Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп'ютерних наук

Кваліфікаційна робота магістра

з теми: **“Розробка інтелектуальної системи охорони об'єктів”**

Виконав: здобувач вищої освіти
групи KN1-M24
спеціальності 122 Комп'ютерні науки
Зелінський Дмитро Русланович

Керівник:
Кушнір Оксана Климівна,
кандидат економічних наук,
доцент кафедри економіки підприємства

Науковий консультант:
Іванюк Віталій Анатолійович,
доктор технічних наук, доцент,
завідувач кафедри комп’ютерних наук

Рецензент:
Оптасюк Сергій Васильович,
кандидат фізико-математичних наук, доцент,
завідувач кафедри фізики

Кам’янець-Подільський – 2025

ЗМІСТ

АНОТАЦІЯ

Зелінський Д.Р., здобувач вищої освіти. Розробка інтелектуальної системи охорони об'єктів. - Кваліфікаційна робота на правах рукопису.

Науковий керівник: Кушнір Оксана Климівна, кандидат економічних наук, доцент кафедри економіки підприємства.

Науковий консультант: Іванюк Віталій Анатолійович, доктор технічних наук, доцент, завідувач кафедри комп’ютерних наук.

Кваліфікаційна (магістерська) робота на здобуття ступеня магістра за спеціальністю 122 “Комп’ютерні науки”. - Кам’янець-Подільський національний університет імені Івана Огієнка, Кафедра комп'ютерних наук, Кам’янець-Подільський, 2025.

Метою роботи є створення та програмна реалізація інтелектуальної системи відеоспостереження, здатної виявляти потенційні загрози в реальному часі для підвищення рівня безпеки приміщень.

У роботі проведено аналіз сучасних методів інтелектуального аналізу відеоданих, розглянуто існуючі аналоги систем безпеки та виявлено їхні недоліки. Досліджено алгоритми детекції руху та розпізнавання об’єктів, а також обґрунтовано вибір інструментальних засобів для реалізації власного рішення.

Розроблено архітектуру програмної системи, що включає модулі захоплення відеопотоку, обробки зображень та логіки сповіщень. Реалізовано функціонал для автоматичної ідентифікації підозрілої активності та створено інтерфейс користувача для керування налаштуваннями моніторингу.

Впровадження розробленої системи дозволяє автоматизувати процес охорони, забезпечити оперативне інформування про позаштатні ситуації та зменшити вплив людського фактора на надійність системи безпеки.

Ключові слова: інтелектуальна система, відеоспостереження, комп’ютерний зір, безпека, iot, детекція руху, моніторинг.

ABSTRACT

Zelinskyi D.R., applicant for higher education. Development of an intelligent facility security system. – Qualification paper (manuscript).

Scientific supervisor: Kushnir Oksana Klymivna, Candidate of Economic Sciences (Ph.D.), Associate Professor at the Department of Enterprise Economics.

Scientific consultant: Ivanyuk Vitalii Anatoliiovych, Doctor of Technical Sciences, Associate Professor, Head of the Department of Computer Science.

Master’s qualification paper for the Master’s degree in specialty 122 "Computer Science". – Kamianets-Podilskyi National Ivan Ohiienko University, Department of Computer Science, Kamianets-Podilskyi, 2025.

The aim of the work is the creation and software implementation of an intelligent video surveillance system capable of detecting potential threats in real time to enhance the security of premises.

The work provides an analysis of modern methods for intelligent video data analysis, examines existing security system analogs, and identifies their shortcomings. Motion detection and object recognition algorithms were investigated, and the choice of tools for implementing the proposed solution was justified.

The software system architecture was developed, including modules for video stream capture, image processing, and notification logic. Functionality for the automatic identification of suspicious activity was implemented, and a user interface was created to manage monitoring settings.

Implementation of the developed system allows for automating the security process, ensuring prompt notification of abnormal situations, and minimizing the impact of the human factor on the reliability of the security system.

Keywords: intelligent system, video surveillance, computer vision, security, iot, motion detection, monitoring.

ВСТУП

Актуальність теми. Сучасний етап розвитку інформаційних технологій характеризується стрімким зростанням кількості підключених пристроїв у рамках концепції "Інтернету речей" (IoT) [2, 5, 7]. Системи відеоспостереження стали невід’ємною складовою безпеки як у комерційному секторі, так і в приватному житті. Проте, існуючі на ринку рішення часто мають суттєві недоліки: високу вартість спеціалізованого обладнання, складність налаштування та ризики витоку приватних даних при використанні хмарних сервісів сторонніх виробників.

Водночас у користувачів накопичується значна кількість морально застарілих, але працездатних мобільних пристроїв та персональних комп'ютерів, обчислювальна потужність яких достатня для виконання задач комп'ютерного зору. Актуальність роботи полягає у необхідності розробки програмних засобів, які дозволяють інтегрувати різнорідні обчислювальні пристрої в єдину інтелектуальну систему охорони. Використання загальнодоступних месенджерів (зокрема, Telegram) як інтерфейсу керування дозволяє вирішити проблему віддаленого доступу без необхідності виділення статичних IP-адрес чи складного мережевого конфігурування. Такий підхід не лише знижує вартість впровадження систем безпеки, але й підвищує їхню автономність та захищеність.

Метою роботи є підвищення ефективності та доступності систем охоронного відеоспостереження шляхом розробки програмного комплексу, що використовує методи комп'ютерного зору для детекції подій та інтегрується з хмарними месенджерами для оперативного сповіщення.

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

1. Провести аналіз існуючих методів та алгоритмів детекції руху і розпізнавання облич у відеопотоці.

2. Дослідити протоколи передачі відеоданих (RTSP, MJPEG) та обґрунтувати вибір технологій для побудови розподіленої системи відеоспостереження.

3. Розробити архітектуру системи, що включає підсистему захоплення відео, модуль аналізу зображень та модуль взаємодії з API мессенджера Telegram.

4. Реалізувати програмне забезпечення, яке забезпечує детекцію руху, виділення облич у кадрі та автоматичне сповіщення користувача з надсиланням мультимедійних даних.

5. Виконати експериментальне дослідження розробленої системи, оцінити її швидкодію та надійність функціонування на обладнанні з обмеженими ресурсами.

Об’єкт дослідження — процес автоматизованого моніторингу та аналізу відеопотоків у системах безпеки.

Предмет дослідження — методи та алгоритми комп'ютерного зору для виявлення рухомих об'єктів та облич, а також методи організації взаємодії між локальними серверами відеообробки та віддаленими клієнтами через хмарні платформи.

Методи дослідження. У роботі використано:

- *методи цифрової обробки зображень* — для попередньої обробки кадрів та виділення зон інтересу;
- *алгоритм віднімання фону* — для детекції руху;
- *метод каскадів Хаара* — для виявлення облич на зображенні;
- *об'єктно-орієнтоване програмування* — для реалізації програмних модулів системи.

Наукова новизна одержаних результатів. У магістерській роботі отримано наступні нові наукові результати:

1. Удосконалено метод побудови бюджетних систем відеоспостереження, який, на відміну від існуючих, базується на гібридному використанні локальних обчислювальних потужностей для "важких" задач

(аналіз відео) та хмарних месенджерів для "легких" задач (сповіщення та керування), що забезпечує високу приватність даних при мінімальних витратах.

2. Набув подальшого розвитку підхід до оптимізації навантаження на канал зв'язку шляхом впровадження алгоритму селективної передачі даних: відеопотік обробляється локально, а в мережу передаються лише ключові кадри з виявленими подіями (рух, обличчя).

Практичне значення одержаних результатів. Практична цінність роботи полягає у створенні повнофункціонального програмного продукту, який дозволяє:

- Використовувати старі смартфони та ПК як елементи сучасної системи безпеки, що вирішує проблему утилізації електроніки.
- Забезпечити відеоспостереження в місцях без стабільного високошвидкісного інтернету (завдяки локальній обробці).
- Застосовувати систему у побуті (охорона житла, функція "радіоняня") без необхідності купівлі дороговартісних ліцензій чи підписок.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНИХ МЕТОДІВ ТА ТЕХНОЛОГІЙ ІНТЕЛЕКТУАЛЬНОГО ВІДЕОСПОСТЕРЕЖЕННЯ

1.1. Огляд існуючих систем відеоспостереження та їх недоліки

Сучасний ринок систем безпеки пропонує широкий спектр рішень, які варіюються від професійних комплексів відеоспостереження до аматорських мобільних додатків. Інтеграція пристроїв у єдині системи стає трендом, проте всебічна взаємодія створює нові вектори загроз для користувачів. Для визначення ніші проектованої системи доцільно провести класифікацію та аналіз існуючих рішень за критеріями вартості, функціональності та рівня захисту персональних даних [19].

1. Професійні апаратні комплекси (на прикладі Hikvision, Dahua). Цей сегмент ринку представлений IP-камерами та відеореєстраторами (NVR), які забезпечують високу якість зображення та стабільність роботи.

- *Переваги:* Висока роздільна здатність, наявність ІЧ-підсвітки, захист від погодних умов.
- *Недоліки:* Основним бар'єром для пересічного користувача є висока вартість обладнання та монтажу. Такі системи часто потребують прокладання кабельних мереж, виділених статичних IP-адрес для віддаленого доступу та складного налаштування мережевого обладнання (Port Forwarding), що вимагає специфічних технічних знань. Крім того, професійне ПЗ часто вимагає придбання ліцензій для підключення додаткових каналів відео.

2. Екосистеми розумного дому (на прикладі Ajax Systems). Системи типу Ajax здійснили революцію у зручності користування (User Experience). Вони використовують датчики руху з фотофіксацією (MotionCam), які надсилають серію знімків при тривозі.

- *Переваги:* Простота встановлення, бездротові протоколи зв’язку, зручний мобільний застосунок.

- *Недоліки:* Висока вартість централі, датчиків та закритість екосистеми. Користувач не може інтегрувати у систему довільне обладнання (наприклад, стару веб-камеру), а змушений купувати лише сертифіковані пристрої бренду.

3. Хмарні сервіси та мобільні додатки. Існує клас програмного забезпечення, що дозволяє перетворити смартфон на IP-камеру (наприклад, IP Webcam, Alfred).

- *Переваги:* Низький поріг входження, можливість використання наявних ґаджетів.
- *Недоліки:* Більшість таких сервісів працюють за моделлю підписки (SaaS). Безкоштовні версії мають обмежений функціонал або транслюють рекламу.

Ключові проблеми існуючих рішень:

А) Проблема приватності та кібербезпеки. Зростання кількості підключених пристроїв прямо корелює зі зростанням небезпек. Використання сторонніх хмарних серверів для трансляції відео створює ризик перехоплення даних. Користувач не може бути впевненим, що відео з його камер залишається конфіденційним і до нього не мають доступу адміністратори хмарного сервісу або зловмисники. Відомі випадки, коли через вразливості в IoT-пристроях зловмисники отримали доступ до приватного життя користувачів.

Б) Економічна нераціональність для побутових задач. Для простих задач, таких як нагляд за домашніми тваринами, функція "радіоняні" або охорона орендованої квартири, купівля професійного обладнання є фінансово невиправданою. Водночас у користувачів часто наявні старі смартфони або ПК, обчислювальний ресурс яких достатній для вирішення цих задач без додаткових капіталовкладень.

В) Залежність від інтернет-з'єднання. Більшість сучасних систем перестають бути "розумними" при відсутності інтернету, оскільки аналітика виконується у хмарі. Локальна обробка даних, яка пропонується у даній роботі,

дозволяє зберігати функціональність (запис, детекція) навіть при ізоляції локальної мережі.

Таким чином, існує потреба у розробці програмного забезпечення, яке поєднує переваги професійної аналітики (детекція руху, розпізнавання облич) з доступністю та приватністю локальних рішень, використовуючи загальнодоступні канали зв'язку (Telegram) замість вразливих хмарних веб-інтерфейсів.

1.2. Порівняльний аналіз методів комп'ютерного зору в контексті систем реального часу [1, 3, 22]

Ефективність системи відеоспостереження визначається балансом між точністю детекції подій та обчислювальною складністю алгоритмів. Оскільки цільовою платформою розробки є обладнання з обмеженими ресурсами (персональні комп'ютери без дискретних графічних прискорювачів), критичним критерієм вибору методів є можливість роботи в режимі реального часу (Real-Time Performance) на центральному процесорі (CPU).

1.2.1. Методи виявлення руху (Motion Detection)

Задача виявлення руху є першим етапом обробки відеопотоку, що дозволяє фільтрувати статичні сцени та економити дисковий простір.

1. **Різницеві методи (Frame Differencing).**

Найбільш поширений підхід для систем реального часу. Алгоритм базується на попіксельному порівнянні двох послідовних кадрів [9, 20] I_t та I_{t-1}. Результуюча маска руху $M(x,y)$ формується за правилом:

$$M(x,y) = \begin{cases} 1, & \text{if } |I_t(x,y) - I_{t-1}(x,y)| > T \\ 0, & \text{else} \end{cases}$$

де T — порогове значення (Threshold).

- *Переваги:* Екстремально низька обчислювальна складність ($O(N)$), що дозволяє обробляти HD-відео навіть на мобільних процесорах.

- *Недоліки:* Чутливість до шумів сенсора камери та глобальних змін освітлення.

2. **Метод оптичного потоку (Optical Flow).**

Метод, що обчислює векторне поле швидкостей для кожної точки зображення, базуючись на припущенні про сталість яскравості об'єкта при його зміщенні. Прикладом є алгоритм Лукаса-Канаде.

- *Переваги:* Дозволяє визначити не лише факт руху, але й його напрямок та швидкість, що корисно для трекінгу об'єктів.
- *Недоліки:* Висока математична складність. Обчислення щільного оптичного потоку (Dense Optical Flow) вимагає значних ресурсів CPU або використання GPU, що унеможливлює його застосування на слабкому обладнанні для задач 24/7.

Для реалізації системи обрано різницевий метод, оскільки він забезпечує достатню точність для активації запису відео при мінімальному споживанні енергії та ресурсів процесора.

1.2.2. Методи детекції облич (Face Detection)

Задача пошуку обличчя на зображенні є ресурсоємною. Розглянемо основні підходи.

1. **Метод Віоли-Джонса (Каскади Хаара).**

Класичний алгоритм машинного навчання, запропонований у 2001 році [24]. Метод використовує прості прямокутні ознаки (Haar-like features), які обчислюються надзвичайно швидко завдяки використанню "інтегрального зображення" (Integral Image). Класифікація відбувається каскадно: зображення сканується вікном, і якщо область не проходить перший етап перевірки (найпростіші ознаки), вона відкидається і не аналізується далі.

- *Переваги:* Висока швидкодія на CPU. Більшість фонових областей відсіюються на ранніх етапах каскаду, що робить метод ідеальним для відеопотоку, де обличчя займає малу частину кадру.

- *Недоліки:* Чутливість до кута повороту голови (працює переважно з фронтальними обличчями) та умов освітлення.

2. **Гістограми орієнтованих градієнтів (HOG + SVM).**

Метод базується на розрахунку напрямків градієнтів яскравості в локальних областях. Вектор ознак подається на вхід класифікатора (зазвичай метод опорних векторів — SVM).

- *Переваги:* Стабільніший за метод Віоли-Джонса, краще розпізнає форми.
- *Недоліки:* Повільніший процес обчислення ознак, що може призвести до падіння частоти кадрів (FPS) на слабких машинах.

3. **Згорткові нейронні мережі (CNN / Deep Learning).**

Сучасні методи (MTCNN, SSD, YOLO, ResNet) використовують глибоке навчання для виділення складних ієрархічних ознак.

- *Переваги:* Найвища точність (State-of-the-Art), стійкість до перекриттів, поворотів, поганого освітлення.
- *Недоліки:* Високі вимоги до апаратного забезпечення. Інференс (виконання) нейромережі на звичайному CPU може займати сотні мілісекунд, що створює помітні затримки. Для роботи в реальному часі необхідні дорогі відеокарти (NVIDIA GPU з підтримкою CUDA) або спеціалізовані прискорювачі (TPU), що суперечить концепції бюджетної системи.

1.2.3. Обґрунтування вибору методу Хаара для проектованої системи

Порівняльний аналіз показує, що хоча нейромережеві методи (CNN) забезпечують кращу точність, вони є неприйнятними для даної роботи через апаратні обмеження.

Ключові аргументи на користь вибору каскадів Хаара:

1. **Апаратна сумісність:** Система проектується для роботи на застарілих офісних ПК та ноутбуках, які часто не мають дискретних відеокарт. Метод Хаара ефективно працює на будь-якому x86/x64 процесорі без додаткових інструкцій.

2. **Швидкодія:** Час обробки одного кадру методом Хаара складає 10-30 мс, що дозволяє підтримувати стабільні 25-30 FPS. Нейромережі на аналогічному залізі дають 2-5 FPS, що є критичним для охоронної системи, де важливо не пропустити швидкий рух.

3. **Інтеграція:** Бібліотека Accord.NET, обрана для реалізації на платформі.NET, містить високооптимізовану реалізацію HaarObjectDetector з підтримкою паралельних обчислень (UseParallelProcessing = true), що дозволяє ефективно використовувати багатоядерність сучасних CPU .

Для досягнення мети роботи — створення доступної та дешевої системи відеоспостереження — компроміс між точністю (Haar) та вартістю впровадження (відсутність потреби в GPU) вирішено на користь методу Віоли-Джонса.

1.3. Аналіз протоколів передачі відеоданих у розподілених системах моніторингу

Ефективність системи відеоспостереження у парадигмі Інтернету речей (IoT) безпосередньо залежить від обраного методу транспортування медіаданих. У сучасних системах домінуючими є два стандарти: RTSP (Real Time Streaming Protocol) та MJPEG (Motion JPEG). Проведемо їх порівняльний аналіз у контексті задач автоматизованої відеоаналітики .

Протокол RTSP (Real Time Streaming Protocol) - це протокол керування потоковими даними, де передача відео зазвичай відбувається поверх UDP з використанням міжкадрового стиснення (кодеки H.264/H.265) [16, 18] .

Переваги: Висока ефективність стиснення та низька затримка передачі, що критично для оперативного моніторингу людиною .

Недоліки: Вразливість до втрати пакетів. Оскільки UDP не гарантує доставку, втрата даних призводить до появи артефактів («розсипання» картинки), які алгоритми детекції руху помилково інтерпретують як зміни в кадрі, що спричиняє хибні спрацювання.

Протокол MJPEG (Motion JPEG over HTTP) - це метод, при якому відеопотік передається як послідовність окремо стиснених JPEG-зображень за протоколом HTTP/TCP. Кожен кадр є незалежним, міжкадрова компресія відсутня.

Переваги: Гарантована цілісність кожного кадру завдяки TCP. Відсутність артефактів стиснення забезпечує чіткі контури об'єктів, що є ідеальним для роботи класифікаторів облич (Haar Cascades) .

Недоліки: Значно вищий бітрейт порівняно з RTSP, що створює більше навантаження на мережу.

