Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: Комп'ютерна програма для визначення частоти серцебиття на основі аналізу зміни кольору обличчя

Виконав: студент IV курсу
групи 4ПІ–21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Присяжнюк В.В.
(прізвище та ініціали)

Керівник: к.т.н., професор каф. ПЗ Романюк О.Н.
(прізвище та ініціали)

Рецензент:. к.т.н., доц. каф ОТ Снігур А.В,
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ О.Н. Романюк
«09» 06 2025 р.

ВНТУ – 2025

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н._____
« 21 » березня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Присяжнюку Валентину Вадимовичу

1. Тема роботи – «Комп'ютерна програма для визначення частоти серцебиття на основі аналізу зміни кольору обличчя»

Керівник роботи: Романюк Олександр Никифорович, д.т.н., завідувач кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025р. №97

2. Строк подання студентом роботи 2 червня 2025 р.

3. Вихідні дані до роботи: програма з вимірюванням пульсу за зміною кольору обличчя; діапазон зміни пульсу від 30 до 180; середовище розробки Visual Studio Code; мова розробки Python; операційна система – Windows 10, тип відеокамери: від HD та вище.

4. Зміст розрахунково–пояснювальної записки: загальна характеристика роботи, аналіз даних та програмних засобів, розробка інтерфейсу та алгоритмів програмного застосунку, розробка модуля для розпізнавання обличчя, вимірювання пульсу, архівні сховища вимірюваних серцевих скорочень, обгрунтування вибору мови програмування та середовища розробки, блок–схеми та діаграми, висновки, список використаних джерел, додатки.

5. Перелік графічного матеріалу: демонстрація роботи аналогів, блок–схема функціонування програмного застосунку, структурна блок–схема додатку.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Романюк О.Н., д.т.н., професор кафедри ПЗ	24.03.25	01.06.25

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз методів і засобів для дистанційного вимірювання пульсу веб–камерою	25.03.25 – 07.04.25	вик
2	Розробка методів для дистанційного вимірювання частоти пульсу серцебиття	08.04.25 – 18.04.25	вик
3	Обґрунтування вибору засобів програмної реалізації	19.04.25 – 01.05.25	вик
4	Розробка програмних засобів для вимірювання серцебиття з використанням веб–камери	02.05.25 – 09.05.25	вик
5	Тестування програми	10.05.25 – 20.05.25	вик
6	Оформлення матеріалів до захисту БКР	21.05.25 – 30.05.25	вик

Студент _____ Присяжнюк В.В.
(підпис) (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи _____ Романюк О.Н.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У бакалаврській кваліфікаційній роботі розроблено комп’ютерну програму для дистанційного вимірювання частоти серцевих скорочень людини шляхом аналізу відеозображення обличчя з вебкамери. Розробка базується на принципі фотоплетизмографії, що дозволяє виявляти мікроскопічні зміни кольору шкіри, пов’язані з пульсацією кровотоку.

У ході дослідження було проаналізовано існуючі технології та програмні рішення, визначено їх недоліки та сформульовано завдання для створення ефективнішого застосунку. Запропоновано нову математичну модель аналізу зміни кольору шкіри з урахуванням умов освітлення та рухів, що забезпечило підвищення точності до 15%. Реалізовано гібридний підхід із використанням елементів машинного навчання, методів фільтрації та спектрального аналізу сигналів.

Програмна система створена мовою Python у середовищі Visual Studio Code, використовує бібліотеки OpenCV, NumPy, SciPy та scikit-learn. Вона забезпечує реальний час роботи, візуалізацію графіків пульсу, стійкість до змін освітлення та незначних рухів користувача. Результати тестування демонструють високу надійність роботи програми за різних умов.

Розроблене рішення має практичне значення для домашнього використання, телемедицини, спортивного моніторингу та попередньої діагностики стану серцево-судинної системи. Робота має наукову новизну, апробована на конференціях і підтверджує актуальність безконтактного моніторингу здоров’я.

Ключові слов**а**: пульс, вебкамера, фотоплетизмографія, аналіз кольору обличчя, комп’ютерне зір, Python, неінвазивна діагностика.

ABSTRACT

In the bachelor's qualification work, a computer program was developed for remote measurement of human heart rate by analyzing a video image of a face from a webcam. The development is based on the principle of photoplethysmography, which allows detecting microscopic changes in skin color associated with blood flow pulsation.

During the study, existing technologies and software solutions were analyzed, their shortcomings were identified, and tasks were formulated to create a more effective application. A new mathematical model for analyzing skin color changes was proposed, taking into account lighting conditions and movements, which provided an increase in accuracy of up to 15%. A hybrid approach was implemented using elements of machine learning, filtering methods, and spectral analysis of signals.

The software system was created in Python in the Visual Studio Code environment, using the OpenCV, NumPy, SciPy, and scikit-learn libraries. It provides real-time operation, visualization of pulse graphs, resistance to changes in education and minor user movements. The test results demonstrate high reliability of the program under various conditions.

The developed solution has practical significance for home use, telemedicine, sports monitoring and preliminary diagnostics of the cardiovascular system. The work is scientifically novel, tested at conferences and confirms the relevance of contactless health monitoring.

Keywords: pulse, webcam, photoplethysmography, facial color analysis, computer vision, Python, non-invasive diagnostics.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Сучасний етап розвитку цифрових технологій та підвищення обчислювальної потужності персональних комп'ютерів створюють нові можливості у сфері неінвазивної медичної діагностики. Дистанційне вимірювання життєво важливих показників людини, зокрема частоти серцевих скорочень, набуває особливої актуальності з огляду на глобальні тенденції до персоналізації медицини та віддаленого моніторингу здоров'я.

Технології вимірювання пульсу за допомогою веб–камери базуються на методі фотоплетизмографії, що дозволяє виявляти циклічні зміни кольору шкіри обличчя, які відповідають пульсації кровотоку. Такий підхід не потребує спеціалізованого обладнання і може бути реалізований на стандартних пристроях з вбудованою камерою, що робить його доступним для широкого загалу користувачів.

Останні дослідження у галузі комп'ютерного зору та обробки цифрових сигналів демонструють можливість досягнення високої точності вимірювань пульсу за допомогою стандартних веб–камер. Проте, існуючі рішення часто страждають від обмежень, пов'язаних з умовами освітлення, рухом об'єкта та іншими факторами, що впливають на якість вхідного сигналу.

У комп'ютерних системах вимірювання пульсу високу ефективність забезпечують методи та засоби, що дають змогу оптимально використовувати обчислювальні ресурси для обробки відеопотоку та аналізу змін кольору шкіри. Існуючі методи розпізнавання та аналізу пульсу характеризуються суттєвою обчислювальною складністю, що значною мірою впливає на швидкість та точність вимірювань.

Для забезпечення високої точності вимірювання пульсу використовують не тільки простий аналіз змін кольору, а й допоміжні методи фільтрації шумів, компенсації рухів та врахування умов освітлення, що дозволяє отримати

надійніші результати. Це суттєво ускладнює обчислювальний процес і вимагає розробки ефективних алгоритмів обробки сигналів.

Тому існуючі методи вимірювання пульсу за допомогою веб–камери потребують вдосконалення для підвищення точності та надійності в різних умовах використання. Тому актуальними є питання розробки нових методів і програмних засобів для аналізу змін кольору обличчя з метою вимірювання пульсу.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є автоматизація дистанційного визначення частоти серцебиття по зображенню обличчя людини.

Основними задачами дослідження є:

– провести аналіз існуючих методів і систем безконтактного вимірювання пульсу за допомогою веб–камери для визначення напрямків підвищення їх точності та надійності;

– запропонувати нові:

– методи розпізнавання обличчя та виділення зон інтересу для аналізу пульсового сигналу;

– методи обробки відеопотоку для виділення та аналізу змін кольору шкіри, пов'язаних з пульсацією кровотоку;

– методи фільтрації шумів та компенсації рухів для підвищення якості вхідного сигналу;

– розробити програмні модулі та систему вимірювання пульсу на основі запропонованих методів;

– провести експериментальні дослідження розроблених засобів вимірювання пульсу.

Об'єкт дослідження – процес розпізнавання та аналізу змін кольору шкіри обличчя для вимірювання пульсу за допомогою веб–камери.

Предмет дослідження – методи та засоби комп'ютерного зору й обробки сигналів для виділення та аналізу пульсового сигналу з відеопотоку.

Методи дослідження. В процесі досліджень використовувались: методи цифрової обробки сигналів, комп'ютерного зору, математичної статистики, спектрального аналізу, теорія розпізнавання образів та машинного навчання для розробки моделей та методів вимірювання пульсу; комп'ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна отриманих результатів:

1. Вперше запропоновано математичну модель аналізу змін кольору шкіри обличчя, яка враховує умови освітлення та рухи об'єкта для реалізації адаптивної фільтрації сигналу, що дозволило підвищити точність визначення частоти серцевих скорочень до 15%.

2. Запропоновано комбінований метод обробки відеосигналу з використанням алгоритмів незалежного компонентного аналізу та адаптивної фільтрації, що дозволяє підвищити точність вимірювання пульсу за рахунок ефективного виділення корисного сигналу з шумів.

Практичне значення отриманих результатів полягає в тому, що на основі отриманих в бакалаврській кваліфікаційної роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби вимірювання пульсу за допомогою веб–камери для систем неінвазивного моніторингу здоров'я. Розроблена комп'ютерна програма може бути використана у домашніх умовах, телемедицині, спортивних тренуваннях та для первинної діагностики стану серцево–судинної системи.

Особистий внесок здобувача. Усі наукові результати, викладені у БКР, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: комп’ютерне визначення частоти серцебиття шляхом аналізу кольорових змін обличчя[1], статистика серцево-судинних захворювань[2], особливості використання штучного інтелекту для визначення ритму серцебиття[3].

Апробація матеріалів бакалаврської кваліфікаційної роботи. Описані у роботі положення доповідались на конференціях: «XXV Всеукраїнської науково–технічної конференції молодих вчених (ВНТУ, 2025)»; «XXVII Міжнародної науково–технічної конференції «Технологія-2025» (СНУ, 2025)»; «XII Всеукраїнської науково–практичній конференції молодих учених «Інформаційні технології» (КСУ, 2025)».

Публікації. За темою бакалаврської кваліфікаційної роботи надруковано 3–х праць у матеріалах конференціях.

Структура та обсяг БКР. БКР складається зі вступу, чотирьох розділів, висновків, списку літератури, що містить 20 найменувань, 4 додатків. Робота містить 24 ілюстрації, 3 таблиць. Загальний обсяг роботи складає 103 сторінок, основний зміст викладено на 68 сторінках друкованого тексту.

1 АНАЛІЗ МЕТОДІВ І ЗАСОБІВ ДЛЯ ДИСТАНЦІЙНОГО ВИМІРЮВАННЯ ПУЛЬСУ ВЕБ–КАМЕРОЮ

1.1 Аналіз стану технологій вимірювання пульсу з використанням веб–камерою

У сучасних умовах стрімкого розвитку інформаційних технологій системи безконтактного вимірювання фізіологічних показників набувають особливого значення. Вимірювання пульсу за допомогою веб–камери – це інноваційна технологія, що дозволяє отримувати дані про частоту серцевих скорочень без використання спеціалізованих фізичних датчиків. Ця технологія належить до галузі комп'ютерного зору та є перспективним напрямом досліджень у сфері біометрії та телемедицини.

Головне завдання галузі в тому, щоб за допомогою програмних засобів дублювати здатності людського зору і виділення образів на зображенні. Виділення образів на зображенні можна сприймати, як обробку та аналіз інформації з даних зображення за допомогою моделі, побудованої з використанням геометрії, фізики, статистики та теорії навчання.

Технологія вимірювання пульсу за зміною кольору обличчя базується на принципі фотоплетизмографії, що дозволяє виявляти незначні зміни кольору шкіри, пов'язані з циркуляцією крові. Під час серцевого циклу змінюється об'єм крові в капілярах, що призводить до мікроскопічних змін відтінку шкіри, які непомітні людському оку, але можуть бути зафіксовані за допомогою сучасних камер та програмних алгоритмів.

Один із ключових викликів у розробці таких систем 0 – це фільтрація шумів, пов'язаних із рухом об'єкта, зміною освітлення та іншими факторами, що впливають на якість відеосигналу. Сучасні підходи до вирішення цієї проблеми включають застосування алгоритмів машинного навчання, зокрема згорткових нейронних мереж, та методів цифрової обробки сигналів для виділення корисного сигналу.

Область комп'ютерного зору може бути охарактеризована як відносно нова та динамічна. Хоча існують більш ранні роботи в цьому напрямі, інтенсивне вивчення проблеми почалось лише з кінця 1970–х років, коли комп'ютери змогли обробляти великі набори даних, зокрема зображення. Однак, ці дослідження зазвичай починались з інших галузей, і, відповідно, не існує стандартного формулювання проблеми комп'ютерного зору.

1.2 Аналіз методів відстежування кольорових змін обличчя для оцінки пульсу

Розробка програмних модулів комп'ютерної системи вимірювання пульсу за зміною кольору обличчя вимагає ретельного вибору технічних підходів, щоб забезпечити високу точність вимірювань та зручність використання. Існує кілька стратегій реалізації такої системи, кожна з яких має свої переваги та недоліки, що впливають на загальну ефективність та придатність розробки. У цьому розділі розглядаються найбільш ефективні методики створення програмних модулів для вимірювання пульсу за допомогою веб–камери.

Один із підходів передбачає використання методу Ейлерівської відеомагніфікації, який дозволяє візуально підсилювати непомітні для людського ока зміни кольору шкіри, пов'язані з пульсовою хвилею. Цей метод забезпечує візуалізацію мікроскопічних змін кольору, що полегшує їх подальший аналіз та виділення пульсової складової. Однак такий підхід може створити додаткове обчислювальне навантаження на систему, оскільки потребує складних математичних перетворень для кожного кадру відеопотоку, що може обмежувати можливість роботи в реальному часі на пристроях з обмеженими ресурсами.

Альтернативний підхід передбачає використання методів просторово–часового аналізу, які фокусуються на виявленні періодичних змін у колірних компонентах обраних областей інтересу на обличчі. Такий підхід є менш вимогливим до обчислювальних ресурсів і забезпечує достатню точність для більшості застосувань. Крім того, він дозволяє ефективно фільтрувати шуми,

пов'язані з рухом об'єкта або зміною освітлення, що підвищує надійність вимірювань у реальних умовах.

Використання методів машинного навчання, зокрема згорткових нейронних мереж, для аналізу відеопотоку та виділення пульсової складової є ще одним перспективним підходом. Цей метод дозволяє системі самостійно визначати найінформативніші області та характеристики зображення для вимірювання пульсу, що значно покращує адаптивність до різних умов зйомки та індивідуальних особливостей користувачів. Проте, такий підхід потребує значного обсягу даних для навчання нейронної мережі та оптимізації її параметрів, що ускладнює початкове впровадження системи [4].

Після аналізу переваг та недоліків кожного з методів було вирішено обрати гібридний підхід, що поєднує просторово–часовий аналіз з елементами машинного навчання для оптимізації процесу виділення та аналізу пульсового сигналу. Такий підхід дозволить створити збалансоване рішення, що забезпечує високу точність вимірювань при помірних вимогах до обчислювальних ресурсів, а також здатне адаптуватися до різних умов використання.

Для реалізації обраного підходу було вирішено використовувати мову програмування Python, яка надає широкі можливості для обробки та аналізу відеоданих завдяки наявності спеціалізованих бібліотек, таких як OpenCV для комп'ютерного зору, NumPy для математичних обчислень, SciPy для обробки сигналів та scikit–learn для застосування методів машинного навчання. Це забезпечить гнучкість у розробці та дозволить ефективно реалізувати всі компоненти системи.

1.3 Аналіз аналогів

Зі стрімким розвитком технологій комп'ютерного зору сфера безконтактного вимірювання фізіологічних показників зазнає суттєвих змін, особливо у контексті автоматизації моніторингу пульсу за допомогою веб–камери.

Завдяки системам безконтактного вимірювання пульсу користувачі можуть регулярно контролювати свій серцевий ритм без спеціалізованого медичного

обладнання, отримувати дані про стан здоров'я та відстежувати динаміку змін у часі. У цьому контексті з'являється все більше програмних рішень та мобільних застосунків, які дозволяють ефективно вимірювати пульс через веб–камеру. Такі програми надають користувачам зручні функції, як–от відстеження варіабельності серцевого ритму, візуалізацію результатів у вигляді графіків, можливість експорту даних та багато інших сервісів, що роблять моніторинг здоров'я комфортнішим та інформативнішим.

Сучасні алгоритми машинного навчання та комп'ютерного зору значно підвищили точність таких вимірювань. Використання методів глибокого навчання для фільтрації шумів, компенсації руху користувача та оптимізації обробки відеосигналу дозволяє досягти результатів, порівнянних з традиційними медичними приладами. Особливо ефективними виявляються нейронні мережі, здатні адаптуватися до різних умов освітлення, типів шкіри та індивідуальних особливостей користувачів.

Практичне застосування таких систем простягається далеко за межі домашнього використання. У телемедицині ці технології відкривають нові можливості для дистанційного моніторингу пацієнтів, особливо в умовах обмеженого доступу до медичних закладів. Фітнес-індустрія також активно впроваджує безконтактні рішення для контролю навантаження під час тренувань, а в корпоративному секторі з'являються програми моніторингу стресу співробітників та профілактики професійного вигорання.

Серед численних аналогів розроблюваної системи варто виділити декілька популярних рішень: Cardiio, FaceBeat, Pulse Rate Monitor, RPPG Health, кожен з яких пропонує свої унікальні можливості та підходи до вимірювання пульсу через камеру.

На рисунку 1.1 зображено Cardiio [5], який є мобільним застосунком для безконтактного вимірювання пульсу, що використовує фронтальну камеру смартфона для аналізу відбиття світла від обличчя користувача. Він робить акцент на простоті використання та забезпечує точність вимірювань близьку до медичних пульсоксиметрів у оптимальних умовах освітлення.

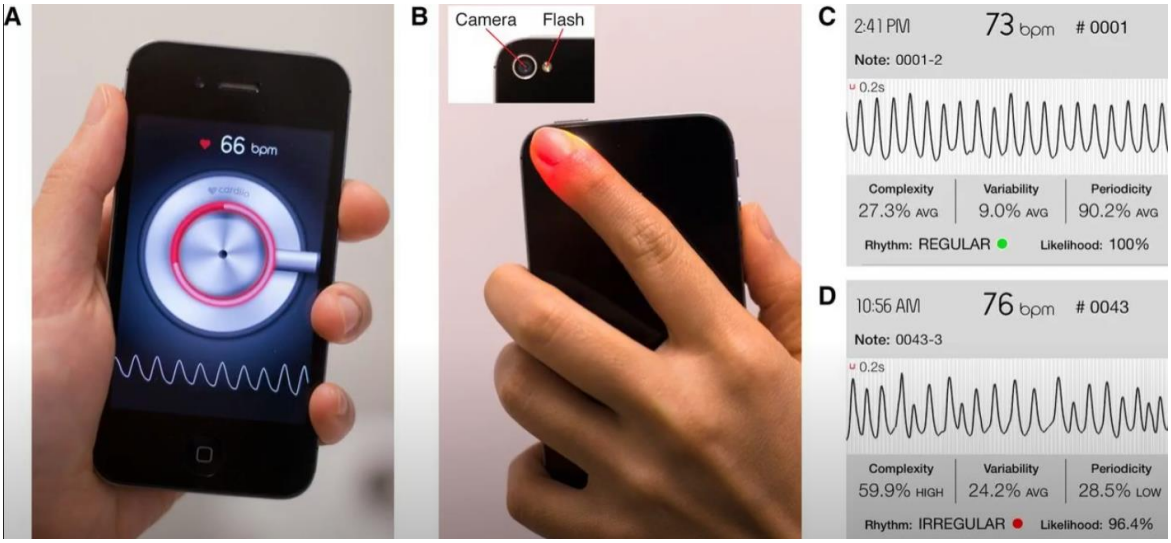


Рисунок 1.1 — Приклад роботи застосунку Cardiio

На рисунку 1.2 зображено FaceBeat [6], який є програмним рішенням для ПК, що використовує алгоритми машинного навчання для аналізу відеопотоку з веб–камери. Застосунок здійснює детекцію обличчя та визначає області інтересу для аналізу пульсової хвилі. Перевагами FaceBeat є можливість довготривалого моніторингу та експорт даних для подальшого аналізу.

Рисунок 1.2 — Інтерфейс програми FaceBeatНа

На рисунку 1.3 зображено Pulse Rate Monitor [7], який є крос–платформним рішенням, що підтримує як мобільні пристрої, так і ПК. Основна особливість цього застосунку полягає у використанні спектрального аналізу для виділення пульсового сигналу з відеопотоку. Застосунок також пропонує функції аналізу варіабельності серцевого ритму для оцінки стресу та загального стану здоров'я.

Рисунок 1.3 — Приклад вимірювання пульсу у Pulse Rate Monitor

На рисунку 1.4 зображено RPPG Health [8], який є веб–сервісом, що дозволяє користувачам вимірювати пульс через веб–камеру без необхідності встановлення додаткового програмного забезпечення. Він забезпечує проведення вимірювань у реальному часі, збереження результатів в особистому кабінеті та візуалізацію даних у вигляді графіків та діаграм, що сприяє кращому розумінню динаміки серцевого ритму

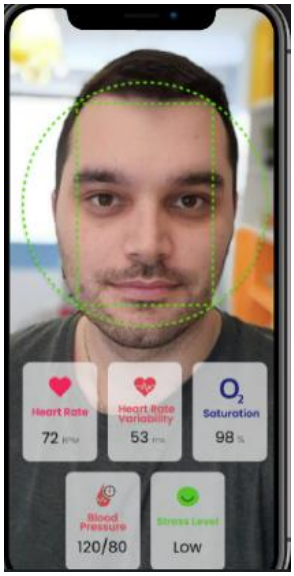


Рисунок 1.4 — Приклад інтерфейсу сервісу RPPG Health

На рисунку 1.5 зображено FaceHR [9], який є дослідницькою платформою для вимірювання пульсу через аналіз відеопотоку, що орієнтована на наукові та медичні дослідження. Система надає можливість детального налаштування параметрів алгоритмів обробки сигналу, вибору різних методів фільтрації та аналізу, а також дозволяє експортувати необроблені дані для подальшої обробки в спеціалізованому програмному забезпеченні.

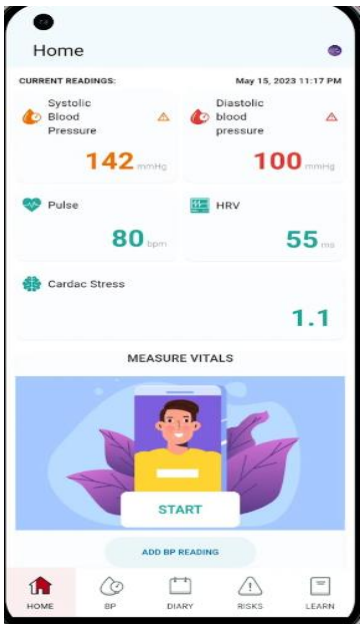


Рисунок 1.5 — Приклад використання програми FaceHR

Таблиця 1.1 – Порівняльні характеристики систем вимірювання пульсу веб–камерою

	Cardiiot	FaceBeatt	Pulse Rate Monitor	RPPG Health	Власна розробка
Точність вимірювання пульсу	1	1	1	1	1
Можливість роботи при різному освітленні	0	1	1	1	1
Стійкість до рухів користувачів	0	0	1	0	1
Аналіз варіабельності серцевого ритму	1	1	1	0	1
Візуалізація даних у реальному часі	0	0	0	0	1
Сумарний коефіцієнт	2	3	4	2	5

Відповідно до наведеної таблиці порівняльного аналізу, розробка власної системи для вимірювання пульсу за зміною кольору обличчя з використанням мови програмування Python є виправданою та має значні переваги порівняно з деякими існуючими рішеннями на ринку, такими як Cardiio, FaceBeat та RPPG Health. Запропонований продукт включатиме всі ключові функції, що пропонуються конкурентами, а також запровадить унікальні можливості, які роблять його більш зручним та функціональним для користувачів.

