НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Радіотехнічний факультет

Кафедра радіоконструювання та виробництва радіоапаратури

До захисту допущено:
В.о. зав.кафедрою
Євгеній НЕЛІН
«___»_____ 20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інтелектуальні технології мікросистемної радіоелектронної техніки»

за спеціальністю172 «Телекомунікації та радіотехніка»

на тему: «Пристрій комп`ютерного зору»

Виконав:
студент IV курсу, групи РІ-71

Швець Денис Олегович
Прізвище, ім'я, по батькові
Керівник:
доцент, к.т.н. Тарабаров Сергій Борисович
Посада, науковий ступінь, вчене звання,
Прізвище, ім'я, по батькові

Рецензент:

доцент, к.т.н. каф.РОС Сушко Ірина Олександрівна
Посада, науковий ступінь, вчене звання,
Прізвище, ім'я, по батькові

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент

Київ – 2021 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Радіотехнічний факультет

Кафедра радіоконструювання та виробництва радіоапаратури

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 172 «Телекомунікації та радіотехніка»

Освітньо-професійна програма «Інтелектуальні технології мікросистемної радіоелектронної техніки»

ЗАТВЕРДЖУЮ

В.о.зав. кафедрою

Євгеній НЕЛІН

«_12_»____06_____2021 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Швецю Денису Олеговичу

1. Тема проєкту «Пристрій комп`ютерного зору», керівник проєкту Тарабаров Сергій Борисович, доцент, к.т.н. , затверджені наказом по університету від «18»травня 2021 р. №1205-с

2. Термін подання студентом проєкту 09 червня 2021 року

3. Вихідні дані до проєкту: Мультибіометричний термінал контролю доступу ZKTeco SpeedFace-V5L(TD); Hive UVF – система управления доступом автомобильного транспорта на базе IP-камеры; Создание приложений с помощью Mediapipe – доступу - https://habr.com/ru/post/502440/

4. Зміст пояснювальної записки: аналіз ТЗ, огляд аналогів, функціональна схема та характеристика елементів, програмне забезпечення, розробка макета та тестування.

5. Перелік графічного матеріалу: Схема електрична принципова, Передня кришка, Нижня кришка, Верхня кришка, Захисна кришка, Кільце 1, Кільце 2, Кільце 3, Складальний кресленик.

6. Дата видачі завдання 12 квітня 2021 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Розробка ТЗ	12.05.2021	
2	Аналіз ТЗ	16.05.2021	
3	Огляд аналогів	20.05.2021	
4	Характеристики елементів	24.05.2021	
5	Програмне забезпечення	30.05.2021	
6	Розробка макету та тестування	04.05.2021	
7	Розробка графічної документації	09.06.2021	

Студент Денис ШВЕЦЬ

Керівник Сергій ТАРАБАРОВ

АНОТАЦІЯ

Дипломний проект на тему «Пристрій комп`ютерного зору» виконано на 63 сторінках, що включають 23 ілюстрації, 2 таблиці та 5 додатків.

Метою даного проекту є розробка пристрою комп'ютерного зору, для приймання та аналізу візуальних команд від людини. Також пристрій повинен мати можливість приймати рішення та контактувати з іншими пристроями.

На основі аналізу існуючих технологій виготовлено пристрій та написана необхідна кодова база.

Ключові слова: raspberry , mediapipe, OpenCV, зображення, розпізнавання, сервер, акумулятор.

ANNOTATION

Thesis project on "Computer Vision Device" is made on 63 pages, including 23 illustrations, 2 tables and 5 appendices.

The aim of this project is to develop a computer vision device for receiving and analyzing visual commands from humans. The device must also be able to make decisions and communicate with other devices.

Based on the analysis of existing technologies, the device is made and the necessary code base is written.

Keywords: raspberry, mediapipe, OpenCV, image, recognition, server, battery.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту

на тему: «Пристрій комп`ютерного зору»

Київ — 2021 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ЦП — Центральний процесор

ПК — Персональний комп’ютер

В — Вольт

А — Ампер

г — грам

Гб — Гігабайт

к/с — кадри в секунду

Мп — Мегапіксель

ІЧ — Інфрачервоний

м — Метри

µm — Мікрометри

API — Application Programming Interface, Прикладний програмний інтерфейс

GPU — graphics processing unit Графічний процесор

Web — «павутина», інтернет-простір

PoE — Power-over-Ethernet живлення через Ethernet кабель

BT — Bluetooth

BLE — Bluetooth Low Energy

ВСТУП

Комп'ютерне бачення це те як машини (комп'ютери) можуть бачити цей світ. Сама концепція комп’ютерного бачення це створення системи, яка на основі аналізу зображення може виконувати виявлення об’єктів та стеження за ними. Зображення для такої системи можуть бути подані в різних формах:

- зображення;
- відео;
- відео потік (зображення з камери в прямому етері).

Наразі системи комп’ютерного зору дуже швидко розвиваються. їх використовують в:

- медицині;
- системи керування (промислові) (різноманітні камери для керуванням процесу виробництва на фабриках, конвейерах та заводах);
- системи контролю дорожнього руху (системи фіксації дорожніх правопорушень);
- системи соціального рейтингу (наприклад система соціально рейтингу Китаю);
- автопілоти як автомобільні так і різноманітні авіаційні та космічні (наприклад навігація марсохода Curiosity);
- різноманітні системи навігації;
- системи доповненої реальності;
- топографічне моделювання;
- системи взаємодії людина-машина;

В наш час комп’ютерний зір використовується все в більшій кількості галузей, розвиток комп’ютерного бачення робить життя людей простішим, цікавішим та безпечнішим. Так в наш час словосполучення «інтелектуальне відеоспостереження» вже стало досить звичним та є частиною нашого життя.

Системи керування через комп’ютерне бачення спрощують життя людей, замінюючи використання складного інтерфейсу додатків.

Переваги комп’ютерного зору:

- безпека (практично у всіх сферах застосовні системи контролю доступу на основі розпізнавання осіб: від бізнес-центрів та офісів компаній до банків і ресторанів);
- сервіс (за рахунок швидкої ідентифікації по обличчю можна скоротити час обслуговування клієнта і запропонувати персональні послуги);
- посилення людських можливостей (комп'ютерний зір дозволяє побачити те, що людина може не помітити; особливо актуально це для аналізу рентгенівських та інших знімків в медицині, виявлення браку в промисловості тощо);
- скорочення часу на рутинні завдання (розпізнавання, як правило, займає кілька секунд);
- автономність (без комп'ютерного зору неможливий розвиток безпілотного транспорту і роботів).

Однією з ключових проблем при створенні систем відеоспостереження реального часу є досягнення високої швидкості оброблення зображень зі збереженням якості отриманого результату. Однак, існує ряд складнощів, пов'язаних з ефективним розпаралелюванням обчислень на кілька потоків при програмній реалізації алгоритмів, а також труднощі апаратної реалізації на високопродуктивних обчислювальних системах.

Мета розробки пристрою комп’ютерного зору це створення можливості гнучкого керування різноманітними пристроями за допомогою комп’ютерного бачення. Наприклад ввімкнення світла, коли в кімнаті знаходиться людина, керування побутовими приладами за допомогою жестів, керування комп’ютером за допомогою жестів тощо.

1 АНАЛІЗ ТЗ

1.1 Віддалений модуль

Для системи комп’ютерного бачення потрібен центральний процесор ним в цій системі буде слугувати мікрокомп’ютер. Він буде приймати відео потік від камери та передавати його через інтернет на сервер, або зберігати відео у внутрішній пам’яті, якщо розірвано зв’язок з сервером. ЦП вибирається за багатьма критеріями, основні з них:

- потужність процесора;
- кількість оперативної пам’яті;
- наявність Ethernet порта;
- можливість обробки та передачі відео потоку;
- наявність розвинутого ком’юніті;

Для системи комп’ютерного бачення потрібна одна або декілька камер, вони будуть «очима» всієї системи. Камери вибираються за багатьма критеріями, в залежності від:

- середовища використання;
- потрібної роздільної здатності камери;
- можливості зуму, оптика.

Від середовища використання залежить рівень захисту пристрою. Захист від вологи, якщо пристрій встановлено в приміщенні з високим рівнем вологи. Покращене відведення тепла, якщо потрібно експлуатувати пристрій в умовах підвищених температур. Захист від вандалізму, якщо потрібно використовувати пристрій в загальнодоступному місці. Тощо.

Зі збільшенням роздільної здатності камери збільшуються і вимоги до пристроїв обробки зображення та швидкості передавання інформації.

В деяких випадках пристрій повинен мати можливість аналізувати навколишнє середовище на досить великі відстані, для цього можуть використовуватись камери з великим оптичним зумом.

Для віддаленого модулю потрібен вологозахищений корпус з можливістю направити камеру в будь-яку сторону.

1.2 Сервер

Сервер повинен відповідати по характеристикам вимогам наведеним в ТЗ, а саме:

- напруга живлення 220 В;
- потужність БЖ 450 Вт;
- відеокарта не гірше ніж nvidia geforce gtx 1050;
- процесор не гірше intel i5 7400;
- не менше 8 Гб оперативної пам’яті DDR3;
- жорсткий диск об’ємом 1 Тб;

Для того, щоб сервер міг керувати зовнішніми приладами потрібно використати пристрій, що має виводи GPIO та розробити програмне забезпечення для зв’язку між сервером та зовнішніми приладами. Такий пристрій повинен відповідати деяким критеріям:

- можливість програмування;
- наявність інтерфейсу зв’язку з сервером;
- наявність GPIO виводів.

2 ОГЛЯД АНАЛОГІВ

Так як абсолютно ідентичних пристрою, що розробляється в цьому дипломному проекті аналогів не існує в вільному продажу, порівняння буде проводитись з системами зі схожим функціоналом.

Найбільш поширені системи наведені нижче.

2.1 Мультибіометричний термінал контролю доступу ZKTeco SpeedFace-V5L(TD)

На рисунку 2.1 зображено 2.1 Мультибіометричний термінал контролю доступу ZKTeco SpeedFace-V5L(TD).

