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

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Система моніторингу уваги людини з використанням аналізу відеопотоку в реальному часі
Назва теми

КвРКІ 210124.21.01.72 ПЗ
Шифр

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

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

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

Виконав: студент IV курсу, група КІ2-21-1 _____ Андрій СЕВАСТЬЯНОВ
Підпис — Ініціали, прізвище

Керівник _____ Володимир КИСІЛЬ
Підпис, дата — Ініціали, прізвище

Нормоконтролер _____ Тетяна КИСІЛЬ
Підпис, дата — Ініціали, прізвище

До захисту допускаю:
зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Ольга ПАВЛОВА
Підпис — Ініціали, прізвище

«12» червня 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень Бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Ольга ПАВЛОВА

" 10 " 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Севастьянову Андрію Юрійовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Система моніторингу уваги людини з використанням аналізу відеопотоку в реальному часі

Керівник проекту (роботи) Кисіль Володимир Володимирович, асистент

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 07.02.2025 р. № 23

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2025 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

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

Система моніторингу уваги людини на основі відеоаналізу: загальний опис, призначення та обґрунтування актуальності.

Проєктування моніторингу уваги людини у кіберфізичній системі на базі контролю погляду в умовах реального часу.

Програмно-апаратна реалізація кіберфізичної системи моніторингу уваги людини.

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

Апаратні з'єднання компонентів

Блок-схеми алгоритму роботи програми

Діаграма послідовності для системи моніторингу уваги водія

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата завдання видав	завдання прийня
Нормоконтроль	Тетяна КИСІЛЬ, доцент кафедри КІС		
Антиплагіат	Андрій НІЧЕПОРУК, доцент кафедри КІС		

7. Дата видачі завдання «10» 01 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примі
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2025	виконан
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2025	виконан
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2025	виконан
4	Робота над розділом 2 – вибір компонентів для проєктування системи моніторингу уваги людини	01.04.2025	виконан
5	Робота над розділом 3 – проєктування системи моніторингу уваги людини з аналізом відеопотоку в реальному часі	29.04.2025	виконан
6	Оформлення пояснювальної записки згідно вимог	25.05.2025	виконан
7	Попередній захист ВКР	26.05.2025	виконан
8	Захист ВКР на засіданні ЕК	Червень 2025 року	

Студент _____ Андрій СЕВАСТЬЯНОВ
Підпис Ініціали, прізвище

Керівник роботи _____ Володимир КИСІЛЬ
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ скз	Примітка
			Текстові документи			
1		КвРКІ 210124.21.01.72 ПЗ	Пояснювальна записка	63		
			Графічні матеріали			
2		КвРКІ 210124.21.01.72 Е8	Апаратні з'єднання	1		
			компонентів			
3		КвРКІ 210124.21.01.72 Е8	Блок-схеми алгоритму	1		
			роботи програми			
4		КвРКІ 210124.21.01.72 Е8	Діаграма послідовності	1		
			для системи моніторингу			
			уваги водія			

КвРКІ 210124.21.01.72 ВП

Зм	Арк	№ докум	Підпис	Дата	Відомість проекту	Літера	Аркуш	Аркушів
Розробив		Севастьянов				У	1	1
Перевір.		Кисіль						
Н. контр.		Кисіль				ХНУ, КІ2-21-1		
Затв.		Павлова						

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Система моніторингу уваги людини з використанням аналізу відеопотоку в реальному часі».

Автор роботи: Андрій СЕВАСТЬЯНОВ.

Керівник роботи: Володимир КИСІЛЬ.

Пояснювальна записка: 63 с., 30 рис., 2 табл., 4 дод., 56 джерел.

Графічна частина: 3 креслення.

СИСТЕМА МОНІТОРИНГУ УВАГИ, КІБЕРФІЗИЧНА СИСТЕМА, АНАЛІЗ ВІДЕОПОТОКУ, ВИЯВЛЕННЯ ВІДВОЛІКАННЯ, RASPBERRY PI.

Метою дипломної роботи є розробка та дослідження ефективності системи моніторингу уваги людини на основі аналізу відеопотоку в реальному часі для своєчасного виявлення ознак зниження концентрації або відволікання.

Об'єктом дослідження є функціонування системи моніторингу уваги людини, яка використовує методи комп'ютерного зору.

Предметом дослідження є оцінка ефективності та швидкодії методів і алгоритмів аналізу відеопотоку, реалізованих на мікрокомп'ютері Raspberry Pi, для визначення стану уваги людини в умовах реального часу.

Під час проведення даного дослідження був використаний підхід системного проектування та розробки програмного забезпечення, а також методи експериментальної оцінки для вивчення продуктивності системи в реальних умовах.

Підпис студента

30.05.2025

Дата

ЗМІСТ

ВСТУП

Забезпечення високого рівня уваги та концентрації є критично важливим у багатьох сферах професійної діяльності, де помилки, спричинені неуважністю, можуть мати серйозні наслідки для безпеки та ефективності. Особливо це стосується таких областей, як керування транспортними засобами, робота зі складним промисловим обладнанням або виконання завдань, що вимагають тривалої та безперервної зосередженості. Зниження рівня концентрації або відволікання уваги є однією з ключових причин аварій, виробничих інцидентів та зниження загальної продуктивності. Статистичні дані свідчать, що людський фактор, зокрема втрата концентрації, стає причиною більшості нещасних випадків у транспортній та промисловій галузях, що підкреслює гостру необхідність впровадження систем контролю стану операторів.

У зв'язку з цим зростає потреба в розробці надійних та доступних методів і інструментів для об'єктивного моніторингу стану уваги людини. Традиційні підходи, засновані на суб'єктивній оцінці або періодичному тестуванні, не забезпечують безперервного контролю та здатні пропустити критичні моменти зниження концентрації. Водночас методи, що базуються на фізіологічних показниках, часто вимагають спеціального обладнання, яке може бути дороговартісним та незручним у повсякденному використанні.

Сучасний розвиток комп'ютерного зору та доступність компактних, але достатньо потужних обчислювальних платформ відкривають нові можливості для створення автоматизованих систем контролю уваги. Використання відеоаналізу дозволяє безконтактно оцінювати візуальні ознаки стану людини, як-от: положення голови, очей, міміка. Аналіз цих параметрів в динаміці дозволяє з високою точністю визначати моменти зниження концентрації та передбачати потенційно небезпечні стани ще до виникнення критичної ситуації.

Реалізація на вбудованих системах робить такі рішення портативними та економічно доцільними для потенційного широкого застосування. Переваги такого

підходу полягають у можливості інтеграції системи моніторингу безпосередньо в робоче місце оператора без суттєвих змін існуючої інфраструктури. Крім того, локальна обробка даних забезпечує конфіденційність та зменшує вимоги до пропускної здатності комунікаційних каналів, що особливо важливо для мобільних застосувань або віддалених об'єктів. Розробка ефективної системи, здатної функціонувати в реальному часі на обмежених ресурсах, є актуальним науково-технічним завданням. Воно вимагає оптимального поєднання точності розпізнавання та обчислювальної ефективності, а також вирішення ряду супутніх проблем, включаючи енергоефективність, стійкість до різних умов експлуатації та мінімізацію хибних спрацьовувань.

Метою дипломної роботи є розробка, реалізація та експериментальне дослідження програмно-апаратної системи моніторингу уваги людини на основі аналізу відеопотоку в реальному часі. Центральним завданням є створення комплексу, здатного вчасно виявляти типові ознаки зниження концентрації уваги або відволікання оператора чи користувача, наприклад: сонливість, тривалий погляд вбік, часте кліпання, забезпечуючи можливість своєчасного оповіщення для запобігання небезпечним ситуаціям та підвищення безпеки.

Об'єктом дослідження є функціонування програмно-апаратної системи моніторингу уваги людини, що базується на відеоаналізі. Предметом дослідження виступають методи та алгоритми комп'ютерного зору для визначення стану уваги людини, їх оптимізація для роботи на обмежених обчислювальних ресурсах та інтеграція в єдину ефективну систему моніторингу.Предметом дослідження є методи та алгоритми аналізу відеопотоку для визначення стану уваги людини, а також оцінка їх ефективності, швидкодії та придатності для реалізації на вбудованій платформі Raspberry Pi в умовах реального часу.

Під час проведення даного дослідження був використаний підхід системного проектування, методи обробки зображень та комп'ютерного зору, а також методи експериментальної оцінки продуктивності та надійності розробленої системи.

1 КІБЕРФІЗИЧНА СИСТЕМА МОНІТОРИНГУ УВАГИ ЛЮДИНИ З ВИКОРИСТАННЯМ АНАЛІЗУ ВІДЕОПОТОКУ В РЕАЛЬНОМУ ЧАСІ ТА ПОСТАНОВКА ЗАДАЧІ ЩОДО ЇЇ РЕАЛІЗАЦІЇ

1.1 Аналіз предметної області і виявлення наявних проблем і завдань

Існує багато сфер діяльності людини, де увага над процесом діяльності відіграє ключову роль. Це стосується як водіїв, так і працівників на небезпечних виробничих лініях, операторів промислового обладнання, диспетчерів у критичних системах, а також інших спеціалістів, чия діяльність вимагає постійної концентрації. У подібних умовах навіть короткочасна втрата уваги може мати серйозні наслідки - травмування самого працівника, загроза життю оточуючих, пошкодження обладнання, екологічні катастрофи або людські жертви.

У транспортній галузі, зокрема у водіїв вантажівок, автобусів чи таксі, сонливість або відволікання - одна з основних причин аварій. У промисловості - працівник, що втратив концентрацію, може неправильно взаємодіяти з обладнанням, активувати механізми у невірний момент або не помітити критичний збій. Це особливо небезпечно в умовах безперервного виробництва, де людська участь є вирішальною.

Усі ці задачі об’єднує одна спільна мета - виявити неуважність або втому якомога раніше, до того як це призведе до негативних наслідків. Саме тому виникає потреба в автоматизованих, надійних і адаптивних системах, які можуть працювати в реальному часі, розпізнавати ознаки втрати уваги та застосовувати превентивні заходи, наприклад, застереження звуковим сигналом або навіть зупинка роботи обладнання.

Сучасні досягнення у сфері комп’ютерного зору, зокрема використання бібліотек OpenCV [8, 20, 21], dlib [25], MediaPipe [27], а також легких нейронних мереж, наприклад, TensorFlow Lite, відкривають можливості для створення подібних систем. Вони можуть функціонувати на доступному апаратному забезпеченні, наприклад, на Raspberry Pi, що робить їх дешевими,

енергоефективними та зручними у встановленні як у транспортних засобах, так і на виробництві.

Таким чином, створення системи моніторингу уваги на основі аналізу відеопотоку в реальному часі є актуальним завданням, яке може значно покращити безпеку в різних сферах діяльності. У центрі такого підходу - ідея, що людина не завжди може самостійно контролювати рівень своєї зосередженості, тому технології мають її в цьому підтримати.

У процесі проектування подібної системи необхідно враховувати особливості кожного середовища, в якому вона буде використовуватися. Наприклад, для водія транспортного засобу було б ідеально виявляти моменти, коли він відволікається від дороги, закриває очі на тривалий час, нахиляє голову або дивиться вбік - і в таких випадках подавати звукове або вібраційне застереження [1, 5, 30]. Це дозволяє оперативно повернути увагу водія до дороги та зменшити ймовірність аварії.

Однак, варто підкреслити, що контролювати увагу водія, коли авто нерухоме - недоцільно. У таких ситуаціях, наприклад на світлофорі або в заторі, водій має право на короткий відпочинок, і сигнали тривоги в цей момент можуть лише спричинити стрес чи роздратування. Тому система має враховувати умови руху транспортного засобу. Це можна реалізувати за допомогою акселерометра, GPS, або ж підключенням до CAN-шини авто для визначення швидкості. Таким чином, контроль уваги активується лише під час руху.

У протилежному прикладі - на виробничій лінії навпаки, контроль має бути постійним, оскільки працівник взаємодіє з потенційно небезпечним обладнанням, і навіть коротке відволікання може призвести до травм або поломки. Тут тригером для активації системи може слугувати стан увімкнення верстата, датчики присутності, або сигнал від виробничої системи. Система повинна починати моніторинг лише тоді, коли працівник дійсно виконує активні дії, і завершувати спостереження, коли зміна закінчилась або устаткування не працює.

Виходячи з цього, можна сформувати вимогу до архітектури системи - вона має бути модульною, адаптивною та легко модифікованою під різні сценарії використання.

У кожному випадку система має спиратися на єдине ядро, яке відповідальне за:

– Виявлення неуважності (на основі аналізу відео: напрям погляду, моргання, положення голови).

– Фіксацію інцидентів та збереження статистики.

– Виведення повідомлень або запуск сигналів тривоги.

Проте при цьому до ядра підключається контекстний модуль, який визначає "умови для активації контролю". Це може бути:

– Модуль руху для авто (GPS, акселерометр, швидкість) [35, 36].

– Модуль активності обладнання на виробництві (стан увімкнення/вимкнення), або навіть модуль розкладу чергування (для персоналу, що працює за графіком).

Така архітектура дозволяє легко масштабувати систему на інші сфери, додаючи лише нові модулі активації, не змінюючи базову логіку ядра. Наприклад, система може однаково добре працювати:

– в дорожньому транспорті: активується при початку руху;

– в цехах і підприємствах: активується при включенні станка;

– диспетчерській: активна весь час, але знижує чутливість у нічний період;

– медичному закладі: контролює стан лікаря або персоналу в критичних відділеннях (реанімація, операційна).

Крім того, система може мати механізм фізичного втручання або зворотного зв’язку. Якщо користувач отримує попередження про втрату уваги, а поведінка не змінюється, система може:

– Активувати резервне джерело сигналу (гучніший звук, вібрація).

– Повідомити старшого зміну або іншу особу.

– Зупинити устаткування або заблокувати керування (за певних умов, наприклад, у транспорті - за допомогою зовнішньої інтеграції).

У перспективі, така система може бути централізовано підключена до серверної частини, яка буде збирати статистику за днями, змінами, типами помилок. Це дає змогу не тільки підвищити безпеку, але й аналізувати загальну ефективність роботи, виявляти періоди зниження уваги, оптимізувати навантаження та прогнозувати ризики ще до того, як вони переростуть у проблеми.

Тобто саме модульність і відкритість до конфігурацій і нових змін такої системи надають їй великих перспектив у великих проектах де можуть бути інтегровані десятки компонентів. Можливість створення модифікації для будь яких умов експлуатації роблять таку систему відкритою до змін. Ця гнучкість досягається за рахунок архітектури, що складається з взаємозамінних та конфігурованих функціональних блоків [15]. Зокрема, можна виділити модуль аналізу візуальних даних, відповідальний за розпізнавання ознак неуважності (стан очей, положення голови, напрям погляду). Цей модуль може використовувати різні алгоритми або навіть моделі машинного навчання, оптимізовані під конкретні умови освітлення, якість зображення або тип камери. Поруч функціонує модуль контексту, який інтегрується з зовнішніми джерелами інформації, що характеризують поточне операційне середовище - це можуть бути дані про швидкість транспортного засобу, статус роботи промислового обладнання, параметри мікроклімату тощо. Нарешті, модуль реагування визначає тип та інтенсивність зворотного зв'язку - від візуальних та звукових сповіщень до активації зовнішніх систем (наприклад, блокування керування або зупинка обладнання).

Таким чином, система моніторингу уваги розглядається як набір адаптивних компонентів, які можуть бути налаштовані та скомбіновані для формування специфічної логіки роботи, що повністю відповідає унікальним вимогам та умовам експлуатації в будь-якій галузі, де потрібен такий контроль. Це забезпечує безпрецедентну гнучкість у впровадженні та масштабуванні рішення.

1.2 Порівняльний аналіз переваг та недоліків існуючих рішень

У ході даної роботи система моніторингу уваги розглядатиметься переважно в контексті використання для водіїв транспортних засобів, оскільки саме ця сфера на сьогодні є однією з найактуальніших у плані попиту та практичного застосування подібних технологій. Втома, сонливість або неуважність водія є основними причинами дорожньо-транспортних пригод, особливо під час тривалих рейсів, нічного водіння або при монотонному русі автомагістралями [1, 2]. Саме тому більшість існуючих рішень орієнтовані на контроль стану водія, а завдання створення доступної, автономної та надійної системи моніторингу уваги є актуальним як з технічної, так і з соціальної точки зору.

Системи моніторингу уваги водія належать до класу допоміжних інтелектуальних систем безпеки (ADAS - Advanced Driver Assistance Systems та DAA - Driver Attention Alert) [18], основна функція яких - виявлення ознак втоми, неуважності або втрати контролю над дорожньою ситуацією. Такі системи реалізуються на основі аналізу візуальної інформації, отриманої з відеокамер, у поєднанні з алгоритмами комп'ютерного зору та машинного навчання, яке в свою чергу, дає змогу системі навчитися розпізнавати закономірності у поведінці водія на основі великої кількості прикладів: наприклад, визначати ознаки сонливості (через індикатори на кшталт PERCLOS), виявляти відволікання або розпізнавати індивідуальні особливості користувача, використовуючи класифікаційні алгоритми чи нейронні мережі [4, 47]. Штучний інтелект, як надбудова над цими технологіями, забезпечує прийняття рішень, адаптацію системи до змінних умов, інтеграцію з іншими електронними модулями автомобіля та подальше вдосконалення поведінки програми на основі накопиченого досвіду. Такий підхід є особливо ефективним у контексті задачі моніторингу уваги водія, адже дозволяє створити гнучку, адаптивну та високоточну систему, що працює безконтактно, в реальному часі та може реагувати на критичні зміни у стані водія, своєчасно попереджаючи про потенційну небезпеку.

Типовими технічними характеристиками подібних рішень є:

– використання фронтальної камери, зафіксованої на панелі або торпедо автомобіля;

– обробка відеопотоку в реальному часі;

– виявлення та трекінг обличчя, очей, напрямку погляду, міміки [26, 30, 31, 34];

– генерація тривожного сигналу у випадку виявлення ознак втоми або неуважності;

– можливість інтеграції з іншими модулями транспортного засобу (гальма, рульове управління, CAN-шина) [15, 16].

Сучасні реалізації систем моніторингу можна умовно поділити на три категорії:

1. Комерційні промислові рішення, інтегровані у сучасні транспортні засоби (зазвичай - системи від Bosch, SmartEye, Mobileye тощо);

2. Програмні рішення для мобільних пристроїв, що використовують фронтальну камеру смартфона або планшета;

3. DIY/науково-дослідницькі системи на базі платформи Raspberry Pi, Arduino або інших мікроконтролерів у поєднанні з бібліотеками OpenCV, MediaPipe, Dlib, TensorFlow [20] [21].

Кожна з категорій має свої переваги й обмеження, зумовлені доступними ресурсами, точністю аналізу, швидкістю обробки та вартістю впровадження. Особливої уваги заслуговують системи, які можуть бути реалізовані з використанням недорогого та енергоефективного обладнання (як-от Raspberry Pi), адже вони дозволяють протестувати сучасні алгоритми без потреби у дорогих обчислювальних ресурсах. Має бути здійснено порівняння основних типів таких систем за критеріями точності, вартості, складності реалізації та адаптивності до умов експлуатації.

Існує кілька підходів до реалізації подібних систем, зокрема на основі спеціалізованого апаратного забезпечення, програмного забезпечення з

комп’ютерним зором, а також гібридних рішень. Проведемо аналіз найбільш популярних варіантів.

Комерційні системи (наприклад, SmartEye, Seeing Machines [6], Bosch Driver Monitoring System). Переваги:

– Висока точність і надійність розпізнавання втоми або відволікання.

– Використовують інфрачервоні камери, що працюють в умовах поганої видимості.

– Можливість роботи в різних погодних та освітлених умовах.

– Недоліки:

– Висока вартість обладнання і програмного забезпечення.

– Потребують інтеграції з CAN-шиною або іншим транспортним інтерфейсом.

– Закритий код, неможливість гнучкого налаштування або розширення.

Рисунок 1.1 – Інтегрована система моніторингу водія та ADAS від Seeing Machines та Ambarella [6]

Системи на базі мобільних пристроїв (наприклад, Android/iOS-додатки типу DriveSafe, Drowsy Driver Alert);

Переваги:

– Дешевизна, не потребують окремого обладнання (лише смартфон).

– Простота встановлення та користування.

– Використовують вбудовану фронтальну камеру.

– Недоліки:

– Залежність від положення та фіксації телефону, що впливає на точність.

– Високе навантаження на акумулятор телефону та обмеженість обчислювальних ресурсів.

– Часто не працюють у фоновому режимі або під час використання GPS.

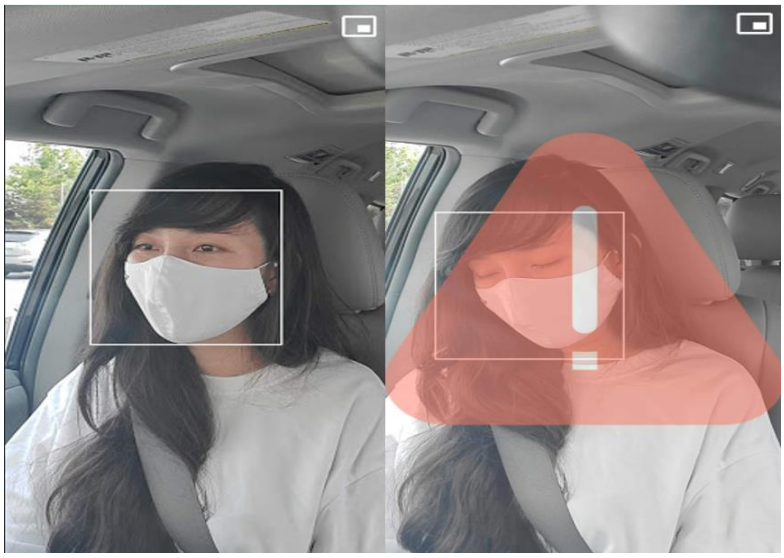


Рисунок 1.2 – Система оповіщення про водіння у стані сонливості від Drowsy Driver Alert

DIY-рішення на базі Raspberry Pi, OpenCV і камери (наприклад, Deep Learning системи, детекція очей/обличчя)

Переваги:

– Низька вартість компонентів, можливість гнучкої кастомізації.

– Відкритий код, можливість використання алгоритмів OpenCV, Dlib, MediaPipe, TensorFlow тощо.