Основною перевагою власної розробки є комплексність та універсальність. На відміну від конкурентів, наша система забезпечує не лише точне вимірювання пульсу, а й підтримує функціонал для аналізу варіабельності серцевого ритму, що

є важливим показником для оцінки загального стану серцево–судинної системи. Крім того, розроблювана система демонструє підвищену стійкість до рухів користувача, що є значною перевагою для реальних умов використання, де абсолютна нерухомість не завжди можлива.

Унікальною особливістю є візуалізація даних у реальному часі, що дозволяє користувачам наочно спостерігати за зміною пульсової хвилі та краще розуміти її характеристики. Це може бути особливо корисним у навчальних та дослідницьких цілях, а також для підвищення залученості користувачів до процесу моніторингу здоров'я.

Таким чином, власна розробка системи вимірювання пульсу за зміною кольору обличчя демонструє найвищий сумарний коефіцієнт серед порівнюваних рішень, що підтверджує її конкурентоспроможність та актуальність у сучасному середовищі телемедичних сервісів. Така система не лише забезпечить високий рівень задоволеності користувачів завдяки зручності та ефективності моніторингу пульсу, але й сприятиме покращенню користувацького досвіду завдяки впровадженню нових функцій, таких як стійкість до рухів та візуалізація даних у реальному часі.

Проаналізувавши дані з таблиці 1.1, можна зробити висновок, що власна розробка демонструє кращі результати за розглянутими критеріями порівняно з аналогами Cardiio та RPPG Health на 60% (100% – 2/5 × 100% = 60%), у порівнянні з FaceBeat — на 40% (100% – 3/5 × 100% = 40%). Таким чином, розробка та впровадження власної системи для вимірювання пульсу веб–камерою має всі шанси на успіх, пропонуючи ринку унікальне рішення, яке не лише задовольнить потреби сучасних користувачів у доступному та зручному моніторингу здоров'я, але й покращить точність та надійність вимірювань у різних умовах використання.

1.4 Постановка задачі дослідження

На основі проведеного аналізу існуючих рішень для безконтактного вимірювання пульсу з використанням веб–камери були виявлені певні

обмеження та можливості для вдосконалення. Для створення ефективної комп'ютерної системи вимірювання пульсу за зміною кольору обличчя необхідно виконати наступні дослідницькі завдання:

– розробити надійний компонент для відстеження обличчя користувача в режимі реального часу за допомогою алгоритмів комп'ютерного зору, що працюватиме в різних умовах освітлення та при наявності незначних рухів.

– розробити метод для аналізу мікроскопічних змін кольору шкіри з відеопотоку, використовуючи зелений канал RGB як найбільш інформативний для виявлення пульсової хвилі.

– розробити технології цифрової обробки сигналів для нормалізації даних, усунення шумів та артефактів, підвищення чіткості корисного сигналу за допомогою швидкого перетворення .

– розробити модуль графічного інтерфейсу користувача для відображення відеопотоку з накладанням кольорової індикації пульсу, візуалізацією графіків сигналу в часовій та частотній областях, трендів пульсу протягом сеансу.

– розробити функціонал для запису результатів вимірювань у CSV–файли, експорту графіків та автоматичного аналізу тенденцій пульсу, що дозволить проводити довгострокове спостереження за змінами стану серцево–судинної системи.

– адаптувати алгоритми для швидкої роботи на пристроях з обмеженими ресурсами, щоб забезпечити плавну роботу системи в реальному часі з мінімальними затримками та високою частотою кадрів.

Кінцевою метою дослідження є створення програмного забезпечення, що дозволить точно та надійно вимірювати частоту серцевих скорочень за допомогою звичайної веб–камери, забезпечуючи інтуїтивно зрозумілу візуалізацію та можливість тривалого моніторингу. Це розширить доступність технологій для самостійного контролю стану здоров'я і створить основу для

подальших досліджень у галузі безконтактної діагностики фізіологічних параметрів.

1.5 Висновки

У першому розділі було проаналізовано розвитку програм для вимірювання пульсу за допомогою веб–камери та обґрунтовано актуальність розробки програмного застосунку для вимірювання пульсу за зміною кольору обличчя. Були розглянуті програмні засоби такі як Cardiiot, FaceBeat, Pulse Rate Monitor, RPPG Health. Кожен з цих засобів має свої переваги і недоліки, що були проаналізовані.

Проаналізувавши переваги та недоліки існуючих програмних засобів, було сформульовано завдання для подальшої розробки власного програмного застосунку для вимірювання пульсу. На основі отриманої інформації було сформовано перелік задач, які необхідно виконати для розробки власних програмних засобів.

2 РОЗРОБКА МЕТОДІВ ДЛЯ ДИСТАНЦІЙНОГО ВИМІРЮВАННЯ ЧАСТОТИ ПУЛЬСУ

2.1 Розробка методу аналізу зміни кольору обличчя для визначення пульсу

Розробка системи дистанційного визначення пульсу на основі аналізу кольору обличчя передбачає використання методів комп’ютерного зору та цифрової обробки сигналів для безконтактного моніторингу серцевого ритму людини. Така система є перспективною альтернативою традиційним методам вимірювання пульсу, оскільки не потребує фізичного контакту з тілом і може використовуватись у телемедицині, системах безпеки, спортивному моніторингу та в повсякденному житті.

Однією з ключових рекомендацій при розробці таких систем є забезпечення високої якості відеозапису. Важливо використовувати камери з високою роздільною здатністю та частотою кадрів не менше 30 кадрів на секунду для збереження точності аналізу зміни кольору шкіри, яка є надзвичайно незначною (див. рис. 2.1).

Рисунок 2.1 – Вигляд відеокамери з HD роздільною здатністю та 30 кадрів в секунду

Оптимальне освітлення відіграє також вирішальну роль: слід уникати надмірних тіней, змін освітлення та мерехтіння світла. Рекомендовано використовувати стабільне, розсіяне природне або штучне світло (див. рис. 2.2).

Рисунок 2.2 – Вигляд оптимального освітлення робочого місця

Для визначення пульсу з очищеного сигналу використовують частотний аналіз, зокрема швидке перетворення Фур’є або вейвлет-перетворення, що дозволяють виявити домінантну частоту, яка відповідає частоті серцебиття (див. рис. 2.3).

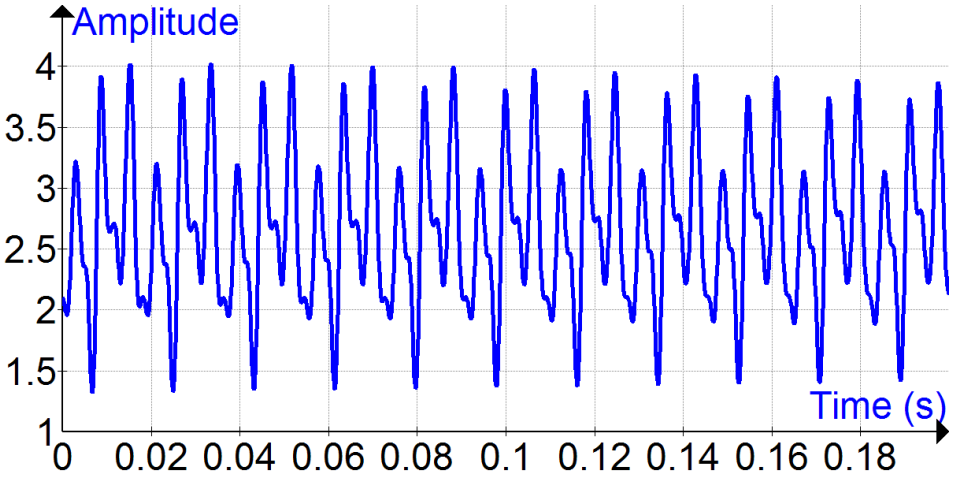


Рисунок 2.3 – Графік аналізу Фур’є

Основна ідея дистанційного визначення пульсу полягає у виявленні мікроскопічних змін кольору шкіри обличчя, зумовлених пульсацією крові. Ці зміни, хоч і малопомітні для людського ока, можуть бути зафіксовані звичайною

камерою та проаналізовані за допомогою спеціалізованих алгоритмів. Для ефективної реалізації такої системи слід враховувати низку важливих рекомендацій.

Необхідно забезпечити якісне відеозахоплення. Камера повинна мати достатню роздільну здатність та частоту кадрів, яка не менше 30 кадрів за хвилину, що дозволяє точно фіксувати дрібні зміни кольору в області шкіри. Освітлення сцени має бути рівномірним, без надмірних тіней чи мерехтіння, оскільки нестабільне освітлення може істотно знизити точність вимірювань. Бажано уникати джерел світла з низькою частотою мерехтіння, наприклад, деяких люмінесцентних ламп.

Далі важливим етапом є детекція обличчя та виділення зони інтересу, або ROI – Region of Interest, зазвичай в області чола або щік, де шкіра менш піддається рухам та має відносно стабільний колір. Для цього використовуються алгоритми машинного зору, зокрема методи на основі глибокого навчання, такі як Haar-каскади, MTCNN або MediaPipe Face Detection (див. рис. 2.4).

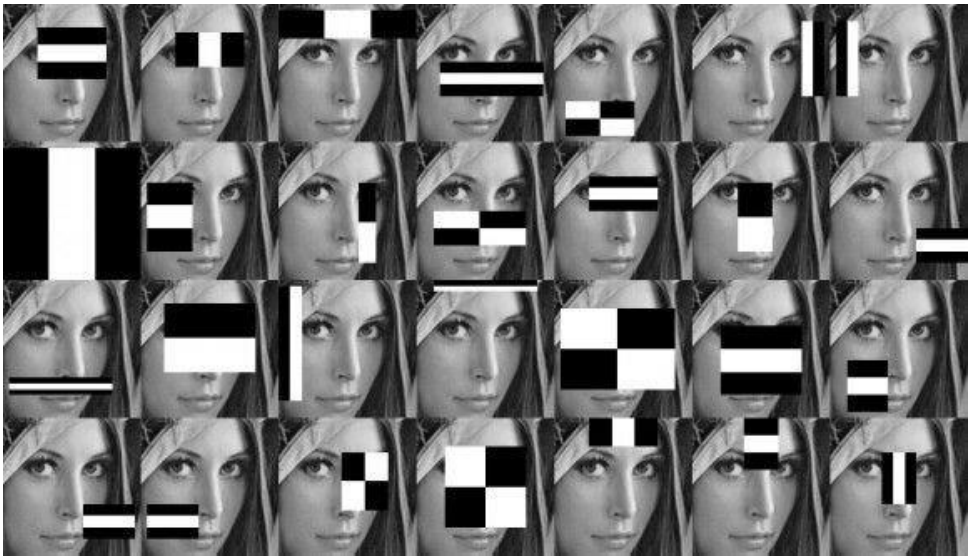


Рисунок 2.4 – Робота виявлення обличчя з використанням Haar Cascades

Після виділення ROI необхідно здійснити аналіз зміни кольорових компонентів у часі (див. рис. 2.5). Найчастіше використовується простір RGB, хоча у деяких випадках кращі результати дає перетворення в інші кольорові

простори, наприклад, HSV чи YCbCr. На основі часових рядів кольорових сигналів застосовуються фільтрація шумів, та спектральний аналіз, для виділення домінантної частоти, що відповідає серцевому ритму.

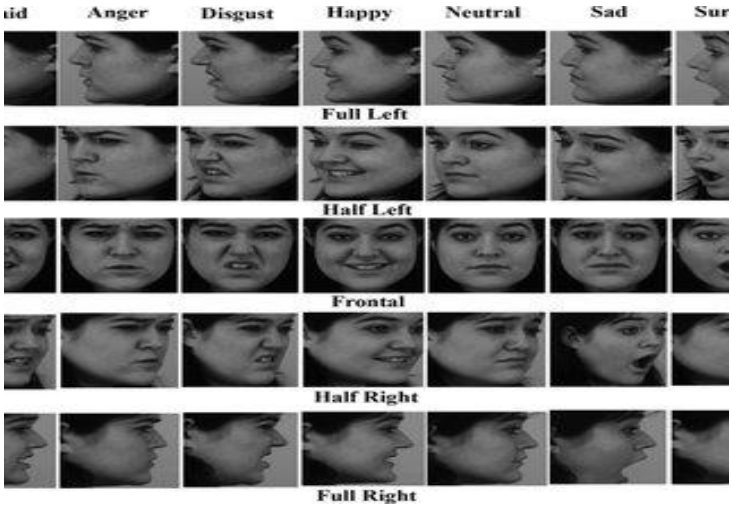


Рисунок 2.5 – Робота Region of Interest на обличчі

Отже, варто враховувати вплив артефактів – рухів голови, міміки, змін освітлення та інших факторів. Для їх компенсації використовуються методи стабілізації зображення, алгоритми відстеження ROI у часі, а також адаптивні фільтри. Для підвищення точності рекомендується застосовувати комбінацію кількох зон обличчя та усереднення результатів.

2.1.1 Рекомендація по вибору відеокамер для визначення пульсу

Вибір відеокамери є одним із ключових аспектів при розробці систем дистанційного визначення пульсу на основі аналізу кольору обличчя. Від якості відеозображення, стабільності кольору, частоти кадрів та рівня шумів безпосередньо залежить точність і надійність вимірювання пульсу. важливо забезпечити високу частоту кадрів. Рекомендується використовувати камери з частотою не менше 30 кадрів на секунду, а оптимально – 60 кадрів і вище, оскільки серцевий ритм людини зазвичай перебуває в межах від 1 до 2 Гц, або 60–120 ударів на хвилину, і лише при достатній частоті кадрів можливо коректно фіксувати дрібні зміни кольору шкіри, зумовлені кровоплином.

Роздільна здатність відеокамери також має велике значення. Висока чіткість дозволяє точніше виділяти область інтересу на обличчі – зазвичай у ділянці чола або щік, – та забезпечує більшу кількість корисних пікселів для аналізу. Мінімально рекомендованою є роздільна здатність HD, проте перевагу слід надавати Full HD або навіть 4K у разі наявності відповідних обчислювальних ресурсів. Особливу увагу слід приділяти точності кольоропередачі – камера повинна стабільно відтворювати кольори, особливо у зеленому каналі, оскільки саме він найбільш чутливий до змін, викликаних пульсацією крові. Камери з підтримкою sRGB або AdobeRGB, з мінімальною компресією кольору, забезпечують вищу точність вимірювань (див. рис. 2.6).

Рисунок 2.6 – Вигляд камери з роздільною здатністю Full HD

Дуже важливою є наявність ручного керування основними параметрами – експозицією, чутливістю, балансом білого. Автоматичні режими часто викликають коливання яскравості або кольору, що негативно впливає на якість сигналу пульсу. Камера має також стабільно працювати при штучному освітленні. Зокрема, варто перевірити, чи має вона функцію компенсації мерехтіння, яка дозволяє уникнути небажаних перешкод при зйомці під світильниками з частотою 50/60 Гц.

Ще одним важливим критерієм є інтерфейс підключення камери. Для режимів реального часу бажано використовувати інтерфейси з високою пропускною здатністю і малою затримкою – такі як USB 3.0, HDMI або Ethernet.

Це дозволяє обробляти відео без затримок і забезпечує більш стабільну роботу системи.

Тип камери також відіграє важливу роль. Веб-камери, наприклад Logitech C920 або Brio 4K, є зручними для створення прототипів і підтримують HD або навіть 4K якість, але не завжди дозволяють повністю вручну керувати параметрами зйомки (див. рис. 2.7).

Рисунок 2.7 – Вигляд веб–камери Logitech

Промислові камери, такі як Basler, FLIR або IDS, забезпечують точне налаштування, високу стабільність сигналу, можливість захоплення відео без компресії та краще підходять для наукових і медичних застосувань (див. рис. 2.8).

Рисунок 2.8 – Вигляд промислової камери Basler

Камери класу DSLR або бездзеркальні з можливістю виводу відео через HDMI забезпечують відмінну якість зображення, але потребують додаткових пристроїв захоплення (див. рис. 2.9).

Рисунок 2.9 – Вигляд бездзеркальних камер Canon

У спеціалізованих системах можуть використовуватися також інфрачервоні або комбіновані RGB+IR камери, які дозволяють покращити точність вимірювання в умовах низького освітлення або з високим рівнем шумів (див. рис. 2.10)

Рисунок 2.10 – Вигляд комбінованої RGB+IR камери

Додатково рекомендовано враховувати рівень шуму камери, особливо при слабкому освітленні, а також її можливість фіксації на стабільній платформі, щоб уникнути тремтіння зображення. Важливою є також сумісність камери з платформами обробки відео, такими як OpenCV, MATLAB або Python, для зручної інтеграції в програмну частину системи.

Отже, вибір камери має базуватись на комплексному врахуванні технічних характеристик – частоти кадрів, роздільної здатності, кольоропередачі, рівня шуму, підтримки ручних налаштувань, стабільності при штучному освітленні, типу інтерфейсу, типу камери та її сумісності з оброблювальним програмним забезпеченням. Раціональний вибір обладнання значною мірою визначає точність, стійкість та ефективність роботи всієї системи дистанційного визначення пульсу.

2.1.2 Рекомендація по освітленню обличчя для точного визначення серце

Освітлення обличчя є критично важливим фактором при розробці систем дистанційного визначення пульсу на основі аналізу кольору шкіри. Якість освітлення безпосередньо впливає на точність виявлення слабких змін кольору, спричинених пульсацією крові в капілярах. Нерівномірне, нестабільне або недостатнє освітлення може призвести до спотворення сигналу, додавання шумів, помилкових вимірювань або взагалі унеможливлення аналізу. Тому до організації освітлення слід підходити з особливою увагою, дотримуючись низки ключових рекомендацій.

Освітлення повинно бути рівномірним і м’яким, без різких тіней, відблисків або пересвітів. Ідеальним варіантом є дифузне освітлення, що досягається за допомогою світлорозсіювачів, м’яких світильників або розсіяного денного світла. Точкове світло або спрямовані джерела світла слід уникати, оскільки вони створюють нерівномірні зони яскравості, які можуть ускладнити алгоритмам виявлення кольорових змін (див. рис. 2.11).

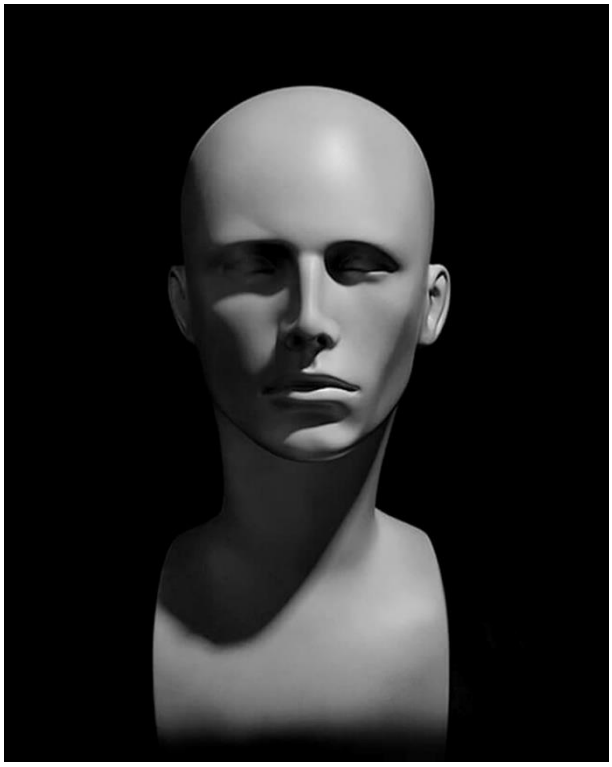


Рисунок 2.11 – Приклад оптимального освітлення обличчя.

Важливу роль відіграє стабільність освітлення у часі. Освітлення не повинно мерехтіти або змінювати яскравість під час зйомки. Багато штучних джерел світла, зокрема дешеві світлодіодні лампи чи люмінесцентні трубки, мають ефект мерехтіння, особливо при низькій частоті електромережі, 50 або 60 Гц. Ці мерехтіння, хоч і непомітне для ока, може бути зафіксоване камерою і призводить до помилкових коливань у відео, які система може сприйняти як пульсацію. В таких випадках, потрібно використовувати джерела світла з технологією flicker-free, або перевірені професійні лампи, які забезпечують стабільний світловий потік.

Головним фактором для якісного параметром є колірна температура освітлення. Найбільш придатною для аналізу шкіри є колірна температура у межах 4500–5500 К, що відповідає природному денному освітленню (див. рис. 2.12).

Рисунок 2.12 – Приклад колірних температур від 1K до 10K

Світло має бути нейтрально-білим, без виражених жовтих або синіх відтінків, оскільки ці кольори можуть викривлювати реальні зміни у шкірі, які залежать від кровообігу. Бажано, щоб колірна температура залишалась сталою протягом усього періоду відеозапису.

Рисунок 2.13 – Приклад використання колірного освітлення при вимірюванні пульсу

Уникнення поєднання декількох джерел світла з різними колірними температурами, оскільки це створює неоднорідність колірного профілю в кадрі. За можливості, варто застосовувати одне основне джерело світла з однорідною характеристикою або декілька однакових джерел, розташованих симетрично.

Рівень освітленості повинен бути достатнім для уникнення шумів при зйомці. Камери знімають з найменшим рівнем шуму при середньому або високому рівні освітлення, тоді як при слабкому освітленні автоматично підвищується чутливість сенсора, що призводить до зростання цифрового шуму. Тому оптимальним є освітлення, яке дозволяє камері працювати на низькому чутливому сенсору з короткою витримкою.

Не меншим важливим фактором є розміщення джерелу світла. Найкращим, є те, що коли основне джерело світла розташоване фронтально, трохи вище рівня очей людини, або під кутом 45 градусів до обличчя. Це дозволяє уникнути небажаних тіней і одночасно надає обличчю об’єм. Використання допоміжних заповнюючих джерел з боків допомагає ще більше пом’якшити тіні (див. рис. 2.14).

Рисунок 2.14 – Приклад правильного розміщення світлу

Отже, для забезпечення надійного освітлення при дистанційному вимірюванні пульсу необхідно застосовувати рівномірне, стабільне та природне

за спектром світло з колірною температурою близькою до денного, уникати мерехтіння, забезпечити сталу інтенсивність освітлення протягом всього періоду аналізу та уникати сильних тіней і відблисків. Лише при дотриманні цих умов система зможе точно виявляти слабкі зміни кольору шкіри та надійно визначати пульс користувача.

2.1.3 Рекомендація по організації робочого місця

Організація робочого місця для системи дистанційного визначення пульсу на основі аналізу кольору обличчя відіграє вирішальну роль у забезпеченні стабільності, надійності та точності вимірювань. Від умов навколишнього середовища, правильного розміщення обладнання, освітлення, фону та мінімізації зовнішніх впливів залежить ефективність функціонування всієї системи. Основною метою при організації робочого простору є створення контрольованого, стабільного і відтворюваного середовища для збору відеоданих високої якості.