Рисунок 2.1 — ZKTeco SpeedFace-V5L(TD)

Мультибіометричний термінал контролю доступу [1] використовується для контролю доступу в приміщення за допомогою біометричної ідентифікації виявлення високої температури і індивідуальна ідентифікація в масках;

Тип доступу: геометрія особи (6000 шт.), Відбиток пальця (6000 шт.), Відбиток долоні (3 000 шт.);

Швидкість зчитування 0,3 сек; Камера 2 Мп Low Light WDR; Дисплей: 5-дюймовий сенсорний екран;

Зчитування на відстані: 30-50см долоні, до 3 метрів зчитування особи; Журнал подій 200 000 записів;

Інтерфейси зв'язку: TCP / IP, WiFi (опціонально), вхід / вихід Wiegand, RS485;

Операційна система Linux; Двоядерний процесор з частотою 900 МГц; ОЗУ 512 МБ, Flash 8G; Зв'язок: TCP / IP, WiFi (опціонально), вхід / вихід Wiegand, RS485;

Біометричні алгоритми: ZKFace V5.8 & ZKFinger V10.0 & ZKPalm V12.0; Датчик двері, кнопка виходу, тривожний вихід;

Живлення: 12В 3А

2.2 IP камера Hive UVF

На рисунку 2.2 зображена камера Hive UVF

Рисунок 2.2 — IP камера Hive UVF

Камера Hive UVF [2] — це система розпізнавання автомобільних номерів і управління доступом транспорту призначена для використання на паркінгах, КПП, автомийках, СТО та інших.

Програмне забезпечення інтегроване в камеру, налаштування проводиться через web-інтерфейс IP-камери, релейні виходи камери дозволяють

керувати воротами або шлагбаумом. Інформація зберігається на SD-карті пам'яті живлення здійснюється через PoE.

Характеристики Hive UVF:

Матриця: 1/3" CMOS

Розширення: 4 Мп (2592x1520)

Швидкість запису: 2592×1520 – 20 к/с; 2560×1440 та нижче – 25 к/с

Фокусна відстань: 2.8-12мм

Кути обзору: 95°-29°

Формат стиснення: H.265/H.264

ІК-підсвічування: 30м

Чутливість: 0.02Lux@F1.4, 0Lux з ИК

Мережевий порт: 10M/100M Base-TX Ethernet

Мережеві функції: L2TP, IPv4, IGMP, ICMP, ARP, TCP, UDP, DHCP, PPPoE, RTP, RTSP, RTCP, DNS, DDNS, NTP, FTP, UPnP, HTTP, HTTPS, SMTP, 802.1x.

Сумісність: ONVIF, API

Ступінь захисту: IP66

Робоча температура: -50...+60°C

Розміри: 253.4x86x71.7мм

3 ФУНКЦІОНАЛЬНА СХЕМА ПРИСТРОЮ ТА ХАРАКТЕРИСТИКА ЕЛЕМЕНТІВ

Функціональна схема пристрою комп’ютерного зору наведена на рис. 3.1.

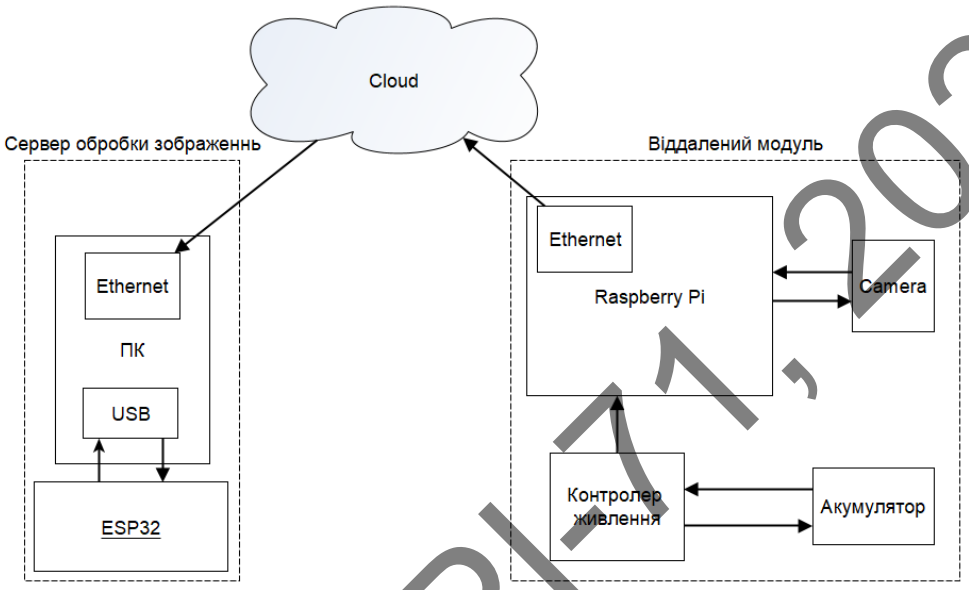

Рисунок 3.1 — Функціональна схема

Пристрій складається з двох основних модулів:

- віддалений модуль;
- сервер оброблення зображень

Віддалений модуль містить в собі камеру для захоплення зображення або відео потоку, ЦП (Raspberri pi 3B+) для передачі відео потоку на сервер через мережу Ethernet та зберігання відео якщо є проблеми з з’єднанням, акумулятор та контролер живлення для забезпечення роботи приладу при відсутності енергопостачання.

Сервер оброблення зображень складається з таких блоків: ПК (персонального комп’ютера) для прийняття і обробки відео потоку та прийняття рішень, ESP32, що буде слугувати для підключення приладів якими керує сервер.

3.1 Центральний процесор віддаленого модуля

За основу віддаленого модуля взято Raspberry PI 3 Model B+, зовнішній вигляд показано на рис 3.2. Raspberry Pi це одноплатний комп'ютер за розмірами наближений до розмірів банківської картки. Raspberry PI був розроблений британським фондом Raspberry Pi Foundation. Головна мета цього пристрою була зробити навчання комп'ютерних наук простішим та доступнішим. Щоб зробити навчання максимально доступним ціна на Raspberry PI починається від 20 доларів для версії A розробленої в лютому 2013р, та закінчується 100 доларами за версію 400.

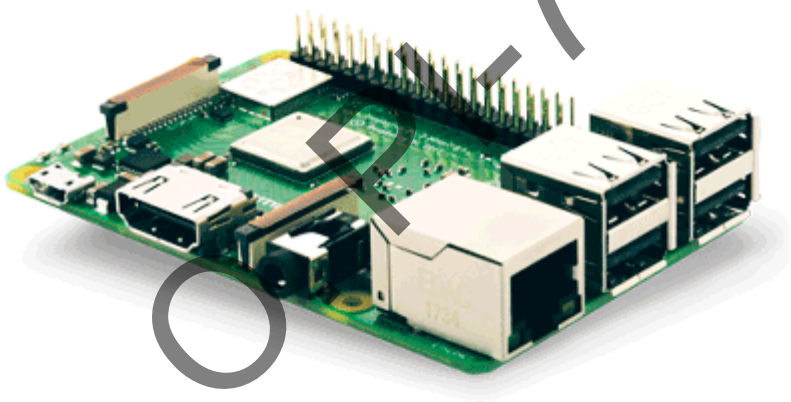

Рисунок 3.2— Зовнішній вигляд Raspberry PI 3 Model B+

Цей мікрокомп'ютер поєднує в собі достатню обчислювальну потужність та всі необхідні інтерфейси (Ethernet, Wi-Fi, CSI) для передачі відеопотоку високої якості. Також великою перевагою саме цього мікрокомп'ютера є його невелика ціна. Версія обрана для виконання цієї дипломної роботи коштує близько 35 доларів на момент написання роботи.

Ще одним стимулом до використання саме цього мікрокомп'ютера стала наявність у нього роз'єму для камери та великої кількості різноманітних ка-

мер модулів, від самих простих, ціна за які варіюється від 10 доларів до досить потужних камер ціною в декілька сотень доларів. Також до кожної камери є багато прикладів по створенню програмного забезпечення для роботи з нею.

З великої кількості операційних систем доступних для даного мікрокомп'ютера обрано Raspberry Pi OS. Raspberry Pi OS (Raspbian) це операційна система, що базується на Debian (Linux). Ця операційна система офіційно представлена Raspberry Pi Foundation як основна операційна система для мікрокомп'ютерів Raspberry Pi. Переваги саме цієї операційної системи в тому, що вона розроблена спеціально для мікрокомп'ютера Raspberry Pi. Raspberry Pi OS має дуже велике комюніті, що постійно розвивається та зростає. В Raspberry Pi OS є дуже багато готових рішень які можна імплементувати в своїх проектах. Це значно спрощує розробку на цій платформі та відкриває широкий спектр можливостей.

Характеристики Raspberry PI 3 Model B+:

- вага 45 г;
- габарити 85 × 56 × 17 мм;
- процесор Broadcom BCM2837B0 64-bit, Cortex-A53;
- робоча частота: 1,4ГГц;
- графічний процесор: VideoCore 4 3D;
- оперативна пам'ять: 1 ГБ;
- мережеві інтерфейси: Gigabit Ethernet;
- бездротові інтерфейси: Wi-Fi (2.4GHz / 5GHz) IEEE 802.11 і Bluetooth 4.2 LE (на основі чіпу Cypress CYW43455);
- відео вихід: HDMI;
- аудіо вихід: 3,5 мм;
- GPIO: 40 контактів;
- роз'єм для камери: CSI;
- роз'єм для дисплею: Raw LCD (DSI);
- вимоги до живлення: 5 В 2.1 А;

- операційні системи: Raspbian, Windows 10 IoT Core, OpenELEC, OSMC, Pidora, Arch Linux, RISC OS.

3.2 Камера віддаленого модуля

В якості камери віддаленого модуля використано Raspberry Pi High Quality Camera, що зображена на рис 3.3. Цей модуль камера розроблена на основі матриці сенсора Sony IMX477R. Для роботи цієї камери потрібен об’єктив, що встановлюється за допомогою адаптера типу C або CS.

Рисунок 3.3 — Камера Raspberry Pi High Quality Camera без об’єктиву

Дана камера розроблена спеціально для мікрокомп’ютера Raspberry PI, це забезпечить максимальну сумісність цієї камери з обраним ЦП.

Характеристики Raspberry Pi High Quality Camera:

- матриця (сенсор): Sony IMX477R;
- роздільна здатність: 12.3MP, 4056 (Висота) × 3040 (Ширина);
- розміри пікселя: 1.55µm (Висота) × 1.55µm (Ширина).

