Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Кіберфізична система контролю евакуації людей з приміщень організації
Назва теми

Рівень вищої освіти *перший (бакалаврський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Шифр КвРКІ 250203.23.02.48 ПЗ

Виконав здобувач ІІІ курсу, група КІ2с-23-2 Артем КОРОЛЬЧУК
Підпис Ініціали, прізвище

Керівник к.т.н., доцент Марія КАПУСТЯН
Науковий ступінь, учене звання Підпис Ініціали, прізвище

Нормоконтролер д.т.н., професор Сергій ЛИСЕНКО
Науковий ступінь, учене звання Підпис Ініціали, прізвище

До захисту допускаю:
завідувач кафедри КІІС Ольга ПАВЛОВА
«01» червня 2026 р. Підпис Ініціали, прізвище
дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Рівень вищої освіти перший (бакалаврський)

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КПС
Ольга ПАВЛОВА

" 10 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Корольчуку Артему Вадимовичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Кіберфізична система контролю евакуації людей з приміщень організації.

Керівник проекту (роботи) Капустян Марія Вікторівна, к.т.н., доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.03.2026 р. № 5

2. Термін подання здобувачем роботи на кафедру 01.06.2026 р.

3. Вихідні дані до роботи Завдання на кваліфікаційну роботу

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

Система відеомоніторингу та автоматизованого підрахунку кількості людей у приміщенні під час штатного режиму та евакуації, постановка задачі щодо підвищення точності контролю присутності людей

Проєктування структури інформаційної системи відеоспостереження, алгортимів детекції, відстеження та підрахунку людей у приміщенні

Програмно-апаратна Програмна реалізація веборієнтованої системи моніторингу кількості людей, тестування роботи системи та оцінка ефективності застосування під час евакуації

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

Структурно-функціональна схема системи контролю евакуації людей

Блок-схеми алгоритмів підрахунку людей та контролю евакуації

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Сергій ЛИСЕНКО, професор кафедри КІІС		
Антиплагіат	Андрій НІЧЕПОРУК, доцент кафедри КІІС		

7. Дата видачі завдання «_10_» _____01_____ 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2026	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2026	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2026	виконано
4	Робота над розділом 2 – вибір програмних і технічних компонентів для проєктування системи автоматизованого відеомоніторингу та підрахунку людей у приміщенні під час евакуації	01.04.2026	виконано
5	Робота над розділом 3 – проєктування системи автоматизованого відеомоніторингу, детекції, відстеження та підрахунку людей у приміщенні для контролю процесу евакуації	29.04.2026	виконано
6	Оформлення пояснювальної записки згідно вимог	24.05.2026	виконано
7	Попередній захист ВКР	25.05.2026	виконано
8	Захист ВКР на засіданні ЕК	Червень 2026 року	

Здобувач _____ Артем КОРОЛЬЧУК
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Марія КАПУСТЯН
Підпис Ім'я, ПРІЗВИЩЕ

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 250203.23.02.48 ПЗ	Пояснювальна записка	56		
			Графічні матеріали			
2		КвРКІ 250203.23.02.48 Е8	Структура програмного забезпечення системи контролю евакуації людей	1		
3		КвРКІ 250203.23.02.48 Е8	Схема взаємодії підсистем системи контролю евакуації людей	1		
4		КвРКІ 250203.23.02.48 Е8	Блок-схеми алгоритмів підрахунку людей та контролю евакуації	1		

КвРКІ 250203.23.02.48 ВП

Зм	Арк	№ докум	Підпис	Дата
Розробив		Корольчук		
Перевір.		Капустян		
Н. контр.		Лисенко		
Затв.		Павлова		01.06

Відомість проекту

Літера	Аркуш	Аркушів
У	1	1

ХНУ, КІ2с-23-2

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Кіберфізична система контролю евакуації людей з приміщень організації».

Автор роботи: Артем КОРОЛЬЧУК.

Керівник роботи: Марія КАПУСТЯН.

Пояснювальна записка: 56 с., 29 рис., 1 табл., 4 дод., 64 джерел.

Графічна частина: 3 креслення.

АРХІТЕКТУРА, ВЕБСИСТЕМА, ВІДЕОМОНІТОРИНГ, ЕВАКУАЦІЯ, ЖУРНАЛ ПОДІЙ, КОМП'ЮТЕРНИЙ ЗІР, ПІДРАХУНОК ЛЮДЕЙ.

Кваліфікаційна робота бакалавра присвячена розробці та дослідженню системи автоматизованого відеомоніторингу та контролю евакуації людей із приміщення на основі методів комп'ютерного зору. Актуальність теми зумовлена зростанням вимог до безпеки людей у будівлях та необхідністю оперативного контролю кількості осіб у приміщенні в умовах надзвичайних ситуацій. Своєчасне визначення кількості людей, які перебувають у приміщенні, входять до нього та виходять із нього, дає змогу підвищити ефективність організації евакуації, зменшити вплив людського фактора, скоротити час перевірки приміщень і знизити ризик того, що під час евакуації всередині можуть залишитися люди.

Метою роботи є проєктування, реалізація та тестування програмно-технічного засобу для виявлення, відстеження, підрахунку людей та візуалізації результатів моніторингу у реальному часі. Для досягнення поставленої мети було виконано аналіз сучасних підходів до контролю присутності людей і відеоаналітики, обрано програмні засоби реалізації, розроблено структурну та функціональну схеми системи, спроєктовано алгоритми підрахунку людей і контролю евакуації, а також реалізовано вебінтерфейс користувача для перегляду відеопотоку, статистичних показників і журналу подій.

30.05.2026

Підпис здобувача

Дата

ЗМІСТ

ВСТУП

У сучасних умовах функціонування організацій різного призначення, таких як офісні центри, навчальні заклади, промислові підприємства та громадські установи, особливого значення набуває забезпечення безпеки людей. Одним із найважливіших аспектів безпеки є своєчасна та ефективна евакуація людей у разі виникнення надзвичайних ситуацій, зокрема пожеж, техногенних аварій, задимлення або оголошення повітряної тривоги. У таких ситуаціях від швидкості та організованості евакуації безпосередньо залежить життя та здоров’я людей.

Незважаючи на наявність нормативних документів, інструкцій та планів евакуації, на практиці часто виникають труднощі з контролем повноти евакуації. Зокрема, відповідальні особи не завжди мають можливість оперативно перевірити всі приміщення та переконатися у відсутності людей. Це може бути пов’язано з великими розмірами об’єкта, обмеженим часом, панікою або іншими факторами, що ускладнюють процес перевірки. У результаті існує ризик того, що окремі люди можуть залишитися в небезпечній зоні без своєчасного виявлення.

Традиційні методи контролю евакуації переважно базуються на ручній перевірці приміщень або використанні списків присутності, що не забезпечує достатнього рівня оперативності та точності. У зв’язку з цим виникає потреба у впровадженні сучасних автоматизованих систем, які здатні здійснювати моніторинг присутності людей у приміщеннях та надавати актуальну інформацію відповідальним особам у режимі реального часу.

Одним із перспективних напрямів вирішення даної проблеми є використання кіберфізичних систем. Кіберфізичні системи являють собою інтеграцію фізичних компонентів, таких як датчики та пристрої збору інформації, з програмним забезпеченням, яке забезпечує обробку даних, аналіз та прийняття рішень. Завдяки цьому забезпечується взаємодія між фізичним середовищем та інформаційною системою.

Використання сучасних технологій, зокрема методів комп’ютерного зору, дозволяє автоматично визначати присутність людей у приміщеннях за допомогою відеокамер. Це дає можливість здійснювати підрахунок кількості людей, контролювати їх переміщення та визначати, чи всі особи залишили приміщення під час евакуації. Такий підхід дозволяє підвищити ефективність контролю евакуації, зменшити вплив людського фактора та забезпечити своєчасне реагування у разі виявлення людей, які залишилися у небезпечній зоні.

Розробка кіберфізичної системи контролю евакуації є актуальним завданням, оскільки її впровадження дозволить підвищити рівень безпеки в організаціях, автоматизувати процес контролю та забезпечити отримання достовірної інформації про стан евакуації. Крім того, використання сучасних інформаційних технологій відповідає загальним тенденціям цифровізації та автоматизації систем управління.

Таким чином, розробка кіберфізичної системи контролю евакуації людей з приміщень є важливим і актуальним завданням, спрямованим на підвищення рівня безпеки та ефективності реагування у надзвичайних ситуаціях.

1 КІБЕРФІЗИЧНА СИСТЕМА АВТОМАТИЗОВАНОГО ВІДЕОМОНІТОРИНГУ ТА КОНТРОЛЮ ЕВАКУАЦІЇ ЛЮДЕЙ З ПРИМІЩЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ ЩОДО ЇЇ УДОСКОНАЛЕННЯ

1.1 Аналіз проблеми контрою евакуації людей з приміщення

Забезпечення безпеки людей у приміщеннях є важливим завданням для сучасних установ і підприємств. Особливо актуальним це стає під час надзвичайних ситуацій, таких як пожежі, аварії або повітряні тривоги, коли необхідно швидко організувати евакуацію людей із небезпечної зони. При цьому важливо не лише забезпечити вихід людей, а й контролювати повноту евакуації, тобто визначити, чи всі особи залишили приміщення.

Евакуація зазвичай виконується відповідно до планів евакуації, які визначають маршрути руху та розташування виходів. Приклад типового плану евакуації людей із будівлі наведено на рисунку 1.1.

Рисунок 1.1 – Приклад плану евакуації людей з будівлі [51]

Після оголошення сигналу тривоги або виникнення небезпечної ситуації люди повинні залишити приміщення відповідно до визначених маршрутів. Загальний процес евакуації людей із приміщення представлено на рисунку 1.2.

Рисунок 1.2 – Процес евакуації людей з приміщення [52]

Однією з головних проблем є відсутність ефективних засобів контролю присутності людей після початку евакуації. У більшості випадків перевірка приміщень здійснюється вручну відповідальними особами, що є тривалим і небезпечним процесом, особливо у великих будівлях або в умовах задимлення. Додаткову складність створює людський фактор, оскільки в стресових ситуаціях людей можна не помітити.

Ще однією проблемою є відсутність оперативної інформації про кількість людей у приміщенні на момент початку евакуації. Більшість сучасних систем безпеки виконують лише функцію оповіщення та не забезпечують контроль фактичної присутності людей. Приклад системи оповіщення наведено на рисунку 1.3.

Рисунок 1.3 – Система оповіщення про надзвичайну ситуацію [53]

У зв’язку з цим виникає необхідність використання автоматизованих систем контролю евакуації, які дозволяють визначати кількість людей у приміщенні та контролювати процес їх виходу в режимі реального часу. Використання таких систем зменшує вплив людського фактора, підвищує точність контролю та скорочує час перевірки приміщень.

Особливо актуальним впровадження автоматизованих рішень є для об’єктів із великою кількістю людей, де ручний контроль є малоефективним.

Таким чином, проблема контролю евакуації потребує застосування сучасних технічних засобів, здатних забезпечити ефективний контроль присутності людей і підвищити рівень безпеки.

1.2 Аналіз існуючих методів контролю присутності людей у приміщенні

Для забезпечення безпеки в будівлях важливим є контроль присутності людей та їх переміщення, особливо під час надзвичайних ситуацій. Існує кілька методів контролю присутності людей, які відрізняються принципом роботи, точністю та складністю реалізації.

До основних методів контролю присутності людей у приміщенні належать:

1) пасивні інфрачервоні датчики руху;
2) інфрачервоні бар’єри;
3) системи контролю доступу з використанням електронних карт або RFID-міток;
4) тепловізійні камери;
5) системи розпізнавання звуку;
6) ультразвукові датчики;
7) мікрохвильові радари.

Далі розглянемо наведені методи більш детально.

Пасивні інфрачервоні датчики руху визначають зміну теплового випромінювання людини в зоні дії датчика. Вони є простими у використанні, мають невисоку вартість та широко застосовуються в охоронних системах і системах автоматичного освітлення. Основним недоліком такого підходу є те, що датчики реагують лише на рух і не дозволяють визначити кількість людей у приміщенні. Якщо людина перебуває без руху, система може не зафіксувати її присутність. Приклад пасивного інфрачервоного датчика руху наведено на рисунку 1.4.

Рисунок 1.4 – Пасивний інфрачервоний датчик руху [54]

Інфрачервоні бар’єри працюють за принципом переривання інфрачервоного променя між передавачем і приймачем. Це дозволяє фіксувати проходження людей через контрольну зону та вести облік входів і виходів. Такий метод забезпечує вищу точність порівняно з PIR-датчиками, однак при одночасному проходженні кількох людей результати можуть бути неточними. Приклад такого рішення наведено на рисунку 1.5.

Рисунок 1.5 – Інфрачервоний датчик підрахунку людей у дверному прорізі [55]

Системи контролю доступу з використанням електронних карт або RFID-міток передбачають ідентифікацію користувача під час входу та виходу з приміщення. Їх перевагою є можливість ведення точного обліку конкретних осіб, фіксації часу проходу та збереження історії переміщення користувачів у межах об’єкта. Такі системи широко використовуються в офісних будівлях, навчальних закладах, підприємствах та на об’єктах із обмеженим доступом. Водночас під час евакуації люди можуть залишати приміщення без використання карт, що унеможливлює визначення реальної кількості людей усередині будівлі. Крім того, система не дозволяє визначити точне місцезнаходження людини всередині

приміщення після проходу через контрольну точку. Приклад системи контролю доступу наведено на рисунку 1.6.

Рисунок 1.6 – Система контролю доступу з використанням електронної карти [56]

Тепловізійні камери визначають присутність людей за тепловим випромінюванням. Вони можуть ефективно працювати в умовах темряви або задимлення, що робить їх корисними під час пожеж або інших аварійних ситуацій. Основними недоліками є висока вартість обладнання та складність інтеграції в систему контролю. Приклад тепловізійної камери наведено на рисунку 1.7.

Рисунок 1.7 – Приклад тепловізійної камери для виявлення людей [57]

Системи розпізнавання звуку використовують мікрофони та алгоритми аналізу аудіосигналів для визначення присутності людей за звуками кроків, голосу або іншими шумами. Перевагою такого підходу є можливість роботи без камер та незалежність від рівня освітлення. Водночас на точність можуть впливати сторонні шуми, а за відсутності звукової активності присутність людини може не визначатися. Приклад такого підходу наведено на рисунку 1.8.

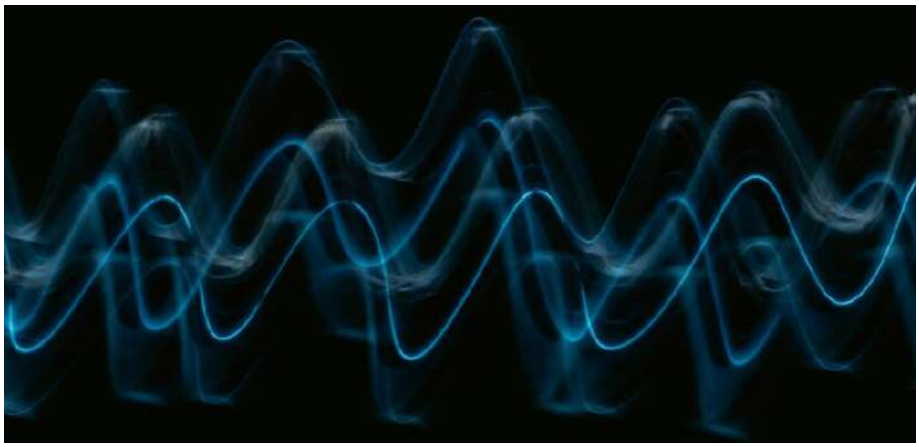


Рисунок 1.8 – Визначення присутності людей за допомогою звукових сенсорів [58]

Ультразвукові датчики визначають наявність людей за допомогою випромінювання ультразвукових хвиль та подальшого аналізу сигналу, відбитого від об’єктів у зоні контролю. На основі часу повернення відбитого сигналу система може визначати відстань до об’єкта та фіксувати його появу або переміщення в межах контрольованої області.

Такі датчики часто використовуються в системах автоматизації, охоронних системах, а також для контролю заповненості приміщень або окремих зон. До їх переваг належать невисока вартість, простота встановлення та використання, а також незалежність від рівня освітлення в приміщенні. Крім того, ультразвукові датчики можуть стабільно працювати як удень, так і в повній темряві. Недоліками такого підходу є обмежена зона дії, залежність точності від конфігурації приміщення та наявності сторонніх об’єктів, які можуть впливати

на відбиття сигналу. Також вони не забезпечують точного підрахунку кількості людей у разі перебування кількох осіб у зоні дії датчика. Приклад такого датчика наведено на рисунку 1.9.

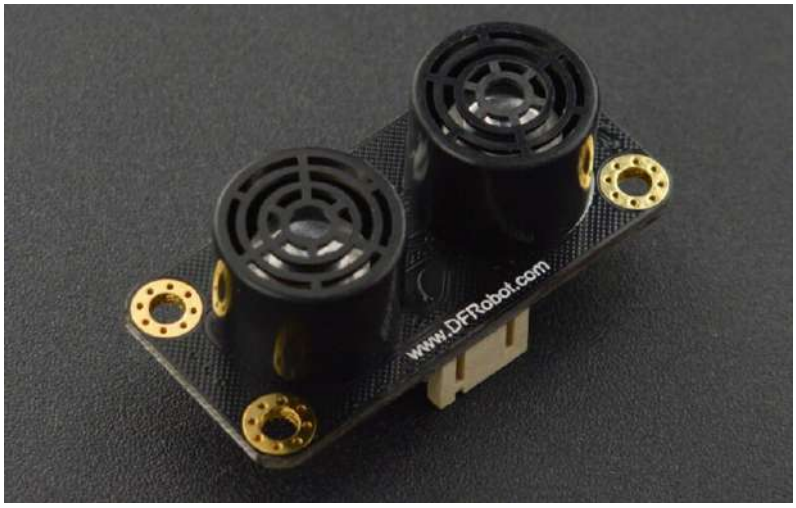


Рисунок 1.9 – Ультразвуковий датчик присутності людей [59]

Мікрохвильові радари працюють на основі ефекту Доплера та дозволяють виявляти рух і присутність людей шляхом аналізу відбитих радіохвиль. Принцип їх роботи полягає у випромінюванні високочастотного сигналу та аналізі змін частоти відбитого сигналу, які виникають при русі об’єктів у зоні дії радара. Завдяки цьому такі системи здатні визначати не лише сам факт присутності, але й характер руху об’єкта. Вони можуть працювати навіть через окремі неметалічні перешкоди, що робить їх ефективними в умовах обмеженої видимості, задимлення або складної конфігурації приміщень. Мікрохвильові радари широко застосовуються у сфері безпеки, рятувальних операціях, промислових системах моніторингу та спеціалізованих охоронних комплексах. Разом із цим вони мають нижчу точність підрахунку кількості людей у порівнянні з візуальними або комбінованими методами та можуть реагувати на сторонні рухи в зоні дії. Тому на практиці їх часто використовують у поєднанні з іншими сенсорами для підвищення точності системи. Приклад використання мікрохвильового радара наведено на рисунку 1.10.

Рисунок 1.10 – Використання портативного радарного пристрою для виявлення руху у роботі поліції [60]

Таким чином, існує значна кількість методів контролю присутності людей, кожен із яких має свої переваги та недоліки. Частина методів є простою у реалізації та має низьку вартість, проте не забезпечує достатньої точності. Інші методи є більш точними, але потребують значних фінансових затрат та складного обладнання. Це створює необхідність подальшого аналізу існуючих рішень з метою визначення найбільш ефективного методу контролю присутності людей у приміщенні.

1.3 Аналіз систем контролю евакуації на основі відеокамер

Системи контролю евакуації на основі відеокамер використовують засоби комп’ютерного зору та алгоритми штучного інтелекту для виявлення людей, підрахунку їх кількості та аналізу переміщення в межах приміщення. Основою таких рішень є безперервний відеомоніторинг зон евакуації, включаючи приміщення, коридори та виходи.

Отриманий відеопотік обробляється за допомогою бібліотек комп’ютерного зору, зокрема OpenCV, яка дозволяє виконувати виявлення руху, виділення об’єктів, аналіз контурів та відстеження їх траєкторій. На основі методів фонової сегментації відбувається виділення рухомих об’єктів, що дає змогу визначати присутність людей у кадрі та контролювати їх переміщення.

Для підвищення точності в сучасних системах застосовуються моделі глибокого навчання, такі як YOLO, SSD або OpenPose. Вони дозволяють розпізнавати людей навіть за умов часткового перекриття об’єктів, складного фону або зниження якості зображення. Крім того, такі моделі забезпечують визначення напрямку руху та щільності скупчення людей, що є важливим для аналізу процесу евакуації. Приклад роботи системи відеоаналізу для розпізнавання людей наведено на рисунку 1.11.

Рисунок 1.11 – Приклад розпізнавання людей на основі відеокамер [61]

Використання відеокамерних систем має низку переваг, серед яких можливість віддаленого моніторингу, автоматичний аналіз ситуації та оперативне реагування на надзвичайні події. Інтеграція з алгоритмами штучного інтелекту підвищує точність розпізнавання та дозволяє більш ефективно оцінювати процес евакуації в реальному часі.

Водночас такі системи мають певні обмеження. Їх ефективність залежить від умов освітлення, ракурсу встановлення камер та наявності прямої видимості об’єктів. Ускладнення також виникають при щільному скупченні людей або складній конфігурації приміщень. Додатково, обробка відеопотоку в режимі реального часу потребує значних обчислювальних ресурсів.

Попри зазначені обмеження, поєднання відеокамер із алгоритмами комп’ютерного зору та моделями штучного інтелекту є ефективним підходом до контролю евакуації, оскільки дозволяє здійснювати безперервний моніторинг та аналіз руху людей у надзвичайних ситуаціях.

1.4 Порівняльний аналіх існуючих рішень

Контроль присутності людей у приміщеннях є важливим елементом забезпечення безпеки, особливо під час надзвичайних ситуацій та евакуації. Ефективна система контролю повинна дозволяти визначати кількість людей у приміщенні, напрямок їх руху та оцінювати щільність скупчень у режимі реального часу. Існуючі методи відрізняються принципом роботи, точністю та складністю реалізації.

Пасивні інфрачервоні датчики є простими та недорогими рішеннями, однак не дозволяють визначати точну кількість людей у приміщенні та не фіксують нерухомі об’єкти. Інфрачервоні бар’єри забезпечують базовий підрахунок входів і виходів через дверні прорізи, проте їх точність знижується у випадках одночасного проходу кількох людей. Системи контролю доступу на основі електронних карт або RFID-міток дозволяють вести облік персоналу, однак під час евакуації не гарантують отримання актуальної інформації про фактичну кількість людей у приміщенні.

Ультразвукові та звукові сенсори також можуть використовуватися для виявлення присутності людей, однак їх ефективність залежить від конфігурації приміщення, наявності перешкод та рівня сторонніх шумів. Тепловізійні камери

та мікрохвильові радари здатні працювати в умовах недостатнього освітлення або задимлення, що є важливим під час надзвичайних ситуацій. Разом із тим висока вартість обладнання та складність інтеграції обмежують їх широке застосування. Нижче наведено приклад автоматизованого підрахунку людей у щільному потоці наведено на рисунку 1.12.

