МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри,
д-р. тех. наук, проф.

_____ І. М. Журавська

« __ » _____ 2023 р.

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

СИСТЕМА COMPUTER VISION ДЛЯ AUTOMOTIVE НА БАЗІ RASPBERRY PI

Спеціальність 123 Комп'ютерна інженерія
123 – КМР.1 – 605.21820501

Студент _____ Я. В. Смолянік
підпис
«__»_____2023 р.

Керівник доцент, канд. фіз.-мат. наук _____ С. В. Пузирьов
підпис
«__»_____2023 р.

Миколаїв – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 5
ВСТУП 6
1 АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРИ ТА ПАТЕНТНОЇ ІНФОРМАЦІЇ У СФЕРІ АПАРАТНО-ПРОГРАМНИХ МОДУЛІВ 9
1.1 Порівняння аналогічних проектів 13
1.2 Підключення відеокамер для спостереження і виявлення об’єктів 20
Висновки до розділу 1 22
2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ СИСТЕМ АПАРАТНО-ПРОГРАМНОГО МОДУЛЮ 23
2.1 Комплектуючі апаратного модулю 23
2.2 Плата Raspberry Pi 4 B 26
2.3 Інструменти для проектування програмної частини 31
2.4 Мови програмування 34
2.5 Математичні методи застосовані в використанні класифікаторів OpenCV 38
Висновки до розділу 2 39
3 АПАРАТНО-ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ COMPUTER VISION НА БАЗІ RASPBERRY PI 41
3.1 GStreamer framework 41
3.2 OpenCV бібліотека 49
Висновки до розділу 3 55
4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА ТЕСТУВАННЯ СИСТЕМИ COMPUTER VISION 57
4.1 Підключення до одноплатного комп’ютера та завантаження програмного коду 57
4.2 Створення каскадного класифікатора 64
4.3 Блок-схема роботи алгоритму для системи computer vision 72

4.4 Потенційна сфера застосування 74
Висновки до розділу 4 76
ВИСНОВКИ 77
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 79
ДОДАТОК А Програмний код апаратно-програмного комплексу 81

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI	–	Artificial intelligence
ARM	–	Advanced RISC Machine
CV	–	Computer Vision
FPS	–	Frame per second
GPU	–	Graphics processing unit
GUI	–	Graphical user interface
HDMI	–	High Definition Multimedia Interface
HTTP	–	HyperText Transfer Protocol
IP	–	Internet Protocol
MP	–	Megapixel
MS	–	Microsoft
RAII	–	Resource Acquisition Is Initialization
RAM	–	Random Access Memory
RTSP	–	Real Time Streaming Protocol
SD	–	Secure Digital
SDK	–	Software development kit
SDI	–	Snappy Driver Installer
SPI	–	Serial Peripheral Interface
SSH	–	Security Shell Protocol
TCP	–	Transmission Control Protocol
UART	–	Universal asynchronous receiver/transmitter
USB	–	Universal Serial Bus
VNC	–	Virtual Network Computing
Wi-Fi	–	Wireless Fidelity
ІЧ	–	Інфрачервоний
ОС	–	Операційна система
ШІМ	–	Широко-імпульсна модуляція

ВСТУП

Актуальністю написання магістерської роботи є формування навичок і професійних умінь щодо прийняття особистих рішень під час професійної діяльності в реальних умовах. Також важливо відточення придбаних знань на практиці та їх закріплення. Опанування сучасних технологій, методів в галузі майбутньої професії. Набуття практичного досвіду з роботи на підприємстві. Основною проблемою існуючих систем є їх досить значна часова затримка між отриманням кадру і накладання на нього фільтрів, та іншого процессінгу.

Актуальність роботи зумовлена великим розвитком комп’ютерного зору (computer vision) для різноманітних машин, зменшенням комп’ютерів та комплектуючих для обробки великого потоку даних з датчиків та камер, та їх доступністю для широкого загалу, а також популярністю сучасних automotive машин, таких як (Tesla, BMW, Audi, Porsche тощо).

Різноманітні датчики та камери на сучасних автомобілях людей надають велику об’єм даних, за допомогою яких можна налаштувати автомобіль так, щоб він зміг прорахувати наперед відповідні дії водія, для запобігання аварій, та збереженню життя водію та навколишніх людей.

Однією з таких задач є самостійне керування автотранспортом за допомогою камер і систем доповненої реальності для з розпізнавання об’єктів, людей та розмітку на трасі. Усе це можливо реалізувати на одноплатному комп’ютері і це дає можливість швидко отримувати та обробляти вхідні відео дані з декількох камерних модулів та датчиків у одному спеціально написаному застосунку, і мінімальним зайнятим місцем у автомобілі. Актуальність кваліфікаційної роботи полягає в розробці апаратно-програмного комплексу який дозволяє мінімізувати затримку між отриманням кадрів і їх процессінгом.

Мета: Проектування апаратно-програмного модулю для збору та обробки відео потоку на основі Raspberry Pi і оптимізація витраченого часу на обробку кадру у фреймворку для застосування технології computer vision.

Об’єкт: Технологія інтеграції мікропроцесорної системи зі спеціалізованим фреймворком обробки кадрів.

Предмет: Система computer vision для automotive на базі Raspberry Pi.

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

1. Виконати огляд проектів з існуючих систем комп’ютерного зору.
2. Вдосконалити алгоритм потокової обробки даних від апаратного модуля до фреймворку з обробки кадру.
3. Натренувати апаратно-програмний комплекс на розпізнавання певного класу об’єктів .
4. Протестувати прототип апаратно-програмного комплексу у реальному часі.

Практичне значення отриманих результатів:

1. Зручний і компактний одноплатний комп’ютер для розпізнавання об’єктів та людей на борту automotive.
2. Широкі можливості для інтеграції з різноманітними датчиками, наприклад у системі autoparktronic. Наприклад, це дасть можливість розробити автоматичні автономне паркування з урахуванням небезпек, таких як найближчий автомобіль, людина, бордюр, та обмаль місця.

Наукова новизна

1. Швидка обробка потокових даних на одноплатному комп’ютері з обмеженими ресурсами.
2. Вдосконалено методику розпізнавання об’єктів у машинах automotive.
3. Вдосконалення формату передачі даних для більш швидкої обробки та синхронізації з реальним часом.

Гіпотеза дослідження полягає у тому, що існують системи для обробки великих масивів відео даних у реальному часі при використанні, невеликого одноплатного комп’ютера та правильно підібраного фреймворку.

Методи досліджень

Беручи до уваги цієї магістерської роботи, поставлені завдання, об’єкт та предмет дослідження, використано низку методів наукового пізнання, зокрема: метод передачі даних на обробку, побудови патернів для розпізнавання об’єктів, синхронізації кадру з реальним часом.

Робота пройшла апробацію під час XXV Всеукраїнської науково-практичної конференції «Могилянські читання» (Миколаїв, 07–11 листопада 2022 р.).

Публікації. Основні положення та результати магістерської роботи опубліковані у збірнику матеріалів XXV Всеукраїнської науково-практичної конференції «Могилянські читання–2022»[21].

1 АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРИ ТА ПАТЕНТНОЇ ІНФОРМАЦІЇ У СФЕРІ АПАРАТНО-ПРОГРАМНИХ МОДУЛІВ

Незважаючи на дві різні виникаючі думки, немає реальної різниці між розпізнаванням об’єкта та розпізнаванням наданого зображення. Фактично, обидва вони стосуються технологій, які можуть розпізнавати певні цільові суб’єкти за допомогою певних алгоритмів, таких як глибоке навчання. Вони тісно пов’язані з комп’ютерним зором, який ми визначаємо як мистецтво та науку, що дозволяє комп’ютерам оброблювати та розуміти зображення і відділяти об’єкти на них [24].

Розпізнавання об’єктів складається з розпізнавання, ідентифікації та визначення місцезнаходження об’єктів на зображенні із заданим ступенем упевненості.

Цей процес виділяє чотири основні завдання для ефективної роботи алгоритму:

1. Класифікація.
2. Тегування.
3. Виявлення.
4. Сегментація.

Одним з найголовніших завдань у розпізнаванні об’єктів є визначення того, що знаходиться на зображенні та з яким рівнем впевненості розпізнаний об’єкт. Зазвичай вказується відсоток ймовірності на рисунку, його приклад показаний нижче. [27] (рис. 1.1).

Рисунок 1.1 – Класифікація та тегування

Класифікація сама по собі розпізнає лише один клас заданих об’єктів, а додавання тегів може розпізнавати декілька для певного зображення. Під час класифікації алгоритм запам’ятає лише те, що на рисунку собака, ігноруючи всі інші класи. Але під час додавання тегів він намагатиметься повернути всі найкращі класи, які відповідають зображенню.

Визначившись з об’єктом на зображенні потрібно виділити об’єкт. Для цього використовується виявлення та сегментація.

Виявлення видає прямокутник, який також називають обмежувальною рамкою, де знаходяться об’єкти. Це дуже надійна технологія, схильна до незначних помилок і неточностей. Однак сегментація визначає об’єкти для кожного пікселя на зображенні, в результаті чого створюється дуже точна карта. Але сегментація потребує тривалого навчання нейронної мережі для більш точного результату.

На основі AI використовується дуже велика кількість електронних речей для спрощення роботи у різноманітних сферах діяльності. Наприклад далі показана гістограма використання штучного інтелекту у різних системах світу, на відмінок від людей (рис. 1.2).

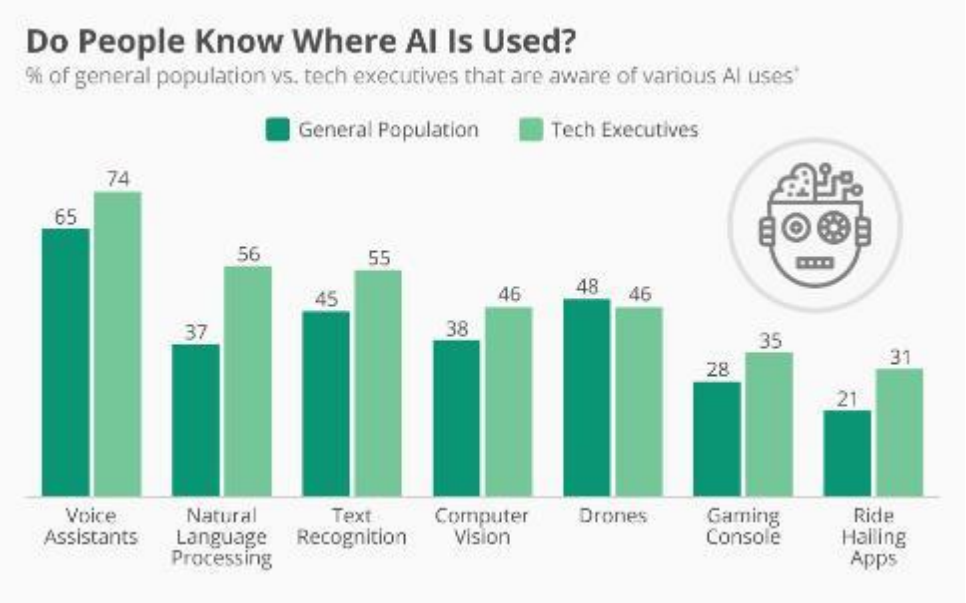


Рисунок 1.2 – Тенденції використання штучного інтелекту у світі

За допомогою сучасного смартфону та відповідним застосункам можна знімати на камеру з використанням інфрачервоного датчику та розпізнавати різноманітні об’єкти, які вже заложенні у програмний код та сама система доповненої реальності вже навчена на спеціальних класах[22]. Але це більш для розваг та відпочинку, та й коштують такі застосунки дорого. У цьому проєкті використовується модель Raspberry Pi 4 та модуль камери для відслідковування об’єктів на дорозі та розпізнавання їх. Це допоможе уникнути великої кількості аварій та допомога у прийнятті рішень у критичній ситуації.

Комп’ютерний зір є основою технології автономних автомобілів. Автомобілі використовують алгоритми виявлення об’єктів у поєднанні з сучасними камерами та датчиками, які можна групувати у масиви та використати з одною чи декількома платами, щоб аналізувати оточення і передавати відповідні дані з датчиків в режимі реального часу та обробляти і розпізнавати речі, які зазвичай знаходяться на шляху транспортного засобу, наприклад дорожні знаки, пішоходи, відбійники, дорожні смуги та інші транспортні засоби, для безпечного руху по дорозі у місті чи шосе. [28] Стрімкий розвиток автомобільних камер, датчиків і комп’ютерного зору

наблизив їх як ніколи до стандартів безпеки, заслуживши визнання громадськості та досягнувши комерційної доступності. Тим є приклад таких концернів як Tesla, BMW, Audi та ін. (рис. 1.3).

Рисунок 1.3 – Концепт комп’ютерного зору об’єктів на дорозі

Глобальний автомобільний сектор стикається з різними проблемами, включаючи проблему дефіциту мікросхем для створення плат, юридичні проблеми тощо. Керівники шукають різні цифрові рішення для подолання цих проблем, і одним із цих рішень є комп’ютерний зір [29]. Саме тому дана робота направлена для поліпшення цієї проблеми, тим що можна використовувати популярну плату і відповідні датчики для проектування майже ідентичної системи розпізнавання об’єктів на дорозі і проектування системи передбачень у критичних ситуаціях на дорозі. І набагато дешевше.

Комп’ютерний зір дозволяє автомобільному сектору стати розумнішим, безпечнішим і ефективнішим. Для інвестування великої кількості грошей у дану галузь, треба перш за все висвітити основні переваги даної системи і її перспективність у розвитку. Прогнозується, що світовий ринок штучного інтелекту, найбільшу частку якого становить комп’ютерний зір, зросте з 1,5 мільярда доларів США у 2022 році до 7,6 мільярда доларів США до 2028 року.

Один із самих популярних шляхів використання технології комп’ютерного зору – це автомобілі з авто керуванням. Від звичайних седанів, до важких логістичних вантажівок з підтримкою AI, які роблять прорив у автомобільному секторі.

Сучасні автомобілі оснащені датчиками і камерами які працюють у фоновому режимі, та виявляють небезпеку на дорозі, та різноманітні об’єкти [30] (рис. 1.4).

Рисунок 1.4 – Системи датчиків обробки інформації

1.1 Порівняння аналогічних проектів

Велика кількість відкритих (opensource) проектів для оснащення техніки яка використовує computer-vision знаходяться на таких популярних платформах як Kickstarter та ArduinoHub. Далі буде розглянуто декілька прикладів з них.

Перший проект для демонстрації – це робот який стежить за обличчям на основі одноплатного комп’ютера Raspberry Pi 4 Model B, RP Camera module, Arduino Nano R3, SparkFun Dual H-Bridge motor drivers L298 та програмним забезпеченням для розпізнавання образів – OpenCV. Raspberry Pi використовує фреймворк OpenCV для впізнання обличчя та слідкування за ним за допомогою камери. Arduino використовується у якості драйвера

двигуна. Коли програма дізнається де розташовані обличчя, Raspberry Pi використовує USB-порт, для пе

\редачі сигналу Arduino, як треба рухати двигуни. (рис. 1.5).

Рисунок 1.5 – Програмування слідкуючого робота за допомогою Raspberry Pi

На рис. 1.6 показано також створений веб-інтерфейс для керування роботом в ручну.

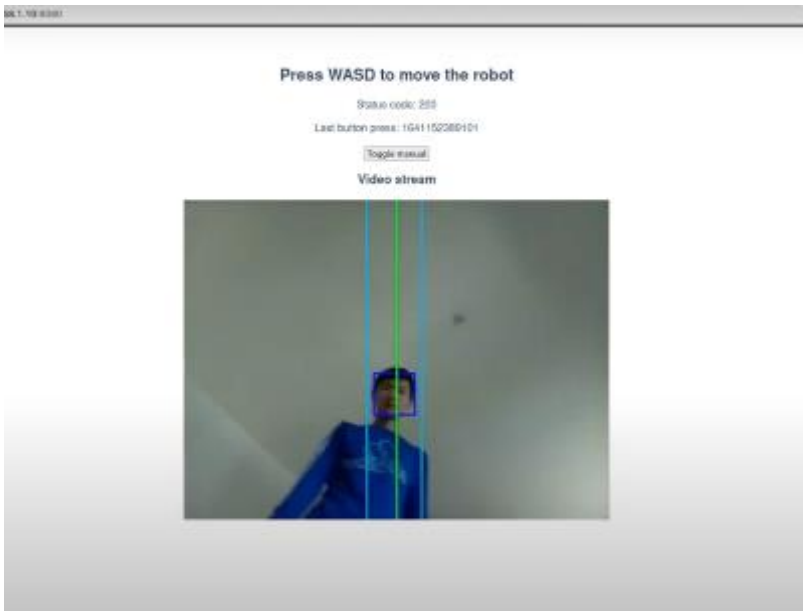


Рисунок 1.6 – Веб-інтерфейс керування

Він базується на підніманні веб-серверу за допомогою кастомної бібліотеки на мові програмування Python, та для побудови самого інтерфейсу був використаний фреймворк Vue.js.

У наступному проекті представлена система, за допомогою якої можна вираховувати перешкоди які знаходяться перед камерою [8]. Ця система корисна у таких механізмах як парктронік та ін. Для реалізації цього проекту також знадобилися веб-камери та фреймворк розпізнавання об’єктів OpenCV. (рис. 1.7).

Рисунок 1.7 – Робот для виявлення перешкод

Метод передбачає захоплення зображення, перероблюючи його і представляючи як відтінок сірого, злегка розмиття, а також для виділення країв на зображенні було використано методу виявлення країв. Використання зробленого зображення з розпізнаним краєм, починаючи з лівого боку та уздовж ширини зображення з відповідними інтервалами, сканується від нижньої частини зображення, доки не буде досягнуто білого пікселя, що вказує на перший край який зустрівся (рис. 1.8).

