МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ГРОМОВИЧ ОЛЕКСІЙ СЕРГІЙОВИЧ

Допускається до захисту:
В.о. завідувача кафедри
інформаційних технологій
канд. техн. наук, доцент
_____ Оксана ЗЕЛІНСЬКА
«__»_____2024 р.

ДОСЛІДЖЕННЯ МЕТОДІВ ЗБОРУ, ОБРОБКИ ТА АНАЛІЗУ ДАНИХ З МОБІЛЬНИХ СЕНСОРІВ.

Спеціальність 122 «Комп'ютерні науки»
Кваліфікаційна (магістерська) робота

Керівник:
П.В. Римар, старший викладач
кафедри інформаційних технологій
Науковий консультант:
Ю.С. Антонов, доцент кафедри
інформаційних технологій
к.фіз.-мат.наук, доцент

Оцінка:_____/_____ /_____
(бали/за шкалою ЄКТС/за національною шкалою)
Голова ЕК:_____
(підпис)

Вінниця – 2024

АНОТАЦІЯ

Громович О. С. Дослідження методів збору, обробки та аналізу даних з мобільних сенсорів. Спеціальність 122 «Комп’ютерні науки», Освітньої програми «Комп’ютерні технології обробки даних». Донецький національний університет імені Василя Стуса, Вінниця, 2024.

У кваліфікаційній роботі досліджено основні аспекти збору, обробки та аналізу даних з мобільних сенсорів. Показано різноманітні типи датчиків та методи їх збору, а також основні проблеми та обмеження сучасних систем збору даних. Встановлено, що сучасні методи часто стикаються з високими енергетичними витратами та обчислювальними обмеженнями. Наведено нові підходи та алгоритми для оптимізації цих процесів. Додатково розглянуто розробку додатку на основі сучасних технологій, який демонструє ефективність динамічного збору даних для оптимізації ресурсів мобільних пристроїв.

Ключові слова: мобільні сенсори, методи збору, аналіз даних

Табл. 1. Рис. 19. Бібліограф.:44 найм

ABSTRACT

Hromovych O. Research of methods of collection, processing and analysis of data from mobile sensors. Specialty 122 «Computer Sciences», Programme «Data Science». Vasyl' Stus Donetsk National University, Vinnytsia, 202.

In the qualification work, the main aspects of data collection, processing, and analysis from mobile sensors were investigated. Various types of sensors and their collection methods were demonstrated, as well as the primary challenges and limitations of modern data collection systems. It was established that current methods often face high energy consumption and computational constraints. New approaches and algorithms for optimizing these processes were introduced. Additionally, the development of an application based on modern technologies was discussed, which showcases the effectiveness of dynamic data collection in optimizing mobile device resources.

Keywords: mobile sensors, collection methods, data analysis

Tabl. 1. Fig. 19. Bibliography: 44 items

ЗМІСТ

ВСТУП

Мобільні сенсори, вбудовані у наші смартфони та інші мобільні пристрої, можна порівняти з невидимими помічниками, які діють непомітно в реальному світі. Вони сприймають навколишнє середовище і перетворюють його на цифрові дані, що робить їх надзвичайно важливими. Ці сенсори постійно забезпечують нам потік інформації про наші дії і навколишній світ, відслідковуючи, наприклад, наші рухи за допомогою акселерометрів і визначаючи точну локацію за допомогою GPS-модулів.

Мобільні сенсори перетворили нашу взаємодію з технологіям на невербальний обмін інформацією завдяки їхній потужності та незамітності. Сучасні смартфони та планшети потребують їх, щоб реагувати на дотик, ідентифікувати наші рухи та відтворювати звук. Їхні функції не обмежуються цим, і разом із розвитком технологій їхні можливості розширюються.

Однак чому саме ця тема є важливою? Як сучасні технологічні досягнення та наше повсякденне життя впливають на мобільні сенсори? Яким чином вони сприяють збору та аналізу даних? Це магістерська робота, яка спрямована на дослідження методів збору, обробки та аналізу даних з мобільних сенсорів, має розглянути ці питання. У цьому контексті варто обговорити важливість цієї теми, її актуальність для сучасної інформаційної суспільності та потенційні переваги її досліджень і практичних застосувань.

Актуальність теми, яка стосується методів збору, обробки та аналізу даних з мобільних сенсорів, становить собою ключовий аспект сучасного інформаційного ландшафту. Швидкий розвиток мобільних технологій та зростання кількості смартфонів призвели до значного збільшення обсягу та доступності даних, зібраних за допомогою мобільних сенсорів. Цей феномен обумовлює наступні ключові аргументи, які підкреслюють актуальність теми.

По-перше, мобільні смартфони стали не просто комунікаційними засобами, а повноцінними міні-лабораторіями для збору різноманітної інформації. Вони

оснащені різними видами сенсорів, які дозволяють вимірювати рух, звук, світло, температуру та інші параметри навколишнього середовища. Це робить їх потужними інструментами для збору даних в реальному часі.

По-друге, зростання кількості смартфонів викликає необхідність вдосконалення методів збору та аналізу даних. Переважна більшість користувачів не лише споживають інформацію, але і створюють її. Вони залишають цифровий слід у вигляді фотографій, відео, текстових повідомлень та інших даних, які можна аналізувати для розуміння їхньої поведінки, звичок і вподобань.

По-третє, застосування мобільних сенсорів не обмежується лише особистими смартфонами. Вони використовуються в різних сферах, таких як медицина (наприклад, для моніторингу здоров'я пацієнтів), транспорт (для покращення систем навігації та безпеки на дорогах), спорт (для трекінгу фізичної активності), наука (для збору даних у дослідженнях) та інші.

Отже, швидкий розвиток мобільних технологій та зростання кількості смартфонів створюють актуальний контекст, в якому ефективні методи збору та аналізу даних з мобільних сенсорів стають необхідними. Подальший розвиток цієї технологічної галузі вимагає наукових досліджень, щоб оптимізувати збір та використання цих даних у різних аспектах життя сучасного суспільства.

Об'єкт дослідження цієї магістерської роботи - це процеси та явища, пов'язані з збором, обробкою та аналізом даних з мобільних сенсорів. Дослідження спрямоване на покращення методів цього процесу та визначення можливостей використання отриманих даних у практичних задачах, що має важливий практичний і науковий інтерес.

Предметом дослідження цієї магістерської роботи є конкретні аспекти методів збору, обробки та аналізу даних, які отримуються за допомогою мобільних сенсорів. Робота спрямована на розгляд різних аспектів цих методів, включаючи алгоритми збору та передачі даних, обробку і фільтрацію сигналів, а також статистичний та математичний аналіз зібраних даних.

Мета дослідження: Дослідити методи збору, обробки та аналізу даних з мобільних сенсорів з метою вдосконалення та оптимізації їхнього використання у практичних задачах.

Перш за все, мета цього дослідження полягає в систематичному аналізі інформації про методи збору, обробки та аналізу даних з мобільних сенсорів. Цей аналіз має на меті визначити поточний стан справ у даній галузі та ідентифікувати можливість покращення та оптимізації існуючих методів.

Далі, мета роботи передбачає розробку нових підходів, методів чи алгоритмів, які можуть бути використані для збору, обробки та аналізу даних з мобільних сенсорів з більшою ефективністю, точністю та надійністю.

Крім того, мета дослідження передбачає вивчення можливостей використання отриманих даних у практичних застосуваннях, таких як розробка нових технологічних продуктів, моніторинг стану здоров'я або покращення процесів прийняття рішень.

Загалом, мета цієї магістерської роботи полягає в тому, щоб внести важливий науковий внесок у розуміння та використання методів збору, обробки та аналізу даних з мобільних сенсорів, а також надати практичні рекомендації та розробити нові підходи для їх вдосконалення та застосування.

Підвищення ефективності застосувань: Дослідження спрямоване на розробку більш ефективних методів використання даних з мобільних сенсорів у різних сферах. Наприклад, це може включати покращення систем моніторингу здоров'я, підвищення точності навігації у транспорті, розвиток інтерактивних додатків та ігор та багато інших застосувань.

Науковий внесок: Дослідження може сприяти розвитку наукових знань у галузі обробки сигналів, інформатики та інших наукових дисциплін. Всі вдосконалення методів та технологій додадуться до загального фонду знань і можуть бути використані в подальших дослідженнях.

Практична користь: В результаті дослідження можуть бути розроблені практичні інструменти та рекомендації для використання даних з мобільних

сенсорів у практичних задачах. Це може допомогти покращити якість життя людей, розвивати нові технології та підвищувати конкурентоспроможність підприємств та організацій.

Отже, дане дослідження є важливим не лише з технічної, але і з наукової та практичної точки зору. Воно спрямоване на вирішення конкретних проблем та відкриття нових можливостей у використанні даних з мобільних сенсорів, що має великий потенціал для покращення сучасного інформаційного середовища.

На основі цієї мети можна сформулювати конкретні завдання дослідження, які будуть послідовними кроками на шляху досягнення цієї мети. Декілька прикладів таких завдань:

1. Вивчити сучасні методи збору даних з мобільних сенсорів: Аналіз існуючих методів для зрозуміння їх переваг і недоліків.
2. Розробити алгоритми обробки даних: Створення нових або вдосконалення існуючих алгоритмів для підвищення точності та надійності зібраних даних.
3. Дослідити можливості використання даних в практиці: Визначення можливих застосувань отриманих даних у конкретних областях, таких як медицина, транспорт, наука та інші.
4. Провести експерименти та аналіз результатів: Виконання практичних експериментів для перевірки та обґрунтування розроблених методів та алгоритмів.
5. Підготовка наукового звіту та рекомендацій: Написання звіту, який міститиме результати дослідження та практичні рекомендації для використання отриманих знань та методів у практичних завданнях.

Ці завдання утворюють послідовність дій, спрямованих на досягнення визначеної мети дослідження і дозволяють систематично вирішувати задачі роботи з мобільними пристроями.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих типів мобільних сенсорів

На сьогодні мобільні пристрої обладнані різними датчиками, такими як GPS, акселерометр, гіроскоп, магнітометр, датчик освітлення, мікрофон і багато інших.(ДОДАТОК А) Ця розмаїтість датчиків дозволяє смартфонам і планшетам здійснювати безліч функцій і забезпечувати користувачам найсучасніші можливості.[1]

GPS (глобальна система позиціонування) дозволяє визначати точний географічний розташування пристрою (рис. 1.1.1), що стало необхідністю для навігації, визначення місця зйомки фотографій і відео, а також використання локаційних сервісів.

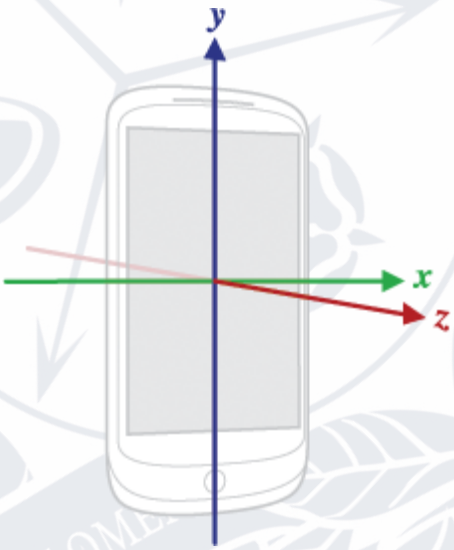

Рис. 1.1.1. Система координат(відносно пристрою)

Акселерометр і гіроскоп служать для визначення орієнтації пристрою в просторі(рис. 1.1.2). Це дозволяє автоматично повертати екран в горизонтальне або вертикальне положення, використовувати жести для керування додатками і групами, а також вдосконалювати віртуальну реальність і доповнену реальність.

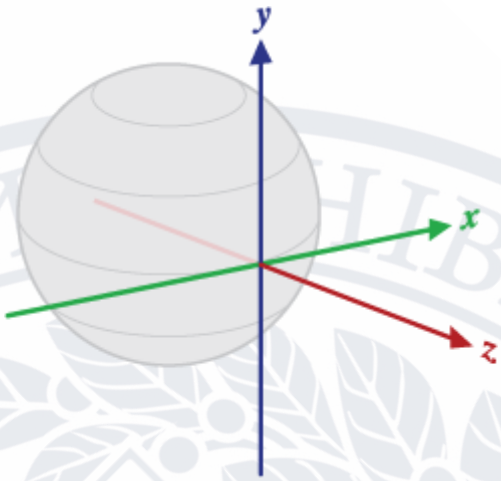

Рисунок 1.1.2 Система координат, датчика вектора обертання

Магнітометр визначає магнітну орієнтацію пристрою і може використовуватися для навігації в місцях, де GPS-сигнал слабкий, а також для розширених реалістичних ігор і додатків.

Датчик освітлення допомагає автоматично регулювати яскравість екрану в залежності від освітлення навколишнього середовища, що зберігає енергію батареї та покращує комфорт користувача.

