Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ СИСТЕМИ ВІДЕОСПОСТЕРЕЖЕННЯ З АВТОМАТИЧНИМ ВИЯВЛЕННЯМ РУХУ ТА ПЕРЕДАЧЕЮ МУЛЬТИМЕДІЙНИХ ДАНИХ ЧЕРЕЗ БОТА В TELEGRAM
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-22-1
Петренко К.А.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник ст. викл. Путятіна О.Є.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Петренку Кирилу Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення системи відеоспостереження з автоматичним виявленням руху та передачею мультимедійних даних через бота в Telegram

затверджена наказом університету від 26 травня 2026 року № 435Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 19 червня 2026 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література з комп'ютерного зору, відеоспостереження, оброблення зображень та штучного інтелекту; матеріали наукових конференцій; нормативні та методичні документи; ресурси мережі Інтернет; бібліотеки OpenCV, ONNX Runtime, aiogram; засоби розроблення Python; документація Telegram Bot API.

4. Перелік питань, що потрібно опрацювати в роботі
1.Аналіз сучасних підходів до виявлення руху та розпізнавання людини у відеопотоці.
2.Обґрунтування вибору методів, алгоритмів та архітектури програмної системи відеоспостереження.
3.Розроблення програмної системи автоматичного виявлення руху, розпізнавання людини та збереження подій.
4.Реалізація механізму надсилання мультимедійних повідомлень користувачу через Telegram-бота.
5.Проведення тестування розробленої системи та оцінювання ефективності її роботи.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Ілюстрація «Еволюція підходів до відеоспостереження», схема «Основні методы виявлення руху та їх переваги», схема «Структура взаємодії модулів програмної системи», блок-схема «Алгоритм формування події у програмній системі», ілюстрація «Структура каталогів програмного застосунку», схема «Асинхронна взаємодія компонентів програмної системи», блок-схема «Алгоритм виявлення руху в програмній системі», блок-схема «Режими розпізнавання людини у програмній системі», схема «Взаємодія користувача з Telegram-ботом», схема «Збереження стану та матеріалів подій у програмній системі», блок-схема «Схема перевірки працездатності програмної системи».

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	13.04.2026	
2	Аналіз завдання, підбір літератури	14.04.26-19.04.26	
3	Аналіз літератури з проблеми, що вирішується	20.04.26-22.04.26	
4	Аналіз існуючих методів відеомоніторингу	23.04.26-27.04.26	
5	Формування кроків вирішення проблеми	28.04.26-05.05.26	
6	Програмна реалізація	06.05.26-24.05.26	
7	Аналіз отриманих результатів	25.05.26-01.06.26	
8	Оформлення пояснювальної записки	02.06.26-09.06.26	
9	Перевірка на нормоконтроль	10.06.26-19.06.26	
10	Перевірка на плагіат	11.06.26-30.06.26	
11	Рецензування	20.06.26-30.06.26	
12	Підготовка презентації та доповіді	20.06.26-30.06.26	
13	Занесення роботи в електронний архів	30.06.26-09.07.26	
14	Попередній захист кваліфікаційної роботи	30.06.26-09.07.26	

Дата видачі завдання 13 квітня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Путятіна О.Є.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 86 с., 19 табл., 12 рис., 1 дод., 31 джерело.

ВИЯВЛЕННЯ РУХУ, ВІДЕОСПОСТЕРЕЖЕННЯ, КАМЕРА, КОМП'ЮТЕРНИЙ ЗІР, OPENCV, PYTHON, TELEGRAM-БОТ, YOLO.

Об'єктом роботи є розроблення програмної системи відеоспостереження з автоматичним виявленням руху та сповіщенням користувача.

Метою роботи є розроблення програмного застосунку для відеомоніторингу, розпізнавання людини в кадрі та надсилання сповіщень через Telegram-бота.

У роботі використано методи комп'ютерного зору, OpenCV, YOLO, aiogram, JSON та мову програмування Python.

Практична цінність полягає в автоматизації контролю приміщення або робочої зони та оперативному інформуванні користувача.

У результаті розроблено застосунок, що забезпечує виявлення руху, розпізнавання людини, збереження подій і надсилання фото- або відеосповіщень.

Область застосування: домашні системи безпеки, офісний моніторинг, автоматизоване відеоспостереження.

ARTIFICIAL INTELLIGENCE, CAMERA, COMPUTER VISION, MOTION DETECTION, OPENCV, PYTHON, TELEGRAM BOT, VIDEO SURVEILLANCE, YOLO.

The objective of this work is the development process of a video surveillance software system with automatic motion detection and user notification.

The goal of this work is to develop a software application for video monitoring, human recognition, and Telegram notifications.

The work uses computer vision methods, OpenCV, YOLO, aiogram, JSON, and Python.

The practical value lies in automating room or working area monitoring and promptly informing the user.

As a result, an application was developed that detects motion, recognizes humans, stores events, and sends photo or video alerts.

Applications: home security systems, office monitoring, and automated video surveillance.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

ШІ – штучний інтелект

aiogram – асинхронна бібліотека для створення Telegram-ботів мовою Python

asyncio – вбудована бібліотека Python для організації асинхронного введення-виведення

alert-worker – асинхронна фонова задача для оброблення та відправлення черги сповіщень

API – Application Programming Interface (інтерфейс прикладного програмування)

API Telegram – Telegram Bot Application Programming Interface (інтерфейс програмування застосунків для взаємодії з платформою Telegram)

COCO – Common Objects in Context (набір даних із поширеними об’єктами в контексті для навчання нейромереж)

DEBUG – Debugging Mode (режим налагодження та відображення діагностичних даних на зображенні)

DNN – Deep Neural Network (глибока нейронна мережа)

DOTENV – Environment Variables Loader (модуль завантаження службових параметрів із файлу оточення)

FPS – Frames Per Second (кадри за секунду)

GaussianBlur – Gaussian Blur Filter (алгоритм згладжування зображення за Гауссом для зменшення цифрового шуму)

HOG – Histogram Of Oriented Gradients (гістограма орієнтованих градієнтів для виявлення силуетів)

IP – Internet Protocol (інтернет-протокол)

JPEG – Joint Photographic Experts Group (формат стиснення та збереження растрових зображень)

JSON – JavaScript Object Notation (текстовий формат обміну структурованими даними для збереження налаштувань)

MOG2 – Mixture Of Gaussians 2 (модель суміші гауссівських розподілів другого покоління для фонового віднімання)

MotionDetector – Motion Detector Class (програмний клас детекції руху, що реалізує аналіз відеопотоку)

MP4 – Moving Picture Experts Group 4 (медіаконтейнер для збереження та передавання відеофрагментів)

ObjectIdentifier – Object Identifier Class (програмний клас розпізнавання людини за допомогою YOLO або OpenCV)

ONNX – Open Neural Network Exchange (відкритий формат обміну нейронними мережами)

OpenCV – Open Source Computer Vision Library (відкрита бібліотека комп'ютерного зору)

Pillow – Python Imaging Library Fork (допоміжна бібліотека для роботи з графічними ресурсами й іконками трея)

polling – Long Polling Mode (метод постійного опитування сервера Telegram для отримання нових команд бота)

python-dotenv – розширена бібліотека для зчитування конфігураційних параметрів з ізольованого файлу оточення

Python – мова програмування високого рівня, використана для реалізації програмної системи

ROI – Region Of Interest (область інтересу на кадрі для обмеження зони аналізу)

RotatingFileHandler – Rotating File Handler (інструмент циклічного журналювання з обмеженням розміру файлів логів)

RTSP – Real Time Streaming Protocol (протокол потокового передавання даних у реальному часі)

SQLite – Structured Query Language Lite (вбудована легковагова реляційна система керування базами даних)

VideoCapture – Video Capture Interface (інтерфейс OpenCV для захоплення та зчитування послідовності кадрів)

YOLO – You Only Look Once (нейромережева модель одноетапного виявлення та розпізнавання об’єктів)

ВСТУП

Комп’ютерний зір є одним із напрямів інформаційних технологій та штучного інтелекту, що забезпечує автоматичне отримання, оброблення й аналіз зображень або відеопотоків. Основним завданням комп’ютерного зору є виділення корисної інформації з візуальних даних для подальшого прийняття рішень програмною системою. До таких задач належать виявлення руху, розпізнавання об’єктів, аналіз контурів, визначення положення об’єктів у кадрі та формування повідомлень про зафіксовані події.

Одним із практичних напрямів застосування комп’ютерного зору є автоматизоване відеоспостереження. Такі системи використовуються для контролю житлових, офісних, навчальних і виробничих приміщень. Традиційні системи відеоспостереження переважно виконують функцію запису або передавання відеопотоку, тому виявлення важливих подій часто потребує постійної участі користувача або подальшого перегляду збережених матеріалів. Унаслідок цього знижується оперативність реагування на події, що відбуваються в зоні спостереження.

Сучасний розвиток інформаційних технологій дає змогу доповнювати системи відеоспостереження засобами автоматичного аналізу відеопотоку. Застосування методів фонового віднімання, аналізу контурів і розпізнавання об’єктів дозволяє виявляти рух у кадрі, визначати область активності та встановлювати наявність людини. Завдяки цьому система може реагувати не на весь відеопотік, а лише на події, що відповідають заданим умовам.

Актуальність роботи полягає у необхідності розроблення програмної системи відеомоніторингу, здатної автоматично виявляти рух, розпізнавати присутність людини в кадрі та оперативно передавати повідомлення користувачу. Такий підхід дає змогу зменшити потребу в постійному ручному контролі відеопотоку, підвищити швидкість реагування на події та забезпечити зручний спосіб віддаленого інформування.

Необхідність розроблення системи зумовлена тим, що наявність руху в кадрі не завжди свідчить про появу людини. Спрацювання можуть бути спричинені змінами освітлення, тінями, шумами камери або коливанням окремих об’єктів. Тому доцільним є поєднання методів виявлення руху з алгоритмами розпізнавання людини. Це дає змогу підвищити інформативність повідомлень і зменшити кількість незначущих спрацювань.

Тема кваліфікаційної роботи – розроблення програмної системи відеоспостереження з виявленням руху, розпізнаванням людини в кадрі та Telegram-сповіщенням користувача.

Основні проєктні рішення полягають у використанні мови програмування Python, бібліотеки OpenCV для оброблення відеопотоку, методів фонового віднімання та аналізу контурів для виявлення руху, моделі YOLO у форматі ONNX для розпізнавання людини, а також бібліотеки aiogram для реалізації Telegram-бота. Для збереження параметрів роботи системи та історії подій застосовано формат JSON. Такий підхід забезпечує модульність програмної системи, можливість зміни налаштувань і передавання сповіщень у віддаленому режимі.

Напрямами використання результатів кваліфікаційної роботи є домашні системи безпеки, офісний моніторинг, контроль навчальних приміщень, спостереження за робочими зонами та подальше розроблення програмних засобів відеоаналітики. Розроблена система може бути використана як приклад, прикладного застосування методів комп’ютерного зору та штучного інтелекту для автоматизації відеоспостереження.

1 СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Історія та розвиток систем відеоспостереження

Відеоспостереження є одним із основних напрямів технічного забезпечення безпеки, що сформувався внаслідок розвитку оптичних пристроїв, засобів передавання сигналу, цифрового оброблення зображень та інформаційних систем. Спочатку такі системи виконували переважно функцію візуального контролю, коли оператор безпосередньо спостерігав за зображенням з камери. Згодом до процесу було додано запис відеоматеріалів, що дало змогу зберігати інформацію про події та аналізувати її після виникнення інциденту. Проте навіть за наявності запису основне навантаження залишалося на користувачеві, оскільки виявлення важливих моментів потребувало ручного перегляду великого обсягу відеоданих.

Розвиток аналогових систем відеоспостереження був пов’язаний із використанням камер, коаксіальних кабелів, моніторів та пристроїв запису. Такі рішення забезпечували безперервний контроль об’єкта, однак мали обмеження щодо якості зображення, масштабування та автоматичного аналізу подій. У разі потреби контролю декількох зон одночасно необхідно було збільшувати кількість обладнання та залучати оператора, здатного обробляти візуальну інформацію в реальному часі. Унаслідок цього традиційна модель відеоспостереження залишалася залежною від людини.

Перехід до цифрових відеореєстраторів дав змогу підвищити якість збереження даних, спростити пошук відеофрагментів та забезпечити гнучкіше керування відеозаписами. Цифрове подання відеопотоку створило передумови для подальшої автоматизації, оскільки кадри стало можливим обробляти програмними засобами. Саме на цьому етапі почали активно застосовуватися алгоритми виявлення руху, які порівнювали поточний кадр із попереднім або з еталонним фоном і визначали області змін.

Подальший розвиток мережевих технологій спричинив поширення IP-камер і систем віддаленого доступу. Завдяки цьому відеопотік став доступним через локальну мережу або інтернет, а користувач отримав змогу переглядати зображення з різних пристроїв. Водночас простий віддалений перегляд не розв'язує проблему автоматичного виявлення подій. Наявність підключення до мережі забезпечує передавання відео, але не гарантує його змістовного аналізу. Тому наступним етапом розвитку стали системи відеоаналітики, орієнтовані на автоматичну інтерпретацію кадрів.

Сучасні системи відеоспостереження дедалі частіше поєднуються з методами комп'ютерного зору. У таких системах джерелом даних є відеокамера, а програмний модуль виконує попереднє оброблення зображення, виділення рухомих об'єктів, аналіз форми, розпізнавання класів об'єктів та формування повідомлень. Такий метод змінює цілі системи, оскільки він не тільки зберігає відео, але й виконує основне прийняття рішень відповідно до визначених правил.

Можливість автоматичного сповіщення користувача є особливо важливою. Подія може бути виявлена запізненням, якщо система лише записує відео. Користувач може швидше відреагувати, якщо система створює повідомлення одразу після виявлення руху або об'єкта в кадрі. Відеоаналітика, таким чином, поширюється в сучасних прикладних рішеннях разом з мобільними сервісами, повідомленнями, електронною поштою та чат-ботами.

З огляду на розвиток програмних бібліотек, камер загального призначення та відкритих моделей комп'ютерного зору, створення власної системи відеомоніторингу стало можливим без використання складного спеціалізованого обладнання. Для невеликих об'єктів спостереження достатньо персонального комп'ютера, вебкамери або IP-камери, програмного модуля виявлення руху та каналу сповіщення користувача. Такий підхід є доцільним для домашнього використання, контролю навчальних приміщень, невеликих офісів і робочих зон.

Розвиток систем відеоспостереження демонструє поступовий перехід від пасивного запису до активного аналізу відеопотоку. У результаті актуальним стає розроблення програмної системи, що поєднує детекцію руху, розпізнавання людини та надсилання повідомлень користувачу. Узагальнену еволюцію підходів до відеоспостереження зображено на рисунку 1.1.