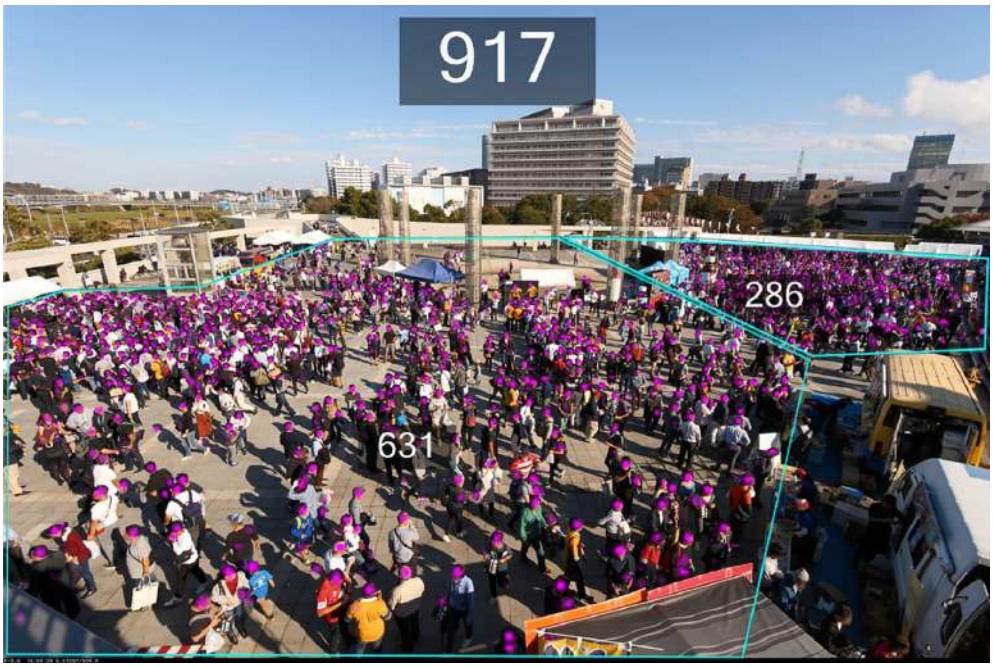


Рисунок 1.12 – Приклад підрахунку натовпу людей [62]

Найбільш перспективним підходом для контролю евакуації є використання систем на основі відеокамер та алгоритмів комп'ютерного зору. Такі системи дозволяють визначати присутність людей, аналізувати напрямок руху та оцінювати щільність скупчень у режимі реального часу. Для цього використовуються бібліотеки комп'ютерного зору, зокрема OpenCV, а також алгоритми розпізнавання об'єктів, такі як HOG, Haar Cascades або нейронні мережі. Це дозволяє виділяти людей у кадрі та відстежувати їх переміщення між окремими зонами спостереження.

Приклад розпізнавання людей за допомогою методів комп'ютерного зору наведено на рисунку 1.13.

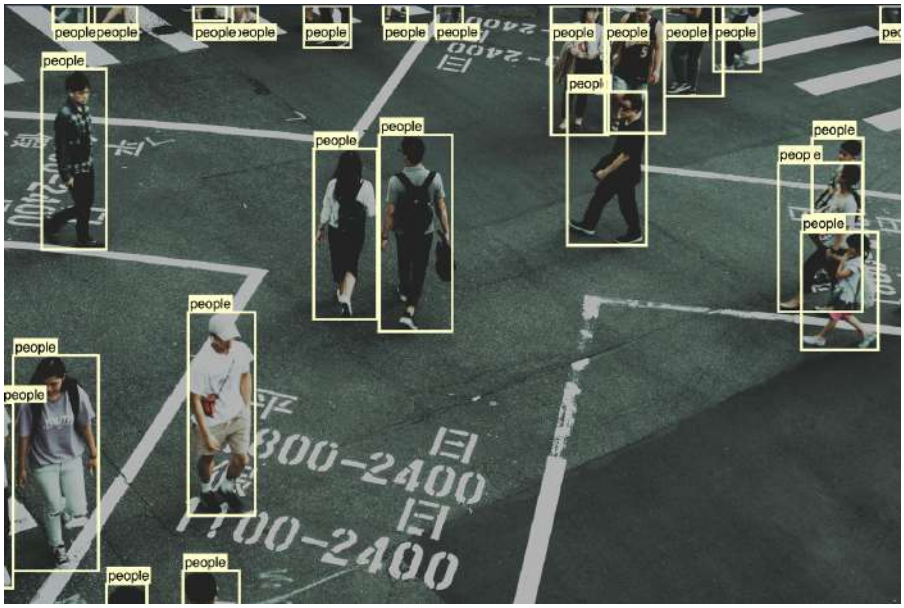


Рисунок 1.13 – Приклад розпізнавання людей за допомогою комп’ютерного зору [63]

Одним із поширених способів застосування відеоаналізу є підрахунок людей під час перетину контрольної лінії або визначеної зони на відеозображенні. У таких системах програмно задається віртуальна лінія, перетин якої використовується для фіксації входу або виходу людини. Це дозволяє автоматично вести облік кількості людей у приміщенні та контролювати зміну їх кількості в режимі реального часу.

Після виявлення людини в кадрі система відстежує її траєкторію між кадрами, визначає напрямок руху та виконує зміну лічильника присутніх осіб. Для реалізації такого підходу спочатку виконується виявлення рухомих об’єктів у кадрі, після чого застосовується класифікація для підтвердження, що виявлений об’єкт є людиною. Далі виконується відстеження об’єкта між кадрами та аналіз моменту перетину контрольної зони.

Додатково можуть використовуватись алгоритми, що запобігають повторному підрахунку однієї й тієї ж особи або дозволяють коректно працювати зі щільними групами людей, коли люди рухаються поруч або частково перекривають одна одну в кадрі. Такий підхід забезпечує вищу точність підрахунку порівняно з традиційними датчиками та дає змогу отримувати

оперативну інформацію про переміщення людей у приміщенні. Приклад такого підрахунку пішоходів із використанням контрольної лінії наведено на рисунку 1.14.

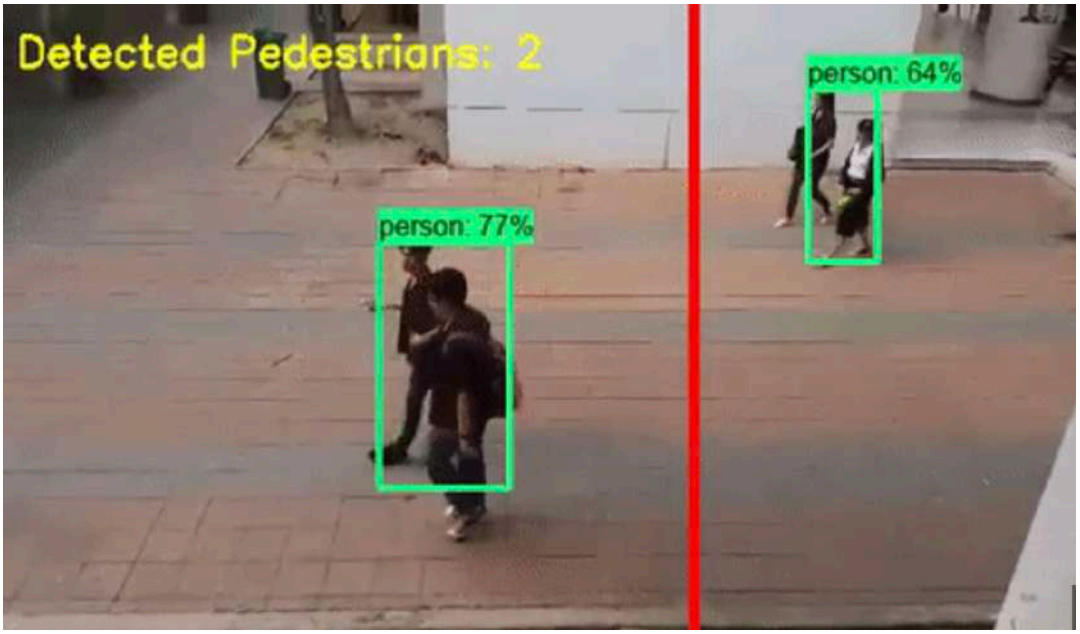


Рисунок 1.14 – Приклад підрахунку пішоходів [64]

Порівняльний аналіз показує, що традиційні методи контролю присутності людей не завжди забезпечують необхідну точність та оперативність для використання під час евакуації у великих або складно організованих приміщеннях. Використання відеокамер у поєднанні з алгоритмами обробки відеопотоку та комп'ютерного зору дозволяє поєднати точний підрахунок людей, аналіз траєкторій руху та роботу в режимі реального часу. Це робить такі системи найбільш перспективними для контролю евакуації та підвищення рівня безпеки людей у приміщенні.

1.5 Обґрунтування підходу до вирішення задачі

Оскільки основною метою дослідження є розробка системи контролю евакуації на основі підрахунку кількості людей, які входять і виходять з

приміщення, доцільним є використання існуючих камер відеоспостереження та методів комп’ютерного зору. Такий підхід дозволяє отримувати інформацію в режимі реального часу без необхідності встановлення додаткових спеціалізованих датчиків, що значно зменшує вартість впровадження системи та спрощує її інтеграцію в існуючу інфраструктуру будівлі.

Для реалізації обробки відеопотоку доцільно використовувати бібліотеку OpenCV, яка є однією з найпоширеніших відкритих бібліотек для комп’ютерного зору. Вона надає широкий набір функцій для захоплення відео з камери, обробки зображень, виявлення рухомих об’єктів, розпізнавання людей та відстеження їх переміщення. Використання OpenCV дозволяє створити систему, яка може автоматично визначати появу людини у кадрі, аналізувати її рух та фіксувати момент перетину контрольної зони.

Як програмне середовище доцільно використовувати мову програмування Python, оскільки вона має простий синтаксис, велику кількість бібліотек для обробки зображень та підтримує інтеграцію з OpenCV. Крім того, Python широко використовується у сфері штучного інтелекту та відеоаналітики, що дозволяє у майбутньому розширити функціональні можливості системи.

Принцип роботи системи полягає у отриманні відеопотоку з камери, його покадровій обробці та аналізі. На кожному кадрі система виконує виявлення людей, після чого відстежує їх переміщення у просторі. У зоні дверного прорізу або іншого контрольного проходу визначається віртуальна контрольна лінія.

Коли людина перетинає цю лінію, система фіксує напрямок її руху та відповідно збільшує або зменшує значення лічильника. Таким чином формується інформація про кількість людей, які знаходяться у приміщенні в даний момент часу.

Використання такого підходу дозволяє забезпечити автоматичний, безконтактний та точний контроль кількості людей у приміщенні, що є важливим для підвищення ефективності евакуації та забезпечення безпеки у надзвичайних ситуаціях.

1.6 Постановка задачі

Завданнями роботи є:

- провести аналіз особливостей процесу евакуації людей з приміщень та існуючих методів контролю присутності людей;
- обґрунтувати вибір апаратних і програмних засобів для побудови кіберфізичної системи, зокрема відеокамери, обчислювального пристрою та бібліотек комп'ютерного зору (наприклад, OpenCV);
- розробити алгоритм виявлення та підрахунку людей на основі відеопотоку з використанням методів комп'ютерного зору;
- розробити алгоритм контролю евакуації, який забезпечує визначення кількості людей, що залишили приміщення, та виявлення випадків невідповідності кількості евакуйованих осіб;
- реалізувати програмне забезпечення кіберфізичної системи контролю евакуації людей;
- провести тестування розробленої системи та оцінити ефективність її роботи;

На основі цього розробити працездатну кіберфізичну систему контролю евакуації людей з приміщень організації на базі відеокамери та методів комп'ютерного зору, яка забезпечить автоматичний підрахунок кількості людей, контроль процесу евакуації та виявлення приміщень, у яких можуть залишатися люди, у режимі реального часу.

1.7 Висновки до першого розділу

У межах першого розділу проведено аналіз проблеми контролю евакуації людей з приміщень та досліджено існуючі методи визначення присутності людей у приміщенні. Розглянуто переваги та обмеження різних технічних рішень, зокрема пасивних інфрачервоних датчиків, інфрачервоних бар'єрів, систем

контролю доступу на основі електронних карт, тепловізійних камер, мікрохвильових радарів та ультразвукових сенсорів. У результаті аналізу встановлено, що жоден із традиційних методів не забезпечує одночасно точного підрахунку людей, відстеження напрямків їх руху та роботи в режимі реального часу, що є важливим для ефективного контролю евакуації.

Особливу увагу приділено системам на базі відеокамер і методів комп'ютерного зору, які дозволяють автоматизувати процес виявлення людей у кадрі, здійснювати їх підрахунок у контрольних зонах та контролювати напрямок переміщення. Розглянуто принцип роботи систем підрахунку людей за допомогою відеоаналітики, зокрема використання контрольної лінії, виявлення рухомих об'єктів, розпізнавання людей та формування лічильників входу і виходу. Такий підхід має суттєву перевагу над традиційними сенсорними засобами, оскільки дає змогу використовувати вже наявні камери відеоспостереження, отримувати візуальне підтвердження результатів роботи системи та адаптувати алгоритми під особливості конкретного приміщення.

На основі проведеного аналізу визначено, що найбільш перспективним підходом до розв'язання поставленої задачі є використання існуючих камер відеоспостереження, алгоритмів комп'ютерного зору та програмних засобів обробки відеопотоку. Крім того, встановлено, що поєднання автоматизованого підрахунку людей із функцією контролю стану евакуації дозволяє не лише фіксувати присутність людей у приміщенні, а й підвищувати оперативність прийняття рішень у надзвичайних ситуаціях. Отримані результати стали основою для постановки задачі дипломної роботи та подальшої розробки системи автоматизованого відеомоніторингу і контролю евакуації людей із приміщення.

2 ПРОЄКТУВАННЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ АВТОМАТИЗОВАНОГО ВІДЕОМОНІТОРИНГУ ТА КОНТРОЛЮ ЕВАКУАЦІЇ ЛЮДЕЙ

2.1 Визначення апаратних і програмних підсистем програмно-технічного засобу

На основі проведеного аналізу предметної області та існуючих методів контролю присутності людей визначено склад програмно-технічного засобу для автоматизованого відеомоніторингу та контролю евакуації. Основною метою системи є підрахунок кількості людей у приміщенні та визначення повноти евакуації на основі аналізу відеопотоку. Для реалізації поставленої задачі система повинна забезпечувати контроль присутності людей у приміщенні та відображення результатів користувачу.

До складу програмно-технічного засобу входять апаратні та програмні підсистеми, які працюють у взаємозв’язку між собою. Апаратна частина забезпечує отримання та передавання відеоданих, тоді як програмна частина виконує їх обробку, аналіз та візуалізацію результатів. Такий підхід забезпечує автоматизований контроль присутності людей і підтримку процесу евакуації.

До складу апаратних підсистем належать відеокамера та обчислювальний пристрій. Відеокамера забезпечує безперервне отримання відеопотоку із зони контролю. Якість роботи системи при цьому залежить від характеристик камери, зокрема роздільної здатності, кута огляду та умов освітлення. Обчислювальний пристрій виконує аналіз отриманого відеопотоку, обробку результатів та роботу вебінтерфейсу. У його ролі може використовуватись персональний комп’ютер, ноутбук або інший пристрій із достатньою продуктивністю. Взаємозв’язок між основними програмними модулями та послідовність передавання даних між ними визначають логіку функціонування всієї системи. Загальну структуру програмного забезпечення системи автоматизованого відеомоніторингу та контролю евакуації людей наведено на рисунку 2.1.

Рисунок 2.1 – Структура програмного забезпечення системи автоматизованого відеомоніторингу та контролю евакуації людей

Як показано на рисунку 2.1, центральним елементом системи є аналітичне ядро, яке координує взаємодію між основними програмними модулями. Воно забезпечує обробку відеопотоку, запуск алгоритмів виявлення та відстеження людей, підрахунок кількості присутніх осіб і формування даних для подальшого відображення та збереження.

Програмна частина системи включає декілька взаємопов’язаних підсистем. Підсистема отримання та попередньої обробки відеопотоку відповідає за приймання відеоданих, зчитування кадрів та підготовку зображень до подальшого аналізу. Її функціями є масштабування кадрів, зміна формату та передавання даних до модуля комп’ютерного зору.

Однією з ключових складових є підсистема виявлення людей у кадрі, яка реалізується із використанням алгоритмів комп’ютерного зору та моделей

детекції об’єктів. Її завданням є визначення координат областей зображення, у межах яких виявлено людей. Для аналізу переміщення використовується підсистема відстеження об’єктів. Вона дозволяє пов’язувати виявлені об’єкти між послідовними кадрами, будувати траєкторії руху та уникати повторного врахування однієї й тієї ж людини.

Підсистема підрахунку людей виконує аналіз перетину віртуальної контрольної лінії та фіксує події входу або виходу. На основі цих даних формується поточна кількість людей у приміщенні. Логічним продовженням цієї складової є підсистема контролю евакуації, яка використовує результати підрахунку для визначення кількості людей, що залишилися всередині приміщення, а також для оцінки завершення евакуації та відображення відповідного статусу користувачу.

Для збереження результатів роботи використовується підсистема реєстрації подій, яка забезпечує фіксацію моментів входу та виходу людей, часу подій і поточного стану лічильників у журналі подій. Наявність такої підсистеми дозволяє не лише відображати інформацію в реальному часі, але й аналізувати історію роботи системи та оцінювати її ефективність.

Окремою складовою є підсистема інтерфейсу, реалізована у вигляді вебінтерфейсу. Вона забезпечує перегляд відеопотоку з результатами обробки, відображення кількості людей у приміщенні, статусу евакуації та журналу подій, а також керування режимами роботи системи.

Таким чином, програмно-технічний засіб автоматизованого відеомоніторингу та контролю евакуації людей складається з апаратних підсистем отримання та обробки відеоданих і програмних підсистем обробки відеопотоку, детекції та відстеження людей, підрахунку, контролю евакуації, реєстрації подій та вебвізуалізації результатів. Сукупність зазначених підсистем забезпечує реалізацію задачі автоматизованого контролю присутності людей і підтримки процесу евакуації.

2.2 Розробка структурної схеми та опис взаємодії підсистем

Логіка взаємодії підсистем програмно технічного засобу визначається після встановлення його апаратної та програмної структури. На попередньому етапі було описано склад системи та функціональне призначення окремих компонентів, тоді як у даному підрозділі розглядається організація інформаційних потоків між ними та принципи їх узгодженої роботи в межах єдиної архітектури. Це дозволяє перейти від структурного опису до аналізу фактичного функціонування системи як цілісного програмного рішення.

Розроблювана система автоматизованого відеомоніторингу та контролю евакуації людей належить до класу програмно технічних засобів, у яких обробка інформації реалізується через послідовне перетворення вхідних даних. При такій організації вихід кожного модуля використовується як вхід для наступного, що забезпечує безперервний ланцюг обробки та уточнення результатів.

У межах такої моделі реалізується поетапна обробка інформації, починаючи від первинного відеопотоку і завершуючи формуванням аналітичних показників. Вхідними даними системи є відеопотік, що надходить із камери спостереження або тестового відеофайлу. Далі він проходить через послідовність етапів обробки, на кожному з яких формуються проміжні результати, що уточнюють стан об’єктів у кадрі.

У підсумку система повинна забезпечувати користувачеві не лише відеозображення, а й структуровану інформацію щодо кількості осіб, які зайшли до приміщення, кількості осіб, які його залишили, поточної кількості людей усередині та стану процесу евакуації. Така організація дозволяє простежити повний маршрут проходження інформації між функціональними компонентами системи.

Отримані на кожному етапі результати формують послідовний ланцюг взаємодії підсистем, який відображає загальну архітектуру програмно

технічного засобу. Схему взаємодії підсистем програмно технічного засобу наведено на рисунку 2.2.

Рисунок 2.2 – Схема взаємодії підсистем програмно-технічного засобу

На схемі відображено рух даних від джерела відео через аналітичні модулі до вебінтерфейсу користувача. Система побудована як послідовний конвеєр обробки відеоданих.

Першим етапом є отримання та підготовка відеопотоку, де кадри зчитуються, приводяться до єдиного формату та передаються далі для обробки. Від коректної роботи цього модуля залежить стабільність усієї системи. Центральним елементом є аналітичне ядро, яке координує взаємодію модулів і забезпечує передачу кадрів до підсистем детекції, відстеження та підрахунку.

Модуль детекції виконує визначення людей у кадрі та формує координати об’єктів. Отримані дані передаються до модуля відстеження, який пов’язує об’єкти між послідовними кадрами та формує їх траєкторії. Це забезпечує відстеження конкретних осіб у часі без дублювання.

Далі виконується підрахунок перетинів контрольної лінії: при русі у напрямку входу збільшується лічильник входів, у протилежному напрямку —

виходів. На основі цих даних визначається поточна кількість людей у приміщенні.

Після активації режиму евакуації система додатково контролює кількість осіб, що залишилися всередині, та динаміку її зміни. На основі різниці між входами та виходами формується поточний стан евакуації.

Результати обробки передаються до допоміжних компонентів. Журнал подій фіксує всі входи та виходи з часовими мітками, а статистичний модуль формує дані для відображення. Вебінтерфейс забезпечує візуалізацію відеопотоку, статистики та журналу подій, а також керування системою, зокрема запуск аналізу, зміну параметрів і активацію режиму евакуації.

Таким чином, система реалізована як послідовний конвеєр обробки даних, де кожен модуль працює з результатами попереднього, що забезпечує структуровану та передбачувану поведінку програмного засобу.

2.3 Проєктування алгоритму виявлення, відстеження та підрахунку людей

Одним із ключових етапів розробки програмно-технічного засобу автоматизованого відеомоніторингу та контролю евакуації людей є проєктування алгоритму підрахунку, який забезпечує виявлення людей у відеопотоці, їх відстеження та визначення кількості осіб, що входять і виходять із приміщення. Завдання включає не лише детекцію об’єктів класу «людина», а й аналіз їх переміщення в послідовності кадрів із встановленням відповідності між детекціями, визначенням напрямку руху та фіксацією перетину контрольної лінії. Узагальнена послідовність роботи алгоритму включає отримання та обробку кадру, виявлення людей, відстеження об’єктів, аналіз траєкторій руху та оновлення лічильників входу і виходу, що наведено на рисунку 2.3.

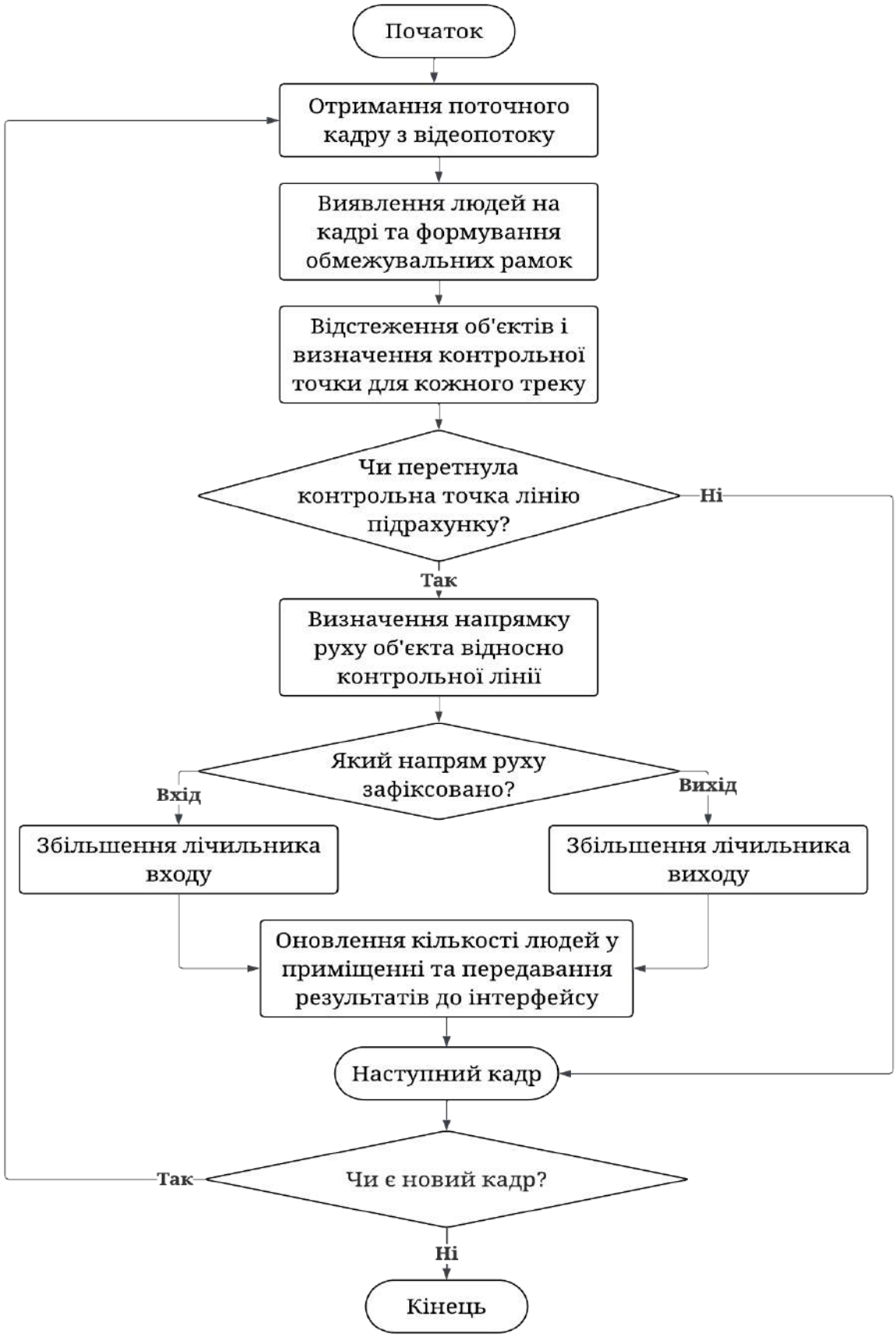


Рисунок 2.3 – Блок-схема алгоритму підрахунку людей за відеопотоком

Першим етапом роботи алгоритму є отримання відеокадру з джерела даних, яким може бути камера відеоспостереження або тестовий відеофайл. На цьому етапі здійснюється попередня підготовка зображення до аналізу: приведення кадру до потрібного формату, зміна розміру та перевірка коректності даних. Основним завданням цього етапу є забезпечення стабільного та безперервного надходження вхідних даних для подальшої обробки.

