КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра радіотехніки та радіоелектронних систем

«На правах рукопису»

Робота допущена до захисту в ЕК
рішенням кафедри радіотехніки та радіоелектронних систем
від __ травня 2024 року, протокол № ___.
Завідувач кафедри доктор фіз.-мат. наук, професор
_____ Ігор АНІСІМОВ

ДИПЛОМНА РОБОТА МАГІСТРА
на тему:
«СИСТЕМА ВІДЕОНАГЛЯДУ
З ВИКОРИСТАННЯМ КОМП'ЮТЕРНОГО ЗОРУ»

Виконав:
студент 2-го курсу магістратури
денної форми навчання
спеціальності 172 - Телекомунікації та радіотехніка
ОНП «Інформаційна безпека телекомунікаційних систем і мереж»
Шаповаленко Владислав Анатолійович_____

Науковий керівник:
Асистент кафедри радіотехніки
та радіоелектронних системб
кандидат фіз.-мат. наук
Котов Михайло Миколайович_____

Рецензент:
Завідувач кафедри кібербезпеки та комп'ютерної інфенерії,
Київського національного університету будівництва та архітектури,
професор, доктор технічних наук
Хлапонін Юрій Іванович_____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів без
відповідних посилань
Студент_____ Владислав ШАПОВАЛЕНКО

Київ – 2024

РЕФЕРАТ

Дипломна робота магістра: 56 сторінок, 28 рисунків, 18 джерел, 2 додатки.

МІКРОКОНТРОЛЕР, ARDUINO, C++ PROGRAMMING LANGUAGE, КОМП'ЮТЕРНИЙ ЗІР, PYTHON PROGRAMMING LANGUAGE, БІБЛІОТЕКА OPENCV, КАСКАД ХААРА, ОПТИЧНИЙ ТРЕКІНГ

Об'єкт розроблення – макет системи відеонагляду з використанням комп'ютерного зору.

Мета роботи – розробити систему відеотрекінгу з можливістю відцентровування позиції камери відносно відстежуваного об'єкту .

Розроблена система відеонагляду на основі платформи Arduino та його модулів, а також програми мовою Python з використанням бібліотеки OpenCV для відстеження зміни положення обличчя. Система працює в двох режимах: за допомогою джойстика та автоматично.

- В режимі управління джойстиком користувач може керувати положенням камери за допомогою джойстика, який підключений до мікроконтролера.
- В автоматичному режимі програма з використанням комп'ютерного зору самостійно шукає обличчя та наводить на нього камеру.

Розроблена система може бути використана в якості основи для інших проектів для роботи з комп'ютерним зором та автоматичним відеотрекінгом, а також поглибленого дослідження цієї технології. В роботі наведений детальний опис створення такого роду системи.

В подальшому систему можна вдосконалити таким чином, щоб вона була повністю автономною і не залежала від персонального комп'ютера, що виконує пошук обличчя. Функцію обробки відеопотоку, а також управління системою позиціонування можна перенести на одноплатний комп'ютер (наприклад Raspberry Pi), що зробить систему автономною.

Список рисунків

Зміст

Вступ

Відеоспостереження - це технологія, яка використовується для запису та відтворення відео з метою спостереження за людьми або об'єктами. Вона існує вже понад 80 років і використовується в різних сферах, включаючи безпеку, охорону здоров'я та промисловість [1].

Зараз набувають все більшого розповсюдження системи відеоспостереження з використанням комп’ютерного зору. Для прикладу можна розглянути такі об’єкти їх застосування.

Міжнародні аеропорти є важливими об’єктами критичної інфраструктури, тому система їх безпеки складається з багатьох важливих елементів, спрямованих на гарантування безпеки пасажирів і працівників аеропорту. Необхідно розвинути аспекти безпеки на найвищому рівні, зокрема слід проаналізувати можливості підтримки операторів відеоспостереження в аеропорту за допомогою сучасних рішень штучного інтелекту (AI) [2].

Аналізу дорожнього трафіку. Відео спостереження за дорожнім рухом у реальному часі потребують постійного нагляду для моніторингу та вжиття відповідних заходів у разі смертельних аварій. Однак безперервний контроль за ними під наглядом людини є втомливим і може призводити до помилок. Тому було запропоновано підхід глибокого навчання для автоматичного виявлення та локалізації дорожньо-транспортних пригод, формулюючи проблему як виявлення аномалій. Модель виконано на основі реальних наборів даних відеоспостереження за дорожнім рухом і досягнуто значних результатів як якісно, так і кількісно [3].

Останнім часом розробка автоматизованих систем відеоспостереження (АСВ) стала надзвичайно важливою для забезпечення безпеки населення під час масових заходів. В технологіях машинного навчання (ML) і глибокого навчання (DL) вже з’являються можливості тренування та навчання окремих компонентів. Тож важливо проводити огляд можливостей впровадження таких технологій в системи відеонагляду [4].

Підсумовуючи, можна зробити висновок, що системи відеоспостереження з відеотрекінгом на основі машинного навчання набувають все більшої популярності завдяки своїй універсальності та ефективності. Однак слід зазначити, що вартість таких систем може бути доволі високою, що робить їх недоступними для багатьох людей, які хочуть експериментувати з машинним зором.

Мета та актуальність роботи

Метою даної дипломної роботи є розробка системи відеотрекінгу з можливістю відцентровування позиції відеокамери відносно відстежуваного об’єкта.

Для досягнення цієї мети були поставлені такі задачі:

- Розробити систему позиціонування камери за допомогою сервоприводів та платформи Arduino.
- Розробити програму керування сервоприводами на платформі Arduino з можливістю отримання координат об'єкта з програми на Python.
- Реалізувати алгоритм розпізнавання та відстеження об'єктів у відеопотоці камери за допомогою бібліотеки OpenCV та методів машинного навчання.
- Інтегрувати роботу програми на Python з платформою Arduino для відправлення координат об'єкта та керування сервоприводами.

Наразі системи відеонагляду з використанням технологій комп’ютерного зору користуються великим попитом. Оскільки вони можуть суттєво спростити роботу операторам таких систем якщо мова йде, на приклад, про обробку великих обсягів даних чи про точне управління системами позиціонування.

Однак вартість подібних систем відеотрекінгу може бути доволі високою, що робить їх недоступними для багатьох людей, які хочуть експериментувати з машинним зором.

Отже, створення макету такої системи з доступних компонентів, а також створення докладного посібника з розробки такого роду систем може дати можливість здобути знання та навички в цій галузі, що і робить роботу актуальною.

1. Технології комп’ютерного зору та їх застосування

1.1. Машинне навчання

Машинне навчання (МН, Machine Learning, ML) — великий підрозділ технології штучного інтелекту, що вивчає методи побудови алгоритмів, здатних навчатися. Цей розділ досліджує методи, що дозволяють комп'ютерам покращувати свої характеристики на основі отриманого досвіду [5]. До основних етапів машинного навчання входять:

1. Збір та підготовка даних:
   1.1. Збір великої кількості даних, які містять інформацію про вхідні змінні та бажаний результат.
   1.2. Підготовка даних, включаючи їх очищення, нормалізацію та розділення на навчальний та тестовий набори.
2. Вибір математичної моделі або алгоритму, який буде використовуватися для вирішення завдання на основі даних. Це може бути лінійна регресія, дерево рішень, нейронна мережа, тощо.
3. Навчання моделі. Використання навчального набору даних для навчання обраної моделі. Модель аналізує дані та навчається визначати залежності між вхідними змінними та бажаним результатом.
4. Оцінка моделі. Використання тестового набору даних для оцінки точності та ефективності моделі. Це допомагає виявити, наскільки добре модель вирішує завдання.
5. Налаштування та оптимізація. В разі необхідності внесення змін у модель або її гіперпараметри для покращення результатів.
6. Впровадження навченої моделі для вирішення практичних завдань, які вимагають класифікації, прогнозування або інших операцій на основі вхідних даних.

1.2. Трекінг об’єктів. Оптичний трекінг

Існує кілька видів трекінгу об'єктів [6]:

- оптичний трекінг (Optical Tracking) використовує оптичні сенсори та камери для відстеження руху об'єктів. Цей метод часто використовується у відеоіграх та віртуальній реальності;
- інерціальний трекінг (Inertial Tracking) базується на вимірюваннях прискорення та обертової швидкості об'єкта за допомогою гіроскопів та акселерометрів. Використовується в носимих пристроях та навігаційних системах;
- магнітний трекінг (Magnetic Tracking) використовує магнітні датчики для визначення положення об'єкта у відношенні до магнітних полів. Застосовується в хірургії та віртуальній реальності;
- акустичний трекінг (Acoustic Tracking) базується на вимірюваннях звукових хвиль для визначення положення об'єктів. Застосовується, наприклад, у водному дослідженні;
- радіочастотний трекінг (Radio Frequency Tracking) використовує радіочастотні сигнали для визначення місцезнаходження об'єктів. Застосовується у системах GPS та в мережах зв'язку.

Кожен вид трекінгу має свої особливості та застосування у різних галузях, зокрема у науці, медицині та багатьох інших. Однак в рамках даної роботи більш докладно розглянемо саме оптичний трекінг.

Визначення об’єктів, що рухаються є важливою задачею у сфері комп’ютерного зору для різних роботизованих пристроїв. Комп'ютерний зір включає в себе отримання цифрового зображення, його обробку, аналіз і розпізнавання зображень, використовуючи статистичні методи і моделі побудовані за допомогою фізики, геометрії, статистики і теорії статистичного навчання.

Завдання розробки алгоритмів систем відеоспостереження в даний час є надзвичайно актуальним і зачіпає багато аспектів життя людини. З появою дешевих цифрових відеореєстраторів стало можливим обробляти вимірювальні дані за допомогою комп'ютера.

Прикладами реалізації таких завдань є охорона периметра і внутрішньої території об'єкта, виявлення і розпізнавання осіб, розпізнавання руху. Системи відеоспостереження знаходять широке застосування в задачах боротьби з тероризмом і моніторингу дорожнього руху.