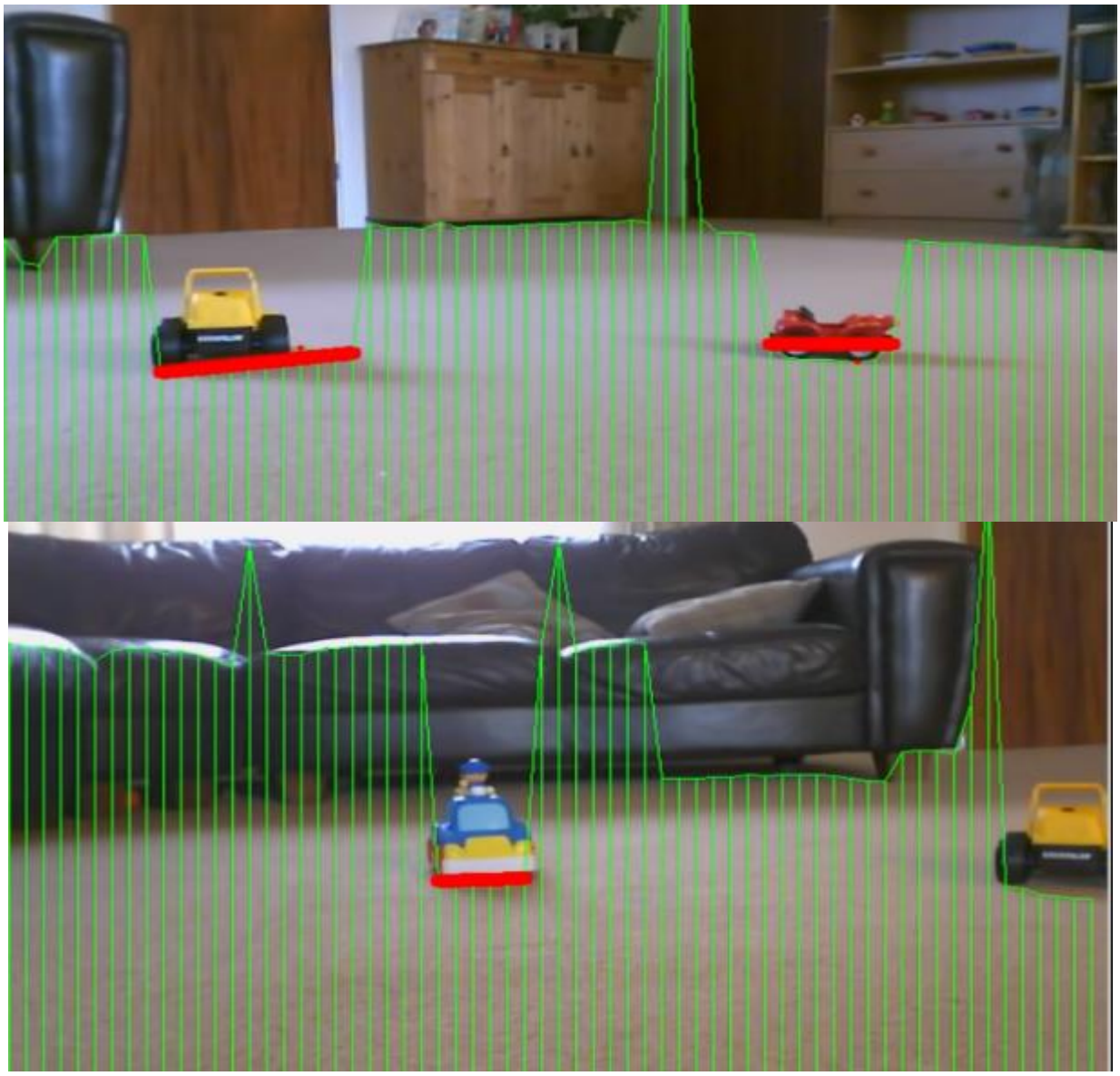


Рисунок 1.8 – Виділення перешкод та їх висоти

Також генерується масив, який містить координати перших ребер, знайдених перед роботом. Використовуючи цей масив, можна шукати зміни в напрямку нахилу країв, які чітко визначають де знаходиться об’єкт перед камерою. На рисунках які вказані вище, ігнорується більшість деталей які знаходяться у верхній частині рисунку. Все знайдене, буде досить далеко від робота, щоб що сильно турбуватися про це. Дані виділення змінюватимуться залежно від кута нахилу камери. Якщо камера дивиться вниз, до землі, відповідно все в сцені може бути сприйнятим за перешкоди. Виявивши зміни нахилу, проходить сканування в обох напрямках, щоб спробувати знайти

край об’єкта, на який вказує різка зміна значень у масиві найближчих країв. Відповідно для іншого середовища може знадобитися додаткове налаштування проекту.

Третій проект базується на розпізнаванні сміття, та його прибиранні. За основу також взято плато Raspberry pi та фреймворк OpenCV(рис. 1.9).

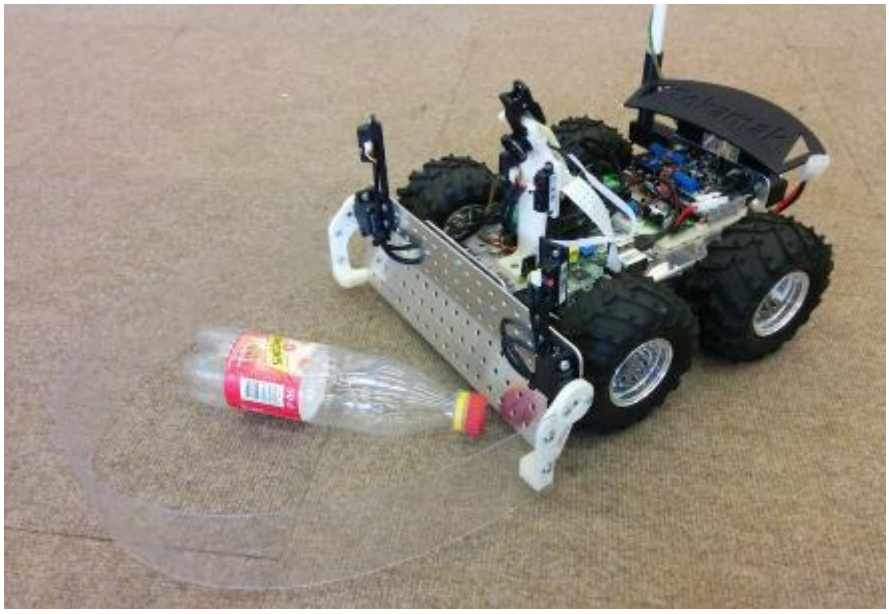


Рисунок 1.9 – Робот для виявлення сміття

Усе виконано за допомогою Raspberry Pi з Arduino (клоном) для керування мобільною платформою. Написаний програмний код для розпізнавання пляшок базується на каскадному класифікаторі Haar який використовує відповідні функції для виявлення об’єктів (у цьому випадку пляшок) через OpenCV. [9] Основний код на Raspberry Pi написаний на язику програмування Python. Дане рішення пропонувало швидший час розробки, ніж C/C++, але з більшими витратами.

Далі на рисунку можна побачити детальну схему, по якій працює даний проект (рис. 1.10).

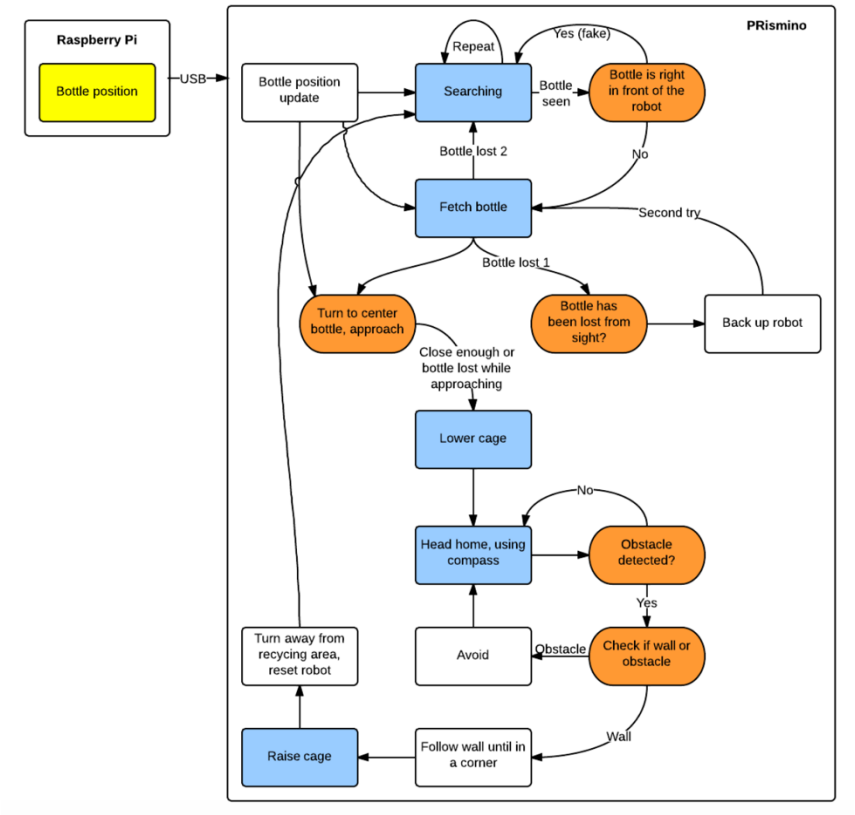


Рисунок 1.10 – Схема роботи пристрою

Робот обходить перешкод, доки камерою не буде виявлено пляшку на її шляху. Після виявлення пляшки починається обробка події та за допомогою написаного скрипту Python, запущеного на Raspberry Pi, позиція сміття пересилається на плату Arduino, яка містить основну програму в керуванні моторами, яка переходить до стану взяття пляшки.

Після отримання пляшки розпізнавальним роботом, алгоритм подій буде таким: робот відправляється у пункт переробки, шо зазначено точку у програмному коді (кут карти) використовуючи компаса, і якщо на шляху постає стіна, робот буде йти за нею, поки не буде виявлено кут та після цього робот має відпустити пляшку. (рис. 1.11).

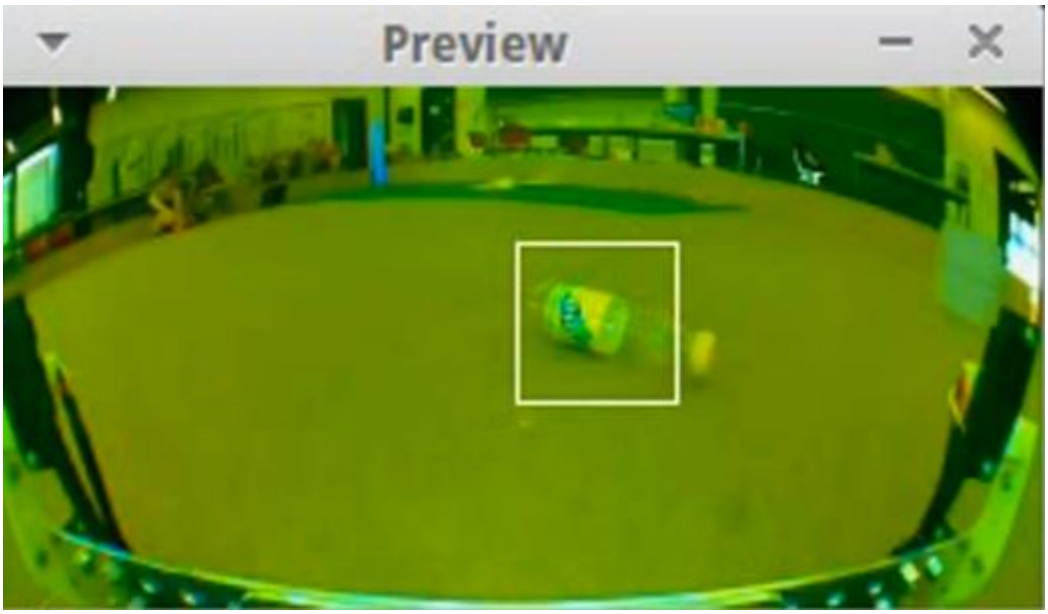


Рисунок 1.11 – Виявлення пляшки для збору

Початковий план полягав у використанні 2 ІЧ-датчиків і камери для виявлення перешкод, але камера виявилася повільною для даної задачі і наступні тести виявили, що більша кількість ІЧ-датчиків була б простішим рішенням цієї задачі і не поступалося би ефективністю, тому було обрано використовувати 4 ІЧ-датчики і одну камеру лише для виявлення пляшок.

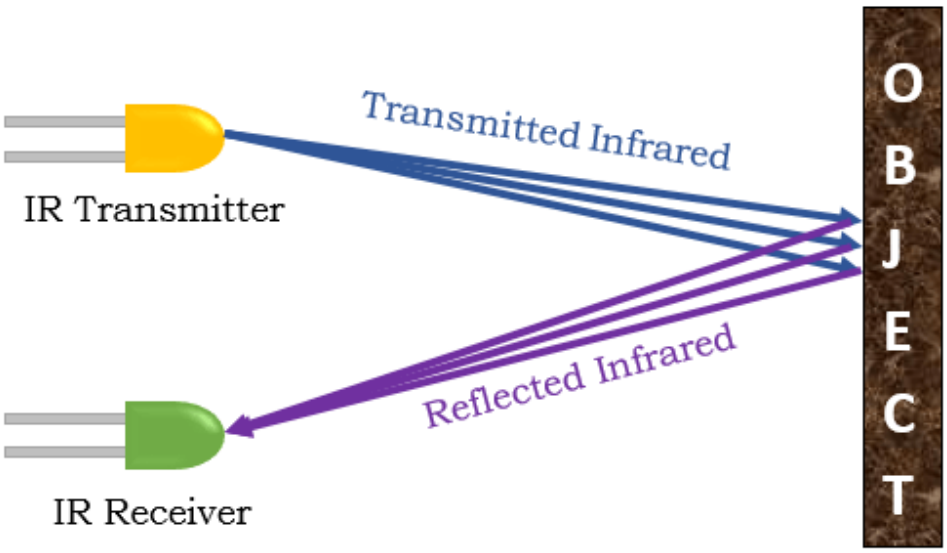


Рисунок 1.12 – Робота IR-датчика

1.2 Підключення відеокамер для спостереження і виявлення об’єктів

Багато користувачів Raspberry Pi помічають, що на платі присутній тільки один слот для підключення камери. Процесори на сучасних моделях які представленні для купівлі, таких як Pi 3 і Pi 4, йде підтримку двох камер з заводу, але лише одна з них доступна на стандартних платах Raspberry Pi. Не дивлячись на те, що одноплатні модулі Raspberry Pi дозволяють використовувати обидва, вони все ще потребують налаштованих плат носіїв і не так популярні у просторі розробників.

Найбільш розповсюдженими проекти де можна використовувати масив камер:

– Зйомка кругового відео на 360 градусів.
– IP-камери для спостереження за територією яка охороняться.
– Створення системи для машин які розпізнають об’єктів на дорозі.
– Розпізнавання та слідкування за об’єктами у переміщенні.

Що робити, якщо вам потрібно кілька камер для відповідного застосування? Найпростішим рішенням буде з’єднання кількох плат Raspberry Pi і камер. Це можливо, але це дійсно збільшує складність підключення усіх елементів та проводки, електроспоживання та зайняття великої кількості міста. Також вартість розгортання такої системи може значно перевищити очікування. (рис. 1.13).

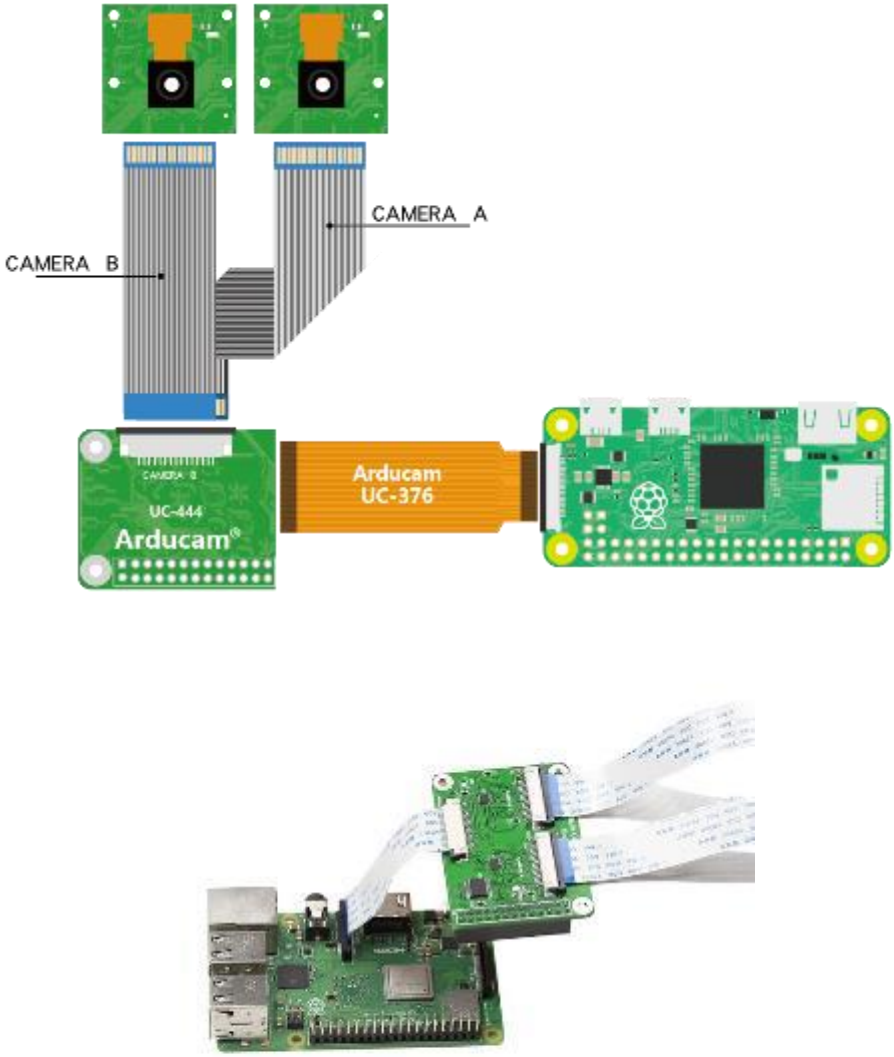


Рисунок 1.13 – Схематичне та реальне підключення масиву камер

Ключовим моментом і фундаментальною теорією для побудови цих багатокамерних адаптерних плат є мультиплексування. Raspberry Pi дозволяє підключати одну камеру та розпізнавати лише цю підключену камеру, тому перемикання між різними каналами дозволяє передавати відеопотоки різних камер на бортовий порт плати CSI RPi по одному каналу. Таким чином, Raspberry Pi все ще підключається до однієї камери, але вивід йде через кілька камер які будуть підключені до адаптеру.

Висновки до розділу 1

У цьому розділі було досліджено та розглянуто невелику кількість схожих проектів на основі Raspberry Pi та модулями камер, також виконано порівняння підключення однієї камери та масиву. Також було показано застосування додаткових компонентів, таких як IR-модуль.

Показаний основний принцип роботи камери з розпізнаванням речей, його взаємодія з предметами на дорозі, і описані основні пункти переваг даного модуля над іншими моделями.

Аналітичний огляд схожих проектів, які побудовані на апаратно-програмному модулі Raspberry Pi дав змогу виділити для розуміння важливі технічні нюанси при роботі з даним одноплатним комп’ютером, побачити його потенціал у майбутньому розвитку, та ефективну взаємодію з відповідними технологіями, як комп’ютерний зір, переведення в автоматизацію та ін.

Висвітлена основна концепція побудови одноплатного модуля для розпізнавання об’єктів за допомогою computer vision у системах automotive, описані методи за допомогою яких можна під’єднати масиви камери до однієї плати.

Зваживши усі переваги та недоліки даного одноплатного комп’ютера, можна зрозуміти що Raspberry Pi це дуже енергоефективна та потужна плата в усіх відповідних аспектах. За не дуже велику суму можна отримати чудовий одноплатний комп’ютер, який спроможний справлятися с великим спектром важких задач і використовувати відповідні модулі для цього.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ СИСТЕМ АПАРАТНО-ПРОГРАМНОГО МОДУЛЮ

Першою частиною розробки та проектування даного модуля є написання відповідного алгоритму для роботи, обирання відповідних фрейморків та язиків програмування, які будуть коректним чином взаємодіяти друг з другом і вписуватимуться у бізнес-логіку даного проекту. Це відповідно впливає на працездатність і саму швидкість роботи програми, та модулю.

2.1 Комплектуючі апаратного модулю

Проект computer vision для automotive базується на стрімінговому потоці та розпізнаванні об’єктів перед собою, або перешкод за допомогою відповідних датчиків та камер. Апаратною частиною для даного пристрою можна обрати наступні одноплатні комп’ютери на базі:

- Raspberry Pi 4
- Asus Tinker Board S R2.0
- NVIDIA Jetson Nano Developer kit
- Odroid-XU4

У даний час одним з найпоширеніших одноплатних комп’ютерів з вбудованою підтримкою 4K та швидким процесором для обробки поточної інформації є Raspberry Pi 4. Побудована апаратно-програмний модуль на одноплатному комп’ютері Raspberry Pi 4 отримує потокове відео, та оброблює відповідний фрейм для визначення об’єктів та перешкод у системах automotive (рис. 2.1).

Рисунок 2.1 – Одноплатний комп’ютер Raspberry Pi 4 Model B

Плата Raspberry Pi 4 Model B це покращена плата у порівняні з Raspberry Pi 3 моделі B+ попереднього покоління [2]. Швидкість процесора була значно збільшена, вдосконалена мультимедіа та пам'ять, а також оновлені порти підключення периферії. Далі на рисунку можна побачити порівняльну таблицю двох плат Pi3 та Pi4, яка представлена нижче. (рис. 2.2).

