МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Литвиненко О.Є.

«___»_____2021 р.

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: Модуль розпізнавання жестів рук для керування квадрокоптером

Виконавець: Левчук В.В.

Керівник: Глазок О.М.

Нормоконтролер: Тупота Є.В.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет Кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Спеціальність 123 «Комп'ютерна інженерія»
(шифр, найменування)

Освітньо-професійна програма «Системне програмування»
Форма навчання денна

ЗАТВЕРДЖУЮ
Завідувач кафедри

Литвиненко О.Є.
«_____»_____2021 р.

ЗАВДАННЯ
на виконання дипломної роботи (проєкту)

Левчука Владислава Володимировича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи (проєкту) «Модуль розпізнавання жестів рук для керування квадрокоптером»

затверджена наказом ректора від « 04 » лютого 2021 р. № 135/ст.

2. Термін виконання роботи (проєкту): з 17 травня 2021 р. по 20 червня 2021 р.

3. Вихідні дані до роботи (проєкту): мова програмування *C++*, середовище програмування *Visual Studio Code*, бібліотека *MediaPipe*, бібліотека *OpenCV*, одноплатний комп'ютер *Raspberry Pi*

4. Зміст пояснювальної записки:

1) Аналіз засобів, методів та апаратного забезпечення, які застосовують при створенні інтерактивних систем керування жестами;

2) Проектування модуля розпізнавання жестів рук для керування квадрокоптером;

3) Програмна та апаратна реалізація модуля розпізнавання жестів рук для керування квадрокоптером.

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:

1) Програмне забезпечення (Діаграма прецедентів);

2) Програмне забезпечення (Діаграма послідрвності);

3) Програмне забезпечення (Діаграма класів);

4) Апаратне забезпечення (Схема структурна);

5) Апаратно-програмний модуль (Ескіз поєднання модуля і коптера)

6. Календарний план

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1.	Ознайомитися з літературою за темою дипломного проекту	17.05.2021-19.05.2021	
2.	Провести аналіз та огляд існуючих систем розпізнавання	19.05.2021-21.05.2021	
3.	Ознайомитися із перспективними апаратними платформами, які підходять для реалізації модуля.	21.05.2021-22.05.2021	
4.	Ознайомитися з існуючими бібліотеками для роботи із комп'ютерним зором.	22.05.2021-24.05.2021	
5.	Розробити план реалізації апаратної частини.	24.05.2021-26.05.2021	
6.	Реалізувати апаратну частину модуля і провести його відлагодження.	26.05.2021-29.05.2021	
7.	Розробити план реалізації програмної частини.	29.05.2021-01.06.2021	
8.	Реалізувати програмну частину модуля і провести його відлагодження.	01.06.2021-05.06.2021	
9.	Написати пояснювальну записку	05.06.2021-10.06.2021	
10.	Підготувати графічний матеріал	10.06.2021-12.06.2021	
11.	Підготувати презентацію та доповідь	12.06.2021-13.06.2021	

7. Дата видачі завдання: «_17_»___травня___ 2021 р.

Керівник дипломної роботи (проєкту) _____ Глазок О.М.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Левчук В.В.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Модуль розпізнавання жестів рук для керування квадрокоптером»: 58 с., 25 рис., 1 табл., 28 літературних джерел.

ШТУЧНИЙ ІНТЕЛЕКТ, ГЛИБОКЕ НАВЧАННЯ, ОДНОПЛАТНА ЕОМ, *Raspberry Pi*, *MediaPipe*, *OpenCV*, ЖЕСТ, КВАДРОКОПТЕР.

Об’єкт дослідження – розпізнавання жестів рук на основі нейромережевих програмних засобів.

Предмет дослідження – модуль розпізнавання жестів рук для керування квадрокоптером.

Мета дипломного проекту – створення модуля на основі одноплатної ЕОМ для розпізнавання жестів рук з використанням алгоритмів штучного інтелекту з подальшою інтерпретацією результатів роботи модуля у команди для керування квадрокоптером.

Метод проєктування – здійснення розробки, збирання та налаштування апаратної частини модуля, написання і відлагодження коду, тестування модуля.

Галузь застосування – дистанційне жестове керування динамічними системами.

Значущість роботи та висновки – модуль жестикуляційного керування забезпечує просте та інтуїтивне керування рухомими системами, такими як квадрокоптер.

Прогнози та припущення щодо розвитку об’єкта дослідження – створення робочого зразка модуля, інтеграція зразка у квадрокоптер та використання його для керування. Подальший розвиток системи передбачає додавання нового функціоналу у модуль, зокрема: розпізнавання та ідентифікація обличчя власника або кола персон, що мають допуск до керування, розширення спектру жестів, які підлягають розпізнаванню, додавання можливості розпізнавання звукових команд.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ОЗП – Оперативний запам’ятовуючий пристрій

ПЗЗ – Пристрій із зарядним зв’язком

CCD – *Charge-coupled device*

КМОН – Комплементарна структура метал-оксид-напівпровідник

CMOS – *Complementary-symmetry/metal-oxide semiconductor*

Мп – Мегапіксель

ЗПП – Зворотне поширення помилки

ЦАП – Цифро-аналоговий перетворювач

LCD – *Liquid crystal display*

HSV – *Hue, Saturation, Value*

GPIO – *General-purpose input/output*

USB – *Universal Serial Bus*

HDML – *Handheld Device Markup Language*

DSI – *Display Serial Interface*

CSI – *Camera Serial Interface*

PWM – *Pulse-width modulation*

ШІМ – Широтно-імпульсна модуляція

JSON – *JavaScript Object Notation*

ВСТУП

Актуальність теми. На сьогоднішній день збільшується використання електронних систем у повсякденній діяльності людей [1-3]. Ріст попиту людей на задоволення своїх потреб як у побуті, так у вирішенні низки проблем економічного та логістичного характеру безпрецедентно спонукає до розвитку науково-технічної галузі систем. Виникає дедалі більший запит на інтерактивні системи людино-комп'ютерної взаємодії, які забезпечують керування людиною різноманітними технічними засобами за допомогою дедалі більш простого інтерфейсу [4]. Можна виділити кілька причин для спрощення систем інтерфейсу керування людино-машинних комплексів, а саме:

– легкість в освоєні користувачем простого та зручного інтерфейсу;

– зменшення витрат на навчання персоналу для керування комплексом;

– інтуїтивно зрозумілий інтерфейс зменшує ризик виникнення помилок під час освоєння та експлуатації техніки.

Однією з таких систем інтерфейсу керування є керування за допомогою жестикуляції рук людини, що вимагає використання автоматичного розпізнавання жестів [5].

Інтерактивна система розпізнавання жестів – це комплекс, який розкриває нові можливості для керування різноманітними системами. Системи такого типу інтуїтивно зрозумілі, завдяки тому, що керування здійснюється жестикуляцією рук оператора. Сам процес розпізнавання жестів рук ділиться на кілька етапів, протягом яких система отримує інформацію, аналізує її, обробляє і на основі неї здійснює висновок щодо жестикуляцій. Інтерфейси керування за допомогою жестикуляції рук лише починають впроваджуватися, поки що вони знаходяться переважно на стадії досліджень.

Дані системи влаштовані на основі процесів виявлення об'єктів та алгоритмів розпізнання характеристик, яким наділені виявленні об'єкти. Під процесом виявлення об'єктів мається на увазі технології комп'ютерного зору та обробки зображень, а алгоритми розпізнання втілюють у собі алгоритми штучно-

го інтелекту. Тісне поєднання даних технологій забезпечує можливість функціонувати системам розпізнавання жестів, зокрема здатність виявляти руки людей і їх окремих елементів (наприклад кисті руки) і встановлювати характеристики цих рухів та положення рук людей. На основі отриманої інформації від оператора, система може примати відповідні рішення або здійснювати певні дії. Хоча такі інтерактивні системи розпізнавання жестів є складним, але у той же час, надзвичайно вагомим знаряддям людино-машинного інтерфейсу. Окрім раніше відзначених переваг, такі системи надають вибірку додаткових аргументів у бік переваг систем керування такого виду, а саме:

– виключення фізичного контакту оператора з комплексом, що досить актуально, зважаючи на епідеміологічний стан у світі;

– можливість застосувати даний спосіб керування як додатковий, або такий, що розширює «класичні» способи керування коптерами – керування на основі радіоуправлінні;

– можливість поєднання з системами, які вже впровадженні у коптерах і також використовують ті самі апаратні елементи (відеокамера), що потрібні для системи розпізнавання жестів, наприклад з системою розпізнавання обличчя.

Одним з можливих рішень у застосуванні інтерактивних систем керування жестами, на думку автора, є керування літальними апаратами – мультикоптерами. Натхненний швидким та буремним розвитком коптеробудування та появою нових алгоритмів, методик та інструментів у галузях комп’ютерного зору та глибокого навчання (*deep learning*), автор загорівся бажанням створити власний модуль розпізнавання жестів рук для керування квадрокоптером. Даний вид керування дозволить розширити можливості по управлінню літальними апаратами даного типу, а не лише застосовувати загальноприйняті пульти керування на радіоуправлінні.

Зважаючи на вищевисвітлені аргументи на користь інтерактивних систем керування жестами, було вирішено спроектувати та розробити систему керування такого виду, яка надасть достатні можливості у плані здійснення повноцінного керування літальний апаратами з трьома і більше несучими гвинтами – мультикоптерами.

На основі цього визначається подальша мета, предмет, завдання та зміст дипломного проектування.

Об'єкт дослідження – розпізнавання жестів рук на основі нейромережевих програмних засобів.

Предмет дослідження – модуль розпізнавання жестів рук для керування квадрокоптером.

Мета дипломного проекту – створення модуля на основі одноплатної ЕОМ для розпізнавання жестів рук з використанням алгоритмів штучного інтелекту з подальшою інтерпретацією результатів роботи модуля у команди для керування квадрокоптера.

Для досягнення цієї мети необхідно виконати наступні завдання:

1. Обрати відповідне апаратне забезпечення;
2. Зібрати та налаштувати апаратну частину модуля;
3. Написати програмне забезпечення для роботи модуля;
4. Відладити програмне забезпечення;
5. Перевірити роботу програмного забезпечення з апаратною частиною модуля.

Метод проєктування – здійснення розробки, збирання та налаштування апаратної частини модуля, написання і відлагодження коду, тестування модуля.

Галузь застосування – дистанційне жестове керування динамічних систем.

Значущість роботи та висновки – модуль жестикуляційного керування забезпечує просте та інтуїтивне керування рухомими системами, такими як квадрокоптер.

Прогнози та припущення щодо розвитку об'єкта дослідження – створення робочого зразка модуля, інтеграція зразка у квадрокоптер та використання його для керування. Подальший розвиток системи передбачає додавання нового функціоналу у модуль, зокрема: розпізнавання та ідентифікація обличчя власника, розширення спектру жестів, які підлягають розпізнаванню, додавання можливості розпізнавання звукових команд.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Загальна схема роботи інтерактивних систем розпізнавання жестів

Розвиток технологічного прогресу, поява масово доступного апаратного забезпечення датчиків та обчислювальних машин, систем штучного інтелекту, привів до можливості створення систем людино-машинного інтерфейсу, заснованого на розпізнаванні зображення людини з відеокамери. До таких систем відносяться системи керування жестами, які здатні розрізняти певні жести оператора і на основі них виконувати ті чи інші дії. Важливо, що така система має бути здатна виявляти та розрізняти серед усього різноманіття рухів людини множину конкретних жестів, проводити аналіз даних жестів та інтерпретувати їх, перетворюючи у послідовність команд або іншого вигляду інформацію. У такий спосіб досягається обмін інформацією між людиною (оператором) і машиною, у коло задач якої входить виявлення жестів оператора і виконання відповідної послідовності дій, залежно від наданого жеста.

В загальному розумінні, під визначенням «інтерактивна система керування жестами» розуміється система, яка здатна виявляти та розрізняти серед стохастичного набору рухів людини набір конкретних жестів, проводити аналіз даних жестів та інтерпретувати їх у вибірку команд або іншого вигляду інформацію, яка може використовуватися як самою системою так і іншими системами чи комплексами. Дана система має складну структуру, яку можна розподілити на окремі блоки. Кожен з таких блоків володіє набором функції, які мають бути виконані на певному етапі роботи системи.

На рис. 1.1 наведено загальну структуру таких систем [6-8].

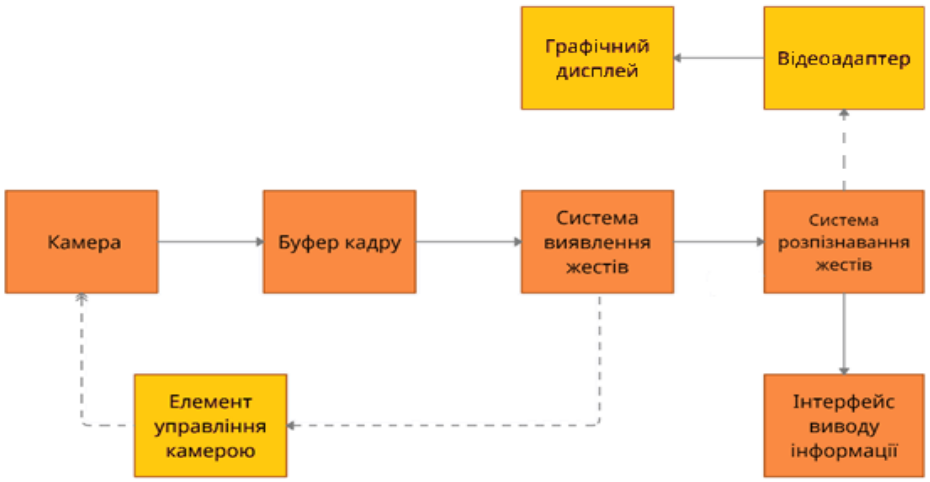

Рис. 1.1. Загальна схема інтерактивної системи керування жестами

У даній системі наявні наступні основні вузли: камера, буфер кадру, система виявлення жестів, система розпізнавання жестів, інтерфейс виведення інформації (відеоадаптер та графічний дисплей необов'язкові елементи), елемент управління камерою (необов'язковий елемент).

Розглянемо більш детально кожен із даних елементів і взаємодію між ними.

Камера є основним елементом, який забезпечує систему інформацією у вигляді кадрів відеопотоку або окремих зображень. Кадри зображення є основною інформацією, якою оперує система [9].

Необхідно описати основні характеристики, за якими обирають камери, для систем керування подібного типу. Серед основних характеристик камер, можна виділити: тип матриці, розмір матриці, фокусна відстань та кут огляду, роздільна здатність, додаткові можливості.

Більшість камер володіють одним із двох типів матриць: ПЗЗ-матрицю (англ. *CCD*) або матрицю на основі КМОН-технології (англ. *CMOS*). Основна відмінність – спосіб зчитування інформації. У ПЗЗ-матриць зчитування відбувається послідовно, у КМОН-матриць зчитування відбувається у довільний спосіб. Переваги ПЗЗ полягають у більш якісному зображені, ніж у КМОН, проте у КМОН формування кадру зображення відбувається значно швидше. Фізичний розмір матриці визначає якість як самої матриці, так якість формування зображення. Якість зображення залежить саме від розміру окремого елемента

матриці – комірки. Збільшення розміру даної комірки призводить до покращення характеристик якості кадру відео чи зображення.