Великий інтерес в області алгоритмів супроводження руху зумовило поширення потужних комп’ютерів, доступність високоякісних камер із невеликою ціною та збільшення потреб для автоматизованого аналізу відео.

Основні ключові кроки відеоаналітики: розпізнавання об’єкта, що рухається, відстеження об’єкта кадр за кадром, і аналіз об’єктів для визначення їх поведінки.

Оптичний трекінг дозволяє створювати досить точні та динамічні системи відстеження руху, що робить його важливою технологією для багатьох сучасних додатків і дисциплін.

1.3. Особливості і проблематика відстеження об’єктів

При створенні та роботі відстежувача виникають деякі основні проблеми, пов’язані з виглядом об’єктів та їх схожістю на інші об’єкти на сцені. Наприклад, вигляд інших об’єктів може накладатися на задній фон, що ускладнює пошук об’єкта, який ми очікуємо побачити. На рисунку 1.1 зображено приклад помилкового визначення об’єкту.

Рисунок 1.1 – Приклад помилкового визначення обличчя в кадрі.

Крім того, існують інші фактори, які ускладнюють визначення об’єктів в площині об’єктива:

- Зміна позиції: об’єкт, що рухається, може змінювати свій вигляд на площині зображення, наприклад, якщо він обертається.
- Зміна освітлення: колір, напрямок та інтенсивність освітлення впливають на вигляд об’єкта. Зміна освітлення може викликати проблеми в сусідніх сценах, впливаючи на сприйняття зображення через лінзу камери.
- Шум: процес отримання зображення пов’язаний з певною долею шуму, який залежить від якості матриці камери.
- Перекриття об’єктів: спостерігання за ціллю у випадках, коли вона частково або повністю перекрита іншим об’єктом на зображенні.

1.4. Способи визначення об’єктів

В задачах спостереження об’єкт може бути визначений як будь-що важливе для подальшого аналізу. Об’єкти представлені їх виглядом та формою [7]. Способи представлення, що широко використовуються для відстеження, зображено на рисунку 1.2.

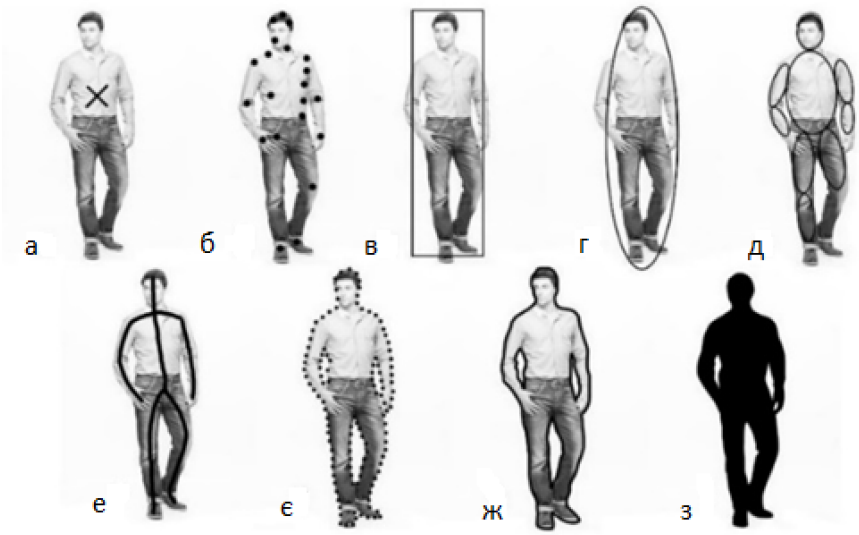

Рисунок 1.2 – Способи виділення об’єкта: а – центр об’єкта, б – особливі точки, в – форма об’єкта у вигляді прямокутника, г – об’єкт у формі еліпса, д – поєднання фігур, е – скелетна модель, є та ж – контур об’єкта, з – силует.

Визначати та виділяти об’єкти можна такими методами:

- об’єкти представляють собою точки, або набір точок (рис. 1.2б). Підходить для стеження за невеликими областями на зображенні;
- форма об’єкта представлена прямокутником, еліпсом тощо (рис. 1.2в,г). Підходить для відстежування твердих об’єктів;
- контур представляє собою границі об’єкта (рис. 1.2є, ж)). Силует знаходиться всередині контуру (рис. 1.2з). Підходить для стеження за нетвердими об’єктами;

- з’єднані об’єкти складаються з частин тіла, що тримаються разом за рахунок суглобів. Наприклад, тіло людини - це з’єднаний об’єкт, що складається з ніг, рук, голови, торсу та суглобів (рис. 1.2д).
- Скелетна модель – одна з найпоширеніших моделей для розпізнавання об’єктів суглобів (рис. 1.2е).

У системах відеоспостереження важливу роль відіграє визначення правильних ознак для відстеження об'єктів. Ці візуальні ознаки повинні бути унікальними, щоб можна було чітко відрізнити об'єкт від інших. Вони також тісно пов'язані з представленням об'єкта в кадрі. Наприклад, контур об'єкта є характерною ознакою, яку можна відобразити за допомогою алгоритмів виділення контурів.

До основних візуальних ознак належать:

- *видимий колір об'єкта*, який може бути представлений у просторах RGB або HSV;
- *границі об'єкта*, що створюють значні зміни в інтенсивності зображення, тому їх можна використовувати для виявлення та відстеження об'єктів. Важливою перевагою границь є те, що вони менш чутливі до змін освітлення, порівняно з візуальними ознаками кольору;
- *оптичний потік*, який описує видимий рух об'єктів, поверхонь та граней у візуальній сцені, який спостерігається під час відносного руху між камерою та сценою;
- *текстура*, що описує міру зміни інтенсивності поверхні, що дає інформацію про такі характеристики, як рівність та постійність.

1.5. Використання штучного інтелекту в галузі відеоспостереження

Останнім часом як виробники обладнання для систем відеоспостереження, так і фахівці-інсталятори все частіше звертають увагу на штучний інтелект (ШІ/AI) та Технології Глибинного Навчання (ТГН/DL), обговорюючи їх потенціал у сфері електронної безпеки. І не без причини – ці інноваційні розробки активно використовуються для реалізації різноманітних функцій інтелектуальної відеоаналітики. Завдяки штучному інтелекту системи можуть автоматично розпізнавати обличчя та автомобільні номери, вести відслідковування руху конкретної особи за допомогою камер відеоспостереження та виконувати інші завдання з високою ефективністю [4].

В системах відеоспостереження зазвичай використовуються такий тип штучного інтелекту, що допомагає обробляти великі об’єми даних та вишукує в них певні закономірності.

Більша частина функцій інтелектуальної відеоаналітики в системах спостереження ґрунтується на таких уміннях штучного інтелекту, як:

Машинне навчання: наприклад, функція розпізнавання облич IP-камерами відеоспостереження – чим більше облич потрапляє до бази даних, тим більше людей програма зможе ідентифікувати і знайти, якщо отримає команду на пошук.

Комп’ютерний зір: виявлення, класифікація та стеження за визначеними типами об'єктів – наприклад, камеру спостереження на автомагістралі можна віддалено налаштувати на виявлення автомобілів червоного кольору, що може вказувати на злочин. Таким чином, якщо камера виявить автомобіль червоного кольору, служба охорони отримає тривожне повідомлення.

Глибокий аналіз даних: структурування, аналіз та виділення корисної інформації з невпорядкованого масиву різноманітних потоків інформації.

1.6. Переваги використання штучного інтелекту в системах відеонагляду та відеоспостереження

До переваг використання штучного інтелекту в системах відеонагляду та відеоспостереження належать:

- Підстрахування або часткова заміна людей в службі охорони – у минулому, за епохи аналогового відеоспостереження, жива людина слідкувала за зображенням, отриманим з камери, та швидко реагувала на будь-які неадекватні або злочинні дії. Тепер, завдяки штучному інтелекту, сигнали тривоги автоматично генеруються без участі людини. Штучний інтелект виконує складні функції, такі як розпізнавання забутого або залишеного предмета, виявлення руху шукаючої особи (трекінг) та інші. На відміну від людини, штучний інтелект завжди пильний і не пропустить жодного завдання.
- Багатозадачність – відеокамери зі штучним інтелектом, крім основної функції ведення відеоспостереження, виконують різноманітні завдання відеоаналітики, включаючи збір статистики та контроль пропуску осіб чи транспортних засобів на об'єкт. Деякі моделі відеокамер навіть вимірюють температуру тіла людини на відстані.
- Робота з архівом – одна з найбільш поширених і важливих функцій штучного інтелекту – це робота з відеоархівом. Штучний інтелект ефективно класифікує, впорядковує та швидко знаходить необхідні відеофрагменти. Наприклад, шляхом введення параметрів пошуку, можна миттєво знайти всі відеозаписи з виявленою розшукуваною особою [8].

В цій роботі основна увага буде приділена функції так званого «підстраховування», коли штучний інтелект стає, так би мовити, партнером по роботі.

1.7. Каскад Хаара

В даній роботі для пошуку обличчя використовуються вже навчені моделі, що запропоновані OpenCV, а саме каскади Хаара.

Каскад Хаара – статистичний метод, який використовується для виявлення об'єктів у зображеннях або відео. Головною перевагою є швидкість роботи і висока ефективність виявлення об'єктів. Для використання цього методу спочатку потрібно навчити класифікатор моделі зображенням позитивних і негативних зразків. Позитивні зразки містять зображення об'єкта, який потрібно виявити, тоді як негативні зразки містять зображення без цього об'єкта. На їх основі створюється класифікатор, який може використовуватися для виявлення об'єктів у нових зображеннях або відео [9].

Характеристики Хаара виділяються з прямокутних областей зображення. Значення функції залежить від інтенсивності пікселів. Зазвичай він розраховується за допомогою розсувного вікна та області у вікні, що поділяється на дві або більше прямокутних областей. Характеристика Хаара – це різниця в сумі інтенсивностей пікселів між цими областями.