Другим етапом є виявлення людей у кадрі. Для цього використовуються алгоритми комп'ютерного зору та моделі детекції об'єктів, навчені розпізнавати клас «людина». Результатом роботи модуля є набір обмежувальних рамок, які визначають положення кожної людини в межах кадру, а також рівень достовірності розпізнавання. Це дозволяє відфільтровувати хибні спрацювання та формувати первинний опис поточного стану сцени.

Оскільки аналіз лише окремих кадрів не дає можливості визначити напрямок руху людини, наступним етапом є відстеження об'єктів у часі. Для кожної виявленої людини формується умовний ідентифікатор, який зберігається на послідовних кадрах доти, доки об'єкт перебуває в полі зору камери. Відстеження здійснюється шляхом порівняння поточних виявлень із даними попередніх кадрів з урахуванням координат і розмірів обмежувальних рамок. Це дозволяє формувати траєкторію руху кожної людини та уникати повторного підрахунку.

На наступному етапі виконується аналіз траєкторії руху об'єкта відносно контрольної лінії, яка відповідає місцю проходу людей через дверний проріз або вхід до приміщення. Для кожного об'єкта визначається контрольна точка, наприклад центр обмежувальної рамки. Якщо на попередньому кадрі ця точка знаходилась з одного боку контрольної лінії, а на поточному – з іншого, система фіксує факт перетину. Напрямок перетину визначає тип події: вхід або вихід із приміщення.

Після визначення напрямку руху система оновлює лічильники входу та виходу, а також формує поточну кількість людей у приміщенні. Важливою

складовою алгоритму є обробка ситуацій часткового перекриття людей або тимчасового зникнення об’єкта з кадру. У таких випадках модуль відстеження повинен зберігати траєкторію певний час навіть за відсутності нового виявлення, а модуль підрахунку має враховувати лише достатньо достовірні переходи через контрольну лінію. Це дозволяє підвищити стійкість роботи системи в реальних умовах експлуатації.

Після оновлення лічильників алгоритм формує поточний стан системи. На цьому етапі визначаються кількість людей, що увійшли до приміщення, кількість людей, що вийшли, поточна кількість осіб усередині та інформація про останню подію. Якщо активовано режим контролю евакуації, додатково визначається кількість людей, які ще залишаються в приміщенні, та стан завершеності евакуації.

Узагальнений алгоритм роботи системи можна подати у вигляді послідовності дій:

1) отримання кадру з джерела відеопотоку;
2) попередня обробка кадру;
3) виявлення людей у кадрі;
4) формування або оновлення траєкторій об’єктів;
5) визначення положення контрольної точки для кожного об’єкта;
6) перевірка факту перетину контрольної лінії;
7) визначення напрямку руху;
8) оновлення лічильників входу, виходу та поточної кількості людей;
9) формування стану системи контролю евакуації;
10) збереження інформації про події та передавання результатів до вебінтерфейсу.

Запропонований підхід має низку переваг. Він дозволяє виконувати автоматичний підрахунок людей без використання додаткових фізичних датчиків, використовуючи лише відеопотік із камер спостереження. Поєднання детекції та трекінгу забезпечує аналіз реального руху об’єктів, а можливість

налаштування контрольної лінії дозволяє адаптувати систему до різних умов експлуатації. Водночас точність роботи алгоритму залежить від якості відеопотоку, правильного розташування камери, рівня освітлення та стабільності роботи моделі детекції.

У межах розроблюваного програмно-технічного засобу доцільно передбачити можливість ручного налаштування положення контрольної лінії через вебінтерфейс користувача. Це забезпечує адаптивність системи до різних ракурсів камер і варіантів розташування проходів без необхідності зміни програмного коду.

Отже, проєктування алгоритму виявлення, відстеження та підрахунку людей базується на послідовному виконанні операцій детекції, трекінгу, аналізу траєкторії та фіксації факту перетину контрольної лінії. Запропонований підхід забезпечує автоматизоване формування інформації про кількість людей у приміщенні та створює основу для реалізації функції контролю евакуації в режимі реального часу.

2.4 Проєктування інформаційних ресурсів та структури збереження даних

Важливою складовою проєктування програмно-технічного засобу автоматизованого відеомоніторингу та контролю евакуації людей є визначення інформаційних ресурсів системи, способів їх формування, зберігання та використання. Інформаційні ресурси забезпечують фіксацію результатів роботи системи, формування її поточного стану, передавання даних до інтерфейсу користувача та подальший аналіз подій. Тому на етапі проєктування необхідно визначити структуру даних і принципи їх організації.

Особливе значення мають дані, пов’язані з режимом евакуації. У цьому режимі система повинна не лише реєструвати входи та виходи людей, а й визначати, чи всі особи залишили приміщення. Логіку роботи алгоритму контролю евакуації наведено на рисунку 2.4.

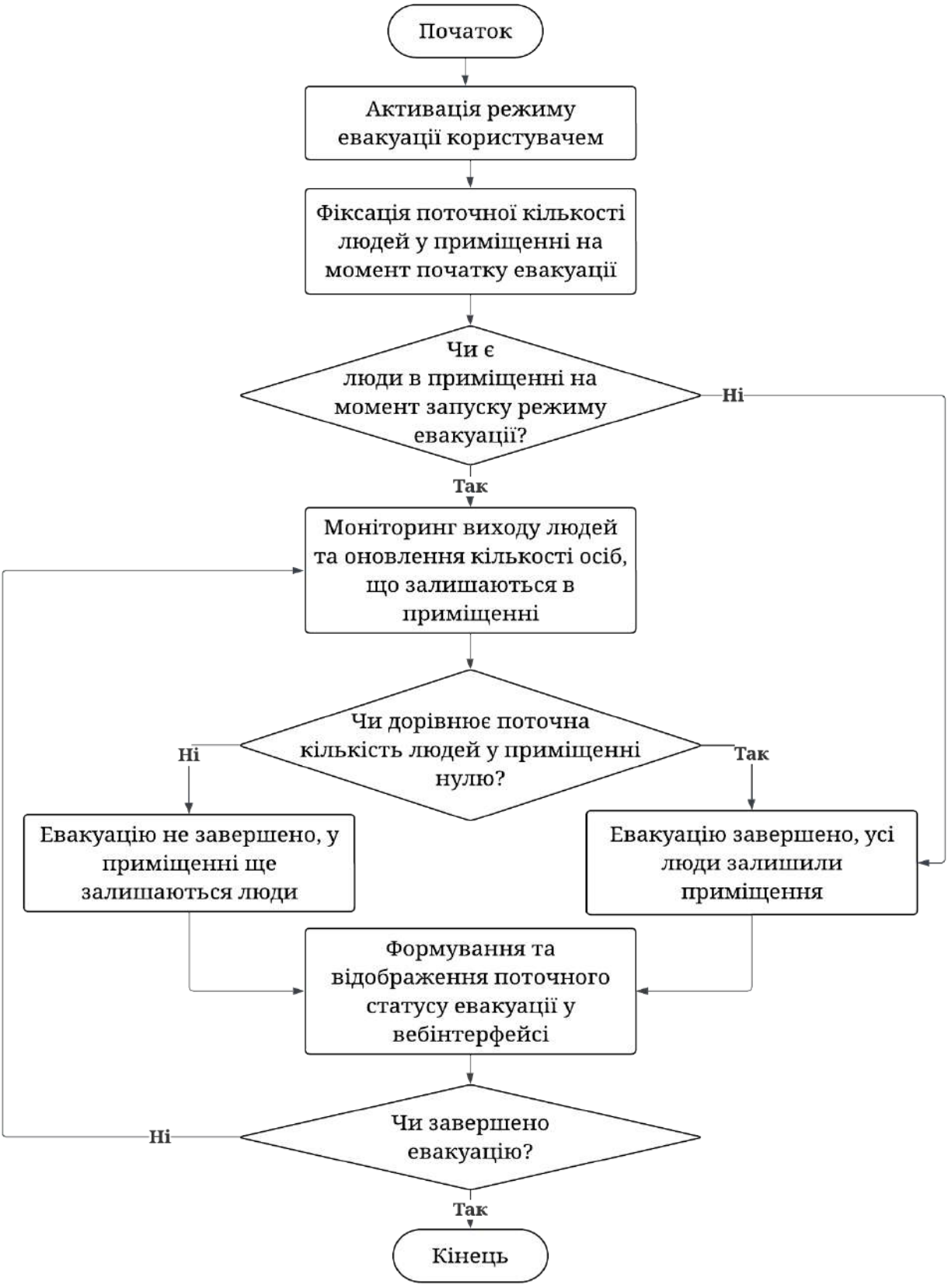


Рисунок 2.4 – Блок-схема алгоритму контролю евакуації

Після активації режиму евакуації система фіксує поточну кількість людей у приміщенні та виконує моніторинг їх виходу. Якщо людей у приміщенні немає, система одразу формує повідомлення про завершення евакуації. У протилежному випадку після кожної події виходу оновлюється кількість осіб, які залишаються всередині, та формується відповідний статус. Якщо поточна кількість людей стає нульовою, система фіксує завершення евакуації.

Інформаційні ресурси системи доцільно поділити на дві групи: оперативні та накопичувані дані. Оперативні дані використовуються під час поточної роботи системи та характеризують її стан у конкретний момент часу. До них належать значення лічильників входу і виходу, поточна кількість людей у приміщенні, службова інформація про відстежувані об’єкти та параметри режиму евакуації. Накопичувані дані призначені для довготривалого збереження результатів роботи системи. Основним прикладом таких даних є журнал подій, у якому фіксуються всі події, пов’язані з функціонуванням системи.

Поточний стан системи формується під час обробки відеопотоку та відображає основні показники: кількість людей, які увійшли та вийшли, поточну кількість осіб у приміщенні, стан режиму евакуації та кількість людей, які ще повинні залишити приміщення. Саме ці дані використовуються для оновлення вебінтерфейсу користувача та відображення результатів роботи системи.

Службова інформація про відстежувані об’єкти використовується для роботи алгоритмів детекції, трекінгу та підрахунку. До таких даних належать ідентифікатор об’єкта, координати його положення, траєкторія руху та параметри перевірки перетину контрольної лінії. Ці дані мають короткочасний характер і зберігаються лише під час обробки відеопотоку.

Основним ресурсом довготривалого збереження є журнал подій. Його призначення полягає у фіксації моменту виникнення події, типу події, поточної кількості людей у приміщенні та стану евакуації. Це дозволяє аналізувати історію функціонування системи та перевіряти коректність роботи алгоритмів.

Для збереження інформації доцільно використовувати табличний формат CSV. Такий вибір пояснюється простотою реалізації, відсутністю потреби у використанні повноцінної системи керування базами даних та можливістю подальшого аналізу даних у табличних редакторах. Структура запису журналу повинна містити дату і час події, ідентифікатор об’єкта, тип події, поточну кількість людей у приміщенні та стан режиму евакуації.

Інформаційні ресурси системи пов’язані з функціональними модулями програмного забезпечення. Дані про виявлені об’єкти формуються модулем детекції, передаються до модуля відстеження, а далі використовуються модулем підрахунку для оновлення поточного стану системи та формування записів журналу подій. Після цього результати відображаються у вебінтерфейсі користувача.

Важливою вимогою до структури збереження даних є її простота та достатність. Оскільки розроблюваний засіб має навчально-прикладний характер, у межах роботи не використовується повноцінна реляційна база даних, а довготривале збереження реалізується у вигляді журналу подій. Водночас така структура допускає можливість подальшої модернізації системи та перенесення даних до бази даних у майбутньому.

Отже, у процесі проєктування інформаційних ресурсів було визначено, що для функціонування системи необхідно використовувати оперативний стан системи, службові дані відстежуваних об’єктів та журнал подій. Обрана структура збереження даних є достатньою для розв’язання поставлених задач і відповідає архітектурі розроблюваної системи.

2.5 Вибір методів і засобів комп’ютерної інженерії для реалізації системи

Після визначення структури програмно-технічного засобу та основних алгоритмів необхідно обрати конкретні інструменти реалізації, які забезпечать обробку відеопотоку, виконання алгоритмів комп’ютерного зору та роботу

вебінтерфейсу. Вимоги до стеку включають підтримку роботи з відеоданими, можливість реалізації детекції та трекінгу об’єктів, достатню продуктивність для обробки кадрів у режимі, наближеному до реального часу, а також просту інтеграцію між програмними компонентами системи.

Базовою мовою реалізації обрано Python. Вибір зумовлений наявністю великої кількості бібліотек для обробки зображень, комп’ютерного зору та роботи з нейромережами, а також можливістю швидкої розробки та тестування алгоритмів. Python застосовується як основна мова для реалізації логіки обробки відеопотоку, взаємодії між модулями та організації загального робочого процесу системи.

Для роботи з відеоданими використовується бібліотека OpenCV. Вона забезпечує зчитування відеопотоку з камери або відеофайлу, попередню обробку кадрів, зміну розміру зображень, перетворення форматів та накладання графічних елементів поверх відео. Це дозволяє забезпечити узгоджену подачу кадрів у модулі детекції без втрати послідовності даних. Окремо OpenCV використовується для організації потоку кадрів у реальному часі та синхронізації етапів обробки.

Для задач виявлення людей застосовуються моделі сімейства YOLO. Їх використання обумовлено високою швидкістю роботи та достатньою точністю для задач аналізу відеопотоку. Модель повертає координати bounding box’ів, клас об’єкта та ймовірність детекції, що дозволяє виконувати фільтрацію результатів і зменшувати кількість хибних спрацювань у складних сценах з перекриттями або рухомим фоном.

Відстеження об’єктів між кадрами реалізується за допомогою алгоритмів трекінгу, які дозволяють зберігати ідентифікатори об’єктів у часі та формувати їх траєкторії руху. Це забезпечує можливість аналізу переміщення конкретної людини між кадрами та є основою для визначення факту перетину контрольної лінії.

Для математичної обробки координат і виконання операцій над масивами даних використовується NumPy. Бібліотека застосовується для розрахунку центрів об’єктів, обчислення відстаней між позиціями на різних кадрах та виконання операцій порівняння положень у рамках алгоритму трекінгу.

Серверна частина вебсистеми реалізується на основі Flask. Фреймворк використовується для організації API, передачі результатів обробки відеопотоку, керування режимами роботи системи та синхронізації даних між серверною та клієнтською частиною. Flask забезпечує мінімальну затримку при обміні даними та підходить для невеликих прикладних систем реального часу.

Клієнтська частина реалізується за допомогою HTML, CSS та JavaScript. HTML відповідає за структуру інтерфейсу, CSS за візуальне оформлення панелі керування, а JavaScript використовується для динамічного оновлення статистики, отримання даних з сервера та відправки команд без перезавантаження сторінки. Це дозволяє забезпечити роботу інтерфейсу в режимі наближеному до реального часу.

Для збереження результатів роботи системи використовується формат CSV. У ньому фіксуються події входу та виходу з часовими мітками, а також супровідна інформація про стан системи. Використання CSV дозволяє спростити збереження даних без впровадження окремої бази даних, а також забезпечує подальший аналіз результатів у табличних редакторах або аналітичних інструментах.

Вибраний стек характеризується модульною архітектурою, де кожен компонент виконує окрему функцію. OpenCV відповідає за відеообробку, YOLO за детекцію об’єктів, трекер за відстеження, NumPy за обчислення, Flask за серверну логіку, вебтехнології за інтерфейс користувача, а CSV за збереження подій. Такий підхід спрощує масштабування системи та дозволяє замінювати окремі модулі без зміни загальної архітектури.

Отже, обрані методи і засоби реалізації забезпечують виконання основних функцій системи: обробку відеопотоку, виявлення та відстеження людей,

підрахунок подій перетину контрольної лінії та відображення результатів через вебінтерфейс у режимі, наближеному до реального часу.

2.6 Висновки до другого розділу

У другому розділі визначено апаратні та програмні підсистеми програмно-технічного засобу автоматизованого відеомоніторингу та контролю евакуації людей, а також встановлено логіку їх взаємодії. У результаті проєктування сформовано структуру системи, що включає підсистеми обробки відеопотоку, детекції та відстеження людей, підрахунку входу і виходу, контролю евакуації, журналу подій і вебвізуалізації результатів.

У межах розділу розроблено структурну схему програмно-технічного засобу та визначено послідовність проходження інформації між його модулями. Встановлено, що функціонування системи базується на перетворенні вхідного відеопотоку в структуровану інформацію про кількість людей у приміщенні, напрямок їх руху та стан евакуації.

Також спроєктовано алгоритм виявлення, відстеження та підрахунку людей на основі відеопотоку. Визначено основні етапи його роботи: отримання кадру, детекція людей, формування траєкторій руху, перевірка перетину контрольної лінії та оновлення лічильників входу, виходу і поточної кількості людей у приміщенні. Окрему увагу приділено реалізації логіки контролю евакуації.

Крім того, обґрунтовано вибір основних методів і засобів комп’ютерної інженерії для реалізації системи. Для розробки доцільно використовувати мову програмування Python, бібліотеку OpenCV, моделі детекції YOLO, алгоритми трекінгу, мікрофреймворк Flask та стандартні вебтехнології. Отримані результати створюють основу для практичної реалізації програмно-технічного засобу у наступному розділі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ АВТОМАТИЗОВАНОГО ВІДЕОМОНІТОРИНГУ ТА КОНТРОЛЮ ЕВАКУАЦІЇ ЛЮДЕЙ

3.1 Опис реалізації модулів програмно-технічного засобу

На основі розроблених у попередньому розділі структурної схеми, алгоритмів функціонування та інтерфейсу було виконано програмну реалізацію системи автоматизованого відеомоніторингу та контролю евакуації людей. Реалізований програмно-технічний засіб забезпечує отримання відеопотоку, виявлення людей у кадрі, відстеження їх переміщення, підрахунок кількості осіб, які входять та виходять із приміщення, а також контроль стану евакуації з відображенням результатів у вебінтерфейсі.

Реалізація системи побудована за модульним принципом, що дозволило розділити програмне забезпечення на окремі функціональні частини. Такий підхід спрощує розробку, налагодження та подальшу модернізацію системи. До основних модулів належать: модуль отримання відеопотоку, модуль виявлення людей, модуль відстеження об’єктів, модуль підрахунку входу та виходу, модуль контролю евакуації, модуль збереження журналу подій, модуль вебсервера та модуль вебінтерфейсу користувача.

Модуль отримання та попередньої обробки відеопотоку забезпечує зчитування кадрів із відеофайлу або камери та їх підготовку до подальшого аналізу. Саме цей модуль відповідає за стабільне надходження відеоданих до аналітичної частини системи. Реалізація взаємодії між модулями передбачає централізовану обробку даних з подальшим передаванням результатів до інтерфейсної частини системи для візуалізації.

Для відображення загального стану системи та результатів роботи програмних модулів використовується вебінтерфейс користувача, що наведено на рисунку 3.1.

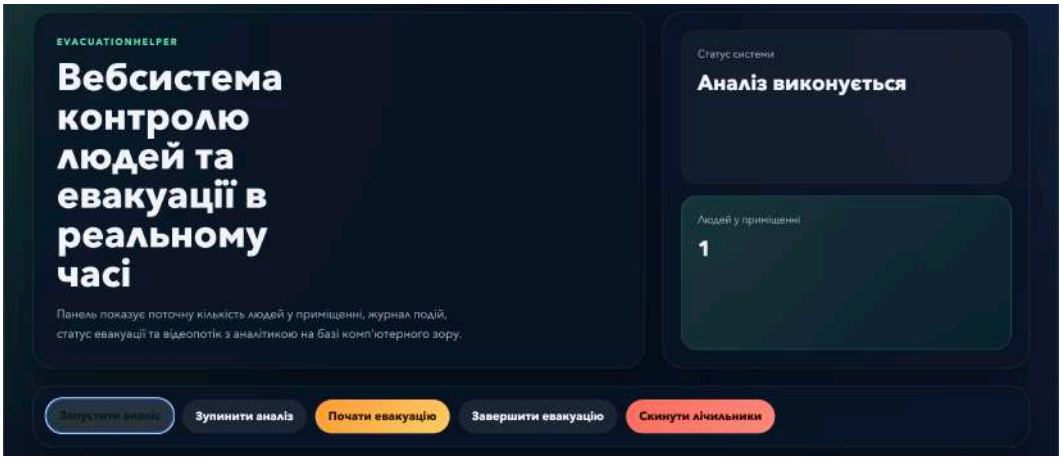


Рисунок 3.1 – Загальний вигляд вебінтерфейсу системи автоматизованого відеомоніторингу та контролю евакуації людей

Модуль виявлення людей реалізує використання моделі комп’ютерного зору для розпізнавання об’єктів класу «людина». На кожному кадрі система формує обмежувальні рамки навколо виявлених осіб. Після цього результати передаються до модуля відстеження, який зберігає траєкторії руху людей та присвоює кожному об’єкту унікальний ідентифікатор. Це дозволяє аналізувати переміщення людей у часі та уникати повторного підрахунку одного й того самого об’єкта.

Модуль підрахунку входу та виходу реалізує основну логіку системи. Для кожного відстежуваного об’єкта визначається контрольна точка та перевіряється її положення відносно контрольної лінії. Якщо точка змінює сторону відносно межі, система фіксує подію входу або виходу та оновлює значення лічильників і поточну кількість людей у приміщенні.

Окремим функціональним компонентом є модуль контролю евакуації. Після активації спеціального режиму система визначає кількість людей, які залишаються в приміщенні, та формує статус евакуації. Це дозволяє оператору оцінювати поточний стан приміщення під час евакуації. Результати роботи алгоритмів комп’ютерного зору, включаючи виявлення людей та накладання контрольної лінії, наведено на рисунку 3.2.