Мікрофони дозволяють записувати аудіо, використовувати голосові команди і спілкуватися в реальному часі через додатки для обміну повідомленнями та додатки відеодзвінків.

Сенсори серцевого ритму стали важливими для моніторингу здоров'я та фітнес-додатків. Загалом, сенсори серцевого ритму відкривають нові можливості для моніторингу та підтримки здоров'я, роблять мобільні пристрої більш корисними для щоденного використання та допомагають покращувати якість життя користувачів.

Ця інтеграція різних датчиків робить мобільні пристрої більш функціональними і корисними для користувачів у їх повсякденних завданнях і розвагах.

1.2 Дослідження методів збору даних з мобільних сенсорів.

Аналізуючи інформацію з різних джерел можна згрупувати методи за подібними ознаками, сферою використання, елементами взаємодії та масштабом.[2-4] В результаті виділяються наступні групи:

- Пасивний збір даних. Мобільні датчики безперервно збирають дані без дій користувача. Наприклад дані GPS можна реєструвати з часом для відстеження руху користувача.
- Активний збір даних: Користувачі отримують запити на надання даних через певні дії або опитування. Це може включати в себе запит користувачів оцінити свій настрій або зробити фотографії для конкретних дослідницьких цілей.
- Краудсорсинг: Збір даних від великої кількості користувачів або пристроїв. Наприклад, дані про трафік або забруднення можна збирати з різних мобільних пристроїв для створення карт в реальному часі.
- Пристрої з підтримкою носіння: Дослідження також може включати в себе датчики для носіння, такі як фітнес-трекери або смарт-годинники, для збору біометричних даних.

Зазначені методи збору даних мають свої переваги та недоліки. Пасивний збір і пристрої з підтримкою носіння надають можливість неперервного моніторингу даних без активної участі користувача. Однак вони також схильні до збільшеного споживання енергії та залежать від правильності використання. У свою чергу, активний збір даних дозволяє контролювати, коли і які дані збираються, і може надати якісну інформацію завдяки взаємодії з користувачами (рис. 1.2.1). Проте цей метод може бути вимогливим до залучення користувачів та сприйматися суб'єктивним. Краудсорсинг дозволяє збирати дані від багатьох користувачів, але потребує відсіву некоректних інформації та забезпечення приватності. Остаточний вибір методу залежить від конкретних завдань та контексту дослідження.

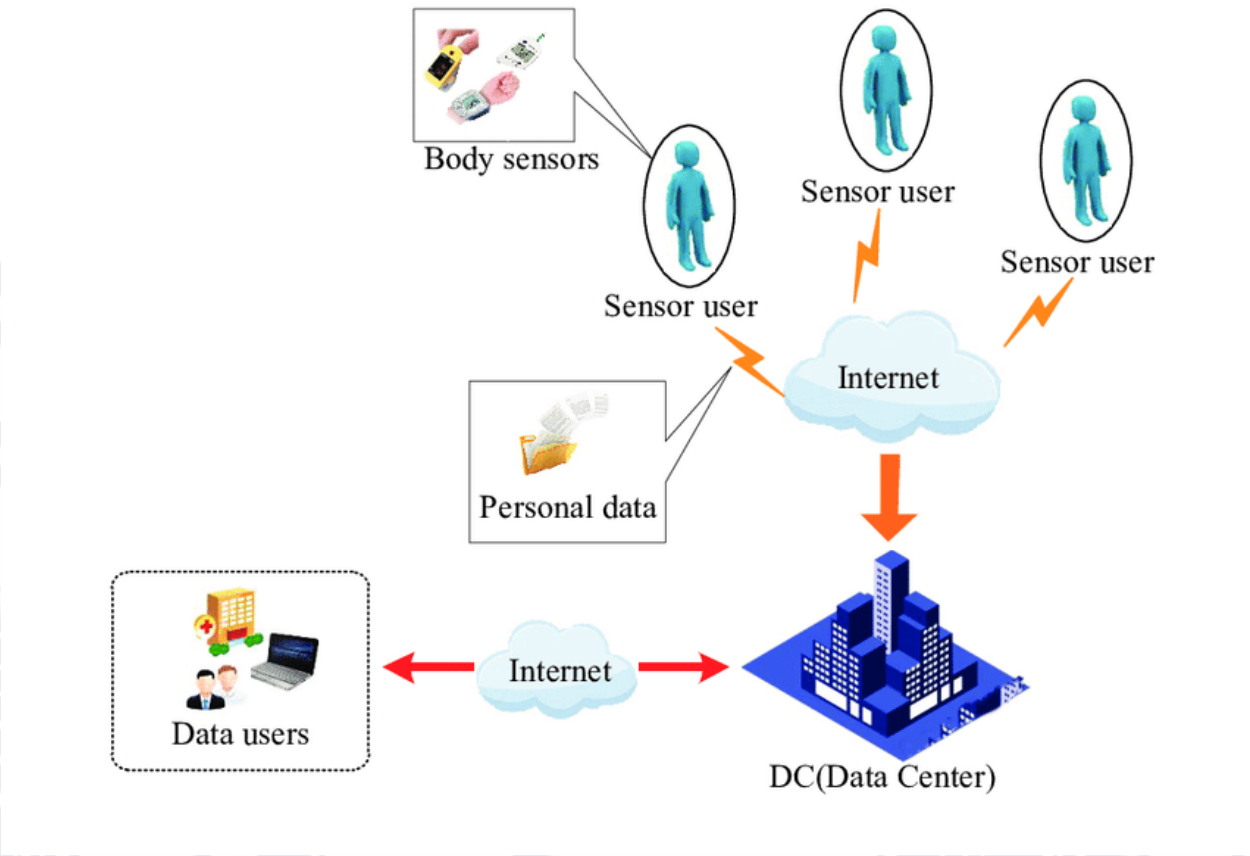

Рисунок 1.2.1 Процес збору даних від користувачів

Взаємодія між цими методами може бути дуже корисною для створення комплексних систем моніторингу та збору даних. Наприклад, можна використовувати активний збір даних для отримання специфічної інформації від користувачів, а пасивний збір для автоматичного надання контексту та додаткової інформації. Краудсорсинг може служити джерелом даних для валідації та розширення обсягу інформації, а пристрої з підтримкою носіння можуть надавати постійний потік біометричних даних.

Також спільною проблемою може стати приватність даних. Збирання без прямого дозволу користувача, небажання розкривати свої дані, безпека даних. Важливо бути обережними із збором та зберіганням особистих даних користувачів, дотримуючи відповідних норм і законів про приватність.

Кожен мобільний пристрій обладнаний різними сенсорами, і кожен з них має свої унікальні можливості та особливості передачі зібраних даних (ДОДАТОК Б).

Потрібно аналізувати їхній принцип роботи, можливості та точність вимірювань, а також досліджувати інтерфейси та API, які надають доступ до сенсорів та зібраних даних.[5]

Точність сенсорів може варіюватися в залежності від виробника та якості пристрою. Для деяких додатків, таких як вимірювання кроків або навігація, точність грає важливу роль

Для доступу до сенсорів та зібраних даних розробники мають доступ до різних інтерфейсів та API. Ці інструменти надають можливість програмно звертатися до сенсорів і обробляти отримані дані. Наприклад, Android надає API для роботи з акселерометром, гіроскопом, GPS і багатьма іншими сенсорами.[6] Як правило ці елементи незалежні від логіки додатків, та знають про пристрої з якими працюють, як це показано на рисунку 1.2.2

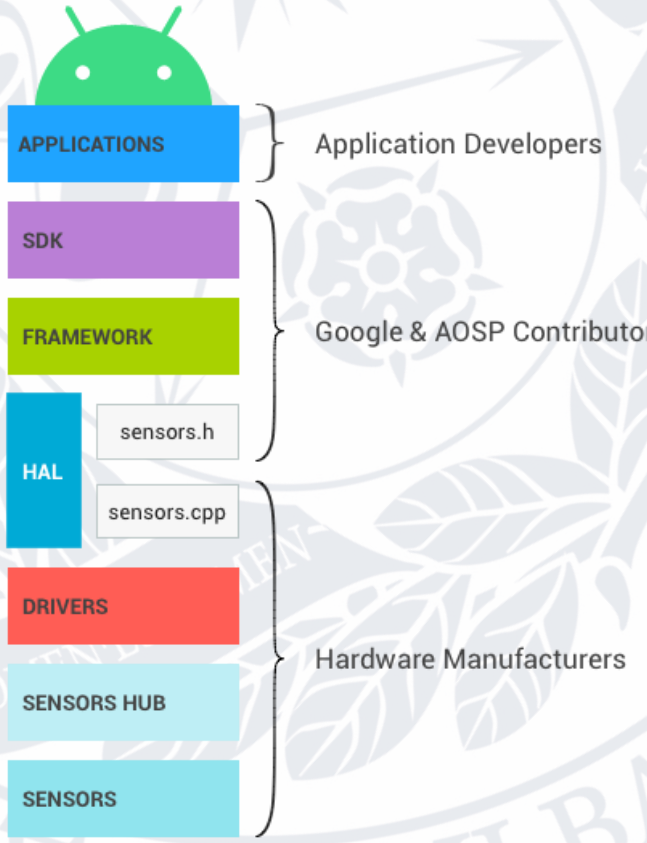

Рисунок 1.2.2 Шари стека Android сенсорів

1.3 Аналіз доступних засобів обробки даних з мобільних сенсорів

Попередня обробка даних (рис. 1.3.1) є важливим етапом в аналізі інформації з мобільних сенсорів. Ось деякі основні аспекти цього процесу[7-9]:

- Очищення даних: Під час очищення даних виконується видалення шуму, викидів та помилкових даних. Це допомагає забезпечити якість зібраних даних і уникнути спотворення результатів аналізу. Наприклад, якщо акселерометр зареєстрував випадкові відлуння, їх можна видалити під час очищення.
- Видобуток ознак: Важливо визначити важливі ознаки з сирої інформації датчиків. Цей процес спрощує дані, зменшує їхню розмірність і полегшує подальший аналіз. Наприклад, з акселерометра можуть бути виділені характеристики, такі як середнє значення або максимальне значення прискорення.
- Об'єднання даних: Поєднання даних з різних датчиків може надати більше повного розуміння оточуючої обстановки або поведінки користувача. Наприклад, комбінування даних з GPS і акселерометра може допомогти визначити швидкість руху та напрямок.

Алгоритми обробки даних використовуються для ряду важливих цілей і завдань, що допомагають отримати корисну інформацію і здійснити раціональне прийняття рішень.[10] Для цього завдання можна використовувати різні алгоритми та методи, в залежності від характеристик даних та потреб аналізу.

Рисунок 1.3.1 Етапи попередньої обробки даних

Для очищення даних можна використовувати різні алгоритми фільтрації[11-12], наприклад:

- Середнє значення (Mean Filter). Цей метод обчислює середнє значення даних в околі кожного точки і замінює цю точку середнім значенням. Використовується для видалення високочастотного шуму з сигналів.
- Медіанний фільтр (Median Filter). Цей метод обчислює медіану значень даних в околі кожного точки і замінює цю точку медіаною. Ефективний для видалення викидів з сигналів.
- Фільтр Калмана (Kalman Filter). Цей рекурсивний фільтр використовує прогноз і корекцію для обробки даних. Він оцінює стан системи і робить прогнози на основі попередніх даних та коригує їх на основі нових вимірювань. Зазвичай використовується для фільтрації сигналів, що мають деякий рівень шуму.
- Loess-регресія (Locally Weighted Scatterplot Smoothing). Цей метод застосовується для наближення функції залежності між змінними у декількох точках. Він використовує вагові коефіцієнти для регресії, що дозволяє враховувати локальні особливості даних. Використовується для видалення шуму та викидів у великих наборах даних.
- Видалення викидів на основі стандартних відхилень (Outlier Removal based on Standard Deviation). Цей метод визначає викиди, порівнюючи кожне значення зі стандартним відхиленням і видаляє значення, які перевищують певний поріг. Допомагає видаляти очевидні викиди з даних.
- Низькочастотний фільтр (Low-Pass Filter). Низькочастотний фільтр допускає сигнали з низькими частотами (нижче певного порогового значення) і пригнічує сигнали з високими частотами. Це означає, що він пропускає повільні зміни і згладжує високочастотний шум. Низькочастотні фільтри добре підходять для видалення високочастотного шуму з сигналів, таких як аудіо, відео, сигнали від сенсорів тощо. Вони також використовуються для вимірювання низькочастотних явищ, таких як середнє значення.

- Високочастотний фільтр (High-Pass Filter). Високочастотний фільтр допускає сигнали з високими частотами і пригнічує сигнали з низькими частотами. Він виділяє швидкі зміни і викиди у сигналі. Високочастотні фільтри можуть бути корисними для виділення і відокремлення швидких та короткочасних явищ у сигналі, наприклад, для виявлення різких переходів в аудіо, видалення початкового "удару" у звукозаписах, викидів у фізичних сенсорах тощо.