3.3 Об’єктив для камери віддаленого модуля

Об’єктивом для камери віддаленого модуля обрано Raspberry PI Wide Angel Lens 6mm, що зображено на рисунку 3.4. Це високоякісний промисловий ширококутний об'єктив з фокусною відстанню 6 мм, кутом огляду 63°, адаптером CS-Mount. Сумісний з камерою високої якості Raspberry Pi.

Рисунок 3.4 — Об’єктив Raspberry PI Wide Angel Lens 6mm

Характеристики Raspberry PI Wide Angel Lens 6mm:

- фокусна відстаннь: 6мм;
- діафрагма: F1.2;
- адаптер кріплення: CS-mount;
- кут огляду: 63°;
- задній фокус: 7.53 мм;
- мінімальна відстань до об’єкта: 0.2 мм;
- розмір: 30.00 × 34.00 мм;
- вага: 53 г.

Через досить широкий кут огляду в 63° камера з цим об’єктивом зможе захопити більше об’єктів ніж з іншими об’єктивами. Об’єктив має ручне налаштування діафрагми.

На рисунку 3.5 зображений модуль камери Raspberry Pi High Quality Camera з встановленим об’єктивом Raspberry PI Wide Angel Lens 6mm

Рисунок 3.5 — Камера Raspberry Pi High Quality Camera з встановленим об’єктивом Raspberry PI Wide Angel Lens 6mm

3.4 Акумулятор для віддаленого модуля

В технічному завданні вказано, що пристрій повинен працювати автономно не менше 6 годин.

Розрахунок необхідної батареї:

$$Q = \frac{P \cdot t}{V \cdot k} \quad (1)$$

Q - необхідна ємність акумулятора, А·год;

P - потужність, Вт;

V - напруга кожної акумуляторної батареї, В;

t - час резервування, год;

k - коефіцієнт використання ємності акумуляторів (кількості електричної енергії, допустимої до використання споживачами).

Розрахунок потужності:

$$\boldsymbol{P = I \cdot V} \quad \textbf{(2)}$$

P - потужність, Вт;

V - напруга кожної акумуляторної батареї, В;

I – струм, А

Потужність raspberry pi:

$$\boldsymbol{P = I \cdot V = 2.1 \cdot 5 = 10.5}\ \text{Вт} \quad \textbf{(3)}$$

$$\boldsymbol{Q = \frac{P \cdot t}{V \cdot k} = \frac{10.5 \cdot 6}{3.7 \cdot 0.7} = 24.3}\ \text{А} \cdot \text{год} \quad \textbf{(4)}$$

Як видно з (4) для автономної роботи пристрою впродовж 6 годин, потрібно забезпечити ємність акумулятора 25 А·год.

Для забезпечення такої ємності вибрано 8 акумуляторів Panasonic NCR18650B (рис 3.6) ємністю 3.4 А·год кожний, зєднаних паралельно.

Рисунок 3.6 — Акумулятор Panasonic NCR18650B

Сумарна ємність такої збірки складає 27,2 А·год.

3.5 Модуль зарядки акумулятора

Модуль зарядки акумуляторів потрібен для захисту елементів живлення від перепадів напруги перезарядки або глибокої розрядки.

Акумулятори обрані в пункті 3.4 Panasonic NCR18650B це li-ion акумулятори, вони мають досить вагому перевагу: велика ємність при малих габаритах, але вони не позбавлені недоліків. Основний недолік акумуляторів li-ion це велика чутливість до перезарядки та глибокої розрядки, саме ці недоліки компенсує модуль зарядки.

Для захисту акумуляторів взято модуль зарядки на основі мікросхеми TP4056, що зображено на рисунку 3.7.

Рисунок 3.7 — Модуль зарядки MicroUSB для Li-ion акумуляторів TP4056 (TC4056)

Плата зарядки акумуляторів TP (TC) 4056 1A Micro USB з захистом видає струм заряду 1 А і відключається коли напруга акумуляторів перевищує 4,2 В. Крім того, коли напруга падає нижче 2,5 В, інтегральна схема захисту відключить напругу, щоб захистити акумулятор від глибокого розряду. Також контролер відключить акумулятор якщо струм навантаження перевищить 3 А.

Як видно з характеристик в пункті 3.1 для живлення Raspberry pi потрібно 2.1 А. Так як максимальний струм заряду цієї плати 1 А, було використано два таких модулі з’єднані паралельно.

3.6 Мікроконтролер для керування зовнішніми приладами

Для розширення можливостей системи комп’ютерного бачення потрібно використовувати універсальні піни вводу виводу GPIO. Так як персональний комп’ютер не має пінів вводу виводу це потрібно компенсувати шляхом введення додаткового пристрою через який сервер зможе контролювати будь-які прилади.

Для керування зовнішніми приладами використано ESP32 WROOM DevKit v1.

ESP32 WROOM DevKit v1 це модуль розроблений на основі мікромодуля ESP-WROOM-32 від компанії Espressif. Цей модуль призначений для широкого спектра застосувань, він має Wi-Fi, BT, BLE. На модулі зібрана вся необхідна переферія для роботи ESP-WROOM-32. На рисунку 3.8 наведений зовнішній вигляд модуля ESP32 WROOM DevKit v1.

Рисунок 3.8 — Мікроконтролер ESP32 WROOM DevKit v1

ESP-WROOM-32 виконаний на базі двоядерного процесора ESP32, із тактовою частотою від 80 МГц до 240 МГц.

Однією з особливостей модуля є наднизьке споживання і гнучкий вибір «сплячих» режимів, що дозволяють отримати цифри до 20мкА (deep sleep mode).

Для спілкування модуля з сервером використано інтерфейс USB/UART

Технічні характеристики мікроконтролера**:**

- USB-UART конвертер CP2102;
- напруга живлення 5В;
- максимальний струм стабілізатора напруги: 800мА;

- процесор: 32-бітний двоядерний мікропроцесор Xtensa LX6;
- бездротовий зв'язок: Wi-Fi 802.11b / g / n до 150 Мб / с, Bluetooth 4.2 BR / EDR / BLE;
- підтримка STA / AP / STA + AP режимів, вбудований стек TCP/ IP;
- UART, SPI, I2C, I2S інтерфейси, ШІМ, SD контролери, ємнісні сенсорні, АЦП, ЦАП;
- тактова частота процесора: 80, 160 або 240 МГц;
- флеш-пам’ять: 4 МБ;
- кількість виводів: 30;
- кількість GPIO: 25;
- крок виводів: 2.54 мм;
- розміри: 51 х 28 х 13 мм.

Модуль ESP-WROOM-32 розроблений для переносної і автономної електроніки та інтернет-речей, виконаний в корпусі 25,5 мм х 18 мм, має на борту Flash пам'ять, кварц 40 МГц і PCB антену.

3.7 Корпус пристрою комп’ютерного зору

Корпус розроблено в програмному забезпечені Solid Works згідно з технічним завданням. Зовнішній вигляд моделі готового пристрою зображений на рисунку 3.9.

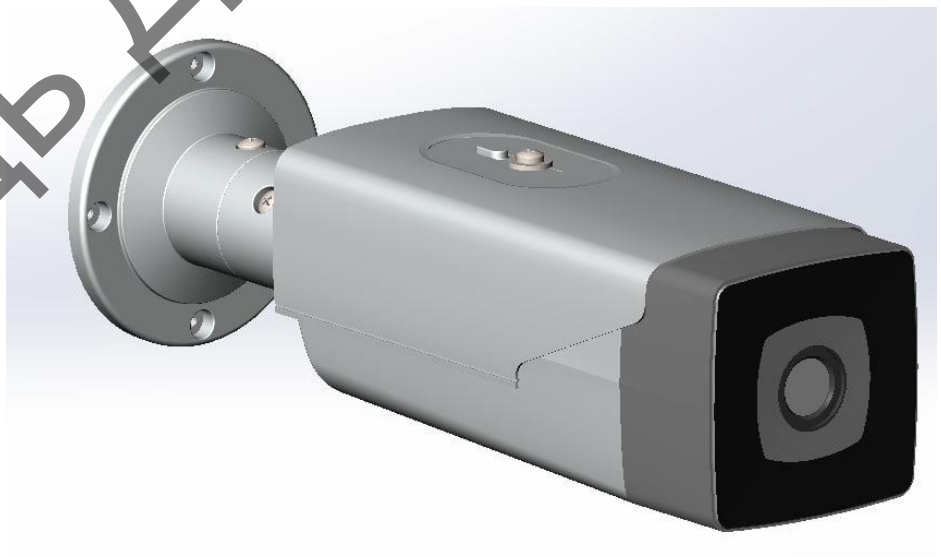

Рисунок 3.9 — Зовнішній вигляд готового пристрою

Корпус складається з 8 частин (за виключенням деталей кріплення):

1) плата живлення;
2) передня кришка;
3) нижня кришка;
4) верхня кришка;
5) захисна кришка;
6) кільце 1;
7) кільце 2;
8) кільце 3.

Деталі виконані з пластику литтям. Нижня кришка кріпиться до передньої і верхньої на фіксатори та герметизується герметиком за потреби. Raspberry PI 3 Model B+ кріпиться до нижньої кришки на 4 гвинта ANSI / ASME B18.6.7M -M2.5 x 0.45 x 5. Плата вставляється в спеціальні пази на нижній кришці. Raspberry Pi High Quality Camera кріпиться на 4 гвинта ANSI / ASME B18.6.7M -M2.5 x 0.45 x 5 до передньої кришки. Захисна кришка кріпиться до верхньої кришки гвинтом ANSI / ASME B18.6.7M -M5 x 0.8 x 16, та затягується гайкою. Корпус кріпиться до поверхні та наводиться в потрібному напряму за допомогою трьох кілець, які фіксуються трьома гвинтами ANSI / ASME B18.6.7M - M5 x 0.8 x 6.

3.8 Друкована плата

Друкована плата розроблена в програмному забезпеченні Altium. Зовнішній вигляд друкованої плати в 3D наведено на рисунку 3.10. Друкована плата розроблена за схемою електричною принциповою, що наведена в графічній частині дипломної роботи РІ71.421457.001 Е3.