Вважається, що присутність об’єкта спотворить зміну інтенсивності пікселів. Наприклад, фон зазвичай має однорідний візерунок, у який об’єкт переднього плану не вписується. Перевіряючи інтенсивність пікселів між сусідніми прямокутними областями, ви зможете помітити різницю. Тому це вказує на присутність об’єкта.

Ключова ідея каскаду Хаара полягає в тому, що лише невелика кількість пікселів у всьому зображенні пов’язана з об’єктом, про який йдеться. Тому важливо якомога швидше відкинути невідповідну частину зображення. Під час процесу виявлення каскад Хаара сканує зображення в різних масштабах і місцях, щоб усунути невідповідні області.

2. Макет системи відеонагляду

2.1. Вибір інструментів розробки

2.1.1. Платформа Arduino

Мікроконтролер - це невеликий комп'ютер, який містить центральний процесор (CPU), пам'ять, периферійні пристрої та інші компоненти на одній мікросхемі. Мікроконтролери використовуються в широкому спектрі пристроїв, включаючи промислові контролери, побутову електроніку, медичні пристрої та багато іншого [10].

Arduino (Ардуіно) — апаратна обчислювальна платформа для аматорського конструювання, основними компонентами якої є плата мікроконтролера з елементами вводу/виводу та середовище розробки Processing/Wiring на мові програмування, що є спрощеною підмножиною C/C++ [11].

В якості основи для створення проекту було обрано платформу Arduino завдяки наступним перевагам:

- Низька вартість. Плати Arduino є відносно недорогими, що дозволяє швидко і легко створювати прототипи.
- Простота використання. Плати Arduino прості у використанні, навіть для початківців.
- Широкий спектр можливостей. Плати Arduino підтримують широкий спектр датчиків і компонентів, що дозволяє створювати прототипи для різних цілей.
- Мобільність. Плати Arduino компактні і портативні, що дозволяє легко транспортувати їх і використовувати в різних місцях.

2.1.2. Середовище програмування Arduino IDE

Arduino IDE - це безкоштовне та відкрите середовище розробки (IDE) для програмування плат Arduino. Воно дозволяє розробникам створювати та компілювати програми для плат Arduino за допомогою мови програмування C/C++ [12]. Основні функції Arduino IDE:

- інтегрований редактор коду, який дозволяє розробникам створювати та редагувати програми для плат Arduino;
- компілятор, який компілює програми, написані на мові C/C++, у формат, який може бути завантажений на плату;

  завантажувач, який завантажує програми з комп'ютера на плату Arduino;
- віртуальний симулятор, який дозволяє розробникам тестувати програми для плат Arduino без необхідності підключення плати до комп'ютера.

2.1.3. Мова програмування Python

Python - це універсальна, динамічна мова програмування, що використовується в різноманітних сферах, зокрема:

- Веброзробка: Django, Flask
- Аналіз даних: NumPy, Pandas
- Машинне навчання: TensorFlow, PyTorch
- Системне адміністрування: Ansible, SaltStack
- Наукові розрахунки: SciPy, Matplotlib
- Ігрова розробка: PyGame, Godot

Мову Python було обрано завдяки низці переваг:

- Простота. Легко читати та писати код, що робить його доступним для новачків.
- Швидкість. Python - інтерпретована мова, що робить її швидкою та ефективною.
- Широкий спектр бібліотек. Існує бібліотека для майже будь-якого завдання.
- Безкоштовний та відкритий код. Доступний для всіх безкоштовно [13].

2.1.4. OpenCV: Бібліотека комп'ютерного зору

OpenCV (Open Source Computer Vision Library) - це потужна та універсальна бібліотека для комп'ютерного зору в реальному часі та обробки зображень [14]. Вона широко використовується в різних сферах, зокрема:

Обробка зображень та відео:

- Читання, запис та відображення зображень та відео
- Зміна розміру, кадрування, обертання та фільтрація зображень
- Виявлення та відстеження об'єктів на відео
- Розпізнавання облич
- Аналіз руху

До можливостей розпізнавання об'єктів входять:

- Ідентифікація об'єктів на зображеннях та відео
- Розпізнавання штрих-кодів, QR-кодів та тексту
- Класифікація об'єктів за різними категоріями

В OpenCV вже є вбудовані навчені моделі для розпізнавання рук, облич, посмішок, фігур і тд. У рамках виконання дипломної роботи було використано вже навчену модель з офіційного Git-репозиторію проекту.

2.1.5. PySerial: бібліотека для роботи з послідовними портами

PySerial - це популярна бібліотека Python, яка робить простішим налаштування та використання послідовних портів на комп'ютері [15]. Вона дозволяє програмі написаною мовою Python спілкуватися з пристроями, підключеними до серійного порту, такими як плата Arduino, модеми, інші мікроконтролери тощо.

Основні можливості PySerial:

- Відкриття, закриття та керування послідовними портами.
- Налаштування параметрів порту, таких як швидкість передачі даних, біти даних, стоповий біт і контроль парності.
- Надсилання та отримання даних через послідовний порт у вигляді байтів, рядків або інших типів даних.
- Читання та запис файлів через послідовний порт.
- Використання подій для отримання сповіщень про доступність даних, помилки тощо.

За допомогою бібліотеки PySerial реалізовано обмін даними між моделлю машинного навчання і платою Arduino.

2.2. Апаратна частина макету

Завданням дипломної роботи є розробка макету системи відеотрекінгу. Апаратна частина створена на основі платформи Arduino та її модулів. Загалом система працює в двох режимах, між якими можна перемикатись:

- Ручний режим роботи передбачає управління положенням камери за допомогою джойстика, для пошуку обличчя в тому разі, якщо воно знаходиться поза кадром. Після того, як обличчя буде знайдено, почне світитись червоний індикатор і можна буде переключитись на автоматичний режим.
- В автоматичному режимі система позиціонування буде наводити камеру самостійно таким чином, щоб обличчя лишалося в центрі кадру.

На рисунку 2.1 наведена принципова схема макету.

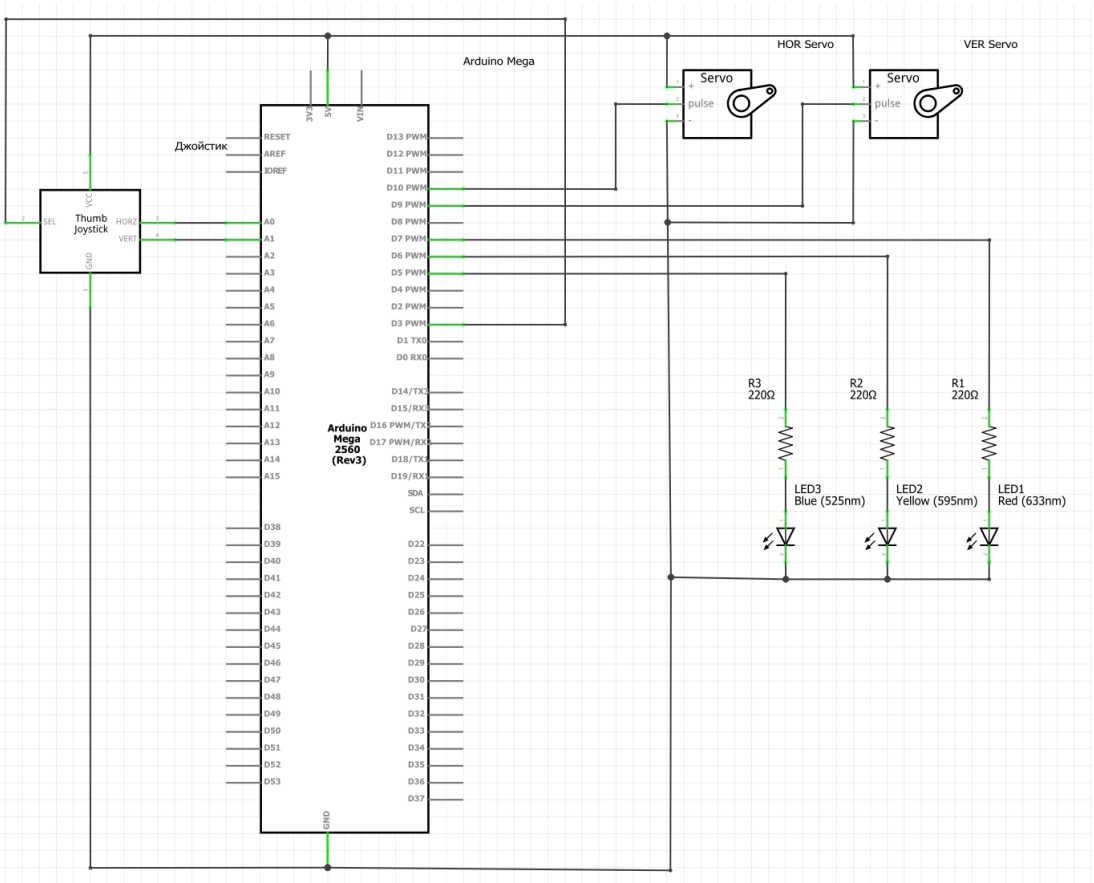

Рисунок 2.1 – Принципова схема макету.

Макет складається з таких модулів:

- плата Arduino Mega в якості основного керуючого компонента (опис плати наведено в розділі 2.3.1.)
- джойстик
- три світлодіоди (LED)
- резистори
- два сервомотори

На рисунку 2.2. наведена фотографія зібраного макету системи.