– Можливість додавання звукового або візуального сигналу тривоги.

– Простота розгортання в автомобілі або іншому середовищі.

– Недоліки:

– Обмежені апаратні ресурси Raspberry Pi - складність обробки відео в реальному часі.

– Потреба в оптимізації моделей для ARM-архітектури.

– Потреба у базових знаннях схемотехніки та програмування.

– Обмежені можливості роботи в темряві без додаткового інфрачервоного підсвічування.

Зважаючи на співвідношення ціни, відкритості платформи, можливості налаштування та масштабування, найдоцільнішим варіантом для реалізації системи в умовах дипломної роботи є DIY-рішення на базі Raspberry Pi та OpenCV. Воно дозволяє створити функціональну систему моніторингу уваги водія із застосуванням алгоритмів комп'ютерного зору з можливістю подальшого вдосконалення.

Обчислювальний модуль побудовано на одноплатному комп'ютері Raspberry Pi [9, 10, 11], який використовує Broadcom SoC із вбудованим ARM-сумісним процесором та відеоядром GPU. Це дозволяє виконувати зважені обчислення з обробки зображень і відео у реальному часі. На ньому зазвичай працює операційна система Linux із підтримкою Python та бібліотеки OpenCV [8, 20]. Raspberry Pi має мережеві інтерфейси (наприклад, вбудований Wi-Fi та Bluetooth) для бездротового зв'язку та 40-контактний GPIO-роз'єм для підключення периферії. Така архітектура забезпечує достатню обчислювальну потужність і гнучкість для виконання алгоритмів трекінгу обличчя, очей і оцінки напрямку погляду.

Програмна складова базується на бібліотеці OpenCV, яка оптимізована для комп'ютерного зору в реальному часі. OpenCV надає готові алгоритми для виявлення облич, очей, оцінки пози обличчя та інших характеристик (наприклад,

трекінг жестів та взаємодію людина-комп'ютер). Це робить її ефективною для задач визначення уваги водія: алгоритм захоплює кадри з камери, детектує на них обличчя й очі (наприклад, за допомогою каскадних класифікаторів або методів глибокого навчання) й оцінює напрямок погляду та відхилення очей. OpenCV підтримує апаратне прискорення (GPU), що дозволяє обробляти відеопотік із мінімальною затримкою. У разі виявлення відволікання чи сонливості програма може активувати візуальну або звукову сигналізацію через виходи GPIO або мультимедійну систему.

Система реалізована в модульному вигляді: кожний компонент (камера, обчислювальний блок, джерело живлення, інтерфейс керування) є окремим модулем, який можна замінити або модернізувати без істотних змін архітектури. Наприклад, камеру можна легко замінити на іншу модель - навіть ІЧ-камеру (Pi NoIR) для роботи у темряві або високопродуктивну USB-камеру. Raspberry Pi надає 40-контактний GPIO-інтерфейс [9] та інші порти (I2C, SPI, UART, USB), що дозволяють підключати різні сенсори (температури, освітленості, тиску, акселерометри тощо). Підтримується стандарт HAT (Hardware Attached on Top) - плат розширення з EEPROM, які ОС Raspberry Pi автоматично розпізнає та налаштовує. Модульна архітектура також спрощує масштабування: можна під'єднати сторонні API через інтернет (через Wi-Fi/Bluetooth) або додати нові платформи обробки. Завдяки цьому система легко адаптується до нових завдань і може розвиватися без повної переробки системи. Це дозволяє з часом розширювати її функціональність - наприклад, додати підтримку машинного навчання для адаптивного виявлення індивідуальних особливостей водіїв, інтегрувати модулі зі збору статистики про поїздки, або підключити до хмарної платформи для централізованого моніторингу автопарку.

Також можлива реалізація додаткових сценаріїв: виявлення емоцій водія, розпізнавання втоми за мімікою, або навіть інтеграція з CAN-шиною автомобіля для адаптивної взаємодії з керуванням транспортним засобом (наприклад, зниження гучності музики чи обмеження швидкості при виявленні сонливості). За

допомогою Python та бібліотек MQTT або HTTP [41] можлива інтеграція з мобільним додатком або веб-інтерфейсом для сповіщень, перегляду логів та дистанційного налаштування [40].

У підсумку, DIY-рішення на базі Raspberry Pi не тільки відповідає вимогам робочого проєкту, але й створює реальну базу для подальших досліджень та розробки комерційного або відкритого продукту. Його відкритість, модульність і масштабованість дозволяють глибоко зануритись у питання комп'ютерного зору, вбудованих систем і безпеки транспорту, демонструючи практичне застосування сучасних технологій у сфері інтелектуального моніторингу.

1.3 Використання алгоритмів комп'ютерного зору для аналізу уваги людини на основі відеопотоку

Система моніторингу уваги, реалізована за допомогою комп'ютерного зору, передбачає автоматичне виявлення та аналіз низки фізіологічних маркерів - положення голови, моргання, зівання, напрямку погляду, які можуть сигналізувати про втому або сонливість. Основу подібної системи становить обробка відеопотоку в реальному часі за допомогою бібліотеки OpenCV та .

OpenCV - це відкрита бібліотека комп'ютерного зору, яка містить широкий спектр алгоритмів: класичні інструменти комп'ютерного зору, такі як детектори особливих точок (SIFT, SURF, ORB), алгоритми оптичного потоку (Lucas-Kanade, Farnebäck), різноманітні трансформації зображень (Фур'є, Хафа) та методи сегментації, потужні модулі машинного навчання з підтримкою традиційних алгоритмів, інтеграцією з глибинним навчанням через dnn модуль та каскадними класифікаторами Хаара для детекції контрольних точок обличчя чи очей [8, 20] [21]. Для специфічних завдань аналізу уваги OpenCV пропонує ефективні інструменти, включаючи каскадні класифікатори для швидкої детекції, різноманітні алгоритми відстеження об'єктів (KCF, MedianFlow) та функції обробки контурів та, що використовуються для відстеження положення зіниці в

задачах аналізу погляду. Натомість Dlib [25] пропонує спеціалізовані, більш точні методи виявлення обличчя та рис обличчя. Ці бібліотеки чудово доповнюють одна одну, тому розробники часто використовують OpenCV для базової обробки відео та зображень, а потім вже залучають dlib для точної роботи з обличчями - аналіз погляду та виразів.

Головний аспект контролю уваги з відеопотоку включає локалізацію обличчя в кадрі, після чого виконується побудова сітки з 68 контрольних точок (face landmarks) [5], яка охоплює ключові зони: очі, брови, ніс, рот, контури обличчя, які можна побачити на рисунку 1.3. Для точного визначення цих контрольних точок широко використовується бібліотека Dlib, яка пропонує предиктор орієнтирів обличчя, що забезпечує високу точність розміщення 68 точок навіть при зміні положення голови та виразу обличчя. Розпізнавання обличчя в OpenCV реалізується за допомогою різних алгоритмів, але одними із найпопулярніших є каскадні класифікатори Хаара, що є методом машинного навчання. Також підтримує методи глибокого навчання для розпізнавання облич, на основі нейронних мереж, які можуть працювати в тандемі з Dlib для створення високоефективних систем моніторингу уваги.

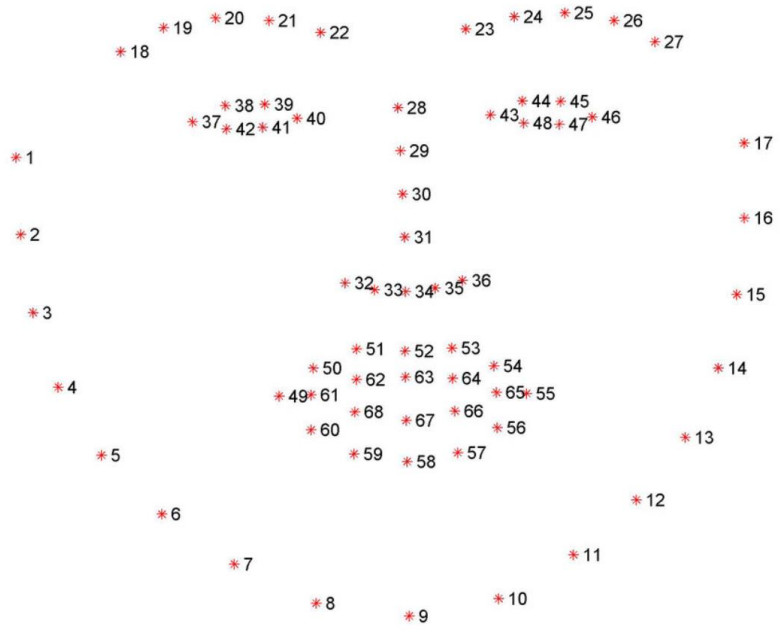
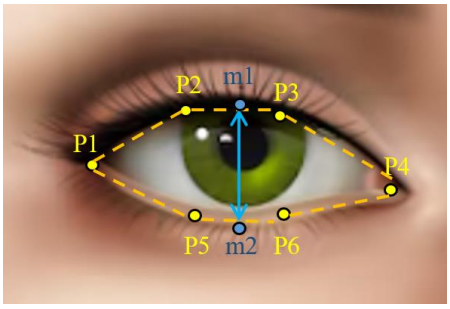
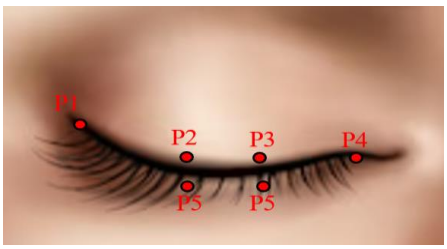


Рисунок 1.3 – Сітка контрольних точок на обличчі людини

Одним із ключових фізіологічних індикаторів втрати уваги або настання стану сонливості є моргання та тривале закриття очей [5, 33]. У стані втоми частота моргання зазвичай стає повільнішим і менш частим, а його тривалість подовжується. Автоматизований аналіз моргань у системах моніторингу уваги реалізується шляхом обробки відеозображення обличчя в реальному часі, зокрема за допомогою алгоритму обчислення коефіцієнта аспекту ока - EAR (Eye Aspect Ratio) [30]. Щоб виявити моргання, необхідно визначити співвідношення горизонтальної та вертикальної протяжності очей. Для лівого ока горизонтальна лінія буде визначатися відстанню між точками P1 і P4, а вертикальна - відстанню між середньою точкою відрізка P2-P3 та середньою точкою відрізка P5-P6, що зображені на рисунку 1.4. Аналогічні розрахунки будуть проведені для правого ока, використовуючи відповідні координати. Для цього ми розробимо функції для обчислення середньої точки та евклідової відстані між заданими координатами.



а)



б)

а) – у відкритому положенні, б) – у закритому положенні

Рисунок 1.4 – Контрольні точки очей

Окрім визначення стану очей, контрольні точки обличчя дозволяють ідентифікувати інші поведінкові ознаки втрати уваги, зокрема позіхання, міміку або рухи рота [4]. Наприклад, аналізуючи відстані між точками, що описують контури ротової порожнини, можна обчислити коефіцієнт відкриття рота, який зростає при позіханні. Таким чином, система на основі контрольних точок здатна відстежувати не лише моргання, а й низку інших проявів втоми чи неуважності, що суттєво підвищує точність і надійність моніторингу.

Не менш важливим є напрямок погляду зіниць, адже в більшості застосувань існують допустимі сектори перед обличчям користувача, в межах яких увага вважається зосередженою. Відхилення погляду за межі цих секторів - наприклад, вліво чи вправо - не завжди є безпечним або допустимим і залежить від конкретної сфери застосування, зокрема в автомобільних, виробничих чи операторських системах. На рисунку 1.5 представлено приклад роботи програми, реалізованої за допомогою бібліотек OpenCV та Dlib, яка виконує аналіз положення зіниць у реальному часі. Знаючи координати зіниць, можна визначити, у який сектор спрямований погляд користувача. Це дозволяє судити про рівень його концентрації та своєчасно реагувати на ознаки відволікання.

Рисунок 1.5 – Демонтсрація знаходження координат зіниці, використовуючи аналіз відеопотоку OpenCV [8]

1.4 Постановка задачі

Забезпечення безпеки дорожнього руху є актуальною проблемою, зокрема в контексті зниження кількості аварій, спричинених людським фактором. Однією з поширених причин ДТП є втрата концентрації або відволікання водія під час керування транспортним засобом. У зв'язку з цим виникає потреба у створенні інтелектуальної системи, здатної в реальному часі контролювати рівень уваги водія та оперативно реагувати у разі його неуважності.

Проєкт спрямований на розробку програмно-апаратного комплексу, що використовує Raspberry Pi, камеру, GPS-модуль, LCD-дисплей, звуковий сигналізатор (buzzer) та світлодіод (LED) для моніторингу стану уваги користувача. Система використовує бібліотеки OpenCV і Dlib для виявлення обличчя, його орієнтації та координат зіниць. На основі цієї інформації здійснюється оцінка напрямку погляду з порівнянням із допустимими зонами спостереження. У разі фіксації відволікання, користувач отримує звукове та візуальне повідомлення.

Система повинна реалізовувати такі основні функції:

1) Інтегрувати апаратні компоненти: камера, GPS-модуль, LCD-дисплей, buzzer та LED на платформі Raspberry Pi для створення єдиного програмно-апаратного комплексу.

2) Реалізувати обробку відео з використанням OpenCV та Dlib для детекції обличчя, положення голови та координат зіниць у реальному часі.

3) Розробити алгоритм аналізу уваги на основі порівняння напрямку погляду з допустимими зонами, із запуском сповіщень при виявленні відволікання.

4) Реалізувати керування індикацією: виведення повідомлень на LCD, генерація звукових сигналів з варіативною гучністю, та візуальна індикація через LED.

5) Розробити механізм збору та передачі статистики, включаючи координати GPS і часові мітки, на віддалений сервер через визначені інтервали.

1.5 Висновки до першого розділу

У межах розділу 1 проведено аналіз структурної та функціональної організації кіберфізичної системи моніторингу уваги людини на основі аналізу відеопотоку в реальному часі. Розглянуто особливості побудови систем контролю стану оператора, наведено класифікацію існуючих підходів до виявлення ознак втоми на основі фізіологічних і поведінкових маркерів. Здійснено порівняльний аналіз переваг і недоліків сучасних аналогів систем моніторингу уваги, зокрема щодо точності, складності реалізації, вимог до обчислювальних ресурсів та здатності до роботи в реальному часі. Проведено огляд бібліотек комп'ютерного зору, таких як OpenCV та Dlib, і алгоритмів, які застосовуються для детекції обличчя, очей, рота, положення голови та інших ознак, що характеризують рівень уваги. На основі проведеного аналізу сформульовано проблеми і рішення розроблення власної кіберфізичної системи для контролю уваги водія із використанням засобів аналізу відеопотоку та попередження про ознаки сонливості або втрати концентрації.

Проведене дослідження показало високу перспективність реалізації систем моніторингу уваги людини у форматі DIY-рішень на базі одноплатного комп'ютера Raspberry Pi. Такі системи вирізняються низькою вартістю, компактністю та достатньою обчислювальною потужністю для виконання базових алгоритмів комп'ютерного зору в реальному часі. Використання відкритих бібліотек, зокрема OpenCV та Dlib, дозволяє швидко розгорнути функціональний прототип без потреби у складному програмно-апаратному середовищі. Важливою перевагою є енергоефективність і можливість автономної роботи в обмежених умовах, зокрема в транспортних засобах. Це робить Raspberry Pi оптимальним вибором для побудови недорогих, адаптивних та мобільних кіберфізичних систем.

2 ПРОЄКТУВАННЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ ДЛЯ МОНІТОРИНГУ УВАГИ ЛЮДИНИ З ВИКОРИСТАННЯМ АНАЛІЗУ ВІДЕОПОТОКУ В РЕАЛЬНОМУ ЧАСІ

2.1 Визначення апаратних і програмних підсистем програмно-технічного засобу

Для реалізації програмно-технічного засобу моніторингу відволікання водія, який поєднує в собі обробку відео та передачу даних, були визначені ключові апаратні та програмні підсистеми, що найкраще підходять для реалізації поставленої задачі, проведено їхній аналіз та знайдено оптимальні конфігурації та компоненти.

Апаратні підсистеми:

1) Одноплатний комп'ютер Raspberry Pi 5, що виступає центральним обчислювальним елементом системи. Його продуктивність дозволяє обробляти відеопотік в реальному часі для детекції відволікання та забезпечує взаємодію з іншими периферійними пристроями.

2) Камера Raspberry Pi Camera Module 3, що використовується для захоплення відеопотоку з високою роздільною здатністю, що є критично важливим для точного аналізу погляду та міміки обличчя. Камера підключається до спеціального порту CSI на Raspberry Pi.

3) LCD-екран 16x2, який забезпечує виведення текстової інформації для водія або оператора, такої як поточний стан системи, попередження про відволікання, або інші службові повідомлення. Підключається до Raspberry Pi через GPIO піни, зазвичай по інтерфейсу I2C або паралельно.

4) Зуммер (Buzzer), який використовується для подачі звукових сигналів попередження у випадку виявлення відволікання водія. Підключається до одного з GPIO пінів Raspberry Pi.

5) Червоний LED-індикатор, як додатковий візуальний індикатор для сигналізації про стан системи або попередження. Також підключається до GPIO пінів Raspberry Pi.

6) GPS-модуль (U-blox NEO-6M [36] або аналогічний) з антеною, що надає дані про поточне місцезнаходження та швидкість транспортного засобу [35, 37]. Ця інформація може бути використана для запису подій відволікання (наприклад, реєстрація відволікання на певних ділянках дороги) та для додавання до статистики, що передається на сервер. Підключається до Raspberry Pi через послідовний порт (UART).

7) USB 4G/LTE модем (з підтримкою SIM-карти), який використовується для забезпечення мобільного доступу до інтернету. Це дозволяє системі передавати зібрану статистику та дані про інциденти відволікання на віддалений сервер, коли є доступ до мережі. Підключається до одного з USB-портів Raspberry Pi.

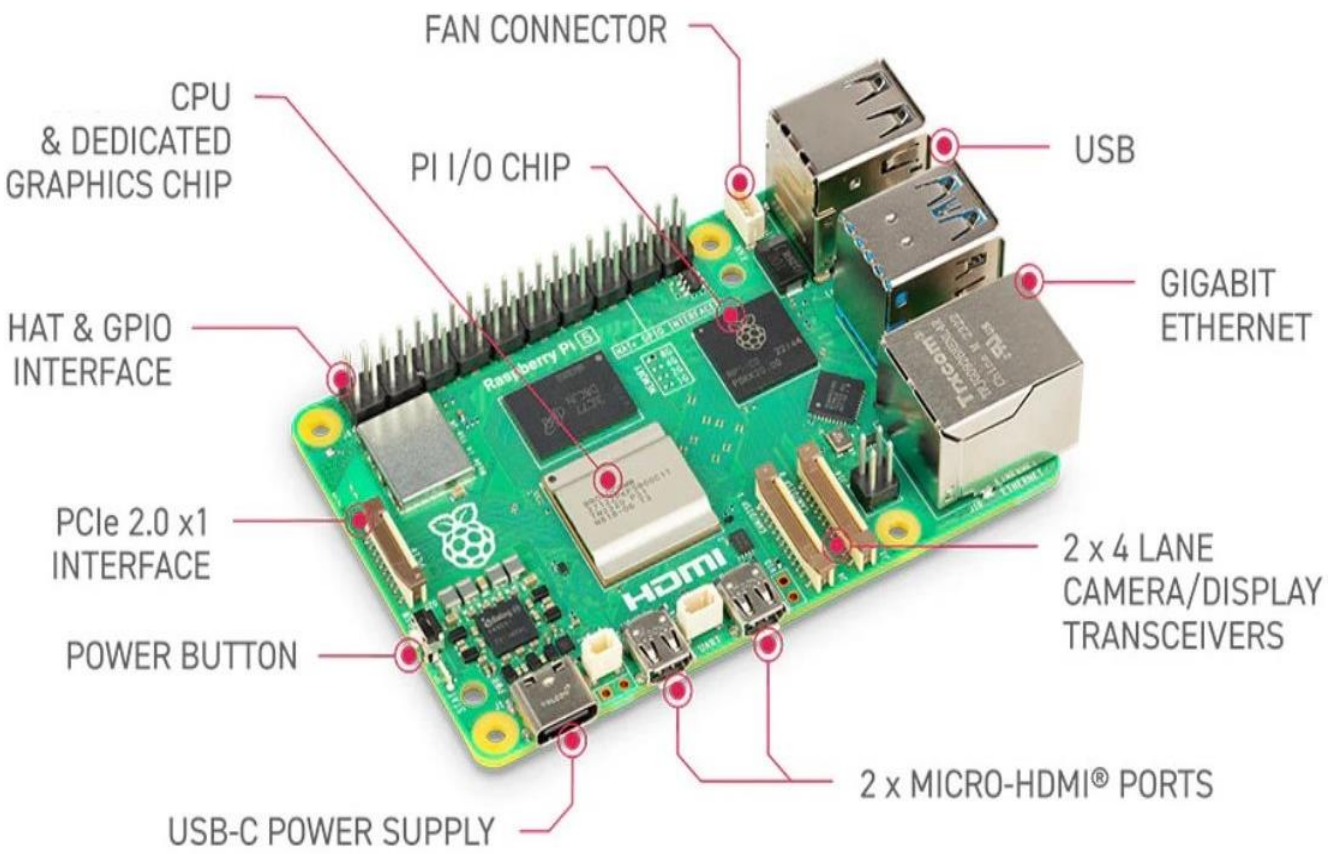


Рисунок 2.1 – Схема Raspberry Pi 5 [9]

Було обрано модель Raspberry Pi 5 через низку ключових переваг, що забезпечують її оптимальність для даного проекту та майбутнього розвитку. По-перше, значно потужніший процесор BCM2712 гарантує високу обчислювальну продуктивність, необхідну для ефективної обробки відеопотоку в реальному часі за допомогою бібліотек OpenCV та Dlib, що є критично важливим для точної детекції відволікання за поглядом і мімікою обличчя. По-друге, доступні конфігурації оперативної пам'яті (2ГБ, 4ГБ, 8ГБ, 16ГБ) надають достатній ресурс для завантаження та роботи складних моделей машинного навчання, а також для одночасної обробки значних обсягів візуальних даних. По-третє, обрання Raspberry Pi 5 створює значний "запас міцності" для оновлення та розширення проекту під нові задачі: майбутні вимоги до більш складних алгоритмів аналізу, інтеграції додаткових сенсорів або розширення функціоналу системи будуть легко реалізовані завдяки надлишковій продуктивності та гнучкості платформи. Нарешті, велика кількість та різноманітність доступних портів, включаючи кілька USB-портів (зокрема USB 3.0 для швидкої передачі даних), два CSI/DSI порти для камери та дисплея, а також розширений GPIO, PCIe FFC та вбудовані модулі Wi-Fi 5 та Bluetooth 5, забезпечує безперешкодне підключення всіх необхідних апаратних підсистем, включаючи USB 4G/LTE модем, GPS-модуль, LCD-екран та інші периферійні пристрої, що робить Raspberry Pi 5 ідеальним вибором для комплексного програмно-технічного засобу.