Перед наступним кроком, потрібно вибрати тихе, слабо динамічне приміщення, де відсутній потік людей, різкі зміни освітлення або шум, що можуть викликати мікрорухи особи або впливати на обчислення. Робоче місце повинне бути ізольованим від зовнішніх джерел змінного освітлення, таких як вікна з прямим сонячним світлом або проходи з рухомими об'єктами, які можуть створювати тіні та змінювати рівень освітлення під час зйомки. Якщо вікна присутні, бажано застосовувати щільні штори або жалюзі для регулювання природного світла та запобігання його флуктуаціям.

Робоча поверхня, на якій встановлена камера, повинна бути стабільною, нерухомою, без вібрацій чи коливань. Камеру слід жорстко закріпити на штативі або монтажній платформі з можливістю точного регулювання висоти та кута нахилу. Висота камери повинна відповідати рівню очей або дещо вище, щоб забезпечити прямий або злегка нахилений кут огляду на обличчя користувача. Це сприяє кращому виявленню області інтересу на лобі та щоках, де зміни кольору найпомітніші.

Окрім камери, необхідно правильно розташувати джерела освітлення. Освітлювальні прилади повинні бути розміщені симетрично або фронтально щодо обличчя, бажано трохи вище рівня голови, щоб уникати тіней на обличчі. Освітлення має бути рівномірним та м’яким, без різких контрастів. Важливо, щоб джерела світла не потрапляли в об’єктив камери, оскільки це може спричинити засвічування та втрату контрасту.

Фон, на якому розміщене обличчя, також має значення. Рекомендується використовувати однотонний, нейтральний фон, наприклад, сірого або світло-блакитного кольору, який не відволікає систему візуального аналізу та не створює додаткового навантаження при обробці зображень. Слід уникати фону з високим контрастом, строкатими елементами або рухомими об’єктами.

Щодо позиціонування користувача, слід забезпечити фіксацію голови в стабільному положенні. У лабораторних умовах це може бути спеціальна підставка для підборіддя, а в побутових – рекомендації користувачу не рухати головою під час запису. Відстань від користувача до камери має бути такою, щоб обличчя займало достатню частину кадру, при цьому не створюючи надмірного наближення, що може спричинити спотворення.

Робоче місце також має бути вільним від зайвих джерел електромагнітних перешкод і шумів, які можуть впливати на електроніку камери або комп’ютера. Обчислювальна техніка має бути розміщена поряд, щоб забезпечити стабільний відеопотік без втрат кадрів. Рекомендується використовувати дротові підключення, щоб уникати бездротових з'єднань, які можуть мати затримки або перебої.

Для користувача бажано передбачити зручне сидіння, яке не обмежує рухів, але водночас дозволяє сидіти нерухомо протягом кількох хвилин. Відстань до монітора, якщо використовується зворотний зв'язок або інтерфейс, має бути комфортною. Варто також уникати джерел вібрацій поруч із камерою.

Отже, ефективна організація робочого місця для системи дистанційного визначення пульсу повинна забезпечувати стабільність положення камери та користувача, контрольоване і однорідне освітлення, відсутність змін

навколишнього середовища, належний фон та відсутність перешкод, які можуть спотворити відеосигнал. Створення таких умов є запорукою точної та стабільної роботи системи в реальному часі.

2.2 Розробка математичної моделі для методу аналізу зміни кольору обличчя для підвищення точності визначення частоти серцебиття

Для вимірювання пульсу по змінах кольору обличчя використовують відеоплетизмографію. Це дає можливість зафіксувати незначні коливання інтенсивності кольору шкіри, що пов'язані з пульсацією крові. Для цього використовують звичайну веб–камеру, яка робить множину кадрів і потім аналізує зміни кольору у послідовності зображень (див. рис. 2.15).

Рисунок 2.15 – Процес роботи відеоплетизмографії

Для визначення пульсу людини пропонується розробити метод, який аналізує зміну кольору обличчя за допомогою відеозахоплення. Далі виконується аналіз, який полягає у відстеженні змін кольорових компонентів в регіоні інтересу на обличчі. Цей підхід дозволяє фіксувати мікроскопічні зміни кольору шкіри, які відбуваються синхронно з серцебиттям.

У відеоплетизмографії на основі веб–камери важливим є вибір правильної області обличчя для аналізу. Найчастіше використовуються області з великою кількістю кровоносних судин, такі як лоб та щоки. Ці ділянки обличчя менше

піддаються впливу міміки та рухів, що дозволяє отримати більш стабільний сигнал для аналізу пульсу.

Точність визначення пульсу в значній мірі залежить від якості відео та освітлення, оскільки більша інтенсивність освітлення дозволяє краще фіксувати зміни кольору шкіри. Для покращення якості сигналу використовуються різні методи фільтрації шуму та компенсації руху голови.

Застосування алгоритмів обробки зображень, таких як ROI, дозволяє виділити необхідні ділянки обличчя для аналізу. Після сегментації отримується часовий ряд змін інтенсивності кольору, який представляє собою набір значень, що описують зміну кольору шкіри у часі. Цей часовий ряд може бути використаний для подальшого аналізу та розрахунку частоти пульсу.

Щоб визначити пульс, потрібно розбити процес на етапи: захоплення відео за допомогою веб– камери, виділення області обличчя, визначення зміни кольору шкіри в регіоні інтересу, фільтрація та обробка сигналу, аналіз частотних характеристик сигналу для визначення частоти пульсу. Цей підхід до визначення пульсу забезпечує безконтактне вимірювання серцевого ритму, що може бути особливо корисним в медичних та моніторингових системах.

Для точного визначення пульсу людини ефективно використовувати аналіз зеленого каналу колірної моделі RGB, оскільки він має найкращу чутливість до змін кольору, пов'язаних із пульсацією крові. За допомогою аналізу часових рядів можна точно визначити частоту пульсу, враховуючи його періодичну природу. Цей підхід надає можливість отримати деталізовану інформацію про серцевий ритм та його варіабельність, що допомагає в дослідженнях з медичної діагностики.

Нехай $I(x, y, t)$ — інтенсивність зеленого каналу в точці з координатами (x, y) в момент часу t. Визначимо функцію $f(t)$ так, що $f(t)$ = середнє значення $I(x, y, t)$ в області ROI. Це означає, що значення функції відповідає середній інтенсивності зеленого каналу в області інтересу в момент часу t.

Для отримання часового ряду $f(t)$, будемо використовувати значення $I(x, y, t)$ на послідовності кадрів. Часовий ряд $f(t)$ можна визначити як:

$$f(t) = \frac{1}{N}\sum_{n=1}^{N} I(x, y, t)\,, \tag{2.1}$$

де N – кількість пікселів у області ROI.

Для аналізу змін кольору обличчя розглянемо функцію $f(t)$, яка представляє часовий ряд змін інтенсивності кольору. Для визначення частоти серцевих скорочень необхідно знайти домінуючу частоту у цьому сигналі. Для цього можна використати перетворення спектру:

$$F(\omega) = \int f(t) e^{(-j\omega t)} dt\,, \tag{2.2}$$

де ω – частота, а $F(\omega)$ – амплітуда сигналу на цій частоті.

Визначимо вираз для спектральної густини потужності $P(\omega)$ як квадрат модуля $F(\omega)$:

$$P(\omega) = |F(\omega)|^2, \tag{2.3}$$

Для визначення частоти пульсу знаходимо максимум функції $P(\omega)$ в діапазоні частот, що відповідають нормальному діапазону пульсу людини (зазвичай 0.7–4 Гц, що відповідає 42–240 ударам на хвилину):

$$\omega_{pulse} = \mathrm{a}rgmax\big(P(\omega)\big) \tag{2.4}$$

Частота пульсу в ударах на хвилину (BPM) обчислюється за формулою:

$$BPM = 60 \times \omega_{pulse}, \tag{2.5}$$

Отже, було виконано розробку методів аналізу зміни кольору обличчя для визначення пульсу на основі відеозахоплення. Було отримано формули та методики, що дозволяють обрахувати частоту серцевих скорочень з максимальною точністю, використовуючи засоби обробки зображень та аналізу часових рядів. Розроблені методи дозволяють врахувати особливості людського обличчя та умови зйомки, що сприяє отриманню точних результатів вимірювань.

2.3 Розробка методу фільтрації сигналу та спектрального аналізу для виділення пульсації

Для виділення корисного сигналу пульсації з отриманого часового ряду змін кольору необхідно розробити метод фільтрації та спектрального аналізу. Необхідно враховувати, що сирий сигнал містить шуми, артефакти руху та інші перешкоди, які потрібно відфільтрувати.

Програмне забезпечення буде здатне обробляти часові ряди даних, отримані з відеопотоку веб–камери. Наступним кроком програмне забезпечення буде виконувати фільтрацію сигналу для видалення шумів та виділення компонентів, пов'язаних із пульсацією крові.

Одним з ефективних методів попередньої обробки сигналу є нормалізація часового ряду. Для цього від вихідного сигналу віднімається його середнє значення і результат ділиться на стандартне відхилення:

$$f_{norm}(\mathrm{e}) = \frac{(f(t) - \mu)}{\sigma} \tag{2.6}$$

де μ – середнє значення сигналу, σ – стандартне відхилення.

Після нормалізації доцільно застосувати смуговий фільтр для виділення частотного діапазону, що відповідає нормальному діапазону частоти серцевих скорочень, наприклад від 0.7 до 4 Гц. Для цього використовується фільтр Баттерворта:

$$H(\omega) = \frac{1}{\sqrt{1 + (\frac{\omega}{\omega_c})^{2n}}}, \quad (2.7)$$

де ω_c – частота зрізу, n – порядок фільтра.

Відфільтрований сигнал $f\_\text{filtered}(t)$ можна отримати шляхом згортки вихідного сигналу з імпульсною характеристикою фільтра або шляхом множення їх спектрів у частотній області:

$$F_{filtered}(\omega) = F(\omega) \times H(\omega), \quad (2.8)$$

Для спектрального аналізу сигналу застосовується швидке перетворення спектру, яке дозволяє визначити спектральний склад сигналу та виявити основну частоту пульсації. Для підвищення надійності та точності оцінки частоти пульсу можна також використовувати вдосконалені методи спектрального аналізу, такі як:

Метод Уелча для оцінки спектральної густини потужності, який полягає в усередненні модифікованих періодограм:

$$P_{welch}(\omega) = \frac{1}{K}\sum_{k=1}^{K}|F_k(\omega)|^2, \quad (2.9)$$

де K – кількість сегментів, а $F\_k(\omega)$ – перетворення спектру k–го сегмента.

Авторегресійний спектральний аналіз, який моделює сигнал як вихід авторегресійного фільтра, що збуджується білим шумом:

$$P_{AR}(\omega) = \frac{\sigma^2}{|\sum a_k \times e^{(-j\omega k)}|^2}, \quad (2.10)$$

де σ^2 – дисперсія білого шуму, a_k – коефіцієнти авторегресійної моделі.

Після виконання спектрального аналізу для визначення основної частоти пульсації шукається пік у спектральній густині потужності в діапазоні фізіологічно можливих частот пульсу. Частота, що відповідає цьому піку, перераховується в удари на хвилину шляхом множення на 60.

Отже, було виконано розробку методів фільтрації сигналу та спектрального аналізу для виділення пульсації з часового ряду змін кольору обличчя. Запропоновані методи дозволяють ефективно відфільтрувати шуми та артефакти руху, виділити корисний сигнал пульсації та провести точний аналіз його частотних характеристик для визначення частоти серцевих скорочень.

2.4 Розробка методу розрахунку частоти серцевих скорочень на основі відеоданих

Для розрахунку частоти серцевих скорочень на основі даних відеоаналізу зміни кольору обличчя необхідно розробити метод, який перетворить отриманий відфільтрований сигнал у конкретні значення пульсу.

Для визначення частоти серцевих скорочень у часовій області пропонується використовувати метод виявлення піків. Оскільки періодичність сигналу пульсу проявляється у вигляді локальних максимумів на графіку зміни інтенсивності кольору, можна знаходити ці піки і обчислювати часовий інтервал між ними.

Алгоритм виявлення піків включає наступні кроки:

- визначення локальних максимумів у відфільтрованому сигналі;
- відкидання малих піків, які можуть бути шумом;
- обчислення часового інтервалу між сусідніми піками;
- обчислення середнього значення цих інтервалів;
- перетворення середнього інтервалу між піками в частоту серцевих скорочень.

Математично це можна виразити наступним чином:

Нехай t_i – момент часу, в який спостерігається і–й пік. Тоді інтервал між сусідніми піками:

$$\Delta t_i = t_{i+1} - t_i, \tag{2.11}$$

Середній інтервал між піками:

$$\Delta t_{avg} = \frac{1}{n-1} \times \sum t_i\,, \tag{2.12}$$

де n – загальна кількість виявлених піків.

Частота серцевих скорочень в ударах на хвилину:

$$BPM_{time} = \frac{60}{\Delta t_{avg}}, \tag{2.13}$$

Для підвищення надійності оцінки частоти серцевих скорочень рекомендується комбінувати результати частотного аналізу, отримані за допомогою FFT та інших методів спектрального аналізу та часового аналізу:

$$BPM = \alpha \times BPM_{freq} + (1 - a) \times BPM_{time}, \tag{2.13}$$

де α – ваговий коефіцієнт, який можна налаштовувати в залежності від умов вимірювання та якості сигналу.

Отже, було виконано розробку методу розрахунку частоти серцевих скорочень на основі відеоданих. Запропонований метод поєднує частотний та часовий аналіз сигналу для підвищення надійності та точності вимірювань. Використання ковзного вікна та адаптивної фільтрації дозволяє враховувати варіабельність серцевого ритму та підвищувати стійкість методу до змін умов зйомки та руху об'єкта.

2.5 Розробка прикладної системи візуалізації

Інтерфейс програмного додатку розроблений з використанням бібліотеки OpenCV, яка забезпечує можливість візуалізації відеопотоку та виведення результатів аналізу в реальному часі. Графічний інтерфейс відіграє ключову роль у забезпеченні інтуїтивної взаємодії користувача з системою, тому кожна деталь і елемент вікна програми було ретельно продумано. На рисунку 2.16 , представлено схематичне зображення головного вікна програми з пронумерованими секціями, кожна з який має чітке функціональне призначення.

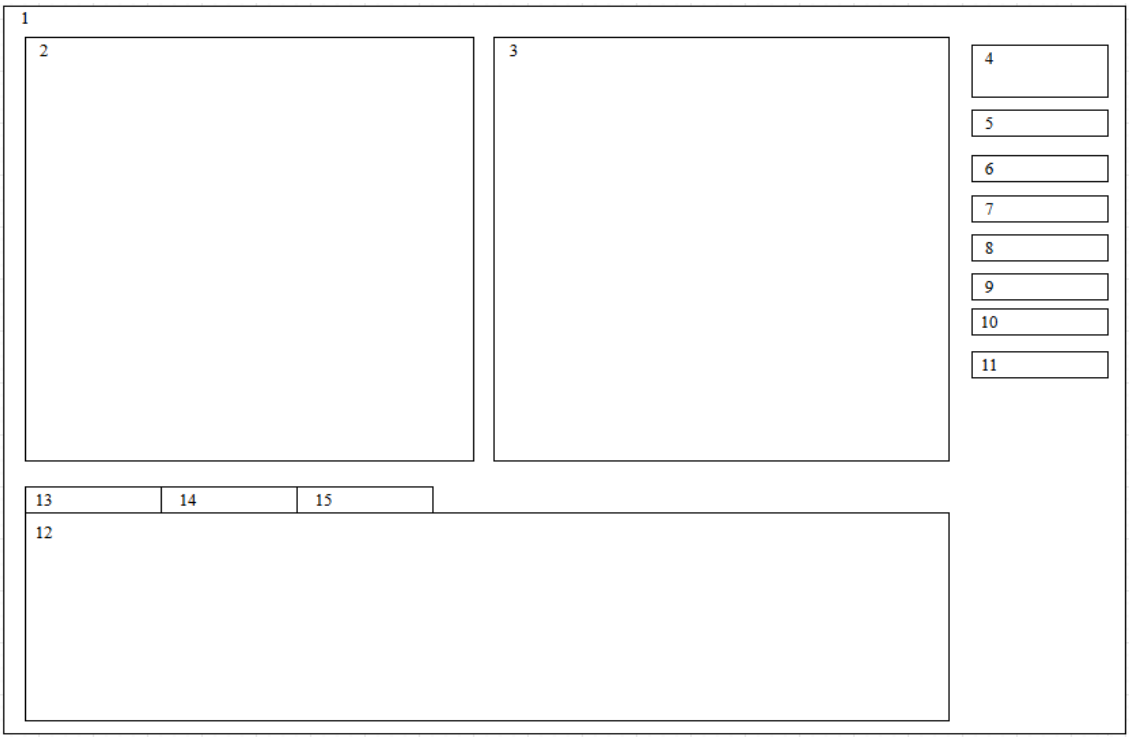


Рисунок 2.16 – Графічна схема інтерфейсу головного вікна програми

Основними елементами головного вікна є:

1. Головне вікно програми;
2. Область відображення відеопотоку з веб–камери в реальному часі.
3. Область відображення обробленого відеопотоку з виділеним обличчям і кольоровим маркуванням пульсу.
4. Область відображення частоти серцебиття з області відеопотоку.
5. Кнопка для запуску або зупинки обробки відеопотоку з веб–камери.
6. Кнопка для початку або завершення запису даних пульсу.

7. Кнопка для експорту графіка сигналу у форматі PNG.
8. Кнопка для експорту графіка частотного спектра у форматі PNG.
9. Кнопка для експорту графіка тренду пульсу у форматі PNG.
10. Кнопка для експорту всіх даних у PDF–звіт.
11. Кнопка для виклику довідкового вікна з інформацією про програму.
12. Область відображення поточного значення пульсу в ударах за хвилину.
13. Вкладка для перегляду графіка сигналу пульсу.
14. Вкладка для перегляду графіка частотного спектра.
15. Вкладка для перегляду графіка тренду пульсу.

Функціональність експорту результатів реалізована через спеціалізовані кнопки для збереження графіків сигналу, частотного спектра та тренду пульсу у форматі PNG, а також для створення комплексного PDF-звіту з усіма даними. Додатково передбачена кнопка виклику довідкового вікна з детальною інформацією про роботу програми.

Інформаційна панель інтерфейсу містить область відображення поточного значення пульсу в ударах за хвилину та секцію оцінки якості сигналу з практичними рекомендаціями щодо оптимізації умов вимірювання. Навігаційна система побудована на основі вкладок, які забезпечують швидкий перехід між різними типами графіків: сигналом пульсу, частотним спектром та трендом серцебиття.

Така організація інтерфейсу створює цілісне робоче середовище, де користувач може ефективно контролювати процес вимірювання, аналізувати отримані результати та зберігати дані для подальшого використання. Система рекомендацій щодо покращення якості сигналу допомагає користувачеві оптимізувати умови освітлення та положення обличчя, що підвищує точність вимірювань. Реалізована функціональність експорту забезпечує гнучкість у роботі з результатами, дозволяючи зберігати як окремі графіки, так і комплексні звіти. Загалом, розроблений інтерфейс успішно поєднує зручність використання з потужними аналітичними можливостями, створюючи ефективний інструмент для моніторингу серцевого ритму.

2.6 Висновки

У другому розділі було проведено розробку методів вимірювання пульсу з використанням змін кольору обличчя. Основними результатами розділу є:

Розроблено методи аналізу зміни кольору обличчя для визначення пульсу, які дозволяють виділяти мікроскопічні зміни кольору шкіри, що пов'язані з пульсацією крові. Обґрунтовано вибір зеленого каналу колірної моделі RGB як найбільш інформативного для аналізу пульсації.

Запропоновано формули обчислення часового ряду зміни інтенсивності кольору в регіоні інтересу та його аналізу для виявлення частоти пульсу. Отримані математичні моделі дозволяють ефективно екстрагувати інформацію про серцеві скорочення з відеоданих.

Розроблено методи фільтрації сигналу та спектрального аналізу для виділення пульсації, що включають нормалізацію часового ряду, застосування смугового фільтра та використання вдосконалених методів спектрального аналізу, таких як метод Уелча та авторегресійний спектральний аналіз.

Розроблено метод розрахунку частоти серцевих скорочень на основі відеоданих, який поєднує частотний та часовий аналіз сигналу для підвищення надійності та точності вимірювань. Запропоновано використання ковзного вікна та адаптивної фільтрації для врахування варіабельності серцевого ритму та підвищення стійкості методу до зовнішніх факторів.

Розроблені методи забезпечують безконтактне вимірювання пульсу за допомогою звичайної веб–камери, що має важливе практичне значення для систем моніторингу стану здоров'я, телемедицини та біометричної аутентифікації. Запропоновані рішення враховують особливості людського обличчя, умови зйомки та можливі джерела шуму, що сприяє отриманню точних результатів вимірювань у реальних умовах.

3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ВИМІРЮВАННЯ ПУЛЬСУ З ВИКОРИСТАННЯМ ВЕБ–КАМЕРИ

3.1 Обґрунтування вибору засобів програмування

Було розглянуто декілька мов програмування таких як Python, C++ та Java.

Python була обрана як оптимальне рішення для імплементації системи вимірювання пульсу через сукупність технічних переваг. Python демонструє значну ефективність у вирішенні задач комп'ютерного зору завдяки розвиненій екосистемі спеціалізованих бібліотек [10]. Зокрема, бібліотека OpenCV забезпечує надійні алгоритми для виявлення та відстеження обличчя, що є критичним компонентом безконтактного вимірювання пульсу [11].

Ефективність обробки сигналів забезпечується бібліотеками NumPy та SciPy, які надають оптимізовані методи для роботи з масивами даних та реалізацію швидкого перетворення спектру [12]. Це дозволяє виділяти частотні компоненти, характерні для серцевих скорочень, із потоку часових даних, отриманих з відеопослідовності.

C++ – мова програмування високого рівня з підтримкою кількох парадигм програмування: об'єктно–орієнтованої, узагальненої та процедурної [13]. ри створенні C++ прагнули зберегти сумісність з мовою C. Більшість програм на C справно працюватимуть і з компілятором C++. C++ має синтаксис, заснований на синтаксисі C.

Java – об'єктно–орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано [14]. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно–орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині.

Таблиця 3.1 – Порівняння мов програмування

Критерій	Python	C++	Java
Швидкість розробки	+	+/–	+/–
Наявність бібліотек комп’ютерного зору	+	+	+
Кросплатформеність	+	–	+
Виразність та простота написання програмного коду	+	–	–
Використання бібліотек OpenCV	+	+	+

Багатоплатформність реалізації програмного рішення забезпечується через кросплатформеність Python та обраних бібліотек. Система здатна функціонувати на різних операційних системах без суттєвих модифікацій коду, що значно розширює потенційну аудиторію користувачів. Реалізація графічного інтерфейсу через фреймворк PyQt5 надає можливості для створення інтуїтивно зрозумілого користувацького досвіду з підтримкою багатопотокової обробки даних [15].

Середовище розробки Visual Studio було обрано через інтегровані засоби профілювання та відлагодження, що дозволяють оптимізувати критичні ділянки коду для підвищення загальної продуктивності системи [16]. Інтеграція з системою контролю версій забезпечує надійне керування розробкою програмного продукту.

При аналізі можливих альтернатив розглядались мови C++ та Java, проте вони демонструють нижчу швидкість розробки та менш зручну екосистему для задач комп'ютерного зору. C++ має перевагу у швидкодії, але комплексність синтаксису та управління пам'яттю збільшують час розробки [17]. Java

забезпечує кросплатформеність, але має обмежені можливості інтеграції з низькорівневими системними компонентами [18].