Features/Specs	Raspberry Pi 4 Model B	Raspberry Pi 3 Model B+
Release Date	24th June 2019	14th March 2018
SoC Type (Processor)	Broadcom BCM2711 (with metal cover)	Broadcom BCM2837B0 (with metal cover)
Core Type	Cortex-A72 64-bit (ARMv8)	Cortex-A53 64-bit (ARMv8)
No. of Cores	Quad-Core	
GPU	VideoCore VI	VideoCore IV
Multimedia	H.265 decode (4Kp60) H.264 decode (1080p60) H.264 encode (1080p30) OpenGL ES 1.1, 2.0, 3.0 Graphics	H.264, MPEG-4 decode (1080p30) H.264 encode (1080p30) OpenGL ES 1.1, 2.0 Graphics
CPU Clock	1.5 GHz	1.4 GHz
Memory/OS storage	microSD	
RAM	LPDDR4: 1GB, 2GB, 4GB and 8GB options	LPDDR2 1GB
Ethernet	True Gigabit Ethernet	Gigabit over USB 2.0 (Max 300Mbps)
USB Port	2 x USB 3.0 + 2 x USB 2.0	4 x USB 2.0
HDMI	2 x micro HDMI support Dual Display	1 x full size HDMI
WiFi	802.11 b/g/n/ac (2.4GHz+5GHz & Shielded)	
Bluetooth	5.0 + BLE (Shielded)	4.2 + BLE (Shielded)
Antenna	PCB Antenna (Similar to Rpi Zero W)	
GPIO	40 pins (Fully backwards-compatible with previous boards)	
Operating System	Raspbian (> 24 June 2019)	Raspbian (> March 2018)
Dimension	85mm x 56mm	
Power Input	5V via USB Type C (upto 3A) 5V via GPIO header (upto 3A) Power over Ethernet, requires PoE HAT	5V via USB Micro B (upto 2.5A) 5V via GPIO header (upto 3A) Power over Ethernet, requires PoE HAT

Рисунок 2.2 – Порівняння та відмінності плат Raspberry Pi4 і Pi3

Pi 4 оснащений такими передовими функціями, як вхід живлення USB-C, два відеовиходи, з можливістю відтворення картинки на двох моніторах у 4K, і вибір доповнень до оперативної пам’яті (RAM) — вперше для Raspberry Pi.

2.1.1 Альтернативні версії Raspberry Pi

У підрозділі описано варіацію одноплатних комп’ютерів які можна використовувати у альтернативному випадку, замість Raspberry Pi 4 [1].

1. Raspberry Pi 4 Model B - З точки зору кінцевого використання, важливо відзначити, що швидкість та продуктивність останнього одноплатного комп’ютера Raspberry Pi 4 Model B еквівалентна продуктивності комп’ютера x86 початкового рівня. Частота процесора у Pi 4 1.5GHz у порівнянні з моделлю Pi 3 1.4GHz, що дає нам перевагу у швидкості обробки поточних даних. Також плата Raspberry Pi має 40-контактний розширений роз’єм GPIO, який ідеально підходе для підключення додаткових датчиків або їх масиву. Також Raspberry Pi 4 має два порти USB 3.0 і USB 2.0. Ще на платі присутній USB - type C (5V постійний струм), який можна використовувати для живлення плати, при підключенні більш потужних датчиків.

2. Asus Tinker Board S R2.0 – ця плата має низку приємних функцій, які роблять її ідеальною для любителів. Особливо виділяється кольорове маркування пінів GPIO, яке дозволяє легко розпізнавати відповідні контактні для підключення. Tinkerboard має кілька функцій, які порівнюються з Pi, зокрема 6-ядерний процесор, у якого два ядра що працюють на частоті 2 ГГц, швидший графічний процесор і 16 ГБ вбудованої пам’яті.

3. NVIDIA Jetson Nano Developer kit – являється одним із найкращих одноплатних комп’ютерів (SBC) для розробки додатків зі штучним інтелектом та робототехніки. Модель на 2 ГБ скорочує трохи обсяг оперативної пам’яті, але зберігає 128-ядерний графічний процесор на базі

NVIDIA Maxwell і чотирьох ядерний процесор ARM A57. Багатозадачність на основі графічного інтерфейсу користувача (GUI) може зменшуватися [4].

4. Odroid-XU4 – XU4, який вийшов у 2015 році, є восьми ядерною платою ARM, яку Hardkernel планує продовжувати виробляти до 2023 року. XU4 використовує процесори ARM (32-розрядні від Samsung). XU4 може працювати з Android нативно. Плата Odroid може не мати вбудованого Wi-Fi, але XU4 має Gigabit Ethernet, якого має бути більш ніж достатньо для потокового передавання медіа, обміну файлами та веб-перегляду.

2.2 Плата Raspberry Pi 4 B

Для даного проекту було обрано камеру RasPi cam v2 з сенсором Sony IMX219. [4] Версія 2 є простішою, швидшою та кращою в багатьох важливих аспектах, тому головна перевага тепер це те, що можна робити фото у роздільній здатності 8 Мрх. Це показник важливий, отже фото будуть чіткого відображення з можливістю точно відобразити об’єкти і перешкоди які будуть попадати у поле зору об’єктиву без піксилізації та розмитого зображення [23] (blur) (рис. 2.3).

Рисунок 2.3 – RasPi cam V2

Характеристики до камери наступні:

– Повна назва : Raspberry Pi Camera Board V2
– Роздільна здатність : 8MP (до 3280×2464)
– Сенсор : Sony IMX 219 PQ CMOS, ¼ inches
– Підтримка відео форматів : 1080p (30fps), 720p (60fps), 640 × 480p (90fps)
– Лінза : 33 mm
– Діафрагма : f / 2

Raspberry Pi 4 — це маленький одноплатний комп’ютер, де всі його компоненти, від пам’яті до USB-портів, поміщаються на одній друкованій платі без додаткових плат чи аксесуарів. Розміри даної плати 2,2 на 3,4 дюйма і приблизно 0,6 дюйма у висоту. На додаток до 4 ГБ оперативної пам’яті на борту стоїть чотирьох ядерний процесор Broadcom, що працює на частоті 1,5 ГГц, чотири порти USB Type-A, два відеовиходи micro HDMI, один роз’єм гігабітного Ethernet і приймач сигналу для 802.11ac Wi-Fi і Bluetooth 5.0.

Майже всі модулі використовують 40-контактний роз’єм вводу/виводу загального призначення (GPIO), який може забезпечувати живлення усіх підключених датчиків, а також надсилати та збирати дані з них.

Характеристики плати Raspberry Pi 4 наведено нижче у таблиці 2.1 (посилання на оригінал будуть у переліку посилань магістерської роботи)[3]

Таблиця 2.1 – Характеристики плати Raspberry Pi 4

Бездротова мережа	802.11 ac; Bluetooth 5.0
Центральний процесор	Broadcom BCM2711 (Cortex-A72)
Тактова частота	1.5GHz
Пам’ять	4GB LPDDR4
Інтерфейси	40-pin header (GPIO), USB 3.0, 2.0, type-c
Камера	Sony IMX 219

Графіка	ARM VideoCore VI
Вивід відео	Micro HDMI x2
Підтримка кодеків	OpenGL ES3, h.265, h.264
Дротова мережа	Gigabit Ethernet

Завдяки зміні архітектури, з’явилися наступні вдосконалення. Контролер Ethernet тепер має власний виділений інтерфейс, а порти USB тепер підключені через лінію PCI Express Gen 2, забезпечуючи загальну пропускну здатність 4 Гбіт/с. достатньо для двох портів USB 3.0 і двох портів USB 2.0.

2.2.1 Особливості плати та конекторів

У Raspberry Pi є 40 піновий конектор, який дозволяє підключитися до датчиків, світильників, двигунів та інших пристроїв. Pi використовує 40- або 26-контактний роз’єм залежно від моделі, тому важливо розуміти, як ці виводи розташовані та позначені.

Крім того, плата має додаткові і сучасні конектори які дозволяють під’єднати різноманітну переферію, таку як монітор для виведення картинки у високій якості. Далі показано можливе під’єднання додаткових модулів, для вдосконалення системи розпізнавання об’єктів та перешкод, а також для більш зручного користування звичайній людині(рис. 2.4).

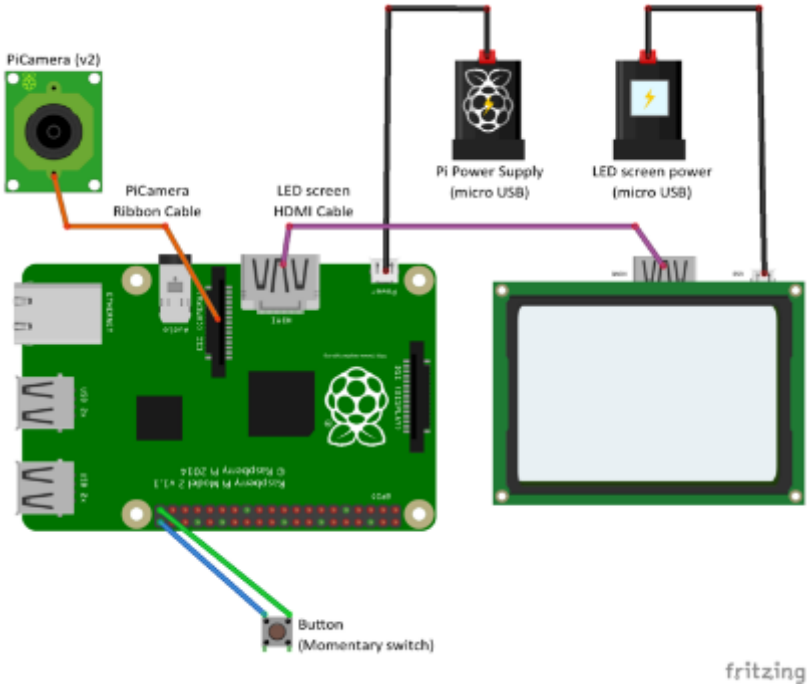


Рисунок 2.4 – Підключення переферії до Raspberry Pi 4

Дана плата має широкий список функцій:

– Процесор Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz

– Швидка оперативна пам'ять 2 GB, 4 GB або 8 GB LPDDR4-3200

– Сучасний спектр частот бездротової мережі 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE

– Можливість підключення Gigabit Ethernet

– Чотири порти 2 x USB 3.0 та 2 x USB 2.0

– 40 піновий GPIO header (зі зворотною сумісністю)

– Два мікро HDMI порти (підтримка до 4kp60)

– Двоканальний порт дисплею MIPI DSI та порт камери MIPI CSI

– Підтримка найпопулярніших кодеків H.265 (4kp60 декодування), H264 (1080p60 декодування, 1080p30 кодування)

– Графічний інтерфейс OpenGL ES 3.1, Vulkan 1.0

– Підтримка Micro-SD для запису та считування інформації

– Можливість під’єднати 5V DC через USB-C або GPIO пін

– Також живлення через порт Ethernet

Також на платі Raspberry Pi 4 встановлений ряд пінів для різних задач і підключення модулів:

– 3В (на 2 контактах)
– 5В (на 2 контактах)
– Земля (на 8 контактах)
– Вхід і вихід загального призначення
– ШІМ (широтно-імпульсна модуляція)
– I2C
– I2S
– SPI
– Серійний

Порядок пінів та їх значення показано на рис. 2.5

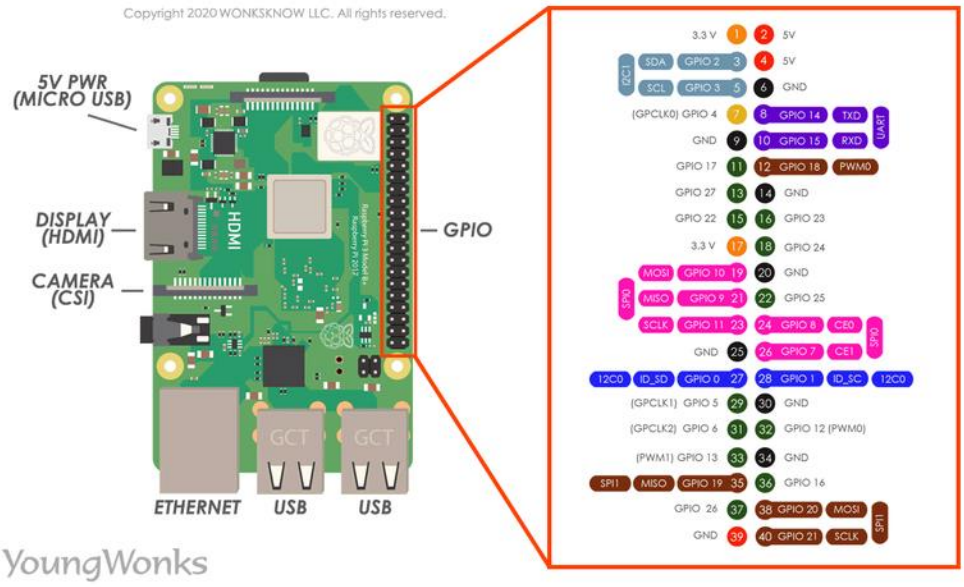


Рисунок 2.5 – Розпіновка плати та їх значення

Raspberry Pi 4 все ще має 40 контактів GPIO, але має кілька додаткових доступних з’єднань I2C, SPI та UART.

GPIO — найпростіший, але доступний аспект Raspberry Pi. У багатьох сучасних платах від нього вже відмовляються, хоча він і досі має велику популярність у підключенні модулів. Виводи GPIO є цифровими, що дають йому мати два основні стани: нуль або одиницю (off or on). Вони можуть

мати напрямок для отримання або надсилання струму (введення, вихід відповідно), а також мати контроль станів і напрямок контактів за допомогою мов програмування, таких як Python, JavaScript, node-RED тощо.

Більшість контактів у заголовку підходять безпосередньо до чіпу Broadcom. Важливо ретельно проектувати компоненти, які будуть приєднуватися, що би не пошкодити плату.

2.3 Інструменти для проектування програмної частини

Для написання програмного коду і продумування його алгоритму використано програмне середовище Microsoft Visual Studio Code.(рис. 2.6).

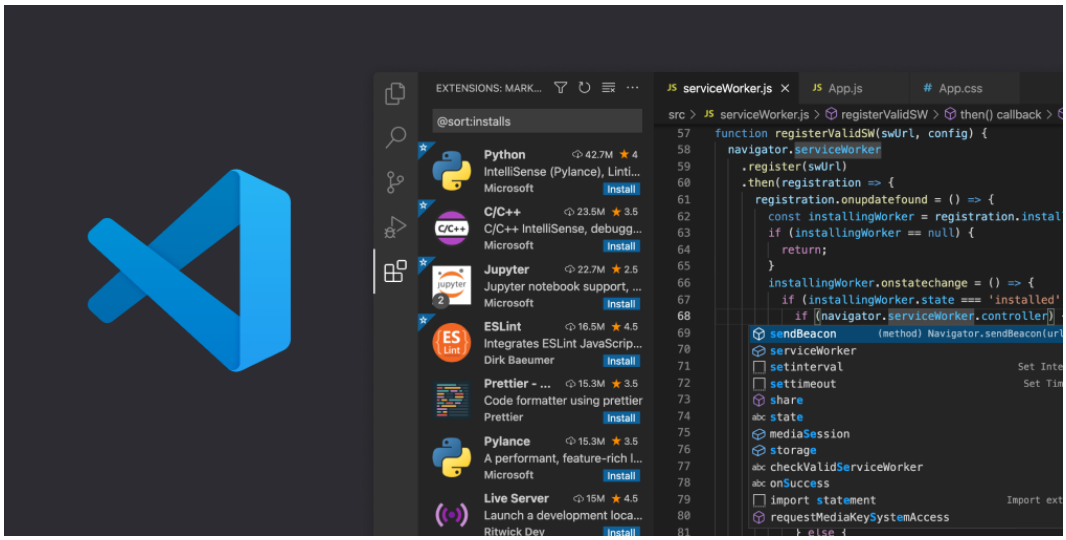


Рисунок 2.6 – Програмне середовище VS Code

Visual Studio Code — легкий і дуже потужний редактор вихідного коду, доступний для ОС Windows, macOS, Linux і Raspberry Pi. Також можна використовувати через браузер. Даний редактор йде з вбудованою підтримкою JavaScript, TypeScript і Node.js і має масштабну бібліотеку розширень для інших мов програмування (таких як C++, C#, Java, Python, Rust і Go), середовища виконання (.NET і Unity), середовища (такі як Docker і Kubernetes). Даний редактор знадобиться для виконання завдання по написанню алгоритма розпізнавання моделей на мові Python, за підтримки бібліотеки OpenCV. Для більш ефективної та швидкої роботи,

використовується розширення до мови Python, яке встановлюється у самому редакторі коду (рис. 2.7).

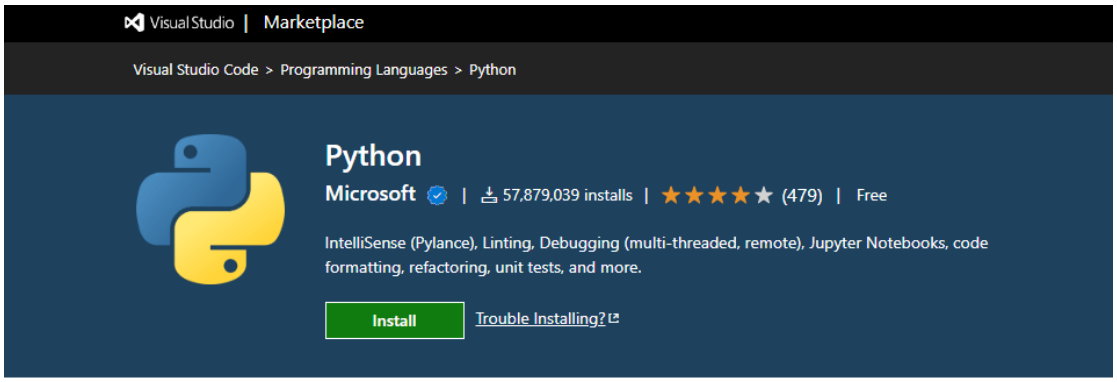


Рисунок 2.7 – Розширення для мови програмування Python

VScode автоматично запропонує це розширення, при створенні файлу з розширенням .py. Функція IntelliSense надає корисні фітчі, як автозавершення коду, навігація та перевірка синтаксису. Також при встановленні автоматично завантажується Pylance, яке надає підтримку при розробці, і розширення Jupyter для використання блокнотів Jupyter. Інші можливості цього розширення дають включають налагодження коду, форматування, рефакторинг і автоматичне перемикання між різними середовищами Python.

Також для проектування програмної частини під Raspberry Pi знадобляться наступні програми. Одної з них буде PuTTY. PuTTY - безкоштовний інструмент, написаний мовою C, який дозволяє легко встановити SSH з’єднання з пристроєм віддаленим від користувача, а також мати повний доступ до нього і його файлової системи. Досить захищене з’єднання дає надію на стабільну роботу і без стороннього втручання.

Цей інструмент знадобиться для підключення обраної плати Raspberry Pi, а також можливістю користуватися графічним інтерфейсом, та написанням самих програм у відповідному терміналі PuTTY і запуском скриптів написаних мовою програмування Python.

Secure Shell або Secure Socket Shell — це мережевий протокол, задача якого полягає у тому, щоб для двох пристроїв досягати цих речей:

спілкуватися та обмінюватися даними. Крім того, протокол SSH також шифрує дані, що робить його ідеальним для незахищених мереж. Для досягнення безпеки, протокол SSH забезпечує не лише шифрування за допомогою захищеного каналу, але також надійну автентифікацію за паролем і автентифікацію з відкритим ключем.

Стандартне вікно підключення до одноплатного комп’ютера Raspberry Pi з можливістю налаштування безпечного з’єднання (рис. 2.8).