Для видобутку ознак з даних можна використовувати алгоритми визначення характеристик[13-15], такі як:

- Метод головних компонент (PCA): PCA використовується для зменшення розмірності даних, отриманих з мобільних сенсорів, і виділення важливих ознак. Це може покращити ефективність аналізу та зменшити обсяг даних для зберігання.
- Аналіз часового ряду (Time Series Analysis): Для аналізу даних, отриманих від датчиків, що фіксують зміни в часі, часто використовуються методи аналізу часових рядів, такі як авторегресійні моделі (AR), ковзні середні (MA), моделі ARIMA, експоненціальне згладжування тощо.
- Аналіз геопросторових даних (Geospatial Analysis): Для обробки геопросторових даних з GPS-датчиків можна використовувати методи геоінформатики, такі як розрахунок відстаней, геокодування, аналіз траєкторій тощо.
- Методи визначення характеристик образу (Image Feature Extraction): Якщо дані містять зображення або відео, то для видобутку ознак іноді використовуються методи обробки зображень, такі як гістограми кольорів, кутові ознаки, габаритні ознаки, глибокі нейронні мережі (наприклад, CNN).
- Визначення активності (Activity Recognition): Для визначення активностей користувача на основі даних з акселерометра і гіроскопа можна використовувати алгоритми класифікації, такі як метод опорних векторів (SVM), дерева рішень, нейронні мережі тощо.

- Аналіз текстових даних (Text Analysis): Якщо дані включають текстову інформацію, то для видобутку ознак з тексту можна використовувати методи обробки тексту, такі як векторизація тексту, тематичне моделювання, аналіз настроїв тощо.
- Автоматичне визначення ключових точок (Key Point Detection): Для обробки зображень або відео, автоматичне визначення ключових точок, таких як SIFT або SURF, може використовуватися для видобутку ознак та визначення об'єктів.
- Аналіз звуку (Audio Analysis): У випадку аудіоданих можуть використовуватися алгоритми аналізу звуку, такі як екстракція характеристик зі спектрограми, розпізнавання мови, аналіз тональності тощо.

Для об’єдання даних [16] популярні наступні алгоритми:

- Об'єднання за ключем (Merge / Join): Цей алгоритм використовується для об'єднання даних з двох або більше таблиць чи наборів даних на основі спільного ключа (унікального ідентифікатора). Це дозволяє створювати одну об'єднану таблицю на основі спільних значень ключа.
- Об'єднання за індексами (Index-Based Join): Цей метод використовує індекси для швидкого об'єднання даних. Він особливо ефективний для великих обсягів даних, коли потрібно зменшити час обробки.
- Об'єднання за допомогою хеш-функцій (Hash Join): Цей алгоритм використовує хеш-функції для об'єднання даних. Він може бути ефективним для об'єднання великих наборів даних.
- Об'єднання за допомогою конкатенації (Concatenation): Цей метод просто об'єднує дані, додаючи один набір даних до іншого. Він використовується, коли даних немає спільного ключа для об'єднання.
- Об'єднання за допомогою зовнішніх ключів (Outer Join): Цей алгоритм дозволяє зберегти всі записи з обох джерел, включаючи записи, які не мають спільних значень ключа.

• Об'єднання за допомогою машинного навчання: Деякі задачі об'єднання даних можна розв'язувати за допомогою алгоритмів машинного навчання, таких як кластеризація або категоризація.

Вибір конкретного методу обробки залежить від структури даних, потреб аналізу і конкретного завдання, що потребую рішення.

Після попередньої обробки даних, інформація стає більш структурованою та придатною для подальшого аналізу, що дозволяє використовувати її для вирішення різноманітних завдань, включаючи моніторинг здоров'я, навігацію, фітнес-вимірювання та багато інших застосувань(рис. 1.3.2).

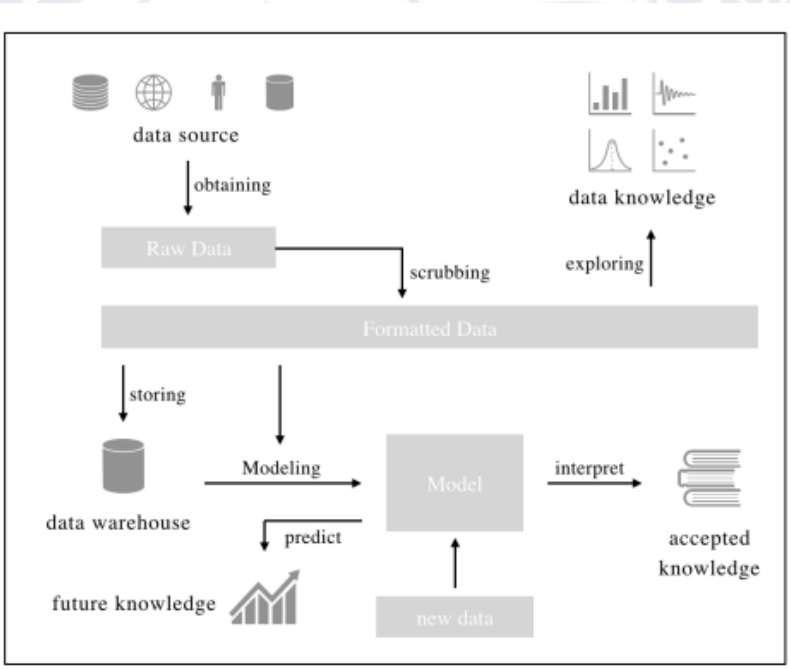

Рисунок 1.3.2 Загальний процес обробки даних

1.4 Дослідження методів аналізу та інтерпретації даних з мобільних пристроїв

Отримані дані можуть бути надзвичайно корисними для багатьох галузей, включаючи медицину, спорт, науку, технології та бізнес. Однак, для використання цих даних вимагається ретельний аналіз та інтерпретація.

Дослідимо, як ці методи можуть бути використані для отримання цінної інформації з сенсорів мобільних пристроїв та як їхні результати можуть застосовуватися в різних галузях.

Зважаючи на різноманітність даних та завдань, пов'язаних з аналізом мобільних даних, розглянемо детальніше деякі техніки аналізу даних.

Машинне навчання[17] - це комп'ютерна методологія, яка дозволяє системам навчатися на основі даних і приймати рішення без явно заданого програмного коду. В мобільних додатках, особливо тих, які збирають дані з різних сенсорів, машинне навчання може бути потужним інструментом для аналізу та розуміння цих даних. Розгляньмо кілька популярних методів машинного навчання та їхні приклади застосування:

- Дерева рішень: Вони використовуються для класифікації та регресійного аналізу. Це структура, що складається з вузлів, які представляють рішення, і гілок, які представляють альтернативи вибору. Можуть бути застосовані до завдань, таких як розпізнавання активності користувача на основі даних з акселерометра.
- Методи опорних векторів (SVM): Використовуються для класифікації і регресії. Ці алгоритми знаходять оптимальну гіперплощину для розділення даних у просторі. Вони можуть використовуватись для вирішення задач розпізнавання образів, наприклад спамовий елемент текстового повідомлення
- Глибоке навчання (Deep Learning): Включає в себе нейронні мережі з багатьма шарами для вирішення складних завдань. Мережі згорткового типу(CNN) можна використати для аналізу зображень, а рекурентні мережі (RNN) для аналізу послідовностей даних, наприклад прогнозу погоди.

Ці методи машинного навчання дозволяють мобільним додаткам аналізувати інформацію з сенсорів і надавати користувачам відповідну інформацію або певні функції, наприклад, рекомендації, розпізнавання образів або голосовий інтерфейс.

Статистичний аналіз[18] - це метод дослідження та інтерпретації даних з метою виявлення закономірностей, зв’язків та шаблонів у цих даних. У мобільних додатках статистичний аналіз може бути використаний для зрозуміння користувацьких поведінок, прогнозування тенденцій та вдосконалення

функціоналу додатків на основі аналізу даних, що збираються з мобільних пристроїв. Включає наступні методи:

- Перевірка гіпотез: Використовується для встановлення статистичної значущості виявлених взаємозв'язків у даних. Наприклад, можна проводити тести чи є статистично значущою різниця в середній кількості кроків між групами користувачів.
- Регресійний аналіз: Використовується для моделювання залежностей між змінними. Наприклад, прогнозувати рівень фізичної активності на основі інших факторів, таких як стать або вік.
- Кореляційний аналіз: Допомагає встановити ступінь взаємозв'язку між двома або більше змінними. Наприклад, може визначати чи існує кореляція між кількістю кроків, пройденими користувачем, і його споживанням калорій.

Аналіз часових рядів полягає в розумінні та прогнозуванні змін у часових рядах[19]. Використовуються наступні види::

- Авторегресійні моделі (AR): Базуються на ідеї, що майбутнє значення часового ряду залежить від його попередніх значень. Чим більше попередніх значень враховується, тим більша точність прогнозу. Прикладом можуть виступати датчики аналізу повітря, температури.
- Аналіз Фур'є: Аналіз Фур'є є потужним інструментом для обробки сигналів і даних з мобільних пристроїв, особливо тих, що мають мітки часу. Мобільні пристрої можуть вимірювати біометричні дані, такі як пульс або кількість кроків. Аналіз Фур'є може допомогти виявити патерни у цих даних, які можуть вказувати на стан здоров'я користувача.
- Сезоний аналіз: Цей метод дозволяє виявляти та аналізувати сезонні коливання у часовому ряді, тобто регулярні патерни або цикли, які повторюються через певний період часу. Наприклад прогнозування пікових навантажень на мережу.

Геопросторовий аналіз[20] є важливою складовою аналізу даних, особливо коли маємо справу з геолокаційними даними. Цей вид аналізу дозволяє

визначити, як дані розподілені в просторі та часі, і використовується для вирішення різноманітних завдань.

- Геолокаційна відстеження: Аналіз даних про розташування дозволяє відстежувати маршрути користувачів, їхню активність у певних регіонах і часові інтервали, проведені в певних локаціях. Можна застосовувати покращення застосування аналізу на основі поведінки користувачів у конкретних місцях та часі.
- Геозонування: Аналіз може розділити територію на геозони або території з певними характеристиками на основі геолокаційних даних. Допомогають орієнтувати користувачів до найближчих об'єктів інтересу, таких як ресторани, магазини або банкомати.
- Виявлення аномалій у геолокаційних даних: Аналіз може виявляти незвичайні або аномальні маршрути користувачів, які можуть свідчити про шахрайську діяльність або несправності датчиків. Дозволяють виявляти шахрайські активності на основі надзвичайних переміщень користувачів.
- Аналіз зображень та відео з мобільних пристроїв[21] - це важлива галузь обробки даних, яка дозволяє отримувати інформацію з візуальних даних, таких як фотографії та відеозаписи.
- Розпізнавання об'єктів: Використовується для ідентифікації об’єктів на фотографіях або відеозаписах. Наприклад розпізнавання обличь для автоматичної індексації і класифікації фотографій користувачів.
- Розпізнавання рукописного тексту: Мобільні пристрої можуть розпізнавати та конвертувати рукописний текст на фотографіях у текстову форму. Можливість сканування рукописних нотаток та конвертації їх у текстовий формат для зручного зберігання
- Автоматична класифікація зображень: Зображення можуть бути класифіковані на основі їхнього вмісту, наприклад, визначення, чи це фотографія природи, їжі або архітектури.

Існує багато інших методів аналізу даних та їх типів, які можна використовувати на мобільних пристроях. Кожен з цих методів має свої переваги та сферу застосування. Вірне застосування цих методів сприяє підвищенню продуктивності та ефективності аналізу даних з мобільних пристроїв. Важливо враховувати специфіку завдання та доступні ресурси пристрою для вибору найбільш відповідного методу. Незалежно від обраного методу, аналіз даних на мобільних пристроях відкриває безліч можливостей для отримання цінної інформації та прийняття обґрунтованих рішень.

1.5 Огляд можливостей візуалізації результатів даних з мобільних пристроїв

Візуалізація є ключовим компонентом аналізу даних з мобільних пристроїв, оскільки вона дозволяє представити інформацію у зрозумілій та зручній для користувача формі. Розглянемо різні можливості візуалізації результатів аналізу даних з мобільних пристроїв та їхню важливість для розуміння та інтерпретації цих даних.

Візуалізація даних з мобільних пристроїв може приймати різні форми, в залежності від характеру даних та мети аналізу[22-24]. Основні типи візуалізації включають:

- Графіки та діаграми: Лінійні графіки, стовпчикові графіки, кругові діаграми та інші типи графіків дозволяють відобразити числові дані в зручній формі.
- Географічні карти: Для даних, які мають геолокаційні відомості, використання географічних карт може візуалізувати просторову розташування даних.
- Графи та мережі: Використовуються для візуалізації взаємозв'язків та залежностей між об'єктами даних.
- Теплові карти: Вони дозволяють відобразити густину даних та їхній розподіл на площині або в часі.