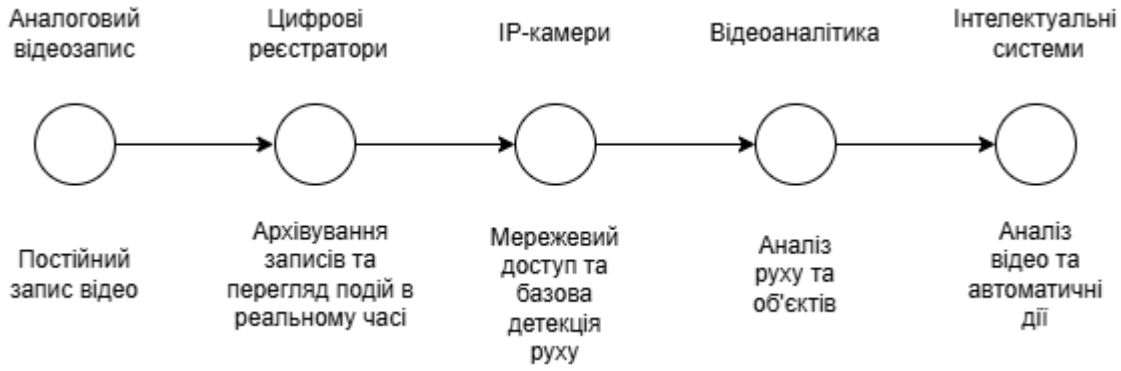

Рисунок 1.1 – Еволюція підходів до відеоспостереження

Більшість досліджень у сфері відеоспостереження розвиваються у двох взаємопов’язаних напрямах. Перший напрямок пов’язаний із підвищенням якості виявлення рухомих об’єктів у складних умовах освітлення, зміни фону, наявності тіней та шумів. Другий напрям передбачає семантичне розпізнавання об’єктів, зокрема людини, транспортних засобів або небезпечних ситуацій. У працях, присвячених глибинному навчанню для відеоспостереження, наголошується, що сучасні моделі можуть бути ефективними для виявлення аномальних подій, однак їх застосування часто пов’язане з потребою у значних обчислювальних ресурсах та підготовлених наборах даних [1].

1.2 Аналіз підходів до виявлення руху у відеопотоці

Виявлення руху є базовим етапом більшості систем відеоспостереження, оскільки саме він визначає момент, коли відеопотік потребує подальшого

аналізу. Виділяють кілька основних груп методів: порівняння послідовних кадрів, фонове віднімання, оптичний потік і методи на основі машинного навчання. Кожна група має власну область ефективного застосування, тому вибір методу залежить від умов спостереження, вимог до швидкодії та характеру очікуваних подій.

Метод порівняння кадрів базується на обчисленні різниці між двома або кількома послідовними зображеннями. Якщо різниця між пікселями перевищує встановлений поріг, відповідна область вважається рухомою. Перевагою цього підходу є простота та невисока обчислювальна складність. Проте метод має істотні обмеження: він чутливий до шумів, не забезпечує стабільного виділення повної форми об’єкта та може втрачати об’єкти, які рухаються повільно або тимчасово зупиняються.

Фонове віднімання передбачає побудову моделі статичного фону сцени та порівняння поточного кадру з цією моделлю. Області, які відрізняються від фону, визначаються як передній план. У роботі [2] розглянуто методи цифрового оброблення зображень, які є основою для попереднього аналізу кадрів, виділення інформативних ознак і подальшого розпізнавання візуальних даних. У роботі [3] проаналізовано застосування моделей класифікації відеоданих із використанням згорткових нейронних мереж, що підтверджує доцільність використання інтелектуальних методів для аналізу відеопотоку.

Водночас фонове віднімання має низку обмежень. У реальних умовах фон не завжди залишається сталим: освітлення може змінюватися протягом доби, у кадрі можуть з’являтися тіні, відблиски, рухомі елементи інтер’єру або дрібні шуми камери. У таких випадках алгоритм може формувати помилкові спрацювання. Фонове віднімання може розглядатися як базовий етап відеоаналітики, який забезпечує прийнятне співвідношення між швидкодією та якістю виявлення руху, особливо за умови використання статичної камери та відносно стабільного фону.

Методи оптичного потоку визначають напрямок і швидкість переміщення пікселів між кадрами. Їх перевагою є можливість отримання детальної інформації про характер руху. Проте обчислення оптичного потоку є складнішим порівняно з порівнянням кадрів або фоновим відніманням. З огляду на це такі методи доцільні у задачах, де потрібно аналізувати траєкторії або поведінку об’єктів, але вони не завжди є оптимальними для простої подієвої системи спостереження.

Сучасні нейромережеві підходи дозволяють поєднувати виявлення руху з розпізнаванням об’єктів або сегментацією сцени. Однак такі методи потребують більшої кількості обчислювальних ресурсів і часто потребують підготовки або добору навчальних даних. Унаслідок цього для локальної системи відеомоніторингу доцільним є використання комбінації простіших методів виявлення руху з подальшою перевіркою семантичного змісту події.

Як видно з порівняння, жоден із підходів не є універсальним. Порівняння кадрів добре працює в простих умовах, але недостатньо добре працює в шумних місцях. Однак фонове віднімання потребує фільтрації помилкових областей, що робить його кращим для статичної камери. Хоча оптичний потік надає детальні дані про рух, він вимагає більшої обчислювальної роботи. Нейромережеві методи забезпечують більшу інформативність, вони можуть бути надмірними для систем локального моніторингу розміром до невеликої.

У дослідженнях, присвячених виявленню руху, простежується позиція, що попереднє визначення активних зон сцени є необхідним. Подібність поглядів авторів полягає в тому, що рух розглядається як основна характеристика події, після чого слід уточнити зміст кадру. У цих дослідженнях відрізняються методи виділення, які використовуються: одні дослідження надають перевагу простим різницевим методам, тоді як інші дослідження зосереджуються на адаптивному фоновому моделюванні або оцінюванні оптичного потоку.

Суперечливість результатів пояснюється різними умовами проведення експериментів. Методи, які демонструють високу точність у стабільних лабораторних умовах, можуть втрачати ефективність у приміщеннях із нерівномірним освітленням, рухомими тінями або відблисками. Водночас прості алгоритми можуть забезпечувати достатній результат у локальних системах, якщо їх доповнено пороговою фільтрацією, аналізом площі рухомої області та перевіркою повторюваності спрацювань.

Отже, для прикладної системи відеомоніторингу доцільним є не ізольоване використання одного підходу, а поєднання первинної детекції руху з подальшою перевіркою значущості події. Такий висновок узгоджується з практичним характером роботи, оскільки система має забезпечувати не максимальну повноту наукового експерименту, а своєчасне виявлення подій у реальних умовах експлуатації.

1.3 Аналіз підходів до розпізнавання людини у відеопотоці

Виявлення руху саме по собі не дає змоги встановити зміст події. Рух у кадрі може бути спричинений появою людини, переміщенням тварини, зміною освітлення, тінню або коливанням предметів. Тому в системах відеоспостереження важливим є додатковий етап розпізнавання людини. Його використання підвищує інформативність повідомлення та дає змогу відрізняти потенційно значущі події від незначущих змін у кадрі.

У навчальному посібнику [4] розглянуто базові методи та алгоритми комп’ютерного зору, які застосовуються для аналізу зображень, виділення ознак, опису об’єктів і подальшого розпізнавання. Зазначений підхід є важливим для задач відеомоніторингу, оскільки кожний кадр відеопотоку може розглядатися як окреме зображення, що потребує попереднього оброблення, виділення інформативних ознак і прийняття рішення про

наявність об’єкта. Отже, розпізнавання людини у відеопотоці ґрунтується не лише на факті руху, а й на аналізі візуальних характеристик об’єкта.

Класичні підходи комп’ютерного зору передбачають використання ознак форми, контурів, текстури, градієнтів і просторового розміщення елементів зображення. Їх перевагою є відносна простота, зрозуміла логіка роботи та можливість застосування на обладнанні з обмеженими обчислювальними ресурсами. Водночас такі підходи залежать від якості вхідного зображення, положення об’єкта, масштабу, освітлення та наявності часткових перекриттів. Унаслідок цього класичні методи можуть бути ефективними в контрольованих умовах, але їх стійкість знижується у складних сценах.

Формування простору ознак у методах класифікації зображень, розглянуто у роботі [5]. Такий напрям є важливим для розпізнавання людини, оскільки якість класифікації значною мірою залежить від того, наскільки повно ознаки описують об’єкт. Якщо ознаковий опис недостатньо відображає форму, структуру або характерні елементи зображення, точність розпізнавання знижується. Отже, для задач відеоспостереження важливим є не лише сам факт виділення об’єкта, а й вибір такого подання ознак, яке дає змогу відокремити людину від інших рухомих об’єктів.

Аналіз класичних підходів показує, що вони мають практичну цінність завдяки невисоким вимогам до ресурсів і можливості використання як допоміжних засобів розпізнавання. Проте їх результати можуть змінюватися залежно від умов зйомки. Людина може перебувати в кадрі в різних позах, на різній відстані від камери, частково перекриватися іншими об’єктами або потрапляти в область зі складним освітленням. Тому класичні методи доцільно розглядати як базовий або резервний рівень аналізу, а не як єдину основу для розпізнавання людини у відеопотоці.

Нейромережеві підходи відрізняються тим, що ознаки об’єкта формуються автоматично під час навчання моделі. У роботі [3] проаналізовано застосування моделей класифікації відеоданих із використанням згорткових

нейронних мереж. Такі моделі здатні враховувати складні просторові залежності у зображеннях, що є важливим для аналізу відеопотоку. Завдяки цьому нейромережеві підходи краще пристосовані до різних положень об’єктів, зміни масштабу та складного фону.

Використання нейромережевих моделей не вирішує всі проблеми, пов’язані з розпізнаванням людини у відеопотоці. Якість навчальних даних, умови зйомки, роздільна здатність кадру, освітлення, кут розміщення камери та обчислювальні можливості пристрою впливають на те, наскільки добре вони працюють. Крім того, якщо умови спостереження значно відрізняються від умов, представлених у навчальних даних, модель, навчена на загальних зображеннях, може показати нижчу якість роботи в конкретному середовищі.

Наявність розбіжностей між якістю розпізнавання та простотою реалізації видно з порівняння класичних і нейромережевих підходів. Класичні методи простіші та потребують менших ресурсів, але вони не так стійкі до складних умов. Хоч нейромережеві моделі пропонують більшу здатність до узагальнення, їх навчання та адаптація вимагає більшої кількості обчислювальних ресурсів і якісної інформації. Порівняння підходів показує, що жоден із методів не є універсальним (рис. 1.2).

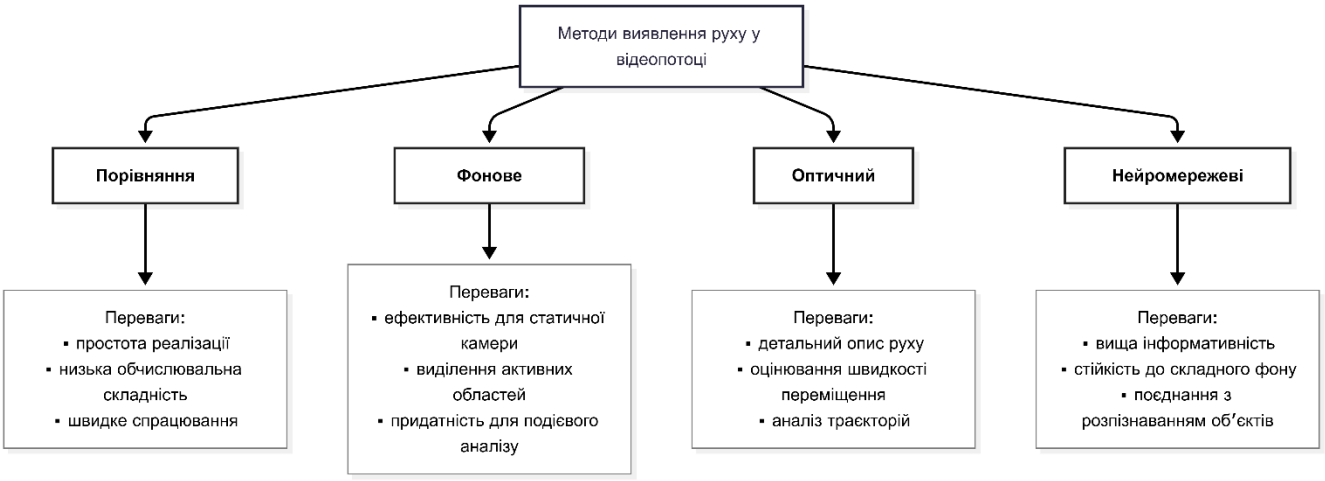

Рисунок 1.2 – Основні методи виявлення руху та їх переваги

Таким чином, власний підхід до задачі полягає в тому, що розпізнавання людини має виконуватися після встановлення факту руху. Спочатку система визначає наявність активності в кадрі, після чого виконується уточнювальний аналіз, спрямований на встановлення присутності людини. Така послідовність обмежує кількість кадрів, які потребують поглибленого аналізу, зменшує обчислювальне навантаження та підвищує практичну доцільність системи для локального використання.

Узагальнення розглянутих підходів дає змогу зробити висновок, що для прикладної системи відеомоніторингу доцільним є комбінований підхід. Детекція руху має виконувати роль первинного фільтра подій, а розпізнавання людини – уточнювального етапу, який підвищує змістову точність сповіщення. Така логіка дає змогу не виконувати складний аналіз для кожного кадру, а зосередити оброблення лише на потенційно значущих фрагментах відеопотоку.

1.4 Критичний аналіз існуючих рішень для задач відеомоніторингу

Існуючі рішення для відеоспостереження можна умовно поділити на кілька груп: традиційні системи відеозапису, системи з базовою детекцією руху, хмарні камери з віддаленим доступом, інтелектуальні системи відеоаналітики та локальні подієві системи сповіщення. Кожна група розв’язує частину задачі, однак не завжди забезпечує оптимальне поєднання доступності, автономності, гнучкості налаштувань і зручності інформування користувача.

Традиційні системи відеозапису забезпечують збереження відеоматеріалів і можливість їх подальшого перегляду. Їх перевагою є стабільність і відносна простота використання. Однак основним недоліком залишається пасивний характер роботи: система фіксує події, але не завжди

здатна самостійно визначити їх значущість і повідомити користувача. У результаті відповідальність за аналіз відео переноситься на людину.

Критичне оцінювання існуючих рішень показує, що традиційний відеозапис забезпечує повноту фіксації, але не розв’язує задачу оперативного виявлення значущих подій. У такому підході основне навантаження переноситься на користувача, який має самостійно переглядати матеріали або постійно спостерігати за трансляцією. Це знижує ефективність системи в умовах, коли потрібно швидко реагувати на появу людини в зоні контролю.

Рішення з базовою детекцією руху частково усувають зазначене обмеження, оскільки дають змогу переходити від безперервного запису до подієвої моделі роботи. Проте реакція лише на факт зміни в кадрі не завжди є достатньою. У реальних умовах рух може бути спричинений не тільки людиною, а й освітленням, шумом, дрібними предметами або іншими незначущими факторами.