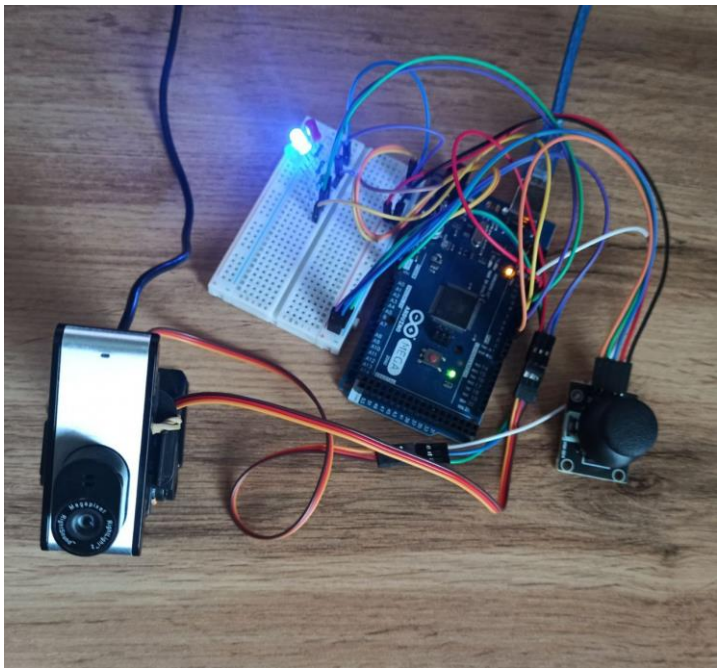

Рисунок 2.2 – Фотографія зібраного макету системи.

В лівій частині фото розташований тримач веб-камери, в який вмонтовані сервомотори. В центральні частині фото розташована макетна плата, що спрощує з’єднування компонентів схеми. Безпосередньо на монтажній платі знаходять світлодіоди, що використовуються для світлової ідентифікації. Справа від монтажної плати розташована плата Arduino Mega 2560 – основна частина схеми. В правій частині фото розташований джойстик – головний елемент управління системою позиціонування камери в ручному режимі роботи.

2.3. Модулі апаратної частини

2.3.1. Плата Arduino Mega

На етапі проектування системи було вирішено використовувати саме плату Arduino Mega 2560, таку, що зображена на рисунку 2.3, оскільки у неї є широкий спектр можливостей для подальшого вдосконалення. Для системи, що була розроблена в ході виконання проекту було б достатньо використати платформи з меншим функціоналом. Наприклад Uno або Nano. Однак вони мають менше можливостей для подальшого удосконалення.

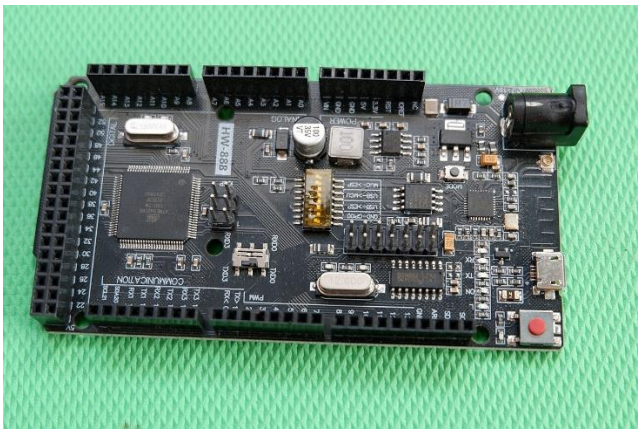

Рисунок 2.3 – Плата Arduino Mega 2560.

Arduino Mega побудована на мікроконтролері ATmega2560. Плата має 54 цифрових входів/виходів, 16 аналогових входів, 4 послідовних порти UART, кварцовий генератор 16 МГц, USB коннектор, роз'єм живлення, роз'єм ICSP і кнопка перезавантаження. Arduino Mega 2560 сумісна з платами розширення, розробленими для платформ Uno або Duemilanove [16].

2.3.2. Джойстик KY-023

Використаний джойстик має 5 виводів: VCC (+ 5V), GND (земля), X (переміщення по осі X, на схемі позначений як HORZ), Y (переміщення по осі Y, на схемі позначений як VERT) і Button (сигналізує чи натиснута кнопка, на схемі позначений як VERT). Загальний вигляд джойстику представлений на рисунку 2.4.

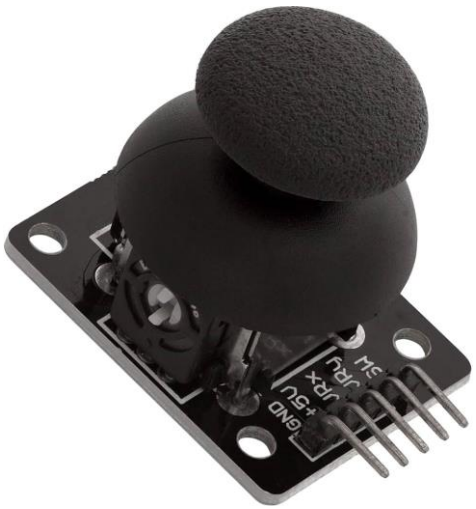

Рисунок 2.4 – Джойстик KY-023.

На виходах X і Y сигнал є пропорційним до кута нахилу ручки. При використанні напруги живлення 5 В, за замовчуванням на аналогових виводах X і Y буде 2.5 В. При переміщенні джойстика в одну сторону напруга буде наростати до 5 В, при русі джойстика в іншу сторону напруга буде падати до 0 В. Виводи підключаються до аналогових входів Arduino. Таким чином можна отримувати точне положення ручки джойстика і реагувати на кут нахилу, а не тільки на сам факт нахилу ручки [17].

2.3.3. Сервомотор SG90

Сервомотори використовуються для позиціонування камери по двох осях. Загальний вигляд SG90 представлено на рисунку 2.5.

Рисунок 2.5 – Сервомотор SG90.

Сервомотор SG90, що був використаний в макеті, може обертатися приблизно на 180º(по 90º у кожному напрямку). Їх можливо використовувати з будь-яким сервокодом, обладнанням або бібліотекою для керування цими сервомоторами.

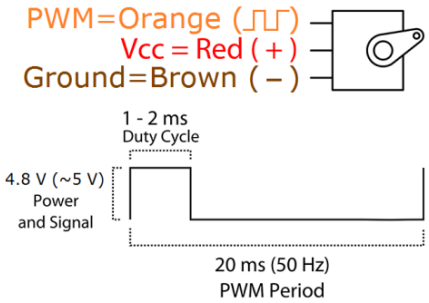

Рисунок 2.6 – Схема підключення сервоприводу SG90 та часова діаграма широтно-імпульсної модуляції.

На рисунку 2.6 показана схема підключення. Цей сервомотор має 3 входи: PWM (підключається до цифрового порту та використовується для управління), VCC (+ 5V), GND (земля). Також наведено графік широтно-імпульсної модуляції. Один робочий цикл складає від 1 до 2 мілісекунд. Положення «0» (імпульс 1,5 мс) це середина, «90» (~2 мс імпульс) повністю праворуч, "-90" (~1 мс імпульс) повністю ліворуч [18].

2.4. Програмний код для Arduino

Повний код програми наведено у ДОДАТКУ А.

2.4.1. Підключення бібліотек. Визначення портів. Функція setup

В першу чергу при написанні програмного коду для Arduino слід підключити всі необхідні бібліотеки. В нашому випадку це лише бібліотека Servo.h що потрібна для управління сервоприводом. Також на початку програми ми визначаємо змінні що будуть в подальшому використовуватися. Ці операції описані в коді на рисунку 2.7.

```
#include <Servo.h>              // Підключення бібліотеки для сервоприводу

#define LED_PIN_Y 5             // Номер порта керування жовтим світлодіодом
#define LED_PIN_B 6             // Номер порта керування синім світлодіодом
#define LED_PIN_R 7             // Номер порта керування червоним світлодіодом

#define JOY_BUTTON 3            // Номер порта для підключення кнопки джойстика
#define JOY_X 0                 // Номер порта для підключення осі X джойстика
#define JOY_Y 1                 // Номер порта для підключення осі Y джойстика

#define SERVO_PIN_X 10          // Номер порта для підключення горизонтального сервоприводу
#define SERVO_PIN_Y 9           // Номер порта для підключення вертикального сервоприводу

Servo servoVer;                 // Вертикальний сервопривід
Servo servoHor;                 // Горизонтальний сервопривід

int width = 640, height = 480;  // Загальне розширення відео
int x_pos = 90, y_pos = 70;     // Початкові положення сервоприводів
int x_com, y_com;               // Значення координат, що надходять з послідовного порту
```

Рисунок 2.7 – Код для підключення бібліотеки, визначення номерів портів для підключення модулів та змінних основних параметрів.

В функцію setup заносяться попередні встановлення для програми. Наприклад, приєднання сервоприводів, визначення режиму роботи портів і т. д. Цей блок програми виконується один раз на самому початку роботи. Дії в цьому блоці описані на рисунку 2.8.

```
void setup() {
  Serial.begin(9600);
  servoHor.attach(SERVO_PIN_X);   // "Приєднання" горизонтального сервоприводу до порту
  servoVer.attach(SERVO_PIN_Y);   // "Приєднання" вертикального сервоприводу до порту
  servoHor.write(x_pos);          // Записування X координати в сервопривід
  servoVer.write(y_pos);          // Записування Y координати в сервопривід
  pinMode(LED_PIN_Y, OUTPUT);     // Визначення режиму роботи цифрового порту світлодіода на вихід
  pinMode(LED_PIN_B, OUTPUT);
  pinMode(LED_PIN_R, OUTPUT);
  pinMode(JOY_BUTTON, INPUT);     // Визначення режиму роботи цифрового порту кнопки на вхід
  digitalWrite(JOY_BUTTON, HIGH); // Подача на порт кнопки "високого" сигналу
}
```

Рисунок 2.8 – Setup блок програми.