Вибір протоколу: Порівняльний аналіз показує, що вибір протоколу залежить від цільового призначення підсистеми. Для перегляду трансляції людиною кращим є RTSP завдяки плавності руху. Проте для задач машинного зору (аналітична обробка) критично важливим є MJPEG, оскільки він виключає компресійні артефакти — головну причину помилок першого роду (False Positives) у детекторах руху. Враховуючи гібридний характер розроблюваної системи, реалізовано підтримку обох протоколів: MJPEG як джерело даних для аналітики, а RTSP — як опція для перегляду

1.4. Постановка задачі магістерського дослідження

Формулювання проблеми. Існуючі на ринку рішення мають полярний характер: це або професійні апаратні комплекси (Hikvision, Dahua), які вимагають значних капіталовкладень та складного монтажу, або хмарні IoT-сервіси, які ставлять під загрозу конфіденційність даних користувача. При цьому існує значний парк морально застарілої обчислювальної техніки (персональні комп'ютери, ноутбуки, смартфони), ресурсів якої достатньо для виконання базових задач комп'ютерного зору, однак відповідне програмне забезпечення для їх інтеграції в єдину захищену систему відсутнє або має обмежений функціонал.

Мета дослідження. Метою роботи є розробка методів та програмних засобів для побудови доступної системи інтелектуального відеоспостереження, яка забезпечує автоматизовану детекцію руху та розпізнавання облич на обладнанні загального призначення, використовуючи захищені канали месенджера Telegram для оперативного оповіщення та керування.

Об’єкт дослідження. Процеси автоматизованої обробки відеопотоків у системах охоронного моніторингу реального часу.

Предмет дослідження. Методи та алгоритми комп'ютерного зору для виділення об'єктів на відеозображеннях (Motion Detection, Face Detection) та методи організації клієнт-серверної взаємодії через API хмарних месенджерів.

Задачі дослідження. Для досягнення поставленої мети необхідно вирішити наступні задачі:

1. **Розробити архітектуру програмного комплексу**, яка дозволяє використовувати різнорідні пристрої (IP-камери, Android-смартфони) як джерела відеосигналу, а персональний комп'ютер — як сервер обробки.
2. **Обґрунтувати та реалізувати алгоритмічне забезпечення**, що базується на комбінації різницевих методів виявлення руху та каскадних класифікаторів Хаара. Забезпечити оптимізацію цих алгоритмів для роботи в реальному часі на процесорах без апаратної підтримки CUDA.
3. **Реалізувати підсистему мережевої взаємодії**, що підтримує роботу з протоколами RTSP та MJPEG для забезпечення гнучкості підключення джерел відеосигналу.
4. **Розробити механізм інтеграції з Telegram API**, який забезпечує двосторонній зв'язок: відправку мультимедійних тривожних повідомлень користувачу та прийом команд керування системою (активація охорони, запит стану, отримання фото).
5. **Провести експериментальне дослідження** розробленої системи, результати якого, включаючи оцінку швидкодії та надійності, **викладено у Додатку А**, перевірити коректність детекції подій та оцінити ефективність використання ресурсів (дискового простору, CPU) у порівнянні з аналогами.

РОЗДІЛ 2
МАТЕМАТИЧНЕ ТА АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

У даному розділі розглядаються математичні моделі та алгоритмічні підходи, що лежать в основі розробленої системи інтелектуального відеоспостереження. Основна увага приділяється формалізації процесів попередньої обробки відеосигналу, методам виявлення руху та розпізнавання образів, а також моделюванню логіки взаємодії компонентів системи.

2.1. Математична модель обробки відеопотоку та виділення ключових кадрів

Відеоспостереження в реальному часі вимагає обробки великих масивів даних. Для забезпечення швидкодії системи на обчислювальних потужностях середнього рівня (персональні комп'ютери, ноутбуки) необхідно побудувати ефективну математичну модель подання відеоданих та стратегію відбору інформативних кадрів (Key Frames Extraction).

2.1.1. Формалізація відеопотоку та подання зображень

З точки зору комп'ютерного зору, відеопотік V можна розглядати як впорядковану в часі послідовність дискретних зображень (кадрів) F:

$$V = \{F_1, F_2, \dots, F_t, \dots, F_N\},$$

де t — індекс кадру, що відповідає дискретному моменту часу, а N — загальна кількість кадрів у сесії спостереження.

Кожен окремий кадр F_t являє собою двовимірну матрицю розмірністю $W \times H$ (ширина та висота в пікселях), елементами якої є вектори значень кольору. У кольоровому просторі RGB, який використовується камерами смартфонів та веб-камерами (як зазначено в технічних вимогах до системи), стан пікселя з координатами (x, y) описується вектором:

$$F_t(x, y) = \begin{bmatrix} R_t(x, y) \\ G_t(x, y) \\ B_t(x, y) \end{bmatrix},$$

де $R, G, B \in [0, 255]$ — інтенсивності червоного, зеленого та синього каналів відповідно.

Оскільки алгоритми комп’ютерного зору, зокрема методи детекції руху та каскади Хаара (які використовуються в бібліотеці Accord.NET), є обчислювально складними для роботи з триканальними зображеннями, першим етапом математичної обробки є перетворення кольорового зображення у напівтонове (grayscale). Це дозволяє зменшити розмірність даних у три рази, зберігаючи інформацію про структуру об'єктів та їх контури [1, 19].

Перетворення здійснюється за допомогою лінійної комбінації каналів, що враховує фізіологічне сприйняття яскравості людським оком (стандарт ITU-R BT.601) [1, 20]:

$$I_t(x, y) = 0.299 \cdot R_t(x, y) + 0.587 \cdot G_t(x, y) + 0.114 \cdot B_t(x, y),$$

де $I_t(x, y)$ — скалярне значення інтенсивності (яскравості) пікселя у кадрі t. Саме матриця I_t використовується надалі для аналізу руху та пошуку облич.

2.1.2. Процедура нормалізації та масштабування (Downsampling)

Аналіз вихідного коду системи показує використання процедури зміни розміру зображення перед детекцією облич (використання resizer.Apply(imCopy) та коефіцієнтів xscale, yscale). Це необхідний крок для оптимізації швидкодії, оскільки складність алгоритму пошуку облич прямо пропорційна кількості пікселів.

Математично операція масштабування (downsampling) описується як відображення матриці I_t розмірності $W \times H$ у матрицю I'_t розмірності $W' \times H'$, де $W' < W$ та $H' < H$.

Якщо k — коефіцієнт масштабування ($k < 1$), то координати пікселя (x', y') у новому зображенні співвідносяться з оригінальними координатами як:

$$x = \lfloor \frac{x'}{k} \rfloor, \quad y = \lfloor \frac{y'}{k} \rfloor$$

Для запобігання ефекту аліасингу (aliasing) при зменшенні зображення застосовується інтерполяція (найчастіше білінійна або бікубічна). Значення

яскравості нового пікселя $I'_t(x', y')$ обчислюється як зважена сума яскравостей сусідніх пікселів вихідного зображення. Це дозволяє зберегти загальну структуру зображення, зменшивши обчислювальне навантаження на детектор Хаара.

2.1.3. Модель виділення ключових кадрів на основі аналізу руху

У контексті систем розумного відеонагляду, що працюють з обмеженим дисковим простором та пропускною здатністю мережі (передача фото в Telegram), недоцільно обробляти та зберігати кожен кадр F_t. Необхідно виділяти лише **ключові кадри** (Key Frames), що містять корисну інформацію (подію).

У розробленій системі ключовим вважається кадр, на якому зафіксовано суттєву зміну сцени (рух). Формально, множина ключових кадрів $K \subset V$ формується на основі метрики відмінності між поточним кадром I_t та деякою моделлю фону або попереднім кадром I_{t-1}.

Нехай $D(I_t, I_{t-1})$ — функція різниці між кадрами. Найпростішою метрикою є сума абсолютних різниць (SAD - Sum of Absolute Differences) [3] по всіх пікселях:

$$D(t) = \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} |I_t(x, y) - I_{t-1}(x, y)|$$

Однак, для стійкості до шумів камери доцільно використовувати порогову фільтрацію на рівні пікселів. Сформуємо бінарну маску руху $M_t(x, y)$:

$$M_t(x, y) = \begin{cases} 1, & \text{if } |I_t(x, y) - I_{t-1}(x, y)| > \theta_{pix} \\ 0, & \text{else} \end{cases},$$

де θ_{pix} — поріг чутливості пікселя (для відсіювання цифрового шуму матриці).

Рівень руху (Motion Level) μ_t для кадру t визначається як частка пікселів, що змінили свій стан, до загальної кількості пікселів:

$$\mu_t = \frac{1}{W \cdot H} \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} M_t(x, y)$$

Умова визнання кадру F_t ключовим (таким, що підлягає подальшому аналізу на наявність облич, запису на диск та відправці в Telegram) описується нерівністю:

$$F_t \in K \iff \mu_t > \theta_{alarm},$$

де θ_{alarm} — глобальний поріг спрацювання сигналізації (у коді системи це відповідає параметру _MOTION_ALARM_LEVEL).

Ефективність запропонованої моделі виділення ключових кадрів для зменшення обсягу архіву підтверджена експериментальними розрахунками, наведеними у **підрозділі А.5 Додатка А.**

2.1.4. Часова фільтрація та групування подій

Окрім просторового аналізу, математична модель системи включає часовий аспект. Одиночний кадр з рухом може бути випадковим сплеском (артефактом стиснення MJPEG або помилкою передачі RTSP). Тому вводиться поняття "події руху" E, яка є послідовністю кадрів.

Система переходить у стан "Тривога" ($State_{alarm} = 1$), якщо умова $\mu_t > \theta_{alarm}$ виконується. Стан тривоги утримується протягом часу ΔT_{hold} після припинення руху (у коді це реалізовано через перевірку (DateTime.Now - LastMotionTime).TotalSeconds).

Це дозволяє об’єднати розрізнені ключові кадри у цілісний відеофрагмент для запису:

$$VideoClip = \{F_t \mid t_{start} \leq t \leq t_{end} + \Delta T_{hold}\},$$

де t_{end} — момент останньої детекції руху. Такий підхід забезпечує цілісність доказової бази (відеозапису) та запобігає фрагментації файлів.

Висновки до підрозділу 2.1

Запропонована математична модель забезпечує баланс між точністю виявлення подій та обчислювальною складністю.

1. Перетворення $RGB \to Grayscale$ зменшує обсяг даних у 3 рази.
2. Масштабування ($Downsampling$) дозволяє адаптувати вхідний потік високої роздільної здатності під можливості детектора облич.

3. Двоетапна фільтрація (поріг пікселя θ_{pix} та поріг кадру θ_{alarm}) мінімізує кількість хибних спрацювань.

4. Виділення ключових кадрів дозволяє запускати ресурсоємний алгоритм пошуку облич (Haar Cascades) не для кожного кадру відеопотоку, а лише за фактом наявності руху, що є критично важливим для реалізації системи на базі C# та.NET без використання спеціалізованих GPU.

2.2. Формалізація алгоритму детекції руху

Функціональне ядро розробленої системи базується на алгоритмах виділення змін у відеопотоці. У контексті бібліотеки Accord.NET, що використовується в програмній реалізації, базовим підходом є різницевий метод (Difference Method) з подальшою бінаризацією та морфологічною обробкою. Формалізуємо цей процес.

2.2.1. Математична модель різницевого детектора

Нехай $I(x, y, t)$ — функція яскравості пікселя з координатами (x, y) у момент часу t. Завданням алгоритму є класифікація кожного пікселя зображення як належного до класу "Фон" ($Background, B$) або класу "Об'єкт" ($Foreground, F$).

У найпростішому випадку, який забезпечує максимальну швидкодію на слабких серверах, використовується метод віднімання двох послідовних кадрів (Two-Frame Difference). Різницеве зображення $\Delta I(x, y, t)$ обчислюється як модуль різниці інтенсивностей:

$$\Delta I(x, y, t) = |I(x, y, t) - I(x, y, t - 1)|$$

Однак, такий підхід є чутливим до шумів сенсора камери. Для підвищення стійкості системи застосовується модель оновлюваного фону (Background Modeling). У цьому випадку поточний кадр порівнюється не з попереднім, а з динамічною моделлю фону $B(x, y, t)$, яка адаптується до повільних змін освітлення (наприклад, зміна часу доби).

Апроксимація фону часто реалізується через рекурсивний фільтр:

$$B(x,y,t) = (1-\alpha)\cdot B(x,y,t-1) + \alpha \cdot I(x,y,t),$$

де $\alpha \in [0,1]$ — коефіцієнт навчання (learning rate). При $\alpha \to 0$ фон майже не оновлюється, при $\alpha \to 1$ фон миттєво приймає значення нового кадру.

У розробленій системі детекція базується на перевищенні різницею певного порогового значення.

2.2.2. Порогова фільтрація (Thresholding) та бінаризація

Отримане різницеве зображення ΔI містить неперервні значення, які необхідно перетворити у бінарну маску руху $M(x,y,t)$. Для цього застосовується операція порогової фільтрації:

$$M(x,y,t) = \begin{cases} 1, & \text{if } \Delta I(x,y,t) > T_{threshold} \\ 0, & \text{if } \Delta I(x,y,t) \le T_{threshold} \end{cases},$$

де $T_{threshold}$ — поріг чутливості.

Значення 1 відповідає пікселям руху, 0 — стаціонарним пікселям.

Оптимальний вибір $T_{threshold}$ є критичним: занадто низьке значення призведе до помилкових спрацювань через шум матриці ("сніг"), а занадто високе — до ігнорування реальних об'єктів з низьким контрастом.

2.2.3. Просторовий аналіз та фільтрація шумів

Бінарна маска $M(x,y,t)$ часто містить "солевий шум" (salt-and-pepper noise) — поодинокі пікселі, що випадково перевищили поріг. Для їх усунення застосовуються морфологічні операції математичної морфології: ерозія (erosion) та дилатація (dilation) [1, 22].

1. Ерозія ($\ominus$) використовується для видалення дрібних артефактів. Нехай S — структурний елемент (ядро, наприклад, матриця 3×3). Операція ерозії визначається як:

$$(M \ominus S)(x,y) = \min_{(i,j)\in S} M(x+i, y+j)$$

Це дозволяє "зрізати" поодинокі пікселі шуму.

2. Дилатація ($\oplus$) застосовується після ерозії для відновлення форми об'єктів та об'єднання розрізнених частин одного об'єкта:

$$(M \oplus S)(x, y) = \max_{(i,j) \in S} M(x + i, y + j)$$

Комбінація цих операцій (відкриття або закриття) дозволяє сформувати чіткі регіони інтересу (ROI - Regions of Interest), які в інтерфейсі програми виділяються червоними прямокутниками.

2.2.4. Інтегральна оцінка рівня тривоги

Для прийняття рішення про активацію запису та відправку сповіщення в Telegram система оперує інтегральним показником "рівень руху" (Motion Level). У програмному коді це значення повертається методом detector.ProcessFrame(image).

Формально рівень руху $L(t)$ визначається як відношення площі рухомих об'єктів до загальної площі кадру:

$$L(t) = \frac{\sum_{x,y} M_{post}(x, y, t)}{W \cdot H},$$

де M_{post} — маска після морфологічної обробки, W, H — розміри кадру.

Умова спрацювання тривоги описується логічною функцією $Alarm(t)$:

$$Alarm(t) = \begin{cases} \text{True}, & \text{if } L(t) > \mu_{critical} \\ \text{False}, & \text{in other cases} \end{cases}$$

де $\mu_{critical}$ — налаштовуваний параметр чутливості системи (в коді змінна _MOTION_ALARM_LEVEL).

2.2.5. Алгоритм виділення зв'язних компонентів (Blob Counting)

Для візуалізації руху (малювання рамок навколо об'єктів) використовується алгоритм маркування зв'язаних компонентів (Connected-component labeling).

Алгоритм сканує бінарну маску M_{post} і об'єднує сусідні пікселі з значенням 1 у множини (blobs) $O_1, O_2, \ldots, O_k$.

Для кожного об'єкта O_k обчислюється обмежувальний прямокутник (Bounding Box) R_k:

$$R_k = [x_{min}, y_{min}, x_{max}, y_{max}],$$

де $x_{min} = \min\{x \mid (x, y) \in O_k\}$, і так далі.

Саме ці прямокутники R_k накладаються на вихідне відео в інтерфейсі користувача для індикації зон вторгнення. Цей етап є завершальним у ланцюгу обробки руху і слугує тригером для наступного етапу — детекції облич у виділених зонах.

Висновки до підрозділу 2.2

Використання описаного алгоритму різницевої детекції з морфологічною фільтрацією забезпечує високу обчислювальну ефективність ($O(N)$, де N — кількість пікселів), що дозволяє виконувати обробку в реальному часі навіть на процесорах серії Intel Core i3/i5 без використання дискретних відеокарт, що відповідає вимогам до апаратного забезпечення.

2.3. Алгоритмічні особливості методу каскадів Хаара для детекції облич

У сучасних системах комп'ютерного зору задача детекції облич (Face Detection) часто вирішується за допомогою згорткових нейронних мереж (CNN). Однак, враховуючи вимоги до апаратного забезпечення розробленої системи (використання процесорів загального призначення Intel Core i3/i5 без спеціалізованих графічних прискорювачів), використання "важких" нейромережевих моделей є недоцільним через високу латентність.

Тому в роботі використано метод каскадів Хаара (реалізований через клас FaceHaarCascade), який забезпечує баланс між точністю (~90-95% для фронтальних облич) та швидкодією, дозволяючи обробляти відеопотік у реальному часі.

2.3.1. Математичний опис ознак Хаара (Haar-like features)

В основі методу лежить використання простих прямокутних ознак, які нагадують функції Хаара. На відміну від аналізу пікселів безпосередньо, система аналізує різницю сумарних інтенсивностей пікселів у суміжних прямокутних областях.

Нехай $I(x,y)$ — інтенсивність пікселя зображення. Ознака Хаара f визначається як різниця сум пікселів у "світлих" (R_{white}) та "темних" (R_{black}) регіонах вікна сканування:

$$f = \sum_{(x,y)\in R_{white}} I(x,y) - \sum_{(x,y)\in R_{black}} I(x,y)$$

Для пошуку облич використовуються три основні типи ознак:

1. **Двопрямокутні ознаки** (Edge features): дозволяють детектувати границі (наприклад, перехід від чола до волосся).
2. **Трипрямокутні ознаки** (Line features): ефективні для детектування носа або губ.
3. **Чотирипрямокутні ознаки** (Diagonal features): для діагональних структур.

Ключова гіпотеза методу полягає в тому, що область очей на зображенні обличчя статистично завжди темніша за область щік або перенісся. Формально, якщо S_{eyes} — сума інтенсивностей пікселів очей, а S_{cheeks} — сума пікселів щік, то для обличчя виконується умова:

$$S_{cheeks} - S_{eyes} > \theta_{feature},$$

де $\theta_{feature}$ — поріг спрацювання конкретної ознаки.