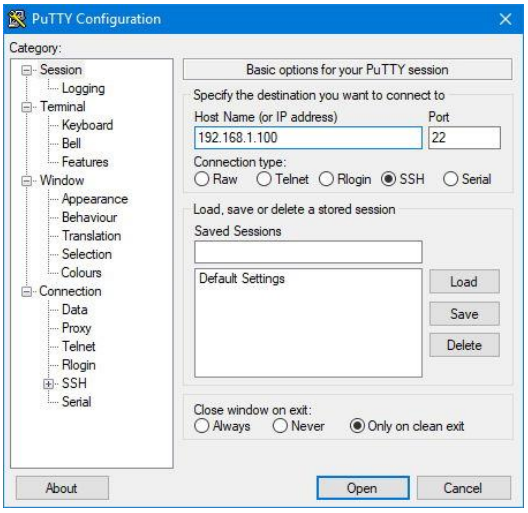


Рисунок 2.8 – Налаштування з’єднання між комп’ютерами за допомогою PuTTY

Також ще одна програма потрібна для відображення графічного інтерфейсу Raspberry Pi (Bullseye). Для цього використовується VNC Connect. VNC означає Virtual Network Computing. [18] Кросплатформна система спільного доступу до екрана, створена для віддаленого керування іншим комп’ютером. Це означає, що екран, клавіатура та миша комп’ютера можуть використовуватися віддаленим користувачем із додаткового пристрою на відстані, наче вони сидять прямо перед ним. Данна програма дуже допомагає у розробці програмного забезпечення, коли немає можливості підключити плату до додаткового монітору. Підключення також відбувається через SSH протокол (рис 2.9).

Рисунок 2.9 – Віддалений інтерфейс

2.4 Мови програмування

Для даного завдання також велику роль грає правильно обрана мова програмування. Вона дозволить нам писати ефективний програмний код під відповідний одноплатний пристрій, який буде працювати швидко і бажано без помилок. Написаний код, або скрипт, сам по собі буде основним чином використовується для автоматизації, збору інформації з датчиків, інтерпретації, обробки, та виводу готових даних та відповідної інформації для користувача.

При розробці розглядаються різні варіанти мов програмування, які мають відповідний інструмент (у вигляді бібліотек, або фреймворків) для більш швидкого і зручного написання коду.

Розглянемо мови програмування які за думкою більшості є найефективнішими у розробці програмної частина, під процесор arm архітектури і самої плати Raspberry Pi.

Почнемо з мови програмування Rust, який на сьогоднішній день є однією з мов що швидко розвивається і не поступається своєю швидкість і збереженню пам’яті, таким мовам як C++, Java, Python (рис. 2.10).

Рисунок 2.10 – Логотип мови Rust

Rust за своєю сутністю є статично типізованою мовою програмування, Яка в першу чергу створена для безпеки і продуктивності, акцентуючи свою увагу на безпечному паралелізму і керуванням пам’яті. Якщо подивитися, то синтаксис подібний до C++.

Ця мова програмування вирішує проблеми, яка існує і сьогодні у розробників C/C++: помилки при виділенні та очищені пам’яті і паралельне програмування. Безпека пам’яті — це сильна сторона Rust, у якій указники пам’яті, що використовуються, завжди вказують на вже виділену пам’ять, тобто яка буде правильного типу/розміру.

Безпека пам’яті — це проблема правильності — програма, яка не є безпечною для пам’яті, може вийти з ладу або створити недетермінований вихід залежно від помилки.

Rust – це технологія яку все частіше використовують у ентерпрайс виробництві, і переписують відповідні ділянки коду на Rust. Також ця мова програмування дозволяє вам писати логіку для низького рівня і контролювати її. Робота з пам’яттю дуже зручна, так як дані можна зберігати у стеку (stack) (виділяється статична пам’ять) чи у купі (heap) (виділяється динамічна пам’яті).

Важливою ідіомою є RAII (Resource Acquisition Is Initialization), яку використовують частіше у мові C++, але також підходить до мови Rust: кожного разу, якщо у написаній програмі, об’єкт виходить за межі області

видимості і не використовується - викликається його деструктор і звільняються його власні ресурси. Це набагато легше, тому не приходиться робити все власноруч, і також прибирається проблема витоку ресурсів.

Наступною мовою програмування, буде розглянута мова Python. Це незамінимий інструмент з великою кількість створених бібліотек, для більш зручної роботи, починаючи з аналізу даних (data analytics) та закінчуючи штучним інтелектом (AI) (рис. 2.11).

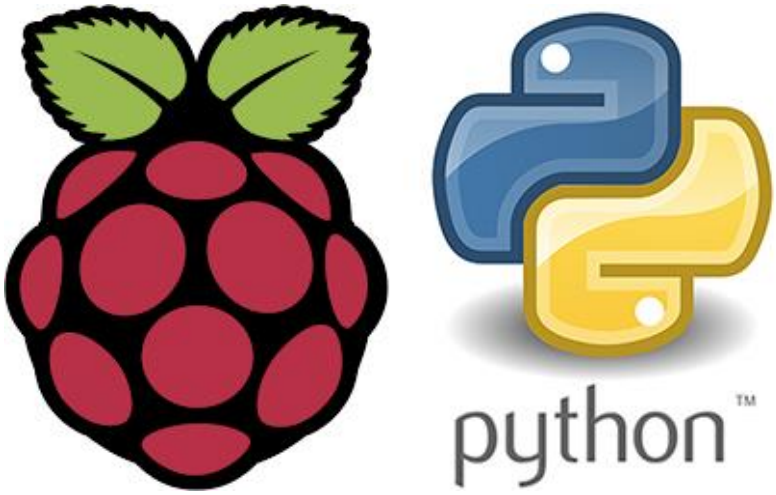


Рисунок 2.11 – Логотип мови Python

Саме ця мова програмування відповідно підходить для написання програмного забезпечення для розпізнавання машин та різноманітних обьектів на дорозі, при підтримки відповідної бібліотеки для комп’ютерного зору OpenCV. Це незамінний інструмент для використання класифікаторів, а також, за бажанням створення особистого класифікатора, та тренування у розпізнаванні об’єктів [7] (рис. 2.12).

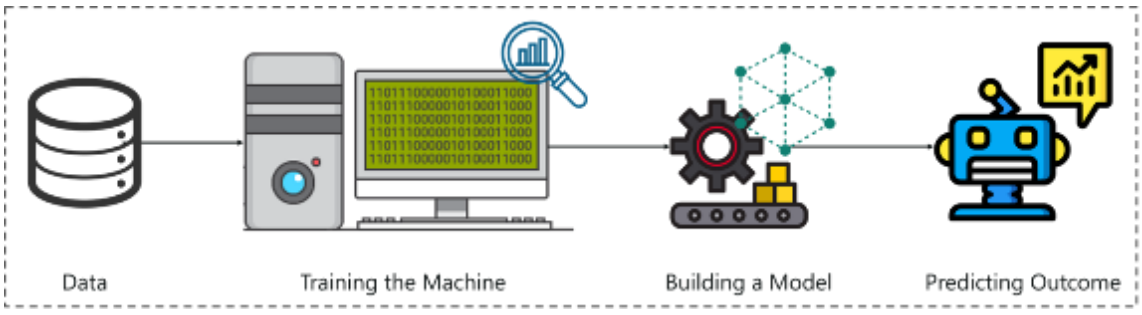


Рисунок 2.12 – Тренування відповідної моделі

Основними перевагами мови Python є:

– Менше коду: впровадження систем доповненої реальності включає безліч алгоритмів. Для попередньо визначених пакетів не потрібно писати алгоритми з нуля. Також Python пропонує методологію «перевірка під час написання коду», яка у свою чергу зменшує навантаження на тестування коду.

– Попередньо зібрані бібліотеки: у Python є сотні готових бібліотек для реалізації різноманітних алгоритмів для написання програми саме під потреби користувача і використовуючи інструменти машинного та глибокого навчання. Тому при розробці алгоритму зі системою доповненої реальності та впровадження величезних наборів даних для тренування свого комп’ютерного зору, лише потрібно встановити та завантажити необхідні пакети у свій проект за допомого команди pip install. Приклади готових бібліотек включають NumPy, OpenCV, Keras, Tensorflow, Pytorch тощо.

– Незалежність від платформи: Python працює за допомогою інтерпретатора, отже може працювати кросплатформенно, включаючи такі системи як Windows, MacOS, Linux, Unix тощо. Переносячи код з однієї платформи на іншу, найкраще всього використовувати такі пакети, як PyInstaller, які вирішують основні проблеми залежностей.

Також можна відзначити таку мову програмування як Go. Мова Go також вміє працювати бібліотекою для комп’ютерного зору, такою як OpenCV (рис. 2.13).

Рисунок 2.13 – Мова програмування Go

Go — статично типізована мова програмування, яка на виході створює двійкові файли які були скомпільовані машиною. Архітектура містить

вбудовані масиви різної довжини, а також карти ключ-значення. Також присутня підтримка стандартної бібліотеки.

Go дозволяє розробнику проводити операції з об'єктами, забезпечувати статичне/суворе введення з паралельним доступом.

Також слід зазначити, що використання OpenCV в Go буде називатися GoCV. GoCV надає доступ до бібліотеки комп'ютерного зору OpenCV 4. Пакет GoCV підтримує останні релізи Go та OpenCV v4.7.0 для ОС Linux, macOS та Windows. Основний задум цього, ці створення сумісності з останніми розробками в екосистемі OpenCV для мови Go. GoCV підтримує CUDA за допомогою графічних процесорів NVIDIA.

2.5 Математичні методи застосовані в використанні класифікаторів OpenCV

Об'єкти прямокутника можна обчислити дуже швидко, використовуючи проміжне представлення зображення, яке називаємо інтегралом. [14] Інтегральне зображення в місці x, y містить суму пікселів вище та ліворуч від x, y, включно

$$ii(x, y) = \sum_{x' \le x, y' \le y} i(x', y')$$

Де $ii(x, y)$ це інтегральне зображення і $i(x', y')$ це оригінальне зображення. Далі представлено функції навчання класифікатора. Таким чином, слабкий класифікатор $h_j(x)$ складається з ознаки f_j, порогового значення θ_j і парності p_j, що вказує напрямок знаку нерівності.

$$h_j(x) = \begin{cases} 1 \text{ якщо } p_j f_j(x) < p_j \theta_j \\ 0 \text{ у іншому випадку} \end{cases}$$

Наведені приклади зображень $x_1 y_1$, ..., $x_n y_n$ де $y_i = 0, 1$ для негативних і позитивних прикладів відповідно.

Ініціалізуємо вагові коефіцієнти $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ для $y_i = 0, 1$ відповідно, де m і l — кількість негативних і позитивних результатів відповідно.

Для $t = 1$, ..., T:

1. Нормалізуйте ваги $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$*, щоб* w_t *був розподілом ймовірностей.*

2. Вибрати найкращий слабкий класифікатор щодо похибки ваги $\sum_i w_t \mid h_j(x_i) - y_i$.

3. Обираємо класифікатор h_j з мінімізованою похибкою.

4. Оновлюємо ваги $w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$ де $e_i = 0$ якщо приклад x_i класифікувався коректно і $e_i = 1$ навпаки, і $\beta_t = \frac{e_t}{1-e_t}$.

Нарешті отримуємо сильний класифікатор:

$$h(x) = \begin{cases} 1 \sum_{t=1}^{T} a_t h_t(x) \geq \frac{1}{2}\sum_{t=1}^{T} a_t \\ 0 \text{ у всіх інших випадках} \end{cases}$$

де $a_t = \log\frac{1}{\beta_t}$.

Висновки до розділу 2

У другому розділі основі проведеного аналізу обрано програмну частину для одноплатного комп'ютера Raspberry Pi 4 для створення алгоритму обробки поточного зображення, для систем automotive за допомогою фреймворку GStreamer та бібліотеки OpenCV.

Порівняно мови програмування та обрано відповідну для написання алгоритму обробки зображення та виділення об'єктів.

Показані та обґрунтовані, характеристик і сумісного ПО, Переваги Raspberry Pi серед інших одноплатних комп'ютерів.

Для написання алгоритму обрана мова програмування Python для написання швидкого алгоритму для розпізнавання об'єктів. Корисність використання готових бібліотек від розробників.

Програмне забезпечення для проектування програми на основі computer vision:

– Microsoft Visual Studio Code – середовище розробки;
– GStreamer – фреймворк для відео потоку;
– OpenCV tools – інструменти для створення класиіфкаторів;

– Python – мова програмування для написання алгоритму оброки зображень.

Описані основні моменти використання програмного забезпечення, таке як Putty, для підключення до одноплатного комп’ютера Raspberry Pi, віддаленим способом. А також продемонстрований розрахунок каскадних класифікаторів.

3 АПАРАТНО-ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ COMPUTER VISION НА БАЗІ RASPBERRY PI

3.1 GStreamer framework

GStreamer — це мультимедійна платформа з відкритим вихідним кодом, яка в основному використовується для розробки медіа-програм (потокове передавання, відтворення медіа, нелінійне редагування тощо) (рис. 3.1).

Рисунок 3.1 – GStreamer framework

Даний фрейморк буде використаний у нашій програмі як пайплайн (pipeline), який передає зображення с потокового відео на обробку, для знаходження машин, перешкод та знаків.

GStreamer — це мультимедійний фреймворк, який здатний робити саму різну обробку медіа даних, наприклад аудіо, відео, запис і потокова відео трансляція тощо. Завдяки цьому, даний фреймворк забезпечує модульність. Наприклад користувач зможе зробити окремий, під свої потреби конвеєри та інтегрувати плагіни у процесі розробки додатку[10].

Так як ці дві бібліотеки OpenCV і GStreamer можуть використовуватися окремо один від одного, їх поєднання дає більшу варіативність та гнучкість для обробки мультимедійного потоку та усіх поступаючих даних.

GStreamer дозволяє нам маніпулювати, надсилати та отримувати дані на нашій машині за різними протоколами [12] (рис. 3.2).

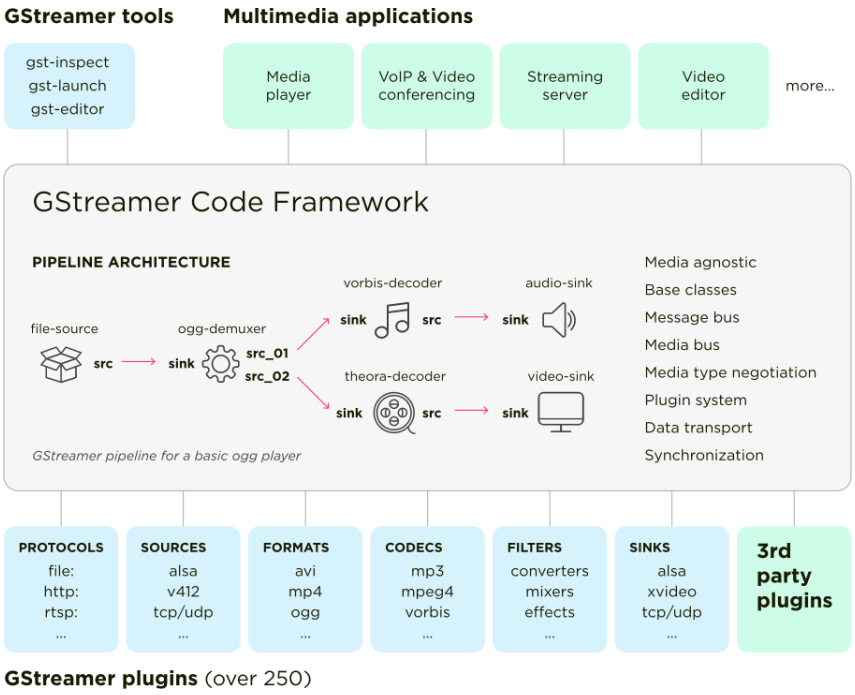


Рисунок 3.2 – Проходження вихідного файлу через GStreamer

Проектування OpenCV за допомогою GStreamer у середовищі Linux може бути простим у порівнянні з Windows через наявність більшої онлайн-підтримки та процедури встановлення.

Наприклад, створюється генерація машинних даних RTP, та зчитування їх. Також можна створити сервер RTSP, для якого буде використовуватися певне джерело поточних даних, наприклад RTP або файл формату (WAV/MP3), і буде дозволено клієнтам RTSP протоколу отримувати доступ до цього потоку [26].

RTSP це протокол передачі потокових даних (у нашому випадку відео дані), який дає змогу працювати з мережевими джерелами, такими як камери у машині, безпеки чи домашні ip-камери, медіа-сервери тощо.

GStreamer дозволяє реалізувати ці функції просто додавши однорядкові команди для виконання у терміналі. Основна частина GStreamer полягає в тому, що ми будуємо конвеєри (pipelines). Конвеєр описує потік операцій, який починається від джерела, проходить через вбудовані інструменти програми, які будуть вказані у терміналі, на всьому шляху до фінального результату на виході.

Також одним з самих головних аспектів цього фреймворку є кодування та декодування різних форматів файлів. В ньому є такі основні функції, як підтримка декодування WebM Alpha, підтримка підкадрів відеодекодера, багатопотокове перетворення відео та мікшування, підтримка MPEG-2 і VP9 Linux без збереження стану, а також підтримка інтелектуального кодування (пропуску) для VP8, VP9 і H.265.

Приклад перекодування файлу:

Ми хочемо взяти файл із записом подкасту у форматі wav, і треба перетворити його на mp3.

Ми можемо визначити файл як джерело -> проаналізувати з wav-файлу необроблений аудіо -> закодувати його за допомогою mp3 -> зберегти це необроблене аудіо до місця призначення, яке є новим файлом mp3.

Теж саме можна зробити, але для поточного відео, без збереження файлу, а обробці у реальному часі.

3.1.1 Способи передачі потоку GStreamer

Фреймворк GStreamer має дуже велику кількість протоколів для передачі потокового зображення. Кожен з них корисний при відповідній задачі, але зараз буде розглянуто самі популярні, які використовуються, та будуть використовуватися ще у недалекому майбутньомус[11] (рис. 3.3).

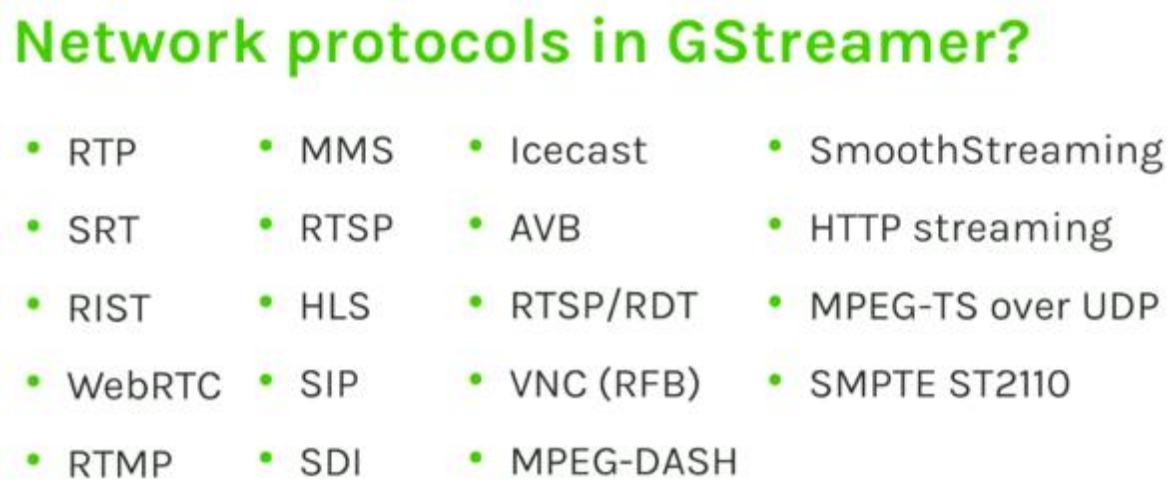


Рисунок 3.3 – Мережеві протоколи GStreamer

Головна їх відмінність у низької затримці і високій надійності. Тобто, під надійністю мається на увазі зниження буфеоізації, чим більше буде заповнений буфер обміну, тим вище вірогідність переривання відео потоку, або аудіо потоку. Якщо буфер буде становити приблизно одну хвилину, то і відео буде не працювати 1 хвилину.