Для захоплення відеопотоку була обрана саме Pi Camera v3 завдяки її високій роздільній здатності (12 Мп), чудовій якості зображення та підтримці HDR, що забезпечує чітке відстеження погляду та міміки обличчя навіть в умовах змінної освітленості. Ключовими факторами вибору також стали вбудований автофокус, який спрощує налаштування, та легка інтеграція з Raspberry Pi через спеціальний CSI-порт.

Рисунок 2.2 – Raspberry Pi Camera Module 3

Для інтеграції функціоналу визначення місцезнаходження та швидкості був обраний GPS-модуль (наприклад, U-blox NEO-6M). Його вибір зумовлений високою точністю позиціонування, що дозволяє прив'язувати дані про відволікання до конкретних географічних координат та маршруту руху. Наявність вбудованої антени забезпечує стабільний прийом сигналу, а легкість підключення до Raspberry Pi через UART-інтерфейс робить його зручним компонентом для збору контекстуальних даних, необхідних для аналізу та передачі на сервер.

Рисунок 2.3 – GPS модуль NEO-6M v2 [36]

Для забезпечення стабільного мобільного доступу до Інтернету в складі системи використовується USB 4G/LTE модем з підтримкою SIM-карти, який підключається до одного з USB-портів Raspberry Pi. Це дозволяє пристрою отримувати інтернет-з’єднання незалежно від наявності Wi-Fi мережі, що критично важливо для передачі даних про інциденти у реальному часі під час руху

транспортного засобу. Мобільне з’єднання використовується для надсилання інформації на віддалений сервер [42, 43], включаючи координати, швидкість, час, а також зображення водія у момент виявлення неуважності.

Однією з практичних моделей для реалізації цього компонента є Huawei E3372h-320 - USB LTE-модем, який підтримує мережі 4G LTE Cat.4 з максимальною швидкістю завантаження до 150 Мбіт/с і відвантаження до 50 Мбіт/с. Пристрій має компактний формфактор, сумісний із Raspberry Pi, працює з будь-якими SIM-картами українських мобільних операторів і не потребує складної конфігурації. Його основна перевага полягає у стабільній роботі без необхідності встановлення драйверів - модем розпізнається системою як мережевий інтерфейс (через модемний режим або режим Ethernet over USB).

Рисунок 2.4 – USB 4G/LTE модем Huawei E3372h-320

Оскільки USB-модем також є споживачем електроенергії, важливо забезпечити належне живлення для коректної та безперебійної роботи всієї системи. Для цього рекомендується використовувати якісний блок живлення на 5 В з вихідним струмом не менше 3 А, що дозволить одночасно підтримувати роботу Raspberry Pi, USB-модему, камери та інших периферійних пристроїв. Уникнення просідання напруги критичне, оскільки в протилежному випадку модем може перезавантажуватися або втрачати з’єднання під час активної передачі даних.

У проєкті також використовуються звуковий сигналізатор (buzzer) та світлодіод (LED) для оповіщення користувача про стан його уваги. Якщо система виявляє, що користувач відволікся або закрив очі на тривалий час, активується звуковий сигнал з різною інтенсивністю залежно від рівня небезпеки або

тривалості неуважності. Світлодіод служить як візуальний індикатор стану, наприклад, може мигати або змінювати колір. Крім того, використовується екран LCD, який відображає системну інформацію - зокрема, повідомлення про те, що камера спрямована на обличчя або що система перебуває в режимі пошуку. Під час пошуку обличчя подається тихий звуковий сигнал, який інформує користувача про активний процес ініціалізації без надмірного відволікання.

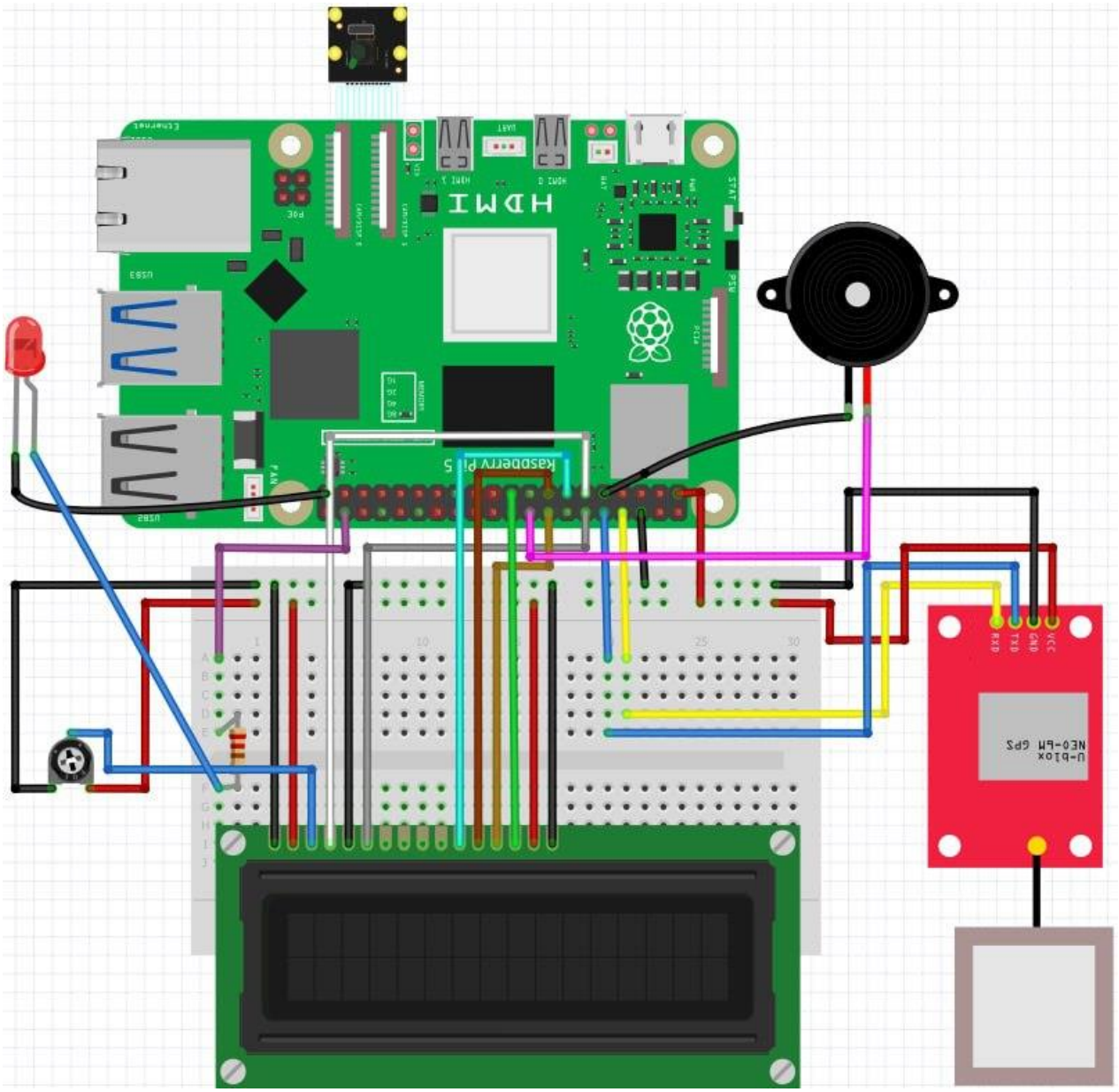


Рисунок 2.5 – Креслення апаратних з’єднаннь усіх компонентів

Програмні підсистеми:

1) Операційна система Raspberry Pi OS (Debian-based) [10], яка забезпечує базове середовище для роботи апаратних компонентів та виконання програмного забезпечення.

2) OpenCV (Open Source Computer Vision Library), як основна бібліотека для роботи з відеопотоком, обробки зображень та виконання комп'ютерного зору. OpenCV буде використовуватися для захоплення відео з камери, попередньої обробки кадрів та взаємодії з dlib.

3) Dlib Library [25], як бібліотека машинного навчання, що буде застосовуватися для детекції та відстеження обличчя, розпізнавання ключових точок обличчя (landmarks) та аналізу міміки та напрямку погляду. Dlib надає потужні інструменти для реалізації алгоритмів виявлення відволікання на основі візуальних даних.

4) Розробити алгоритми для аналізу погляду та міміки, які розроблені з використанням OpenCV та Dlib, це програмне забезпечення буде аналізувати відеопотік з камери, визначати положення голови, напрямок погляду та виявляти ознаки відволікання (наприклад, тривала відсутність погляду на дорогу, закриті очі), підрахунок EAR значення.

5) Програмне забезпечення для керування периферійними пристроями, реалізує логіку взаємодії з LCD-екраном (виведення інформації), зуммером та LED-індикатором (видача попереджень).

6) Модуль для роботи з GPS-даними, як програмний компонент для зчитування та парсингу даних з GPS-модуля.

7) Модуль для управління інтернет-з'єднанням та передачею даних відповідає за встановлення та підтримку інтернет-з'єднання через USB-модем, а також за форматування та надсилання статистичних даних про відволікання (з прив'язкою до часу та GPS-координат) на віддалений сервер. Для передачі даних можуть використовуватися протоколи HTTP/HTTPS або MQTT [41].

У системі моніторингу уваги водія критично важливим є ефективне виявлення станів неуважності в реальному часі, з метою попередження потенційно небезпечних ситуацій на дорозі. Для цього необхідно розробити чіткий програмний алгоритм, який забезпечить своєчасну обробку вхідних візуальних даних, прийняття рішень на основі аналізу фізіологічних та поведінкових ознак, а також фіксацію відповідних подій для подальшого зберігання та передачі на сервер. Такий алгоритм має враховувати як технічні особливості платформи, так і специфіку поведінки користувача.

Робота системи починається з отримання зображення з відеопотоку, що надходить від фронтальної камери, спрямованої на обличчя водія. Далі виконується обробка кадру, зокрема детекція обличчя та визначення ключових точок (landmarks), що відображають положення очей, брів, носа, губ тощо. Ці дані передаються на два незалежних програмних модулі, які працюють паралельно. Перший модуль здійснює аналіз геометрії очей шляхом розрахунку коефіцієнта EAR який дозволяє визначити, чи були очі закритими тривалий час - показник сонливості або зниження концентрації. Другий модуль аналізує напрямок погляду, визначаючи, чи сфокусований він на дорозі або ж відвернутий вбік. Це дозволяє виявити відволікання, не пов'язане зі сном, наприклад, при використанні мобільного телефону. У разі, якщо хоча б один з модулів виявляє неуважність, система негайно подає сигнал тривоги водієві за допомогою зумера, світлодіодного індикатора та текстових повідомлень на дисплеї. Одночасно з цим фіксуються й зберігаються технічні дані інциденту - зображення обличчя, дата, час, координати GPS та швидкість автомобіля. У випадку, якщо увага водія підтверджена обома модулями, система переходить до стандартного режиму моніторингу та виводить службову інформацію. Деталізовану реалізацію цієї логіки представлено на рисунку 2.5 у вигляді блок-схеми алгоритму обробки вхідного відеопотоку.

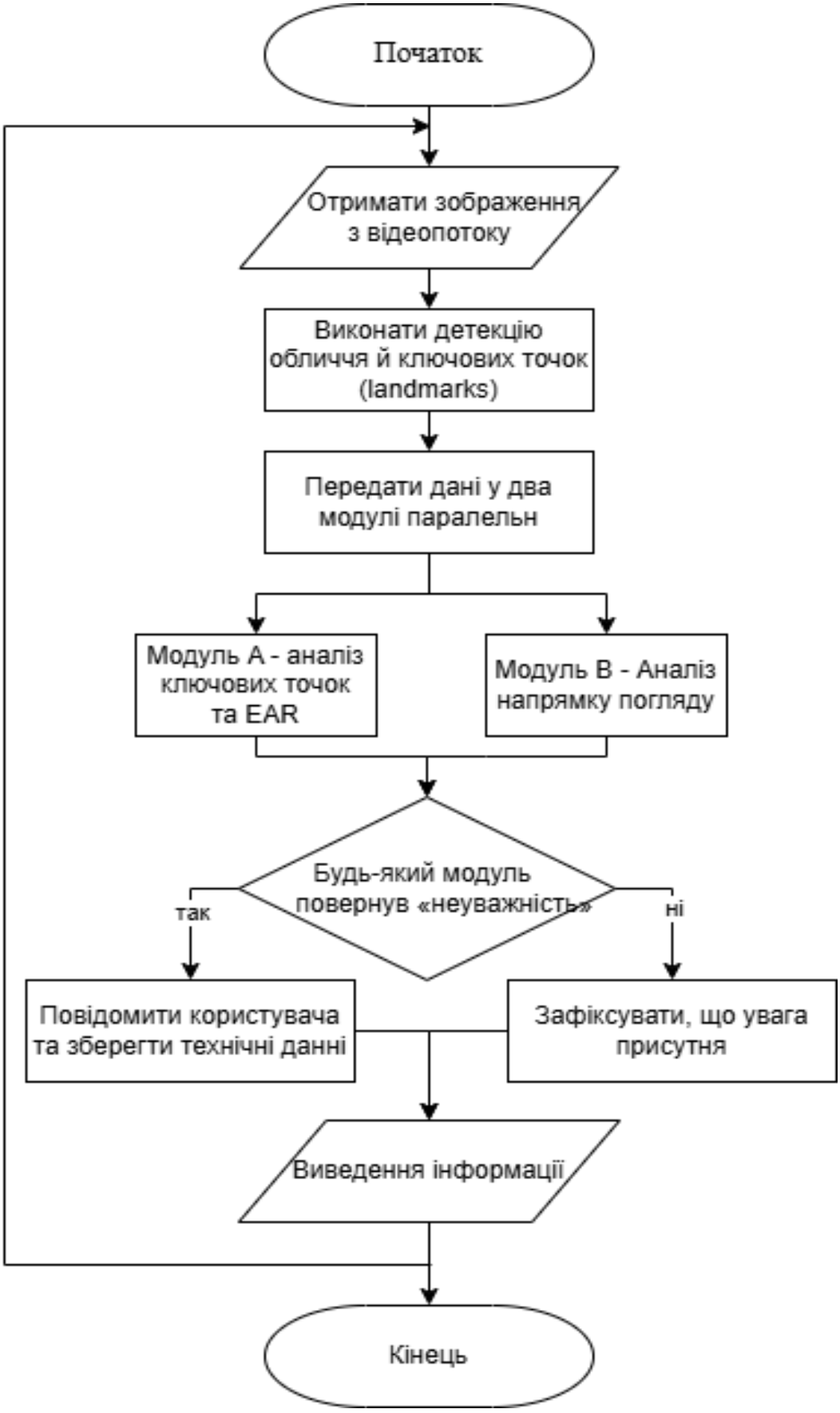


Рисунок 2.6 – Блок-схема алгоритму перевірки погляду на уважність

Виявлення коефіцієнта закритих очей, відомого як Eye Aspect Ratio (EAR), здійснюється на основі математичної формули, яка враховує просторові відстані між ключовими контрольними точками на повіках, що дозволяє точно оцінити ступінь відкритості або закритості очей на зображенні чи відео і визначається за формулою:

$$EAR = \frac{\|P_2 - P_6\| + \|P_3 - P_5\|}{2 \cdot \|P_1 - P_4\|}, \quad (2.1)$$

де *EAR* - коефіцієнт пропорції відкритості ока.

P_1, …, P_6 - точки, що представляють верхню і нижню частини одного з вертикальних сегментів ока, див. рис. 1.4.

EAR розраховується окремо для кожного ока на кожному кадрі відеопотоку. При відкритих очах значення EAR є відносно високим і стабільним. У момент моргання або закриття очей вертикальні відстані між повіками зменшуються, у результаті чого EAR різко падає. Короткочасне зниження (на 1-2 кадри) інтерпретується як нормальне моргання, а тривале зменшення EAR протягом певної кількості кадрів (наприклад, 15-20) є ознакою дрімоти.

На основі емпіричних досліджень було встановлено, що для більшості користувачів:

– EAR > 0.25 - очі відкриті;

– EAR > 0.20 - очі закриті або у стані моргання;

– якщо EAR < 0.20 зберігається протягом ≥ 20 кадрів - виявлено сонливість або втрату концентрації.

Ці порогові значення можуть бути адаптовані під конкретного користувача або контекст шляхом індивідуального калібрування. Алгоритм обробляє кожен кадр у режимі реального часу, тому ефективність обчислень є критичною. Для уникнення хибних спрацьовувань через випадкові моргання застосовується ковзне

вікно з підрахунком тривалості низького EAR. Використання Dlib facial landmark detector забезпечує точну локалізацію контрольних точок, навіть за змінних умов освітлення або часткової зміни положення голови.

Системи моніторингу стану водія, зокрема ті, що відслідковують увагу або втому, мають працювати лише у відповідні моменти - коли транспортний засіб рухається. Це дозволяє уникнути зайвих обчислень та фальшивих тривог у разі, коли автомобіль стоїть або вимкнений. Одним із найбільш надійних способів виявлення руху є використання глобальної навігаційної супутникової системи (GNSS), зокрема модулів на базі GPS. У цьому проєкті застосовується модуль NEO-6M v2, який базується на чипі U-blox 6.

GPS-модуль обробляє сигнали від кількох супутників (мінімум чотирьох), щоб визначити:

1) географічні координати (широта, довгота, висота);
2) час (UTC);
3) швидкість руху об'єкта по земній поверхні;
4) курс руху (напрям);

GPS формує ці дані у вигляді NMEA-повідомлень, де одне з ключових - це $GPRMC (Recommended Minimum Specific GPS/Transit Data). Це повідомлення містить значення швидкості по поверхні землі (Speed Over Ground), яке ми використовуємо для визначення руху автомобіля. Швидкість, яку передає модуль, вже обчислена на основі зміни координат у часі, з урахуванням фільтрації супутникових сигналів. Важливо що, модуль NEO-6M самостійно обчислює швидкість. Обґрунтування використання GPS, а не акселерометра юудується на тому, що хоч деякі мікроконтролери або смартфони мають акселерометри, GPS надає більш стабільний та точний показник швидкості при довготривалому спостереженні. На відміну від акселерометра, який фіксує навіть мікроскопічні рухи або вібрації, GPS фільтрує подібні коливання, оскільки працює з координатами у глобальній системі.

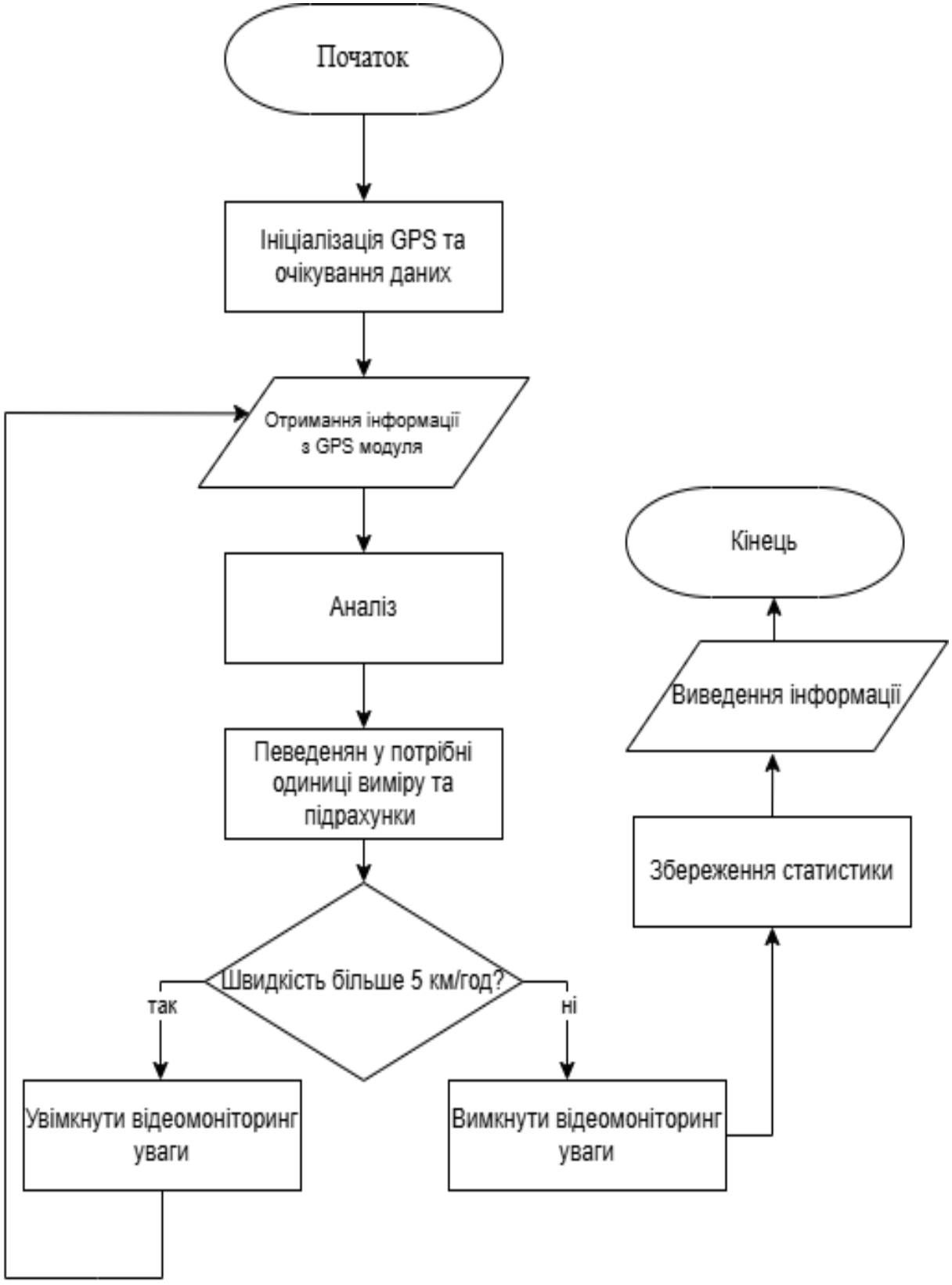


Рисунок 2.7 – Блок-схема роботи програми в залежності від швидкості

2.2 Формування та збереження даних локально

Система моніторингу уваги водія, розроблена на основі одноплатного комп'ютера Raspberry Pi 5, повинна ефективно опрацьовувати та зберігати велику кількість даних, що генеруються в реальному часі. До таких даних належать: зображення з камери, часові мітки інцидентів, геолокаційна інформація з GPS-модуля, швидкість транспортного засобу та службові повідомлення. Через обмеженість внутрішньої пам’яті пристрою важливо правильно організувати процес збереження даних та оптимізувати їх об’єм.