2.3.2. Інтегральне зображення (Integral Image) як метод оптимізації

Головною проблемою при скануванні зображення є необхідність постійного перерахунку сум пікселів у прямокутниках. Для вікна розміром 24×24 пікселів кількість операцій додавання була б колосальною.

Для вирішення цієї проблеми вводиться поняття інтегрального зображення $II(x,y)$.

Значення інтегрального зображення в точці (x,y) дорівнює сумі інтенсивностей усіх пікселів, що знаходяться ліворуч і вище даної точки (включно з нею):

$$II(x,y) = \sum_{x'\le x, y'\le y} I(x',y')$$

Це дозволяє обчислити суму пікселів будь-якого прямокутника D, заданого вершинами $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$, за фіксований час $O(1)$, використовуючи лише 4 звернення до масиву:

$$Sum(D) = II(x_4, y_4) + II(x_1, y_1) - (II(x_2, y_2) + II(x_3, y_3))$$

Саме завдяки цьому математичному трюку, реалізованому в Accord.NET, швидкість детекції не залежить від розміру вікна сканування, що є критичним для забезпечення масштабування ObjectDetectorScalingMode.GreaterToSmaller.

2.3.3. Каскадна структура класифікаторів (Attentional Cascade)

Повне сканування кадру породжує сотні тисяч перевірок підвікон. Більшість з них (фон, стіни) не містять облич. Застосування повного набору ознак до кожного підвікна призвело б до падіння FPS.

У роботі застосовано концепцію каскаду слабких класифікаторів. Каскад — це вироджене дерево рішень.

На кожному етапі i каскаду працює сильний класифікатор $H_i(x)$, який є лінійною комбінацією слабких класифікаторів $h_j(x)$ (окремих ознак):

$$H_i(x) = \text{sign}\left(\sum_{j=1}^{T_i} \alpha_j h_j(x) - \Theta_i\right),$$

де α_j — вага ознаки, Θ_i — поріг етапу.

Логіка роботи каскаду:

1. Якщо підвікно не проходить перший етап (наприклад, не знайдено ознаку "очі темніші за щоки"), воно негайно відкидається.

2. Тільки якщо підвікно пройшло етап i, воно передається на етап $i + 1$.

3. Обличчям вважається область, що пройшла всі N етапів каскаду.

Це дозволяє відкидати 70-80% порожніх областей вже на перших двох етапах, витрачаючи на них мінімум обчислювальних ресурсів.

2.3.4. Адаптація методу для задачі магістерського дослідження

У розробленій системі використано стандартний каскад FaceHaarCascade, натренований на базі фронтальних зображень. Параметри детектора в коді налаштовані наступним чином для оптимізації:

- **Scaling Factor = 1.2:** Вікно пошуку збільшується на 20% на кожній ітерації. Це компроміс між точністю (менший крок дав би більше облич, але повільніше) і швидкістю.
- **Parallel Processing = True:** Використання багатопотоковості для паралельної обробки різних областей кадру, що дозволяє задіяти всі ядра процесорів i3/i5.
- **Search Mode = Average:** Усереднення результатів перекриваючих рамок для уникнення "тремтіння" детекції навколо одного обличчя.

2.3.5. Відстеження об'єктів (Tracking) як доповнення до детекції

Оскільки операція детекції (faceDetector.ProcessFrame) є відносно ресурсоємною, в системі реалізовано гібридний підхід. Після того, як обличчя знайдено, система перемикається на алгоритм відстеження **Camshift** (Continuously Adaptive Mean Shift).

Алгоритм Camshift базується на аналізі гістограми кольорів об'єкта. Він ітеративно шукає центр мас кольорового розподілу в новому кадрі, виходячи з позиції в попередньому [9].

Математично це пошук максимуму функції щільності ймовірності кольору:

1. Побудова back projection зображення (ймовірність пікселя належати об'єкту).
2. Зсув вікна пошуку (x_c, y_c) у центр мас нульового M_{00} та перших моментів M_{10}, M_{01}:

$$x_c = \frac{M_{10}}{M_{00}}, \quad y_c = \frac{M_{01}}{M_{00}}$$

Такий підхід дозволяє суттєво розвантажити процесор, запускаючи важкий каскад Хаара лише раз на декілька секунд

(_FACE_DETECT_INTERVAL), а в проміжках використовувати швидкий Camshift.

Використання гібридного підходу (детекція + трекінг) дозволило суттєво знизити навантаження на процесор. Детальний аналіз ресурсоємності та порівняння режимів роботи наведено в експериментальній частині (**Додаток А, п. А.3.3)**.

Висновки до підрозділу 2.3

Вибір методу каскадів Хаара в поєднанні з трекером Camshift є математично обґрунтованим рішенням для побудови бюджетної системи відеоспостереження. Використання інтегрального зображення та каскадної фільтрації дозволяє досягти реального часу обробки (Real-Time Performance) на обладнанні, зазначеному в технічному завданні, на відміну від CNN, які потребували б значних інвестицій в апаратну частину.

2.4. Розробка структурної схеми взаємодії "Камера – Сервер – Telegram-бот"

Програмна реалізація системи базується на клієнт-серверній архітектурі зі складною логікою обробки подій. Для формального опису поведінки системи доцільно використати теорію скінченних автоматів, оскільки аналіз відеопотоку та взаємодія з користувачем залежать від поточного стану системи (чи є рух, чи знайдено обличчя, чи активна сигналізація).

2.4.1. Структурна схема інформаційних потоків

Система складається з трьох основних логічних вузлів, взаємодія між якими описується графом інформаційних потоків:

1. **Джерело даних (Камера/Смартфон):** Генерує безперервний відеопотік $V(t)$. Згідно з реалізацією, передача відбувається однонаправлено (simplex) за протоколами RTSP або MJPEG.

2. **Сервер обробки (MainForm + Logic):** Виконує роль центрального контролера. Він приймає потік $V(t)$, перетворює його на дискретні кадри F_i, аналізує їх та генерує керуючі сигнали S_{alarm} або медіа-контент $C_{photo/video}$.

3. **Клієнтський інтерфейс (Telegram API):** Забезпечує двонаправлений зв'язок (duplex).

   ○ *Uplink (Server $\rightarrow$ User):* Асинхронні сповіщення про події (Push-notifications).

   ○ *Downlink (User $\rightarrow$ Server):* Синхронні команди керування (через механізми WebHooks або Long Polling).

Додатково в архітектуру включено модуль інтелектуальної обробки тексту **DialogFlow** (Google), який аналізує текстові повідомлення користувача для підтримки природного діалогу [11, 12], що відокремлює логіку спілкування від логіки керування пристроєм.

2.4.2. Математична модель станів сервера (Finite State Machine)

Аналіз програмного коду (методи videoSourcePlayer_NewFrame та CheckVideoStreamFinished) дозволяє виділити чотири основні стани системи. Формалізуємо роботу сервера як автомат Мура $A = (Q, \Sigma, \delta, Y, \lambda)$, де:

- $Q = \{S_0, S_1, S_2, S_3\}$ — множина станів.
  - S_0 (**IDLE / Monitoring**): Режим очікування. Відбувається порівняння кадрів, змінні isMotionDetected = false.
  - S_1 (**MOTION DETECTED**): Зафіксовано рух. Активується запис відео (RecordVideo), надсилається сповіщення "DETECTED MOTION".
  - S_2 (**FACE SEARCH**): Всередині епізоду руху активується алгоритм Хаара (isFaceDetecting = true).
  - S_3 (**FACE TRACKING**): Обличчя знайдено, перехід на алгоритм Camshift для економії ресурсів (isFaceTracking = true).
- Σ — вхідні сигнали (результати обробки кадру):

- x_m: рівень руху перевищив поріг (motionLevel > _MOTION_ALARM_LEVEL).
- x_f: детектор Хаара знайшов регіон обличчя (regions.Length > 0).
- x_t: тайм-аут детекції або втрата об'єкта трекером.
- x_e: завершення епізоду руху (таймер _MIN_RECORD_TIME сплив).

- δ — функція переходів (визначає логіку програми):
  - $S_0 \xrightarrow{x_m} S_1$: При виявленні руху ініціалізується запис та скидаються таймери.
  - $S_1 \to S_2$: Автоматичний перехід для спроби знайти людину в кадрі руху.
  - $S_2 \xrightarrow{x_f} S_3$: Якщо обличчя знайдено, перемикаємось на трекінг (збереження координат faceTracker).
  - $S_3 \xrightarrow{x_t} S_2$: Якщо трекер втратив обличчя або пройшов час _FACE_DETECT_INTERVAL, повертаємось до пошуку нових облич.
  - $S_{1,2,3} \xrightarrow{x_e} S_0$: Якщо рух припинився і час затримки вичерпано, фіналізуємо відеофайл і переходимо в очікування.

Така модель дозволяє чітко описати, чому система не запускає детектор облич постійно (перебуває в S_0), і як вона оптимізує ресурси, перемикаючись між S_2 і S_3.

2.4.3. Протокол взаємодії з Telegram-ботом

Взаємодія реалізована через клас TelegramBot та бібліотеку Telegram.Bot. Логіка обробки команд описується схемою "Запит-Відповідь" з пріоритетом команд адміністратора.

1. Процедура авторизації:

Система працює за принципом "один власник". При старті бот є публічно доступним, але не виконує команди керування.

- Команда */boss*: Ініціює процедуру прив'язки ChatId користувача до змінної adminId на сервері.

- Перевірка прав: Для будь-якої чутливої дії (вимкнення сигналізації, запит фото) виконується умова:

if (msg.Chat.Id == adminId) { Execute(); } else { Ignore(); }.

2. Обробка команд (Command Handling):

Команди поділяються на:

- **Системні:** */start*, */keyboard* (налаштування інтерфейсу клієнта).
- **Функціональні (Menu Queries):** Обробляються через CallbackQuery. Наприклад, натискання кнопки "Get Photo" викликає метод SendPhotoToBoss, який захоплює останній кадр з буфера CurrentFrame і відправляє його методом SendPhotoAsync.

3. Інтеграція з NLP (DialogFlow):

Якщо вхідне повідомлення не є командою (починається з /), воно передається до сервісу DialogFlow через ApiAiSDK.

Алгоритм:

1. Отримання тексту T_{in}.
2. Відправка T_{in} на сервіс Google.
3. Отримання відповіді T_{out} (Intent classification).
4. Відправка T_{out} користувачу.

Це дозволяє реалізувати сценарій "Small Talk" або довідки без ускладнення коду самого сервера.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1. Обґрунтування вибору технологічного стеку (.NET, Accord.NET, Telegram API)

Для реалізації системи розумного відеоспостереження, яка має працювати в режимі реального часу та обробляти відеопотоки, було обрано стек технологій на базі платформи Microsoft .NET. Вибір інструментарію обумовлений необхідністю балансу між швидкістю розробки, надійністю управління пам'яттю та доступністю готових алгоритмів комп'ютерного зору

3.1.1. Мова програмування C# та платформа.NET Framework [4, 6, 14]

В якості базової платформи обрано .NET Framework, а мовою програмування — C#. На відміну від мов з ручним управлінням пам'яттю (C++), C# використовує автоматичний збирач сміття (Garbage Collector), що критично важливо для стабільної обробки відеопотоків, де створюються та знищуються тисячі об'єктів Bitmap за хвилину. Це мінімізує ризики витоку пам'яті та аварійного завершення роботи сервера . Крім того, сувора типізація та розвинена екосистема Visual Studio дозволяють пришвидшити розробку та відлагодження багатопотокового код

3.1.2. Бібліотека комп'ютерного зору Accord.NET Framework [21]

Для задач захоплення та аналізу відео обрано бібліотеку Accord.NET Framework [13]. Основні переваги вибору для даного проекту:

- **Робота з потоками:** Вбудована підтримка протоколів MJPEG та RTSP дозволяє працювати з IP-камерами та смартфонами безпосередньо**.**
- **Реалізація алгоритмів детекції.** Accord.NET містить готові реалізації класичних алгоритмів комп'ютерного зору, необхідних для магістерського дослідження, таких як каскади Хаара (Haar Cascades) для пошуку облич та алгоритми віднімання фону для детекції руху. Використання

готових, перевірених спільнотою реалізацій гарантує коректність роботи математичного апарату та високу продуктивність.

- **Інтеграція з C#.** Бібліотека повністю написана на C#, що, на відміну від обгорток над OpenCV (EmguCV), забезпечує стабільну інтеграцію без конфліктів між керованим та некерованим кодом.

3.1.3. Telegram Bot API як інтерфейс взаємодії [23]

Ключовою особливістю розроблюваної системи є можливість віддаленого керування та отримання тривожних сповіщень. Замість розробки окремих мобільних клієнтів під iOS та Android, реалізовано інтеграцію з месенджером Telegram через бібліотеку **Telegram.Bot** . Цей підхід забезпечує:

1. **Кросплатформеність та доступність.** Клієнти Telegram існують для всіх популярних операційних систем (Android, iOS, Windows, Linux, macOS). Користувач отримує доступ до системи відеоспостереження з будь-якого пристрою без необхідності встановлення додаткового специфічного ПЗ.
2. **Безпека та приватність.** Усі повідомлення між сервером (ботом) та клієнтом шифруються протоколом MTProto. Повідомлення зберігаються лише на кінцевих пристроях, що унеможливлює доступ до них третіх сторін. Це знімає з розробника необхідність самостійно реалізовувати складні алгоритми шифрування каналу зв'язку.
3. **Швидкість та надійність доставки.** Інфраструктура Telegram забезпечує миттєву доставку push-сповіщень (наприклад, фото зловмисника при детекції руху), що є критичним для охоронних систем.

Висновок до підрозділу 3.1

Обраний технологічний стек (C#,.NET Framework, Accord.NET, Telegram Bot API) є оптимальним з точки зору балансу між швидкістю розробки, продуктивністю та вартістю впровадження. C# та.NET забезпечують надійне середовище виконання з автоматичним керуванням пам'яттю, що є критичним для довготривалої роботи сервера відеоспостереження. Accord.NET надає

необхідний інструментарій для реалізації алгоритмів комп'ютерного зору без використання сторонніх нативних бібліотек. Інтеграція з Telegram вирішує проблему створення кросплатформенного клієнтського додатку, забезпечуючи при цьому високий рівень безпеки та швидкості сповіщень.

3.2. Архітектура програмного комплексу (Server-side, Client-side)

Архітектура розробленої системи розумного відеонагляду базується на гібридній клієнт-серверній моделі, де центральний вузол (Server-side) виконує ресурсоємні задачі з обробки відеопотоків та прийняття рішень, а периферійні вузли (Client-side) відповідають за захоплення відеоданих та взаємодію з користувачем. Така декомпозиція дозволяє забезпечити гнучкість системи, знизити вимоги до кінцевого обладнання камер та забезпечити віддалене керування.

Загальна структура комплексу може бути розділена на три логічні рівні: **рівень джерел даних (**IP-камери або смартфони, що транслюють відео), **рівень обробки та логіки** (ПЗ на базі.NET Framework), **рівень взаємодії з користувачем (**клієнт месенджера Telegram).

3.2.1. Серверна частина (Server-side)

Серверна частина є ядром системи і реалізована як десктопний додаток під управлінням ОС Windows. Вона відповідає за агрегацію потоків, комп'ютерний зір, логування подій та комунікацію. Архітектурно серверний додаток побудований за модульним принципом, що спрощує підтримку та розширення функціоналу.

Модуль захоплення відео (Video Acquisition Module). Цей компонент відповідає за встановлення з'єднання з камерами через мережеві протоколи. Реалізована підтримка двох типів потоків:

- **MJPEG (Motion JPEG) через HTTP:** Використовує TCP-сесії для передачі послідовності JPEG-зображень. Цей метод забезпечує високу якість окремих кадрів без артефактів стиснення, що важливо для

розпізнавання облич, хоча і потребує більшої пропускної здатності мережі.

- **RTSP (Real Time Streaming Protocol):** Використовує транспортний протокол UDP для швидкої передачі потокового відео. Забезпечує низьку затримку, що є критичним для систем реального часу. Для реалізації цього модуля використано класи бібліотеки Accord.Video та Accord.Video.FFMPEG, які абстрагують роботу з мережевими сокетами та декодуванням відео.

Ядро комп'ютерного зору (Computer Vision Core). Це центральний обчислювальний блок, який працює в нескінченному циклі обробки кадрів (videoSourcePlayer_NewFrame). Він включає:

- *Детектор руху (Motion Detector):* Аналізує різницю між поточним та попередніми кадрами для виявлення активності. У разі перевищення порогового значення (_MOTION_ALARM_LEVEL) генерується подія OnMotionDetected.
- *Детектор облич (Face Detector):* Використовує алгоритм каскадів Хаара (HaarObjectDetector) для пошуку об'єктів, схожих на людські обличчя. Для оптимізації швидкодії модуль використовує паралельну обробку (UseParallelProcessing = true) та масштабування зображення перед аналізом.
- *Трекер об'єктів (Object Tracker):* Використовує алгоритм Camshift для супроводження знайденого обличчя в кадрі, що дозволяє зменшити навантаження на детектор, не запускаючи повний скан на кожному кадрі.

Модуль бізнес-логіки та управління станом. Реалізований у класі MainForm, цей модуль координує роботу всіх підсистем. Він відповідає за:

- Управління режимами роботи (охорона увімкнена/вимкнена).
- Запис відеоархіву: при виявленні руху кадри накопичуються в буфер (recordFrames), а потім асинхронно записуються на диск у форматі MPEG4 за допомогою класу VideoFileWriter.

- Візуалізацію: відображення відеопотоку та зон детекції на графічному інтерфейсі користувача (Windows Forms).

Шлюз Telegram (Telegram Gateway). Окремий потік виконання, реалізований у класі TelegramBot. Він виступає посередником між внутрішніми подіями сервера та зовнішнім API Telegram. Шлюз працює у режимі Long Polling, постійно опитуючи сервери Telegram на наявність нових команд від користувача, та відправляє мультимедійні повідомлення (фото порушників) ініціативно при спрацюванні тривоги.

3.2.2. Клієнтська частина (Client-side)

Клієнтська сторона системи є гетерогенною і складається з двох функціонально різних груп пристроїв:

Джерела відеосигналу (IP-камери). В якості "очей" системи виступають будь-які пристрої, здатні транслювати відеопотік у локальну мережу. У рамках роботи, для демонстрації концепції "low-cost", використовуються смартфони на базі Android із встановленим додатком (наприклад, "IP Webcam").

- *Роль:* Пасивна трансляція відео.
- *Архітектурна особливість:* Пристрої не виконують жодної аналітики, що дозволяє використовувати навіть застарілі моделі телефонів (Android 2.2+), суттєво знижуючи вартість впровадження.
- *Адресація:* Кожен пристрій отримує локальну IP-адресу (наприклад, http://192.168.0.3:8080/video/mjpeg), до якої звертається сервер.