Використані бібліотеки Python формують комплексну систему інструментів для розв'язання задачі безконтактного вимірювання пульсу:

– OpenCV забезпечує методи виявлення обличчя з використанням каскадних класифікаторів Хаара та виділення регіонів інтересу;

– NumPy надає методи векторизованої обробки даних для ефективної роботи з масивами;

– SciPy реалізує методи частотного аналізу сигналів та фільтрації;

– PyQt5 забезпечує багатопотокову архітектуру програми для одночасної обробки відеопотоку та взаємодії з користувачем;

– Matplotlib створює інтерактивні візуалізації сигналів та спектрів для відображення результатів вимірювання.

Отже врахувавши результати порівняння(див. табл. 3.1) та все вище сказане для розробки було обрано мову програмування Python.

3.2 Розробка алгоритмів вимірювання пульсу на основі веб–камери

Для створення програмного продукту, призначеного для вимірювання пульсу за зміною кольору обличчя, використано поєднання мов програмування Python і бібліотек OpenCV, NumPy, Matplotlib та PyQt5. Ці технології забезпечують обробку відеопотоку, аналіз даних і створення зручного графічного інтерфейсу [19]. Python обрано через його широкий набір бібліотек для обробки зображень і сигналів, що дозволяє ефективно аналізувати зміни кольору обличчя та візуалізувати результати вимірювань. PyQt5 використано для розробки графічного інтерфейсу, який забезпечує зручну взаємодію користувача з програмою, відображення відеопотоку та графіків пульсу. OpenCV застосовується для обробки відеопотоку з вебкамери, виявлення обличчя та аналізу кольорових змін у регіоні інтересу. NumPy забезпечує швидкі обчислення з масивами даних, необхідними для аналізу сигналів, а Matplotlib

використовується для створення інтерактивних графіків, що відображають сигнал пульсу, частотний спектр і тренд пульсу.

Бібліотека OpenCV надає інструменти для роботи з відеопотоком і виявлення обличчя за допомогою каскадного класифікатора Хаара. Цей класифікатор дозволяє точно визначати регіон обличчя у кадрі, що є ключовим для аналізу змін кольору шкіри. Після виявлення обличчя програма обчислює середнє значення RGB–каналів у цій області, що відображає зміни кольору, пов'язані з кровообігом. Отримані дані обробляються за допомогою алгоритму швидкого перетворення спектру, реалізованого через бібліотеку NumPy, для визначення частоти пульсу в діапазоні від 30 до 180 ударів за хвилину. Бібліотека Matplotlib створює три типи графіків: графік сигналу, що відображає зміни інтенсивності кольору, графік частотного спектра, який показує амплітуду частот пульсу, і графік тренду, що ілюструє зміни пульсу з часом [20]. PyQt5 забезпечує інтеграцію всіх компонентів у єдиний графічний інтерфейс, де користувач може запускати вимірювання, переглядати відеопотік, графіки та експортувати результати.

Для кращого розуміння роботи програми розроблено модель її функціонування на прикладі UML–діаграми діяльності. Згідно з цією діаграмою, користувач запускає програму, після чого відкривається головне вікно з графічним інтерфейсом. Користувач натискає кнопку для початку роботи вебкамери, і програма ініціалізує відеопотік. Якщо веб–камера недоступна, видається повідомлення про помилку. У разі успішного запуску програма перевіряє наявність обличчя у кадрі за допомогою каскадного класифікатора. Якщо обличчя не виявлено, відображається відповідне повідомлення, і вимірювання не проводиться. При виявленні обличчя програма аналізує середнє значення RGB–каналів у регіоні обличчя, нормалізує дані та застосовує швидке перетворення спектру для обчислення частоти пульсу. Отриманий пульс відображається у реальному часі разом із графіками сигналу, частотного спектра та тренду. Користувач може увімкнути запис даних у PDF–файл, експортувати графіки у форматі PNG або зупинити вимірювання.

Загальний алгоритм роботи програми включає кілька етапів. Спочатку завантажується графічний інтерфейс, який надає доступ до всіх функцій. Після натискання кнопки запуску програма ініціалізує веб–камеру та перевіряє її працездатність. Якщо камера працює, програма починає обробку відеопотоку, виявляючи обличчя у кожному кадрі. У разі відсутності обличчя користувачу відображається повідомлення. Якщо обличчя виявлено, програма обчислює середнє значення RGB–каналів, формує буфер даних і нормалізує їх. Далі застосовується швидке перетворення спектру для визначення частоти пульсу. Результати оновлюються кожну секунду, а графіки перебудовуються кожні 0,5 секунди. Користувач може зберегти графіки або експортувати всі дані, включаючи логи пульсу.

Рисунок 3.1 – Алгоритм роботи програмного застосунку

Для детального опису обробки сигналу розроблено додатковий алгоритм. На першому етапі програма завантажує відеопотік і перевіряє наявність обличчя. Якщо обличчя виявлено, обчислюється середнє значення RGB–каналів у регіоні обличчя, і дані додаються до буфера. Буфер нормалізується, після чого застосовується швидке перетворення спектру для визначення частот у діапазоні від 30/60 до 180/60 Гц. Частота з максимальною амплітудою переводиться в удари за хвилину. Далі дані використовуються для побудови графіків сигналу, частотного спектра та тренду пульсу. Якщо увімкнено запис, значення пульсу зберігаються у PDF–файл із міткою часу.

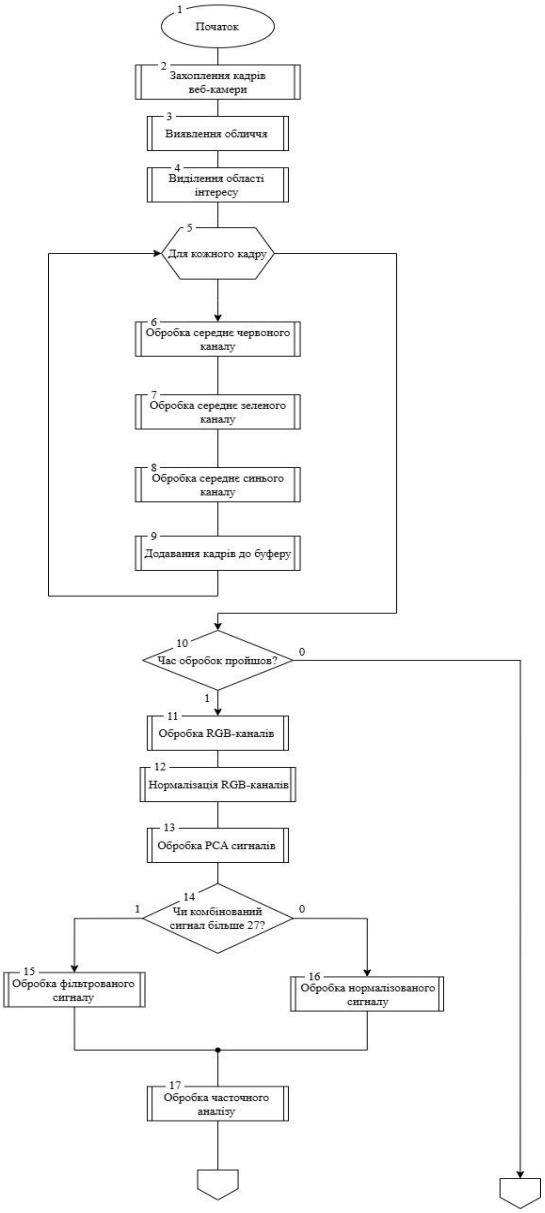


Рисунок 3.2 – Блок–схема алгоритму обробки сигналу пульсу

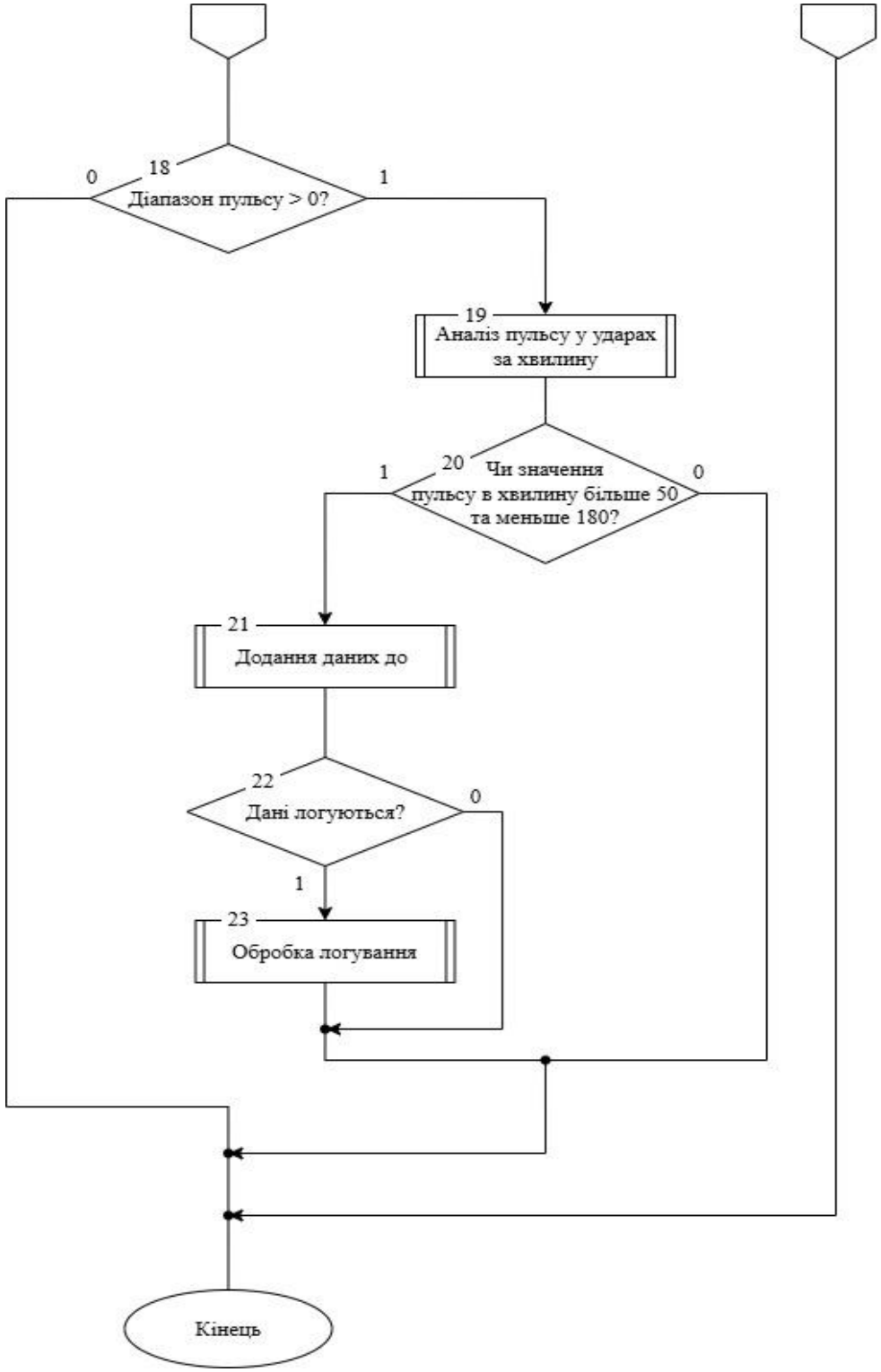


Рисунок 3.3 – Блок–схема алгоритму обробки сигналу пульсу

Таким чином, обґрунтовано вибір технологій Python, OpenCV, NumPy, Matplotlib і PyQt5 для розробки програми. Розроблено основний алгоритм роботи програми та алгоритм обробки сигналу пульсу. Модель роботи програми представлено через UML–діаграми.

3.3 Розробка модуля виявлення та відстежування обличчя

Виявлення обличчя та аналіз його кольору є ключовими етапами для вимірювання пульсу. Для реалізації цього модуля використано бібліотеку OpenCV із каскадним класифікатором Хаара, який забезпечує точне визначення регіону обличчя у відеопотоці. Класифікатор налаштовано для виявлення обличчя з мінімальним розміром 80x80 пікселів, що дозволяє ігнорувати дрібні об’єкти. Після виявлення обличчя програма виділяє регіон інтересу та обчислює середнє значення RGB–каналів, яке відображає зміни кольору шкіри, спричинені кровообігом (див. рис 4.5).

Рисунок 3.4 – Блок–схема функції виявлення обличчя

Після обчислення середнього значення RGB–каналів дані передаються до функції нормалізації, яка усуває вплив освітлення та інших зовнішніх факторів. Нормалізовані дані аналізуються за допомогою швидкого перетворення спектру, яке визначає частоту змін кольору в діапазоні пульсу.

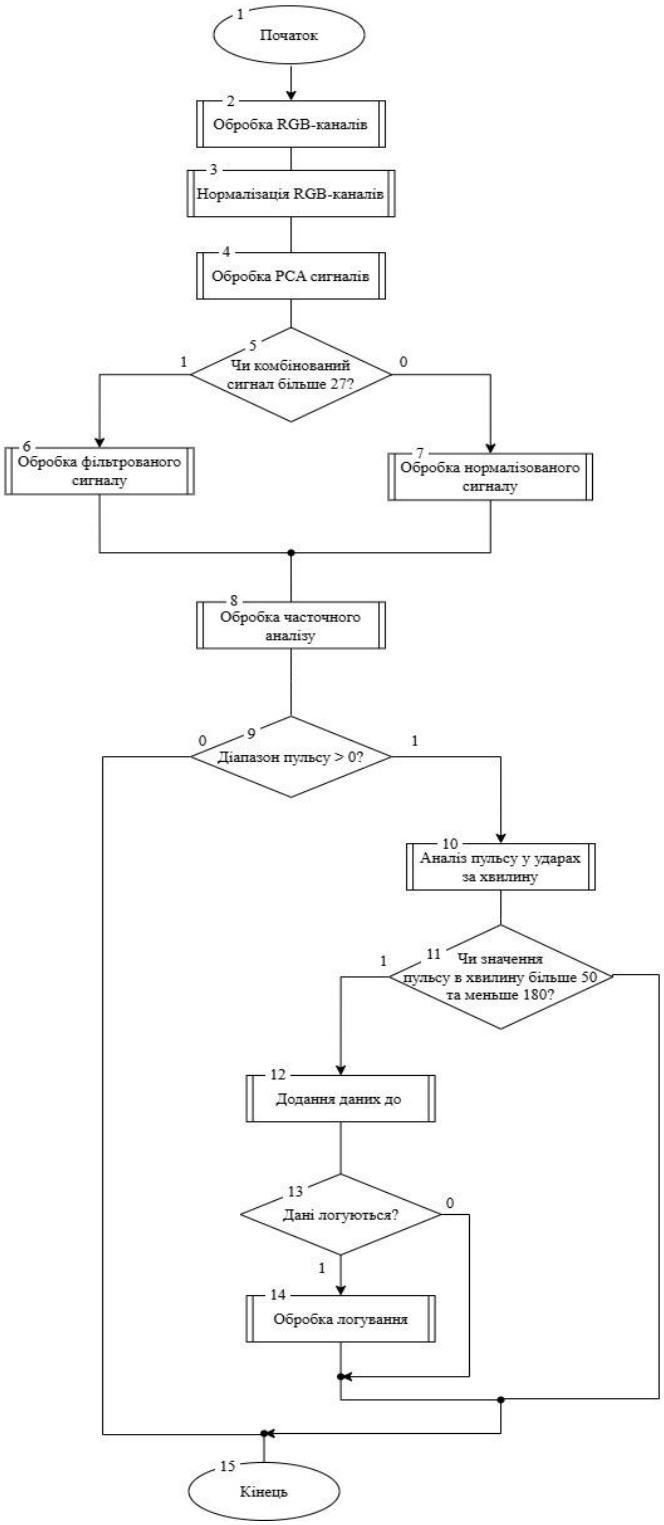


Рисунок 3.5 – Блок–схема функції нормалізації та аналізу сигналу

3.4 Розробка модуля аналізу кольорових змін та розрахунку пульсу

Основним завданням модуля вимірювання пульсу є точне визначення частоти пульсу на основі змін кольору обличчя. Модуль автоматично обчислює середнє значення RGB–каналів, нормалізує дані та застосовує швидке перетворення спектру для визначення частоти пульсу.

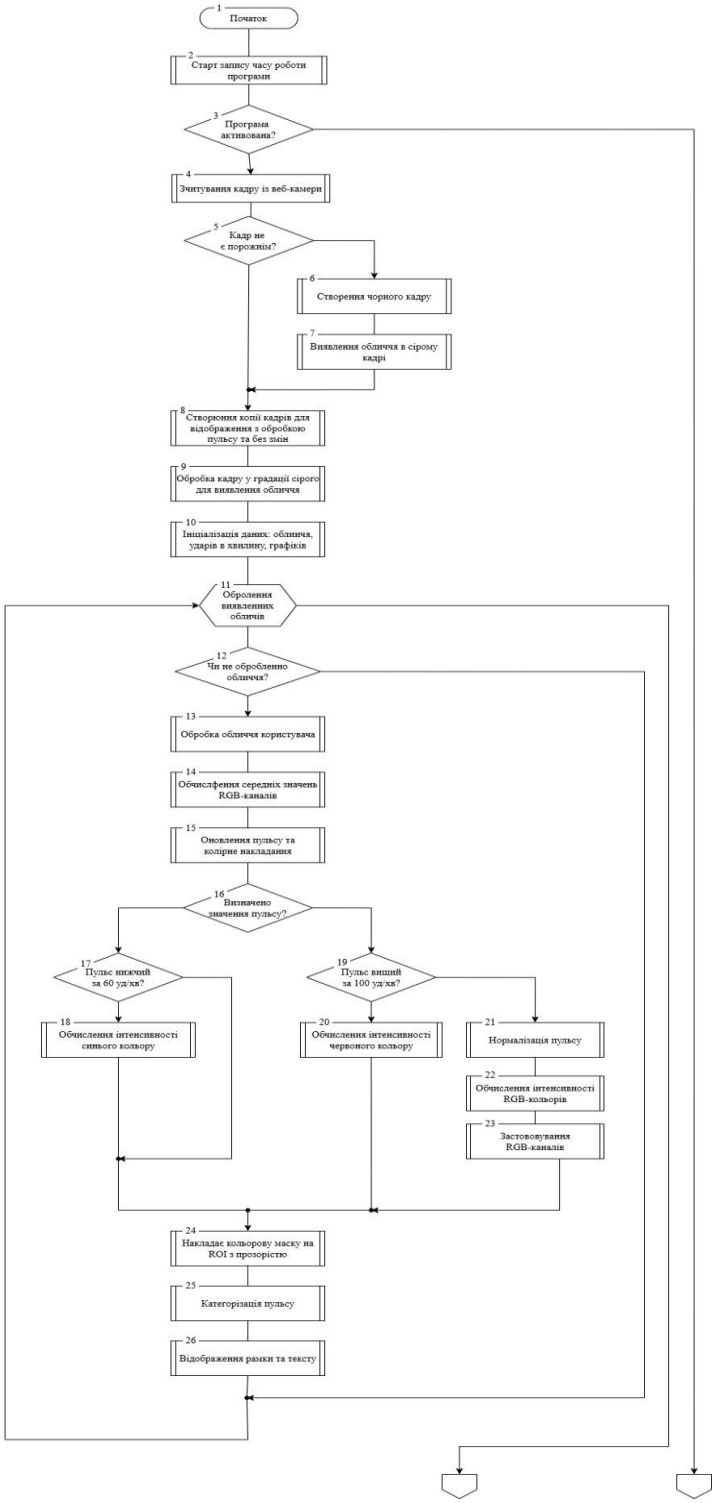


Рисунок 3.6 – Блок–схема функції аналізу відеопотоку

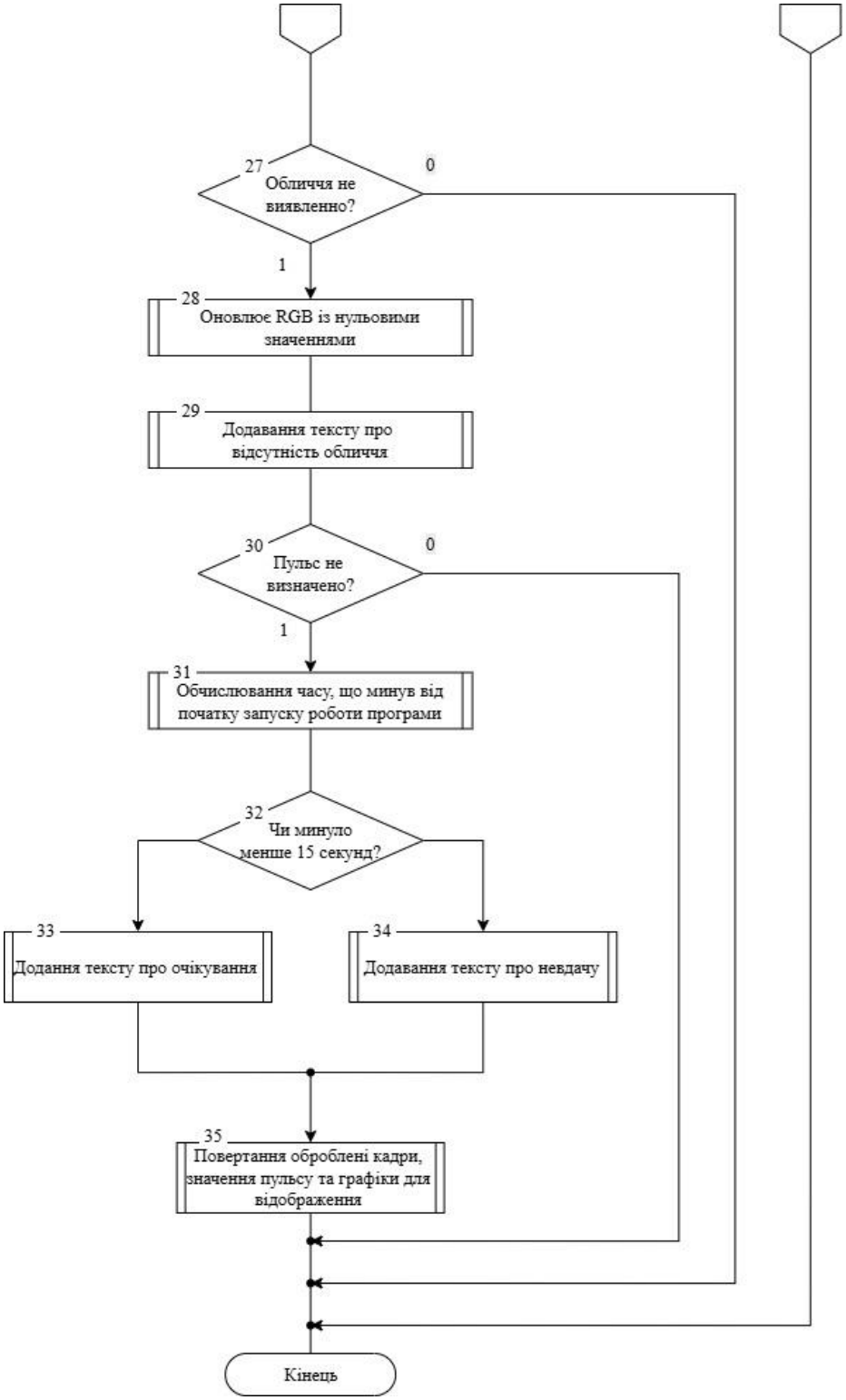


Рисунок 3.7 –Блок–схема функція аналізу відеопотоку

Додатково створено функцію для перевірки відеопотоку на наявність помилок, таких як відсутність сигналу з камери. Якщо помилок немає, модуль обробляє дані та повертає значення пульсу. Для обчислення пульсу використано формулу, яка переводить частоту максимальної амплітуди у удари за хвилину.

