогою додавання основних кольорів один з одним. Наприклад, при змішуванні синього та червоного виходить пурпуровий, зеленого та червоного — жовтий, зеленого та синього — блакитний. Комбінація всіх трьох основних випромінювань дасть білий колір, відсутність випромінювання — чорний. Для зручності та наочності модель RGB часто подають у вигляді колірного куба (рисунок 3.7).

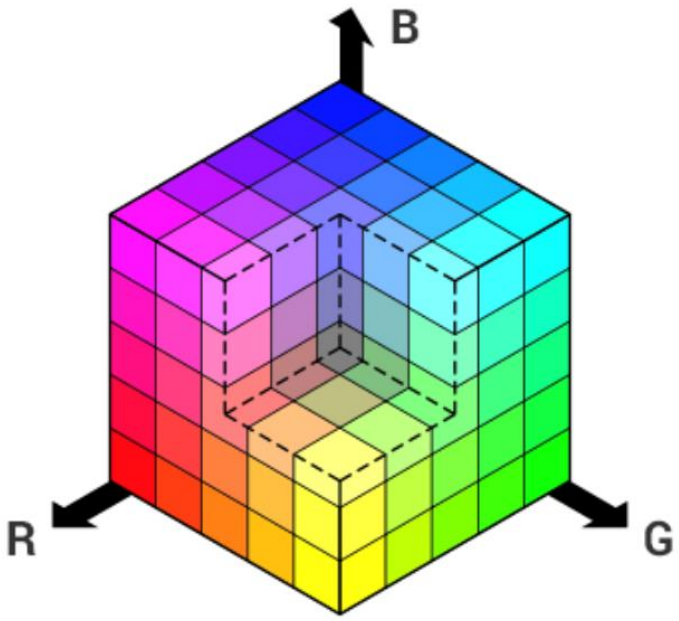

Рисунок 3.7 — RGB модель у вигляді колірного куба

Зображення в моделі RGB можна інтерпретувати як двовимірні масиви або матриці з трьома шарами (каналами), що представляють червоний, зелений і синій кольори. Кожен елемент двовимірного масиву кожного каналу зберігає інтенсивність одного з кольорів певного пікселя зображення. Комбінація відповідних пікселів кожного каналу дає унікальний загальний колір пікселя (рисунок 3.8) [25].

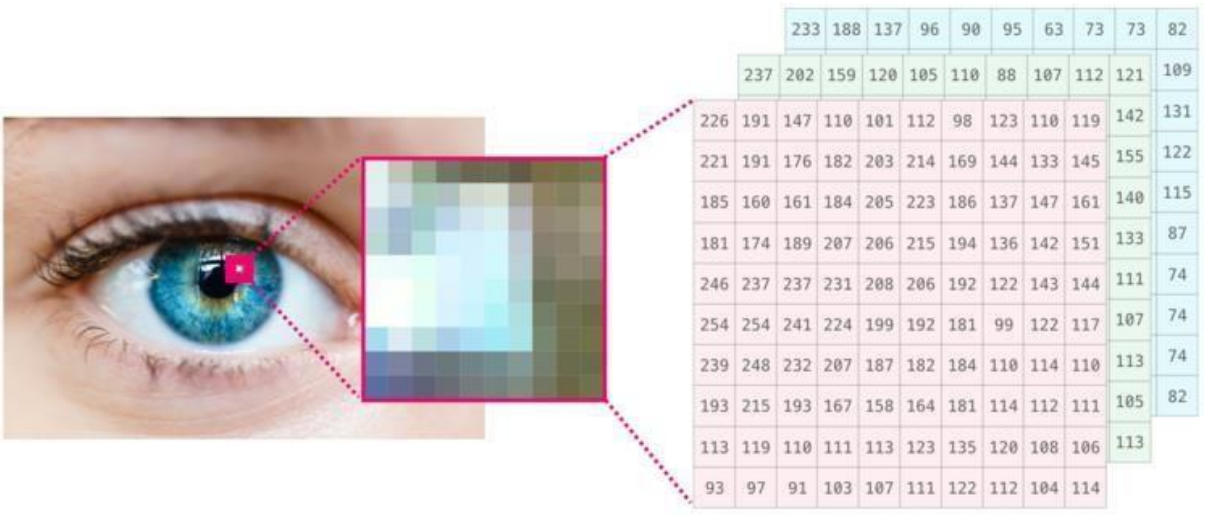

Рисунок 3.8 — Зображення у вигляді матриці

У комп'ютері кожна з координат координат пікселя записується з допомогою одного байта, тобто. в діапазоні чисел від 0 до 255 включно, де 0 — мінімальна, а 255 — максимальна інтенсивність. Так, наприклад, червоний колір буде закодований у вигляді (255, 0, 0), жовтий — (255, 255, 0), помаранчевий — (255, 128, 0), тощо.

Іншим, що часто використовується, колірним простором є модель HSV, координатами в якій виступають відтінок (Hue), насиченість (Saturation) та яскравість (Value). Для зручності та наочності цю модель часто представляють у вигляді конуса кольору (рисунок 3.9).

Відтінки у просторі HSV представлені кутами на колі, що лежать у діапазоні від 0º до 360º. Червоний, зелений і синій кольори, наприклад, ділять основу конуса на три рівні частини, утворюючи кути в 0, 120 і 240 відповідно. Насиченість варіюється в межах від 0 до 100 або від 0 до 1 і відповідає діаметру кола. Чим більший цей параметр, тим «чистіший» колір, тому його ще іноді називають чистотою кольору. Яскравість відповідає висоті конуса і також визначається діапазонами від 0 до 100 або від 0 до 1, де 0 представляє чорний колір, а 1 або 100 — білий.

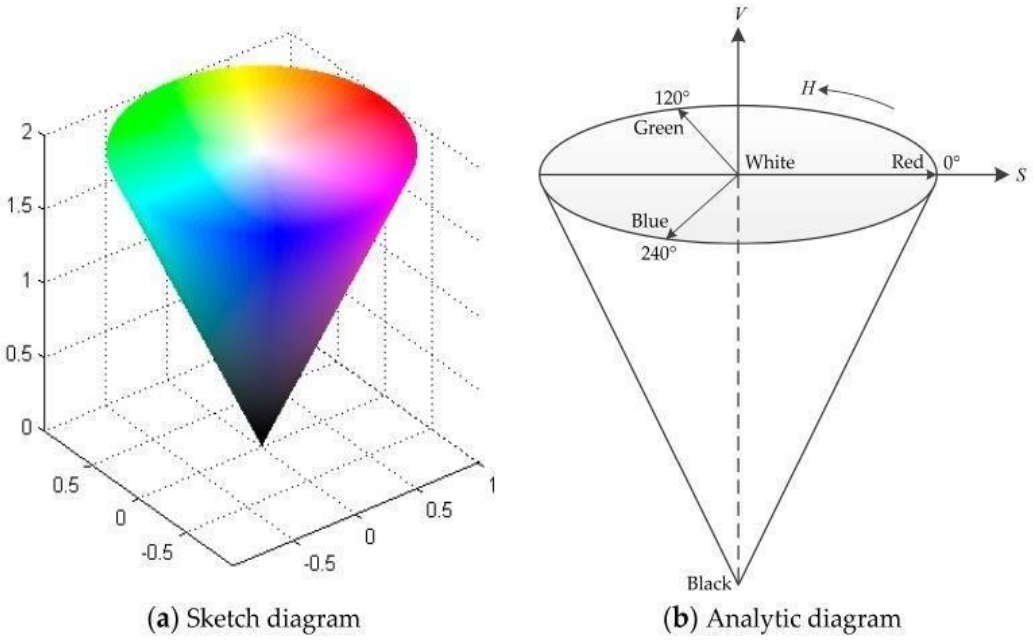

Рисунок 3.9 — HSV модель у вигляді конуса кольору

Завдяки тому, що всі можливі відтінки є набором фіксованих значень, а параметри, що відповідають за насиченість і яскравість, незалежні один від одного, колірний простір HSV краще підходить для виявлення яскраво-синьої напрямної лінії ніж RGB, оскільки дозволяє гнучкіше налаштовувати параметри сегментації зображення і менше схильне до впливу освітлення [25].

У бібліотеці OpenCV за сегментацію зображення за кольором відповідає функція inRange(src, lowerb, upperb, dst), де:

— src — вхідний кадр, який накладається фільтр;

— lowerb — нижня межа колірного діапазону;

— upperb — верхня межа колірного діапазону;

— dst — вихідний кадр, із застосованим фільтром.

Ця функція перетворює надходить їй на вхід кольорове зображення на бінарне, роблячи всі пікселі, які у вказаний колірний діапазон, білими, інші — чорними. Демонстрація роботи функції представлена рисунку 3.10 [25].

Рисунок 3.10 — Демонстрація роботи функції inRange()

3.2.2 Виділення контурів

Ще одним важливим інструментом бібліотеки комп'ютерного зору OpenCV є виділення контурів на зображенні. Даний алгоритм дозволяє аналізувати положення об'єктів у кадрі і може бути використаний для виявлення мобільним роботом напрямної лінії з її зовнішніх кордонів.

У бібліотеці OpenCV за знаходження контурів об'єкта на зображенні відповідає функція findContours(image, contours, hierarchy, mode, method), де:

— image — зображення, що досліджується;

— contours — список усіх знайдених контурів, які представлені у вигляді векторів їх точок;

— hierarchy — структура, що встановлює батьківсько-дочірні відносини між контурами.

Наприклад, при виявленні двох контурів, один із яких розташований усередині іншого, зовнішній контур буде батьківським, а внутрішній — дочірнім. Ієрархія представлена у вигляді масиву, кожен елемент якого є ще одним масивом з чотирьох індексів (Next, Previous, First_Child, Parent), відповідний одному з контурів. Індекс Next позначає наступний контур на тому ж ієрархічному рівні для контуру, що розглядається. Індекс Previous позначає попередній контур на тому ж ієрархічному рівні для контуру, що розглядається. Індекс First_Child позначає перший дочірній контур для контуру, що розглядається. Індекс Parent означає батьківський контур для контуру, що розглядається. Якщо контур, на який

посилається якийсь із індексів, відсутній, то відповідний індекс повертає значення -1 [26].

Параметр mode містить один із чотирьох режимів вилучення знайдених контурів:

— CV_RETR_LIST — витягує всі контури без встановлення будь-яких ієрархічних відносин;

— CV_RETR_EXTERNAL — витягує лише зовнішні контури;

— CV_RETR_CCOMP — витягує всі контури та організує їх у дворівневу ієрархію, на верхньому рівні знаходяться зовнішні межі об'єктів, на нижньому рівні — межі отворів, якщо всередині отвору є інший контур, він все одно поміщається на верхній рівень;

— CV_RETR_TREE — витягує всі контури та групує їх у багаторівневу ієрархію.

Параметр method містить один із чотирьох методів апроксимації контурів:

— CV_CHAIN_APPROX_NONE — упаковка відсутня та всі контури зберігаються у вигляді безлічі відрізків;

— CV_CHAIN_APPROX_SIMPLE — склеює всі горизонтальні, вертикальні та діагональні сегменти, залишаючи тільки їх кінцеві точки;

— CV_CHAIN_APPROX_TC89_L1 — застосовує до контурів один із варіантів алгоритму апроксимації Те-Чіна;

— CV_CHAIN_APPROX_TC89_KCOS — також застосовує до контурів один із варіантів алгоритму апроксимації Те-Чіна.

Для візуалізації контурів на кадрі в бібліотеці OpenCV передбачена функція drawContours(image, contours, contourIdx, color, thickness), де:

— image — зображення, на якому будуть намальовані контури;

— contours — список контурів для візуалізації, які представлені у вигляді векторів їх точок;

— contourIdx — параметр, що вказує конкретний контур для відтворення. Якщо він дорівнює -1, то візуалізовані всі контури;

— color — колір контурів, що візуалізуються;

— thickness — товщина ліній контурів, що візуалізуються.

Приклад знаходження та візуалізації контурів на зображенні представлений на рисунку 3.11.