Хмарні системи забезпечують зручний доступ і централізоване зберігання матеріалів, але водночас створюють залежність від зовнішньої інфраструктури. Для локального відеомоніторингу така залежність не завжди є прийнятною, оскільки стабільність роботи системи має зберігатися навіть за умов обмеженого або нестабільного мережевого з’єднання.

Системи з базовою детекцією руху дають змогу зменшити обсяг збережуваних матеріалів і формувати записи лише за наявності змін у кадрі. Проте більшість таких рішень реагує на будь-який рух. Зміни освітлення, тіні, рух штор, відблиски або шум камери можуть спричиняти зайві повідомлення. Отже, без додаткового розпізнавання об’єктів детекція руху залишається недостатньо інформативною.

Хмарні камери та сервіси з віддаленим доступом забезпечують зручність перегляду матеріалів через мобільні застосунки. Їхньою перевагою є готова інфраструктура зберігання та сповіщення. Проте залежність від зовнішніх сервісів створює обмеження, пов’язані з оплатою, конфіденційністю, доступністю мережі та неможливістю повного контролю над алгоритмами

оброблення відео. Для користувачів, які потребують локального збереження подій, такі рішення не завжди є прийнятними.

Інтелектуальні системи відеоаналітики з використанням глибинного навчання можуть забезпечувати виявлення людей, транспортних засобів, підозрілих дій або аномальної поведінки. Глибинне навчання відкриває можливість аналізу складних сценаріїв, але потребує якісних навчальних даних, ретельного налаштування та значних обчислювальних ресурсів. Для невеликих об’єктів спостереження така складність може бути надмірною.

Локальні подієві системи сповіщення є проміжним варіантом між простим відеозаписом і складною інтелектуальною аналітикою. Вони можуть працювати на персональному комп’ютері, використовувати камеру як джерело відеопотоку, аналізувати кадри в реальному часі та надсилати повідомлення користувачу. Саме такий підхід є доцільним для кваліфікаційної роботи, оскільки він дозволяє розв’язати прикладну задачу автоматизованого контролю без створення надмірно складної системи аналізу поведінки.

Аналіз існуючих рішень показує, що основна проблема полягає не лише у виявленні руху, а й у визначенні значущості події для користувача. Застосунок має не просто зберігати відео, а виконувати попередній аналіз, відбирати потенційно важливі фрагменти та забезпечувати оперативне повідомлення. Тому обґрунтованим є вибір напряму, у якому поєднуються детекція руху, перевірка присутності людини, збереження подій і дистанційне інформування.

1.5 Постановка задачі

Автоматизоване виявлення руху та розпізнавання присутності людини у відеопотоці є актуальним завданням у контексті розвитку систем комп’ютерного зору та прикладних засобів безпеки. Проведений аналіз показав, що традиційне відеоспостереження не забезпечує достатньої

оперативності реагування, тоді як складні системи відеоаналітики можуть бути надмірними для невеликих локальних об’єктів. Тому доцільним є створення програмної системи, яка поєднує виявлення руху, перевірку наявності людини в кадрі, збереження події та передавання повідомлення користувачу.

Об’єктом роботи є розроблення програмної системи відеоспостереження з автоматичним виявленням руху та сповіщенням користувача.

Метою роботи є розроблення програмного застосунку для відеомоніторингу, розпізнавання людини в кадрі та надсилання сповіщень через Telegram-бота.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– проаналізувати сучасний стан проблемної області автоматизованого відеоспостереження;

– розглянути наукові підходи до виявлення руху у відеопотоці;

– проаналізувати методи розпізнавання людини в кадрі та визначити їхні переваги й обмеження;

– розробити структуру програмної системи для захоплення відеопотоку, виявлення руху та розпізнавання людини;

– реалізувати механізм збереження фото- або відеоматеріалів за фактом виявлення події;

– забезпечити надсилання повідомлень користувачу про виявлені події;

– передбачити можливість налаштування основних параметрів роботи системи;

– провести тестування розробленого застосунку та оцінити отримані результати.

2 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДІВ ТА МОДЕЛЮВАННЯ СИСТЕМНИХ ЗВ'ЯЗКІВ ПРОГРАМНОЇ СИСТЕМИ ВІДЕОСПОСТЕРЕЖЕННЯ

2.1 Декомпозиція задачі та критерії вибору методів

Другий розділ присвячено деталізації змісту задач, які потребують вирішення під час розроблення програмної системи відеоспостереження Camera. На відміну від простого опису готових модулів, у розділі виконано порівняння можливих підходів до окремих частин задачі, обґрунтовано вибір методів, змодельовано системні зв'язки та визначено роль кожного компонента у загальному процесі формування події. Теоретичною основою такого аналізу є методи цифрового оброблення зображень і структурного розпізнавання візуальних даних [2, 6].

Загальна задача автоматизованого відеоспостереження складається з кількох локальних задач: отримання кадру з камери, попереднє оброблення відеопотоку, виявлення руху, розпізнавання людини, формування фото- або відеоподії, збереження результатів, передавання сповіщення користувачу та налаштування параметрів роботи системи. Кожна задача може бути розв'язана різними способами, тому вибір підходу має бути обґрунтований вимогами до швидкодії, точності, автономності, простоти супроводу та відповідності фактичній реалізації проєкту. Для обґрунтування вибору також враховано підходи до пошуку й подання візуальних об'єктів у структурних методах класифікації [7, 8].

Основою обраного підходу є комбінування методів класичного комп'ютерного зору та нейромережевого розпізнавання об'єктів. На першому рівні застосовується виявлення руху, морфологічного оброблення та аналізу контурів. Цей рівень виконує роль швидкого фільтра кадрів, що дає змогу відокремити потенційно значущі фрагменти відеопотоку від статичних кадрів.

Далі використовується нейромережева модель, яка визначає наявність людини в кадрі та забезпечує змістову перевірку події.

Для нашої задачі основними обмеженнями є локальне виконання на персональному комп’ютері, робота з камерою загального призначення, можливість використання без спеціалізованого серверного обладнання, підтримка сповіщень, можливість резервного розпізнавання без підключеної нейромережевої моделі та збереження параметрів у файлах конфігурації. Саме ці обмеження визначають доцільність комбінованого підходу: прості методи застосовуються як швидкий первинний фільтр, а складніші методи використовуються лише після появи потенційно значущої події [3, 9].

Декомпозицію задачі відеомоніторингу подано на рисунку 2.1.

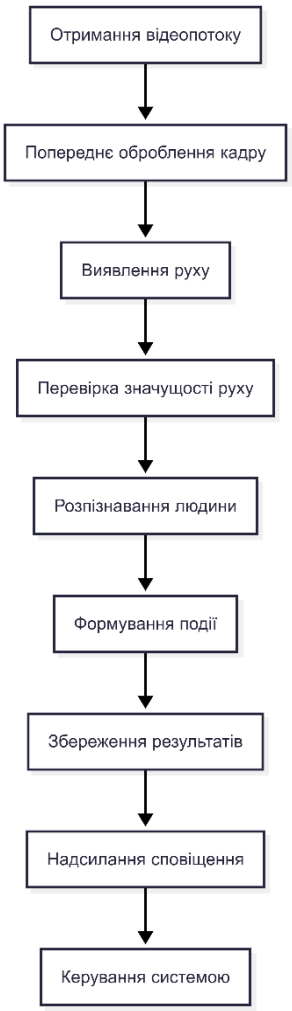

Рисунок 2.1 – Декомпозиція задачі відеомоніторингу

Розроблювана система не є лише програмою для запису відео. Її робота передбачає послідовне прийняття рішень: спочатку визначається наявність змін у кадрі, потім перевіряється зміст події, після чого формується матеріал для користувача. Такий підхід зменшує кількість зайвих записів і обмежує складне розпізнавання лише кадрами, у яких попередньо зафіксовано рух [10, 11].

Також потрібно визначити критерії вибору підходів до локальних задач, та визначити їх важливість. Критерії показані у таблиці 2.1.

На основі зазначених критеріїв у подальшому розглянуто альтернативні підходи до кожної частини задачі, а також обгрунтовано, чому у проєкті обрано саме реалізовану комбінацію методів. Додатково враховано підходи до перетворення описів зображень у класифікаційні ознаки та застосування відеоаналізу для задач підрахунку й контролю об’єктів у потоках із камер спостереження [12, 13].

Таблиця 2.1 – Критерії вибору підходів до локальних задач

Критерій	Зміст критерію	Значення для проєкту
1	2	3
Швидкодія	Час оброблення кадру та можливість роботи в основному циклі моніторингу без значних затримок	Важлива, оскільки кадри обробляються безперервно
Точність	Здатність відрізняти значущу подію від шуму, тіні або випадкової зміни освітлення	Важлива для зменшення кількості хибних сповіщень
Автономність	Можливість роботи та без передавання відеопотоку сторонній інфраструктурі	Важлива для локального відеомоніторингу

Продовження таблиці 2.1

1	2	3
Простота налаштування	Можливість змінювати параметри без редагування програмного коду	Важлива для зручного та швидкого налаштування
Резервування	Збереження працездатності за відсутності додаткової моделі або нестабільності окремого компонента	Важлива, оскільки нейромережева модель може бути відсутньою або неправильно працювати
Масштабованість	Можливість додавання нових режимів, параметрів, детекторів або каналів сповіщення	Середньо важливо для тривалого користування

2.2 Моделювання системних зв’язків та вибір архітектури

Наступною локальною задачею є організація взаємодії модулів. Для системи відеомоніторингу можливі різні архітектурні підходи: монолітний синхронний цикл, модульна синхронна структура та асинхронна модульна структура.

Монолітний цикл є простим, але ускладнює супровід і розширення. Модульна синхронна структура краще розділяє відповідальність, однак може блокувати виконання під час надсилання повідомлень або роботи з файлами. Асинхронна модульна структура дає змогу розділити основний моніторинг, Telegram-бота, очищення сховища та чергу сповіщень, порівняння цих підходів подано у таблиці 2.2 [14, 15].

У системі обрано асинхронну модульну архітектуру. Вона передбачає окреме виконання циклу відеомоніторингу, оброблення черги сповіщень, взаємодії з Telegram-ботом і службового очищення сховища. Сповіщення

передаються через асинхронну чергу, що відокремлює момент виявлення події від моменту фактичного надсилання повідомлення. Такий підхід запобігає блокуванню основного циклу оброблення кадрів під час мережевої взаємодії [16, 17].

Таблиця 2.2 – Порівняння архітектурних підходів

Підхід	Переваги	Недоліки	Результат вибору
Монолітний цикл	Найпростіша реалізація	Складне розширення, ризик блокування під час надсилання файлів	Не обрано
Модульна синхронна структура	Розділення відповідальності між файлами	Операції Telegram і очищення сховища можуть затримувати цикл	Частково використано на рівні класів
Асинхронна модульна структура	Паралельна робота бота, моніторингу, очищення та черги сповіщень	Вища складність керування задачами	Обрано як основну архітектуру

У системі повинні бути виділені окремі компоненти для захоплення кадрів, детекції руху, розпізнавання людини, керування станом, взаємодії з Telegram, локального інтерфейсу та файлового сховища. Структуру взаємодії модулів показано на рисунку 2.2.

Такий поділ зменшує зв’язність модулів і полегшує заміну окремого підходу без повного переписування системи. Призначення основних модулів системи подано у таблиці 2.3.

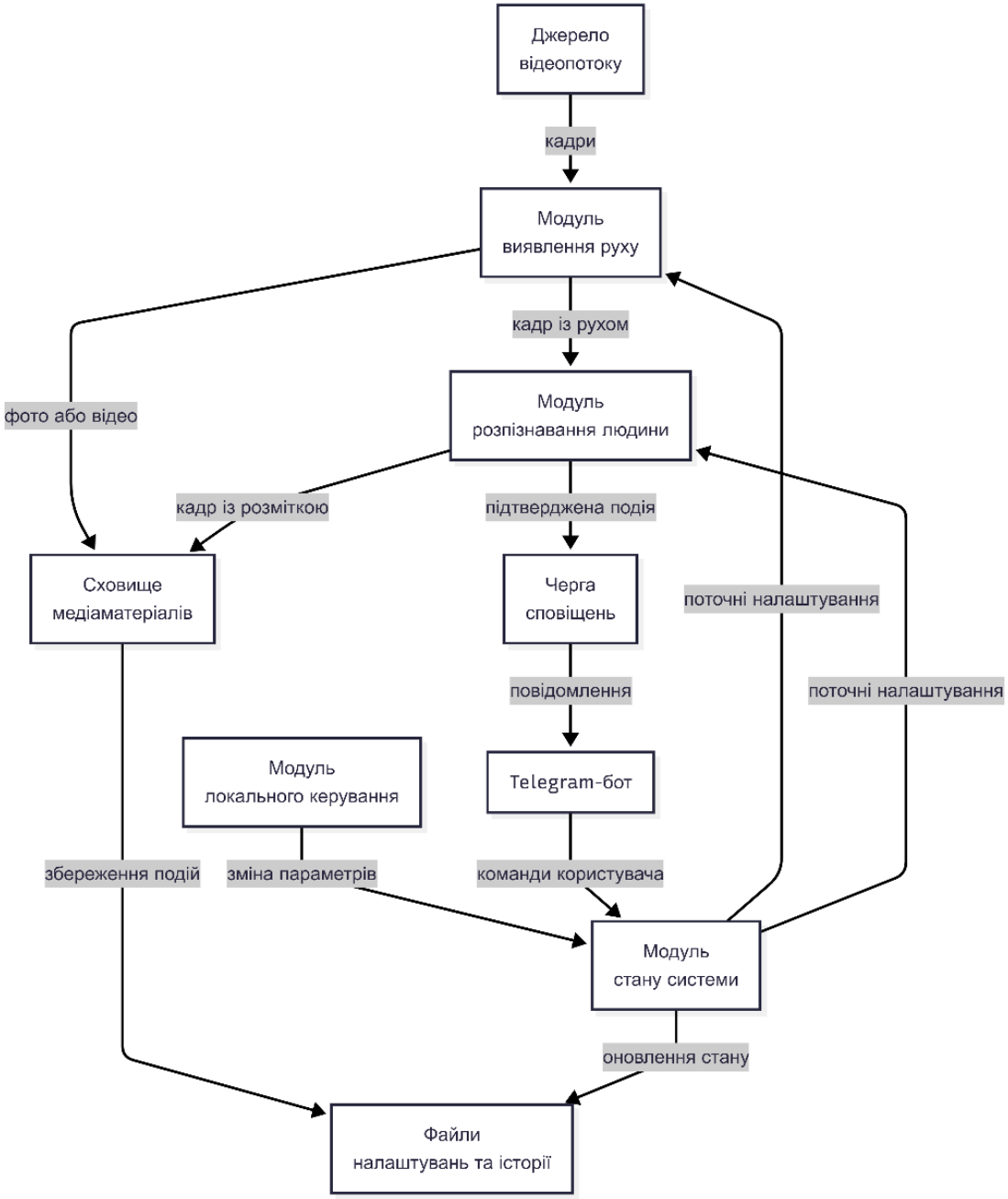

Рисунок 2.2 – Структура взаємодії модулів програмної системи

Таблиця 2.3 – Призначення основних модулів системи

Модуль	Призначення	Результат роботи
1	2	3
Координаційний модуль	Узгоджує роботу циклу відеомоніторингу, черги сповіщень, бота та службових процесів	Керована взаємодія компонентів системи

Продовження таблиці 2.3

1	2	3
Модуль детекції руху	Отримує кадри, застосовує область інтересу, фонове віднімання, контурний аналіз і формування матеріалу події	Ознака руху та підготовлений матеріал події
Модуль розпізнавання людини	Виконує перевірку кадру засобами нейромережевого або резервного класичного підходу	Результат встановлення наявності людини
Модуль Telegram-сповіщень	Обробляє команди користувача, контролює доступ і надсилає повідомлення з медіаматеріалами	Віддалене керування та інформування
Модуль стану системи	Зберігає налаштування, користувачів, розклад і журнал подій	Актуальний стан застосунку
Модуль локального керування	Надає можливість керування через системний інтерфейс без редагування програмного коду	Зміна параметрів у локальному режимі

2.3 Порівняння підходів до отримання та попереднього оброблення відеопотоку

Першою локальною задачею є отримання відеоданих. Для програмної системи відеоспостереження можуть застосовуватися різні підходи: робота з локальною камерою через драйвер операційної системи, підключення до мережевого відеопотоку IP-камери та оброблення попередньо записаного відеофайлу.