Фокусна відстань та кут огляду – два взаємопов'язані параметри, які визначають дальність та кут огляду камери. Зі зменшенням фокусної відстані зростає кут огляду, і зменшується кількість деталей на кадрі.

Параметр роздільної здатності. Роздільна здатність вимірюється у мегапікселях (*Mn*). Кількість мегапікселів визначає розмір і якість зображення.

Буфер кадру слугує місцем, куди поміщається зображення, отримане у ході зйомки. Зображення з камери може зберігатися у різноманітних форматах, які відрізняються якістю, у якій зберігають зображення.

Чим більш якісне зображення, тим системі керування жестами знадобиться здійснити більше роботи над зображенням протягом всього робочого циклу: формування кадру, попередня обробка кадру, опрацювання кадру системою виявлення жестів. Це у свою чергу може сильно відобразитися на продуктивності окремих блоків, що впливатиме на всю систему в цілому. Таким чином, від формату зображення напряму може залежати робота всієї системи і необхідно підібрати формат, що забезпечить оптимальний рівень навантаження обчислювальної системи. Наразі, найбільш популярними форматами зображень є: *GIF*, *BMP*, *JPEG*, *PNG*. Кожен із форматів надає свої переваги, але одночасно має і певні недоліки. Вибираючи формат, необхідно брати до уваги можливості інтерактивної системи керування жестами.

Система виявлення жестів. Отримавши інформацію з камери і провівши збереження кадру в одному із форматів, отримали зображення, яке слугуватиме одиницею інформації у інтерактивній системі розпізнавання жестів. Дане зображення необхідно попередньо обробити спеціальними засобами.

Жест – це така дія або рух людини, що забезпечує передачу інформації чи емоції без застосування мовлення. Жестом можна назвати будь-яку послідовність рухів людини, які можуть бути передати певну інформацію, або нести певний сенс. Дане зображення необхідно попередньо обробити спеціальними засобами.

Жест – це така дія або рух людини, що забезпечує передачу інформації чи емоції без застосування мовлення. Жестом можна назвати будь-яку послідовність

рухів людини, які можуть бути передати певну інформацію, або нести певний сенс. Людина використовує різноманітні жести, такі як: маніпулятивні, комунікаційні, емоційні, ілюстративні жести.

З погляду системи, яка має розпізнавати жести, жест є одиницею інформації. Як було сказано раніше, перед початком розпізнавання та класифікації жесту, необхідно певним чином обробити кадр зображення, щоб мати змогу виявити сам жест. Надалі кадр можна використовувати як для роботи розпізнавання і класифікації жестів, так і для безпосереднього розпізнавання логіки жеста. Засоби і методи, які використовують для виявлення жеста на кадрі зображення, буде детально висвітлено у підрозділі 1.2.

У ранніх дослідженнях по розпізнаванню і керуванню жестами часто використовувалися спеціальні маркери на частинах тіла, зокрема на кінчиках пальців, що дозволяло відслідковувати ключові точки на руці [10]. Ці маркери представляли собою певні невеликі елементи визначеної кольорової гамми. Система аналізувала зображення на наявність цих «кольорових точок», і у разі, якщо колір даних точок входить у визначений діапазон кольорів, точка ідентифікувалася як ключова. На кадрі зображення визначалася вибірка таких точок, які формували двомірну матрицю, готову до обробки засобами розпізнання жестів. Система аналізувала матрицю і у разі успіху ідентифікувала ці точки як жест. Зовнішній вигляд кисті руки для систем подібного вигляду продемонстровано на рис. 1.2.

Рис. 1.2. Рука із кольоровими маркерами

На сьогоднішній день, розпізнавання жестів частіше виконується на основі більш розвинених засобів комп'ютерного зору. Зображення об'єкту розгляду (наприклад, руки) може набувати вигляду однотонних силуетів. Проте, сам силует у його початковому вигляді не використовується безпосередньо при розпізнаванні образа руки через складність і велику витрату ресурсів при обробці такого кадру і ідентифікації на кадрі жесту руки. Зазвичай силует обробляють програмними засобами, які забезпечують отримання контурів руки. Такий спосіб представлення інформації кращий, у порівнянні з усім силуетом руки, через те, що системі необхідно опрацьовувати значно меншу кількість елементів. Хоча, для повноцінної роботи системи розпізнавання, цього може бути недостатньо.

Також відомий спосіб розпізнавання жестів на основі скелетної моделі кисті руки (рис. 1.3) [11]. Він представляє собою поєднання попередньо розглянутих способів: отримання жесту руки, з виділенням ключових точок (точок ламаної) у контурах та обчислення центру мас та побудови скелету по цим даним. Скелетна модель надає можливість отримати деякі ключові вузли на кисті руки, а саме: ключові точки пальців та точки, які описують середину долоні руки.

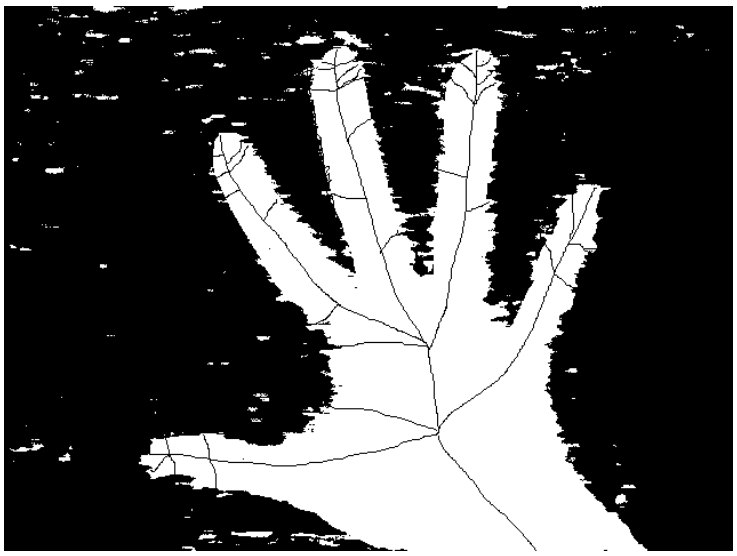

Рис. 1.3. Скелетна модель кисті руки

Після обробки та виявлення контурів руки, зображення передається у **систему розпізнавання жестів**. Розпізнавання – це процес, у якому здійснюється групування та оброблення просторово-часової інформації, отриманої на попередніх етапах і здійснення остаточного висновку – чи є ця інформація жестом, і якщо так, то до якого набору жестів її віднести

(класифікацію жесту). Для цього використовують нейромережеві алгоритми.

Інтерфейс виведення інформації. Через нього система здійснює надсилання інформації отримувачу. Отримувачем може слугувати будь-яка система, який потребує даних від даної інтерактивної системи. На етапі проектування необхідно визначити, який тип даних має надавати нейромережа – чи це буде «класичний» тип даних, наприклад цілочисельне значення, певний символ або значення з плаваючою комою, або користувацький тип даних, такий як структура або клас. Не обов'язково система має повертати цифровий тип даних, це також може бути аналоговий сигнал, як наприклад радіочастота.

Відеоадаптер та графічний дисплей. Дані складові не є обов'язковими у системі. Зазвичай застосовуються для відладки. Під відеоадаптером розуміється програмний блок, який забезпечує отримання інформації від системи розпізнавання жестів, її аналіз, обробку початкового кадру, накладення на нього додаткових графічних елементів та передача зображення на вивід на дисплей.

Елемент управління камерою. Також не є обов'язковим. Залежно від реалізації системи, камера може бути або статично закріпленою, або такою, яка має здатність обертатися. Якщо камера має бути рухомою, необхідно передбачити елемент, що надасть таку можливість. Для цього можна використати альт-азимутальну установку (рис. 1.4).

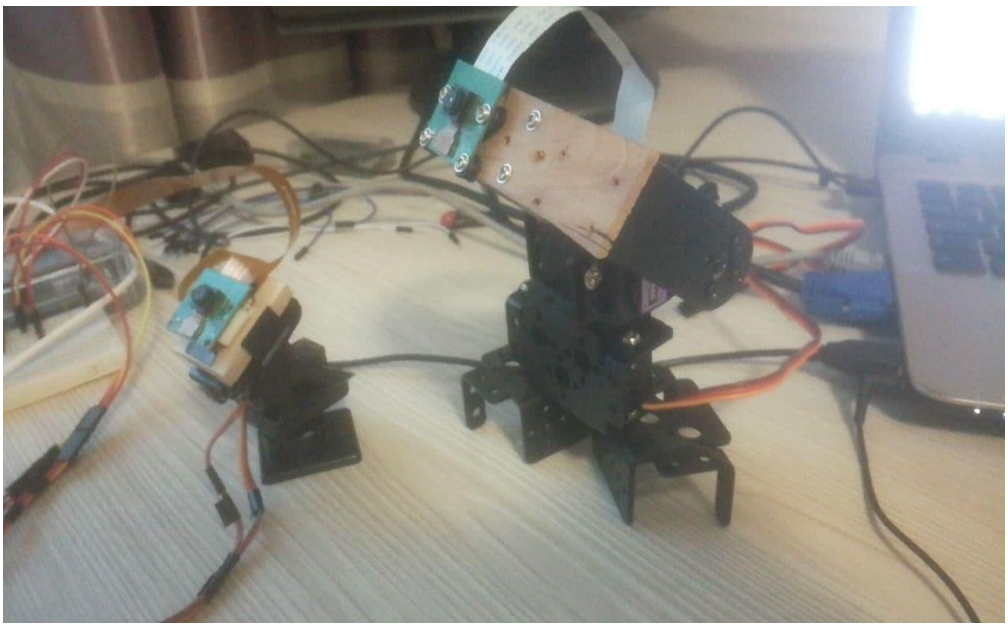

Рис. 1.4. Альт-азимутальні установки із закріпленими камерами

Альт-азимутальна установка – це пристрій, що має вертикальну та горизонтальну осі обертання і завдяки цьому дозволяє спрямувати підвішену на ньому камеру у будь-якому напрямку. Обертання може здійснюватися як вручну так і за допомогою спецзасобів, наприклад сервоприводів.

Розглянемо приклади використання інтерактивних систем розпізнавання жестів у різноманітних проєктах. Наразі поширене явище створення пристроїв, відомих як інтернет речей (англ. *Internet of Things, IoT*). Це система, у якій різноманітні об'єкти здатні бути пов'язаними між собою, мають доступ до виходу у мережу Інтернет (або в локальну мережу), мають різноманітні датчики та сенсори. Спільна робота допомагає у виконанні різноманітного кола задач, з якими людина стискається протягом всього циклу дня. Приклад *IoT* – система «розумний дім». Це система, яка складається з певного набору пристроїв, які розташовуються у будинку і мають виконувати набір дій з списку повсякденних завдань без участі людини. «Розумний дім» може мати набір різноманітних датчиків: датчики освітлення, руху, вологості і т.п.. Одним із таких датчиків може бути інтерактивний модуль керування жестами. Система, буде захоплювати та відстежувати рухи рук господаря та розпізнавати набір жестів, що дозволить керувати технікою у домі, наприклад для збільшення гучності на медіаплеєрі, керування освітленням або увімкненням побутових пристроїв лише одним рухом руки.

Якщо розглядати можливі області застосування подібних інтерактивних систем для керування, варто згадати системи, що впроваджені у мобільних телефонах та мобільних додатках. Зокрема, у них надана можливість захоплення та відстеження рук, розпізнавання та класифікація жестів рук, що надає зможу виконувати широкий спектр дій, як наприклад: здійснення знімку екрана, керування відеозаписом екрана телефона, розблокування пристрою, т.п.

Також прикладом може слугувати система керування жестами у бортових комп'ютерах сучасних автомобілів. За функціоналом вони подібні до мобільних систем, проте у автомобілі система забезпечує керування мультимедією автомобіля. Прості жести, такі як «вказати» або «змахнути» надають можливість викликати певні функції – наприклад, прийняти або відхилити дзвінок.

1.2. Відомості про теорію розпізнавання образів

Виявлення жестів є окремим складним завданням, яке передбачає застосування цілого набору різноманітних операцій та алгоритмів [12].

Теорія розпізнавання образів – це набір теоретичних основ та методик ідентифікації та класифікації предметів, процесів, явищ, за якими закріплена певна вибірка певних специфічних ознак, які допомагають точно ідентифікувати сутність із сукупності інших сутностей. Для цього необхідно виділити набір ознак, притаманних саме цьому об’єкту, та вказати правило або процедуру, яка визначатиме наявність кожної з ознак. Кожен об’єкт має характеристики, які його однозначно ідентифікують: колір, форма, розташування у просторі, певні спеціальні характеристики (наприклад, пальці на долоні однозначно ідентифікують об’єкт як кисть руки) і т.п. Важливо те, що ці характеристики можуть змінюватись і це не має впливати на успіх виявлення об’єкта.

Наприклад, переважна більшість рослин мають зелене забарвлення і це є ключовою ознакою рослин. Також ознакою рослин є наявність листків. Проте, при настанні осені, колір листків змінюється від зеленого до помаранчевого, червоного, а через деякий час рослини взагалі позбуваються листя. Проте це не заважає нам ідентифікувати дані об’єкти як рослини. Отже, колір та листки – це ознаки, а зміна кольору та наявність листків є варіацією цих ознак.

Описати систему виявлення образів (жестів) можна кількома пунктами [13]. Перш за все, необхідно вказати, який саме об’єкт буде підпадати під розпізнавання, і виділити основні ознаки в цих об’єктів. Надалі будується математична модель, яка описуватиме той чи інший об’єкт. Вибирається необхідний алгоритм, або набір алгоритмів, які здійснюватимуть виявлення об’єктів. Здійснюється перетворення початкових даних для успішного опрацювання цих даних вибраним алгоритмом. Виявлення об’єкта зазвичай є процесом «порівняння» певного зразку із частинами зображення, на якому необхідно знайти подібний об’єкт.

Розглядаючи розпізнавання жестів руки, можна сказати, що об’єктом розпізнавання є кисть руки. Ознаки руки – наявність пальців, специфічна форма

долоні та заданий спектр кольорів людської шкіри. За даними ознаками, людина здатна без проблем ідентифікувати кисть руки серед набору інших об’єкті.

Тепер необхідно провести побудову математичної моделі кисті руки. Математичну модель кисті руки можна подати у вигляді структури, яка складається з набору менших підструктур – точок, які описуються кількома параметрами, що описують просторове розміщення цих точок.

Наступним етапом в побудові системи виявлення жестів є побудова математичного алгоритму, який виявлятиме жест. Для вирішення даної задачі дослідниками запропонований вельми великий набір математичних структур та алгоритмів. Серед запропонованих методів – класифікація по формі, яку можна виконувати на основі заданого шаблону або аналітичної формули; по просторовому розміщенню, по кольору та ін. Надалі, необхідно використати складну математичну функцію, яка зважаючи на всі ці критерії (як наприклад, розмір, форма, колір, розміщення об’єкта), могла б видати інтегральний показник, за яким можна зробити висновок щодо розпізнавання об’єкту.

Для побудови таких функцій використовують алгоритми штучного інтелекту, зокрема нейронні мережі.

1.3. Вибір одноплатного комп’ютера

При створенні інтерактивних систем керування жестами, необхідно вирішити питання апаратної частини, тобто на якій електронно-обчислювальній машині буде виконуватися програмний код.