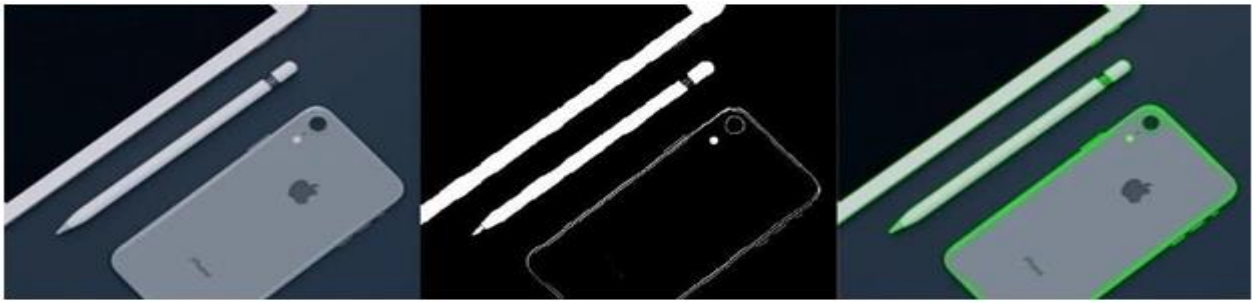

Рисунок 3.11 — Демонстрація роботи функцій findContours() та drawContours()

3.2.3 Моменти зображення

Моменти зображення — це певні середньозважені значення інтенсивності пікселів зображення. Вони визначаються такою формулою:

$$M_{ij} = \sum_x \sum_y x^i \, y^j I(x, y), \tag{3.1}$$

де x, y — координати пікселя;

i, j — потужність, на якій відповідний піксель взятий у сумі з іншими відображеними; I(x, y) — інтенсивність пікселя.

У формулі 3.1 момент M_{ij} визначається як сума по всім пікселям об'єкта значень пікселів у точці x, y, помножених на коефіцієнт $x^i y^j$ [27].

У бінарних зображеннях інтенсивність пікселя I(x, y) може приймати лише два значення: нуль для чорних сегментів та одиниця для білих. Таким чином, момент нульового порядку M_{00} такого зображення буде представляти площу білого об'єкта (суму всіх його пікселів) [27]. Обчислюється він за такою формулою:

$$M_{00} = \sum_x \sum_y I(x, y). \tag{3.2}$$

Моменти M_{10} і M_{01} першого порядку бінарного зображення є сумами x і y

складових всіх пікселів білого об'єкта і знаходяться за формулами (3.3) і (3.4) відповідно.

$$M_{10} = \sum_x \sum_y xI(x,y). \tag{3.3}$$

$$M_{01} = \sum_x \sum_y yI(x,y). \tag{3.4}$$

За допомогою моментів M_{00}, M_{10} і M_{01} можна знайти центроїд білого об'єкта в кадрі, яким буде шукана напрямна лінія. Його координати можуть бути обчислені за допомогою формул (3.5) та (3.6).

$$x = \frac{M_{10}}{M_{00}}, \tag{3.5}$$

$$y = \frac{M_{01}}{M_{00}}, \tag{3.6}$$

У бібліотеці OpenCV за знаходження моментів зображення відповідає функція moments(array, binaryImage), де:

— array — lосліджуване зображення;

— binaryImage — параметр бінарності зображення типу bool. Якщо він приймає значення true, всі ненульові пікселі кадру розглядаються як одиниці.

3.3 Опис та реалізація запропонованих рішень

Досліджуючи представлений алгоритм розпізнавання напрямної лінії, заснованого на принципі пошуку об'єкта кольором через колірний простір HSV з подальшим обчисленням моментів бінарного зображення, запропоновано інші методи реєстрації та розпізнавання напряму руху за допомогою комп’ютерного зору.

3.3.1 Рішення 1

Головною ідеєю запропонованого методу є розпізнавання двох напрямних ліній шляхом перетворення кольорового кадру з веб-камери у відтінки сірого. Досягти цього дозволяє функція cvtColor(), яка приймає як код перетворення запис COLOR_BGR2GRAY.

Розглянемо основні етапи роботи алгоритму.

Далі до кадру, що вийшов, застосовується розмиття по Гауссу, яке покликане згладити зображення і зменшити шуми. Принцип дії даного методу полягає в тому, що він перераховує значення яскравості кожного пікселя кадру, що перетворюється, і встановлює його рівним середньозваженому значенню його околиці, причому сусідні пікселі надають все менший вплив на вихідний піксель у міру збільшення їх відстані до нього. У бібліотеці OpenCV за реалізацію розмиття по Гауссу відповідає функція GaussianBlur (src, dst, ksize, sigma), де:

— src — вхідний кадр, який накладається фільтр;

— dst — вихідний кадр, із застосованим фільтром;

— ksize — розмір ядра, області навколо пікселя, яка враховується при обчисленні нового значення;

— sigma — середньоквадратичне відхилення нормального розподілу функції Гауса, збільшення якого призводить до більш сильного розмиття зображення і згладжування гострих країв.

Наступним кроком є виділення меж об'єктів на зображенні за допомогою оператора Кенні. Робота цього методу складається з 5 етапів:

— згладжування зображення — перш ніж кадр буде оброблений алгоритмом виявлення кордонів, до нього автоматично застосовується розмиття Гауссом для зменшення шумів, у цій роботі цей етап був раніше пройдений за допомогою окремого виклику функції GaussianBlur();

— пошук градієнтів зображення — градієнт показує зміну інтенсивності або яскравості пікселів кадру у різних напрямках, для його знаходження використовується оператор Собеля, результатом застосування якого у кожній точці зображення є вектор градієнта у цій точці;

— пригнічення не-максимумів, мається на увазі, що пригнічуються всі точки, які є локальними максимумами градієнта у напрямі;

— подвійна порогова фільтрація зображення, де застосовуються спеціальні порогові значення визначення того, є даний піксель кадру частиною межі об'єкта

чи ні, при цьому оператор Кенні використовує два пороги фільтрації, якщо значення пікселя вище верхньої межі він набуває максимального значення (кордон вважається достовірним), якщо нижче — піксель пригнічується, а точки зі значенням, що потрапляють у діапазон між кордонами, приймають фіксоване середнє значення (вони будуть уточнені на наступному етапі), таким чином, чим менше поріг, тим більше меж буде знайдено, але тим сприйнятливішим до шуму стане результат, виділяючи зайві об'єкти на зображенні, тоді високий поріг, навпаки, може проігнорувати деякі краї чи видати неповну межу як фрагментів;

— трасування області неоднозначності, а, якщо, простіше, то завдання зводиться до виділення груп пікселів, які отримали на попередньому етапі проміжне значення, і віднесення їх до кордону (якщо вони з'єднані з одним із встановлених кордонів) або їх придушення (інакше), піксель додається до групи, якщо він стикається з нею по одному з 8 напрямків.

У бібліотеці OpenCV за застосування оператора Кенні для знаходження меж об'єктів на зображенні відповідає функція Canny (image, edges, threshold1, threshold2), де:

— image — зображення, що досліджується;

— edges — бінарне зображення, на якому залишаться лише межі об'єктів;

— threshold1 — нижня межа фільтрації;

— threshold2 — верхня межа фільтрації.

Після цього на отриманому кадрі виділяється область інтересу, що містить лише покриття з напрямними лініями, а все інше відкидається.

Останнім кроком у процесі обробки зображення, що надходить із веб-камери, є використання перетворення Хафа для знаходження напрямних ліній. Даний алгоритм заснований на поданні прямої в полярних координатах і використовує наступне рівняння:

$$r = x\cos\theta + y\sin\theta, \tag{3.7}$$

де r — довжина нормалі до прямої, проведеної з початку координат;

θ — кут між цією нормаллю та віссю абсцис;

x та y — координати точки на прямій.

Будь-яка пряма, що проходить через довільну точку (x, y) у прямокутній системі координат, відповідає точка (r, θ) у полярній системі координат. Отже, безлічі прямих, що проходять через довільну точку (x, y) у прямокутній системі координат, відповідає набір точок (r, θ) у полярній системі координат, який, згідно з рівнянням 3.7, являє собою синусоїду, унікальну для даної точки (x, y) і однозначно її визначальну. Таким чином, всі можливі прямі, що проходять через усі точки зображення, можуть бути представлені у вигляді набору синусоїд у полярних координатах. Перетин синусоїд один з одним означає, що відповідні їм точки в прямокутних координатах знаходяться на одній прямій. Кожна така пряма записується в спеціальний двовимірний масив, званий акумулятором, рядки/стовпці якого відповідають показникам r і θ. Розмір даного масиву залежить від вибраного кроку для кута θ і максимально можливої довжини нормалі r обмеженою діагоналлю кадру. З цього всього випливає, що більше перетинів синусоїд, тим більше відповідних їм точок лежать на одних і тих же прямих і тим більше записів в акумуляторі належать цим прямим. Тільки при досягненні певної кількості записів пряма вважається виявленою.

У бібліотеці OpenCV за використання перетворення Хафа для знаходження прямих на зображенні відповідає функція HoughLinesP(image, lines, rho, theta, threshold, minLineLength, maxLineGap), де:

— image — зображення, що досліджується;

— lines — масив типу вектор для знайдених ліній, що зберігає координати кінцевих точок кожного виявленого сегмента лінії;

— rho — показник r, що визначає розмір акумулятора;

— theta — показник θ, визначальний розмір акумулятора;

— threshold — граничне значення кількості записів в акумуляторі, необхідне виявлення лінії;

— minLineLength — мінімальна довжина лінії. Сегменти лінії коротше цього значення відхиляються;

— maxLineGap — максимально допустимий проміжок міжсегментами однієї лінії для їхнього з'єднання.

Далі виявлені напрямні лінії використовуються як основа для побудови навколо них прямокутника, що обертається, орієнтація якого і формує керуючий вплив для коригування траєкторії руху.

Результати всіх перетворень та роботи алгоритму загалом представлені рисунку 3.12. Запропонований метод може бути застосований при позиціонуванні сучасних автомобілів на проїжджих ділянках або в складських приміщеннях з відповідною розміткою.

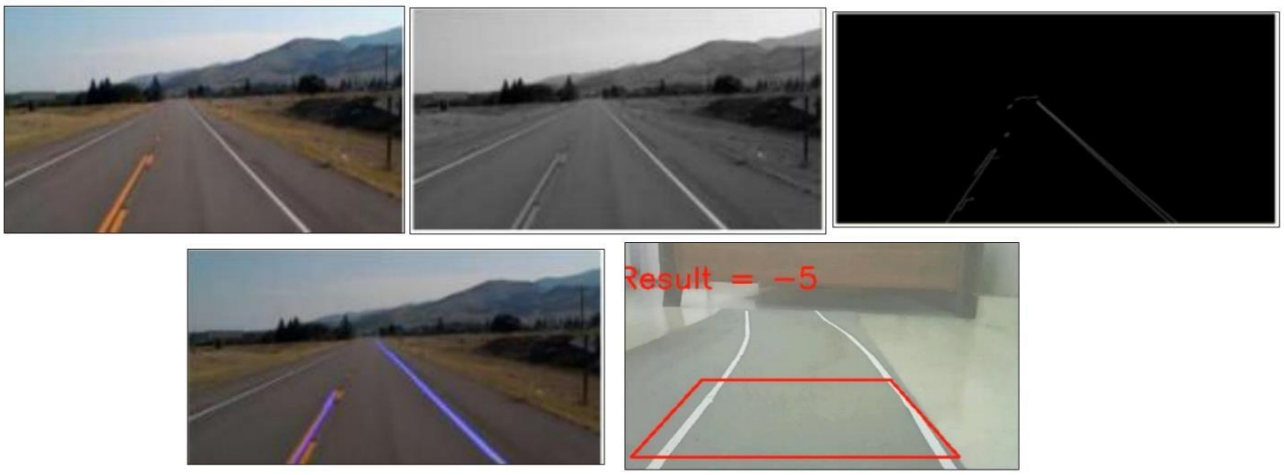

Рисунок 3.12 — Результати роботи алгоритму рішення 1

3.3.2 Рішення 2

Головною ідеєю цього методу, на відміну від вищезапропонованого, є використання спеціальних маркерів замість напрямних ліній.

Розглянемо основні етапи роботи даного алгоритму.

Для зручнішого аналізу кадрів насамперед необхідно програмно змінити перспективу зображень, що надходять із веб-камери. За реалізацію такого перетворення у бібліотеці OpenCV відповідають дві функції. Перша називається getPerspectiveTransform(src, dst), де:

— src — масив, що зберігає координати 4 вершин області на вихідному зображенні, перспективу якої потрібно змінити;

— dst — масив, що зберігає координати 4 вершин прямокутної області на вихідному зображенні, яку буде поміщений перетворений фрагмент.