Кожний підхід має різне призначення. Відеофайл є зручним для тестування, але не забезпечує реального моніторингу, що для системи є критичним. IP-камера підходить для розподілених систем, однак потребує стабільної мережі, та не зосвім підходить для розроблюємої системи моніторингу, але може використатися як доповнення системи. Локальна камера забезпечує найпростішу інтеграцію для навчального та прикладного застосунку [18].

Для розв’язання задачі отримання кадрів обрано інтерфейс захоплення відео OpenCV. Такий вибір є оптимальним для локального застосунку, оскільки не потребує додаткового мережевого протоколу, забезпечує прямий доступ до кадрів і дає змогу змінювати джерело відеопотоку без перебудови архітектури системи, на відміну від IP- камери або хмарної камери. Порівняння джерел відеопотоку подано у таблиці 2.4.

Таблиця 2.4 – Порівняння підходів до отримання відеопотоку

Підхід	Переваги	Обмеження	Доцільність для системи
1	2	3	4
Локальна камера через OpenCV	Пряма інтеграція, просте налаштування, низька затримка	Залежність від доступності пристрою на комп’ютері	Обрано як основний підхід
IP-камера або RTSP-потік	Віддалене розміщення камери, можливість мережевого моніторингу	Потреба у стабільній мережі та додаткових параметрах підключення	Може бути розширенням системи

Продовження таблиці 2.4

1	2	3	4
Попередньо записаний відеофайл	Зручність тестування	Не забезпечує спостереження в реальному часі	Доцільно лише для тестових сценаріїв
Хмарна камера	Готова інфраструктура доступу та збереження	Залежність від стороннього сервісу та можливі обмеження конфіденційності	Не обрано для локального застосунку

Відеопотік розглядається як дискретна послідовність кадрів, що надходять до основного циклу оброблення.

$$V := \{F_t \mid t: 0,1,2, \dots, T\}, \tag{2.1}$$

де V – відеопотік;

F_t – кадр у момент часу t;

T – кількість кадрів, оброблених під час роботи системи.

Після отримання кадру виконується попереднє оброблення. Можливими підходами є аналіз кольорового кадру, перехід до відтінків сірого, просторове згладжування, виділення області інтересу та масштабування, порівняння подано у таблиці 2.5.

Для розроблюваної системи обрано перетворення кадру у відтінки сірого, згладжування фільтром GaussianBlur і можливість використання області інтересу. Такий вибір пояснюється тим, що для виявлення руху достатньо аналізувати зміну інтенсивності, а не повну кольорову інформацію [2, 18].

Таблиця 2.5 – Порівняння підходів до попереднього оброблення кадру

Підхід	Призначення	Переваги	Прийняте рішення
Аналіз кольорового кадру	Використання всіх каналів зображення	Збереження повної інформації про сцену	Не обрано для детекції руху через надлишковість даних
Відтінки сірого	Зменшення кількості каналів до одного	Менше обчислень і достатня інформативність для руху	Обрано як базовий етап
GaussianBlur	Зменшення шуму перед фоновим відніманням	Зменшує випадкові дрібні контури	Обрано як засіб попереднього згладжування
ROI	Обмеження аналізу заданою ділянкою кадру	Зменшує хибні спрацювання в неважливих зонах	Обрано як параметр, що може вмикатися
Масштабування кадру	Приведення кадру до фіксованого розміру	Може прискорити оброблення	Використовується в розпізнаванні, а не в первинній детекції

Область інтересу дозволяє виключити частини кадру, у яких рух не має прикладного значення. Наприклад, у приміщенні може бути рух у зоні вікна або екрана, який не повинен спричиняти повідомлення.

У системі область інтересу задається параметрами положення та розмірів, які можуть змінюватися відповідно до умов спостереження.

$$R_t = F_t[x_0 : x_0 + w,\ y_0 : y_0 + h], \tag{2.2}$$

де R_t – область інтересу поточного кадру;

x_0, y_0 – координати лівого верхнього кута області;

w, h – ширина та висота області.

2.4 Порівняння підходів до виявлення руху

Виявлення руху є основною локальною задачею, оскільки саме воно визначає момент, коли кадр потребує подальшого аналізу. Для цієї задачі можуть застосовуватися різниця між послідовними кадрами, фонове віднімання, оптичний потік, нейромережева сегментація або постійне розпізнавання кожного кадру без окремої детекції руху. У разі подальшого розширення системи можуть використовуватися моделі класифікації та компактного подання ознак, однак для первинного фільтра подій достатнім є менш ресурсомісткий підхід [19 – 22].

Просте порівняння кадрів має невисоку складність, однак погано працює для повільного руху та не формує стабільної моделі фону. Оптичний потік дає змогу оцінити напрям і швидкість руху, але потребує більше обчислень і є надмірним для задачі сповіщення про подію. Нейромережева сегментація може бути точнішою, проте вона збільшує вимоги до ресурсів. Для локальної системи з камерою загального призначення оптимальним є фонове віднімання з подальшою фільтрацією маски.

Для розроблюваної системи обрано фонове віднімання MOG2, реалізоване засобами OpenCV. Його параметри мають налаштовуватися залежно від умов сцени, тому в теоретичному обґрунтуванні не фіксуються конкретні числові значення.

Після побудови маски використовується порогова обробка, морфологічне відкриття, розширення та аналіз зовнішніх контурів. Порівняння підходів до детекції руху подано у таблиці 2.6 [18].

Таблиця 2.6 – Порівняння підходів до виявлення руху

Підхід	Переваги	Недоліки	Оцінка доцільності
Різниця послідовних кадрів	Простота реалізації та висока швидкодія	Чутливість до шуму, нестійкість до повільного руху	Може бути базовим варіантом, але не обрано
Фонове віднімання MOG2	Адаптивна модель фону, прийнятна швидкодія, доступність в OpenCV	Потребує фільтрації тіней і дрібних шумів	Обрано як основний підхід
Оптичний потік	Дає інформацію про напрям і швидкість переміщення	Вища обчислювальна складність	Не обрано через надмірність для подієвої системи
Нейромережева сегментація	Може точніше відокремлювати об’єкти	Потребує моделі, ресурсів і підготовки даних	Не обрано для первинного фільтра
Постійне YOLO-розпізнавання	Семантичний аналіз кожного кадру	Зайве навантаження на систему	Використовується лише після руху

Маска переднього плану формується шляхом порівняння поточного обробленого кадру з фоновою моделлю.

$$M_t(x,y) = \begin{cases} 1, & |G_t(x,y) - B_t(x,y)| \geq \tau, \\ 0, & |G_t(x,y) - B_t(x,y)| < \tau. \end{cases}, \quad (2.3)$$

де $M_t(x, y)$ – бінарна маска руху;

$G_t(x, y)$ – інтенсивність поточного кадру;

$B_t(x, y)$ – значення фонової моделі;

τ– поріг відмінності від фону.

Після одержання маски виконуються морфологічні операції. Відкриття зменшує кількість поодиноких шумових пікселів, а розширення об’єднує близькі ділянки переднього плану. Далі визначаються зовнішні контури рухомих областей. Серед знайдених контурів важливими вважаються ті, площа яких перевищує встановлений поріг значущості [18].

$$S_t = \max_{1 \le k \le n} A(C_k), \tag{2.4}$$

де S_t – найбільша площа рухомої області;

C_k – k-й контур;

$A(C_k)$ – площа контуру;

n – кількість контурів у масці.

Однокадрове перевищення порогу не завжди є достатньою підставою для формування події. Через це використовується підтвердження руху на основі послідовності кадрів. Якщо площа значущого контуру перевищує поріг, ознака руху накопичується. Якщо значущий контур відсутній, накопичення скидається.

$$D_t = \begin{cases} 1, & S_t \ge S_{\min} \wedge q_t \ge N, \\ 0, & \text{якщо умови не виконано.'} \end{cases} \tag{2.5}$$

де D_t – ознака підтвердженого руху;

S_{min} – мінімальна площа контуру;

q_t – довжина поточної серії кадрів із рухом;

N – кількість кадрів, потрібна для підтвердження.

Таким чином, обраний метод детекції руху є компромісом між швидкодією та стійкістю. MOG2 формує адаптивну модель фону,

морфологічна фільтрація зменшує шум, площинний критерій відкидає малі об’єкти, а підтвердження на послідовності кадрів знижує ймовірність випадкового спрацювання [18, 20].

2.5 Порівняння підходів до розпізнавання людини

Після встановлення факту руху необхідно визначити, чи є подія змістовно важливою. Сам рух не доводить появу людини в кадрі, оскільки зміна може бути спричинена тінню, коливанням предмета, шумом камери або іншими факторами. Тому у програмній системі передбачено окремий етап розпізнавання людини.

Для розв’язання цієї локальної задачі було розглянуто декілька підходів до розпізнавання людини: відсутність семантичного розпізнавання, класичні детектори OpenCV, нейромережева модель YOLO у форматі ONNX, хмарні API комп’ютерного зору та навчання власної моделі. Порівняння цих підходів подано у таблиці 2.7 [23, 24].

Таблиця 2.7 – Порівняння підходів до розпізнавання людини

Підхід	Переваги	Недоліки	Рішення для проєкту
1	2	3	4
Тільки факт руху	Найменше навантаження на систему	Не відрізняє людину від інших змін у кадрі	Не використовується як остаточне рішення
HOG + Haar-каскади OpenCV	Не потребує зовнішньої моделі, працює локально	Менша стійкість до складних поз, освітлення та перекриттів	Обрано як резервний режим

Продовження таблиці 2.7

1	2	3	4
YOLO ONNX	Краще розпізнає людину в різних умовах, працює локально через OpenCV DNN	Потребує ONNX-файл і більше обчислень	Обрано як основний режим за наявності моделі
Хмарний API	Може мати високу якість розпізнавання	Залежність від інтернету, передавання кадрів сторонньому сервісу	Не обрано через вимогу локальності
Власне навчання моделі	Можливість адаптації до конкретних умов	Потрібні дані, розмітка, навчання і перевірка якості	Не обрано в межах поточної роботи

Для задачі розпізнавання людини передбачено комбіновану модель. Резервний режим може використовувати класичні детектори OpenCV, що забезпечує працездатність без підключення нейромережевої моделі. Основний режим передбачає використання ONNX-моделі YOLO через OpenCV DNN, що дає змогу виконувати локальне розпізнавання людини в кадрі.

У режимі YOLO кадр приводиться до вхідного подання, прийнятного для моделі, після чого виконується прямий прохід мережі та аналіз вихідних прогнозів. Для кожного прогнозу визначається клас з найбільшою оцінкою. Оскільки задача орієнтована на людину, враховується клас person. Правило вибору найбільш імовірного класу вираховується за формулою [25, 26].

$$c^* = \arg\max_i p_i, \qquad P = \max_i p_i \,, \tag{2.6}$$

де c^* – клас з найбільшою оцінкою;

p_i – оцінка належності до і-го класу;

P – максимальна довіра до прогнозу.

Координати рамки, отримані від нейромережевої моделі, задаються відносно вхідного розміру моделі. Для нанесення рамки на початковий кадр виконується масштабування за шириною та висотою.

$$x = \left(x_c - \frac{w}{2}\right) k_x, \qquad y = \left(y_c - \frac{h}{2}\right) k_y, \tag{2.7}$$

де *x, y* – координати лівого верхнього кута рамки на вихідному кадрі;

xc, yc – координати центра рамки;

w, h – ширина та висота рамки;

k_x, k_y – коефіцієнти масштабування.

Перевагою реалізованого підходу є наявність резервування. Якщо YOLO-модель не вказана або не може бути завантажена, система не припиняє роботу, а може використовувати засоби OpenCV. Якщо модель доступна, розпізнавання виконується на основі нейромережевого підходу. Така модель вибору відповідає навчальному й прикладному характеру роботи, оскільки забезпечує як демонстраційну працездатність, так і можливість підвищення точності.

2.6 Порівняння підходів до формування події та передавання повідомлень

Після виявлення руху та розпізнавання людини система має визначити, який результат слід зберегти і як повідомити користувача. Можливими режимами є безперервний запис, збереження фото за фактом руху, запис відеофрагмента за фактом руху або комбінована модель із вибором режиму. Безперервний запис забезпечує повноту даних, але створює надлишкове навантаження на сховище. Фоторежим є економним, але може не зафіксувати розвиток події. Відеорежим зберігає більше контексту, але потребує обмеження тривалості. Порівняння підходів подано у таблиці 2.8.

Таблиця 2.8 – Порівняння режимів формування події

Режим	Переваги	Недоліки	Використання в системі
Безперервний запис	Фіксує всі події без пропусків	Швидко збільшує обсяг сховища та ускладнює перегляд	Не обрано як основний режим
Фото за фактом руху	Економить сховище, швидко надсилається в Telegram	Може не передати весь розвиток події	Обрано як режим photo
Відео за фактом руху	Зберігає контекст події	Потребує контролю FPS і умови зупинки	Обрано як режим video
Комбінована модель	Дає користувачу вибір залежно від задачі моніторингу	Потребує додаткових параметрів	Обрано як загальну логіку

Для формування подій обрано комбіновану модель із режимами фото та відео. У фоторежимі після підтвердження руху виконується розпізнавання, зберігається знімок і надсилається повідомлення. Для зменшення кількості повторних повідомлень застосовується часове обмеження повторної відправки. У відеорежимі запис починається після появи руху та завершується після відсутності значущих змін у кадрі протягом заданого інтервалу.

Узагальнене правило формування події враховує не тільки рух, а й активність моніторингу, розклад, режим роботи та обмеження повторних повідомлень.

$$E_t = \begin{cases} 1, & D_t = 1 \wedge A_t = 1 \wedge R_t = 1, \\ 0, & \text{якщо хоча б одна умова не виконана.'} \end{cases} \quad (2.8)$$

де E_t – ознака сформованої події;

D_t – ознака підтвердженого руху;

A_t – ознака активного та дозволеного моніторингу;

R_t – виконання режимних обмежень.