Найбільшого поширення у обчислювальних блоках легких БПЛА набуло використання одноплатних ЕОМ через задовільні характеристики та компактність. Одноплатний ЕОМ – це комплекс технічних засобів, елементи, які призначенні для опрацювання інформації в процесі вирішення обчислювальних задач, розміщенні на одній платі. Подібне розташування елементів ЕОМ надає можливість реалізувати різноманітні автономні системи керування жестами, достатньо компактні, щоб мати змогу розмістити на літальному апаратові [14]. У таблиці 1.1 наведена характеристика деяких ЕОМ.

Таблиця 1.1

Порівняння одноплатних ЕОМ

ЕОМ	Ціна	Процесор	Ядра	ОЗП	Пам’ять
Raspberry Pi 3B	35 $	*Broadcom BCM2837*	4*x A53*, 1.2 *GHz*	1 *Gb*	*MicroSD*
Banana Pi M3	74 $	*Allwinner A83T*	8*x A7*, 2 *GHz*	2 *Gb*	*8 GB eMMC, MicroSD*
Orange Pi Prime	30 $	*Allwinner H5*	4*x A53*	2 *Gb*	*MicroSD*
Libre Computer Renegade	50 $	*ARM Cortex*	4*x A53*	2 *Gb*	*MicroSD*

Дані комп’ютери підтримують операційні *UNIX*-системи, а наявність різноманітних портів, надасть можливість підключати різноманітну переферію і модулі, що забезпечує певну «маневреність» при реалізації власних проєктів.

1.4. Висновки до розділу

Під час написання першого розділу, було наведено пояснення, що саме розуміється під терміном «інтерактивна система керування жестами». Описано загальну структуру такої системи, наведені основні і додаткові функціональні блоки. Для кожного блоку наведений короткий опис, який висвітлює роль даного блоку у роботі усієї системи. Наведені відомості з теорії розпізнавання образів. Описано загальні принципи, за якими здійснюється виділення необхідних об’єктів серед інших об’єктів, наведено критерії, дотримуючись яких, можна здійснювати виявлення. Виявлення об’єктів – це складна процедура, яка відбувається в кілька етапів. Для кожного з етапів наведено короткий опис. Проведене порівняння одноплатних комп’ютерів, на яких можна б було побудувати систему розпізнавання жестів. Обрано для подальшої роботи одноплатну ЕОМ *Raspberry Pi 3B+*.

РОЗДІЛ 2

ПРОЕКТУВАННЯ МОДУЛЯ РОЗПІЗНАВАННЯ ЖЕСТІВ РУК ДЛЯ КЕРУВАННЯ КВАДРОКОПТЕРОМ

2.1. Опис апаратних елементів, які застосовувалися під час проектування модуля

Основою для виконання програмного коду та керування елементами модуля, був обраний одноплатна ЕОМ *Raspberry Pi 3B+* (рис. 2.1) [15].

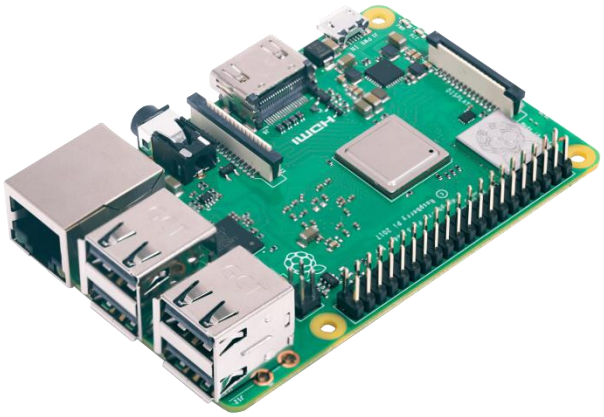

Рис. 2.1. *Raspberry Pi 3B+*

Даний одноплатний комп’ютер відповідає усім необхідним вимогам, які потрібні для успішного проектування та створення автономного модуля, а саме: Компактність, задовільні апаратні можливості, наявність інтерфейсу вводу-виводу загального призначення (*GPIO*), мале споживання електроенергії, архітектура комп’ютера підтримує роботу *UNIX*-подібних операційних систем.

Компактність. Розміри даного одноплатного комп’ютера наступні: довжина 85 *мм*, ширина 56 *мм*, висота 17 *мм*. Задовільні апаратні можливості. Апаратні характеристики даного пристрою задовольняють потреби, які виникають при роботі програмного забезпечення.

Характеристики *Raspberry Pi 3B+*:

– чіп: *Broadcom BCM2837B0;*

– процесор: 64-бітний 4-ядерний *ARMv8 Cortex-A53* 1,4 *ГГц*;

– відеопроцесор: *VideoCore IV 3D*;

– ОЗП: 1 *Гб LPDDR2* (900 *МГц*);

– роз'єми: *HDMI*, 3,5 *мм jack, USB* 2.0 4 шт., роз'єм для *microSD*, 10/100 *Ethernet*, порт для дисплея *DSI*, порт для камери *CSI-2*, *GPIO* 40 *pins*;

– безпроводні інтерфейси: *WiFi* 2,4 *ГГц* і 5 *ГГц IEEE 802.11.b/g/n/ac* 2,4 *ГГц*, *Bluetooth*: 4.2 *Classic*;

– інтерфейс виведення інформації (відеоадаптер та графічний дисплей необов'язкові елементи).

Важливим моментом є наявність інтерфейс вводу-виводу загального призначення (*GPIO*). Даний інтерфейс забезпечую можливість підключення до ЕОМ різноманітних модулей і їх керування, також надає можливість через піни здійснювати надсилання сигналів.

Даний комп'ютер здатний працювати при малому споживанні електроенергії – достатньо напруги в 5 *V* та сили струму в діапазоні 1.5-2 *A*.

Операційна система, вибрана для роботи, є *Raspbian OS* – операційна система, заснована на основі *Debian*.

Сервопривід – це пристрій, що здійснює механічне переміщення регулюючого органу відповідно до сигналів, отримуваних від системи керування. Для забезпечення обертання рухомих частин модуля (альт-азимутальна установка), було вибрано сервоприводи *Tower Pro MG90S* (рис. 2.2).

Рис. 2.2. Сервопривід

Характеристики: робоча напруга 4-6 *V*, пусковий момент 1.8 *кг*см*, кут повороту 180˚, час повороту на 60 ˚ при напрузі 4.8V без навантажень 0.1 с, час повороту на 60˚ при напрузі 6V без навантажень 0.08 с, довжина 32,3 *мм*, ширина 33 *мм*, висота 12,4 *мм*.

Одним із основних елементів модуля є камера, яка здійснює відеозахоплення. Обрано камеру для *Raspberry Pi*, версії 1.3 (рис. 2.3) [16].

Рис. 2.3. Модуль камери для *Raspberry Pi*, версія 1.3

Модуль камери має роздільну здатність матриці в 5 *Mn*, розмір матриці сягає 1/4" або 3,76 × 2,74 *мм*, максимальний розмір зображення, який може бути створений – 2592 × 1944 пікселів. Розміри модуля: довжина: 25 *мм* , ширина: 24 *мм*, висота: 9 *мм*.

Задля забезпечення фіксації і обертання камери, було використано альт-азимутальну установку *Pan/Tilt SG90* (рис. 2.4).

Рис. 2.4. Альт-азимутальна установка

Живлення модуля відбувається за допомогою двох джерел. Для забезпечення роботи одноплатного комп'ютера *Raspberry Pi* та сервоприводів, використовується два зовнішніх акумулятори *Power Bank 3 10000 mAh* (рис. 2.5). Вихідна напруга 5 *V*, сила струму 1.5 *A*, ємність батареї 10000 *mAh*.

Рис. 2.5. Зовнішній акумулятор

2.2. Бібліотеки *Mediapipe* та *OpenCV*

Mediapipe – це кросплатформенна бібліотека рішень задач комп'ютерного зору, на основі використання бібліотеки *OpenCV* та алгоритмів ШІ [17].

OpenCV – бібліотека алгоритмів комп'ютерного зору та алгоритмів для опрацювання зображень та відеопотоків, яка написана на мові *C/C++* [18]. Алгоритми штучного інтелекту, які використовуються у *MediaPipe*, забезпечуються завдяки бібліотеці для машинного навчання *TensorFlow*. Алгоритми *TensorFlow* містять інструменти для створення моделей нейронних мереж і їх навчання.

Робота даної бібліотеки організована у вигляді «конвеєра» або «трубопровода» (*pipeline*) – у систему подаються певні вхідні дані, які проходять через кілька вузлів обробки і користувач отримує на виході результат, залежно від заданих конфігурацій. Таким чином, робочою системою бібліотеки є пайплайн («трубопровід»), який зображується у вигляді графа. Вершинам графа відповідають процедури (методи) обробки даних, їх також називають «калькуляторами». Ребра графа зображують потоки, які передають інформацію у вигляді пакетів. Граф є поєднанням комбінації менших графів (підграфів), які з'єднані потоками. В підграфах здійснюється налаштування вхідних потоків, вихідних потоків та вхідних статичних пакетів. Також підграф може відігравати роль як окремої структурної одиниці в загальному графі.

Після цього всі дані вносяться у один конфігураційний файл з розширенням *.pbtxt*, який зберігає дані про граф. Таким чином, відбувається створення графа і побудова плану, по якому відбувається виконання методів класів (калькуляторів).

Пакет – це одиниця даних, яка курсує від одного калькулятора до іншого, завдяки потокам, які пов'язують калькулятори. Пакет є структурою, яка може містити довільну кількість різноманітних даних. Тип цих даних може бути як користувацьким типом даних (структура, клас), так і одним із типів даних, передбачених у *C/C++* (*bool, int, char, double* і т.п.).

В бібліотеці застосована синхронізація між пакетами задля забезпечення коректної і стабільної роботи системи. Синхронізація у *MediaPipe* досягається шляхом використання у пакетах часових міток і застосування механізму обробки цих міток. Виконання графа *MediaPipe* децентралізоване. Це забезпечує більш високу пропускну здатність за рахунок конвеєрної обробки.

Для того, щоб використати певний калькулятор, необхідно виконати певну послідовність дій. Спочатку необхідно визначити новий підклас – калькулятор, який бажаємо створити. Надалі реалізувати обов'язкові методи класу. Визначити вхідні та вихідні потоки. Провести реєстрацію нового калькулятора.

Для того, щоб створити калькулятор, необхідно створити клас, дочірній класу *CalculatorBase*. Даний клас є батьківським для усіх класів, що реалізовують роботу калькуляторів.

Надалі необхідно провести перевизначення, як мінімум, чотирьох методів: *GetContract(), Open(), Process(), Close()*. Робота даної бібліотеки організована у вигляді «конвеєра» (*pipeline*) – у систему подаються певні вхідні дані, які проходять через кілька вузлів обробки і користувач отримує на виході результат,

Калькулятор буде здійснювати обробку найновіших пакетів, у яких буде співпадати часова мітка із тією міткою, яка приходить з вхідного потоку.

Під час проектування модуля керування жестами, був застосований граф *MediaPipe Hands*. *MediaPipe Hands* забезпечує рішення точного відстеження рук і пальців. Він використовує засоби *OpenCV* та алгоритми машинного навчання, щоб знайти у кадрі 21 ключову точку руки і позначити область, у якій виявлена долоня. Даний метод вирішує проблему виявлення кисті руки з використанням

різних стратегій. Детектором кисті руки є нейромережа, котра пройшла навчання значній кількості даних, ці дані підготовленні засобами *OpenCV*. Таким чином, забезпечується отримання 21 ключової точки на кисті руки (рис. 2.7).

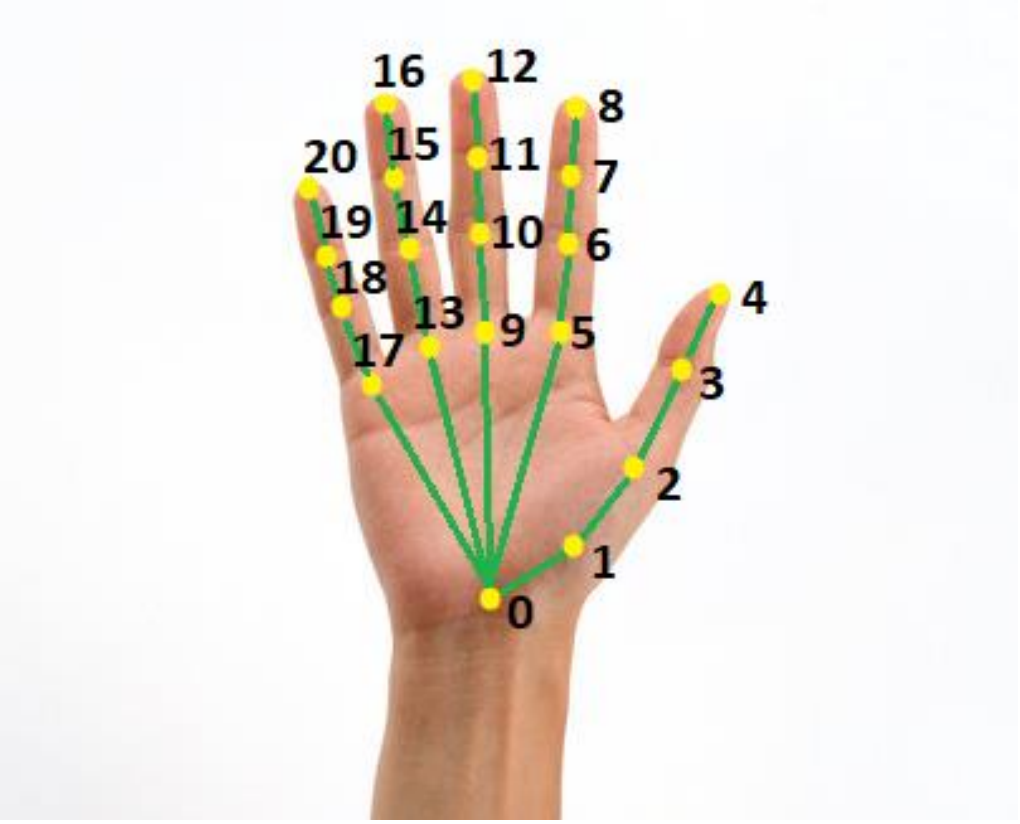

Рис. 2.6. Ключові точки долоні, які виявляються *MediaPipe Hands*

Виявлення відбувається навіть у випадку частково видимих рук або у випадку, якщо одна рука перекриває іншу – завдяки регресії, тобто передбачення координат.

Отже, алгоритм отримує у якості вхідних даних кадр відеопотоку, де ймовірно зображена кисть руки, і у разі, якщо кисть присутня – виявляє її і надає масив з координатами 21 точки, кожна точка описана трьома координатами.

OpenCV – це бібліотека, яка забезпечує можливість використання алгоритмів комп'ютерного зору і деяких інших алгоритмів по обробці зображень. Дана бібліотека написана на мові *C/C++* що надає чудову швидкодійність програмного коду. Вона надає можливість здійснювати інтерпретації зображень, калібровки камери по еталону, ліквідацію шумів на зображені, визначення форми різноманітних об'єктів, здійснювати розділення за кольорами і т.п.

У процесі написання програмного коду, дана бібліотека використовувалася для вирішення низки задач, зокрема обробки зображень, також вона застосовується у роботі раніше згаданої бібліотеки *MediaPipe*. Надалі буде описано вибірку алгоритмів, які були застосовані у проектуванні модуля розпізнавання жестів.