Ця функція повертає матрицю перетворення, необхідну для подальшої роботи. За безпосередню зміну перспективи зображення відповідає функція warpPerspective(src, dst, M, dsize), де:

— src — вхідне зображення, область якого потрібно перетворити;

— dst — вихідне зображення, на якому буде розміщено змінений фрагмент;

— M — матриця перетворення, що є результатом роботи функції getPerspectiveTransform();

— dsize — розмір вихідного зображення.

Результат перетворення перспективи зображення рисунку 3.13.

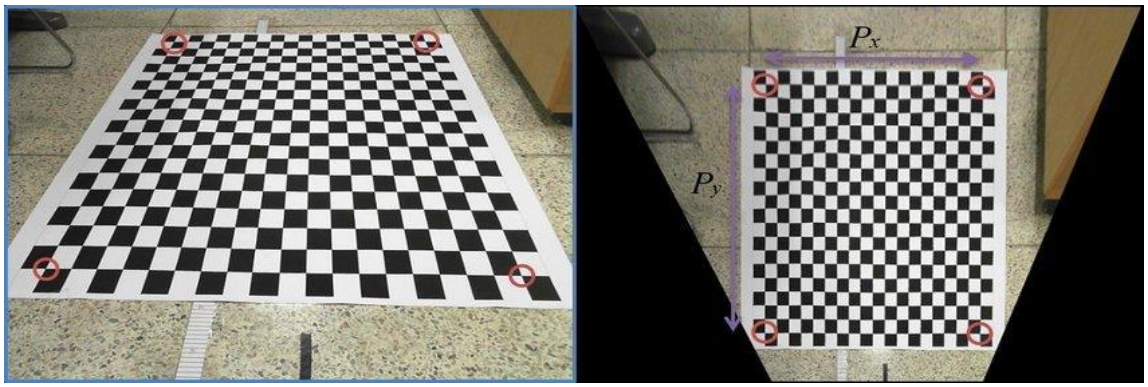

Рисунок 3.13 — Результат перетворення перспективи зображення

Далі отриманий кадр перетворюється на простір HSV і проходить колірну сегментацію виявлення напрямних маркерів. Після цього до бінарного зображення застосовується перетворення Хафа. Знайдені лінії використовуються визначення положення системи у просторі. Так, наприклад, з їхньою допомогою обчислюється орієнтація вказівних символів усередині маркерів, зокрема трикутників, а також відстань від AGV до найближчого маркера. Всі ці дані потім використовуються для формування керуючого впливу та коригування траєкторії руху. Основний принцип роботи алгоритму представлено рисунку 3.14.

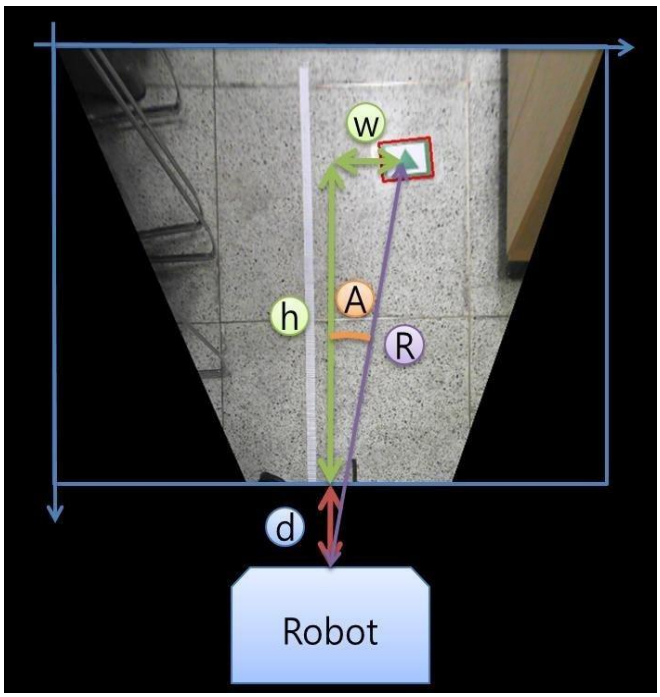

Рисунок 3.14 — Принцип роботи алгоритму рішення 2

3.4 Програмна реалізація рішень

Програма для система реєстрації та розпізнавання напряму руху базується на вище запропонованих методах та буде реалізовуватись для розпізнавання яскраво-синьої напрямної лінії за допомогою комп'ютерного зору із застосуванням принципі пошуку об'єкта в колірному простірі HSV з подальшим обчисленням положення системи в просторі та формуванням впливу, що управляє, для корекції траєкторії його руху. Лістинг програми представлений у Додатку Б.

Загальний лгоритм роботи програми складається з 22 послідовних етапів:

— підключення всіх необхідних роботи програми заголовних файлів і бібліотек;

— оголошення змінних;

— завдання колірного діапазону — визначення верхній і нижньої меж;

— ініціалізація камери;

— відкриття послідовного порту;

— перевірка успішності відкриття послідовного порту, у разі виникнення будь-якої помилки в консоль одноплатного комп'ютера Raspberry Pi виведеться повідомлення Failed open port і програма завершить свою роботу;

— захоплення зображення, що надходить із веб-камери;

— перевірка успішності захоплення кадру,у разі виникнення будь-якої помилки в консоль одноплатного комп'ютера Raspberry Pi виведеться повідомлення Camera error і програма завершить свою роботу;

— перетворення отриманого кадру з колірного простору RGB в колірну модель HSV;

— колірна сегментація за допомогою підібраних раніше меж і вилучення бінарного зображення з виділенням області заданого кольору, ідентифікація виявленого синього об'єкта як спрямовуючої лінії;

— знаходження моментів зображення;

— порівняння площі розпізнаної лінії із встановленим значенням,у разі перевищення порога програма продовжить свою роботу і перейде до подальших математичних перетворень, в іншому випадку в консоль одноплатного комп'ютера Raspberry Pi виведеться повідомлення "Line was not detected", відбудеться пропуск всіх обчислень та виведення знайденого кадру з даними попереднього циклу;

— визначення координат середини напрямної лінії;

— знаходження координат центру кадру;

— обчислення різниці x-складових координат середини лінії та центру кадру та запис отриманого значення змінну delta з метою її подальшого використання як керуючого впливу;

— візуалізація отриманої інформації про стан системи;

— запис значення змінної delta у вигляді окремих цифр у спеціальний масив-буфер для подальшого відправлення;

— відправка елементів, що зберігаються в буфері, через послідовний порт мікроконтролер Arduino;

— примусове обнулення масиву-буфера;

— виведення зображення з веб-камери на дисплей, підключений до одноплатного комп'ютера;

— перевірка натискання користувачем будь-якої клавіші на клавіатурі протягом 10 мс, що означало б завершення роботи програми, за відсутності дій із боку людини відбувається автоматичний перехід початку нового циклу — захоплення нового кадру;

— закриття послідовного порту та завершення програми за умови, що кнопка була натиснута.

Блок-схема загального алгоритму роботи програми представлена рисунку 3.15.

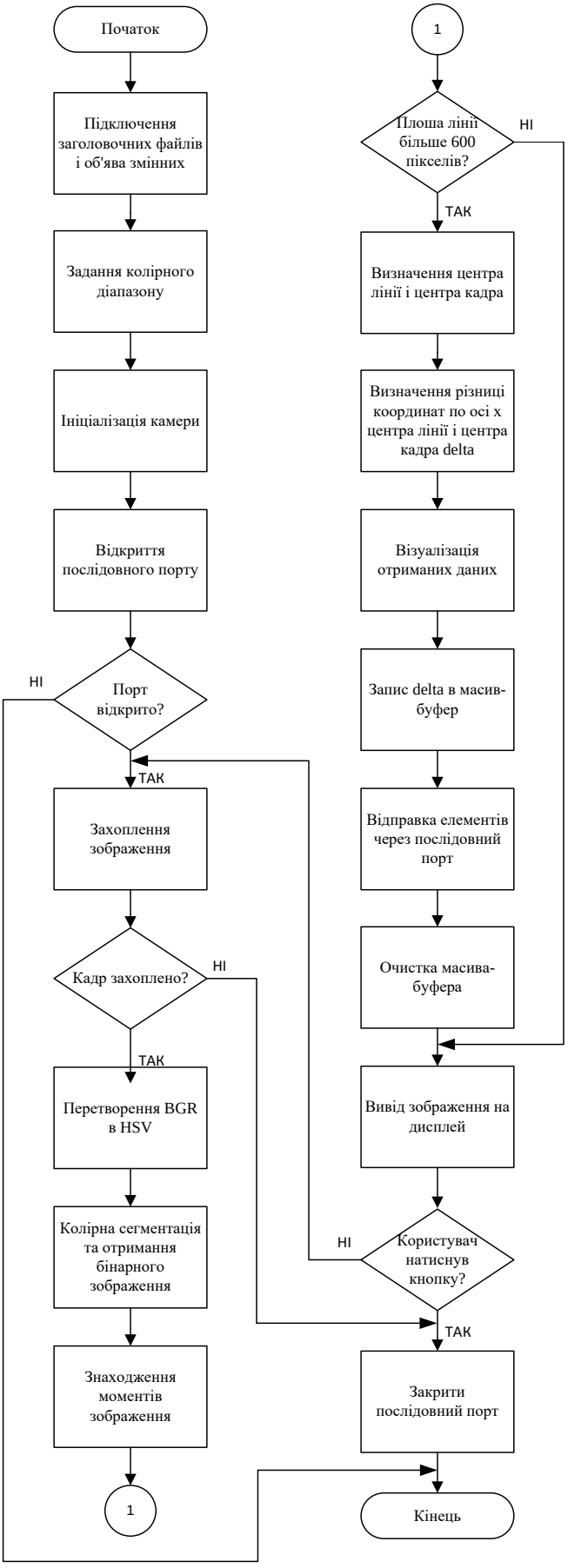

Рисунок 3.15 — Блок-схема загального алгоритму роботи програми

4 ДОСЛІДЖЕННЯ СПРОЕКТОВАНОЇ СИСТЕМИ

4.1 Докладний розбір коду програми

Проведемо детальний аналіз коду програми. Насамперед здійснюється підключення всіх необхідних для роботи заголовних файлів:

— opencv2/core.hpp містить класи та функції, які використовуються для роботи з базовими структурами даних бібліотеки OpenCV: матрицями, векторами, точками тощо, ці об'єкти є основними будівельними блоками до створення додатків комп'ютерного зору;

— opencv2/imgproc.hpp містить функції, які використовуються для обробки зображень, даний модуль дає доступ до інструментів, що дозволяють, наприклад, знаходити межі та контури об'єктів, налаштовувати яскравість та контрастність, змінювати колірні простори та багато іншого;

— opencv2/highgui.hpp містить функції, які використовуються для роботи з графічним інтерфейсом користувача, даний модуль відповідає за створення та відображення вікон, читання та запис зображень тощо;

— opencv2/videoio.hpp містить функції, які використовуються для захоплення відео з камери, а також читання та запису відеофайлів;

— iostream відповідає за введення користувачем інформації з клавіатури та виведення даних у консоль;

— vector дає можливість працювати з однойменною динамічною структурою даних, що забезпечує швидке додавання нових елементів, автоматично змінює свій розмір за потреби та гарантує відсутність витоків пам'яті;

— wiringPi дозволяє звертатися до GPIO інтерфейсу Raspberry Pi;

— wiringSerial відкриває доступ до інструментів для роботи з послідовним портом Raspberry Pi;

— unistd.h містить функцію delay(), яка дозволяє призупиняти виконання програми на певний час, вказаний усередині дужок та вимірюваний у мілісекундах.

Наступним кроком є оголошення всередині основної функції main() всіх змінних та структур, які будуть використовуватись у програмі надалі.

Об'єкти типу Mat є n-мірними масивами або матрицями, що використовуються для зберігання зображень. Так, frame міститиме вихідний кадр із веб-камери у колірному просторі RGB, hsv — той самий кадр, але вже у форматі HSV, а mask — бінарне зображення після сегментації кадру hsv за кольором та виділення напрямної лінії.

Об'єкт m типу Moments зберігатиме моменти зображення двійкового кадру mask.

