Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

Васькова Олександра Вікторовича

на здобуття ступеня вищої освіти Бакалавра

Розумна система на основі
мікроплати для ведення відеонагляду

Галузь знань 12 – Інформаційні технології
Спеціальність 123 – Комп'ютерна інженерія
Освітня програма Програмування та захист комп'ютерних систем і мереж

Шифр КРБКІ.101002.21.01.02 ПЗ

Виконав студент 3 курсу група КІ1с-21-1 Олександр ВАСЬКОВ
Керівник канд. техн. наук, доцент Ігор МУЛЯР
Нормоконтролер старший викладач Сергій МОСТОВИЙ

До захисту допускаю:
Завідувач кафедри кібербезпеки Юрій КЛЬОЦ

13 06 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Кібербезпеки

Рівень вищої освіти Бакалавр

Галузь знань 12 – Інформаційні технології

Спеціальність 123 – Комп'ютерна інженерія

Освітня програма Програмування та захист комп'ютерних систем і мереж

ЗАТВЕРДЖУЮ

Завідувач кафедри кібербезпеки

Юрій КЛЬОЦ

15 лютого 2024 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Васькову Олександру Вікторовичу

1 Тема роботи Розумна система на основі мікроплати для ведення відеонагляду

Керівник роботи Муляр Ігор Володимирович

Затверджено наказом ректора університету від 15 лютого 2024 № 8

2 Строк подання студентом кваліфікаційної роботи на кафедру

3 Вихідні дані до роботи пристрій створений на основі мікроплати Raspberry Pi 5 та камери Raspberry Pi Camera V2.1, що запрограмований на Raspberry OS та мові програмування Python із використанням бібліотек Motion та OpenCV, що призначення для ведення відеонагляду, обробки записів та сповіщення

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз та визначення проблем відеонагляду, порівняльний аналіз існуючих рішень, вибір підходів для розробки системи на базі Raspberry Pi, розробка алгоритмів та пристроїв, програмно-апаратна реалізацію з використанням Raspberry Pi OS, Motion та OpenCV, інструкції користування системою, та висновки з переліком джерел та додатками, що включають креслення та лістинг коду

5 Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Креслення будови та детального підключення плати Raspberry Pi і камери, креслення адаптеру з 15 до 22 Pin FPC, схема алгоритму роботи системи відеонагляду, фото прототипу пристрою

6 Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Мостовий С.В., старший викладач кафедри кібербезпеки		

7 Дата видачі завдання 16 лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
Вибір і затвердження теми кваліфікаційної роботи	Січень-Лютий	виконав
Ознайомлення з предметною областю	Лютий	виконав
Дослідження існуючих рішень	Лютий	виконав
Постановка задачі	Березень	виконав
Визначення загальних принципів рішення задачі	Березень	виконав
Деталізація принципів рішення задачі	Квітень	виконав
Розробка проєктних рішень	Квітень	виконав
Апробація проєктних рішень	Травень	виконав
Оформлення пояснювальної записки згідно вимог	Травень	виконав
Оформлення графічної частини	Червень	виконав
Захист КР	Червень	виконав

Студент Олександр ВАСЬКОВ

Керівник кваліфікаційної роботи Ігор МУЛЯР

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КРБКІ.101002.21.01.02 ПЗ	Пояснювальна записка	62		
			Графічні матеріали			
2		КРБКІ.101002.21.01.02 Е1	Будова Rapberry PI та	1		
			підключення до камери			
3		КРБКІ.101002.21.01.02 Е2	Будова адаптеру з 15 до 22	1		
			Pin FPC			
4		КРБКІ.101002.21.01.02 Е3	Детальна будова Rapberry	1		
			PI та шлефу піделючення			
			камери			
5		КРБКІ.101002.21.01.02 Е8	Алгоритм роботи системи	1		
			відеонагляду			
6		КРБКІ.101002.21.01.02 Е8	Фото прототипу пристрою	1		

КРБКІ.101002.21.01.02 ВП

Зм	Арк	№ докум	Підпис	Дата	Відомість проекту	Літера	Аркуш	Аркушів
Розробив		Васьков О.В.		05.06.24		У	1	1
Перевір.		Муляр І.В.						
Н. контр.		Мостовий С.В.		19.06.24		ХНУ, КІ1с–21–1		
Затв.		Кльоц Ю.П.		19.06.24				

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Розумна система на основі мікроплати для ведення відеонагляду».

Автор роботи: Васьков Олександр Вікторович.

Керівник роботи: Муляр Ігор Володимирович.

Пояснювальна записка: 62 с., 37 рис., 3 дод., 40 джерело.

Графічна частина: 5 креслень.

МІКРОПЛАТА, СИСТЕМА ВІДЕОНАГЛЯДУ, НЕЙРОМЕРЕЖА, МОНІТОРИНГ, РОЗПІЗНАВАННЯ.

Метою кваліфікаційної роботи є планування та розробка розумної системи відеонагляду яка побудована на базі мікроплати та її програмування та налаштування для забезпечення необхідного функціоналу та зручності роботи.

Об'єктом дослідження є система відеонагляду.

Предметом дослідження є оцінка роботи систем відеонагляду та розробка власної системи на базі мікроплати, що використовує нейроалгоритми.

Під час проведення даного дослідження був використаний метод систематичного огляду літератури для вивчення і аналізу предметної області даного дослідження з текстових джерел інформації та для розробки власного проекту.

05.06.2024

ЗМІСТ

					КРБКІ.101002.21.01.02 ПЗ		
Зм.	Арк.	№докум.	Підпис	Дата			
Виконав		Васьков О.В.		05.06.21	Кваліфікаційна робота	Літера	Аркуш 2 / Аркушів 62
Перевір.		Муляр І.В.					
Н.контр.		Мостовий С.В.		10.06.21	Пояснювальна записка	ХНУ, КІ1с-21-1	
Затвер.		Кльоц Ю.П.					

ВСТУП

Відеонагляд стає все більш невід'ємною частиною сучасного життя, відіграючи ключову роль у забезпеченні безпеки та аналітиці. Згідно з дослідженням Grand View Research, прогнозується, що ринок відеонагляду зросте з 45,5 мільярдів доларів у 2020 році до 74,6 мільярдів доларів у 2025 році, що підкреслює збільшення попиту на ці технології. Однак, з розвитком цієї галузі, зростають і вимоги до неї, включаючи потребу в якісному відео, здатності обробляти великі обсяги даних, використанні алгоритмів для розпізнавання та інтеграції з іншими системами безпеки. Це веде до необхідності створення інтелектуальних систем відеонагляду, які використовують передові технології, такі як штучний інтелект, машинне навчання та інтернет речей.

Актуальність цієї теми невпинно зростає, оскільки вона відповідає потребам суспільства, яке стикається зі збільшенням обсягів відеоданих, що вимагають ефективної обробки та аналізу. Дослідження Cisco вказує, що до 2022 року відеотрафік становитиме 82% всього інтернет-трафіку, що свідчить про значний попит на відеосервіси. Розробка інтелектуальних систем відеонагляду, які використовують мікроплати для локальної обробки даних, може підвищити якість та ефективність відеонагляду, зменшуючи навантаження на мережі та сервери. Ці системи також відображають новітні технологічні тенденції та інновації, засновані на штучному інтелекті та машинному навчанні. MarketsandMarkets прогнозує, що ринок інтелектуального відеонагляду зросте з 21,4 мільярда доларів у 2019 році до 39,6 мільярда доларів у 2025 році, що підтверджує високий інтерес до цих технологій. Розумні системи відеонагляду на базі мікроплат можуть пропонувати нові функції, такі як детекція руху, розпізнавання об'єктів, класифікація сцен та аналіз поведінки, відкриваючи нові можливості для застосування у різних секторах. Згідно з IHS Markit, кількість камер відеонагляду у світі досягне 1 мільярда до 2021 року , що свідчить про їх широке поширення [1, 2].

У світлі стрімкого розвитку цифрових технологій, відеоспостереження перетворюється на справжній інструмент інтелектуального аналізу. Використання мікроплат у відеонаглядових системах відкриває нові можливості для автоматизації та оптимізації процесів безпеки. Ці пристрої можуть бути запрограмовані для виконання складних завдань, таких як ідентифікація та класифікація об’єктів у реальному часі, що робить їх незамінними у сферах, де потрібен швидкий та точний аналіз великих обсягів відеоданих. Інтеграція мікроплат з іншими технологіями, такими як дрони та автономні роботи, може створити ще більш ефективні та інноваційні системи відеонагляду, які будуть сприяти підвищенню рівня безпеки та контролю.

Дослідження в області відеонагляду сприяє розвитку мікроплат як потужних інструментів для обробки відеоданих. Мікроплати, які є компактними, енергоефективними та доступними, можуть виконувати широкий спектр завдань, включаючи захоплення, кодування, декодування та розпізнавання. Розробка інтелектуальних систем відеонагляду на базі мікроплат може продемонструвати переваги використання цих пристроїв для відеоспостереження та стимулювати їх застосування у різних секторах.

1 ТЕОРИТИЧНІ ТА ПРАКТИЧНІ ОСНОВИ ПОСТАВЛЕНОГО ЗАВДАННЯ

1.1 Аналіз предметної області і виявлення наявних проблем та завдань

Розумна система відеонагляду представляє собою комплексне рішення, яке здатне аналізувати відеопотоки, ідентифікувати об’єкти, сцени, події та поведінку, а також ініціювати відповідні дії, такі як сповіщення чи автоматизація процесів. Центральним елементом такої системи є мікроплата, яка є мініатюрним, але потужним пристроєм, що включає мікроконтролер, пам’ять, інтерфейси для підключення та бездротові модулі, а також датчики та камери, забезпечуючи обробку та зберігання відеоданих.

Мікроплата пропонує ряд переваг для створення персоналізованої системи відеонагляду:

– низька вартість та компактність, що дозволяє розміщувати пристрої у різних локаціях для відеоспостереження;

– автономність та гнучкість, оскільки мікроплати можуть працювати на акумуляторах або від альтернативних джерел енергії та підключатися до мережі через 3G, 4G, Wi-Fi тощо;

– інтелектуальність та адаптивність, що дозволяє мікроплатам аналізувати відеопотоки та реагувати на різноманітні ситуації.

При створенні системи відеонагляду на базі мікроплати важливо врахувати наступні аспекти:

– вибір мікроплати з необхідними характеристиками та функціоналом;

– забезпечення живлення та мережевого з’єднання, використовуючи, наприклад, батареї, сонячні панелі, SIM-карти або Wi-Fi;

– комплектація системи з мікроплати, камери та накопичувача, а також налаштування мережевого доступу та живлення;

– розміщення мікроплати в оптимальному місці та налаштування кута огляду камери;

– розробка програмного забезпечення для відеонагляду та управління системою;

– моніторинг роботи системи в реальному часі, перегляд записів, отримання сповіщень та налаштування параметрів системи.

Також, при розробці власної системи відеонагляду на базі мікроплати, необхідно враховувати потенційні проблеми, які можуть виникнути під час її розробки та експлуатації.

У контексті сучасних викликів, системи відеонагляду зіштовхуються з рядом проблем, які вимагають уваги та вирішення. Однією з ключових проблем є нестабільність живлення та зв'язку, що може призвести до переривань у моніторингу та втрати відеоданих. Причинами цього можуть бути недоліки в якості джерела живлення, низький заряд батареї, перебої в електромережі, слабкий сигнал чи перешкоди в мережі. Такі проблеми можуть мати серйозні наслідки, включаючи неможливість отримання відеопотоку в реальному часі та втрату критично важливих даних [3].

Проблема обмеженої пам'яті та обчислювальної потужності мікроплати також є важливою. Вона може обмежувати тривалість запису та кількість підключених камер, а також ускладнювати виконання складних аналітичних завдань. Це може бути пов'язано з низькою ємністю пам'яті, швидкістю обробки даних мікроконтролером, пропускною здатністю мережі та складністю алгоритмів.

Нарешті, складність налаштування та обслуговування системи може вимагати від користувачів технічних знань та навичок. Нестандартність та різноманітність компонентів системи, несумісність та неінтегрованість різних частин, а також неінтуїтивність програмного забезпечення можуть ускладнити встановлення, керування та усунення несправностей.

Ці заходи можуть допомогти створити більш надійну та ефективну систему відеонагляду, яка зможе відповідати сучасним вимогам безпеки та аналітики. Головною метою проекту є створення інноваційної цифрової системи відеоспостереження, яка базується на мікроплаті. Система буде оснащена

інтегрованою камерою, засобами для зберігання даних та алгоритмами розпізнавання, що дозволить ефективно уникати зазначених проблем і забезпечить оптимальні результати у процесі розробки [4, 5].

1.2 Порівняльний аналіз переваг та недоліків існуючих рішень

Існуючі рішення систем відеоспостереження можна класифікувати на дві основні категорії: аналогові та цифрові.

Аналогові системи відеоспостереження використовують аналогові відеокамери, які передають відеосигнал до відеореєстратора через коаксіальний кабель. За допомогою цього типу системи, відеосигнал від камери до відеореєстратора передається без цифрового конвертування, що дозволяє зберегти оригінальну якість зображення. Відеореєстратор зберігає відеодані на жорсткому диску, що дозволяє зберегти запис відео на тривалий період часу. За допомогою відеореєстратора, вихідний відеосигнал може бути виведений на монітор або інший пристрій для перегляду в реальному часі або після запису [6].

Рисунок 1.1 – Аналогова система відеонагляду

Цифрові системи відеоспостереження представляють собою передову технологію, що використовує цифрові відеокамери для забезпечення

безперебійного збору відеоданих. Ці камери передають відеосигнал до мережевого відеореєстратора або сервера, що може бути підключений через мережевий кабель або бездротовий зв'язок, такий як Wi-Fi [7].

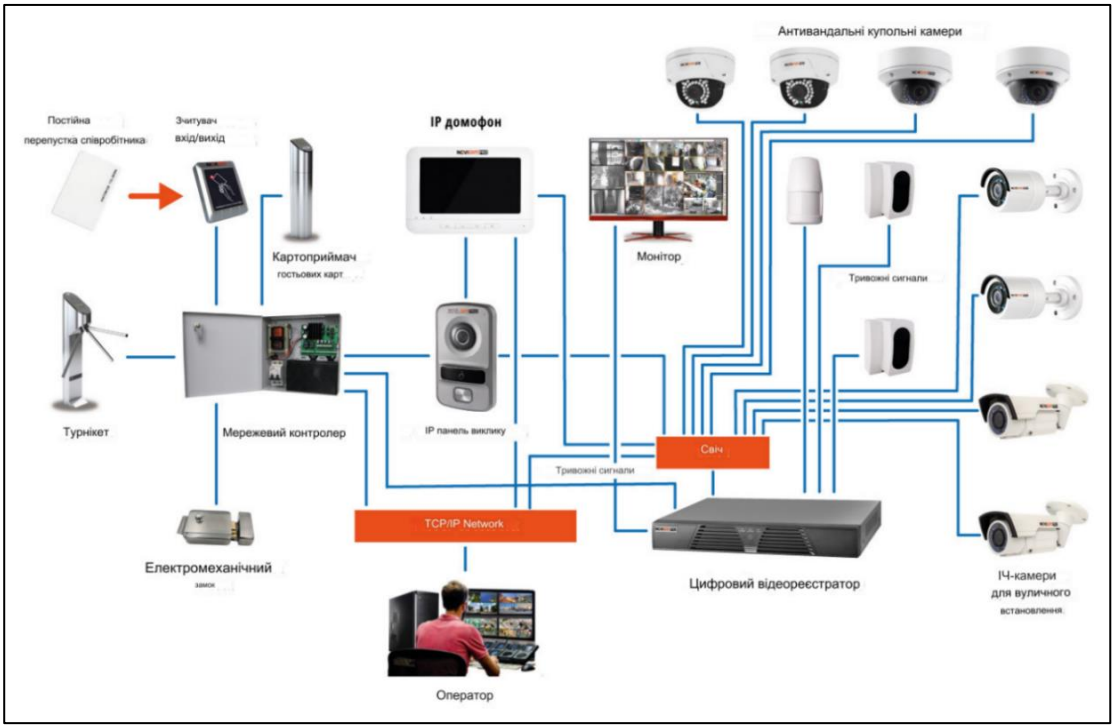


Рисунок 1.2 – Цифрова система відеонагляду

Враховуючи переваги та недоліки цифрових та аналогових систем відеоспостереження, можна виділити такі основні характеристики:

– цифрові системи зазвичай пропонують вищу якість зображення, включаючи кращу роздільну здатність, широкий динамічний діапазон, нічне бачення та стабілізацію зображення, навпроти, аналогові системи мають обмеження щодо роздільної здатності, контрастності, чіткості та колірної гами, що призводить до нижчої якості зображення;

– цифрові системи генерують значно більший обсяг відеоданих за рахунок використання високої роздільної здатності, кадрової частоти та бітрейту, з іншого боку, аналогові системи виробляють менше даних, завдяки використанню нижчої роздільної здатності, кадрової частоти та бітрейту;

– цифрові системи, які використовуються в сучасному світі, вимагають значно вищої пропускної здатності та швидкості передачі даних, це пов'язано з

тим, що вони передають значно більший обсяг відеоданих, що є критично важливим для забезпечення якості, з іншого боку, аналогові системи, які використовуються у менш технологічно розвинених ситуаціях або для менш об'ємних задач, передають значно менший обсяг даних, внаслідок цього, вони вимагають меншої пропускної здатності та швидкості передачі, що робить їх менш вимогливими до інфраструктури.;

– цифрові системи можуть надати вищий рівень надійності та безпеки, використовуючи захищені протоколи та шифрування, а також надавати можливість віддаленого доступу та контролю, натомість аналогові системи мають меншу надійність та безпеку, оскільки вони використовують незахищені протоколи та не шифрують відеодані, а також не надають можливості віддаленого доступу та контролю;

– цифрові системи, характеризуються вищою вартістю та складністю установки та обслуговування, що вимагає використання більш високотехнологічного обладнання, специфічного програмного забезпечення та мережевих компонентів, це потребує кваліфікованих спеціалістів для правильного встановлення та подальшого обслуговування, з іншого боку, аналогові системи відрізняються нижчою вартістю та складністю, використовуючи більш доступне та просте в управлінні обладнання, базове програмне забезпечення та мережеві рішення, це робить їх більш доступними для широкого користувача, який не має глибоких знань в даній галузі.

Після глибокого аналізу та порівняння ефективності аналогових та цифрових систем відеоспостереження, можна з упевненістю стверджувати, що цифрові рішення виявляються найбільш ефективними. Це можна пояснити більш високим рівнем якості відео, великою ємністю зберігання даних, високою швидкістю передачі даних, а також кращою надійністю та безпекою.

У контексті цифрових систем, найпередовішою та найновітнішою є IP система відеоспостереження. IP система відеоспостереження - це високотехнологічна система, яка використовує цифрові відеокамери, що з'єднуються з глобальною мережею Інтернет або локальною мережею. Ця система

надає можливість передати, зберігати, обробляти та відтворювати відеодані на широкому спектрі пристроїв, включаючи комп’ютери, смартфони, планшети та інші.

Як приклад сучасної IP системи відеоспостереження можна навести рішення від відомої компанії Hikvision. На рисунку 1.3 зображено, що ця конкретна система складається з IP-камери та 4-канального мережевого відеохабу[8].

Рисунок 1.3 - Комплект IP відеоспостереження Hikvision

Система відеоспостереження, що розглядається в даному контексті, володіє рядом важливих переваг, які вже встигли добре зарекомендувати себе на ринку. Така система не лише забезпечує безпеку, але і дозволяє контролювати ситуацію в реальному часі, надаючи можливість вчасно реагувати на будь-які події. Вона відкриває широкі можливості для моніторингу і контролю, що робить її незамінною для багатьох сфер діяльності.

Перша і найважливіша перевага полягає в тому, що відеокамери від компанії Hikvision можуть записувати та передавати відео високої роздільної здатності - до 12 мегапікселів. Така висока якість дозволяє чітко розгледіти найдрібніші деталі зображення, включаючи обличчя людей, номери автомобілів, предмети і так далі.

Другою важливою перевагою є те, що ці камери можуть транслювати зображення в реальному часі на різноманітні пристрої, як-от персональні комп'ютери, смартфони чи планшети, і це можливо реалізувати через інтернет чи локальну мережу.

Як і будь-яка технологія, IP система відеоспостереження від Hikvision не є винятком і має певні обмеження та недоліки, які важливо враховувати при її виборі та використанні. Незважаючи на широкий спектр функціональності та передові технологічні рішення, які пропонує Hikvision, користувачі можуть зіткнутися проблемами.

Перший і головний недолік - це те, що камери від Hikvision зазвичай коштують дорожче, ніж аналогові камери. Це пояснюється тим, що вони використовують більш потужну та складну апаратну частину. Крім того, для їх роботи може знадобитися додаткове обладнання, таке як комутатори, маршрутизатори, сервери, що також підвищує вартість системи.

Другий недолік полягає в тому, що камери підключаються до мережі, що створює потенційну загрозу для їх захисту від несанкціонованого доступу, вірусів, хакерських атак і так далі. Тому для забезпечення безпеки необхідно використовувати захищені протоколи, шифрування, паролі, файрволи та інші засоби захисту мережі.

Третій недолік полягає в тому, що ці камери можуть не бути сумісними з обладнанням для відеонагляду від інших виробників, оскільки вони використовують власні стандарти та протоколи. Тому рекомендується використовувати обладнання від одного виробника або перевіряти сумісність перед покупкою.