2.3. Розмиття Гауса

Розмиття Гауса – це алгоритм, який здійснює розмиття зображення [19]. Сам алгоритм був названий на честь німецького математика Карла Фрідріха Гауса.

Розмиття є інструментом обробки зображення. Основною метою застосування є зменшення деталізації зображення, що використовується для багатьох алгоритмів обробки зображень, завдяки зменшенню кількості шумів на зображені. На рис. 2.7 можна побачити приклад результату застосування даної функції до зображення:

Рис. 2.7. Розмиття за Гаусом

2.4. Кольорова модель *HSV*

Кольорова модель *HSV* (вимовляється як «ейчесві») – математична модель представлення кольорів зображення, у якій застосовується певна кількість кортежів (упорядкований набір фіксованої довжини) чисел [20]. Модель

представляється через три значення: *Hue* (тон), *Saturation* (насиченість), *Value* (яскравість). На рис. 2.8 демонструється подання даної кольорової моделі у тривимірному вигляді.

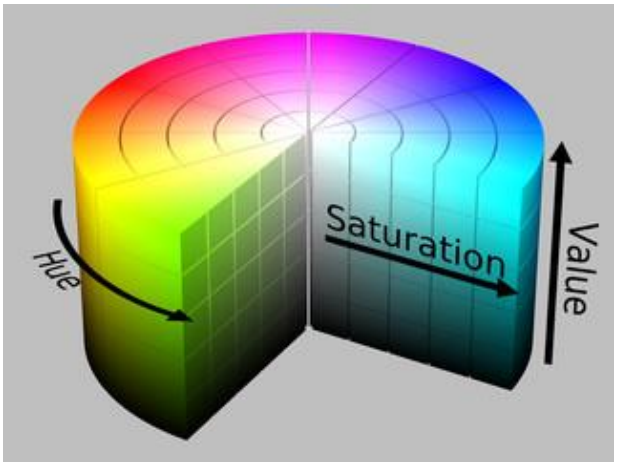

Рис. 2.8. Просторове розміщення *Hue, Saturation, Value*.

Зазвичай компоненти *Saturation, Value* знаходяться у діапазоні від 0 до 100; стандартний діапазон *Hue* від 0 до 360. За необхідності, діапазон може бути масштабований до інших значень.

Дана кольорова модель використовується для попередньої обробки зображень, зокрема для алгоритмів порогових операцій (алгоритми, які дозволяють виділити один певний кольоровий спектр із зображення).

2.5. Порогові операції.

Порогові операції – це методи сегментації заданих, через пороги, кольорів, тобто – виділення певного кольору на зображенні [21]. Якщо розглянути рисунок просторового розташування *HSV* у попередньому розділі (рис. 2.9), то можна побачити, як даний «циліндр» розділений на умовні сегменти. Даний сегмент описується парою координат *Hue, Saturation, Value* («мінімальний» та «максимальний» пороги), які і описують ці сегменти. Сегмент буде містити колір, який необхідно виділити. Після того, як сегмент задано, заданий колір на рисунку набуває білого забарвлення, а все інше – чорного. Зображення стає

бінарним (зображення, яке складається лише з двох кольорів – білого і чорного). На рис. 2.9 продемонстровано приклад виділення жовтого кольору.

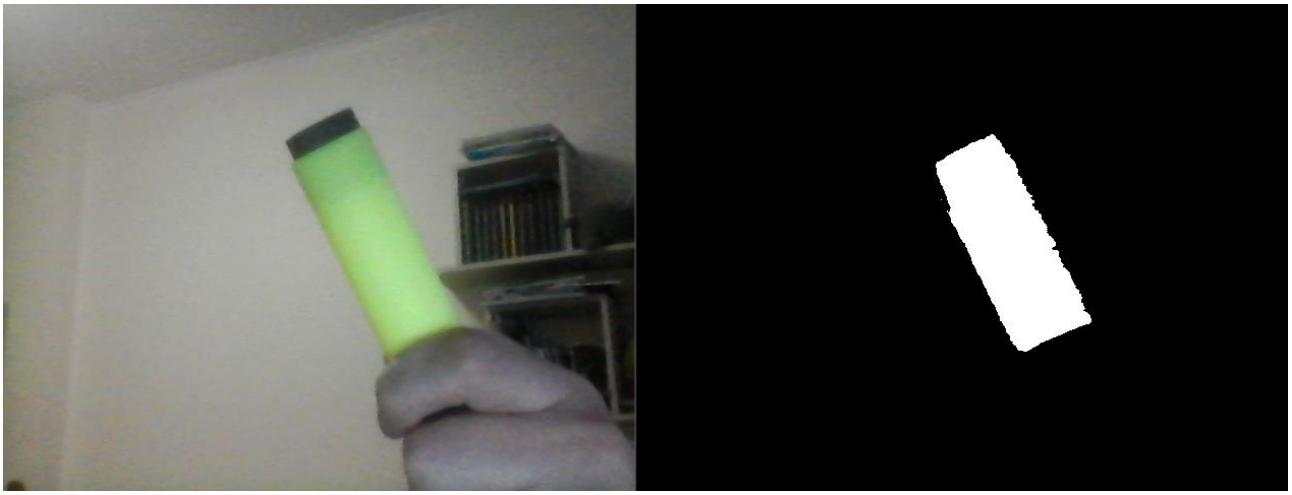

Рис. 2.9. Виділення жовтого кольору

Таким чином, за допомогою заданих порогів, виділяється сегмент, що забезпечує виокремлення потрібного кольору (кольорового спектру).

2.6. Алгоритм виявлення контурів

Алгоритм виявлення контурів, що використовується в *OpenCV*, є алгоритмом японського математика Сатосі Сузукі. Алгоритм відомий як *Suzuki85* [22-23]. Даний алгоритм забезпечує можливість ієрархічних відносин між границями контурів та здатний вказувати зовнішні та внутрішні границі. Загалом, алгоритм працює на основі аналізу пікселів і сусідніх пікселей, до пікселя який наразі розглядається. Обробка пікселей відбувається зверху-вниз і зліва на право. На бінарному зображенні всі чорні пікселі позначаються умовним нулем, білі – одиницею. У ході опрацювання пікселей, можливі два варіанта, які можуть підказати, яка саме границя наразі розглядається – зовнішня чи внутрішня (рис. 2.10).

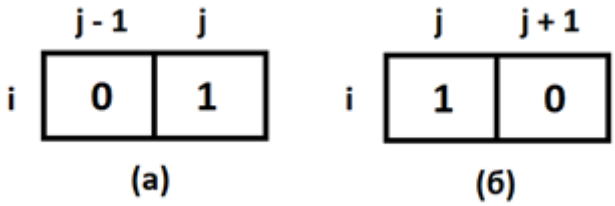

Рис. 2.10. Визначення границі

Якщо перший піксель – нульовий, а наступний є відмінним від нуля, то це повідомляє про виявлення контуру. Якщо поточний піксель позначений як ненульовий а наступний є нульовим, то розглядаємо це як закінчення контуру. Після того, як була досягнута дана ситуація, іде сканування сусідніх пікселів, спочатку за годинниковою стрілкою, потім – проти годинникової. Перевіривши ще низку умов, встановлюємо номери для кожного з контурів. Також необхідно мати змінну-лічильник, яка буде відслідковувати поточну кількість контурів. Таким чином, можна позначити пікселі, які відповідають одним і тим самим умовам, однаковими значеннями. Пікселі, що мають однакові значення, формують окремий контур.

2.7. Алгоритм Рамера-Дугласа-Пекера

При роботі алгоритму по виявленню контурів (наприклад *Suzuki85*) може бути ситуація, коли алгоритм виявить величезну кількість контурів, кожен з контурів буде доволі коротким, а самі контури утворять ламану, яка містить безліч точок (рис. 2.11) [23].

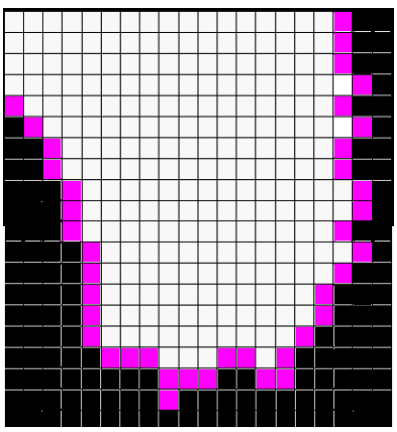

Рис. 2.11. Контур фігури (фіолетовим)

Наявність величезної кількість точок може нашкодити швидкодійності програми, внаслідок збільшення часу опрацювання усіх точок. Для зменшення точок, застосовують алгоритм Рамера-Дугласа-Пекера.

Даний алгоритм здійснює апроксимацію ламаної до ламаної з меншою кількістю точок. Наприклад, якщо застосувати даний алгоритм на контурі з рис. 2.11, то отримаємо більш гладку ламану, зображену на рис. 2.12.

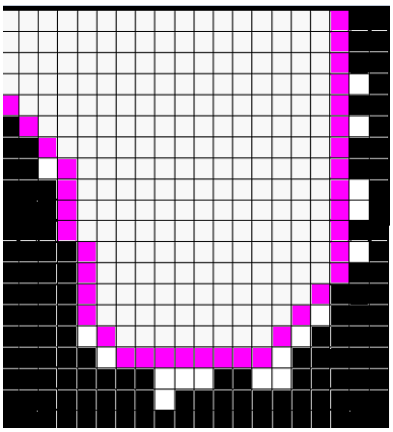

Рис. 2.12. Апроксимований контур

Отриманий контур є ламаною з меншою кількістю точок переломів, що спрощує подальші оперції над контуром. Наша ламана, яку потрібно апроксимувати, розглядається як упорядкована вибірка з точок переломів. Також, задається довільне дійсне число $\varepsilon > 0$. Алгоритм рекурсивно проводить поділ ліній. Чим більше епсілон, тим більш апроксимованим буде контур. Вибирається дві точки (зазвичай початкова і кінцева точки ламаної) і між ними проводиться лінія. Відбувається пошук точки ламаної, яка є найбільш віддалена від даної лінії. Надалі поточні точки сполучаються із цією точкою і дана точка позначається як така, яка буде збережена . Проводиться перевірка відстані між іншими точками, які ще не були задіяними, і проведеними прямими. Якщо відстань між точною і прямою менше за ε, то така точка відкидається. Надалі проводимо нову ітерацію до тих пір, поки будуть оброблені всі точки. Таким чином, отримуємо апроксимовану ламану. На рис. 2.13 показано приклад апроксимації ламаної.

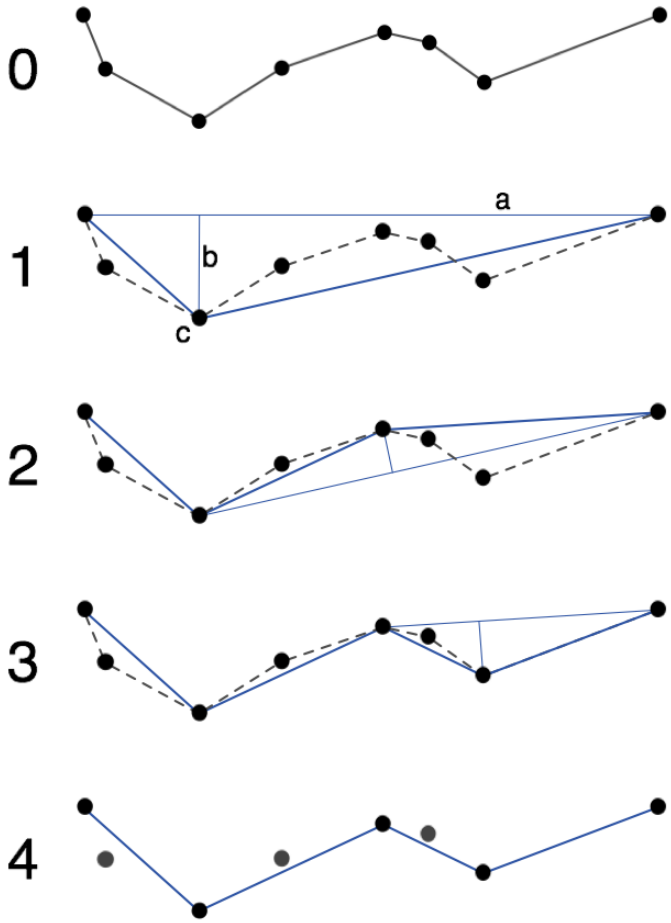

Рис. 2.13. Приклад апроксимація ламаної

2.8. Поєднання роботи алгоритмів

Тепер, оглянувши алгоритми, можна навести приклад їхнього поєднання задля досягнення певних результатів. Наприклад – виявити об’єкт із заданим кольором.

Маємо наступну послідовність дій:

1) Отримуємо зображення з камери у кольоровій моделі *RGB* (у *OpenCV* використовується *BGR*);

2) Застосовуємо Гаусівське розмиття, задля зменшення кількості шумів;

3) Конвертуємо *RGB* в *HSV*-модель;

4) З *HSV*-моделі отримуємо необхідний колір, застосувавши порогові операції;

5) З бінарного зображення отримуємо контури за допомогою аглоритма

виявлення контурів (Сузукі 85);

6) Застосувавши алгоритм Рамера-Дугласа-Пекера, апроксимуємо контур.

Отже, застосувавши алгоритми, отримали замкнутий колір, який позначає на зображенню місцерозташування шуканого об’єкта.

На рис. 2.14 проілюстровано застосування алгоритмів і їх результат.