Для оцінки якості сигналу, отриманого з відеопотоку, створено функцію assess_signal_quality, яка аналізує умови вимірювання пульсу та надає рекомендації для їх покращення. Ця функція оцінює яскравість кадру, стабільність регіону інтересу, амплітуду сигналу RGB-каналів та рівень шуму в частотному спектрі.

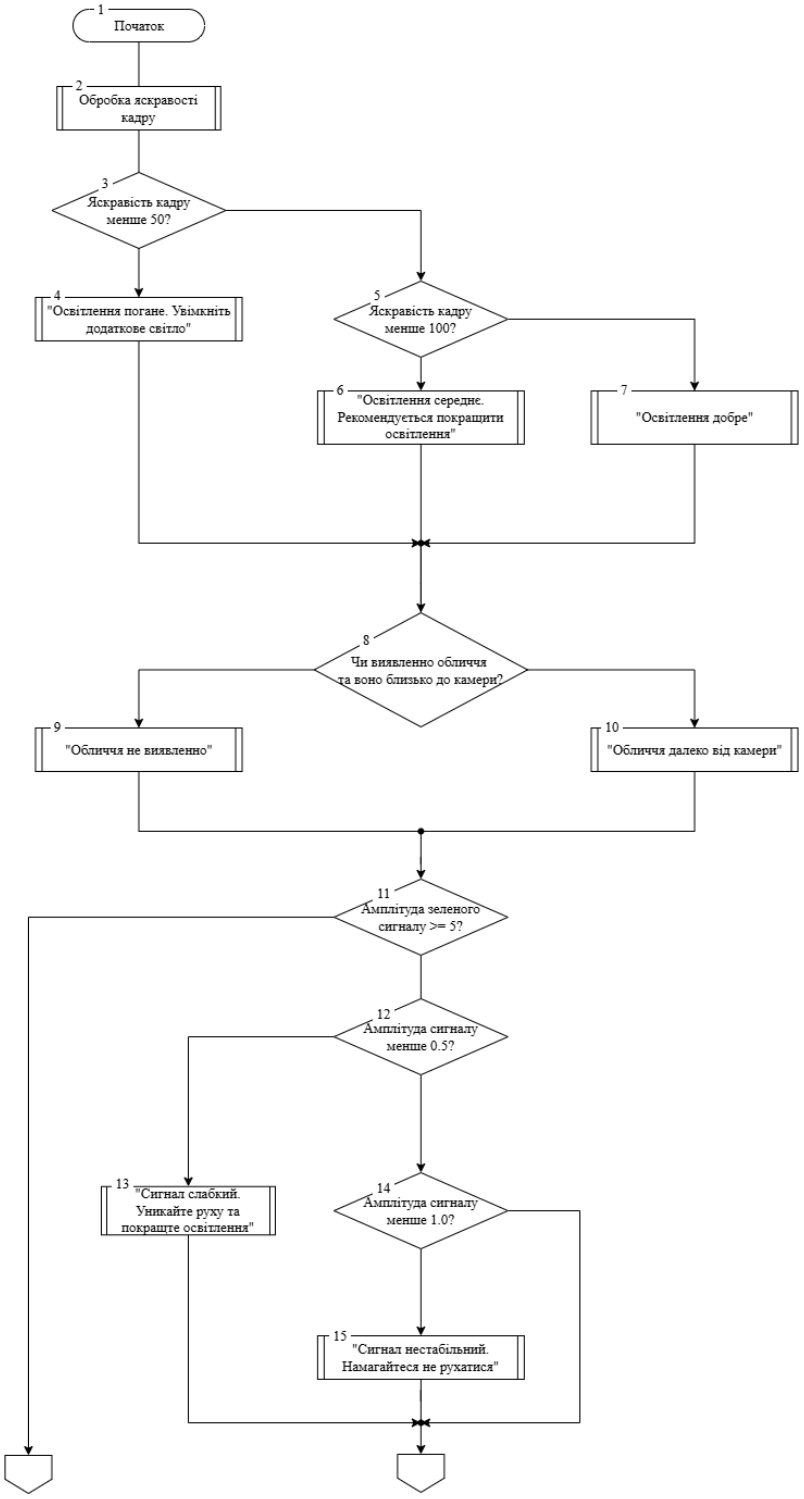


Рисунок 3.8 – Блок-схема оцінки якості сигналу вимірювання серцебиття

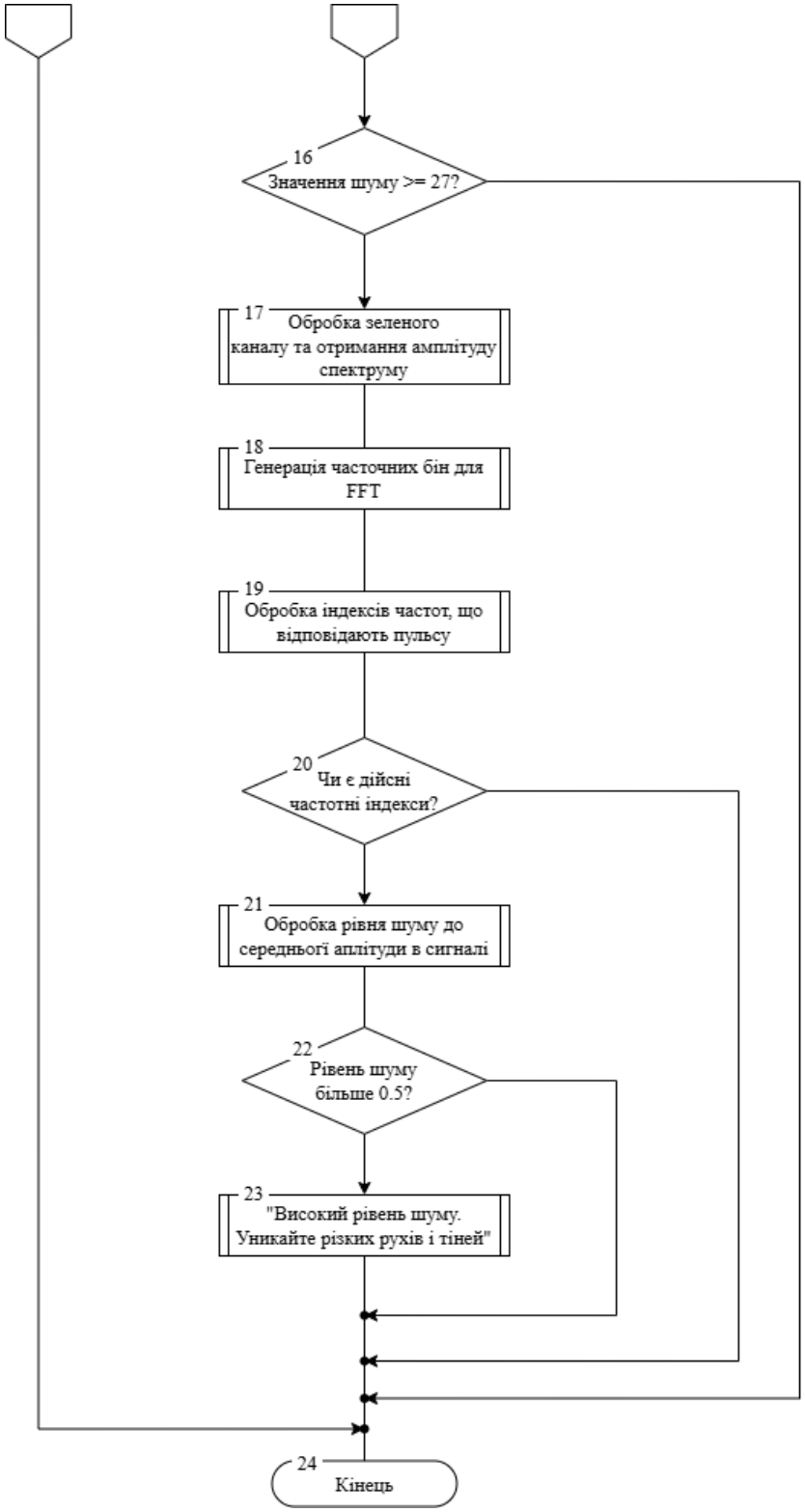


Рисунок 3.9 – Блок-схема оцінки якості сигналу вимірювання серцебиття

На основі цих параметрів визначається якість сигналу як "Хороша", "Середня" або "Погана", а також формулюється повідомлення з рекомендаціями, наприклад, щодо покращення освітлення чи зменшення руху. Функція оновлює

оцінку якості кожні 0,5 секунди, забезпечуючи актуальну інформацію в реальному часі та сприяючи надійному вимірюванню пульсу. Функція оцінки якості сигналу підвищує надійність вимірювань, допомагаючи користувачу оптимізувати умови для точного визначення пульсу [21].

3.5 Розробка модуля візуалізації та інтерфейсу користувача

Інтерфейс програми розроблено з урахуванням зручності та інтуїтивності. Основним кольором обрано зелений, який асоціюється з медициною, із допоміжними чорним і білим для контрасту. При запуску програма відображає головне вікно з двома областями для відеопотоку: оригінального та з накладеним кольоровим ефектом, що відображає пульс.

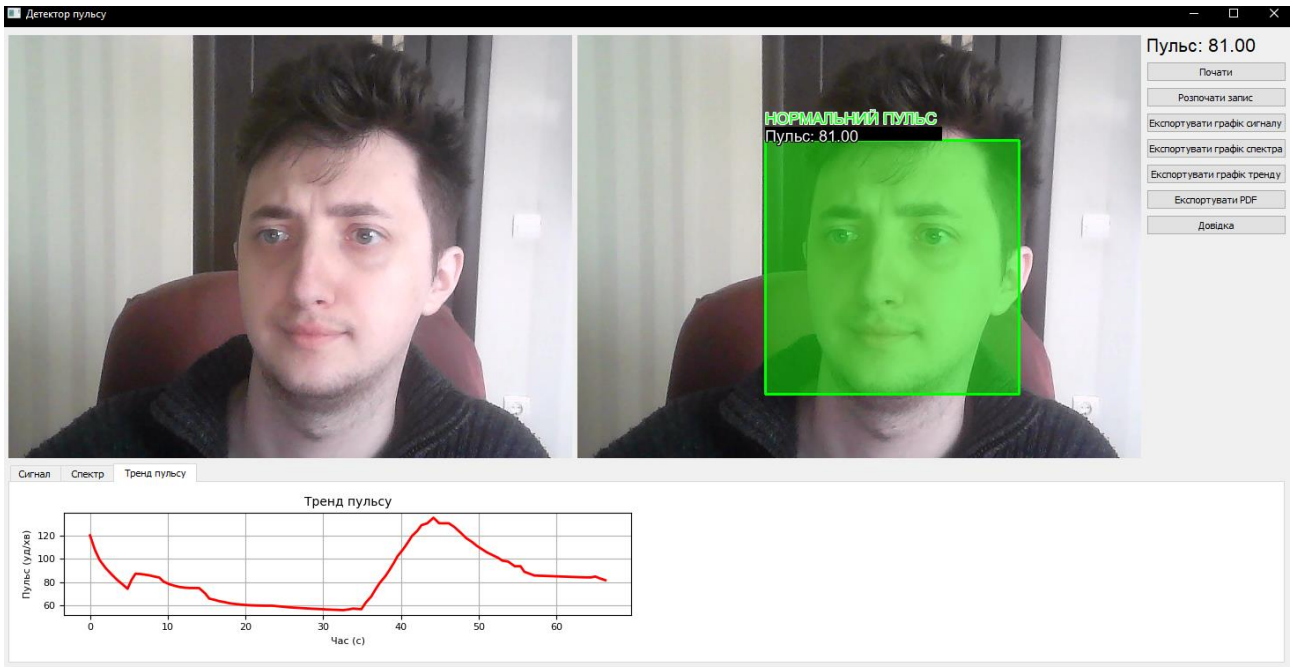


Рисунок 3.10 – Головне вікно програми

Інтерфейс включає вкладки для відображення графіків сигналу, частотного спектра та тренду пульсу. Користувач може запустити або зупинити вебкамеру, увімкнути запис даних у PDF–файл, а також експортувати графіки у форматі PNG. Меню програми містить кнопки для керування вимірюванням і збереженням даних [22].

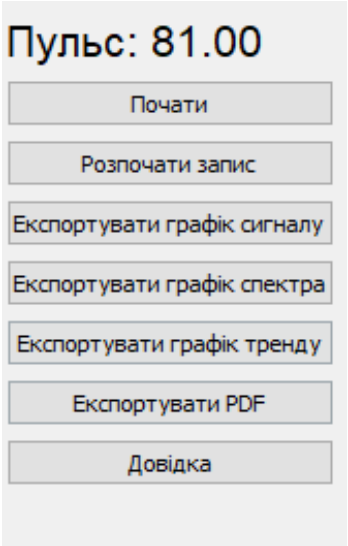


Рисунок 3.11 – Меню програми

Після запуску вимірювання інтерфейс оновлюється, відображаючи значення пульсу та графіки. Користувач може переглянути детальну інформацію про програму на вкладці довідки, яка містить опис функціоналу та відомості про розробника.

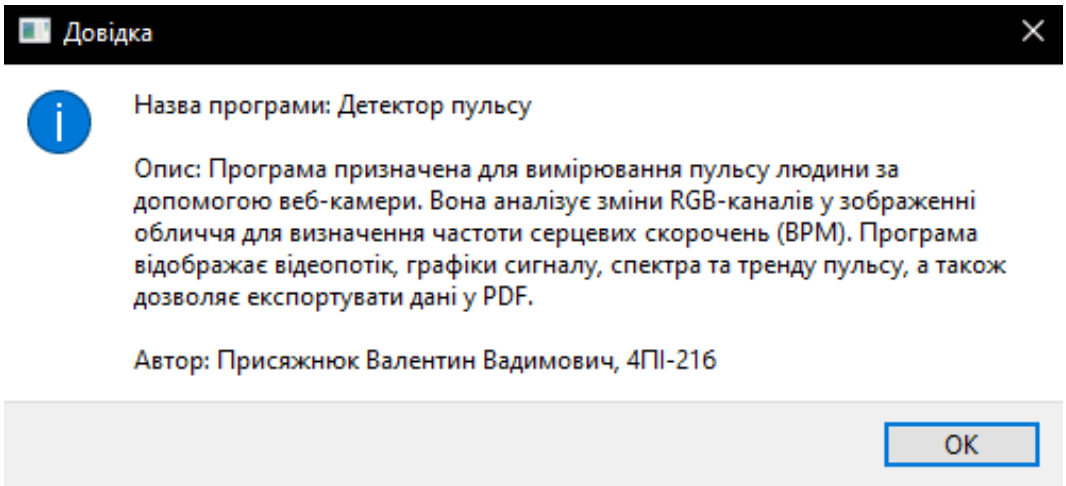


Рисунок 3.12 – Екран довідки

Інтерфейс розроблено у середовищі PyQt5, що забезпечує кросплатформну сумісність і швидке оновлення елементів.

3.6 Висновки

У третьому розділі розроблено алгоритми для вимірювання пульсу за зміною кольору обличчя. Описано використані технології, включаючи Python, OpenCV, NumPy, Matplotlib і PyQt5. Розроблено модулі виявлення обличчя, аналізу кольору, вимірювання пульсу та графічного інтерфейсу, а також представлено UML–діаграми та алгоритми роботи програми.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування програми

Тестування програмного забезпечення для вимірювання пульсу за зміною кольору обличчя є процесом перевірки функціональності програми з метою виявлення помилок, дефектів або невідповідностей вимогам. Основна мета тестування полягає в забезпеченні якості, надійності та точності роботи програми перед її використанням. Для початку тестування необхідно встановити програму на комп'ютер з операційною системою Windows або MacOS, яка відповідає мінімальним технічним вимогам. Перед тестуванням створюється план тестування та забезпечується наявність документації, включаючи специфікації вимог [23].

Першим етапом є функціональне тестування, яке перевіряє коректність роботи програми. Воно включає перевірку правильності виявлення обличчя, обробки відеопотоку з веб–камери, аналізу зміни кольору обличчя та обчислення частоти пульсу. Тестуються сценарії введення некоректних даних, наприклад, відсутність обличчя в кадрі або недостатнє освітлення [24].

Другим етапом є тестування відмов, спрямоване на виявлення поведінки програми в екстремальних умовах. Проводиться аналіз реакції програми на переривання відеопотоку, низьку частоту кадрів або помилки ініціалізації вебкамери. Наприклад, програма має коректно повідомляти користувача про відсутність камери або неможливість її ініціалізації, як показано на рисунку 4.1.

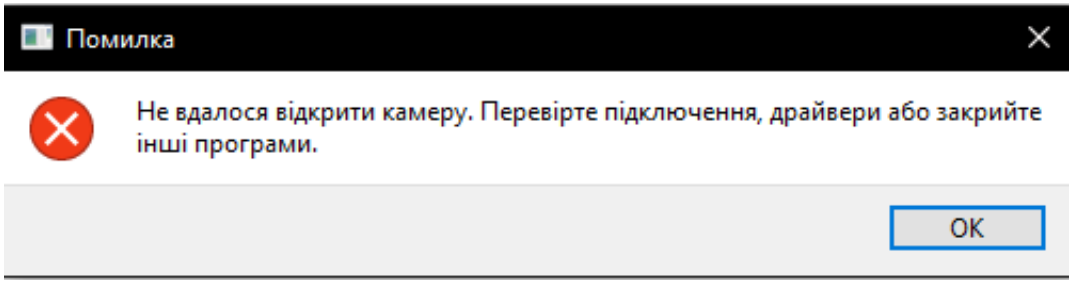


Рисунок 4.1 – Повідомлення про помилку ініціалізації веб–камери

Після функціонального тестування та тестування відмов проводиться тестування продуктивності, яке оцінює швидкість обробки відеопотоку та обчислення пульсу. Вимірюється час відгуку програми на дії користувача, наприклад, запуск або зупинку вимірювання. Результати тестування показали, що програма обробляє відеопотік зі швидкістю не більше 0.5 секунди на кадр, забезпечуючи миттєвий зворотний зв’язок.

Тестування сумісності перевіряє роботу програми на різних операційних системах, включаючи Windows 11, Windows 10, Windows 8.1 та MacOS, а також на пристроях з різними апаратними конфігураціями [25]. Програма успішно функціонує як на високопродуктивних комп’ютерах, так і на пристроях з мінімальними технічними характеристиками, що підтверджує її оптимізацію.

Під час тестування перевірялася реакція програми на відсутність обличчя в кадрі. У таких випадках програма відображає повідомлення "Обличчя не виявлено" та пропонує користувачеві виправити позицію перед камерою, як показано на рисунку 4.2. Це забезпечує зручність використання та чітке інформування про проблему.

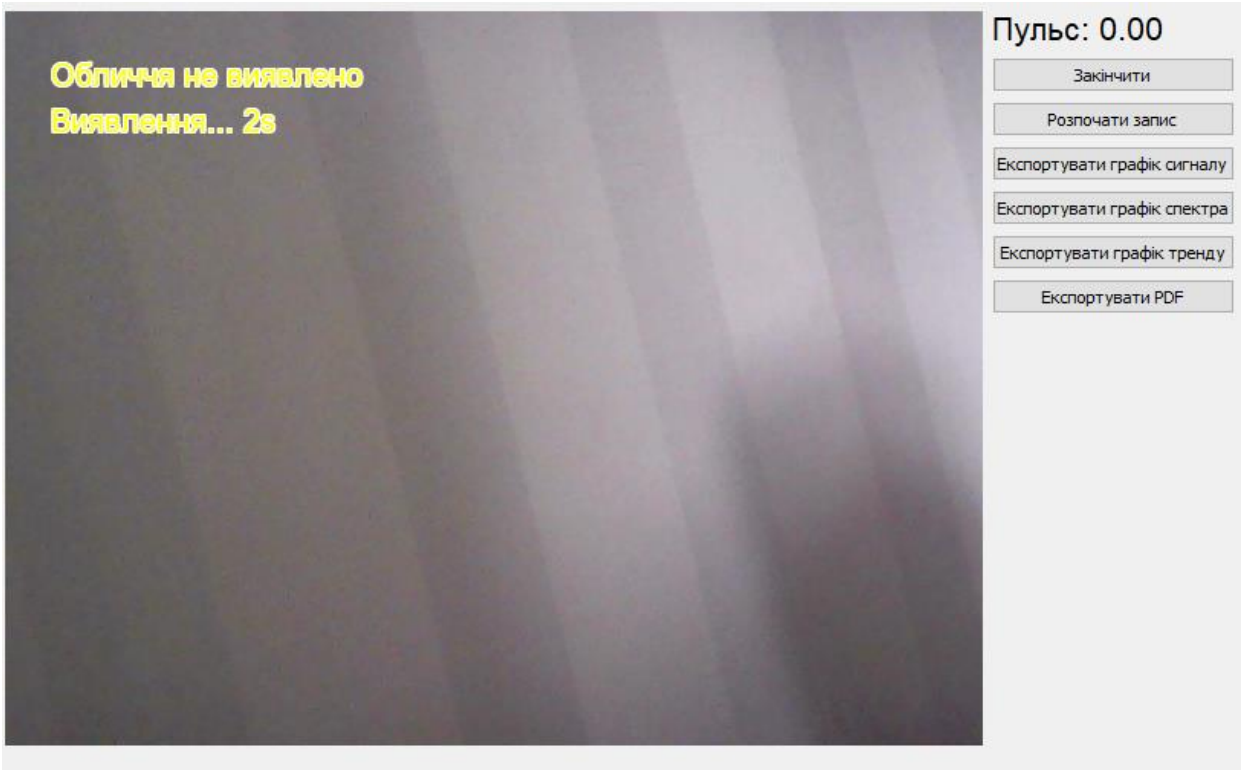


Рисунок 4.2 – Повідомлення про відсутність обличчя в кадрі

Після перевірки реакції на помилкові ситуації тестувалася обробка коректного відеопотоку. Програма успішно виявляла обличчя, аналізувала зміну кольору шкіри та обчислювала частоту пульсу. Результати вимірювання відображалися в реальному часі у вигляді числового значення та графіків, як показано на рисунку 4.3. Тестування підтвердило високу точність обчислень у межах 30–180 ударів за хвилину.

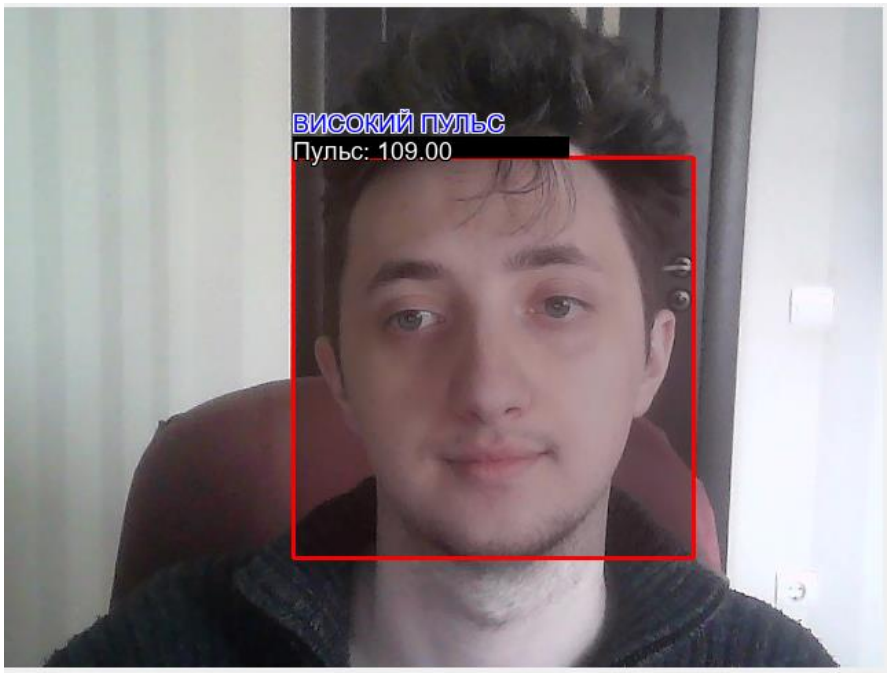


Рисунок 4.3 – Результат вимірювання пульсу з коректним відеопотоком

Було проведено тестування експорту даних. Програма дозволяє зберігати результати вимірювань у форматі PDF та експортувати графіки сигналу, частотного спектру та тренду пульсу у форматі PNG. Результати експорту, показані на рисунку 4.4, демонструють коректне збереження всіх параметрів, включаючи мітки часу та значення пульсу.