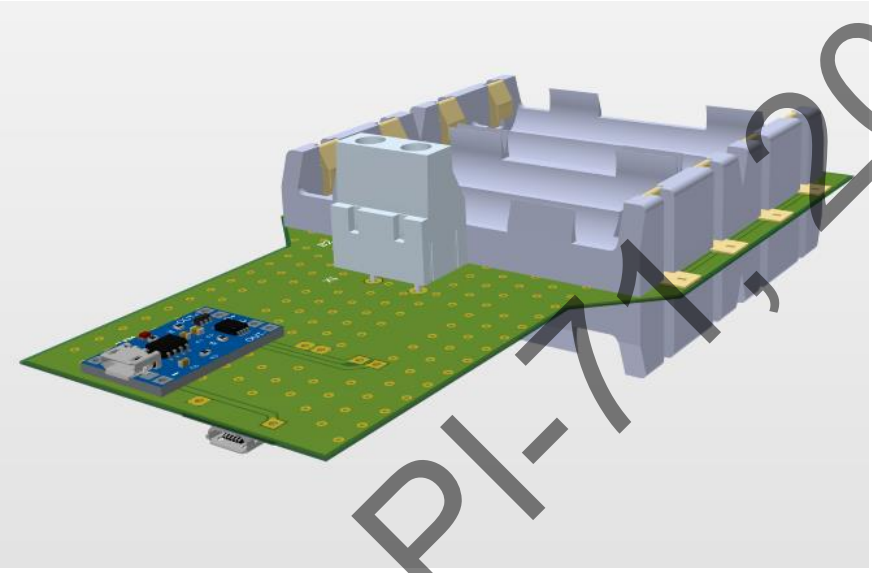

Рисунок 3.10 — Зовнішній вигляд друкованої плати в 3D

Друкована плата зроблена для безперебійного живлення Raspberry PI 3 Model B+. На друкованій платі розташовані 2 тримача для акумуляторів типу 18650 сумарною ємністю 27,2 А·год.

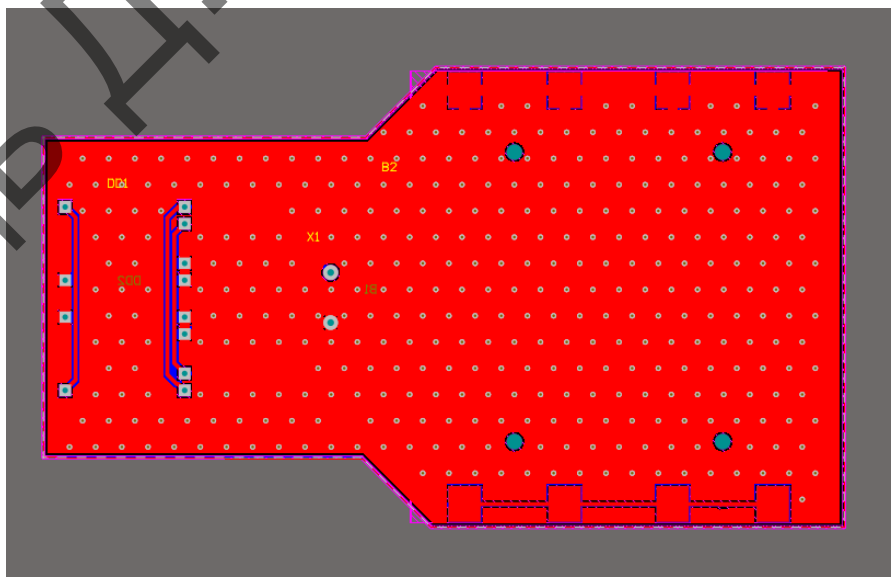

Рисунок 3.11 — Зовнішній вигляд друкованої плати в 2D

За контроль заряду акумуляторів та подачу напруги на живлення Raspberry PI 3 Model B+ відповідає модуль зарядки акумуляторів на основі мікросхеми TP4056. Зовнішній вигляд друкованої плати в 2D з позначенням шарів видно на рисунку 3.11

4 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПРИСТРОЮ КОМП'ЮТЕРНОГО ЗОРУ

Нижче наведено опис необхідного програмного забезпечення та використаних бібліотек.

4.1 Бібліотеки використані для детектування зображення

Для детектування зображення пристрою комп'ютерного зору використано наступні бібліотеки: MediaPipe , TensorFlow , OpenCV.

4.1.1 Бібліотека MediaPipe

Частина з розпізнаванням (детектуванням) рук розроблена на фреймворці з відкритим вихідним кодом MediaPipe. MediaPipe - це фреймворк представлений Google, який допомагає створювати мультимодальні конвеєри машинного навчання. Розробник може створити прототип, не заглиблюючись в написання алгоритмів і моделей машинного навчання, використовуючи існуючі компоненти. Ця структура може використовуватися для різних додатків для обробки зображень та мультимедіа (особливо у віртуальній реальності), таких як виявлення об'єктів, розпізнавання осіб, відстеження рук і сегментація волосся. MediaPipe підтримує різні апаратні та операційні платформи, такі як Android, iOS і Linux, пропонуючи API на C ++, Java, Objective-c і т.д. Це середовище також здатна використовувати ресурси GPU [3].

Mediapipe поставляється з готовими до використання моделями, де розробники можуть почати використовувати його безпосередньо або зі своїми зазначеними модифікаціями. Виявлення заперечень може бути оброблено дуже легко, без споживання великої кількості системних ресурсів. Виявлення об'єктів на основі ML з камери прямої трансляції з частотою кадрів 30 кадрів в секунду зазвичай вимагає великих ресурсів і нездійсненно через тривалий час виведення. MediaPipe досягає цього шляхом паралельного запуску, відстеження та виявлення, тому кожен процес ніколи не буде заблокований іншим [3].

На рисунку 4.1 наведено блок-схему знаходження (детектування) об’єкта за допомогою MediaPipe.

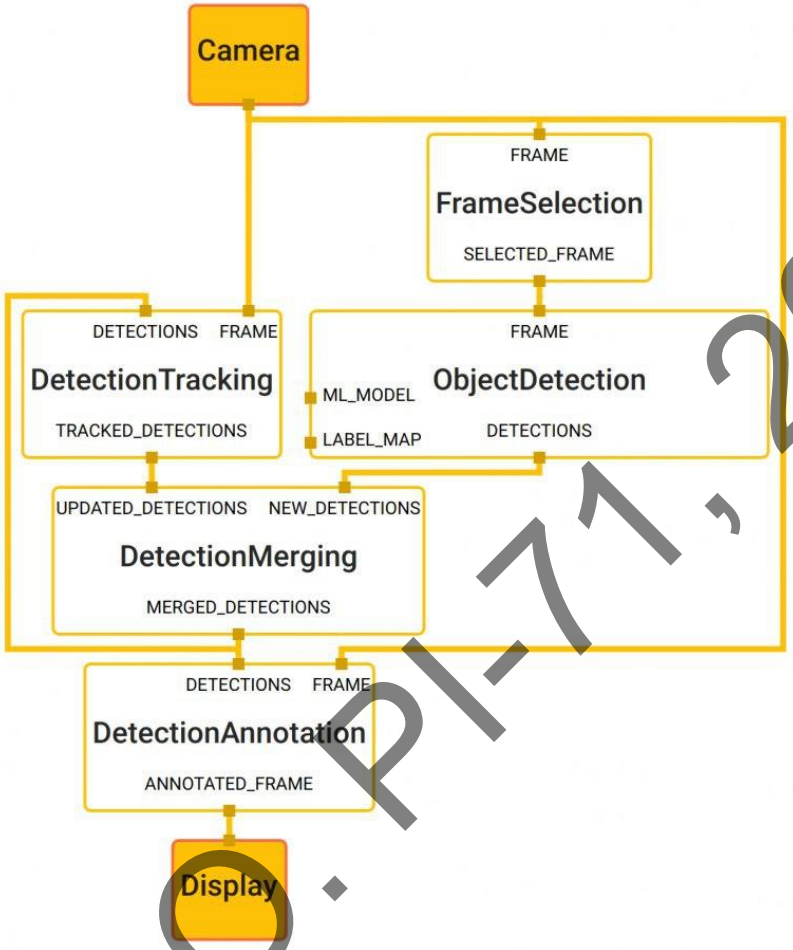

Рисунок 4.1 — Функціональна схема детектування об’єкта за допомогою MediaPipe

З рисунку 4.1: виявлення заперечень відбувається в два окремих процеси - повільний процес виявлення і швидкий процес відстеження. І з конфігурації графа Pipeline, калькулятори запускають паралельні потоки для виконання процесу.

Mediapipe використовується в проекті комп’ютерного зору для розпізнавання візуальних команд від людини.

4.1.2 Бібліотека TensorFlow

TensorFlow [4] — відкрита програмна бібліотека для машинного навчання, розроблена компанією Google для вирішення задач тренування нейронних мереж з метою автоматичного знаходження та класифікації образів, досягаю-

чи якості людського сприйняття. Застосовується як для досліджень, так і для розробки власних продуктів Google. Основний API для роботи з бібліотекою реалізований для Python, також існують реалізації для R, C Sharp, C ++, Haskell, Java, Go і Swift.

TensorFlow потрібен в проекті комп'ютерного зору для бібліотеки Mediapipe.

4.1.3 Бібліотека OpenCV

OpenCV [5] — Open Source Computer Vision Library, бібліотека комп'ютерного зору з відкритим вихідним кодом. Це бібліотека комп'ютерного зору та обробки зображеннь. Бібліотека реалізована на C/C++, Python, Java, Ruby, Matlab, Lua. За допомогою OpenCV можна виконати багато найрізноманітніших операцій з зображеннями та відео, деякі з них:

- імпорт та перегляд зображення (при чому з будь-яких джерел, від збережених на жорсткому диску картинок до відео потоку, що транслюється на сайті);
- зміна розміру;
- поворот зображення;
- переведення зображення в градації сірого;
- розмитість та згладжування зображень;
- нанесення графіки на зображення чи відео (прямокутники, лінії);
- нанесення тексту на зображення чи відео;
- розпізнавання обличь;
- розпізнавання об'єктів.

В проєкті комп'ютерного зору OpenCV використовується для передачі відеопотоку Mediapipe, нанесення графіки та написів на відео.

4.2 Принцип роботи приладу комп'ютерного бачення

Основна мета програмного забезпечення приладу керування через комп'ютерне бачення:

1) передача відеопотоку серверу;

2) аналіз відеопотоку сервером;

3) розпізнавання людської руки на відео;

4) розпізнавання жестів;

5) прийняття рішень на основі розпізнаних жестів;

6) курування зовнішніми приладами.