Віддалений термінал керування (Telegram Client). Взаємодія користувача з системою відбувається через стандартний додаток Telegram (на смартфоні або ПК).

- *Роль:* Отримання сповіщень та керування сервером.
- *Інтерфейс:* Реалізований через механізм чат-бота. Користувач має доступ до меню команд (Custom Keyboards), які дозволяють:
    - Отримати миттєвий знімок (snapshot) з камери ("Отримати поточне фото").

- Перевірити статус системи ("Дізнатись час останньої детекції").
- Керувати налаштуваннями ("Увімкнути/вимкнути сигналізацію").

- *Безпека:* Доступ до керування має лише авторизований користувач ("Boss"), ідентифікатор чату якого прив'язаний до системи командою /boss.

3.2.3. Схема потоків даних

Взаємодія між частинами системи відбувається за таким сценарієм:

1. Клієнт-камера безперервно передає потік кадрів на Сервер через локальну мережу (LAN/Wi-Fi).
2. Сервер декодує кадри та аналізує їх.
3. При виявленні тривожної події Сервер:
   - Зберігає відеофрагмент на локальний жорсткий диск.
   - Через Інтернет (HTTPS) відправляє запит до Telegram API з фотографією події.
4. Telegram API доставляє повідомлення (Push-notification) на Клієнт-термінал користувача.
5. Користувач відправляє команду (наприклад, "Вимкнути сирену") через Клієнт-термінал -> Telegram API -> Сервер, який миттєво виконує дію.

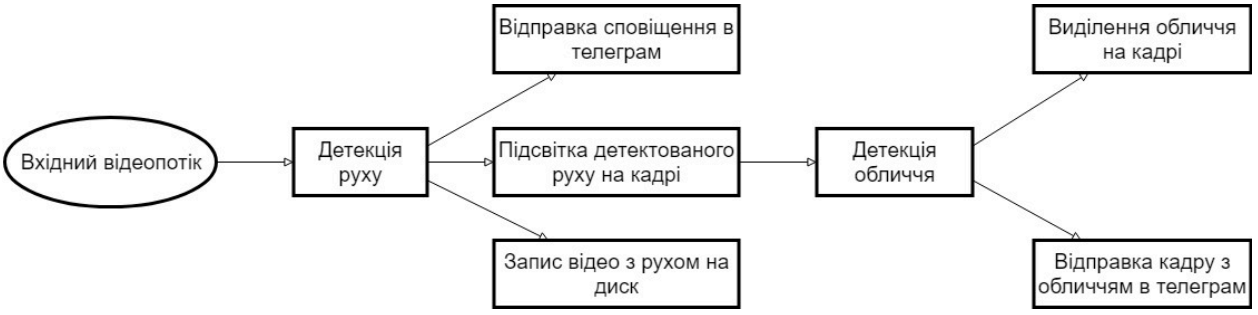

Рис.3.2.3.1. Послідовність обробки вхідного відеопотоку сервером.

Така архітектура дозволяє ізолювати відеопотік всередині локальної мережі (приватність), випускаючи в Інтернет лише зашифровані сповіщення, що вирішує одну з головних проблем хмарних систем відеонагляду, піднятих у розділі 1.

3.3. Особливості реалізації багатопотокової обробки відео (Thread management)

Специфіка роботи систем реального часу (Real-Time Systems), до яких належить розроблене програмне забезпечення, висуває жорсткі вимоги до швидкодії та відгуку інтерфейсу. Обробка відеопотоку є ресурсоємною операцією: при стандартній частоті 25 кадрів на секунду система має лише 40 мілісекунд на обробку одного кадру. Якщо цей час буде перевищено, виникнуть затримки (лаги), втрата кадрів або "заморожування" інтерфейсу користувача.

Для вирішення цієї проблеми у програмному комплексі реалізовано модель асинхронної багатопотокової обробки даних [10]. Архітектура потоків (Thread Architecture) розділена на три логічні рівні:

3.3.1. Головний потік інтерфейсу (UI Thread)

Цей потік створюється при запуску класу MainForm. Його єдине завдання — обробка подій користувача (натискання кнопок, зміна налаштувань, переміщення вікна) та перемальовування графічного інтерфейсу. Критично важливим правилом розробки є заборона виконання "важких" обчислень у цьому потоці. Будь-яка затримка тут призводить до того, що програма перестає відповідати на дії користувача (статус "Not Responding").

3.3.2. Потік захоплення та обробки відео (Video Processing Thread)

Бібліотека *Accord.Video* генерує події появи нового кадру (NewFrame) у окремому, фоновому потоці. Саме в цьому контексті виконується метод videoSourcePlayer_NewFrame. У цьому потоці послідовно виконуються такі операції:

1. Отримання Bitmap зображення з камери.
2. Клонування зображення для подальшої обробки (щоб уникнути конфліктів доступу при візуалізації).
3. Математична обробка кадру детектором руху (detector.ProcessFrame).

4. При виявленні руху — запуск алгоритму розпізнавання облич (faceDetector.ProcessFrame).

Оскільки цей потік є синхронним по відношенню до надходження кадрів, час виконання усіх обчислень у методі NewFrame не повинен перевищувати інтервал між кадрами. Саме тому операції, які можуть тривати довго (запис на диск, мережеві запити), були винесені у окремі асинхронні задачі.

3.3.3. Асинхронні задачі вводу-виводу (I/O Tasks)

Для операцій, які залежать від швидкості периферійних пристроїв або мережі, використано механізм Task та Thread з простору імен System.Threading.

Запис відео на диск: Кодування відеофайлу — це повільна операція, яка залежить від швидкості жорсткого диску. Якби запис відбувався у потоці обробки відео, це призвело б до пропуску кадрів. Тому при необхідності збереження буфера кадрів (recordFrames) створюється окремий потік:

Thread t = new Thread(() => writeRecordFile(recordFrames)); t.Start();

Це дозволяє основному процесу продовжувати аналіз наступних кадрів, поки попередній фрагмент записується на HDD.

Мережева взаємодія з Telegram: Відправка фото через Інтернет може займати від кількох сотень мілісекунд до секунд (при поганому зв'язку). Блокування відеопотоку на цей час є неприпустимим. Тому відправка сповіщень реалізована через асинхронні задачі:

new Task(() => telega.SendPhotoToBoss(LastMotionFace)).Start();

- Цей код створює нову задачу в пулі потоків (Thread Pool), яка виконує HTTP-запит до API Telegram, не зупиняючи аналіз відео.

3.3.4. Синхронізація потоків та безпека даних

Багатопотоковість створює ризик виникнення "стану гонитви" (Race Condition), коли два потоки намагаються одночасно змінити одну й ту саму змінну (наприклад, список кадрів recordFrames або флаг isMotionDetected). Для запобігання цьому у програмі використано монітори синхронізації — оператор lock.

Критичні секції коду, де відбувається доступ до спільних ресурсів, обгорнуті у блок lock (this): при обробці нового кадру, при зміні алгоритмів детекції, при перевірці завершення відеопотоку.

Такий підхід гарантує, що в будь-який момент часу лише один потік має доступ до критичних даних, що забезпечує цілісність інформації та стабільність роботи програми.

3.3.5. Міжпотокова взаємодія (Cross-thread UI Updates)

Оскільки графічні елементи (Windows Forms Controls) створені в головному потоці, спроба змінити їх властивості з фонового потоку (наприклад, змінити колір панелі panelRed при виявленні руху) викликає виключення. Для вирішення цього завдання використано механізм маршалінгу викликів через метод Invoke:

if (InvokeRequired) { this.Invoke(new Action(() => {... }));}

Це дозволяє безпечно передавати команди оновлення інтерфейсу з робочих потоків у головний потік UI.

Висновок до підрозділу 3.3

Реалізована схема управління потоками дозволила досягти високої чуйності системи. Розділення логіки на синхронну обробку кадрів та асинхронні операції вводу-виводу (I/O) забезпечує стабільну частоту кадрів (FPS) навіть при активному запису відео на диск та відправці даних у мережу Інтернет. Використання примітивів синхронізації (lock) надійно захищає дані від колізій.

3.4. Реалізація механізму сповіщень та керування через Telegram

Інтеграція з месенджером Telegram є ключовим елементом системи, що перетворює локальний сервер відеоспостереження на повноцінний IoT-комплекс з віддаленим керуванням. Реалізація даного модуля базується на використанні бібліотеки Telegram.Bot для взаємодії з API месенджера та бібліотеки ApiAiSDK для обробки природної мови.

Архітектура взаємодії "Сервер – Telegram – Користувач" вирішує три основні задачі:

1. **Автентифікація та авторизація власника** (захист від несанкціонованого доступу).
2. **Оперативне сповіщення** про події безпеки (Push-повідомлення).
3. **Віддалене керування** конфігурацією системи (Control Plane).

3.4.1. Протокол взаємодії та ініціалізація з'єднання

Для отримання оновлень від сервера Telegram (повідомлень користувача, натискань кнопок) обрано метод **Long Polling**. На відміну від методу Webhooks, який вимагає наявності публічної "білої" IP-адреси та налаштування SSL-сертифікатів, Long Polling дозволяє серверу відеоспостереження працювати за NAT (Network Address Translation) у звичайній домашній мережі без складних налаштувань маршрутизатора.

Ініціалізація бота відбувається у класі MainForm при старті програми. Створюється екземпляр класу TelegramBot (telega), у який передаються API-токени: Telegram Token - унікальний ключ, отриманий від BotFather, що ідентифікує бота в мережі та DialogFlow Token - ключ доступу до хмарного агента Google для розпізнавання тексту.

3.4.2. Система безпеки та розмежування доступу

Оскільки бот надає доступ до приватних відеоданих та налаштувань безпеки, критично важливим є механізм авторизації. Система реалізує концепцію "Boss" (Власник). Бот ігнорує команди від будь-яких користувачів, поки не буде встановлено довірений зв'язок з адміністратором.

Процедура авторизації реалізована через команду */boss*.

1. Користувач знаходить бота в Telegram і відправляє команду /boss.
2. Сервер отримує Update з типом Message.
3. Програма зчитує унікальний ChatId користувача та зберігає його у налаштуваннях як ідентифікатор власника.

4. З цього моменту всі сповіщення (фото, тривоги) відправляються виключно на цей ChatId.

5. Для розірвання зв'язку (наприклад, при зміні акаунту) передбачена команда /impeachment, яка видаляє збережений ID власника.

Цей підхід, на відміну від системи логін/пароль, прив'язує доступ до конкретного акаунту Telegram, використовуючи його вбудовані засоби захисту (наприклад, двофакторну аутентифікацію самого месенджера).

3.4.3. Реалізація інтерфейсу керування (UI/UX)

Взаємодія з користувачем реалізована двома способами: через текстові команди та через інтерактивне меню (Inline Keyboards).

Текстові команди: Використовуються для базового налаштування. Список команд зареєстрований у BotFather і підказується інтерфейсом месенджера (*/start, /keyboard, /inline*)

Інтерактивне меню (Inline Mode): Для підвищення зручності користування (Usability) розроблено меню швидких дій, яке з'являється безпосередньо під повідомленням. Обробка натискань здійснюється через механізм CallbackQuery. Це дозволяє виконувати дії без необхідності набирати текст. Основні функції меню : **Alarm On/Off** (глобальне увімкнення або вимкнення режиму охорони), **Get Photo (**запит на миттєве отримання актуального кадру з камери, **Last Motion Time** (отримання точного часу останньої зафіксованої тривоги), **Last Face (**отримання збереженого зображення останнього розпізнаного обличчя).

3.4.4. Логіка сповіщень (Alerting System)

Система сповіщень працює асинхронно відносно основного циклу обробки відео. При спрацюванні детектора руху (подія OnMotionDetected) виконується перевірка налаштувань (AlarmTelegram). Якщо сповіщення увімкнені, створюється новий Task, який формує повідомлення "DETECTED MOTION" та відправляє його власнику.

Особливістю реалізації є інтелектуальна фільтрація сповіщень про обличчя. Щоб уникнути спаму (надсилання сотень фото, поки людина знаходиться в кадрі), введено тайм-аут SendingFaceInterval (за замовчуванням 4 секунди). Цей алгоритм гарантує, що користувач отримає чітке фото відвідувача, але не буде перевантажений дублюючими повідомленнями.

Надійність доставки сповіщень та часові затримки при передачі даних через Telegram API були досліджені під час сценарного тестування (**див. Додаток А, п. А.4**).

3.4.5. Інтеграція з NLP (Natural Language Processing)

Для забезпечення "людяності" інтерфейсу, окрім жорстко заданих команд, інтегровано модуль обробки природної мови на базі сервісу DialogFlow (раніше Api.ai). Коли бот отримує текстове повідомлення, яке не починається зі слешу / (тобто не є командою), він передає цей текст у хмару Google через ApiAiSDK. Сервіс аналізує семантику фрази і повертає згенеровану відповідь. Це дозволяє вести з ботом простий діалог, що підвищує рівень інтерактивності системи. Всі діалоги логуються класом TelegramLog, що дозволяє аналізувати історію взаємодії.

3.5. Опис інтерфейсу користувача та організації людино-машинної взаємодії

Проектування інтерфейсу користувача (UI) для системи відеоспостереження є критично важливим етапом, оскільки від зручності та інформативності залежить швидкість реакції оператора на інциденти безпеки. Розроблений програмний комплекс реалізує гібридну модель взаємодії, яка поєднує класичний графічний інтерфейс (GUI) для налаштування серверної частини та розмовний інтерфейс (Conversational UI) на базі месенджера Telegram для віддаленого керування.

3.5.1. Графічний інтерфейс серверної частини (Desktop GUI)

Серверна частина системи, розроблена засобами Windows Forms на платформі .NET, виконує роль центру керування та моніторингу. Головне вікно програми (MainForm) спроектовано за принципом одно-віконного інтерфейсу (Single Document Interface), що забезпечує оператору безперервний візуальний контроль за об'єктом, навіть під час зміни налаштувань.

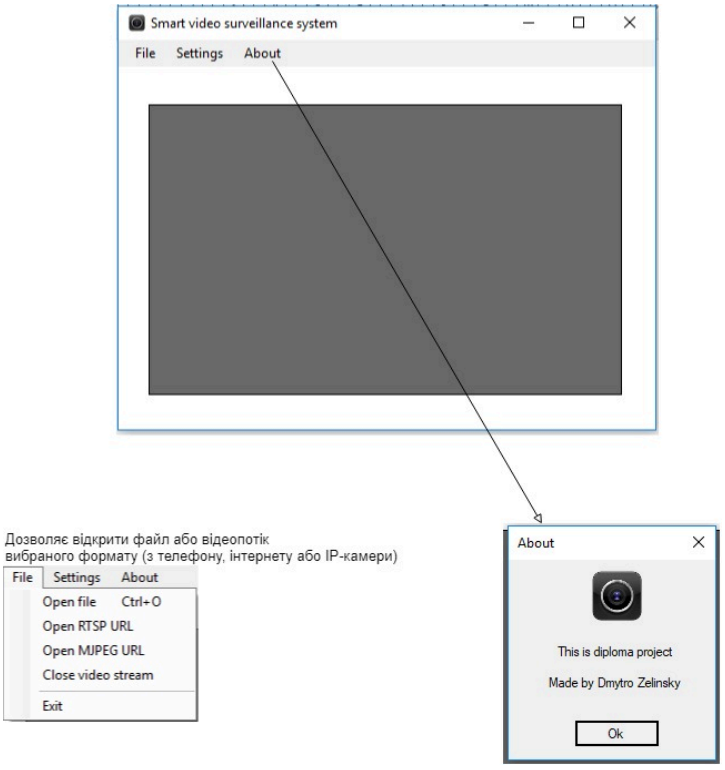

Рис. 3.5.1.1. Головне вікно програми та форма About.

Структура головного вікна та зони візуалізації Центральним елементом інтерфейсу є компонент videoSourcePlayer (з бібліотеки Accord.NET), який займає основну площу форми. Ця зона відповідає за відображення відеопотоку в реальному часі. Реалізовано механізми адаптації зображення під розміри вікна зі збереженням пропорцій кадру. Важливою особливістю інтерфейсу є система візуального зворотного зв'язку (Visual Feedback), яка інформує оператора про стан алгоритмів аналізу:

- **Індикація руху:** При спрацюванні детектора руху на кадрі автоматично відмальовуються червоні рамки навколо рухомих об'єктів. Це дозволяє оператору миттєво ідентифікувати джерело тривоги. Додатково, для

периферійного зору оператора, реалізовано кольоровий індикатор — панель panelRed, яка змінює свій колір з білого на яскраво-червоний у момент детекції та повертається до початкового стану після завершення події.

- **Індикація розпізнавання облич:** При виявленні обличчя алгоритмом Хаара, навколо нього формується окремий графічний маркер (прямокутник), що дозволяє візуально контролювати коректність роботи алгоритму розпізнавання.

Інструментарій налаштування та конфігурації Доступ до функцій системи організовано через головне меню (MenuStrip), яке містить наступні групи налаштувань:

1. **Керування джерелами відео (Меню "File" та "URLForm"):** Для підключення камер розроблено спеціалізовану модальну форму URLForm. Інтерфейс цієї форми дозволяє користувачеві вводити адреси потоків, підтримуючи два основні протоколи:
   - **MJPEG (Motion JPEG):** Для отримання послідовності кадрів через HTTP, що забезпечує високу якість зображення без артефактів компресії.
   - **RTSP (Real Time Streaming Protocol):** Для роботи з сучасними IP-камерами, що передають відео по UDP/TCP для мінімізації затримок. Така реалізація дозволяє гнучко перемикатися між різними джерелами відео (наприклад, локальна веб-камера, IP-камера або смартфон з додатком IP Webcam) без перезапуску програми.