Нижче знаходяться цілі змінні типу int, які братимуть участь у математичних перетвореннях. Width буде містити в собі ширину всіх кадрів у пікселях, x і y — відповідні координати середини напрямної лінії, center — положення центру кадру на осі абсцис, а delta — вже згадану раніше різницю x-складових координат середини лінії та центру кадру, що є керуючим впливом для мобільного робота.

Далі йдуть нижня (lowerBlue) і верхня (upperBlue) межі діапазону відтінків синього для сегментації зображення в колірному просторі HSV, що є двома масивами типу vector, що містять по 3 елементи типу int:

vector<int>lowerBlue = {75, 60, 50};

vector<int>upperBlue = {130, 255, 255}.

Варто зазначити, що у бібліотеці OpenCV значення кольору кодуються числами від 0 до 180, насиченості — від 0 до 255, яскравості — також від 0 до 255.

Масив-буфер buffer типу char, здатний зберігати у собі 5 елементів, використовуватиметься у процесах візуалізації керуючого впливу delta та її відправлення через послідовний порт на мікроконтролер Arduino.

Далі відбувається ініціалізація камери за допомогою екземпляра cap класу VideoCapture бібліотеки OpenCV, призначеного для захоплення відео та читання відеофайлів. До цього об'єкта застосовується метод open(), що приймає індекс пристрою, з якого необхідно здійснити захоплення відеопотоку. Оскільки система, що розробляється, передбачає наявність лише однієї камери, то аргументом для методу open() буде число 0. Запис, що відповідає за це, в коді виглядає наступним чином: VideoCapture cap; cap.open(0).

Наступним кроком є ініціалізація послідовного порту, до якого підключений

мікроконтролер Arduino, за допомогою команди serialOpen(), яка приймає в якості аргументів номер порту та швидкість передачі даних. Контроль за станом порту буде здійснюватися за допомогою змінної serialPort типу int:

int serialPort = serialOpen("/dev/ttyUSB0", 9600).

Далі йде перевірка успішності відкриття послідовного порту за допомогою конструкції if. Якщо змінна serialPort містить значення -1, то це означає, що зв'язок між Raspberry Pi та Arduino не був встановлений, внаслідок чого в консоль одноплатного комп'ютера виведеться повідомлення про помилку "Failed open port", а програма завершить свою роботу.

Далі штучно створюється невелика затримка тривалістю 1 секунду за допомогою функції delay() для забезпечення синхронізації пристроїв та стабільності з'єднання.

Після цього програма потрапляє в нескінченний цикл, в якому здійснюється пошук напрямної лінії та формування керуючого впливу.

Цей процес починається із захоплення зображення, що надходить до одноплатного комп'ютера з веб-камери, за допомогою згаданого раніше об'єкта cap і запису його в матрицю frame: cap >> frame.

Далі йде перевірка успішності захоплення кадру за допомогою конструкції if. Якщо метод empty, застосований до об'єкта frame, повертає логічне значення true, це означає, що під час захоплення відеопотоку сталася помилка і матриця для зберігання зображення залишилася порожньою, внаслідок чого консоль одноплатного комп'ютера Raspberry Pi виведеться повідомлення про помилку «Camera error», а програма завершить свою роботу.

Приклад кадру, отриманого з веб-камери, показано на рисунку 4.1.

Рисунок 4.1 — Кадр із веб-камери

Далі відбувається обчислення ширини захопленого кадру в пікселях за допомогою методу size().width, що застосовується до матриці frame, і запис отриманого значення змінну width. Наступним кроком є перетворення отриманого зображення на колірний простір HSV за допомогою функції бібліотеки OpenCV cvtColor(src, dst. code), де:

— src — вхідний кадр, який потрібно перетворити;

— dst — вихідний перетворений кадр;

— code — код необхідного перетворення.

Запис, що відповідає за це, в коді має такий вигляд: cvtColor(frame, hsv, COLOR_BGR2HSV).

В даному випадку на місці параметра code знаходиться запис COLOR_BGR2HSV, що сигналізує програмі про те, що необхідно провести перетворення зображення, що зберігається в матриці frame, з простору простору BGR в HSV і записати його в матрицю під назвою hsv. Варто зазначити, що в бібліотеці OpenCV як стандартна колірна модель використовується простір BGR замість більш поширеного на сьогоднішній день RGB. Причиною цього стала популярність даного колірного формату серед виробників камер та програмного

забезпечення у минулому, коли бібліотека ще тільки розроблялася.

Приклад кадру колірної моделі HSV представлений на рисунку 4.2.

Рисунок 4.2 — Кадр у колірному просторі HSV

Далі переходимо до колірної сегментації кадру за допомогою функції inRange(). Даний алгоритм перевіряє попадання пікселів вхідного зображення hsv в діапазон синього кольору, заданий за допомогою раніше визначених меж lowerBlue та upperBlue. Якщо піксель перебуває у межах, він приймає значення 1, тобто забарвлюється у білий колір. В іншому випадку пікселю надається значення 0 і він стає чорним. Отримане бінарне зображення записується у матрицю mask. Запис, що відповідає за це, в коді виглядає так:

вRange(hsv, lowerBlue, upperBlue, mask).

Приклад кадру після колірної сегментації представлений рисунку 4.3.

Рисунок 4.3 — Кадр після колірної сегментації

На цьому етапі припускаємо, що єдиним яскраво-синім об'єктом у зору веб-камери є напрямна лінія. З цього випливає, що всі білі області двійкового кадру, що вийшов, інтерпретуватимуться як напрямна лінія.

Нижче відбувається обчислення моментів бінарного зображення mask та запис їх у спеціальний об'єкт m типу Moments: m = moments(mask, 1).

Далі йде порівняння площі розпізнаної лінії з встановленим значенням за допомогою моменту першого порядку 00. При перевищенні порога програма продовжить свою роботу та перейде до подальших математичних операцій. В іншому випадку в консоль Raspberry Pi виведеться повідомлення «Line was not detected» і почнеться повторний пошук — комп'ютер пропустить поточну ітерацію нескінченного циклу і перейде до наступної. Ця перевірка дозволяє системі у разі втрати напрямної лінії на повороті ігнорувати незначні об'єкти синього кольору, які можуть потрапити до поля зору її камери, і продовжувати рухатися згідно з керуючими значеннями, обчисленими раніше, коли лінія ще була присутня в кадрі. Цей прийом значно підвищує шанс для транспортного засобу самостійно повернутись на «трасу».

При виконанні описаного вище умови програма переходить до знаходження

координат x і y середини напрямної лінії за допомогою моментів другого порядку 10 і 01 за формулами (3.5) і (3.6). Далі обчислюється положення центру кадру на осі абсцис, що дорівнює половині ширини зображення width, що записується в змінну center. Після цього відбувається розрахунок керуючого впливу delta, що представляє собою різницю x-складових координат середини лінії та центру кадру.

Наступним кроком є візуалізація отриманих даних на дисплеї, яка допоможе користувачеві легко відстежити те, як система розпізнає напрямну лінію, і оцінити, наскільки коректними є обчислені мікрокомп'ютером значення впливу керуючого delta при різних положеннях системи, для подальшого коригування коду.

Насамперед зобразимо центри лінії та кадру у вигляді кіл червоного та зеленого кольору відповідно. Для цього скористаємося функцією для малювання кіл бібліотеки OpenCV circle(img, center, radius, color, thickness), де:

— img — зображення, на якому буде намальовано коло;

— center — координати центру кола;

— radius — радіус кола;

— color — колір кола;

— thickness — товщина контуру кола.

При негативному значенні даного параметра буде намальовано суцільне коло. Запис, що відповідає за це, у коді виглядає наступним чином:

```
circle(frame, Point(x, y), 5, Scalar(0, 0, 255), -1);
circle(frame, Point(center, y), 5, Scalar(0, 255, 0), -1);
```

Далі зобразимо жовтий відрізок, що з'єднує дані кружки, для візуальної оцінки відстані між серединою лінії та центром кадру. Для цього скористаємось функцією бібліотеки OpenCV line(img, pt1, pt2, color, thickness), де:

— img — зображення, на якому буде намальована лінія;

— pt1 — координати початку відрізка;

— pt1 — координати кінця відрізка;

— color — колір лінії;

— thickness — товщина лінії.

Запис, що відповідає за це, у коді виглядає наступним чином: line(frame,

Point(center, y), Point(x, y), Scalar(0, 255, 255), 2).

Останнім етапом візуалізації є нанесення знайденого значення керуючого на зображення як тексту. Для цього скористаємося раніше створеним масивом-буфером buffer і занесемо до нього число, що зберігається в змінній delta, у вигляді окремих символів типу char за допомогою команди sprintf():

sprintf(buffer, %d, delta).

Потім звернемося до функції бібліотеки OpenCV putText(img, text, org, fontFace, fontScale, color, thickness), де:

— img — зображення, на яке буде виведено напис;

— text — текст, який потрібно вивести;

— org — координати, в яких буде виведено перший символ напису;

— fontFace — тип шрифту;

— fontScale — коефіцієнт масштабування, який множиться на базовий розмір конкретного шрифту;

— color — колір тексту;

— thickness — товщина символів.

Запис, що відповідає за це, в коді виглядає так:

putText(frame, buffer, Point(width/2, y - 20),FONT_HERSHEY_SIMPLEX, 1, Scalar(0, 255, 255), 3);

Результат перерахованих вище дій представлений на рисунку 4.4.

Наступним кроком є відправлення керуючого впливу на мікроконтролер Arduino. Для цього використовується команда serialPrintf(), яка передає через послідовний порт набір символів з масиву-буфера buffer, що зберігає в собі значення, що нас цікавить: serialPrintf(serialPort, "%s\n", buffer);

Після цього відбувається примусове обнулення всіх елементів буфера щоб уникнути будь-яких помилок при подальших обчисленнях за допомогою функції memset(), що приймає як аргументи покажчик на масив, значення, до якого будуть прирівняні його елементи, а також кількість цих елементів:

memset(buffer, 0, sizeof buffer).

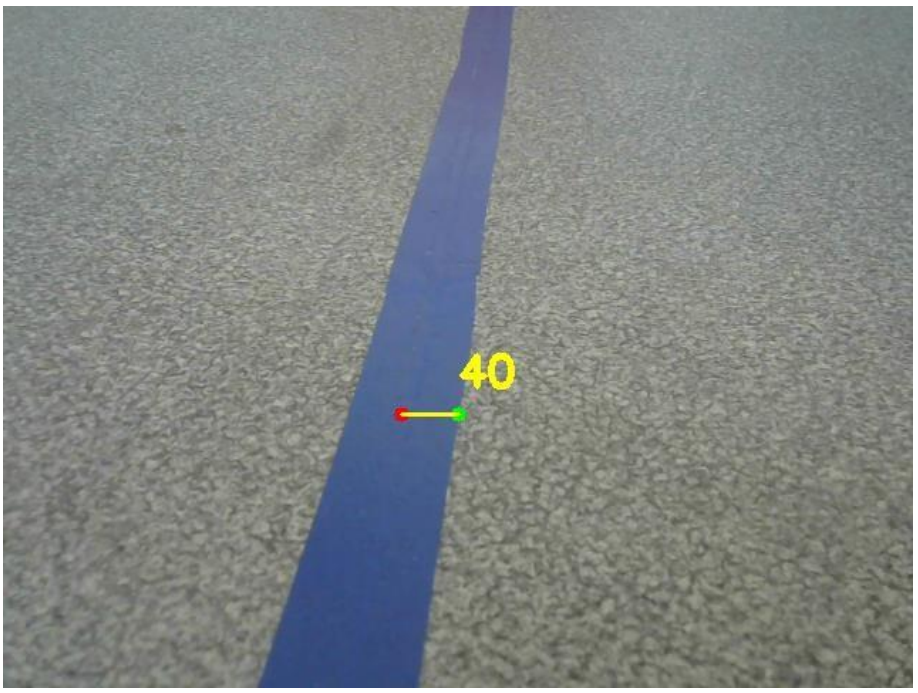

Рисунок 4.4 — Кадр із візуалізацією даних

Далі йдуть конструкції, які відповідають за виведення зображення на екран.

Вікно, через яке відбувається трансляція кадрів із веб-камери, створюється за допомогою функції бібліотеки OpenCV namedWindow(winname, flag), де:

— winname — назва вікна;

— flag — тип відображення вікна.