З метою впорядкування та ефективного доступу до інформації всі локальні дані формуються у структурованому вигляді. Для цього використовується файлово-каталогова система з чіткою ієрархією:

1) Кожен день створюється окрема папка з датою у форматі YYYY-MM-DD.
2) В межах папки зберігаються окремі підпапки для інцидентів (incidents), логів (logs), зображень (images) та службових звітів (reports).
3) Формат запису інциденту базується на JSON-структурі з ключами timestamp, latitude, longitude, speed, image_path та duration, що забезпечує гнучкість та легкість інтеграції з будь-якою системою обробки даних у майбутньому.

Зважаючи на великий об’єм зображень, які зберігаються у випадку виявлення неуважності, критично важливо реалізувати ефективні алгоритми стиснення. Для цього використовується формат JPEG із адаптивною компресією. Основна перевага JPEG - збереження задовільної візуальної якості при значному зменшенні розміру файлу. Залежно від освітленості, кількості деталей на зображенні та фіксованої частоти виявлення інцидентів, рівень якості компресії встановлюється на рівні 60-75%, що дозволяє досягти середнього розміру зображення в межах 150-250 КБ.

Крім JPEG, для подальшого зменшення розміру файлів та збереження супутньої інформації (наприклад, EXIF-метаданих) може застосовуватись формат WebP, який підтримує як втратне, так і безвтратне стиснення. WebP показує кращу ефективність у порівнянні з JPEG для зображень з однаковою візуальною якістю.

Текстові логи, які зберігають інформацію про роботу системи, внутрішні події та службові повідомлення, зберігаються у форматі .log або .txt з можливістю щоденної ротації. Для уникнення накопичення надлишкових даних використовується алгоритм циклічного логування: по досягненню граничного об’єму (наприклад, 1 МБ), старіші записи перезаписуються або архівуються у форматі .gz. Інструменти, як-от logrotate, можна налаштувати безпосередньо в системі Raspberry Pi для автоматичного виконання ротації та стиснення. Такий підхід дозволяє утримувати актуальну інформацію в системі та не перевищувати обмеження по пам’яті.

Зображення, які не мають критичного значення (наприклад, ті, що не містять підтвердженого інциденту), можуть зберігатися в окремій директорії кешу з обмеженням часу життя (TTL -Time to Live). Наприклад, такі файли автоматично видаляються через 24 години. Для цього використовуються скрипти на Bash або Python, які щогодини перевіряють вік файлів у кеш-директорії.

Для зберігання метаданих (наприклад, координат, часу, ID користувача, посилання на зображення) локально застосовується вбудована СУБД SQLite. Цей механізм дозволяє зберігати структуровану інформацію у вигляді таблиць, виконувати SQL-запити, сортування та фільтрацію без потреби в окремому серверному середовищі. SQLite є легковаговою, не вимагає інсталяції та ідеально підходить для вбудованих рішень на Raspberry Pi.

Незважаючи на вжиті заходи з оптимізації, кількість даних, які зберігаються в системі, поступово зростає. Якщо система працює впродовж тривалого часу без передачі інформації на сервер, з’являється ризик заповнення вільної пам’яті пристрою, що може призвести до збоїв у роботі або втрати важливої інформації. Саме тому ключовим етапом архітектури є інтеграція з віддаленим сервером, на який система передає заздалегідь відібрані, стиснені та структуровані дані.

Цей підхід забезпечує надійність, масштабованість, можливість централізованого моніторингу та аналізу історичних даних. У наступному

підрозділі буде розглянуто детальніше логіку взаємодії з сервером, принципи передачі інформації, а також інструменти реалізації бекенд-частини системи.

2.3 Моделювання віддаленого серверу для зберження даних

Оскільки Raspberry Pi має обмежені ресурси та обсяг пам’яті, частину функцій обробки й зберігання даних доцільно передати на простий віддалений сервер, тому що це:

1) розвантажує пристрій;
2) забезпечує масштабованість;
3) дозволяє централізовано зберігати, обробляти та візуалізувати дані;
4) відкриває можливість для історичної аналітики;
5) сервер працює як центральна база даних та аналітична платформа для всіх підключених пристроїв;
6) додаткова обчислювальна потужність;
7) централізоване зберігання: події, GPS-координати, системні логи - усе записується в базу даних;
8) масштабованість: легко додати нові пристрої без значних змін;
9) аналітика та історія: можна аналізувати поведінку користувача у довгостроковій перспективі;
10) гнучкість: налаштування частоти та типів переданих даних (наприклад, раз на 10 сек / 1 хв / 5 хв) ;
11) веб-інтерфейс: зручна панель адміністратора, що показує аналітику в реальному часі.

Після того як пристрій на базі Raspberry Pi виявляє факт неуважності водія - наприклад, зафіксовано, що водій відвів погляд від дороги на тривалий час, закрив очі або відвернув голову - система переходить до етапу збору та структурування даних для передачі на сервер. До таких даних належить точна дата та час події, координати з GPS-модуля (широта, довгота), швидкість руху автомобіля,

тривалість відволікання та зображення водія, зроблене камерою в момент порушення. Ці дані зберігаються локально у тимчасовому буфері на пристрої та упаковуються у формат, придатний для мережевої передачі.

Далі пристрій через інтернет-з'єднання, яке забезпечується за допомогою USB-адаптера з SIM-картою (мобільна мережа), ініціює з'єднання з віддаленим сервером. Передача даних відбувається через HTTPS-запит до веб-сервера, який працює під керуванням Nginx [38]. У цьому запиті можуть використовуватись формати JSON (для текстових і числових даних) та multipart/form-data (для зображення). Запит містить всю інформацію про подію, включно з фотографією, закодованою або у base64, або прикріпленою як файл.

Серверна частина розробленої системи реалізована на мові програмування JavaScript з використанням платформи Node.js [52] та сучасного фреймворку NestJS. Обрання NestJS було зумовлене його архітектурною структурованістю, підтримкою TypeScript, модульністю та широкими можливостями для побудови масштабованих REST API, що особливо важливо для прийому, обробки та збереження великої кількості даних, які надходять із вбудованих пристроїв. Завдяки NestJS сервер надійно обробляє запити, перевіряє авторизацію пристроїв на основі попередньо зареєстрованих користувачів та записує всю інформацію до реляційної бази даних MySQL [54].

Для адміністрування та візуального представлення даних, отриманих від пристроїв, було реалізовано вебінтерфейс на базі фронтенд-фреймворку Vue.js [51]. Панель адміністратора дозволяє переглядати список зареєстрованих пристроїв, статистику інцидентів, а також деталі кожного випадку відволікання водія, включно з геолокацією, зображеннями, швидкістю руху та часовими мітками. Такий підхід дає змогу оператору системи або відповідальній особі в реальному часі аналізувати поведінку водіїв, приймати управлінські рішення або проводити ретроспективний аналіз небезпечних ситуацій. Інтеграція NestJS та Vue.js забезпечує надійний розподіл обов'язків між серверною логікою та візуалізацією,

сприяючи зручності обслуговування системи, легкості масштабування, а також забезпечує високу продуктивність та сучасний користувацький досвід.

Nginx виступає в ролі зворотного проксі та першої точки захисту. Він фільтрує всі вхідні запити та перенаправляє лише дозволені з них на внутрішній сервер, що займається обробкою логіки запиту. Доступ до API обмежений лише для реальних, авторизованих пристроїв, які були попередньо зареєстровані адміністратором системи. Ідентифікація може відбуватися через IP-фільтрацію, API-токени або інші методи аутентифікації, які підтверджують, що пристрій справді належить до списку довірених. Якщо перевірка проходить успішно, бекенд-сервер приймає запит, розпаковує отриману інформацію та виконує перевірку її валідності. Далі ці дані передаються у систему збереження - базу даних MySQL. Тут створюється новий запис, який містить усі параметри події, включно з геолокацією, часом, швидкістю, тривалістю та зображенням водія. Таким чином формується історія інцидентів, яка може бути використана для подальшого аналізу або сповіщення відповідальних осіб.

Після успішного збереження запису сервер відправляє відповідь Raspberry Pi - наприклад, статус 200 OK або інший код, що підтверджує прийняття та збереження даних. Якщо виникла помилка (наприклад, проблема з аутентифікацією або внутрішня помилка бази даних), пристрій отримує відповідне повідомлення і, за потреби, повторює запит через деякий час.

Вся система спроєктована таким чином, щоб працювати автономно й безперервно, забезпечуючи надійний контроль за станом уваги водія. Завдяки використанню мобільного інтернету, передача даних можлива в реальному часі, незалежно від місця перебування автомобіля, а централізоване збереження дає змогу легко здійснювати моніторинг, звітність або реагування на критичні ситуації.

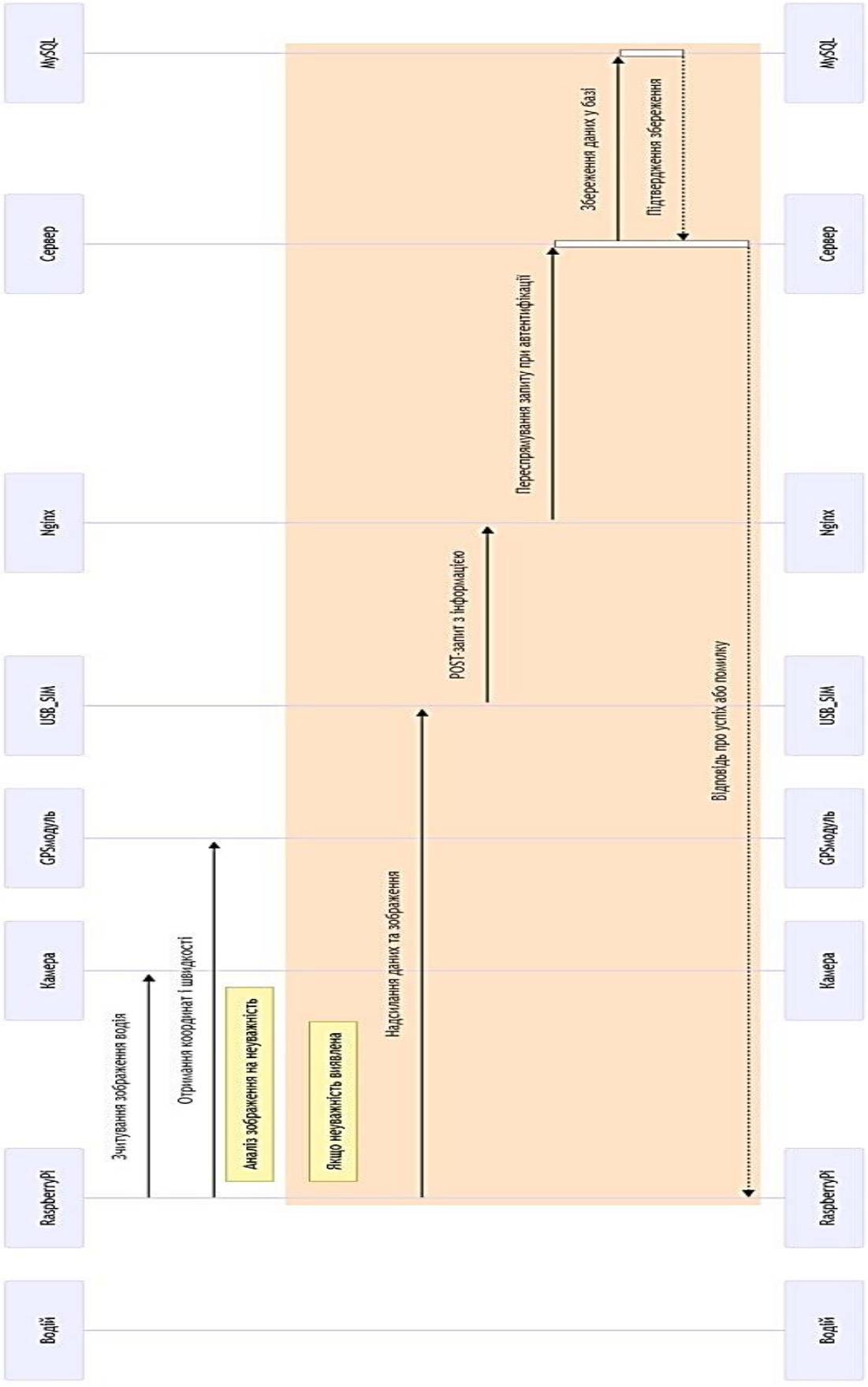


Рисунок 2.8 – Опис діаграми послідовності для системи моніторингу уваги водія

У межах даного проєкту для фіксації, обробки та збереження подій, пов’язаних із неуважністю водія, було спроєктовано логічну структуру реляційної бази даних на основі СКБД MySQL [54]. Розроблена схема охоплює ключові об’єкти системи: користувачів, пристрої, зафіксовані інциденти, пов’язані з ними зображення, а також системні журнали подій. Така побудова дозволяє ефективно організувати збереження, доступ і обробку інформації, яка надходить з пристрою на базі Raspberry Pi, що встановлений у транспортному засобі та виконує функцію моніторингу уваги водія в реальному часі.

Центральним елементом є таблиця incidents, де зберігається інформація про кожен випадок виявленого відволікання - зокрема, координати, швидкість руху, час інциденту, тривалість неуважності та відповідне зображення. Зв’язок з таблицею devices забезпечує прив’язку до конкретного пристрою, а через нього - до зареєстрованого в системі користувача. Додатково таблиця images дає змогу зберігати декілька зображень, пов’язаних з одним інцидентом, наприклад, різні кадри або типи зображення (фото обличчя, повного кадру тощо). Таблиця logs фіксує події на рівні пристрою, що дозволяє моніторити стабільність роботи системи, виявляти помилки або підозрілі дії.

Запропонована SQL-схема є гнучкою, розширюваною та узгодженою з основною метою проєкту - створення надійної та масштабованої інфраструктури для збору й збереження критичних даних, пов’язаних із безпекою дорожнього руху. Така структура дозволяє не лише фіксувати відволікання водіїв, а й надалі проводити статистичний аналіз, здійснювати фільтрацію подій за місцем, часом або користувачем, що значно підвищує прикладну цінність розробленої системи.

Модель бази даних з таблицями та їх зв’язками зображено на рисунку 2.9.

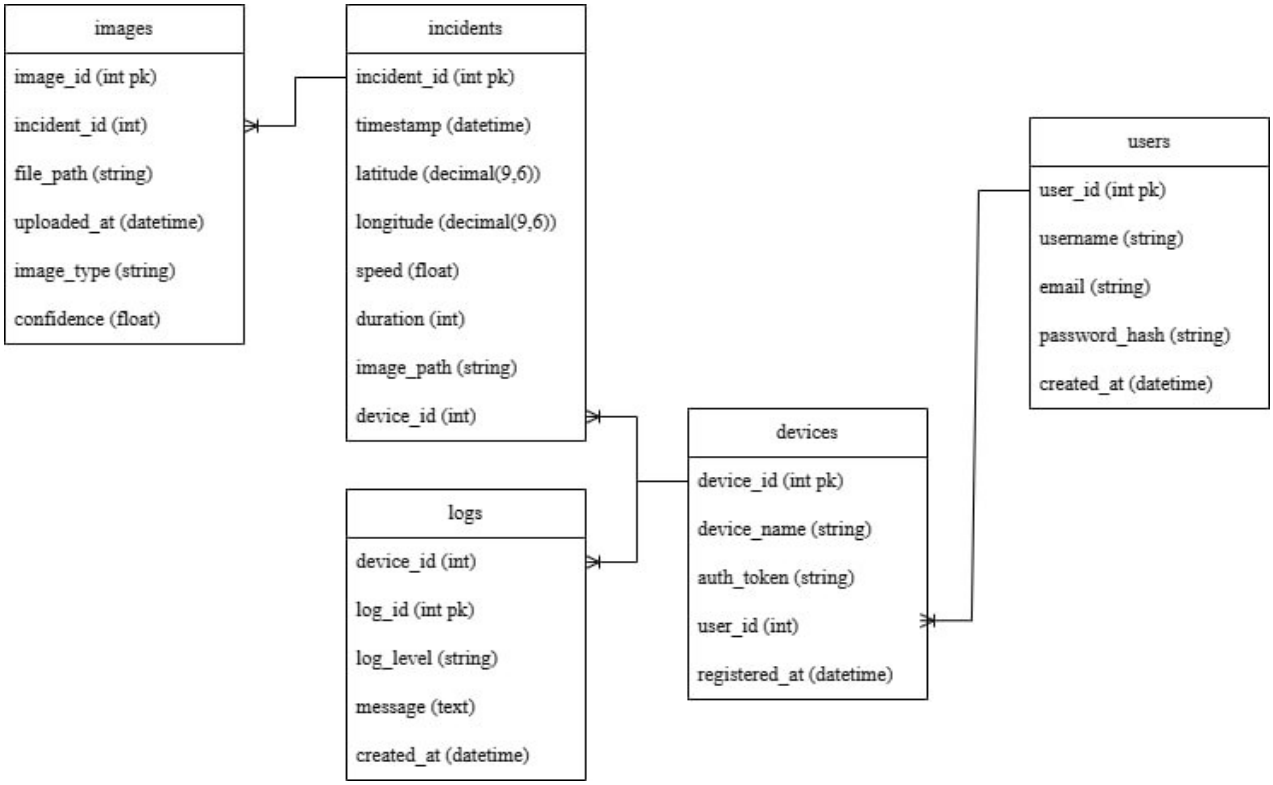


Рисунок 2.9 – ERD-модель бази даних системи виявлення неуважності

2.4 Висновки до другого розділу

У межах даного розділу було здійснено повноцінне попереднє проєктування системи моніторингу уваги водія, що поєднує апаратну й програмну складові. Було детально проаналізовано ключові компоненти системи, визначено їхню функціональність, технічні характеристики та взаємодію між собою. Такий підхід дозволив сформувати цілісну архітектуру, здатну забезпечити стабільну й ефективну роботу в реальному часі.

Особливу увагу приділено вибору обчислювальної платформи Raspberry Pi 5, яка завдяки своїй продуктивності й гнучкості стала основою для реалізації складних задач комп’ютерного зору. У поєднанні з камерою високої чіткості, модулями зв’язку та іншими периферійними пристроями, система набуває комплексного функціоналу: розпізнавання стану користувача, генерація попереджень, збереження та передача статистичних даних. Застосування

паралельної обробки відеоданих для аналізу ключових точок обличчя та напрямку погляду дозволяє підвищити точність визначення моментів неуважності. Логіка роботи алгоритму побудована з урахуванням реальних умов експлуатації: забезпечено як оперативне реагування на загрозливі стани, так і можливість передачі аналітичної інформації на сервер для подальшого аналізу.

Також було передбачено механізми зворотного зв’язку з користувачем, зокрема за допомогою звукових і візуальних сигналів та текстового повідомлення на LCD-екрані. Це сприяє оперативному інформуванню водія про критичні стани без необхідності відволікання від дороги. У результаті виконаного проєктування було сформовано технічно обґрунтовану структуру програмно-апаратного засобу, яка забезпечує як функціональність, так і подальшу масштабованість системи. Обрана архітектура гарантує надійну роботу навіть в умовах обмежених ресурсів, а також відкриває можливості для інтеграції нових функцій у майбутньому.

3 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ ЗАСОБУ ДЛЯ МОНІТОРИНГУ УВАГИ ЛЮДИНИ З ВИКОРИСТАННЯМ АНАЛІЗУ ВІДЕОПОТОКУ В РЕАЛЬНОМУ ЧАСІ

3.1 Апаратна інтеграція та початкове налаштування системи

Реалізація програмно-технічного засобу для моніторингу неуважності водія розпочинається з фізичної інтеграції основних апаратних модулів. Центральним елементом системи виступає одноплатний комп’ютер Raspberry Pi 5 [9], який має достатню обчислювальну потужність для роботи з відеопотоком, обробки зображень у реальному часі та обміну даними з іншими пристроями.

Камера Raspberry Pi Camera Module 3, як головний компонент для візуального моніторингу, підключається до роз'єму CSI (Camera Serial Interface). Такий тип з'єднання дозволяє передавати відео з високою роздільною здатністю з мінімальною затримкою, що є критично важливим для точного визначення відволікання. Для виводу поточного стану системи використовується LCD-дисплей 16x2, який за допомогою I2C-адаптера підключається до GPIO-пінів Raspberry Pi. Використання I2C дозволяє зекономити кількість задіяних пінів, а також спрощує схему підключення.

Зуммер (buzzer) та світлодіодний індикатор (LED) використовуються як засоби оповіщення. Обидва пристрої підключаються безпосередньо до цифрових виходів GPIO. У випадку виявлення ознак неуважності, Raspberry Pi активує ці пристрої відповідними сигналами, що дозволяє оперативно попередити водія як візуально, так і звуково.

Для отримання координат і швидкості руху використовується GPS-модуль типу U-blox NEO-6M, який з’єднується з Raspberry Pi через UART-інтерфейс. Модуль потребує стабільного живлення (3.3-5 В), а передача даних здійснюється через TX/RX-піни. Вказані GPS-дані дозволяють доповнити події відволікання точним місцем фіксації, що може бути використано для подальшого аналізу.