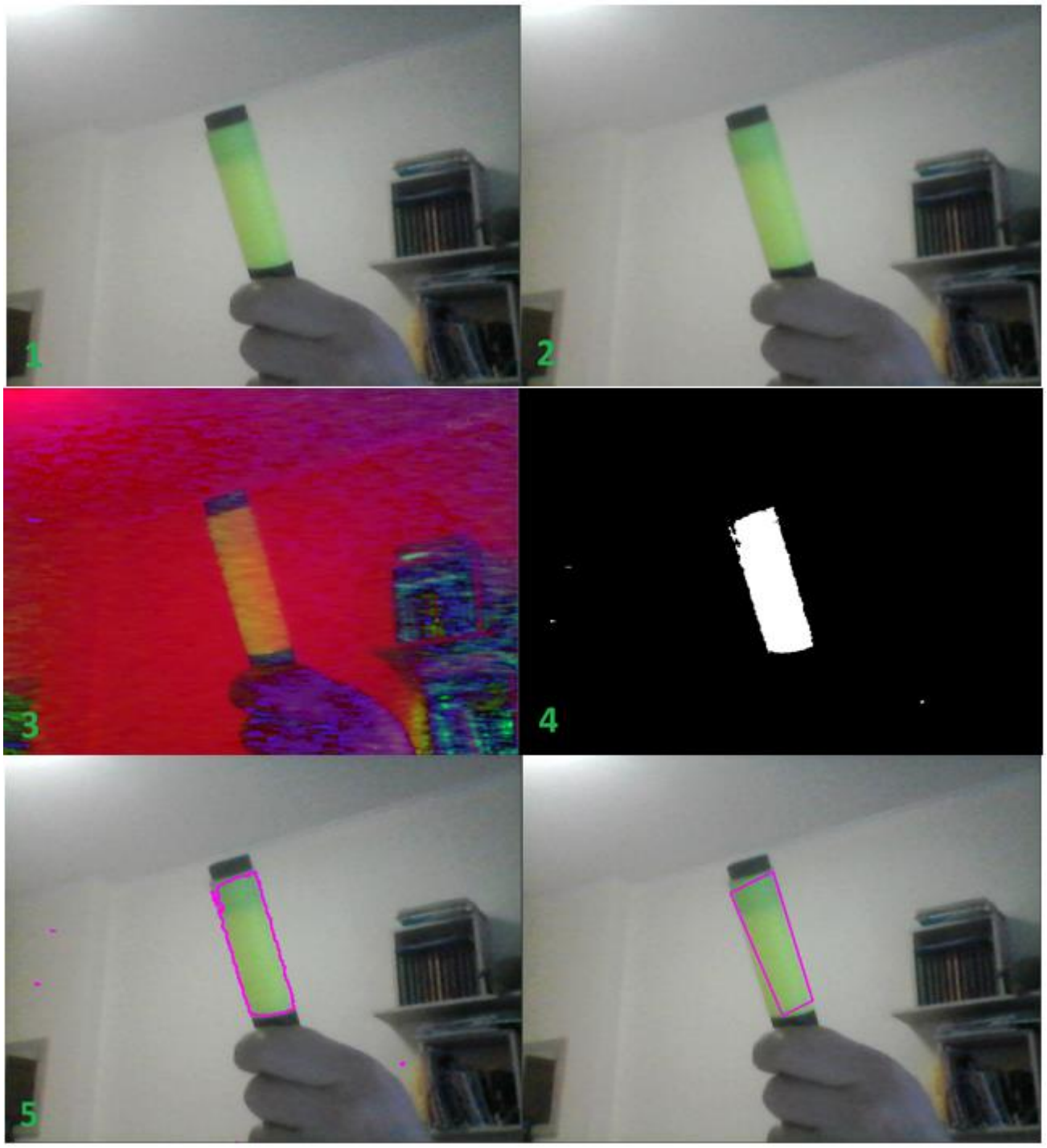

Рис. 2.14. Результати послідовного застосування алгоритмів

2.9. Система збірки *Bazel*

Система збірки *Bazel* – це програмне забезпечення з відкритим кодом, що забезпечує можливість збірки та тестування програм. Має можливість створювати ПЗ для різних платформ. Підтримує наступні мови програмування: *C, C++, Java, Go, Python, Objective-C* и *Bourne shell*-скрипти. Під час розроблення *Bazel*, увага була сфокусована на досягнення прискорення збірки, зокрема застосовуючи розпаралелювання. Також є можливість через даний інструмент підключати код, бібліотеки і інші файли, які знаходяться віддалено, потрібні для компіляції. Основою *Bazel* є робоча область (*WORKSPACE*), конфігураційні файли збірки (*BUILD*) та цілі (*targets*).

WORKSPACE – файл, який має бути в кореневій папці проекту. В даному файлі можна задати глобальні налаштування.

BUILD – кожна окрема папка повинна мати свій власний *BUILD*-файл. Тут прописуються всі налаштування конфігурацій, підключення файлів з програмним кодом, бібліотек і т.п. і вказуються інші конфігураційні файли *BUILD*.

Target – цілі, або правила, які підлягають до виконання. Через них вказуються дії, які необхідно виконати у *BUILD*-файлах.

2.10. Бібліотеки *WiringPi* та *softPwm*

Бібліотеки *WiringPi* та *softPwm* застосовуються для забезпечення зручної роботи з *GPIO* на *Raspberry Pi. GPIO* – це універсальний інтерфейс вводу-виводу, що надає можливість обмінюватися інформацією із різноманітною апаратурою, завдяки змінні напрузі на пінах. *GPIO* у *Raspberry Pi* представляє 40 пінів різного призначення, через які можна подавати, приймати інформацію, подавати постійну напругу 5 *V* або використовувати піни для заземлення (рис. 2.15).

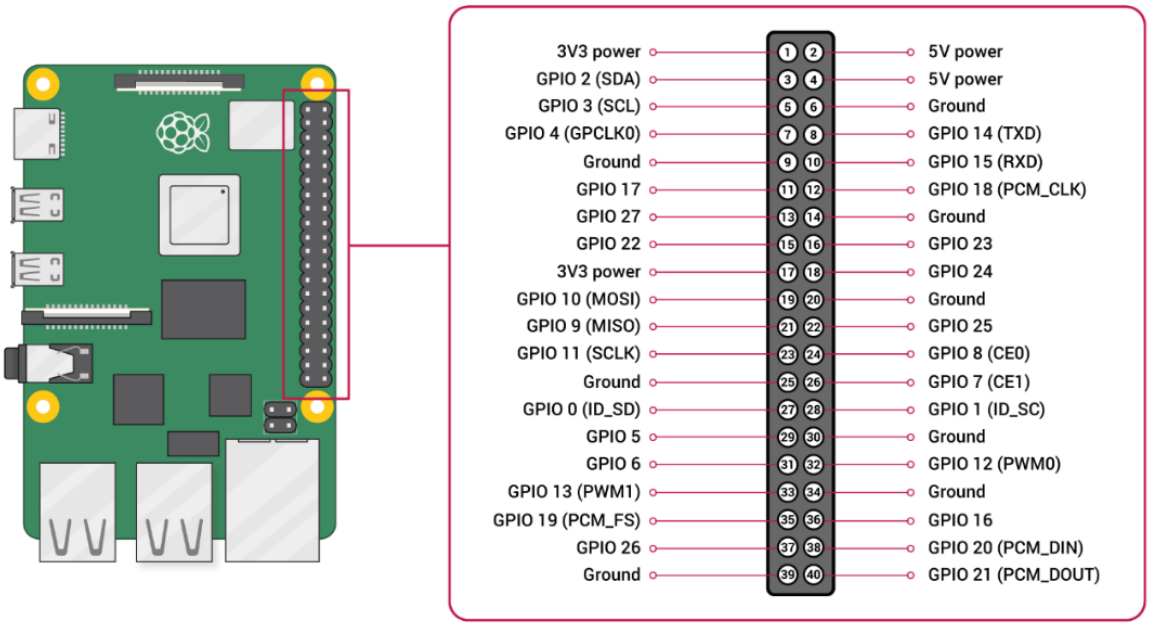

Рис. 2.15. *GPIO Raspberry Pi*

Через те, що безпосередня робота з пінами *Raspberry Pi* є доволі важкою, через необхідність вдаватися до низькорівневого програмування, функції з бібліотек *WiringPi* та *softPwm* надають легкий спосіб використання цих пінів [25]. *WiringPi* - це бібліотека керування за допомогою *GPIO*, написана на мові C для пристроїв *Raspberry Pi*. Дана бібліотека розроблялася виключно для керування загальним інтерфейсом вводу-виводу на *Raspberry Pi*. Функції у бібліотеці наближені до функцій, які використовуються для написання програмного коду для апаратно-програмного пристрою *Arduino* на мові *C/C++*. *softPwm* – бібліотека, яка використовує бібліотеку *WiringPi* для додавання певних можливостей, зокрема для реалізації цифрової *PWM*-модуляції (ШІМ-модуляція). Широтно-імпульсна модуляція – процес керування шириною (тривалістю) високочастотних імпульсів за законом, який задає низькочастотний сигнал. Використовується для керування сервоприводів.

Принцип роботи сервопривода на основі ШІМ – подання імпульсів певною тривалістю. В залежності від тривалості імпульсу, буде задаватися поворот сервопривода. На рис. 2.16 наочно показано, як від тривалості залежить кут повороту сервопривода.

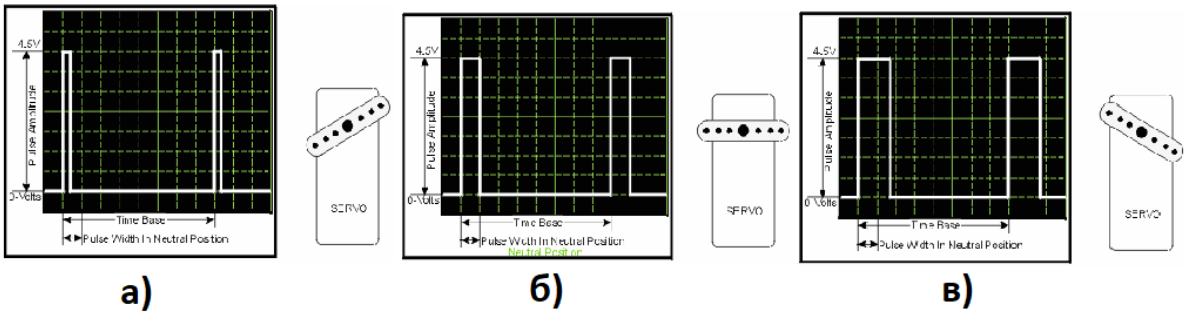

Рис. 2.16. Керування сервоприводом за допомогою ШІМ

Як бачимо, залежно від тривалості сигналу, поворот буде здійснено уліво (варіант а) – тривалість менше тривалості «нейтральної позиції»), управо (варіант в) – тривалість більше тривалості «нейтральної позиції»), або прийняти положення без поворотів (варіант б) – тривалість рівна тривалості «нейтральної позиції»).

2.11 Бібліотека *JSON*

Бібліотека *JSON* (вимовляється як «джейсон») – це бібліотека, яка надає текстовий формат обміну даними між різними ЕОМ. Окрім обміну, бібліотека надає можливості формувати файли для зберігання упорядкованої інформації і проводити читання таких файлів. Таким чином, її використання вимагається саме при записі та читанню файлів із структоризованим вмістом.

Під час виконання дипломного проекту, бібліотеку *JSON* використано для збереження даних, а саме: налаштувань нейромережі і навчальних даних.

2.12. Висновки до розділу

У другому розділі розглянуто ряд використаних технологій, які застосовувалися під час реалізації предмета дослідження дипломного проєкту, наданий короткий опис, а саме:

– розмиття Гауса, що забезпечує зменшення шумів;

– конвертація у кольорову модель HSV, для того, щоб надалі мати змогу застосувати порогові операції;

– порогові операції, які забезпечують виділення заданого кольору на зображенні;

– алгоритм виділення контурів, а саме алгоритм Сузукі 85, який здійснює виокремлення контурів у бінарному зображенні, сформованому пороговими операціями;

– алгоритм Рамера-Дугласа-Пекера, котрий здійснює апроксимацію контурів, задля зменшення точок контурів, що надасть зменшити витрати часу на оброблення точок контурів.

Окрім застосованих технологій і алгоритмів, розділ містить дані про обрані апаратні елементи, які будуть використані на етапі розробки модуля по керуванню жестами рук.

Був наданий перелік апаратних елементів, які будуть використовуватися у процесі реалізації модуля по керуванню жестами, а саме:

– одноплатний комп'ютер *Raspberry Pi*;

– сервоприводи;

– модуль камери;

– альт-азимутальна установка;

– живлення.

Висвітлено процес обробки зображень, що в майбутньому використовується для виявлення об'єктів (жестів). Наданий детальний опис теоретичних відомостей по бібліотекам *MediaPipe* та *OpenCV*. Надано перелік алгоритмів і методів по обробці зображень, а саме: розмивання Гауса, кольорова модель *HSV*, порогові операції, алгоритм виявлення контурів, алгоритм Рамера-Дугласа-Пекера. Для кожного з алгоритмів наведений їх короткий опис і укінці наданий приклад їх застосування для виявлення об'єктів. Наведений опис системи збірки Bazel, яка використовуватиметься під час компіляції і збірки програмного забезпечення. Наданий опис бібліотек *WiringPi* та *softPwm*, які забезпечують можливість простої взаємодії із загальним інтерфейсом вводу/виводу *GPIO*. Описано призначення бібліотеки *JSON*.

РОЗДІЛ 3
ПРОГРАМНА ТА АПАРАТНА РЕАЛІЗАЦІЯ МОДУЛЯ

3.1. Реалізація концепції модуля розпізнавання жестів

На початку робіт, по фізичному втіленню предмета дипломної роботи, а саме модуля розпізнавання жестів рук для керування квадрокоптером, необхідно було продумати загальну концепцію всієї системи. Модуль має відповідати наступним критеріям:

– компактний розмір;

– автономність роботи;

– рухомий елемент, який забезпечуватиме достатній огляд для можливості розпізнання жесту;

– можливість кріплення модуля на коптер;

– інтерфейс, який забезпечить передачу даних від модуля до коптера.

Враховуючи дані вимоги, і загальну схему систем розпізнавання жестів (Розділ 1, рис. 1.1) було обрано елементи для побудови (Розділ 2) а саме:

– Raspberry Pi 3B+;

– 2 сервоприводи *Tower Pro MG90S* ;

– альт-азимутальна установка *Pan/Tilt SG90*;

– модуль камери *Raspberry Pi*;

– портативний акумулятор *Power Bank 3 10000 mAh*;

– роз’єм для живлення *Breadboard Power Supply 5/3.3 V (1 A)* ;

– макетна плата.

На рис 3.1 подано схему поєднання обраних елементів.

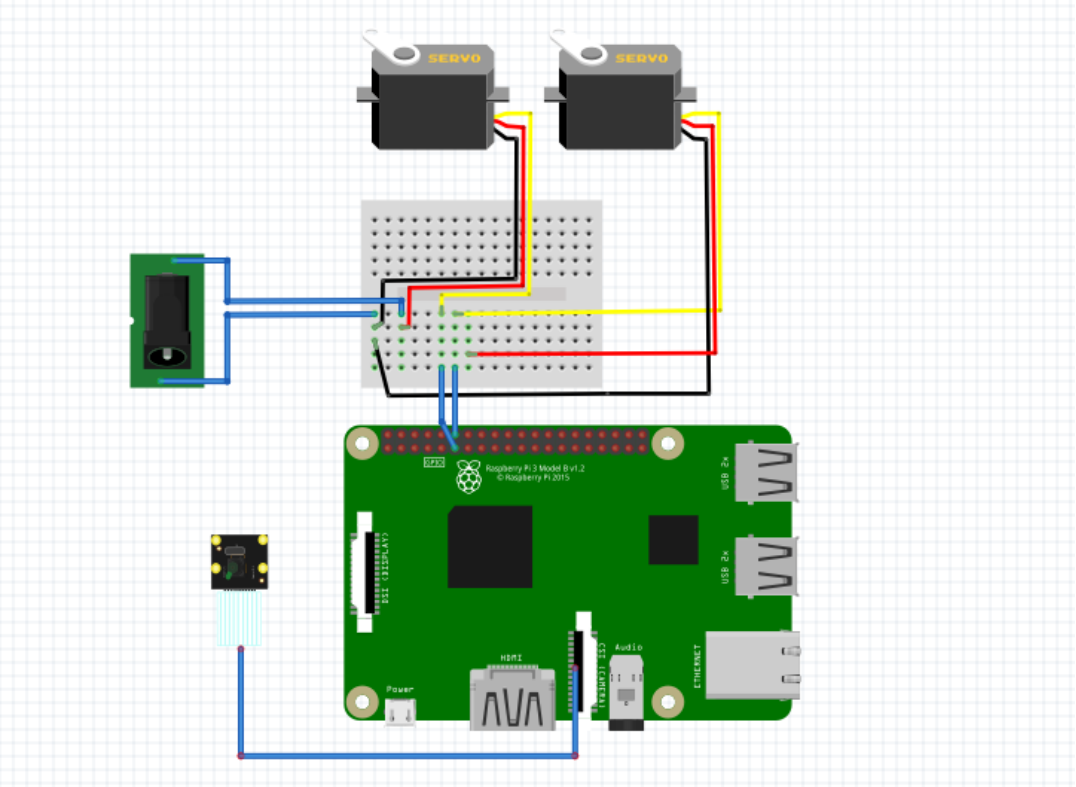

Рис. 3.1. Схема підключення компонентів модуля розпізнавання жестів

Після придбання усіх необхідних компонентів здійснено фізичну збірку модуля (рис. 3.2).