Наприклад, WINDOW_NORMAL дозволяє користувачеві змінювати розміри вікна, тоді як WINDOW_AUTOSIZE автоматично підганяє ширину і висоту вікна під зображення, що відображається без можливості регулювання.

Запис, що відповідає за це, в коді виглядає так: namedWindow("Window", WINDOW_AUTOSIZE).

Для кращого позиціонування вікна на екрані дисплея використовується функція бібліотеки OpenCV moveWindow(winname, x, y), де:

— winname — назва вікна, яке буде пересунуто;

— x — координата точки по осі абсцис, в яку буде переміщений лівий верхній кут вікна, що пересувається;

— y — координата точки по осі ординат, в яку буде переміщений лівий верхній кут вікна, що пересувається.

Запис, що відповідає за це, в коді виглядає так: moveWindow ("Window", 70, 70);

За безпосередню демонстрацію зображення у вікні відповідає функція бібліотеки OpenCV imshow(winname, mat), де:

— winname — назва вікна, у якому показуватиметься кадр;

— mat — зображення, що демонструється.

Запис, що відповідає за це, в коді має такий вигляд: imshow("Window", frame);

Завершує даний блок коду умова, за якої система протягом 10 мс чекає натискання користувачем будь-якої клавіші на клавіатурі для виходу з нескінченного циклу та завершення своєї роботи. Якщо дій зі сторони людини після цього часу не буде зареєстровано, програма перейде до наступної ітерації. Реалізація цієї перевірки заснована на функції бібліотеки OpenCV waitKey(delay), яка приймає як аргумент кількість мілісекунд, протягом яких система чекатиме натискання будь-якої клавіші на клавіатурі. Крім цього ця функція також повертає код кнопки, що використовується. Таким чином, умова if(waitKey(10) >= 0) спрацьовує при натисканні будь-якої клавіші, а if(waitKey(10) == 27) лише при взаємодії з esc.

При виході з нескінченного циклу відбувається руйнування вікна, за допомогою якого здійснювалася трансляція кадрів з веб-камери, за допомогою функції бібліотеки OpenCV destroyWindow(), яка приймає як аргумент ім'я даного вікна, а також закриття послідовного порту за допомогою команди serialClose(), аргументом до Якою є змінна serialPort, що відповідає за зв'язок між Raspberry Pi та Arduino.

Після застосування оператора return 0 робота програми завершується.

4.2 Можливі варіанти модернізації програми

4.2.1 Модифікація №1

Одним із шляхів покращення розробленої програми є додавання додаткових керуючих впливів, що відповідають за детектування відхилень системи від середини напрямної лінії у різних частинах кадру. Лістинг модифікованої програми, що реалізує цей підхід, представлений у Додатку В, а результат її роботи на рисунку 4.5.

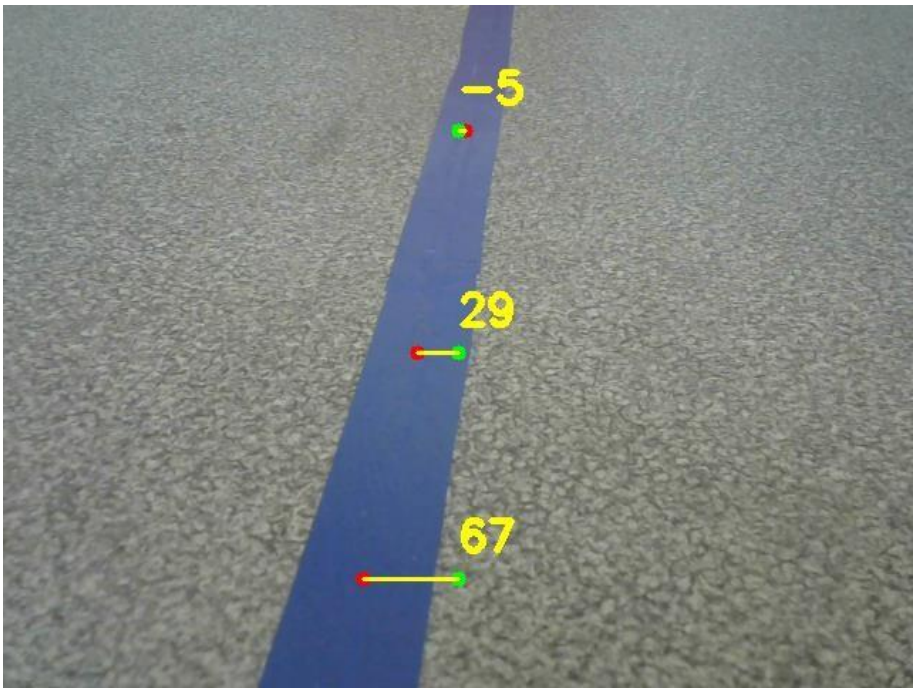

Рисунок 4.5 — Результат роботи програми модифікації №1

Алгоритм роботи даної модифікації ідентичний алгоритму роботи основної програми за винятком того, що тепер пункти 9-16 виконуються по одному разу для кожної з трьох частин кадру: верхньої, середньої і нижньої.

Подібний розділ поля зору камери реалізовано за допомогою методу для обрізання зображень Range(). Його запис відбувається так:

img(Range(start_row, end_row), Range(start_col, end_col)).

Ця функція має кілька параметрів:

— img — зображення, яке необхідно обрізати;

— start_row — y-координата початку ділянки, що зберігається;

— end_row — y-координата кінця ділянки, що зберігається;

— start_col — x-координата початку ділянки, що зберігається;

— end_col — x-координата кінця ділянки, що зберігається.

Головною перевагою даного підходу є отримання більшої кількості даних про положення системи щодо напрямної лінії, що дозволяє точніше налаштувати ПІД-регулятор транспортного засобу і, як наслідок, домогтися збільшення плавності його руху та покращення проходження ним поворотів.

Недоліком даної модифікації є значне збільшення кількості математичних перетворень, що відбуваються в процесі виконання програми, що збільшує споживання обчислювальних ресурсів мікрокомп'ютера та негативно позначається на його швидкодії.

4.2.2 Модифікація №2

Іншим можливим способом покращення розробленої програми може бути використання контурів для знаходження середини напрямної лінії замість моментів зображення.

Основним недоліком пошуку центроїду об'єкта за допомогою моментів зображення є негативний вплив сторонніх об'єктів у кадрі того ж кольору, що шукана лінія. Причина цього полягає в тому, що елементи M_{00}, M_{10} та M_{01} обчислюються з урахуванням абсолютно всіх білих пікселів бінарного зображення незалежно від їх положення та приналежності до того чи іншого предмета. З цього випливає, що поява в полі зору камери великого об'єкта яскраво-синього кольору здатна значно вплинути на процес обчислення впливу, що управляє, і змінити траєкторію руху системи.

Одним з можливих рішень цієї проблеми є знаходження контурів всіх синіх об'єктів у кадрі, визначення найбільшого з них та інтерпретація його як спрямовуючої лінії. Лістинг модифікованої програми, що реалізує згаданий вище підхід, представлений у Додатку Г. Блок-схема алгоритму роботи цієї модифікації представлена рисунку 4.6.

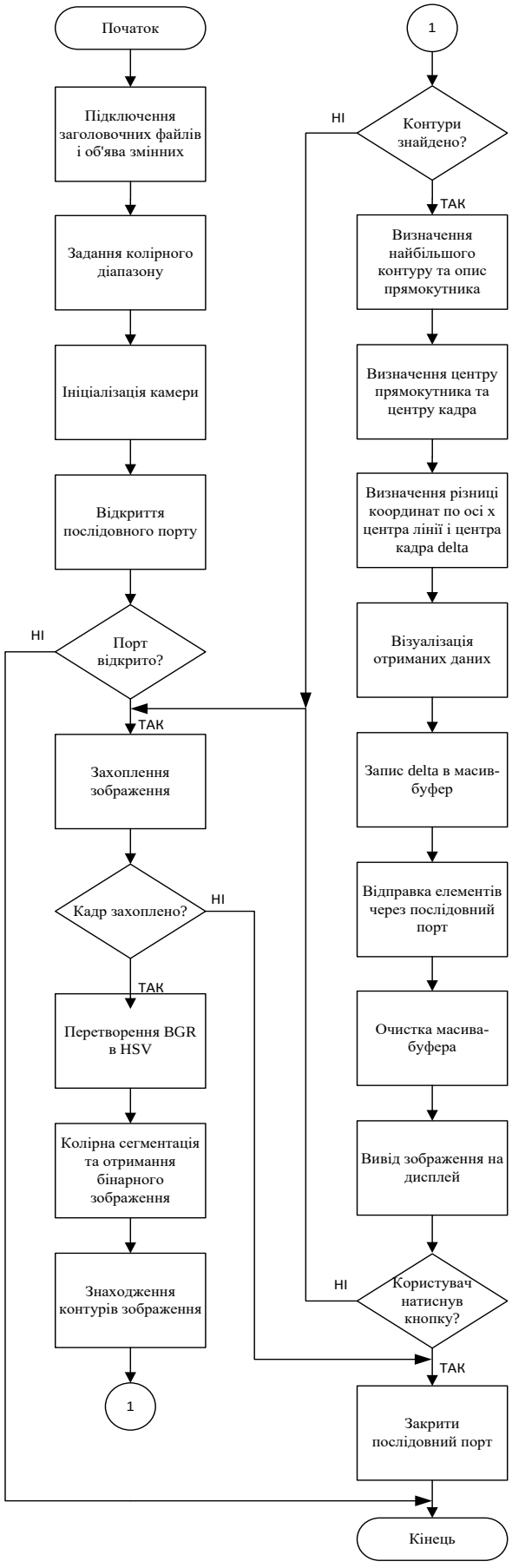

Рисунок 4.6 — Блок-схема алгоритму роботи програми модифікації №2

Проведемо аналіз окремих ключових функцій модифікованої програми. Знаходження кордонів всіх синіх об'єктів у кадрі здійснюється з допомогою функції бібліотеки OpenCV findContours(). Запис, що відповідає за це, в коді виглядає так:

findContours(mask, contours, hierarchy, RETR_TREE, CHAIN_APPROX_NONE).

У даному випадку функція приймає бінарне зображення mask, що пройшло колірну сегментацію, записує всі знайдені контури в масив під назвою contours типу vector<vector<Point>> у вигляді безлічі відрізків, а також організує їх багаторівневу ієрархію, що зберігається в масиві hierarchy типу vector<Vec4 >.

Далі йде перевірка успішності виявлення меж об'єктів за допомогою конструкції if. Якщо метод empty, застосований до об'єкта contours, повертає логічне значення true, це означає, що під час обробки зображення не було знайдено жодного контуру і масив для їх зберігання залишився порожнім, внаслідок чого в консоль Raspberry Pi виведеться повідомлення про помилку «Line was not detected» і розпочнеться повторний пошук — одноплатний комп'ютер пропустить поточну ітерацію нескінченного циклу та перейде до наступної.

Далі в циклі for відбувається перебір всіх виявлених контурів, обчислюються їх площі за допомогою функції contourArea(), що приймає як аргумент елементи масиву contours, які потім порівнюються із заздалегідь встановленим значенням. При перевищенні порога поточний контур візуалізується за допомогою функції drawContours(), а навколо нього описується прямокутник, що обертається, мінімально можливих розмірів за допомогою об'єкта rect типу RotatedRect і функції бібліотеки OpenCV minAreaRect(). Дане граничне значення підбирається в такий спосіб, щоб виявляти межі найбільшого синього об'єкта у кадрі, тобто напрямної лінії. Решта більш дрібні контури просто ігноруються [26].

Нижче відбувається обчислення керуючого впливу delta, яке в даному випадку являє собою різницю x-складових координат центру прямокутника, що описується, і центру кадру. Як додатковий параметр може також використовуватися кут повороту прямокутника angle між його довжиною та віссю

абсцис. Запис, що відповідає за це, в коді виглядає так:

delta = width/2 - rect.center.x; angle = rect.angle;

Наступним кроком є візуалізація отриманих даних за допомогою вже розібраних раніше функцій для малювання ліній та кіл, а також виведення тексту на екран.

Результат роботи цієї модифікації представлений рисунку 4.7.