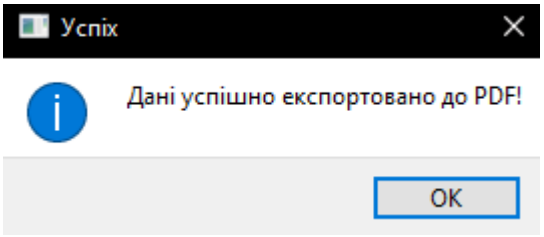


Рисунок 4.4 – Результат експорту даних у формат PDF та PNG

Додатково було проведено тестування експорту даних, якщо камера не була ввімкнена.

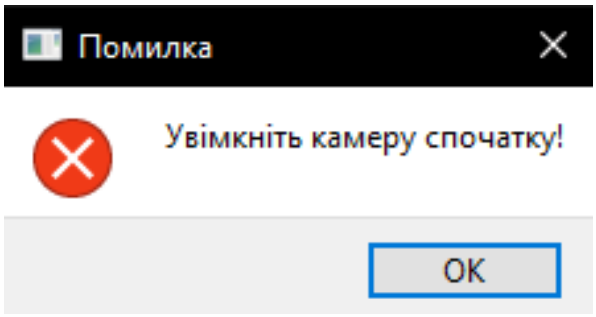


Рисунок 4.5 – Результат експорту даних при вимкненій камері

Тестування також включало аналіз часу обробки даних. Автоматизоване вимірювання пульсу та оновлення графіків займають не більше 5.5 секунд, що забезпечує високу швидкість роботи програми. Швидкий відгук інтерфейсу сприяє зручності використання та позитивному досвіду користувача.

4.2 Розробка інструкції користувача

Розробка інструкції користувача є важливим етапом для забезпечення зручності роботи з програмою вимірювання пульсу. Інструкція описує процес встановлення, запуску та використання програми, а також технічні вимоги до обладнання.

Програма призначена для автоматизованого вимірювання частоти пульсу шляхом аналізу зміни кольору обличчя у відеопотоці з вебкамери. Вона підтримує виявлення обличчя, обробку відеоданих, обчислення пульсу, відображення результатів у текстовому та графічному форматах, а також експорт даних у файли PDF та PNG.

Огляд можливостей:

– виявлення обличчя: програма автоматично ідентифікує обличчя у відеопотоці з вебкамери;

– вимірювання пульсу: аналізує зміну кольору шкіри для визначення частоти пульсу в реальному часі;

– відображення графіків: показує графіки сигналу, частотного спектру та тренду пульсу;

– експорт даних: дозволяє зберігати результати вимірювань у форматі PDF та графіки у форматі PNG;

– повідомлення про помилки: інформує користувача про відсутність обличчя, недостатнє освітлення або проблеми з камерою.

Технічні вимоги до апаратного забезпечення наведено в таблиці 4.1.

Таблиця 4.1 – Вимоги до апаратного забезпечення

Тип процесора	32–розрядний (x86) або 64–розрядний (x64) процесор з тактовою частотою 1,5 ГГц
Об’єм оперативної пам’яті	2 ГБ для 64–разрядної системи
Місце на жорсткому диску	500 МБ
Графічний пристрій	графічний пристрій DirectX9
Веб–камера	30 FPS, роздільна здатність 640x360 або вище

Програма має безкоштовну ліцензію та відкритий код. Процес встановлення виконується автоматично. Після запуску користувач бачить головний інтерфейс з двома вікнами для відображення відеопотоку, числовим значенням пульсу та вкладками з графіками.

4.3 Висновки

У четвертому розділі проведено тестування програмного забезпечення для вимірювання пульсу за зміною кольору обличчя. Перевірено реакцію програми на помилкові ситуації, такі як відсутність обличчя, недостатнє освітлення або проблеми з вебкамерою. Підтверджено коректність обробки відеопотоку, точність обчислення пульсу та можливість експорту даних. Розроблено

інструкцію користувача, яка описує встановлення, запуск та основні функції програми.

ВИСНОВКИ

У бакалаврській кваліфікаційній роботі розроблено методи та програмні засоби для вимірювання пульсу за зміною кольору обличчя з використанням відеопотоку з веб–камери. Проведено аналіз сучасних підходів до вимірювання пульсу шляхом фотоплетизмографії. Обґрунтовано необхідність розробки програмного забезпечення для автоматизованого аналізу зміни кольору шкіри обличчя з метою підвищення точності та зручності вимірювання частоти пульсу.

Вперше запропоновано метод визначення частоти пульсу на основі аналізу середнього значення RGB–каналів у реальному часі, особливість якого полягає у нормалізації сигналу та застосуванні швидкого перетворення спектру для виділення частотного спектру в діапазоні 30–180 ударів за хвилину, що забезпечує високу точність вимірювання. Вперше розроблено метод візуалізації пульсового сигналу, який включає створення графіків сигналу, частотного спектру та тренду пульсу, що дозволяє користувачеві наочно оцінити динаміку вимірювань. Вперше запропоновано метод автоматичного виявлення помилкових ситуацій, таких як відсутність обличчя в кадрі або недостатнє освітлення, з відображенням відповідних повідомлень для користувача, що підвищує надійність програми.

Практичне значення отриманих результатів полягає в розробці програмного забезпечення, яке інтегрує алгоритми обробки відеопотоку, аналізу кольору обличчя та візуалізації результатів у зручному графічному інтерфейсі на основі PyQt5. Розроблено методи вимірювання пульсу, які дозволяють з високою точністю визначати частоту серцевих скорочень у реальному часі. На основі проведених досліджень створено алгоритми для обробки відеоданих, виявлення обличчя за допомогою каскаду Хаара та експорту результатів у формати PDF та PNG.

Проведене тестування підтвердило достовірність теоретичних досліджень методів вимірювання пульсу та засобів їх реалізації. Розроблене програмне

забезпечення можна використовувати в медичній галузі для моніторингу частоти пульсу, а також у фітнес–додатках для оцінки фізичного стану користувачів.

Ключові слова: фотоплетизмографія, вимірювання серцебиття, обробка відеопотоку, аналіз кольору обличчя, програмне забезпечення, веб–камера.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. О. Н. Романюк, В. В. Присяжнюк, «Комп’ютерне визначення частоти серцебиття шляхом аналізу кольорових змін обличчя», *Матеріали XXV Всеукраїнської науково–технічної конференції молодих вчених*, Вінниця: ВНТУ, 2025, с. 53–56.

2. О. Н. Романюк, В. В. Присяжнюк, «Статистика серцево–судинних захворювань», *Матеріали XXVII Міжнародної науково–технічної конференції «Технологія-2025»*, Київ: СНУ, 2025, с. 45–48.

3. О. Н. Романюк, В. В. Присяжнюк, «Використання штучного інтелекту для визначення ритму серцебиття», *Матеріали XII Всеукраїнської науково–практичній конференції молодих учених «Інформаційні технології»,* Київ: КСУ, 2025, с. 80–82.

4. П. І. Ковальчук, О. В. Сидоренко, «Алгоритмічне та програмне забезпечення для автоматизованого аналізу частоти серцевих скорочень на основі відеозапису обличчя», Національний технічний університет України «КПІ імені Ігоря Сікорського», 2022, с. 1–15.

5. Т. Ш. Пащенко, «Система діагностики пульсу людини на основі аналізу відеоінформації», Київський національний університет імені Тараса Шевченка, Київ, Україна, 2021, 1 с.

6. Region of Interest [Електронний ресурс]– Режим доступу: https://en.wikipedia.org/wiki/Region_of_interest

7. Частота_серцевих_скорочень [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Частота_серцевих скорочень.

8. L. Wang and Y. Chen, "FaceHR: A Research Platform for Non–contact Heart Rate Measurement," IEEE Transactions on Biomedical Engineering, vol. 68, no. 4, pp. 1120–1132, 2024.

9. Microsoft. Visual Studio Code – Code Editing. Redefined. URL: https://code.visualstudio.com/date of access: 17.04.2025.

10. Комп'ютерний зір [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Комп%27ютерний_зір.

11. Зір [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Зір.

12. M. Y. Wu, H. E. Chang, and J. H. Liang, "Eulerian Video Magnification for Vital Signs Monitoring," in Proc. ICPRAM 2018, pp. 567–571, 2018.

13. Cardiio, "Cardiio: Heart Rate Monitor App," [Online]. Available: https://www.cardiio.com. [Accessed: 02–May–2025].

14. Y. Liu, "FaceBeat: Non–contact Heart Rate Monitoring System," Tech. Rep., 2023.

15. Теорія розпізнавання образів [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Теорія_розпізнавання_образів.

16. Emgu CV: OpenCV in .Net (C#, VB, C++) [Електронний ресурс] – Режим доступу: http://www.emgu.com/wiki/index.php/Main_Page.

17. OpenCV [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/OpenCV.

18. Діаграма потоків даних [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Діаграма_потоків_даних.

19. Інтерфейс_користувача [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Інтерфейс_користувача.

20. Python [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Python_(programming_language).

21. C++ [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/C%2B%2B.

22. Java [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Java.

23.J. Kim and K. Lee, "Pulse Rate Monitor: Cross–platform Heart Rate Monitoring Solution," Journal of Biomedical Engineering, vol. 45, no. 3, pp. 234–240, 2024.

24. Тестування програмного забезпечення [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Тестування_програмного_забезпечення.

25. RPPG Health Inc., "RPPG Health: Web–based Heart Rate Monitoring Service," [Online]. Available: https://www.rppghealth.com. [Accessed: 02–May–2025].

ДОДАТКИ

Додаток А. Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
«25» 03 2025 р.

Технічне завдання
на бакалаврську кваліфікаційну роботу «Комп'ютерна програма для визначення частоти серцебиття на основі аналізу зміни кольору обличчя»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
д.т.н., проф. Романюк О.Н.
«24» 03 2025 року.
Виконав:
студент гр. 4ПІ-21б Присяжнюк В. В.
«24» 03 2025 року.

м. Вінниця – 2025 рік

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Комп'ютерна програма для визначення частоти серцебиття на основі аналізу зміни кольору обличчя».

Галузь застосування – медичні технології.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол №18 від 18 лютого 2025 р.

3. Мета та призначення розробки.

Метою роботи є покращення визначення серцебиття людини за рахунок використання комп'ютерного зору та обличчя людини, що дозволяє визначити серцеві скорочення.

4. Вихідні дані для проведення НДР

1. X. Sun, Y. Su, X. Hou, X. Yuan, H. Li, and C. Wang, "Research on heart rate detection from facial videos based on an attention mechanism 3D convolutional neural network," *Electronics*, vol. 14, no. 2, p. 269, 2025.

2. B. Li, P. Zhang, J. Peng, and H. Fu, "Non-contact PPG signal and heart rate estimation with multi-hierarchical convolutional network," *arXiv preprint*, arXiv:2104.02260, 2021. [Online]. Available: https://arxiv.org/abs/2104.02260.

3. Y. Qiu, Y. Liu, J. Arteaga-Falconi, H. Dong, and A. El Saddik, "EVM-CNN: Real-time contactless heart rate estimation from facial video," *arXiv preprint*, arXiv:2212.13843, 2022. [Online]. Available: https://arxiv.org/abs/2212.13843

4. C. Xu and X. Li, "Facial video-based remote heart rate measurement via spatial-temporal convolutional network," in *Intelligent Life System Modelling, Image Processing and Analysis*, , vol. 1467, Singapore: Springer, 2021, pp. 33–42.

5. Технічні вимоги

Вхідні дані — зображення з веб-камери користувача.

Вихідні дані — метрики частоти серцебиття.

6. Конструктивні вимоги.

Інтерфейс програми поовинен відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до бакалаврської кваліфікаційної роботи;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз методів і засобів для дистанційного вимірювання пульсу веб–камерою	25.03.25 – 07.04.25
2	Розробка методів для дистанційного вимірювання частоти пульсу серцебиття	08.04.25 – 18.04.25
3	Обґрунтування вибору засобів програмної реалізації	19.04.25 – 01.05.25
4	Розробка програмних засобів для вимірювання серцебиття з використанням веб–камери	02.05.25 – 09.05.25
5	Тестування програми	10.05.25 – 20.05.25
6	Оформлення матеріалів до захисту БКР	21.05.25 – 30.05.25

10. Порядок контролю та прийняття.

Усі етапи бакалаврської роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

Додаток Б. Протокол перевірки кваліфікаційної роботи

Назва роботи: Комп'ютерна програма для визначення частоти серцебиття на основі аналізу зміни кольору обличчя

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ ПЗ, ФІТКІ, 4ПІ–21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 6,4 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Романюк О.Н
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Присяжнюк В.В.
(підпис) (прізвище, ініціали)