Зараз буде розглянуто основні протоколи які можна використовувати і їх основні характеристики і відмінності:

– HTTP streaming (HLS)
– RTP/RTSP
– SDI
– VNC (RFB)

HTTP Live Streaming (HLS) — це широко використовуваний протокол потокового передавання, який може передавати відео/аудіо файли малими фрагментами через HTTP протокол, тому їх можна буде завжди переглядати. Але системні брандмауери легко блокують UDP потік і боряться з обмеженою перевіркою помилок при надсиланні.

HLS у своєму сегменті, був розроблений для нативної роботи через HTTP і подолав ці проблеми, отже сам по собі, більш простий. Оскільки відповідні браузери і пристрої вже мали роботу з HTTP, для HLS не потрібно було ні виділеного сервера, ні власної служби для доставки вмісту (рис. 3.4).

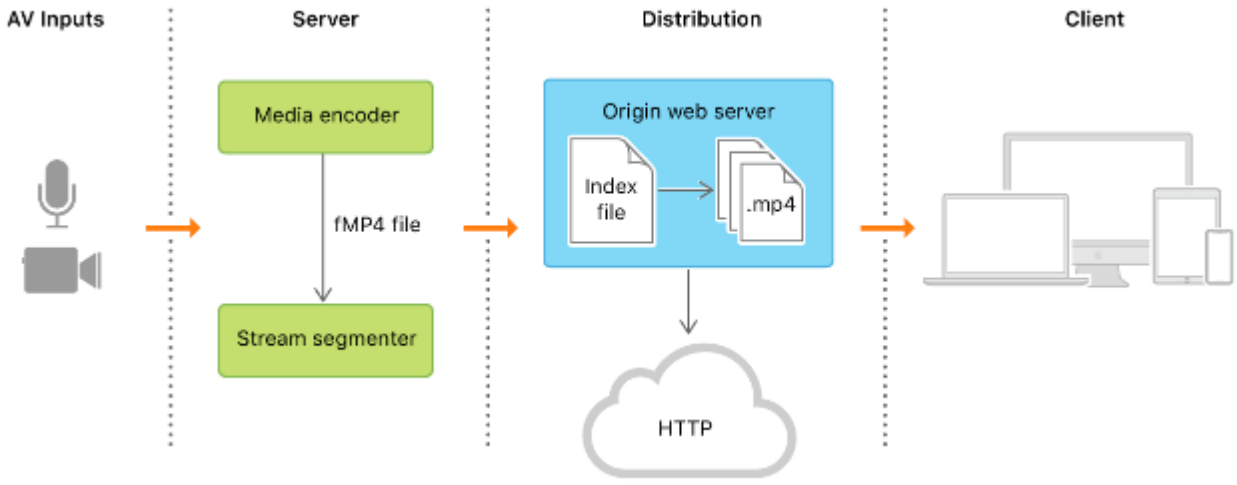


Рисунок 3.4 – Архітектура HLS

Основна і корисна з функції цього протоколу в тому, що HLS кодує відеофайли в декількох бітрейтах. Отже для перегляду відеоматеріалу можна обрати його якість, і в будь-який час доступні кілька версій відеофрагментів різної якості.

Файл зі списку містить назву кожного фрагмента, а також його різну якість для перемикання. Не важливо, чи у клієнта 2G, Wi-Fi або Ethernet, він завжди зможе вибрати якість яка йому підходить.

RTSP (Real Time Streaming Protocol) – протокол прикладного рівня, розроблений для систем телекомунікації, основною задачею якого є керування доставкою відео та аудіо даних. RTSP - сигнальний протокол, він керує сесією передачі даних напряму з медіа сервером (рис. 3.5).

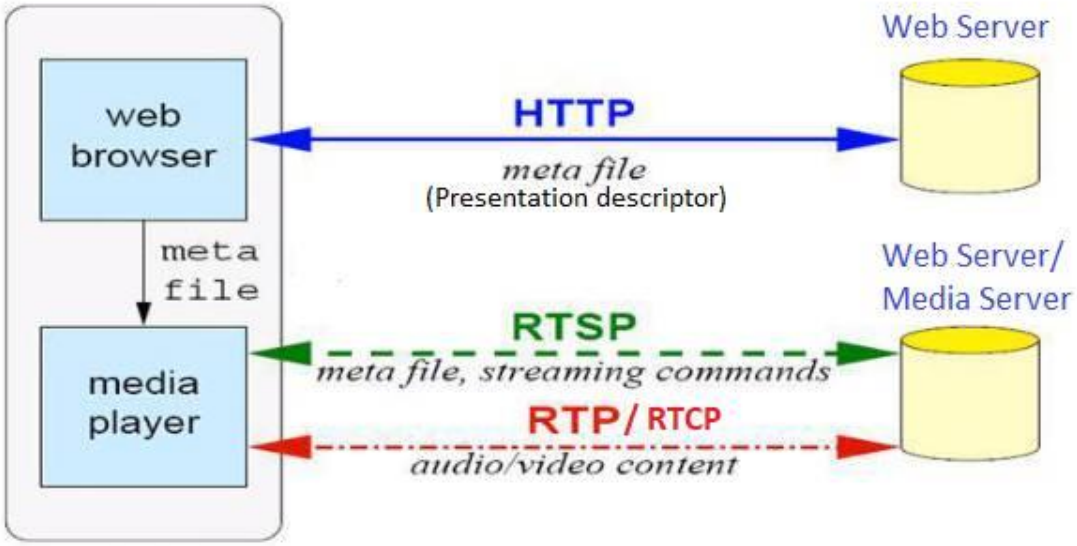


Рисунок 3.5 – Архітектура RTSP

Закладені у RTSP принципи зміг перейти у сучасний стандарт WebRTC протоколу.

На транспортному рівні передачі самого потоку медіа як реального часу використовується протокол RTP (Real-Time Protocol). Роль RTSP порівняно як дистанційне керування сервером потокового медіа. IP-камери можуть використовувати як TCP, так і UDP для передачі потокового відео або аудіо файлу. Однак слід зауважити, що для цього завдання в UDP немає

практичного сенсу. Нікому не хочеться втрачати інформацію при передачі потоку.

SDI протокол використовується локальною мережею. SDI — це стандарт передачі цифрового відео, який зазвичай використовується в телерадіомовленні та інших професійних програмах. Для його роботи обирається пару коаксіальних кабелів, які передають один цифровий сигнал, використовуючи оптичне волокно, швидкість якого може бути від 270 Мбіт/с до 12 Гбіт/с.

SDI — це інтерфейс «точка-точка», що означає, що кожен кабель передає сигнал від одного пристрою до іншого.

Деякі з особливостей SDI включають:

– Відео без стиснення: SDI надсилає відео потік на пряму, без використання стиснення, що і видає картинку високої чіткості.

– Корекція помилок: SDI включає вбудовану корекцію помилок, це дає максимально плавну та надійну передачу відео сигналу.

– Низька затримка: SDI має низьку затримку, часом надсилання сигналу та часом його отримання дає дуже невелика затримка. Це важливо для додатків з прямим потоком, наприклад для прямих трансляцій.

– Передача на великі відстані: SDI може передавати відеосигнали на великі відстані без погіршення якості сигналу.

VNC (віртуальні мережеві обчислення) — це система віддаленого спільного використання робочого стола, яку можна використовувати для керування одним комп’ютером з іншого місця. Інструменти віддаленого робочого столу на основі VNC є легкими та універсальними, доступних на сьогодні, і часто також безкоштовні, які займають вищі місця у безкоштовному ПО для роботи з віддаленим робочим столом (рис. 3.6).

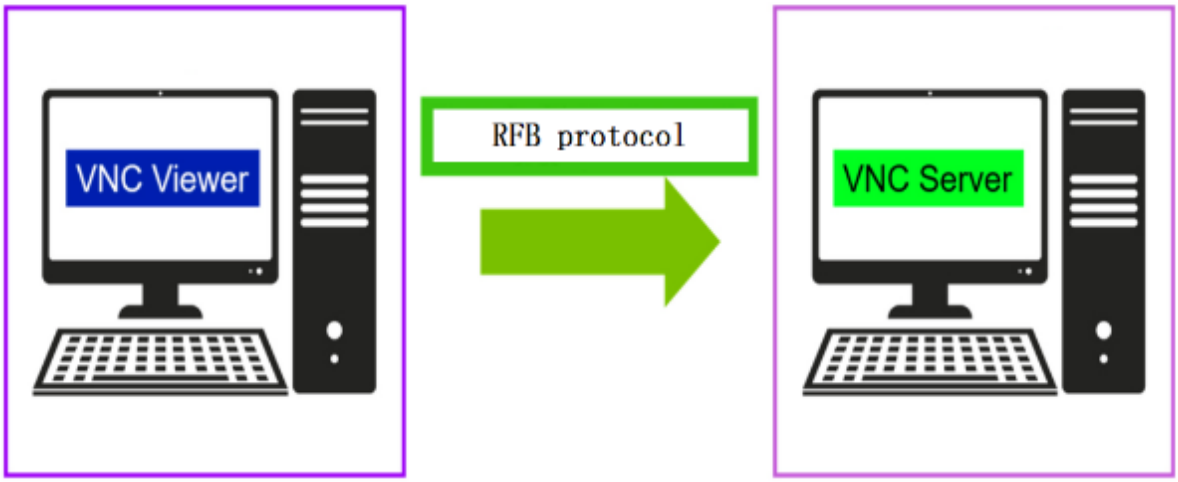


Рисунок 3.6 – Віддалене підключення до через VNC

VNC використовує надійний, але простий протокол під назвою RFB - віддалений кадровий буфер (remote framebuffer). Він транспортує зображення та периферійні вхідні дані у обидві сторони між сервером (з комп’ютером до якого проведено підключення) і клієнтом (комп’ютером, з якого здійснюється підключення). VNC можна запускати на не потужному обладнанні оскільки він не використовує багато ресурсів (процесор і пам’ять).

Протоколу RFB не потрібно багато знати про операційну систему, на якій він працює. Основні інструменти які йому потрібні, це надсилати дані миші та клавіатури від клієнта до сервера, а також дані зображення від сервера до клієнта. Також даний протокол використовується в даній роботі для підключення до віддаленого комп’ютера Raspberry Pi.

3.1.2 GStreamer Pipeline

Елемент — найважливіший клас об’єктів у GStreamer. В основі створюється ланцюжок з обраних елементів, які пов’язані один з іншим (обробка, декодування тощо), після чого вхідні дані проходять через цей створений ланцюжок з елементів. Елемент має одну з конкретних функцію, наприклад може бути декодування цих файлів, або виведення цих даних на звукову карту.

Після об’єднання обраних елементів у ланцюжок, буде створений конвеєр (pipeline), який буде виконувати конкретне завдання, відтворювати потокове віщання або відповідна обробка медіа файлів. GStreamer має велику колекцією елементів у стандартному наборі при встановленні, що робить розробку різноманітних медіа-додатків гнучкою та під конкретні завдання. При бажанні, можна написати нові елементи. [25] Про ці функції, детальніше можна прочитати в Посібнику для автора плагінів GStreamer (рис. 3.7).

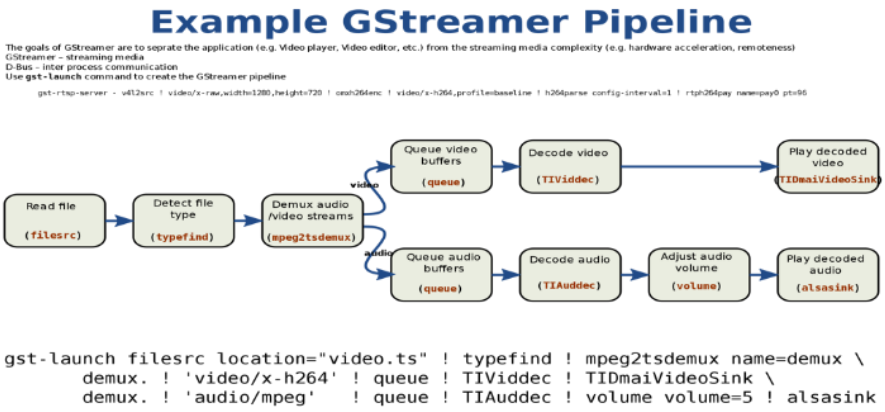


Рисунок 3.7 – GStreamer pipeline

Bin (контейнер) — потрібен для набору елементів. Контейнери самі по собі є підкласами елементів, де можна керувати контейнером так, ніби він був елементом, тим самим обходячи великої складності для програми яка розроблюється.

Pipeline - це контейнер верхнього рівня. Pipeline надає шину для програми та керує синхронізацією для своїх підлеглих елементів. [13] Коли ви встановлюєте його в стан PAUSE або PLAYING, почнеться потік встановлених даних і відбувається обробка цих даних. Програма буде продовжувати працювати і самі конвеєри будуть працювати в окремому потоці, доки вони не будуть зупинені, або не буде досягнуто кінця потоку даних.

3.2 OpenCV бібліотека

OpenCV — це бібліотека комп’ютерного зору з відкритим вихідним кодом, призначена в основному для систем реального часу. Бібліотека написана на C++ і пропонує оптимізований код для обробки даних і візуалізації на різних ОС, Windows, Linux, FreeBSD, macOS тощо. Завдяки правильно написаній бібліотеці розробників і дослідників, ця бібліотека пропонує великий набір інструментів для редактору коду, для різних етапів обробки зображень і поточного відео; починаючи від зйомки зображення/відео, калібрування, попередньої обробки, виділення ознак і закінчуючи класифікацією, або написанням свого класифікатора. Написаний на C++, OpenCV має адаптацію а інші мови програмування, таких як Python, Java, Go (рис. 3.8).

Рисунок 3.8 – Бібліотека OpenCV

Під час розробки бібліотеки OpenCV основна увага приділялася програмам реального часу для підвищення ефективності обчислень. Усі речі написані на оптимізованому C/C++, щоб скористатися перевагами багатоядерної обробки. Саме те що потрібно для опрацювання на одноплатних комп’ютерах, типу Raspberry Pi [6].

Функціональність OpenCV:

– Введення/виведення зображення/відео, обробка, відображення (ядро, imgproc, highgui)

– Виявлення об’єктів/функцій (objdetect, features2d, nonfree)

– Монокулярний або стереокомп’ютерний зір на основі геометрії (calib3d, зшивання, videostab)

– Комп'ютерна фотографія (фото, відео, суперзйомка)

– Машинне навчання та кластеризація (ml, flann)

– Прискорення CUDA (gpu)

Комп’ютер зчитує будь-яке зображення як діапазон значень від 0 до 255. Для будь-якого кольорового зображення є 3 основні канали – червоний, зелений і синій (рис. 3.9).

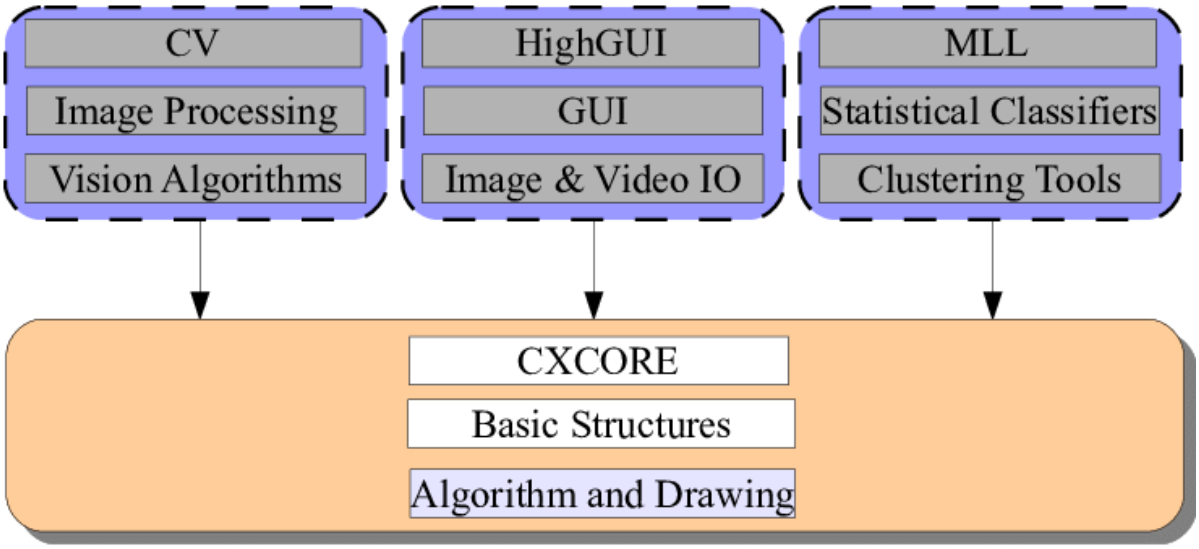


Рисунок 3.9 – Архітектура OpenCV

Ось основні бібліотечні модулі за допомогою який і обробляються дані OpenCV:

– Core Functionality. Основні функції бібліотеки OpenCV працює з основними структурами даних, таких як Scalar, Point, Range тощо. Також Mat використовується для зберігання зображення у багатовимірний масив.

– Image Processing. Даний модуль використовується для обробки зображень, гістограми, геометричні перетворення зображень, фільтрація зображень, перетворення колірного простору тощо.

– Video. Цей модуль відповідає за аналіз потоку даних, оцінка руху, та відстеження об’єктів, віднімання фону.

– Video I/O. Цей модуль відповідає за відео кодеки та захоплення самого відеопотоку.

– Calib3d. Даний модуль містить алгоритми геометрії кількох ракурсів, оцінку пози об’єкта, стереовідповідність та елементи 3D-реконструкції.

– Objdetect. Цей модуль відповідає за виявлення об’єктів попередньо визначених класів, таких як машини, знаки, світлофори, люди, обличчя тощо.

3.2.1 Haar Cascade класифікатор

Для розпізнавання об’єктів повинен бути відповідний файл (класифікатор), модель якого буде навчена відрізняти задані об’єкти, котрі можуть бути розташовані на вихідному відео, або фото, чи рисунку.

Каскадний класифікатор Хаара — це найбільш популярний та ефективний підхід до виявлення об’єктів у кадрі, запропонований Полом Віолою та Майклом Джонсом у статті 2001 року «Швидке виявлення об’єктів за допомогою розширеного каскаду простих функцій».[5]

Що потрібно для створення класифікатора:

– Фотографії об’єкта в реальному середовищі. Чим вибірка більше буде схожа на те середовище де йде розпізнавання, тим краще будуть результати. Головне не брати професій фото зі студії, результат буде гірше, якщо, наприклад, розпізнавання буде на вулиці.

– Обрання негативних фотографій, на яких немає об'єкта розпізнавання. Бажано щоби фото, негативного характеру, були зроблені у тому ж середовищі.

– Бібліотека OpenCV або сторонній gui інтерфейс для створення класифікатора. Якщо робите проект з нуля, краще використовувати OpenCV для повного розуміння як це працює.

Модель розпізнавання об’єкті можна робити як через бібліотеку OpenCV (файли opencv_createsamples.exe використовується для позитивних прикладів і opencv_traincascade.exe для формування самого каскаду). Для коректного розпізнавання знадобиться близько 1000 позитивних і негативних фотографій (рис. 3.10).

Рисунок 3.10 – Приклад вибірки фотографі для розпізнавання об’єктів

Основні етапи роботи алгоритму:

– Розрахунок властивостей Хаара (Haar Features)
– Створення цілісних образів (Integral Images)
– Використання Adaboost
– Реалізація каскадних класифікаторів (Cascading Classifiers)

Функція Хаара — обчислення у певних місцях на фото з об’єктом, з використанням суміжних прямокутних областей для виконання. Основна його задача це підсумовування інтенсивностей пікселів у кожній з виділених області. Після чого йде обчислення різниці між сумами (рис. 3.11).