Через вищеописані проблеми, на мою думку, вибір IP системи відеоспостереження від Hikvision може не бути найкращим рішенням, враховуючи її потенційні недоліки. Зокрема, висока вартість придбання та подальшого обслуговування може стати значним фінансовим тягарем. Крім того, існує ризик уразливості до кібератак, оскільки система підключена до мережі, що може бути вразливою до зловмисників. Також, обмежена сумісність з обладнанням від інших виробників може створити труднощі для інтеграції та масштабування системи. Враховуючи ці аспекти, IP система відеоспостереження від Hikvision може не відповідати потребам ринку та не бути оптимальним вибором для споживачів, які шукають гнучкість та високий рівень безпеки [8, 9].

1.3 Визначення підходу до вирішення поставленої задачі

У рамках визначеного завдання було прийнято рішення створити систему, яка базуватиметься на мікропроцесорі відомої компанії Raspberry Pi. Ця компанія відома своїми компактними та економічно ефективними мікроплатами, які знайшли широке застосування у багатьох сферах, включаючи відеонагляд, робототехніку, ігрову індустрію та освітні проекти[10, 11].

На ринку представлено кілька моделей мікроплат Raspberry Pi, кожна з яких має свої унікальні характеристики, цінові категорії та функціональні можливості, що дозволяє користувачам обрати оптимальний варіант для своїх потреб. Давайте далі розглянемо деякі з них.

Мікроплата Raspberry Pi 5 (рисунок 1.4) представляє собою останнє покоління мікроплат від Raspberry Pi, яке вирізняється своєю високою продуктивністю та багатофункціональністю. Ця мікроплата оснащена потужним 8-ядерним процесором, що працює на частоті 2 ГГц, та пропонує варіанти з 2, 4 або 8 ГБ оперативної пам’яті, що робить її ідеальною для вимогливих завдань.

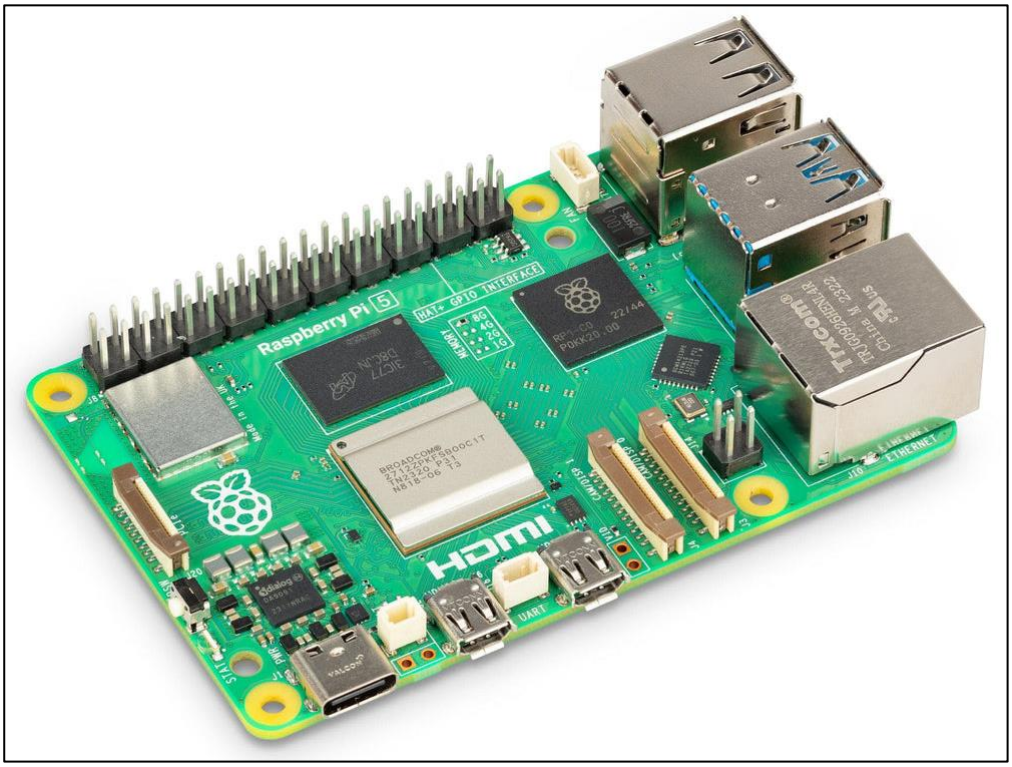


Рисунок 1.4 - Raspberry Pi 5

Мікроплата Raspberry Pi 4 Model B (рисунок 1.5) є попередником останнього випуску мікроплат від Raspberry Pi, пропонуючи солідну продуктивність за поміркованою ціною.

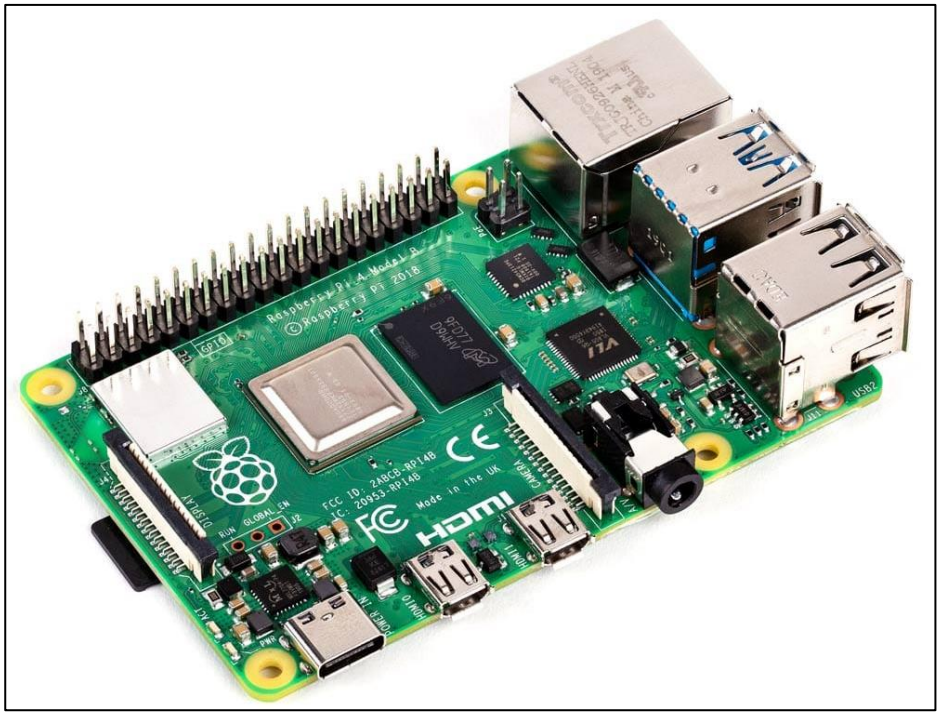


Рисунок 1.5 - Raspberry Pi 4 Model B

Мікроплата Raspberry Pi Zero W (рисунок 1.6) вирізняється своїми компактними розмірами та економічною цінністю, пропонуючи базову функціональність для широкого спектру застосувань.

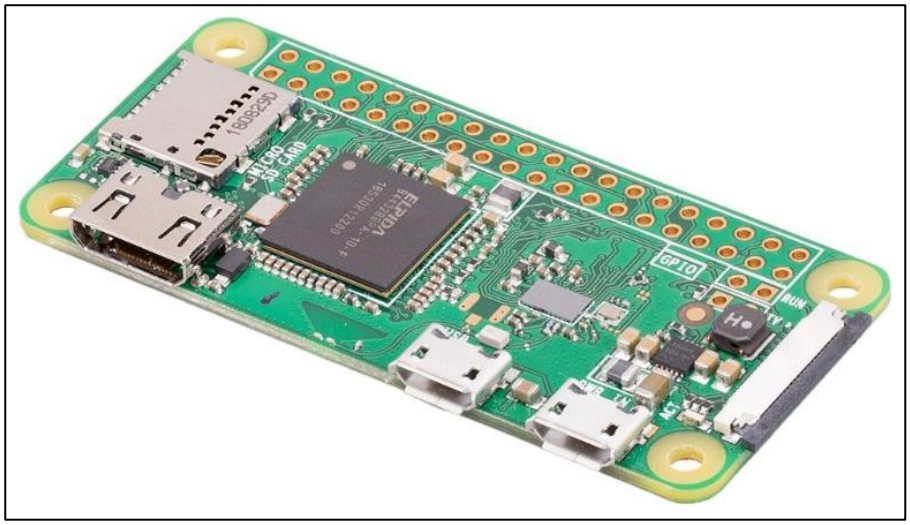


Рисунок 1.6 - Raspberry Pi Zero W

Мікроплата Raspberry Pi Pico (рисунок 1.7) відрізняється своїм компактним розміром та ефективністю. Він оснащений двоядерним процесором, що працює з частотою 133 МГц, та має 264 КБ оперативної пам’яті для виконання коду. З 2 МБ вбудованої флеш-пам’яті, цей мікроконтролер забезпечує достатньо місця для зберігання програм та даних.

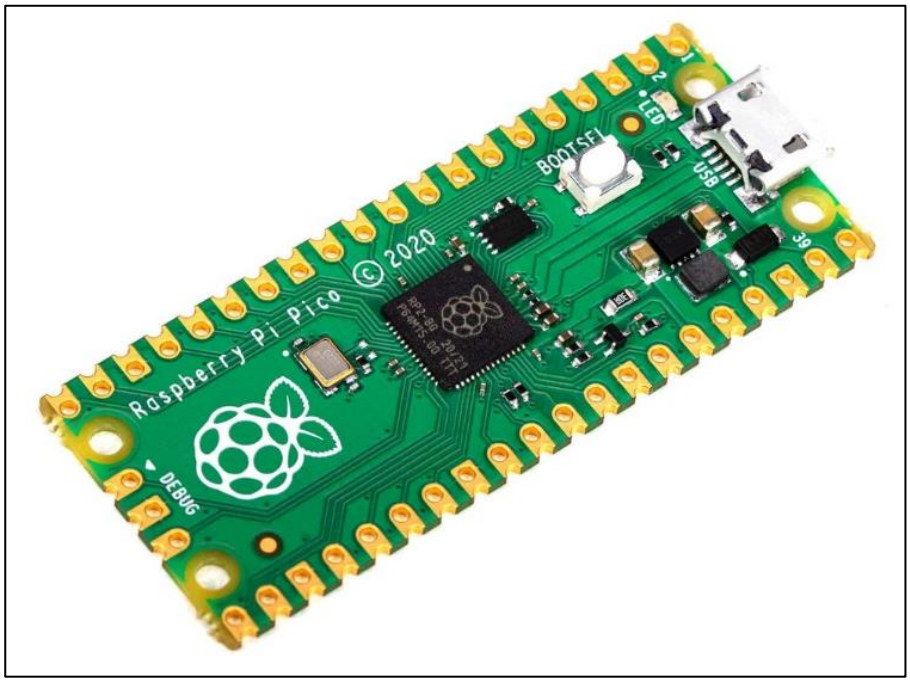


Рисунок 1.7 - Raspberry Pi Pico

Для реалізації системи відеоспостереження необхідно інтегрувати спеціалізований модуль камери. Модулі камери, розроблені для Raspberry Pi, з’єднуються з мікроплатою через інтерфейс CSI-2, що дозволяє пряму передачу відеоданих до відеопроцесора VideoCore на самій платі. Це забезпечує ефективну передачу відео високої якості, оптимізуючи при цьому використання ресурсів мікроплати. На ринку доступні різні моделі та конфігурації модулів камери Raspberry Pi, кожна з яких має свої унікальні характеристики та цінові категорії, надаючи користувачам можливість вибрати оптимальне рішення, яке відповідає їхнім потребам та бюджету.

Камерний модуль Raspberry Pi Camera Module 2 (рисунок 1.8) був представлений у квітні 2016 року, став наступником оригінального камерного модуля. Оснащений 8-мегапіксельним сенсором Sony IMX219, що є значним

покращенням порівняно з 5-мегапіксельним сенсором OmniVision OV5647 у попередній моделі, цей модуль дозволяє здійснювати високоякісну відеозйомку та фотографування. Його інтуїтивно зрозумілий інтерфейс робить його доступним для новачків, водночас пропонуючи розширені функції для тих, хто бажає поглибити свої технічні знання. Цей модуль камери є чудовим інструментом для тих, хто прагне розвивати свої навички у сфері цифрової фотографії та відеозйомки.

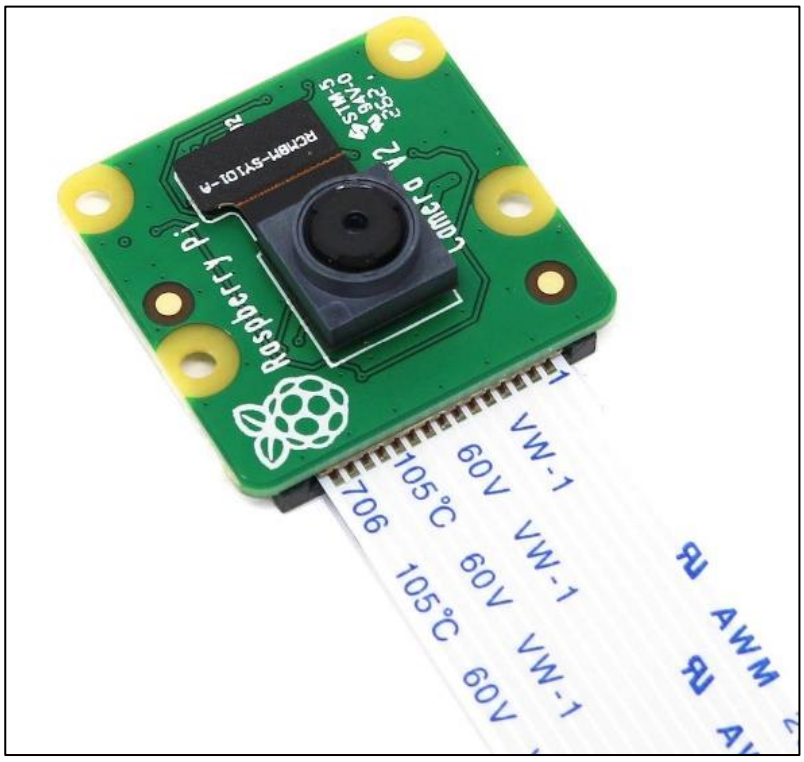


Рисунок 1.8 - Raspberry Pi Camera Module 2

Камерний модуль Raspberry Pi Camera Module 3 (рисунок 1.9) є третім випущеним модулем камери Raspberry Pi, який з’явився на ринку у 2023 році, є значним кроком вперед у лінійці камерних модулів компанії. Цей модуль оснащений 11,9-мегапіксельним датчиком зображення Sony IMX708, який забезпечує фіксований фокус та підтримує високу роздільну здатність зображень до 4608x2592 пікселів. Цей модуль камери є відмінним вибором для тих, хто шукає якісне та гнучке рішення для своїх проектів у сфері цифрового зображення.

Рисунок 1.9 - Raspberry Pi Camera Module 3

Для забезпечення ефективної роботи та надійної передачі даних системи відеонагляду, важливо вибрати відповідні джерела живлення та зв’язку. Використовуючи Raspberry Pi 5 як основу, можна розглянути наступні опції для підключення до мережі:

– Ethernet - це дуже надійний метод підключення, який забезпечує стабільну швидкість передачі даних до 1 Гбіт/с, для підключення достатньо вставити Ethernet-кабель безпосередньо в порт на Raspberry Pi 5 та з’єднати його з роутером або комутатором;

– Wi-Fi - цей бездротовий метод підключення дозволяє забезпечити більшу гнучкість розміщення Raspberry Pi 5.

– Bluetooth - ця технологія дозволяє бездротово з’єднувати Raspberry Pi 5 з такими пристроями, як смартфони, планшети, навушники, клавіатури та миші, вбудований модуль Bluetooth на Raspberry Pi 5 підтримує останні стандарти, включаючи Bluetooth 4.2 та 5.0;

– GSM/GPRS/GNSS - це варіант, що забезпечує зв’язок з мобільними мережами та системами глобальної навігації, для цього потрібно використовувати спеціалізований модуль зв’язку, який можна підключити до Raspberry Pi 5 через GPIO або USB.

Для успішної розробки системи відеонагляду на базі мікроплати Raspberry Pi, ключовим елементом є вибір відповідного програмного забезпечення та бібліотек. Ці інструменти відіграють вирішальну роль у визначенні логіки роботи та функціональних можливостей системи. Використання різноманітних програм та бібліотек дозволяє не тільки контролювати камери, записувати та переглядати відео, але й імплементувати інтелектуальні функції, такі як:

– розпізнавання облич із використанням бібліотек, таких як OpenCV або TensorFlow, може дозволити системі ідентифікувати та відстежувати обличчя в реальному часі;

– детекція руху із алгоритмами, що базуються на зміні пікселів у відеопотоці, можуть виявляти рух, що є корисним для систем безпеки;

– аналіз поведінки за допомогою машинного навчання можна аналізувати поведінку людей або об’єктів, що може бути застосовано в рітейлі для вивчення покупців.

Для реалізації цих функцій, можна використовувати програмні пакети та мови програмування які описані далі.

Python - завдяки своїй гнучкості та широкому спектру бібліотек, Python є ідеальним вибором для розробки систем відеонагляду.

Node.js - це може бути використано для створення веб-серверів для віддаленого доступу та управління системою відеонагляду.

C/C++ - для оптимізації продуктивності та роботи з апаратним забезпеченням, C/C++ можуть бути використані для розробки низькорівневого програмного забезпечення.

Raspbian – це спеціалізована операційна система, створена для Raspberry Pi, яка побудована на основі Debian Linux. Ця система включає в себе комплект драйверів та програмного забезпечення, необхідного для ефективної роботи з Raspberry Pi, в тому числі для управління камерою, Wi-Fi, Bluetooth, GPIO, та іншими компонентами.

Motion - це програмне забезпечення для спостереження, яке дозволяє записувати відео з веб-камер та IP-камер. Ця програма здатна виявляти рух,

зберігати фотографії та відеозаписи, надсилати повідомлення про сповіщення, а також транслювати відео через веб-інтерфейс або локальну мережу.

OpenCV - це велика бібліотека, призначена для роботи з комп'ютерним зором, яка включає понад 2500 алгоритмів обробки зображень та відео.

Pi Camera - це інструментарій для управління камерою Raspberry Pi, який надає можливість користувачам налаштовувати такі параметри камери, як роздільна здатність, частота кадрів, експозиція, баланс білого та ISO. Він також дозволяє здійснювати запис відео, переглядати його, робити фотографії, а також застосовувати різноманітні ефекти та фільтри.

Використовуючи ці інструменти та бібліотеки, можливо розробити ефективну та адаптивну систему відеоспостереження на базі мікроконтролерів Raspberry Pi, яка буде відповідати специфічним вимогам та перевагам користувачів [15].

Сучасні системи відеоспостереження повинні забезпечувати функціонал розпізнавання осіб, що є ключовим для попередження потенційних злочинів та захисту власності. Завдяки бібліотеці OpenCV, система відеонагляду, побудована на Raspberry Pi, також може надавати цю можливість. OpenCV, як було зазначено раніше, дозволяє виконувати широкий спектр завдань, пов'язаних з виявленням, ідентифікацією, слідкуванням, аналізом та модифікацією об'єктів у зображеннях та відео. Цей процес включає два ключові етапи: виявлення обличчя та його подальшу ідентифікацію.

Процес детектування обличчя полягає у виявленні та визначенні місцезнаходження обличчя на фотографії чи у відеоматеріалі. Існує кілька методів для цього, включаючи каскади Хаара та згорткові нейронні мережі.

Каскади Хаара -це традиційний підхід, який використовує прості прямокутні фільтри для ідентифікації характерних рис обличчя, таких як очі, ніс, рот і т.д. Фільтри послідовно застосовуються до різних сегментів зображення для відкидання тих, що не містять обличчя. Цей метод є швидким і ефективним, але може бути неточним і не завжди справляється з різноманітністю умов освітлення, кутів обличчя, виразів тощо.

Згорткові нейронні мережі (CNN) - це більш сучасний підхід, який використовує глибоке навчання для визначення обличчя на зображенні. CNN складається з багатьох шарів, які проводять операції згортки, пулінгу, активації та повнозв'язного з'єднання, де кожен шар вивчає все більш складні особливості обличчя, такі як текстура, форма, контури. Цей метод є високоточним, але вимагає значної кількості даних для тренування та обчислювальних ресурсів для обробки.

Ідентифікація обличчя - це процедура, яка полягає у порівнянні виявленого обличчя з базою даних існуючих обличь, щоб встановити відповідність або ідентифікувати особу. Існують різноманітні методи для цього процесу, зокрема Eigenfaces, Fisherfaces та Local Binary Patterns

Eigenfaces - цей метод використовує принципи лінійної алгебри для виявлення основних компонентів, які відображають варіації обличь у базі даних. Кожне обличчя представляється як комбінація цих основних компонентів, відомих як eigenfaces. Для ідентифікації, обличчя проектується у простір eigenfaces, де шукається найближчий вектор у базі даних

Fisherfaces - цей метод застосовує дискримінантний аналіз для визначення найбільш значущих компонентів варіації обличь у базі даних. Обличчя представляється як комбінація цих компонентів, відомих як fisherfaces. Для ідентифікації, обличчя проектується у простір fisherfaces, де шукається найближчий вектор

Local Binary Patterns (LBP) - цей метод використовує статистичний аналіз для виявлення локальних характеристик обличь у базі даних. Обличчя ділиться на маленькі регіони, і кожен регіон кодується за допомогою бінарного числа, яке представляє розподіл інтенсивності пікселів у регіоні. Для ідентифікації, фрагменти обличчя порівнюються з фрагментами у базі даних, шукаючи найменшу відстань між ними.

OpenCV пропонує імплементації цих та інших методів для детекції та ідентифікації обличь, а також можливість розробки власних моделей та алгоритмів. Завдяки OpenCV, розробка додатків для розпізнавання обличь стає значно спрощеною та доступною [17, 18].