Ще одним важливим елементом є USB 4G/LTE-модем, який забезпечує мобільне підключення до Інтернету. Він підключається до одного з USB-портів Raspberry Pi. Це дозволяє системі надсилати зібрані логи, зображення та GPS-координати на віддалений сервер, забезпечуючи незалежність від локальної інфраструктури. Наявність бездротового зв’язку відкриває можливість для централізованого моніторингу в режимі реального часу. Зібрана система потребує належного живлення. Для стабільної роботи Raspberry Pi 5 та всіх підключених периферійних пристроїв використовується блок живлення на 5В 5А, який підключається через роз’єм USB-C. Окремі компоненти, такі як GPS або дисплей, можуть також вимагати захисту від перенапруг, що реалізується за допомогою зовнішніх модулів стабілізації живлення або логічних рівнів.

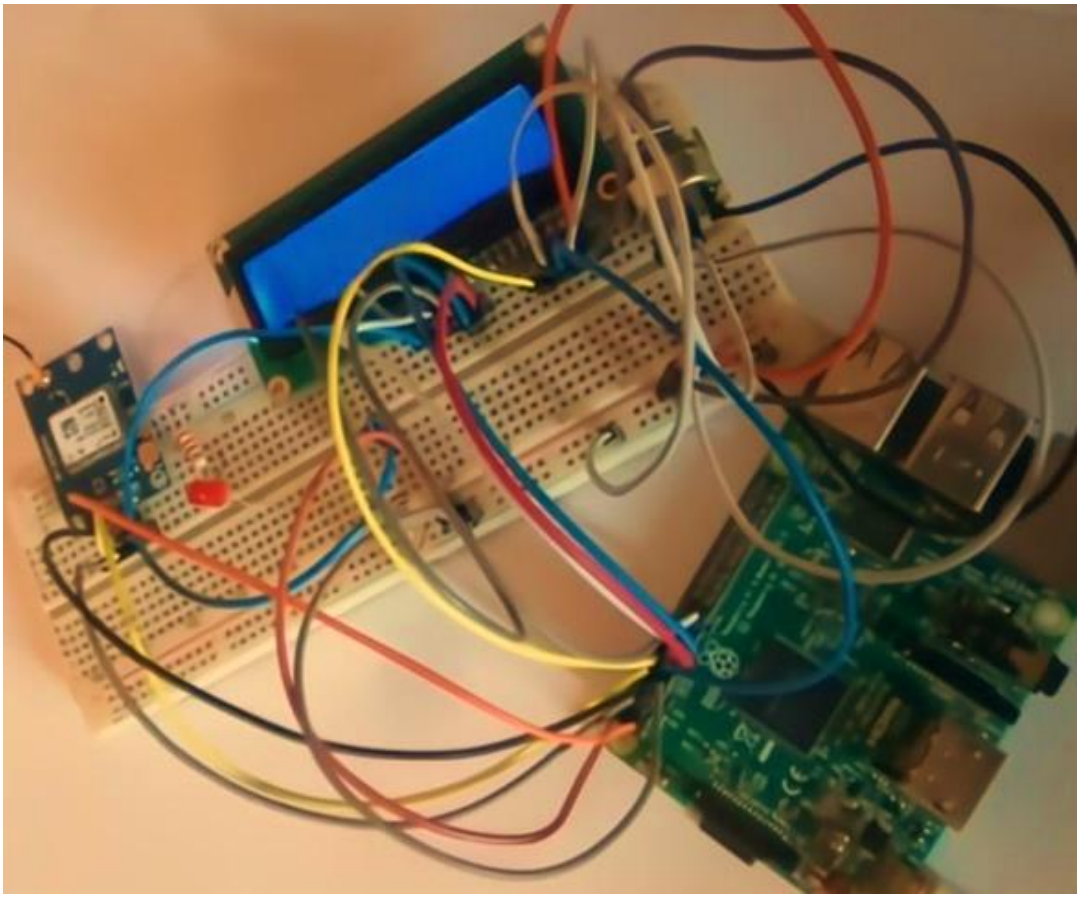


Рисунок 3.1 – Зображення апаратних з’єднань компонентів Raspberry Pi

Підключення всіх апаратних компонентів до Raspberry Pi 5 здійснюється через GPIO-піни з використанням стандартних інтерфейсів. Камера Camera Module 3 під'єднується через CSI-порт для стабільного захоплення відео. LCD-екран 16x2 працює в 4-бітному режимі та підключений до GPIO17, GPIO18, GPIO27, GPIO22, GPIO23 і GPIO10, з підсвіткою на GPIO2. GPS-модуль NEO-6M використовує UART-з'єднання через GPIO14 (TX) і GPIO15 (RX) для передачі координат у форматі NMEA. Звуковий сигналізатор (зуммер) підключено до GPIO24, а червоний LED - до GPIO20, обидва керуються логічними сигналами. Живлення подається через 3.3В або 5В пін, в залежності від вимог пристроїв. Уся система побудована з урахуванням оптимального використання портів та енергоспоживання.

Таблиця 3.1 – Опис підключень компонентів до GPIO Raspberry Pi

	Компонент	GPIO-пін	Фізичний пін	Призначення
1.	LCD RS	GPIO17	Пін 11	Управління регістрами LCD задає режим роботи дисплея.
2.	LCD EN	GPIO18	Пін 12	Сигнал "Enable" активує передачу даних.
3.	LCD D4	GPIO27	Пін 13	Передача даних (біт 4) надсилає частину інформації.
4.	LCD D5	GPIO22	Пін 15	Передача даних (біт 5) доповнює бітовий потік.
5.	LCD D6	GPIO23	Пін 16	Передача даних (біт 6) додає ще один біт.
6.	LCD D7	GPIO10	Пін 19	Передача даних (біт 7) завершує передавання байта.
7	LCD підсвітка	GPIO2	Пін 3	Вмикання підсвітки, освітлює екран.

Кінець таблиці 3.1 – Опис підключень компонентів до GPIO Raspberry Pi

8.	LED-індикатор	GPIO20	Пін 38	Сигнал попередження вмикає світлодіод.
9.	Зумер (buzzer)	GPIO24	Пін 18	Аудіо-сигнал неуважності активує зумер.
10.	GPS TX - RX	GPIO15	Пін 10	Прийом даних з GPS здійснюється через UART
11.	GPS RX - TX	GPIO14	Пін 8	Передача даних до GPS використовується для команд.

3.2 Розгортання операційної системи та підготовка програмного середовища

Для повноцінної роботи програмно-технічного засобу, зокрема обробки відеопотоку, збору телеметрії та комунікації з серверною частиною, необхідно підготувати операційну систему на одноплатному комп’ютері Raspberry Pi 5. Обрана система - Raspberry Pi OS (64-bit версія на базі Debian), яка забезпечує стабільну підтримку драйверів, сумісність із більшістю бібліотек для роботи з камерами, модулями GPS, дисплеями та іншими периферійними пристроями, що використовуються у складі системи. Саме ця ОС дозволяє легко інтегрувати як низькорівневе керування GPIO, так і високорівневі серверні процеси.

На початковому етапі виконується запис образу Raspberry Pi OS на карту пам’яті (типово microSD обсягом 32-64 ГБ) за допомогою утиліт на кшталт Raspberry Pi Imager або balenaEtcher. Після цього здійснюється початкове налаштування - створення користувача, підключення до мережі Wi-Fi або Ethernet, оновлення пакетної бази та активація SSH для віддаленого адміністрування. Також рекомендується увімкнути інтерфейси камери, I2C, SPI, та UART через

конфігураційне меню raspi-config - це забезпечує подальший доступ до всіх фізично підключених компонентів.

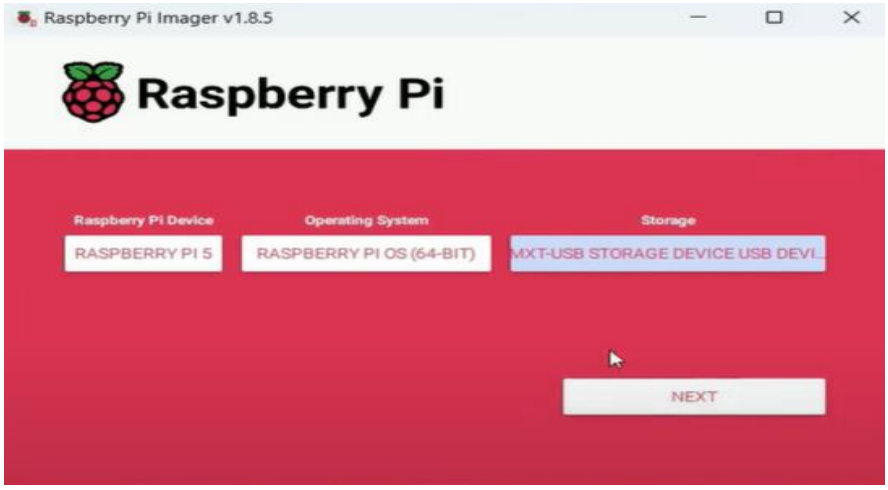


Рисунок 3.2 – Встановлення Raspberry Pi OS на microSD карту

Після встановлення Rasp OS на microSD карту потрібно вставити її в слот Raspberry Pi, підключити живлення, монітор (або активувати SSH для віддаленого доступу), дочекатися першого запуску системи та виконати базове налаштування - вибір мови, часового поясу, підключення до мережі та оновлення пакетів.

Наступним кроком є оптимізація системи - видалення зайвих пакетів, обмеження графічного інтерфейсу або повна його деактивація, що дозволяє зменшити навантаження на систему та зекономити ресурси. Після цього Pi готовий до встановлення всіх компонентів.

Після встановлення Raspberry Pi OS на microSD-карту необхідно оновити систему до останніх версій пакетів. Для цього виконується стандартна команда оновлення - це двохетапний процес, що складається з двох послідовних команд. Перша команда синхронізує списки пакетів з репозиторіями. Вона завантажує найновіші списки доступних пакетів та їхніх версій, але не встановлює жодних оновлень. Цей крок лише "інформує" систему про те, що доступно для оновлення. Друга команда вже безпосередньо завантажує та встановлює ці оновлення. Результат наведено нижче на рисунку 3.3.

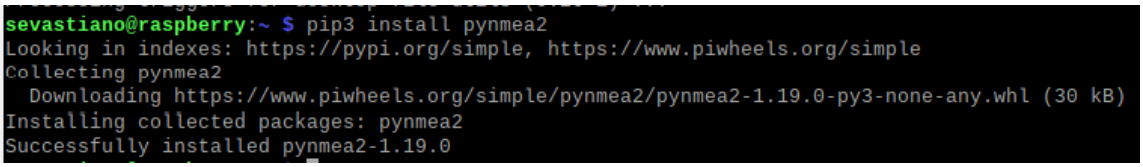
```
sevastiano@raspberry:~ $ sudo apt update && sudo apt upgrade -y
Hit:1 http://deb.debian.org/debian bullseye InRelease
Hit:2 http://deb.debian.org/debian-security bullseye-security InRelease
Hit:3 http://deb.debian.org/debian bullseye-updates InRelease
Hit:4 http://archive.raspberrypi.org/debian bullseye InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
1 package can be upgraded. Run 'apt list --upgradable' to see it.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages were automatically installed and are no longer required:
  libopengl0 libre2-9 libwpe-1.0-1 libwpebackend-fdo-1.0-1 sse3-support
Use 'sudo apt autoremove' to remove them.
The following packages have been kept back:
  linux-image-amd64:amd64
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
```

Рисунок 3.3 – Синхронізація репозиторіїв та системи

Далі встановлюються базові пакети, необхідні для роботи з модулем GPS, обробки послідовних даних та виведення інформації на дисплей. Встановлення усіх необхідних компонентів зображено нижче на рисунку 3.4.

```
sevastiano@raspberry:~ $ sudo apt install gpsd gpsd-clients python3-gps minicom python3-pip -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docutils-common libopengl0 libre2-9 libwpe-1.0-1 libwpebackend-fdo-1.0-1 mypy pylint
  python3-astroid python3-asttokens python3-docutils python3-isort python3-lazy-object-proxy
  python3-logilab-common python3-mccabe python3-mypy python3-mypy-extensions python3-roman
  python3-send2trash python3-typed-ast python3-typing-extensions python3-wrapt sgml-base
  sse3-support xml-core xsel
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  gpsd gpsd-clients minicom python3-gps python3-pip
0 upgraded, 5 newly installed, 0 to remove and 1 not upgraded.
Need to get 1,350 kB/1,687 kB of archives.
After this operation, 6,435 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bullseye/main i386 gpsd i386 3.22-4 [409 kB]
Get:2 http://deb.debian.org/debian bullseye/main i386 python3-gps i386 3.22-4 [164 kB]
Get:3 http://deb.debian.org/debian bullseye/main i386 gpsd-clients i386 3.22-4 [486 kB]
Get:4 http://deb.debian.org/debian bullseye/main i386 minicom i386 2.8-2 [292 kB]
Fetched 1,350 kB in 2s (841 kB/s)
Selecting previously unselected package gpsd.
(Reading database ... 174092 files and directories currently installed.)
Preparing to unpack .../archives/gpsd_3.22-4_i386.deb ...
Unpacking gpsd (3.22-4) ...
Selecting previously unselected package python3-gps.
Preparing to unpack .../python3-gps_3.22-4_i386.deb ...
Unpacking python3-gps (3.22-4) ...
Selecting previously unselected package gpsd-clients.
```

а)



б)

а) – встановлення необхідних пакетів, б) – встановлення бібліотеки Python для протоколу NMEA

Рисунок 3.4 – Встановлення необхідний компонентів для роботи з GPS

Пакет gpsd є ключовим компонентом програмного забезпечення для роботи з GPS на Raspberry Pi. Це системний демон, який працює у фоновому режимі та відповідає за прийом, обробку й передачу GPS-даних, що надходять у форматі NMEA від зовнішніх або внутрішніх GPS-модулів. Його основна функція - уніфікація доступу до даних з різних джерел GPS і надання стандартного API, з яким можуть працювати різні клієнтські програми.

Для тестування та взаємодії з gpsd використовується пакет gpsd-clients, який містить зручні консольні утиліти, зокрема cgps. Утиліта cgps дозволяє у реальному часі відображати координати, кількість супутників, поточний час, швидкість, висоту та інші дані в зручному форматі прямо у терміналі. Вона є незамінною при первинній перевірці правильності підключення GPS-модуля та коректної обробки даних демоном gpsd. Крім того, для програмного доступу до GPS-даних у Python зручно використовувати бібліотеку pynmea2. Ця бібліотека надає функціонал для розбору сирих NMEA-повідомлень, що надходять із GPS-модуля, та дозволяє програмно витягувати з них окремі поля: широту, довготу, час фіксації, якість сигналу, кількість супутників тощо. pynmea2 дозволяє легко інтегрувати GPS-функціональність у більші проєкти, що використовують Raspberry Pi, без необхідності глибоко занурюватися у специфікацію NMEA.

Таким чином, зв’язка gpsd, gpsd-clients та pynmea2 забезпечує повний цикл обробки, тестування та інтеграції GPS-даних - від апаратного рівня до програмної обробки в прикладному рівні.

У ході тестування функціональності системи було проведено перевірку коректності прийому даних від GPS-модуля. Після налаштування з'єднання через послідовний порт Raspberry Pi з модулем типу U-blox NEO-6M було отримано потік NMEA-повідомлень, який містить стандартні GPS-рядки, зокрема $GPGGA, $GPRMC, $GPGSV та інші. Дані включають поточну широту та довготу, кількість видимих супутників, точність позиціювання, висоту над рівнем моря та швидкість руху.

```
$GPGGA,235718.78,4916.2004,N,02725.7305,E,0,3,100.000,0.000,M,,M,0,*7B
$GNGSA,A,1,14,17,30,,,,,,,,,,,,,100.000,100.000,100.000*2F
$GPGSA,A,1,14,17,30,,,,,,,,,,,,,100.000,100.000,100.000*31
$GLGSA,A,1,,,,,,,,,,,,,,,,,100.000,100.000,100.000*2D
$BDGSA,A,1,,,,,,,,,,,,,,,,,100.000,100.000,100.000*20
$QZGSA,A,1,,,,,,,,,,,,,,,,,100.000,100.000,100.000*2D
$GPGSV,7,1,28,2,5,25,,10,8,335,,12,6,235,,13,53,176,*4C
$GPGSV,7,2,28,14,45,61,17,15,55,248,,17,47,99,24,19,39,141,10*7D
$GPGSV,7,3,28,22,64,63,,23,17,295,,24,38,292,,30,18,105,16*44
$GLGSV,7,4,28,65,54,253,,66,32,323,,72,21,192,,74,38,58,*50
$GLGSV,7,5,28,75,69,321,,76,21,262,,83,16,24,,84,24,79,*6A
$BDGSV,7,6,28,201,5,,,202,5,,,203,5,,,210,5,,*62
$BDGSV,7,7,28,211,26,86,,212,15,35,,213,14,320,,214,5,,*6F
$GPRMC,235718.78,V,4916.2004,N,02725.7305,E,0.000,0.000,310525,,E,N*09
```

Рисунок 3.5 – NMEA-повідомлення, отримані від GPS-модуля під час тестування

На рисунку 3.5 продемонстровано результат успішного тестування прийому даних з GPS-модуля. Після запуску модуля через послідовний інтерфейс на Raspberry Pi система почала отримувати велику кількість GPS-рядків у форматі NMEA, серед яких $GPGGA, $GPRMC, $GPGSA, $GPGSV тощо. Ці повідомлення містять різноманітну інформацію: координати, час, кількість супутників, точність фіксації, швидкість руху, висоту над рівнем моря тощо. Під час тестування спостерігалося, що дані оновлюються з певною періодичністю, залежно від рівня сигналу та кількості супутників. Усі отримані повідомлення виводилися в консоль у реальному часі, що дозволяло оперативно контролювати їх зміст і цілісність. Також важливо, що повідомлення містили контрольні суми, які можуть бути використані для перевірки достовірності даних. Для реалізації проекту необхідно обробляти лише ключові повідомлення. Зокрема, рядок $GPGGA використовується

для отримання координат широти та довготи, а $GPRMC - для визначення швидкості руху. Інші дані, хоч і інформативні, не є критично необхідними для поставленої задачі, тому не обробляються. Такий підхід дозволяє оптимізувати роботу системи та зосередитись на найважливіших параметрах.

3.3 Структура програмного забезпечення для моніторингу уваги

Програмне забезпечення, реалізоване в рамках цього проєкту, складається з набору методів, кожен з яких виконує конкретне завдання в системі моніторингу уваги водія. Алгоритм взаємодіє з камерою, аналізує зображення обличчя для визначення стану очей і напрямку погляду, а також обробляє сигнали від GPS-модуля. У разі виявлення ознак неуважності система сповіщає користувача за допомогою світлових та звукових сигналів, а також виводить відповідну інформацію на LCD-екрані. Додатково передбачена можливість дистанційної передачі даних на сервер. В таблиці 3.2 наведено головні методи для роботи ПЗ моніторингу уваги людини з використанням відеопотоку в реальномучасі..

Таблиця 3.2 – Опис методів ПЗ моніторингу уваги людини

	Назва функції	Опис функції
1.	start_camera()	Ініціалізує камеру та починає захоплення відеопотоку.
2.	detect_face()	Визначає наявність обличчя у кадрі для подальшого аналізу.
3.	detect_eyes()	Перевіряє стан очей (відкриті/закриті) на основі кадру з обличчям.
4.	track_pupil()	Визначає напрям погляду на основі координат зіниці.
5.	alert_user()	Увімкнення сигналу (бузер, LED) у разі виявлення неуважності.

Кінець таблиці 3.2 – Опис методів ПЗ моніторингу уваги людини

6.	display_status()	Виводить статус (уважний, обличчя не знайдено тощо) на LCD екран.
7.	read_gps_data()	Зчитує координати та швидкість руху з GPS-модуля.
8.	send_to_server()	Передає дані (координати, стан уваги) на віддалений сервер (якщо увімкнено).
9.	toggle_monitoring()	Вмикає або вимикає моніторинг
10.	toggle_server_sync()	Керує передачею даних на сервер (ON/OFF)

Програмне забезпечення реалізовано як набір взаємодіючих модулів, що забезпечують виявлення ознак неуважності водія в реальному часі. Основний скрипт main.py ініціалізує всі компоненти системи, завантажує конфігурації та запускає циклічний алгоритм обробки відео з камери. Модуль gaze відповідає за аналіз напрямку погляду та визначення стану очей (відкриті чи закриті). На основі координат зіниці виконується аналіз фокусу уваги користувача.

Модуль gps обробляє дані з GPS-приймача, дозволяючи визначити координати і швидкість руху. Вся актуальна інформація виводиться на LCD-дисплей, щоб користувач міг бачити причини спрацювання сигналу або поточний стан системи (наприклад, відсутність обличчя в кадрі). Модуль notifier активує звукову сигналізацію (бузер) та індикацію (LED), якщо система фіксує втрату уваги.

Додатково, передбачена можливість керування системою через кнопки — користувач може вимкнути моніторинг або тимчасово зупинити передачу даних на сервер. Комунікація з віддаленим сервером реалізована в модулі server, де дані передаються в реальному часі або зберігаються у разі відсутності з'єднання. Завдяки модульній структурі, програму легко масштабувати або адаптувати до нових умов.

Структуру проєкту зображено на рисунку 3.6.