З камери Raspberry Pi High Quality Camera відеопоток потрапляє до ЦП віддаленого модуля Raspberry PI 3 Model B+. ЦП віддаленого модуля публікує відео потік через Ethernet. Сервер приймає відеопотік через Ethernet. Далі відбувається обробка зображення лістинг коду обробки зображень наведено в додатку Б. Після обробки зображення, якщо було розпізнано команду, через інтерфейс SPI відправляється команда ESP32. Лістинг серверної частини коду наведено в додатку В, лістинг коду для мікроконтролера наведено в додатку Д. В додатках наведено код мінімального функціонала, він може бути розширений в залежності від потреб користувача.

5 РОЗРОБКА МАКЕТУ ПРИЛАДУ КОМП'ЮТЕРНОГО БАЧЕННЯ ТА ЙОГО ТЕСТУВАННЯ

5.1 Опис макету для тестування

Для тестування працездатності з боку віддаленого модуля використана Raspberry PI 3 Model B+, та камера Raspberry Pi Camera Module v2, по характеристикам Raspberry Pi Camera Module v2 дещо відрізняється від Raspberry Pi High Quality Camera, але вони мають ідентичний інтерфейс підключення до Raspberry Pi та програмно взаємозамінні. Порівняння характеристик цих камер вказано в таблиці 5.1.

Таблиця 5.1 — Порівняння характеристик Raspberry Pi Camera Module v2 та Raspberry Pi High Quality Camera

№ з/п	Характеристика	Raspberry Pi Camera Module v2	Raspberry Pi High Quality Camera
1	Тип сенсора	Sony IMX219	Sony IMX477R
2	Розділювальна здатність	8 МП (3280×2464)	12.3MP (4056× 3040)
3	Відео формати	1080p30, 720p60 та 640 × 480p60 / 90	1080p30, 720p60 та 640 × 480p60 / 90
4	Розміри	25 × 24 × 9 мм	38 x 38 x 18,4 мм (без об'єктиву)
5	Розмір пікселя	1,12 мкм x 1,12 мкм	1,55 мкм x 1,55 мкм

Макет для тестування програмного забезпечення віддаленого модуля зображений на рис 5.1.

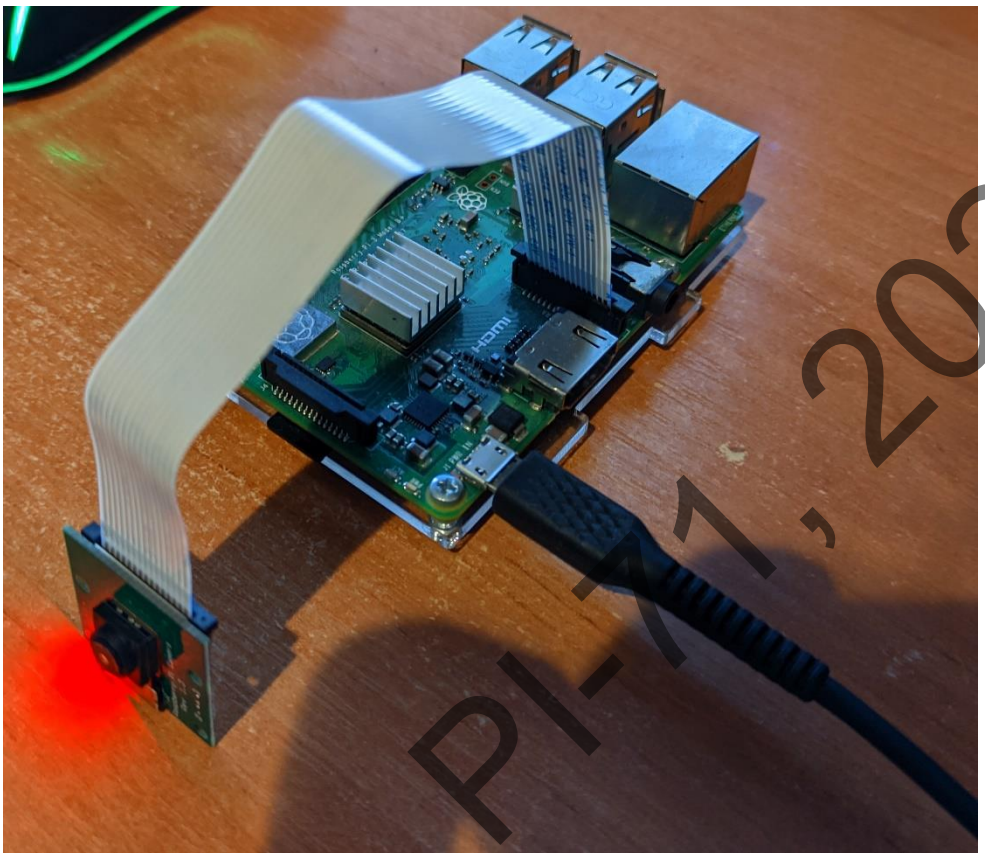

Рисунок 5.1 — Макет для тестування програмного забезпечення віддаленого модуля

Приклад лістингу коду для віддаленого модуля наведено в додатку Г.

Для тестування серверної частини використано персональний комп’ютер з наступними характеристиками:

- процесор: Intel Xeon CPU E5-1650v2 3.50GHz;
- оперативна пам’ять: 32 Гб DDR3;
- відеокарта: Radeon RX 470;
- постійна пам’ять: 2 ТБ HDD 2шт, 512 Гб SSD M.2.

Лістинг коду для серверної частини наведено в додатку Б.

Замість ESP32 використано ESP8266, ця плата має дещо менший функціонал в порівнянні з ESP32, але її функціоналу вистачить для тестування працездатності програмного забезпечення. Порівняння характеристик ESP32 та ESP8266 наведено в таблиці 5.2.

Таблиця 5.2 — Порівняння характеристик ESP32 та ESP8266

№ з/п	Характеристика	ESP8266	ESP32
1	Мікропроцесор	Xtensa Single-core 32-bit L106	Xtensa Dual-core 32-bit LX6 with 600 DMIPS
2	Wi-Fi	HT20	HT40
3	SRAM	Ні	Так
4	Flash	Ні	Так
5	GPIO	17	36
6	Програмна ШІМ	8 каналів	16 каналів
7	SPI/I2C/I2S/UART	2/1/2/2	4/2/2/2
8	АЦП	10 біт	12 біт
9	Ethernet	Ні	Так
10	Сенсор дотику	Ні	Так
11	Давач температури	Ні	Так
12	Давач ефекту холу	Ні	Так
13	Діапазон робочих температур	-40°C до 125°C	-40°C до 125°C

Зовнішній вигляд підключення ESP8266 наведено на рисунку 5.2.

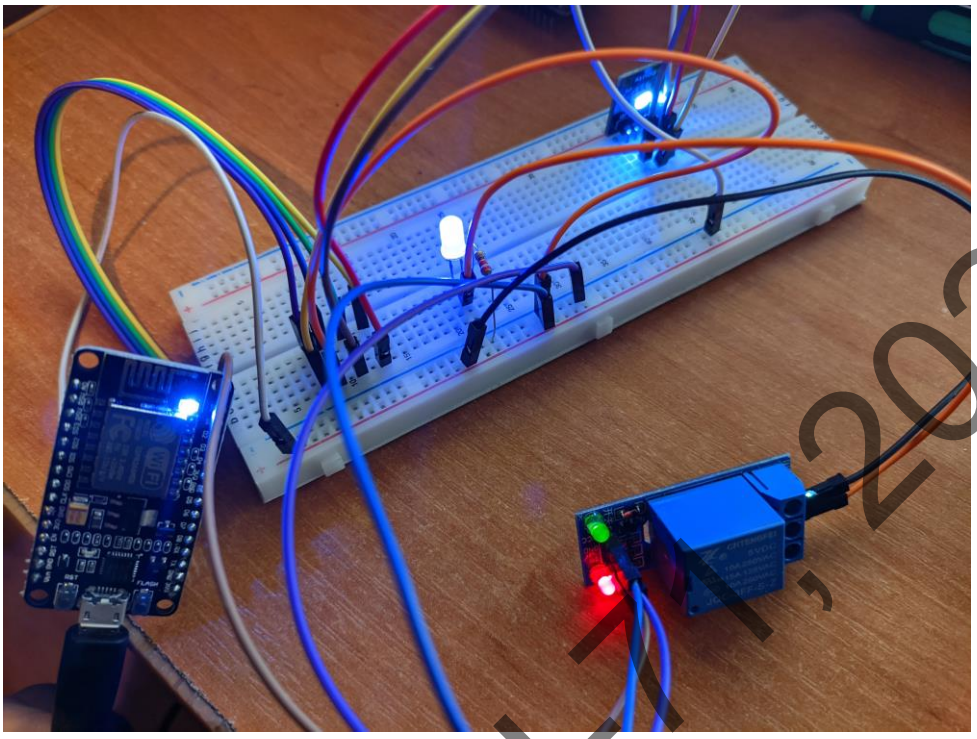

Рисунок 5.2 — Підключення ESP8266

Raspberry PI 3 Model B+ та сервер знаходяться в одній локальній мережі, яку створює роутер TP-WR941ND. Raspberry PI 3 Model B+ підключена до роутера за допомогою Wi-Fi, сервер підключений по проводу. Блок схема підключення для тестування наведена на рисунку 5.3