1.4 Постановка задачі

Виходячи з раніше згаданих елементів, було прийнято рішення використовувати мікроконтролер Raspberry Pi 5 та камеру Raspberry Pi Camera Module 3 для розробки інтелектуальної системи відеоспостереження. Вибір припав на Raspberry Pi 5 через її переваги у вигляді високої продуктивності, об'єму пам'яті, швидкості обробки даних, якості зображення, функціональності, сумісності з іншими пристроями, надійності, безпеки та адаптивності порівняно з іншими доступними моделями. Raspberry Pi Camera Module 3 було обрано завдяки його високій роздільній здатності та якості зображення, а також ефективній передачі даних через інтерфейс CSI-2, який забезпечує пряме підключення до відеопроцесора VideoCore на платі Raspberry Pi 5. Як джерело живлення було вибрано офіційний 15-ватний USB-C адаптер. Програмування системи буде виконано на мові C++, з використанням існуючих бібліотек на Python.

Задачею цього проекту є створення системи відеоспостереження на базі Raspberry Pi, яка здатна моніторити об'єкти (такі як будинки, офіси, склади, гаражі тощо), забезпечувати захоплення, передачу, зберігання, обробку та відображення відеоданих. Система також буде виконувати аналітичні функції, такі як розпізнавання об'єктів, подій, поведінки, та ініціювати дії на основі аналізу відеоданих, включаючи сповіщення, управління, оптимізацію, автоматизацію. Система відеоспостереження на базі Raspberry Pi може бути застосована для широкого спектру цілей, включаючи безпеку, контроль, моніторинг, освіту, дослідження, та може функціонувати в різноманітних умовах, як внутрішніх, так і зовнішніх, при денному чи нічному освітленні, в статичному або динамічному режимах.

Використовуючи платформу Raspberry Pi для створення системи відеоспостереження, можна застосувати наступні техніки:

– управління камерою -цей підхід дозволяє активувати або деактивувати камеру, регулювати її налаштування, перемикати режими, змінювати масштаб, фокусувати та направляти камеру, що з'єднана з мікроконтролером Raspberry Pi;

– обробка відеоданих - ці методи включають в себе процеси збору, збереження, передачі, конвертації, аналізу, фільтрації, шифрування та дешифрування відеоданих, які камера захоплює, а також їх відображення на різних пристроях, таких як монітори, смартфони, планшети чи комп’ютери;

– аналітичні функції - ці методи дозволяють проводити аналіз відеоданих для розпізнавання об’єктів, ситуацій, подій, поведінки, а також визначати параметри, такі як кількість, розмір, форма, колір, швидкість, напрямок, емоції, ідентифікацію, класифікацію, сегментацію, локалізацію;

– реагування на аналіз відеоданих - ці методи включають в себе вживання дій на основі аналізу відеоданих, таких як відправлення сповіщень, управління системою, оптимізація та автоматизація процесів, система може відправляти сповіщення на електронну пошту, телефон, планшет чи комп’ютер у випадку виявлення потенційно небезпечних ситуацій, подій або об’єктів.

2 РОЗРОБКА АЛГОРИТМІВ ТА ПРИСТРОЇВ ДЛЯ СИСТЕМИ ВЕДЕННЯ ВІДЕОНАГЛЯДУ

2.1 Перелік існуючих рішень та обґрунтування вибору плати Raspberry Pi 5

Для побудови системи відеонагляду було обрано взяти за основі мікроплату. Мікроплата - це компактний одноплатний комп’ютер, який містить усі необхідні компоненти для функціонування як повноцінний комп’ютер. Вона включає процесор, пам’ять, вхід/вихід (I/O) порти, і часто має можливість підключення до мережі та розширення за допомогою додаткових модулів.

Є ряд переваг чому варто використовувати мікроплати для побудови систем відеонагляду:

– мікроплати пропонують велику гнучкість у підборі компонентів та налаштуванні системи під конкретні потреби;

– невеликий розмір мікроплат робить їх ідеальними для використання у відеонаглядових системах, де простір може бути обмеженим;

– мікроплати споживають значно менше енергії порівняно з традиційними комп’ютерами, що робить їх економічно вигідними для тривалої експлуатації;

– мікроплати зазвичай коштують менше, ніж стандартні комп’ютерні системи, що робить їх доступними для широкого кола користувачів;

– існує велика спільнота розробників та ентузіастів, які постійно працюють над створенням нових проектів та покращенням існуючих, що забезпечує хорошу підтримку та багато ресурсів для навчання;

– багато мікроплат мають GPIO-піни та інші інтерфейси, які дозволяють підключати різноманітні датчики, модулі камер та інше обладнання.

Загалом, мікроплати пропонують високу продуктивність та багато можливостей для розробки індивідуальних рішень у сфері відеонагляду, будучи при цьому економічно вигідними та легкими у використанні [13, 14].

Серед існуючих на ринку мікроплат можна виділити наступні мікроплати, що описано далі.

Raspberry Pi - це одноплатний комп’ютер, який можна використовувати для

різних проектів, включаючи відеонагляд. Raspberry Pi має невеликі розміри, що дозволяє легко вбудовувати його в різні пристрої. Наявність GPIO-пінів дозволяє підключати датчики, камери та інші пристрої безпосередньо до плати. Raspberry Pi підтримує різні операційні системи, такі як Raspberry Pi OS, Ubuntu та інші. Вартість відносно низька в порівнянні з іншими комп'ютерами. Велика спільнота розробників та ентузіастів активно допомагає один одному у розв'язанні проблем та розробці проектів. Можливості залежать від використання та додаткового обладнання. Raspberry Pi може використовуватися для відеонагляду, серверів, IoT-пристроїв, навчання програмуванню та багатьох інших завдань.

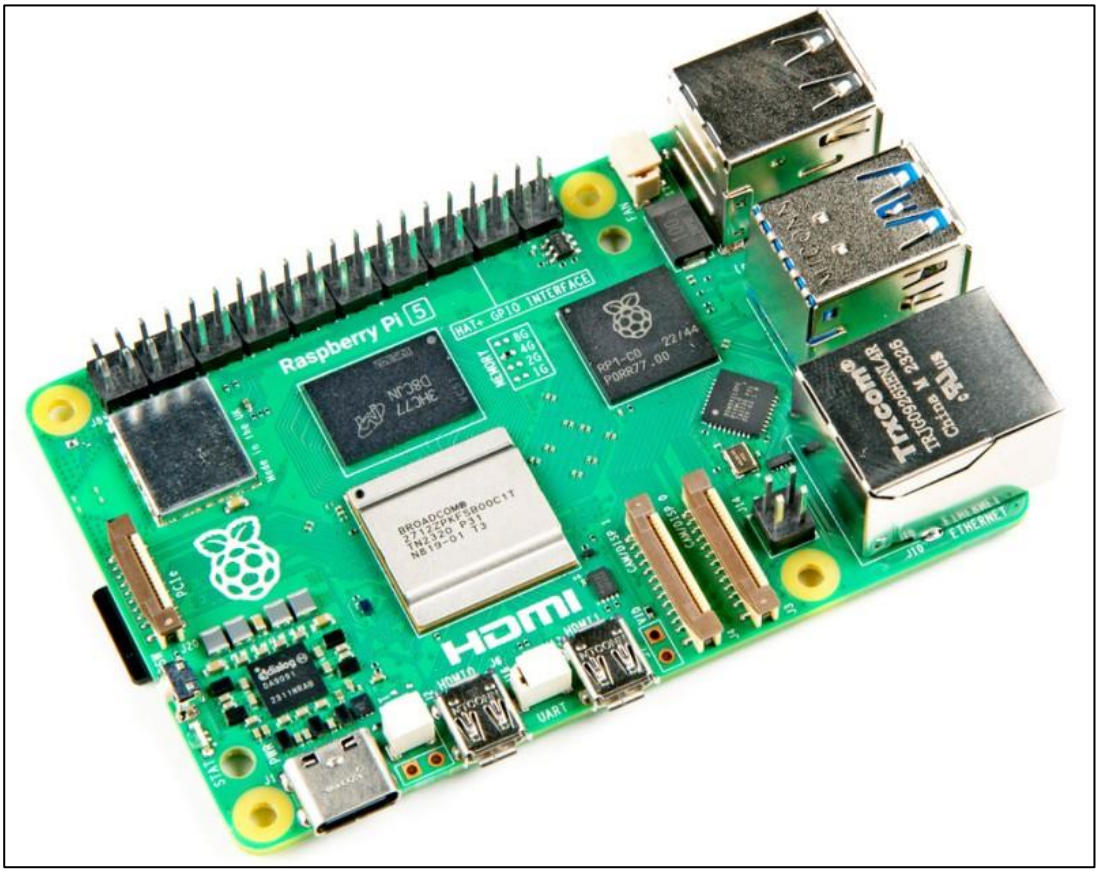


Рисунок 2.1 – Мікроплата Raspberry Pi

Arduino - це платформа для створення інтерактивних електронних пристроїв за допомогою коду та апаратного забезпечення. Arduino IDE - це програмне забезпечення, яке дозволяє писати код та завантажувати його на плату, воно має інтуїтивний інтерфейс та підтримує різні плати Arduino. Користувач може

використовувати бібліотеки для розширення функціональності Arduino. Ядра дозволяють додавати нові плати до Arduino IDE. Arduino можна використовувати для різних проектів: від вбудованих систем до IoT-пристроїв та робототехніки. Вона підтримує різні мови програмування, такі як C++ та MicroPython. Велика спільнота розробників та ентузіастів активно допомагає один одному у розв'язанні проблем та розробці проектів.

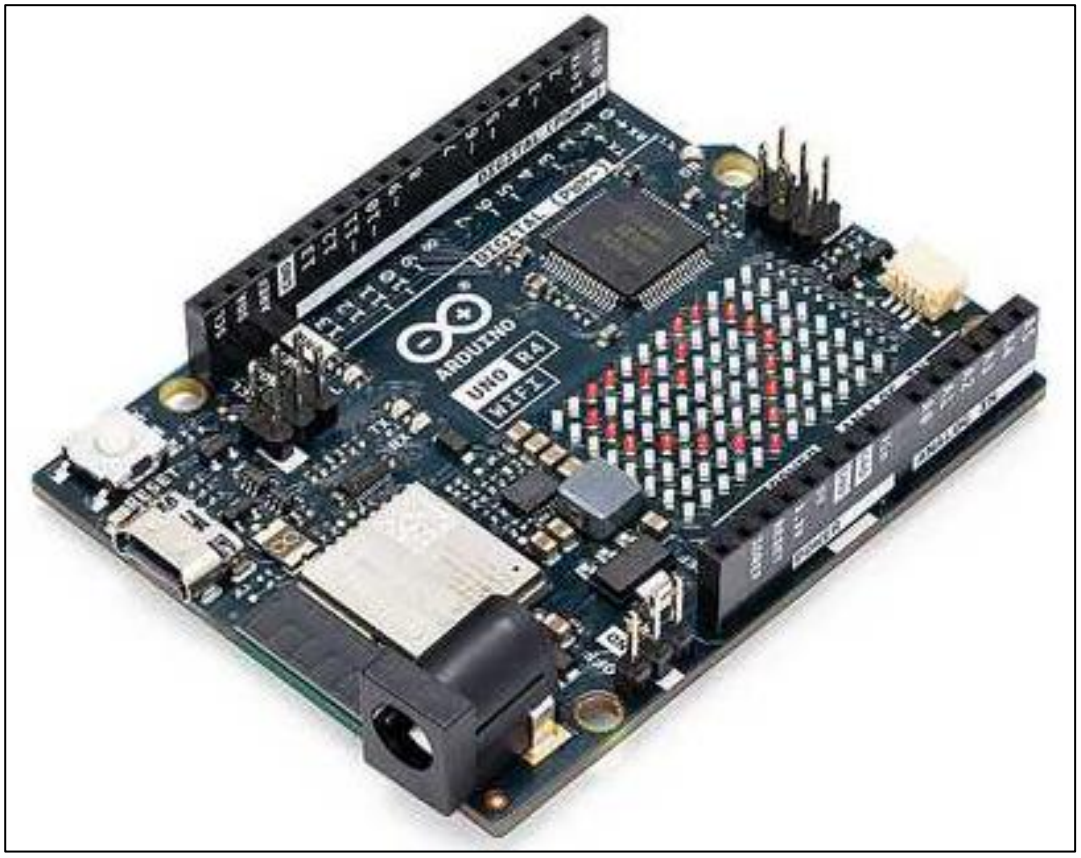


Рисунок 2.2 – Мікроплата Arduino

NVIDIA Jetson Nano - це малий, потужний комп'ютер для вбудованих застосувань та штучного інтелекту (AI IoT), який надає можливість використовувати сучасний AI. Jetson Nano має розміри менше, ніж кредитна картка, але при цьому надає велику продуктивність для розгортання AI на краю. Ця мікроплата надає 472 GFLOPs для сучасних алгоритмів AI. Вона може виконувати кілька нейронних мереж паралельно та обробляти дані з високороздільних сенсорів одночасно.

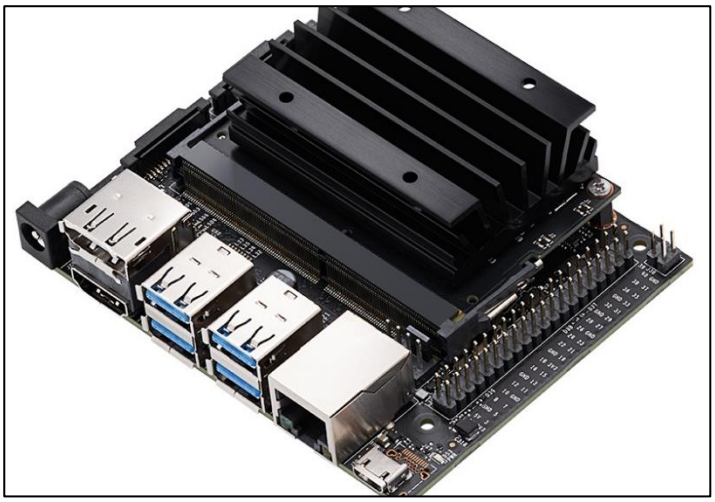


Рисунок 2.3 – Мікроплата NVIDIA Jetson Nano

Orange Pi - це бренд відкритих продуктів, який пропонує різноманітні плати розробки, обчислювальні модулі, клавіатурні комп'ютери та інше. Orange Pi OS - це офіційна операційна система з відкритим вихідним кодом для плат Orange Pi, вона надає робочий стіл Orange Pi для ряду плат розробки Orange Pi, клавіатурних ПК та інших ПК. Orange Pi OS може запускати додатки під управлінням Android, Windows та Linux. Вона є швидкою, безпечною та стабільною операційною системою з красивим інтерфейсом та простим у використанні розширенням. Orange Pi співпрацює з Microsoft і випустила перший набір розробки для Інтернету речей (IoT) на базі Azure Smart Cloud в Китаї.

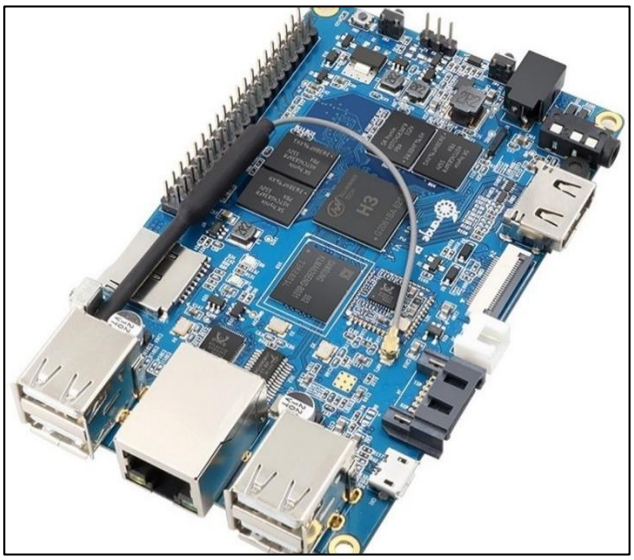


Рисунок 2.4 – Мікроплата Orange Pi

Мій вибір для системи відеонагляду впав на Raspberry Pi 5 і він обумовлений

кількома наступними перевагами:

– вона забезпечує значно вищу продуктивність завдяки оновленому процесору та підтримці більшої кількості оперативної пам’яті;

– дана мікроплата надає можливість підключення двох камер одночасно, що розширює можливості моніторингу без необхідності використання додаткових пристроїв;

– всі дані обробляються локально, що забезпечує більшу конфіденційність та безпеку, оскільки дані не передаються через інтернет;

– легка інтеграція з системами розумного будинку, такими як Home Assistant, що дозволяє створювати більш інтелектуальні та зручні системи відеонагляду;

– дана мікроплата має низьке споживання енергії, що робить його ідеальним для тривалого безперервного використання;

– велика спільнота розробників та користувачів, яка надає підтримку, документацію та численні проекти, які можна використовувати як основу або для натхнення.

Ці переваги зробили для мене Raspberry Pi 5 відмінним вибором для основи для створення ефективної та надійної системи відеонагляду.

2.2 Перелік існуючих рішень та обґрунтування вибору модуля камери Raspberry Pi Camera Module 2

Вибір модуля камери для Raspberry Pi є важливим кроком у створенні системи відеонагляду. Це не лише про технічні характеристики, а й про визначення потреб даного проекту. Ось декілька ключових факторів, які слід врахувати при виборі модуля камери:

– роздільна здатність та якість зображення;

– кут огляду;

– світлочутливість;

– сумісність з Raspberry Pi.

Враховуючи ці фактори, можна зробити обґрунтований вибір, який найкраще відповідає потребам у системі відеонагляду [22]. Серед модулів камери які будуть сумісні із обраною Raspberry Pi 5 можна виділити наступні:

– Raspberry Pi Camera Module 3.

– Raspberry Pi Camera Module 2.

– Raspberry Pi High Quality Camera (HQ Camera).

– Raspberry Pi Global Shutter Camera.

Враховуючи всі вищезазначені параметри, для даного проекту було вирішено обрати Raspberry Pi Camera Module 2. Цей вибір можна обґрунтувати наступними перевагами:

– модуль Camera Module 2 забезпечує високу роздільну здатність, що дозволяє отримувати чіткі зображення, важливі для ідентифікації об’єктів або осіб;

– його малий розмір робить модуль ідеальним для використання в обмежених просторах або для створення дискретних систем відеонагляду;

– цей модуль камери легко інтегрується з Raspberry Pi, що спрощує налаштування та розробку системи;

– велика спільнота користувачів та розробників Raspberry Pi надає численні ресурси, які можуть допомогти у вирішенні проблем та покращенні проекту.

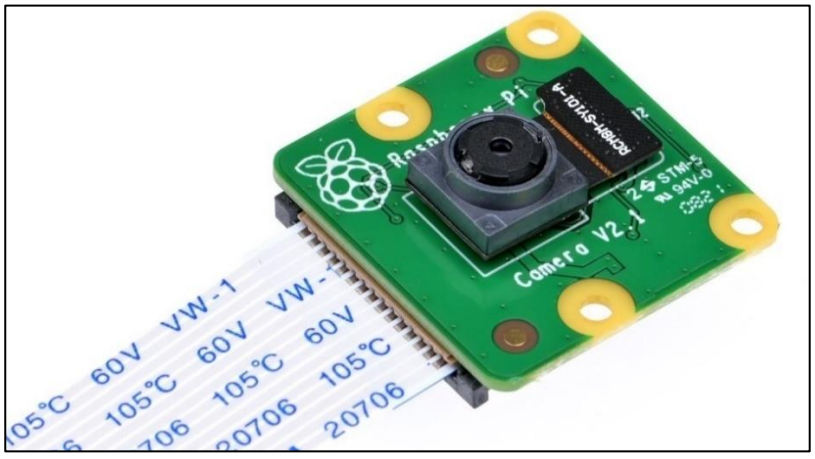


Рисунок 2.5 – Модуль камери Raspberry Pi Camera Module 2

Ці переваги роблять Raspberry Pi Camera Module 2 відмінним вибором камери для створення ефективної та надійної системи відеонагляду.

2.3 Перелік існуючих рішень та обґрунтування вибору середовища програмування та програмного забезпечення для плати Raspberry Pi 5

Перед тим як поглиблюватися у програмування Raspberry Pi 5, давайте розглянемо різноманітні мови програмування, які можуть бути використані у цій роботі. Кожна з них має свої переваги та застосування.

Python - це перша мова програмування, яка приходить на думку, коли йдеться про Raspberry Pi. Вона встановлена за замовчуванням у Raspberry Pi OS і має відмінну підтримку з усіма апаратними компонентами екосистеми (GPIO, камера, HATs тощо). Python також є чудовим вибором для початківців, оскільки він близький до природної мови, легко читається та не надто складно пишеться. Він має багато бібліотек для різних завдань, що дозволяє створювати широкий спектр проектів.

C/C++ - це мови програмування, які також підтримуються Raspberry Pi. Вони дозволяють писати ефективний та швидкий код, що особливо важливо для деяких завдань, таких як обробка сигналів, робототехніка та інші.

Java також можна використовувати на Raspberry Pi. Вона підтримується на різних платформах та дозволяє розробляти різноманітні додатки.

JavaScript можна використовувати для веб-розробки та інших завдань.

Scratch - це проста мова програмування, яка призначена для навчання дітей та молоді. Вона використовується для створення інтерактивних історій, ігор та анімацій.

HTML/CSS можна використовувати для веб-розробки та створення веб-інтерфейсів.

PyQt та Tkinter - це популярні бібліотеки Python для створення графічних інтерфейсів. Вони прості у використанні та мають всі необхідні функції для

розробки інтерфейсів на Raspberry Pi.