На рисунку 2.3 показано, що в процесі формування події важливим є послідовне відокремлення випадкових змін у кадрі від дійсно значущих ситуацій. Для цього система не переходить до збереження матеріалів одразу після появи окремої зміни у відеопотоці. Спочатку виконується перевірка наявності руху, після чого аналізуються контури та площа активної області. Такий підхід дає змогу зменшити кількість хибних спрацювань, які можуть виникати через зміну освітлення, появу тіней, цифровий шум камери або незначні коливання об'єктів у зоні спостереження.

Після підтвердження руху система переходить до уточнювального етапу, пов'язаного з розпізнаванням людини. Цей етап є необхідним, оскільки факт руху не завжди означає появу користувацьки значущої події. У разі виявлення людини формується подія, яка містить інформацію про час спрацювання, тип збереженого матеріалу та результат аналізу кадру. Якщо людину не виявлено, система повертається до циклу оброблення відеопотоку без створення повідомлення.

Окрему роль у запропонованому алгоритмі відіграє вибір режиму збереження матеріалу події. У фоторежимі формується окремий знімок, який фіксує момент спрацювання та швидко передається користувачу. У відеорежимі зберігається короткий фрагмент, що дає змогу переглянути розвиток ситуації в часі. Вибір режиму залежить від налаштувань системи та характеру задачі відеомоніторингу. Для зменшення надлишкового навантаження на сховище тривалість відеофрагмента та частота повторних сповіщень мають обмежуватися встановленими параметрами.

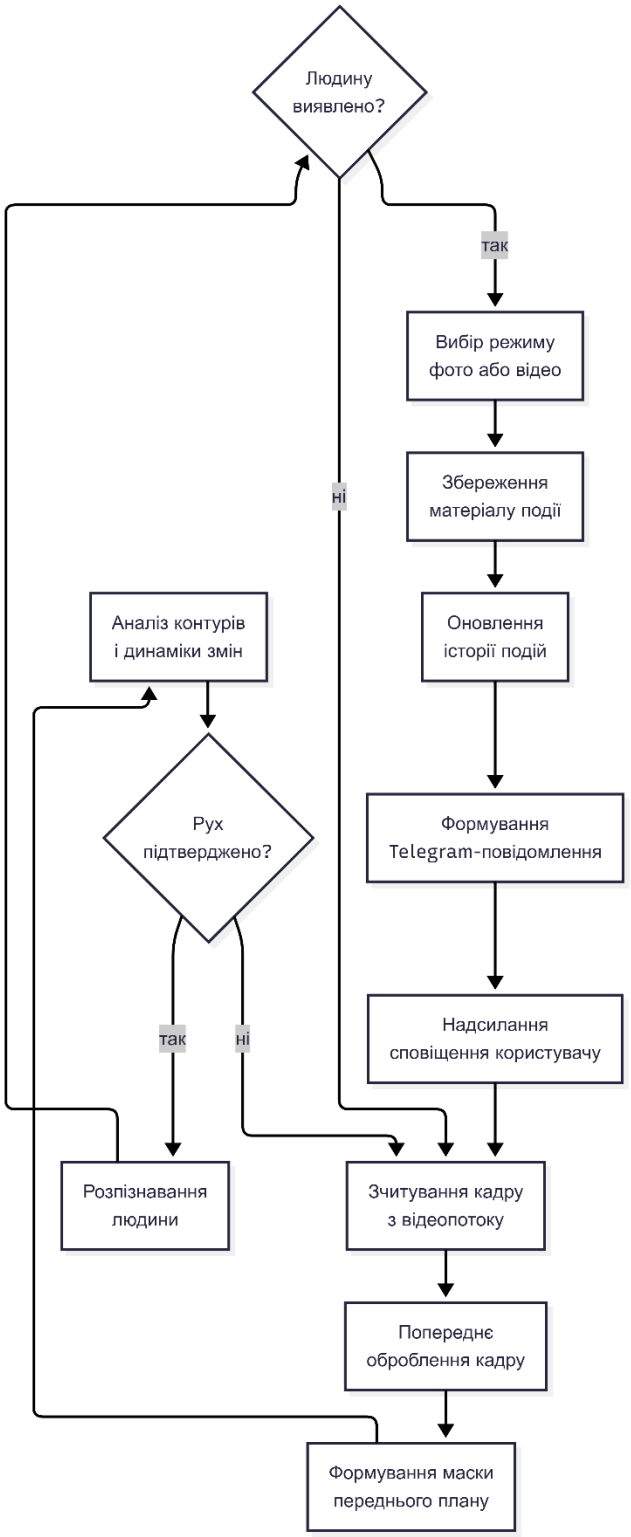

Рисунок 2.3 – Алгоритм формування події у програмній системі

Для передавання повідомлень також розглянуто кілька підходів: локальний журнал, електронна пошта, власний мобільний застосунок, push-сервіс і Telegram-бот. Telegram-бот обрано як оптимальний канал, оскільки він дає змогу надсилати текстові повідомлення, фото і відео, підтримує віддалені

команди та не потребує розроблення окремого мобільного клієнта. Порівняння каналів сповіщення подано у таблиці 2.9 [15, 17].

Таблиця 2.9 – Порівняння каналів сповіщення користувача

Канал	Переваги	Недоліки	Рішення
Локальний журнал	Простота реалізації та відсутність мережевих залежностей	Користувач не отримує оперативне повідомлення віддалено	Використовується як історія подій
Електронна пошта	Підтримка вкладень і широке поширення	Повільніше сприйняття, складніше керування командами	Не обрано, із-за затримок в сповіщені користувача
Власний мобільний застосунок	Повний контроль над інтерфейсом	Потребує окремої розробки та підтримки	Не обрано, із-за скзадності та незручності
Push-сервіс	Швидкі сповіщення	Потреба у додатковій інфраструктурі	Не обрано
Telegram-бот	Команди, текст, фото, відео, контроль доступу, швидка взаємодія	Потребує токен бота та інтернет-з’єднання	Обрано як основний канал

У межах обраного підходу Telegram-бот відповідає за реєстрацію користувача, перегляд стану системи, зміну доступних параметрів, отримання поточного знімка, формування звіту та перевірку працездатності системи.

Контроль доступу має здійснюватися через список дозволених користувачів, що зменшує ризик несанкціонованого керування системою [17].

2.7 Порівняння підходів до збереження налаштувань, подій і керування сховищем

Для подієвої системи відеоспостереження важливим є не лише розпізнавання, а й збереження параметрів, історії та матеріалів подій. Можливими підходами до збереження стану є текстові конфігураційні файли, JSON-файли, реляційна база даних або вбудована база SQLite. Кожний варіант має різний рівень складності та масштабованості, їх порівняння подано у таблиці 2.10 [27].

Таблиця 2.10 – Порівняння підходів до збереження стану системи

Підхід	Переваги	Недоліки	Доцільність
Службові конфігураційні файли	Зручні для зберігання службових параметрів, шляхів і значень за замовчуванням	Не призначені для частого оновлення складних структур	Використано для конфігурації середовища
JSON-файли	Прості, читабельні, підтримують вкладені структури	Не мають складних запитів і транзакцій	Використано для налаштувань, користувачів та історії
SQLite	Підтримує структуровані запити та більші обсяги історії	Потребує схеми таблиць і додаткового коду доступу	Може бути подальшим удосконаленням
Серверна база даних	Підходить для багатьох камер і користувачів	Надмірна для локального застосунку	Не обрано

У системі використовується змішаний підхід: службові параметри відокремлюються від робочих налаштувань, користувацькі дані та історія подій зберігаються у структурованому файловому форматі. Такий вибір є достатнім для локальної системи, оскільки не потребує встановлення серверної бази даних і забезпечує зрозумілу структуру збереження.

Керування файлами подій виконується окремою процедурою очищення сховища. Вона видаляє застарілі файли за строком зберігання та додатково обмежує загальний розмір сховища. Це важливо, оскільки у відеорежимі обсяг даних може зростати швидше, ніж у фоторежимі. Правила очищення мають задаватися через параметри строку зберігання та граничного обсягу сховища [27].

Таблиця 2.11 – Обґрунтування вибраних параметрів і механізмів системи

Елемент	Порівняні альтернативи	Обране рішення	Причина вибору
1	2	3	4
Детекція руху	Різниця кадрів, MOG2, оптичний потік, нейромережева сегментація	MOG2 + контури + підтвердження кадрами	Баланс швидкодії та стійкості для статичної камери
Розпізнавання людини	Без розпізнавання, HOG + Haar, YOLO, хмарний API	YOLO ONNX з резервом OpenCV	Локальна робота та можливість резервного режиму

Продовження таблиці 2.11

1	2	3	4
Сповіщення	Журнал, email, push, власний застосунок, Telegram	Telegram-бот	Швидке та віддалене керування медіа
Архітектура	Монолітна, синхронна модульна, асинхронна модульна	asyncio + черга сповіщень	Відокремлення оброблення кадрів від мережевої взаємодії
Стан системи	Службові конфігураційні файли, структуровані файли, SQLite, серверна база	Службова конфігурація та структуровані файли	Достатня простота для локального застосунку

Такий підхід до збереження стану відповідає масштабу розробленої системи. Структуровані файли забезпечують просту зміну параметрів через інтерфейс, службова конфігурація відокремлює технічні значення, а періодичне очищення сховища запобігає неконтрольованому накопиченню файлів .

3 РОЗРОБЛЕННЯ, ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМНОЇ СИСТЕМИ ВІДЕОСПОСТЕРЕЖЕННЯ

У третьому розділі подано практичну реалізацію програмної системи відеоспостереження з автоматичним виявленням руху, розпізнаванням людини у відеопотоці та передаванням мультимедійних повідомлень через Telegram-бота. Цей розділ орієнтовано на опис фактично розроблених програмних компонентів, їх взаємодії, налаштування, тестування та оцінювання результатів.

Практична реалізація кваліфікаційної роботи, виконана як локальний застосунок Camera, що працює на персональному комп’ютері, отримує кадри з камери, виконує первинну детекцію руху, за потреби застосовує розпізнавання людини, зберігає матеріали події та надсилає повідомлення дозволеним користувачам. Такий підхід відповідає поставленій задачі, оскільки система не обмежується пасивним відеозаписом, а переходить до подієвої моделі роботи. Подія формується лише після виконання визначених умов: активний моніторинг, допустимий розклад, наявність руху, виконання режимних обмежень і можливість передавання результату користувачу.

3.1 Обґрунтування вибору інструментальних засобів реалізації

Для реалізації застосунку обрано мову програмування Python. Її використання зумовлене наявністю розвиненої екосистеми бібліотек для комп’ютерного зору, асинхронної мережевої взаємодії, роботи з файлами конфігурації та швидкого створення прикладних програм. У межах поставленої задачі Python забезпечує достатню швидкість розроблення, зручну інтеграцію з OpenCV, підтримку бібліотеки aiogram і можливість розширення функціональності без істотної перебудови архітектури.

Основним засобом отримання та оброблення відеопотоку обрано бібліотеку OpenCV. Вона використовується для роботи з камерою через VideoCapture, перетворення кадру у відтінки сірого, згладжування зображення, фонового віднімання, морфологічного оброблення, пошуку контурів, збереження знімків і запису відеофрагментів. Використання OpenCV є доцільним, оскільки бібліотека підтримує як класичні методи комп'ютерного зору, так і модуль DNN для локального запуску нейромережевих моделей [18].

Для розпізнавання людини реалізовано два режими. Основний режим передбачає використання YOLO-моделі у форматі ONNX, що виконується локально через OpenCV DNN. Такий вибір поєднує переваги нейромережевого розпізнавання з відсутністю потреби передавати кадри стороннім сервісам. Резервний режим базується на класичних детекторах OpenCV, зокрема HOG-детекторі силуету людини та Haar-каскадах для пошуку обличчя, профілю обличчя й верхньої частини тіла. Наявність резервного режиму підвищує стійкість системи у випадку відсутності або неправильного зазначення файлу моделі.

Для організації Telegram-бота застосовано бібліотеку aiogram. Вона підтримує асинхронну модель роботи, оброблення команд, callback-кнопок, надсилання текстових повідомлень, фото та відео. Це відповідає архітектурі системи, у якій основний цикл моніторингу не повинен блокуватися під час мережевих операцій. Для збереження налаштувань і журналу подій використано формат JSON, що є простим для читання, редагування та серіалізації даних [27].

Перелік використаних інструментальних засобів реалізації наведено у таблиці 3.1.

Обраний набір інструментальних засобів дає змогу реалізувати локальну систему відеоспостереження без складної серверної інфраструктури, забезпечити подієве збереження матеріалів і організувати зручний канал віддаленого сповіщення. Перелік використаних інструментальних засобів реалізації наведено у таблиці 3.1.

Таблиця 3.1 – Використані інструментальні засоби реалізації

Засіб	Призначення	Обґрунтування застосування
Python	Реалізація логіки застосунку, асинхронних задач, роботи з файлами	Забезпечує швидке створення модульного прикладного рішення
OpenCV	Захоплення кадрів, оброблення зображень, MOG2, контури, DNN	Надає єдине середовище для класичного й нейромережевого аналізу відеопотоку
YOLO ONNX	Розпізнавання людини в кадрі	Дає змогу виконувати локальну перевірку класу person без хмарного API
aiogram	Реалізація Telegram-бота та інтерактивних команд	Підтримує асинхронну взаємодію і передавання медіафайлів
JSON	Збереження налаштувань, користувачів та історії подій	Є достатнім для локальної системи та не потребує серверної БД
python-dotenv	Завантаження службових параметрів із .env	Відокремлює токени та технічні параметри від програмного коду
Pillow	Допоміжна робота з графічними ресурсами застосунку	Використовується для підготовки зображень та іконок

3.2 Структура розробленого програмного застосунку

Розроблений застосунок має модульну структуру. Основні програмні складові винесено в окремі каталоги та файли, що зменшує зв’язність

компонентів і полегшує подальший супровід. Координаційний модуль відповідає за запуск асинхронних задач, модуль детекції руху за аналіз відеопотоку, модуль розпізнавання за визначення наявності людини, модуль Telegram-бота за віддалене керування та сповіщення, а модуль локального інтерфейсу відповідає за роботу із системним треєм.

Узагальнену структуру каталогів програмного застосунку винесено до додатка А. Така структура відповідає архітектурному рішенню, обґрунтованому в другому розділі: система складається з відносно незалежних модулів, кожний з яких виконує власну локальну задачу.

Файл main.py виконує роль точки входу до застосунку. У ньому ініціалізуються детектор руху, ідентифікатор об’єктів, черга сповіщень, цикл очищення сховища, Telegram-бот і локальний інтерфейс. Файл config.py відповідає за завантаження змінних середовища, шляхів до службових файлів і параметрів за замовчуванням. Каталог motion_detection містить клас MotionDetector, у якому зосереджено логіку отримання кадрів, виявлення руху та збереження матеріалів. Каталог image_processing містить клас ObjectIdentifier і допоміжний модуль detection_mode.py, що забезпечує перемикання між режимами OpenCV та YOLO.