2.4.2. Функція loop. Програмування кнопки-перемикача

В функцію loop заносяться дії, які мікроконтролер буде виконувати після попередніх установок. Все, що занесене в цей блок, буде виконуватися циклічно. Таким чином, можна визначити перелік дій мікроконтролера, які потрібно виконувати протягом усього часу його роботи. Наприклад, це може бути управління джойстиком, реагування на натискання кнопок і т. д.

```
void loop() {
  int button_reading = digitalRead(JOY_BUTTON);      // Зчитування стану кнопки
  if (button_reading != last_button_state) {         // Дебаунсинг
    last_debounce_time = millis();
  }
  if ((millis() - last_debounce_time) > debounce_delay) {
    if (button_reading != cur_button_state) {        // Зміна стану на протилежний
      cur_button_state = button_reading;
      if (cur_button_state == HIGH) {
        program_mode = (program_mode + 1) % 2;       // Визначення режиму роботи
        if (program_mode == 1) {                     // У разі, якщо режим роботи
          x_pos = servoHor.read();                   // автоматичний, то ми зчитуємо
          y_pos = servoVer.read();                   // попередні значення на серво
        }
      }
    }
  }
  last_button_state = button_reading;                // оновлення стану
```

Рисунок 2.9 – Код для програмування кнопки – перемикача.

На рисунку 2.9 наведена частина коду, що відповідає за роботу кнопки в якості перемикача. Зчитується стан кнопки, потім визначається час останнього змінення стану (так званий дебаунсинг) для уникнення “брязкоту”, далі безпосередньо зміна режиму програми і оновлення робочого стану. Отже, цей код забезпечує ефективний спосіб визначення натискань кнопки з врахуванням можливого "брязкоту" при фізичному контакті кнопки. Якщо кнопка натиснута, то програма змінює режим між 0 і 1 при кожному натисканні кнопки.

Також слід зауважити, що при перемиканні в автоматичний режим роботи ми маємо зберегти попередній стан сервоприводів, щоб після перемикання вони не повертались в початкове положення, а лишились в попередньому положенні.

2.4.3. Запис координат з послідовного порту

Код, наведений на рисунку 2.10 виконує запис даних, що надходять з послідовного порту у відповідні змінні.

```
if (Serial.available() > 0) {      // Якщо послідовний порт відкрився
  if (Serial.read() == 'X') {      // Якщо в послідовному потоці зустрічаємо X
    x_com = Serial.parseInt();     // То записуємо наступне ціле числове значення в змінну x_com
    if (Serial.read() == 'Y')      // Якщо в послідовному потоці зустрічаємо Y
      y_com = Serial.parseInt();   // То записуємо наступне ціле числове значення в змінну Y_com
  }
}
```

Рисунок 2.10 – Код для зчитування даних з послідовного порту.

Запис даних відбувається в три етапи:

1. Перевірка наявності даних у послідовному порті за допомогою методу Serial.available().
2. Якщо дані доступні, перевіряється перший символ, що надійшов від послідовного порту. Якщо це символ 'X', програма читає наступне ціле число, вважаючи його значенням змінної x_com.
3. Після цього перевіряється, чи наступний символ у послідовному потоці - 'Y'. Якщо так, програма читає наступне ціле число, вважаючи його значенням змінної y_com.

Отже, цей код призначений для отримання даних з послідовного порту. Якщо вхідні дані мають формат "X<number>Y<number>", то він зчитує числа після 'X' та 'Y' і зберігає їх у відповідних змінних.

2.4.4. Увімкнення індикації знаходження обличчя в кадрі

Програмою Python, що буде розглянута пізніше, передбачено, що розмір екрану становитиме 640×480 пікселів. Відповідно більшими за ці значення координати бути не можуть. Відповідно в тому разі, якщо обличчя не було знайдено, надсилаються дані X999Y999, щоб формат даних, що надсилаються був однаковим. З одного боку, вийшовши за рамки кадру, ці дані не будуть порушувати роботу програми, з іншого, формат лишається таким самим, що не порушить процес розшифрування даних, що надходять з послідовного порту. Код для цієї обробки наведено на рисунку 2.11.

```
if (Serial.available() > 0) {           // Якщо послідовний порт відкрився
  if (x_com == 999 && y_com == 999) {   // Якщо отримано повідомлення "X999Y999"
    digitalWrite(LED_PIN_R, LOW);       // То загасити червоний світлодіод
  } else {
    digitalWrite(LED_PIN_R, HIGH);      // В іншому разі - увімкнути
  }
}
```

Рисунок 2.11 Код для індикації знайдення обличчя в кадрі

Отже, за умови, що дані надходять з послідовного порту, при отриманні повідомлення «X999Y999» за допомогою методу digitalWrite ми надсилаємо сигнал низького рівня (логічний 0) на порт, до якого під’єднано червоний світлодіод, тобто, ми його гасимо. В іншому випадку, коли ми отримуємо будь-які інші значення (програмою Python передбачено, що це значення в районі розширення екрану, тобто 0<x<640 та 0<y<450, усі інші значення можна використати в якості кодів для іншої індикації), то цей світлодіод почне світитись.

2.4.5. Ручний режим роботи макету

Код, наведений на рисунку 2.12 виконує управління позицією камери в ручному режимі.

```
if (program_mode == 0) {                                        // Якщо це ручний режим роботи
  digitalWrite(LED_PIN_B, HIGH);                                // Вмикаємо синій світлодіод
  digitalWrite(LED_PIN_Y, LOW);                                 // Вимикаємо жовтий світлодіод

  float easing_speed = 0.05;                                    // Визначаємо швидкість згладжування

  int target_servo_x = analogRead(JOY_X);                       // Зчитуємо значення X з джойстика
  target_servo_x = map(target_servo_x, 0, 1023, 0, 180);        // Перетворюємо аналогове зачення в кут повороту

  int target_servo_y = analogRead(JOY_Y);                       // Зчитуємо значення Y з джойстика
  target_servo_y = map(target_servo_y, 0, 1023, 25, 115);       // Перетворюємо аналогове зачення в кут повороту

  float prev_servo_x = servoHor.read();                         // Зчитуємо попереднє положення серво X
  float prev_servo_y = servoVer.read();                         // Зчитуємо попереднє положення серво Y

  servoHor.write(lerp(prev_servo_x, target_servo_x, easing_speed)); // Застосовуємо лінійну інтерполяцію для плавного
  servoVer.write(lerp(prev_servo_y, target_servo_y, easing_speed)); // руху по осям X та Y відповідно

  delay(10);                                                    // Для того, щоб контролер встиг зреагувати, ставимо затримку
}
```

Рисунок 2.12 – Код для управління позицією камери за допомогою джойстика.

Таким чином ручне управління відбувається за таким алгоритмом:

1. Якщо змінна program_mode дорівнює 0, що вказує на ручний режим роботи.
2. Увімкнення синього світлодіода (LED_PIN_B) і вимкнення жовтого світлодіода (LED_PIN_Y).
3. Визначається змінна easing_speed, яка встановлює швидкість згладжування руху сервоприводів.
4. Зчитуються значення джойстика по осях X і Y (JOY_X і JOY_Y) за допомогою функції analogRead.
5. Значення зчитаних аналогових сигналів масштабуються в діапазоні від 0 до 180 (для кута повороту сервопривода) за допомогою функції map.
6. Зчитуються поточні положення сервоприводів по осях X і Y за допомогою методів servoHor.read() і servoVer.read().
7. Застосовується лінійна інтерполяція (функція lerp) для згладження руху від поточного положення до цільового за швидкістю easing_speed.
8. Встановлюється затримка на 10 мілісекунд для того, щоб мікроконтролер встиг реагувати на зміни.

2.4.6. Автоматичний режим роботи макету

Код, наведений на рисунку 2.13 виконує управління позицією камери в автоматичному режимі.

```
if (program_mode == 1) {                                    // Якщо це автоматичний режим роботи
  digitalWrite(LED_PIN_B, LOW);                             // Вимикаємо синій світлодіод
  digitalWrite(LED_PIN_Y, HIGH);                            // Вмикаємо жовтий світлодіод
  if (Serial.available() > 0 && x_com != 999 && y_com != 999) {   // За умови, що на послідовному порті не "X999Y999"
    if (x_com > width / 2 - 30)                             // У разі відхилення від центру праоруч на 30 пікселів
      x_pos += angle;                                       // Збільшуємо кут повороту по горзоналі на визначену к-ть градусів
    if (x_com < width / 2 + 30)                             // Аналогічно для всіх інших відхилень
      x_pos -= angle;
    if (y_com < height / 2 - 30)
      y_pos -= angle;
    if (y_com > height / 2 + 30)
      y_pos += angle;
    x_pos = constrain(x_pos, 0, 180);                       // Обмежуємо значеня повороту по X в межах від 0 до 180 градусів
    y_pos = constrain(y_pos, 0, 180);                       // Обмежуємо значеня повороту по Y в межах від 0 до 180 градусів
    servoHor.write(x_pos);                                  // Надсилаємо значення x_pos на горзонтальний сервопривід
    servoVer.write(y_pos);                                  // Надсилаємо значення y_pos на вертикальний сервопривід
  }
}
```

Рисунок 2.13 – Код для управління позицією камери в автоматичному режимі.

Таким чином автоматичне управління відбувається за таким алгоритмом:

1. Перевірка, чи змінна program_mode дорівнює 1, що вказує на автоматичний режим роботи.
2. Вимкнення синього світлодіода (LED_PIN_B) і увімкнення жовтого світлодіода (LED_PIN_Y).
3. Перевірка наявності даних у послідовному порті (Serial) та значення-маркеру для відсутності даних.
4. В залежності від значень x_com і y_com, якщо вони більше або менше від певних порогів (ширина/висота екрану поділена на 2, з врахуванням відхилення на 30 пікселів), збільшує або зменшує значення x_pos і y_pos на певну кількість градусів (angle).
5. Обмеження значення x_pos і y_pos в межах від 0 до 180 градусів за допомогою функції constrain.
6. Запис значення x_pos і y_pos у відповідні сервоприводи за допомогою методів servoHor.write() і servoVer.write().

Отже, цей фрагмент коду відповідає за автоматичне управління рухом сервоприводів на основі отриманих координат x_com і y_com через послідовний порт.