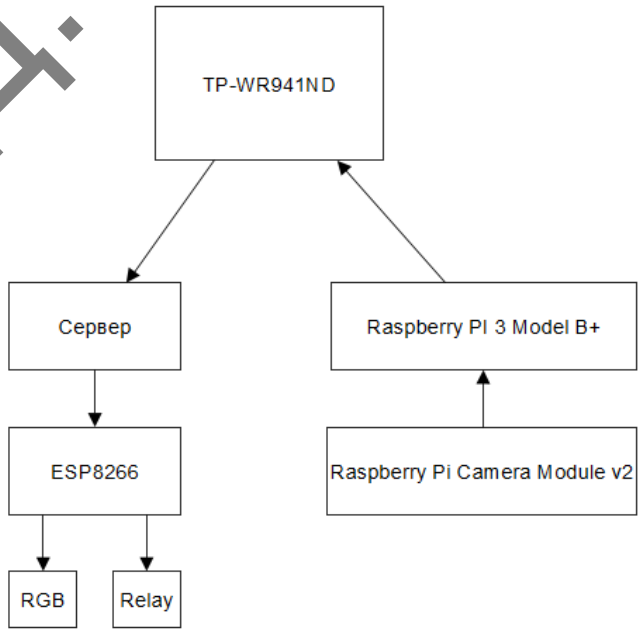

Рисунок 5.3 — Блок схема підключення для тестування

На рисунку 5.4 зображено найменування виводів ESP8266

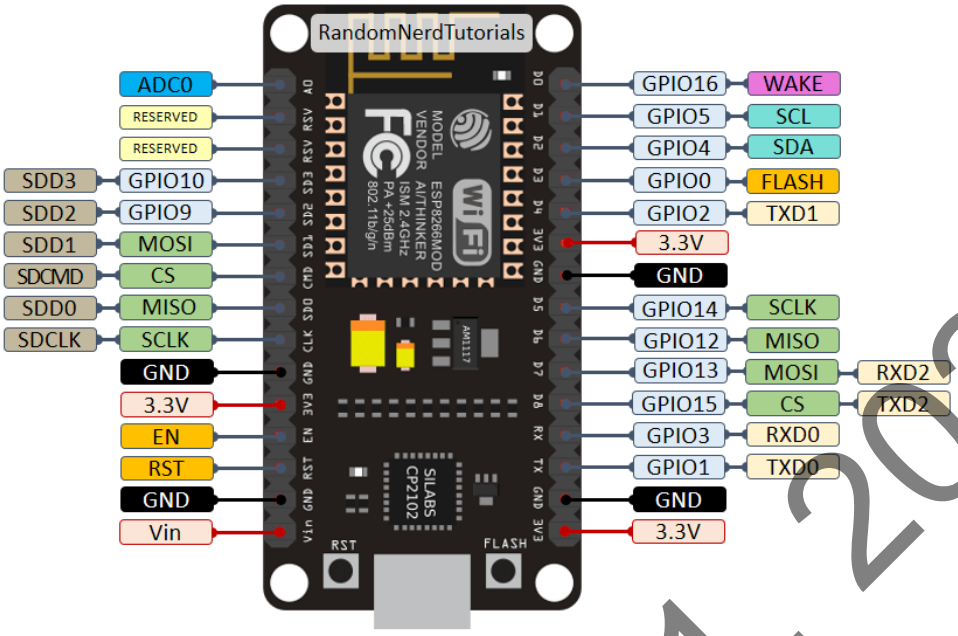

Рисунок 5.4 — pinout ESP8266

До ESP8266 підключено RGB світлодіод на виводи GPIO14 (D5), GPIO2 (D4), GPIO4 (D2) та реле на вивід GPIO15 (D8). До реле підключений синій світлодіод, для наглядності. Замість світлодіода до реле можна підключити будь що, чим можна керувати ввімкненням та вимкненням струму. Лістинг коду для ESP8266 наведено в додатку Д.

5.2 Опис проведення тесту

5.2.1 Запуск віддаленої частини

Raspberry PI 3 Model B+ підключена до джерела живлення через micro usb та до роутера через Wi-Fi. До Raspberry PI 3 Model B+ підключена камера Raspberry Pi Camera Module v2, зовнішній вигляд тестового макету віддаленого модуля зображено на рисунку 5.1. На сервері запущено програму VNC Viewer для підключення до Raspberry PI та запуску коду транслювання зображення, лістинг коду транслювання зображення наведено в додатку Г. Після запуску коду транслювання зображення можна перевірити чи він працює правильно зайшовши на raspberry_ip:8000 через будь-який браузер. Якщо все запущено правильно сторінка на сторінці буде трансляція з камери.

На рисунку 5.5 зображено вигляд сторінки трансляції відео потоку в браузері, ip raspberi яка приймала участь в тестуванні 192.168.0.12.

Рисунок 5.5 — Вигляд сторінки трансляції в браузері

5.2.2 Запуск серверної частини

Після того як почалася трансляція в браузері можна запускати код розпізнавання, лістинг коду розпізнавання наведено в додатку Б.

Трохи детальніше про функціональність:

Наразі до сервера підключено ESP8266 за допомогою UART. До ESP8266, як зазначалось вище, підключений RGB світлодіод та реле. Сервер розпізнає положення руки та передає просторову координату кінчика вказівного пальця як інтенсивність світіння відповідного кольору. Тобто координата X це R — червоний колір, координата У це G — зелений колір а Z це B — синій колір. Для фіксування кольору потрібно зігнути вказівний палець тоді напис “Tracking ON” зміниться на напис “Tracking OFF” а значення R, G, B перестануть змінюватись. Для керування реле використовується середній палець — коли він розігнутий реле ввімкнене а напис біля руки “Relay ON”, коли середній палець зігнуто реле вимкнено а напис статусу реле змінюється на “Relay OFF”.

На рисунку 5.6 зображено розпізнавання руки, на зображення руки накладено 21 ключову точку, координати кожної з них відстежуються. Габаритні розміри руки обведені прямокутником.

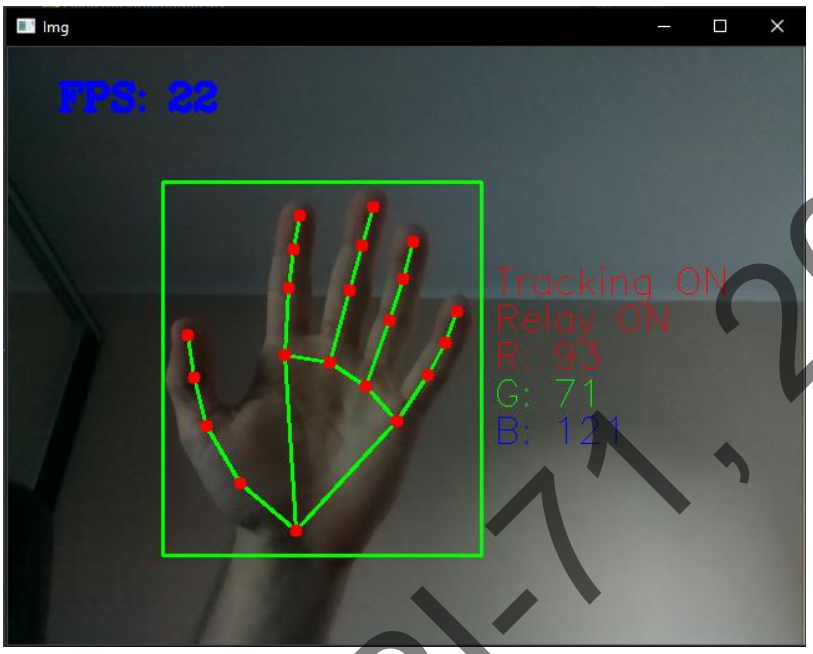

Рисунок 5.6 — Оброблене зображення з Raspberry Pi Camera Module v2

Правіше руки знаходиться інформація про розпізнані команди. Так як і вказівний і середній пальці підняті ввімкнені реле та зміна кольору RGB світлодіоду, про це свідчать написи “Tracking ON”, “Relay ON” та значення R, G, B, що змінюються. В верхньому лівому куті знаходиться лічильник кадрів за секунду. Вигляд тестової установки при подачі таких команд наведено на рисунку 5.2.

На рисунку 5.7 зображено оброблене зображення з Raspberry Pi Camera Module v2 з розпізнаною командою на вимкнення зміни кольору RGB світлодіоду.

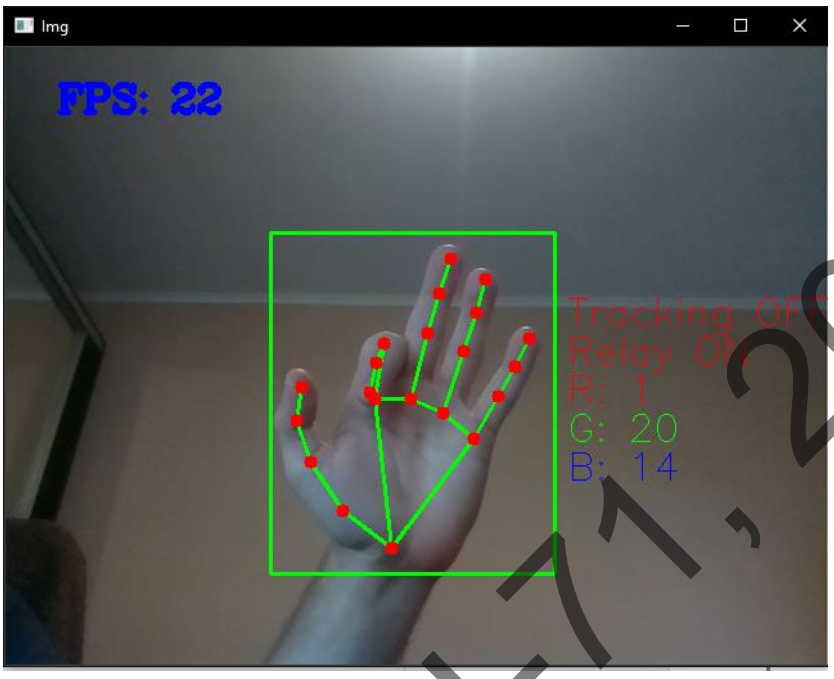

Рисунок 5.7 — Оброблене зображення з Raspberry Pi Camera Module v2 з командою “Tracking OFF”

На рисунку 5.7 видно, що вказівний палець зігнуто — це команда зафіксувати значення RGB. Команда зафіксувати значення розпізнана і застосована, це видно з надпису “Tracking OFF” та значенням RGB які притаманні лівому верхньому куту зображення.

На рисунку 5.8 зображено оброблене зображення з Raspberry Pi Camera Module v2 з розпізнаними командами на вимкнення зміни кольору RGB світлодіоду та вимкнення реле.