Каталог bot_handler містить два ключові файли. У файлі bot.py реалізовано Telegram-команди, інтерактивні кнопки, надсилання медіаматеріалів і розсилання сповіщень дозволеним користувачам. У файлі state.py реалізовано збереження та оновлення стану застосунку. Каталог desktop_ui містить модулі системного трея й автозапуску. Окремі файли JSON використовуються для робочих налаштувань, історії подій, списку відомих користувачів і користувачів, які очікують підтвердження.

Призначення основних файлів і модулів застосунку наведено у таблиці 3.2.

Особливістю структури є відокремлення службової конфігурації від змінюваного стану. Статичні або рідко змінювані параметри розміщуються у файлі .env, а робочі налаштування, що можуть змінюватися з Telegram або

локального інтерфейсу, зберігаються у settings.json. Завдяки цьому користувач може змінювати режим роботи, камеру, область інтересу, пороги детекції, розклад і спосіб розпізнавання без внесення змін до програмного коду.

Таблиця 3.2 – Призначення основних файлів і модулів застосунку

Файл або каталог	Призначення	Результат роботи
main.py	Запуск асинхронних задач і координація роботи системи	Одночасна робота моніторингу, бота, очищення сховища та трея
config.py	Завантаження .env, шляхів і параметрів за замовчуванням	Єдине джерело службової конфігурації
motion_detection/detector.py	Виявлення руху, формування фото та відеозапису	Ознака руху та файл події
image_processing/identifier.py	Розпізнавання людини в режимах YOLO або OpenCV	Список виявлених об’єктів і кадр з розміткою
bot_handler/bot.py	Telegram-команди, кнопки, надсилання сповіщень	Віддалене керування та інформування користувача
bot_handler/state.py	Збереження налаштувань, користувачів, розкладу та історії	Актуальний стан системи у JSON-файлах
desktop_ui/tray.py	Локальне керування через системний трей	Швидкий доступ до параметрів без редагування коду
tests	Модульні тести конфігурації, режимів і автозапуску	Перевірка окремих службових функцій

3.3 Реалізація основного циклу роботи системи

Основний цикл роботи системи реалізовано в main.py. Його завдання полягає не тільки в послідовному обробленні кадрів, а й в узгодженні декількох паралельних процесів: Telegram-бота, черги сповіщень, очищення сховища, системного трея та безпосереднього відеомоніторингу. Для цього використано засоби asyncio, що дають змогу створювати окремі асинхронні задачі й не блокувати один компонент роботою іншого.

Загальну схему асинхронної взаємодії компонентів винесено до додатка А. Основна ідея полягає в тому, що цикл аналізу кадрів не виконує мережеву операцію надсилання повідомлення безпосередньо. Замість цього подія передається до черги, після чого окремий працівник alert-worker здійснює розсилання через Telegram. Такий підхід зменшує ризик пропуску кадрів у разі тимчасової затримки мережі.

Під час запуску застосунку створюється черга сповіщень, запускається Telegram-бот у режимі polling, активується цикл очищення сховища, ініціалізується системний трей і запускається основний цикл відеомоніторингу. Якщо один із процесів завершується з критичною помилкою, виконання застосунку фіксується в журналі, а всі активні задачі коректно скасовуються. Для журналювання використано RotatingFileHandler, що обмежує розмір одного файлу журналу та кількість резервних копій.

Лістинг 3.1 Фрагмент реалізації асинхронної черги сповіщень:

```
async def enqueue_alert(alert_queue: asyncio.Queue, message_text: str,
                        file_path: str = None, file_type: str = "photo"):
    await alert_queue.put({
        "message_text": message_text,
        "file_path": file_path,
        "file_type": file_type,
    })
```

```
async def alert_worker(alert_queue: asyncio.Queue):
    while True:
        alert = await alert_queue.get()
        try:
            if alert is ALERT_WORKER_STOP:
                return
            await broadcast_alert(
                alert["message_text"],
                alert.get("file_path"),
                alert.get("file_type", "photo"),
            )
        except Exception as exc:
            logger.error("Помилка надсилання сповіщення з черги: %s", exc,
exc_info=True)
        finally:
            alert_queue.task_done()
```

У лістингу 3.1 наведено фрагмент, який демонструє відокремлення моменту формування події від моменту її передавання користувачу. Функція enqueue_alert створює повідомлення у вигляді словника з текстом, шляхом до файлу та типом медіа. Функція alert_worker очікує нові елементи черги, передає їх до broadcast_alert і позначає задачу як оброблену. У разі помилки її зміст фіксується в журналі, але основний цикл моніторингу не припиняється.

У головному циклі постійно оновлюються налаштування, отримані зі стану застосунку. Це потрібно для того, щоб зміни, внесені через Telegram або системний трей, починали діяти без перезапуску програми. До таких параметрів належать індекс камери, режим фото або відео, поріг площі контуру, кількість кадрів для підтвердження руху, режим розпізнавання, шлях до YOLO-моделі, розклад і область інтересу.

Окрему увагу приділено відновленню після помилок камери. Якщо застосунок не може отримати кадр, лічильник помилок збільшується. Після досягнення граничної кількості невдалих спроб камера звільняється, у журнал подій додається запис про помилку, а система намагається повторно підключитися до того самого пристрою. Користувачу надсилається повідомлення про недоступність камери та про її подальше відновлення. Такий механізм підвищує надійність застосунку під час тривалої роботи.

3.4 Реалізація модуля виявлення руху

Модуль виявлення руху реалізовано у класі MotionDetector. Він відповідає за отримання кадрів з камери, попереднє оброблення зображення, побудову маски переднього плану, фільтрацію шуму, пошук контурів, підтвердження руху, збереження знімків і запис відео. Для захоплення кадрів використано cv2.VideoCapture, а розмір кадру встановлюється відповідно до параметрів FRAME_WIDTH і FRAME_HEIGHT.

Алгоритм виявлення руху винесено до додатка А. Після отримання кадру система може застосувати область інтересу. Далі кадр перетворюється у відтінки сірого та згладжується фільтром GaussianBlur. Після цього використовується фонове віднімання MOG2, яке формує маску переднього плану. Маска додатково обробляється пороговою операцією, морфологічним відкриттям і розширенням, після чого визначаються зовнішні контури.

Використання MOG2 у розробленому застосунку обґрунтовано необхідністю пристосування до сцени зі статичною камерою. Порівняно з простим порівнянням сусідніх кадрів, модель фону краще враховує поступові зміни сцени. Водночас вона може реагувати на тіні або дрібний шум, тому після побудови маски застосовано додаткові фільтри. Порогове значення 244 використовується для відокремлення найбільш виражених ділянок переднього

плану, а морфологічні операції зменшують кількість дрібних областей, які не мають прикладного значення.

Лістинг 3.2 Фрагмент реалізації виявлення руху:

```
def get_frame_with_motion_status(self):
    if not self.is_running or self.previous_frame is None:
        return None, False

    original_frame = self._read_frame()
    if original_frame is None:
        return None, False

    detection_frame = self._crop_roi(original_frame)
    current_processed_frame = self._process_frame(detection_frame)
    foreground_mask =
self.background_subtractor.apply(current_processed_frame)
    _, thresh = cv2.threshold(foreground_mask, 244, 255,
cv2.THRESH_BINARY)
    thresh = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, None,
iterations=1)
    thresh = cv2.dilate(thresh, None, iterations=2)
    contours, _ = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
                                   cv2.CHAIN_APPROX_SIMPLE)

    raw_motion_detected = any(cv2.contourArea(contour) > self.min_area
                              for contour in contours)
    if raw_motion_detected:
        self.motion_streak += 1
    else:
        self.motion_streak = 0
```

```
            motion_detected = self.motion_streak >=
self.motion_confirmation_frames
            return original_frame, motion_detected
```

У лістингу 3.2 показано ключову частину алгоритму детекції руху. Після отримання кадру виконується обрізання за ROI, попереднє оброблення, застосування фонового віднімача та пошук контурів. Не кожний знайдений контур вважається значущим: рух фіксується лише тоді, коли площа хоча б одного контуру перевищує встановлений поріг min_area. Додатково використано motion_streak, який накопичує кількість послідовних кадрів із рухом. Це дає змогу зменшити вплив випадкових одиничних спрацювань.

Параметри min_area і motion_confirmation_frames не зафіксовані жорстко в коді. Вони можуть змінюватися через файл налаштувань або інтерфейс керування. Такий підхід важливий, оскільки різні сцени потребують різних порогів. Для невеликого приміщення поріг площі може бути нижчим, а для сцени з великою кількістю дрібних рухів – вищим. Аналогічно, кількість кадрів для підтвердження руху впливає на баланс між швидкістю реакції та стійкістю до шуму.

Область інтересу реалізовано як кортеж координат x, y, width і height. Перед обрізанням значення обмежуються межами кадру, що запобігає помилкам у випадку некоректного налаштування. Якщо ROI вимкнено, аналізується весь кадр. Якщо ROI увімкнено, система ігнорує неважливі частини сцени, наприклад зону вікна, екрана або рухомих відблисків. Це зменшує кількість хибних спрацювань без ускладнення основного алгоритму.

Результатом роботи модуля є пара значень: оригінальний кадр і логічна ознака підтвердженого руху. Така форма результату є зручною для головного циклу, оскільки кадр може бути використаний для подальшого розпізнавання, збереження або оновлення latest_frame.jpg, а ознака руху визначає, чи потрібно переходити до формування події.

3.5 Реалізація модуля розпізнавання людини

Модуль розпізнавання людини реалізовано у класі ObjectIdentifier. Його призначення полягає в тому, щоб після підтвердження руху встановити, чи пов’язана подія з появою людини. У системі передбачено два режими роботи: YOLO ONNX і резервний режим OpenCV. Перемикання виконується через параметр detection_backend, а нормалізація значення здійснюється в модулі detection_mode.py. Якщо встановлено режим YOLO, але файл моделі відсутній або не завантажується, система повертає ознаку помилки розпізнавання і не завершує роботу всього застосунку.

Загальну логіку розпізнавання винесено до додатка А. Основний режим використовує модель YOLO у форматі ONNX. Кадр перетворюється на blob розміром 640 на 640 пікселів, нормалізується до діапазону від 0 до 1 і подається на вхід мережі. Після прямого проходу аналізуються вихідні прогнози. Для задачі відеоспостереження використовується тільки клас person, який у наборі класів COCO має індекс 0. Прогнози з нижчою довірою відкидаються.

Лістинг 3.3 Фрагмент реалізації YOLO-розпізнавання людини:

```
def _detect_people_yolo(self, frame):
    input_size = 640
    blob = cv2.dnn.blobFromImage(frame, 1 / 255.0,
                                 (input_size, input_size), swapRB=True, crop=False)
    self.yolo_net.setInput(blob)
    outputs = self.yolo_net.forward()
    predictions = np.squeeze(outputs)

    if predictions.ndim == 1:
        predictions = np.expand_dims(predictions, axis=0)
```

```
        if predictions.shape[0] < predictions.shape[1] and predictions.shape[0]
in (84, 85):
            predictions = predictions.T

        frame_height, frame_width = frame.shape[:2]
        x_scale = frame_width / input_size
        y_scale = frame_height / input_size
        detections = []

        for row in predictions:
            class_scores = row[4:]
            class_id = int(np.argmax(class_scores))
            confidence = float(class_scores[class_id])
            if class_id != 0 or confidence < self.yolo_confidence_threshold:
                continue
            center_x, center_y, width, height = row[:4]
            x = int((center_x - width / 2) * x_scale)
            y = int((center_y - height / 2) * y_scale)
            w = int(width * x_scale)
            h = int(height * y_scale)
            detections.append((x, y, w, h, f"person-yolo {confidence:.2f}"))
        return detections
```

Лістинг 3.3 відображає процес аналізу вихідних даних YOLO-моделі. Оскільки різні ONNX-експорти можуть мати різну орієнтацію масиву прогнозів, у коді передбачено перевірку розмірності та транспонування матриці за потреби. Після цього для кожного прогнозу визначається клас із найбільшою оцінкою. Якщо клас не відповідає людині або довіра нижча за встановлений поріг, прогноз не використовується. Для прийнятого прогнозу координати рамки масштабуються відповідно до розміру початкового кадру.

Резервний режим OpenCV використовується тоді, коли нейромережева модель не потрібна або не підключена. У цьому режимі застосовуються HOG-детектор і Haar-каскади для пошуку ознак людини. HOG орієнтований на виявлення силуету, а Haar-каскади на пошук обличчя, профілю обличчя та верхньої частини тіла. Такий режим менш стійкий до складних умов, проте забезпечує працездатність без файлу моделі. Фрагмент реалізації резервного режиму OpenCV наведено в лістингу 3.4.

Лістинг 3.4 Фрагмент реалізації резервного OpenCV-розпізнавання:

```
elif self.detection_backend == DETECTION_BACKEND_OPENCV and self.pose_detector is not None:
    resized_frame = self._resize_for_detection(frame)
    detections, weights = self.pose_detector.detectMultiScale(
        resized_frame,
        winStride=(8, 8),
        padding=(8, 8),
        scale=1.05,
    )
    if len(detections) > 0:
        max_weight = max((float(weight) for weight in weights), default=0.0)
        if max_weight >= self.person_confidence_threshold:
            detections_found.extend(
                self._scale_hog_detections(
                    detections,
                    original_shape=frame.shape,
                    resized_shape=resized_frame.shape,
                    label="person-hog",
                )
            )
    detections_found.extend(self._contains_person_features(frame))
```

```
def _contains_person_features(self, frame):
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray = cv2.equalizeHist(gray)
    detections_found = []
    for detector, scale_factor, min_neighbors, min_size, label in (
        (self.face_detector, 1.1, 5, (40, 40), "face"),
        (self.profile_face_detector, 1.1, 5, (40, 40), "profile-face"),
        (self.upper_body_detector, 1.05, 4, (60, 60), "upper-body"),
    ):
        if detector is None:
            continue
        detections = detector.detectMultiScale(
            gray,
            scaleFactor=scale_factor,
            minNeighbors=min_neighbors,
            minSize=min_size,
        )
        for x, y, w, h in detections:
            detections_found.append((int(x), int(y), int(w), int(h), label))
    return detections_found
```

У лістингу 3.4 наведено фрагмент резервного розпізнавання, який застосовується за вибору OpenCV-режиму. Спочатку кадр зменшується до робочого розміру, після чого HOG-детектор виконує пошук силуету людини. Додатково кадр переводиться у відтінки сірого, вирівнюється за гістограмою та перевіряється каскадними класифікаторами обличчя, профілю обличчя і верхньої частини тіла. Така комбінація підвищує імовірність виявлення людини без використання нейромережевої моделі.

Після виявлення кандидатів виконується об’єднання вкладених рамок. Це потрібно, оскільки різні детектори можуть знайти одну й ту саму людину кілька разів, наприклад через силует, обличчя та верхню частину тіла. Метод _merge_nested_detections залишає більші рамки та відкидає кандидати, центр яких міститься всередині вже прийнятої рамки. Завдяки цьому повідомлення користувачу не містить зайвого збільшення кількості виявлених об’єктів.