2.4.7. Лінійна інтерполяція координат

Оскільки інформація з джойстика зчитується практично миттєво, то відповідно так само швидко спрацьовують сервоприводи, якщо відправляти інформацію напряму лише за допомогою масштабування аналогового значення в кут повороту. Це миттєве спрацювання не є практичним при ручному управлінні, оскільки управління за допомогою джойстика передбачає деякі похибки. Тому є сенс застосувати лінійну інтерполяцію для більш плавного спрацювання сервоприводів.

```
float lerp(float start, float end, float t) {
  return start + (end - start) * t;
}
```

Рисунок 2.14 – Функція лінійної інтерполяції.

Лінійна інтерполяція використовується для плавного переміщення сервоприводів з їх поточного положення до нового, визначеного значення.

```
servoHor.write(lerp(prev_servo_x, target_servo_x, easing_speed));
servoVer.write(lerp(prev_servo_y, target_servo_y, easing_speed));
```

Рисунок 2.15 – Застосування функції лінійної інтерполяції.

Спочатку вимірюється поточне положення сервоприводів (prev_servo_x і prev_servo_y). Зчитуються нові значення кута повороту для сервоприводів (target_servo_x і target_servo_y), які були перетворені з аналогових значень з джойстика за допомогою функції map().

Функція lerp() виконує лінійну інтерполяцію між поточним та новим значеннями сервоприводів. Ця функція використовує згладжування (easing_speed), що визначає швидкість зміни значень. Нові значення, отримані в результаті лінійної інтерполяції, записуються на сервоприводи за допомогою servoHor.write() і servoVer.write().

2.5. Програмний код для визначення координат обличчя

Мовою Python з використанням бібліотеки OpenCV написаний програмний код, що відповідає за визначення координат обличчя у кадрі і відправлення цих координат на плату Arduino для їх подальшої обробки. Відправлення цих даних забезпечують методи бібліотеки PySerial.

Приклад визначення обличчя на відео показаний на рисунку 2.16.

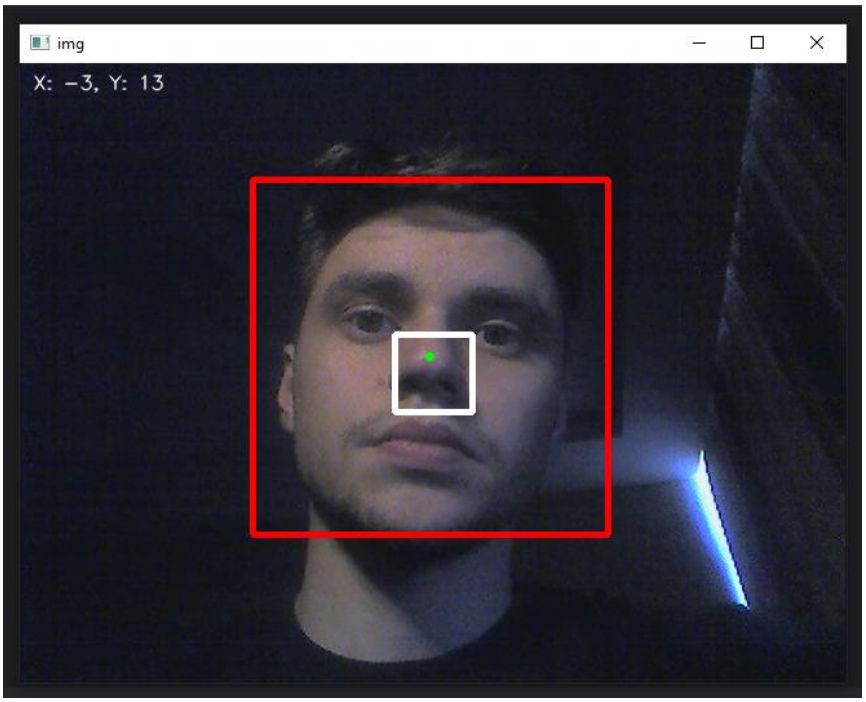

Рисунок 2.16 – Визначення положення обличчя у відеопотоці.

Повний код програми наведено у ДОДАТКУ Б.

2.5.1. Підключення бібліотек та початок роботи

На початку роботи потрібно підключити всі необхідні бібліотеки та модулі, а також виконати стартові налаштування. Налаштування показані на рисунку 2.17.

```
import cv2
import serial
import time

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')

cap = cv2.VideoCapture(0)

ArduinoSerial = serial.Serial( port: 'com1',  baudrate: 9600, timeout=0.1)

time.sleep(1)
```

Рисунок 2.17 – Код для підключення бібліотек і стартових налаштувань.

Підключення модулів відбувається за допомогою import. В нашому випадку ми використовуємо такі модулі:

1. Модуль cv2 – частина бібліотеки OpenCV, що в даному проекті використовується для роботи із відео
2. Модуль serial - потрібен для передачі потокової інформації на плату Arduino за допомогою послідовного порту
3. Модуль time – потрібен для встановлення пауз

Далі відбувається завантаження класифікатору Haar Cascade для виявлення облич у змінну face_cascade.

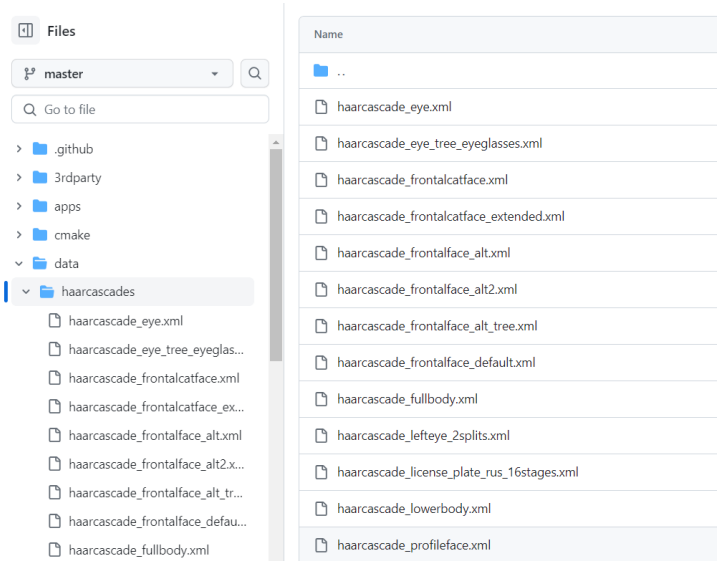

Рисунок 2.18 – Доступні класифікатори з GitHub сторінки OpenCV.

На офіційній сторінці OpenCV на GitHub, знімок якої наведено на рисунку 2.18, є класифікатори для визначення облич, рук, долонь, очей, посмішок і т. д. Тому такий класифікатор було завантажено звідти. Слід зазначити, що є можливість протестувати кілька каскадів під час роботи над проектом. У випадку цієї дипломної роботи було обрано haarcascade_frontalface_alt.xml, оскільки він показав себе найкраще в умовах нерівномірного освітлення. Таким чином, підключаючи будь-який з наявних класифікаторів, існує можливість вибору, за чим саме камера буде слідкувати і по чому саме вона буде центруватись. В ході виконання роботи, під час тестування в умовах обмеженого простору камера відслідковувала долоню, з чим установка справилась.

Повернімося до коду. Далі відкривається відеопотік з вебкамери за допомогою cv2.VideoCapture(0). В аргументи методу передається індекс відеокамери.

Далі за допомогою метода Serial встановлюється з'єднання з Arduino через порт COM1 з швидкістю передачі 9600 біт за секунду. З його допомогою на Arduino будуть передаватись координати обличчя.

І в кінці очікуємо одну секунду для стабілізації роботи. Це відбувається за допомогою time.sleep(1).

2.5.2. Виявлення обличчя у відеопотоці

Після закінчення початкових налаштувань потрібно виявити обличчя в кадрі. На рисунку 2.19 наведено код для обробки кадрів відеопотоку.

```
while cap.isOpened():
    ret, frame = cap.read()
    frame = cv2.flip(frame, flipCode: 1)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = face_cascade.detectMultiScale(gray, scaleFactor: 1.1, minNeighbors: 6)
```

Рисунок 2.19 – Код для обробки кадрів відеопотоку.

Обробка відбувається і циклі while cap.isOpened(), тобто поки відеопотік відкритий і готовий до читання.

Метод cap.read() читає кадр з відеопотоку. Він повертає два значення: ret (булеве значення), що показує, чи був успішно прочитаний кадр, та frame, який представляє сам кадр.

Рядок frame = cv2.flip(frame, 1) здійснює відображення кадру по горизонталі, тобто віддзеркалює його відносно вертикальної осі.

Рядок gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) перетворює кадр у відтінки сірого кольору. Деякі алгоритми обробки обличчя працюють з кадрами в градаціях сірого кольору, тому потрібно перетворити кадр з формату кольору BGR (або RGB) у сірий колір.

Рядок faces = face_cascade.detectMultiScale(gray, 1.1, 6) використовує класифікатор Haar Cascade для виявлення облич у кадрі. face_cascade - це об'єкт класифікатора, який був завантажений на початку програми. Метод detectMultiScale застосовує класифікатор до сірого кадру з певними параметрами, такими як масштаб фото та мінімальна кількість сусідів, які повинні бути знайдені для визначення обличчя. В результаті отримуємо список прямокутників, які відображають позиції знайдених облич на кадрі.

Параметр scaleFactor визначає наскільки зменшується розмір зображення під час кожної ітерації сканування. Наприклад, якщо scaleFactor = 1.1, то зображення буде зменшуватися на 10% кожного разу. Це допомагає виявляти об'єкти різних розмірів на зображенні

Кількість сусідів – це параметр, який визначає мінімальну кількість сусідніх зон, які повинні бути знайдені під час виявлення обличчя. Це допомагає уникнути помилкового виявлення і допомагає збільшити точність виявлення.