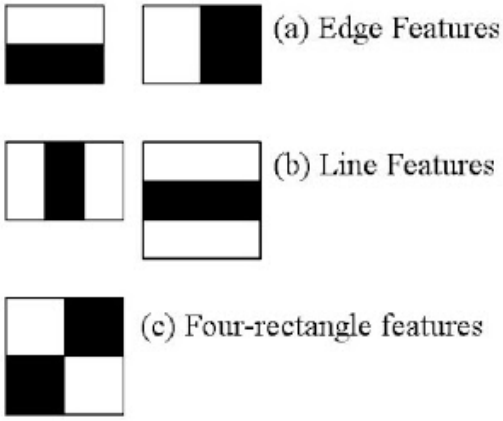


Рисунок 3.11 – Функції Haar

Інтегральні зображення дуже прискорюють обчислення характеристик Хаара. Створюються підпрямокутники та посилання на масив для кожного з цих підпрямокутників, щоб не обчислювати кожен піксель окремо. Дані масиви далі використовуються для обчислення характеристик Хаара (рис. 3.12). На цьому етапі важливі лише характеристики об’єкта, тому майже усі функції Хаара будуть не релевантні.

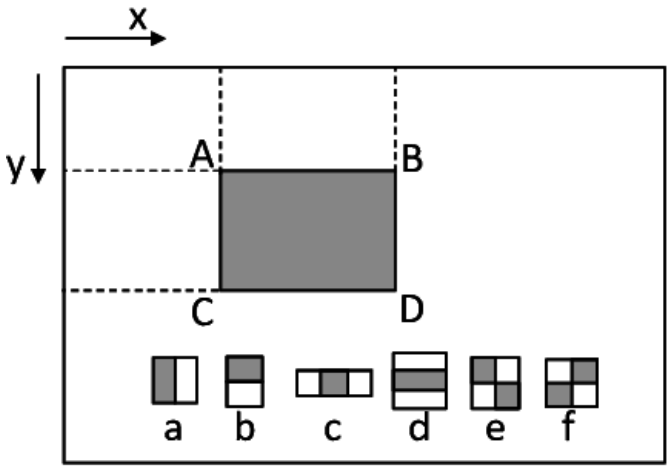


Рисунок 3.12 – Інтегральні зображення

За допомогою Adaboost вибираються найкращі функції і навчає класифікаторів їх використовувати. Adaboost використовує виділення «слабких класифікаторів (weak classifier)», щоб створити «сильний класифікатор (strong classifier)», після чого алгоритм використовує їх для виявлення об’єктів.

Слабко навчені створюються шляхом переміщення вікна над вхідним зображенням і обчислення характеристик Хаара для кожного підрозділу зображення. Після чого різниця порівнюється з вивченим порогом, який сортує відсутність відповідного необ’єкту і об’єкти. Для формування сильного класифікатора необхідна велика кількість ознак Хаара для білш високої точності (рис. 3.13).

Після чого йде об’єднання дуже великої кількості weak learners у strong learners за допомогою каскадного класифікатора [17].

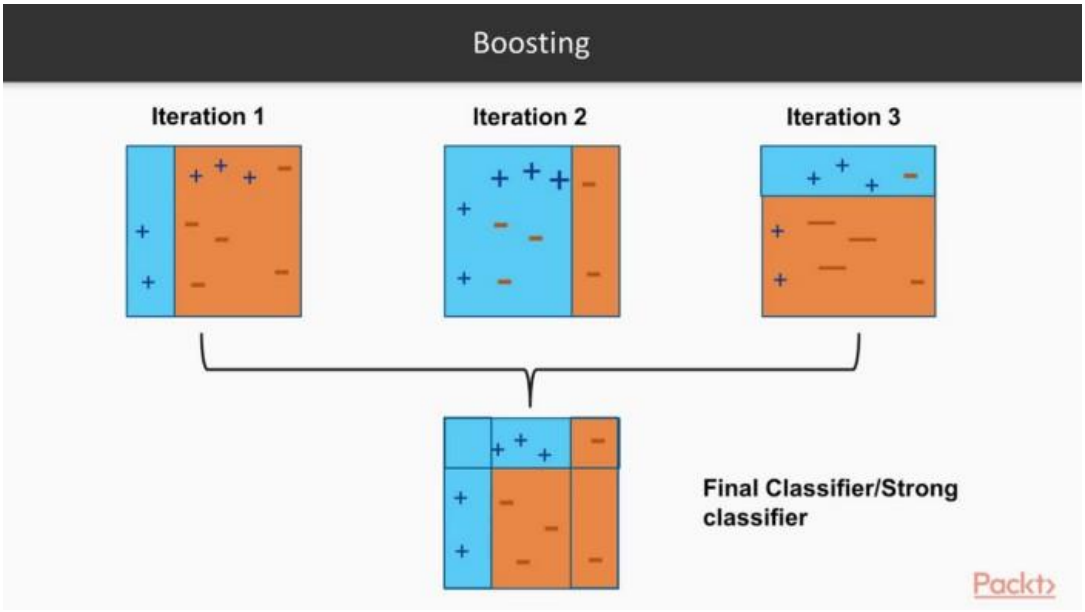


Рисунок 3.13 – Adaboost алгоритм

Каскадний класифікатор складається з серії етапів. Кожен з етапів це набором слабких учнів. Слабкі учні навчаються за допомогою стимулювання, що дозволяє отримати високоточний класифікатор із середнього прогнозу всіх слабких учнів (рис. 3.14).

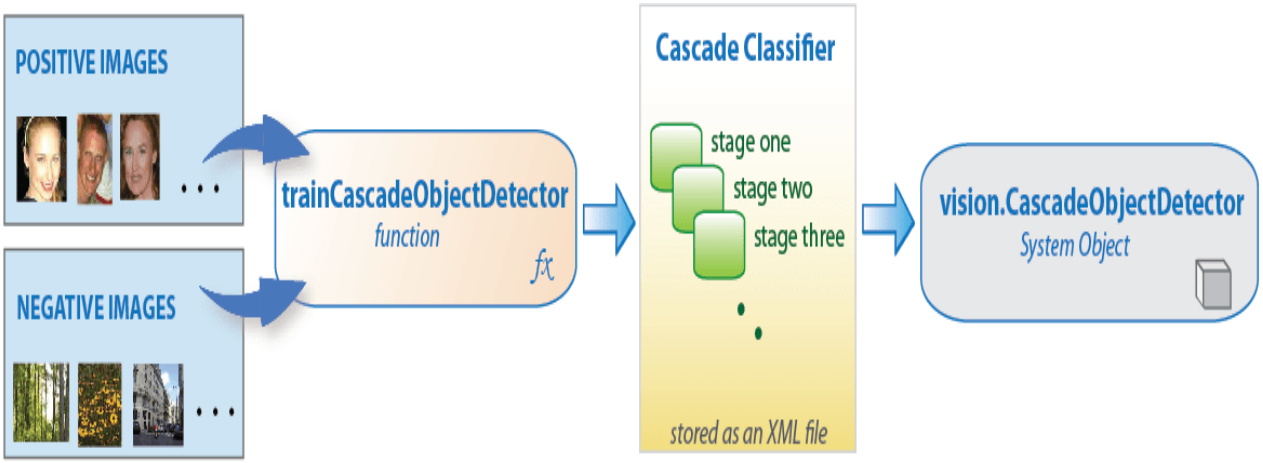


Рисунок 3.14 – Створення каскадного класифікатора

На основі цього передбачення класифікатор вирішує або вказати, що об’єкт знайдено (позитивний), або перейти до наступної області (негативний).

На виході отримується .xml файл, який у подальшому можна використовувати у алгоритмі для роботи з цим класифікатором (car_detect = cv2.CascadeClassifier("TruckClassifier.xml")).

Висновки до розділу 3

У третьому розділі обрано апаратно програмне забезпечення для даного проекту, використовуючи фреймворк GStreamer та бібліотеки комп’ютерного зору OpenCV.

Обраний відповідний пайплайн, за допомогою фреймворку GStreamer та розглянуті протоколи передачі зображення з обробкою до кінцевого користувача.

Були висвітлені наступні пункти розробки:

- Плюси у використанні з стрімінговим фреймворком;
- Працездатність pipeline (магістралі);
- Робота класифікатора та його створення;
- Використання computer vision з бібліотекою OpenCV;
- Особливість протоколів передачі даних.

Використовуючи ідеально сумісну бібліотеку з готовим функціоналом для створення, та використання класифікаторів.

Описані основні пункти по проектуванню програмного забезпечення для поставленої задачі, та відповідні ризики, які можуть бути у процесі розробки.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА ТЕСТУВАННЯ СИСТЕМИ COMPUTER VISION

Беручи до уваги інформацію яка вищевикладена, можна зробити опис основних вимог до апаратно-програмного комплексу на Raspberry Pi:

Hardware

– Одноплатний комп’ютер Raspberry Pi 4
– Raspberry Pi camera module v2
– USB-Type C power supply
– Кабель HDMI
– Монітор для виводу картинки
– Корпус розроблений за допомогою 3d-принтера (за бажанням)

Software

– Програмне середовище для розробки алгоритму та написання каскадного класифікатора, і завантаження на плату– Microsoft Visual Studio Code
– Фреймворк для обробки меді-потоку та створення пайплайнів - Gstreamer
– Бібліотека для розпізнавання об’єктів та створення класифікаторів - OpenCV
– SSH термінал Putty для віддаленого підключення
– Fritzing - середовище для візуалізації з’єднання схеми
– Cascade training GUI

Після цього можна підключати одноплатний комп’ютер, робити налаштування, та тестувати програмне забезпечення.

4.1 Підключення до одноплатного комп’ютера та завантаження програмного коду

На сам перед, для роботи з модулем треба його підключити то комп’ютера на якому робите, а також встановити відповідну для цієї плати

операційну систему, з графічним інтерфейсом, для виводу зображення, здобреного на камеру.

Для початку під’єднується SD картка до ПК де і буде встановлюватися ОС для Raspberry Pi. Після цього було завантажено “Raspberry Pi Imager” для запису ОС під плату (рис. 4.1).

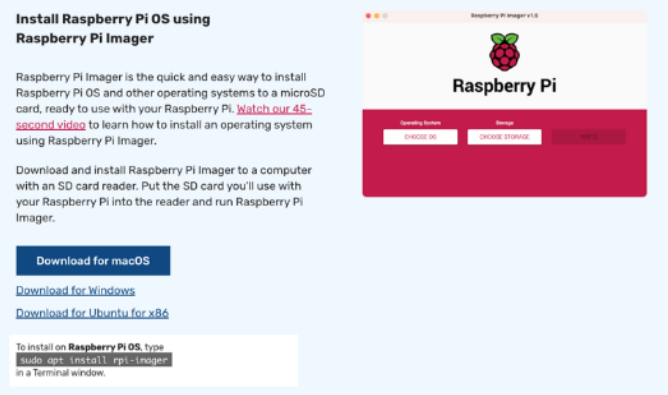
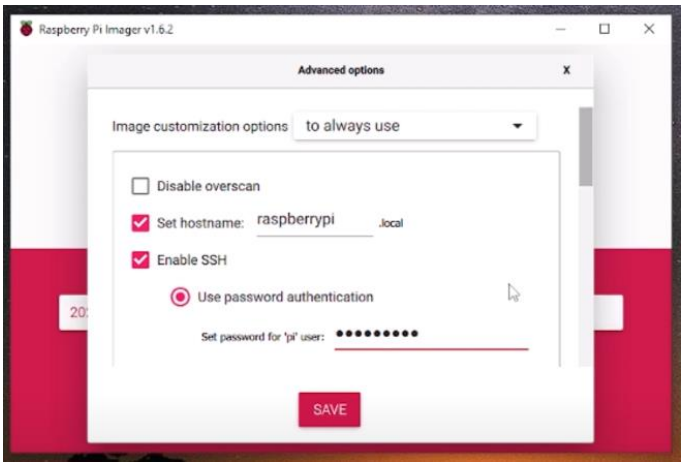


Рисунок 4.1 – Встановлення Raspberry Pi Imager

Після встановлення було обрану однин з трьох образів (Lite, with desktop, with desktop and recommended software), with desktop. Далі одним з найголовніших пунктів, це налаштування образу, де вказується ім’я користувача (hostname), встановлюється дозвіл на SSH з’єднання, та встановлюється пароль. Також для віддаленого підключення встановлюється конфігурація для Wi-Fi мережі з SSID та паролем. Також після цього можна налаштувати локальні данні (часовий пояс, мову клавіатури тощо.) (рис. 4.2, 4.3).



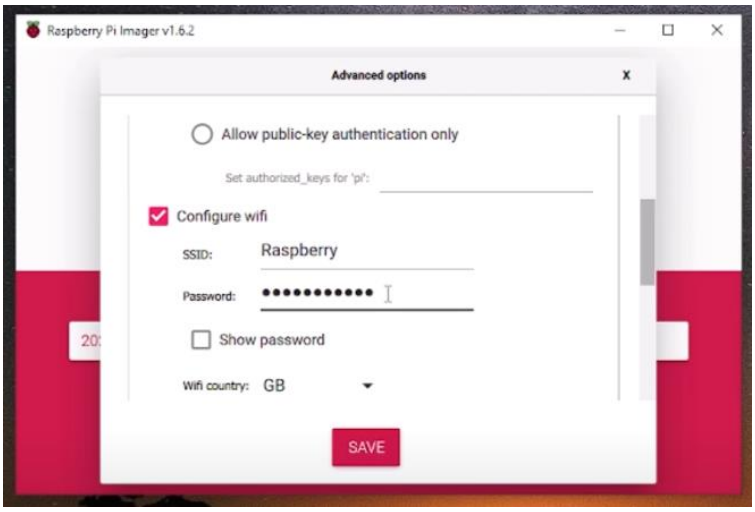


Рисунок 4.2, 4.3 – Конфігурація ОС для підключення

Після усіх налаштувань тиснеться кнопка «Write» і відповідний образ записується на SD-карту. Після запису, карта витягується і вставляється в Raspberry Pi.

4.1.1 SSH підключення через Putty та VNC

Далі було зроблено підключення до плати, та перевірка працездатності, за допомогою програми Putty буде зроблено ssh з’єднання з платою. При бажанні, також можна використовувати Ethernet кабель, але вже треба буде вписувати локальне ім’я плати, яке встановлювалося.

Після встановлення Putty, треба дізнатися адрес своєї плати, зробити це швидко можна у налаштуваннях свого маршрутизатору. Після цього можна перевірити за допомогою команди ping у консолі (ping <ip-адреса плати>). Далі у конфігураторі Putty вводиться адреса плати, і 22 порт за замовченням (рис. 4.4).

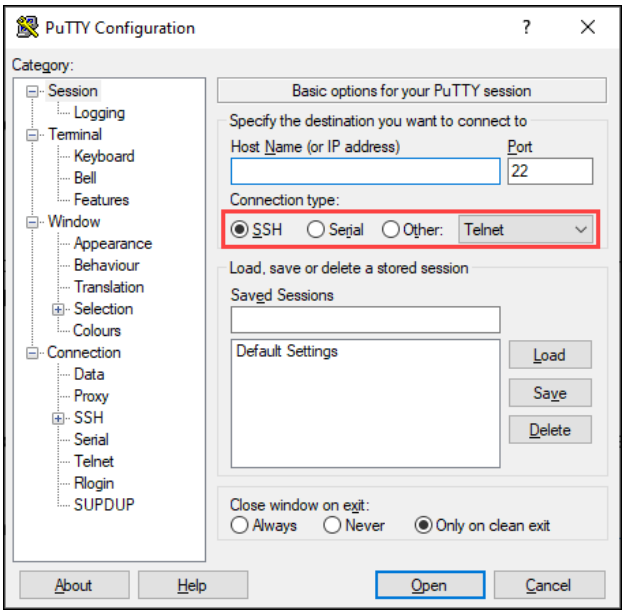


Рисунок 4.4 – Підключення до плати через SSH з’єднання

З’єднавшись з платою, вона попросить ввести логін і пароль користувача, після чого можна буде повністю керувати платою віддалено, та зробити відповідні конфігурації. Далі потрібно дозволити VNC підключення до плати. Це робиться у конфігураційному файлі Raspberry Pi.

Для потрапляння у це меню вводиться команда `sudo raspi-config` після чого переходимо в меню Interface Options і дозволяємо підключення VNC (рис. 4.5, 4.6). Далі налаштуавння з’єднання зберігаються і можна перейти до завантаження самої програми VNC Viewer, яка і буде виводити графічний інтерфейс плати Raspberry Pi.

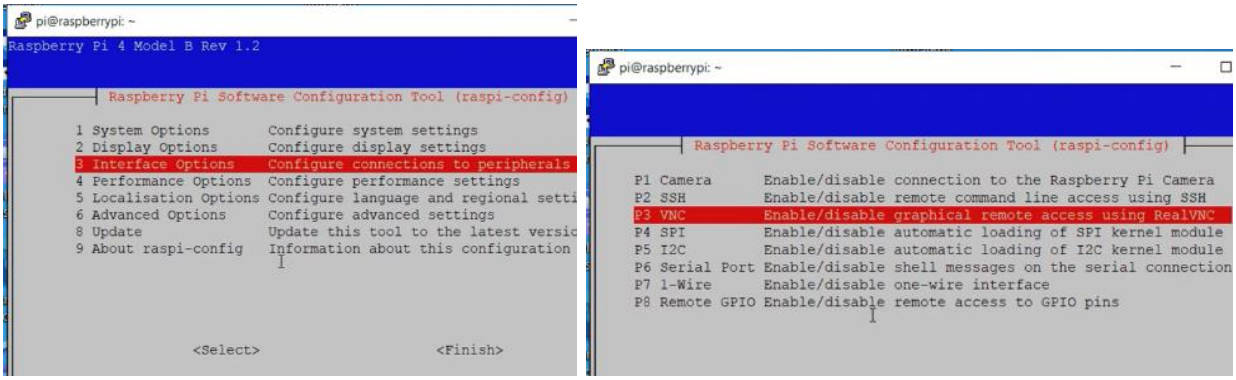


Рисунок 4.5,4.6 – Конфігураційний файл системи

Запустивши програму VNC Viewer, за таким же принципом вводимо ip-адресу плати, після чого програма попросить також ввести логін і пароль користувача. Після підключення з’явиться робочий стіл плати (рис. 4.7, 4.8).

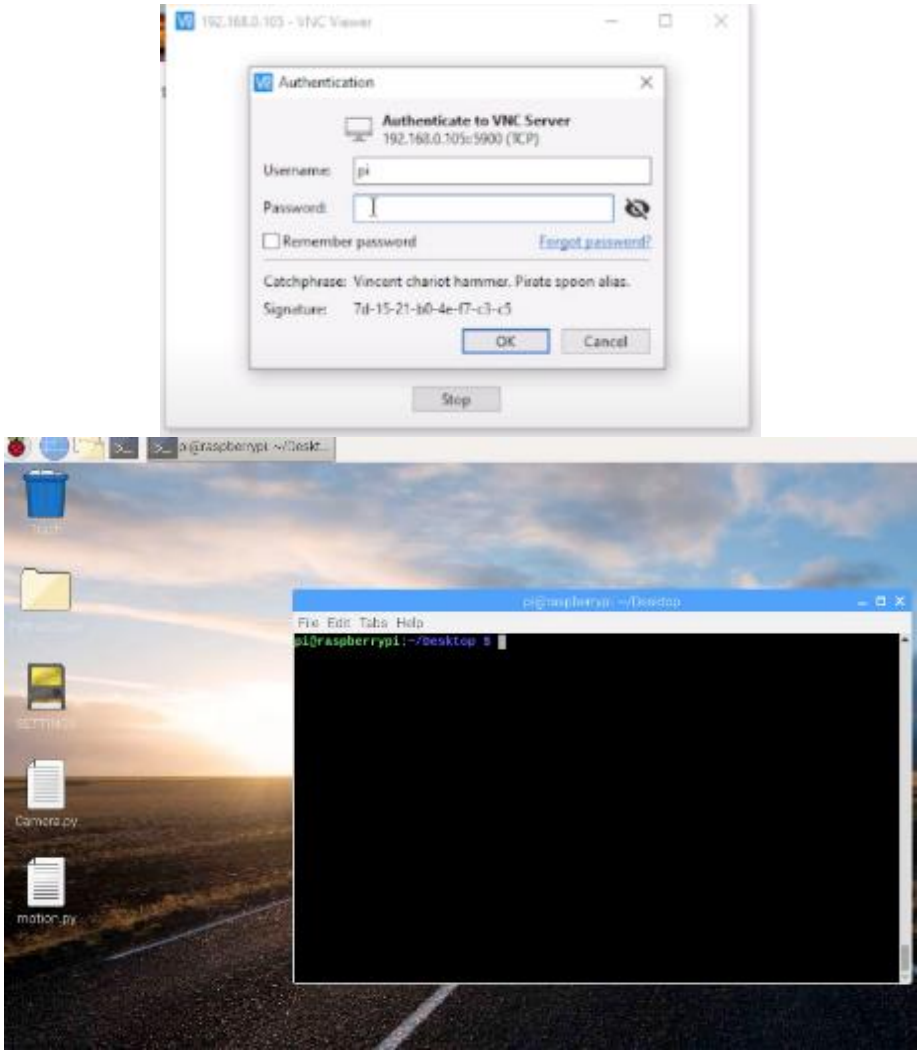


Рисунок 4.7, 4.8 – Підключення до GUI Raspberry Pi

4.1.2 Тестування роботи камери

Для тестування камери, її треба підключити до плати. Знеструмлюємо її і відключаймося від серверу. На платі знаходиться порт для підключення шлейфу камери, біля HDMI порту. Піднімається замок вгору і в середину вставляється шлейф, синьою міткою у сторону аудіо роз’єму. Після чого замок закривається (рис. 4.9).