Для діагностики роботи розпізнавання передбачено debug-кадр. Якщо виявлено людину, на зображення наноситься рамка та підпис із типом детектора або довірою YOLO. Якщо людину не виявлено, на кадр додається відповідний текст. У фоторежимі користувач може отримувати саме такий debug-кадр, що спрощує перевірку правильності розпізнавання.

3.6 Реалізація режимів фото- та відеофіксації подій

У системі реалізовано два режими фіксації подій: photo і video. Режим photo призначений для швидкого надсилання окремого знімка після виявлення руху. Режим video призначений для збереження короткого відеофрагмента, який фіксує розвиток події в часі. Перемикання режимів здійснюється через Telegram-бота або локальний інтерфейс, а поточне значення зберігається у settings.json.

У фоторежимі після підтвердження руху система перевіряє часовий інтервал photo_cooldown_period. Якщо з моменту останнього фото минуло недостатньо часу, нове повідомлення не формується. Це запобігає надмірній кількості однакових сповіщень у ситуації, коли людина тривалий час перебуває в кадрі. Якщо обмеження виконано, кадр передається до ObjectIdentifier, після чого зберігається знімок. У повідомлення додається короткий опис виявлених об’єктів.

У відеорежимі логіка відрізняється. Після підтвердження руху система починає запис відео, якщо запис ще не виконується. Під час запису кадри

додаються з урахуванням установленого FPS. Якщо рух не фіксується протягом video_no_motion_stop_delay секунд, запис завершується, файл передається до черги сповіщень, а в історію подій додається запис про завершений відеофрагмент. Такий підхід дає змогу зберігати не весь відеопотік, а тільки фрагменти, пов’язані з подіями. Основні параметри налаштування режимів подієвої фіксації наведено у таблиці 3.3.

Таблиця 3.3 – Основні параметри налаштування режимів подієвої фіксації

Параметр	Значення у реалізації	Вплив на роботу системи
current_mode	photo або video	Визначає спосіб фіксації події
photo_cooldown_period	Типово 10 с	Обмежує частоту повторних фото
video_fps	Налаштовується, у settings.json може бути 15	Впливає на плавність і розмір відеофайлу
video_no_motion_stop_delay	Типово 5 с	Визначає час очікування перед завершенням запису
debug_photo_enabled	true або false	Визначає, чи надсилати кадр із рамками розпізнавання
min_contour_area	Типово 1000	Відсіює дрібні рухомі області
motion_confirmation_frames	Типово 2	Підтверджує рух на кількох кадрах
roi_enabled	true або false	Дозволяє аналізувати лише задану частину кадру

Подієва модель роботи має перевагу перед безперервним записом. Вона зменшує навантаження на сховище, спрощує перегляд матеріалів і підвищує інформативність повідомлень. Водночас система не втрачає можливість

зберегти контекст події, оскільки користувач може перейти з фоторежиму до відеорежиму залежно від умов спостереження.

3.7 Реалізація Telegram-бота та механізму сповіщень

Telegram-бот є основним каналом взаємодії користувача з розробленою системою. Він виконує дві групи функцій. Перша група пов’язана з керуванням: увімкнення та вимкнення моніторингу, вибір режиму фото або відео, увімкнення debug-фото, перегляд статусу, отримання поточного кадру та формування звіту. Друга група пов’язана зі сповіщенням: бот надсилає повідомлення про події, помилки камери, початок і завершення відеозапису, а також тестові повідомлення.

Схему взаємодії з Telegram-ботом винесено до додатка А. Користувач надсилає команду або натискає кнопку, бот перевіряє право доступу, виконує дію і за потреби оновлює повідомлення зі статусом. Якщо подія формується не користувацькою командою, а основним циклом моніторингу, вона потрапляє до черги сповіщень і надсилається усім дозволеним користувачам.

Контроль доступу реалізовано через список allowed_user_ids. Якщо користувач не входить до цього списку, він не може отримувати сповіщення або керувати системою. При взаємодії з ботом інформація про користувача може зберігатися у списку відомих або очікуваних користувачів. Такий підхід створює базовий рівень безпеки для локальної системи, оскільки Telegram-токен і дозволені ідентифікатори не повинні бути відкрито вбудовані у публічний код. Основні команди та дії Telegram-бота наведено у таблиці 3.4.

Під час надсилання медіа враховується тип файлу. Для фото використовується надсилання зображення, для відео – надсилання відеофайлу, а для службових повідомлень – текстовий формат. Такий поділ важливий, оскільки Telegram API має різні методи для різних типів контенту [17]. У разі відсутності файлу повідомлення може бути надіслане як текстове,

що використовується для службових подій, помилок камери або тестових повідомлень.

Таблиця 3.4 – Основні команди та дії Telegram-бота

Команда або дія	Призначення	Результат
/start	Початковий запуск діалогу з ботом	Відображення меню та базової інформації
/status	Перегляд поточного стану системи	Повернення активності моніторингу, режиму, розкладу, ROI та сховища
/snapshot	Отримання останнього кадру з камери	Надсилання latest_frame.jpg, якщо він доступний
/report	Формування короткого звіту	Повернення кількості подій і поточних параметрів
/test	Перевірка каналу сповіщень	Надсилання тестового повідомлення
/myid	Отримання Telegram ID користувача	Відображення ідентифікатора для внесення до дозволених
callback-кнопки	Швидке перемикання моніторингу, режиму й debug-фото	Оновлення settings.json і повідомлення про зміну

У лістингу 3.5 показано, що повідомлення надсилаються не одному фіксованому адресату, а всім користувачам зі списку дозволених. Для кожного користувача створюється окрема асинхронна задача, після чого виконується gather з return_exceptions=True. Це дає змогу зафіксувати помилку надсилання одному користувачу без зриву розсилання іншим дозволеним користувачам.

Лістинг 3.5 Фрагмент реалізації розсилання сповіщень:

```
async def broadcast_alert(message_text: str, file_path: str = None,
                          file_type: str = "photo"):
    allowed_user_ids = bot_state.app_state.get_allowed_user_ids()
    tasks = [
        send_alert_to_user(user_id, message_text, file_path, file_type)
        for user_id in allowed_user_ids
    ]
    if not tasks:
        logger.warning("Немає дозволених користувачів для надсилання сповіщення.")
        return
    results = await asyncio.gather(*tasks, return_exceptions=True)
    for i, result in enumerate(results):
        if isinstance(result, Exception):
            logger.error("Помилка надсилання користувачу %s: %s",
                         allowed_user_ids[i], result)
```

3.8 Реалізація локального керування та збереження стану системи

Окрім Telegram-бота, у застосунку передбачено локальне керування через системний трей. Такий інтерфейс потрібен для користувачів, які працюють безпосередньо на комп’ютері, де запущено систему. Через трей можна переглядати або змінювати основні параметри, не редагуючи JSON-файли вручну. Окремий модуль autostart.py формує сценарій автозапуску для Windows, що дає змогу запускати застосунок разом з операційною системою.

Збереження стану системи організовано за допомогою кількох файлів. Службові параметри, що стосуються токена Telegram-бота, початкових

дозволених користувачів, шляхів до сховища, розмірів кадру та інших значень за замовчуванням, зберігаються у .env. Робочі параметри, які змінюються під час експлуатації, зберігаються у settings.json. Історія подій записується в event_history.json. Окремо ведуться списки відомих і очікуваних користувачів Telegram.

Схему збереження стану та матеріалів подій винесено до додатка А. Вона демонструє, що система не потребує окремого сервера бази даних. Для локального застосунку достатньо структурованих файлів, оскільки обсяг налаштувань і журналу подій є помірним. Водночас така модель може бути розширена до SQLite або серверної бази даних у разі підтримки кількох камер і багатьох користувачів.

У settings.json зберігаються ознака активності моніторингу, поточний режим, стан debug-фото, індекс камери, параметри детекції, відеозапису, розкладу, ROI, автозапуску, режиму розпізнавання та шлях до YOLO-моделі. Під час завантаження ці значення проходять нормалізацію. Числові параметри перетворюються до цілих значень і обмежуються мінімальними або максимальними межами. Наприклад, індекс камери не може бути меншим за нуль, кількість кадрів підтвердження не може бути меншою за один, а години розкладу обмежуються діапазоном від 0 до 23.

Журнал подій використовується для подальшого аналізу роботи системи. У нього можуть потрапляти записи про фото, початок відеозапису, завершення відеозапису, помилки камери та відновлення камери. Для кожного запису доцільно зберігати тип події, текстовий опис, шлях до файлу та список виявлених об’єктів. У звіті, який формується Telegram-ботом, ці записи можуть узагальнюватися для користувача.

Для запобігання неконтрольованому накопиченню файлів реалізовано цикл очищення сховища. Він видаляє застарілі файли за строком зберігання та додатково обмежує загальний розмір сховища. Такий підхід особливо важливий для відеорежиму, оскільки відеофрагменти займають більше місця,

ніж окремі знімки. Очищення виконується періодично і не блокує роботу основного циклу.

3.9 Інструкція щодо використання застосунку

Для використання розробленої системи необхідно підготувати середовище виконання, встановити залежності, налаштувати службові параметри та запустити основний файл застосунку. Перед запуском потрібно переконатися, що до комп'ютера підключено камеру або доступний інший відеопристрій, який може бути відкритий через OpenCV. Для типового локального використання застосовується камера з індексом 0.

Підготовка середовища передбачає встановлення Python, створення віртуального середовища та встановлення залежностей із requirements.txt. До переліку залежностей входять aiogram, opencv-python, numpy, Pillow і python-dotenv. Після встановлення залежностей потрібно створити файл .env на основі .env.example. У ньому зазначаються токен Telegram-бота, дозволені ідентифікатори користувачів, параметри камери, каталоги збереження матеріалів і додаткові значення.

Токен Telegram-бота повинен зберігатися тільки у файлі .env або в захищеному середовищі. Для додавання користувача до списку дозволених може використовуватися команда /myid, після чого отриманий ідентифікатор додається до ALLOWED_USER_IDS або через механізми керування користувачами. Такий підхід зменшує ризик несанкціонованого доступу до системи.

Після запуску застосунок створює поточний кадр latest_frame.jpg у каталозі runtime. Це дає змогу команді /snapshot повертати актуальне зображення навіть тоді, коли подія ще не була сформована. Якщо моніторинг вимкнено або розклад не дозволяє спостереження в поточний момент, система

все одно може оновлювати останній кадр, але не формує подій і не надсилає зайвих повідомлень.

Послідовність запуску та первинного налаштування системи наведено у таблиці 3.5.

Таблиця 3.5 – Послідовність запуску та первинного налаштування системи

Етап	Дія користувача	Очікуваний результат
1	Встановити Python і створити віртуальне середовище	Підготовлено ізольоване середовище запуску
2	Встановити залежності з requirements.txt	Доступні OpenCV, aiogram, numpy, Pillow і dotenv
3	Створити .env на основі .env.example	Налаштовано токен, користувачів, камеру, сховище та пороги
4	За потреби вказати шлях до yolov8n.onnx	Увімкнено режим YOLO ONNX
5	Запустити main.py	Створено асинхронні задачі та відкрито камеру
6	Відкрити Telegram-бота і виконати /start	Отримано меню керування
7	Увімкнути моніторинг і вибрати режим photo або video	Система переходить до подієвої фіксації руху
8	Перевірити /snapshot, /status і /test	Підтверджено роботу камери, стану й каналу сповіщень

Для налаштування чутливості детекції слід змінювати min_contour_area і motion_confirmation_frames. Якщо система реагує на дрібні зміни, доцільно збільшити мінімальну площу контуру або кількість кадрів підтвердження.

Якщо система пропускає реальні події, ці значення можна зменшити. Для сцен зі стабільною важливою областю доцільно увімкнути ROI, щоб зменшити вплив руху в неважливих частинах кадру.

Для вибору способу розпізнавання використовується detection_backend. Режим yolo доцільний за наявності коректного ONNX-файлу моделі. Режим opencv може використовуватися для перевірки системи без моделі або як резервний варіант. Під час практичного використання важливо враховувати, що YOLO забезпечує змістовнішу перевірку людини, але потребує більше обчислювальних ресурсів.

3.10 Тестування програмної системи

Тестування програмної системи виконувалося з урахуванням її подієвої логіки. Перевірялися як окремі службові функції, так і основні сценарії використання: запуск застосунку, оброблення конфігурації, перемикання режимів, виявлення руху, розпізнавання людини, формування фото, запис відео, надсилання Telegram-повідомлень, робота з користувачами та очищення сховища. Узагальнену схему перевірки винесено до додатка А.

Автоматизованими тестами охоплено службові функції конфігурації, нормалізації режимів розпізнавання та формування сценарію автозапуску. Тести конфігурації перевіряють перетворення списку дозволених користувачів із рядка у список числових ідентифікаторів, серіалізацію цього списку та оброблення некоректних значень. Тести режимів перевіряють, що значення yolo приймається як YOLO-режим, а всі інші або порожні значення нормалізуються до резервного режиму OpenCV. Перевірка автозапуску має Windows-орієнтований характер, оскільки модуль формує .bat-файл для каталогу Startup.

Функціональне тестування здійснювалося за сценаріями, які відповідають типовому використанню системи. Для кожного сценарію

визначено початкові умови, дію, очікуваний результат і критерій успішності. Основні сценарії наведено у таблиці 3.6.

Таблиця 3.6 – Сценарії функціонального тестування програмної системи

Сценарій	Початкові умови	Дія	Очікуваний результат
Запуск застосунку	Наявний .env і підключена камера	Запустити main.py	Створюються задачі бота, моніторингу, очищення та трея
Перевірка камери	Камера доступна за індексом 0	Виконати /snapshot	Користувач отримує останній кадр
Фоторежим	monitoring_active=true, current_mode=photo	Створити рух у кадрі	Зберігається JPG і надсилається фото
Відеорежим	current_mode=video	Створити рух і припинити його	Починається запис, після паузи формується MP4
YOLO-розпізнавання	detection_backend=yolo, модель доступна	Потрапляння людини в кадр	На debug-кадрі формується рамка person-yolo
Резервний режим	detection_backend=opencv	Потрапляння людини або її ознак у кадр	Використовуються HOG і Haar-детектори
ROI	roi_enabled=true	Рух поза областю інтересу	Подія не формується або кількість спрацювань зменшується
Контроль доступу	Користувач не входить до allowed_user_ids	Надіслати команду боту	Керування системою не надається
Очищення сховища	Є файли, що перевищують строк або розмір	Запустити цикл очищення	Застарілі або надлишкові файли видаляються

Під час перевірки фоторежиму основним критерієм була поява файлу у каталозі motion_screenshots і надсилання відповідного повідомлення через

Telegram. Якщо debug-фото увімкнено, на знімку повинні відображатися результати розпізнавання. Якщо людину виявлено YOLO-моделлю, на кадрі очікується рамка з підписом person-yolo і значенням довіри. Якщо людина не виявлена, система повертає повідомлення про невідомий об’єкт. Такий результат є корисним для аналізу хибних або незначущих спрацювань.