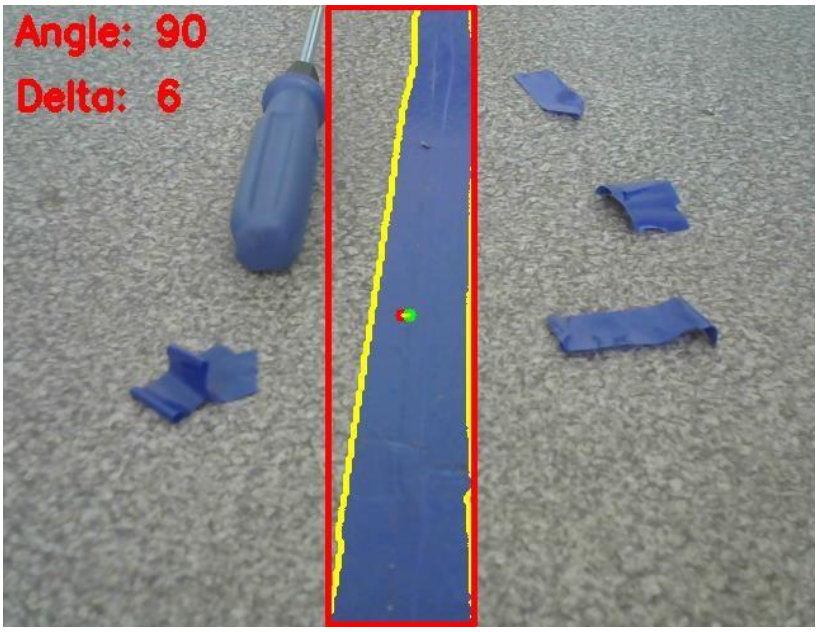

Рисунок 4.7 — Результат роботи програми модифікації №2

Головною перевагою описаного вище підходу з використанням контурів замість моментів зображення є можливість усунення негативного впливу випадкових перешкод і сторонніх об'єктів синього кольору в кадрі на рух системи.

Недоліками даної модифікації є збільшення кількості математичних перетворень, що відбуваються в процесі виконання програми, а також більше використання графічних функцій для візуалізації отриманих даних, що збільшує споживання обчислювальних ресурсів мікрокомп'ютера та негативно позначається на його швидкодії.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку системи реєстрації та розпізнавання напряму руху за допомогою комп’ютерного зору. Особливістю розробки є системи реєстрації та розпізнавання напряму руху за допомогою комп’ютерного зору.

Актуальність полягає у використанні систем комп'ютерного зору при розробці автоматично керованих засобів та автономних мобільних роботів у сфері складської логістики.

Аналогом може бути Робот Kiva компанії Amazon за ціною 30000 $, який використовує аналогічне програмне забезпечення.

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

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

Бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продукту в реальних умовах
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження табл. 5.1

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

Продовження табл. 5.1

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

Усі дані по кожному параметру занесено в таблиці 5.2

Таблиця 5.2 — Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	3	4	4
Наявність аналогів на ринку	3	3	4
Цінова політика	4	4	4
Технічні та споживчі властивості виробу	4	4	4
Експлуатаційні витрати	4	4	3
Ринок збуту	4	3	4
Конкурентоспроможність	3	4	3
Фахівці з технічної і комерційної реалізації	4	3	4
Фінансування	4	4	4
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	4	4	4
Супровідна документація	4	3	4
Сума	44	43	45
Середньоарифметична сума балів	(44+43+45) / 3 = 44		

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

Таблиця 5.3 — Рівні комерційного потенціалу розробки

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

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок використання систем комп'ютерного зору при розробці автоматично керованих засобів та автономних мобільних роботів у сфері складської логістики.

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

5.2.1 Основна заробітна плата розробників, яка розраховується за формулою:

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

де M — місячний посадовий оклад конкретного розробника (дослідника), грн.;

$T_р$ — число робочих днів за місяць, 20 днів;

t — число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 5.4.

Таблиця 5.4 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	33000	1650,00	45	74250,000
Програміст	26000	1300,00	45	58500,000
Всього				132750,00

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

5.2.2 Додаткова заробітна плата розробників, які брати участь в розробці обладнання/програмного продукту.

Додаткову заробітну плату прийнято розраховувати як 14 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 14\ \% / 100\ \% \qquad (5.2)$$

$$З_д = (132750{,}00 \cdot 14\ \% / 100\ \%) = 18585{,}00 \ \text{(грн.)}$$

5.2.3 Нарахування на заробітну плату розробників.

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

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \qquad (5.3)$$

$$Н_з = (132750{,}00 + 18585{,}00) \cdot 22\ \% / 100\ \% = 33293{,}70 \ \text{(грн.)}$$

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

5.2.4 Амортизація обладнання, яке використовувалось для проведення розробки в спрощеному вигляді розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \quad (5.4)$$

де Ц — балансова вартість обладнання, грн.;

Т — термін корисного використання обладнання згідно податкового законодавства, років;

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

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 20000 грн., термін його корисного використання згідно податкового законодавства — 2 роки, а термін його фактичного використання — 2,25 міс.

$$A_{обл} = \frac{20000}{2} \times \frac{2,25}{12} = 1875 \; грн.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 5.5. Так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн, то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю, $B_{нем.ак.}$ = 11000 грн.

Таблиця 5.5 — Амортизаційні відрахування на матеріальні та нематеріальні ресурси для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна периферія	20000	2	2,25	1875,000
Офісне обладнання (меблі)	21500	4	2,25	1007,813
Приміщення	850000	20	2,25	7968,750
Всього				10851,56

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

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (5.5)$$

де В – вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П – встановлена потужність обладнання, кВт. П = 0,5 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

$К_п$ – коефіцієнт використання потужності, $К_п$ = 0,9.

$$В_е = 0,9 \cdot 0,5 \cdot 8 \cdot 45 \cdot 6,2 = 1004,4 \text{ (грн.)}$$

5.2.5 Інші витрати та загальновиробничі витрати.

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

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.6)$$

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

$$I_в = 132750{,}00 * 80\% / 100\% = 106200 \text{ (грн.)}$$

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$H_{нзв} = (З_о + З_р) \cdot \frac{H_{нзв}}{100\%}, \quad (5.7)$$

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

$$H_{нзв} = 132750{,}00 * 133\ \% / 100\ \% = \ 176558 \text{ (грн.)}$$

5.2.6 Витрати на проведення науково-дослідної роботи.

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

$$В_{заг} = 132750{,}00+18585{,}00+33293{,}70+10851{,}56+11000+1004{,}40+106200+$$
$$+176558 = 490242{,}16 \text{ грн.}$$

5.2.7 Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів. Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \text{ (грн)}, \quad (5.8)$$

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

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

$$ЗВ = 490242,16 / 0,5 = 980484 \text{ грн.}$$

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

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

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

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

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

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

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

При розрахунку економічної ефективності потрібно обов'язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

— абсолютного економічного ефекту (чистого дисконтованого доходу);

— внутрішньої економічної дохідності (внутрішньої норми дохідності);

— терміну окупності (дисконтованого терміну окупності).

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

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

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

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (5.9)$$

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

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

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

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

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

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

p — коефіцієнт, який враховує рентабельність продукту;

ϑ — ставка податку на прибуток, у 2023 році ϑ =18%.

Припустимо, що при прогнозованій ціні 11200 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на *400* грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року — на 1500 шт., протягом другого року – на 1800 шт., протягом третього року на 2000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$$\Delta П_1 = (0*400 + (11200 + 400)*1500)* 0{,}8333* 0{,}33) * (1 - 0{,}18) = 3788399{,}848 \text{ грн.}$$

$$\Delta П_2 = (0*400 + (11200 + 400)*(1500+1800)* 0{,}8333* 0{,}33) * (1 - 0{,}18) = 8632139{,}655 \text{ грн.}$$

$$\Delta П_3 = (0*400 + (11200 + 400)*(1500+1800+2000)* 0{,}8333* 0{,}33) * (1 - 0{,}18) = 13863739{,}445 \text{ грн.}$$

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 26284278,95 грн.

5.3.2 Розрахунок ефективності вкладених інвестицій та періоду їх окупності. Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.10)$$

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

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

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,05...0,15$;

t — період часу (в роках).

Збільшення прибутку ми отримаємо, починаючи з першого року:

$$ПП = (3788399,848/(1+0,1)^1)+(8632139,655/(1+0,1)^2)+(13863739,445/$$
$$/(1+0,1)^3) = 3443999,86 + 7133999,715 + 10416032,64 = 20994032,22 \text{ грн.}$$

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

$$PV = k_{інв} * ЗВ, \quad (5.11)$$

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

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

$$PV = 2 * 980484 = 1960968{,}65 \text{ грн.}$$

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

$$E_{абс} = ПП - PV, \tag{5.12}$$

$$E_{абс} = 20994032{,}22 - 1960968{,}65 = 19033063{,}57 \text{ грн.}$$

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

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

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

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.13}$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{(1 + 19033063{,}57/1960968{,}65} - 1 = 1{,}204$$

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

$$\tau = d + f, \tag{5.14}$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d = (0,09…0,14);

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

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

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

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

$$T_{ок} = \frac{1}{E_в}, \tag{5.15}$$

$$T_{ок} = 1 / 1,204 = 0,83 \text{ р.}$$

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,83 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 980484 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний

продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,83 роки.

ВИСНОВКИ

У рамках виконання магістерської кваліфікаційної роботи було розроблено систему реєстрації та розпізнавання напряму руху за допомогою комп’ютерного зору.

У першому розділі було підтверджено актуальність використання систем комп'ютерного зору при розробці автоматично керованих систем та автономних мобільних роботів у сфері складської логістики.

У другому розділі було зроблено вибір одноплатного комп'ютера та веб-камери для системи, що розробляється. Було складено структурну схему розроблюваної системи та представлено перелік всіх використовуваних у ній елементів. Складено схему електричних з'єднань системи, що розробляється, а також представлені програми та блок-схеми алгоритмів передачі та прийому даних по послідовному порту з одноплатного комп'ютера Raspberry Pi на мікроконтролер Arduino.

У третьому розділі було зроблено вибір мови програмування, бібліотеки комп'ютерного зору та середовища розробки для реалізації програмної частини. Докладно описані алгоритми та функції бібліотеки комп'ютерного зору OpenCV, необхідні для поетапного перетворення відеоінформації, що отримується з веб-камери. Запропоновано методи реєстрації та розпізнавання напряму руху за допомогою комп'ютерного зору, а також здійснено розбір функцій бібліотеки OpenCV, що використовувалися в них.

У четвертому розділі було досліджено та докладно описано програму для розпізнавання напрямної лінії за допомогою комп'ютерного зору. Запропоновано можливі модифікації основної програми, а також розглянуто їх основні переваги та недоліки.

Проведено економічний розрахунок системи, що розробляється.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Навчальний посібник з дисципліни Системи візуалізації та розпізнавання образів [навчальний посібник] / Смолій В.В., Савицька Я.А., Місюра М.Д., Шкарупило В.В. // - К.: ФОП Ямчинський О.В., 2020.- 200 с.

2. Вовк С.М., Гнатушенко В.В., Бондаренко М.В. Методи обробки зображень та комп'ютерний зір. Навчальний посібник. – Д.:«ЛІРА», 2016. – 148 с.

3. Бодянський Є. В. Аналіз та обробка потоків даних засобами обчислювального інтелекту: Монографія / Є. В. Бодянський, Д. Д. Пелешко, О. А. Винокурова, С. В.Машталір, Ю. С. Іванов. Львів : Видавництво Львівської політехніки, 2016. 236 с.

4. Довбиш А.С. Основи теорії розпізнавання образів: навч. посіб. : у 2 ч. / А.С. Довбиш, І.В. Шелехов. Суми: Сумський державний університет, 2015. Ч.1. 109 с.

5. Методи, алгоритми і програмні засоби опрацювання біомедичних зображень / Березький О. М., Батько Ю.М., Березька К.М., Вербовий С.О., Дацко Т.В., Дубчак Л.О., Ігнатєв І.В., Мельник Г.М., Николюк В.Д., Піцун О.Й. Тернопіль: Економічна думка, ТНЕУ, 2017. 330 с.