Коли класифікатор виявляє обличчя, він аналізує різні області на зображенні і перевіряє їх на відповідність шаблону обличчя. Потім він об'єднує ці області у групи, які мають достатньо високу відповідність. Кількість сусідів визначається пороговим значенням для збирання цих областей у групи.

Більша кількість сусідів означає більш жорсткий поріг, що може призвести до меншої кількості помилкових виявлень, але може також призвести до пропуску деяких облич, якщо вони не відповідають строго визначеному шаблону. Тому цей параметр слід підбирати для окремих випадків.

2.5.3. Кодування відсутності обличчя

В розділі 2.4.4. був описаний алгоритм увімкнення та вимкнення індикації при знаходженні обличчя. За кодування відсутності обличчя відповідає код, що наведений на рисунку 2.20.

```
if len(faces) == 0:
    string = 'X999Y999'
    print(string)
```

Рисунок 2.20 – Код для визначення, що обличчя в потоці відсутнє.

Цей код перевіряє, чи було знайдено обличчя на зображенні. Умова len(faces) == 0 перевіряє, чи список faces, який містить координати обличчя на зображенні, порожній. Якщо умова виконується (тобто не було знайдено обличчя), то виводиться рядок 'X999Y999'.

Цей рядок представляє координати обличчя у форматі, що використовується для відправлення через послідовний порт Arduino. Значення 999 вказує на те, що обличчя не було знайдено. Таким чином, якщо на зображенні не виявлено обличчя, Arduino отримає рядок 'X999Y999', що інформує його про відсутність обличчя на зображенні.

2.5.4. Визначення і відправка координат обличчя

Коли обличчя знайдено у відеопотоці, то потрібно визначити його координати та відправити по послідовному порту. Обрахунок координат центру обличчя і відправлення відповідного повідомлення описано на рисунку 2.21.

```
for (x, y, w, h) in faces:
    string = 'X{0:d}Y{1:d}'.format( *args: (x + w // 2), (y + h // 2))
    print(string)
    ArduinoSerial.write(string.encode('utf-8'))
```

Рисунок 2.21 – Код для визначення і відправки координат.

Цей фрагмент коду використовує координати та розміри знайденого обличчя для створення рядка string, який містить координати центру обличчя у форматі 'X{координата X}Y{координата Y}'. Використовуються такі параметри:

Параметри x(координата верхнього лівого кута області обличчя по осі X), y(координата верхнього лівого кута області обличчя по осі Y), w(ширина області обличчя) та h(висота області обличчя) дозволяють точно визначити межі та положення обличчя на зображенні, щоб можна було взяти координати центру обличчя або виконати будь-які інші дії, пов'язані з обробкою обличчя.

Після створення рядка координат string він надсилається через послідовний порт за допомогою об'єкта ArduinoSerial. Кожен рядок конвертується в байтовий об'єкт за допомогою кодування utf-8 за допомогою методу encode('utf-8').

Також, виводиться рядок string для візуального відстеження координат, які відправляються до Arduino.

```
cv2.circle(frame, center: (x + w // 2, y + h // 2), radius: 2, color: (0, 255, 0), thickness: 2)
cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 3)

cv2.putText(frame, text: f"X: {x + w // 2 - 640 // 2}, Y: {480 // 2 - y - h // 2}", org: (10, 20),
            cv2.FONT_HERSHEY_SIMPLEX,
            fontScale: 0.5, color: (255, 255, 255), thickness: 1, cv2.LINE_AA)
```

Рисунок 2.22 – Код для візуалізації знаходження обличчя.

Окрім того відбувається візуалізація знайденого в кадрі обличчя. Код наведено на рисунку 2.22.

Команда cv2.circle() створює точку зеленого кольору в області центру обличчя. Параметри команди:

- frame: зображення, на якому буде намальований коло.
- (x + w // 2, y + h // 2): координати центра обличчя.
- 2: радіус кола.
- (0, 255, 0): колір кола (у форматі BGR, тут це зелений).
- 2: товщина контуру кола.

Команда cv2.rectangle() створює прямокутник навколо області обличчя. Параметри команди:

- frame: зображення, на якому буде намальований прямокутник.
- (x, y): верхній лівий кут прямокутника.
- (x + w, y + h): нижній правий кут прямокутника.
- (0, 0, 255): колір прямокутника (у форматі BGR, тут це червоний).
- 3: товщина контуру прямокутника.

Команда cv2.putText() додає текст з координатами центру обличчя на зображення. Параметри команди:

- frame: зображення, на якому буде виведено текст.
- f"X: {x + w // 2 - 640 // 2}, Y: {480 // 2 - y - h // 2}": рядок тексту.
- (10, 20): координати верхнього лівого кута тексту.
- cv2.FONT_HERSHEY_SIMPLEX: шрифт тексту.
- 0.5: розмір шрифту.
- (255, 255, 255): колір тексту (у форматі BGR, тут це білий).
- 1: товщина лінійки тексту.
- cv2.LINE_AA: тип лінійки тексту.

2.5.5. Виокремлення області інтересу

Також на зображенні є додатковий інтерфейс для калібрування установки. Код для відображення цього інтерфейсу наведено на рисунку 2.23.

```
cv2.rectangle(frame, (640 // 2 - 30, 480 // 2 - 30),
              (640 // 2 + 30, 480 // 2 + 30),
              (255, 255, 255), 3)
```

Рисунок 2.23 – Код для відображення додаткового прямокутника.

Команда cv2.rectangle() створює додатковий прямокутник у центрі зображення для виділення так званої «зони інтересу». Параметри команди:

- frame: зображення, на якому буде намальований прямокутник.
- (640 // 2 - 30, 480 // 2 - 30): координати верхнього лівого кута прямокутника. Тут (640 // 2 - 30, 480 // 2 - 30) визначає верхній лівий кут прямокутника, зміщений на 30 пікселів вліво та 30 пікселів вгору від центра зображення.
- (640 // 2 + 30, 480 // 2 + 30): координати нижнього правого кута прямокутника. Тут (640 // 2 + 30, 480 // 2 + 30) визначає нижній правий кут прямокутника, зміщений на 30 пікселів вправо та 30 пікселів вниз від центра зображення. В коді Arduino, що наведений в розділі 2.4.6. було описано, що сервоприводи реагуватимуть на відхилення обличчя від центру на 30 пікселів. Власне, спрацювання перевіряється виходом точки центру обличчя за межі цього прямокутника.
- (255, 255, 255): колір прямокутника (у форматі BGR, тут це білий).
- 3: товщина контуру прямокутника.

2.5.6. Оновлення кадрів

Після всіх маніпуляцій з кадром відеопотоку його потрібно оновити та відобразити. Це відбувається за допомогою команди на рисунку 2.24.

```
cv2.imshow( winname: 'img', frame)
```

Рисунок 2.24 – Код для відображення оновленого зображення.

Команда cv2.imshow('img', frame) відображає оновлене зображення з рамкою навколо обличчя, точкою центру обличчя, текстом координат обличчя відносно центру кадру), а також додатковий прямокутник «зони інтересу» (рисунок 2.24). Параметри команди:

- 'img': назва вікна, у якому буде відображатися зображення.
- frame: зображення, яке буде відображене.

Це дозволяє візуалізувати результати аналізу обличчя та області інтересу у відеопотоці.

2.5.7. Завершення роботи програми

Оскільки обробка зображень відбувається у нескінченному циклі, то потрібно визначити дію для виходу з цього циклу. Такі дії наведені на рисунку 2.25.

```
k = cv2.waitKey(30) & 0xff
if k == 27:
    break
```

Рисунок 2.25 – Код для виходу із циклу.

Цей фрагмент коду відповідає за очікування натискання клавіші протягом 30 мілісекунд після кожної ітерації циклу.

Метод cv2.waitKey(30) очікує натискання клавіші від користувача протягом 30 мілісекунд. Він призначений для переривання виконання програми, поки користувач не натисне будь-яку клавішу або не мине вказаний інтервал часу.

Операція & 0xff виконується для того, щоб отримати нижні 8 біт зі значення, яке повертає cv2.waitKey(). Це необхідно для того, щоб гарантувати, що значення клавіші не перевищує 255.

Після отримання значення натиснутої клавіші, програма перевіряє, чи це значення відповідає коду клавіші ESC (код 27). Це забезпечує можливість вийти з циклу та завершити програму, якщо користувач натиснув клавішу ESC.

Якщо умова if виконується, тобто користувач натиснув клавішу ESC, виконання циклу припиняється за допомогою команди break, і виконання програми переходить до наступного рядка після циклу.

Після завершення циклу обробки зображень потрібно звільнити ресурси та зачинити всі вікна відеопотоків. Для цього використовуються команди, що наведені на рисунку 2.26.

```
cap.release()
cv2.destroyAllWindows()
```

Рисунок 2.26 – Код для коректного завершення роботи програми.

Ці дві команди виконуються після завершення роботи з відеопотоком і вікном відображення зображення.

Команда cap.release() відключає або звільняє захоплення відеопотоку або камери. Вона звільняє ресурси, які були зайняті для захоплення відео, що дозволяє іншим програмам або процесам використовувати цю камеру. Після виклику цієї команди подальший доступ до камери буде неможливим, поки її знову не буде ініціалізовано.

Команда cv2.destroyAllWindows() закриває всі вікна, які були створені під час виконання програми з використанням бібліотеки OpenCV. Вона прибирає вікна зображення, які відкривалися для відображення поточного відео або зображення. Після виклику цієї команди всі вікна будуть закриті, і користувач повернеться до робочого середовища.

Виконана робота

У ході виконання дипломної роботи була розроблена система відеонагляду на основі плати Arduino та її модулів, а також програми мовою Python для відстеження зміни положення обличчя. Система працює в двох режимах: за допомогою джойстика та автоматично.

Система працює коректно в обох режимах.