Додаток В. Лістинг програмного забезпечення

```
import cv2
import numpy as np
import time
from collections import deque
import matplotlib
matplotlib.use('Agg')  # Non-interactive backend
import matplotlib.pyplot as plt
from matplotlib.backends.backend_agg import FigureCanvasAgg
from matplotlib.backends.backend_pdf import PdfPages
from PIL import Image, ImageDraw, ImageFont
from PyQt5 import QtWidgets, QtGui, QtCore
import sys
import threading
from pathlib import Path
from datetime import datetime
import shutil
from scipy import signal
import traceback
import os
import warnings
import tempfile

# Функція для отримання шляху до ресурсів у .exe або під час розробки
def resource_path(relative_path):
    """Отримує абсолютний шлях до ресурсу, працює як у розробці, так і в .exe"""
    if hasattr(sys, '_MEIPASS'):
        return str(Path(sys._MEIPASS) / relative_path)
    return str(Path(__file__).parent / relative_path)

# Отримуємо директорію програми (.exe або скрипта)
def get_program_dir():
    """Повертає директорію, де розташований .exe або скрипт"""
    if getattr(sys, 'frozen', False):
        return Path(sys.executable).parent
    return Path(__file__).parent

PROGRAM_DIR = get_program_dir()

# Функція для безпечного запису до лог-файлу
def safe_log(log_file, message):
    """Спробувати записати до лог-файлу, з фолбеком на тимчасовий файл."""
    try:
        with open(log_file, "a", encoding="utf-8") as f:
            f.write(message)
    except (PermissionError, OSError):
        # Фолбек на тимчасовий файл
        temp_log = Path(tempfile.gettempdir()) / "pulse_detector_temp_log.txt"
        try:
```

```
            with open(temp_log, "a", encoding="utf-8") as f:
                f.write(message)
        except Exception:
            pass  # Якщо і це не вдалося, ігноруємо логування

class PulseDetector:
    def __init__(self, buffer_size=150, fps=30):
        """
        Ініціалізація детектора пульсу для RGB-каналів.

        Параметри:
            buffer_size (int): Максимальний розмір буфера даних (зразків).
            fps (float): Частота кадрів веб-камери (кадрів/с).
        """
        self.buffer_size = buffer_size
        self.fps = fps
        self.data_buffer_red = deque(maxlen=buffer_size)  # Буфер для червоного каналу
        self.data_buffer_green = deque(maxlen=buffer_size)  # Буфер для зеленого каналу
        self.data_buffer_blue = deque(maxlen=buffer_size)  # Буфер для синього каналу
        self.times = deque(maxlen=buffer_size)  # Часові мітки
        self.bpm = 0  # Поточний пульс
        self.bpm_history = deque(maxlen=15)  # Історія для усереднення
        self.bpm_trend = deque(maxlen=3600)  # Довгостроковий тренд
        self.bpm_times = deque(maxlen=3600)  # Часові мітки тренду
        self.last_update = time.time()  # Останнє оновлення пульсу
        self.last_graph_update = time.time()  # Останнє оновлення графіків
        self.signal_graph = np.zeros((200, 720, 3), dtype=np.uint8)  # Графік сигналу
        self.fft_graph = np.zeros((200, 720, 3), dtype=np.uint8)  # Графік FFT
        self.trend_graph = np.zeros((200, 720, 3), dtype=np.uint8)  # Графік тренду
        self.logging_enabled = False  # Статус логування
        self.csv_data = []  # Дані для експорту
        # Смуговий фільтр Butterworth (0.83–3 Гц, 50–180 уд/хв)
        self.b, self.a = signal.butter(4, [0.83, 3], btype='bandpass', fs=fps)
        self.log_file = PROGRAM_DIR / "export_log.txt"

    def start_logging(self):
        """Почати логування даних пульсу."""
        if not self.logging_enabled:
            self.csv_data = []
            self.logging_enabled = True
            safe_log(self.log_file, f"{datetime.now()}: Started logging\n")

    def stop_logging(self):
        """Зупинити логування."""
        if self.logging_enabled:
            self.logging_enabled = False
            safe_log(self.log_file, f"{datetime.now()}: Stopped logging\n")

    def update(self, rgb_intensity, face_detected):
        """
```

```
    Оновити дані пульсу на основі RGB-каналів.

    Параметри:
        rgb_intensity (list or np.ndarray): Середні значення [R, G, B] у ROI.
        face_detected (bool): Чи виявлено обличчя.

    Повертає:
        tuple: (bpm, signal_graph, fft_graph, trend_graph)
    """
    current_time = time.time()

    safe_log(self.log_file, f"{datetime.now()}: PulseDetector.update called, face_detected={face_detected}, rgb_intensity={rgb_intensity}\n")

    if not face_detected or not isinstance(rgb_intensity, (list, np.ndarray)) or len(rgb_intensity) != 3:
        return self.get_bpm(), self.signal_graph, self.fft_graph, self.trend_graph

    # Додавання значень до буферів
    self.data_buffer_red.append(rgb_intensity[0])
    self.data_buffer_green.append(rgb_intensity[1])
    self.data_buffer_blue.append(rgb_intensity[2])
    self.times.append(current_time)

    # Логування стану буферів
    safe_log(self.log_file, f"{datetime.now()}: Buffer sizes - Red: {len(self.data_buffer_red)}, Green: {len(self.data_buffer_green)}, Blue: {len(self.data_buffer_blue)}\n")

    # Оновлення пульсу кожні 0.5 секунди
    if current_time - self.last_update >= 0.5 and len(self.data_buffer_red) >= 15:
        self.last_update = current_time

        # Формування масивів сигналів
        red_data = np.array(self.data_buffer_red)
        green_data = np.array(self.data_buffer_green)
        blue_data = np.array(self.data_buffer_blue)

        # Нормалізація сигналів
        red_norm = (red_data - np.mean(red_data)) / np.std(red_data) if np.std(red_data) != 0 else red_data
        green_norm = (green_data - np.mean(green_data)) / np.std(green_data) if np.std(green_data) != 0 else green_data
        blue_norm = (blue_data - np.mean(blue_data)) / np.std(blue_data) if np.std(blue_data) != 0 else blue_data

        # Комбінація сигналів через PCA
        signals = np.vstack((red_norm, green_norm, blue_norm))
        covariance_matrix = np.cov(signals)
        eigenvalues, eigenvectors = np.linalg.eigh(covariance_matrix)
        principal_component = eigenvectors[:, np.argmax(eigenvalues)]
        combined_signal = np.dot(principal_component, signals)

        safe_log(self.log_file, f"{datetime.now()}: Combined signal length: {len(combined_signal)}\n")
```

```
            # Фільтрація комбінованого сигналу
            try:
                if len(combined_signal) >= 15:  # Зменшено поріг для filtfilt
                    filtered_signal = signal.filtfilt(self.b, self.a, combined_signal, padlen=14)
                    normalized_signal = (filtered_signal - np.mean(filtered_signal)) / np.std(filtered_signal)
                else:
                    normalized_signal = (combined_signal - np.mean(combined_signal)) / np.std(combined_signal)

                # Частотний аналіз
                fft_data = np.abs(np.fft.rfft(normalized_signal))
                freqs = np.fft.rfftfreq(len(normalized_signal), 1.0/self.fps)

                valid_range = np.where((freqs >= 50/60) & (freqs <= 180/60))[0]
                if len(valid_range) > 0:
                    max_idx = valid_range[np.argmax(fft_data[valid_range])]
                    bpm_value = freqs[max_idx] * 60

                    if 50 <= bpm_value <= 180:
                        self.bpm_history.append(bpm_value)
                        self.bpm = np.mean(self.bpm_history)
                        self.bpm_trend.append(self.bpm)
                        self.bpm_times.append(current_time)

                        if self.logging_enabled:
                            timestamp = datetime.fromtimestamp(current_time).strftime("%Y-%m-%d %H:%M:%S")
                            self.csv_data.append([timestamp, self.bpm, rgb_intensity[0], rgb_intensity[1], rgb_intensity[2]])
            except Exception as e:
                safe_log(self.log_file, f"{datetime.now()}: Exception in signal processing: {str(e)}\n")
                safe_log(self.log_file, "Traceback:\n" + "".join(traceback.format_exc()) + "\n")

        # Оновлення графіків кожні 0.2 секунди
        if current_time - self.last_graph_update >= 0.2:
            self.last_graph_update = current_time
            self.update_signal_graph()
            self.update_fft_graph()
            self.update_trend_graph()

        return self.get_bpm(), self.signal_graph, self.fft_graph, self.trend_graph

    def get_bpm(self):
        """Отримати поточне значення пульсу."""
        return int(self.bpm)

    def update_signal_graph(self):
        """Оновити графік сигналу (на основі зеленого каналу для відображення)."""
        if len(self.data_buffer_green) < 2:
            return

        fig = plt.figure(figsize=(7.2, 2), dpi=100)
```

```
    ax = fig.add_subplot(111)

    data_array = np.array(list(self.data_buffer_green))
    data_normalized = (data_array - np.min(data_array)) / (np.max(data_array) - np.min(data_array))

    ax.plot(data_normalized, 'g-', linewidth=2, marker='o', markersize=3, label='Сигнал пульсу')
    ax.set_xlim([0, self.buffer_size])
    ax.set_ylim([0, 1])
    ax.set_title(f'Пульс: {self.get_bpm()} уд/хв', fontsize=10)
    ax.set_xlabel('Зразок', fontsize=8)
    ax.set_ylabel('Інтенсивність', fontsize=8)
    ax.grid(which='both', linestyle='--', linewidth=0.5)
    ax.set_facecolor('#f0f0f0')
    ax.axhline(y=0.2, color='blue', linestyle='--', alpha=0.5, label='Норма (50–100)')
    ax.axhline(y=0.8, color='blue', linestyle='--', alpha=0.5)
    ax.legend(fontsize=8, loc='upper right')
    ax.tick_params(labelsize=8)
    fig.tight_layout()

    canvas = FigureCanvasAgg(fig)
    canvas.draw()
    graph = np.array(canvas.renderer.buffer_rgba())
    plt.close(fig)

    self.signal_graph = cv2.cvtColor(graph, cv2.COLOR_RGBA2BGR)

def update_fft_graph(self):
    """Оновити графік частотного спектра (на основі зеленого каналу)."""
    if len(self.data_buffer_green) < 2:
        return

    fig = plt.figure(figsize=(7.2, 2), dpi=100)
    ax = fig.add_subplot(111)

    normalized_data = np.array(self.data_buffer_green)
    normalized_data = (normalized_data - np.mean(normalized_data)) / np.std(normalized_data)
    fft_data = np.abs(np.fft.rfft(normalized_data))
    freqs = np.fft.rfftfreq(len(normalized_data), 1.0/self.fps)

    ax.plot(freqs*60, fft_data, 'b-', linewidth=2)
    ax.set_xlim([50, 180])
    ax.set_title('Частотний спектр', fontsize=10)
    ax.set_xlabel('Частота (уд/хв)', fontsize=8)
    ax.set_ylabel('Амплітуда', fontsize=8)
    ax.grid(True)
    ax.tick_params(labelsize=8)
    fig.tight_layout()

    canvas = FigureCanvasAgg(fig)
    canvas.draw()
```

```
        graph = np.array(canvas.renderer.buffer_rgba())
        plt.close(fig)

        self.fft_graph = cv2.cvtColor(graph, cv2.COLOR_RGBA2BGR)

    def update_trend_graph(self):
        """Оновити графік тренду пульсу."""
        if len(self.bpm_trend) < 2:
            return

        fig = plt.figure(figsize=(7.2, 2), dpi=100)
        ax = fig.add_subplot(111)

        relative_times = [(t - self.bpm_times[0]) for t in self.bpm_times]

        ax.plot(relative_times, list(self.bpm_trend), 'r-', linewidth=2)
        ax.set_title('Тренд пульсу', fontsize=10)
        ax.set_xlabel('Час (с)', fontsize=8)
        ax.set_ylabel('Пульс (уд/хв)', fontsize=8)
        ax.grid(True)
        ax.tick_params(labelsize=8)
        fig.tight_layout()

        canvas = FigureCanvasAgg(fig)
        canvas.draw()
        graph = np.array(canvas.renderer.buffer_rgba())
        plt.close(fig)

        self.trend_graph = cv2.cvtColor(graph, cv2.COLOR_RGBA2BGR)

    def save_graph(self, graph_type, temp_dir):
        """Зберегти графік як PNG."""
        timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
        graph = getattr(self, f"{graph_type}_graph")
        if graph.shape[0] > 0:
            png_path = temp_dir / f"{graph_type}_graph_{timestamp}.png"
            try:
                cv2.imwrite(str(png_path), graph)
                safe_log(self.log_file, f"{datetime.now()}: Saved {graph_type} graph to {png_path}\n")
                return png_path
            except Exception as e:
                safe_log(self.log_file, f"{datetime.now()}: Failed to save {graph_type} graph: {str(e)}\n")
        return None

    def export_all_data(self):
        """Експортувати всі дані у PDF у директорію програми за допомогою matplotlib."""
        timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
        log_file = PROGRAM_DIR / "export_log.txt"

        try:
```

```
            # Створюємо тимчасову папку в PROGRAM_DIR
            temp_dir = PROGRAM_DIR / "export_temp"
            temp_dir.mkdir(exist_ok=True)

            # Перевірка прав доступу
            if not os.access(temp_dir, os.W_OK):
                safe_log(log_file, f"{datetime.now()}: No write permissions to {temp_dir}.\n")
                raise Exception(f"No write permissions to {temp_dir}.")

            # Збереження графіків
            signal_path = self.save_graph("signal", temp_dir)
            fft_path = self.save_graph("fft", temp_dir)
            trend_path = self.save_graph("trend", temp_dir)

            # Створюємо PDF
            pdf_path = PROGRAM_DIR / f"pulse_data_{timestamp}.pdf"
            with PdfPages(pdf_path) as pdf:
                # Сторінка 1: Заголовок і таблиця даних
                fig = plt.figure(figsize=(8.3, 11.7))  # A4 у дюймах
                plt.suptitle(f"Звіт про вимірювання пульсу - {timestamp}", fontsize=16)

                # Опис
                plt.text(0.1, 0.9, "Цей звіт містить дані вимірювання пульсу, отримані за допомогою веб-камери,
включаючи значення BPM та інтенсивність RGB-каналів, а також відповідні графіки.", fontsize=10,
wrap=True)

                # Таблиця з даними
                if self.csv_data:
                    table_data = [['Час', 'Пульс (уд/хв)', 'Червоний', 'Зелений', 'Синій']]
                    for row in self.csv_data[-50:]:  # Обмежуємо до останніх 50 записів
                        timestamp, bpm, red, green, blue = row
                        table_data.append([timestamp, f"{bpm:.2f}", f"{red:.1f}", f"{green:.1f}", f"{blue:.1f}"])

                    table = plt.table(cellText=table_data, loc='center', cellLoc='center', colWidths=[0.3, 0.15, 0.15, 0.15,
0.15])
                    table.auto_set_font_size(False)
                    table.set_fontsize(8)
                    table.scale(1.2, 1.2)

                plt.axis('off')
                pdf.savefig(fig)
                plt.close(fig)

                # Сторінки для графіків
                for graph_path, title in [
                    (signal_path, "Графік сигналу"),
                    (fft_path, "Графік частотного спектра"),
                    (trend_path, "Графік тренду пульсу")
                ]:
                    if graph_path and os.path.exists(graph_path):
```

```
                    fig = plt.figure(figsize=(8.3, 11.7))
                    img = plt.imread(graph_path)
                    plt.imshow(img)
                    plt.title(title, fontsize=14)
                    plt.axis('off')
                    pdf.savefig(fig)
                    plt.close(fig)
                else:
                    safe_log(log_file, f"{datetime.now()}: Graph not found: {graph_path}\n")

            safe_log(log_file, f"{datetime.now()}: PDF exported to {pdf_path}\n")

        except Exception as e:
            safe_log(log_file, f"{datetime.now()}: Exception during PDF export: {str(e)}\n")
            safe_log(log_file, "Traceback:\n" + "".join(traceback.format_exc()) + "\n")
            raise

        finally:
            # Очистка тимчасової папки
            shutil.rmtree(temp_dir, ignore_errors=True)
            safe_log(log_file, f"{datetime.now()}: Cleaned up temporary directory\n")

        if self.logging_enabled:
            self.stop_logging()

def put_text_with_pil(img, text, position, font_size=20, color=(255, 255, 255), outline_color=(0, 0, 0)):
    """Додати текст до зображення з контуром."""
    img_pil = Image.fromarray(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    draw = ImageDraw.Draw(img_pil)

    try:
        font = ImageFont.truetype("arial.ttf", font_size)
    except Exception:
        font = ImageFont.load_default()

    outline_offset = 1
    for x_offset in [-outline_offset, 0, outline_offset]:
        for y_offset in [-outline_offset, 0, outline_offset]:
            if x_offset != 0 or y_offset != 0:
                draw.text((position[0] + x_offset, position[1] + y_offset), text, font=font, fill=outline_color)

    draw.text(position, text, font=font, fill=color)
    img_cv = cv2.cvtColor(np.array(img_pil), cv2.COLOR_RGB2BGR)
    return img_cv

class WebcamThread(QtCore.QObject):
    frame_signal = QtCore.pyqtSignal(np.ndarray, np.ndarray, int, np.ndarray, np.ndarray, np.ndarray)
    error_signal = QtCore.pyqtSignal(str)

    def __init__(self):
```

```
        super().__init__()
        self.running = False
        self.cap = None
        self.pulse_detector = None
        self.log_file = PROGRAM_DIR / "export_log.txt"

    def try_open_camera(self):
        """Спробувати відкрити камеру з різними бекендами."""
        backends = [
            (cv2.CAP_MSMF, "CAP_MSMF"),  # Спочатку пробуємо MSMF
            (cv2.CAP_DSHOW, "CAP_DSHOW"),
            (cv2.CAP_ANY, "CAP_ANY")
        ]

        for index in range(5):  # Спробуємо індекси 0–4
            for backend, backend_name in backends:
                with warnings.catch_warnings(record=True) as w:
                    warnings.simplefilter("always")
                    cap = cv2.VideoCapture(index, backend)
                    for warning in w:
                        safe_log(self.log_file, f"{datetime.now()}: OpenCV Warning: {str(warning.message)}\n")
                    if cap.isOpened():
                        safe_log(self.log_file, f"{datetime.now()}: Successfully opened webcam with index={index},
backend={backend_name}\n")
                        return cap, backend_name
                    cap.release()

        safe_log(self.log_file, f"{datetime.now()}: Failed to open webcam with any index or backend.\n")
        return None, None

    def run(self):
        """Основна логіка потоку веб-камери."""
        try:
            self.cap, backend_name = self.try_open_camera()
            if not self.cap:
                self.error_signal.emit("Не вдалося відкрити камеру. Перевірте підключення, драйвери або
закрийте інші програми.")
                return

            # Логування роздільної здатності камери
            width = int(self.cap.get(cv2.CAP_PROP_FRAME_WIDTH))
            height = int(self.cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
            safe_log(self.log_file, f"{datetime.now()}: Camera resolution: {width}x{height}\n")

            self.cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)

            fps = self.cap.get(cv2.CAP_PROP_FPS)
            if fps <= 0 or fps > 120:
                fps = 30
```

```
        self.pulse_detector = PulseDetector(fps=fps)

        cascade_path = Path(resource_path("haarcascades/haarcascade_frontalface_default.xml"))
        if not cascade_path.exists():
            cascade_path = Path(cv2.data.haarcascades) / "haarcascade_frontalface_default.xml"
            if not cascade_path.exists():
                self.error_signal.emit("Не вдалося знайти haarcascade_frontalface_default.xml")
                return

        face_cascade = cv2.CascadeClassifier(str(cascade_path))
        if face_cascade.empty():
            self.error_signal.emit("Не вдалося завантажити haarcascade_frontalface_default.xml")
            return

        start_time = time.time()
        bpm = 0

        while self.running:
            ret, frame = self.cap.read()
            if not ret or frame is None:
                frame = np.zeros((480, 640, 3), dtype=np.uint8)
                frame = put_text_with_pil(frame, "Помилка камери", (50, 240), font_size=24, color=(255, 0, 0),
outline_color=(255, 255, 255))
                self.frame_signal.emit(frame, frame, bpm, self.pulse_detector.signal_graph,
self.pulse_detector.fft_graph, self.pulse_detector.trend_graph)
                QtCore.QThread.msleep(30)
                continue

            frame_pulse = frame.copy()

            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            faces = face_cascade.detectMultiScale(gray, 1.2, 5, minSize=(100, 100))

            safe_log(self.log_file, f"{datetime.now()}: Faces detected: {len(faces)}\n")

            face_detected = False
            face_roi = None

            for (x, y, w, h) in faces:
                if not face_detected:
                    face_detected = True
                    face_roi = frame[y:y+h, x:x+w]

                    # Обчислення середніх значень RGB-каналів
                    mean_red = np.mean(face_roi[:, :, 2])
                    mean_green = np.mean(face_roi[:, :, 1])
                    mean_blue = np.mean(face_roi[:, :, 0])
                    rgb_intensity = [mean_red, mean_green, mean_blue]
```

```
                bpm, signal_graph, fft_graph, trend_graph = self.pulse_detector.update(rgb_intensity,
face_detected)

                color_overlay = np.zeros_like(face_roi, dtype=np.float32)
                alpha = 0.5

                if bpm > 0:
                    if bpm < 60:
                        blue_intensity = max(0, min(1.0, 1.0 - bpm/60.0))
                        color_overlay[:, :, 0] = 255 * blue_intensity
                        alpha = 0.3 + blue_intensity * 0.3
                    elif bpm > 100:
                        red_intensity = min(1.0, (bpm - 100) / 80.0)
                        color_overlay[:, :, 2] = 255 * red_intensity
                        alpha = 0.3 + red_intensity * 0.3
                    else:
                        norm_pulse = (bpm - 60) / 40.0
                        blue_intensity = max(0, 1.0 - norm_pulse * 2.0)
                        green_intensity = 1.0 - abs(norm_pulse - 0.5) * 2.0
                        red_intensity = max(0, (norm_pulse * 2.0) - 1.0)
                        color_overlay[:, :, 0] = 255 * blue_intensity
                        color_overlay[:, :, 1] = 255 * green_intensity
                        color_overlay[:, :, 2] = 255 * red_intensity

                face_colored = cv2.addWeighted(face_roi, 1.0 - alpha, color_overlay.astype(np.uint8), alpha, 0)
                frame_pulse[y:y+h, x:x+w] = face_colored

                if bpm < 60:
                    rect_color = (255, 0, 0)
                    pulse_level_text = "НИЗЬКИЙ ПУЛЬС"
                elif bpm > 100:
                    rect_color = (0, 0, 255)
                    pulse_level_text = "ВИСОКИЙ ПУЛЬС"
                else:
                    rect_color = (0, 255, 0)
                    pulse_level_text = "НОРМАЛЬНИЙ ПУЛЬС"

                cv2.rectangle(frame_pulse, (x, y), (x+w, y+h), rect_color, 2)
                frame_pulse = put_text_with_pil(frame_pulse, pulse_level_text, (x, y-35), font_size=18,
color=rect_color, outline_color=(255, 255, 255))
                pulse_text = f"Пульс: {bpm:.2f}"
                cv2.rectangle(frame_pulse, (x, y-15), (x+200, y), (0, 0, 0), -1)  # Black background
                frame_pulse = put_text_with_pil(frame_pulse, pulse_text, (x, y-15), font_size=18, color=(255,
255, 255), outline_color=(0, 0, 0))

            if not face_detected:
                bpm, signal_graph, fft_graph, trend_graph = self.pulse_detector.update([0.0, 0.0, 0.0], False)
                frame_pulse = put_text_with_pil(frame_pulse, "Обличчя не виявлено", (30, 30), font_size=20,
color=(255, 255, 0), outline_color=(255, 255, 255))
                if bpm == 0:
```

```
                elapsed_time = int(time.time() - start_time)
                if elapsed_time < 15:
                    frame_pulse = put_text_with_pil(frame_pulse, f"Виявлення... {elapsed_time}s", (30, 60),
font_size=20, color=(255, 255, 0), outline_color=(255, 255, 255))
                else:
                    frame_pulse = put_text_with_pil(frame_pulse, "Не вдалося виявити пульс", (30, 75),
font_size=20, color=(255, 0, 0), outline_color=(255, 255, 255))

            self.frame_signal.emit(frame, frame_pulse, bpm, signal_graph, fft_graph, trend_graph)
            QtCore.QThread.msleep(30)

        self.pulse_detector.stop_logging()
        if self.cap and self.cap.isOpened():
            self.cap.release()
            self.cap = None
            safe_log(self.log_file, f"{datetime.now()}: Webcam released\n")

    except Exception as e:
        safe_log(self.log_file, f"{datetime.now()}: Exception in WebcamThread.run: {str(e)}\n")
        safe_log(self.log_file, "Traceback:\n" + "".join(traceback.format_exc()) + "\n")
        self.error_signal.emit(f"Помилка потоку камери: {str(e)}")

  def start(self):
    self.running = True
    threading.Thread(target=self.run, daemon=True).start()

  def stop(self):
    self.running = False
    if self.cap and self.cap.isOpened():
        self.cap.release()
        self.cap = None
        safe_log(self.log_file, f"{datetime.now()}: Webcam stopped\n")

class MainWindow(QtWidgets.QMainWindow):
  def __init__(self):
    super().__init__()
    self.setWindowTitle("Детектор пульсу")
    self.resize(1280, 720)

    self.central_widget = QtWidgets.QWidget()
    self.setCentralWidget(self.central_widget)
    self.main_layout = QtWidgets.QVBoxLayout(self.central_widget)

    self.camera_layout = QtWidgets.QHBoxLayout()
    self.main_layout.addLayout(self.camera_layout)

    self.normal_label = QtWidgets.QLabel()
    self.normal_label.setMinimumSize(640, 480)
    self.camera_layout.addWidget(self.normal_label)
```

```
        self.pulse_label = QtWidgets.QLabel()
        self.pulse_label.setMinimumSize(640, 480)
        self.camera_layout.addWidget(self.pulse_label)

        self.control_layout = QtWidgets.QVBoxLayout()
        self.camera_layout.addLayout(self.control_layout)

        self.bpm_label = QtWidgets.QLabel("Пульс: 0")
        self.bpm_label.setFont(QtGui.QFont("Arial", 16))
        self.control_layout.addWidget(self.bpm_label)

        self.start_stop_button = QtWidgets.QPushButton("Почати")
        self.start_stop_button.clicked.connect(self.toggle_webcam)
        self.control_layout.addWidget(self.start_stop_button)

        self.log_button = QtWidgets.QPushButton("Розпочати запис")
        self.log_button.clicked.connect(self.toggle_logging)
        self.control_layout.addWidget(self.log_button)

        self.export_signal_button = QtWidgets.QPushButton("Експортувати графік сигналу")
        self.export_signal_button.clicked.connect(lambda: self.webcam_thread.pulse_detector.save_graph("signal",
PROGRAM_DIR))
        self.control_layout.addWidget(self.export_signal_button)

        self.export_fft_button = QtWidgets.QPushButton("Експортувати графік спектра")
        self.export_fft_button.clicked.connect(lambda: self.webcam_thread.pulse_detector.save_graph("fft",
PROGRAM_DIR))
        self.control_layout.addWidget(self.export_fft_button)

        self.export_trend_button = QtWidgets.QPushButton("Експортувати графік тренду")
        self.export_trend_button.clicked.connect(lambda: self.webcam_thread.pulse_detector.save_graph("trend",
PROGRAM_DIR))
        self.control_layout.addWidget(self.export_trend_button)

        self.export_all_button = QtWidgets.QPushButton("Експортувати PDF")
        self.export_all_button.clicked.connect(self.export_all_data)
        self.control_layout.addWidget(self.export_all_button)

        self.help_button = QtWidgets.QPushButton("Довідка")
        self.help_button.clicked.connect(self.show_help)
        self.control_layout.addWidget(self.help_button)

        self.control_layout.addStretch()

        self.tabs = QtWidgets.QTabWidget()
        self.main_layout.addWidget(self.tabs)

        self.signal_label = QtWidgets.QLabel()
        self.signal_label.setFixedSize(720, 200)
        self.fft_label = QtWidgets.QLabel()
```

```
        self.fft_label.setFixedSize(720, 200)
        self.trend_label = QtWidgets.QLabel()
        self.trend_label.setFixedSize(720, 200)

        self.tabs.addTab(self.signal_label, "Сигнал")
        self.tabs.addTab(self.fft_label, "Спектр")
        self.tabs.addTab(self.trend_label, "Тренд пульсу")

        self.webcam_thread = WebcamThread()
        self.webcam_thread.frame_signal.connect(self.update_frames)
        self.webcam_thread.error_signal.connect(self.show_error)

        self.is_running = False
        self.is_logging = False

    def toggle_webcam(self):
        if not self.is_running:
            self.webcam_thread.start()
            self.start_stop_button.setText("Закінчити")
            self.is_running = True
            safe_log(PROGRAM_DIR / "export_log.txt", f"{datetime.now()}: Webcam thread started\n")
        else:
            self.webcam_thread.stop()
            self.start_stop_button.setText("Почати")
            self.is_running = False
            self.is_logging = False
            self.log_button.setText("Розпочати запис")
            safe_log(PROGRAM_DIR / "export_log.txt", f"{datetime.now()}: Webcam thread stopped\n")

    def toggle_logging(self):
        if not self.is_running:
            QtWidgets.QMessageBox.critical(self, "Помилка", "Увімкніть камеру спочатку!")
            return

        if not self.is_logging:
            self.webcam_thread.pulse_detector.start_logging()
            self.log_button.setText("Закінчити запис")
            self.is_logging = True
        else:
            self.webcam_thread.pulse_detector.stop_logging()
            self.log_button.setText("Розпочати запис")
            self.is_logging = False

    def scale_frame(self, frame, target_width, target_height):
        """Масштабувати кадр до цільового розміру, зберігаючи пропорцію."""
        h, w = frame.shape[:2]
        if h == 0 or w == 0:
            return frame

        # Обчислюємо співвідношення сторін
```

```
    aspect_ratio = w / h
    target_aspect = target_width / target_height

    if aspect_ratio > target_aspect:
        # Кадр ширший, ніж цільовий розмір
        new_w = int(target_height * aspect_ratio)
        new_h = target_height
    else:
        # Кадр вищий, ніж цільовий розмір
        new_w = target_width
        new_h = int(target_width / aspect_ratio)

    # Масштабуємо кадр
    frame = cv2.resize(frame, (new_w, new_h), interpolation=cv2.INTER_AREA)

    # Обрізаємо або додаємо рамки, якщо потрібно
    if new_w > target_width:
        start_x = (new_w - target_width) // 2
        frame = frame[:, start_x:start_x + target_width]
    elif new_h > target_height:
        start_y = (new_h - target_height) // 2
        frame = frame[start_y:start_y + target_height]

    if new_w < target_width:
        left = (target_width - new_w) // 2
        right = target_width - new_w - left
        frame = cv2.copyMakeBorder(frame, 0, 0, left, right, cv2.BORDER_CONSTANT, value=[0, 0, 0])
    if new_h < target_height:
        top = (target_height - new_h) // 2
        bottom = target_height - new_h - top
        frame = cv2.copyMakeBorder(frame, top, bottom, 0, 0, cv2.BORDER_CONSTANT, value=[0, 0, 0])

    if frame.shape[:2] != (target_height, target_width):
        frame = cv2.resize(frame, (target_width, target_height), interpolation=cv2.INTER_AREA)

    return frame

def update_frames(self, frame_normal, frame_pulse, bpm, signal_graph, fft_graph, trend_graph):
    """Оновити відображення кадрів."""
    safe_log(PROGRAM_DIR / "export_log.txt", f"{datetime.now()}: Updating frames, bpm={bpm}\n")

    # Масштабуємо кадри до розміру віджетів
    target_width, target_height = self.normal_label.width(), self.normal_label.height()
    frame_normal = self.scale_frame(frame_normal, target_width, target_height)
    frame_pulse = self.scale_frame(frame_pulse, target_width, target_height)

    frame_normal = cv2.cvtColor(frame_normal, cv2.COLOR_BGR2RGB)
    h, w, c = frame_normal.shape
    qimage_normal = QtGui.QImage(frame_normal.data, w, h, w * c, QtGui.QImage.Format_RGB888)
    self.normal_label.setPixmap(QtGui.QPixmap.fromImage(qimage_normal))
```

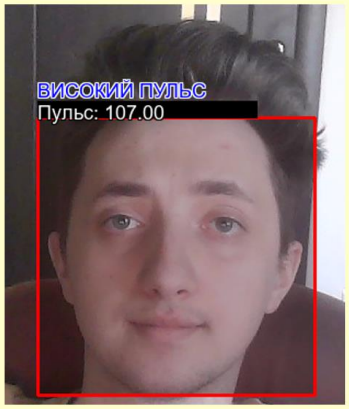
```
        frame_pulse = cv2.cvtColor(frame_pulse, cv2.COLOR_BGR2RGB)
        qimage_pulse = QtGui.QImage(frame_pulse.data, w, h, w * c, QtGui.QImage.Format_RGB888)
        self.pulse_label.setPixmap(QtGui.QPixmap.fromImage(qimage_pulse))

        self.bpm_label.setText(f"Пульс: {bpm:.2f}")

        for graph, label in zip([signal_graph, fft_graph, trend_graph], [self.signal_label, self.fft_label,
self.trend_label]):
            if graph.shape[0] > 0:
                graph = cv2.cvtColor(graph, cv2.COLOR_BGR2RGB)
                gh, gw, gc = graph.shape
                qimage = QtGui.QImage(graph.data, gw, gh, gw * gc, QtGui.QImage.Format_RGB888)
                label.setPixmap(QtGui.QPixmap.fromImage(qimage).scaled(720, 200))

    def export_all_data(self):
        if not self.is_running:
            QtWidgets.QMessageBox.critical(self, "Помилка", "Увімкніть камеру спочатку!")
            return

        try:
            self.webcam_thread.pulse_detector.export_all_data()
            QtWidgets.QMessageBox.information(self, "Успіх", "Дані успішно експортовано до PDF!")
            safe_log(PROGRAM_DIR / "export_log.txt", f"{datetime.now()}: Export PDF successful\n")
        except Exception as e:
            safe_log(PROGRAM_DIR / "export_log.txt", f"{datetime.now()}: Export PDF failed: {e}\n")
            QtWidgets.QMessageBox.critical(self, "Помилка", f"Не вдалося експортувати дані: {str(e)}")

    def show_error(self, message):
        QtWidgets.QMessageBox.critical(self, "Помилка", message)

    def show_help(self):
        """Показати діалог довідки з інформацією про програму."""
        safe_log(PROGRAM_DIR / "export_log.txt", f"{datetime.now()}: Help dialog opened\n")
        help_text = (
            "Назва програми: Детектор пульсу\n\n"
            "Опис: Програма призначена для вимірювання пульсу людини за допомогою веб-камери. "
            "Вона аналізує зміни RGB-каналів у зображенні обличчя для визначення частоти серцевих
скорочень (BPM). "
            "Програма відображає відеопотік, графіки сигналу, спектра та тренду пульсу, а також дозволяє
експортувати дані у PDF.\n\n"
            "Автор: Присяжнюк Валентин Вадимович, 4ПІ-21б"
        )
        QtWidgets.QMessageBox.information(self, "Довідка", help_text)

    def closeEvent(self, event):
        self.webcam_thread.stop()
        event.accept()

if __name__ == "__main__":
```

```
    try:
        # Перевірка прав доступу
        if not os.access(PROGRAM_DIR, os.W_OK):
            safe_log(PROGRAM_DIR / "export_log.txt", f"{datetime.now()}: No write permissions to
{PROGRAM_DIR}\n")
            print(f"Error: No write permissions to {PROGRAM_DIR}")
            sys.exit(1)

        if os.name == 'nt':
            try:
                import psutil
                p = psutil.Process(os.getpid())
                p.nice(psutil.HIGH_PRIORITY_CLASS)
            except ImportError:
                pass

        app = QtWidgets.QApplication(sys.argv)
        window = MainWindow()
        window.show()
        sys.exit(app.exec_())
    except Exception as e:
        safe_log(PROGRAM_DIR / "export_log.txt", f"{datetime.now()}: Main program exception: {e}\n")
        safe_log(PROGRAM_DIR / "export_log.txt", "Traceback:\n" + "".join(traceback.format_exc()) + "\n")
        print(f"Error: {e}")
        traceback.print_exc()
```

Додаток Г. Графічна частина

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«Комп'ютерна програма для визначення серцевих скорочень на основі аналізу змін кольору обличчя»



Студент гр. 4ПІ-21Б Присяжнюк В.В.
Науковий керівник - д.т.н. проф. Романюк О.Н.