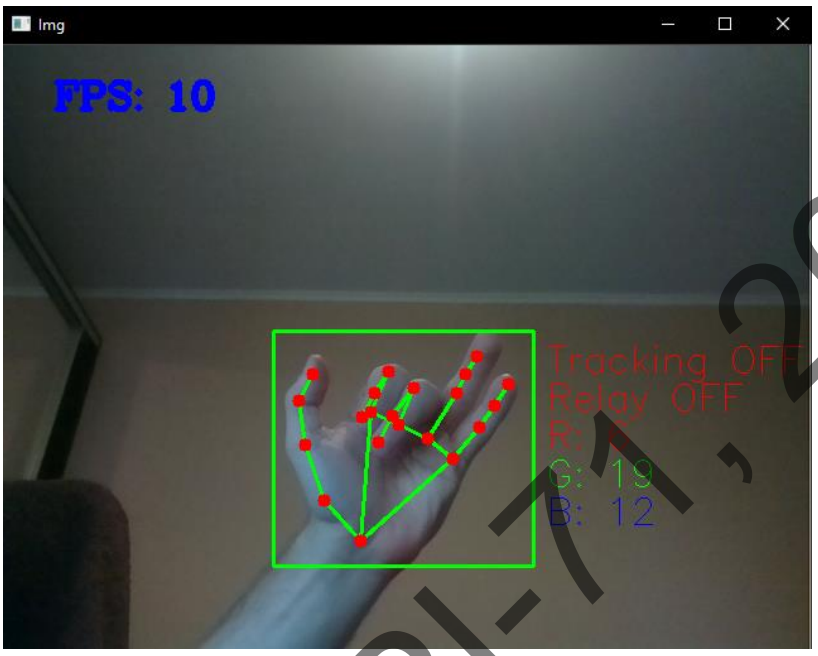

Рисунок 5.8 — Оброблене зображення з Raspberry Pi Camera Module v2 з командами “Tracking OFF” та “Relay OFF”

На рисунку 5.8 видно, що вказівний та середній пальці зігнуто — це команди зафіксувати значення RGB та вимкнути реле.

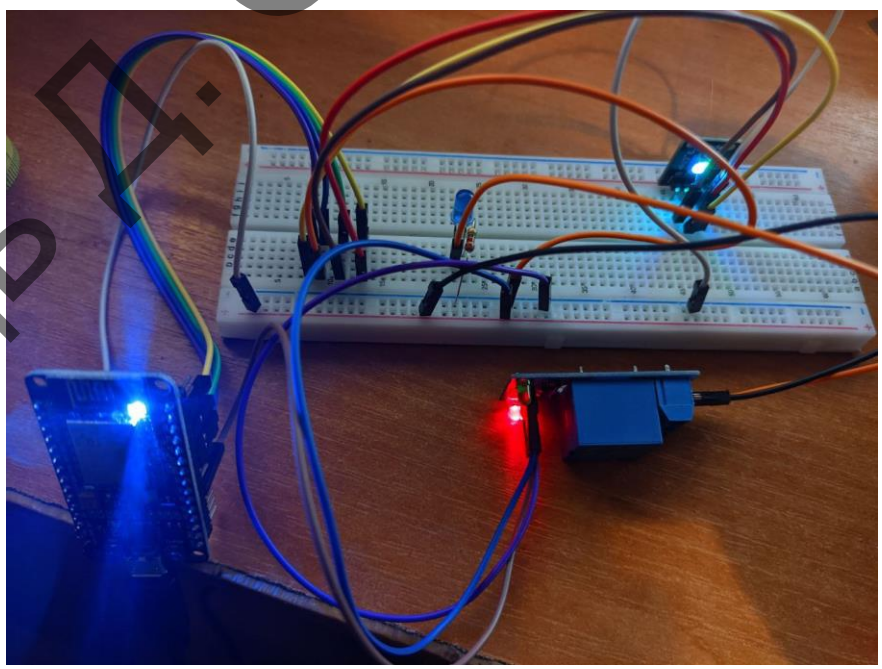

Рисунок 5.9 — Стан ESP8266 з командами “Tracking OFF” та “Relay OFF”

Команда зафіксувати значення розпізнана і застосована, це видно з надпису “Tracking OFF” та значенням RGB які притаманні лівому верхньому куту зображення на рис 5.8. Команда вимкнути реле розпізнана і застосована, це видно з надпису “Relay OFF”. Стан ESP8266 можна побачити на рисунку 5.9.

ВИСНОВКИ

В цьому дипломному проекті розроблено пристрій комп'ютерного зору згідно з технічним завданням.

Орієнтовна собівартість приладу (без вартості серверу та програмного забезпечення) складає близько 255$. Остання, головним чином, обумовлена використанням якісного модуля камери, що забезпечує більшу гнучкість пристрою комп'ютерного зору при використанні.

Пристрій комп'ютерного зору може об'єднати в собі функціонал різноманітних пристроїв аналізу візуальної інформації, за наявності необхідного програмного забезпечення.

Проведені тести довели працездатність приладу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Мультибіометричний термінал контролю доступу ZKTeco SpeedFace-V5L(TD) — [Електронний ресурс] — Режим доступу https://mhd.com.ua/p1325385205-zkteco-speedface-v5ltd.html?source=merchant_center&gclid=Cj0KCQjwkZiFBhD9ARIsAGxFX8BVMeZDZyf70HCWIPI5jHwFnsqDvmm1IiH8eOePOVyOmYW6YSSZCsQaAsAmEALw_wcB
2. Hive UVF – система управления доступом автомобильного транспорта на базе IP-камеры — [Електронний ресурс] — Режим доступу https://svn.kiev.ua/p735634510-kamera-hive-uvf.html
3. MediaPipe — это фреймворк с открытым исходным кодом, разработанный Google — [Електронний ресурс] — Режим доступу https://cnx-software.ru/2019/12/26/mediapipe-eto-frejmvork-s-otkrytym-ishodnym-kodom-razrabotannyj-google/
4. Комплексная платформа машинного обучения с открытым исходным кодом — [Електронний ресурс] — Режим доступу https://www.tensorflow.org/
5. About OpenCV (Open Source Computer Vision Library) — [Електронний ресурс] — Режим доступу https://opencv.org/about/
6. Создание приложений с помощью Mediapipe — [Електронний ресурс] — Режим доступу https://habr.com/ru/post/502440/
7. On-Device, Real-Time Hand Tracking with MediaPipe — [Електронний ресурс] — Режим доступу https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html

ДОДАТОК А. Технічне завдання (копія)

Вимоги призначення для віддаленого модуля

Віддалений модуль повинен бути сконструйований з можливістю встановлення поза приміщенням.

Віддалений модуль буде займатися збором та передачею інформації на сервер, для нього потрібно забезпечити живлення з постійною напругою 5 В та струмом 2.1 А.

Для уникнення проблем з живленням при перебоях в енергопостачанні потрібна система безперебійного живлення, яка забезпечить мінімум 6 годин безперервної роботи.

Для швидкісної передачі інформації (відеопотоку) потрібно забезпечити або підключення до мережі Ethernet, або локальну мережу з сервером

Для правильного розпізнавання потрібно використовувати камеру з роздільною здатністю не менше 720р.

Для збереження інформації у випадку розриву зв'язку з сервером потрібно забеспечети пристрій цифровим накопичувачем з об'ємом пам'яті мінімум 32 Гб.

Вимоги призначення для сервера обробки зображення

Сервер буде займатися прийомом, обробкою, зберіганням інформації та прийманням рішень.

Для сервера потрібно забезпечити живлення з змінною напругою 220 В. Сервер повинен бути обладнаний блоком живлення потужністю мінімум на 450 Вт.

Для швидкого оброблення зображення сервер повинен мати відеокарту nvidia geforce gtx 1050 або краще, процесор intel i5 7400 або краще, 8 або більше гігабайт оперативної пам'яті стандарту DDR3 або краще.

Для збереження інформації жорсткий диск (або декілька) об'ємом мінімум 1 ТБ, бажане використання систем резервування типу RAID.

Для прийому інформації від віддаленого модуля потрібен мережевий адаптер з швидкістю понад 100 Мбіт/с та або підключення до мережі Ethernet, або локальна мережа з віддаленим модулем.

Для керування зовнішніми приладами потрібен мікроконтролер з виводами GPIO.

Вимоги життєздатності та стійкості до зовнішніх впливів і чинників для віддаленого модуля

Віддалений модуль повинен бути виконаний із пиле та волого захистом IP66 (повний захист від пилу та захист від струменів води під високим тиском протягом 3хв)

Вимоги конструкції

Вид виконання корпусу блочний. Корпус складається з чотирьох блоків:

1) передня кришка;
2) нижня кришка;
3) верхня кришка;
4) шарнірний блок кріплення.

Останній, в свою чергу, складається ще з трьох блоків:

1) блок кріплення до поверхні;
2) блок обертання;
3) блок горизонтального та вертикального регулювання.

В корпусі повинен бути отвір під кабелі живлення та Ethernet.

Спосіб кріплення камери та центрального процесора віддаленого модуля на гвинтах. Центральний процесор кріпиться чотирма гвинтами до нижньої внутрішньої поверхні корпусу віддаленого модуля. Камера вставляється в спеціальне кріплення на передній панелі та фіксується чотирма гвинтами.

Вимоги дизайну

Довжина корпусу 230 мм, ширина корпусу 90 мм, висота корпусу 90 мм (без врахування розмірів шарнірного кріплення).