- В режимі управління джойстиком користувач може керувати положенням камери за допомогою джойстика, який підключений до мікроконтролера.
- В автоматичному режимі програма з використанням комп'ютерного зору самостійно шукає обличчя та наводить на нього камеру.
- Також була додана система світлової індикації для перевірки перемикання схеми між режимами роботи, а також відстеження обличчя.

Розроблена система може бути використана в якості основи для інших проектів для роботи з комп’ютерним зором та автоматичним відеотрекінгом. А також поглибленого дослідження цієї технології. В роботі наведений детальний опис створення такого роду системи. Універсальність, а також зростаюча розповсюдженість і відповідно зацікавленість в такого роду системах робить дану роботу актуальною.

Розроблена в ході виконання дипломної роботи система має ряд переваг, таких як:

- Простота розробки та експлуатації.
- Низька вартість.
- Широкий спектр можливостей для подальшого вдосконалення.

Подальше вдосконалення системи може передбачати покращення системи позиціонування, а також впровадження інших моделей для відслідковування об’єктів.

Висновки

В ході виконання дипломної роботи було встановлено, що для більшої плавності роботи сервоприводів в макетах з використанням комп’ютерного зору краще використовувати лінійну інтерполяцію, а не лише масштабування значень отриманих з джойстика та подальший їх запис за допомогою функції write, оскільки останній спосіб спрацьовує дуже різко, що призводить до втрати стабілізації та спотворення зображень.

Зазначено, що при обміні даними через послідовний порт формат представлення має залишатись незмінним, інакше це може призвести до некоректної роботи схеми. Для кодування особливих випадків у роботі запропоновано використання значень координат, що виходять за межі вікна. Тобто службові повідомлення також матимуть формат координат, що робить декодування даних більш стійким.

Зазначено, що для тестування роботи систем оптичного трекінгу в кадр відеопотоку доцільно додавати так звану «область інтересу». Таким чином можна дослідити точність спрацювання системи для подальшого її калібрування.

Список літературних джерел

1. Історія відеоспостереження [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: Video Surveillance History. – 2023. – Retrieved from: https://ipvm.com/reports/history-video-surveillance
2. Artificial Intelligence Systems for Supporting Video Surveillance Operators at International Airport / [M. Sujkowski, J. Kozuba, P. Uchroński та ін.]. // Transportation Research Procedia. – 2023. – №74. – С. 1284–1291
3. Pawar K. Deep learning based detection and localization of road accidents from traffic surveillance videos / K. Pawar, V. Attar. // ICT Express. – 2022. – №8. – С. 379–387
4. Video surveillance using deep transfer learning and deep domain adaptation: Towards better generalization / [Y. Himeur, S. Al-Maadeed, H. Kheddar та ін.]. // Engineering Applications of Artificial Intelligence. – 2023. – №119
5. Machine Learning, ML [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://www.it.ua/knowledge-base/technology-innovation/machine-learning
6. Object Tracking Methods: A Review / [Z. Soleimanitaleb, M. A. Keyvanrad, A. Jafari] // 2019 9th International Conference on Computer and Knowledge Engineering (ICCKE). – 2019.
7. A real-time object detection algorithm for video / [L. Shengyu, W. Beizhan, W. Hongji та ін.]. // Computers & Electrical Engineering. – 2019. – №77. – С. 398–408.
8. Використання функцій штучного інтелекту (AI) в галузі відеоспостереження [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://tvtdigital.com.ua/vykorystannia-funktsiy-shtuchnoho-intelektu-ai-v-haluzi-videosposterezhennia/
9. Tam A. Using Haar Cascade for Object Detection [Електронний ресурс] / Adrian Tam // Machine Learning Mastery. – 2024. – Режим доступу до ресурсу: https://machinelearningmastery.com/using-haar-cascade-for-object-detection/.

10. Microcontroller. Матеріал з Вікіпедії – вільної енциклопедії. [Електронний ресурс]. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Microcontroller
11. Arduino. Матеріал з Вікіпедії – вільної енциклопедії. [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Arduino
12. Arduino IDE. Матеріал з Вікіпедії – вільної енциклопедії. [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Arduino_IDE
13. About Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/about/
14. About OpenCV [Електронний ресурс] – Режим доступу до ресурсу: https://opencv.org/about/
15. PySerial Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://pythonhosted.org/pyserial/
16. Mega 2560 Rev3 [Електронний ресурс] // Arduino Official Site – Режим доступу до ресурсу: https://docs.arduino.cc/hardware/mega-2560/
17. KY-023 Joystick module (XY-Axis) [Електронний ресурс] // Joy-IT. – 2017. – Режим доступу до ресурсу: https://datasheetspdf.com/mobile/1402034/Joy-IT/KY-023/
18. SG90 Micro Servo [Електронний ресурс] // Tower Pro. – 2014. – Режим доступу до ресурсу: https://datasheetspdf.com/mobile/791970/TowerPro/SG90/

ДОДАТОК А

Програма для Arduino для управління системою позиціонування, перемикання між режимами роботи програм та перемикання світлової індикації.

```
#include <Servo.h>
#define LED_PIN_Y 5
#define LED_PIN_B 6
#define LED_PIN_R 7
#define JOY_BUTTON 3
#define JOY_X 0
#define JOY_Y 1
#define SERVO_PIN_X 10
#define SERVO_PIN_Y 9

Servo servoVer;
Servo servoHor;

int width = 640, height = 480;
int x_pos = 90, y_pos = 70;
int x_com, y_com;

int cur_button_state = HIGH;
int last_button_state = HIGH;
unsigned long last_debounce_time = 0;
unsigned long debounce_delay = 50;
int program_mode = 0;
const int angle = 1;

void setup() {
  Serial.begin(9600);
  servoHor.attach(SERVO_PIN_X);
  servoVer.attach(SERVO_PIN_Y);
  servoHor.write(x_pos);
  servoVer.write(y_pos);
  pinMode(LED_PIN_Y, OUTPUT);
  pinMode(LED_PIN_B, OUTPUT);
  pinMode(LED_PIN_R, OUTPUT);
  pinMode(2, INPUT);
  digitalWrite(JOY_BUTTON, HIGH);
}

void loop() {
  int button_reading = digitalRead(JOY_BUTTON);
  if (button_reading != last_button_state) {
    last_debounce_time = millis();
  }
  if ((millis() - last_debounce_time) > debounce_delay) {
```

```
  if (button_reading != cur_button_state) {
    cur_button_state = button_reading;
    if (cur_button_state == HIGH) {
      program_mode = (program_mode + 1) % 2;
      if (program_mode == 1) {
        x_pos = servoHor.read();
        y_pos = servoVer.read();
      }
    }
  }
}
last_button_state = button_reading;

if (Serial.available() > 0) {

  if (Serial.read() == 'X') {
    x_com = Serial.parseInt();m
    if (Serial.read() == 'Y')
      y_com = Serial.parseInt();
  }
}

if (Serial.available() > 0) {
  if (x_com == 999 && y_com == 999) {
    digitalWrite(LED_PIN_R, LOW);
  } else {
    digitalWrite(LED_PIN_R, HIGH);
  }
}

if (program_mode == 0) {
  digitalWrite(LED_PIN_B, HIGH);
  digitalWrite(LED_PIN_Y, LOW);

  float easing_speed = 0.1;

  float target_servo_x = analogRead(JOY_X);
  target_servo_x = map(target_servo_x, 0, 1023, 0, 180);

  float target_servo_y = analogRead(JOY_Y);
  target_servo_y = map(target_servo_y, 0, 1023, 25, 115);

  float prev_servo_x = servoHor.read();
  float prev_servo_y = servoVer.read();

  servoHor.write(lerp(prev_servo_x, target_servo_x, easing_speed));
  servoVer.write(lerp(prev_servo_y, target_servo_y, easing_speed));

  delay(10);
}
```

```
  if (program_mode == 1) {
    digitalWrite(LED_PIN_B, LOW);
    digitalWrite(LED_PIN_Y, HIGH);

    if (Serial.available() > 0 && x_com != 999 && y_com != 999) {
      if (x_com > width / 2 + 30)
        x_pos += angle;
      if (x_com < width / 2 - 30)
        x_pos -= angle;
      if (y_com < height / 2 - 30)
        y_pos -= angle;
      if (y_com > height / 2 + 30)
        y_pos += angle;

      x_pos = constrain(x_pos, 0, 180);
      y_pos = constrain(y_pos, 0, 180);

      servoHor.write(x_pos);
      servoVer.write(y_pos);
    }
  }
}

float lerp(float start, float end, float t) {
  return start + (end - start) * t;
}
```

ДОДАТОК Б

Програма, написана мовою Python, для пошуку обличчя у відеопотоці, визначення його координат, відправлення цих координат на плату Arduino, а також виведення на екран додаткових інформаційних показників.

```
import cv2
import serial
import time

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')

cap = cv2.VideoCapture(0)

ArduinoSerial = serial.Serial('com1', 9600, timeout=0.1)

time.sleep(1)

while cap.isOpened():
    ret, frame = cap.read()
    frame = cv2.flip(frame, 1)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = face_cascade.detectMultiScale(gray, 1.1, 6)

    if len(faces) == 0:
        string = 'X999Y999'
        print(string)

    for (x, y, w, h) in faces:
        string = 'X{0:d}Y{1:d}'.format((x + w // 2), (y + h // 2))
        print(string)
        ArduinoSerial.write(string.encode('utf-8'))
```

```
        cv2.circle(frame, (x + w // 2, y + h // 2), 2, (0, 255, 0), 2)
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 3)

        cv2.putText(frame, f"X: {x + w // 2 - 640 // 2}, Y: {480 // 2 - y - h // 2}", (10, 20),
                    cv2.FONT_HERSHEY_SIMPLEX,
                    0.5, (255, 255, 255), 1, cv2.LINE_AA)

    cv2.rectangle(frame, (640 // 2 - 30, 480 // 2 - 30),
                  (640 // 2 + 30, 480 // 2 + 30),
                  (255, 255, 255), 3)

    cv2.imshow('img', frame)

    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

cap.release()
cv2.destroyAllWindows()
```