• 3D візуалізація: Для деяких типів даних, таких як дані з акселерометрів у тривимірному просторі, може бути корисною 3D візуалізація.

• Анімація: Використання анімації для візуалізації динамічних процесів у даних.

Результати візуалізації даних з мобільних пристроїв стають більш інформативними та легше сприймаються користувачами. Інформація, представлена візуально, не лише полегшує аналіз та дослідження, але й дозволяє швидше виявляти закономірності, зрозуміло інтерпретувати дані та приймати обґрунтовані рішення. Завдяки візуалізації даних, відкриваються нові можливості для вдосконалення та розвитку різних галузей, роблячи їх більш продуктивними та конкурентоспроможними.

Візуалізація даних з мобільних пристроїв має широкий спектр застосувань:

• Моніторинг здоров'я: Візуалізація біометричних даних, таких як пульс, кров'яний тиск та інші показники здоров'я, дозволяє користувачам моніторити свій стан і приймати своєчасні заходи для поліпшення здоров'я.

• Спортивний аналіз: У спорті візуалізація даних може бути використана для аналізу фізичних параметрів спортсменів, їхнього руху та тренувального процесу.

• Маркетинг і реклама: Візуалізація даних може допомогти аналізувати поведінку користувачів, їхній інтерес до певних товарів та послуг, що дає можливість вдосконалити маркетингові стратегії.

• Моніторинг навколишнього середовища: Географічні карти та графіки дозволяють візуалізувати дані про якість повітря, рівень шуму та інші показники навколишнього середовища.

Візуалізація даних є важливим інструментом для прийняття управлінських та стратегічних рішень на основі даних з мобільних пристроїв. Вона допомагає виявляти закономірності, тренди та незвичайні явища, які можуть залишитися непоміченими в числових таблицях, як це видно з таблиці 1.5. Таким чином,

візуалізація робить дані більш зрозумілими і доступними для широкого кола користувачів, що сприяє прийняттю обґрунтованих рішень.

Таблиця 1.1 Порівняння можливостей діаграм та таблиць

Параметри	Діаграми	Таблиці
Визначення	Відображення фактів за допомогою різних форм та символів.	Стовпці та рядки, що використовуються для коротких підсумків.
Типи	Кругова, гістограма, стовпчикова, блок-схема, радар, лінійна діаграма тощо.	Стовпці, рядки, комірки, заголовок, динамічні дані.
Представлення	Розподіли, тенденції та інші статистичні моделі.	Кількості, числа, медіа або тексти в рядку/стовпці.
Тенденції та моделі	+	-
Тип даних	Сирі та оригінальні	Відфільтровані та організовані

1.6 Приватність та безпека збору, обробки та аналізу даних

Захист даних є надзвичайно важливою аспектом аналізу та обробки інформації з мобільних пристроїв. Забезпечення приватності та безпеки даних користувачів є високою пріоритетною задачею[5, 25-27].

Методи захисту особистих даних на мобільних пристроях включають в себе широкий спектр заходів, таких як шифрування, аутентифікація, контроль доступу та безпека мережі. Шифрування даних може бути застосовано для захисту інформації під час її передачі та зберігання на пристроях. Додатково, методи аутентифікації, такі як відбиток пальця, розпізнавання обличчя чи двофакторна аутентифікація, забезпечують впевненість у тому, що лише правомісний користувач має доступ до даних.

Контроль доступу визначає, які користувачі мають право переглядати та модифікувати дані, забезпечуючи обмеження прав доступу до конфіденційної

інформації. Безпека мережі включає в себе заходи для запобігання несанкціонованому доступу до мережевого з'єднання між мобільними пристроями та серверами, як це показано на рисунку 1.6.1.

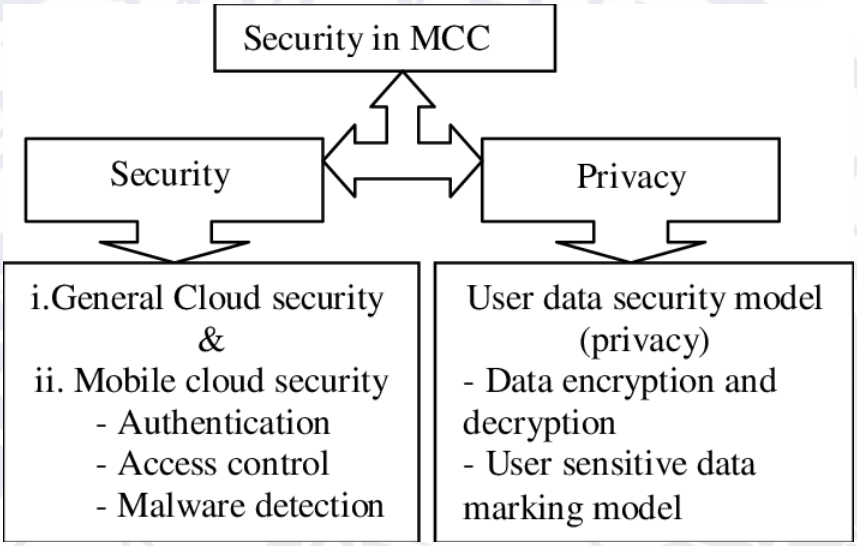

Рис. 1.6.1 Класифікація безпеки та конфіденційності мобільних хмарних обчислень

Крім того, необхідно враховувати правові аспекти та вимоги до обробки особистих даних, зокрема відповідно до регулюючих законів і нормативів, таких як Загальний регламент з регулювання захисту даних (GDPR) в Європейському Союзі. Велика увага також повинна бути приділена захисту від вразливостей та зловмисних програм, які можуть загрожувати безпеці даних.

Основною метою є забезпечення конфіденційності, цілісності та доступності даних, збереження довіри користувачів та дотримання всіх законодавчих норм.

Загальна мета цих заходів - забезпечити приватність та безпеку користувачів, запобігти витокам особистої інформації та забезпечити дотримання вимог щодо захисту даних. Такий захист стає невід'ємною частиною роботи з мобільними даними та їх аналізу в сучасному цифровому світі.

Висновки

У результаті огляду і дослідження існуючих типів мобільних сенсорів було виявлено різноманітні типи датчиків, які можуть бути використані для збору різноманітних даних з мобільних пристроїв.

Методи збору даних з мобільних сенсорів були розглянуті, включаючи методи отримання даних з акселерометрів, гіроскопів, GPS-датчиків та багатьох інших. Кожен метод має свої переваги і обмеження, і вибір методу залежить від конкретного завдання дослідження.

Аналіз доступних засобів обробки даних з мобільних сенсорів показав різноманітні алгоритми та методи, які можуть бути застосовані для обробки та підготовки даних до аналізу.

Дослідження методів аналізу та інтерпретації даних з мобільних пристроїв показало, що існують різноманітні методи, такі як машинне навчання, статистичний аналіз, аналіз часових рядів та інші, які можуть бути використані для отримання цінної інформації з даних.

В огляді можливостей візуалізації результатів даних з мобільних пристроїв було представлено різні методи та інструменти для представлення даних у зрозумілій та інформативній формі.

Нарешті, приватність та безпека збору, обробки та аналізу даних зазнали ретельного аналізу. Важливість захисту особистих даних користувачів була підкреслена, і розглянуті різні аспекти безпеки, такі як шифрування, захист від несанкціонованого доступу та відповідність законодавству.

Усі ці аспекти разом створюють основу для подальших досліджень та розробки нових методів та технологій у сфері аналізу даних з мобільних пристроїв.

РОЗДІЛ 2
ДОСЛІДЖЕННЯ ТА ВПРОВАДЖЕННЯ МЕТОДІВ ЗБОРУ, ОБРОБКИ ТА АНАЛІЗУ ДАНИХ З МОБІЛЬНИХ СЕНСОРІВ

2.1 Виявлення та аналіз недосконалостей існуючих методів

В ході аналізу наукової літератури та технічних звітів було виявлено декілька ключових обмежень існуючих методів збору, обробки та аналізу даних з мобільних сенсорів.

Однією з основних проблем є висока затрата ресурсів[28-29]. Багато методів збору даних вимагають високої потужності для передачі даних або постійного з'єднання з мережею, що може швидко виснажувати батарею мобільного пристрою(рис. 2.1.1). Деякі алгоритми обробки та аналізу даних можуть бути високообчислювальними і вимагати значних обчислювальних ресурсів, що може бути проблематичним для мобільних пристроїв з обмеженими ресурсами.

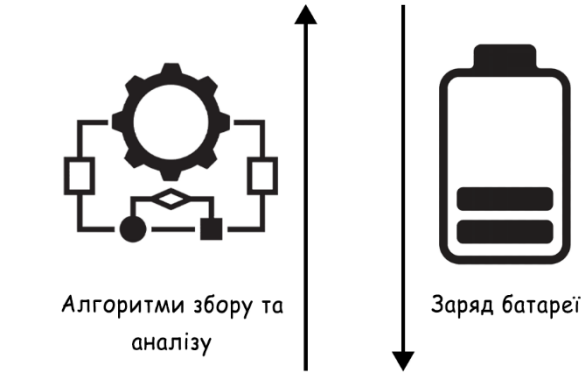

Рис. 2.1.1 Відношення заряду батареї від навантаження алгоритмів збору та аналізу

Іншою важливою проблемою є недостатня точність. Мобільні сенсори можуть виробляти дані з помилками через внутрішні та зовнішні фактори, такі як шум, помилки калібрування або зовнішні завади. Деякі сенсори можуть мати обмежену роздільну здатність, що впливає на точність та якість зібраних даних.

Конфіденційність та безпека[25-27] також є важливими питаннями. Під час передачі даних між мобільним пристроєм та сервером може виникнути ризик

витоку чутливих даних через незахищені мережеві з'єднання або атаки зловмисників. Відсутність сильних механізмів аутентифікації та авторизації може призвести до несанкціонованого доступу до даних або систем обробки даних.

Обмеження зберігання даних також є важливим аспектом. Мобільні пристрої можуть мати обмежену пам'ять для зберігання даних, що вимагає ефективних стратегій управління даними та зберігання.

Недостатній аналіз та інтерпретація даних є іншим важливим аспектом. Існуючі інструменти та платформи можуть мати обмеження у функціональності, що ускладнює аналіз та інтерпретацію зібраних даних.

Ці та інші обмеження вказують на необхідність подальшого дослідження та розробки нових методів та технологій для подолання виявлених проблем і поліпшення ефективності збору, обробки та аналізу даних з мобільних сенсорів.

2.2 Визначення потреби в оптимізації та вдосконаленні

Після виявлення основних обмежень методів збору, обробки та аналізу даних з мобільних сенсорів, наступним кроком є визначення потреби в їх оптимізації та вдосконаленні. Це включає аналіз існуючих рішень та виявлення областей для вдосконалення, а також обґрунтування необхідності розробки нових методів.

Для визначення потреби в оптимізації та вдосконаленні методів збору, обробки та аналізу даних з мобільних сенсорів було проведено ретельний аналіз існуючого стану справ у даній галузі. Цей аналіз був спрямований на вивчення технологічних та методологічних аспектів, які впливають на ефективність та надійність систем збору даних. Основні етапи аналізу включали в себе огляд наукової літератури, аналіз існуючих технологічних рішень та вивчення реальних випадків використання мобільних сенсорних систем.

В ході огляду наукової літератури було виявлено, що багато сучасних систем, спрямованих на збір даних, зазнають серйозних труднощів у вигляді високих

енергетичних витрат[28] і обчислювальних обмежень[29-31]. Це стало особливо очевидним у контексті обробки даних у реальному часі або коли обсяги даних великі за розміром. Наприклад, стаття Xu та співавторів вказує на проблеми надійності передачі даних у бездротових мережах сенсорів, що може впливати на ефективність збору та аналізу даних у реальному часі[32]. Автори вказують на проблеми надійності передачі даних у бездротових мережах сенсорів, зокрема на важливість прогнозування відмов та оптимізації мобільних шляхів для ефективного збору даних. Також вони розглядають методи злиття даних для забезпечення точності та надійності інформації у мережі.

Також було виявлено, що існуючі методи обробки та аналізу даних не завжди гарантують відповідний рівень конфіденційності і безпеки, що є насущно важливими в цифровому суспільстві сьогодення. Стаття Tang та співавторів розглядає архітектуру терміналу інформаційної безпеки для мобільного Інтернету речей у сфері енергетичного транспортування на основі аналізу великих даних, що підкреслює важливість захисту даних у мобільних системах[33-34]. Автори пропонують архітектурний дизайн терміналів безпеки інформації для ефективного управління даними про електроенергію, враховуючи великий обсяг даних та різноманітність типів мобільної інформації. Експериментальні результати показали, що запропонований термінал безпеки інформації може зменшувати використання комунікаційних ресурсів та знижувати вартість комунікацій під час агрегації багатовимірних даних, підкреслюючи важливість захисту даних у таких системах.