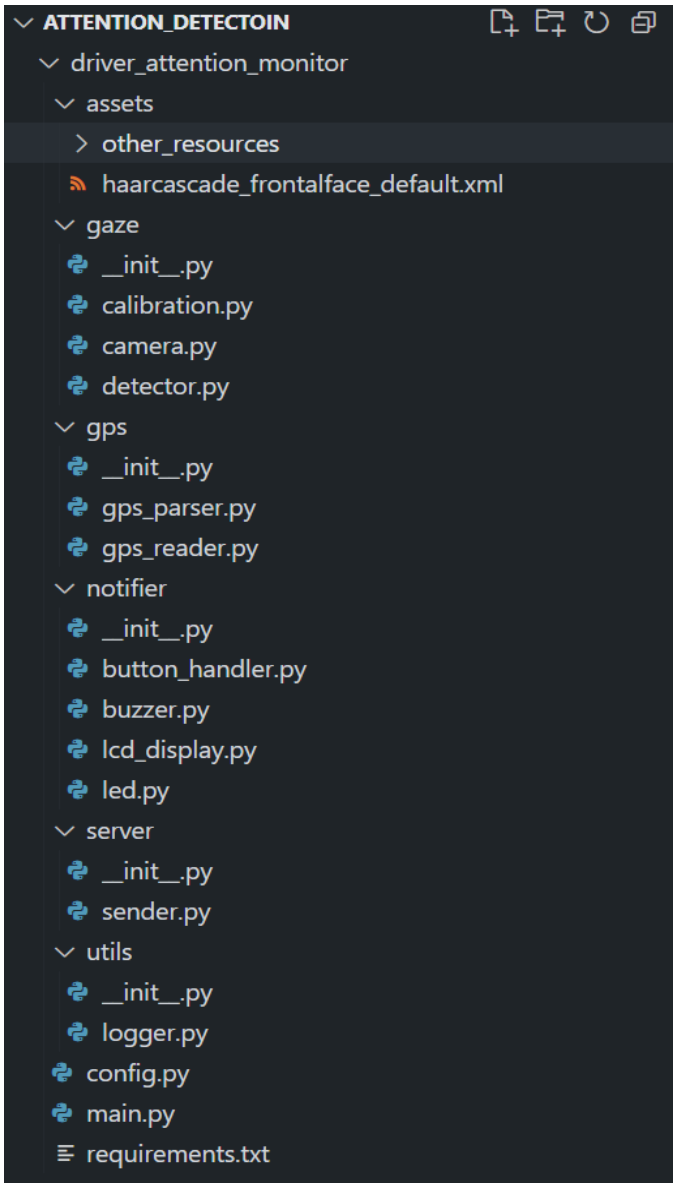


Рисунок 3.6 – Структура проєкту моніторингу уваги людини

Для забезпечення ефективної роботи системи на Raspberry Pi реалізовано локальну буферизацію даних перед їх передачею на сервер. Усі дані (зображення, координати, час, статус уваги водія) тимчасово зберігаються у внутрішній структурі папок, що організована за датами та категоріями. Інциденти кодуються у форматі JSON і доповнюються супутньою інформацією в локальній базі даних SQLite, запит до якої зображено на рисунку 3.7.

Зображення стискаються у формат JPEG або WebP з якістю 60-75%, що дозволяє зменшити їх розмір до 150-250 КБ при збереженні розбірливості.

Нестратегічні кадри (без підтверджених інцидентів) розміщуються в кеші з обмеженим терміном життя - зазвичай 24 години. Для очищення кешу застосовуються періодичні скрипти на Bash або Python.

Логи роботи системи ведуться у форматі .log з циклічною ротацією (logrotate), після чого можуть архівуватись у .gz. У разі успішної відправки даних на сервер, пов’язані локальні записи видаляються або маркуються як передані. Це дозволяє уникнути надмірного заповнення пам’яті, підтримувати актуальність даних і забезпечити безперервність роботи системи навіть при нестабільному інтернет-з'єднанні.

```
sevastiano@raspberry:~$ sqlite3 -header -column incidents.db "SELECT * FROM incidents;"
id  timestamp            latitude  longitude  speed  image_path                      duration
--  -------------------  --------  ---------  -----  ------------------------------  --------
13  2025-04-29T08:15:22  49.27     27.4203    47.5   images/2025-04-29/incident1.jpg  3
14  2025-04-29T09:03:47  49.2698   27.4205    53.2   images/2025-04-29/incident2.jpg  4
15  2025-04-29T10:45:11  49.2701   27.4199    0.0    images/2025-04-29/incident3.jpg  2
```

Рисунок 3.7 – SQL запит до локальної бази даних на пристрої Raspberry Pi

Віддалений сервер виконує функцію централізованого вузла збору та обробки телеметричних даних, які надсилаються з пристроїв моніторингу водія на Raspberry Pi. Сервер приймає HTTP-запити з JSON-пакетами, що містять координати, швидкість, статус уваги, зображення обличчя водія та часову мітку події. Усі запити проходять через авторизаційний фільтр, що перевіряє унікальний токен пристрою, і зберігаються в базі даних MySQL за допомогою окремих сервісів NestJS-модулів

Архітектура побудована на розділенні обов’язків: кожен модуль відповідає за свою частину - облік пристроїв, реєстрацію інцидентів, логування подій, та доступ до API для фронтенду. Кожен пристрій має свій унікальний запис у таблиці devices, зв’язаний із переданими ним інцидентами в таблиці incidents, а зображення та GPS-дані зберігаються в окремих таблицях або файловій системі з посиланням у БД.

Фронтенд-частина реалізована як SPA-додаток на Vue.js, який через REST API звертається до серверу для отримання даних. Основні функції адмінпанелі включають:

1) інтерактивну карту з поточними геокоординатами кожного активного пристрою (реалізовано через Leaflet.js);

2) сторінку пристроїв, де видно їх статус (онлайн/офлайн), останній зв'язок, загальна кількість інцидентів;

3) журнал інцидентів з фільтрацією за датою, пристроєм, координатами або рівнем тривоги;

4) відображення зображень із камери та короткої інформації (швидкість, час, статус уваги) для кожної події;

Пристрої, як-от Raspberry Pi з камерою, періодично або подієво (наприклад, під час інцидентів) надсилають дані на сервер через HTTP-запити (зазвичай POST або PUT). Ці запити містять JSON-структуровану інформацію: координати GPS, швидкість руху, тип події (наприклад, "закриті очі").

Адміністративна панель звертається до сервера через HTTP GET-запити для отримання актуального стану пристроїв, їх статусів, історії інцидентів та статистики. Усі відповіді повертаються у вигляді JSON і рендеряться React-компонентами. Адмін може застосовувати фільтри до запитів, щоб зменшити обсяг трафіку і спростити обробку даних на фронтенді.

Візуальну складову адміністративної панелі подано на рисунках 3.8, 3.9 та 3.10.

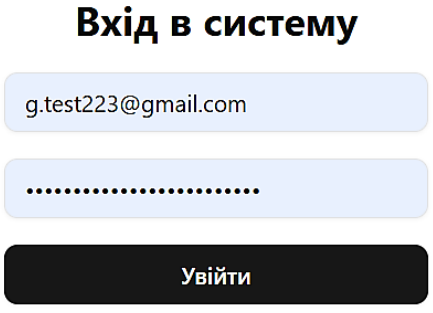
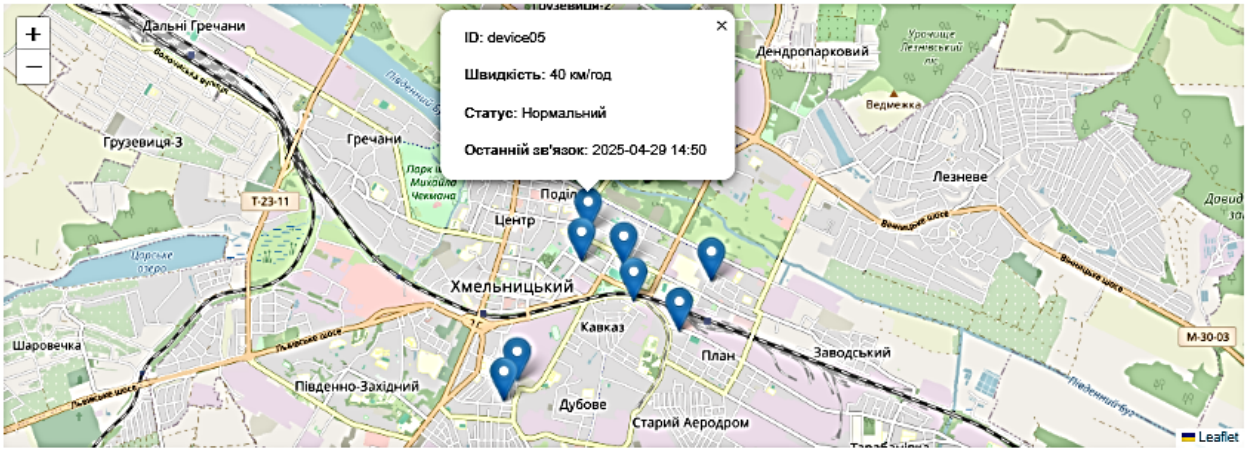


Рисунок 3.8 – Сторінка входу у панель адміністратора

Карта активних пристроїв



Список пристроїв

Пошук за ID...

ID	Назва	Статус	Останній зв'язок	Інциденти
device01	Raspberry Pi #1	Online	2025-04-29 14:35	3
device02	Raspberry Pi #2	Online	2025-04-29 14:40	1
device03	Raspberry Pi #3	Offline	2025-04-29 14:42	0
device04	Raspberry Pi #4	Online	2025-04-29 14:45	5
device05	Raspberry Pi #5	Online	2025-04-29 14:50	2
device06	Raspberry Pi #6	Online	2025-04-29 14:55	0
device07	Raspberry Pi #7	Offline	2025-04-29 15:00	1
device08	Raspberry Pi #8	Online	2025-04-29 15:05	4
device09	Raspberry Pi #9	Online	2025-04-29 15:10	0
device10	Raspberry Pi #10	Online	2025-04-29 15:15	2

Інциденти

ID	Пристрій	Тип	Час	Координати	Швидкість	Тривалість	Зображення
1	device01	Закриті очі	2025-04-29 14:32	49.4216, 26.9965	42 км/год	5 сек	Переглянути
2	device02	Не знайдено лице	2025-04-29 14:38	49.4312, 26.985	50.5 км/год	3 сек	Переглянути
3	device04	Небезпечний напрямок погляду	2025-04-29 14:46	49.408, 26.978	58 км/год	10 сек	Переглянути
4	device05	Закриті очі	2025-04-29 14:52	49.425, 26.991	39 км/год	7 сек	Переглянути
5	device08	Не знайдено лице	2025-04-29 15:07	49.42, 27.01	34.5 км/год	15 сек	Переглянути
6	device01	Небезпечний напрямок погляду	2025-04-29 14:33	49.4217, 26.996	43 км/год	4 сек	Переглянути
7	device02	Закриті очі	2025-04-29 14:41	49.431, 26.9855	51 км/год	6 сек	Переглянути
8	device04	Не знайдено лице	2025-04-29 14:47	49.4082, 26.9785	59 км/год	8 сек	Переглянути

Рисунок 3.9 – Сторінка перегляду статистики пристроїв

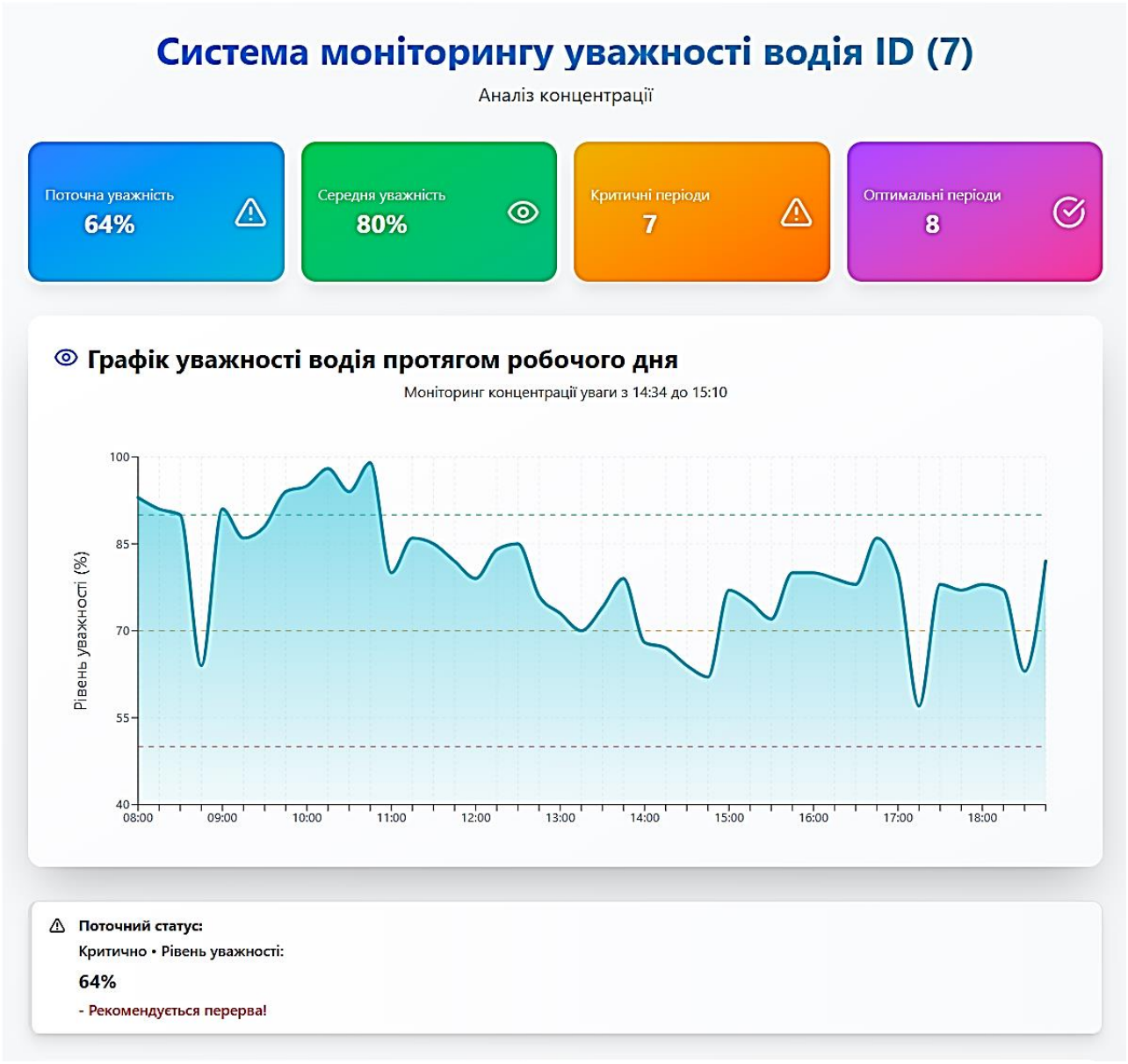


Рисунок 3.10 – Сторінка перегляду статистики для окремого користувача

Розроблена система демонструє виняткову чутливість до найменших змін рівня уважності користувача, успішно детектуючи навіть незначні флуктуації концентрації з точністю до 2-3%. Пристрій ефективно відстежує циркадні коливання уваги протягом дня, фіксуючи характерні піки активності. Завдяки тому, що дані можуть зберігатися локально доки не будуть відправлені на віддалений сервер, система дає змогу отримати всю інформацію для статистичних даних і порівнянь, тому рішення зберігати дані локально оправдовує себе.

3.4 Розгортання серверної інфраструктури та налаштування віддаленого доступу через NGINX

Для забезпечення надійного віддаленого доступу до серверної частини системи було прийнято рішення розгорнути серверну інфраструктуру у хмарному середовищі Amazon Web Services (AWS). Такий підхід забезпечує високу доступність, гнучкість масштабування та спрощує адміністрування. Серверна частина, реалізована за допомогою NestJS (REST API) та вебінтерфейс адміністратора на Vue.js, розгортаються на окремому віртуальному екземплярі (EC2), який виконує роль бекенд-сервера.

Інфраструктура побудована з урахуванням можливості горизонтального масштабування. У разі збільшення кількості пристроїв система може бути масштабована за допомогою балансувальника навантаження AWS Elastic Load Balancer (ELB) та автоматичного створення додаткових EC2-екземплярів (Auto Scaling Group).

Для зберігання зображень, які надходять із пристроїв (наприклад, кадрів з детекцією відволікання), використовується сервіс AWS S3. Зберігання зображень у S3 має ряд переваг:

1) висока доступність і стійкість до втрати даних;
2) автоматичне масштабування обсягів зберігання;
3) можливість контролю доступу до кожного об'єкта;
4) інтеграція з іншими сервісами AWS для аналітики або архівації.

Це дозволяє розвантажити основний сервер від обробки великих файлів, зосередивши його ресурси на логіці та обробці запитів API. Таким чином, архітектура системи є гнучкою, розширюваною та придатною для подальшого розвитку і використання у промислових умовах. Створення S3 bucket подано на рисунку 3.11.

Для оптимальної роботи було обрано екземпляр з достатньою кількістю оперативної пам’яті (від 2 до 4 ГБ), підтримкою постійного інтернет-з’єднання та

автоматичним резервним копіюванням. NGINX використовується як зворотній проксі-сервер, який приймає вхідні HTTP/HTTPS-запити, перенаправляє їх на відповідні сервіси (NestJS API, фронтенд) і забезпечує захист за допомогою SSL-сертифікатів (наприклад, через Let's Encrypt).

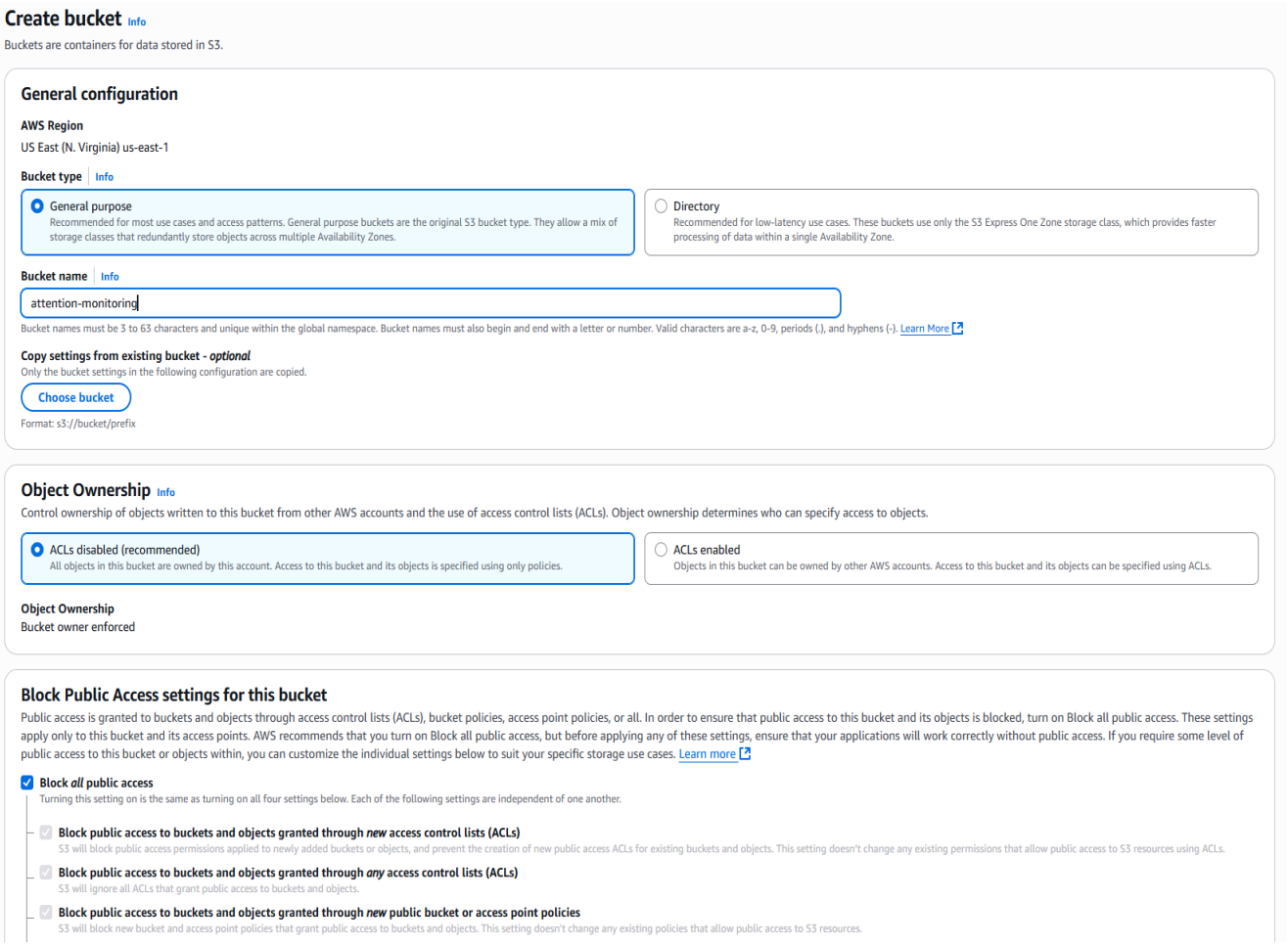


Рисунок. 3.11 – Створення сервісу AWS S3 для збереження файлів

Ці налаштування забезпечують безпечне та централізоване зберігання даних з можливістю подальшої інтеграції з бекендом або CDN. Для розгортання серверної частини було створено інстанс в AWS EC2 з назвою attention-monitoring-system. Обрано образ Ubuntu Server 24.04 LTS - стабільну та підтримувану систему, яка ідеально підходить для веб-серверів. Тип інстансу t2.micro обрано як безкоштовний (Free Tier), достатній для невеликого навантаження.

Створення віртуальних серверних ресурсів подано на рисунку 3.12 та 3.13.