Серед важливих рішень, які будуть використані у цьому проекті варто виділити OpenCV (Open Computer Vision Library) - це найбільша бібліотека комп'ютерного зору у світі. OpenCV є відкритим і доступним для всіх та підтримує мови програмування, такі як Python, C++, Java, JavaScript та інші. Цей інструмент надає різноманітні рішення для роботи з зображеннями, відео, обробки сигналів, розпізнавання обличь, машинного навчання та багато іншого [27].

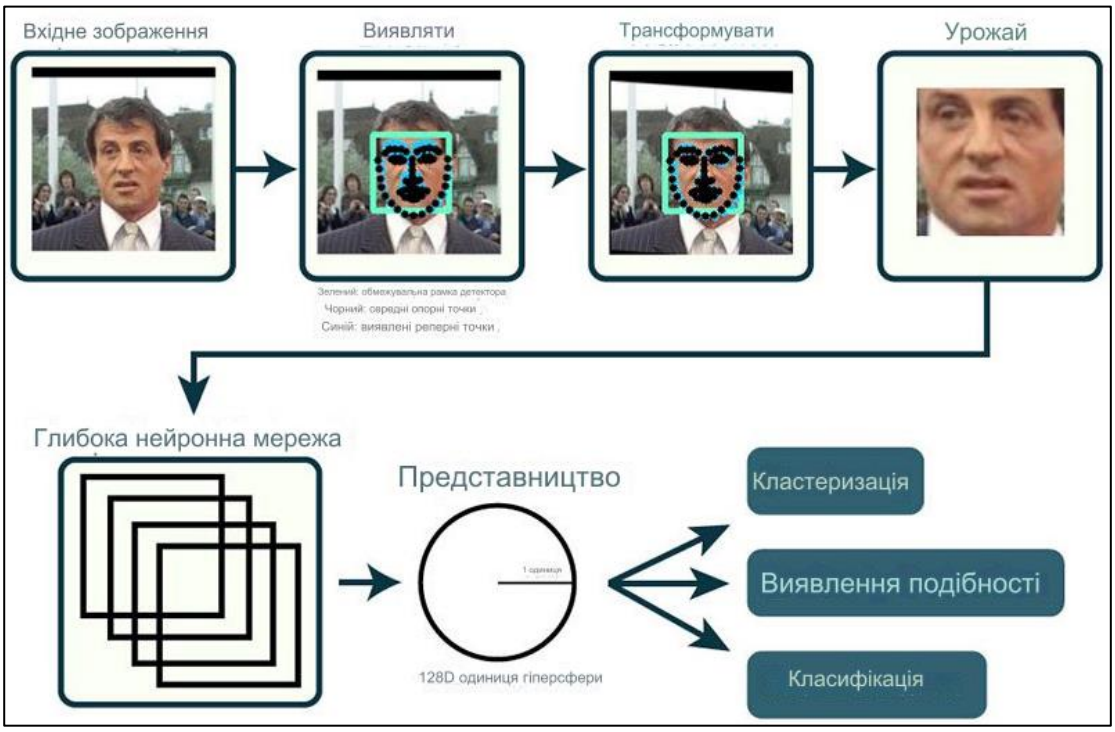


Рисунок 2.6 – Алгоритм розпізнавання OpenCV

Іншим важливим інструментом у цій роботі виступає PiCamera - це бібліотека Python, яка дозволяє керувати модулем камери Raspberry Pi. Вона надає чистий Python інтерфейс до модуля камери Raspberry Pi для версій Python 2.7 (або вище) або Python 3.2 (або вище). Загалом це потужний інструмент для роботи з камерою Raspberry Pi, який дозволяє створювати різноманітні проекти з обробки зображень та відеонагляду.

Серед переліку доступних мов програмування та інструментів мною було обрано наступні:

– Python є найкращим вибором для Raspberry Pi з багатьма перевагами, він

встановлений за замовчуванням у Raspberry Pi OS а також легкий для вивчення, з читабельним синтаксисом, має багато бібліотек для роботи з GPIO, камерою, датчиками та іншими компонентами;

– OpenCV - найбільша бібліотека комп’ютерного зору, вона дозволяє обробляти зображення, відео, розпізнавати обличчя та об’єкти, чим ідеально підходить для систем відеонагляду;

– PiCamera - бібліотека Python для керування модулем камери Raspberry Pi, вона проста у використанні та дозволяє знімати відео та фотографії.

2.4 Підключення плати Raspberry Pi 5 і модуля камери Raspberry Pi Camera Module 2 та тестування їх спільної роботи

Для роботи повністю функціонуючої системи нам потрібно приєднати Raspberry Pi Camera Module 2 до Raspberry Pi 5 за допомогою спеціального CSI FPC (22pin to 15pin, 200мм) шлейфу.

Рисунок 2.7 – CSI FPC (22pin to 15pin, 200мм) шлейф

Особливістю цього шлейфу є те що він перетворює сигнал із 22 пін на сигнал

у 15 пін, тобто виступає адаптером у цьому підключені, схему будови цього адаптеру зображено на рисунку 2.8.

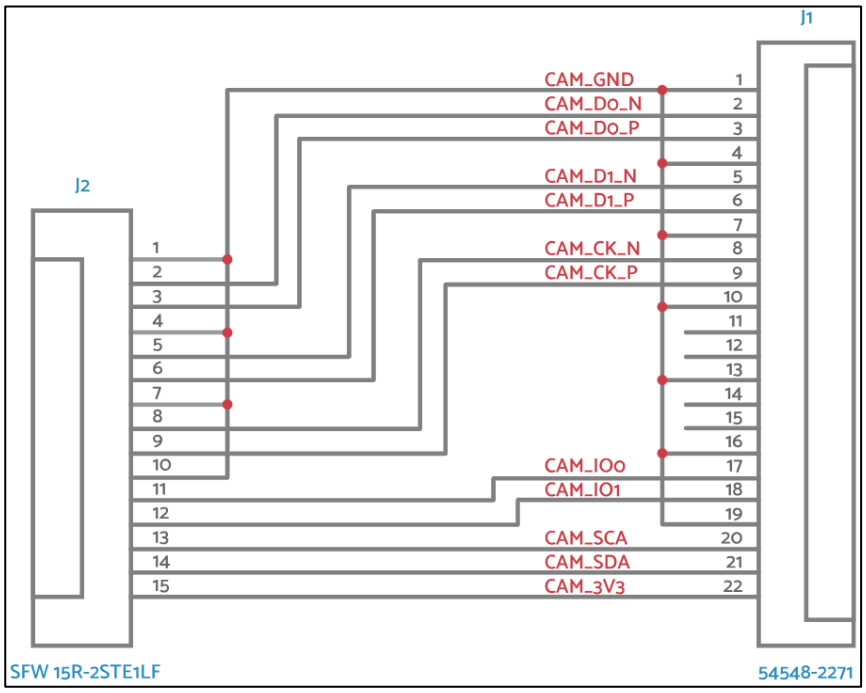


Рисунок 2.8 – Схема будови адаптеру CSI FPC 22pin to 15pin

Край шлейфу на 22 піни вставляється у саму камеру а інший край на 15 пінів у спеціальне гніздо для підключення камери чи дисплею на самій Raspberry PI як показано на схемі рисунку 2.9 [21].

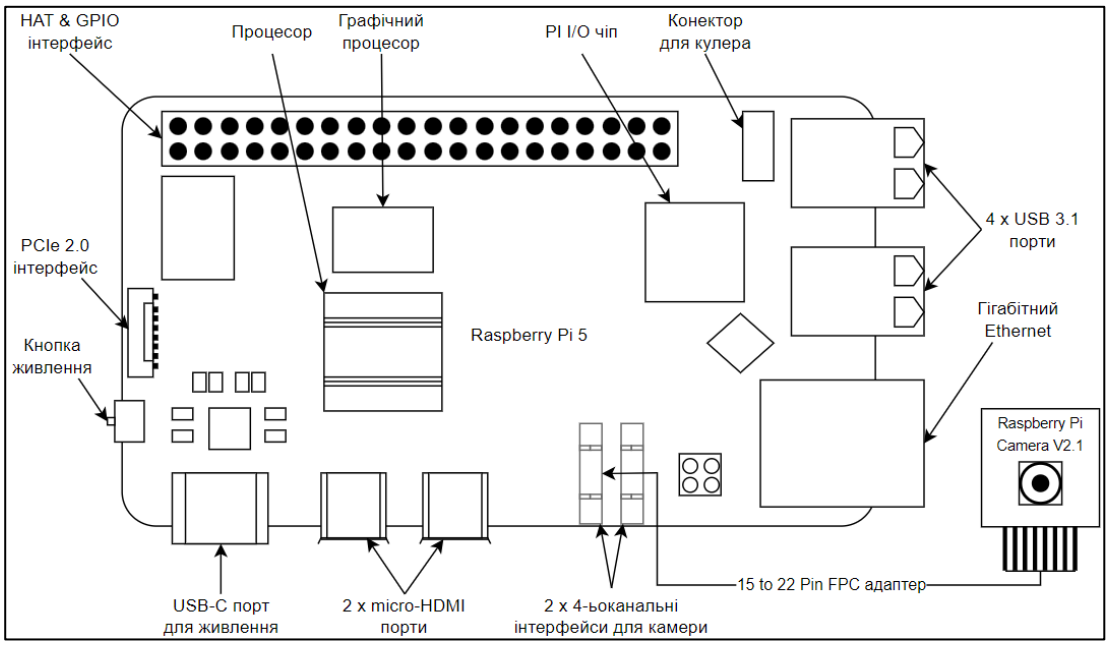


Рисунок 2.9 – Схема підключення камери до мікроплати Raspberry Pi

Щоб розглянути дане підключення детальніше, модна звернутись до рисунку 2.10

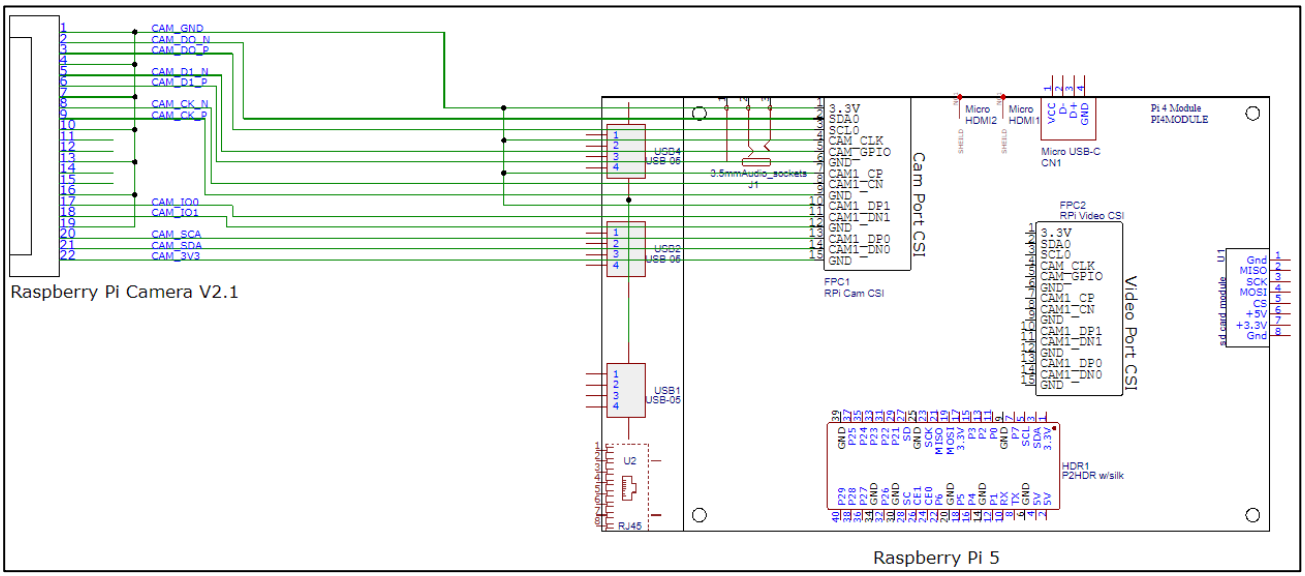


Рисунок 2.10 – Схема детального підключення камери до мікроплати

На мікроплаті Raspberry PI 5 є два таких гнізда, тому шлейф камери я підключив до гнізда DISP/CAM0, як і зображено на рисунку 2.11.

Рисунок 2.11 – Підключена камера до мікроплати

2.5 Висновок

Важливість мікроплати Raspberry Pi 5 у системі відеонагляду не можна недооцінювати. Ця мікроплата служить як мозок системи, керуючи збором даних з камери, їх обробкою та зберіганням. Вона забезпечує необхідну обчислювальну потужність для виконання складних завдань, таких як аналіз відео в реальному часі, розпізнавання об’єктів або інтеграцію з іншими системами. В свою чергу, модуль камери є важливою частиною Raspberry Pi, яка дозволяє захоплювати відео та фотографії. Різні моделі мають різні характеристики, такі як роздільна здатність, кут огляду та можливості зйомки в умовах недостатнього освітлення. Середовища програмування, такі як Python, OpenCV та PiCamera, дозволяють створювати різноманітні проекти з обробки зображень та відеонагляду на Raspberry Pi. Обравши ці інструменти, ми отримуємо потужний інструментарій для розробки системи відеонагляду.

Сама взаємодія між мікроплатою та камерою є ключовою для успішної реалізації системи відеонагляду. Мікроплата Raspberry Pi 5 з її високою продуктивністю та широкими можливостями підключення ідеально підходить для цього завдання. Вона може одночасно підтримувати кілька модулів камер, забезпечуючи гнучкість у виборі кута огляду та роздільної здатності. Також, завдяки підтримці різноманітних мов програмування та бібліотек, як-от Python, OpenCV та PiCamera, розробники мають свободу вибору інструментів для створення потужних та ефективних систем відеонагляду.

Враховуючи це, можна зробити висновок, що мікроплата Raspberry Pi 5 разом з відповідно підібраним модулем камери та програмним забезпеченням становить основу для створення високоефективної системи відеонагляду, яка може бути адаптована до різних потреб та середовищ.

3 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ РОЗУМНОЇ СИСТЕМИ НА ОСНОВІ МІКРОПЛАТИ ДЛЯ ВЕДЕННЯ ВІДЕОНАГЛЯДУ

3.1 Встановлення середовища Raspberry Pi OS та його налаштування для роботи зі всіма підключеними компонентами плати

Спочатку нам потрібно встановити операційну систему та програмне забезпечення на Raspberry Pi перед тим, як збирати все разом. Операційна система — це базове програмне забезпечення, яке керує апаратним забезпеченням Raspberry Pi. Linux ідеально підходить для цього завдання. Я обрав Raspberry Pi OS, оскільки це одна з найбільш розвинутих операційних систем для Raspberry Pi, до того ж в інтернеті є багато підтримки та навчальних матеріалів.

Нам необхідно підготувати SD-карту, щоб запустити Raspberry Pi OS на Raspberry Pi. Для цього знадобиться карта пам’яті із мінімальним об’ємом в 8 GB, я вирішив обрати карту пам’яті на 64 GB (рисунок 3.1), адже для збереження відеозаписів буде необхідно більший об’єм пам’яті також можна додати до системи зовнішній HDD чи SSD накопичувач, але в даному випадку я використовую збірку без нього [26, 27].

Рисунок 3.1 – Карта пам’яті Kodak та адаптер для її підключення до пк

Тепер коли є обрана SD-карта, наступним кроком буде встановлення операційної системи на SD-карту. Раніше це було дещо складно, особливо для початківців, через різні основні операційні системи (macOS, Windows, Ubuntu тощо) та їхні різні методи доступу до SD-карти. Тепер існує новий інструмент під назвою Raspberry Pi Imager, який виконує всю складну роботу за користувача [31].

Після завантаження програмного забезпечення, відкриваєм додаток. Спочатку оберем операційну систему, яку ви маєм встановити на SD-карту. У даному випадку вибір впав на Raspberry Pi OS, але є і чимало інших варіантів. Далі вставляєм SD-карту в комп'ютер та вибираєм її у додатку. Коли все обрано правильно натискаємо на «Записати», після чого почнеться процес завантаження, запису та перевірки OS на карту пам'яті, як це зображено на рисунку 3.2.

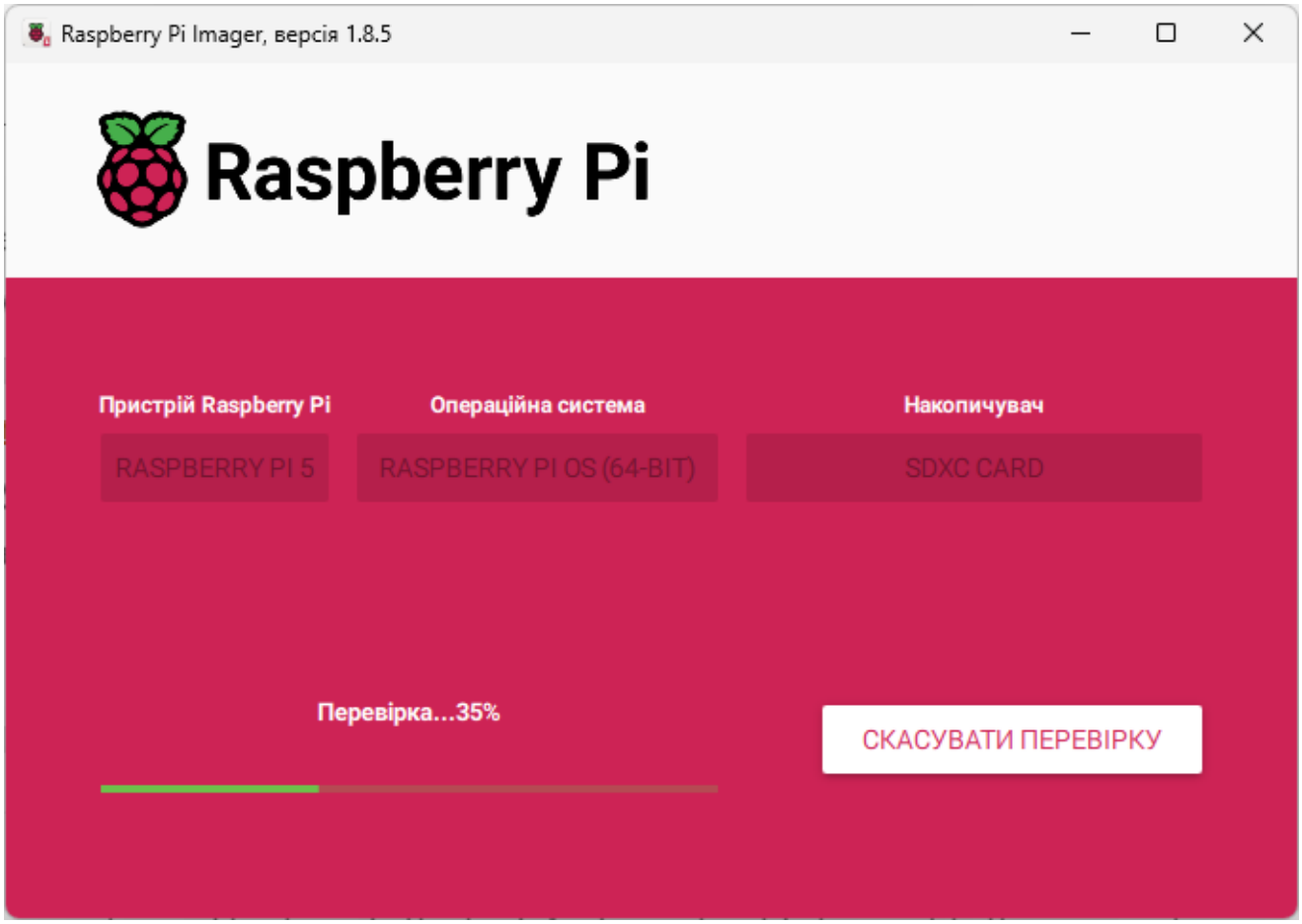


Рисунок 3.2 – Запис операційної системи на SD картку

Якщо весь процес пройшов успішно то ми почуєм звуковий сигнал а також програма напише про успішне завершення форматування SD картки та

встановлення на неї образу операційної системи, як показано на рисунку 3.3.