Ці відкриття підкреслили необхідність розробки нових методів та технологій, які можуть вирішити існуючі проблеми та вдосконалити процеси збору, обробки та аналізу даних з мобільних сенсорів. Особливу увагу слід приділити розробці більш ефективних алгоритмів для обробки та аналізу даних, які здатні зменшити обчислювальні витрати та підвищити точність результатів, а також поліпшенню механізмів безпеки для захисту даних від несанкціонованого доступу та можливих зловживань[35].

Далі було проведено аналіз різних напрямків для можливого вдосконалення ситуації. Один із ключових напрямків - це розробка більш ефективних алгоритмів для обробки та аналізу даних, які здатні зменшити обчислювальні витрати та підвищити точність результатів[36]. Іншим важливим аспектом є поліпшення механізмів безпеки для захисту даних від несанкціонованого доступу та можливих зловживань.

Потреба у розробці нових методів визначається наявними обмеженнями та можливостями для вдосконалення. Діючі методи нерідко не в змозі відповісти на зростаючі вимоги до швидкості обробки, точності аналізу та безпеки даних. З цієї причини розробка нових методів, спрямованих на ефективне вирішення цих завдань, є критично важливою для подальшого розвитку галузі збору, обробки та аналізу даних, отриманих за допомогою мобільних сенсорів.

2.3 Розробка інноваційних методів та підходів збору, обробки та аналізу даних з мобільних сенсорів

Розробка ефективних методів та підходів для збору, обробки та аналізу цих даних є важливим завданням для науковців та інженерів. Цей розділ розглядає інноваційні методи та підходи, які були запропоновані для вирішення цих викликів.

Однією з ключових областей інновацій є розробка нових методів збору даних з мобільних сенсорів. Існує ряд можливих вдосконалень, які можна внести у методи збору даних з мобільних сенсорів. Для більш детального опису і огляду можливих методів можна проаналізувати реальні підходи. Наприклад, у дослідженні Андреа Каппоні та ін., під назвою "A Cost-Effective Distributed Framework for Data Collection in Cloud-Based Mobile Crowd Sensing Architectures"[37-38] було представлено новий розподілений та економічно ефективний фреймворк для збору даних в архітектурах мобільного збору даних на основі хмари(рис. 2.3.1). Автори концентрують свою увагу на зменшенні

витрат на збір та звітність даних, одночасно підвищуючи корисність збору даних та якість наданої інформації.

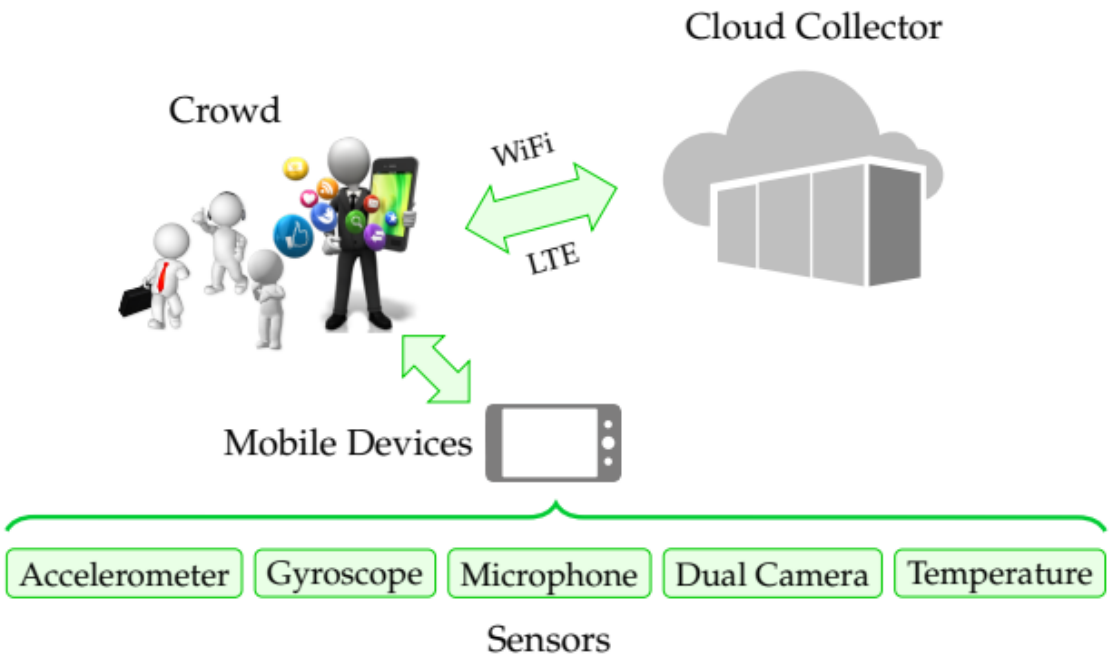

Рис. 2.3.1 Хмарна система MCS (мобільний краудсенсінг)

Цей підхід може служити хорошим прикладом для розробників мобільних додатків та сервісів, які прагнуть оптимізувати використання ресурсів мобільних пристроїв та хмарних сервісів для збору та обробки даних.

Відштовхуючись від цього дослідження, можна запропонувати декілька власних вдосконалень. По-перше, можна розробити алгоритми для динамічного управління частотою збору даних в залежності від контексту користувача та стану системи(рис. 2.3.2). Наприклад, у випадках, коли користувач перебуває в статичному стані, частоту збору даних можна знизити, щоб зекономити енергію батареї. Навпаки, у динамічних сценаріях, таких як водіння автомобіля або фізична активність, частоту збору даних можна збільшити для отримання більш точної інформації.

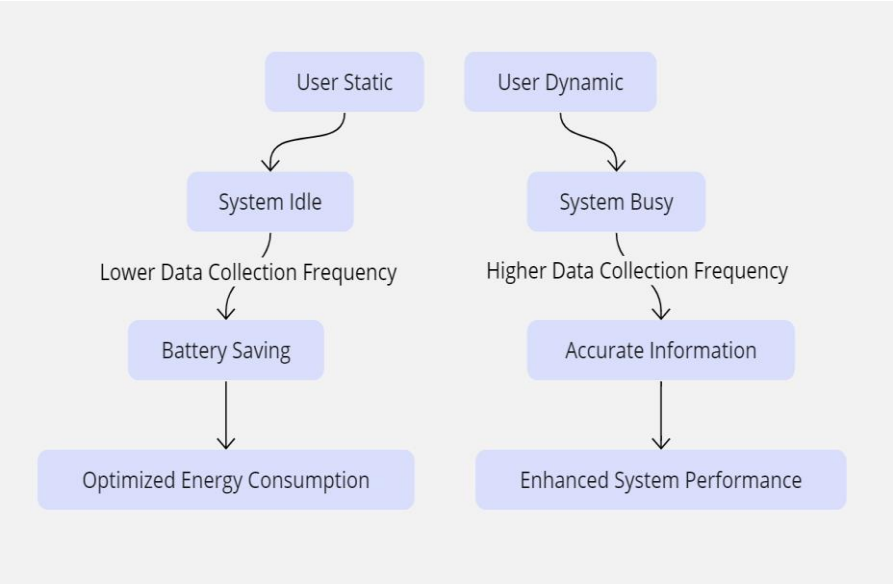

Рис. 2.3.2 Динамічне управління частотою збору даних

По-друге, важливо розробити методи для ефективного управління приватністю та безпекою даних під час збору та передачі даних до хмарних сервісів. Це може включати в себе застосування криптографічних технологій для шифрування даних на пристрої користувача перед їх передачею, а також розробку політик доступу та аутентифікації для захисту даних від несанкціонованого доступу.

По-третє, можна розглянути можливість використання машинного навчання та інших методів аналізу даних безпосередньо на мобільних пристроях для виявлення цінних закономірностей та отримання нової корисної інформації без необхідності передачі великих обсягів даних до хмари. Це може допомогти зменшити завантаження на мережеву інфраструктуру та забезпечити кращу приватність користувачів.

Крім того, інтеграція різних типів сенсорів та джерел даних може надати більш повну картину для аналізу та виведення висновків[39]. Наприклад, комбінація даних з GPS, акселерометрів, гіроскопів та інших сенсорів може допомогти створити більш точні моделі поведінки користувачів та їхнього оточення, приклад чого можна побачити на рисунку 2.3.3.

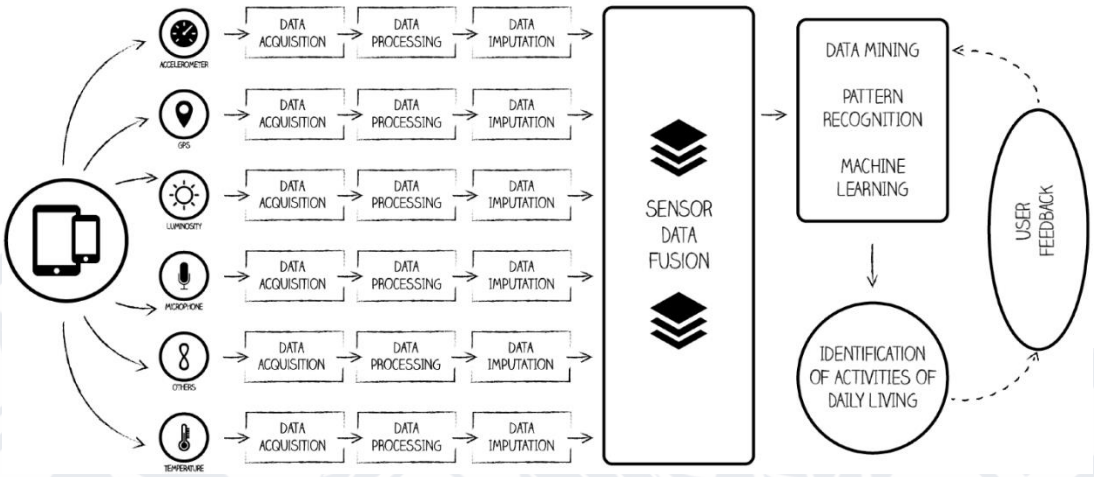

Рис. 2.3.3 Схема мультисенсорної системи для розпізнавання повсякденної діяльності

З усіма цими вдосконаленнями, нові методи збору даних з мобільних сенсорів можуть відкрити нові можливості для розробки інноваційних мобільних додатків та сервісів, які можуть краще відповідати на вимоги користувачів та забезпечувати високий рівень безпеки та приватності даних.

Обробка та аналіз даних з мобільних сенсорів є важливими для виявлення цінних інсайтів та підтримки високої продуктивності мобільних та хмарних систем. У дослідженні К. Хасіб та ін., під назвою "DDR-ESC: A Distributed and Data Reliability Model for Mobile Edge-Based Sensor-Cloud", було представлено розподілену модель надійності даних для мобільних систем на основі периферійних датчиків та хмарних обчисленнях[40]. Цей підхід спрямований на покращення продуктивності передачі даних та забезпечення контролю безпеки, що є критично важливим у сучасних мобільних та хмарних архітектурах.

Відштовхуючись від цього дослідження, можна запропонувати декілька власних вдосконалень (рис. 2.3.4). По-перше, важливо розробити алгоритми для ефективного управління потоками даних між мобільними пристроями та хмарними сервісами, що дозволить оптимізувати використання мережевих ресурсів та зменшити затримки. По-друге, можна розробити методи для автоматичного виявлення аномалій та відхилень у даних з мобільних сенсорів у реальному часі, що дозволить швидко реагувати на можливі проблеми та

забезпечити високу якість даних. По-третє, важливо зосередитись на розробці методів для забезпечення конфіденційності та безпеки даних на всіх етапах їх обробки та аналізу, включаючи застосування сучасних криптографічних технологій та протоколів безпеки.