Рисунок 4.9 – Підключення шлейфу камери до плати

Далі плату можна підключити до живлення і вже через графічний інтерфейс треба активувати камеру у платі. Це все також робиться через конфігураційний файл, як і у випадку з VNC. Тобто у терміналі вводитьсяс команда `sudo raspi-config` і обирається Interface Options -> Cameras (Enable) (рис. 4.10).

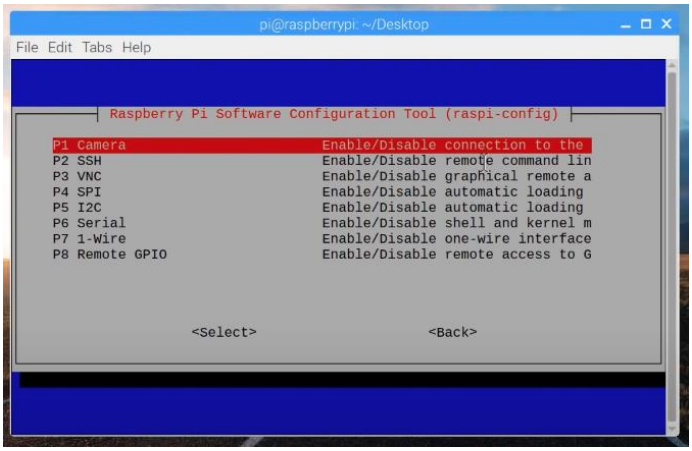


Рисунок 4.10 – Активація камери

Що б перевірити як працює камера, можна написати невеликий скрипт на Python. Для цього створюється файл і прописується такий код:

```
from picamera import Picamera
from time import sleep
camera  = PiCamera()
camera.start_preview(alpha=192)
sleep(1)
```

```
camera.capture(“/home/pi/Desktop/test.jpg”)
camera.stop_preview()
```

Після запуску даного скрипта, камера повинна зробити знімок та зберегти його на робочому столі (рис. 4.11).

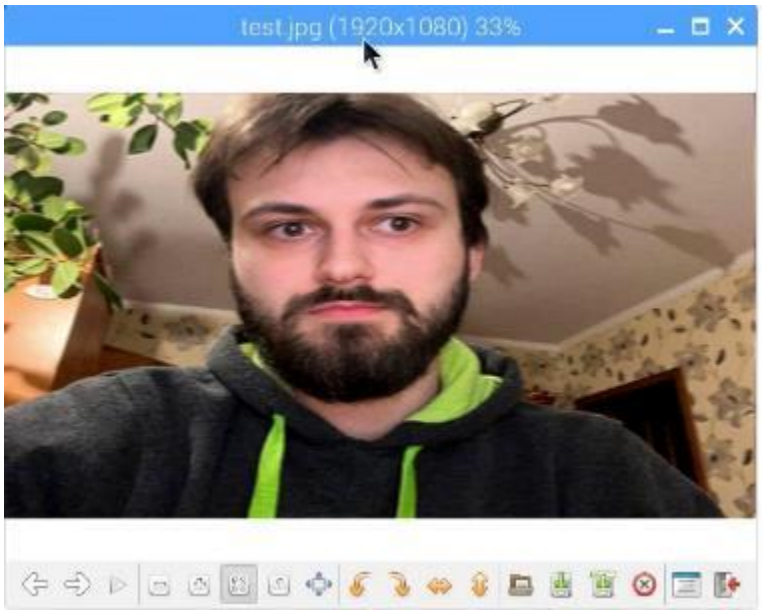


Рисунок 4.11 – Результат роботи камери

4.1.3 Тестування медіа потоку з камери на розпізнавання предметів

Тепер можна протестувати камеру з розпізнаванням об’єктів у ній. Для цього тестування буде написаний код для розпізнавання обличчя, з готовим класифікатором, який пропонує OpenCV (рис. 4.12).

```
# Face rec cascade
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# To capture video from Raspi-cam.
cap = cv2.VideoCapture(0)

while True:
  # Read the frame
  _, img = cap.read()
  # Convert to grayscale
  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  # Detect the faces
  faces = face_cascade.detectMultiScale(gray, 1.1, 4)
  # Draw the rectangle around each face
  for (x, y, w, h) in faces:
      cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
  # Display
  cv2.imshow('img', img)
  # Stop if escape key is pressed
  k = cv2.waitKey(30) & 0xff
  if k==27:
      break
# Release the VideoCapture object
cap.release ()
```

Рисунок 4.12 – Код тестування розпізнавання обличчя з камери

Так як відео складається з кадрів, ми їх перетворюємо у сірий формат та пропускаємо через бібліотеку OpenCV, і виконуємо виявлення заданого об’єкту на кожному кадрі. Після запуску написаної програми отримуємо відповідний результат [15] (рис. 4.13).

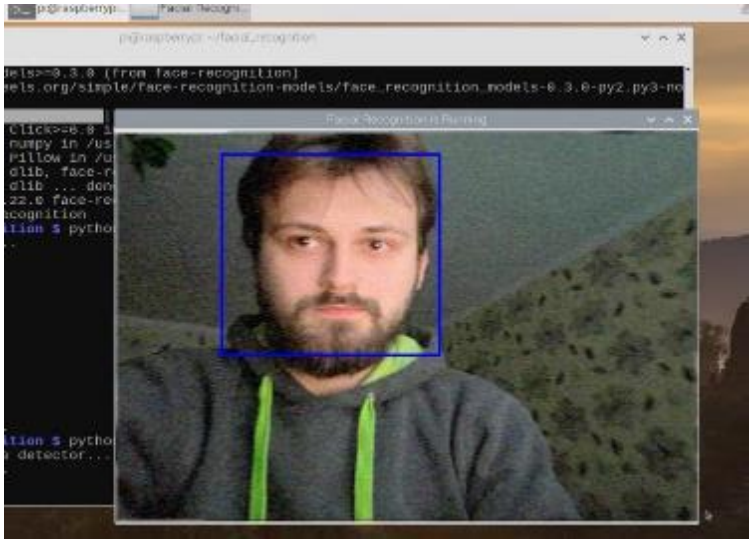


Рисунок 4.13 – Відео потік з розпізнаванням

4.2 Створення каскадного класифікатора

Для розпізнавання об’єктів на дорозі, нам потрібні відповідні класифікатори. Вони створюються на основі датасетів відповідного об’єкту, після чого на них створюються позитивні та негативні варіанти, йде тренування класифікатора, і все це зберігається у файлі з розширенням XML.

Для тренування був обраний датасет автомобілів, який містить у собі близько 8 тисяч фото для тренування і також кількість для тестування [16] (рис. 4.14).

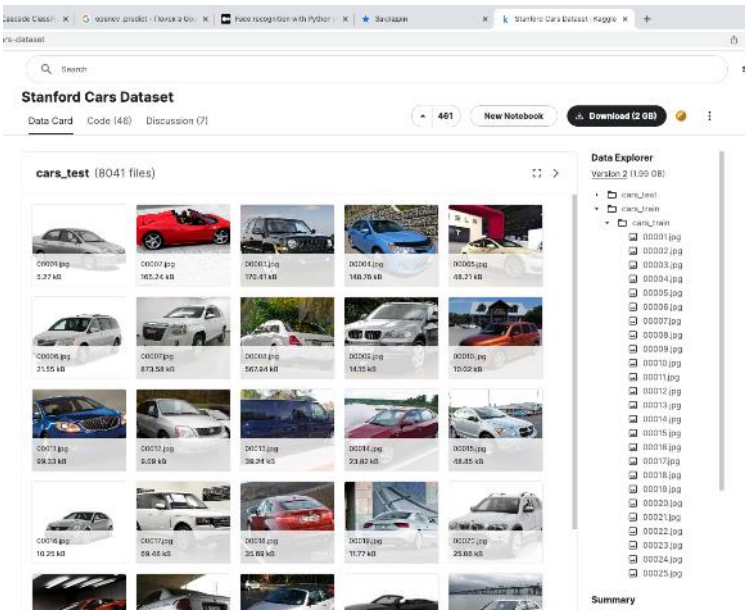


Рисунок 4.14 – Датасет автомобілів

Для тренування та тестування, було обрано меншу кількість наданих фото. Також слід пам’ятати що при русі, автомобіль відзнятий камерою може бути розмитий та не чіткий, тому якщо класифікатор тренувався тільки на статичних зображеннях, алгоритм може його не детектувати, або робити це з помилками. Отже треба відповідний датасет для навчання на окреме створення класифікатору машин у русі. Це також стосується і інших класифікаторів, які можуть бути використані у програмі, такі як знаки дорожнього руху, люди тощо.

Після цього створюється дві папки, позитивні та негативні фото (positive and negative). У позитивних фото знаходяться наші автомобілі, а у негативних можуть знаходитися траси, або дорожні вулиці з людьми тощо. Після цього робиться скрипт для створення текстового файлу для негативних результатів який буде вказувати тренуванню де ці файли.

```
def generate_negative_description_file():
    with open('neg.txt', 'w') as f:
        for filename in os.listdir('negative'):
            f.write('negative/' + filename + '\n')
```

Усі фали з кодом будуть представлені у кінці магістерської роботи. Після запуску, створюється текстовий файл з усіма фото, які були у негативній папці (рис. 4.15).

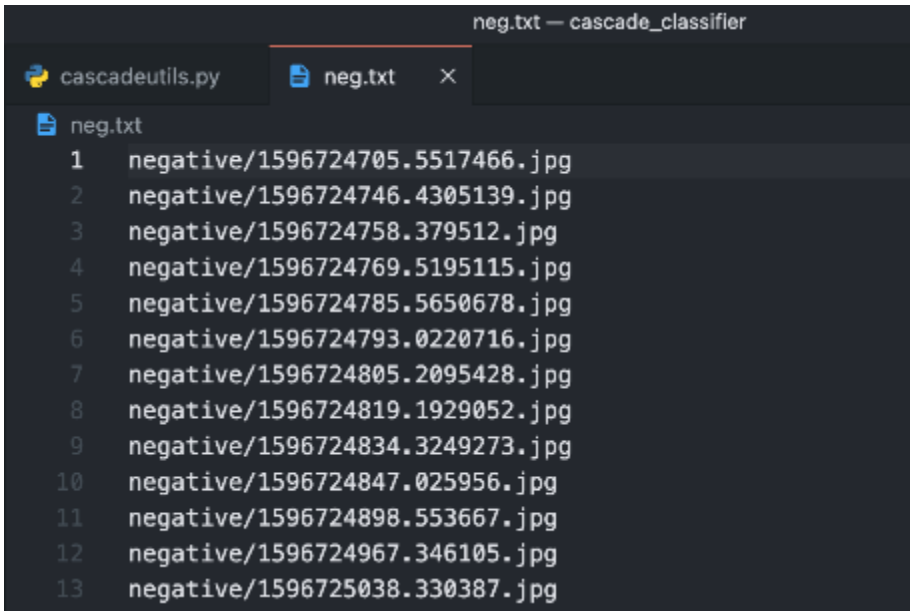


Рисунок 4.15 – Файл негативних варіантів

Таку ж саму операцію, треба зробити для позитивних рішень, але на позитивних фото потрібна бути зелена рамка, яка вказує на об’єкт, якому буде навчатися класифікатор.[20] Для цього використовується, завантажені інструменти OpenCV (opencv_annotation.exe). Викликається він командою «`C:/Users/…/opencv/build/x64/vc15/bin/opencv_annotation.exe --annotations=pos.txt --images=positive/`» і виділяються об’єкти на кожній фото, клавішою «C» перемикається на зелений квадрат для позитивного об’єкта, після чого створюється такий же текстовий список з позитивними моделями (рис. 4.16).

Рисунок 4.16 – Виділення об’єктів

Після цього треба створити векторний файл з позитивними значеннями, які будуть використовуватися у навчанні, за допомогою встановлених інструментів «`C:/Users/…/opencv/build/x64/vc15/bin/opencv_createsamples.exe -info pos.txt -w 24 -h 24 -num 1000 -vec pos.vec`»

– -info pos.txt – Розміщення позитивних значень

– -w 24 -h 24 – Ширина та висота (у пікселях) вихідних зразків

– -num 1000 – Кількість позитивних зразків для створення (зелені прямокутники, вказується більше чим є)

– -vec pos.vec – Назва вихідного файлу, що містить позитивні зразки для навчання

Отримаємо значення позитивних результатів (рис. 4.17).

```
pos.txt(106) : parse errorDone. Created 253 samples
(venv)
```

Рисунок 4.17 – Позитивні векторні результати

Тепер можна приступати до створення самого класифікатора. Використовуємо команду «`C:/Users/…/opencv/build/x64/vc15/bin/opencv_traincascade.ex`

```
e -data cascade/ -vec pos.vec -bg neg.txt -precalcValBufSize
6000 -precalcIdxBufSize 6000 -numPos 200 -numNeg 1000 -
numStages 12 -w 24 -h 24 -maxFalseAlarmRate 0.4 -minHitRate
0.999»
```

Нові команди для навчання:

– -data cascade/ – Директорія для збереження класифікатора

– -bg neg.txt – Негативні варіанти, або задній фон

– -precalcValBufSize 6000 – Розмір буфера пам’яті для швидкої обробки

– -precalcIdxBufSize 6000 – Розмір буфера пам’яті для швидкої обробки

– -numPos 200 – Кількість позитивних проб, використаних у навчанні для кожного етапу класифікатора, не повинні перевищувати вихідного значення

– -numNeg 400 – Кількість негативних проб, використаних у навчанні для кожного етапу класифікатора, найкращій варіант у два рази більше позитивних

– -numStages 12 – Кількість етапів каскаду для навчання (чим більше, тим краще буде результат)

– -maxFalseAlarmRate 0.4 – Максимальна бажана частота помилкових тривог для кожного ступеня класифікатора.

– -minHitRate 0.999 – Мінімальний бажаний показник попадання для кожного етапу класифікатора

Після вдалого тренування буде отримано каскадний класифікатор у форматі .xml, який збережеться у відповідну директорію. При старті роботи алгоритму, цей файл завантажуються у пам’ять лише один раз і використовуються для знаходження об’єктів у полі зору камери. Тобто при кожній обробці він не буде відвантажуватися кожного разу знов і знов, і

заповнювати пам’ять, що могло би призвести до сповільнення роботи алгоритму і зниженням частоти кадрів медіа-даних.

Після вдалого тренування, можна попробувати написати легкий скрипт для перевірки працездатності (рис. 4.18).

Рисунок 4.18 – Результат готового класифікатора

Усі ці дії потрібно повторити для кожного об’єкта, який буде розпізнаватися.

4.2.2 Реалізація системи compute vision для automotive

Після перевірок та тестування, можна зібрати систему в одне ціле, використовуючи саму плату, створені класифікатори, та фреймворк GStreamer для обробки відео потоку.

Для початку написаний програмний код буде у додатку до магістерської роботи. Також до нього необхідно додати відповідній інструкції по обробці вихідного зображення з камери, яке буде проходити через фреймворк GStreamer, а вже після потрапляти на розпізнавання об’єктів за допомогою OpenCV.

Пропускання відео через фреймворк дає змогу його швидшій обробці для розпізнавання об’єктів, завдяки стисненню кадрів, вони становяться меншого розміру, з мінімальною втратою якості. У коді прописаний пайплайн для обробки відео з камери (рис. 4.19).

```
camSet='libcamerasrc ! video/x-h264, width=640, height=480, framerate=30/1 ! \
h264parse ! videoconvert ! videoscale ! format=BGRx ! autovideosink'
```

Рисунок 4.19 – Пайплайн обробки відео

У даному пайплайні стоять відповідні команди, як пропускати вхідне відео з камери, ось основні з них:

- Video/x-h264, h264parse – Парсинг потоку h264
- Width, height – Висота та ширина кадру
- Framerate – Частота кадрів
- Videoconvert – Перетворюйте відеокадри між великою різноманітністю відеоформатів
- Format – формат відео
- Autovideosink – Це відеоприймач, який автоматично визначає відповідний відеоприймач для використання

Після запуску застосунку у консолі вводиться ще одна команда, для виводу відео-потоку на монітор «`gst-launch-1.0 shmsrc socket-path=/tmp/tmpsock ! 'video/x-h264, format=(string)I420, width=(int)1024, height=(int)768, framerate=(fraction)20/1' ! videoconvert ! ximagesink`». По ній ми звертаємося до створеного сокету у відповідній папці.

Після чого картинка з’являється з камери Raspberry Pi (рис. 4.20)

Рисунок 4.20 – Результат тестування системи computer vision для automotive

Як можна побачити з результату, було розпізнано людину, ближні дві машини, одну дальню і деякі знаки.

Також можна побачити помилкове розпізнавання знаку на будці з газетами. Для уникнення цієї проблеми, бажано перевчити класифікатор знаків і обрати більший масив даних для навчання.

Також як було вказано раніше, максимальна похибка для розпізнавання 40 відсотків, отже класифікатор не буде розпізнавати об’єкти на дорозі, в яких він не впевнений хоча б на 40%, адже на дорозі не повинно бути помилкових ситуацій.

У таблиці 4.1 представлено результати розпізнавання машини, використовуючи різні кути нахилу камери. Вона допомагає вибрати правильний кут нахилу камери, яка може бути встановлена у середині транспортного засобу.

Таблиця 4.1 – Порівняльна таблиці розпізнавання автомобіля.

Орієнтація обличчя	**Швидкість виявлення**	**Швидкість розпізнавання**
0° (перед і зад машини)	96,5 %	96 %
45° (уверх)	90,1 %	90 %

90°(уверх)	54,2 %	48,4 %
-45°(униз)	45 %	46 %
-90°(униз)	0 %	0 %

На сам перед, можна сказати що класифікатор працює чудово, але якщо опускати камеру униз машини, то точність і розпізнавання самого об’єкту стрімко падає. А при піднятті камери угору – тримається стабільно, хоча з виду на кришу також падає. Тому оптимальний кут для розпізнавання об’єкту це 0° - 60°. Для більш досконалого результату, треба більший датасет з усіх боків, а також гарне освітлення камери.