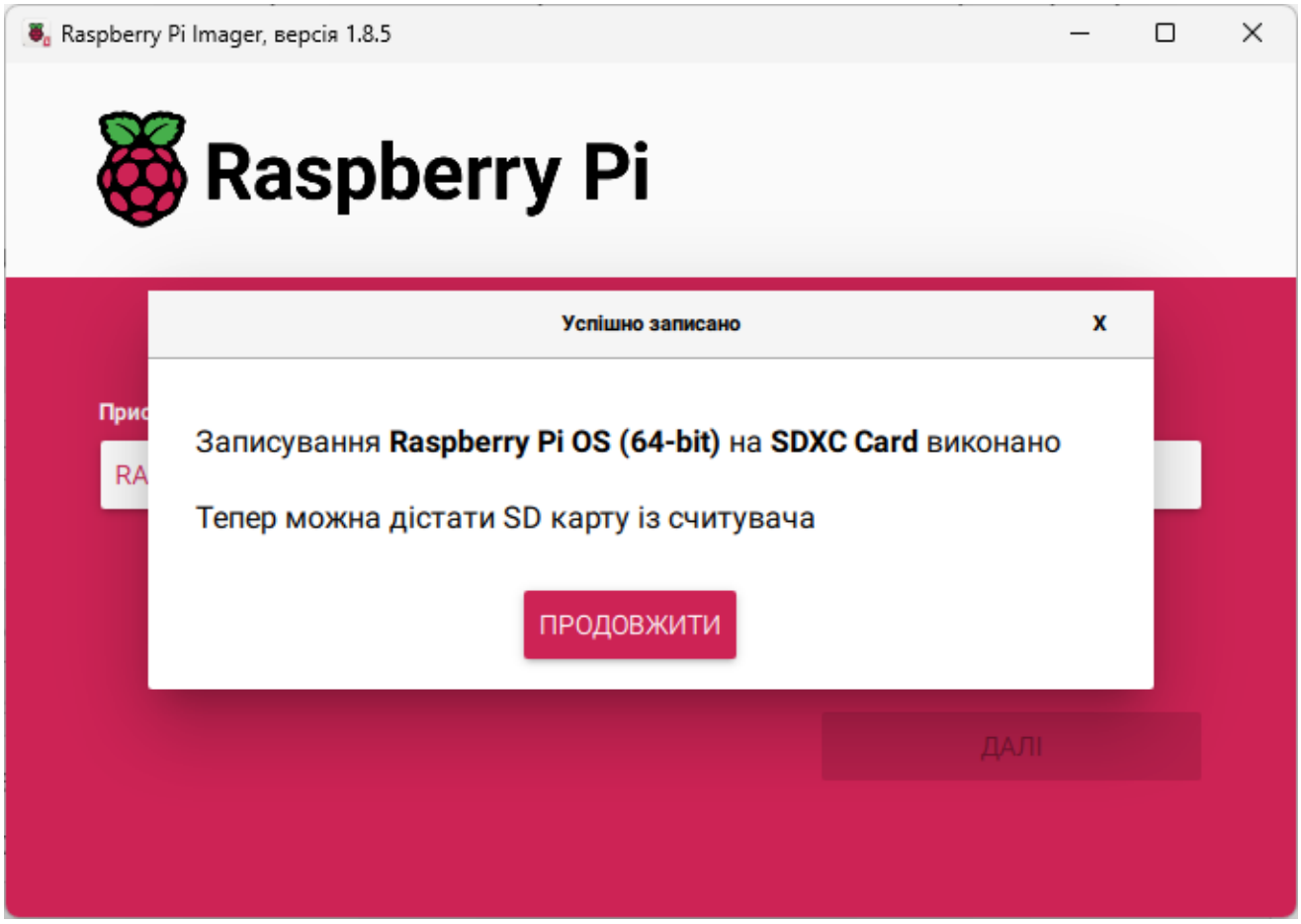


Рисунок 3.3 – Вікно успішного завершення встановлення ОС

Тепер тимчасово підключаємо нашу плату Raspberry Pi до LAN-кабелю, монітора (HDMI to micro HDMI) та USB-клавіатури для базового налаштування. Вставляємо підготовлену SD-карту з інсталятором Raspberry Pi OS та підключаємо блок живлення. Raspberry Pi має завантажитися та провести нас через процес налаштування. Після цього у нас має бути запущена базова операційна система Raspberry Pi OS. Варто також увімкнути SSH у Raspberry Pi OS, щоб ми могли керувати пристроєм Raspberry Pi, навіть коли до нього не підключені монітор та клавіатура. Також слід увімкнути камеру з меню налаштувань Raspberry Pi OS, щоб ми могли використовувати її для виявлення руху та відеоспостереження [33, 35]].

Тепер, переконаємось, що наш Raspberry Pi 5 оновлено до останньої версії, виконаємо в терміналі, команди що зображено на рисунку 3.4.

```
# Встановлення інструменту rpi-update для оновлення прошивки на Raspberry Pi
sudo apt-get install rpi-update

# Виконання оновлення прошивки до останньої версії
sudo rpi-update

# Оновлення списку доступних пакетів та їх версій
sudo apt update

# Оновлення всіх встановлених пакетів до останніх версій
sudo apt upgrade

# Оновлення EEPROM (електрично стираємої програмованої постійної пам'яті) до останньої версії
sudo rpi-eeprom-update -a
```

Рисунок 3.4 – Команди для оновлення програмного забезпечення плати

Ці команди дозволять оновити нашу систему та EEPROM1. Останнє (на момент виконання цієї роботи) оновлення EEPROM від 17 квітня 2024 року включає важливі поліпшення, такі як підвищення продуктивності, зміцнення безпеки та кращу сумісність з апаратним забезпеченням. Після виконання цих команд перезавантажуємо наш Raspberry Pi, щоб зміни вступили в силу.

У моїй роботі я здійснив базове налаштування та забезпечив можливість підключення до Raspberry Pi з мого комп’ютера. Це дозволяє мені керувати консоллю Linux з будь-якого пристрою у моїй локальній мережі, ніби я працюю безпосередньо з Raspberry Pi. Такий підхід є ключовим, адже коли Raspberry Pi встановлено на відстані від робочого місця, я маю можливість оновлювати систему та змінювати конфігурацію камери в будь-який час, не від’єднуючи її від стіни.

Варто зазначити, що ця камера — це не просто камера, але й потужний комп’ютер з операційною системою Linux, що відкриває широкі можливості для розширення функціоналу. Моя система не обмежена лише тим, що описано у цій роботі; вона здатна адаптуватися до нових розробок програмного забезпечення, дозволяючи мені встановлювати оновлення та додаткові модулі у майбутньому.

Для підключення до консолі Linux на Raspberry Pi я використовую програму PuTTY, яка є безкоштовною та доступною для завантаження з офіційного сайту. Після встановлення PuTTY я можу підключитися до Raspberry Pi, і з цього

моменту мені більше не потрібні монітор та клавіатура для роботи з пристроєм [30].

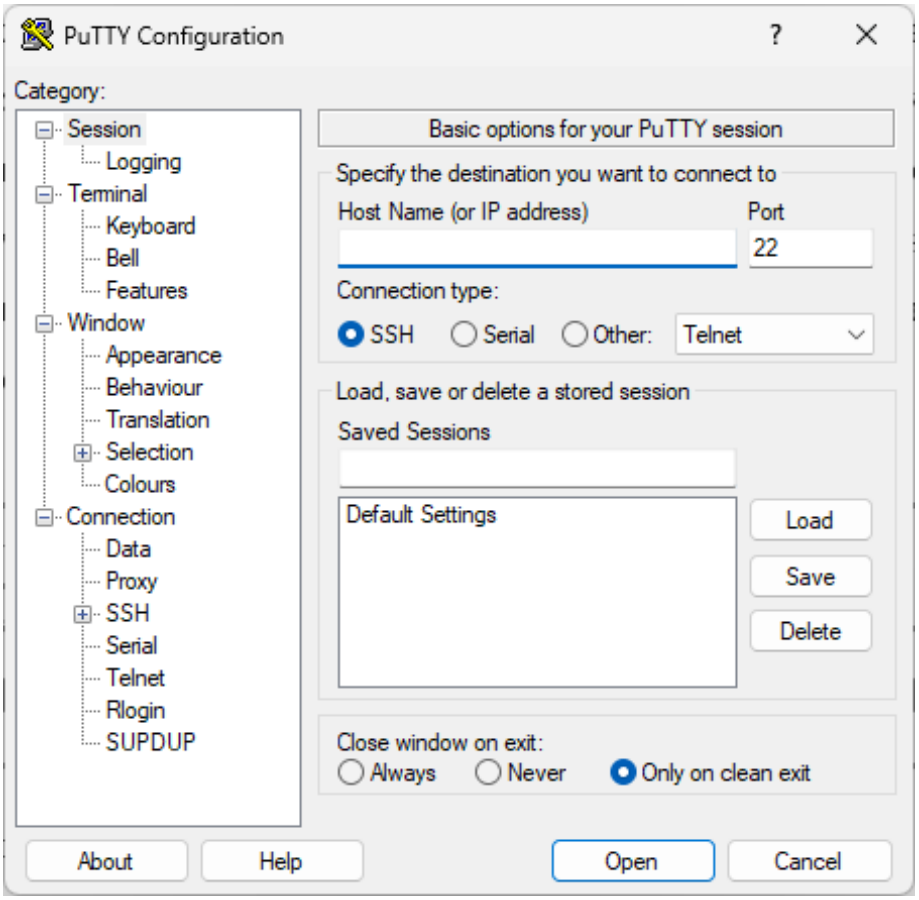


Рисунок 3.5 – Конфігурація відділеного підключення в PuTTY

Наступним кроком необхідно буде налаштувати Wi-Fi мережу для того щоб наша плата могла працювати без підключеної дротової мережі. Для цього з консолі (вікна PuTTY), я редагую мережеві налаштування Raspberry Pi за допомогою команди «sudo nano /etc/network/interfaces». Для налаштування мережевих властивостей Raspberry Pi, додаємо рядки, як заображено на рисунку 3.6 та рисунку 3.7, в кінець файлу /etc/network/interfaces або змінюєм існуючі рядки так, щоб вони відповідали цим.

```
auto eth0
iface eth0 inet static
    address 192.168.1.100
    netmask 255.255.255.0
    gateway 192.168.1.1
    dns-nameservers 8.8.8.8 8.8.4.4
```

Рисунок 3.6 – Налаштування інтерфейсу для використання статичної IP-адреси

```
# Цей рядок дозволяє системі реагувати на підключення бездротового інтерфейсу wlan0
allow-hotplug wlan0

# Налаштування інтерфейсу wlan0 для автоматичного отримання IP-адреси через DHCP
iface wlan0 inet dhcp

# Вказівка SSID вашої бездротової мережі
wpa-ssid "YOUR NETWORK SSID"

# Введення пароля для підключення до бездротової мережі
wpa-psk "WIFI PASSWORD"
```

Рисунок 3.7 – Налаштування бездротової мережі

Ці рядки конфігурації налаштовують бездротову мережу та встановлюють статичну IP-адресу для інтерфейсу eth0, вказують маску підмережі, шлюз за замовчуванням та DNS-сервери. Після додавання або зміни рядків, зберігаємо зміни у файлі, натиснувши Ctrl+O та виходимо з редактора nano, натиснувши Ctrl+X. Щоб застосувати нові налаштування, перезавантажуємо мережевий інтерфейс або Raspberry Pi.

Таким чином наша Raspberry Pi налаштована та готова до початку роботи із конфігурацією її як камери відеонагляду.

3.2 Встановлення та налаштування системи реагування на рух Motion

Дуже хорошим (та безкоштовним з відкритим кодом) програмним забезпеченням для виявлення руху та відеонагляду, яке має багато опцій налаштування, є програма Motion. Вона пропонує широкий спектр можливостей для конфігурації та адаптації до різних сценаріїв використання, що робить її ідеальним вибором для систем відеонагляду.

Програми Motion та MotionPlus - це високоналаштовувані системи, які стежать за відеосигналами з різних типів камер і, в залежності від їх конфігурації, виконують дії при виявленні руху. Motion - це оригінальне програмне забезпечення, створене ще у 2000 році, тоді як MotionPlus - це новий продукт. MotionPlus було розроблено як новий додаток, щоб дозволити видалення старих,

рідко використовуваних функцій, а також додавання нових можливостей.

До можливостей цих програм належать:

– створення відео або збереження знімків активності;

– пряме записування з багатьох IP-камер;

– перегляд потокового відео з камер у реальному часі;

– виклик скриптів при виявленні активності;

– запис активності у різні типи баз даних;

– повністю налаштовувані маски для конфіденційності або виявлення руху;

– повна підтримка tls(https) з аутентифікацією для веб-контролю та потоків.

Сумісна з багатьма типами пристроїв, таких як:

– мережеві камери через RTSP, RTMP та HTTP;

– камери PI;

– веб-камери V4L2;

– відеокарти захоплення;

– існуючі відеофайли.

Ці програми надають широкі можливості для створення систем відеонагляду, які можуть бути точно налаштовані під конкретні потреби користувача.

Для початку роботи буде необхідно встановити цю бібліотеку програм в терміналі за допомогою команди «sudo apt-get install motion»

У процесі створення моєї системи відеонагляду я встановив необхідні пакети, використовуючи команду «y» для продовження інсталяції. Оскільки поточна версія програми Motion ще не підтримує модуль камери Raspberry, мені довелося завантажити та встановити спеціальну збірку, яка підтримує цей модуль камери [35, 36].

Після завантаження файлу я розпакував його у директорію /tmp – «tar zxvf motion-mmal.tar.gz».

Далі я оновив встановлену версію Motion за допомогою завантаженої збірки, як зображено на рисунку 3.8

```
# Ця команда переміщує виконавчий файл 'motion' до директорії /usr/bin,
# що є стандартним місцем для системних виконавчих файлів
sudo mv motion /usr/bin/motion

# Ця команда переміщує файл конфігурації 'motion-mmalcam.conf' до /etc/motion.conf,
# замінюючи стандартний файл конфігурації Motion на спеціалізований для модуля камери MMAL
sudo mv motion-mmalcam.conf /etc/motion.conf
```

Рисунок 3.8 – Переміщення конфігурації Motion.

Також я активував Motion daemon, щоб програма Motion завжди працювала командою «sudo nano /etc/default/motion» і змінив рядок на: «start_motion_daemon=yes»

Я впевнений, що офіційна збірка Motion незабаром також підтримуватиме модуль камери Raspberry. Для редагування файлу конфігурації Motion я використовував команду «sudo nano /etc/motion.conf»

Зауважу, що у стандартній інсталяції Motion файл motion.conf знаходиться в /etc/motion/, але у спеціальній збірці motion-mmal він знаходиться в /etc/.

Я також переконався, що права доступу до файлів встановлені правильно: коли я встановлював Motion через ssh, будучи залогіненим як користувач «pi», я мав переконатися, що надав користувачу “motion” права для запуску Motion як сервісу після перезавантаження, що зображено на рисунку 3.9.

```
# Змінює права доступу до файлу конфігурації Motion, дозволяючи власнику та групі читати та редагувати файл,
# а іншим користувачам - лише читати
sudo chmod 664 /etc/motion.conf

# Змінює права доступу до виконавчого файлу Motion, дозволяючи власнику та групі читати та виконувати файл,
# а іншим користувачам - лише читати
sudo chmod 755 /usr/bin/motion

# Створює новий файл журналу для Motion, якщо він не існує
sudo touch /tmp/motion.log

# Змінює права доступу до файлу журналу Motion, дозволяючи власнику та групі читати, редагувати та виконувати файл,
# а іншим користувачам - читати та виконувати
sudo chmod 775 /tmp/motion.log
```

Рисунок 3.9 – Зміна прав доступу для Motion

Було зроблено ряд змін до основного конфігураційного файлу Motion, з метою щоб налагодити кращу продуктивність проекту, ці зміну будуть описані далі.

У рамках мого проекту я забезпечив, щоб програма Motion завжди працювала в фоновому режимі як daemon «daemon on» [36].

Лог-файл я вирішив зберігати в директорії /tmp, щоб користувачі з автозапуском мали до нього доступ, а не в папці /home/pi/ «logfile /tmp/motion.log».

Для забезпечення високоякісного відеоспостереження я встановив роздільну здатність 1280x720: «width 1280», «height 720».

Реальний час відео мені не потрібен, достатньо 2 зображень за секунду для наших потреб «framerate 2».

Також є ще одна дуже зручна функція програми Motion – це записувати деякі (у нашій конфігурації 2) кадри до та після виявлення руху на зображенні: «pre_capture 2», «post_capture 2».

Я не хочу нескінченних відео. Натомість, я хочу мати максимум 10-хвилинні частини відео з рухом. Ця опція конфігурації була перейменована з «max_movie_time» на «max_mpeg_time» у Motion. При використанні збірки «motion-mmal», ця опція працюватиме. Але якщо виникатиме помилка «Unknown config option max_mpeg_time», треба змінити цю опцію на «max_movie_time» або переконатися, що зо дійсно використовується збірка «motion-mmal», як було вказано мною раніше. Це пункт конфігурації «max_mpeg_time 600».

Оскільки деякі медіаплеєри, такі як VLC, не можуть відтворювати записані фільми, я змінив кодек на msmpeg4. Тоді фільми коректно відтворюються у всіх плеєрах: «ffmpeg_video_codec msmpeg4».

Я дозволив доступ до прямої трансляції з будь-якого місця. Інакше лише локальний хост (= пристрій Raspberry) мав би доступ до прямої трансляції: «stream_localhost off».

Для того щоб захистити захистити пряму трансляцію за допомогою імені користувача та пароля, слід увімкнути наступні конфігураційні пункти:

– stream_auth_method 2;

– stream_authentication (логін) (пароль).

Всі параметри конфігурації детально пояснені в документації конфігурації Motion. Після змін у файлі motion.conf, необхідно перезавантажити Raspberry Pi командою «sudo reboot».

Після перезавантаження червоне світло модуля камери має бути увімкнене, що показує, що програма Motion використовує камеру для виявлення будь-якого руху.

У моєму проекті я вирішив використовувати обмежені ресурси SD-карти Raspberry Pi більш ефективно, тому дозволив Raspberry Pi зберігати відео на одному з наших Windows-серверів. Це було досить просто:

Спочатку я поділився папкою з Windows-машини. Потім я відкрив конфігурацію fstab на Raspberry Pi через консоль PuTTY або безпосередньо з пристрою «sudo nano /etc/fstab».

Додав додатковий рядок з конфігурацією моєї папки, яка спільно використовується в мережі Windows:

«//(назва серверу) /(назва поширеної папки на сервері) /mnt/camshare cifs username=(ім’я користувача), password=(пароль), iocharset=utf8, file_mode=0777, dir_mode=0777 0 0»

Я переконався, що користувач має відповідні права для збереження файлів у цій спільній папці.

Після перезавантаження Raspberry Pi мала додаткову папку /mnt/camshare, яка була підключена до спільної папки Windows. Тепер у файлі motion.conf я встановив «target_dir /mnt/camshare» таким чином, програма Motion зберігає всі відео у спільній папці на Windows-машині.

У процесі роботи над моїм проектом я зіткнувся з проблемою що програма Motion не запускалася автоматично після перезавантаження Raspberry Pi. Я виявив, що це сталося тому, що підключена папка з Windows-машини ще не була готова в момент, коли Motion намагалася отримати до неї доступ.

Проблему вдалося швидко вирішити: Я просто відредагував файл Motion за

допомогою команди «sudo nano /etc/init.d/motion» і туди додав рядок «sleep 30» в початок сценарію запуску. Це дозволило забезпечити достатньо часу для підготовки папки перед тим, як Motion спробує звернутися до неї, і вирішило проблему з автозапуском.

В результаті в нас буде повністю функціонувати автоматичне реагування на рух та запис відео у нашій системі відеонагляду.

3.3 Встановлення та налаштування системи ідентифікації обличчя OpenCV

Щоб розпочати роботу з бібліотекою OpenCV на Raspberry Pi OS, мені потрібно було виконати кілька кроків для її встановлення. Процес встановлення включає в себе використання командного рядка для введення специфічних команд, які дозволять завантажити та скомпілювати бібліотеку на моєму пристрої. Ці команди зазвичай включають оновлення списку пакетів, встановлення необхідних залежностей, завантаження самої бібліотеки OpenCV з репозиторію, а також її компіляцію та інсталяцію.

Для встановлення OpenCV на Raspberry Pi OS я використав перелік команд, зображений на рисунку 3.10.

```
# Встановлення CMake, інструментів для збірки проектів, pkg-config для виявлення бібліотек та Git для управління версіями
sudo apt install cmake build-essential pkg-config git

# Встановлення розробницьких бібліотек для роботи з форматами зображень JPEG, TIFF, Jasper, PNG, WebP та OpenEXR
sudo apt install libjpeg-dev libtiff-dev libjasper-dev libpng-dev libwebp-dev libopenexr-dev

# Встановлення розробницьких бібліотек для роботи з відео кодеками та форматами, а також інтерфейсом камери V4L2
sudo apt install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev libxvidcore-dev libx264-dev
    libdc1394-22-dev libgstreamer-plugins-base1.0-dev libgstreamer1.0-dev

# Встановлення бібліотек для розробки графічного інтерфейсу користувача GTK+ 3 та Qt
sudo apt install libgtk-3-dev libqtgui4 libqtwebkit4 libqt4-test python3-pyqt5

# Встановлення бібліотек для наукових обчислень, таких як ATLAS, LAPACKE та компілятора Fortran
sudo apt install libatlas-base-dev liblapacke-dev gfortran

# Встановлення бібліотек для роботи з форматом даних HDF5
sudo apt install libhdf5-dev libhdf5-103

# Встановлення інструментів для розробки на Python 3, включаючи PIP та бібліотеку NumPy для наукових обчислень
sudo apt install python3-dev python3-pip python3-numpy
```

Рисунок 3.10 – Команди для встановлення OpenCV

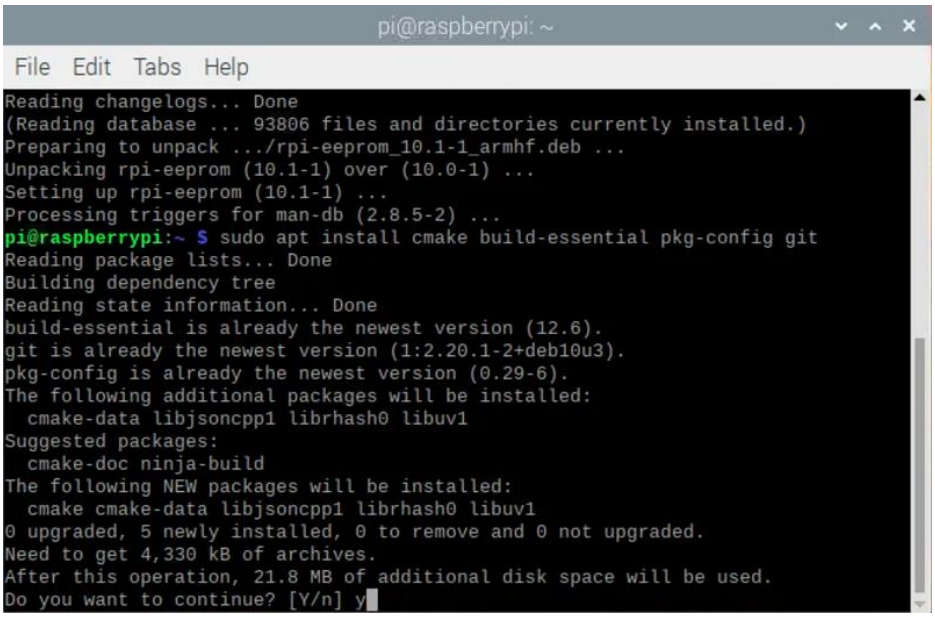


Рисунок 3.11 – Процес встановлення OpenCV

У процесі встановлення пакетів для системи розпізнавання облич на Raspberry Pi я вирішив тимчасово збільшити файл підкачки перед виконанням наступних команд. Для цього я відкрив файл dphys-swapfile для редагування командою «sudo nano /etc/dphys-swapfile». Коли файл був відкритий, я закоментував рядок «CONF_SWAPSIZE=100» і додав «CONF_SWAPSIZE=2048». Потім натиснув Ctrl-X, Y і Enter, щоб зберегти зміни у файлі dphys-swapfile [34].