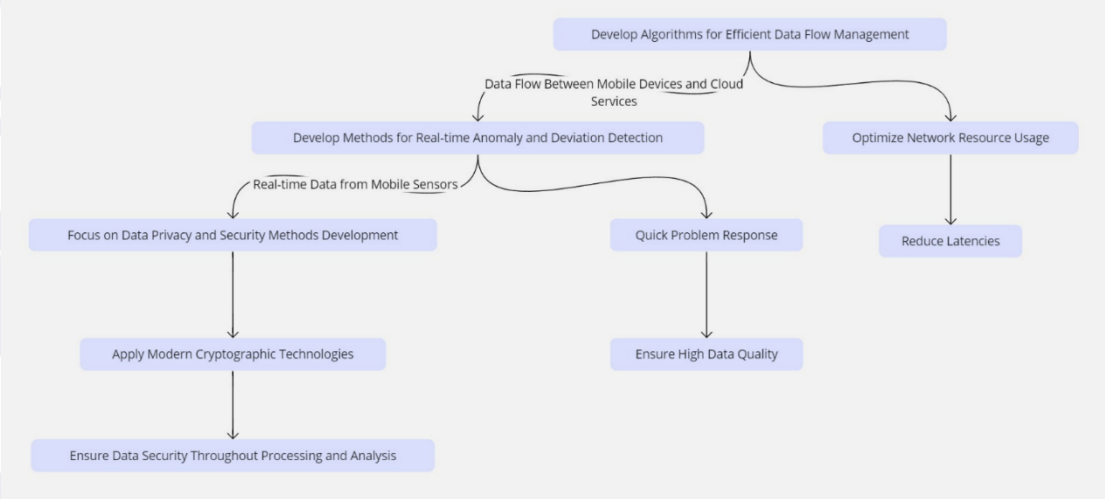

Рис. 2.3.4 Послідовний процес вдосконалень обробки та аналізу даних з мобільних сенсорів

Також можна розглянути можливість використання технологій штучного інтелекту (далі ШІ) та машинного навчання (далі МН) безпосередньо на мобільних пристроях для проведення обробки та аналізу даних. Однією з ключових переваг є можливість проведення аналізу даних в реальному часі. Мобільні пристрої здатні швидко реагувати на зміни в даних, виявляти аномалії та надавати користувачам відгуки миттєво, що є критично важливим у багатьох сценаріях, наприклад, у моніторингу здоров'я або безпеки. Крім того, обробка даних на пристрої значно зменшує необхідність передачі великих обсягів даних до хмари, що може зекономити ресурси мережі та зменшити витрати на передачу даних (рис. 2.3.5). Це також може прискорити процес аналізу, оскільки дані не потребують бути переданими через мережу до сервера для обробки.

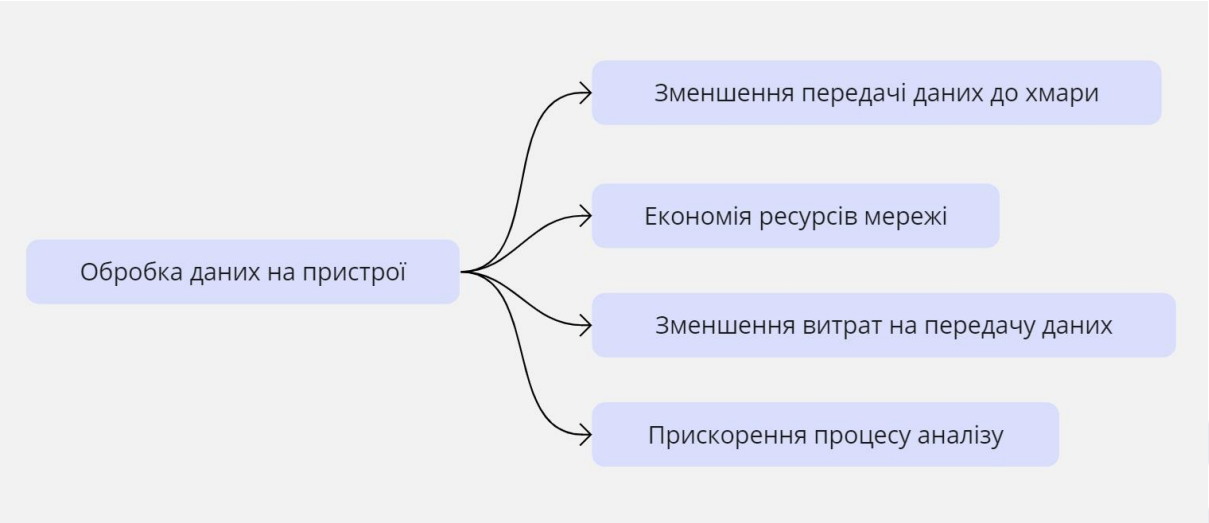

Рис. 2.3.5 Переваги обробки даних на пристроях

Обробка даних локально на пристрої також забезпечує кращий контроль над конфіденційністю та безпекою даних. Дані можуть бути захищеними від несанкціонованого доступу та витоку, оскільки вони залишаються на пристрої користувача. Ще однією важливою перевагою є можливість персоналізації досвіду користувача. ШІ та МН можуть аналізувати дані з сенсорів та адаптувати функціональність мобільного додатку під конкретного користувача, що може покращити задоволення користувача та забезпечити більш особистий досвід.

Нарешті, автоматичне навчання та оптимізація є іншим важливим аспектом використання ШІ та МН на мобільних пристроях. Мобільні пристрої можуть використовувати дані з сенсорів для автоматичного навчання та оптимізації своєї роботи, що може покращити продуктивність пристрою та забезпечити краще використання ресурсів (рис. 2.3.6). У цілому, використання технологій ШІ та МН безпосередньо на мобільних пристроях відкриває нові горизонти для розробки мобільних додатків, покращення продуктивності та забезпечення кращого досвіду користувача.

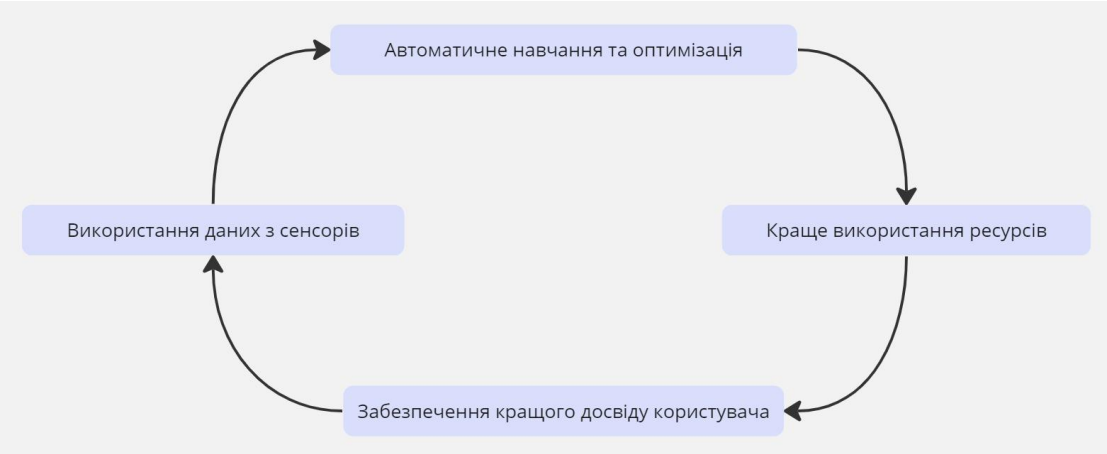

Рис. 2.3.6 Автономне використання ШІ та МН на пристроях

Крім того, важливо розробити стандарти та протоколи для забезпечення сумісності між різними мобільними та хмарними платформами, що дозволить легко інтегрувати нові методи обробки та аналізу даних в існуючі системи та сприяти широкому їх розповсюдженню у галузі.

Також варто відзначити, що розробка нових методів та підходів у цій області вимагає глибокого розуміння викликів, пов'язаних з обробкою великих обсягів даних у реальному часі, а також забезпечення конфіденційності та безпеки даних. Ці інноваційні методи та підходи відкривають нові можливості для розробки мобільних додатків та сервісів, які можуть ефективно використовувати дані з мобільних сенсорів для надання цінної інформації користувачам та підтримки прийняття рішень у реальному часі.

2.4 Аналіз та оцінка відмінностей від існуючих підходів

Розроблений підхід до динамічного управління частотою збору даних відкриває нові горизонти у відношенні до ефективності та точності обробки даних.

У порівнянні з традиційними методами, які використовують статичну частоту збору даних або потребують ручного управління від користувача, даних підхід автоматизує цей процес, адаптуючи частоту збору даних відповідно до контексту користувача та стану системи (див. рис. 2.3.2). Наприклад, коли

користувач перебуває в статичному стані, частоту збору даних можна знизити для економії енергії батареї. У динамічних сценаріях, таких як водіння автомобіля або фізична активність, частоту збору даних можна збільшити для отримання більш точної інформації. Це не тільки оптимізує використання ресурсів, але і забезпечує більш точний збір даних у критичних ситуаціях.

Хоча деякі існуючі системи можуть вже використовувати адаптивні стратегії, даних підхід може пропонувати більш глибокий аналіз контексту та стану системи для ще більш точного управління частотою збору даних.

Запропонований підхід до обробки та аналізу даних відображає глибоке розуміння важливих аспектів управління даними в сучасних мобільних та хмарних системах. Перш за все, підкреслюється важливість ефективного управління потоками даних між мобільними пристроями та хмарними сервісами, що є ключовим для оптимізації використання мережевих ресурсів та зменшення затримок. Це може відігравати важливу роль у підтримці високої продуктивності та відгуку системи.

Далі, пропонується розробка методів для автоматичного виявлення аномалій та відхилень у даних з мобільних сенсорів у реальному часі. Це важливий аспект, який може допомогти швидко виявляти та реагувати на можливі проблеми, що в свою чергу забезпечить високу якість даних та надійність системи. Нарешті, зосереджується увага на важливості забезпечення конфіденційності та безпеки даних на всіх етапах їх обробки та аналізу. Включення сучасних криптографічних технологій та протоколів безпеки є критично важливим для захисту конфіденційності користувачів та забезпечення відповідності нормативним вимогам.

У цілому, цей підхід відображає комплексний погляд на управління, аналіз та захист даних у мобільних та хмарних середовищах, що може значно покращити ефективність та безпеку систем обробки даних.

Використання технологій штучного інтелекту (ШІ) та машинного навчання (МН) безпосередньо на мобільних пристроях відкриває нові можливості для

обробки та аналізу даних. Розроблений підхід акцентує на можливості аналізу даних в реальному часі на мобільних пристроях, що дозволяє швидко реагувати на зміни в даних, виявляти аномалії та надавати користувачам відгуки миттєво, що є критично важливим у багатьох сценаріях, наприклад, у моніторингу здоров'я або безпеки.

Такий підхід також сприяє економії ресурсів мережі, оскільки обробка даних на пристрої зменшує необхідність передачі великих обсягів даних до хмари, що може зекономити ресурси мережі та зменшити витрати на передачу даних. Крім того, процес аналізу може бути прискорений, оскільки дані не потребують бути переданими через мережу до сервера для обробки.

Крім того підхід намагається вирішити деякі з основних проблем, пов'язаних з централізованою обробкою даних, використовуючи можливості ШІ та МН безпосередньо на мобільних пристроях. Це може привести до покращення швидкодії, зменшення витрат на передачу даних та покращення здатності реагувати на зміни в реальному часі. З іншого боку, існуючі методи зазвичай включають централізовану обробку, де дані збираються з мобільних пристроїв та передаються на сервери для подальшої обробки та аналізу. Це може викликати затримки та високі витрати на передачу даних, особливо у регіонах з високими тарифами на передачу даних. Централізована обробка також може призвести до затримок у виявленні аномалій та наданні відгуків користувачам, особливо у критичних ситуаціях.

2.5 Дослідження покращення методів збору даних про місце положення на основі акселерометра та історії рухів користувача.

Основна мета даного дослідження - зменшення енерговитрат системи позиціонування при забезпеченні достатньо точної інформації про положення.

У вимірюванні місцезнаходження в міських умовах використання GPS часто зустрічає труднощі через меншу точність сигналу в густонаселених районах. Відсутність необхідності в постійному використанні GPS може значно зменшити

енерговитрати систем позиціонування. У цьому контексті вивчення покращеного методу збору даних, що базується на акселерометрі та історії рухів користувача, набуває особливого значення.

Техніка використання акселерометра:

- Використання обертового акселерометра для ефективної оцінки руху користувача.
- Сенсор детектує рух та вимірює відношення активності - частку часу, коли користувач знаходиться в русі між двома оновленнями позиції.
- Запобігання активації GPS, коли користувач не рухається, з використанням відношення активності для оцінки поточної швидкості.

Збереження історії рухів користувача:

- Аналіз просторово-часової історії рухів для точного визначення моменту активації GPS.
- Розрахунок середньої швидкості відносно попередньої позиції та асоціація з попередньою координатою просторово-часового простору.
- Використання середніх значень швидкості та відношення активності для точної оцінки поточної швидкості користувача, що дозволяє ефективно активувати GPS тільки при перевищенні невизначеності положення порогу точності.

У контексті мінімізації активації GPS, застосовується використання акселерометра для виявлення руху, що відрізняє цей підхід двома ключовими аспектами.

Перш за все, на відміну від попередніх досліджень, де акселерометр використовується як бінарний датчик руху або обмежується виявленням лише пішоходів, даних підхід полягає в використанні акселерометра для вимірювання відношення активності. Це визначає частку часу, протягом якого користувач перебуває в русі. Далі цей коефіцієнт активності використовується разом з історією інформації про швидкість для оцінки поточної швидкості користувача.

З урахуванням високого енергоспоживання акселерометра, увага приділяється ретельному налаштуванню його режиму роботи для збереження енергії без суттєвого погіршення точності результатів.