Також проведені тестування при обробці кадру з pipeline та без нього. Тестування частоти кадрів проводилося на поточному відео. Усі ці дані можна побачити нижче (рис. 4.21)

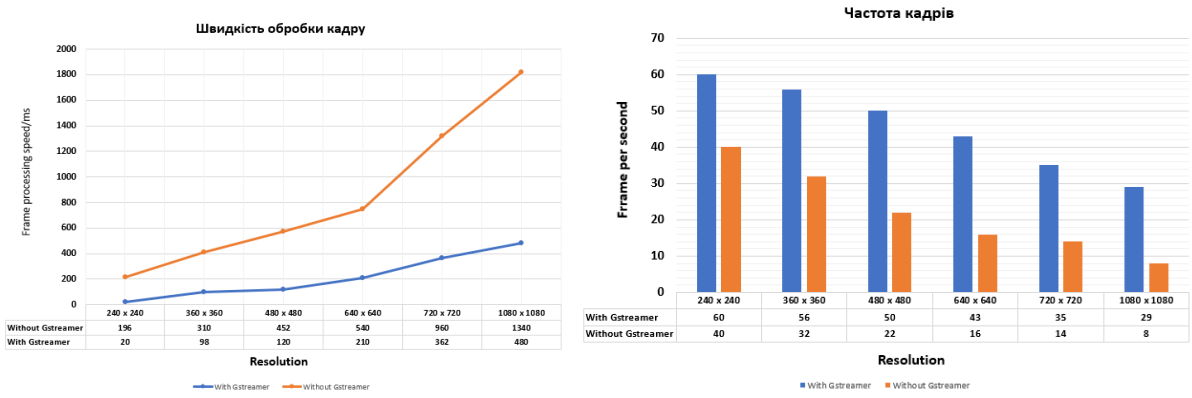


Рисунок 4.21 – Тестування швидкості обробки даних

Як зазначено на діаграмі, видно приріст швидкості і стабільного фреймрйту з використанням пайплайнів.

4.3 Блок-схема роботи алгоритму для системи computer vision

Далі представлено блок-схему алгоритму роботи написаного коду для системи computer vision під управлінням Raspberry Pi, та покроковий опис роботи алгоритму (рис. 4.22).

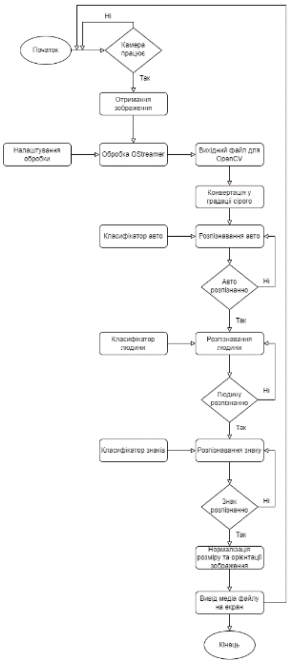


Рисунок 4.22 – Блок-схема алгоритму

Для початку на плату Raspberry Pi подається живлення, після чого можна запустити написаний код. Також важливо щоб до плати була під’єднана камера, з якої саме буде ти відео-потік і йде перевірка на працездатність камери. Після чого він зразу проходить через фреймворк GStreamer, з установленими налаштуваннями для обробки потоку.(рис. 4.23).

Рисунок 4.23 – Перевірка камери, та обробка медіа-файлу

Після обробки, кадр переходить до аналізу за допомогою бібліотеки для комп’ютерного зору OpenCV. (рис. 4.24).

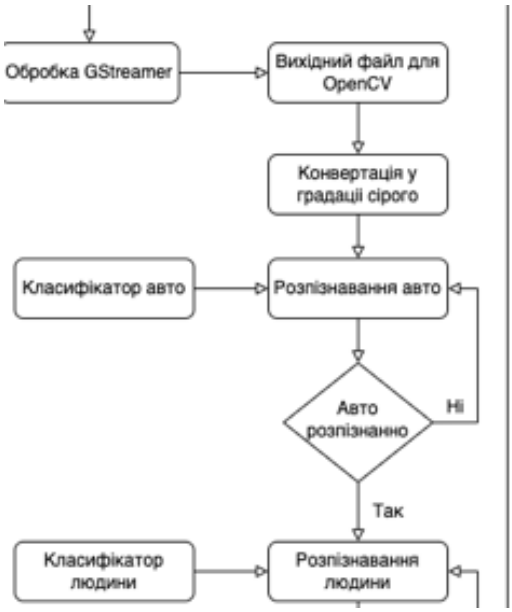


Рисунок 4.24 – Перевірка користувача

Картинка конвертується у градації сірого кольору і починається визначення об’єктів з потокового відео. Йде кожна перевірка на відповідний об’єкт, де до кожного встановлений свій, відповідний класифікатор.

Якщо об’єкт не знайдений, процес повторюється. Ця операція проводиться для кожного встановленого класифікатора. Після обробки кадру, картинка нормалізується і на вихід вже подається відео відповідного формату, з обробкою об’єктів і їх визначенням на медіа-файлі.

4.4 Потенційна сфера застосування

Також не слід забувати про вдосконалення спроектованого апаратно-програмного модулю і його потенціал у майбутньому, додавання нових модулів, та вдосконалення самого програмного забезпечення під відповідні задачі.

Перш за все можна почати з легкого «cover» кейсу, який буде захищати камеру від зовнішніх факторів (вода, пил, бруд), а також від невеликого падіння.[19] Самий простий спосіб це створення корпусу на 3D принтері, під розмір усіх компонентів (рис. 4.25).

Рисунок 4.25 – ESP32-CAM у корпусі фіктивної камери

Також у сьогоднішніх реаліях, з агресією країни-терорист проти України, даний апаратно-програмний модуль може бути вдосконалені у проект з дронами, для розпізнавання ворожої піхоти та техніки за допомогою цього програмного коду. Це допоможе більш ефективніше і швидше ЗСУ реагувати і нищіти окупантів на місцевості, де їх важко помітити. (рис. 4.26).

Рисунок 4.26 – Майбутні сфери застосування

Далі хорошою ідеєю буде розроблення переносного живлення для плати, щоб можна було з нею працювати не тільки в домашніх умовах, а й де інде (рис. 4.27).

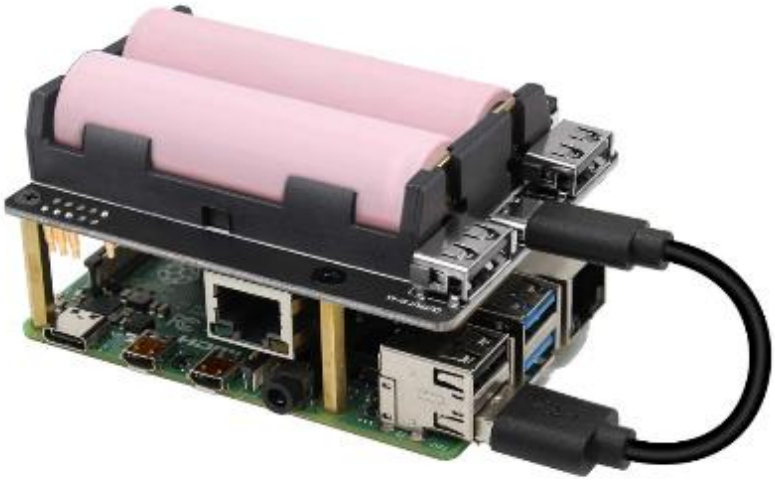


Рисунок 4.27 – Переносне живлення для плати

Висновки до розділу 4

У четвертому розділі зібрано та протестована система computer vision і написаний програмний код для роботи з модулями, за допомогою мови програмування Python. Створений пайплайн для обробки вхідного відео-потоку та поставлені налаштування для вихідного зображення.

Створені класифікатори для відповідних об’єкті, які треба детекувати. Застосована бібліотека для розпізнавання об’єктів на медіа-файлі і вивід зображення кінцевому користувачу.

Спроектована і описана блок-схема роботи алгоритму і розписаний принцип його роботи.

Надані варіанти вдосконалення системи шляхом вироблення захисного кейсу для плати і камери, використання даного проекту сумісно з дронами, та подача живлення, не від розетки.

ВИСНОВКИ

У період виконання магістерської роботи було розроблено план поетапної розробки даного проекту.

Проаналізовано різноманітну літературу та патентну інформацію для проектування апаратно програмного модуля для розпізнавання об’єктів, у automotiv за допомогою одноплатного комп’ютера Raspberry Pi, та додавання масиву камер.

Зібрав такий модуль можна зекономити гроші на купівлі дорогих аналогів для automotive з комп’ютерним зором. Сам зібраний модуль нічим не поступається високо бюджетним аналогам, лише за якістю картинки, але цей фактор вирішується звичайною модернізацією камери на модулі.

По завершенню магістерської роботи була отримана система computer vision для automotive на основі Raspberry Pi для розпізнавання об’єктів на дорозі та перешкод, з використанням спеціальних модулів і фреймворків.

– Вирішено проблему відсутності відповідної кількості портів для підключення декількох камер одночасно.

– Зібрано прототип апаратно програмного модуля для розпізнавання об’єктів на дорозі.

– Застосований фреймворк для швидкої обробки кадру.

– Знайдені та висвітлені переваги одноплатного комп’ютера Raspberry Pi над іншими моделями даного ряду.

– Запропоновано ймовірні шляхи розвитку даного проекту у різноманітних сферах діяльності.

– Модуль працює від різних джерел живлення.

Цей проект може стати частиною більш масштабного проекту з використанням додаткових датчиків та модулів в одній зв’язці. Його можна використати що б зробити свій оригінальний проект, який у майбутньому може стати дуже прибутковим та визнаним.

Завдяки цій магістерській роботі можна вивести популярність штучного інтелекту (а саме комп’ютерного зору) з використанням одноплатного комп’ютера Raspberry Pi й в загалом технологій computer vision на новий рівень. Зацікавити як молоде покоління, так і більш досвідчених розробників.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. The best Raspberry Pi alternatives and other cheaper options, 2022, URL: https://zd.net/3IrVwPe (Last accessed: 12.01.2023).

2. Raspberry Pi 3 vs 4, 2021 URL: https://bit.ly/3I9qFWj (Last accessed: 14.01.2023).

3. Raspberry Pi 4 Review, 2019 URL: https://bit.ly/3K9GoY0 (Last accessed: 12.01.2023).

4. NVIDIA Jetson Nano 2GB Developer Kit Review, 2020.URL: https://bit.ly/3YzBiZB (Last accessed: 20.12.2022).

5. Mohamed Elgendy. Deep Learning for Vision Systems, Volume: 1, 2020, p. 480.

6. Joseph Howse. Learning OpenCV 4 Computer Vision with Python 3, Volume: 3, 2020, p. 372.

7. Leonard Eddison. Coding: Raspberry Pi & Python, Volume: 1, 2018, p. 184.

8. Посилання на проект BFRMR1 Obstacle detection using Raspberry Pi and OpenCV. URL: https://bit.ly/3S3wvwK (дата звернення: 24.12.2022).

9. Посилання на проект Object Tracking Robot Using Raspberry Pi, 2019, URL: https://bit.ly/3Ec57XT (дата звернення: 26.12.2022).

10. Wim Taymans. GStreamer Application Development. 2019, p. 164.

11. Olivier Crête. GStreamer Conference. CC BY-SA 3.0, 2019. URL: https://bit.ly/3jWjB7v (Last accessed: 18.01.2023).

12. Схематичне зображення пайплайну What is GStreamer. 2019, URL: https://bit.ly/3XzKdJa (дата звернення: 20.01.2023).

13. Creating Gstreamer Multimedia Pipeline. 2020, URL: https://bit.ly/3S4mVd3 (Last accessed: 20.01.2023).

14. The Maths behind Contour Moments from OpenCV. 2021 URL: https://bit.ly/3YXpQ9E (Last accessed: 14.01.2023).

15. Ashwin Pajankar. Raspberry Pi Computer Vision Programming.Volume: 2, 2020, p. 306.

16. База даних Stanford Cars Dataset. 2018, URL: https://bit.ly/3KlOHjk (дата звернення: 22.01.2023).

17. Nora Kassner. Real-time face detection with Haar cascades. 2021. URL: https://bit.ly/3XElwv3 (Last accessed: 22.01.2023).

18. VNC remote access technology. 2020, URL: https://bit.ly/3XFSxqC (Last accessed: 05.01.2023).

19. Housing for Telraam Traffic counting camera. Jan. 02, 2021. URL: https://bit.ly/3YAXm65 (Last accessed: 26.01.2023).

20. Документація OpenCV training data. URL: https://bit.ly/3Kd5tBe (дата звернення: 24.01.2023).

21. Смолянік Я. В., Пузирьов С. В. Застосування GStreamer для передачі стрімінгового відео на базі Raspberry Pi. Могилянські читання – 2022 : тези доп. XXV Всеукр. наук.-метод. конф. Миколаїв, 7–11 листоп. 2022 р. Миколаїв : Чорном. нац. ун-т ім. Петра Могили, 2022. С. 88–91.

22. Ngurah Desnanjaya I. G. M., Arsana I. N. A. Home security monitoring system with IoT-based Raspberry Pi. *Indonesian Journal of Electrical Engineering and Computer Science*. 2021. Вип. 22, № 3.

23. Bowman R. W., Vodenicharski B., Collins J. T., Stirling J. Flat-Field and Colour Correction for the Raspberry Pi Camera Module. *Journal of Open Hardware*. 2020. Вип. 4, № 1.

24. Artificial Artificial Intelligence / L. Chan et al. AIES '20: AAAI/ACM Conference on AI, Ethics, and Society, New York NY USA. New York, NY, USA, 2020. URL: https://doi.org/10.1145/3375627.3375870 (Last accessed: 20.12.2022).

25. Automotive Grade Linux Software Architecture for Automotive Infotainment System / P. Sivakumar et al. 2020 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 26–28 February 2020. 2020. URL: https://doi.org/10.1109/icict48043.2020.9112556 (Last accessed: 02.01.2023).

26. Baker S., Taymans W., Wingo A. GStreamer Application Development 1.10.1. 12th Media Services, 2018. 164 p.

27. Computer Vision. Elsevier, 2018. URL: https://doi.org/10.1016/c2015-0-05563-0 (Last accessed: 28.12.2022).

28. Computer Vision / Y. Wang et al. KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event CA USA. New York, NY, USA, 2020. URL: https://doi.org/10.1145/3394486.3406710 (Last accessed: 03.01.2023).

29. Marinchak C. M., Forrest E., Hoanca B. Artificial Intelligence. International Journal of E-Entrepreneurship and Innovation. 2018. Vol. 8, no. 2. P. 14–24. URL: https://doi.org/10.4018/ijeei.2018070102 (Last accessed: 15.12.2022).

30. Trovao J. P. Trends in Automotive Electronics [Automotive Electronics]. IEEE Vehicular Technology Magazine. 2019. Vol. 14, no. 4. P. 100–109. URL: https://doi.org/10.1109/mvt.2019.2939757 (Last accessed: 17.12.2022).

ДОДАТОК А

Програмний код апаратно-програмного комплексу

#Файл automotive_launch

```
import cv2
print(cv2.__version__)

car_detect = cv2.CascadeClassifier("TruckClassifier.xml")
human_detect = cv2.CascadeClassifier("HumanClassifier.xml")
sign_detect = cv2.CascadeClassifier("SignClassifier.xml")
#Additional classifiers are used for blury images,
#on separate datasets

def detect_truc_func(img):
    cameraSource = img.copy()

    #Init Classifiers
    car_rectangle = car_detect.detectMultiScale(cameraSource)
    human_rectangle = human_detect.detectMultiScale(cameraSource)
    sign_rectangle = sign_detect.detectMultiScale(cameraSource)

    #Props
    carName = "car"
    humanName = "human"
    signName = "sign"
    fontFace = cv2.FONT_HERSHEY_SIMPLEX
    fontScale = .8
    colorCar = (0, 255, 0)
    colorHuman = (255, 0, 0)
    colorSign = (255, 255, 0)
    lineType = cv2.LINE_4

    #Draw rectangles and names for all objects

    for (x, y, w, h) in car_rectangle:
        cv2.rectangle(cameraSource, (x, y), (x + w, y + h), (0, 255, 0), 2)
        top = y - 15 if y - 15 > 15 else y + 15
        cv2.putText(cameraSource, carName, (x, top), fontFace,
                    fontScale, colorCar, lineType)

    for (z, v, b, n) in human_rectangle:
        cv2.rectangle(cameraSource, (z, v), (z + b, v + n), (255, 0, 0), 2)
        top = y - 15 if y - 15 > 15 else y + 15
        cv2.putText(cameraSource, humanName, (x, top), fontFace,
                    fontScale, colorHuman, lineType)

    for (q, w, e, r) in sign_rectangle:
        cv2.rectangle(cameraSource, (q, w), (q + e, w + r), (255, 255, 0), 2)
        top = y - 15 if y - 15 > 15 else y + 15
        cv2.putText(cameraSource, signName, (x, top), fontFace,
                    fontScale, colorSign, lineType)

    return cameraSource


# Cam properties
fps = 25
```

```
frame_width = 720
frame_height = 720

dispW=1280
dispH=720
flip=0

#Gstreamer pipeline
camSet='libcamerasrc ! video/x-h264, width=640, height=480, framerate=30/1 ! \
h264parse ! videoconvert ! videoscale ! format=BGRx ! autovideosink'
cam= cv2.VideoCapture(camSet)

gst_str = "queue ! identity ! shmsink wait-for-connection=1 socket-path=/tmp/tmpsock
shm-size=20000000 sync=true"

# Create videowriter as a SHM sink

fourcc = cv2.VideoWriter_fourcc('I','4','2','0')
out = cv2.VideoWriter(gst_str, fourcc, fps, (frame_width, frame_height), True)


while True:
    ret, outVid = cam.read()
    car_frame = detect_truc_func(outVid)
    #live
    cv2.imshow('Output',outVid)
    #frame = cv2.flip(frame,1)
    #to remote
    out.write(outVid)

    if cv2.waitKey(1)==ord('q'):
        break
cam.release()
cv2.destroyAllWindows()

#gst-launch-1.0 shmsrc socket-path=/tmp/tmpsock ! 'video/x-raw, format=(string)I420,
width=(int)1024,   height=(int)768,   framerate=(fraction)20/1'   !   videoconvert   !
ximagesink
#for launch proj
```

#Файл FaceCam_detect

```
import cv2
# Face rec cascade
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# To capture video from Raspi-cam.
cap = cv2.VideoCapture(0)

while True:
  # Read the frame
  _, img = cap.read()
  # Convert to grayscale
  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  # Detect the faces
  faces = face_cascade.detectMultiScale(gray, 1.1, 4)
  # Draw the rectangle around each face
  for (x, y, w, h) in faces:
      cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
```

```
  # Display
  cv2.imshow('img', img)
  # Stop if escape key is pressed
  k = cv2.waitKey(30) & 0xff
  if k==27:
      break
# Release the VideoCapture object
cap.release ()
```

#Файл cascadeutils

```
import os

def generate_negative_description_file():
    # open the output file for writing. will overwrite all existing data in there
    with open('neg.txt', 'w') as f:
        # loop over all the filenames
        for filename in os.listdir('negative'):
            f.write('negative/' + filename + '\n')

# generate positive description file using:
# $ C:/Users/…/opencv/build/x64/vc15/bin/opencv_annotation.exe --annotations=pos.txt
--images=positive/

# You click once to set the upper left corner, then again to set the lower right
corner.
# Press 'c' to confirm.
# Or 'd' to undo the previous confirmation.
# When done, click 'n' to move to the next image.
# Press 'esc' to exit.
# Will exit automatically when you've annotated all of the images

# generate positive samples from the annotations to get a vector file using:
# $ C:/Users/…/opencv/build/x64/vc15/bin/opencv_createsamples.exe -info pos.txt -w 24
-h 24 -num 1000 -vec pos.vec

# train the cascade classifier model using:
# $ C:/Users/…/opencv/build/x64/vc15/bin/opencv_traincascade.exe -data cascade/ -vec
pos.vec -bg neg.txt -precalcValBufSize 6000 -precalcIdxBufSize 6000 -numPos 200 -
numNeg 1000 -numStages 12 -w 24 -h 24 -maxFalseAlarmRate 0.4 -minHitRate 0.999
```