Це зміна була лише тимчасовою, я повернувся до попередніх налаштувань після завершення встановлення OpenCV.

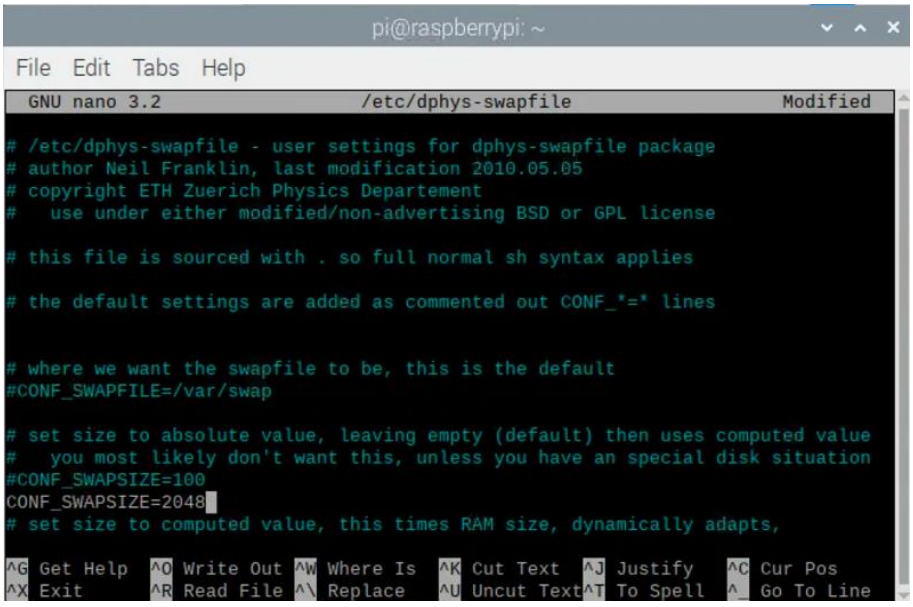


Рисунок 3.12 – Збільшення розміру файлу підкачки

Щоб наші зміни набули чинності, нам потрібно перезапустити файл підкачки за допомогою команди «sudo systemctl restart dphys-swapfile». Ця дія забезпечить використання нових налаштувань файлу підкачки, які я вніс раніше. Після виконання цієї команди система почне використовувати збільшений файл підкачки.

Після чого можна продовжити інсталяцію всіх необхідних бібліотек командами, як зображено на рисунку 3.13

```
# Клонування основного репозиторію OpenCV з GitHub для отримання вихідного коду
git clone https://github.com/opencv/opencv.git

# Клонування додаткового репозиторію OpenCV (opencv_contrib) з GitHub, який містить додаткові модулі
git clone https://github.com/opencv/opencv_contrib.git

# Створення нової папки 'build' у домашньому каталозі opencv для збірки проекту
mkdir ~/opencv/build

# Зміна поточного каталогу на створену папку 'build'
cd ~/opencv/build

# Налаштування CMake для збірки OpenCV з вказанням типу збірки, шляху встановлення,
# шляху до додаткових модулів, оптимізації для ARM архітектури (NEON, VFPV3),
# відключення тестів, прикладів Python, включення не вільних алгоритмів (non-free),
# налаштування лінкера та відключення збірки прикладів
cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \
-D ENABLE_NEON=ON \
-D ENABLE_VFPV3=ON \
-D BUILD_TESTS=OFF \
-D INSTALL_PYTHON_EXAMPLES=OFF \
-D OPENCV_ENABLE_NONFREE=ON \
-D CMAKE_SHARED_LINKER_FLAGS=-latomic \
-D BUILD_EXAMPLES=OFF ..

# Запуск процесу збірки з використанням всіх ядер процесора для прискорення збірки
make -j$(nproc)

# Встановлення зібраної версії OpenCV у систему
sudo make install

# Оновлення конфігурації лінкера, щоб він міг знайти ново встановлені бібліотеки OpenCV
sudo ldconfig
```

Рисунок 3.13 – Продовження інсталяції OpenCV

Після успішного встановлення OpenCV я повернув файл підкачки до його початкового стану. Для цього у терміналі я ввів «sudo nano /etc/dphys-swapfile». Коли файл був відкритий, я розкоментував «CONF_SWAPSIZE=100» і

закоментував «CONF_SWAPSIZE=2048». Натиснувши Ctrl-X, Y і потім Enter, я зберіг зміни у файлі dphys-swapfile. Потім я знову перезапустив файл підкачки за допомогою команди «sudo systemctl restart dphys-swapfile».

Для встановлення пакету face_recognition використано команду «pip install face-recognition», та для встановлення imutils команду «pip install imutils».

В процесі у мене виникли помилки з повідомленням “No module named imutils” та “No module named face-recognition”, для їх виправлення я просто знову встановив ці бібліотеки використавши pip2 замість pip [32].

Для тренування моделі розпізнавання обличчя з використанням Raspberry Pi та OpenCV, є наступні вимоги:

– підготовка даних - зробити набір зображень обличь для створення датасету, даний експеримент проводиться на мені тому це мають бути мої фото з різних ракурсів та в різних освітленнях;

– встановлення необхідних пакетів - OpenCV, face_recognition та imutils на Raspberry Pi;

– використання скрипту train_model.py для аналізу зображень у моєму датасеті та створення відповідності між іменами та обличчями у файлі encodings.pickle;

– модифікування скрипту facial_rec.py, щоб він міг відправляти сигнали на GPIO-піни та керувати сервоприводом або іншими пристроями, коли обличчя було розпізнано.

Перейшовши до детальних кроків роботи, далі описано виконану мною роботу з тренування моделі розпізнавання обличь за допомогою Raspberry Pi та OpenCV.

Я відкрив новий термінал на Raspberry Pi, натиснувши Ctrl-T.

Далі я створив файли з Python кодом, який мені потрібен, частина використаного коду буде описана далі.

Потім я створив набір даних для тренування Pi. З робочого столу Raspberry Pi я відкрив Файловий Менеджер.

Я перейшов до папки facial_recognition, а потім до папки dataset.

У папці dataset я створив нову папку, клацнувши правою кнопкою миші та вибравши "Нова папка", де я ввів своє ім'я як назву новоствореної папки.

Я натиснув OK, щоб завершити створення папки. Саме тут я зберігав фотографії для тренування моделі.

Далі я продовжив користуватися Файловим Менеджером, перейшов до папки facial_recognition і відкрив headshots.py в Geany.

У файлі headshots.py я додав своє ім'я у третьому рядку коду, зберігаючи лапки навколо імені. Пояснення до зміненого мною коду та його роботи зображено на рисунку 3.14.

```
import cv2

# Ім'я користувача, яке буде використовуватися для створення папки з фотографіями
name = 'Oleksandr'

# Ініціалізація камери
cam = cv2.VideoCapture(0)

# Створення вікна для відображення зображення з камери
cv2.namedWindow("press space to take a photo", cv2.WINDOW_NORMAL)
cv2.resizeWindow("press space to take a photo", 500, 300)

# Лічильник зображень
img_counter = 0

# Основний цикл для зчитування кадрів з камери
while True:
    ret, frame = cam.read()
    if not ret:
        print("failed to grab frame")
        break
    cv2.imshow("press space to take a photo", frame)

    # Очікування натискання клавіші
    k = cv2.waitKey(1)
    if k%256 == 27:
        # Якщо натиснута клавіша ESC, вихід з програми
        print("Escape hit, closing...")
        break
    elif k%256 == 32:
        # Якщо натиснута клавіша SPACE, збереження фотографії
        img_name = "dataset/"+ name +"/image_{}.jpg".format(img_counter)
        cv2.imwrite(img_name, frame)
        print("{} written!".format(img_name))
        img_counter += 1

# Звільнення ресурсів камери
cam.release()

# Закриття усіх вікон
cv2.destroyAllWindows()
```

Рисунок 3.14 – Пояснення до коду ідентифікації обличчя

Я запустив headshots.py, натиснувши на іконку паперового літака в Geany.

Після чого направив веб-камеру на своє обличчя і натиснув пробіл, щоб зробити фото. Я рекомендую зробити близько 10 фотографій обличчя під різними кутами.

Далі я перевірив свої фотографії, повернувшись до папки dataset і папки з моїм ім'ям у Файловому Менеджері. Тепер, після створення набору даних, модель готова до тренування.

У новому терміналі я перейшов до facial_recognition, ввівши «cd facial_recognition». Де я запустив команду для тренування моделі, ввівши «python train_model.py». Сам код та пояснення до його роботи зображено на рисунку 3.15, а результат його роботи на рисунку 3.16.

```
# Шляхи до зображень знаходяться у папці dataset
print("[INFO] start processing faces...")
imagePaths = list(paths.list_images("dataset"))

# Ініціалізація списків для зберігання кодувань та імен
knownEncodings = []
knownNames = []

# Перебір шляхів до зображень
for (i, imagePath) in enumerate(imagePaths):
    # Витягування імені особи з шляху до зображення
    print("[INFO] processing image {}/{}".format(i + 1,
        len(imagePaths)))
    name = imagePath.split(os.path.sep)[-2]

    # Завантаження вхідного зображення та конвертація його з формату RGB (формат OpenCV)
    # до формату RGB (формат dlib)
    image = cv2.imread(imagePath)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Визначення координат рамок обличчя на зображенні
    boxes = face_recognition.face_locations(rgb,
        model="hog")

    # Обчислення кодувань для обличчя
    encodings = face_recognition.face_encodings(rgb, boxes)

    # Перебір отриманих кодувань
    for encoding in encodings:
        # Додавання кожного кодування та імені до нашого набору відомих імен та кодувань
        knownEncodings.append(encoding)
        knownNames.append(name)

# Запис кодувань облич та імен на диск
print("[INFO] serializing encodings...")
data = {"encodings": knownEncodings, "names": knownNames}
f = open("encodings.pickle", "wb")
f.write(pickle.dumps(data))
f.close()
```

Рисунок 3.15 – Пояснення до коду тренування моделі

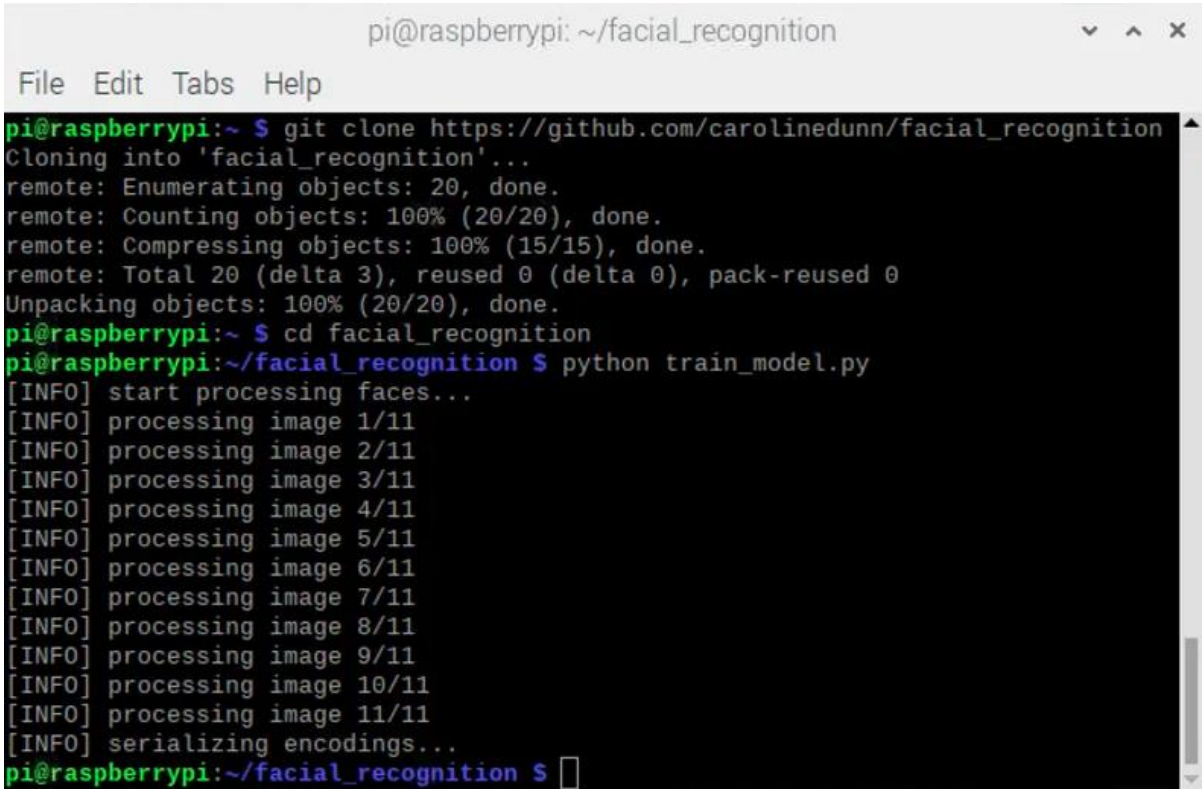


Рисунок 3.16 – Результат роботи тренування моделі в консолі

Код, який використовується для збору даних, демонструє процес створення набору даних для тренування моделі розпізнавання обличь. Він включає кілька важливих етапів, таких як збір фотографій, їх організація у відповідних папках, та автоматизація процесу зйомки за допомогою скрипта [25, 26]..

Перший етап передбачає використання камери для зйомки серії фотографій. Код автоматично контролює камеру, роблячи знімки через певні інтервали часу або у відповідь на конкретні події. Це дозволяє зібрати великий обсяг даних без необхідності постійного ручного втручання. Фотографії зберігаються в структурованих папках, що полегшує їх подальше використання і обробку.

Код також містить інструкції для зміни параметрів скрипта, щоб відповідати конкретним потребам користувача, таким як ім’я папки для збереження фотографій, роздільна здатність зображень, кількість знімків та інтервали між ними. Це робить скрипт надзвичайно гнучким і дозволяє налаштувати його для збору даних в різних умовах та для різних цілей.

Для підвищення ефективності, скрипт включає функції попередньої обробки зображень, такі як нормалізація яскравості та контрастності, а також виявлення і

кадрування обличь. Це дозволяє отримати більш однорідний набір даних, що важливо для тренування моделі.

Додатково, код передбачає можливість автоматичної анотації зображень, що включає виявлення ключових точок на обличчях, таких як очі, ніс і рот. Це може бути корисним для тренування більш складних моделей, які потребують точного розпізнавання окремих рис обличчя.

Завдяки цьому коду, я можу зібрати різноманітні зображення обличь, які будуть використані для створення точної та надійної моделі розпізнавання обличь. Це важливий етап у розробці систем відеоспостереження, які можуть використовуватися для безпеки, ідентифікації осіб або інших застосунків, де необхідне автоматичне розпізнавання обличь. Крім того, цей процес може бути адаптований для різних типів камер і середовищ, що робить його універсальним інструментом для широкого спектру завдань.

Цей код забезпечує не тільки зручність і гнучкість у зборі даних, але й високу якість зібраних зображень. Це досягається завдяки можливостям попередньої обробки та автоматичної перевірки якості знімків. У підсумку, створення такого набору даних є критично важливим для подальшого успішного тренування моделей розпізнавання обличь, що в кінцевому результаті сприяє підвищенню рівня безпеки та ефективності застосунків, що використовують технології розпізнавання обличь.

Код для тренування моделі є частиною системи розпізнавання обличь, яка використовує бібліотеку face_recognition для створення бази даних кодувань обличь. Він автоматизує процес збору зображень, визначення рамок обличь та обчислення унікальних кодувань для кожного обличчя [33].

На першому етапі, код забезпечує автоматизоване збирання зображень, що може здійснюватися як через підключену камеру, так і через завантаження вже наявних зображень. Для цього використовуються функції з бібліотеки OpenCV, які дозволяють налаштувати параметри камери, такі як роздільна здатність та частота кадрів. Це дозволяє збирати якісні зображення для подальшої обробки.

На наступному етапі відбувається визначення рамок обличь на зображеннях.

Для цього використовується модель “hog” (Histogram of Oriented Gradients), яка забезпечує баланс між швидкістю та точністю. Цей метод є особливо корисним для пристроїв з обмеженими обчислювальними ресурсами, таких як Raspberry Pi, оскільки він не потребує великої кількості обчислень та ресурсів.

Після визначення обличь на зображеннях, код обчислює унікальні кодування для кожного обличчя. Кодування представляють собою вектор ознак, який унікально ідентифікує кожну особу. Ці кодування зберігаються разом з іменами осіб у файлі encodings.pickle. Файл pickle є ефективним форматом для зберігання даних, оскільки він дозволяє швидко завантажувати та використовувати кодування під час процесу розпізнавання обличь у реальному часі.

Для забезпечення точності та надійності системи, код також включає функції для перевірки якості зібраних зображень і кодувань. Це включає видалення дублікатів, відсів некоректних або розмитих зображень, та виявлення і виправлення можливих помилок у процесі обчислення кодувань. Такий підхід гарантує, що база даних містить лише високоякісні та точні кодування обличь.

Код ефективно оптимізує процес тренування моделі, забезпечуючи основу для подальшої інтеграції з системами безпеки або іншими застосунками, що потребують ідентифікації особи. Зокрема, отримані кодування можуть бути використані для створення системи відеоспостереження, яка здатна в режимі реального часу розпізнавати осіб та повідомляти про їх присутність. Також, цей підхід може бути інтегрований з іншими системами, такими як контроль доступу, автоматичні каси, або навіть соціальні застосунки, де потрібно ідентифікувати користувачів.

Використання моделі “hog” для визначення обличь забезпечує баланс між швидкістю та точністю, роблячи цей метод підходящим для пристроїв з обмеженими обчислювальними ресурсами, таких як Raspberry Pi. Це дозволяє розгортати систему розпізнавання обличь навіть у віддалених місцях або на невеликих пристроях, що відкриває широкі можливості для її використання у різноманітних умовах та сценаріях.

Цей процес дозволив мені ефективно тренувати модель розпізнавання обличь за допомогою Raspberry Pi та OpenCV.

Загалом весь новий механізм взаємодії всередині системи відеонагляду можна описати алгоритмом взаємодії, який зображений на рисунку 3.17

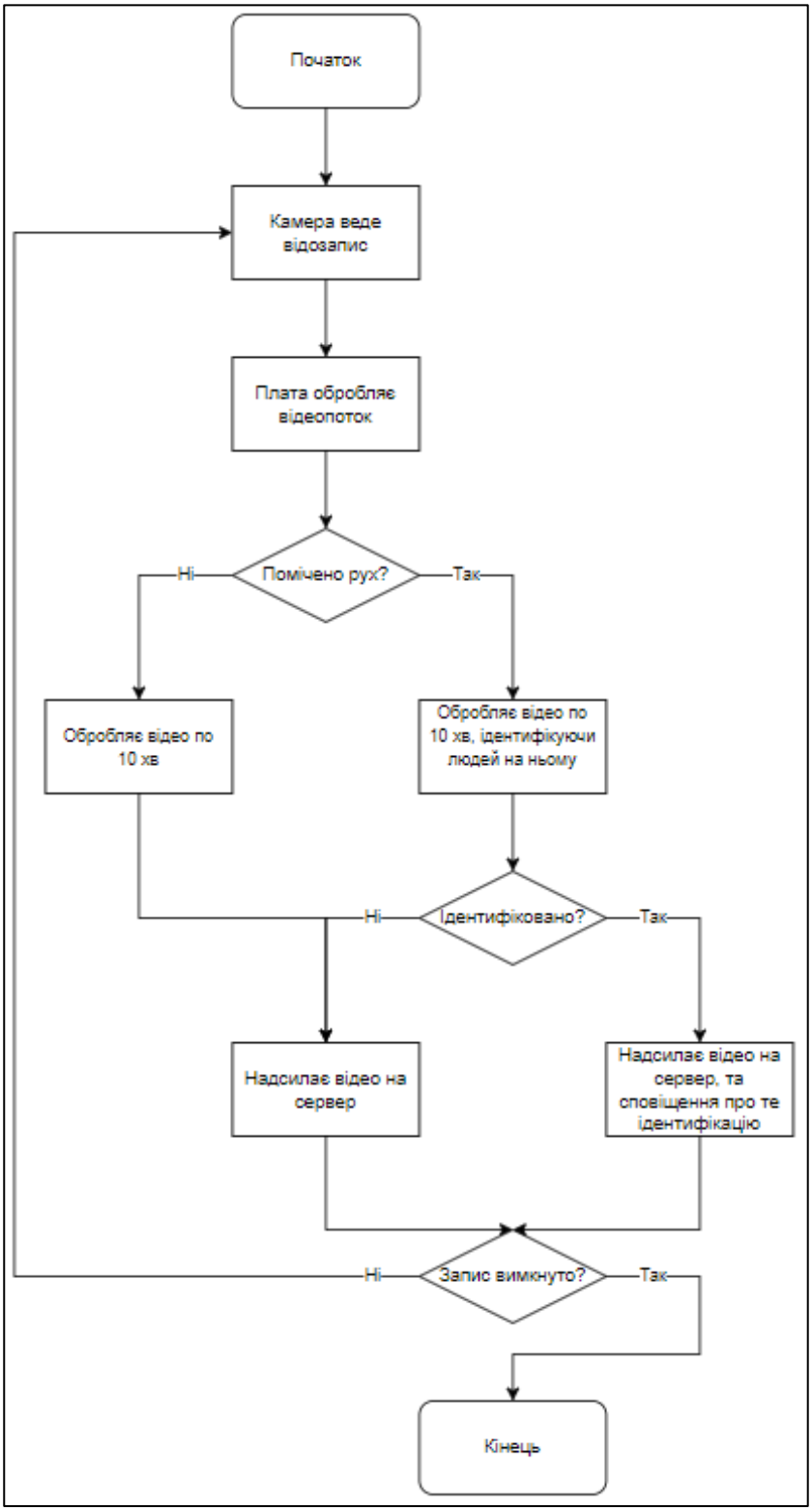


Рисунок 3.17 – Алгоритм роботи системи відеонагляду

3.4 Інструкція із використання розробленої системи відеонагляду

Я зміг налаштувати пряму трансляцію з камери, яку тепер можна переглядати з будь-якого браузера, використовуючи URL-адресу, що вказує на IP-адресу Raspberry Pi та порт 8080, який я налаштував у файлі конфігурації motion.conf. Це дозволило мені легко моніторити простір, не залежно від мого місцезнаходження.