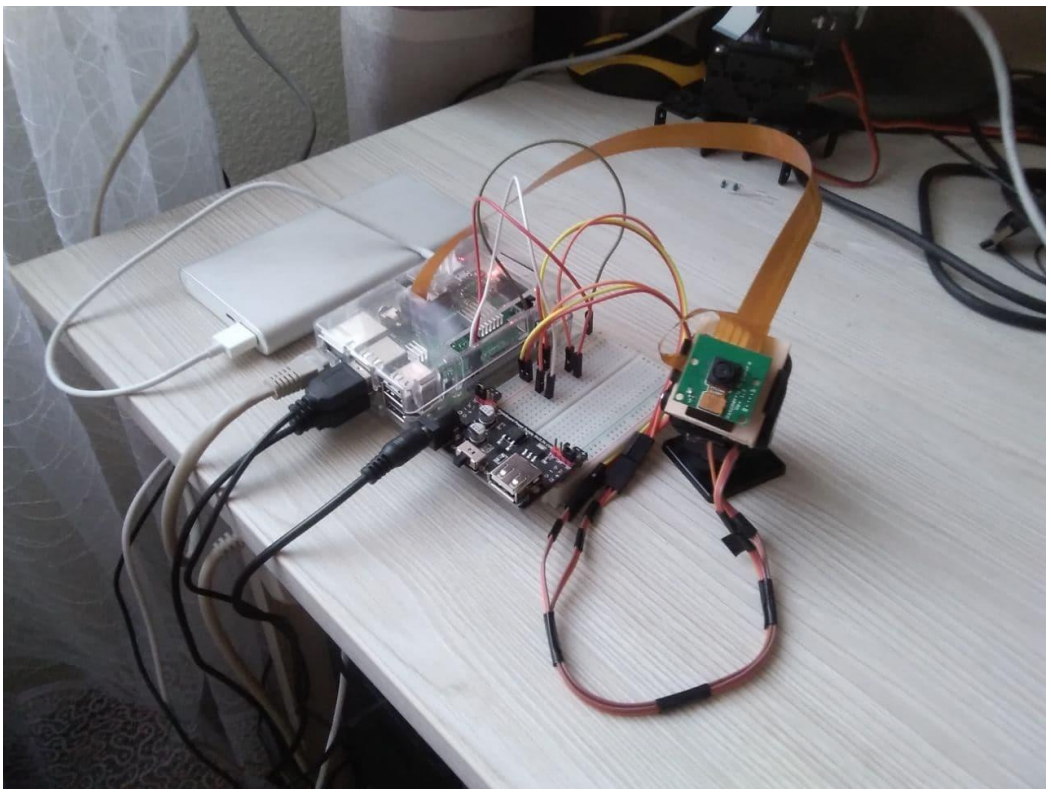

Рис. 3.2. Фізична реалізація модуля

Розміщення апаратного модуля на квадрокоптері показано на ескізі (рис. 3.3).

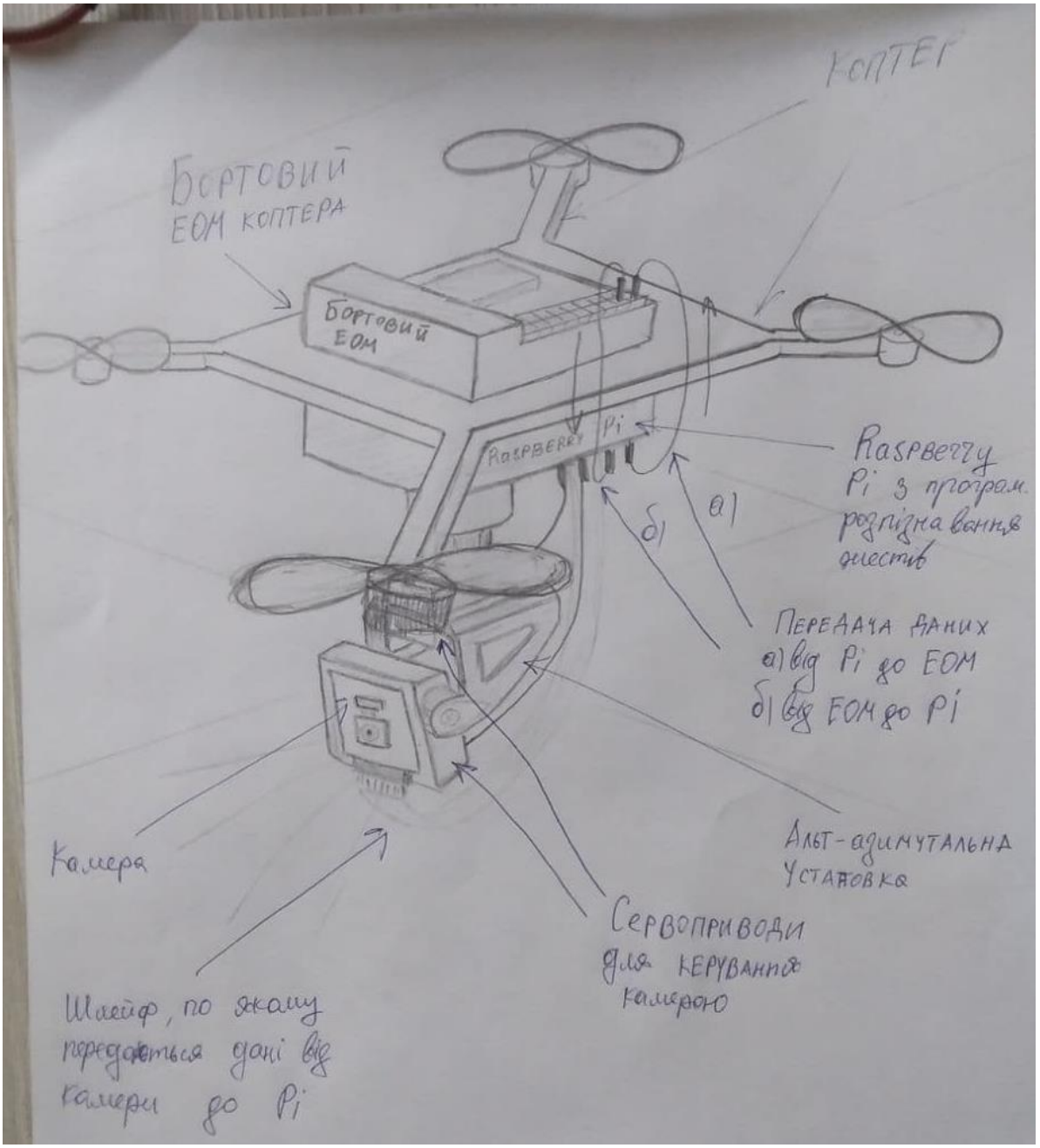

Рис. 3.3. Ескіз поєднання модуля і квадрокоптера

За допомогою системи *AutoCAD* була створена *3D* модель квадрокоптера у поєднані з модулем розпізнавання жестів (рис. 3.4).

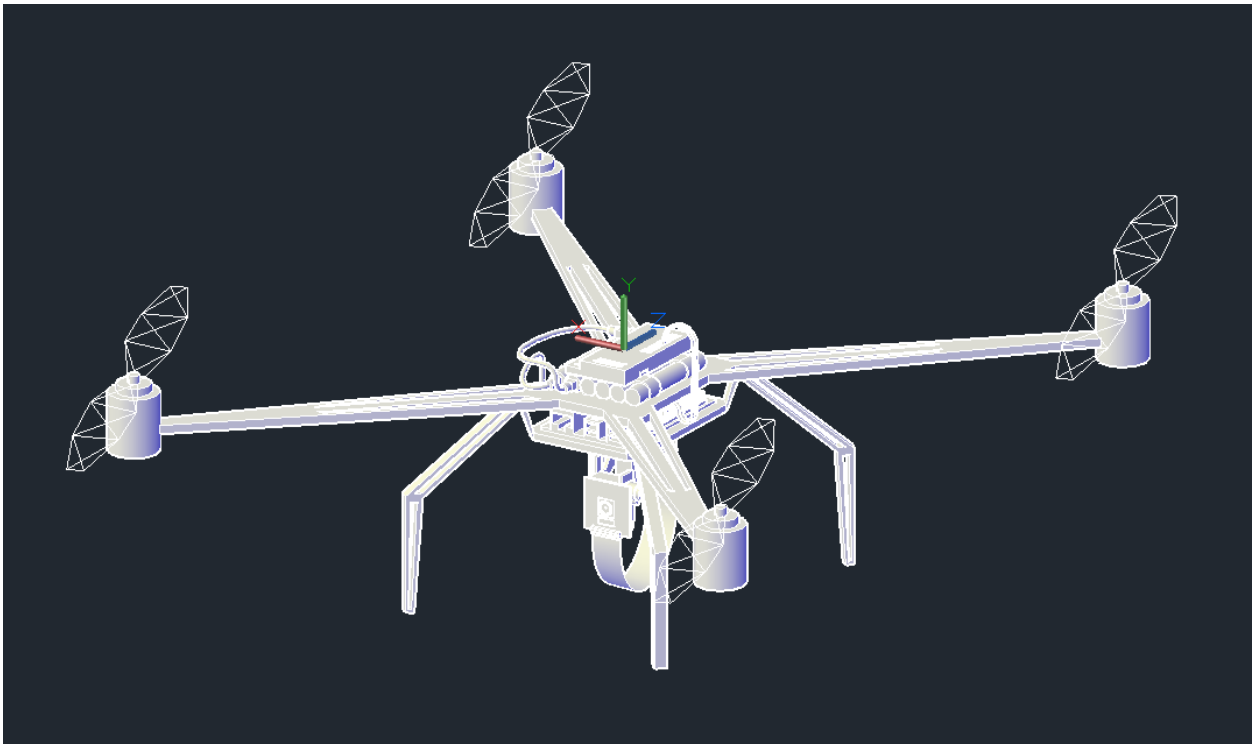

Рис. 3.4. Модель у *AutoCAD*

Наступним етапом є створення програмного забезпечення, яке надасть можливість здійснювати функції по розпізнаванню жестів і надання необхідних команд квадрокоптеру.

Дане ПЗ має виконувати перелік вимог: здійснювати керування сервоприводами для керування альт-азимутальної установкою, реалізовувати процес отримання кадру з відеопотоку і здійснювати попередню обробку, реалізовувати процес виявлення жестів, реалізовувати процес розпізнавання жестів, надавати інформацію на бортовий комп'ютер квадрокоптера.

Була створена діаграма прецедентів (відображена у обов'язковому графічному матеріалі.). На основі діаграми прецедентів, був створений програмний код (опис реалізації окремих класів у підрозділі 3.2) у якому реалізовані класи, що надають можливість виявляти жест, розпізнавати жест, і відносно нього здійснювати ту чи іншу команду і клас, що забезпечує обертання камери. Враховуючи написаний програмний код, була реалізована діаграма послідовностей (відображена у обов'язковому графічному матеріалі.). На основі програмного коду, для легшого розуміння структури програми, була реалізована діаграми класів (відображена у обов'язковому графічному матеріалі.).

Загальний план роботи програми виглядає наступним чином. Після запуску програми через термінал з одним із аргументів: "*default*" (або не вказуючи ніяких параметрів), "*learning*", "*make_data*", виконується один із сценаріїв програми.

Сценарій при запуску з аргументом "*default*" (або без аргументів). Програма запускається у звичайному режимі. Створюється об'єкти класів *HandTracking* та *AltAzimuth* і запускається основний метод *processTracking()*, у якому відбувається основна логіка програми. Паралельно, в окремому процесі відбувається виконання методу *mainLoop()*, об'єкта класа *AltAzimuth*, який забезпечує обертання альт-азимутальної установки. Поки не буде виявлена кисть, альт-азимутальна установка буде проводити сканування, у разі виявлення кисті – буде фокусувати камеру на кисті руки. Під час виконання методу *processTracking()*, здійснюються наступні дії:

1) Об'єкт класу *NeuralNetwork* (нейромережа) здійснює завантаження конфігураційних даних;

2) Створюється об'єкт класу *MediaPipeHandsDetector*, що забезпечуватиме роботу по виявленню жестів

3) Розпочинається робота циклу, який працюватиме до моменту, поки працює програма (або до аварійного завершення);

4) Отримуємо зображення з камери;

5) Передаємо зображення об'єкту класу *MediaPipeHandsDetector* на аналіз кисті руки. Якщо кисть була виявленна, вектор ключових точок кисті руки передається на обробку у нейромережу, яка аналізує дані і виявляє логіку жеста. У разі успіху, передає сигнал на пін;

6) Якщо жест був виявлений, здійснюється обробка кадру для подальшого знаходження центра мас кисті руки;

7) Відбувається обчислення центра мас кисті руки і відносно положенню центра мас на зображенні, здійснюється корегування положення камери;

8) Здійснюється відображення обробленого кадру зображення (якщо підключений дисплей);

9) Відбувається наступна ітерація циклу.

Сценарій при запуску з аргументом "*learning*". Створюється об’єкт класу *HandTracking* і запускається метод *processLearning*. Під час роботи методу *processLearning*() здійснюється навчання нейромережі.

Сценарій при запуску з аргументом "*make_data*" (або без аргументів). Створюється об’єкти класів *HandTracking* та *AltAzimuth* і запускається метод, який забезпечує створення навчальних даних. Паралельно, в окремому процесі відбувається виконання методу *mainLoop()*, об’єкта класа *AltAzimuth*, який забезпечує обертання альт-азимутальної установки. Поки не буде виявлена кисть, альт-азимутальна установка буде проводити сканування, у разі виявлення кисті – буде фокусувати камеру на кисті руки. Після захоплення кисті руки оператор має натиснути одну з клавіш 0-8, що відповідає одному із дев’яти жестів; таким чином здійснюється формування навчальних даних. На рис. 3.5 відображені приклади захоплення руки.