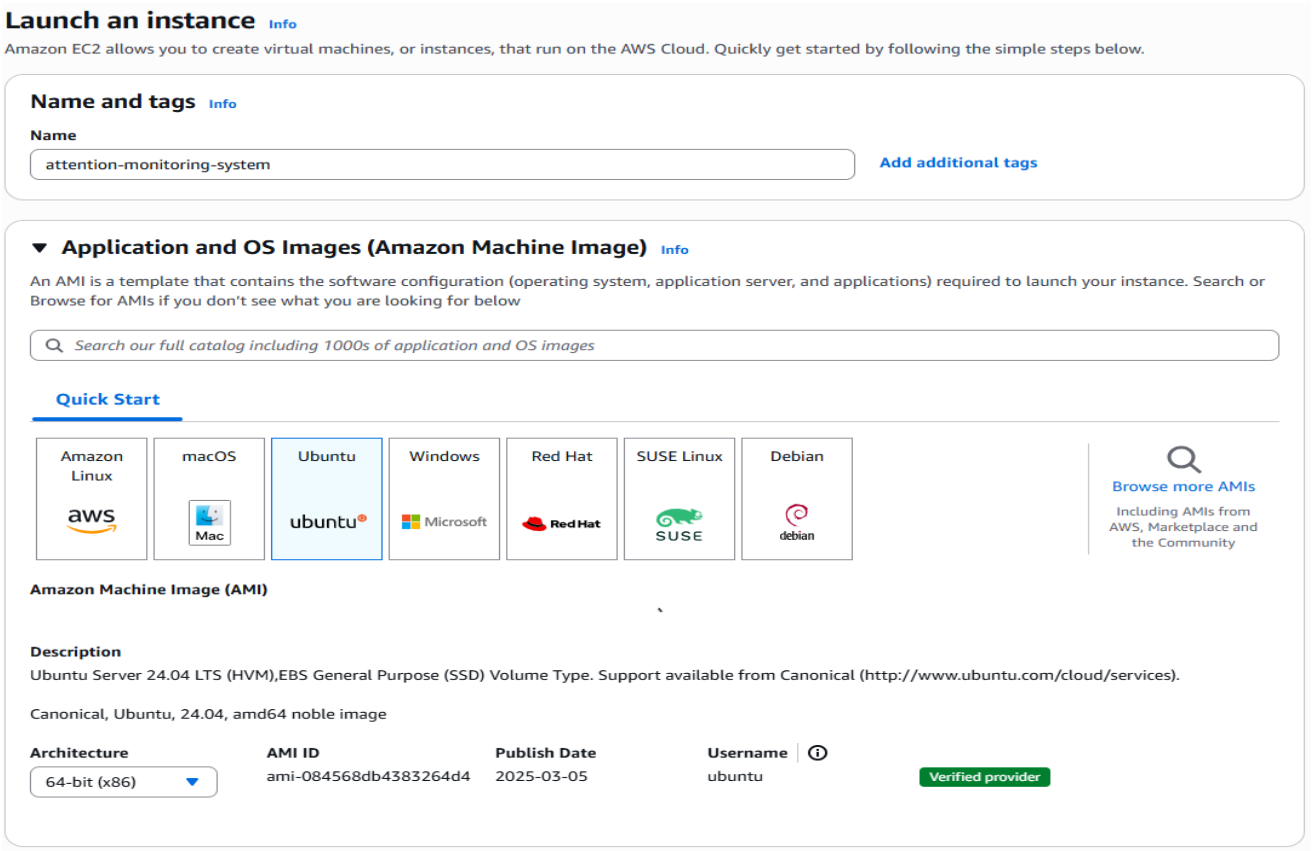


Рисунок. 3.12 – Створення сервісу EC2 для вивантаження віддаленого серверу

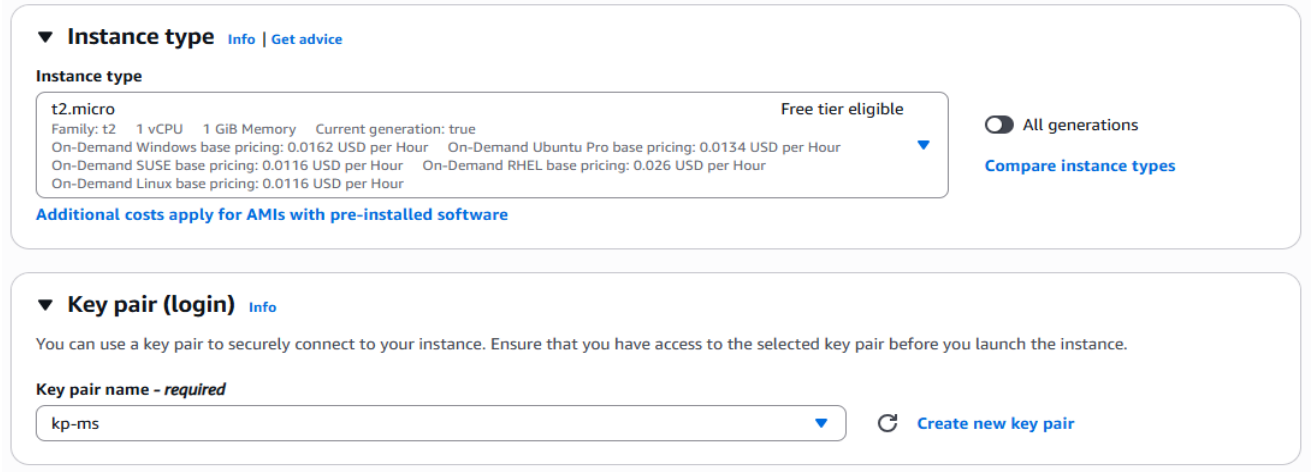


Рисунок. 3.13 – Вибір тип інстансу та встановлення RSA ключів для з'єднання

Для забезпечення зв'язку між пристроєм Raspberry Pi і віддаленим сервером, використовується легкий веб-сервер Nginx, встановлений на віртуальному сервері Amazon EC2. Це дозволяє приймати POST-запити, які містять координати GPS та

статус уважності водія (наприклад, "drowsy" або "alert"). На рисунку 3.14 зображено використання команд для встановлення веб-серверу Nginx.

```
ubuntu@ip-172-31-83-154:~$ sudo apt install nginx -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nginx is already the newest version (1.24.0-2ubuntu7.3).
0 upgraded, 0 newly installed, 0 to remove and 9 not upgraded.
```

Рисунок. 3.14 – Встановлення веб-серверу Nginx.

Далі потрібно відредагувати файл конфігурації Nginx щоб правильно опрацьовувати і перенапрвляти наші запити до запущеного на EC2 серверу оброки інформації. Основна зміна - додано обробку запитів на маршрут /upload та проксирування їх до локального серверу, або запис у лог. Для простої перевірки працездатності системи цього достатньо. Налаштування файлу конфігурації Nginx зображено на рисунку .3.15.

```
server {
    listen 80;
    server_name drowsy-attentoin-bgproj.com;

    # Logs
    access_log /var/log/nginx/attention_access.log;
    error_log  /var/log/nginx/attention_error.log;

    # Gzip compression
    gzip on;
    gzip_types text/plain application/json application/javascript text/css application/xml;
    gzip_min_length 256;

    location /upload {
        proxy_pass http://127.0.0.1:5000/upload;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_cache_bypass $http_upgrade;

        # Timeout settings
        proxy_connect_timeout 10;
        proxy_send_timeout 10;
        proxy_read_timeout 10;
```

Рисунок. 3.15 – Редагування конфігураційного файлу Nginx

Далі потрібно запустити веб сервер і перевірити чи все правильно працює, рисунок 3.16. У перспективі можлива активація HTTPS шляхом підключення TLS/SSL-сертифікатів (наприклад, через Let's Encrypt), однак для початкової версії системи передача даних допускається у незашифрованому вигляді (HTTP), оскільки мова йде про прототип та роботу в контрольованому середовищі. Після чого сервер готовий приймати запити і працювати, як довгострокове сховище інформації.

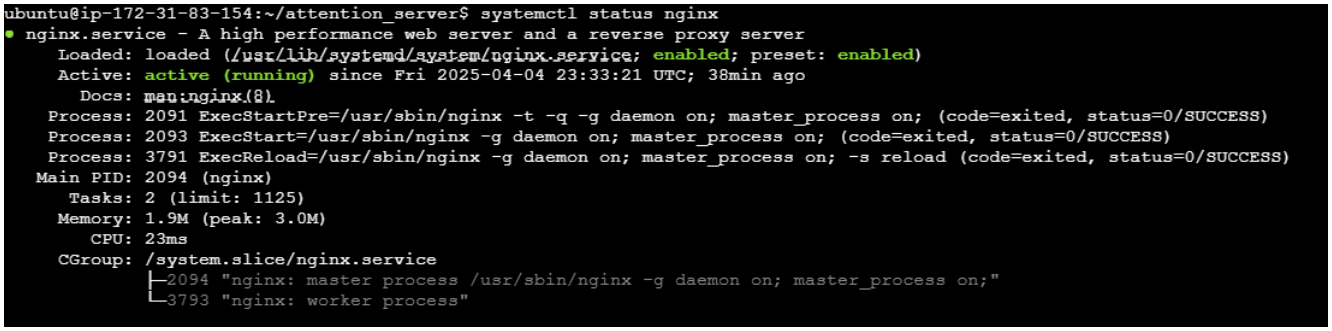


Рисунок. 3.16 – Перевірка роботи веб серверу

3.5. Висновки до третього розділу

У ході виконання третього розділу було здійснено повноцінну програмно-апаратну реалізацію пристрою для моніторингу уваги людини, який базується на платформі Raspberry Pi. На цьому етапі було обґрунтовано вибір ключових компонентів, зокрема самої одноплатної комп’ютерної системи Raspberry Pi як основи для інтеграції різних датчиків та модулів, які забезпечують функціонування системи. Основними критеріями вибору стали доступність, гнучкість конфігурації, достатній рівень обчислювальних потужностей, а також широке поширення підтримки спільнотою розробників. Визначальним кроком також стало встановлення на пристрій операційної системи Raspberry Pi OS та налаштування необхідного середовища для запуску користувацького програмного забезпечення. Були інстальовані всі необхідні бібліотеки та пакети, зокрема ті, що відповідають

за роботу з GPS-модулем, збереження даних, обробку зображення та мережеву взаємодію.

Особливу увагу було приділено тестуванню апаратної складової, зокрема GPS-модуля NEO-6M. Було проведено серію перевірок працездатності модуля, зокрема коректність зчитування координат, стабільність підключення до супутників та швидкість оновлення даних. У результаті тестів підтверджено відповідність модуля вимогам до точності та швидкодії, що дозволяє використовувати його як надійне джерело геолокаційної інформації в рамках системи моніторингу.

Зважаючи на обмежені ресурси самого Raspberry Pi, у процесі розробки було прийнято рішення реалізувати віддалений сервер для зберігання та обробки зібраної статистики, що дозволило зменшити навантаження на основний пристрій. Цей сервер було спроєктовано з урахуванням масштабованості та безпеки, що передбачає розмежування доступу та використання сучасних протоколів передачі даних. Крім того, для зручності адміністрування та перегляду зібраної інформації була реалізована веб-орієнтована адміністративна панель, яка надає змогу у зручному форматі здійснювати моніторинг, аналіз та контроль параметрів системи.

Окремо було розглянуто можливість інтеграції хмарних сервісів, таких як AWS S3 та EC2, з метою підвищення надійності зберігання даних, масштабування серверної частини, а також організації відмовостійкої інфраструктури. Таке рішення дозволяє за необхідності розширити функціональні можливості системи, зберігаючи при цьому високу продуктивність та стабільність роботи. В результаті проробленої роботи було досягнуто синергії між апаратною та програмною частинами системи, що забезпечує повноцінне функціонування пристрою моніторингу уваги людини в умовах реального використання.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було досліджено та реалізовано повноцінну кіберфізичну систему для моніторингу уваги людини в реальному часі, зокрема в контексті підвищення безпеки дорожнього руху. Основна мета полягала у створенні рішення, здатного виявляти втрату уваги водієм на основі аналізу відеопотоку з камери, із подальшою фіксацією інцидентів і передачею їх на віддалений сервер.

У першому розділі було проведено огляд актуальної проблематики, пов'язаної з людським фактором на транспорті. Зокрема, втрати уваги, втома або засинання водія є однією з ключових причин дорожньо-транспортних пригод. Розглянуто сучасні підходи до моніторингу уваги водіїв, проаналізовано переваги використання комп'ютерного зору, глибокого навчання та мікрокомп'ютерів у складі вбудованих систем.

У другому розділі було спроєктовано архітектуру кіберфізичної системи, що поєднує апаратне та програмне забезпечення. Вибір платформи Raspberry Pi як основного обчислювального пристрою був обґрунтований з огляду на її оптимальне співвідношення ціни, енергоефективності та достатньої обчислювальної потужності для виконання задач реального часу. У розробці системи також були використані камера, GPS-модуль, сигнальні пристрої (бузер, LED), а також дисплей для виведення стану пристрою. Особливу увагу було приділено модульності рішень: окремо оброблявся відеопотік, окремо - дані про координати та швидкість.

З програмного боку було реалізовано програму для розпізнавання стану очей та напрямку погляду за допомогою бібліотек OpenCV, MediaPipe, а також попередньо натренованих моделей на базі нейронних мереж. Система дозволяє ідентифікувати ознаки неуважності, такі як закриті очі або відведений погляд. При фіксації таких інцидентів генерується відповідне повідомлення, яке зберігається локально у SQLite, а також дублюється у файл логів, із можливістю подальшого

завантаження на віддалений сервер. Для оптимізації збереження зображень використовувалися формати JPEG/WebP, а також ротація логів для уникнення переповнення пам’яті.

У третьому розділі було безпосередньо реалізовано готову систему. Окремо розгорнуто віддалений сервер, на який передаються метадані та зображення, що зафіксували інциденти. Для цього було використано хмарну платформу AWS, де сервер та адміністраторська панель були розгорнуті на віртуальному сервері EC2 з використанням NGINX як зворотного проксі та балансувальника навантаження. Самі файли інцидентів (зображення, метадані) зберігаються у Amazon S3 - надійному, масштабованому та безпечному сховищі, що дозволяє зручно управляти файлами та контролювати доступ.

Також було реалізовано веб-інтерфейс для адміністратора, що дозволяє:

1) переглядати карту з пристроями у реальному часі;
2) аналізувати список зареєстрованих пристроїв;
3) переглядати історію інцидентів з фільтрацією;
4) переглядати зображення інцидентів;
5) шукати пристрої за ID або іншими критеріями.

Інтерфейс був побудований з урахуванням зручності користувача, з використанням сучасних бібліотек для таблиць, мап та компонентів UI.

Підсумовуючи результати виконаної роботи, стратегічне рішення про розгортання системи в хмарному середовищі Amazon Web Services (AWS) виявилося ключовим для досягнення поставлених цілей та забезпечення високої ефективності, надійності й безпеки. Цей вибір був зумовлений глибоким аналізом вимог до системи та прагненням створити рішення, що відповідатиме сучасним викликам і майбутнім потребам.

Насамперед, вибір AWS дозволив досягти безпрецедентної масштабованості. У контексті роботи, де передбачається постійне збільшення кількості пристроїв та обсягу даних, можливість легко додавати нові пристрої без потреби перебудовувати сервер є не просто зручністю, а фундаментальною вимогою.

Традиційні локальні рішення вимагали б значних інвестицій у фізичне обладнання та простоїв для його інтеграції, що неприйнятно для динамічного розвитку проекту. Хмарна інфраструктура AWS дозволяє автоматично адаптуватися до зростаючих навантажень, забезпечуючи безперебійну роботу системи та оптимізуючи витрати, оскільки ми сплачуємо лише за фактично використані ресурси.

У підсумку, поставлену задачу було виконано повністю. Було не лише досліджено проблему неуважності водіїв, але й реалізовано робочий прототип кіберфізичної системи з аналітикою, збором, передачею та збереженням даних, що може бути використаний у реальних умовах або надалі масштабований до комерційного продукту. Отримані результати демонструють потенціал подібних рішень у сфері безпеки транспорту, а запропонована архітектура - зручність масштабування, гнучкість і практичність.

Загалом, результати виконаної роботи демонструють повний цикл створення інтелектуальної системи моніторингу уваги людини - від аналізу проблеми та формулювання технічного завдання, до проєктування, реалізації та розгортання готового рішення. Успішна інтеграція апаратної та програмної складових, застосування методів комп’ютерного зору та нейронних мереж, а також впровадження інтерфейсу віддаленого адміністрування свідчать про практичну життєздатність запропонованого підходу. Це рішення є гнучким, масштабованим і придатним до розширення, а отже, може ефективно використовуватися як основа для впровадження подібних систем у реальних умовах

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Dasgupta A., George A., Happy S. L., Routray A. A. Vision-Based System for Monitoring the Loss of Attention in Automotive Drivers. *IEEE Transactions on Intelligent Transportation Systems*. 2013. Vol. 14, No 4. P. 1825–1838.
2. Vicente J., Laguna P., Bartra A., Bailón R. Drowsiness detection using heart rate variability. *Medical & Biological Engineering & Computing*. 2016. Vol. 54. P. 927–937.
3. Bergasa L. M., Nuevo J., Sotelo M. A., Barea R., López M. E. Real-time system for monitoring driver vigilance. *IEEE Transactions on Intelligent Transportation Systems*. 2016. Vol. 7, No 1. P. 63–77.
4. Mehmood R. M., Lee H. J. Emotion classification of speech and EEG using deep learning. *Neurocomputing*. 2017. Vol. 243. P. 26–38.
5. Soukupová T., Čech J. Real-Time Eye Blink Detection using Facial Landmarks. *21st Computer Vision Winter Workshop*. Rimske Toplice, Slovenia, 2016.
6. Seeing Machines. Driver Monitoring Systems. URL: https://www.seeingmachines.com/ (дата звернення: 05.04.2025).
7. Lee B. G., Chung W. Y. Driver Alertness Monitoring Using Fusion of Facial Features and Bio-Signals. *IEEE Sensors Journal*. 2012. Vol. 12, No 7. P. 2416–2422.
8. OpenCV Team. OpenCV Documentation. URL: https://docs.opencv.org (дата звернення: 05.04.2025).
9. Raspberry Pi Foundation. Raspberry Pi Documentation. URL: https://www.raspberrypi.com/documentation (дата звернення: 05.04.2025).
10. Raspberry Pi OS. URL: https://www.raspberrypi.com/software/operating-systems/ (дата звернення: 05.04.2025).
11. Raspberry Pi Imager. URL: https://www.raspberrypi.com/software/ (дата звернення: 05.04.2025).
12. Upton E., Halfacree G. Raspberry Pi User Guide. Wiley, 2014. 312 p.
13. Halfacree G. The Official Raspberry Pi Handbook 2021. Raspberry Pi Press, 2020. 200 p.

14. Artificial intelligence and robotics in healthcare. Siddique, H. Adeli. CRC Press, 2018. 208 p.

15. Duo W., Zhou M., Abusorrah A. A survey of cyber attacks on cyber physical systems: Recent advances and challenges. *IEEE/CAA Journal of Automatica Sinica*. 2022. Vol. 9, No 5. P. 784–800.

16. Lee E. A., Seshia S. A. Introduction to Embedded Systems: A Cyber-Physical Systems Approach. Lee & Seshia, 2015. 563 p.

17. Rajkumar R. Cyber-physical systems: The next computing revolution. *Design Automation Conference (DAC)*. 2010. P. 731–736.

18. Sanfilippo C. A review on driver monitoring systems. *IEEE Transactions on Intelligent Transportation Systems*. 2018. Vol. 19, No 12. P. 3788–3797.

19. Derler P., Putz S., Schabauer H. Cyber-physical systems: Foundations, challenges and future developments. Springer, 2019. 302 p.

20. Minichino, J. & Howse, J. Learning OpenCV 3 Computer Vision with Python. Packt Pub Ltd, 2nd ed., 2015. 266 p.

21. OpenCV Documentation. URL: https://docs.opencv.org/4.x/ (дата звернення: 05.04.2025).

22. OpenCV for Raspberry Pi. URL: https://docs.opencv.org/4.x/d7/d9f/tutorial_linux_install.html (дата звернення: 05.04.2025).

23. Mittal A., Gupta M. OpenCV 4 Computer Vision Application Programming Cookbook. Packt Publishing, 2019. 494 p.

24. Pulli K. Mobile visual computing: A survey. *Computer Vision and Image Understanding*. 2012. Vol. 116, No 3. P. 385–401.

25. Kidger P., Garcia C. Equinox: A JAX Library Based on a Single Idea. *Journal of Machine Learning Research*. 2021. Vol. 22. P. 1–8.

26. Kazemi V., Sullivan J. One millisecond face alignment with an ensemble of regression trees. *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*. 2014. P. 1867–1874.

27. MediaPipe Face Mesh. URL: https://developers.google.com/mediapipe/solutions/vision/face_landmarker (дата звернення: 05.04.2025).

28. TensorFlow Documentation. URL: https://www.tensorflow.org/api_docs (дата звернення: 05.04.2025).

29. ONNX: Open Neural Network Exchange. URL: https://onnx.ai/ (дата звернення: 05.04.2025).

30. Terence M. P. Real-time drowsiness detection using facial landmarks. *Int. Conf. on Communications and Signal Processing (ICCSP)*. 2017. P. 1916–1920.

31. Danelljan M. Global and local context for gaze estimation. *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*. 2019. P. 8234–8243.

32. Wang L. Driver drowsiness detection based on PERCLOS and yawn. *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017. P. 1–6.

33. Soukupová T., Cech J. Real-time eye blink detection using facial landmarks. *21st Computer Vision Slovak National Conference (CVCN)*. 2016. P. 40–47.

34. Krafka K. Eye tracking for everyone. Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. 2016. P. 2184.

35. Kaplan E. D., Hegarty C. J. Understanding GPS: Principles and applications. Artech House, 2017. 1064 p.

36. NEO-6M GPS Module Datasheet. URL: https://www.u-blox.com/en/product/neo-6-series (дата звернення: 05.04.2025).

37. Teunissen P., Montenbruck O. (Eds.) Springer Handbook of Global Navigation Satellite Systems. Springer, 2017. 1327 p.

38. Nginx Documentation. URL: https://nginx.org/en/docs/ (дата звернення: 05.04.2025).

39. Tevault D. A. Mastering Linux Security and Hardening. 2nd ed. Packt Publishing, 2020. 566 p.

40. Marx A., Roth R. Nginx: The definitive guide to HTTP & HTTPS on Linux. O'Reilly Media, 2018. 329 p.

41. Hunt C. TCP/IP Network Administration. 3rd Edition. O'Reilly Media, 2019. 746 p.

42. AWS Documentation. URL: https://docs.aws.amazon.com/ (дата звернення: 05.04.2025).

43. Amazon S3 User Guide. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html (дата звернення: 05.04.2025).

44. EC2 Concepts. URL: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html (дата звернення: 05.04.2025).

45. IAM Documentation. URL: https://docs.aws.amazon.com/IAM/latest/UserGuide/what-is-iam.html (дата звернення: 05.04.2025).

46. Hale S., Bostic J. AWS Certified Solutions Architect Official Study Guide: Associate Exam. Sybex, 2019. 342 p.

47. Weidman A. Deep Learning from Scratch. O'Reilly Media, 2019. 235 p.

48. Morena A. Artificial Vision and Language Processing for Robotics. Packt Publishing, 2019. 356 p.

49. Geron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, 2019. 848 p.

50. Chollet F. Deep Learning with Python. 2nd ed. Manning Publications, 2021. 504 p.

51. Vue.js Documentation. URL: https://vuejs.org/ (дата звернення: 05.04.2025).

52. Node.js Documentation. URL: https://nodejs.org/docs/ (дата звернення: 05.04.2025).

53. MongoDB Documentation. URL: https://www.mongodb.com/docs/ (дата звернення: 05.04.2025).

54. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/ (дата звернення: 05.04.2025).