Вінниця – 2025

1

Рисунок Г.1

Галузі застосування програми

- **Сфери використання:**
  - **Медицина:**
    - Попередній скринінг пульсу
    - Моніторинг пацієнтів без контакту
    - Телемедицина та віддалені консультації
  - **Спорт та фітнес:**
    - Контроль навантаження під час тренувань
    - Моніторинг відновлення після фізичної активності
  - **Дослідження:**
    - Збір даних для медичних досліджень
    - Аналіз впливу стресу на серцевий ритм
  - **Експорт:**
    - Збереження даних в PNG та в PDF форматах
    - Логування всіх вимірювань

2

Рисунок Г.2

Мета і завдання дослідження

- **Метою роботи є автоматизація дистанційного визначення частоти серцебиття по зображенню обличчя людини.**
- **Основними задачами дослідження є:**
- **провести аналіз існуючих методів і систем безконтактного визначення частоти серцебиття;**
- **запропонувати нові:**
- **методи визначення розпізнавання обличчя та виділення зон інтересу для аналізу пульсового сигналу;**
- **моделі обробки відеопотоку для виділення та аналізу змін кольору шкіри, пов'язаних з пульсацією кровотоку;**
- **методи фільтрації шумів та компенсації рухів;**
- **розробити програмні модулі та програму визначення частоти серцебиття на основі запропонованих методів;**
- ***Об'єкт дослідження* – процес розпізнавання та аналізу змін кольору шкіри обличчя для визначення частоти серцебиття.**
- ***Предмет дослідження* – методи та засоби комп'ютерного зору й обробки сигналів для виділення та аналізу пульсового сигналу з відеопотоку.**
- ***Методи дослідження.* В процесі досліджень використовувались: методи цифрової обробки сигналів, комп'ютерного зору, математичної статистики, спектрального аналізу, теорія розпізнавання образів та машинного навчання для розробки моделей та методів вимірювання пульсу; комп'ютерне моделювання для аналізу та перевірки отриманих теоретичних положень..**

3

Рисунок Г.3

Наукова новизна отриманих результатів:

- Вперше запропоновано математичну модель аналізу змін кольору шкіри обличчя, яка враховує умови освітлення та рухи об'єкта для реалізації адаптивної фільтрації сигналу, що дозволило підвищити точність визначення частоти серцевих скорочень до 15%.
- Запропоновано комбінований метод обробки відеосигналу з використанням алгоритмів незалежного компонентного аналізу та адаптивної фільтрації, що дозволяє підвищити точність вимірювання пульсу за рахунок ефективного виділення корисного сигналу з шумів.

- **Практичне значення отриманих результатів** полягає в тому, що на основі отриманих в бакалаврській кваліфікаційної роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби вимірювання пульсу за допомогою веб–камери для систем неінвазивного моніторингу здоров'я. Розроблена комп'ютерна програма може бути використана у домашніх умовах, телемедицині, спортивних тренуваннях та для первинної діагностики стану серцево–судинної системи..

4

Рисунок Г.4

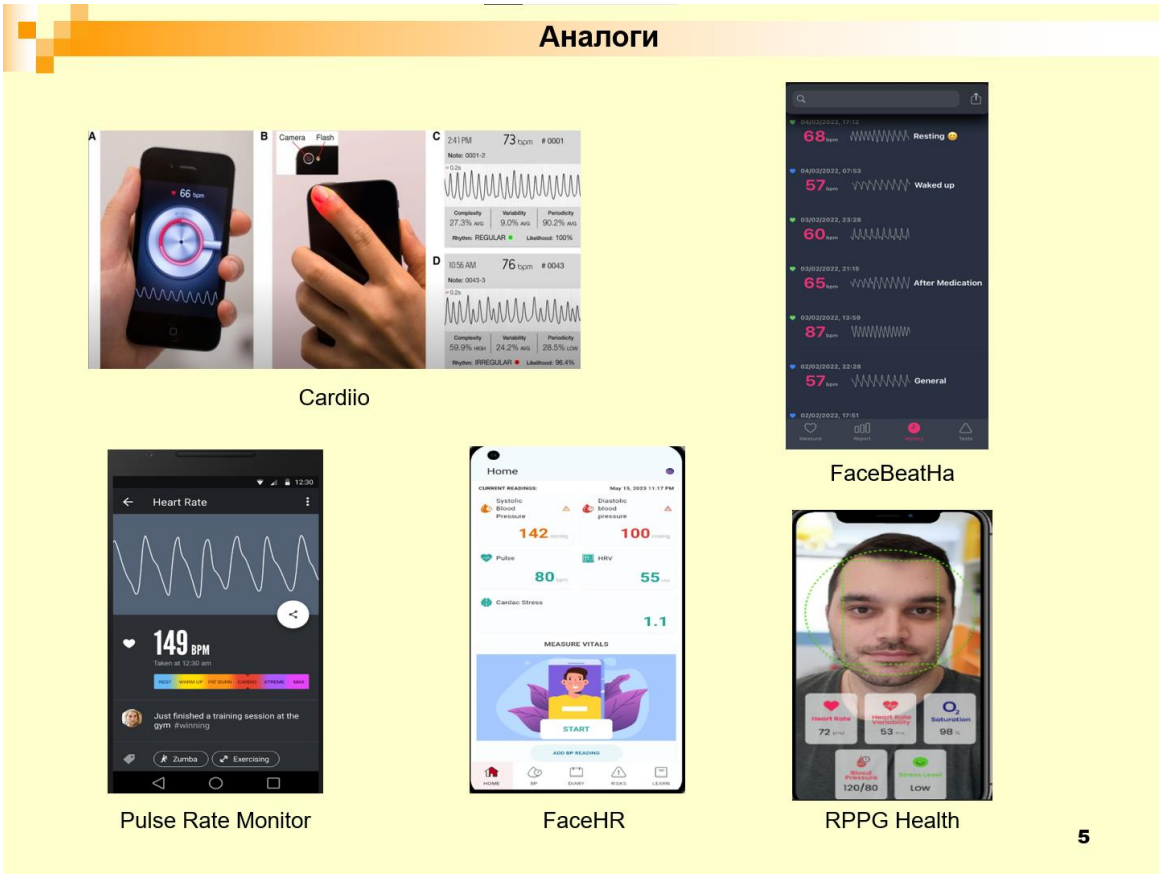


Рисунок Г.5

	Cardiiot	FaceBeat	Pulse Rate Monitor	RPPG Health	Власна розробка
Точність вимірювання пульсу	1	1	1	1	1
Можливість роботи при різному освітленні	0	1	1	1	1
Стійкість до рухів користувачів	0	0	1	0	1
Аналіз варіабельності серцевого ритму	1	1	1	0	1
Візуалізація даних у реальному часі	0	0	0	0	1
Сумарний коефіцієнт	2	3	4	2	5

Рисунок Г.6

Формула роботи математичної моделі для методу аналузі зміни кольору обличчя

1. Усереднення кольору зображення

$$f(t) = \frac{1}{N}\sum_{n=1}^{N} I(x, y, t),$$

де N – кількість пікселів у області ROI.

2. Перетворення Фур'є

$$F(\omega) = \int f(t)e^{(-j\omega t)}dt,$$

де ω – частота, а $F(\omega)$ – амплітуда сигналу на цій частоті.

3. Швидке перетворення Фур'є

$$P(\omega) = |F(\omega)|^2,$$

4. Частота пульсу

$$\omega_{pulse} = argmax(P(\omega))$$

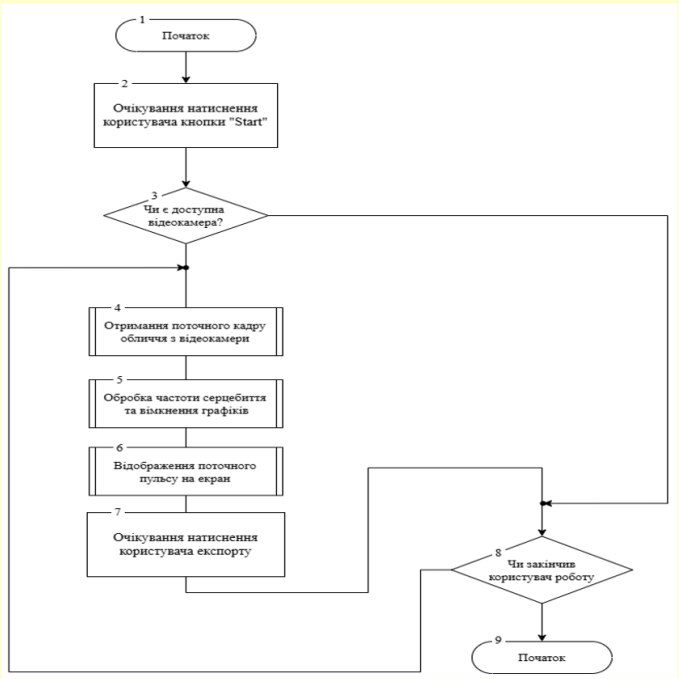
5. Кінцеве обчислення частоти пульсу

$$BPM = 60 \times \omega_{pulse},$$

7

Рисунок Г.7

Блок-схема роботи програмного застосунку



8

Рисунок Г.8

Формули методу фільтрацій сигналу та спектрального аналізу для виділення пульсації

1. Нормалізація сигналу

$$f_{norm}(e) = \frac{(f(t) - \mu)}{\sigma}$$

де μ – середнє значення сигналу, σ – стандартне відхилення.

2. Ідеальний низькочастотний фільтр

$$H(\omega) = \frac{1}{\sqrt{1 + (\frac{\omega}{\omega_c})^{2n}}},$$

де ω_c – частота зрізу, n – порядок фільтра.

3. Фільтрація сигналу в частотній області

$$F_{filtered}(\omega) = F(\omega) \times H(\omega),$$

4. Метод Уелча для оцінки спектральної потужності

$$P_{welch}(\omega) = \frac{1}{K}\sum_{k=1}^{K} |F_k(\omega)|^2,$$

де K – кількість сегментів, а F_k(ω) – перетворення спектру k–го сегмента.

5. Спектральна оцінка методом автокореляційної моделі

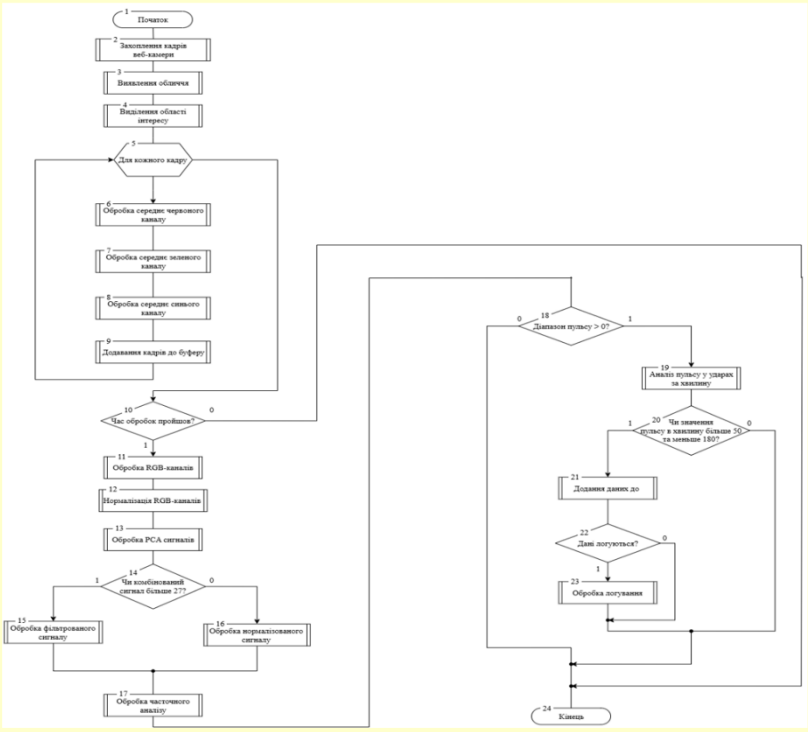
$$P_{AR}(\omega) = \frac{\sigma^2}{|\sum a_k \times e^{(-j\omega k)}|^2},$$

де σ² – дисперсія білого шуму, a_k – коефіцієнти авторегресійної моделі.

9

Рисунок Г.9

Блок-схема алгоритму обробки сигналу пульсу



10

Рисунок Г.10

Формули методу розрахунку частоти серцевих скорочень на основі відеоданих

1. Інтервал між піками

$$\Delta t_i = t_{i+1} - t_i,$$

2. Середній інтервал між серцевими ударами

$$\Delta t_{avg} = \frac{1}{n-1} \times \sum t_i,$$

3. Розрахунок пульсу з часової області

$$BPM_{time} = \frac{60}{\Delta t_{avg}},$$

4. Гібридна модель оцінки серцебиття

$$BPM = \alpha \times BPM_{freq} + (1 - \alpha) \times BPM_{time},$$

11

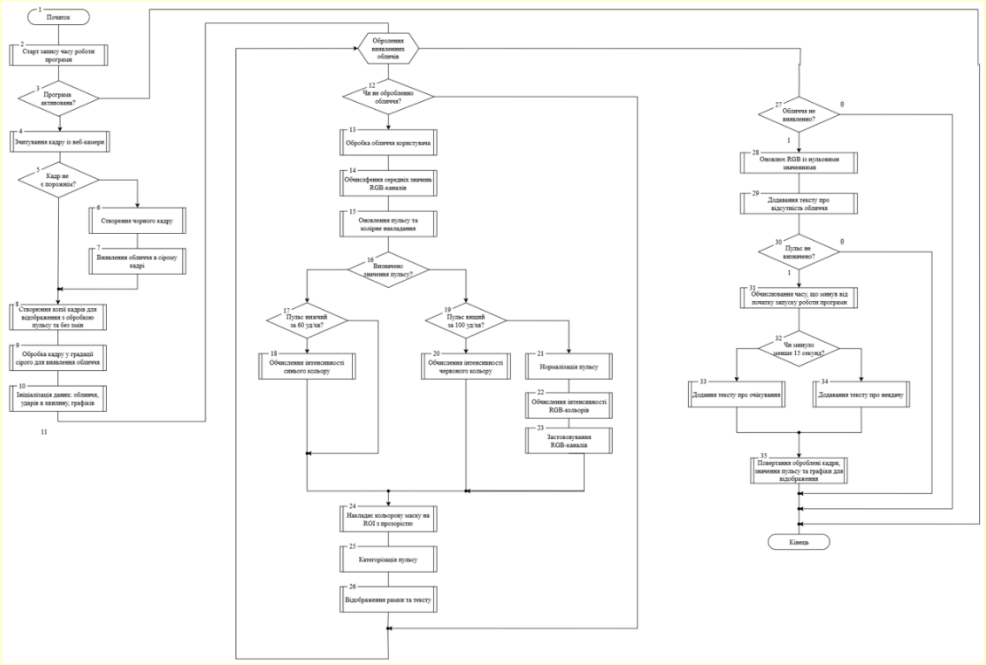
Рисунок Г.11

Блок-схема функції виявлення обличчя

12

Рисунок Г.12

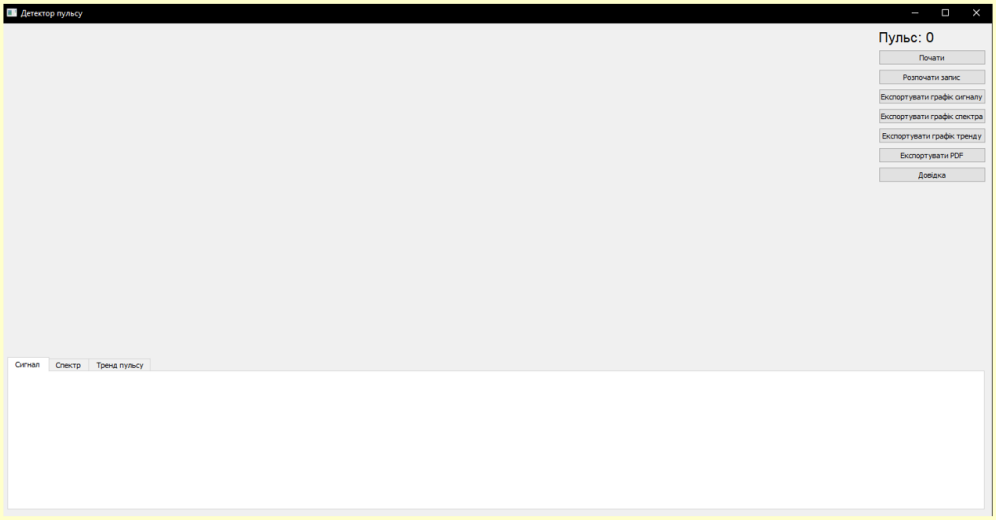
Блок-схема функції аналізу відеопотоку



13

Рисунок Г.13

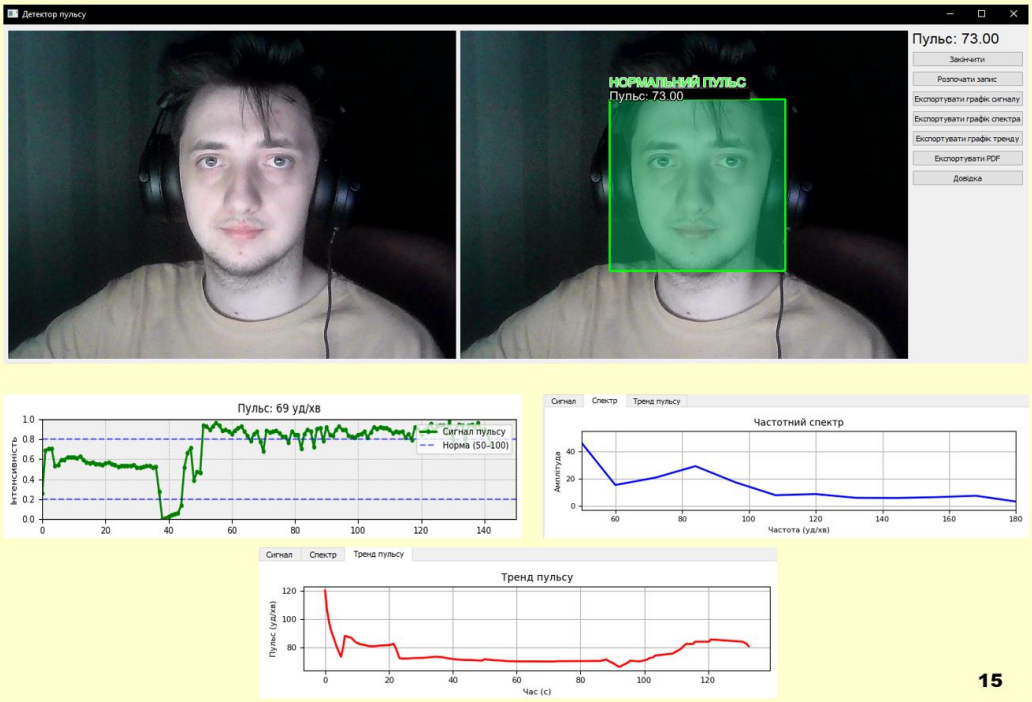
Стартове головного вікна програми



14

Рисунок Г.14

Програма під час початку роботи та її графіки



15

Рисунок Г.15

- **Апробація матеріалів бакалаврської кваліфікаційної роботи.** Описані у роботі положення доповідались на конференціях: «*XXV Всеукраїнської науково–технічної конференції молодих вчених* (ВНТУ, 2025)»; «XXVII Міжнародної науково–технічної конференції «Технологія-2025» (СНУ, 2025)»; «XII Всеукраїнської науково–практичній конференції молодих учених «Інформаційні технології» (КСУ, 2025)».
- **Публікації.** За темою бакалаврської кваліфікаційної роботи надруковано 3–х праць у матеріалах конференціях.

16

Рисунок Г.16

Висновки

- **Було виконано такі задачі:**
- проведено аналіз існуючих методів безконтактного вимірювання частоти серцебиття та обґрунтовано доцільність використання фотоплетизмографії на основі аналізу кольору шкіри обличчя;
- розроблено алгоритм виділення області інтересу (ROI) на обличчі з відеопотоку в реальному часі та побудови часового сигналу пульсації на основі зміни кольорових компонентів зображення;
- реалізовано попередню обробку сигналу, включаючи нормалізацію, фільтрацію та компенсацію впливу освітлення й шуму;
- впроваджено спектральні методи аналізу (швидке перетворення Фур'є, метод Уелча, AR-модель) для визначення частоти серцебиття;
- реалізовано гібридну модель обчислення частоти серцебиття, що поєднує оцінки з частотної та часової областей для підвищення точності результатів;
- створено зручний графічний інтерфейс користувача, який відображає відео, графік сигналу та числові показники частоти серцебиття в реальному часі;
- проведено тестування розробленої програми за різних умов освітлення та позиції обличчя, що підтвердило її працездатність і ефективність.

17

Рисунок Г.17