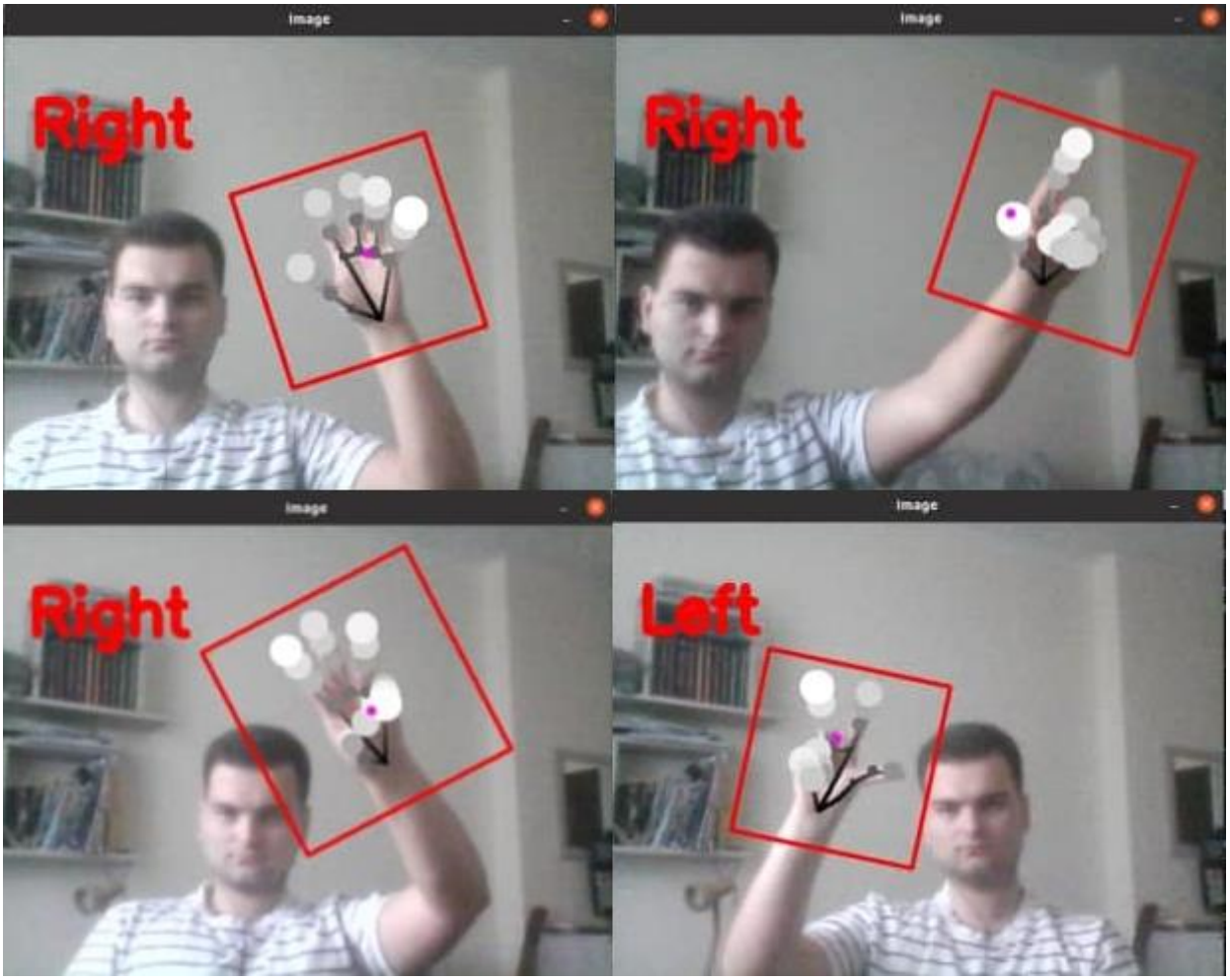

Рис. 3.5. Приклади захоплення кисті руки

3.2. Програмна реалізація модуля

3.2.1. Клас *AltAzimuth*

Загальний опис класу *AltAzimuth.*

Клас *AltAzimuth* реалізовує логіку обертання альт-азимутальної установки, на яку вмонтована камера, яка забезпечує надання системі зображень. В загальному, в класі реалізовано два основних метода, які спрацьовують залежно від обставин. На початку роботи модуля, спрацьовує метод пошуку кисті руки (*void searching()*), який здійснює сканування області. Сканування області відбувається шляхом обертання камери горизонтально у межах 0˚ - 180˚. Вертикально, наведення альт-азимутальної установки здійснюється у межах 0˚ - 165˚. Якщо кисть була виявлена, розпочнеться виконання методу відстеження (*void tracking()*). Даний метод працює на основі місцерозташування центру мас точок кисті руки, відносно центру зображення. Надаючи загальний опис даному методу, обертання альт-азимутальної установки здійснюється наступним чином: якщо центр мас знаходиться правіше від центру зображення, камеру необхідно повернути ліворуч, і навпаки, якщо центр мас знаходиться лівіше від центру зображення – необхідно здійснити обертання праворуч. Подібним чином, логіка задана і для вертикального обертання – якщо центр мас вище – опускаємо камеру, якщо нижче – камера піднімається. Якщо центр мас знаходиться доволі близько від центру зображення (не далі відстані, що відповідає 15% половини висоти або ширини зображення.

Поля класу *AltAzimuth*.

bool isWork – змінна, яка відстежує роботи системи. Якщо значення змінної істино, то робота альт-азимутальної установки продовжується, інакше – припинити роботу.

cv::Point point – змінна, яка зберігає координати центра мас кисті руки. Це клас бібліотеки *OpenCV*, містить дві змінні *int x* та *int y*. Якщо дані змінні рівні –1, отже кисть не виявленна.

cv::Point centre – координати центра зображення.

int width, height – змінні зберігають ширину і висоту зображення відповідно.

int calibrationW, calibrationH - відстанні, які будуть викоритовуватися при заданні зони у якій, при знаходжені *point*, сервоприводи не будуть рухатися (15% від половини ширини і довжини зображення).

int pin0, pin1 – змінні зберігають номера пінів на *GPIO Raspberry Pi*, через які відбувається модуляція ШІМ (у роботі використані піни 24 та 23 відповідно).

int range – задає діапазон для ШІМ (у роботі використано значення 100).

int value – задає частоту для ШІМ (у роботі використано значення 50).

Методи класу *AltAzimuth*.

void tracking(int delayQuantity = 50) – даний метод реалізовує логіку обертання камери, для здійснення пошуку кисті руки. Здійснюється др того моменту, поки *isWork* є істиною і значення змінних *x* та *y* змінної *point* рівні –1. Аргумент *delayQuantity* задає швидкість обертання камери.

void searching(int delayQuantity = 150) – метод, що реалізовує обертання альт-азимутальної установки, що імітує процес відстеження кисті руки. Як було раніше сказано, логіка методу побудована на основі обчислення положення цетру мас кисті руки відносно центра кадра зображення. Аргумент *delayQuantity* задає швидкість обертання камери.

void setPositionX(int delayQuantity) – здійснює обертання альт-азимутальної установки у вертикальній площині. Аргумент *delayQuantity* задає швидкість обертання камери.

void setPositionY(int delayQuantity) – здійснює обертання альт-азимутальної установки у горизонтальній площині. Аргумент *delayQuantity* задає швидкість обертання камери.

AltAzimuth(bool &isWork, cv::Point &point, int width = 640, int height = 480, pin0 = 23, pin1 = 24, range = 100, value = 50) – конструктор класу. Також проводить налаштування *GPIO* за допомогою бібліотек *wiringPi* та *softPwm*.

void mainLoop() – головний метод класу. Залежно від умов, активує метод пошуку кисті руки, або метод відстеження.

virtual ~AltAzimuth() – деструктор класу.

3.2.2. Клас *HandTracking*

Загальний опис класу *HandTracking*.

Клас *HandTracking* є головним класом у даному програмному забезпеченні. Він об’єднує у собі логіку всіх інших класів (окрім класу *AltAzimuth*, з яким зв’язаний опосередковано, надаючи дані про наявність кисті руки і центр мас). Даний клас може здійснювати три сценарії, залежно від аргументу, який задається у терміналі при запуску програми.

Якщо не було задано ніякого аргументу або задано аргумент «*default*», то розпочинається звичайне виконання роботи програми. Розпочинає виконання метод *processTracking()*. Спочатку здійснюється завантаження конфігураційних даних нейромережі і відбувається її ініціалізація. Створюється об’єкт класу *MediaPipeHandsDetector*, який відповідає за процес виявлення кисті руки і передачі даних у нейромережу на аналіз. Надалі відбувається входження програми у цикл, який завершується при умові, якщо змінна *isWork* набуде хибного значення. Ітерація циклу виглядає наступним чином:

– отримуємо зображення із камери;

– дане зображення перевертається по діагоналі (розташування альт-азимутальної установки передбачається перевернутим);

– у разі, якщо ніяке зображення не було отримано, це сигналізує закінчення роботи. Змінні *isWork* присвоюється значення *false* і ітерація циклу переривається оператором break;

– здійснюємо виклик методу *RunMPPGraph(cv::Mat &camera_frame_raw)* об’єкту класа *MediaPipeHandsDetector*. Даний метод здійснює обробку зображення і аналіз його на наявність кисті руки. У разі виявлення кисті руки, дані передаються у нейромережу на розпізнання жесту;

– якщо жест виявлено, здійснюємо обробку кадра зображення, для отримання центру мас кисті руки;

– проводимо обчислення центру мас кисті руки і передачу координат у об’єкт класа *AltAzimuth*;

– якщо жест не виявлено, у об’єкт класа *AltAzimuth* передаємо значення координат центру мас рівні –1 (що означає, що кисть не виявлена);

– на кінець ітерації, відображаємо оброблений кадр зображення.

Якщо був переданий аргумент «*learning*», то розпочинається процес навчання нейромережі. Розпочинає виконання метод *processLearning()*. Здійснюється виклик функції *study(std::string type)*, який розпочинає процес навчання нейромережі. Після закінчення навчання, нейромережа зберігає дані.

Якщо був переданий аргумент «*make_data*», то розпочинається процес створення навчальних даних для майбутнього навчання нейромережі. Робота схожа на роботу програми при запуску з аргументом «*default*», проте нейромережа неактивна і при натиснені клавіш цифр 0-8, здійснюється зберігання даних кисті руки з міткою відповідного жеста.

Поля класу *HandTracking*.

bool isWork – змінна, яка відстежує роботи системи. Якщо значення змінної істино, то робота класу продовжується, інакше – припинити роботу.

bool isDetected – змінна, яка приймає значення істини, у разі, якщо кисть руки була виявлена, інакше змінна приймає значення хибності.

cv::VideoCapture capture – змінна типу даних з бібліотеки *OpenCV* для захоплення відеопотоку.

bool grab_frames – змінна позначає стан відеопотоку (*capture*). Якщо істина – процес захоплення відеопотоку працює, інакше – завершити процес захоплення відеопотоку.

cv::Mat camera_frame_raw – змінна зберігає кадр відеопотоку. *cv::Mat* – тип даних бібліотеки *OpenCV* і собою представляє матрицю, комірки якого описують зображення у певній кольоровій моделі (*RGB, HSV*).

cv::Scalar lower{0,250,250} та *cv::Scalar upper{5,255,255}* – змінні задають кольоровий спектр для порогових операцій.

cv::Mat frameBlur, cv::Mat frameHSV, cv::Mat frameMask – буферні змінні для обробки зображень.

std::vector <std::vector<cv::Point>> contours – змінна зберігає контури, які отримано після застосування алгоритма Сузукі85.

std::vector <std::vector<cv::Point>> contoursPoly – змінна зберігає апроксимовані контури.

cv::Point point – змінна зберігає координати центра мас кисті руки.

std::vector<int> hiddenLayers – вектор, що містить кількість персоптронів у прихованих шарах. Кількість елементів *hiddenLayers* позначає кількість прихованих шарів.

int inputPerceptrons – зберігає кількість перцептронів у вхідному шарі.

int outputPerceptrons – зберігає кількість перцептронів у вихідному шарі.

*NeuralNetwork *neuralNetwork* – вказівник на клас *NeuralNetwork*. На даний вказівник здійснюється динамічне виділення пам’яті, що забезпечує створення нейромережі.

Методи класу *HandTracking*.

void calculateCenterOfMass() – даний метод здйснює обчислення центру мас кисті руки, у разі її виявлення. Обчислення центру мас – це обчислення середнього арифметичного для координат *x* та *y* усіх ключових точок кисті руки. Після підрахунку координат центру мас, по координатам рисується на зображені невелике рожеве коло, що візуально позначає центр мас на зображенні.Формат *JPEG*.

void frameProcessing() – метод здійснює обробку зображення для виявлення центру мас кисті руки. У процесі роботи метода застосовуються такі алгоритми, як: розмиття Гауса, конвертація у іншу кольорову модель (*HSV*), порогові операції, алгоритм знаходження контурів (Сузукі85), апроксимація контурів за алгоритмом Рамера-Дугласа-Пекера.

HandTracking(int inputPerceptrons, int outputPerceptrons, std::vector<int> hiddenLayers) – конструктор класу *HandTracking*. У даному конструкторі відбувається створення об’єкту класа *NeuralNetwork* і його ініціалізація. Зокрема вхідні аргумени задаються кількість перцептронів у вхідному та вихідному шару нейромережі, також задається кількість прихованих шарів і їхня кількість перцептронів.

~HandTracking() – деструктор класа *HandTracking*.

void processTracking() – основний метод класа. Містить у собі виклики

ключових методів об’єктів класів і задає логіку роботи даного програмного забезпечення.

void processLearning() – метод, що реалізує процес навчання нейромережі і зберігання результату навчання.

bool &getIsWork() – метод повертає посилання на змінну *isWork*.

cv::Point &getPoint() – повертає посилання на змінну *point*.

3.2.3. Клас *MediaPipeHandsDetector*

Загальний опис класу *MediaPipeHandsDetector*.

Клас *MediaPipeHandsDetector* реалізовує логіку виявлення жестів за допомогою засобів бібліотеки *MediaPipe*. Принцип роботи бібліотеки, зокрема логіка виявлення кисті руки, висвітлено у Розділі 2. Загалом, принцип роботи даного класу полягає в наступному:

– ініціалізація графу;

– створення захоплювачів, через які буде здійснено отримання *Hand Landmarks*, кількість виявлених долонь (у програмному коді бібліотеки був змінений параметр, який відповідав за кількість одночасно виявлених кисть рук на зображені до однієї), обробленого зображення із графічними мітками ключових точок кисті руки;

– передача кадру зображення у граф і здійснення аналізу даного зображення графом на наявність кисті руки;

– отримання за допомогою захоплювачів необхідних даних;

– передача вектора з ключовими точками у нейромережу.

Поля класу *MediaPipeHandsDetector*.

bool isDetected – змінна, яка приймає значення істини, у разі, якщо кисть руки була виявлена, інакше змінна приймає значення хибності.

mediapipe::CalculatorGraph graph – граф *MediaPipe*.

std::unique_ptr<mediapipe::OutputStreamPoller> outputVideoPoller – захоплювач, через який здійснюється отримання обробленого зображення із графічними мітками ключових точок кисті руки на ньому.

std::unique_ptr<mediapipe::OutputStreamPoller> outputLandmarksPoller – захоплювач, через який здійснюється отримання ключових точок кисті руки.

std::unique_ptr<mediapipe::OutputStreamPoller> outputHandCountPoller – захоплювач, через який здійснюється отримання кількості виявлених кисть рук.

std::string calculator_graph_config_contents - Змінна зберігає шлях до конфігураційного файлу (**.pbtxt*).

mediapipe::CalculatorGraphConfig config - Змінна зберігає конфігураційні дані.

sizeframe_timestamp_us_t – змінна, що зберігає поточну часову мітку.

Методи класу *MediaPipeHandsDetector*.

absl::Status Setup(std::string calculator_graph_config_file) – метод здійснює встановленя деяких конфігурацій для роботи графа.

absl::Status Shutdown() – метод для коректного завершення роботи об'єкта класу *MediaPipeHandsDetector*.

MediaPipeHandsDetector(bool isDetected, std::string calculator_graph_config_file = "mediapipe/graphs/hand_tracking/custom_hand_tracking_desktop_live.pbtxt") - конструктор класа *MediaPipeHandsDetector*. Здійснює встановлення змінної *isDetected* і виклик метода *Setup()*.

absl::Status RunMPPGraph(cv::Mat &camera_frame_raw) – основний метод, що здійснює опрацювання кадру відеопотоку і надає можливість отримати необхідну інформацію у вигляді вектора ключових точок кисті руки. Повертає *absl::StatusOk*, що є свідченням успішного виконання метода.