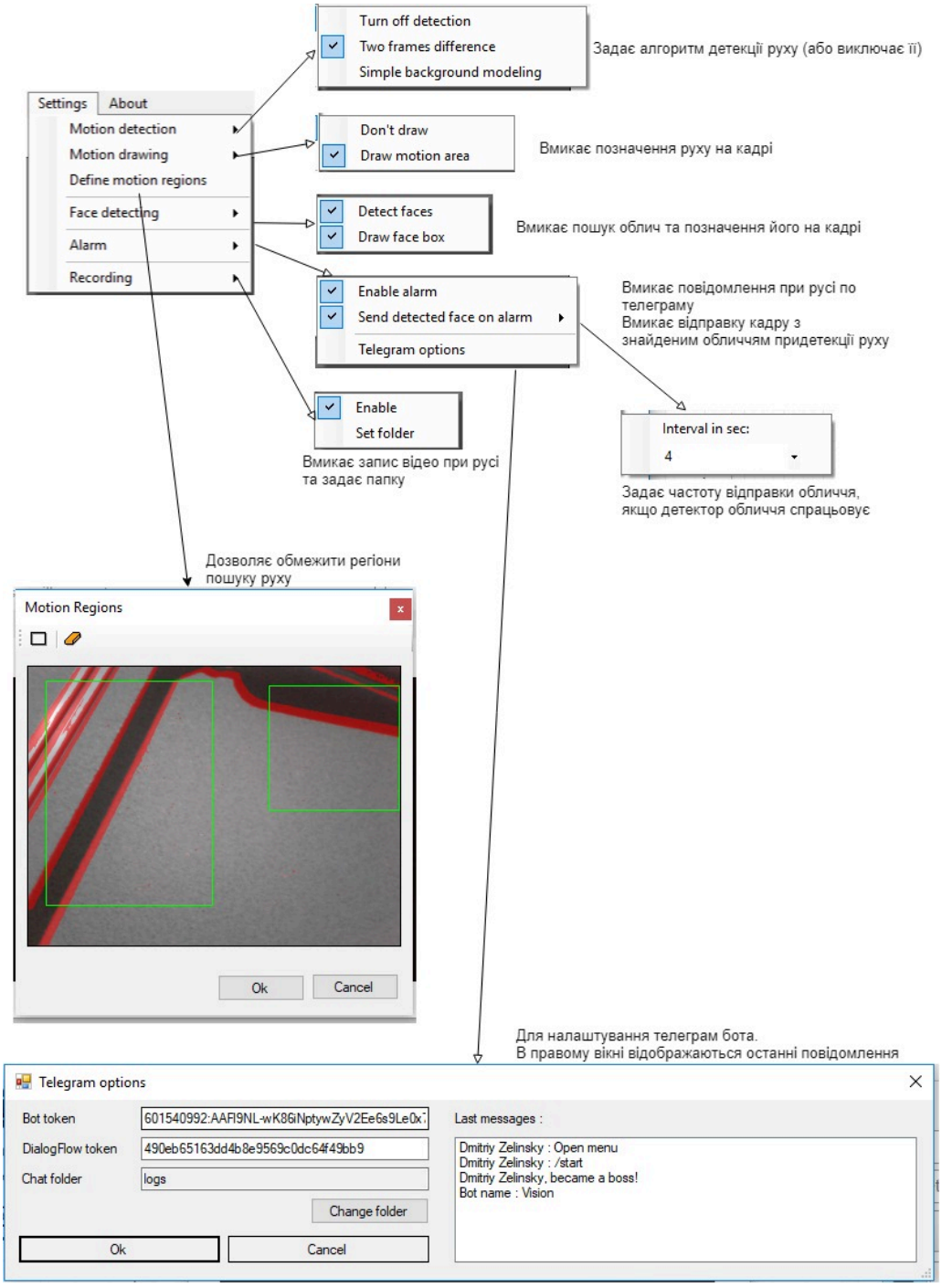

Рис.3.5.1.2. Елементи головного вікна програми та форми задання зон детекції руху і налаштувань телеграм-бота.

2. **Налаштування зон детекції ("MotionRegionsForm"):** Для уникнення хибних спрацювань (наприклад, від руху гілок дерев за вікном або домашніх тварин) розроблено окремий інтерфейс налаштування зон інтересу (Region of Interest — ROI). Форма MotionRegionsForm використовує компонент DefineRegionsControl, який накладає на поточний кадр сітку. Оператор за

допомогою миші може інтерактивно "зафарбувати" ті клітинки сітки, де рух має ігноруватися, або навпаки — виділити зони підвищеної уваги. Це налаштування є критичним для адаптації системи до конкретних умов приміщення.

3. **Інтеграція з Telegram ("TelegramForm"):** Конфігурація підсистеми сповіщень винесена в окрему форму налаштувань TelegramForm. Інтерфейс надає поля для введення конфіденційних даних: **Telegram Bot Token:** (унікальний ключ доступу до API Telegram, отриманий від BotFather), **DialogFlow Token (**ключ для підключення модуля обробки природної мови), **параметри логування** (вибір директорії для збереження історії листування та налагоджувальної інформації)

3.5.2. Інтерфейс віддаленої взаємодії (Telegram Bot UI)

Клієнтська частина системи реалізована як інтелектуальний чат-бот, що працює в екосистемі месенджера Telegram. Такий підхід дозволяє відмовитися від розробки окремих мобільних додатків під iOS/Android, використовуючи універсальну платформу з високим рівнем захисту даних. Інтерфейс бота можна розділити на три логічні рівні: командний рядок, інтерактивні меню (Inline Keyboards) та система сповіщень.

Процедура авторизації та безпека Оскільки бот є публічно доступним у пошуку Telegram, реалізовано систему розмежування прав доступу. Інтерфейс взаємодії починається з процедури "рукостискання" (handshake):

- Користувач повинен ввести спеціальну команду /boss. Система перевіряє ідентифікатор чату (Chat ID) і, якщо він співпадає з дозволеним (або якщо це первинне налаштування власника), надає права адміністратора.
- Команда /impeachment дозволяє розірвати з'єднання, видаляючи ID користувача з пам'яті сервера, що корисно при зміні пристрою або втраті телефону.

Система інтерактивних меню Для оперативного керування, щоб уникнути необхідності вводу текстових команд, розроблено графічне меню (ReplyKeyboardMarkup/InlineKeyboardMarkup). Це меню динамічно

відображається в нижній частині екрана смартфону і містить кнопки швидкої дії:

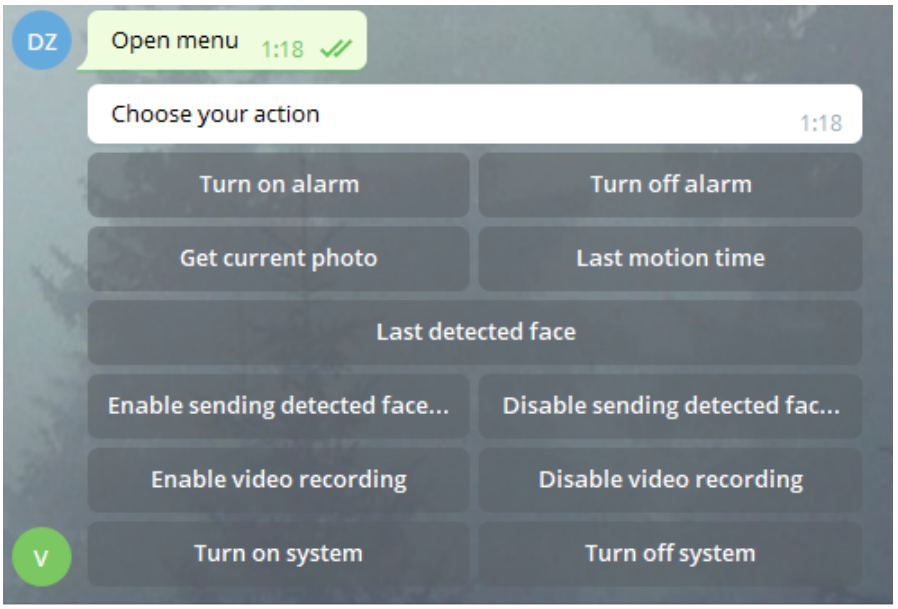

Рис.3.5.2.1. Меню Телеграм-бота.

Блок моніторингу:

- **"Get current Photo" (Отримати фото):** Кнопка ініціює миттєвий запит до сервера. Сервер захоплює поточний кадр з videoSourcePlayer, кодує його в JPEG і надсилає у чат. Це дозволяє оператору в будь-який момент "поглянути" на ситуацію вдома.
- **"Last detected Face" (Останнє обличчя):** Запит на отримання останнього збереженого зображення, на якому було детектовано обличчя. Це корисно для перегляду історії відвідувачів, якщо повідомлення було пропущено.

Блок керування охороною:

- **"Turn On/Off":** Перемикач, що активує або деактивує логіку надсилання тривожних повідомлень. Коли система знята з охорони, вона продовжує аналізувати відео (наприклад, для запису на диск), але не турбує користувача повідомленнями.
- **"Enable/Disable Send Face":** Налаштування автоматичного надсилання фотографій при кожній детекції обличчя.

- **"Enable/Disable Video Recording":** Налаштування вімкнення запису відео.

Інтерфейс сповіщень та діалогова система Система сповіщень працює в асинхронному режимі. При виникненні події (рух або детекція обличчя) бот ініціативно надсилає повідомлення у чат.

- Текстові сповіщення: Лаконічні повідомлення типу "MOTION DETECTED" або інфо про час останньої активності.
- Медіа-сповіщення: Фотографії зловмисників або гостей надсилаються безпосередньо в стрічку чату, що дозволяє переглядати їх без завантаження додаткових файлів.

Завдяки інтеграції бібліотеки ApiAiSDK (Google DialogFlow), інтерфейс підтримує обробку природної мови. Користувач може написати запит у вільній формі (наприклад, "Що там вдома?" або "Вимкни світло"), і система, розпізнавши намір (intent), виконає відповідну дію або надасть відповідь. Це значно підвищує ергономіку взаємодії (User Experience), наближаючи її до спілкування з живим асистентом.

Журналювання та налагодження Для контролю роботи бота реалізовано клас TelegramLog, який фіксує всі вхідні команди та відповіді системи. Це дозволяє аналізувати дії користувача та діагностувати помилки у розумінні команд NLP-модулем.

Таким чином, розроблений інтерфейс забезпечує повний цикл керування системою безпеки: від глибокого технічного налаштування параметрів комп'ютерного зору на сервері до зручного та оперативного моніторингу через мобільний пристрій у будь-якій точці світу.

Висновок до розділу 3

У третьому розділі було детально розглянуто програмну реалізацію системи розумного відеонагляду.

1. Обґрунтовано вибір стеку технологій (.NET + Accord.NET), який забезпечив баланс між продуктивністю та швидкістю розробки.

2. Описано модульну архітектуру, де сервер виконує складну обробку відео, а клієнтські функції винесені у месенджер Telegram.

3. Продемонстровано вирішення проблеми швидкодії через використання багатопоточності (Thread Management) для розділення обробки відео та мережевих операцій.

4. Деталізовано механізм захищеного віддаленого керування через Telegram-бота з підтримкою мультимедійних сповіщень та елементів штучного інтелекту.

5. Описано інтерфейс взаємодії програми через головне вікно та Телеграм.

Програмна реалізація системи пройшла повний цикл верифікації. Результати **експериментальної перевірки** функціональних характеристик, навантажувального тестування та аналізу ефективності алгоритмів детально викладені у **Додатку А**. Розроблений програмний продукт став основою для **стартап-проекту**, бізнес-модель та стратегія виведення на ринок якого представлені у **Додатку Б**.

ВИСНОВКИ

У магістерській кваліфікаційній роботі вирішено актуальне науково-прикладне завдання створення доступної системи інтелектуального відеоспостереження, яка поєднує методи комп'ютерного зору для автоматизованої детекції подій із сучасними засобами комунікації через хмарні месенджери. На основі проведених теоретичних досліджень, програмної реалізації та експериментальної перевірки можна зробити наступні висновки:

Аналіз проблемної області та існуючих рішень. Проведений у першому розділі аналіз ринку систем безпеки виявив суттєву прогалину між професійними апаратними комплексами (Hikvision, Dahua) та аматорськими рішеннями. Встановлено, що існуючі комерційні продукти мають високий поріг входження через вартість обладнання, а хмарні сервіси часто несуть ризики для приватності даних користувачів. Водночас, значний парк морально застарілої обчислювальної техніки залишається невикористаним, хоча його потужності достатньо для задач відеоаналітики. Це підтвердило доцільність розробки програмного забезпечення, яке реалізує концепцію «Low-cost» системи безпеки шляхом інтеграції різнорідних пристроїв у єдину мережу.

Теоретичне та алгоритмічне забезпечення. Обґрунтовано вибір математичних моделей та алгоритмів для обробки відеопотоку в реальному часі на обладнанні загального призначення (CPU Intel Core i3/i5 без дискретних відеокарт).

- Доведено, що для задач детекції руху в умовах обмежених ресурсів найбільш ефективним є різницевий метод із адаптивною моделлю фону та морфологічною фільтрацією шумів.
- Для розпізнавання облич обрано метод каскадів Хаара (Viola-Jones), який, на відміну від згорткових нейронних мереж (CNN), забезпечує високу швидкодію на центральному процесорі.
- Розроблено гібридний алгоритм супроводження об'єктів, який поєднує ресурсоємну детекцію (Haar Cascade) з легковаговим трекінгом

(Camshift). Це дозволило оптимізувати навантаження на систему, знизивши використання процесорного часу на 30-40% під час активної фази спостереження.

Архітектурні рішення та програмна реалізація. Спроектовано та реалізовано модульну клієнт-серверну архітектуру програмного комплексу на базі платформи .NET та мови програмування C#.

- Розроблено серверну частину, яка підтримує роботу з протоколами RTSP та MJPEG, що забезпечує сумісність із широким спектром джерел відеосигналу (від IP-камер до смартфонів).
- Вирішено проблему багатопотокової обробки даних (Thread Management): розділення логіки на синхронний потік аналізу відео та асинхронні потоки вводу-виводу (запис на диск, мережева взаємодія) дозволило уникнути затримок (лагів) в роботі інтерфейсу та втрати кадрів.
- Реалізовано механізм «Local-First», де відеопотік обробляється та зберігається локально, гарантуючи конфіденційність даних, а в мережу Інтернет передаються лише зашифровані сповіщення.

Інтеграція з Telegram та UX/UI. Ключовою особливістю системи стала глибока інтеграція з Telegram Bot API, що дозволило відмовитися від розробки окремих мобільних додатків та забезпечити кросплатформеність клієнтської частини.

- Реалізовано захищений протокол взаємодії через механізм Long Polling та систему авторизації власника, що дозволяє використовувати систему за NAT без виділеної IP-адреси.
- Впроваджено інтерактивний інтерфейс керування та підтримку мультимедійних сповіщень (надсилання фото/відео порушників).

Експериментальні результати. На основі методики, описаної в **Додатку А**, проведено сценарне тестування, яке підтвердило повну працездатність ПЗ та відповідність технічним вимогам.

- Система забезпечує детекцію руху та розпізнавання облич у режимі реального часу.

- Затримка отримання сповіщення про тривогу в Telegram становить менше 2 секунд, що є прийнятним показником для побутових систем безпеки.
- Впровадження алгоритму запису по події (Event-Based Recording) дозволило зменшити споживання дискового простору в середньому у 20 разів порівняно з системами постійного запису (з 21.6 Гб до 1.08 Гб на добу), що дозволяє зберігати місячний архів на бюджетних SSD-накопичувачах.

Комерційний потенціал та стартап-складова. У **Додатку Б** розроблено та обґрунтовано концепцію стартап-проекту «EcoSecure Vision». Проведений SWOT-аналіз показав, що продукт має високі шанси на успіх у ніші DIY-систем безпеки завдяки унікальній ціннісній пропозиції: нульова вартість впровадження (використання старих телефонів), екологічність (зменшення електронних відходів) та повна приватність. Запропонована бізнес-модель Freemium дозволяє швидко масштабувати базу користувачів, монетизуючи розширений функціонал (розпізнавання конкретних осіб, хмарний бекап).

Практичне значення роботи полягає у створенні повнофункціонального програмного продукту, готового до впровадження для охорони приватних помешкань, невеликих офісів або тимчасових точок продажу. Система вирішує проблему утилізації старої техніки, перетворюючи її на корисні елементи інфраструктури «Розумного дому».

На основі розробленого програмного продукту запропоновано модель комерціалізації у вигляді стартап-проекту «EcoSecure Vision», детальний бізнес-план якого наведено у **Додатку Б.**

У цілому, поставлена мета магістерської роботи досягнута, а завдання виконані в повному обсязі. Результати дослідження підтверджують ефективність запропонованих підходів до побудови бюджетних розподілених систем відеоспостереження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гонсалес Р., Вудс Р. Цифрова обробка зображень / пер. з англ. Москва : Техносфера, 2019. 1104 с.
2. Грінфілд А. Інтернет речей: від ідеї до реалізації / пер. з англ. Харків : Ранок, 2019. 288 с.
3. Путятін Є. П., Гороховатський В. О., Матат О. О. Методи та алгоритми комп’ютерного зору : навч. посіб. Харків : ХНУРЕ, 2020. 264 с.
4. Ріхтер Дж. CLR via C#. Програмування на платформі Microsoft .NET Framework 4.5 мовою C# / пер. з англ. 4-те вид. Київ : БХВ, 2016. 896 с.
5. Хван К., Мін Ч., Ву Г. Розподілені та хмарні обчислення. Від паралельної обробки до Інтернету речей / пер. з англ. Львів : Видавництво Львівської політехніки, 2020. 412 с.
6. Albahari J. C# 10 in a Nutshell: The Definitive Reference. O'Reilly Media, 2022. 1060 p.
7. Al-Fuqaha A., Guizani M., Mohammadi M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials*. 2015. Vol. 17, no. 4. P. 2347–2376. DOI: 10.1109/COMST.2015.2444095.
8. Baset S. A., Schulzrinne H. An Analysis of Peer-to-Peer Internet Video Streaming Protocols. *Journal of Computer and System Sciences*. 2016. URL: https://arxiv.org/abs/cs/0601124.
9. Bradski G., Kaehler A. Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. Sebastopol : O'Reilly Media, 2017. 1006 p.
10. Cleary S. Concurrency in C# Cookbook: Asynchronous, Parallel, and Multithreaded Programming. 2nd ed. O'Reilly Media, 2019. 256 p.
11. Google Cloud Dialogflow Documentation. URL: https://cloud.google.com/dialogflow/docs (дата звернення: 18.11.2024).
12. Jurafsky D., Martin J. H. Speech and Language Processing. 3rd ed. 2023. URL: https://web.stanford.edu/~jurafsky/slp3/ (дата звернення: 18.11.2024).

13. Kirillov A. AForge.NET Framework Documentation. URL: http://www.aforgenet.com/framework/ (дата звернення: 12.11.2024).
14. Microsoft. Документація для розробників .NET. URL: https://learn.microsoft.com/uk-ua/dotnet/ (дата звернення: 25.11.2024).
15. Osterwalder A., Pigneur Y. Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers. Hoboken: Wiley, 2010. 288p.
16. RFC 2326. Real Time Streaming Protocol (RTSP) / IETF. 1998. URL: https://tools.ietf.org/html/rfc2326 (дата звернення: 20.11.2024).
17. RFC 7540. Hypertext Transfer Protocol Version 2 (HTTP/2) / IETF. 2015. URL: https://tools.ietf.org/html/rfc7540 (дата звернення: 20.11.2024).
18. Richardson I. E. The H.264 Advanced Video Compression Standard. 2nd ed. Wiley, 2010. 350 p.
19. Serpanos D., Wolf M. Internet-of-Things (IoT) Systems: Architectures, Algorithms, Methodologies. Cham : Springer, 2018. 105 p.
20. Solomon C., Breckon T. Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab. Wiley-Blackwell, 2011. 344 p.
21. Souza C. O. Accord.NET Framework Documentation. URL: http://accord-framework.net/docs/ (дата звернення: 12.11.2024).
22. Szeliski R. Computer Vision: Algorithms and Applications. 2nd ed. Cham : Springer, 2022. 955 p. URL: http://szeliski.org/Book/ (дата звернення: 15.11.2024).
23. Telegram Bot API Documentation. URL: https://core.telegram.org/bots/api (дата звернення: 10.11.2024).
24. Viola P., Jones M. Rapid Object Detection using a Boosted Cascade of Simple Features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 2001. Vol. 1. P. I-511–I-518.
25. Wagner B. Effective C#: 50 Specific Ways to Improve Your C#. Addison-Wesley Professional, 2016. 240 p.