6. Рашкевич Ю.М. Нейроподібні методи, алгоритми та структури обробки сигналів і зображень у реальному часі: монографія. / Ю.М. Рашкевич, Р.О. Ткаченко, Цмоць І.Г., Д.Д. Пелешко. Львів: Видавництво Львівської політехніки, 2017. 256 с. Допоміжна

7. William K. Pratt Digital image processing / Third Edition / John Wiley & Sons, Inc. – 2019. – 723 с.

8. Shalkoff R. J. Digital image processing and computer vision / R. J. Shalkoff. – New York-Chichester-Brisbane-TorontoSingapore: John Wiley & Sons, 1989. – 489p.

9. Duda R. O. Pattern Classification, second ed. / R. O. Duda, P. E. Hart, D. G. Stork. – John Wiley & Sons, New York, 2001. 738 p.

10. Зайченко Ю. П. Основи проектування інтелектуальних систем: навчальний посібник / Ю. П. Зайченко. – К. :106 Видавничий Дім «Слово», 2004. – 352 с.

11. Hastie T. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. – 2nd ed. / T. Hastie, R. Tibshirani, J. Friedman. – Springer-Verlag, 2009. – 746 p. Інформаційні ресурси в Інтернет

12. Machine vision. Article from Wikipedia, the free encyclopedia [Електронний ресурс]: – Режим доступу: https://en.wikipedia.org/wiki/Machine_vision

13. Introducing Myriad X: Unleashing AI at the Edge. [Електронний ресурс]: – Режим доступу: https://newsroom.intel.com/editorials/introducing-myriad-xunleashingai-at-the-edge/#gs.owkeej

14. Computer Vision: Algorithms and Applications by Richard Szeliski. Доступно безкоштовно онлайн. http://szeliski.org/Book/

15. Computer Vision: A Modern Approach (Second Edition) by David Forsyth and Jean Ponce. Доступно безкоштовно онлайн. http://luthuli.cs.uiuc.edu/~daf/CV2E-site/cv2eindex.html14.

16. What is Machine Vision?. Retrieved May 1, 2018, from website: https://machinevision.co.uk/machine-vision-products/vision-hardware/

17. Очима робота: що таке «машинний зір». Retrieved May 1, 2018, від website: https://www.popmech.ru/technologies/238704-glazami-robota-chto-takoe-mashinnoe-zrenie/

18. Robots: a new kind of manufacturing workforce. Retrieved May 1, 2018, від website: https://www.manufacturingtomorrow.com/article/2016/03/robots-a-new-kind-of-manufacturing-workforce/7797

19. The Role of Machine Vision в Automotive Industry. Retrieved May 1, 2018, від website:https://www.photonics.com/a58196/The_Role_of_Machine_Vision_in_the_Autom otive

20. KR 40 PA. Retrieved May 1, 2018, from website: https://www.kuka.com/en-de/products/robot-systems/industrial-robots/kr-40-pa

21. Serial Communication [Електронний ресурс]. URL:https://learn.sparkfun.com/tutorials/serial-communication/all (Дата звернення: 07.11.2023).

22. Contours in OpenCV. Retrieved May 1, 2018, from website: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_contours/py_contours_begin/py_contours_begin.html#contours-getting-started

23. Camera Calibration. Retrieved May 1, 2018, from website https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_calib3d/py_calibration/py_calibration.html#calibration

24. CodeBlocks – середовище програмування мовою C/C++ [Електронний ресурс]. URL: https://progtips.ru/instrumenty- programmista/codeblocks.html (Дата звернення: 16.04.2023).

25. Color Detection & Object Tracking [Електронний ресурс]. URL:https://www.opencv-srf.com/2010/09/object-detection-using-color-seperation.html (Дата звернення: 20.11.2023).

26. Проходження лінії на основі OpenCV [Електронний ресурс]. URL:https://habr.com/ua/articles/426675/ (Дата звернення: 08.11.2023).

27. Davies, ER Computer and Machine Vision: Theory, Algorithms, Practicalities / ER Davies. – Oxford: Academic Press is imprint of Elsevier, 2012. – ISBN: 978-0-12-386908-1.

ДОДАТОК А

Технічне завдання

Міністерство освіти та 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

«___» _____ 20__ р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

«Система реєстрації та розпізнавання напряму руху за допомогою комп'ютерного зору»

08-54.МКР.018.00.000 ПЗ

Науковий керівник

к.т.н., доц. каф. ОТ

_____ Колесник І.С.

виконав:

магістрант 2 курсу,

_____Ткачук В.М..

Вінниця 2023

1. Підстава виконання магістерської кваліфікаційної роботи

1.1 Одним із найбільш актуальних напрямів розвитку штучного інтелекту в останні роки став комп'ютерний зір. Здатність бачити — найважливіша властивість людини. Саме за допомогою зору ми отримуємо найбільшу кількість інформації про навколишній світ.

1.2 Наказ про затвердження теми МКР

2 Мета і призначенням МКР

2.1 Метою роботи є вдосконалення системи реєстрації та розпізнавання напряму руху за допомогою комп’ютерного зору.

2.2 Призначення розробки — виконання магістерської кваліфікаційної роботи.

3 Вихідні дані для виконання МКР

Вихідні дані для виконання МКР: методи засновані на класичних алгоритмах машинного навчання, мова програмування C++, бібліотека комп'ютерного зору з відкритим вихідним кодом OpenCV та кросплатформове середовище розробки Code:: Blocks.

4 Вимоги до виконання МКР

МКР повинна задовольняти такі вимоги:

— забезпечити протокол взаємодії апаратної частини з комп’ютером;

— провести моделювання та тестування системи;

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в табл. А.1.

6 Матеріали, що подаються до захисту МКР

До захисту МКР подаються: пояснювальна записка МКР, ілюстративні та графічні матеріали, протокол попереднього захисту МКР на кафедрі, відзив наукового керівника, відзив опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами, довідка про відповідність оформлення МКР діючим вимогам.

Таблиця А.1 — Етапи МКР

№ з/п	Назва етапів виконання магістерської роботи	Строк виконання етапів роботи	Прим.
1	Постановка мети та задач роботи	21.10.23	
2	Основні функціональні аспекти комп'ютерного зору	25.10-30.10.23	
3	Проектування апаратного забезпечення системи	31.10-08.11.23	
4	Складання структурної схеми пристрою, що розробляється	09.11-15.11.23	
5	Складання електричної схеми пристрою, що розробляється	16.11-.20.1123	
6	Використання інструментів розробки	21.11-25.11.23	
7	Програмна реалізація рішень	26.11-31.11.23	
8	Можливі варіанти модернізації програми	01.12-04.12.23	
9	Розрахунок економічної частини роботи	01.12-04.12.23	
10	Оформлення пояснювальної записки та ілюстративного матеріалу	05.12.23	
11	Аналіз виконання роботи, висновки, додатки		
12	Перевірка якості виконання магістерської роботи та усунення недоліків		

7 Порядок контролю виконання та захисту МКР

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

8 Вимоги до оформлення МКР

8.1 При оформлюванні МКР використовуються:

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

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

— Методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп'ютерна інженерія». Кафедра обчислювальної техніки ВНТУ 2022.

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

ДОДАТОК Б

Програмний код основного алгоритму роботи

```
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/videoio.hpp>
#include <iostream>
#include <vector>
#include <wiringPi.h>
 #include <wiringSerial.h>
#include <unistd.h>
        using namespace cv; using namespace std;
        int main(int argc, char** argv)
        {
                Mat frame, hsv, mask;
                Moments m;
                int width, x, y, center, delta;
                vector<int>lowerBlue = { 75, 60, 50 };
                vector<int>upperBlue = { 130, 255, 255 };
                char buffer[5] = { 0 };
                VideoCapture cap; cap.open(0);
                int serialPort = serialOpen("/dev/ttyUSB0", 9600);
                if (serialPort == -1)
                {
                        cout << "Failed open port" << endl; return 0;
                        }
                delay(1000);
                for (;;)
                {
```

```
cap >> frame;
if (frame.empty())
{
                cout << "Camera error" << endl; break;
            }
            width = frame.size().width;
            cvtColor(frame, hsv, COLOR_BGR2HSV);
            inRange(hsv, lowerBlue, upperBlue, mask);
            m = moments(mask, 1);
            if (m.m00 > 600)
            {
             x = m.m10 / m.m00;
             y = m.m01 / m.m00;
             center = width / 2;
             delta = center - x;

             circle(frame, Point(x, y), 5, Scalar(0, 0, 255), -1);
             circle(frame, Point(center, y), 5, Scalar(0, 255, 0), -1);
             line(frame, Point(center, y), Point(x, y), Scalar(0, 255, 255),
            2); sprintf_s(buffer, "%d", delta);
             putText(frame, buffer, Point(width / 2, y - 20),
            FONT_HERSHEY_SIMPLEX, 1, Scalar(0, 255, 255), 3);
  serialPrintf(serialPort, "%s\n", buffer);
  memset(buffer, 0, sizeof buffer);
                }
            else
  {
  cout << "Line was not detected" << endl;

            }
```

```
 namedWindow("Window", WINDOW_AUTOSIZE);
moveWindow("Window", 70, 70);
 imshow("Window", frame);
 if (waitKey(10) >= 0) break;
}
destroyWindow("Window");
serialClose(serialPort);
return 0;
}
```

ДОДАТОК В

Програмний код модифікації №1 програми

```
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/videoio.hpp>
#include <iostream>
#include <vector>
#include <wiringPi.h>
 #include <wiringSerial.h>
#include <unistd.h>
        using namespace cv;
        using namespace std;
        int main(int argc, char** argv)
        {
        Mat frame, hsv, mask, top, mid, bot, maskTop, maskMid, maskBot;
         Moments m, mTop, mMid, mBot;
        int height, width;
        int topX = 0, topY = 0, midX = 0, midY = 0, botX = 0, botY = 0, topDelta = 0,
midDelta = 0, botDelta = 0, delta;
        vector<int>lowerBlue = { 75, 60, 50 };
        vector<int>upperBlue = { 130, 255, 255 };
        char showDeltaTop[5] = { 0 };
        char showDeltaMid[5] = { 0 };
        char showDeltaBot[5] = { 0 };
        char buffer[5] = { 0 };
        VideoCapture cap; cap.open(0);
        int serialPort = serialOpen("/dev/ttyUSB0", 9600);
        if (serialPort == -1)
```

```
{
cout << "Failed open port" << endl;
return 0;
}
delay(1000);
for (;;)
{
cap >> frame;
if (frame.empty())
{
cout << "Camera error" << endl;
break;
}
height = frame.size().height;
width = frame.size().width;
cvtColor(frame, hsv, COLOR_BGR2HSV);
top = hsv(Range(0, (height / 3)), Range(0, width));
mid = hsv(Range((height / 3), 2*(height / 3)), Range(0, width)); bot = hsv(Range(2 * (height / 3), height), Range(0, width));
inRange(top, lowerBlue, upperBlue, maskTop); mTop = moments(maskTop, 1);
inRange(mid, lowerBlue, upperBlue, maskMid); mMid = moments(maskMid, 1);
inRange(bot, lowerBlue, upperBlue, maskBot); mBot = moments(maskBot, 1);
if (mTop.m00 > 600)
{
topX = mTop.m10 / mTop.m00;
topY = mTop.m01 / mTop.m00;
topDelta = width / 2 - topX;
```

```
sprintf_s(showDeltaTop, "%d", topDelta);
circle(frame, Point(topX, topY), 5, Scalar(0, 0, 255), -1);
circle(frame, Point(width / 2, topY), 5, Scalar(0, 255, 0), -1);
putText(frame, showDeltaTop, Point(width / 2, topY - 20), FONT_HERSHEY_SIMPLEX, 1, Scalar(0, 255, 255), 3);
line(frame, Point(width / 2, topY), Point(topX, topY),
Scalar(0, 255, 255), 2);
}
else
{
cout << "The top of the line was not detected" << endl;
}
if (mMid.m00 > 600)
{
midX = mMid.m10 / mMid.m00;
midY = mMid.m01 / mMid.m00;
midDelta = width / 2 - midX;
sprintf_s(showDeltaMid, "%d", midDelta);
circle(frame, Point(midX, midY + (height / 3)), 5, Scalar(0, 0, 255), -1);
circle(frame, Point(width / 2, midY + (height / 3)), 5, Scalar(0, 255, 0), -1);
putText(frame, showDeltaMid, Point(width / 2, midY + (height / 3) - 20), FONT_HERSHEY_SIMPLEX, 1, Scalar(0, 255, 255), 3);
line(frame, Point(width / 2, midY + (height / 3)), Point(midX, midY + (height / 3)), Scalar(0, 255, 255), 2);
}
else
{
cout << "The middle part of the line was not detected" << endl;
}if (mBot.m00 > 600)
{
```

```
botX = mBot.m10 / mBot.m00;
botY = mBot.m01 / mBot.m00;
botDelta = width / 2 - botX;
sprintf_s(showDeltaBot, "%d", botDelta);
circle(frame, Point(botX, botY + (2 * height / 3)), 5, Scalar(0, 0, 255), -1);
circle(frame, Point(width / 2, botY + (2 * height / 3)), 5,
Scalar(0, 255, 0), -1);
putText(frame, showDeltaBot, Point(width / 2, botY + (2 * height / 3) – 20),
FONT_HERSHEY_SIMPLEX, 1, Scalar(0, 255, 255), 3);
line(frame, Point(width / 2, botY + (2 * height / 3)), Point(botX, botY + (2 *
height / 3)), Scalar(0, 255, 255), 2);
}
else
{
cout << "The bottom of the line was not detected" << endl;
}
topDelta = 0.25 * topDelta;
midDelta = 0.5 * midDelta;
botDelta = 2 * botDelta;
delta = (topDelta + midDelta + botDelta) / 3;
serialPrintf(serialPort, "%s\n", buffer);
memset(buffer, 0, sizeof buffer);
namedWindow("Window", WINDOW_AUTOSIZE);
moveWindow("Window", 70, 70); imshow("Window", frame);
if (waitKey(10) >= 0) break;
}
destroyWindow("Window");
serialClose(serialPort);
return 0;
}
```