Проте слід відзначити, що неперервне використання акселерометра неможливо через високі витрати енергії, які воно вимагає. Аналіз показує, що включення акселерометра на протязі 5 хвилин витрачає більше енергії, ніж активація GPS протягом 1 хвилини. Таким чином, постійна робота акселерометра виявляється неефективною з точки зору споживання енергії. Рекомендацією є рідше активувати GPS, забезпечуючи оптимальний рівень енергії системи. З урахуванням основної мети — енергоефективності — можна зробити висновок, що безперервна робота акселерометра не є прийнятним варіантом для даного дизайну.

Ключовим елементом запропонованого методу є просторово-часова таблиця, яка утримує інформацію щодо минулої історії переміщень користувачів. Зокрема, ця таблиця включає середню швидкість користувача та середнє оцінене співвідношення активності для кожної одиниці просторово-часової координати у 3-вимірній системі (широта, довгота, час доби). При вирішенні питань, пов'язаних із включенням GPS, використовується історія середньої швидкості користувача та співвідношення активності для поточної точки в просторі та часі. Потім здійснюється розрахунок поточної невизначеності положення на основі оцінених значень поточної швидкості та співвідношення активності.

Основна ідея за цією концепцією полягає в тому, що поведінка користувача в конкретній точці просторово-часової системи координат часто є систематичною. Наприклад, людина зазвичай залишається нерухомою протягом тривалого періоду, будучи в офісі під час робочого часу або вдома під час ночі. Рухаючись певним маршрутом, людина зазвичай дотримується середньої швидкості ходьби, а по дорозі, якою вона зазвичай їздить, швидкість її руху вища.

Алгоритм оновлює та використовує цю просторово-часову історію, проводячи квантування як по простору, так і по часу. Просторові координати

округлюються до трьох знаків після коми та квантуються у двовимірну сітку розміром 0,001’ x 0,001’, що відповідає приблизно 100 метрам. Квантування часу здійснюється у контейнери розміром 30 хвилин кожен.

У даному методі кожному блоку сітки в квантованій системі координат призначаються дві величини: історія середньої швидкості та середній коефіцієнт активності, видимий у полі. При кожному оновленні положення користувача, система обчислює швидкість між попереднім та новим положеннями, оновлює середню швидкість та коефіцієнт активності для відповідного блоку сітки. Ця просторово-часова історія використовується для обчислення невизначеності положення в майбутньому, використовуючи рівняння, що залежать від швидкості та коефіцієнта активності. Невизначеність визначається як результат функцій швидкості та активності в минулому положенні. Якщо інформація про поточне положення відсутня, застосовується попереднє відоме положення.

У нашому методі використано коефіцієнт активності як заміну для швидкості з метою поліпшення точності визначення часу активації GPS та зменшення енергоспоживання, порівняно з іншими методами, такими як використання історії або виявлення руху. Давайте розглянемо це на прикладі.

Припустимо, що користувач рухається до позиції A, стоїть там протягом 4 одиниць часу, а потім рухається з позиції A зі швидкістю 10 м/с і коефіцієнтом активності 0,5 протягом одиниці часу. За цей період, середні значення швидкості та коефіцієнта активності для позиції A становлять {0,0,0,0,10} та {0,0,0,0,0.5}, що в результаті дає збережені середні значення 2 м/с та 0,1 в історії.

Пізніше, коли користувач повертається на позицію A, можливі два сценарії. Якщо користувач нерухомий, коефіцієнт поточної активності R(t) буде дорівнювати нулю, що призведе до оціненої швидкості V(t) рівно нулю, відображаючи відсутність руху.

У іншому випадку, якщо користувач виходить з позиції A з коефіцієнтом R(t) близьким до 0,5, визначена швидкість V(t) стає $2 * 0,5/0,1 = 10$ м/с. Це означає,

що система правильно оцінює швидкість користувача та відповідну невизначеність.

Такий підхід дозволяє системі ефективно та енергоефективно оцінювати рух користувача, використовуючи GPS лише в ситуаціях, коли це дійсно необхідно, що є важливим для збереження енергії та оптимізації роботи додатку.

Отже, цей підхід дозволяє нам ефективно оцінювати рух користувача та включати GPS лише в необхідних випадках для збереження енергії. Інтуїтивно зрозуміло, що використання коефіцієнта активності як сурогату для швидкості може покращити точність визначення часу активації GPS та зменшити енергоспоживання, порівняно з оцінкою швидкості через акселерометр, що завжди увімкнено.

2.6 Визначення потенційних проблем та недоліків розроблених методів

Розробка та впровадження запропонованих методів може мати декілька потенційних недоліків або викликів. По-перше, технічна складність є однією з основних проблем. Розробка, тестування та оптимізація алгоритмів для ефективної роботи в різних контекстах може вимагати значних зусиль та ресурсів.

По-друге, точність збору даних може бути під загрозою. Зменшення частоти збору даних у статичних сценаріях може призвести до пропуску важливої інформації, тоді як збільшення частоти збору даних у динамічних сценаріях може забезпечити більш точну інформацію, але також може збільшити навантаження на систему.

Третім аспектом є відгук системи. Збільшення частоти збору даних може вплинути на швидкість відгуку системи, що може призвести до затримок у відгуках, особливо у критичних ситуаціях. Це може бути неприйнятним у сценаріях, які вимагають миттєвого відгуку.

Наступною проблемою є енергетична ефективність. Неправильно налаштовані алгоритми можуть неефективно використовувати енергію, що може призвести до швидкого розряджання батареї мобільних пристроїв.

Далі, вартість є іншим важливим фактором. Впровадження сучасних криптографічних технологій та протоколів безпеки може бути вартісним, особливо для малих та середніх підприємств. Це може стати великим фінансовим навантаженням для організацій з обмеженими ресурсами.

Також важливо врахувати продуктивність. Хоча динамічне управління потоками даних може оптимізувати використання мережевих ресурсів, але воно також може вплинути на продуктивність системи, особливо у випадках високої мережевої затримки або обмеженого пропускної здатності.

Приватність та безпека даних є іншим важливим викликом. Незважаючи на застосування криптографічних технологій, існує ризик витоку або зловживання даними, особливо у випадку вразливостей або атак зловмисників.

Сумісність та інтеграція з існуючими системами та технологіями може бути викликом, особливо у випадках, коли стандарти або протоколи відрізняються. Це може вимагати додаткових зусиль для забезпечення гладкої інтеграції.

Нарешті, підтримка та оновлення системи є іншим важливим аспектом. Потреба у постійній підтримці та оновленні системи для забезпечення її надійності, безпеки та актуальності може бути трудомісткою та вартісною.

Ці та інші потенційні недоліки та виклики варто враховувати під час планування та впровадження підходу до обробки та аналізу даних.

Висновки

Основні проблеми обмеження існуючих методів збору, обробки та аналізу даних включають високі затрати ресурсів, недостатню точність, питання конфіденційності та безпеки, обмеження зберігання даних та недостатній аналіз та інтерпретація даних.

Було виявлено, що сучасні системи збору даних часто стикаються з високими енергетичними витратами та обчислювальними обмеженнями, особливо у контексті обробки даних у реальному часі або при великих обсягах даних. А також, що існуючі методи обробки та аналізу даних не завжди можуть забезпечити належний рівень конфіденційності та безпеки, що є критично важливим у сучасному цифровому суспільстві. Один із ключових напрямків - це розробка більш ефективних алгоритмів для обробки та аналізу даних. Іншим важливим аспектом є поліпшення механізмів безпеки захисту даних.

В результаті були запропоновані нові методи та підходи збору, обробки та аналізу даних з мобільних сенсорів. Розробка алгоритмів для динамічного управління частотою збору даних може оптимізувати енергетичні витрати та забезпечити точність зібраної інформації. Такий підхід дозволяє балансувати між економією енергії та якістю даних. Виявлена потреба в розробці алгоритмів для ефективного управління потоками даних між мобільними пристроями та хмарними сервісами.

Був проведений аналіз та оцінка відмінностей від існуючих підходів, в результаті чого розроблений підхід до динамічного управління частотою збору даних показав переваги на існуючими методами. Зазначено важливість ефективного управління потоками даних між мобільними пристроями та хмарними сервісами. А також використання технологій штучного інтелекту та машинного навчання відкриває нові можливості для обробки та аналізу даних у реальному часі.

Крім того були визначені потенційні проблеми та недоліки розроблених методів.

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНА РОЗРОБКА ДОДАТКУ ТА ЙОГО АЛГОРИТМУ ОПТИМІЗАЦІЇ ЗБОРУ ДАНИХ ДЛЯ МОБІЛЬНИХ СЕНСОРІВ

3.1 Стартова концепція та теоретичні основи розробки

Метою розробки є створення експериментального мобільного додатку, який демонструє можливість оптимізації процесу збору даних з сенсорів на основі розробленого алгоритму. Цей алгоритм реагує на умовні сценарії користувача, такі як період не активності або активності, для динамічного управління частотою збору даних. Це дозволяє теоретично зекономити енергію батареї, оптимізувати використання ресурсів пристрою та забезпечити точний та релевантний збір даних, незважаючи на відсутність доступу до реальних даних користувача.

Теоретичний алгоритм оптимізації збору даних базується на динамічному управлінні частотою опитування сенсорів в залежності від контексту користувача та стану системи(рис. 2.3.2). Ось його основні етапи та логіка роботи:

1. Ініціалізація:
   - Запуск додатку та ініціалізація необхідних модулів.
   - Встановлення початкових параметрів збору даних.
2. Моніторинг стану системи:
   - Перевірка рівня заряду батареї.
   - Визначення стану пам'яті та процесора.
3. Аналіз контексту користувача:
   - Генерація умовних даних або зчитування даних з сенсорів.
   - Визначення поточного сценарію користувача (спокій, фізична активність тощо).
4. Оптимізація частоти збору даних:
   - Якщо користувач у стані "спокій" та система не перевантажена, зниження частоти збору даних.

- У випадку активності користувача або важливих системних подій, збільшення частоти опитування.

5. Зберігання та обробка даних:

- Зберігання зібраних даних у локальну базу даних.
- Аналіз даних для подальшого використання або відображення користувачеві.

6. Завершення роботи:

- Збереження всіх змін та параметрів.
- Закриття додатку або перехід у режим очікування.

Для експериментального оцінювання алгоритму спростимо логіку роботи, обмежуючись генерацією умовних даних які дозволять порівняти стани системи відповідно до стану активності. Для цього дозволимо встановлення початкових параметрів збору даних користувачів, а також можливості налаштування частоти стану не активності.

Експериментальний додаток буде спрямований на демонстрацію роботи алгоритму оптимізації збору даних. Основні компоненти додатку:

1Інтерфейс користувача для відображення статусу збору даних.

- Відображення тестових налаштувань для експерименту
- Відображення процесу збору даних
- Графіки, які демонструють умовні одиниці навантаження під час збору інформації.

2. Модуль алгоритму для динамічного управління частотою збору даних.

- Автоматичний режим, де алгоритм регулює частоту збору даних на основі заданих тестових налаштувань.
- Збір даних одночасно статичним та динамічним методом.
- Можливість перезаписати зібрати дані на основі інших налаштувань.

3. Модуль генерації умовних даних.

- Автоматична генерація даних, які імітують роботу реальних сенсорів.
- Встановлення параметрів для генерації даних: інтервал збору, умовні одиниці навантаження тощо.
- Генерація однакових даних для статичного та динамічного метода

Для оцінки ефективності розробленого алгоритму оптимізації збору даних буде проведено ряд експериментів. Основні етапи та критерії оцінки включають дані для збору. Умовні одиниці навантаження будуть генеруватися автоматично та імітувати реальні сценарії користувача, такі як "спокій" або "фізична активність". Частота збору даних буде записуватися як кількість опитувань сенсорів за одиницю часу. Щодо методики вимірювання ефективності, буде проведено порівняння споживаної енергії при роботі алгоритму з базовим режимом збору даних.

Для забезпечення коректної роботи та досягнення мети експерименту, додаток повинен відповідати наступним вимогам:

1. Функціональність:
   - Автоматична генерація умовних даних, що імітують реальні сценарії користувача.
   - Динамічне управління частотою збору даних на основі розробленого алгоритму.
   - Відображення поточного статусу збору даних та умовних одиниць навантаження.
   - Можливість користувача визначати свої параметри для генерації даних.
2. Обмеження:
   - Додаток повинен працювати в умовах обмеженого доступу до реальних даних.
   - Робота додатку має бути оптимізована для мінімізації споживання енергії.

- Всі дані, що генеруються або обробляються, повинні зберігатися локально без передачі на зовнішні сервери.

3. Технічні характеристики:

- Підтримка операційної системи Android, версія 8.0 та вище.
- Мінімальний обсяг оперативної пам'яті: 2 ГБ.
- Мінімальний обсяг вбудованої пам'яті: 100 МБ вільного простору.
- Підтримка інтернет-з'єднання не є обов'язковою, але може бути корисною для отримання оновлень.