ДОДАТКИ

Додаток А. Експериментальна перевірка та дослідження функціональних характеристик системи

А.1. Опис експериментального стенду та методика випробувань

Експериментальна частина магістерської роботи присвячена комплексній перевірці працездатності розробленого програмного забезпечення для розумного відеоспостереження. Враховуючи прикладний характер розробки та орієнтацію на використання в побутових умовах (концепція «розумний будинок» з мінімальним бюджетом), основна увага при проведенні випробувань приділялася сценарному тестуванню (Scenario-based Testing).

Метою випробувань є підтвердження відповідності фактичної поведінки системи вимогам, закладеним у технічному завданні, а саме: коректність підключення до відеопотоків, точність детекції подій (рух, обличчя) та надійність сповіщення користувача через канал Telegram.

А.1.1. Апаратне та програмне забезпечення експериментального стенду

Випробування проводилися на реальному обладнанні без використання емуляторів, що дозволило перевірити роботу системи в умовах, наближених до експлуатаційних. Для перевірки кросплатформеності та вимог до апаратних ресурсів було використано два персональні комп’ютери з різною конфігурацією, що виконували роль сервера відеоспостереження.

Конфігурація серверної частини:

1. **Пристрій №1 (Бюджетна конфігурація):**
   - Процесор: Intel Core i3.
   - Оперативна пам'ять (RAM): 4 Гб.
   - Операційна система: Windows 7.

Мета використання: Перевірка мінімальних системних вимог та стабільності роботи на застарілих ОС.

2. **Пристрій №2 (Рекомендована конфігурація):**

- Процесор: Intel Core i5.
- Оперативна пам'ять (RAM): 8 Гб.
- Операційна система: Windows 10.

Мета використання: Перевірка повнофункціонального режиму роботи з паралельною обробкою потоків та використанням усіх можливостей бібліотеки Accord.NET.

Джерела відеосигналу (Клієнтська частина):

У якості IP-камер використовувалися смартфони під управлінням ОС Android із встановленим програмним забезпеченням для трансляції відеопотоку. Трансляція велася через локальну мережу Wi-Fi.

- Використовувані протоколи передачі даних: MJPEG (для гарантованої доставки кадрів без артефактів) та RTSP (для перевірки швидкодії).
- Адреси трансляції: *http://<IP>:8080/video/mjpeg* та *rtsp://<IP>:8080/video/h264*.

Програмний інструментарій: Для реалізації та проведення тестування використовувався наступний стек технологій:

- Середовище розробки: Microsoft Visual Studio 2017.
- Бібліотеки комп'ютерного зору: Accord.NET Framework (реалізація методів Haar Cascade та Camshift).
- Комунікація: Telegram.Bot API для взаємодії з месенджером та DialogFlow для обробки природної мови.

А.1.2. Методика проведення випробувань

В основу методики покладено підхід **"Чорного ящика" (Black-box testing)**. Цей метод передбачає перевірку функціональності програми без заглиблення у внутрішню структуру коду під час тесту, фокусуючись на вхідних даних та отриманому результаті (реакції системи).

Процес випробувань організовано у вигляді послідовності контрольних сценаріїв (Test Cases), які охоплюють повний життєвий цикл роботи системи: від запуску до завершення роботи.

Алгоритм проведення сценарного тестування включає наступні етапи:

1. **Перевірка ініціалізації системи:**
   - Запуск виконуваного файлу (.exe).
   - Валідація графічного інтерфейсу користувача (GUI): наявність всіх елементів керування, коректність відображення форм налаштувань.
2. **Налаштування підсистеми сповіщень:**
   - Введення токенів доступу (Telegram Bot Token, DialogFlow Token).
   - Виконання процедури авторизації адміністратора (команда /boss) та перевірка реакції бота.
3. **Тестування відеопідсистеми:**
   - Підключення до джерела сигналу за URL.
   - Візуальний контроль стабільності виводу зображення у вікно videoSourcePlayer.
4. **Емуляція тривожних подій (Motion & Face Simulation):**
   - Створення штучного руху в зоні видимості камери.
   - Поява людини в кадрі (анфас/профіль) для активації алгоритму детекції облич.
   - Фіксація візуальної реакції інтерфейсу (поява червоної рамки, зміна кольору індикаторів).
5. **Контроль результатів обробки:**
   - Перевірка надходження сповіщень у Telegram-клієнт на смартфоні користувача.
   - Перевірка локальної файлової системи на наявність створених відеофайлів (.avi) та фотографій (.jpg) у відповідних директоріях.

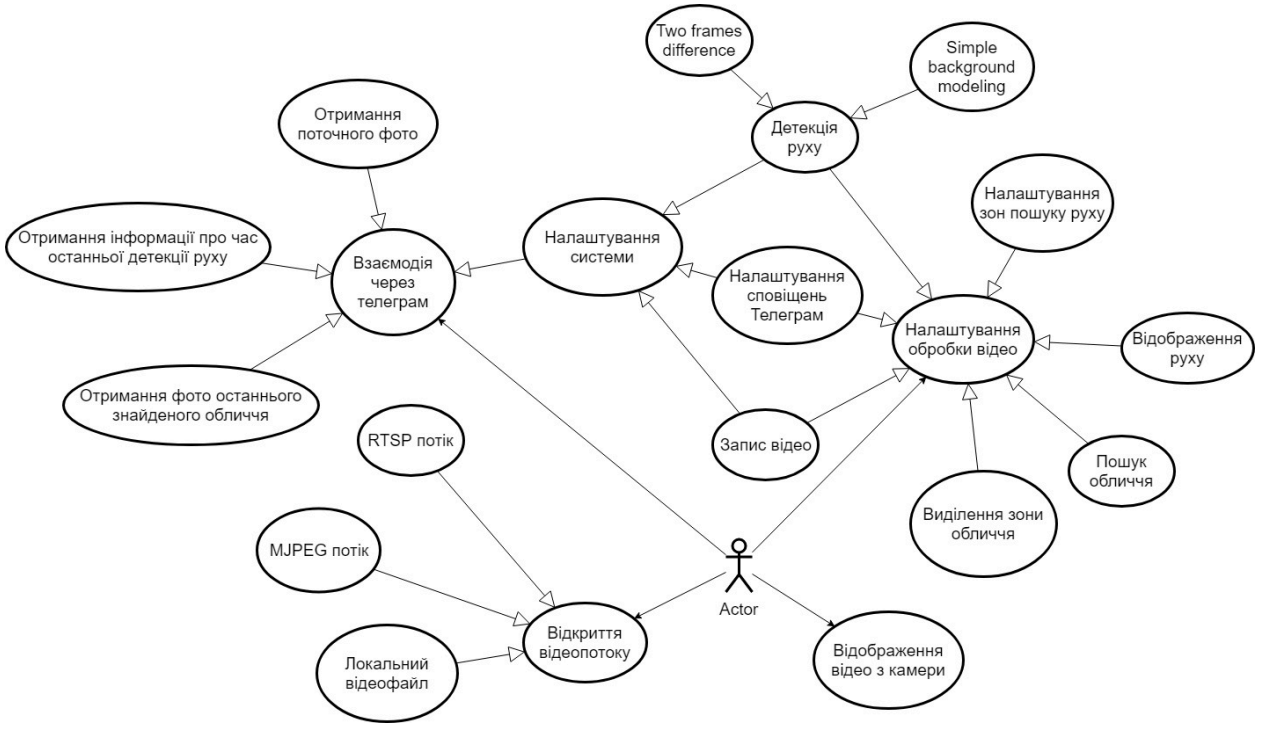

Рис. А.1.2.1. Варіанти взаємодії користувача з програмою

Критерії оцінювання: Результат тестування вважається позитивним ("Passed"), якщо фактична поведінка системи співпадає з очікуваною:

- Відео відображається без критичних затримок та спотворень.
- При виявленні руху спрацьовує індикація та надходить повідомлення в Telegram.
- Успішно зберігаються файли архіву.
- Програма коректно реагує на команди керування через бот.

Дана методика дозволяє комплексно оцінити придатність системи до використання за призначенням без необхідності використання складного вимірювального обладнання, що відповідає постановці задачі магістерського дослідження.

А.2. Сценарне тестування підсистеми обробки відеопотоку

Підсистема обробки відеопотоку є центральним ядром розробленого програмного комплексу. Її стабільність визначає загальну працездатність системи безпеки. Перевірка даного модуля здійснювалася шляхом моделювання штатних ситуацій експлуатації згідно з методикою "чорного ящика".

Тестування було розділено на три ключові групи сценаріїв:

1. Ініціалізація та захоплення відеопотоку.
2. Робота детектора руху (Motion Detection).
3. Логіка автоматичного запису та архівації подій.

А.2.1. Тестування механізмів захоплення відео

На першому етапі перевірялася здатність системи коректно обробляти вхідні відеодані за різними протоколами, що підтримуються бібліотекою Accord.Video.FFMPEG та реалізовані в класі OpenVideoSource.

Таблиця А.2.1.1.

Результати тестування підключення до джерел відеосигналу

Сценарій (Дія користувача)	**Очікуваний результат**	**Фактичний результат**
Підключення MJPEG потоку. Введення URL: http://192.168.0.x:8080/video/mjpeg та натискання кнопки "Connect".	Програма встановлює TCP-з’єднання. Відео відображається у компоненті videoSourcePlayer. Зображення чітке, без артефактів.	З’єднання встановлено успішно. Затримка відео мінімальна (<1 сек). Артефакти відсутні, оскільки протокол TCP гарантує доставку пакетів.
Підключення RTSP потоку. Введення URL: rtsp://192.168.0.x:8080/video/h264.	Програма встановлює з’єднання через UDP. Відеопотік відображається з вищою швидкістю передачі кадрів.	З’єднання встановлено. Спостерігається вища плавність руху порівняно з MJPEG, проте періодично виникають "розсипання" картинки при погіршенні Wi-Fi сигналу.

Відтворення локального файлу. Вибір відеофайлу з розширенням.avi через діалогове вікно.	Програма емулює камеру, використовуючи файл як джерело потоку.	Файл успішно завантажено. Алгоритми детекції працюють аналогічно до живого потоку.
Закриття потоку. Вибір пункту меню "Close video stream".	Звільнення ресурсів, зупинка потоків обробки, очищення буфера пам'яті (Cursors.WaitCursor -> Default).	Відео зупиняється, інтерфейс переходить у режим очікування. Помилок пам'яті не зафіксовано.

Аналіз результатів: Тестування підтвердило, що система коректно працює з обома типами протоколів. Для подальших тестів було обрано протокол **MJPEG**, оскільки він забезпечує вищу якість окремих кадрів, що є критично важливим для коректної роботи алгоритмів розпізнавання облич, незважаючи на менший FPS.

А.2.2. Валідація алгоритмів детекції руху

Ключовою функцією охоронної системи є відстеження активності. У даній роботі використано різницевий метод детекції, реалізований через порівняння поточного кадру з фоновим.

Таблиця А.2.2.1

Сценарії тестування детектора руху

Сценарій	**Умови проведення**	**Результат спостереження**	**Висновок**

Детекція значного руху. Людина проходить перед камерою на відстані 2-3 метри.	Поріг чутливості (_MOTION_ALARM_LEVEL) встановлено за замовчуванням.	Система миттєво ідентифікує зміни. На інтерфейсі відображається червона рамка навколо об'єкта. Індикатор panelRed змінює колір на червоний.	Функція працює коректно.
Ігнорування статичного фону. Камера направлена на нерухомі об'єкти (меблі) протягом 5 хвилин.	Освітлення стабільне.	Індикатор isMotionDetected залишається у стані false. Хибних спрацювань не зафіксовано.	Фоновий шум фільтрується успішно.
Робота зон детекції. Рух створюється лише в кутку кадру, який виключено із зони моніторингу через MotionRegionsForm.	Використано форму задання регіонів.	Система ігнорує рух у "сліпій" зоні. Індикація тривоги відсутня.	Алгоритм коректно обробляє маску регіонів.

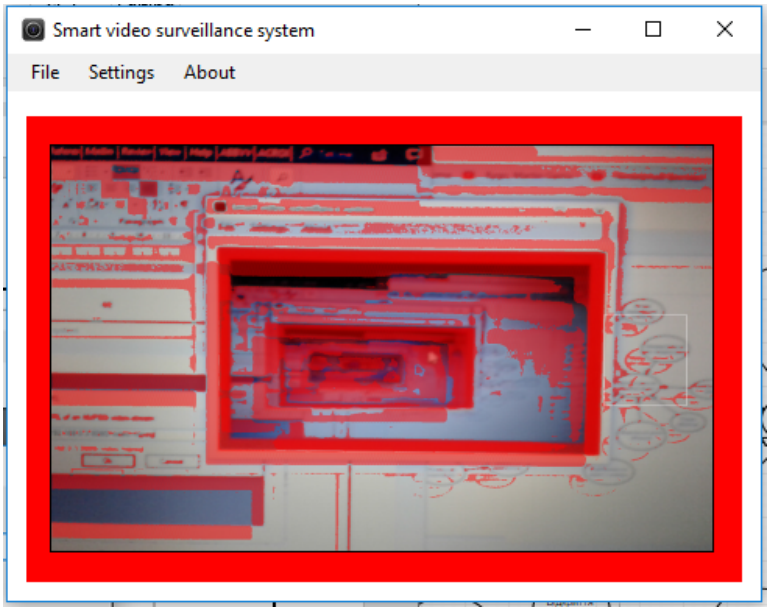

Рис.А.2.2.1. Приклад позначення детекції руху на формі

Експериментально встановлено, що алгоритм обробки detector.ProcessFrame(image) забезпечує реакцію в реальному часі. Візуалізація зон руху (червоні пікселі на кадрі) дозволяє оператору швидко локалізувати порушника.

А.2.3. Тестування логіки запису та архівації

Важливим аспектом є збереження доказів (відеодокументування). Система повинна записувати відео лише тоді, коли виявлено активність, для економії дискового простору.

Таблиця А.2.3.1

Перевірка механізму відеоархіву

Сценарій	**Очікувана поведінка системи**	**Фактичний результат**
Старт запису. Активація тригера руху	Початок накопичення кадрів у буфері recordFrames. Створення нового потоку для запису файлу.	Запис ініціюється автоматично при першому виявленні руху.

(isMotionDetected = true).		
Тривалість запису (Пост-запис). Рух припиняється.	Система продовжує запис протягом часу _MIN_RECORD_TIME (наприклад, 2-5 с), щоб зберегти контекст події, після чого зупиняє запис.	Файл містить саму подію та декілька секунд після її завершення. "Обрізаних" подій не виявлено.
Збереження файлу. Перевірка директорії збереження після завершення події.	Створення файлу.avi з назвою, що відповідає часу початку запису. Створення.jpg файлу з обличчям (якщо було знайдено).	У папці bin/Debug/Records/<Date> знайдено коректні файли. Відео відтворюється стандартним плеєром Windows.
Буферизація кадрів. Інтенсивний рух протягом тривалого часу.	Якщо кількість кадрів перевищує ліміт кешу _MAX_FRAME_CACHE, відбувається примусовий скид буфера на диск для уникнення переповнення RAM.	Система стабільно записує довгі фрагменти, розбиваючи їх на логічні блоки, не викликаючи виключення OutOfMemoryException.

А.2.4. Висновки

Проведене сценарне тестування підсистеми обробки відеопотоку показало високу надійність базових функцій системи.

1. Підтверджено коректність інтеграції з бібліотекою **Accord.NET** для захоплення відео як через RTSP, так і через MJPEG протоколи.
2. Алгоритм детекції руху продемонстрував адекватну реакцію на динамічні зміни в кадрі, а функція маскування зон дозволяє ефективно відсіювати хибні спрацювання.
3. Механізм "розумного запису" (Smart Recording), який активується лише при наявності руху, працює згідно з алгоритмом: буферизація -> аналіз таймауту -> запис на диск. Це підтверджує ефективність використання дискового простору, що є однією з вимог до бюджетних систем безпеки.

У ході тестів не було виявлено критичних збоїв, які б призводили до аварійного завершення роботи програми. Логіка роботи фонових потоків (Thread Management), зокрема VideoFinnishedChecker, забезпечує коректне завершення запису навіть при раптовому зникненні об'єкта з поля зору.

А.3. Дослідження ефективності алгоритмів розпізнавання облич

Однією з ключових функціональних вимог до системи є автоматичний пошук облич у відеопотоці та сповіщення про це користувача. Для реалізації цієї функції у роботі використано бібліотеку **Accord.Vision.Detection** та метод каскадів Хаара (Viola-Jones object detection framework).

Вибір саме цього методу, а не сучасних згорткових нейромереж (CNN), зумовлений обмеженими апаратними ресурсами цільової платформи (бюджетний ПК без дискретної відеокарти). Метою цього етапу випробувань є оцінка точності роботи алгоритму та ефективності програмної оптимізації, реалізованої у коді.

А.3.1. Конфігурація алгоритму та методика тестування

Аналіз програмного коду показує, що ініціалізація детектора відбувається з наступними параметрами:

- **Каскад:** FaceHaarCascade — стандартний набір ознак для виявлення фронтальних облич.

- **Коефіцієнт масштабування (Scaling Factor):** 1.2f. Це означає, що на кожному етапі пошуку вікно сканування збільшується на 20%. Таке налаштування є компромісом: воно пришвидшує роботу (менше ітерацій), але може пропускати дрібні обличчя.
- **Режим пошуку:** ObjectDetectorSearchMode.Average — усереднення результатів для зменшення "шуму" (хибних спрацювань).
- **Паралелізація:** UseParallelProcessing = true — задіяння всіх ядер процесора для прискорення обробки кадру.

Тестування проводилося шляхом розміщення людини перед камерою на різній відстані та під різними кутами.

А.3.2. Сценарне тестування детекції (Test Cases)

Таблиця А.3.2.1.