ДОДАТОК Г

Програмний код модифікації №2 програми

```
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/videoio.hpp>
 #include <iostream>
#include <vector>
#include <wiringPi.h>
#include <wiringSerial.h>
#include <unistd.h>
using namespace cv; using namespace std;
int main(int argc, char** argv)
{
Mat frame, hsv, mask, blur;
vector<int>lowerBlue = { 75, 60, 50 };
vector<int>upperBlue = { 130, 255, 255 };
vector<vector<Point>>contours;
vector<Vec4i> hierarchy;
int delta, width, angle;
double area, maxArea = 12000;
char buffer[5] = { 0 }; RotatedRect rect;
Point2f rectPoints[4];
VideoCapture cap; cap.open(0);
int serialPort = serialOpen("/dev/ttyUSB0", 9600);
 if (serialPort == -1)
{
cout << "Failed open port" << endl; return 0;
}
```

```
delay(1000);
for (;;)
{
cap >> frame;
if (frame.empty())
{
 cout << "Camera error" << endl;
break;
}
width = frame.size().width;
cvtColor(frame, hsv, COLOR_BGR2HSV);
inRange(hsv, lowerBlue, upperBlue, mask);
findContours(mask, contours, hierarchy, RETR_TREE,
CHAIN_APPROX_NONE);
if (contours.empty())
{
cout << "Line was not detected" << endl;
continue;
}
for (int i = 0; i < contours.size(); i++)
{
area = contourArea(contours[i]);
 if (area > maxArea)
{
drawContours(frame, contours, i, Scalar(0, 255, 255), 3); rect =
minAreaRect(contours[i]);
}
}
rect.points(rectPoints);
delta = width / 2 - rect.center.x; angle = rect.angle;
```

```
if (rect.size.width < rect.size.height)
{
angle += 90;
}
for (int j = 0; j < 4; j++)
{
line(frame, rectPoints[j], rectPoints[(j + 1) % 4], Scalar(0, 0, 255), 3);
}
circle(frame, rect.center, 5, Scalar(0, 0, 255), -1);
circle(frame, Point(width / 2, rect.center.y), 5, Scalar(0, 255, 0), -1);
line(frame, rect.center, Point(width / 2, rect.center.y), Scalar(0, 255, 255), 2);
sprintf_s(buffer, "%d", angle);
putText(frame, "Angle:", Point(10, 30), FONT_HERSHEY_SIMPLEX, 1,
Scalar(0, 0, 255), 3);
putText(frame, buffer, Point(120, 30), FONT_HERSHEY_SIMPLEX, 1,
Scalar(0, 0, 255), 3);
memset(buffer, 0, sizeof buffer);
sprintf_s(buffer, "%d", delta);
putText(frame, "Delta:", Point(10, 80), FONT_HERSHEY_SIMPLEX, 1,
Scalar(0, 0, 255), 3);
putText(frame, buffer, Point(120, 80), FONT_HERSHEY_SIMPLEX, 1,
Scalar(0, 0, 255), 3);
serialPrintf(serialPort, "%s\n", buffer);
memset(buffer, 0, sizeof buffer);
namedWindow("Window", WINDOW_AUTOSIZE);
moveWindow("Window", 70, 70);
 imshow("Window", frame);
if (waitKey(10) >= 0) break;
destroyWindow("Window"); serialClose(serialPort); return 0;
```

ДОДАТОК Д

Схема електричних з'єднань

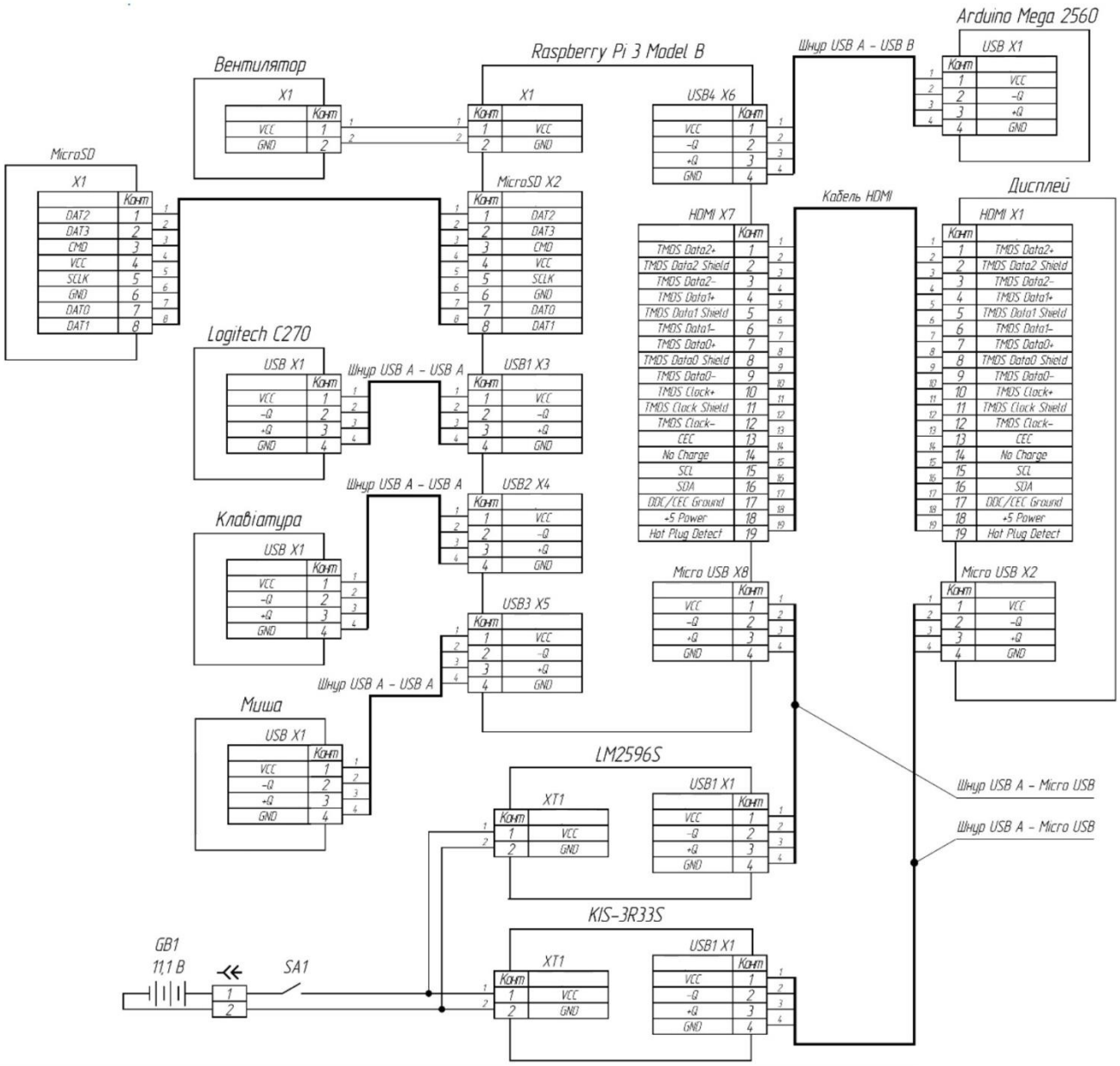

Рисунок Д.1 — Схема електричних з'єднань

ДОДАТОК Е

Блок-схема загального алгоритму роботи програми

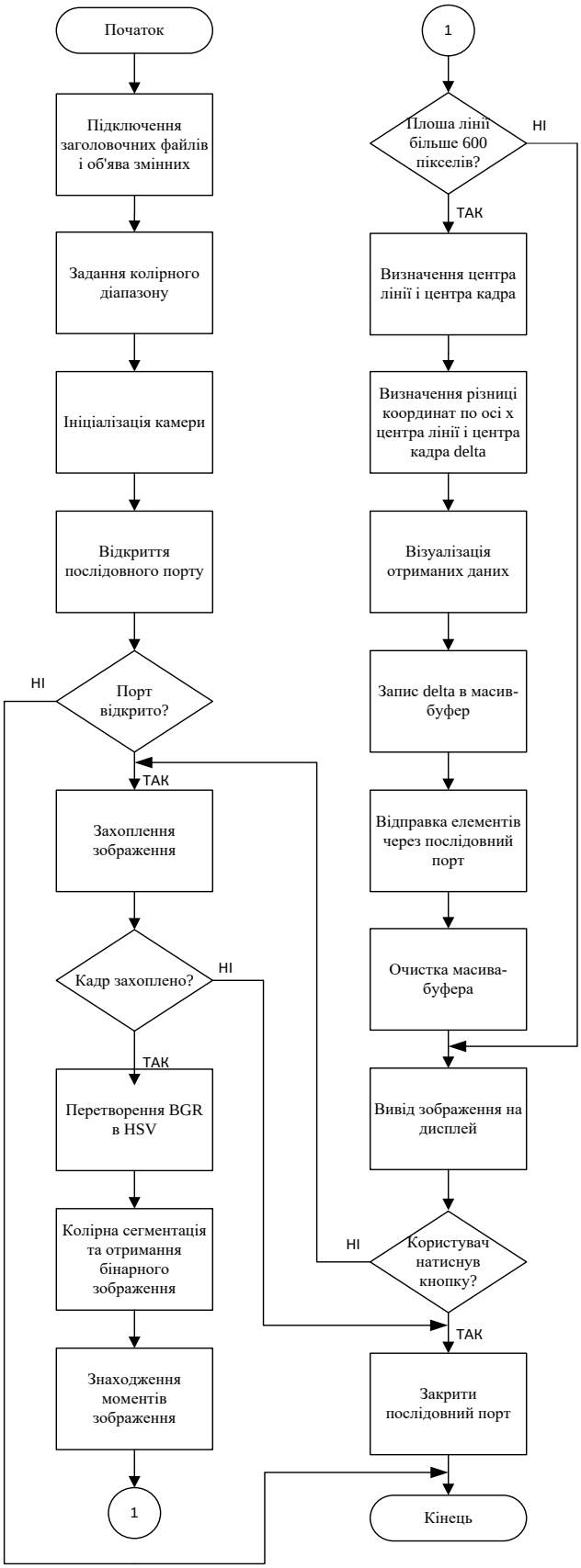

Рисунок Е.1 — Блок-схема загального алгоритму роботи програми

ДОДАТОК Ж

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Система реєстрації та розпізнавання напряму руху за допомогою комп'ютерного зору

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 95,4 Схожість 4.6%

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 _____ Ткачук В.М
(підпис) (прізвище, ініціали)

Керівник роботи _____ Колесник І.С.
(підпис) (прізвище, ініціали)