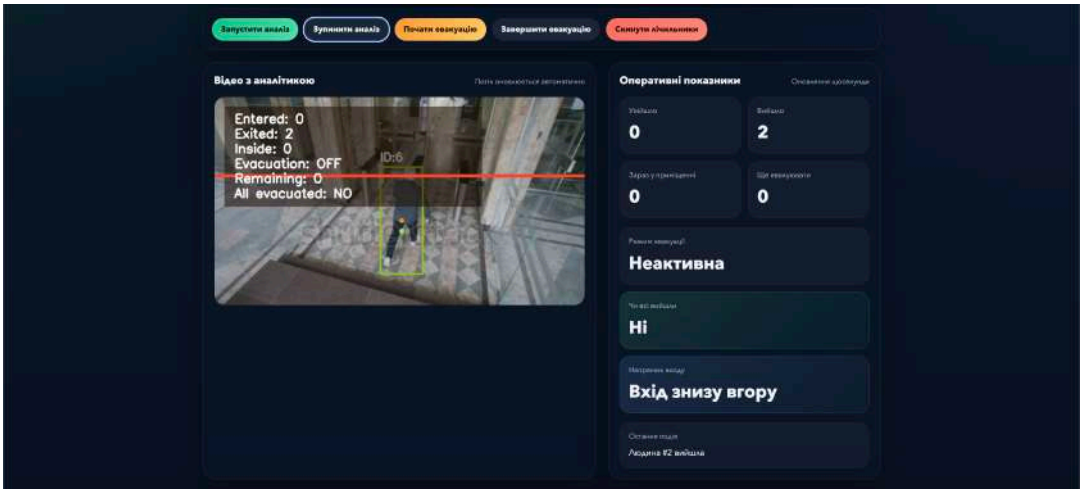


Рисунок 3.2 – Приклад відображення результатів виявлення та підрахунку людей у відеопотоці

Для фіксації результатів роботи реалізовано модуль збереження журналу подій. У журналі зберігаються відомості про час події, ідентифікатор об’єкта, напрямок руху та поточну кількість людей у приміщенні. Наявність цього модуля дозволяє аналізувати історію функціонування системи після завершення роботи.

Реалізація вебдоступу до системи здійснюється за допомогою модуля вебсервера, який забезпечує зв’язок між аналітичною частиною та вебінтерфейсом. Через нього передаються відеопотік, поточний стан системи та команди користувача. Модуль вебінтерфейсу призначений для відображення відеопотоку, статистичних показників, режиму евакуації та журналу подій. Крім того, у вебінтерфейсі реалізовано елементи керування, які дозволяють запускати та зупиняти аналіз, активувати режим евакуації, скидати лічильники та змінювати положення контрольної лінії.

У процесі реалізації особливу увагу приділено взаємодії між модулями. Модуль отримання відеопотоку передає кадри до аналітичної частини, далі модуль виявлення формує дані про знайдені об’єкти, які надходять до модуля відстеження. Після побудови траєкторій результати передаються до модуля підрахунку та контролю евакуації, а потім до журналу подій і вебінтерфейсу користувача. Візуалізація накладання результатів аналізу на відеопотік,

включаючи обмежувальні рамки, ідентифікатори об’єктів та контрольну лінію, наведена на рисунку 3.3.

Рисунок 3.3 – Елемент керування положенням контрольної лінії у вебінтерфейсі системи

Програмне забезпечення також забезпечує накладання результатів аналізу безпосередньо на відеокадр. На зображенні відображаються обмежувальні рамки навколо людей, ідентифікатори об’єктів і контрольна лінія підрахунку. Це підвищує наочність роботи системи для оператора.

Окремо реалізовано функцію ручного налаштування контрольної лінії через вебінтерфейс. Це дозволяє адаптувати систему до різних умов зйомки та розташування проходів без внесення змін до програмного коду.

Таким чином, у межах програмної реалізації створено сукупність взаємопов’язаних модулів, які забезпечують отримання відеопотоку, виявлення людей, відстеження їх руху, підрахунок кількості осіб у приміщенні, контроль стану евакуації та відображення результатів у вебінтерфейсі. Модульна структура підтверджує правильність рішень, прийнятих на етапі проєктування, та створює основу для подальшого тестування системи.

3.2 Опис програмної реалізації алгоритму підрахунку людей та контролю евакуації

Центральним елементом розробленого програмно-технічного засобу є алгоритм підрахунку людей та контролю евакуації, який забезпечує

автоматизоване визначення кількості осіб у приміщенні, а також фіксацію входів і виходів на основі аналізу відеопотоку. У процесі роботи алгоритм інтегрує модулі детекції, відстеження об’єктів та аналізу траєкторій руху.

Програмна реалізація базується на покадровій обробці відеопотоку, під час якої кожен кадр послідовно проходить етапи виявлення людей, формування траєкторій, аналізу положення контрольної точки та перевірки перетину контрольної лінії з подальшим оновленням лічильників.

Результат роботи алгоритму детекції об’єктів та формування обмежувальних рамок наведено на рисунку 3.4.

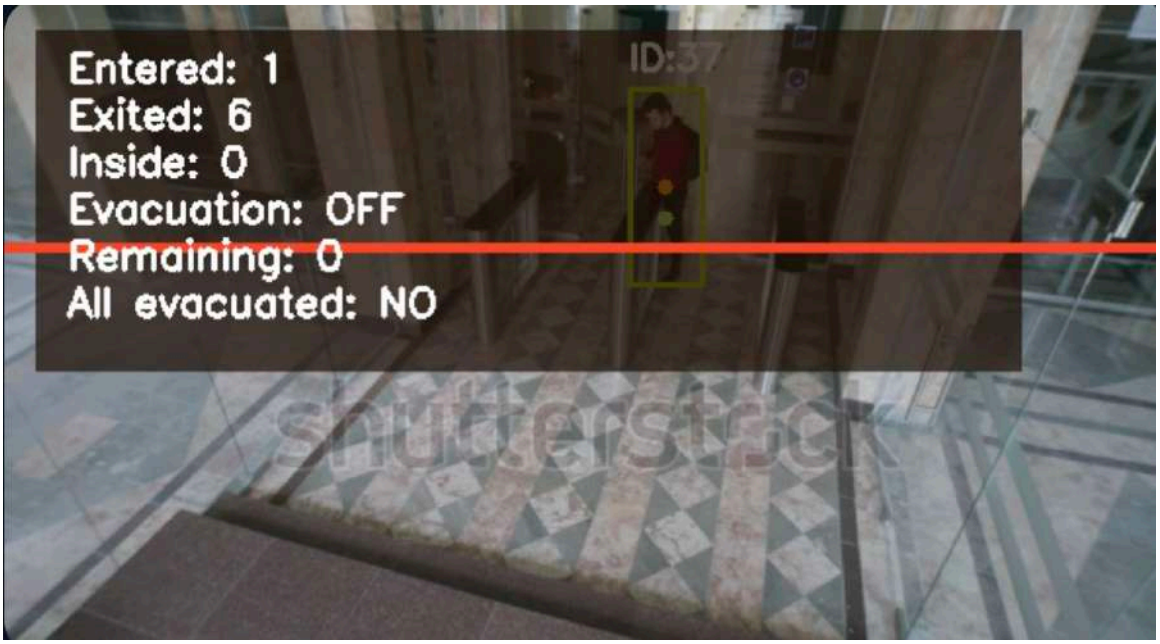


Рисунок 3.4 – Реалізація алгоритму виявлення та підрахунку людей у відеопотоці

На етапі детекції кадр обробляється моделлю розпізнавання об’єктів, яка формує координати обмежувальних рамок для класу «людина». Для зменшення кількості хибних спрацювань застосовується порогове значення впевненості.

Далі координати передаються до модуля відстеження, який забезпечує формування та супровід траєкторій руху об’єктів між кадрами, а також присвоєння кожному об’єкту унікального ідентифікатора. Це дозволяє

виключити повторний підрахунок та забезпечити коректне відстеження переміщення в часі.

Для кожного об’єкта визначається контрольна точка, положення якої відносно контрольної лінії використовується для фіксації факту перетину. У разі зміни сторони відносно межі система реєструє подію входу або виходу з оновленням відповідних лічильників та поточної кількості людей у приміщенні. Після формування подій алгоритм оновлює стан системи, включаючи кількість входів, виходів та поточну кількість осіб. У режимі евакуації додатково визначається кількість людей, які залишаються в приміщенні, та статус завершення евакуаційного процесу.

Результат відображення стану евакуації в інтерфейсі системи наведено на рисунку 3.5.

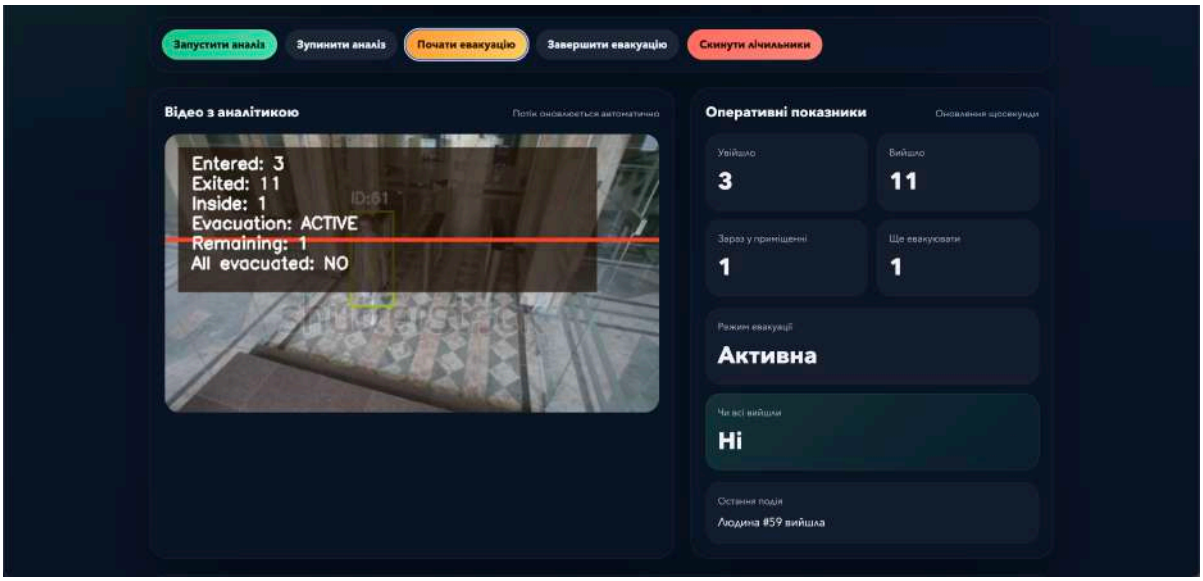


Рисунок 3.5 – Відображення результатів контролю евакуації у вебсистемі

Окремо реалізовано механізм формування журналу подій, у якому фіксуються час події, ідентифікатор об’єкта, напрямок руху та стан системи. Це забезпечує можливість подальшого аналізу роботи алгоритму.

Також реалізовано можливість зміни положення контрольної лінії через вебінтерфейс, що забезпечує адаптацію алгоритму до різних умов зйомки без

внесення змін у програмний код. Алгоритм контролю евакуації побудований на основі логіки підрахунку людей і використовує отримані дані для оцінки завершеності евакуації.

Таким чином, у межах програмної реалізації створено алгоритм, який забезпечує виявлення людей у відеопотоці, відстеження їх переміщення, підрахунок входу і виходу, формування поточної кількості людей у приміщенні та визначення стану евакуації. Реалізований підхід є основою функціонування всього програмно-технічного засобу та підтверджує можливість практичного використання системи для автоматизованого контролю присутності людей.

3.3 Опис реалізації вебінтерфейсу користувача

Важливою складовою програмно-технічного засобу є вебінтерфейс користувача, оскільки саме через нього оператор взаємодіє із системою автоматизованого відеомоніторингу та контролю евакуації людей. Вебінтерфейс забезпечує відображення результатів роботи системи та надає користувачеві засоби керування її режимами.

У межах реалізації було використано веборієнтований підхід, який дозволяє працювати із системою через звичайний браузер без встановлення додаткового програмного забезпечення. Це забезпечує зручність використання на різних пристроях та централізоване відображення результатів аналізу в одному середовищі.

З точки зору функціонального призначення вебінтерфейс реалізує набір дій, які може виконувати оператор під час роботи із системою. До них належать запуск і зупинка аналізу відеопотоку, перегляд відео та статистичних показників, активація і завершення режиму евакуації, зміна положення контрольної лінії, зміна напряму входу та виходу, скидання лічильників і перегляд журналу подій.

Вебінтерфейс реалізовано у вигляді інформаційної панелі, яка поєднує відеопотік, статистичні показники, елементи керування та результати підрахунку

людей. Центральним елементом є блок відеопотоку з накладеними результатами аналізу: рамками навколо людей, ідентифікаторами об’єктів, контрольною точкою та лінією підрахунку. Це дозволяє оператору контролювати правильність роботи алгоритму. Поруч відображаються статистичні показники: кількість входів, виходів, людей у приміщенні та осіб, яких необхідно евакуювати. Окремо реалізовано блок статусу евакуації.

У вебінтерфейсі також передбачено запуск і зупинку аналізу, скидання лічильників та керування режимом евакуації. Для налаштування системи реалізовано інтерактивний повзунок зміни положення контрольної лінії та можливість зміни напрямку підрахунку. Це дозволяє адаптувати систему до різних умов зйомки.

Вебінтерфейс створено за допомогою HTML, CSS та JavaScript. Дані оновлюються в режимі реального часу без перезавантаження сторінки. Для забезпечення можливості аналізу роботи системи та контролю коректності обробки відеопотоку реалізовано журнал подій, у якому фіксуються всі зареєстровані входи та виходи людей, а також супутня службова інформація. Приклад відображення журналу подій у вебінтерфейсі наведено на рисунку 3.7.

Рисунок 3.7 – Журнал подій у вебсистемі контролю евакуації людей

Таким чином, реалізований вебінтерфейс поєднує функції відображення відеопотоку, статистичних показників, контролю евакуації, журналювання подій та керування роботою системи. Побудова інтерфейсу у вигляді єдиної інформаційної панелі забезпечує зручність використання програмно-технічного засобу та підтримує його практичне застосування.

3.4 Інструкція користувача щодо роботи із системою

Для коректної експлуатації програмно-технічного засобу автоматизованого відеомоніторингу та контролю евакуації людей користувач повинен дотримуватись визначеної послідовності дій, яка включає запуск, налаштування та завершення роботи системи.

Перед початком роботи необхідно переконатися у наявності встановленого середовища Python, необхідних бібліотек комп'ютерного зору та веброзробки, а також моделі детекції людей. Як джерело відеоданих може використовуватись IP-камера або відеофайл. Доступ до системи здійснюється через веббраузер.

Запуск системи виконується шляхом ініціалізації серверної частини програмного засобу. Після запуску завантажується модель детекції, ініціалізується обробка відеопотоку та відкривається вебінтерфейс користувача.

Основний вигляд вебінтерфейсу з елементами керування та відображенням результатів роботи системи наведено на рисунку 3.8.

Після відкриття інтерфейсу користувач виконує налаштування положення контрольної лінії відповідно до геометрії проходу. Це дозволяє забезпечити коректний підрахунок входів і виходів у реальних умовах експлуатації.

У процесі роботи система відображає відеопотік, результати детекції та статистичні показники у реальному часі. Користувач має можливість контролювати поточний стан системи та за необхідності коригувати параметри роботи. Для аналізу історії подій використовується журнал подій, у якому фіксуються всі входи та виходи, часові мітки та службова інформація про роботу

системи. Відображення журналу подій у вебінтерфейсі системи наведено на рисунку 3.9.

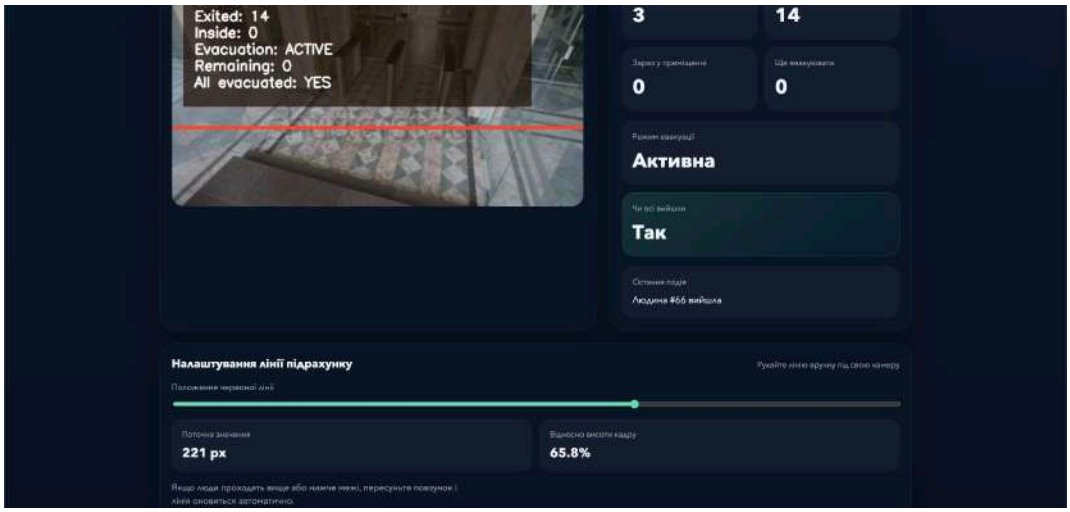


Рисунок 3.8 – Налаштування положення контрольної лінії у вебінтерфейсі системи

Рисунок 3.9 – Перегляд журналу подій у процесі роботи із системою

Журнал подій дозволяє відстежувати послідовність подій та перевіряти коректність роботи алгоритму підрахунку.

Для початку нового сеансу аналізу користувач може скористатися функцією скидання лічильників. Після цього очищуються статистичні показники та журнал подій, а система переходить у початковий стан.

Для завершення роботи оператор спочатку зупиняє обробку відеопотоку через вебінтерфейс, після чого завершує роботу серверної частини програми. Така послідовність дозволяє коректно завершити роботу системи та уникнути втрати даних.

Важливою умовою ефективної роботи системи є правильне розташування камери. Контрольована зона повинна добре проглядатися, а проходи людей

мають відповідати положенню контрольної лінії. За необхідності користувач повинен скоригувати положення камери або параметри системи.

Отже, робота користувача із системою включає запуск програмного засобу, відкриття вебінтерфейсу, налаштування контрольної лінії, запуск аналізу відеопотоку, використання режиму евакуації, перегляд журналу подій та завершення роботи системи. Наведена інструкція підтверджує придатність системи до практичного використання та простоту взаємодії з нею.

3.5 Технічні характеристики та умови використання системи

Для коректного функціонування програмно-технічного засобу автоматизованого відеомоніторингу та контролю евакуації людей необхідно визначити технічні характеристики системи та умови її використання. Вони встановлюють вимоги до апаратного і програмного середовища, у межах якого забезпечується стабільна робота алгоритмів комп’ютерного зору, обробки відеопотоку та вебінтерфейсу.

Система орієнтована на використання на персональному комп’ютері або ноутбуці, який має достатню обчислювальну потужність для покадрової обробки відеопотоку. Мінімально достатнім апаратним забезпеченням є сучасний багатоядерний процесор, не менше 8 ГБ оперативної пам’яті та наявність вільного місця на диску для збереження програмних компонентів і журналу подій. Для стабільнішої роботи рекомендується використовувати 16 ГБ оперативної пам’яті. За наявності графічного прискорювача швидкість роботи моделі детекції може бути вищою, проте базова реалізація системи може працювати і без GPU.

До складу необхідного обладнання входить джерело відеоданих камера відеоспостереження або відеофайл. Для коректної роботи алгоритму камера повинна забезпечувати достатню якість зображення, стабільну частоту кадрів та

належне освітлення контрольованої зони. Якість відеопотоку безпосередньо впливає на точність виявлення людей і підрахунку подій.

З боку програмного забезпечення система потребує операційної системи, яка підтримує Python і необхідні бібліотеки комп'ютерного зору та веброзробки. У межах реалізації можуть використовуватися Windows, Linux або macOS. Основною умовою є можливість запуску Python, OpenCV та локального вебсервера.

Для роботи системи необхідно встановити інтерпретатор Python і додаткові бібліотеки для обробки відео, реалізації алгоритмів детекції та трекінгу об'єктів, а також роботи вебінтерфейсу. Важливою умовою є наявність файлу моделі детекції людей, який використовується модулем комп'ютерного зору для аналізу відеопотоку.

Для взаємодії із системою використовується веббраузер, що підтримує HTML, CSS та JavaScript. Через браузер користувач відкриває локальний вебінтерфейс системи та працює з його елементами керування.

Ефективність роботи системи значною мірою залежить від умов спостереження. Камера повинна бути встановлена так, щоб зона проходу людей добре проглядалася, а контрольна лінія відповідала реальному напрямку переміщення осіб. Бажано уникати сильних перекриттів об'єктів, різких змін освітлення та сторонніх факторів, які можуть ускладнювати роботу алгоритму детекції.

Для коректного використання системи важливо дотримуватися правильної послідовності запуску. Спочатку запускається серверна частина, яка ініціалізує відеопотік і модель виявлення людей, після чого у браузері відкривається вебінтерфейс та запускається аналіз відео.

Однією з переваг системи є можливість локального функціонування без зовнішнього мережевого підключення. Оскільки вебінтерфейс працює через локальний сервер, система може використовуватися на одному комп'ютері або в межах локальної мережі без розгортання додаткової інфраструктури.

Система підтримує два режими роботи: аналіз відеопотоку в реальному часі та аналіз заздалегідь підготовленого відео. Це дозволяє використовувати її як для поточного моніторингу, так і для тестування або демонстрації роботи алгоритмів.

Отже, розроблена система має помірні технічні вимоги та може функціонувати на сучасному персональному комп'ютері або ноутбуці за умови наявності встановленого середовища Python, необхідних бібліотек, моделі детекції людей, джерела відеопотоку та веббраузера. Дотримання визначених умов забезпечує стабільну роботу алгоритмів і коректне функціонування програмно-технічного засобу.

3.6 Організація та методика тестування програмно-технічого засобу

Після завершення програмної реалізації системи автоматизованого відеомоніторингу та контролю евакуації людей було виконано її тестування з використанням реального відеоматеріалу. Як тестовий приклад застосовано відео 2_test.mp4, що містить інтенсивний потік людей у зоні входу через обертові двері. Вибраний фрагмент є інформативним для перевірки, оскільки характеризується одночасним рухом кількох осіб, частковими перекриттями об'єктів та зміною напрямків переміщення.

Метою тестування було підтвердження працездатності основних функцій програмно-технічного засобу, зокрема: запуску та роботи вебінтерфейсу, обробки відеопотоку, виявлення та відстеження людей, фіксації перетину контрольної лінії, формування лічильників входу та виходу, визначення поточної кількості осіб у приміщенні, а також ведення журналу подій. Додатково перевірялася коректність роботи механізмів ручного налаштування контрольної лінії та зміни напрямку підрахунку.

Тестування проводилось на відео тривалістю 14,9 с із роздільною здатністю 640×360 пікселів та частотою 29,97 кадр/с. Загальна кількість кадрів

становила 447. Камера була зафіксована всередині приміщення та спрямована на зону обертових дверей, що забезпечило стабільне спостереження за потоком людей. У межах тестового сценарію напрямок руху “вхід” визначався як переміщення зверху вниз у кадрі.

Методика перевірки передбачала послідовну оцінку повного циклу роботи системи. Спочатку виконувався запуск серверної частини та вебінтерфейсу, після чого активувався режим обробки відеопотоку. Далі перевірялася коректність детекції людей, формування обмежувальних рамок, відображення ідентифікаторів об’єктів, контрольної точки та контрольної лінії. На наступному етапі оцінювалась правильність фіксації подій входу та виходу, а також актуалізація статистичних показників Entered, Exited та Inside.

Окрему увагу приділено роботі системи в умовах щільного руху людей, характерного для реальних сценаріїв евакуації. Складність тестового відео полягала у частковому перекритті об’єктів та короткочасній втраті окремих людей у кадрі, що дозволило перевірити стійкість алгоритму відстеження.

Результати виконання алгоритмів комп’ютерного зору та загальну поведінку системи під час обробки тестового відео 2_test.mp4 наведено на рисунку 3.10.

Основна увага приділялась функціональності системи в умовах щільного потоку людей. Відео 2_test.mp4 є складним прикладом через груповий рух людей, часткові перекриття та короткочасне зникнення об’єктів із кадру. Це дозволило оцінити роботу системи в умовах, наближених до реальної експлуатації.