Під час тестування я виявив, що Google Chrome не може безпосередньо відтворювати цей потік через помилку в проекті Chromium, на якому він базується. Я обійшов цю проблему, створивши простий HTML-файл, який містить велике зображення з URL-адресою потоку камери, що дозволило Chrome також відображати пряму трансляцію. Сподіваюся, що розробники Chrome виправлять цю проблему у своєму браузері.

Інші браузери, такі як FireFox, Safari, а також медіаплеєр VLC, успішно відтворювали пряму трансляцію з камери. Однак, я не зміг запустити пряму трансляцію в Internet Explorer, оскільки він не підтримує Motion JPEG. Кеннет Лаврсен, розробник Motion, описав обхідний шлях для прямої трансляції в Internet Explorer.

Для доступу до прямої трансляції з будь-якої точки світу я налаштував динамічний домен для моєї локальної мережі, що дозволяє підключатися до локальної IP-адреси ззовні, навіть якщо локальна IP-адреса змінюється. Я використовую безкоштовний сервіс dyn.com, який інтегрований у багато маршрутизаторів.

Після налаштування динамічного IP-URL, я можу отримати доступ до потоку камери з будь-якого місця через браузер (наприклад, http://(вашдинамічнийдомен):8080). Це також працює з браузера на мобільному пристрої.

3.5 Висновок

Проект зі створення системи відеонагляду на базі Raspberry Pi та інтеграції засобів розпізнавання обличь був виконаний з великою увагою до деталей та технічної точності. Від початку процесу, який включав підготовку Raspberry Pi та встановлення необхідного програмного забезпечення, до збільшення файлу підкачки для оптимізації процесу збірки OpenCV, кожен крок був ретельно продуманим та виконаним з метою створення надійної та ефективної системи.

Значну увагу було приділено налаштуванню мережевих параметрів та зберіганню відеоданих на зовнішньому сервері, що дозволило оптимізувати використання обмежених ресурсів Raspberry Pi. Встановлення та конфігурація програми Motion, а також налаштування параметрів зберігання відео, були виконані з урахуванням потреб проекту, забезпечуючи високу якість відеоспостереження та легкість доступу до записів.

Особлива увага була приділена безпеці та стабільності системи, зокрема, через коректне налаштування прав доступу до файлів та директорій. Завершальним етапом проекту стало встановлення бібліотеки розпізнавання обличь та інструментів для роботи з Python, що відкрило шлях для подальшого розвитку та інтеграції розширених функцій розпізнавання обличь у систему.

У цілому, виконана робота демонструє глибоке розуміння технічних аспектів розробки систем на базі Raspberry Pi та використання відкритих технологій для створення інноваційних рішень у сфері відеонагляду. Цей досвід може бути застосований для подальших проектів, які вимагають інтеграції апаратного та програмного забезпечення для створення комплексних систем безпеки.

ВИСНОВКИ

Під час кваліфікаційної роботи було здійснено теоретичний аналіз, вивчено недоліки існуючих рішень, проведено дослідження та розроблено розумну систему відеонагляду на основі мікроплати. За допомогою передових технологій штучного інтелекту, машинного навчання, комп'ютерного зору та інтернету речей, ця система може виконувати різноманітні функції, такі як детектування руху, розпізнавання об'єктів, класифікація сцен, аналіз поведінки та інші.

Результати дослідження показують, що розумна система відеонагляду на основі мікроплати має великий потенціал для застосування в різних сферах діяльності. Вона може використовуватись в промисловості для контролю процесів та безпеки, у сільському господарстві для моніторингу росту та здоров'я рослин, в транспорті для виявлення порушень правил дорожнього руху, в освіті для забезпечення безпеки учнів та викладачів, в медицині для відстеження пацієнтів та багатьох інших галузях.

Окрім того, розроблена розумна система відеонагляду може значно покращити якість та ефективність відеоспостереження. Вона забезпечує високу якість зображення та може обробляти великі обсяги відеоданих, що дозволяє забезпечити точний аналіз та ідентифікацію об'єктів. Крім того, система може інтегруватися з іншими системами безпеки, такими як датчики, сигналізації, керування доступом, що робить її більш універсальною та цінною для користувачів.

Дослідження також вказує на великий попит на розумні системи відеонагляду. Ринок відеоспостереження очікується зростати в наступні роки, що свідчить про високий інтерес до цих технологій. Розробка розумної системи відеонагляду на основі мікроплати може дати нові можливості для реалізації різних функцій розумного відеоспостереження та сприяти подальшому розвитку цього ринку.

Отже, дана робота відображає актуальні виклики та потреби сучасного суспільства в обробці та аналізі великих обсягів відеоданих. Вона показує

переваги та можливості використання мікроплати для створення розумних систем відеонагляду, а також стимулює інтерес та застосування цього пристрою в різних сферах діяльності. Результати дослідження можуть бути корисними для фахівців у галузі відеоспостереження, розробників програмного забезпечення та інженерів, які працюють над розробкою нових продуктів та сервісів у цій сфері.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Сидоренко О. В., Коваль В. О., Литвиненко А. В. Розумні системи відеоспостереження на основі мікроплат. К.: НТУУ “КПІ”, 2019. 192 с.
2. Singh S. K., Singh A. K., Kumar A. Smart Video Surveillance System Based on Raspberry Pi. New Delhi: Springer, 2018. 150 c.
3. Romanowich J., Chin D., Lento T. Smart Video Security Handbook: A Practical Guide for Catching Intruders Before They Act. New York: McGraw-Hill, 2020. 240 c.
4. Cob-Parro A. C., Losada-Gutiérrez C., Marrón-Romera M., Gardel-Vicente A., Bravo-Muñoz I. Smart Video Surveillance System Based on Edge Computing. Basel: MDPI, 2021. 18 c.
5. Danescu R., Borza D., Itu R. Detecting Micro-Expressions in Real Time Using High-Speed Video Sequences. London: IntechOpen, 2019. 14 c.
6. Damjanovski V. CCTV – From Light to Pixels. Butterworth-Heinemann, 2013. 616 c.
7. Kruegle H. CCTV Surveillance: Video Practices and Technology 2nd Edition. Butterworth-Heinemann, 2006. 672 c.
8. Romanowich J., Chin D., Lento T. V. Smart Video Security Handbook: A Practical Guide for Catching Intruders Before They Act Paperback. Video Valley Press, 2015. 88 c.
9. Anthony C. C. Digital Video Surveillance and Security 2nd Edition. Butterworth-Heinemann, 2014. 440 c.
10. Nilsson F. Intelligent Network Video: Understanding Modern Video Surveillance Systems, Second Edition 2nd Edition. CRC Press, 2016. 390 c.
11. Cieszynski J. Closed Circuit Television 3rd Edition. Newnes, 2007. 336 c.
12. Клинко, О. В. Розумне спостереження: як камери відеонагляду стають смартфонами для бізнесу. 2020. С. 12-17. URL: https://itce.vntu.edu.ua/index.php/itce/article/ iew/791 (дата звернення: 16.05.2024).

13. Відеоспостереження та системи відеонагляду: захист і безпека 24/7 від VidiMost. 2020. URL: https://vidimost.com.ua/ (дата звернення: 16.05.2024).

14. В. О. Литвиненко. Розумний пошук у системах відеонагляду для підвищення рівня безпеки міст. 2020. С. 42-49. URL: http://radap.kpi.ua/ radiotechnique/article/view/1569 (дата звернення: 19.04.2024).

15. Розумні функції в камерах відеоспостереження: як технології штучного інтелекту підвищують рівень безпеки. 2024. URL: https://viatec.ua/news/765 (дата звернення: 24.04.2024).

16. 10 найкращих генераторів відео зі штучним інтелектом. 2022. URL: https://www.imena.ua/blog/10-best-ai-video-generators/ (дата звернення: 24.04.2024).

17. Використовуємо штучний інтелект для поліпшення якості відео. 2023. URL: https://evergreens.com.ua/ua/articles/video-enhancement.html (дата звернення: 21.04.2024).

18. Штучний інтелект у школі: від загальної концепції до практичного застосування. 2021. URL: https://naurok.com.ua/webinar/shtuchniy-intelekt-u-shkoli-vid-zagalno-koncepci-do-praktichnogo-zastosuvannya (дата звернення: 24.04.2024).

19. Raspberry Pi Documentation - Raspberry Pi. 2021. URL: https://raspberry pi.org /documentation/raspberry-pi/ (дата звернення: 24.04.2024).

20. Модуль зв’язку SIM7600E-H LTE Cat-4 4G/3G/2G, GNSS для Raspberry Pi. 2021. URL: https://www.waveshare.com/sim7600e-h-4g-for-raspberry-pi.htm (дата звернення: 22.05.2024).

21. Камери і об’єктиви для мікрокомп’ютерів | Raspberry Pi | Arduino. 2021. URL: https://botland.com.ua/uk/14-kamery-i-ob-yektyvy-dlya-mikrokompyuteriv (дата звернення: 19.05.2024).

22. OpenCV - Open Source Computer Vision Library. 2021. URL: https://open cv.org/ (дата звернення: 30.04.2024).

23. В. О. Литвиненко, А. В., Сидоренко. Розпізнавання обличь на базі OpenCV, Face Recognition Library і Raspberry Pi. 2018. С. 57-64. URL: http://radap. kpi.ua/ radiotechnique/article/view/1448 (дата звернення: 16.05.2024).

24. А. В. Сидоренко, О. В., Коваль, В. О. Литвиненко. Алгоритм розпізнавання обличчя людей на базі згорткової нейронної мережі. 2019. С. 48-55. URL: http://radap.kpi.ua/radiotechnique/article/view/1530 (дата звернення: 30.04.2024).

25. Шевченко, А. Працюємо з нейромережами: як ми навчили камеру розпізнавати обличчя, щоб обійтися без перепусток в офіс. 2018. URL: https://dou.ua/lenta/articles/neural-networks-for-face-recognition/ (дата звернення: 30.04.2024).

26. А. В. Сидоренко, О. В. Коваль, В. О. Литвиненко. Розпізнавання обличь на потоковому відеоряді за допомогою бібліотеки OpenCV. 2017. С. 49-56. URL: http://radap.kpi.ua/radiotechnique/article/view/1311 (дата звернення: 30.04.2024).

27. OpenCV - Вікіпедія. 2021. URL: https://uk.wikipedia.org/wiki/OpenCV (дата звернення: 19.05.2024).

28. M. A. Al-Mashhadani, A. A. Al-Hindawi, A. A. Al-Saedi. Smart Video Surveillance System Based on Raspberry Pi. 2016. С. 1-6. URL: https://www.ijcaonline.org/volume136/almashhadani-2016-ijca-910413.pdf (дата звернення: 16.05.2024).

29. S. K. Singh, A. K. Singh. Design and Implementation of a Smart Video Surveillance System Based on Raspberry Pi. 2017. С. 1-5. URL: https://www.ijert.org/research/design-and-implementation-of-a-smart-video-surveillance-system-based-on-raspberry-pi-IJERTV6IS040055.pdf (дата звернення: 02.05.2024).

30. S. S. Patil, S. S. Kulkarni, S. S. Patil. Raspberry Pi Based Smart Surveillance System/ 2018/ С. 1739-1744. URL: https://www.ijircce.com/upload/2018/march/181_Raspberry.pdf (дата звернення: 02.05.2024).

31. Artificial Intelligence for Video Surveillance. 2021. URL: https://www.scati.com/en/artificial-intelligence-video-surveillance/ (дата звернення: 20.02.2024).

32. M. A. Saleem Durai. Intelligent video surveillance: a review through deep learning techniques for crowd analysis. 2019. URL: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0212-5 (дата звернення: 20.05.2024).

33. S. K. Singh, A. K. Kumar. Smart Surveillance System using Raspberry Pi Mobile Computing. 2017. С. 1-5. URL: https://www.ijcsmc.com/docs/papers/ April2017/V6I4201712.pdf (дата звернення: 21.02.2024).

34. Smart Video Surveillance System using Raspberry Pi and OpenCV. URL: https://www.ijarcs.info/index.php/Ijarcs/article/view/4150 (дата звернення: 22.05.2024).

35. A Smart Video Surveillance System Based on Raspberry Pi and OpenCV. URL: https://www.ijecs.in/index.php/ijecs/article/view/3760 (дата звернення: 22.05.2024).

36. Design and Implementation of Smart Video Surveillance System using Raspberry Pi. URL: https://www.ijecs.in/index.php/ijecs/article/view/3761 (дата звернення: 19.04.2024).

37. A. C. Losada-Gutiérrez. Smart Video Surveillance System Based on Edge Computing. 2021. С. 2958-2975. URL: https://www.mdpi.com/1424-8220/21/9/2958 (дата звернення: 19.05.2024).

38. R. Rajavel, S. K. Ravichandran. IoT-based smart healthcare video surveillance system using edge computing. 2021. С. 3195-3207. URL: https://link. springer.com/article/10.1007/s12652-021-03157-1 (дата звернення: 21.05.2024).

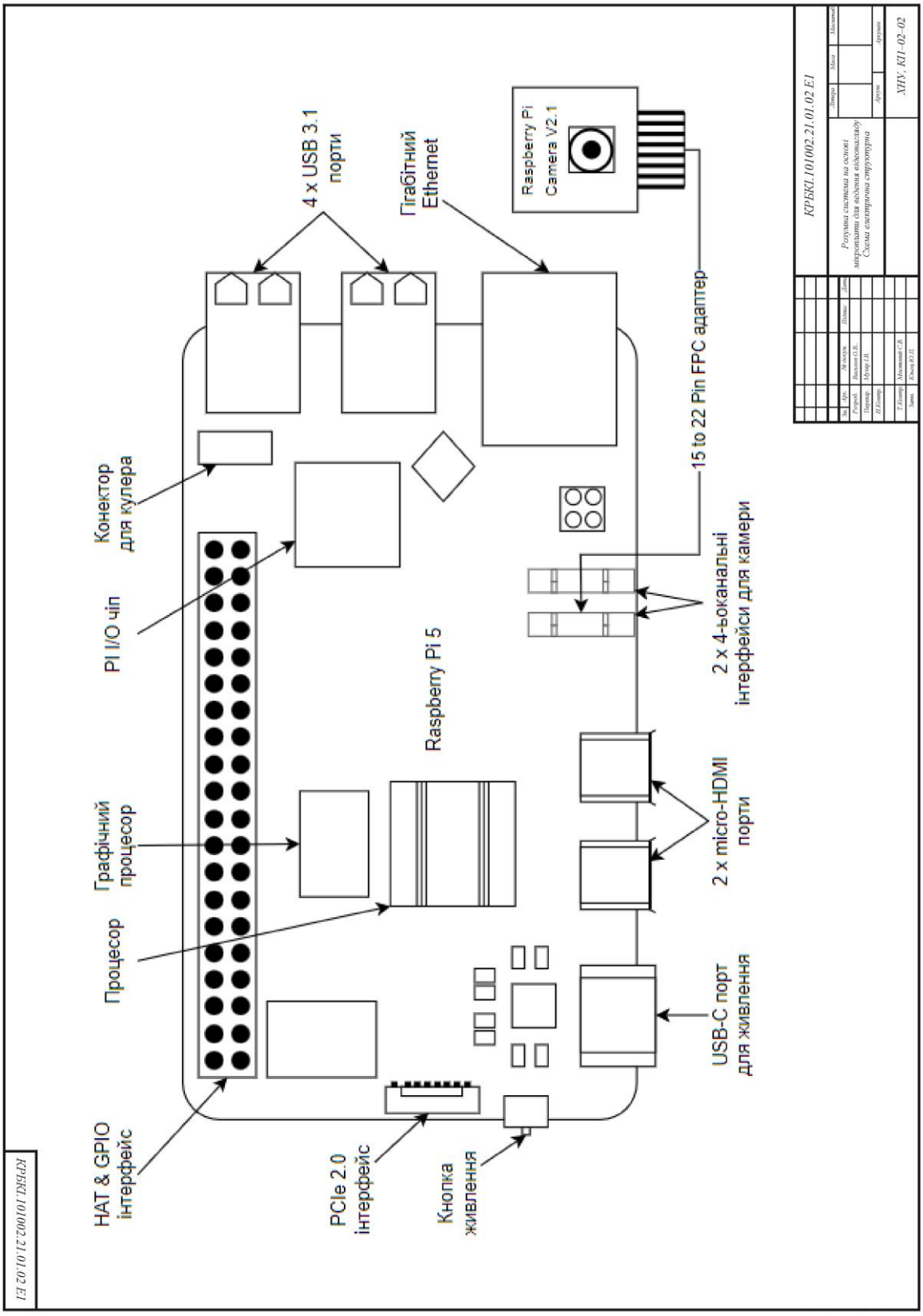
39. A. K. Kumar. Design and Implementation of a Smart Video Surveillance System Based on Raspberry. 2019. С. 1-5. URL: https://www.ijert.org/research/ design-and-implementation-of-a-smart-video-surveillance-system-based-on-raspberry-pi-IJERTV6IS040055.pdf (дата звернення: 20.05.2024).

40. S. S. Patil, S.S. Kulkarni. Raspberry Pi Based Smart Surveillance System. 2018. URL: https://www.ijircce.com/upload/2018/march/181_Raspberry.pdf (дата звернення: 22.05.2024).