Під час перевірки відеорежиму контролювалися три умови: початок запису після підтвердження руху, додавання кадрів з урахуванням video_fps і завершення запису після відсутності руху протягом установленої затримки. Окремо перевірялося, що під час перемикання з відеорежиму у фоторежим активний запис зупиняється. Це запобігає ситуації, коли відеофайл залишається відкритим після зміни режиму користувачем.

Перевірка відмовостійкості охоплювала випадки недоступності камери та відсутності YOLO-моделі. Якщо камера не відкривається, система не завершується безконтрольно, а виконує повторні спроби підключення з установленою затримкою. Якщо кадри перестають надходити під час роботи, після досягнення граничної кількості невдалих спроб виконується перепідключення. Якщо YOLO-модель не знайдена, система фіксує відповідну ситуацію та може використовувати резервний режим OpenCV. Узагальнені результати перевірки працездатності системи наведено у таблиці 3.7.

Таблиця 3.7 – Узагальнені результати перевірки працездатності системи

Критерій оцінювання	Отриманий результат	Висновок
1	2	3
Працездатність захоплення кадрів	Кадри отримуються через OpenCV VideoCapture і оновлюються в runtime/latest_frame.jpg	Камера може використовуватися як джерело локального відеопотоку

Продовження таблиці 3.7

1	2	3
Стійкість детекції руху	MOG2, морфологія, площа контуру та підтвердження кадрами зменшують випадкові спрацювання	Алгоритм придатний для статичної камери
Інформативність події	Після руху виконується розпізнавання людини та формується текстовий опис	Повідомлення містить не лише факт руху
Швидкість сповіщення	Подія передається через асинхронну чергу до Telegram-бота	Основний цикл не блокується мережевою взаємодією
Гнучкість налаштувань	Параметри зберігаються у JSON і можуть змінюватися без редагування коду	Система адаптується до різних умов спостереження
Локальність оброблення	Кадри аналізуються на комп’ютері, а Telegram отримує тільки сформовані матеріали подій	Зменшено залежність від зовнішніх сервісів аналізу
Обмеження	Якість роботи залежить від освітлення, розміщення камери, моделі та порогів	Потрібне початкове налаштування під конкретну сцену

3.11 Аналіз отриманих результатів та перспективи розвитку

За результатами реалізації встановлено, що розроблена програмна система виконує основні функції, визначені у постановці задачі. Забезпечено отримання кадрів із камери, автоматичне виявлення руху, додаткове розпізнавання людини, збереження фото- або відеоматеріалів, ведення історії подій і надсилання сповіщень користувачу через Telegram-бота. Система також підтримує зміну основних параметрів без редагування програмного коду, що підвищує зручність практичного використання.

Практична цінність розроблення полягає в переході від пасивного перегляду відеопотоку до подієвого інформування. Користувач не повинен постійно спостерігати за камерою, оскільки система самостійно реагує на рух, формує матеріал події і передає повідомлення. Наявність режимів photo і video дає змогу вибрати баланс між економією сховища та повнотою фіксації події. Наявність Telegram-команд забезпечує базове віддалене керування без розроблення окремого мобільного застосунку.

Реалізована система має обмеження, характерні для локальних систем комп’ютерного зору. Якість виявлення руху залежить від стабільності освітлення, шуму камери, наявності тіней, відблисків і рухомих предметів у кадрі. Якість розпізнавання людини залежить від положення об’єкта, масштабу, часткових перекриттів і якості моделі. Через це для реальної сцени потрібне налаштування порогу площі контуру, кількості кадрів підтвердження, ROI і порогу довіри YOLO.

Окремим обмеженням є використання файлового зберігання історії. JSON-файли є достатніми для локального застосунку та помірної кількості подій, проте вони не забезпечують складних запитів, індексування та транзакційності. Якщо система буде розширюватися до кількох камер, багатьох користувачів або тривалого архівного зберігання, доцільно перейти до SQLite або серверної бази даних. Перспективи подальшого розвитку програмної системи наведено у таблиці 3.8.

Таблиця 3.8 – Перспективи подальшого розвитку програмної системи

Напрям розвитку	Сутність удосконалення	Очікуваний ефект
Підтримка кількох камер	Додавання незалежних потоків моніторингу та окремих налаштувань для кожної камери	Розширення системи на кілька зон спостереження
SQLite або серверна база	Перенесення історії подій із JSON до структурованого сховища	Зручніший пошук, фільтрація та статистичний аналіз
Вебінтерфейс	Створення локальної панелі перегляду стану, кадрів і журналу	Зручніше адміністрування без редагування файлів
Покращена модель розпізнавання	Використання новішої або донавченої моделі для конкретної сцени	Зменшення кількості помилкових або пропущених спрацювань
Аналітика подій	Побудова статистики за часом, типом подій і кількістю виявлень	Оцінювання завантаженості контрольованої зони
Розширені права доступу	Розподіл користувачів на адміністратора і спостерігачів	Підвищення безпеки керування системою
Захист конфігурації	Шифрування службових параметрів і обмеження доступу до .env	Зменшення ризику витоку токена та ідентифікаторів

Перспективним напрямом також є розширення механізму прийняття рішення про подію. У поточній реалізації подія формується на основі руху, режиму моніторингу, розкладу, ROI та результату розпізнавання. У подальшому можна додати класифікацію типів подій, наприклад поява

людини, тривале перебування в кадрі, рух у забороненій зоні або повернення камери після втрати сигналу. Це дасть змогу формувати різні рівні важливості повідомлень.

Ще одним напрямом удосконалення є оптимізація швидкодії. Для цього можна зменшувати роздільну здатність кадру на етапі первинної детекції, запускати YOLO не для кожного кадру з рухом, а з певним інтервалом, використовувати апаратне прискорення за наявності відповідного обладнання або переносити частину аналізу в окремий процес. Такі зміни можуть бути важливими для слабших комп’ютерів або одночасної роботи з кількома відеопотоками.

З погляду безпеки доцільно розширити контроль доступу. У поточній реалізації застосовується список дозволених Telegram-користувачів. У подальшому можна передбачити рольову модель, журнал дій користувачів, підтвердження критичних команд і повідомлення адміністратору про спроби доступу з невідомих облікових записів. Це підвищить керованість системи під час використання в офісному або навчальному середовищі.

Окрему увагу приділено експлуатаційним обмеженням системи. Для зменшення ризиків передбачено збереження службових параметрів у файлі .env, перевірку дозволених Telegram-користувачів, налаштування області інтересу, вибір режиму фіксації подій і можливість подальшого розгортання застосунку як локального програмного засобу. Ці рішення забезпечують практичну придатність системи без перевантаження основного тексту надмірними службовими деталями.

ВИСНОВКИ

У рамках кваліфікаційної роботи розроблено і реалізовано програмну систему відеоспостереження з автоматичним виявленням руху та передачею мультимедійних даних через бота в Telegram. Робота охоплює повний цикл створення застосунку: від аналізу проблемної області до реалізації функціоналу із використанням сучасних інструментів, та вирішено такі завдання:

– проведено аналіз сучасного стану проблемної області автоматизованого відеоспостереження та існуючих рішень, що дало можливість виявити недоліки традиційного пасивного відеозапису та обґрунтувати доцільність переходу до подієвої моделі роботи;

– проведено аналіз методів виявлення руху та підходів до розпізнавання людини у відеопотоці, що дало можливість детально вивчити їх обмеження та переваги для подальшого вибору найкращої комбінації алгоритмів;

– сформовано критерії вибору методів та архітектурних рішень, що дало можливість розробити структуру програмної системи та ефективно змоделювати взаємодію її модулів;

– розроблено програмний застосунок, що поєднує модуль первинної детекції руху, уточнювальний модуль розпізнавання людини та механізм надсилання сповіщень, що дозволило задовольнити мету кваліфікаційної роботи.

Розглянуто існуючі рішення для задач відеомоніторингу, зокрема системи відеозапису, хмарні камери з віддаленим доступом та складні системи інтелектуальної відеоаналітики. Проаналізовано наукові джерела, вивчено існуючі методи аналізу відеопотоку, особливо ті, які пов’язані з фоновим відніманням, оптичним потоком та виділенням ознак об’єктів у кадрі.

Для отримання кадрів та детекції руху використано методи бібліотеки комп’ютерного зору OpenCV, зокрема алгоритм MOG2. Для розпізнавання людини застосовано нейромережеву модель YOLO у форматі ONNX, а також

резервний класичний підхід на базі HOG-детекторів та Haar-каскадів. Як інтерфейс для віддаленого керування та сповіщення обрано бібліотеку aiogram для взаємодії з Telegram, а вся логіка системи реалізована мовою програмування Python.

Проведено тестування працездатності системи, зокрема перевірку захоплення відеопотоку, стійкості алгоритмів виявлення руху, а також тестування режимів формування і відправлення фото- та відеоматеріалів подій дозволеним користувачам.

Розглянуто перспективи покращення системи, які включають у себе інтеграцію підтримки кількох камер спостереження, перенесення історії подій до структурованої бази даних SQLite, розроблення локального вебінтерфейсу, а також вдосконалення аналітики подій і розширення прав доступу.

Результати кваліфікаційної роботи можуть бути корисними у сфері домашніх систем безпеки, для автоматизації контролю офісних та робочих приміщень, а також при подальшому розробленні прикладних засобів інтелектуальної відеоаналітики.

Результати роботи апробовано у вигляді тез доповіді під час X Всеукраїнської студентської наукової конференції «Розвиток сучасної науки: актуальні питання теорії та практики» [31].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Duong, H.-T., Le, V.-T., & Hoang, V. T. (2023). Deep learning-based anomaly detection in video surveillance: A survey. Sensors, 23(11), 5024.

2. Кобилін О.А., Творошенко І.С. (2021). Методи цифрової обробки зображень: навч. посібник. *Харків: ХНУРЕ*.

3. Tvoroshenko, I., Pomazan, V., Gorokhovatskyi, V., & Kobylin, O. (2023). Application of video data classification models using convolutional neural networks. International Journal of Academic and Applied Research, 7(11), 134–145.

4. Путятін, Є. П., Гороховатський, В. О., & Матат, О. О. (2006). Методи та алгоритми комп'ютерного зору: навчальний посібник. Харків: ХНУРЕ.

5. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, *Сучасні інформаційні системи*, 6(3), С. 5-12.

6. Daradkeh, Y.I., Tvoroshenko, I., Gorokhovatskyi, V., Latiff, L.A., and Ahmad, N. (2021) Development of Effective Methods for Structural Image Recognition Using the Principles of Data Granulation and Apparatus of Fuzzy Logic, *IEEE Access*, 9, pp. 13417-13428.

7. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Tools for Fast Metric Data Search in Structural Methods for Image Classification, IEEE Access, 10, pp. 124738-124746.

8. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

9. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, *IEEE Access*, vol. 11, pp. 126938-126949.

10. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Development of an application for recognizing emotions using convolutional neural networks, *International Journal of Academic Information Systems Research*, 7(7), pp. 25-36.

11. Gorokhovatskyi, V., Gadetska, S., & Stiahlyk, N. (2023). Accelerating image classification based on a model for estimating descriptor-to-class distance. International Journal of Computing, 22(4), 485–492.

12. Gorokhovatskyi V., Tvoroshenko I., and Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 33, no. 1, pp. 113-125.

13. Yakovleva O., Matúšová S., Tvoroshenko I., and Isaiev Y. (2024) Visitor counting based on video stream analysis from surveillance cameras to solve various business problems, *Verejná správa a regionálny rozvoj ekonómia, manažment a marketing*, XX(1), pp. 67-87.

14. Suprun A., Tvoroshenko I., Gorokhovatskyi V., and Yakovleva O. (2025) Development and research of a method for the combined use of large language models for text generation, *International Journal of Academic and Applied Research*, 9(10), pp. 249-263.

15. Bohdan N., Tvoroshenko I., Gorokhovatskyi V., and Kobylin O. (2025) Development of a hybrid method to enhance context memory for a chatbot application based on large language models, *International Journal of Academic Information Systems Research*, 9(10), pp. 7-18.

16. Python Software Foundation. (n.d.). asyncio — Asynchronous I/O. Python Documentation. URL: https://docs.python.org/3/library/asyncio.html (дата звернення 05.06.2026).

17. Telegram. (n.d.). Telegram Bot API. URL: https://core.telegram.org/bots/api (дата звернення 05.06.2026).

18. OpenCV. (n.d.). OpenCV documentation: VideoCapture, image processing, background subtraction, morphology and contours. URL: https://docs.opencv.org/ (дата звернення 05.06.2026).

19. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set, *IEEE Access*, vol. 12, pp. 73376-73385.

20. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2024) Improving the effectiveness of image classification structural methods by compressing the description according to the information content criterion, *Computers, Materials & Continua*, vol. 80, no. 2, pp. 3085-3106.

21. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., and Hudáková M. (2025) Image description compression in classification structural methods, *IEEE Access*, vol. 13, pp. 43631-43641.

22. Gorokhovatskyi V., Chmutov Y., Tvoroshenko I., and Kobylin O. (2025) Reducing computational costs by compressing the structural description in image classification methods, *Advanced Information Systems*, vol. 9, no. 1, pp. 5–12.

23. Gorokhovatskyi V., Tvoroshenko I. (2024) An effective method for transforming an image description into a compact vector for classification. *Information Technology and Implementation (Satellite): Conference Proceedings, November 21, 2024, Kyiv, Ukraine / Ministry of Education and Science of Ukraine, Taras Shevchenko National University of Kyiv and [etc]; Vitaliy Snytyuk (Editor). – Kyiv: Publishing House «Caravela»*, pp. 25-28.

24. Kobylin, O. A., & Kharchenko, A. I. (2024). Classification techniques for computer vision. Information Processing Systems, 3(178), 33–41.

25. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, real-time object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 779–788. URL: https://doi.org/10.1109/CVPR.2016.91 (дата звернення 05.06.2026).

26. ONNX. (n.d.). Open Neural Network Exchange documentation. URL: https://onnx.ai/ (дата звернення 05.06.2026).

27. Bray, T. (2017). The JavaScript Object Notation (JSON) Data Interchange Format (RFC 8259). Internet Engineering Task Force. URL: https://doi.org/10.17487/RFC8259 (дата звернення 04.06.2026).

28. Python Dotenv. (n.d.). python-dotenv. URL: https://pypi.org/project/python-dotenv/ (дата звернення 05.06.2026).

29. Pytest. (n.d.). pytest documentation. URL: https://docs.pytest.org/en/stable/ (дата звернення 05.06.2026).

30. PyInstaller Development Team. (2026). PyInstaller Manual: PyInstaller 6.20.0 documentation. URL: https://pyinstaller.org/ (дата звернення 05.06.2026).

31. Петренко К.А. (2026) Розроблення системи відеоспостереження з автоматичним виявленням руху та передаванням мультимедійних даних через бота в Telegram. *Розвиток сучасної науки: актуальні питання теорії та практики: матеріали X Всеукраїнської студентської наукової конференції (22 травня 2026 року). Тернопіль: ГО «Молодіжна наукова ліга»,* С. 414-416.