Отже, тестування системи базувалося на перевірці повного циклу її роботи: від запуску та налаштування параметрів до фіксації результатів підрахунку та їх відображення у вебінтерфейсі. Такий підхід дозволив оцінити працездатність програмно-технічного засобу та його придатність для задач контролю присутності людей і підтримки процесу евакуації.

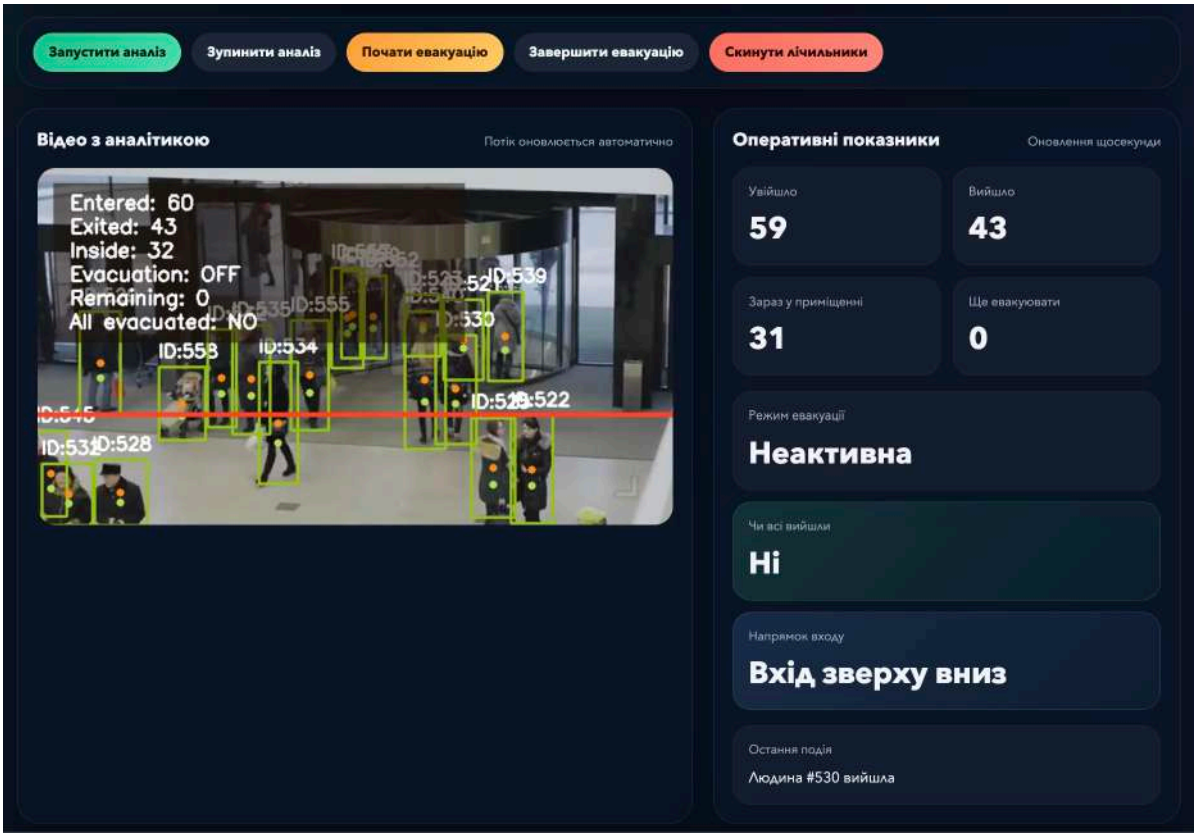


Рисунок 3.10 – Приклад роботи системи під час тестування на відео 2_test.mp4

3.7 Результати тестування системи

У результаті тестування на відеофайлі 2_test.mp4 було отримано практичні результати роботи розробленого програмно-технічного засобу в умовах інтенсивного переміщення людей.

Відео містить сцену з обертовими дверима, де одночасно спостерігаються значний потік людей, часткові перекриття об’єктів та різні напрямки руху. Це дозволило оцінити працездатність системи в умовах, наближених до реальної експлуатації. Отримані результати роботи програмно-технічного засобу під час аналізу відеофайлу 2_test.mp4 наведено на рисунку 3.11.

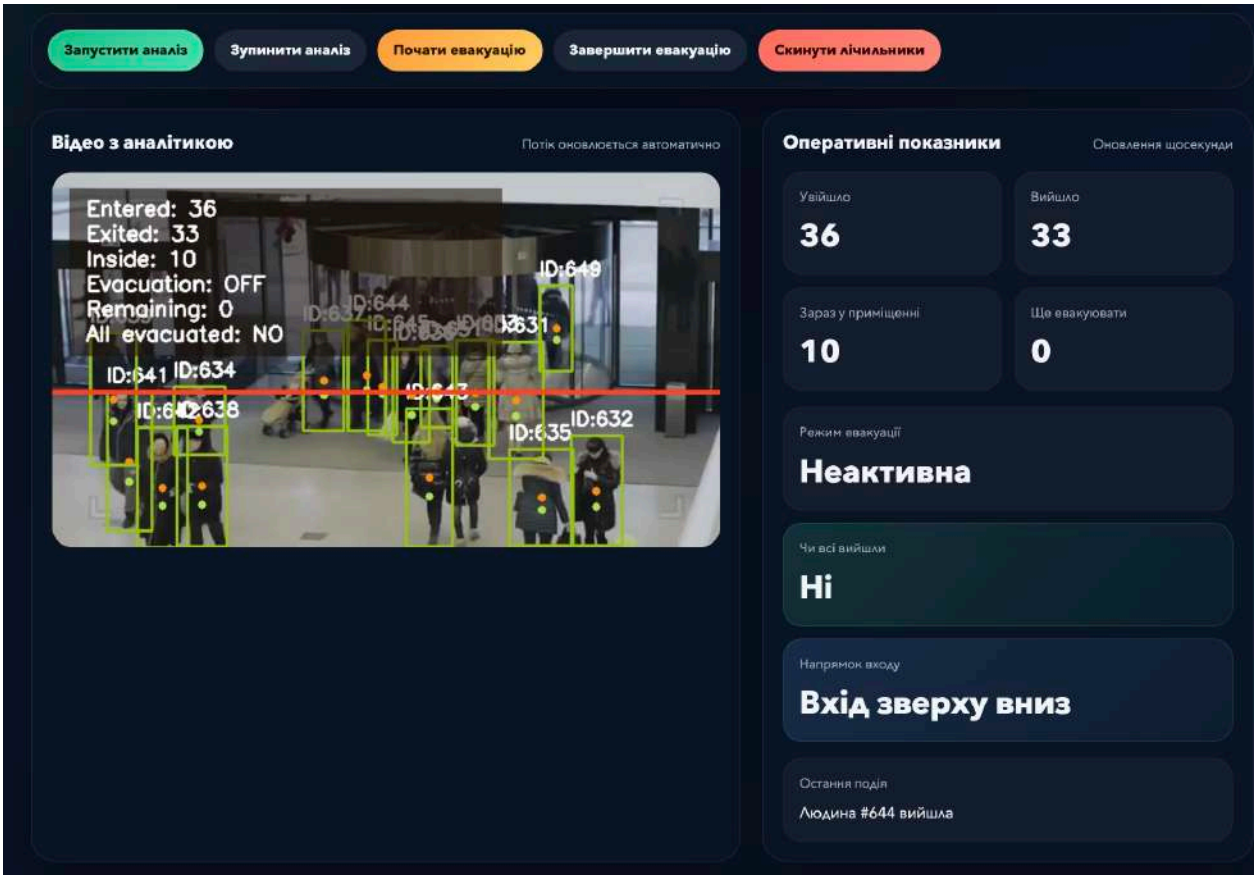


Рисунок 3.11 – Результат роботи системи під час тестування на відео 2_test.mp4

У межах цього тесту система виконувала виявлення людей у кадрі, відстеження їх переміщення, фіксацію перетину контрольної лінії та оновлення основних статистичних показників. Наприкінці проходження відео система сформувала такі оперативні показники: Увійшло - 36, Вийшло - 33, Зараз у приміщенні - 10. Отримане співвідношення відповідає загальному візуальному сприйняттю сцени, оскільки у даному відеофрагменті потік людей, що входять до приміщення, є більшим за потік людей, які виходять.

Це свідчить про те, що система правильно відображає загальну тенденцію руху людей у кадрі. Для більшої наочності основні результати тестування наведено в таблиці 3.1.

Аналіз результатів показав, що система реалізує основні функції, передбачені технічним завданням. Вона забезпечує обробку відеопотоку, виявлення та відстеження людей, підрахунок подій входу і виходу, визначення поточної кількості людей у приміщенні та відображення результатів у

вебінтерфейсі. Також було підтверджено коректну роботу додаткових функцій інтерфейсу, зокрема зміни напряму підрахунку та збереження положення контрольної лінії.

Таблиця 3.1 – Основні результати тестування системи на відео 2_test.mp4

Параметр перевірки	Отриманий результат
Відкриття вебінтерфейсу	виконано успішно
Запуск аналізу відео	виконано успішно
Відображення відеопотоку	працює коректно
Виявленн людей у кадрі	працює
Відстеження об’єктів	працює
Кількість зафіксованих входів	20
Кількість зафіксованих виходів	8
Кількість людей у приміщенні після завершення тесту	12
Формування журналу подій	працює коректно
Зміна напрямку входу/виходу	працює
Зміна положення контрольної лінії	працює
Збереження положення лінії після повторного відтворення відео	працює

Разом із тим слід враховувати, що відео 2_test.mp4 є складним для аналізу через високу щільність потоку людей, часткові перекриття та наявність обертових дверей. У таких умовах точність визначення окремих траєкторій може залежати від параметрів налаштування системи та особливостей сцени. Проте результати тестування підтвердили працездатність розробленого програмно-технічного засобу та можливість його використання для автоматизованого контролю кількості людей і підтримки процесу евакуації.

3.8. Висновки до третього розділу

У третьому розділу було виконано програмну реалізацію системи автоматизованого відеомоніторингу та контролю евакуації людей та проведено її тестування на основі реального відеоматеріалу. У результаті створено працездатний програмно-технічний засіб, який забезпечує обробку відеопотоку, виявлення людей, відстеження їх переміщення, підрахунок входу та виходу, визначення поточної кількості людей у приміщенні та відображення результатів через вебінтерфейс користувача.

У розділі описано реалізацію основних програмних модулів системи: отримання відеопотоку, детекції людей, відстеження об’єктів, підрахунку входу та виходу, контролю евакуації, журналу подій та вебінтерфейсу. Особливу увагу приділено реалізації алгоритму підрахунку людей на основі перетину контрольної лінії, а також механізму зміни напряму підрахунку та ручного налаштування положення контрольної межі.

Крім того, було розглянуто порядок використання системи, визначено технічні характеристики та умови її експлуатації. Це підтвердило, що розроблений засіб є придатним до практичного використання завдяки наявності зрозумілого вебінтерфейсу, елементів керування та засобів візуального контролю результатів роботи.

Проведене тестування на відео 2_test.mp4 підтвердило працездатність системи та правильність реалізації її основних функцій. Було встановлено, що система успішно виконує аналіз відео, відображає результати у вебінтерфейсі, веде підрахунок входу та виходу людей, формує журнал подій і дозволяє змінювати параметри підрахунку без редагування програмного коду. Отримані результати показали можливість роботи системи в умовах реальної сцени з інтенсивним потоком людей.

Разом із тим тестування показало, що в складних умовах при щільному потоці людей, часткових перекриттях та складній геометрії проходу, точність

підрахунку може знижуватися. Це свідчить про можливість подальшого вдосконалення системи, зокрема в частині підвищення точності відстеження об’єктів та стабільності роботи алгоритму.

Отримані результати показали можливість роботи системи в умовах реальної сцени з інтенсивним потоком людей. Отримані результати можуть бути використані як основа для подальшої оптимізації алгоритмів комп’ютерного зору та підвищення точності роботи системи в умовах складних сцен.

Отже, у третьому розділі підтверджено, що розроблений програмно-технічний засіб реалізує основні вимоги системи автоматизованого контролю кількості людей і підтримки процесу евакуації. Результати програмної реалізації та тестування свідчать про досягнення поставленої мети дипломної роботи та підтверджують перспективність подальшого розвитку системи для практичного використання.

ВИСНОВКИ

У кваліфікаційній роботі за результатами виконаних теоретичних і практичних досліджень розв’язано задачу розробки програмно-технічного засобу автоматизованого відеомоніторингу та контролю евакуації людей із приміщення. Актуальність роботи зумовлена необхідністю підвищення безпеки людей у будівлях та зменшення залежності від ручного контролю під час евакуації. У більшості випадків перевірка того, чи всі люди залишили приміщення, виконується відповідальними особами вручну, що потребує часу, збільшує навантаження на персонал і не виключає впливу людського фактора. Саме тому розробка системи, яка автоматично підраховує людей за відеопотоком і дозволяє контролювати повноту евакуації, та є доцільною.

Метою роботи було створення системи, яка на основі аналізу відеоданих забезпечує визначення кількості людей, що входять у приміщення та виходять із нього, обчислення поточної кількості осіб усередині, а також перевірку того, чи всі люди залишили приміщення під час евакуації. Для досягнення цієї мети було виконано аналіз предметної області, обґрунтовано вибір підходів і засобів реалізації, спроєктовано структуру програмно-технічного засобу, реалізовано його програмну частину та проведено тестування на реальному відеоматеріалі.

У першому розділі проведено аналіз проблеми контролю евакуації людей із приміщень та досліджено існуючі методи визначення присутності людей. Було розглянуто різні технічні рішення, зокрема інфрачервоні датчики, бар’єри, системи контролю доступу, тепловізійні та інші засоби. У результаті аналізу встановлено, що традиційні методи не забезпечують одночасно достатньої точності підрахунку, визначення напрямку руху та роботи в режимі реального часу. Це дозволило обґрунтувати доцільність використання систем відеоспостереження та методів комп’ютерного зору як найбільш перспективного підходу для вирішення поставленої задачі.

У другому розділі виконано проєктування програмно-технічного засобу автоматизованого відеомоніторингу та контролю евакуації людей. Було

визначено основні апаратні та програмні підсистеми, описано способи їх взаємодії, спроєктовано структуру програмного забезпечення, алгоритм виявлення, відстеження та підрахунку людей, а також інформаційні ресурси та людино-машинний інтерфейс системи. Крім того, обґрунтовано вибір основних засобів реалізації, серед яких Python, OpenCV, сучасні моделі детекції людей, алгоритми трекінгу, Flask та вебтехнології. У другому розділі сформовано логічну та технічну основу для практичної реалізації системи.

У третьому розділі здійснено програмну реалізацію розробленого засобу та проведено його тестування. Було реалізовано модулі обробки відеопотоку, виявлення людей, відстеження об’єктів, підрахунку входу та виходу, контролю евакуації, журналювання подій та вебінтерфейсу користувача. Особливу увагу приділено створенню зручного інтерфейсу, який дозволяє переглядати відеопотік з результатами аналізу, статистичні показники, журнал подій, а також змінювати положення контрольної лінії та напрямок підрахунку залежно від конкретного відео. Проведене тестування підтвердило працездатність системи та правильність реалізації її основних функцій.

Отримані результати показали, що розроблений програмно-технічний засіб забезпечує автоматизований підрахунок людей у приміщенні, дозволяє оцінювати поточну кількість осіб усередині та підтримує контроль процесу евакуації. Водночас тестування на сценах із щільним потоком людей, частковими перекриттями та складною геометрією проходу показало, що точність підрахунку в таких умовах може знижуватися. Це свідчить про те, що система є функціонально придатною та працездатною, однак має потенціал для подальшого вдосконалення, насамперед у частині підвищення стійкості алгоритмів відстеження та підрахунку.

Практичне значення роботи полягає в тому, що впровадження такого програмно-технічного засобу дозволяє скоротити витрати часу на ручну перевірку приміщень, зменшити вплив людського фактора та підвищити оперативність отримання інформації про кількість людей у небезпечній зоні.

Розроблена система може бути корисною для адміністративних будівель, навчальних закладів, офісних центрів, торговельних приміщень та інших об’єктів, де важливим є контроль присутності людей і перевірка повноти евакуації.

Перспективами подальшого розвитку роботи є підвищення точності алгоритму в складних сценах, використання більш стійких методів трекінгу, підтримка декількох камер, додавання бази даних для збереження історії подій та розширення аналітичних можливостей системи. Отже, у роботі досягнуто поставленої мети, а розроблений програмно-технічний засіб може розглядатися як основа для подальшого розвитку систем автоматизованого контролю кількості людей і підтримки евакуації в приміщеннях.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. OpenCV. People Detection and Tracking. URL: https://docs.opencv.org/4.x/d7/d00/tutorial_meanshift.html (дата звернення: 16.02.2026).

2. PyImageSearch. Pedestrian Detection with OpenCV and Python. URL: https://www.pyimagesearch.com/2015/11/09/pedestrian-detection-opencv/ (дата звернення: 17.02.2026).

3. OpenCV. Background Subtraction Techniques. URL: https://docs.opencv.org/4.x/d1/dc5/tutorial_background_subtraction.html (дата звернення: 17.02.2026).

4. LearnOpenCV. YOLOv8 Object Tracking and Counting with OpenCV. URL: https://learnopencv.com/yolov8-object-tracking-and-counting-with-opencv/ (дата звернення: 17.02.2026).

5. Canon Global. Counting People in Crowds with AI. URL: https://global.canon/en/technology/count2019.html (дата звернення: 17.02.2026).

6. RoboFlow. People Counting Using Computer Vision. URL: https://blog.roboflow.com/people-counting-computer-vision-software/ (дата звернення: 17.02.2026).

7. Geeks For Geeks. Pedestrian Detection using OpenCV-Python. URL: https://www.geeksforgeeks.org/python/pedestrian-detection-using-opencv-python/ (дата звернення: 17.02.2026).

8. Bewley A., Ge Z., Ott L., Ramos F., Upcroft B. Simple Online and Realtime Tracking. *2016 IEEE International Conference on Image Processing (ICIP)*. 2016. P. 3464–3468. DOI: https://doi.org/10.1109/ICIP.2016.7533003.

9. Wojke N., Bewley A., Paulus D. Simple Online and Realtime Tracking with a Deep Association Metric. *arXiv*. 2017. URL: https://arxiv.org/abs/1703.07402 (дата звернення: 10.04.2026).

10. Conte D., Foggia P., Percannella G., Tufano F., Vento M. A Method for Counting Moving People in Video Surveillance Videos. *EURASIP Journal on Advances in Signal Processing*. 2010. Vol. 2010. Article 231240. DOI: https://doi.org/10.1155/2010/231240.

11. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. *arXiv*. 2016. URL: https://arxiv.org/abs/1506.02640 (дата звернення: 10.04.2026).

12. Hassen K. B. A., Machado J. J. M., Tavares J. M. R. S. Convolutional Neural Networks and Heuristic Methods for Crowd Counting: A Systematic Review. *Sensors*. 2022. Vol. 22, No. 14. Article 5286. DOI: https://doi.org/10.3390/s22145286.

13. Szczodrak M., Czyzewski A. Video analytics-based algorithm for monitoring egress from buildings. *Multimedia Tools and Applications*. 2016. Vol. 75. P. 10733–10743. DOI: https://doi.org/10.1007/s11042-014-2143-7.

14. Deng H., Ou Z., Zhang G., Deng Y., Tian M. BIM and Computer Vision-Based Framework for Fire Emergency Evacuation Considering Local Safety Performance. *Sensors*. 2021. Vol. 21, No. 11. Article 3851. DOI: https://doi.org/10.3390/s21113851.

15. Ultralytics YOLO Documentation. Introducing Ultralytics YOLO26. URL: https://docs.ultralytics.com/ (дата звернення: 11.04.2026).

16. Flask Documentation. Quickstart. URL: https://flask.palletsprojects.com/en/latest/quickstart/ (дата звернення: 19.04.2026).

17. Python Documentation. csv - CSV File Reading and Writing. URL: https://docs.python.org/3.9/library/csv.html (дата звернення: 11.04.2026).

18. NumPy. NumPy Documentation. URL: https://numpy.org/doc/stable/ (дата звернення: 22.04.2026).

19. OpenCV Documentation. VideoCapture class reference. URL: https://docs.opencv.org/4.x/d8/dfe/classcv_1_1VideoCapture.html (дата звернення: 11.04.2026).

20. Ultralytics YOLO Documentation. Track mode. URL: https://docs.ultralytics.com/modes/track/ (дата звернення: 11.04.2026).

21. Flask Documentation. API. URL: https://flask.palletsprojects.com/en/latest/api/ (дата звернення: 11.04.2026).

22. Dalal N., Triggs B. Histograms of Oriented Gradients for Human Detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).* 2005. Vol. 1. P. 886–893. DOI: https://doi.org/10.1109/CVPR.2005.177.

23. Viola P., Jones M. J. Rapid Object Detection using a Boosted Cascade of Simple Features. *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 2001. Vol. 1. DOI: https://doi.org/10.1109/CVPR.2001.990517.

24. Bochkovskiy A., Wang C.-Y., Liao H.-Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv.* 2020. URL: https://arxiv.org/abs/2004.10934 (дата звернення: 14.04.2026).

25. Redmon J., Farhadi A. YOLO9000: Better, Faster, Stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2017. P. 6517–6525. DOI: https://doi.org/10.1109/CVPR.2017.690.

26. Lin T.-Y., Goyal P., Girshick R., He K., Dollar P. Focal Loss for Dense Object Detection. *2017 IEEE International Conference on Computer Vision (ICCV).* 2017. P. 2999–3007. DOI: https://doi.org/10.1109/ICCV.2017.324.

27. Zhang Y., Zhou D., Chen S., Gao S., Ma Y. Single-Image Crowd Counting via Multi-Column Convolutional Neural Network. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2016. P. 589–597. DOI: https://doi.org/10.1109/CVPR.2016.70.

28. Li Y., Zhang X., Chen D. CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2018. P. 1091–1100. DOI: https://doi.org/10.1109/CVPR.2018.00120.

29. Cao Z., Hidalgo G., Simon T., Wei S.-E., Sheikh Y. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence. 2021.* Vol. 43, No. 1. P. 172–186. DOI: https://doi.org/10.1109/TPAMI.2019.2929257.

30. Wojke N., Bewley A., Paulus D. Simple Online and Realtime Tracking with a Deep Association Metric. *2017 IEEE International Conference on Image Processing (ICIP).* 2017. P. 3645–3649. DOI: https://doi.org/10.1109/ICIP.2017.8296962.

31. Zhang Y., Wang C., Wang X., Zeng W., Liu W. FairMOT: On the Fairness of Detection and Re-Identification in Multiple Object Tracking. *International Journal of Computer Vision*. 2021. Vol. 129. P. 3069–3087. DOI: https://doi.org/10.1007/s11263-021-01513-4.

32. Zhang Y., Sun P., Jiang Y., Yu D., Yuan Z., Luo P., Liu W., Wang X. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. *European Conference on Computer Vision (ECCV).* 2022. P. 1–21. DOI: https://doi.org/10.1007/978-3-031-20047-2_1.

33. Bergstra J., Bengio Y. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research.* 2012. Vol. 13. P. 281–305. URL: https://jmlr.org/papers/v13/bergstra12a.html (дата звернення: 22.04.2026).

34. Kalman R. E. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*. 1960. Vol. 82, No. 1. P. 35–45. DOI: https://doi.org/10.1115/1.3662552.

35. Bar-Shalom Y., Li X. R., Kirubarajan T. Estimation with Applications to Tracking and Navigation. *New York : John Wiley & Sons*. 2001. 584 p.

36. Szeliski R. Computer Vision: Algorithms and Applications. 2nd ed. Cham : Springer, 2022. 925 p. DOI: https://doi.org/10.1007/978-3-030-34372-9.

37. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. URL: https://www.deeplearningbook.org/ (дата звернення: 22.04.2026).