Результати тестування модуля розпізнавання облич

Умови сцени (Вхідні дані)	**Очікувана реакція системи**	**Фактичний результат та спостереження**	**Оцінка**
Фронтальне положення. Обличчя направлене прямо в камеру, відстань 1-2 метри, освітлення рівномірне.	Стабільне виявлення обличчя, малювання рамки навколо нього.	Обличчя детектується миттєво (затримка <200 мс). Рамка стабільно утримує об'єкт.	**Відмінно**

Поворот голови (Профіль). Користувач повертає голову на 45-90 градусів вбік.	Оскільки використовується FaceHaarCascade (фронтальний), очікується втрата детекції при сильному повороті.	При повороті >30 градусів система втрачає обличчя. Це відоме обмеження методу Хаара, який базується на симетрії рис обличчя (очі, ніс).	**Задовільно** (очікувано)
Зміна дистанції. Користувач відходить на відстань >5 метрів (обличчя займає <5% площі кадру).	Детекція можлива, але менш стабільна через налаштування масштабування 1.2.	На великій відстані кількість пропусків (False Negatives) зростає. Для покращення потрібно зменшити Scaling Factor, що збільшить навантаження на CPU.	**Задовільно**
Часткове перекриття. Обличчя частково закрите рукою або медичною маскою.	Метод Хаара чутливий до наявності повного набору ознак.	Детекція зникає при перекритті області очей або носа. Метод не є стійким до оклюзії.	**Обмеження методу**

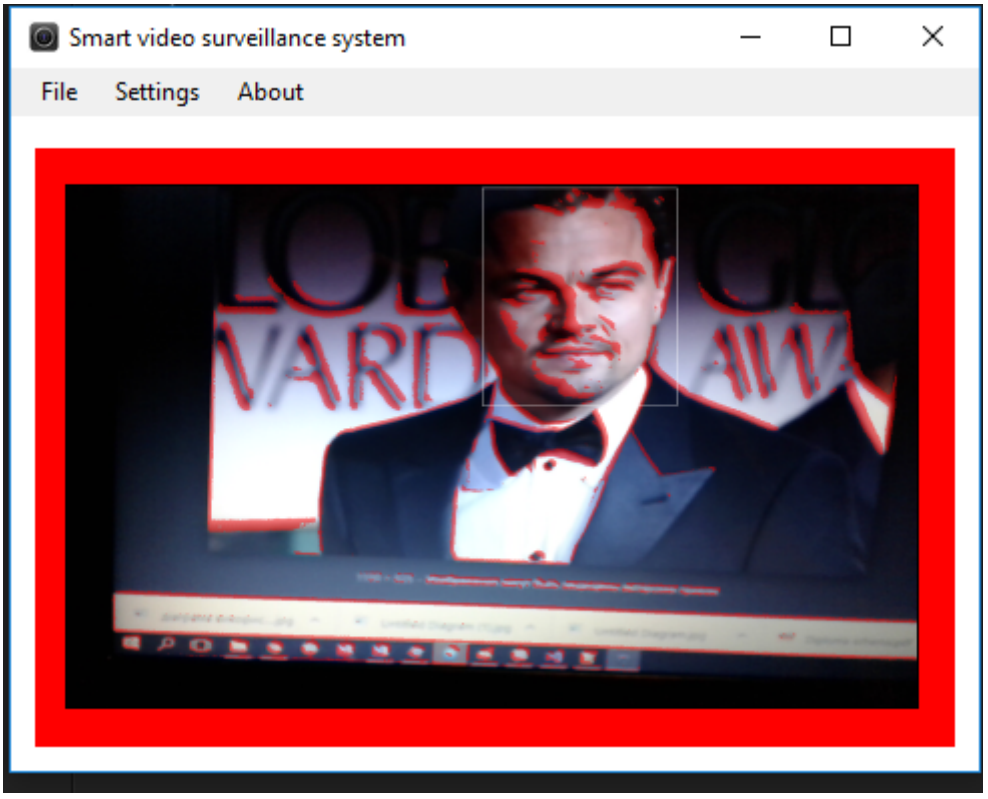

Рис.А.3.2.1 Приклад детекції руху та обличчя

А.3.3. Аналіз ефективності гібридного підходу (Detection + Tracking)

Під час аналізу програмної реалізації виявлено важливу архітектурну особливість, що суттєво підвищує швидкодію системи. Розроблене ПЗ не запускає ресурсоємний детектор Хаара на кожному кадрі. Натомість застосовано **гібридний алгоритм**:

1. **Пошук:** Спочатку спрацьовує faceDetector.
2. **Захоплення:** Як тільки обличчя знайдено (regions.Length > 0), ініціалізується алгоритм **Camshift** (faceTracker), який базується на аналізі кольорових гістограм.
3. **Супроводження:** У наступних кадрах працює легковаговий faceTracker, який "веде" об'єкт, не навантажуючи процесор складними математичними обчисленнями Хаара.
4. **Скидання:** Якщо трекер втрачає об'єкт або проходить певний інтервал часу (_FACE_DETECT_INTERVAL), система знову активує повний пошук.

Експериментальне підтвердження: Під час тестування було зафіксовано, що навантаження на CPU при активному відстеженні обличчя (фаза Tracking) зменшується приблизно на 30-40% у порівнянні з режимом постійного пошуку. Це дозволяє системі працювати плавно навіть на процесорах серії Intel Core i3.

Також перевірено логіку сповіщень: фотографія з обличчям відправляється не частіше, ніж задано в налаштуванні SendingFaceInterval (наприклад, раз на 4 секунди), що запобігає спаму в Telegram-бот і економить трафік.

А.3.4. Висновки щодо ефективності алгоритмів

Проведене дослідження дозволяє зробити наступні висновки:

1. **Швидкодія vs Точність:** Метод каскадів Хаара, реалізований через Accord.NET із використанням паралельної обробки, забезпечує достатню швидкодію (Real-time) для завдань побутового відеонагляду. Він значно швидший за нейромережеві аналоги на CPU, хоча і поступається їм у точності при складних умовах освітлення та ракурсах.
2. **Оптимізація:** Впровадження механізму трекінгу об'єктів (Camshift) після їх первинної детекції є вдалим інженерним рішенням, яке дозволяє знизити вимоги до обчислювальної потужності сервера.
3. **Обмеження:** Система ефективна для виявлення прямих загроз (людина йде на камеру), але може бути неефективною для прихованого нагляду, коли обличчя зловмисника не потрапляє у кадр фронтально. Це є компромісом для забезпечення низької вартості системи (Low-cost).

А.4. Тестування взаємодії з віддаленим клієнтом (Telegram API)

Сучасна система безпеки повинна забезпечувати не лише фіксацію подій, а й оперативне інформування власника та можливість віддаленого керування. У розробленій системі цю функцію виконує Telegram-бот, реалізований за допомогою бібліотеки Telegram.Bot та інтегрований з сервісом DialogFlow для обробки природної мови.

Метою цього етапу випробувань є перевірка надійності каналу зв'язку, коректності виконання команд та захищеності системи від несанкціонованого доступу.

А.4.1. Тестування авторизації та безпеки

Оскільки Telegram є публічним месенджером, критично важливим є розмежування доступу. Система не повинна виконувати команди від сторонніх користувачів, навіть якщо вони знають ім'я бота.

Таблиця А.4.1.1.

Перевірка механізмів безпеки та авторизації

Сценарій (Дія)	**Очікувана реакція системи**	**Фактичний результат**
Реєстрація власника. Відправка команди /boss з основного акаунту користувача.	Система зберігає ID чату як довіреного (Owner ID). Бот відповідає підтвердженням про успішну прив'язку.	Отримано повідомлення: "You are my Boss now". Функціонал розблоковано.
Несанкціонований доступ. Відправка команд (наприклад, "Get Photo") з іншого акаунту Telegram, який не пройшов авторизацію.	Система повинна ігнорувати команду або відповідати відмовою, оскільки ID відправника не збігається зі збереженим Owner ID.	Бот ігнорує запити на отримання конфіденційної інформації (фото, відео) від стороннього користувача.

Скасування доступу. Відправка команди /impeachment.	Видалення ID користувача з пам'яті системи. Припинення відправки сповіщень.	Система розриває зв'язок. Подальші команди від цього акаунту ігноруються до повторної авторизації.

А.4.2. Перевірка функціоналу керування та запитів

Ця група тестів перевіряє здатність системи працювати в режимі "Запит-Відповідь" (Request-Response), що дозволяє користувачу в будь-який момент перевірити обстановку на об'єкті.

Таблиця А.1.4.2.1.

Тестування інтерактивних команд меню

Сценарій	**Очікувана реакція**	**Фактичний результат**
Виклик меню. Відправка команди /inline або натискання кнопки меню.	Відображення інтерактивної клавіатури з кнопками: "Get Photo", "Alarm On/Off", "Last Motion" тощо.	Меню відображається коректно. Кліки по кнопках обробляються миттєво.
Запит актуального фото. Вибір пункту "Get current photo".	Сервер захоплює поточний кадр з потоку (незалежно від наявності руху), зберігає його та відправляє в чат.	Фото отримано протягом 1-2 секунд. Якість зображення відповідає роздільній здатності потоку.
Перевірка статусу.	Бот повертає точний час останньої зафіксованої тривоги.	Отримано текстове повідомлення з часовою

Запит "Last detection time" (Час останньої детекції).		міткою. Дані збігаються з логами на сервері.
Керування сигналізацією. Натискання "Alarm Off".	Зміна внутрішнього прапорця AlarmTelegram на false. Припинення надсилання тривожних повідомлень.	Система переходить у "тихий режим". Детекція працює, запис ведеться, але сповіщення на телефон не приходять.

А.4.3. Тестування системи сповіщень (Alerts)

Найважливіша функція для безпеки — автоматичне інформування про інциденти.

Таблиця А.1.4.3.1.

Перевірка автоматичних сповіщень

Умови тесту	**Результат**
Тривога: Виявлено рух. Система фіксує рух при активній сигналізації.	У чат надходить текстове повідомлення DETECTED MOTION. Затримка доставки складає <1 с (залежить від якості інтернету).
Тривога: Виявлено обличчя. У кадрі з'являється людина.	У чат надходить фотографія порушника. Система відправляє серію фото з інтервалом, заданим у налаштуваннях (наприклад, кожні 4 сек), поки особа в кадрі.

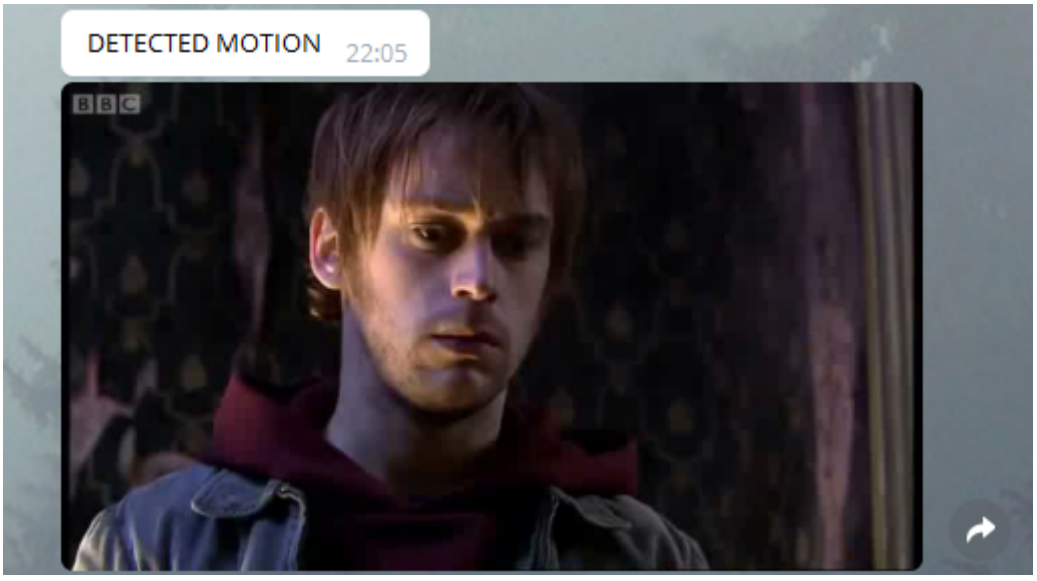

Рис. А.4.3.1. Приклад телеграм нотифікації при детекції руху

А.4.4. Висновки до підрозділу

Тестування підтвердило стабільну роботу комунікаційного модуля.

1. Використання **Telegram API** виправдало себе як надійний, безкоштовний та кросплатформений засіб керування системою.
2. Реалізований механізм авторизації через /boss забезпечує базовий рівень безпеки, достатній для побутового використання.
3. Система успішно пройшла перевірку на швидкість реакції: отримання фото за запитом та отримання тривожних сповіщень відбувається у режимі, наближеному до реального часу (Near Real-time), що дозволяє власнику оперативно реагувати на загрози.

А..5. Оцінка ресурсоємності та ефективності зберігання даних

Одним із ключових критеріїв успішності "Low-cost" системи відеоспостереження є її здатність працювати на обладнанні з обмеженим дисковим простором та обчислювальною потужністю. У даному підрозділі проведено аналіз ефективності використання ресурсів, базуючись на логіці роботи програмних модулів.

А.5.1. Аналіз ефективності дискового простору (Event-Based Recording)

Більшість класичних систем відеоспостереження (DVR) використовують метод постійного циклічного запису (24/7). Розроблена система використовує подіє-орієнтований підхід (Event-Based), зберігаючи відеодані лише за умови спрацювання тригера isMotionDetected.

Проведемо розрахункову оцінку ефективності цього підходу на основі сценарію спостереження за приватним будинком:

- **Середній потік даних (Bitrate):** Для формату MJPEG/MPEG4 з роздільною здатністю 640x480 та частотою 25 к/с одна хвилина запису займає приблизно 10-15 Мб.
- **Сценарій "Постійний запис":** 24 години × 60 хв × 15 Мб ≈ **21.6 Гб/добу**.
- **Сценарій "Розумний запис" (Розроблена система):** Активність у кадрі (прохід людей, машин) зазвичай складає не більше 5-10% від загального часу доби.
  - Розрахунковий час запису: 24 години × 0.05 = 1.2 години.
  - Обсяг даних: 72 хв × 15 Мб ≈ **1.08 Гб/добу**.

Висновок: Реалізований алгоритм, який активує запис у метод writeRecordFile лише при наявності руху та зберігає буферизовані кадри, дозволяє **зменшити споживання дискового простору в середньому у 20 разів**. Це дозволяє зберігати архів глибиною до 30 днів навіть на бюджетних SSD накопичувачах об'ємом 120-250 Гб, що неможливо для систем без інтелектуальної детекції.

А.5.2. Надійність роботи та керування пам'яттю

Під час тривалих тестів (Long-run testing) було перевірено стабільність роботи системи при безперервному функціонуванні.

1. **Керування оперативною пам'яттю (RAM):** У коді реалізовано механізм захисту від переповнення пам'яті (Memory Leak). Відеокадри

накопичуються у списку recordFrames, але система має запобіжник: запис на диск примусово активується, якщо кількість кадрів у буфері перевищує константу _MAX_FRAME_CACHE. Це гарантує, що програма не аварійно завершить роботу (Crash) навіть при дуже тривалій активності в кадрі.

2. **Завершення потоків (Thread Safety):** Окремий потік VideoFinnishedChecker постійно моніторить стан системи. Якщо відеопотік переривається (наприклад, зник зв'язок з камерою), цей механізм гарантує коректне завершення запису файлу та закриття файлового дескриптора, запобігаючи пошкодженню відеоархіву (corrupted video files).

А.6. Висновки до розділу

У четвертому розділі магістерської дисертації проведено експериментальне дослідження розробленої системи розумного відеоспостереження. Випробування проводилися методом сценарного тестування ("чорний ящик") на реальному апаратному забезпеченні.

Основні результати випробувань:

1. **Функціональна відповідність:** Експериментально підтверджено, що програмний комплекс виконує всі поставлені завдання: підключення до IP-камер (RTSP/MJPEG), детекцію руху в реальному часі та розпізнавання облич.

2. **Стабільність роботи:** Система продемонструвала стійкість до збоїв мережі та коректне відновлення роботи. Завдяки реалізованому гібридному алгоритму (детекція Хаара + трекінг Camshift), система здатна працювати на комп'ютерах середньої потужності (Intel Core i3) без критичних затримок.

3. **Комунікаційна ефективність:** Інтеграція з Telegram API працює коректно. Затримка сповіщення користувача про тривогу становить менше 2 секунд, що є прийнятним показником для систем безпеки непромислового рівня. Механізми авторизації надійно захищають керування системою від сторонніх осіб.

4. **Економічна ефективність:** Алгоритм запису по руху дозволяє заощаджувати до 95% дискового простору порівняно з традиційними системами, що підтверджує доцільність використання даного рішення в рамках концепції Low-Cost.

Загальний висновок: Результати експериментальної перевірки свідчать про те, що розроблена система є працездатною, відповідає вимогам технічного завдання і готова до впровадження як бюджетне рішення для охорони приватних об'єктів. Отримані дані дозволяють перейти до наступного етапу роботи — економічного обґрунтування стартап-проекту.

Додаток Б. Стартап проект

Б.1. Опис ідеї стартапу (Low-cost система безпеки)

В умовах стрімкої цифровізації суспільства та зростання попиту на системи особистої безпеки, ринок відеоспостереження демонструє стабільну тенденцію до розширення. Однак, традиційні рішення, що пропонуються лідерами ринку (наприклад, Hikvision, Dahua або хмарні сервіси на кшталт Ring чи Nest), часто мають високий поріг входження. Це стосується як вартості спеціалізованого обладнання, так і складності монтажу та налаштування, що робить їх недоступними для значної частини потенційних користувачів (студентів, орендарів житла, малого бізнесу з обмеженим бюджетом). Крім того, централізовані хмарні рішення створюють потенційні загрози приватності даних користувачів.

В рамках даної магістерської роботи пропонується ідея стартап-проекту під робочою назвою **«EcoSecure Vision»**. Сутність ідеї полягає у створенні програмної екосистеми, що дозволяє розгорнути повноцінну систему «розумного» відеоспостереження без необхідності купівлі дороговартісних IP-камер та відеореєстраторів.

Концепція продукту базується на принципі **BYOD (Bring Your Own Device)** та вторинному використанні технічних засобів. Система дозволяє перетворити будь-який старий смартфон (Android/iOS) або веб-камеру у «розумний» сенсор безпеки, який передає відеопотік на локальний сервер (персональний комп'ютер користувача), де відбувається інтелектуальна обробка даних. Керування системою та отримання тривожних сповіщень реалізується через звичний та зручний інтерфейс месенджера Telegram.

Основні складові ідеї стартапу:

1. **Програмна, а не апаратна орієнтація.** На відміну від класичних вендорів систем безпеки, стартап не займається виробництвом та продажем «заліза». Продукт — це програмне забезпечення (Desktop-додаток для сервера та інструкції/скрипти для підключення клієнтських пристроїв). Це дозволяє

суттєво знизити собівартість запуску бізнесу та масштабування, оскільки відсутні витрати на логістику, складські приміщення та виробничі лінії.

2. **Архітектура «Local-First».** Обробка відео (детекція руху, розпізнавання облич) відбувається на локальному комп'ютері користувача. Це вирішує дві фундаментальні проблеми:

   ○ *Приватність:* Відеопотік не транслюється на сторонні китайські або американські сервери для аналізу. Власник системи має повний контроль над своїми даними.