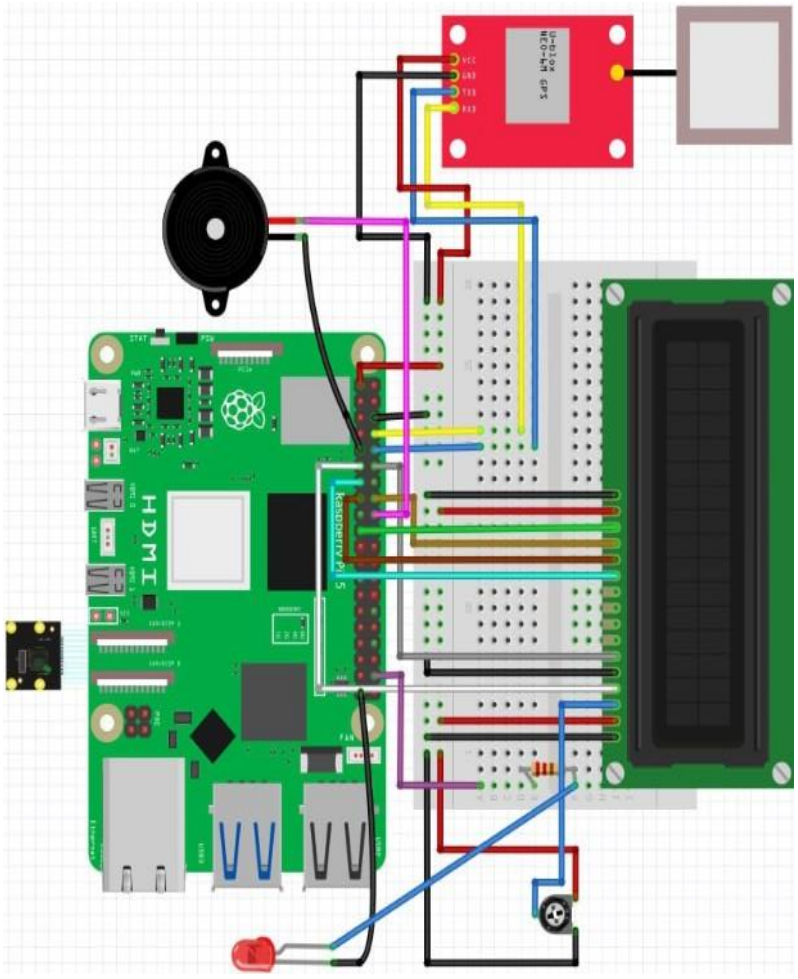
55. SQLite Documentation. URL: https://www.sqlite.org/docs.html (дата звернення: 05.04.2025).

56. Express.js Documentation. URL: https://expressjs.com/ (дата звернення: 05.04.2025).

Додаток А

(обов’язковий)

КОПІЯ КРЕСЛЕННЯ «АПАРАТНІ З’ЄДНАННЯ КОМПОНЕНТІВ»



Креслення апаратних з’єднань усіх компонентів

КвРКІ 210124.21.01.72Е8

Система моніторингу уваги людини з використанням аналізу відеопотоку в реальному часі

Апаратні з’єднання компонентів

Літера | Маса | Масштаб

Аркуш 1 | Аркушів 1

ХНУ,ГР.КІ2-21-1

Зм. Арк. | Номер докум. | Підпис | Дата

Розроб. Семень в.в. А.С.

Перевір. Хижа в.в.

Н.контр.

Т.контр. Кель т.м.

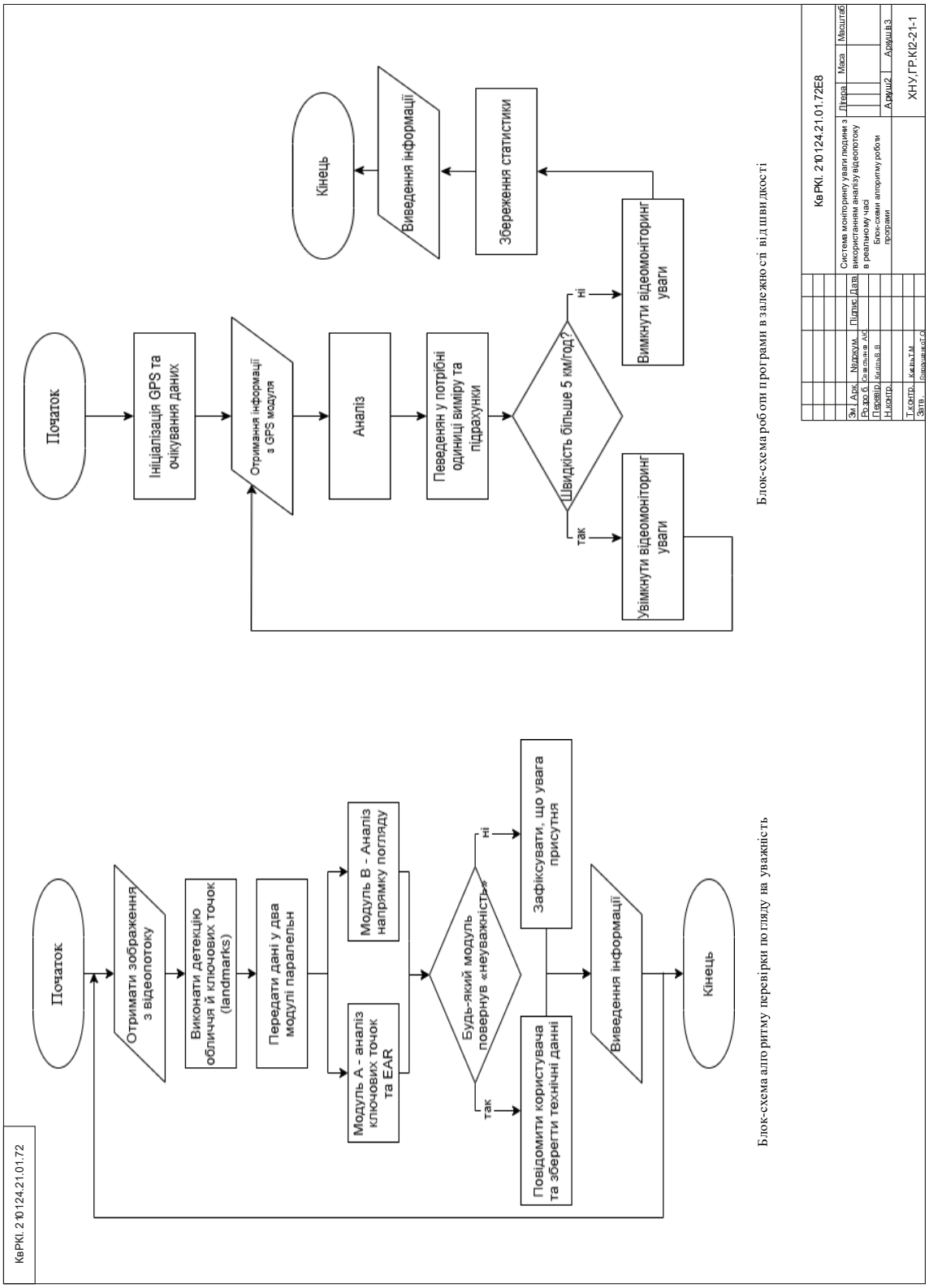
Затв. Пожоменко о.с.

КвРКІ 210124.21.01.72

Додаток Б

(обов’язковий)

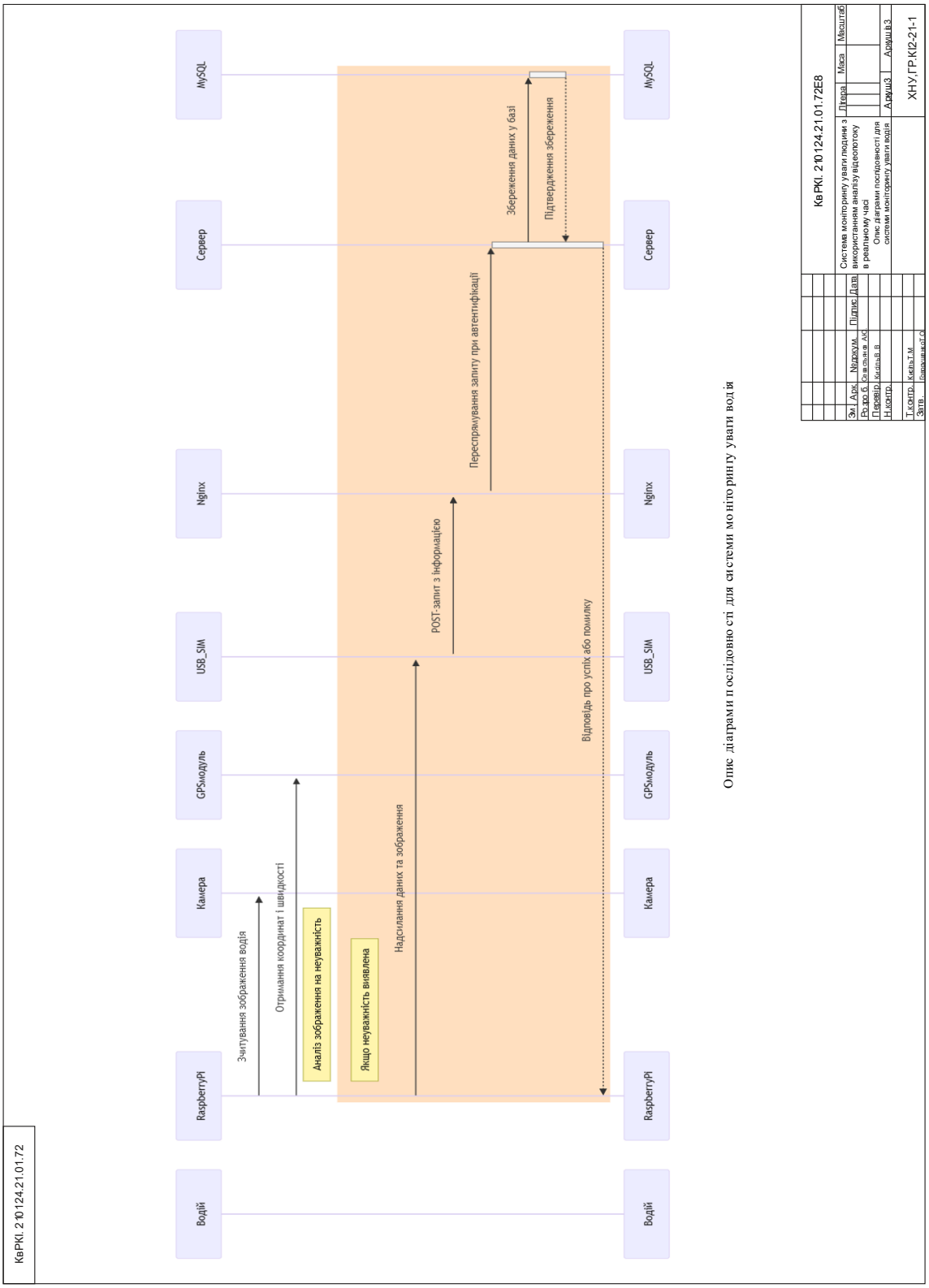
КОПІЯ КРЕСЛЕННЯ «БЛОК-СХЕМИ АЛГОРИТМУ РОБОТИ ПРОГРАМИ»



Додаток В

(обов’язковий)

КОПІЯ КРЕСЛЕННЯ «ДІАГРАМА ПОСЛІДОВНОСТІ ДЛЯ СИСТЕМИ МОНІТОРИНГУ УВАГИ ВОДІЯ»



Додаток Г

(обов'язковий)

ЛІСТИНГ ЧАСТИНИ ПРОГРАМНОГО КОДУ

```
    import logging
    import threading
    import time
    import sys

    # Attempt to import RPi.GPIO and Adafruit_CharLCD, setting
flags for availability
    try:
        import RPi.GPIO as GPIO
        GPIO.setmode(GPIO.BCM)
        GPIO_AVAILABLE = True
    except ImportError:
        GPIO_AVAILABLE = False
    except Exception:
        GPIO_AVAILABLE = False # Catch other potential RPi.GPIO
init errors

    try:
        import Adafruit_CharLCD as LCD
        LCD_AVAILABLE = True
    except ImportError:
        LCD_AVAILABLE = False



    class Logger:
        def __init__(self, log_file):
            logging.basicConfig(level=logging.INFO,
format='%(asctime)s - %(message)s',
```

```
handlers=[logging.FileHandler(log_file), logging.StreamHandler()])
            self.logger = logging.getLogger(__name__)

        def info(self, message):
            self.logger.info(message)

        def log_attention_event(self, event_type, details=None):
            message = f"ATTENTION EVENT: {event_type}"
            if details:
                message += f" - {details}"
            self.info(message)


    class HardwareController:
        def __init__(self, led_pin, buzzer_pin, lcd_pins):
            self.led_pin = led_pin
            self.buzzer_pin = buzzer_pin
            self.lcd_pins = lcd_pins

            self.led_blinking = False
            self.buzzer_buzzing = False
            self.lcd = None
            self.lcd_lock = threading.Lock()

            if GPIO_AVAILABLE:
                GPIO.setup(self.led_pin, GPIO.OUT)
                GPIO.output(self.led_pin, GPIO.LOW)
                GPIO.setup(self.buzzer_pin, GPIO.OUT)
                GPIO.output(self.buzzer_pin, GPIO.LOW)

            if LCD_AVAILABLE:
                try:
                    self.lcd = LCD.Adafruit_CharLCD(
```

```
                        rs=lcd_pins['rs'], en=lcd_pins['en'],
d4=lcd_pins['d4'],
                        d5=lcd_pins['d5'], d6=lcd_pins['d6'],
d7=lcd_pins['d7'],
                        cols=16, lines=2, backlight=2
                    )
                    self.lcd.clear()
                except Exception:
                    self.lcd = None # LCD initialization failed


        def toggle_led(self, state):
            if GPIO_AVAILABLE:
                GPIO.output(self.led_pin, GPIO.HIGH if state else
GPIO.LOW)

        def start_led_blink(self, interval=0.5):
            if not GPIO_AVAILABLE: return
            self.led_blinking = True
            threading.Thread(target=self._blink_loop,
args=(interval,), daemon=True).start()

        def stop_led_blink(self):
            self.led_blinking = False
            self.toggle_led(False)

        def _blink_loop(self, interval):
            while self.led_blinking:
                self.toggle_led(True)
                time.sleep(interval)
                self.toggle_led(False)
                time.sleep(interval)

        def toggle_buzzer(self, state):
            if GPIO_AVAILABLE:
```

```
                GPIO.output(self.buzzer_pin, GPIO.HIGH if state
else GPIO.LOW)

        def start_buzzer_beep(self, on_time=0.5, off_time=0.5):
            if not GPIO_AVAILABLE: return
            self.buzzer_buzzing = True
            threading.Thread(target=self._buzz_loop, args=(on_time,
off_time), daemon=True).start()

        def stop_buzzer_beep(self):
            self.buzzer_buzzing = False
            self.toggle_buzzer(False)

        def _buzz_loop(self, on_time, off_time):
            while self.buzzer_buzzing:
                self.toggle_buzzer(True)
                time.sleep(on_time)
                self.toggle_buzzer(False)
                time.sleep(off_time)

        def display_lcd(self, line1="", line2=""):
            if self.lcd:
                with self.lcd_lock:
                    self.lcd.clear()

self.lcd.message(f"{line1[:16].ljust(16)}\n{line2[:16].ljust(16)}")

        def cleanup_gpio(self):
            if GPIO_AVAILABLE:
               GPIO.cleanup()
class EyeGazeDetector: def __init__(self, ear_threshold=0.25):
self.ear_threshold = ear_threshold def calculate_ear(self,
eye_landmarks): A = dist.euclidean(eye_landmarks[1],
eye_landmarks[5]) B = dist.euclidean(eye_landmarks[2],
eye_landmarks[4]) C = dist.euclidean(eye_landmarks[0],
```

```
eye_landmarks[3]) return (A + B) / (2.0 * C) def is_eye_closed(self,
ear_value): return ear_value < self.ear_threshold def
extract_eye_landmarks(self, face_landmarks, eye_type="left"): start,
end = (36, 42) if eye_type == "left" else (42, 48) return
[(face_landmarks.part(i).x, face_landmarks.part(i).y) for i in
range(start, end)] class SensorManager: def __init__(self,
camera_idx=0, gps_port="/dev/ttyAMA0", gps_baudrate=9600): # Camera
self.cap = cv2.VideoCapture(camera_idx)
self.cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
self.cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480) self.camera_frame =
None self.camera_running = False self.camera_lock = threading.Lock()
# GPS self.gps_port, self.gps_baudrate = gps_port, gps_baudrate
self.ser = None self.gps_running = False self.gps_location =
{"latitude": None, "longitude": None, "speed": None} self.gps_lock =
threading.Lock() def start_camera(self): if not self.cap.isOpened():
return False self.camera_running = True
threading.Thread(target=self._capture_frames, daemon=True).start()
return True def _capture_frames(self): while self.camera_running:
ret, frame = self.cap.read() if ret: with self.camera_lock:
self.camera_frame = frame time.sleep(0.03) def
get_camera_frame(self): with self.camera_lock: return
self.camera_frame.copy() if self.camera_frame is not None else None
def stop_camera(self): self.camera_running = False if self.cap:
self.cap.release() def start_gps(self): try: self.ser =
serial.Serial(self.gps_port, baudrate=self.gps_baudrate,
timeout=0.5) self.gps_running = True
threading.Thread(target=self._read_gps_data, daemon=True).start()
return True except serial.SerialException: return False def
_read_gps_data(self): while self.gps_running: try: if
self.ser.in_waiting > 0: raw_data =
self.ser.readline().decode('ascii', errors='replace').strip()
self._parse_nmea_sentence(raw_data) except Exception: pass
time.sleep(0.1)
```

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: Андрій СЕВАСТЬЯНОВ

Співавтор:

Назва: Севастьянов_Система моніторингу уваги людини з використанням аналізу відеопотоку в реальному часі

Експерт:

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:3.5%

Коефіцієнт подібності 2:0.2%

Мікропробіли: 17

Заміна букв: 5

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-06-11 18:41:03.0

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 вимогам НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обґрунтування:

2025-06-11

Дата

Доцент Андрій Ніченорук

експерт

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 16.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 10%**

ID: 245096 Title: БКР Система моніторингу уваги людини з використанням аналізу відеопотоку в реальному часі Added in a DB: 2025-06-11 Authors: Андрій СЕВАСТЬЯНОВ Heads: Володимир КИСІЛЬ Consultants: Opponents:	Document		Sum coincidence on the DB	
	Symbols	Lexemes	Symbols	Lexemes
	89467	679	16376 (18%)	140 (21%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes
240732	Title: Звіт з ПДП Система моніторингу уваги людини з використанням аналізу відеопотоку в реальному час Added in a DB: 2025-05-01 Authors: А. Ю. Севастьянова Heads: Кисіль В.В. Consultants: Opponents:	14264 (16.0%)	109 (16.0%)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Севастьянов Андрій Юрійович

Тема: Система моніторингу уваги людини з використанням аналізу відеопотоку в реальному часі

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3___ Кількість сторінок записки ___70___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є розробка та дослідження ефективності системи моніторингу уваги людини на основі аналізу відеопотоку в реальному часі для своєчасного виявлення ознак зниження концентрації або відволікання

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 кваліфвкаційної роботи було проведено огляд проблематики пов'язаної з людським фактором на транспорті та доцільністю використання комп'ютерного зору для вирішення даної проблеми. Також було виконано постановку задачі дослідження. В другому розділі було спроектовано архітектуру що поєднує апаратну та програмну частини кіберфізичної системи та вибрано для неї Raspberry Pi так як це є дешевою і доступною платформою та присутнє в університеті на кафедрі. У розробці було вибрано мінімальні необхідні модулі — мініятюрний динамік, GPS модуль, камеру, дисплей. Також розробка є модульною що дозволяє обробляти різні данні в реальному часі. З програмної сторони було використано бібліотеки OpenCv, MediaPipe, а також натреновано моделі на базі нейронних мереж що дозволяє реагувати на напрямок погляду та на закриття очей. Результати спостережень зберігаються в локальній базі даних SQLite з можливістю подальшого завантаження на віддалений сервер. В третьому розділі було безпосередньо реалізовано готову

на віддалений сервер. В третьому розділі було безпосередньо реалізовано готову систему — окремо розгорнуто тестовий сервер на який передавались данні із реалізованої та встановленої системи та були відповідно зафіксовані інциденти.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: місцями опис недостатній — можна було б додати більшу деталізацію на скрінах чи схемах.

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 науково-технічному рівні.

8. Інші зауваження: _____

_____

_____

9. Оцінка дипломної роботи: Розглянувши негативні та позитивні сторони роботи вважаю що робота виконана на відмінному рівні та заслуговує оцінки "відмінно" 4.99 (A)

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

доцент кафедри ТПЗ Лисенко О. М.

_____

"___" _____2025 р. _____(підпис)

Завідувачу кафедри КІС
д-р. філософії, доц. Ользі ПАВЛОВІЙ

Андрія СЕВАСТЬЯНОВА
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-21-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

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

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

Сєв
22.04 2025 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

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

Назва: Система моніторингу уваги людини з використанням аналізу відеопотоку в реальному часі

Автор: Андрій СЕВАСТЬЯНОВ

Спеціальність: 123– Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Володимир КИСІЛЬ, ассистент

Після аналізу звіту подібності зроблено такий висновок:

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

Підтвердження:

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

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з 10-40 джерелами на один фрагмент речення;
4) зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту, до мікропробілів які помилково поставлені в деяких місцях в гомоглатичних кількостях.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ідентичності/схожості StrikePlagiarism, складає 3.45% і адресується до 34 першоджерела; та системою Anti-Plagiarism складає 16% з попереднім звітом з готовності роботи, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Володимир Кисіль

Гарант ОП _____ Андрій Ніченорук

Завідувач кафедри КІІС _____ Ольга ПАВЛОВА