38. Bishop C. M. Pattern Recognition and Machine Learning. New York : Springer. 2006. 738 p.

39. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 4th ed. Hoboken : Pearson. 2021. 1168 p.

40. Gonzalez R. C., Woods R. E. Digital Image Processing. 4th ed. New York : Pearson, 2018. 1024 p.

41. MDN Web Docs. JavaScript Guide. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 22.04.2026).

42. W3C. Cascading Style Sheets (CSS). URL: https://www.w3.org/Style/CSS/Overview.en.html (дата звернення: 22.04.2026).

43. OpenCV Documentation. HOGDescriptor class reference. URL: https://docs.opencv.org/4.x/d5/d33/structcv_1_1HOGDescriptor.html (дата звернення: 22.04.2026).

44. OpenCV Documentation. Object Detection. URL: https://docs.opencv.org/4.x/d5/d54/group__objdetect.html (дата звернення: 22.04.2026).

45. OpenCV Documentation. Motion Analysis and Object Tracking. URL: https://docs.opencv.org/4.x/dc/d6b/group__video__track.html (дата звернення: 22.04.2026).

46. Ultralytics YOLO Documentation. Predict mode. URL: https://docs.ultralytics.com/modes/predict/ (дата звернення: 22.04.2026).

47. Python Documentation. threading - Thread-based parallelism. URL: https://docs.python.org/3.9/library/threading.html (дата звернення: 22.04.2026).

48. OpenCV Documentation. Image Processing (imgproc module). URL: https://docs.opencv.org/4.x/d7/dbd/group__imgproc.html (дата звернення: 22.04.2026).

49. OpenCV Documentation. Image file reading and writing (imgcodecs module). URL: https://docs.opencv.org/4.x/d4/da8/group__imgcodecs.html (дата звернення: 22.04.2026).

50. Python Documentation. pathlib - Object-oriented filesystem paths. URL: https://docs.python.org/3.9/library/pathlib.html (дата звернення: 22.04.2026).

51. Зразки плану евакуації людей. URL: https://antifire.ua/video/plan-evakuacii-2.pdf (дата звернення: 19.05.2026).

52. Показова навчальна евакуація на випадок пожежі. URL: https://vysh.gov.ua/pokazova-navchalna-evakuatsiya-na-vypadok-pozhezhi-u-vssh-suzir-ya/ (дата звернення: 19.05.2026).

53. Базові вимоги пожежної безпеки для громадських об’єктів. URL: https://kolo.news/category/suspilstvo/29877 (дата звернення: 19.05.2026).

54. Інфрачервоний датчик руху. URL: https://crow.ua/products/genius (дата звернення: 19.05.2026).

55. ІЧ-бар'єр. URL: https://protection-key.com.ua/ich-baryer-lightwell-lbx-100-af-z-rezhimom-antituman (дата звернення: 19.05.2026).

56. Chelmsford Lock & Key. URL: https://chelmsfordlock.com/products/ (дата звернення: 19.05.2026).

57. Тепловизійна камера Hikvision. URL: https://hikvision.co.ua/ua/hikvision-ds-2td4228-10w-10-mm/ (дата звернення: 19.05.2026).

58. Introduction to Audio Analysis and Processing. URL: https://blog.paperspace.com/introduction-to-audio-analysis-and-synthesis/ (дата звернення: 19.05.2026).

59. Ultrasonic Distance Sensor. URL: https://www.dfrobot.com/product-1832.html (дата звернення: 19.05.2026).

60. Вимірювання швидкості руху приладами TruCAM. URL: https://sud.ua/uk/news/ukraine/294477-patrulnaya-politsiya-uvelichivaet-kolichestvo-uchastkov-na-kotorykh-budut-izmeryat-skorost-dvizheniya-priboramitrucam (дата звернення: 19.05.2026).

61. Medium. Using AI to Detect Social Distancing Violations. URL: https://medium.com/swlh/using-ai-to-detect-social-distancing-violations-4707301844be (дата звернення: 19.05.2026).

62. Canon. Video-analysis technology. URL: https://global.canon/en/news/2019/20191219.html (дата звернення: 19.05.2026).

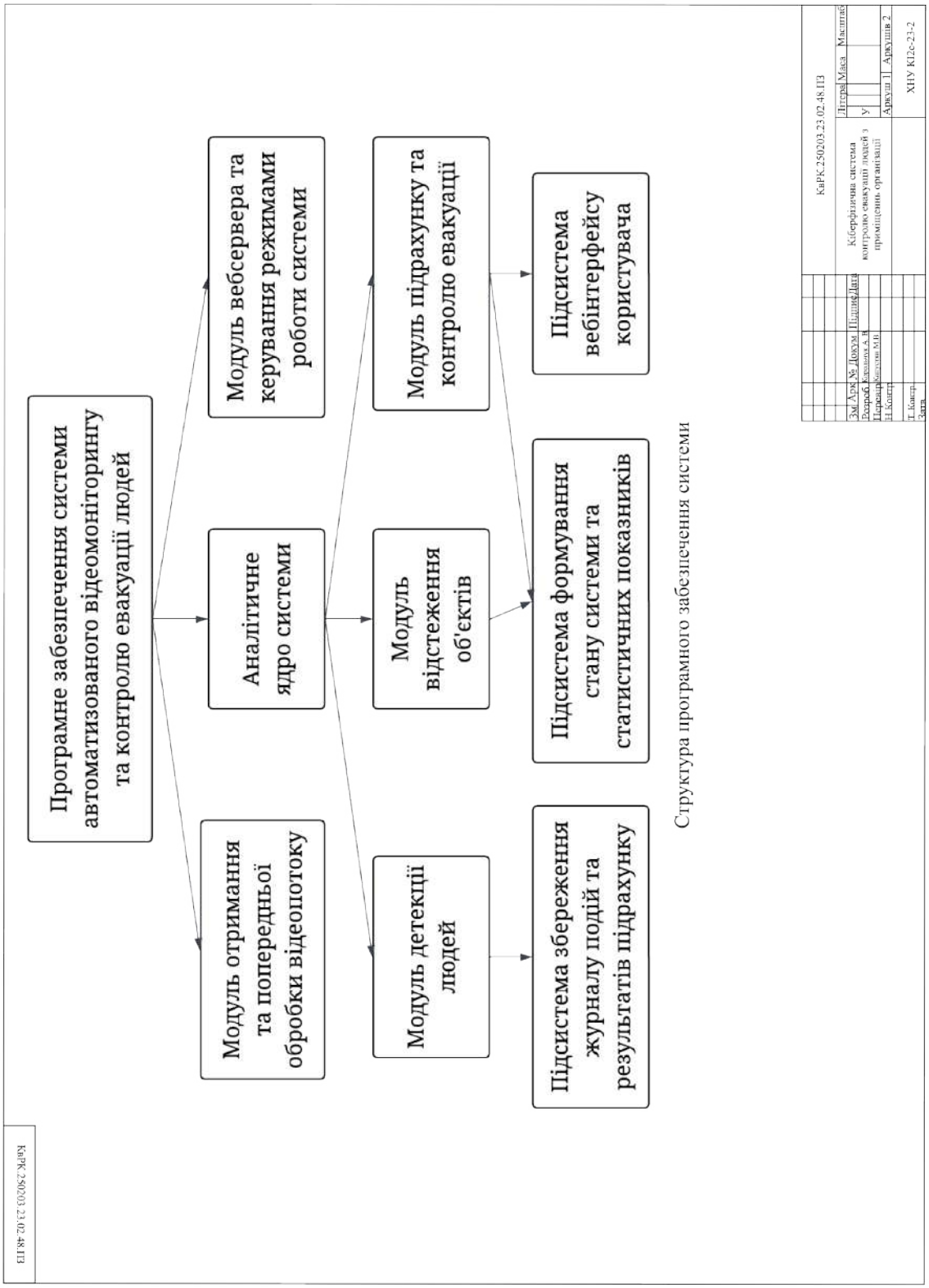
63. Roboflow. People Counting Using Computer Vision. URL: https://blog.roboflow.com/people-counting-computer-vision-software/ (дата звернення: 19.05.2026).

64. GitHub. Tensorflow object counting apiPublic. URL: https://github.com/ahmetozlu/tensorflow_object_counting_api (дата звернення: 19.05.2026).

ДОДАТОК А

(обов’язковий)

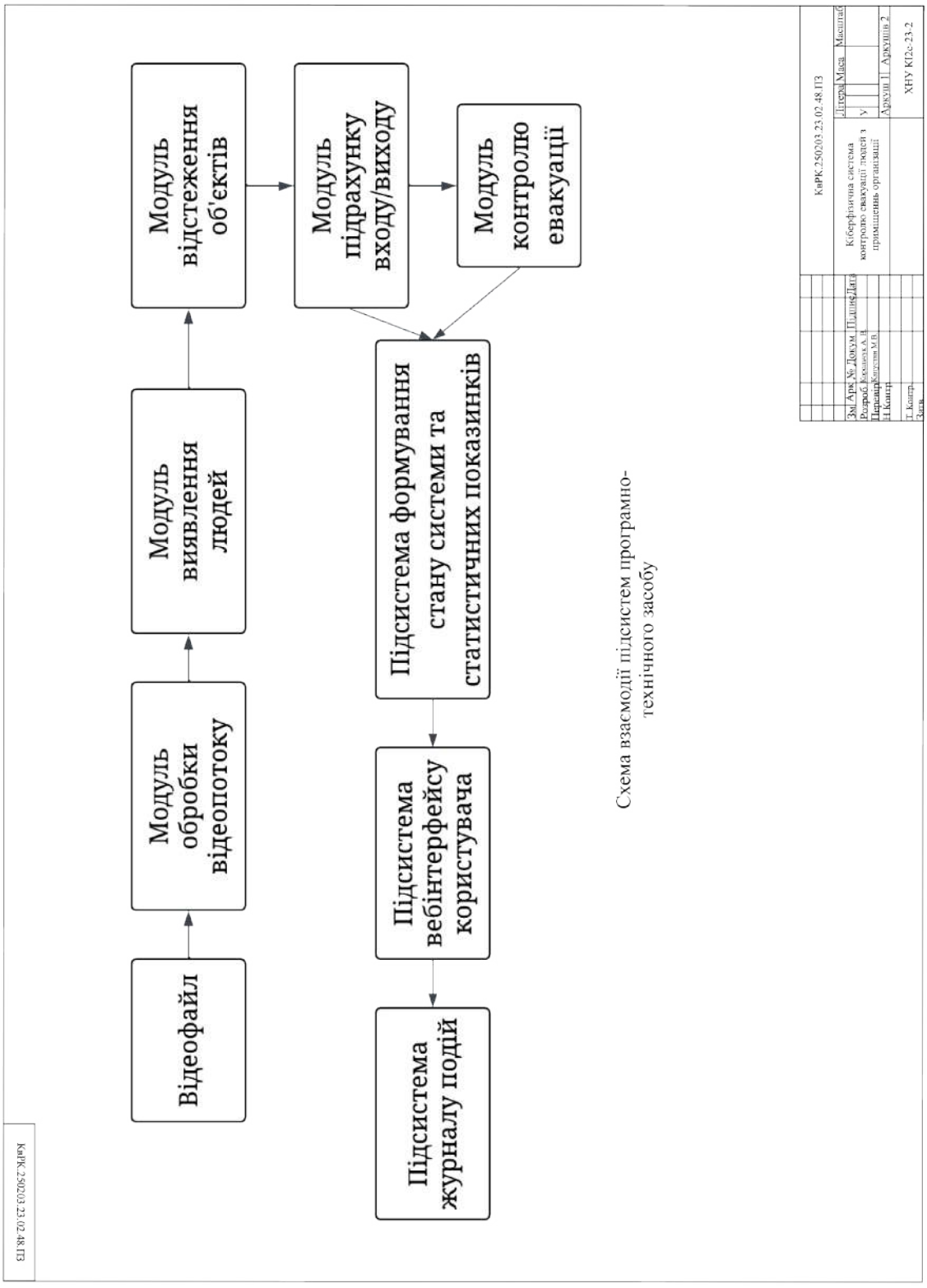
Копія креслення «Структура програмного забезпечення системи контролю евакуації людей»



ДОДАТОК Б

(обов'язковий)

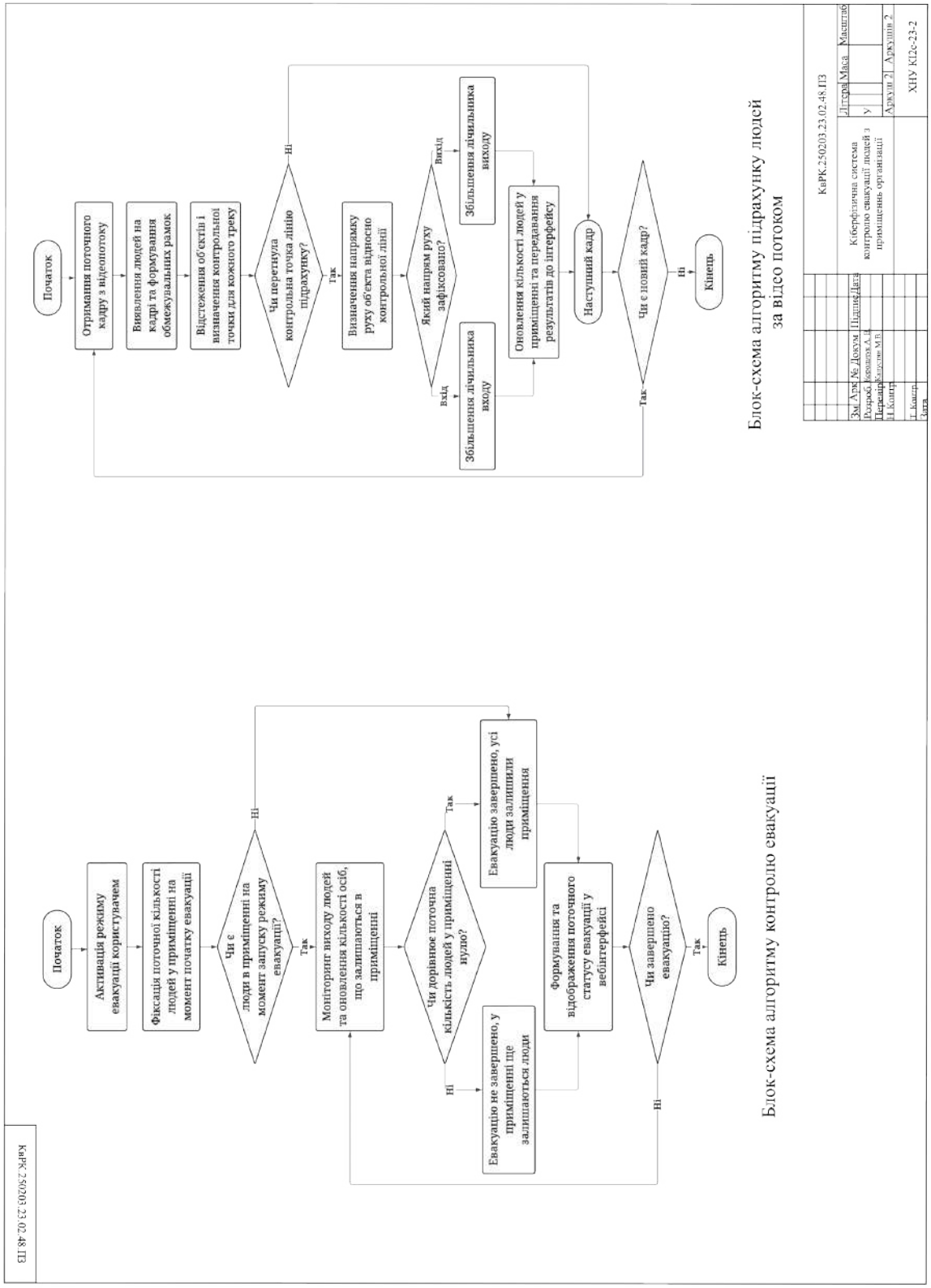
Копія креслення «Схема взаємодії підсистем системи контролю евакуації людей»



ДОДАТОК В

(обов’язковий)

Копія креслення «Блок-схеми алгоритмів підрахунку людей та контролю евакуації»



ДОДАТОК Г

(обов'язковий)

Лістинг програмного коду

requirements.txt

```
opencv-python
ultralytics
numpy
filterpy==1.4.5
scipy
flask
```

src/main.py

```
from pathlib import Path

from flask import Flask, Response, jsonify, render_template, request

from people_counter import PeopleCounter

# Шляхи файлів
project_root = Path(__file__).resolve().parents[1]
video_source = project_root / "data" / "1_test.mp4"
model_path = Path(__file__).with_name("yolov8n.pt")

# Flask-додаток
app = Flask(
    __name__,
    template_folder=str(Path(__file__).with_name("templates")),
    static_folder=str(Path(__file__).with_name("static")),
)

# Лічильник
counter = PeopleCounter(video_source=str(video_source), model_path=model_path)


# Головна сторінка
@app.get("/")
def index():
    return render_template("index.html")


# Відеопотік
@app.get("/video-feed")
def video_feed():
    counter.start_processing()
    return Response(
        counter.frames(),
        mimetype="multipart/x-mixed-replace; boundary=frame",
    )
```

```
# Поточний стан
@app.get("/api/status")
def api_status():
    return jsonify(counter.get_snapshot_data())


# Запуск аналізу
@app.post("/api/control/start")
def api_start():
    counter.start_processing()
    return jsonify({"ok": True, "message": "Обробку відео запущено."})


# Зупинка аналізу
@app.post("/api/control/stop")
def api_stop():
    counter.stop_processing()
    return jsonify({"ok": True, "message": "Обробку відео зупинено."})


# Скидання даних
@app.post("/api/control/reset")
def api_reset():
    counter.reset_counts()
    return jsonify({"ok": True, "message": "Лічильники скинуто."})


# Налаштування лінії
@app.post("/api/control/line")
def api_update_line():
    payload = request.get_json(silent=True) or {}
    line_position = payload.get("line_position")
    if line_position is None:
        return jsonify({"ok": False, "message": "Не передано нову позицію
лінії."}), 400

    updated_line = counter.set_line_position(int(line_position))
    return jsonify(
        {
            "ok": True,
            "message": "Положення лінії оновлено.",
            "line_position": updated_line,
        }
    )


# Напрям руху
@app.post("/api/control/direction")
def api_update_direction():
    payload = request.get_json(silent=True) or {}
    entry_direction = payload.get("entry_direction")
```

```
    if entry_direction is None:
        return jsonify({"ok": False, "message": "Не передано напрямок входу."}),
400

    try:
        updated_direction = counter.set_entry_direction(str(entry_direction))
    except ValueError as error:
        return jsonify({"ok": False, "message": str(error)}), 400

    return jsonify(
        {
            "ok": True,
            "message": "Напрямок підрахунку оновлено.",
            "entry_direction": updated_direction,
        }
    )


# Початок евакуації
@app.post("/api/control/evacuation/start")
def api_start_evacuation():
    counter.start_evacuation()
    return jsonify({"ok": True, "message": "Режим евакуації активовано."})


# Завершення евакуації
@app.post("/api/control/evacuation/stop")
def api_stop_evacuation():
    counter.stop_evacuation()
    return jsonify({"ok": True, "message": "Режим евакуації вимкнено."})


# Джерело відео
@app.post("/api/control/source")
def api_change_source():
    payload = request.get_json(silent=True) or {}
    return jsonify(
        {
            "ok": False,
            "message": (
                "Зміна джерела відео через сайт поки не реалізована. "
                "За потреби можу додати окреме поле для камери або іншого файлу."
            ),
            "payload": payload,
        }
    ), 400


# Запуск сервера
def main() -> None:
    app.run(host="127.0.0.1", port=5050, debug=False, threaded=True)
```

```
if __name__ == "__main__":
    main()
```

src/app_state.py

```
from dataclasses import dataclass, field
from datetime import datetime
from typing import List, Optional


# Подія лічильника
@dataclass
class CounterEvent:
    timestamp: str
    track_id: int
    direction: str
    occupancy: int
    evacuation_active: bool
    remaining_to_evacuate: int


# Знімок стану
@dataclass
class CounterSnapshot:
    total_entered: int = 0
    total_exited: int = 0
    occupancy: int = 0
    evacuation_active: bool = False
    evacuation_target: int = 0
    remaining_to_evacuate: int = 0
    evacuation_started_at: Optional[str] = None
    evacuation_completed: bool = False
    last_event: str = "Система очікує запуску"
    recent_events: List[CounterEvent] = field(default_factory=list)


# Менеджер стану
class OccupancyManager:
    def __init__(self, event_history_limit: int = 12):
        self.event_history_limit = event_history_limit
        self.snapshot = CounterSnapshot()

    # Реєстрація входу
    def register_entry(self, track_id: int) -> CounterEvent:
        self.snapshot.total_entered += 1
        self.snapshot.occupancy += 1
        return self._register_event(track_id=track_id, direction="Entry")

    # Реєстрація виходу
    def register_exit(self, track_id: int) -> CounterEvent:
        self.snapshot.total_exited += 1
        self.snapshot.occupancy = max(0, self.snapshot.occupancy - 1)
        return self._register_event(track_id=track_id, direction="Exit")
```

```
    # Старт евакуації
    def start_evacuation(self) -> None:
        self.snapshot.evacuation_active = True
        self.snapshot.evacuation_completed = self.snapshot.occupancy == 0
        self.snapshot.evacuation_target = self.snapshot.occupancy
        self.snapshot.remaining_to_evacuate = self.snapshot.occupancy
        self.snapshot.evacuation_started_at = now_str()
        self.snapshot.last_event = (
            "Евакуацію розпочато"
            if self.snapshot.occupancy > 0
            else "Евакуацію розпочато, приміщення вже порожнє"
        )

    # Стоп евакуації
    def stop_evacuation(self) -> None:
        self.snapshot.evacuation_active = False
        self.snapshot.evacuation_completed = False
        self.snapshot.remaining_to_evacuate = 0
        self.snapshot.evacuation_target = 0
        self.snapshot.evacuation_started_at = None
        self.snapshot.last_event = "Режим евакуації вимкнено"

    # Скидання стану
    def reset(self) -> None:
        self.snapshot = CounterSnapshot(last_event="Лічильники скинуто")

    # Запис події
    def _register_event(self, track_id: int, direction: str) -> CounterEvent:
        if self.snapshot.evacuation_active:
            self.snapshot.remaining_to_evacuate = self.snapshot.occupancy
            self.snapshot.evacuation_completed = self.snapshot.occupancy == 0

        direction_label = "увійшла" if direction == "Entry" else "вийшла"
        self.snapshot.last_event = f"Людина #{track_id} {direction_label}"

        event = CounterEvent(
            timestamp=now_str(),
            track_id=track_id,
            direction=direction,
            occupancy=self.snapshot.occupancy,
            evacuation_active=self.snapshot.evacuation_active,
            remaining_to_evacuate=self.snapshot.remaining_to_evacuate,
        )
        self.snapshot.recent_events.insert(0, event)
        self.snapshot.recent_events = self.snapshot.recent_events[
            : self.event_history_limit
        ]
        return event


# Поточний час
def now_str() -> str:
    return datetime.now().strftime("%Y-%m-%d %H:%M:%S")
```