ДОДАТОК А

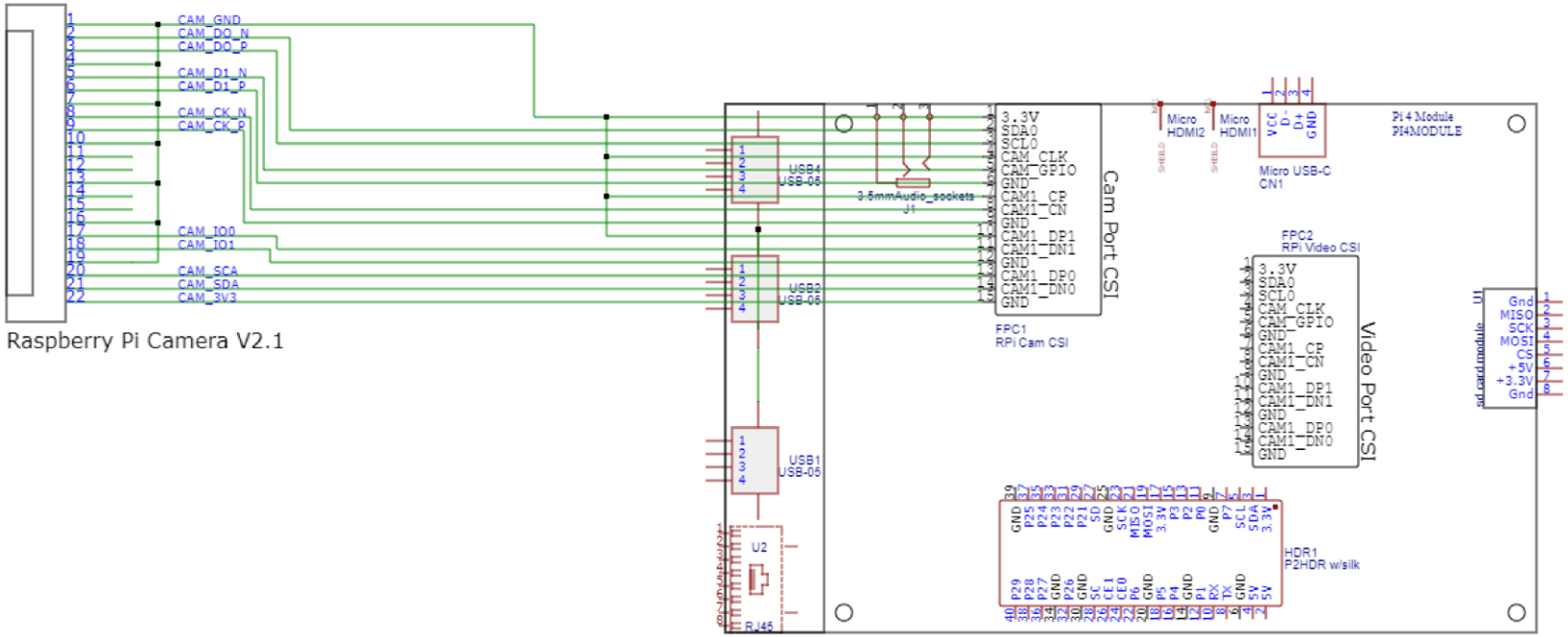
(обов’язковий)

Копія графічної частини



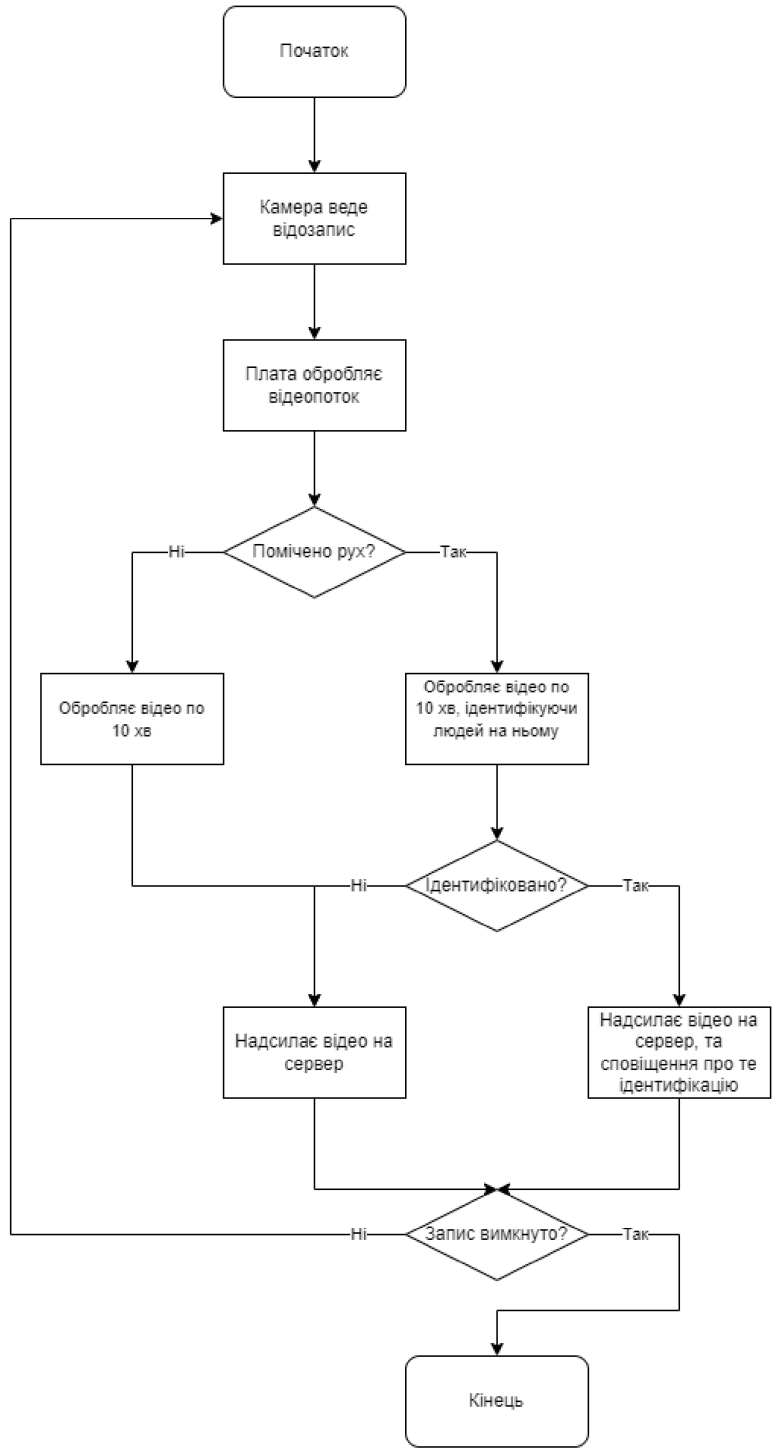
J1
J2
54548-2271
SFW 15R-2STE1LF
CAM_GND
CAM_D0_N
CAM_D0_P
CAM_D1_N
CAM_D1_P
CAM_CK_N
CAM_CK_P
CAM_IO0
CAM_IO1
CAM_SCA
CAM_SDA
CAM_3V3
КРБКЦ.101002.21.01.02 Е2
Розумна система на основі мікроплати для керування підсмоктувачем
Схема електрична функціональна
ХНУ, КІ1–02–02

КРБКІ.101002.21.01.02 Е3



						КРБКІ.101002.21.01.02 Е3	
Зм.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Васьков О.В.			Розумна система на основі мікроплати для ведення відеонагляду Схема електрична принципова	Літера	Маса	Масштаб
Перевір.		Мулар І.В.						
Н.Контр.						Аркуш	Аркушів	
Т.Контр.		Мостовий С.В.				ХНУ, КІ-02-02		
Затв.		Кльоц Ю.П.						

КРБКІ.101002.21.01.02 Е8



Зм.	Арк.	№ докум.	Підпис	Дата
Розроб.		Васьков О.В.		
Перевір.		Мулар І.В.		
Н.Контр.				
Т.Контр.		Мостовий С.В.		
Затв.		Кільо Ю.П.		

КРБКІ.101002.21.01.02 Е8

Розумна система на основі мікроплати для ведення відеонагляду
Алгоритм роботи

Літера	Маса	Масштаб
Аркуш	Аркушів	

ХНУ, КП-02-02

					КРБКІ.101002.21.01.02 Е8	
Зм.	Арк.	№ докум.	Підпис	Дата	Розумна система на основі мікроплати для ведення відеонагляду Фото прототипу пристрою	Літера / Маса / Масштаб
Розроб.		Васьков О.В.				
Перевір.		Мулар І.В.				
Н.Контр.						Аркуш / Аркушів
Т.Контр.		Мостовий С.В.				
Затв.		Кльоц Ю.П.				ХНУ, КІ1–02-02

ДОДАТОК Б

(обов’язковий)

Лістинг коду для розпізнавання обличь

```
#! /usr/bin/python

# import the necessary packages
from imutils.video import VideoStream
from imutils.video import FPS
import face_recognition
import imutils
import pickle
import time
import cv2

#Initialize 'currentname' to trigger only when a new person is identified.
currentname = "unknown"
#Determine faces from encodings.pickle file model created from train_model.py
encodingsP = "encodings.pickle"

# load the known faces and embeddings along with OpenCV's Haar
# cascade for face detection
print("[INFO] loading encodings + face detector...")
data = pickle.loads(open(encodingsP, "rb").read())

# initialize the video stream and allow the camera sensor to warm up
# Set the ser to the followng
# src = 0 : for the build in single web cam, could be your laptop webcam
# src = 2 : I had to set it to 2 inorder to use the USB webcam attached to my laptop
vs = VideoStream(src=2,framerate=10).start()
#vs = VideoStream(usePiCamera=True).start()
time.sleep(2.0)

# start the FPS counter
fps = FPS().start()
```

```
# loop over frames from the video file stream
while True:
	# grab the frame from the threaded video stream and resize it
	# to 500px (to speedup processing)
	frame = vs.read()
	frame = imutils.resize(frame, width=500)
	# Detect the fce boxes
	boxes = face_recognition.face_locations(frame)
	# compute the facial embeddings for each face bounding box
	encodings = face_recognition.face_encodings(frame, boxes)
	names = []

	# loop over the facial embeddings
	for encoding in encodings:
		# attempt to match each face in the input image to our known
		# encodings
		matches = face_recognition.compare_faces(data["encodings"],
			encoding)
		name = "Unknown" #if face is not recognized, then print Unknown

		# check to see if we have found a match
		if True in matches:
			# find the indexes of all matched faces then initialize a
			# dictionary to count the total number of times each face
			# was matched
			matchedIdxs = [i for (i, b) in enumerate(matches) if b]
			counts = {}

			# loop over the matched indexes and maintain a count for
			# each recognized face face
			for i in matchedIdxs:
				name = data["names"][i]
				counts[name] = counts.get(name, 0) + 1

			# determine the recognized face with the largest number
```

```
                    # of votes (note: in the event of an unlikely tie Python
                    # will select first entry in the dictionary)
                    name = max(counts, key=counts.get)

                    #If someone in your dataset is identified, print their name on the screen
                    if currentname != name:
                        currentname = name
                        print(currentname)

                # update the list of names
                names.append(name)

        # loop over the recognized faces
        for ((top, right, bottom, left), name) in zip(boxes, names):
                # draw the predicted face name on the image - color is in BGR
                cv2.rectangle(frame, (left, top), (right, bottom),
                        (0, 255, 225), 2)
                y = top - 15 if top - 15 > 15 else top + 15
                cv2.putText(frame, name, (left, y), cv2.FONT_HERSHEY_SIMPLEX,
                        .8, (0, 255, 255), 2)

        # display the image to our screen
        cv2.imshow("Facial Recognition is Running", frame)
        key = cv2.waitKey(1) & 0xFF

        # quit when 'q' key is pressed
        if key == ord("q"):
                break

        # update the FPS counter
        fps.update()

# stop the timer and display FPS information
fps.stop()
print("[INFO] elasped time: {:.2f}".format(fps.elapsed()))
```

```
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()



#! /usr/bin/python

# import the necessary packages
from imutils.video import VideoStream
from imutils.video import FPS
import face_recognition
import imutils
import pickle
import time
import cv2
import requests

#Initialize 'currentname' to trigger only when a new person is identified.
currentname = "unknown"
#Determine faces from encodings.pickle file model created from train_model.py
encodingsP = "encodings.pickle"
#use this xml file
cascade = "haarcascade_frontalface_default.xml"

#function for setting up emails
def send_message(name):
    return requests.post(
        "https://api.mailgun.net/v3/YOUR_DOMAIN_NAME/messages",
        auth=("api", "YOUR_API_KEY"),
        files = [("attachment", ("image.jpg", open("image.jpg", "rb").read()))],
        data={"from": 'hello@example.com',
            "to": ["YOUR_MAILGUN_EMAIL_ADDRESS"],
            "subject": "You have a visitor",
```

```
            "html": "<html>" + name + " is at your door.  </html>"})

# load the known faces and embeddings along with OpenCV's Haar
# cascade for face detection
print("[INFO] loading encodings + face detector...")
data = pickle.loads(open(encodingsP, "rb").read())
detector = cv2.CascadeClassifier(cascade)

# initialize the video stream and allow the camera sensor to warm up
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
# vs = VideoStream(usePiCamera=True).start()
time.sleep(2.0)

# start the FPS counter
fps = FPS().start()

# loop over frames from the video file stream
while True:
            # grab the frame from the threaded video stream and resize it
            # to 500px (to speedup processing)
            frame = vs.read()
            frame = imutils.resize(frame, width=500)

            # convert the input frame from (1) BGR to grayscale (for face
            # detection) and (2) from BGR to RGB (for face recognition)
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

            # detect faces in the grayscale frame
            rects = detector.detectMultiScale(gray, scaleFactor=1.1,
                        minNeighbors=5, minSize=(30, 30),
                        flags=cv2.CASCADE_SCALE_IMAGE)

            # OpenCV returns bounding box coordinates in (x, y, w, h) order
```

```
# but we need them in (top, right, bottom, left) order, so we
# need to do a bit of reordering
boxes = [(y, x + w, y + h, x) for (x, y, w, h) in rects]

# compute the facial embeddings for each face bounding box
encodings = face_recognition.face_encodings(rgb, boxes)
names = []

# loop over the facial embeddings
for encoding in encodings:
	# attempt to match each face in the input image to our known
	# encodings
	matches = face_recognition.compare_faces(data["encodings"],
		encoding)
	name = "Unknown"

	# check to see if we have found a match
	if True in matches:
		# find the indexes of all matched faces then initialize a
		# dictionary to count the total number of times each face
		# was matched
		matchedIdxs = [i for (i, b) in enumerate(matches) if b]
		counts = {}

		# loop over the matched indexes and maintain a count for
		# each recognized face face
		for i in matchedIdxs:
			name = data["names"][i]
			counts[name] = counts.get(name, 0) + 1

		# determine the recognized face with the largest number
		# of votes (note: in the event of an unlikely tie Python
		# will select first entry in the dictionary)
		name = max(counts, key=counts.get)
```

```
                    #If someone in your dataset is identified, print their name on the screen
                    if currentname != name:
                        currentname = name
                        print(currentname)
                        #Take a picture to send in the email
                        img_name = "image.jpg"
                        cv2.imwrite(img_name, frame)
                        print('Taking a picture.')

                        #Now send me an email to let me know who is at the door
                        request = send_message(name)
                        print ('Status Code: '+format(request.status_code)) #200 status code means email sent successfully

                # update the list of names
                names.append(name)

        # loop over the recognized faces
        for ((top, right, bottom, left), name) in zip(boxes, names):
                # draw the predicted face name on the image - color is in BGR
                cv2.rectangle(frame, (left, top), (right, bottom),
                        (0, 255, 225), 2)
                y = top - 15 if top - 15 > 15 else top + 15
                cv2.putText(frame, name, (left, y), cv2.FONT_HERSHEY_SIMPLEX,
                        .8, (0, 255, 255), 2)

        # display the image to our screen
        cv2.imshow("Facial Recognition is Running", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
                break

        # update the FPS counter
```

```
        fps.update()

# stop the timer and display FPS information
fps.stop()
print("[INFO] elasped time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()



import cv2

name = 'Caroline' #replace with your name

cam = cv2.VideoCapture(0)

cv2.namedWindow("press space to take a photo", cv2.WINDOW_NORMAL)
cv2.resizeWindow("press space to take a photo", 500, 300)

img_counter = 0

while True:
    ret, frame = cam.read()
    if not ret:
        print("failed to grab frame")
        break
    cv2.imshow("press space to take a photo", frame)

    k = cv2.waitKey(1)
    if k%256 == 27:
        # ESC pressed
        print("Escape hit, closing...")
        break
```

```
    elif k%256 == 32:
        # SPACE pressed
        img_name = "dataset/"+ name +"/image_{}.jpg".format(img_counter)
        cv2.imwrite(img_name, frame)
        print("{} written!".format(img_name))
        img_counter += 1

cam.release()

cv2.destroyAllWindows()



import cv2
from picamera import PiCamera
from picamera.array import PiRGBArray

name = 'Caroline' #replace with your name

cam = PiCamera()
cam.resolution = (512, 304)
cam.framerate = 10
rawCapture = PiRGBArray(cam, size=(512, 304))

img_counter = 0

while True:
    for frame in cam.capture_continuous(rawCapture, format="bgr", use_video_port=True):
        image = frame.array
        cv2.imshow("Press Space to take a photo", image)
        rawCapture.truncate(0)

        k = cv2.waitKey(1)
        rawCapture.truncate(0)
        if k%256 == 27: # ESC pressed
            break
```

```
        elif k%256 == 32:
            # SPACE pressed
            img_name = "dataset/"+ name +"/image_{}.jpg".format(img_counter)
            cv2.imwrite(img_name, image)
            print("{} written!".format(img_name))
            img_counter += 1

    if k%256 == 27:
        print("Escape hit, closing...")
        break

cv2.destroyAllWindows()



#! /usr/bin/python

# Imports
import requests

def send_simple_message():
    print("I am sending an email.")
    return requests.post(
        "https://api.mailgun.net/v3/YOUR_DOMAIN_NAME/messages",
        auth=("api", "YOUR_API_KEY"),
        data={"from": 'hello@example.com',
            "subject": "Visitor Alert",
            "html": "<html> Your Raspberry Pi recognizes someone. </html>"})

request = send_simple_message()
print ('Status: '+format(request.status_code))
print ('Body:'+ format(request.text))



#! /usr/bin/python
```

```
# import the necessary packages
from imutils import paths
import face_recognition
#import argparse
import pickle
import cv2
import os

# our images are located in the dataset folder
print("[INFO] start processing faces...")
imagePaths = list(paths.list_images("dataset"))

# initialize the list of known encodings and known names
knownEncodings = []
knownNames = []

# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
	# extract the person name from the image path
	print("[INFO] processing image {}/{}".format(i + 1,
		len(imagePaths)))
	name = imagePath.split(os.path.sep)[-2]

	# load the input image and convert it from RGB (OpenCV ordering)
	# to dlib ordering (RGB)
	image = cv2.imread(imagePath)
	rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

	# detect the (x, y)-coordinates of the bounding boxes
	# corresponding to each face in the input image
	boxes = face_recognition.face_locations(rgb,
		model="hog")

	# compute the facial embedding for the face
	encodings = face_recognition.face_encodings(rgb, boxes)
```

```
        # loop over the encodings
        for encoding in encodings:
                # add each encoding + name to our set of known names and
                # encodings
                knownEncodings.append(encoding)
                knownNames.append(name)

# dump the facial encodings + names to disk
print("[INFO] serializing encodings...")
data = {"encodings": knownEncodings, "names": knownNames}
f = open("encodings.pickle", "wb")
f.write(pickle.dumps(data))
f.close()
```

Завідувачу кафедри кібербезпеки
к.т.н., доц. Кльоцу Ю.П.
Васькова Олександра Вікторовича
ПІБ здобувача вищої освіти

Студента ФІТ, 3 курсу, групи КІ1с-21-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 31.08.2023, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

05.06.2024
дата

підпис

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 10%**

	Документ		Сумарний збіг по Базі Даних	
ID: 131007 Назва: Розумна система на основі мікроплати для ведення відеонагляду Додано в БД: 2024-06-17 Автора: Васьков О.В. Керівники: Муляр І.В. Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	63415	950	853 (1%)	9 (1%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра кібербезпеки

ID перевірки:
1016368245

Дата перевірки:
17.06.2024 21:21:48 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
17.06.2024 22:26:08 EEST

ID користувача:
100008300

Назва документа: Васьков_О_В_КІ1с_21_1 (новий

Кількість сторінок: 57 Кількість слів: 9767 Кількість символів: 75198 Розмір файлу: 1.94 MB ID файлу: 1016174946

1.83%
Схожість

Найбільша схожість: 0.68% з джерелом з Бібліотеки (ID файлу: 1016166830)

1.16% Джерела з Інтернету	94	Сторінка 59
1.33% Джерела з Бібліотеки	97	Сторінка 60

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Розумна система на основі мікроплати для ведення відеонагляду

Автор: Васьков Олександр Вікторович

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Муляр Ігор Володимирович, канд. техн. наук, доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Оригінальність тексту роботи за результатами перевірки системою Unichek складає 98,17%, оригінальність тексту роботи за результатами перевірки системою Anti-Plagiarism v-15.257 складає 99%.

Згідно з Положенням про систему забезпечення академічної доброчесності у ХНУ (https://khmnu.edu.ua/wp-content/uploads/normatyvni-dokumenty/polozhennya/pro-systemu-zabezpechennya-akademichnoyi-dobrochesnosti.pdf, Додаток В) кваліфікаційна робота, виконана за , освітньо-професійною програмою, кількісні показники рівня унікальності тексту у відсотках до загального обсягу матеріалу в якій складає 75-100 %, визнається роботою з високою унікальністю тексту: «Текст вважається унікальним і не потребує додаткових дій щодо запобігання неправомірним запозиченням».

Керівник роботи _____ Ігор МУЛЯР

Завідувач кафедри кібербезпеки _____ Юрій КЛЬОЦ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «бакалавр»

Дипломник Васьков Олександр Вікторович

Тема Розумна система на основі мікроплати для ведення відеонагляду

Спеціальність 123 Комп'ютерна інженерія

Обсяг кваліфікаційної роботи освітньо-кваліфікаційного рівня «бакалавр»:

кількість листів креслень 5 ; кількість сторінок записки 62

1. Короткий зміст роботи та прийнятих рішень У кваліфікаційній роботі спроектовано та розроблено розумну систему відеонагляду, що дозволяє віднести роботу до категорії "відеоспостережень" .

2. Висновок про відповідність кваліфікаційної роботи завданню Кваліфікаційна робота у повній мірі відповідає поставленому завданню як в теоретичній, так і в практичній частині роботи

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У вступі подана загальна характеристика поставленої задачі, сформульована актуальність. Визначені задачі, які необхідно вирішити для досягнення поставленої мети, У першому розділі проведено огляд та аналіз існуючих технологій реалізації систем відеоспостережень та безпеки даних у них, виконана постановка задачі. В другому розділі виконано аналіз наявних засобів для проектування та розробки системи відеоспостереження та обґрунтовано вибір пакету для роботи. В третьому розділі розроблено систему відеоспостереження на основі мікроплати, обрано структуру її будови, проведено її основне конфігурування, написано та налаштовано всі програмні засоби для її роботи.

4. Позитивні сторони роботи Кваліфікаційна робота має практичну цінність. Практична цінність результатів дослідження полягає в обґрунтуванні вибору засобів розробки та компонентів системи та їх налаштуванню для побудови системи відеоспостереження

5. Негативні сторони роботи В роботі відсутній деталізований опис розробки

6. Оцінка графічного оформлення та пояснювальної записки роботи Графічне оформлення виконане відповідно до теми кваліфікаційної роботи з дотриманням стандартів. В загальному графічне оформлення виконане якісно, пояснювальна записка відповідає нормам щодо її оформлення.

7. Відгук про роботу в цілому В загальному кваліфікаційна робота заслуговує позитивної оцінки. Весь матеріал кваліфікаційної роботи структурований, чіткий та послідовний. Усі розділи роботи послідовні та логічні, що дозволяє чітко розуміти викладений матеріал в рамках тематики кваліфікаційної роботи. Графічний матеріал дозволяє наочно побачити доцільність та ефективність рішень, які були прийняті за основу для досягнення поставленої мети.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Враховуючи всі позитивні та негативні сторони представленої кваліфікаційної роботи, можна зробити висновок, що вона заслуговує оцінку «відмінно».

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи)
Мішан Віктор Володимирович, кандидат технічних наук, доцент

« 12 » 06 2024р. (підпис)