ДОДАТОК Б. Лістинг коду розпізнавання (PYTHON)

Hand_tracking_modul

```
import cv2
import mediapipe as mp
import time
import math



class handDetector():
    def __init__(self, mode=False, maxHands=2, detectionCon=0.5,
trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.detectionCon = detectionCon
        self.trackCon = trackCon

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands,
                                        self.detectionCon, self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]

    def find_hands(self, img, draw=True):
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)
        # print(results.multi_hand_landmarks)

        if self.results.multi_hand_landmarks:
            for handLms in self.results.multi_hand_landmarks:
                if draw:
```

```
                self.mpDraw.draw_landmarks(img, handLms,
                                           self.mpHands.HAND_CONNECTIONS)
        return img

    def find_position(self, img, handNo=0, draw=True):
        xList = []
        yList = []
        bbox = []
        self.lmList = []
        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[handNo]
            for id, lm in enumerate(myHand.landmark):
                # print(id, lm)
                h, w, c = img.shape
                cx, cy = int(lm.x * w), int(lm.y * h)
                xList.append(cx)
                yList.append(cy)
                # print(id, cx, cy)
                self.lmList.append([id, cx, cy])
                if draw:
                    cv2.circle(img, (cx, cy), 5, (255, 0, 255), cv2.FILLED)
            xmin, xmax = min(xList), max(xList)
            ymin, ymax = min(yList), max(yList)
            bbox = xmin, ymin, xmax, ymax

            if draw:
                cv2.rectangle(img, (bbox[0] - 20, bbox[1] - 20),
                              (bbox[2] + 20, bbox[3] + 20), (0, 255, 0), 2)

        return self.lmList, bbox
```

```
    def fingers_up(self):
        fingers = []
        # Thumb
        if self.lmList[self.tipIds[0]][1] > self.lmList[self.tipIds[0] - 1][1]:
            fingers.append(1)
        else:
            fingers.append(0)
        # 4 Fingers
        for id in range(1, 5):
            if self.lmList[self.tipIds[id]][2] < self.lmList[self.tipIds[id] - 2][2]:
                fingers.append(1)
            else:
                fingers.append(0)
        return fingers

    def find_distance(self, p1, p2, img, draw=True):

        x1, y1 = self.lmList[p1][1], self.lmList[p1][2]
        x2, y2 = self.lmList[p2][1], self.lmList[p2][2]
        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2

        if draw:
            cv2.circle(img, (x1, y1), 15, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (x2, y2), 15, (255, 0, 255), cv2.FILLED)
            cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), 3)
            cv2.circle(img, (cx, cy), 15, (255, 0, 255), cv2.FILLED)

        length = math.hypot(x2 - x1, y2 - y1)
        return length, img, [x1, y1, x2, y2, cx, cy]
```

```
def main():
    pTime = 0
    cap = cv2.VideoCapture(1)
    detector = handDetector()
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList = detector.findPosition(img)
        if len(lmList) != 0:
            print(lmList[4])

        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime

        cv2.putText(img, str(int(fps)), (10, 70), cv2.FONT_HERSHEY_PLAIN,
3,
                    (255, 0, 255), 3)

        cv2.imshow("Image", img)
        cv2.waitKey(1)


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

ДОДАТОК В. ЛІСТИНГ КОДУ КЕРУВАННЯ RGB СВІТЛОДІОДОМ ТА РЕЛЕ

```
import cv2
import time
import numpy as np
import HandTrackingModule as htm
import math
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
import serial

def arduino_map(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min



################################
wCam, hCam = 640, 480#1280, 720
################################

cap = cv2.VideoCapture("http://192.168.0.12:8000/stream.mjpg")
cap.set(3, wCam)
cap.set(4, hCam)
pTime = 0

detector = htm.handDetector(detectionCon=0.7, maxHands=1)

devices = AudioUtilities.GetSpeakers()
interface = devices.Activate(
    IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
```

```
volume = cast(interface, POINTER(IAudioEndpointVolume))
# volume.GetMute()
# volume.GetMasterVolumeLevel()
volRange = volume.GetVolumeRange()
minVol = volRange[0]
maxVol = volRange[1]
vol = 0
volBar = 400
volPer = 0
area = 0
colorVol = (255, 0, 0)

# flag_RGB = 0
# flag_rele = 0



port = 'COM3'
try:
    realport = serial.Serial(port, int(115200))
except Exception as e:
    print(e)

tipIds = [4, 8, 12, 16, 20]

while True:
    success, img = cap.read()

    # Find Hand
    img = detector.find_hands(img,draw=True,mirror=False)
    lmList, bbox = detector.find_position(img, draw=True)
    if len(lmList) != 0:
```

```
# Find Distance between index and Thumb
length, img, lineInfo = detector.find_distance(4, 8, img,draw=False)

# Convert Volume
volBar = np.interp(length, [50, 200], [400, 150])
volPer = np.interp(length, [50, 200], [0, 100])

# Reduce Resolution to make it smoother
smoothness = 10
volPer = smoothness * round(volPer / smoothness)
```

```
if len(lmList) != 0:
    fingers = []

    # Thumb
    # if lmList[tipIds[0]][1] > lmList[tipIds[0] - 1][1]:
    if lmList[tipIds[2]][2] < lmList[tipIds[2] - 2][2]:
        fingers.append(1)
        relay_text = 'Relay ON'
        command_rele = '2,1;'
    else:
        fingers.append(0)
        command_rele = '2,0;'
        relay_text = 'Relay OFF'
```

```
            realport.write(command_rele.encode('utf-8'))

            if lmList[tipIds[1]][2] < lmList[tipIds[1] - 2][2]:
                x, y = lmList[8][1], lmList[8][2]
                x = math.fabs(arduino_map(x, 0, wCam, 0, 255))
                y = math.fabs(arduino_map(y, 0, hCam, 0, 255))
                z = math.fabs(arduino_map(bbox[2] - bbox[0], 140, 300, 0, 255))



                command_RGB = '1,' + str(x) + ',' + str(y) + ',' + str(z) + ';'
                realport.write(command_RGB.encode('utf-8'))
                RGB_text = 'Tracking ON'

            else:
                RGB_text = 'Tracking OFF'

            cv2.putText(img=img,                                text=RGB_text,
fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=1,
                        org=(int(bbox[2] + 30), int((bbox[1] + bbox[3]) / 2) - 60),
color=(0, 0, 255))
            cv2.putText(img=img,                                text=relay_text,
fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=1,
                        org=(int(bbox[2] + 30), int((bbox[1] + bbox[3]) / 2) - 30),
color=(0, 0, 255))
            cv2.putText(img=img,          text='R:          '          +          str(int(x)),
fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=1,
                        org=(int(bbox[2] + 30), int((bbox[1] + bbox[3]) / 2) ),
color=(0, 0, 255))
            cv2.putText(img=img,          text='G:          '          +          str(int(y)),
fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=1,
```

```
                org=(int(bbox[2] + 30), int((bbox[1] + bbox[3]) / 2)+30),
color=(0, 255, 0))
            cv2.putText(img=img, text='B: ' + str(int(z)),
fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=1,
                org=(int(bbox[2] + 30), int((bbox[1] + bbox[3]) / 2) + 60),
color=(255, 0, 0))



            # print(fingers)
            totalFingers = fingers.count(1)
            print(totalFingers)









        # Frame rate
        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime
        cv2.putText(img, f'FPS: {int(fps)}', (40, 50),
cv2.FONT_HERSHEY_COMPLEX,
            1, (255, 0, 0), 3)

        cv2.imshow("Img", img)
        k = cv2.waitKey(1)
        if k == 27:
            realport.close()
            break
```

ДОДАТОК Г. ЛІСТИНГ КОДУ ТРАНСЛЮВАННЯ ЗОБРАЖЕННЯ

```
import io
import picamera
import logging
import socketserver
import cv2
from threading import Condition
from http import server

PAGE="""\
<html>
<head>
<title>Raspberry Pi - Surveillance Camera</title>
</head>
<body>
<center><h1>Raspberry Pi - Surveillance Camera</h1></center>
<center><img src="stream.mjpg" width="640" height="480"></center>
</body>
</html>
"""
class StreamingOutput(object):
    def __init__(self):
        self.frame = None
        self.buffer = io.BytesIO()
        self.condition = Condition()

    def write(self, buf):
        if buf.startswith(b'\xff\xd8'):
            self.buffer.truncate()
```

```
                with self.condition:
                    self.frame = self.buffer.getvalue()
                    self.condition.notify_all()
                self.buffer.seek(0)
            return self.buffer.write(buf)


    class StreamingHandler(server.BaseHTTPRequestHandler):
        def do_GET(self):
            if self.path == '/':
                self.send_response(301)
                self.send_header('Location', '/index.html')
                self.end_headers()
            elif self.path == '/index.html':
                content = PAGE.encode('utf-8')
                self.send_response(200)
                self.send_header('Content-Type', 'text/html')
                self.send_header('Content-Length', len(content))
                self.end_headers()
                self.wfile.write(content)
            elif self.path == '/stream.mjpg':
                self.send_response(200)
                self.send_header('Age', 0)
                self.send_header('Cache-Control', 'no-cache, private')
                self.send_header('Pragma', 'no-cache')
                self.send_header('Content-Type',          'multipart/x-mixed-replace;
    boundary=FRAME')
                self.end_headers()
                try:
                    while True:
                        with output.condition:
                            output.condition.wait()
```

```
                frame = output.frame
            self.wfile.write(b'--FRAME\r\n')
            self.send_header('Content-Type', 'image/jpeg')
            self.send_header('Content-Length', len(frame))
            self.end_headers()
            self.wfile.write(frame)
            self.wfile.write(b'\r\n')
        except Exception as e:
            logging.warning(
            'Removed streaming client %s: %s',
            self.client_address, str(e))
    else:
        self.send_error(404)
        self.end_headers()

class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):
    allow_reuse_address = True
    daemon_threads = True

with picamera.PiCamera(resolution='640x480', framerate=24) as camera:
    output = StreamingOutput()

    camera.rotation = 180
    camera.hflip = True
    camera.start_recording(output, format='mjpeg')
    try:
        address = ('', 8000)
        server = StreamingServer(address, StreamingHandler)
        server.serve_forever()
    finally:
        camera.stop_recording()
```

ДОДАТОК Д. ЛІСТИНГ КОДУ ESP8266 (C++)

```
#include <Arduino.h>



#define RED_PIN D5
#define GREEN_PIN D4
#define BLUE_PIN D2
#define RELAY_PIN D8

void setup() {


  pinMode(RED_PIN,OUTPUT);
  pinMode(GREEN_PIN,OUTPUT);
  pinMode(BLUE_PIN,OUTPUT);
  pinMode(RELAY_PIN,OUTPUT);
  Serial.begin(115200);

}

void loop() {



  if (Serial.available()){
    char data[30];
    int amount = Serial.readBytesUntil(';',data,30);
    data[amount] = NULL;
```

```
int int_data[10];
int count = 0;
char* offset = data;
while (true)
{
 int_data[count++] = atoi(offset);
 offset = strchr(offset, ',');
 if (offset) offset++;
 else break;
}
switch (int_data[0])
{
case 1:
 analogWrite(RED_PIN,int_data[1]);
 analogWrite(GREEN_PIN,int_data[2]);
 analogWrite(BLUE_PIN,int_data[3]);
 break;
case 2:
 if(int_data[1] == 0){
  digitalWrite(RELAY_PIN,1);
  Serial.println("-");
 }
 if(int_data[1] == 1){
  digitalWrite(RELAY_PIN,0);
  Serial.println("+");
 }

default:
 break;
}
```

```
    }
}
```