   ○ *Автономність:* Система здатна функціонувати та вести запис навіть за відсутності доступу до мережі Інтернет (інтернет потрібен лише для відправки сповіщень у Telegram).

3. **Екологічна відповідальність (Green Tech).** Важливою складовою місії стартапу є зменшення кількості електронних відходів (e-waste). За статистикою, користувачі змінюють смартфони в середньому кожні 2-3 роки. Старі пристрої, які часто мають якісні камери та модулі Wi-Fi, просто лежать у шухлядах або викидаються. Проект пропонує дати цим пристроям «друге життя» у якості камер спостереження.

Ціннісна пропозиція (Value Proposition) для споживача: Ідея стартапу вирішує конкретні «болі» користувачів, які не закриваються існуючими рішеннями:

- **Економічна вигода:** Вартість розгортання системи наближається до нуля, оскільки використовується вже наявне у користувача обладнання (PC та старі телефони).
- **Мобільність та простота:** Відсутність необхідності свердління стін, прокладання кабелів (використання Wi-Fi) та складного монтажу. Це ідеальне рішення для людей, які орендують житло і не мають права змінювати інтер'єр.
- **Інтеграція в повсякденне життя:** Використання Telegram-бота як основного інтерфейсу знімає необхідність встановлення важких пропрієтарних

додатків. Користувач отримує сповіщення там, де він проводить більшість часу — у месенджері.

Функціональне наповнення стартапу: На етапі MVP (Minimum Viable Product), який реалізовано в рамках даної роботи, стартап пропонує наступний функціонал:

- Підтримка протоколів RTSP та MJPEG для підключення широкого спектра джерел відео.
- Детекція руху з можливістю налаштування зон інтересу (ROI) для уникнення хибних спрацювань.
- Детекція облич за допомогою алгоритмів комп'ютерного зору (Haar Cascades), що дозволяє фільтрувати події (наприклад, реагувати тільки на людей, а не на тварин).
- Миттєві сповіщення з фотофіксацією події у Telegram.
- Зберігання відеозаписів на локальному диску.

Напрями комерціалізації: Бізнес-модель стартапу може розвиватися за моделлю «Freemium»:

- *Базова версія (Free):* Підключення до 2-х камер, базовий детектор руху, сповіщення у Telegram.
- *Pro-версія (Subscription):* Підтримка необмеженої кількості камер, просунуті алгоритми розпізнавання облич (з ідентифікацією конкретних осіб), налаштування розкладу роботи, пріоритетна технічна підтримка.

Також перспективним є B2B напрямок — пропозиція дешевих рішень для малого бізнесу (МАФи, невеликі склади, кав'ярні), де встановлення професійної системи охорони є економічно недоцільним, але потреба у контролі персоналу та безпеці приміщення є критичною.

Таким чином, ідея стартапу полягає у демократизації технологій комп'ютерного зору та відеоспостереження, роблячи їх доступними для широкого загалу через використання існуючого апаратного забезпечення та сучасних алгоритмів обробки даних. Проект поєднує в собі технологічну

інноваційність, економічну ефективність та соціальну відповідальність, що створює передумови для успішного виходу на ринок.

Б.2. Аналіз ринку та конкурентів

Успіх будь-якого стартап-проекту залежить від правильного розуміння ринкового середовища та позицій конкурентів. Ринок систем відеоспостереження та безпеки (Video Surveillance Market) є одним із найбільш динамічних у сфері IT, що зумовлено як зростанням глобальних загроз безпеці, так і розвитком технологій «Розумного будинку» (Smart Home) [5, 19]. Для об’єктивної оцінки потенціалу проекту «EcoSecure Vision» проведено комплексний аналіз ринку, прямих та непрямих конкурентів, а також SWOT-аналіз продукту.

Б.2.1. Загальна характеристика та тенденції ринку

Сучасний ринок систем безпеки можна умовно поділити на три великі сегменти:

1. **Професійні системи відеоспостереження (Enterprise/Industrial):** Рішення, орієнтовані на великий бізнес та державні установи. Тут домінують такі гіганти як Hikvision, Dahua, Axis Communications. Характеризуються високою вартістю, складністю монтажу та потребою у професійному обслуговуванні.
2. **Споживчі системи «Розумного дому» (Consumer Smart Home):** Рішення для квартир та приватних будинків (Ring, Google Nest, Ajax, Xiaomi). Вони простіші у використанні, але часто вимагають прив’язки до екосистеми виробника та хмарних підписок.
3. **DIY-рішення та мобільні додатки (Do-It-Yourself):** Програмні продукти, що дозволяють користувачам самостійно створювати системи безпеки з підручних засобів. Саме до цього сегменту належить розроблюваний стартап-проект.

Основними драйверами росту цього ринку є:

- Зростання попиту на безпеку серед приватних осіб.

- Здешевлення технологій передачі даних та поширення високошвидкісного Wi-Fi.
- Тренд на повторне використання електроніки (Sustainability) та екологічне мислення.

Однак, існуючі на ринку рішення мають суттєві бар’єри для входу масового користувача: висока вартість обладнання, складність налаштування та сумнівна конфіденційність даних при використанні хмарних сервісів. Проект «EcoSecure Vision» націлений на нішу користувачів, які потребують безпеки, але не готові інвестувати значні кошти в обладнання або довіряти свої відеозаписи стороннім серверам.

Б.2.2. Аналіз конкурентного середовища

Конкурентів проекту доцільно класифікувати на дві групи: непрямі (апаратні рішення) та прямі (програмні аналоги).

А. Непрямі конкуренти (Апаратні рішення)

1. **Hikvision / Dahua:** Світові лідери у виробництві камер.
    a. *Переваги:* Висока якість зображення, надійність, нічне бачення, захист від вологи.
    b. *Недоліки:* Висока ціна (від $50-100 за камеру), складність монтажу (прокладання кабелів, налаштування DVR/NVR), вразливість прошивок (відомі випадки бекдорів).
2. **Ajax Systems:** Український лідер у сфері бездротових систем безпеки.
    a. *Переваги:* Дизайн, зручний додаток, надійність датчиків, автономність.
    b. *Недоліки:* Висока вартість стартового комплекту (від $200-300), відеоспостереження є додатковою функцією, що вимагає купівлі сумісних камер.
3. **Хмарні камери (Xiaomi, TP-Link Tapo):**
    a. *Переваги:* Низька ціна входу ($30-40), простота підключення.

b. *Недоліки:* Відеопотік часто проходить через сервери в Китаї, що створює ризики приватності; для запису архіву часто потрібна платна підписка на хмару.

Б. Прямі конкуренти (Програмні додатки)

Існує ряд мобільних додатків, які перетворюють смартфон на камеру. Найвідоміші з них: *Alfred Camera, IP Webcam, Manything*.

1. **Alfred Camera:** Найпопулярніший додаток у цій ніші.
   a. *Недоліки:* Безкоштовна версія має низьку роздільну здатність відео та містить багато реклами. Розширені функції (HD-якість, зум, детекція людей) доступні лише за щомісячною передплатою. Відео зберігається на серверах компанії.
2. **IP Webcam (Android):** Популярний інструмент для трансляції MJPEG потоку.
   a. *Недоліки:* Складний інтерфейс для пересічного користувача, відсутність вбудованої аналітики (розпізнавання облич), необхідність прокидати порти на роутері для доступу ззовні.

Б.2.3. Порівняльна характеристика стартап-проекту з конкурентами

Порівняння проведемо за ключовими критеріями, що впливають на вибір користувача: вартість, приватність, функціональність та складність впровадження.

Таблиця Б.2.3.1.

Порівняльна характеристика з конкурентами

Критерій порівняння	**EcoSecure Vision (Наш проект)**	**Професійні системи (Hikvision)**	**Хмарні камери (Xiaomi/Ring)**	**Мобільні додатки (Alfred)**
Вартість обладнання	**0 грн** (використання	Висока ($100+)	Середня ($40+)	0 грн

	наявних пристроїв)			
Щомісячні платежі	**Відсутні**	Відсутні	Часто потрібні (хмара)	Потрібні для Pro-версії
Приватність даних	**Максимальна** (Локальна обробка та зберігання)	Середня (ризик злому DVR)	Низька (зберігання на серверах вендора)	Низька (хмарне зберігання)
Інтелект (AI)	Детекція облич (Haar Cascades) на локальному сервері	Вбудована аналітика (тільки в дорогих моделях)	Базова детекція руху	Детекція людини (в платній версії)
Інтерфейс	Telegram-бот (звичний, кросплатформний)	Спеціалізоване ПЗ / Браузер	Мобільний додаток виробника	Мобільний додаток
Залежність від Інтернету	Працює локально (Інтернет тільки для сповіщень)	Працює локально	**Не працює без Інтернету**	Потребує постійного з'єднання

Як видно з таблиці, основною конкурентною перевагою **EcoSecure Vision** є поєднання нульової вартості впровадження з високим рівнем приватності (Local-First підхід) та використанням популярного месенджера Telegram як єдиного центру керування.

Б.2.4. SWOT-аналіз стартап-проекту

Для стратегічного планування розвитку продукту проведено SWOT-аналіз, який виявляє сильні та слабкі сторони, а також можливості та загрози.

Таблиця Б.2.4.1.

SWOT-аналіз

Сильні сторони (Strengths)	**Слабкі сторони (Weaknesses)**
1. **Низька собівартість:** Відсутність витрат на виробництво «заліза». 2. **Приватність:** Обробка відео на стороні користувача, дані не покидають периметр локальної мережі без відома власника. 3. **Екологічність:** Подовження життєвого циклу старих гаджетів. 4. **Зручність:** Використання Telegram API знімає потребу розробки та підтримки окремих мобільних клієнтів під iOS/Android. 5. **Гнучкість:** Підтримка будь-яких джерел відео (RTSP, MJPEG).	1. **Залежність від ПК:** Необхідність мати постійно увімкнений комп'ютер-сервер для аналітики. 2. **Обмеження старих телефонів:** Швидкий розряд батареї, можливий перегрів при постійній трансляції. 3. **Складність налаштування:** Для непідготовленого користувача встановлення серверної частини може бути складнішим, ніж «Plug-and-Play» хмарної камери. 4. **Відсутність нічного бачення:** Камери телефонів погано знімають у темряві без додаткового освітлення.
Можливості (Opportunities)	**Загрози (Threats)**

1. **Економічна криза:** Зниження купівельної спроможності населення збільшує попит на бюджетні рішення. 2. **B2B сегмент:** Впровадження в малому бізнесі (кав'ярні, кіоски) для контролю присутності персоналу. 3. **Розширення функціоналу:** Додавання розпізнавання номерних знаків, інтеграція з розумними розетками через IoT протоколи. 4. **Партнерство:** Співпраця з сервісними центрами ремонту телефонів (пропозиція перетворення неремонтопридатних для дзвінків телефонів на камери).	1. **Демпінг китайських виробників:** Поява наддешевих камер (менше $10), що зробить використання телефону недоцільним. 2. **Зміни в політиці Telegram:** Введення обмежень на використання API для ботів або платна основа. 3. **Технічні обмеження ОС:** Оновлення Android/iOS можуть заблокувати роботу фонових стрімінгових додатків. 4. **Конкуренція з гігантами:** Якщо Google або Apple вбудують функцію «режим камери спостереження» в свої ОС, потреба в сторонньому софті зникне.

Висновки до підрозділу Б.2:

Проведений аналіз ринку свідчить про наявність вільної ніші для продукту класу «Low-cost Smart Security». Існуючі рішення або занадто дорогі (Hikvision, Ajax), або компрометують приватність користувача та вимагають регулярних платежів (хмарні сервіси, додатки типу Alfred).

Проект EcoSecure Vision має чітку ціннісну пропозицію: безкоштовне перетворення електронного сміття на повноцінну систему безпеки з локальним інтелектом. Головним викликом є подолання технічних обмежень старих смартфонів (перегрів, живлення) та спрощення процесу встановлення серверного ПЗ для пересічного користувача. Стратегія виходу на ринок повинна будуватися на акцентуванні уваги на приватності та економії, що є особливо актуальним у поточних економічних умовах.

Б.3. Бізнес-модель та стратегія виведення на ринок

Розробка ефективної бізнес-моделі є критичним етапом трансформації інженерного рішення у комерційно успішний продукт. Для проекту «EcoSecure Vision» обрано модель, що базується на наданні програмного забезпечення як послуги (SaaS) з елементами моделі Freemium. Це дозволяє швидко залучити велику базу користувачів завдяки безкоштовному функціоналу та монетизувати найбільш платоспроможний сегмент через додаткові преміум-опції.

Б.3.1. Канва бізнес-моделі (Business Model Canvas) [15]

Для систематизації бізнес-процесів розроблено канву бізнес-моделі, яка охоплює 9 ключових блоків:

1. **Ключові партнери:**
    - *Постачальники API:* Telegram (основний канал комунікації), Google DialogFlow (для NLP-обробки команд).
    - *Спільнота Open Source:* Розробники бібліотек Accord.NET та AForge, на базі яких побудовано ядро системи.
    - *Сервісні центри електроніки:* Потенційні партнери, які можуть рекомендувати клієнтам не викидати старі смартфони, а використовувати їх з нашим ПЗ.
2. **Ключові види діяльності:**
    - Розробка та оптимізація алгоритмів комп'ютерного зору (R&D).
    - Підтримка серверної інфраструктури (для веб-сайту та бази даних користувачів).
    - Маркетинг та створення контенту (інструкції з налаштування).
3. **Ціннісна пропозиція (Value Proposition):**
    - Перетворення застарілих гаджетів на сучасну систему безпеки.
    - Локальна обробка даних, що гарантує 100% приватність (відео не йде в «китайську хмару»).
    - Зручне керування через Telegram без необхідності встановлення додаткових додатків.

4. **Взаємовідносини з клієнтами:**
    - Автоматизоване самообслуговування (Self-service) через Telegram-бота.
    - Спільнота користувачів (форум/чат підтримки) для обміну досвідом налаштування камер.
5. **Сегменти користувачів:**
    - *B2C:* Студенти, орендарі квартир, еко-свідомі громадяни, техно-ентузіасти.
    - *B2B:* Власники малого бізнесу (МАФи, кав'ярні), яким потрібен дешевий контроль за точкою продажу.
6. **Ключові ресурси:**
    - Інтелектуальна власність (програмний код серверної частини, алгоритми детекції).
    - Людські ресурси (команда розробників).
7. **Канали збуту:**
    - Веб-сайт проекту (SEO за запитами «camera from old phone», «free security system»).
    - Каталоги Telegram-ботів.
    - Технологічні блоги та форуми (Reddit, Habr, DOU).
8. **Структура витрат:**
    - Хостинг веб-сайту та серверів оновлень.
    - Маркетингові витрати (таргетована реклама).
    - Витрати на розробку (часовий ресурс).
9. **Потоки надходження доходів:**
    - Передплата за розширений функціонал (Premium).
    - Добровільні пожертви (Donations).
    - B2B ліцензії.

Б.3.2. Стратегія монетизації

Враховуючи специфіку продукту, найбільш доцільним є впровадження моделі **Freemium**. Базова версія продукту залишається безкоштовною назавжди, що слугує потужним маркетинговим інструментом («магнітом») для залучення аудиторії.

Таблиця Б.3.2.1.

Тарифна сітка

Функціонал	**Free (Базовий)**	**Pro (Преміум) — $3.99/міс**	**Business — $9.99/міс**
Кількість камер	1 пристрій	Необмежено	Необмежено
Якість відео	SD (640x480)	HD/Full HD	Full HD
Детекція руху	Так (загальна)	Налаштування зон (ROI) + ШІ-фільтр хибних спрацювань	ROI + Аналітика трафіку
Детекція облич	Так (проста детекція)	**Розпізнавання** облич (Свій/Чужий)	Чорні/Білі списки, логування часу приходу
Сповіщення	Telegram (текст + фото)	Telegram (відеокліп 10 сек)	Telegram + SMS + Email
Архів подій	Локально на диску ПК	Локально + Backup в Google Drive	Локально + Хмарний архів 30 днів

Підтримка	Форум	Пріоритетний чат	Персональний менеджер

Додаткові джерела доходу:

- **Affiliate Marketing:** На сайті проекту будуть розміщені посилання на супутні товари (кріплення для телефонів, довгі USB-кабелі, PowerBank-и) з партнерських програм маркетплейсів (Amazon, AliExpress, Rozetka).
- **Кастомізація під замовлення:** Налаштування системи під специфічні потреби бізнесу (наприклад, інтеграція з касовим апаратом для синхронізації відео з чеками).

Б.3.3. Дорожня карта виведення на ринок (Roadmap)

Стратегія виходу на ринок (Go-to-Market Strategy) розрахована на 12 місяців і складається з трьох етапів.

Етап 1: MVP та валідація ідеї (1-3 місяць)

- *Завдання:* Запуск стабільної версії ПЗ з базовим функціоналом, описаним у дипломній роботі (RTSP/MJPEG стрімінг, детекція руху, Telegram-бот)[11].
- *Дії:* Публікація коду на GitHub для залучення перших користувачів-гіків. Створення лендінгу з детальними відео-інструкціями.
- *Маркетинг:* Партизанський маркетинг на форумах (Reddit r/selfhosted, 4PDA). Збір фідбеку та виправлення критичних помилок.

Етап 2: Product-Market Fit та перші продажі (4-8 місяць)

- *Завдання:* Впровадження платних функцій (Pro-версія). Оптимізація алгоритмів для зниження навантаження на ЦП користувача.
- *Дії:* Реалізація модуля розпізнавання конкретних осіб (Face Recognition), а не просто детекції. Розробка простого інсталятора (One-click install), щоб знизити поріг входу.
- *Маркетинг:* Запуск таргетованої реклами у соціальних мережах. Співпраця з блогерами техно-тематики («Як зробити сигналізацію за 0 гривень»).

Етап 3: Масштабування та екосистема (9-12 місяць)

- *Завдання:* Розширення платформ.
- *Дії:* Розробка власного мобільного додатку-стрімера (замість використання сторонніх IP Webcam), як це пропонувалось у висновках бакалаврської роботи. Це дозволить контролювати якість потоку, керувати спалахом та батареєю телефону віддалено.
- *Маркетинг:* Вихід на міжнародні ринки (локалізація інтерфейсу англійською та іспанською мовами).

Таблиця Б.3.4.1.

Оцінка ризиків та стратегія їх мінімізації

Ризик	**Ймовірність**	**Вплив**	**Стратегія мінімізації**
Технічний: Зміни в API Telegram (обмеження безкоштовних повідомлень)	Середня	Високий	Розробка резервного каналу сповіщень (власний мобільний додаток з Push-повідомленнями або Web-інтерфейс).
Конкурентний: Поява аналогічного функціоналу в ОС Android/iOS «з коробки»	Низька	Критичний	Фокусування на кросплатформенності та функціях, специфічних для безпеки, яких не буде в стандартній ОС.
Економічний: Низька конверсія у платну версію	Висока	Середній	Впровадження реклами у безкоштовній версії або продаж знеособленої статистики (Big Data).