people_counter.py

```
from dataclasses import asdict
from pathlib import Path
import threading
import time
from typing import Callable, Optional, Union

import cv2
import numpy as np
from ultralytics.models import YOLO

from app_state import CounterSnapshot, OccupancyManager
from tracker import Tracker
from utils import CsvEventLogger


SnapshotCallback = Callable[[CounterSnapshot], None]


# Основний клас
class PeopleCounter:
    def __init__(
        self,
        video_source: Union[int, str] = 0,
        model_path: Optional[Union[str, Path]] = None,
        csv_path: Optional[Union[str, Path]] = None,
        snapshot_callback: Optional[SnapshotCallback] = None,
    ):
        self.video_source = video_source
        self.model_path = Path(model_path or
Path(__file__).with_name("yolov8n.pt"))
        self.csv_path = Path(csv_path or
Path(__file__).with_name("people_count.csv"))
        self.snapshot_callback = snapshot_callback

        # Модель YOLO
        self.model = YOLO(str(self.model_path))
        self.tracker = Tracker()
        self.state_manager = OccupancyManager()
        self.logger = CsvEventLogger(self.csv_path)

        # Параметри кадру
        self.cap = None
        self.frame_width = 0
        self.frame_height = 0
        self.line_position: Optional[int] = None
        self.entry_direction = "up"
        self.smoothing_alpha = 0.72
        self.track_memory = {}
        self.frame_index = 0
```

```
        # Потокова обробка
        self.latest_frame_jpeg: Optional[bytes] = None
        self.running = False
        self.frame_lock = threading.Lock()
        self.worker_thread: Optional[threading.Thread] = None
        self.is_camera_source = isinstance(video_source, int)

        self._open_capture()

    @property
    def snapshot(self) -> CounterSnapshot:
        return self.state_manager.snapshot

    # Дані статусу
    def set_snapshot_callback(self, callback: SnapshotCallback) -> None:
        self.snapshot_callback = callback
        self._notify_snapshot()

    def get_snapshot_data(self) -> dict:
        snapshot_data = asdict(self.snapshot)
        snapshot_data["processing_active"] = self.running
        snapshot_data["line_position"] = self.line_position
        snapshot_data["frame_height"] = self.frame_height
        snapshot_data["line_position_percent"] = (
            round((self.line_position / self.frame_height) * 100, 1)
            if self.frame_height and self.line_position is not None
            else 0
        )
        snapshot_data["entry_direction"] = self.entry_direction
        snapshot_data["entry_direction_label"] = (
            "Вхід зверху вниз" if self.entry_direction == "down" else "Вхід знизу
вгору"
        )
        return snapshot_data

    # Кадр відео
    def get_latest_frame(self) -> Optional[bytes]:
        with self.frame_lock:
            return self.latest_frame_jpeg

    # Запуск потоку
    def start_processing(self) -> None:
        if self.worker_thread is not None and self.worker_thread.is_alive():
            return

        self.running = True
        self.worker_thread = threading.Thread(target=self._processing_loop,
daemon=True)
        self.worker_thread.start()
        self._notify_snapshot()

    # Зупинка потоку
```

```
    def stop_processing(self) -> None:
        self.running = False

    # Позиція лінії
    def set_line_position(self, line_position: int) -> int:
        if self.frame_height <= 0:
            return self.line_position or 0

        min_position = 20
        max_position = max(min_position, self.frame_height - 20)
        self.line_position = max(min_position, min(int(line_position),
max_position))
        return self.line_position

    # Напрям входу
    def set_entry_direction(self, entry_direction: str) -> str:
        if entry_direction not in {"up", "down"}:
            raise ValueError("Некоректний напрямок входу.")

        self.entry_direction = entry_direction
        self.track_memory.clear()
        self.frame_index = 0
        self._notify_snapshot()
        return self.entry_direction

    # Старт евакуації
    def start_evacuation(self) -> None:
        self.state_manager.start_evacuation()
        self._notify_snapshot()

    # Стоп евакуації
    def stop_evacuation(self) -> None:
        self.state_manager.stop_evacuation()
        self._notify_snapshot()

    # Скидання рахунку
    def reset_counts(self) -> None:
        self.state_manager.reset()
        self.tracker = Tracker()
        self.track_memory.clear()
        self.frame_index = 0
        self.logger.reset()
        self._notify_snapshot()

    # Потік кадрів
    def frames(self):
        while True:
            frame = self.get_latest_frame()
            if frame is None:
                time.sleep(0.1)
                continue
```

```
            yield (
                b"--frame\r\n"
                b"Content-Type: image/jpeg\r\n\r\n" + frame + b"\r\n"
            )
            time.sleep(0.03)

    # Цикл обробки
    def _processing_loop(self) -> None:
        if self.cap is None or not self.cap.isOpened():
            self._open_capture()

        while self.running:
            ret, frame = self.cap.read()
            if not ret:
                if self.is_camera_source:
                    time.sleep(0.1)
                    continue

                self._restart_video_file()
                continue

            processed_frame = self._process_frame(frame)
            self._draw_overlay(processed_frame)
            self._store_frame(processed_frame)

        self._cleanup_capture()
        self._notify_snapshot()

    # Обробка кадру
    def _process_frame(self, frame: np.ndarray) -> np.ndarray:
        self.frame_index += 1
        results = self.model(frame, verbose=False)[0]
        detections = []

        # Детекція людей
        for box, cls, conf in zip(
            results.boxes.xyxy,
            results.boxes.cls,
            results.boxes.conf,
        ):
            if int(cls) != 0 or float(conf) < 0.20:
                continue

            x1, y1, x2, y2 = box.tolist()
            detections.append([x1, y1, x2, y2, float(conf)])

        # Оновлення треків
        if detections:
            tracks = self.tracker.update(np.array(detections, dtype=float))
        else:
            tracks = self.tracker.update(np.empty((0, 5)))
```

```
        for track in tracks:
            x1, y1, x2, y2, track_id = track
            track_id = int(track_id)

            # Пам'ять треку
            previous_state = self.track_memory.get(track_id)
            if previous_state is None:
                raw_reference_y = int(y1 + (y2 - y1) * 0.65)
                self.track_memory[track_id] = {
                    "smoothed_box": [float(x1), float(y1), float(x2), float(y2)],
                    "last_reference_y": raw_reference_y,
                    "last_side": self._get_side(raw_reference_y),
                    "count_frame": -1,
                    "frames_seen": 1,
                }
                previous_state = self.track_memory[track_id]
            else:
                previous_state["frames_seen"] += 1
                previous_state["smoothed_box"] = self._smooth_box(
                    previous_state["smoothed_box"], [x1, y1, x2, y2]
                )

            smoothed_x1, smoothed_y1, smoothed_x2, smoothed_y2 = previous_state[
                "smoothed_box"
            ]
            cx = int((smoothed_x1 + smoothed_x2) / 2)
            cy = int((smoothed_y1 + smoothed_y2) / 2)
            reference_y = int(y1 + (y2 - y1) * 0.65)

            # Перетин лінії
            last_side = previous_state["last_side"]
            current_side = self._get_side(reference_y)
            is_mature_track = previous_state["frames_seen"] >= 2
            recently_counted = (self.frame_index - previous_state["count_frame"])
< 8
            crossed_up = last_side == "below" and current_side == "above"
            crossed_down = last_side == "above" and current_side == "below"

            if (
                (crossed_up or crossed_down)
                and is_mature_track
                and not recently_counted
            ):
                # Подія проходу
                if self._is_entry_crossing(crossed_up, crossed_down):
                    event = self.state_manager.register_entry(track_id)
                else:
                    event = self.state_manager.register_exit(track_id)

                self.logger.append_event(event)
                self.track_memory[track_id]["count_frame"] = self.frame_index
                self._notify_snapshot()
```

```
            # Малювання треку
            self.track_memory[track_id]["last_reference_y"] = reference_y
            self.track_memory[track_id]["last_side"] = current_side
            self._draw_track(
                frame,
                smoothed_x1,
                smoothed_y1,
                smoothed_x2,
                smoothed_y2,
                cx,
                cy,
                reference_y,
                track_id,
            )

        return frame

    # Рамка людини
    def _draw_track(
        self,
        frame: np.ndarray,
        x1: float,
        y1: float,
        x2: float,
        y2: float,
        cx: int,
        cy: int,
        reference_y: int,
        track_id: int,
    ) -> None:
        cv2.rectangle(
            frame,
            (int(x1), int(y1)),
            (int(x2), int(y2)),
            (16, 194, 150),
            2,
        )
        cv2.circle(frame, (cx, cy), 4, (0, 145, 255), -1)
        cv2.circle(frame, (cx, reference_y), 4, (83, 224, 176), -1)
        cv2.putText(
            frame,
            f"ID:{track_id}",
            (int(x1), int(y1) - 10),
            cv2.FONT_HERSHEY_SIMPLEX,
            0.6,
            (255, 255, 255),
            2,
        )

    # Статистика кадру
    def _draw_overlay(self, frame: np.ndarray) -> None:
```

```
        snapshot = self.snapshot
        overlay = frame.copy()
        panel_height = 176
        cv2.rectangle(overlay, (16, 16), (430, panel_height), (8, 18, 32), -1)
        cv2.addWeighted(overlay, 0.56, frame, 0.44, 0, frame)

        cv2.line(
            frame,
            (0, self.line_position),
            (self.frame_width, self.line_position),
            (38, 68, 255),
            3,
        )

        stats = [
            f"Entered: {snapshot.total_entered}",
            f"Exited: {snapshot.total_exited}",
            f"Inside: {snapshot.occupancy}",
            f"Evacuation: {'ACTIVE' if snapshot.evacuation_active else 'OFF'}",
            f"Remaining: {snapshot.remaining_to_evacuate}",
            f"All evacuated: {'YES' if snapshot.evacuation_completed else 'NO'}",
        ]

        for index, text in enumerate(stats):
            cv2.putText(
                frame,
                text,
                (32, 42 + index * 24),
                cv2.FONT_HERSHEY_SIMPLEX,
                0.68,
                (255, 255, 255),
                2,
            )

    # Збереження кадру
    def _store_frame(self, frame: np.ndarray) -> None:
        success, buffer = cv2.imencode(".jpg", frame)
        if not success:
            return

        with self.frame_lock:
            self.latest_frame_jpeg = buffer.tobytes()

    # Оновлення стану
    def _notify_snapshot(self) -> None:
        if self.snapshot_callback is not None:
            self.snapshot_callback(self.snapshot)

    # Відкриття відео
    def _open_capture(self) -> None:
        self._cleanup_capture()
        self.cap = cv2.VideoCapture(self.video_source)
```

```
        if not self.cap.isOpened():
            raise RuntimeError(f"Не вдалося відкрити джерело відео:
{self.video_source}")

        self.frame_width = int(self.cap.get(cv2.CAP_PROP_FRAME_WIDTH) or 0)
        self.frame_height = int(self.cap.get(cv2.CAP_PROP_FRAME_HEIGHT) or 0)
        if self.frame_height:
            default_line_position = int(self.frame_height * 0.38)
            if self.line_position is None:
                self.line_position = default_line_position
            else:
                self.set_line_position(self.line_position)

    # Повтор відео
    def _restart_video_file(self) -> None:
        self._open_capture()
        self.tracker = Tracker()
        self.track_memory.clear()
        self.frame_index = 0

    # Закриття відео
    def _cleanup_capture(self) -> None:
        if self.cap is not None and self.cap.isOpened():
            self.cap.release()

    # Згладження рамки
    def _smooth_box(self, previous_box, current_box):
        alpha = self.smoothing_alpha
        return [
            previous_value * alpha + current_value * (1.0 - alpha)
            for previous_value, current_value in zip(previous_box, current_box)
        ]

    # Сторона лінії
    def _get_side(self, y: int) -> str:
        return "above" if y < (self.line_position or 0) else "below"

    # Тип перетину
    def _is_entry_crossing(self, crossed_up: bool, crossed_down: bool) -> bool:
        if self.entry_direction == "down":
            return crossed_down
        return crossed_up
```

tracker.py

```
import numpy as np

from sort_simple import Sort


# Клас трекера
class Tracker:
    def __init__(self):
```

```
        self.tracker = Sort(max_age=25, min_hits=2, iou_threshold=0.12)

    # Оновлення треків
    def update(self, detections: np.ndarray) -> np.ndarray:
        """
        Оновлює трекери на основі детекцій.

        detections: np.ndarray, shape (N,5) - [[x1,y1,x2,y2,score], ...]
        Повертає: np.ndarray, shape (M,5) - [[x1,y1,x2,y2,track_id], ...]
        """
        if detections is None or len(detections) == 0:
            detections = np.empty((0, 5))

        tracks = self.tracker.update(detections)
        return tracks
```

sort_simple.py

```
# Оригінал SORT
"""
SORT: A Simple, Online and Realtime Tracker
Copyright (C) 2016-2020 Alex Bewley alex@bewley.ai

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
"""

from __future__ import print_function

import numpy as np
from filterpy.kalman import KalmanFilter

np.random.seed(0)


# Призначення пар
def linear_assignment(cost_matrix):
    try:
        import lap

        _, x, y = lap.lapjv(cost_matrix, extend_cost=True)
        return np.array([[y[i], i] for i in x if i >= 0])  #
    except ImportError:
```

```
        from scipy.optimize import linear_sum_assignment

        x, y = linear_sum_assignment(cost_matrix)
        return np.array(list(zip(x, y)))


# Перетин рамок
def iou_batch(bb_test, bb_gt):
    """
    From SORT: Computes IOU between two bboxes in the form [x1,y1,x2,y2]
    """
    bb_gt = np.expand_dims(bb_gt, 0)
    bb_test = np.expand_dims(bb_test, 1)

    xx1 = np.maximum(bb_test[..., 0], bb_gt[..., 0])
    yy1 = np.maximum(bb_test[..., 1], bb_gt[..., 1])
    xx2 = np.minimum(bb_test[..., 2], bb_gt[..., 2])
    yy2 = np.minimum(bb_test[..., 3], bb_gt[..., 3])
    w = np.maximum(0.0, xx2 - xx1)
    h = np.maximum(0.0, yy2 - yy1)
    wh = w * h
    o = wh / (
        (bb_test[..., 2] - bb_test[..., 0]) * (bb_test[..., 3] - bb_test[..., 1])
        + (bb_gt[..., 2] - bb_gt[..., 0]) * (bb_gt[..., 3] - bb_gt[..., 1])
        - wh
    )
    return o


# Рамка у стан
def convert_bbox_to_z(bbox):
    """
    Takes a bounding box in the form [x1,y1,x2,y2] and returns z in the form
      [x,y,s,r] where x,y is the centre of the box and s is the scale/area and r
is
      the aspect ratio
    """
    w = bbox[2] - bbox[0]
    h = bbox[3] - bbox[1]
    x = bbox[0] + w / 2.0
    y = bbox[1] + h / 2.0
    s = w * h  # scale is just area
    r = w / float(h)
    return np.array([x, y, s, r]).reshape((4, 1))


# Стан у рамку
def convert_x_to_bbox(x, score=None):
    """
    Takes a bounding box in the centre form [x,y,s,r] and returns it in the form
      [x1,y1,x2,y2] where x1,y1 is the top left and x2,y2 is the bottom right
    """
    w = np.sqrt(x[2] * x[3])
```

```
    h = x[2] / w
    if score == None:
        return np.array(
            [x[0] - w / 2.0, x[1] - h / 2.0, x[0] + w / 2.0, x[1] + h / 2.0]
        ).reshape((1, 4))
    else:
        return np.array(
            [x[0] - w / 2.0, x[1] - h / 2.0, x[0] + w / 2.0, x[1] + h / 2.0,
score]
        ).reshape((1, 5))


# Калман-трекер
class KalmanBoxTracker(object):
    """
    This class represents the internal state of individual tracked objects
observed as bbox.
    """

    count = 0

    def __init__(self, bbox):
        """
        Initialises a tracker using initial bounding box.
        """
        # define constant velocity model
        self.kf = KalmanFilter(dim_x=7, dim_z=4)
        self.kf.F = np.array(
            [
                [1, 0, 0, 0, 1, 0, 0],
                [0, 1, 0, 0, 0, 1, 0],
                [0, 0, 1, 0, 0, 0, 1],
                [0, 0, 0, 1, 0, 0, 0],
                [0, 0, 0, 0, 1, 0, 0],
                [0, 0, 0, 0, 0, 1, 0],
                [0, 0, 0, 0, 0, 0, 1],
            ]
        )
        self.kf.H = np.array(
            [
                [1, 0, 0, 0, 0, 0, 0],
                [0, 1, 0, 0, 0, 0, 0],
                [0, 0, 1, 0, 0, 0, 0],
                [0, 0, 0, 1, 0, 0, 0],
            ]
        )

        self.kf.R[2:, 2:] *= 10.0
        self.kf.P[4:, 4:] *= (
            1000.0  # give high uncertainty to the unobservable initial velocities
        )
        self.kf.P *= 10.0
        self.kf.Q[-1, -1] *= 0.01
```

```
        self.kf.Q[4:, 4:] *= 0.01

        self.kf.x[:4] = convert_bbox_to_z(bbox)
        self.time_since_update = 0
        self.id = KalmanBoxTracker.count
        KalmanBoxTracker.count += 1
        self.history = []
        self.hits = 0
        self.hit_streak = 0
        self.age = 0

    # Оновлення рамки
    def update(self, bbox):
        """
        Updates the state vector with observed bbox.
        """
        self.time_since_update = 0
        self.history = []
        self.hits += 1
        self.hit_streak += 1
        self.kf.update(convert_bbox_to_z(bbox))

    # Прогноз рамки
    def predict(self):
        """
        Advances the state vector and returns the predicted bounding box estimate.
        """
        if (self.kf.x[6] + self.kf.x[2]) <= 0:
            self.kf.x[6] *= 0.0
        self.kf.predict()
        self.age += 1
        if self.time_since_update > 0:
            self.hit_streak = 0
        self.time_since_update += 1
        self.history.append(convert_x_to_bbox(self.kf.x))
        return self.history[-1]

    # Поточна рамка
    def get_state(self):
        """
        Returns the current bounding box estimate.
        """
        return convert_x_to_bbox(self.kf.x)


# Зіставлення об'єктів
def associate_detections_to_trackers(detections, trackers, iou_threshold=0.3):
    """
    Assigns detections to tracked object (both represented as bounding boxes)

    Returns 3 lists of matches, unmatched_detections and unmatched_trackers
    """
    if len(trackers) == 0:
```

```
        return (
            np.empty((0, 2), dtype=int),
            np.arange(len(detections)),
            np.empty((0, 5), dtype=int),
        )

    iou_matrix = iou_batch(detections, trackers)

    if min(iou_matrix.shape) > 0:
        a = (iou_matrix > iou_threshold).astype(np.int32)
        if a.sum(1).max() == 1 and a.sum(0).max() == 1:
            matched_indices = np.stack(np.where(a), axis=1)
        else:
            matched_indices = linear_assignment(-iou_matrix)
    else:
        matched_indices = np.empty(shape=(0, 2))

    unmatched_detections = []
    for d, det in enumerate(detections):
        if d not in matched_indices[:, 0]:
            unmatched_detections.append(d)
    unmatched_trackers = []
    for t, trk in enumerate(trackers):
        if t not in matched_indices[:, 1]:
            unmatched_trackers.append(t)

    # filter out matched with low IOU
    matches = []
    for m in matched_indices:
        if iou_matrix[m[0], m[1]] < iou_threshold:
            unmatched_detections.append(m[0])
            unmatched_trackers.append(m[1])
        else:
            matches.append(m.reshape(1, 2))
    if len(matches) == 0:
        matches = np.empty((0, 2), dtype=int)
    else:
        matches = np.concatenate(matches, axis=0)

    return matches, np.array(unmatched_detections), np.array(unmatched_trackers)


# SORT-трекер
class Sort(object):
    def __init__(self, max_age=1, min_hits=3, iou_threshold=0.3):
        """
        Sets key parameters for SORT
        """
        self.max_age = max_age
        self.min_hits = min_hits
        self.iou_threshold = iou_threshold
        self.trackers = []
        self.frame_count = 0
```

```
    # Оновлення кадру
    def update(self, dets=np.empty((0, 5))):
        """
        Params:
          dets - a numpy array of detections in the format
[[x1,y1,x2,y2,score],[x1,y1,x2,y2,score],...]
        Requires: this method must be called once for each frame even with empty
detections (use np.empty((0, 5)) for frames without detections).
        Returns the a similar array, where the last column is the object ID.

        NOTE: The number of objects returned may differ from the number of
detections provided.
        """
        self.frame_count += 1
        # get predicted locations from existing trackers.
        trks = np.zeros((len(self.trackers), 5))
        to_del = []
        ret = []
        for t, trk in enumerate(trks):
            pos = self.trackers[t].predict()[0]
            trk[:] = [pos[0], pos[1], pos[2], pos[3], 0]
            if np.any(np.isnan(pos)):
                to_del.append(t)
        trks = np.ma.compress_rows(np.ma.masked_invalid(trks))
        for t in reversed(to_del):
            self.trackers.pop(t)
        matched, unmatched_dets, unmatched_trks =
associate_detections_to_trackers(
            dets, trks, self.iou_threshold
        )

        # update matched trackers with assigned detections
        for m in matched:
            self.trackers[m[1]].update(dets[m[0], :])

        # create and initialise new trackers for unmatched detections
        for i in unmatched_dets:
            trk = KalmanBoxTracker(dets[i, :])
            self.trackers.append(trk)
        i = len(self.trackers)
        for trk in reversed(self.trackers):
            d = trk.get_state()[0]
            if (trk.time_since_update < 1) and (
                trk.hit_streak >= self.min_hits or self.frame_count <=
self.min_hits
            ):
                ret.append(
                    np.concatenate((d, [trk.id + 1])).reshape(1, -1)
                )  # +1 as MOT benchmark requires positive
            i -= 1
            # remove dead tracklet
```

```
            if trk.time_since_update > self.max_age:
                self.trackers.pop(i)
        if len(ret) > 0:
            return np.concatenate(ret)
        return np.empty((0, 5))
```

utils.py

```
import csv
from dataclasses import asdict
from datetime import datetime
from pathlib import Path
from typing import Iterable

from app_state import CounterEvent


# Поля CSV
CSV_HEADERS = [
    "timestamp",
    "track_id",
    "direction",
    "occupancy",
    "evacuation_active",
    "remaining_to_evacuate",
]


# CSV-журнал
class CsvEventLogger:
    def __init__(self, csv_path: Path):
        self.csv_path = Path(csv_path)
        self.csv_path.parent.mkdir(parents=True, exist_ok=True)
        self._ensure_file()

    # Створення файлу
    def _ensure_file(self) -> None:
        if self.csv_path.exists() and self.csv_path.stat().st_size > 0:
            return

        with self.csv_path.open("w", newline="", encoding="utf-8") as file:
            writer = csv.DictWriter(file, fieldnames=CSV_HEADERS)
            writer.writeheader()

    # Очищення журналу
    def reset(self) -> None:
        with self.csv_path.open("w", newline="", encoding="utf-8") as file:
            writer = csv.DictWriter(file, fieldnames=CSV_HEADERS)
            writer.writeheader()

    # Додавання події
    def append_event(self, event: CounterEvent) -> None:
        with self.csv_path.open("a", newline="", encoding="utf-8") as file:
```

```
            writer = csv.DictWriter(file, fieldnames=CSV_HEADERS)
            writer.writerow(asdict(event))

    # Додавання списку
    def append_many(self, events: Iterable[CounterEvent]) -> None:
        with self.csv_path.open("a", newline="", encoding="utf-8") as file:
            writer = csv.DictWriter(file, fieldnames=CSV_HEADERS)
            for event in events:
                writer.writerow(asdict(event))


# Поточний час
def now_str() -> str:
    return datetime.now().strftime("%Y-%m-%d %H:%M:%S")
```