3.2.4. Класи *NeuralNetwork* та *Perceptrons*

Загальний опис класу *Perceptrons*.

Даний клас є допоміжним для *NeuralNetwork* і реалізовує шар перцептронів [26]. У тілі класу надано визначення структури *perceptron*, яка реалізовує окремий перцептрон. Клас *Perceptrons* містить у собі вектор, тип даних якого є *perceptron*. Даний вектор є втіленням шару перцептронів.

Поля класу *Perceptrons*.

struct perceptron – структура, що реалізовує окремий перцептрон. Містить наступні змінні: *double value* – значення перцептрону, *double d_value* – значення помилки перцептрону, *std::vector<double> weights* – вектор ваг, які ідуть від перцептрона, *std::vector<double> d_weights* – вектор помилок ваг, які ідуть від перцептрона, *double derivative* – буферна змінна, яка використовується для обчислень.

std::vector<perceptron> perceptronsLayer – вектор, який реалізовує шар перцептронів.

perceptron bias – здвиг.

Методи класу *Perceptrons*.

Perceptrons(int numbOfPerceptrons, int numbOfWeights) – конструктор даного класу. Здійснює ініціалізацію шару пецептронів, в залежності вхідних аргументів.

~Perceptrons() – деструктор класу *Perceptrons*.

int getQuantityOfPerceptrons() – метод повертає кількість перцептронів у шару.

int getQuantityOfWeights() - метод повертає кількість ваг, що ідуть від будь-якого перцептрону у даному шару.

Загальний опис класу *NeuralNetwork*.

Клас *NeuralNetwork* здійснює реалізацію нейронної мережі на основі алгоритму зворотного поширення помилки. Детальний опис алгоритму наведено у другому розділі. Метод *void study(std::string type)* здійснює процес навчання.

Поля класу *NeuralNetwork*.

int countLayers - зберігає кількість слоїв у нейромережі.

std::vector<Perceptrons> layers* - масив вказівників на слої нейромережі.

std::string typeOfData – задає тип жесту.

int numbOfMotion – задає номер жесту.

std::vector<double> data - вектор, що містить дані для вхідного шару нейромережі.

std::vector<double> arr - Буферна змінна, для зберігання значень вихідного шару.

Методи класу *NeuralNetwork*.

NeuralNetwork(int inputPerceptrons, int outputPerceptrons, std::vector<int> hiddenLayers) – конструктор класу *NeuralNetwork.* За допомогою вхідних аргументів, задається кількість перцептронів на вхідному та вихідному шарах, кількість прихованих перцептронів і кількість перцептронів у кожному із прихованих шарів.

~NeuralNetwork() – деструктор класу *NeuralNetworlk.*

void setInputPerceptrons(std::string type, std::vector<double> &data) - функція встановлює значення на вхідному шарові персептрона. *vector<double> data* - вектор, що містить значення вхідного слою персептронів.

void setPrimaryPerseptrons() - метод виставляє початкові значення для значень персептронів.

void setPrimaryWeights() - метод виставляє початкові значення для ваг персептронів.

double sigm(double x) – метод, що реалізовує сигмоїду.

double d_sigm(double x) - метод, що реалізовує похідну сигмоїди.

void calculatePerceptrons(int begin, int end, int layer) - метод обраховує персептрони наступного слою.

void calculateNeuralNetwork() - метод обраховує значення нейромережі.

void setTrueAnswer(int numbPerceptron) - метод виставляє в останьому слої нейромережі в змінних *d_x* "правильні значення" персептронів. Потім відбувається обрахування *d_x = x - d_x.*

void backPropagationBias(int layer) - обчислення помилки для здвигу за методом ЗПП.

void backPropagationWeights(int layer) - обчислення помилки для ваг за методом ЗПП.

void weightsDifference(double coefficient = 0.02) – додає до ваг знайдені «посилки» (відбувається процес коренування ваг). *double coefficient* – задає швидкість навчання нейромережі.

void backPropagationPerceptrons(int layer) – метод, що реалізовує розповсбдження помилки на перцептрони.

void backPropagation() - реалізація методу ЗПП.

double levelOfEducation() – функція визначає рівень навчиності нейромережі. Чим значення *double*, що повертає функція, ближче до нуля, тим більше навчина нейромережа.

bool saveWeights(std::string path) – збереження даних нейромережі, а саме ваг.

bool loadWeights(std::string path, int numbOfLayer) - завантаження даних нейромережі, а саме ваг.

void loadData(std::string type) – метод реалізує завантаження параметрів нейромережі, а саме кількість шарів нейромережі і кількість перцептронів у цих шарах.

void study(std::string type) – один із основних методів нейромережі. Здійснює виклик вибірки методів, що забезпечує повний цикл навчання за алгоритмом ЗПП.

void saveTrainData(std::string type , int numbOfFolder, const std::vector<double> &data) – метод реалізовує логіку збереження навчальних даних по конкретному жесту.

void loadTrainData(std::string type, int numbOfFolder, int numbOfFile, std::string &type_out, int &numb_out, std::vector<double> &data_out) - метод реалізовує логіку завантаження навчальних даних по конкретному жесту.

3.3. Інструкція по експлуатації

Початок експлуатації модуля не є важким. Перш за все, необхідно переконатися у тому, що *Raspberry Pi* увімкнута. Після цього, необхідно відкрити термінал і перейти у теку, в якій знаходиться папка з проектом і зайти у дану папку.

Після цього запустити команду:

bazel-bin/mediapipe/examples/desktop/controle_module/controle_module <аргумент>

де замість <*аргумент*> необхідно підставити один із наступних аргументів командного рядка:

– *default* – запускає модуль у звичайному режимі роботи;

– *learning* – запускає процес навчання нейромережі;

– *make_data* – запускає процес формування навчальних даних. Для того, щоб створити навчальний жест, необхідно надати кисті руки необхідного вигляд необхідного жесту, і натиснути клавіші 0-8 на клавіатури, що позначає необхідний жест.

У разі, якщо не було надано ніяких аргументів, або аргумент був прописаний невірно, програма запускається у режимі *default.*

3.4. Висновки до розділу

У даному розділі було надано детальний опис процесу реалізації предмету дипломного дослідження, а саме створення модуля розпізнавання жестів рук для керування квадрокоптером.

Було проведено роботу по висвітленню всього процесу побудови модуля. можливість керувати підключеними до нього модулями. На основі одноплатного комп’ютера була проведена збірка модуля розпізнавання жестів, у яку увійшли камера, альт-азимутальна установка, сервоприводи та елементи живлення.

Після того, як була створена апаратна частина модуля, розпочалася реалізація програмного забезпечення. Для забезпечення роботи програмного забезпечення, були обдумані функції, які має виконувати дане ПЗ. На основі даних функції, була створена діаграма прецедентів, опираючись на яку, розпочалася робота над програмним кодом. Після написання коду, була проведена робота по забезпеченню кращого розуміння коду, тому були створенні діаграми послідовності та класів.

Таким чином, був реалізований предмет дипломного проектування. Окрім цього, даний модуль був створений таким чином, щоб в майбутньому була можливість проводити його модернізацію і розширювати можливості системи.

ВИСНОВКИ

Під час виконання дипломного проекту було створено модуль розпізнавання жестів рук для керування квадрокоптером. Розробка є інтерактивною системою розпізнавання жестів, має апаратну та програмну частини. Модуль виконує необхідні задачі, а саме:

– здійснює захоплення кадрів відеопотоку та забезпечує обертання відеокамери на основі руху альт-азимутальної установки;

– здійснює виявлення кисті руки;

– здійснює процес розпізнавання жестів;

– має можливість виводити оброблені кадри відеопотоку на графічний дисплей.

Вся робота по реалізації модуля розпізнавання жестів проводилася у кілька етапів, які забезпечили послідовне та структуроване виконання поставленої задачі:

1. Було наведено пояснення, що саме розуміється під терміном «інтерактивна система керування жестами»;

2. Здійснено вичерпне опрацювання теоретичного матеріалу по темі дипломного проекту. Була розглянута низка технічної літератури по комп’ютерному зору та штучному інтелекту, а саме опрацьовано і вивчено матеріал, який стосується нейронних мереж. Крім технічної літератури, проведене ознайомлення із науковими статтями, які стосуються теми дипломної роботи;

3. Надано опис алгоритмів та методик, які використовуються при проєктуванні модуля розпізнавання жестів. Висвітлено відомості з теорії розпізнавання образів, відомості про загальний принцип виявлення об’єктів та поясненно, якими інструментами та методами досягається результат такої роботи. Викладені загальні положення про нейронні мережі, більш детально розглянуто алгоритм зворотного поширення помилки (ЗПП), який використовувався при реалізації нейронної мережі. Розглянутні поширені алгоритми, які використовуються у системах керування жестами, які також

застосовуються під час виявлення об’єктів;

4. Проведена робота по ознайомленню з апаратними засобами і елементами, які використані у процесі реалізації модуля розпізнавання жестів або могли б бути використані. Оглянуту технічні характеристики даних апаратних засобів;

5. Була проведена робота по збірці апаратної частини модуля розпізнавання жестів. Сам модуль збирався з апаратних елементів, таких як: одноплатна ЕОМ *Raspberry Pi 3B+*, сервоприводи, альт-азимутальна установка, камера та блок живлення. Проведені роботи по перевірці працездатності даної системи;

6. Було здійснено написання програмного коду, що надавав можливість здійснювати необхідну логіку модуля, яка відповідає меті даної дипломної роботи. Для забезпечення якісної роботи програмного забезпечення, були обдумані функції, які має виконувати дане ПЗ. На основі даних функцій, була створена діаграма прецедентів, опираючись на яку, розпочалася робота над програмним кодом. Була проведена робота по забезпеченню кращого розуміння коду, тому були створенні діаграми послідовності та класів. У ході написання програмного коду, були виконані роботи по перевірці даного програмного забезпечення;

7. Були проведені перевірку роботи програмного коду на базі зібраної апаратної системи.

Передбачається подальша можливість здійснювати модернізацію розробленої системи, заради розширення функціоналу і можливостей системи.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розанов Ю.К. Электрические и электронные аппараты. Силовые электронные аппараты: учебник для студентов высших учебных заведений. – Издательский центр «Академия». – 2010. – 412 с.

2. Шехтман М.Б. Тенденции развития систем промышленной автоматизации и новые требования к подготовке специалистов по автоматизации [Електронний ресурс]. – 2008. – Режим доступу: *https://www.krug2000.ru/publications/451.html*

3. Мыльник А.В.Тенденции развития автоматизации в промышленном производстве и эффективность ее этапов автоматизации [Електронний ресурс]. – 2008. – Режим доступу: *https://cyberleninka.ru/article/n/tendentsii-razvitiya-avtomatizatsii-v-promyshlennom-proizvodstve-i-effektivnost-ee-etapov/viewer*

4. Богданов А.А., Жиденко И.Т. Создание и исследование робототехнической системы с интерактивным управлением [Електронний ресурс]. – 2012. – Режим доступу: *https://cyberleninka.ru/article/n/sozdanie-i-issledovanie-robototehnicheskoy-sistemy-s-interaktivnym-upravleniem/viewer*

5. Нюнькин К.М. Система распознавания жестов [Електронний ресурс]. – 2004. – Режим доступу: *https://cyberleninka.ru/article/n/sistema-raspoznavaniya-zhestov/viewer*

6. Васильев В.И. Распознающие системы. Справочник //«Наукова думка» . – Киев. – 1983. – 425 с.

7. Девятков В.В., Алфимцев А.Н. Распознавание манипулятивных жестов [Електронний ресурс]. – 2007. – Режим доступу: *https://cyberleninka.ru/article/n/raspoznavanie-manipulyativnyh-zhestov/viewer*

8. Степурко К.В., Абдугалимова Е.Г. Реализация интерактивных систем распознавание жестов [Електронний ресурс]. – 2014. – Режим доступу: *https://cyberleninka.ru/article/n/raspoznavanie-zhestov-na-osnove-neyrosetevogo-podhoda/viewer*

9. Характеристики видеокамер [Електронний ресурс]. – 2016. – Режим доступу: *https://lantorg.com/article/kak-vybrat-videokameru-ch2-harakteristiki-videokamer*

10. *Pitching a Baseball — Tracking High-Speed Motion with Multi-Exposure Images* [Електронний ресурс]. – 2004. – Режим доступу: *http://resources.mpi-inf.mpg.de/VirtualHumans/publ/sig2004.pdf*

11. Носов А.В. Алгоритм распознавание жестов рук на основе скелетной модели кисти руки [Електронний ресурс]. – 2014. – Режим доступу: *https://cyberleninka.ru/article/n/algoritm-raspoznavaniya-zhestov-ruk-na-osnove-skeletnoy-modeli-kisti-ruki/viewer*

12. Вапник В.Н. Теория распознавания образов. Статические проблемы обучения / В. Н. Вапник. – Москва: «Наука», 1974. – 283 с.

13. Фарсайт Д.А. Компьютерное зрение: современный подход / Д. А. Фарсайт, Ж. В. Понс. – Вильямс: «Нива», 2004. – С. 261-273.

14. Альтернативы *Raspberry Pi* [Електронний ресурс]. – 2019. – Режим доступу: *https://habr.com/ru/post/457666/*

15. *Raspberry Pi 3 Model B* [Електронний ресурс]. – Режим доступу: *https://www.raspberrypi.org/ products/raspberry-pi-3-model-b-plus/*

16. *Camera Module* [Електронний ресурс]. – Режим доступу: *https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/*

17. *MediaPipe* [Електронний ресурс]. – Режим доступу: *https://mediapipe.dev/*

18. *OpenCV* [Електронний ресурс]. – Режим доступу: *https://docs.opencv.org/master/index.html*

19. *Computer Vision*: *Algorithms and Applications 2nd* [Електронний ресурс]. – 2021. – Режим доступу: *https://www.dropbox.com/s/45wg0fkblicrm3g/Szeliski BookDraft_20210607.pdf?dl=0*

20. Цветовая модель *HSV* [Електронний ресурс]. – Режим доступу: *https://studme.org/43365/ informatika/tsvetovaya_model_hsv*

21. *Thresholding Operations using inRange* [Електронний ресурс]. – Режим доступу: *https://docs.opencv.org/ 3.4/da/d97/tutorial_threshold_inRange.html*

22. *S. Suzuki. Topological structural analysis of digitized binary images by border following// Computer Vision, Graphics, and Image Processing. – 1985. – No. 30(1). – Pp. 32-46.*

23. *Suzuki contour algorithm* [Електронний ресурс]. – Режим доступу: *https://theailearner.com/tag/ suzuki-contour-algorithm-opencv/*

24. *Douglas–Peucker algorithm* [Електронний ресурс]. – Режим доступу: *https://cartographyplayground.gitlab.io/playgrounds/douglas-peucker-algorithm/*

25. *WiringPi* [Електронний ресурс]. – Режим доступу: *http://wiringpi.com/*

26. Розенблатт Ф. Принципы нейродинамики. Перцептроны и теория механизмов мозга / Ф. Розенблатт. – Москва: «Мир», 1965. – 480 с.

27. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: НАУ. – 2017. – 63 с.

28. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення / Держстандарт України. – Вид. офіц. – [Чинний від 1995-02- 23]. – Київ, 2007. – 86с.