index.html

```
<!doctype html>
<html lang="uk">
    <head>
        <!-- Метадані -->
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>EvacuationHelper</title>
        <link
            rel="stylesheet"
            href="{{ url_for('static', filename='styles.css') }}"
        />
    </head>
    <body>
        <div class="page-shell">
            <!-- Верхній блок -->
            <header class="hero">
                <div class="hero-copy">
                    <p class="eyebrow">EvacuationHelper</p>
                    <h1>
                        Вебсистема контролю людей та евакуації в реальному часі
                    </h1>
                    <p class="hero-text">
                        Панель показує поточну кількість людей у приміщенні,
                        журнал подій, статус евакуації та відеопотік з
                        аналітикою на базі комп'ютерного зору.
                    </p>
                </div>
                <div class="hero-panel">
                    <div class="hero-badge">
                        <span>Статус системи</span>
                        <strong id="processing-state">Очікує запуску</strong>
                    </div>
                    <div class="hero-badge accent">
                        <span>Людей у приміщенні</span>
                        <strong id="occupancy-highlight">0</strong>
                    </div>
```

```
    </div>
</header>

<!-- Кнопки керування -->
<section class="controls">
    <button class="btn btn-primary" data-action="start">
        Запустити аналіз
    </button>
    <button class="btn" data-action="stop">Зупинити аналіз</button>
    <button class="btn btn-warning" data-action="evac-start">
        Почати евакуацію
    </button>
    <button class="btn" data-action="evac-stop">
        Завершити евакуацію
    </button>
    <button class="btn btn-danger" data-action="reset">
        Скинути лічильники
    </button>
</section>

<!-- Основна панель -->
<section class="dashboard-grid">
    <article class="video-card panel">
        <div class="panel-head">
            <h2>Відео з аналітикою</h2>
            <span class="panel-note"
                >Потік оновлюється автоматично</span
            >
        </div>
        <div class="video-frame">
            <img
                id="video-stream"
                src="/video-feed"
                alt="Відеопотік з аналітикою"
            />
        </div>
    </article>

    <article class="panel status-panel">
        <div class="panel-head">
            <h2>Оперативні показники</h2>
            <span class="panel-note">Оновлення щосекунди</span>
        </div>
        <div class="stats-grid">
            <div class="stat-card">
                <span>Увійшло</span>
                <strong id="total-entered">0</strong>
            </div>
            <div class="stat-card">
                <span>Вийшло</span>
                <strong id="total-exited">0</strong>
            </div>
```

```
                        <div class="stat-card">
                            <span>Зараз у приміщенні</span>
                            <strong id="occupancy">0</strong>
                        </div>
                        <div class="stat-card">
                            <span>Ще евакуювати</span>
                            <strong id="remaining">0</strong>
                        </div>
                    </div>

                    <div class="evacuation-boxes">
                        <div class="evacuation-box">
                            <span>Режим евакуації</span>
                            <strong id="evacuation-state">Неактивна</strong>
                        </div>
                        <div class="evacuation-box success">
                            <span>Чи всі вийшли</span>
                            <strong id="all-evacuated">Ні</strong>
                        </div>
                        <div class="evacuation-box info">
                            <span>Напрямок входу</span>
                            <strong id="direction-summary">Вхід знизу
вгору</strong>
                        </div>
                    </div>

                    <div class="last-event">
                        <span>Остання подія</span>
                        <p id="last-event">Система очікує запуску</p>
                    </div>
                </article>
            </section>

            <!-- Налаштування лінії -->
            <section class="panel tuning-panel">
                <div class="panel-head">
                    <h2>Налаштування лінії підрахунку</h2>
                    <span class="panel-note"
                        >Рухайте лінію вручну під свою камеру</span
                    >
                </div>
                <div class="line-tuning">
                    <div class="direction-block">
                        <button class="btn btn-secondary" id="direction-toggle-
button" type="button">
                            Змінити вхід/вихід
                        </button>
                    </div>
                    <div class="slider-block">
                        <label for="line-position-slider"
                            >Положення червоної лінії</label
                        >
```

```
                <input
                    id="line-position-slider"
                    type="range"
                    min="20"
                    max="300"
                    value="120"
                />
            </div>
            <div class="line-values">
                <div class="line-value-card">
                    <span>Поточне значення</span>
                    <strong id="line-position-value">120 px</strong>
                </div>
                <div class="line-value-card">
                    <span>Відносно висоти кадру</span>
                    <strong id="line-position-percent">0%</strong>
                </div>
            </div>
            <div class="line-actions">
                <p class="line-help">
                    Якщо для іншого відео вхід і вихід поміняліcь місцями,
                    натисніть кнопку зміни напрямку. Положення лінії
                    зберігається навіть після повторного старту відео.
                </p>
            </div>
        </div>
    </section>

    <!-- Журнал подій -->
    <section class="panel table-panel">
        <div class="panel-head">
            <h2>Журнал останніх подій</h2>
            <span class="panel-note">Вхід, вихід і стан евакуації</span>
        </div>
        <div class="table-wrap">
            <table>
                <thead>
                    <tr>
                        <th>Час</th>
                        <th>ID</th>
                        <th>Подія</th>
                        <th>У приміщенні</th>
                        <th>Ще евакуювати</th>
                    </tr>
                </thead>
                <tbody id="events-body">
                    <tr>
                        <td colspan="5" class="empty-state">
                            Подій ще немає
                        </td>
                    </tr>
                </tbody>
```

```
                    </table>
                </div>
            </section>
        </div>

        <div class="toast" id="toast"></div>

        <!-- JS-логіка -->
        <script src="{{ url_for('static', filename='app.js') }}"></script>
    </body>
</html>
```

app.js

```
// API-адреси
const statusUrl = "/api/status";
const controlRoutes = {
  start: "/api/control/start",
  stop: "/api/control/stop",
  reset: "/api/control/reset",
  line: "/api/control/line",
  direction: "/api/control/direction",
  "evac-start": "/api/control/evacuation/start",
  "evac-stop": "/api/control/evacuation/stop",
};

// DOM-елементи
const elements = {
  processingState: document.getElementById("processing-state"),
  occupancyHighlight: document.getElementById("occupancy-highlight"),
  totalEntered: document.getElementById("total-entered"),
  totalExited: document.getElementById("total-exited"),
  occupancy: document.getElementById("occupancy"),
  remaining: document.getElementById("remaining"),
  evacuationState: document.getElementById("evacuation-state"),
  allEvacuated: document.getElementById("all-evacuated"),
  lastEvent: document.getElementById("last-event"),
  eventsBody: document.getElementById("events-body"),
  lineSlider: document.getElementById("line-position-slider"),
  linePositionValue: document.getElementById("line-position-value"),
  linePositionPercent: document.getElementById("line-position-percent"),
  directionSummary: document.getElementById("direction-summary"),
  directionToggleButton: document.getElementById("direction-toggle-button"),
  toast: document.getElementById("toast"),
};

// Стан інтерфейсу
let toastTimer;
let lineSliderDirty = false;
let lineUpdateInFlight = false;
let lineApplyTimer;

// Повідомлення
```

```
function showToast(message) {
  elements.toast.textContent = message;
  elements.toast.classList.add("visible");
  window.clearTimeout(toastTimer);
  toastTimer = window.setTimeout(() => {
    elements.toast.classList.remove("visible");
  }, 2800);
}

// Таблиця подій
function updateTable(events) {
  if (!events || events.length === 0) {
    elements.eventsBody.innerHTML =
      '<tr><td colspan="5" class="empty-state">Подій ще немає</td></tr>';
    return;
  }

  const rows = events
    .map(
      (event) => `
        <tr>
          <td>${event.timestamp}</td>
          <td>${event.track_id}</td>
          <td>${event.direction === "Entry" ? "Вхід" : "Вихід"}</td>
          <td>${event.occupancy}</td>
          <td>${event.remaining_to_evacuate}</td>
        </tr>
      `
    )
    .join("");

  elements.eventsBody.innerHTML = rows;
}

// Оновлення статусу
function renderStatus(data) {
  elements.processingState.textContent = data.processing_active
    ? "Аналіз виконується"
    : "Очікує запуску";
  elements.occupancyHighlight.textContent = data.occupancy;
  elements.totalEntered.textContent = data.total_entered;
  elements.totalExited.textContent = data.total_exited;
  elements.occupancy.textContent = data.occupancy;
  elements.remaining.textContent = data.remaining_to_evacuate;
  elements.evacuationState.textContent = data.evacuation_active ? "Активна" :
"Неактивна";
  elements.allEvacuated.textContent = data.evacuation_completed ? "Так" : "Ні";
  elements.lastEvent.textContent = data.last_event;
  if (elements.lineSlider) {
    elements.lineSlider.max = Math.max(40, data.frame_height || 300);
    if (!lineSliderDirty && !lineUpdateInFlight) {
      elements.lineSlider.value = data.line_position;
```

```
      elements.linePositionValue.textContent = `${data.line_position} px`;
    }
    elements.linePositionPercent.textContent = `${data.line_position_percent}%`;
  }
  if (elements.directionSummary) {
    elements.directionSummary.textContent = data.entry_direction_label;
    elements.directionToggleButton.textContent =
      data.entry_direction === "down"
        ? "Змінити: вхід вниз"
        : "Змінити: вхід вгору";
    elements.directionToggleButton.dataset.currentDirection =
data.entry_direction;
  }
  updateTable(data.recent_events);
}

// Запит статусу
async function refreshStatus() {
  try {
    const response = await fetch(statusUrl, { cache: "no-store" });
    if (!response.ok) {
      throw new Error("Не вдалося отримати статус системи");
    }

    const data = await response.json();
    renderStatus(data);
  } catch (error) {
    showToast(error.message);
  }
}

// Команди керування
async function sendControl(action) {
  const route = controlRoutes[action];
  if (!route) {
    return;
  }

  try {
    const response = await fetch(route, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
    });

    const data = await response.json();
    if (!response.ok || !data.ok) {
      throw new Error(data.message || "Операцію не виконано");
    }

    showToast(data.message);
    await refreshStatus();
  } catch (error) {
```

```
    showToast(error.message);
  }
}

// Зміна лінії
async function applyLinePosition() {
  try {
    lineUpdateInFlight = true;
    const response = await fetch(controlRoutes.line, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({
        line_position: Number(elements.lineSlider.value),
      }),
    });

    const data = await response.json();
    if (!response.ok || !data.ok) {
      throw new Error(data.message || "Не вдалося оновити лінію");
    }

    lineSliderDirty = false;
    await refreshStatus();
  } catch (error) {
    showToast(error.message);
  } finally {
    lineUpdateInFlight = false;
  }
}

// Зміна напряму
async function toggleDirection() {
  const currentDirection = elements.directionToggleButton.dataset.currentDirection
|| "up";
  const nextDirection = currentDirection === "up" ? "down" : "up";

  try {
    const response = await fetch(controlRoutes.direction, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({
        entry_direction: nextDirection,
      }),
    });

    const data = await response.json();
    if (!response.ok || !data.ok) {
      throw new Error(data.message || "Не вдалося оновити напрямок");
    }

    showToast(data.message);
    await refreshStatus();
```

```
  } catch (error) {
    showToast(error.message);
  }
}

// Обробники кнопок
document.querySelectorAll("[data-action]").forEach((button) => {
  button.addEventListener("click", () => {
    sendControl(button.dataset.action);
  });
});

// Повзунок лінії
if (elements.lineSlider) {
  elements.lineSlider.addEventListener("input", () => {
    lineSliderDirty = true;
    elements.linePositionValue.textContent = `${elements.lineSlider.value} px`;
    window.clearTimeout(lineApplyTimer);
    lineApplyTimer = window.setTimeout(() => {
      applyLinePosition();
    }, 180);
  });
}

// Кнопка напряму
if (elements.directionToggleButton) {
  elements.directionToggleButton.addEventListener("click", toggleDirection);
}

// Автооновлення
refreshStatus();
window.setInterval(refreshStatus, 1000);
```

styles.css

```
:root {
  --bg: #07111f;
  --bg-soft: #0d1b2d;
  --panel: rgba(10, 22, 38, 0.78);
  --panel-border: rgba(157, 196, 255, 0.16);
  --text: #f4f7fb;
  --muted: #9fb1c8;
  --accent: #53e0b0;
  --accent-strong: #13c38b;
  --warning: #ffb347;
  --danger: #ff6f61;
  --shadow: 0 24px 60px rgba(0, 0, 0, 0.35);
}

* {
  box-sizing: border-box;
}
```

```
body {
  margin: 0;
  min-height: 100vh;
  font-family: "Avenir Next", "Segoe UI", sans-serif;
  color: var(--text);
  background:
    radial-gradient(circle at top left, rgba(83, 224, 176, 0.18), transparent
28%),
    radial-gradient(circle at top right, rgba(103, 144, 255, 0.14), transparent
32%),
    linear-gradient(180deg, #08111e 0%, #0a1727 46%, #07101d 100%);
}

.page-shell {
  width: min(1360px, calc(100% - 32px));
  margin: 0 auto;
  padding: 28px 0 40px;
}

.hero {
  display: grid;
  grid-template-columns: 1.5fr 0.9fr;
  gap: 24px;
  align-items: stretch;
  margin-bottom: 24px;
}

.hero-copy,
.hero-panel,
.panel,
.controls {
  backdrop-filter: blur(22px);
  background: var(--panel);
  border: 1px solid var(--panel-border);
  border-radius: 28px;
  box-shadow: var(--shadow);
}

.hero-copy {
  padding: 32px;
}

.eyebrow {
  margin: 0 0 14px;
  color: var(--accent);
  text-transform: uppercase;
  letter-spacing: 0.18em;
  font-size: 0.78rem;
  font-weight: 700;
}

.hero h1 {
```

```
  margin: 0 0 14px;
  font-size: clamp(2rem, 3.6vw, 3.7rem);
  line-height: 1.03;
  max-width: 12ch;
}

.hero-text {
  margin: 0;
  color: var(--muted);
  max-width: 60ch;
  font-size: 1.02rem;
  line-height: 1.65;
}

.hero-panel {
  display: grid;
  gap: 16px;
  padding: 24px;
}

.hero-badge {
  padding: 22px;
  border-radius: 22px;
  background: rgba(255, 255, 255, 0.04);
  border: 1px solid rgba(255, 255, 255, 0.05);
}

.hero-badge.accent {
  background: linear-gradient(135deg, rgba(83, 224, 176, 0.16), rgba(83, 224, 176,
0.05));
  border-color: rgba(83, 224, 176, 0.24);
}

.hero-badge span,
.stat-card span,
.evacuation-box span,
.last-event span {
  display: block;
  color: var(--muted);
  font-size: 0.9rem;
  margin-bottom: 10px;
}

.hero-badge strong {
  font-size: 1.8rem;
}

.controls {
  display: flex;
  flex-wrap: wrap;
  gap: 12px;
  padding: 18px;
```

```
  margin-bottom: 24px;
}

.btn {
  border: 0;
  border-radius: 999px;
  padding: 12px 18px;
  font: inherit;
  font-weight: 700;
  color: var(--text);
  background: rgba(255, 255, 255, 0.08);
  cursor: pointer;
  transition: transform 0.18s ease, background 0.18s ease, opacity 0.18s ease;
}

.btn:hover {
  transform: translateY(-1px);
  background: rgba(255, 255, 255, 0.13);
}

.btn-primary {
  background: linear-gradient(135deg, var(--accent-strong), var(--accent));
  color: #072015;
}

.btn-warning {
  background: linear-gradient(135deg, #ff9d3b, #ffd16a);
  color: #331800;
}

.btn-secondary {
  background: linear-gradient(135deg, #72a8ff, #a0d7ff);
  color: #08213f;
}

.btn-danger {
  background: linear-gradient(135deg, #ff6f61, #ff8c7e);
  color: #2f0905;
}

.dashboard-grid {
  display: grid;
  grid-template-columns: 1.3fr 0.9fr;
  gap: 24px;
  margin-bottom: 24px;
}

.tuning-panel {
  margin-bottom: 24px;
}

.panel {
```

```
  padding: 22px;
}

.panel-head {
  display: flex;
  justify-content: space-between;
  align-items: baseline;
  gap: 12px;
  margin-bottom: 18px;
}

.panel-head h2 {
  margin: 0;
  font-size: 1.3rem;
}

.panel-note {
  color: var(--muted);
  font-size: 0.9rem;
}

.video-frame {
  overflow: hidden;
  border-radius: 22px;
  background: #020812;
  border: 1px solid rgba(255, 255, 255, 0.06);
  aspect-ratio: 16 / 9;
}

.video-frame img {
  width: 100%;
  height: 100%;
  object-fit: cover;
  display: block;
}

.stats-grid {
  display: grid;
  grid-template-columns: repeat(2, minmax(0, 1fr));
  gap: 14px;
}

.stat-card,
.evacuation-box,
.last-event {
  border-radius: 22px;
  padding: 18px;
  background: rgba(255, 255, 255, 0.04);
  border: 1px solid rgba(255, 255, 255, 0.06);
}

.stat-card strong,
```

```
.evacuation-box strong {
  font-size: clamp(1.7rem, 2vw, 2.2rem);
}

.evacuation-boxes {
  display: grid;
  gap: 14px;
  margin: 18px 0;
}

.evacuation-box.success {
  background: linear-gradient(135deg, rgba(83, 224, 176, 0.14), rgba(83, 224, 176,
0.04));
  border-color: rgba(83, 224, 176, 0.2);
}

.evacuation-box.info {
  background: linear-gradient(135deg, rgba(114, 168, 255, 0.16), rgba(160, 215,
255, 0.04));
  border-color: rgba(114, 168, 255, 0.24);
}

.last-event p {
  margin: 0;
  font-size: 1.06rem;
  line-height: 1.5;
}

.line-tuning {
  display: grid;
  gap: 18px;
}

.direction-block {
  display: flex;
  flex-wrap: wrap;
  justify-content: flex-start;
  gap: 14px;
  align-items: center;
  border-radius: 20px;
  padding: 16px 18px;
  background: rgba(255, 255, 255, 0.04);
  border: 1px solid rgba(255, 255, 255, 0.06);
}

.slider-block {
  display: grid;
  gap: 10px;
}

.slider-block label {
  color: var(--muted);
```

```
  font-size: 0.95rem;
}

.slider-block input[type="range"] {
  width: 100%;
  accent-color: var(--accent);
}

.line-values {
  display: grid;
  grid-template-columns: repeat(2, minmax(0, 1fr));
  gap: 14px;
}

.line-value-card {
  border-radius: 20px;
  padding: 16px 18px;
  background: rgba(255, 255, 255, 0.04);
  border: 1px solid rgba(255, 255, 255, 0.06);
}

.line-value-card span {
  display: block;
  color: var(--muted);
  margin-bottom: 8px;
  font-size: 0.9rem;
}

.line-value-card strong {
  font-size: 1.45rem;
}

.line-actions {
  display: flex;
  flex-wrap: wrap;
  gap: 14px;
  align-items: center;
}

.line-help {
  margin: 0;
  color: var(--muted);
  max-width: 58ch;
  line-height: 1.5;
}

.table-wrap {
  overflow-x: auto;
}

table {
  width: 100%;
```

```
  border-collapse: collapse;
}

thead th {
  text-align: left;
  color: var(--muted);
  font-size: 0.86rem;
  letter-spacing: 0.05em;
  text-transform: uppercase;
  padding: 0 12px 14px;
}

tbody td {
  padding: 14px 12px;
  border-top: 1px solid rgba(255, 255, 255, 0.08);
}

tbody tr:hover {
  background: rgba(255, 255, 255, 0.03);
}

.empty-state {
  text-align: center;
  color: var(--muted);
  padding: 28px 12px;
}

.toast {
  position: fixed;
  right: 20px;
  bottom: 20px;
  min-width: 260px;
  max-width: 420px;
  padding: 14px 18px;
  border-radius: 18px;
  background: rgba(6, 17, 30, 0.92);
  color: var(--text);
  border: 1px solid rgba(83, 224, 176, 0.18);
  box-shadow: var(--shadow);
  opacity: 0;
  transform: translateY(12px);
  pointer-events: none;
  transition: opacity 0.25s ease, transform 0.25s ease;
}

.toast.visible {
  opacity: 1;
  transform: translateY(0);
}

@media (max-width: 1024px) {
  .hero,
```

```
  .dashboard-grid {
    grid-template-columns: 1fr;
  }
}

@media (max-width: 720px) {
  .page-shell {
    width: min(100% - 20px, 1360px);
    padding-top: 18px;
  }

  .hero-copy,
  .hero-panel,
  .panel,
  .controls {
    border-radius: 22px;
  }

  .hero-copy,
  .panel {
    padding: 18px;
  }

  .stats-grid {
    grid-template-columns: 1fr;
  }

  .line-values {
    grid-template-columns: 1fr;
  }

  .controls {
    padding: 14px;
  }

  .btn {
    width: 100%;
  }

  .direction-block {
    align-items: stretch;
  }
}
```

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Артем КОРОЛЬЧУК

Співавтор:

Назва: Кіберфізична система контролю евакуації людей з приміщень організації

Експерт: Марія КАПУСТЯН

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:6%

Коефіцієнт подібності 2:2.37%

Мікропробіли: 3

Заміна букв: 0

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2026-06-02 07:39:27.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2026-06-02 Доцент Андрій Нічепорук

Дата експерт

Anti-Plagiarism (http://ap.km.ua) v-15.701

Максимальне співпадіння з одним документом 8.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

ID: 273059 Назва: БКР Кіберфізична система контролю евакуації людей з приміщень організації Додано в БД: 2026-06-02 Автора: Артем КОРОЛЬЧУК Керівники: Марія КАПУСТЯН Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	87365	698	9358 (11%)	91 (13%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Корольчук Артем Вадимович

Тема: Кіберфізична система контролю евакуації людей з приміщень організації

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки 56

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є проєктування, реалізація та тестування програмно-технічного засобу для виявлення, відстеження, підрахунку людей та візуалізації результатів моніторингу у реальному часі

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі кваліфікаційної роботи проведено дослідження предметної області автоматизованого відеомоніторингу та контролю евакуації людей з приміщень організації. Виконано аналіз існуючих підходів до підрахунку кількості людей у приміщенні, сучасних методів комп'ютерного зору для виявлення та відстеження людей у відеопотоці, а також сформульовано постановку задачі дослідження та визначено основні вимоги до програмно-технічного засобу. В другому розділі кваліфікаційної роботи виконано проєктування кіберфізичної системи контролю евакуації людей з приміщень організації. Визначено склад апаратних і програмних підсистем, розроблено структурну схему системи та схему взаємодії її компонентів. Описано алгоритм виявлення, відстеження та підрахунку людей у відеопотоці, а також обґрунтовано вибір методів і засобів комп'ютерної інженерії для реалізації системи, зокрема використання алгоритмів комп'ютерного зору, моделі детекції людей та вебтехнологій для відображення результатів. В третьому розділі кваліфікаційної роботи виконано програмну реалізацію кіберфізичної системи контролю евакуації

людей з приміщень організації. Реалізовано модулі обробки відеопотоку, виявлення людей, відстеження їх переміщення, підрахунку входу та виходу, контролю процесу евакуації, журналу подій та вебінтерфейсу користувача. Проведено тестування системи на реальному відеоматеріалі та виконано оцінювання результатів її роботи в умовах інтенсивного потоку людей. Під час виконання роботи використано сучасні засоби комп'ютерного зору, методи аналізу відеоданих та підходи до побудови веборієнтованих програмних систем.

4. Позитивні сторони роботи: висока практична цінність роботи, актуальність тематики дослідження, використання сучасних методів комп'ютерного зору та реалізація працездатного програмно-технічного засобу з можливістю практичного застосування.

5. Негативні сторони роботи: у роботі обмежено розглянуто питання підвищення точності роботи системи в умовах щільного потоку людей, часткових перекриттів об'єктів та складних умов відеоспостереження, що може бути напрямом подальшого розвитку дослідження.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре (B / 86)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Яшина О.М., доцент кафедри УІТ

"01" 06 2026 р. _____(підпис)

Зав. кафедри КІС
д-р. філософії Ользі ПАВЛОВІЙ

Артем КОРОЛЬЧУК
ПІБ здобувача вищої освіти

ФІТ, 3 курсу, групи КІ2с-23-2

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

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

1 червня 2026 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Кіберфізична система контролю евакуації людей з приміщень організації
Автор Артем КОРОЛЬЧУК
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти перший (бакалаврський)
Спеціальність 123 Комп'ютерна інженерія
Науковий керівник: к.т.н., доцент Марія КАПУСТЯН

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;

3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 6% і адресується до 23 першоджерела; та системою Anti-Plagiarism складає 8%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

02.06.2026

Завідувач кафедри — Підпис — Ольга ПАВЛОВА (Ім'я, ПРІЗВИЩЕ)

Гарант освітньої програми — Підпис — Ольга ПАВЛОВА (Ім'я, ПРІЗВИЩЕ)

Керівник кваліфікаційної роботи — Підпис — Марія КАПУСТЯН (Ім'я, ПРІЗВИЩЕ)