На основі розробленого теоритичного алгоритму оптимізації збору даних, можна очікувати наступні показники ефективності та можливі результати:

1. Економія енергії: Завдяки адаптивному збору даних, очікується зниження споживаної енергії на 15-25% порівняно зі стандартним режимом збору даних.

2. Точність збору даних: За допомогою динамічного управління частотою опитування сенсорів, очікується підвищення точності збору даних до 90-95% відповідно до умовних сценаріїв користувача.

3. Швидкість реакції: Алгоритм повинен мати здатність швидко реагувати на зміни контексту, з переключенням режимів збору даних протягом 1-2 секунд.

Можливі труднощі та ризики:

- Технічні обмеження: Реалізація алгоритму на певних пристроях може виявитися складною через обмеження апаратного забезпечення.
- Непередбачувані сценарії користувача: Незважаючи на умовні сценарії, реальна поведінка користувача може відрізнятися, що може вплинути на ефективність алгоритму.
- Проблеми зі стабільністю: Динамічне управління частотою опитування може призвести до нестабільної роботи додатку або перевантаження системи.

Ці прогнози базуються на теоретичних розрахунках та припущеннях. Реальні результати можуть відрізнятися в залежності від умов реалізації та тестування.

В умовах розроблювального експериментального додатка основними показниками ефективності буде умовна завантаженість через процес збору даних датчиків.

3.2 Технічні аспекти та інструменти розробки

У сучасному світі мобільні технології зазнають стрімкого розвитку, а додатки стають все більш складними та функціональними. Це створює попит на ефективні інструменти та методології розробки, які б допомагали створювати високоякісні продукти. У цьому контексті розгляд технічних аспектів та інструментів розробки для мобільних платформ, зокрема Android, є актуальним та важливим для забезпечення якості та ефективності розробки.[41-42]

Операційна система Android стає відмінним вибором для експериментальної розробки додатку, спрямованого на оптимізацію збору даних з мобільних сенсорів. Її велика популярність на світовому ринку мобільних технологій забезпечує доступ до широкої аудиторії користувачів, що відкриває двері для проведення різноманітних дослідницьких ініціатив.

Однією з ключових переваг Android є її відкритий код. Така архітектура дозволяє експертам та розробникам глибоко інтегруватися з операційною системою, надаючи прямий доступ до даних з сенсорів мобільного пристрою. Ця глибока інтеграція сприяє створенню додатків, які можуть збирати, обробляти та аналізувати дані в реальному часі.

Крім того, гнучкість Android дозволяє розробникам безпосередньо взаємодіяти з апаратними компонентами пристрою, такими як акселерометр, гіроскоп, GPS та інші. Ця можливість прямої взаємодії з сенсорами гарантує отримання високоякісних даних для наукового аналізу та дослідження.

Android Studio представляє собою інтегроване середовище розробки (IDE), яке є офіційним інструментом для створення Android-додатків. Це середовище забезпечує розробників комплексом інструментів, необхідних для ефективної розробки, тестування та публікації додатків.

При розробці експериментального додатку, що спрямований на дослідження методів збору, обробки та аналізу даних з мобільних сенсорів, важливо вибрати мову програмування, яка б не тільки була потужною, але й надавала б необхідний рівень гнучкості та безпеки. В цьому контексті Kotlin виявляється відмінним вибором.

Kotlin - це сучасна мова програмування, яка була розроблена з метою вирішення деяких обмежень та проблем, характерних для Java [43]. Це робить її особливо привабливою для розробки новітніх додатків, таких як наш експериментальний проект. Однією з ключових особливостей Kotlin є її безпека. Завдяки строгій системі типів та вдосконаленій системі обробки виключень, розробники можуть бути впевнені в тому, що більшість потенційних помилок буде виявлено ще на етапі компіляції.

Крім того, Kotlin володіє відмінною сумісністю з Java. Це означає, що розробники можуть легко інтегрувати код, написаний на Kotlin, з існуючими Java-бібліотеками або навіть комбінувати обидві мови в рамках одного проекту. Така сумісність є особливо корисною при розробці складних додатків, де може бути необхідно використовувати вже існуючі рішення на Java разом з новітніми можливостями Kotlin.

При розробці експериментального додатку, що зосереджений на дослідженні методів збору, обробки та аналізу даних з мобільних сенсорів, ключовим аспектом є створення інтуїтивного та ефективного інтерфейсу користувача. В цьому контексті Compose виступає як новаторський інструмент для створення UI на платформі Android.

Однією з основних переваг Compose є його декларативний підхід до створення UI [44]. Відмінно від традиційних імперативних методів, де

розробники мають докладно описувати кожен крок для досягнення бажаного вигляду інтерфейсу, Compose дозволяє просто описати, як інтерфейс повинен виглядати. Це не тільки спрощує процес розробки, але й зменшує кількість коду, необхідного для створення UI.

Крім того, Compose було спеціально розроблено для інтеграції з Kotlin. Це означає, що розробники можуть використовувати всі переваги Kotlin, такі як безпека типів, лямбда-вирази та розширення функцій, при створенні декларативних UI компонентів. Така тісна інтеграція забезпечує ефективний та гармонійний процес розробки, що є особливо важливим при створенні експериментальних додатків.

Враховуючи вищезазначене, можна зробити висновок, що правильний вибір технічних інструментів та підходів до розробки є критично важливим для успіху будь-якого проекту. Android Studio, Kotlin та Compose представляють собою потужний набір інструментів, які дозволяють розробникам створювати ефективні, безпечні та відгукові додатки. Використання цих інструментів у комбінації з правильною методологією може значно підвищити якість та швидкість розробки, а також забезпечити високий рівень задоволеності користувачів.

3.3 Реалізація додатку: код, інтерфейс та практичне тестування

В умовах експериментальної розробки був створений мобільний додаток для демонстрації переваг використання розробленого алгоритму, який реагує на умовні сценарії користувача(активність/не активність) для динамічного управління частотою збору даних.

Відповідно до теоретичного плану додаток має наступну структуру:

- Ініціалізація. При запуску додатку ініціалізуються відповідно модулі, встановлюються параметри збору даних.

• Інтерфейс. Користувачу відображається список тестових налаштувань, доступних для зміни, статус процесу збору даних та результат експериментального запуску алгоритму з даними і графіками.

• Модуль генерації умовних даних. Реалізована автоматична генерація, встановлення параметрів, генерація однакових даних для різних методів.

• Завершення роботи. Збереження параметрів, закриття додатку або перехід в режим очікування.

В рамках програмної реалізації додаток розділений на дві основні частини: функціональну логіку та інтерфейсну реалізацію.

Для забезпечення зберігання даних, їх обробки та коректної роботи алгоритму було розроблено клас MainViewModel. При запуску додатку відбувається ініціалізація ключових змінних, які пізніше використовуються як у функціональній логіці, так і для відображення інформації на інтерфейсі.

```
// етап по якому відрізняємо вікна
var stage by mutableStateOf(Stage.Settings)
    private set
// Загальна значення для аналізу. Визначає загальну кількість процесу до якого потрібно дійти за визначений крок
var totalValue by mutableStateOf(1000)
// Кількість яку буде проходити алгоритм за одиницю часу щоб досягти загального значення
var stepForTimeUnit by mutableStateOf(30)
// Шанс, що об'єкт перебуває в статичному стані. Можливість вираховується по умові 1/N. Де N задане число в цьому полі
var randomHoldKey by mutableStateOf(5)
// Значення опрацьованої частини, в межах в 0 до значення totalValue. Використовується для екрану завантаження
var processedValue by mutableStateOf(0)
    private set
```

```
// результат алгоритму аналізу, містить інформацію про навантаження статичними та динамічним методом
var analyzedResult by mutableStateOf<AnalyzedResult?>(null)
    private set
// Допоміжна зміна для процесу обробки даних, зберігає стан процесу
private var processingTask: Job? = null
```

Наступним кроком є визначення функцій, відповідальних за обробку даних. При ініціації процесу збору даних необхідно також перезавантажити початкові параметри та активувати наступний інтерфейсний елемент

```
// Команда для запуску процесу збору даних
fun startProcessing() {
    // переходими в етап обробки
    stage = Stage.Processing
    // обнуляємо значення опрацьованої частини
    processedValue = 0
    // запускаємо завдання опрацювання
    runProcessingTask()
}
```

Процес збору даних реалізовано у відповідному методі. Спочатку ініціалізуються вихідні дані для аналізу навантаження алгоритму. Після цього запускається фонова задача, яка відповідає за генерацію та обробку даних до моменту досягнення завершальних параметрів експерименту.

```
// зупиняємо минулий таймер якщо такий був
processingTask?.cancel()
// Визначаємо списки для збору обробленого завантаження алгоритмами
// Список даних для статичного збору
val constantLoad = mutableListOf(LoadCondition.Hold)
// Список даних для динамічного збору
val dynamicLoad = mutableListOf(LoadCondition.Hold)
```

```
// Кількість запитів в режимі 'простою'
var holdRequests = 0
// створюємо завдання для роботі у фоні
processingTask = viewModelScope.launch {
    // Обробляти дані доки не пройшли всю відстань
    while (processedValue <= totalValue) {
        // Емулювати обробку даних кожну секунду
        delay(1_000)
```

На наступному етапі відбувається заповнення даними про навантаження для кожної ітерації. Окрім цього, імітується подія 'простою', що відображає моменти зупинки користувача чи системи. Ці моменти простою створюють експериментальний контекст, в якому наш алгоритм динамічного збору повинен уникати надмірного навантаження на систему мобільного пристрою.

```
    // Статичний метод завжди завантажений під час обробки
    constantLoad.add(LoadCondition.Load)
    // Генеруємо подію 'простою'
    if (isHoldRequest()) {
        // оновлюємо нове значення кількості запитів в режимі 'простою'
        holdRequests++
        // Динамічний метод в випадку 'простою' нічого не обробляє, тому він не навантажується
        dynamicLoad.add(LoadCondition.Hold)
    } else {
        // Перебуваємо в динамічному стані, тому оновлюємо успішне пересування на визначений крок
        processedValue += stepForTimeUnit
        // Динамічний метод обробляє дані, тому він навантажений
        dynamicLoad.add(LoadCondition.Load)
```

```
    }
}
```

Після завершення процесу збору даних і досягнення тестового кінцевого значення, ми формуємо результати, включаючи дані про статичний та динамічний методи збору. Після цього завершуємо завдання збору даних і переходимо до екрану відображення результатів.

```
    // Генеруємо результат роботи навантаження під час збору даних
    analyzedResult = AnalyzedResult(
        totalRequests = constantLoad.size,
        holdRequests = holdRequests,
        constantLoad = constantLoad.toList(),
        dynamicLoad = dynamicLoad.toList(),
    )
    processingTask = null

    // переходимо на етап показу результатів
    stage = Stage.Result

}
```

В інтерфейсній частині додатку реалізовано три екрани, які були заплановані на теоретичному етапі. Під час запуску додатку користувача зустрічає екран налаштувань алгоритму. Згідно рисунка 3.3.1, на цьому екрані можна змінювати параметри, такі як загальне значення для аналізу, інтервал часу для збору даних та активувати функцію випадкової зупинки. Всі ці параметри безпосередньо впливають на процес отримання експериментальних даних, тривалість збору інформації, обсяг отриманих результатів та ймовірність виникнення простою в роботі додатку.

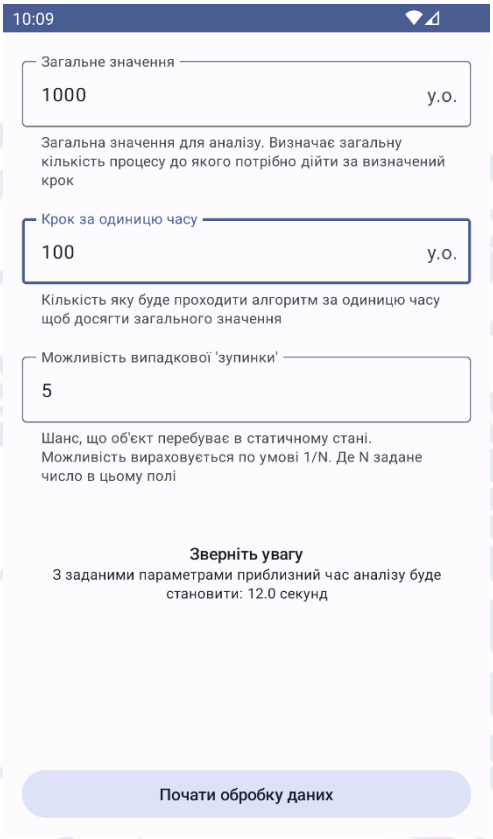

Рис. 3.3.1 Екран налаштувань

Після встановлення потрібних параметрів для збору даних, користувачу пропонується розпочати обробку, натиснувши кнопку «Почати обробку даних». Як тільки фонова задача розпочинається, користувач переходить на екран процесу збору даних, як показано на рисунку 3.3.2. Цей екран відображає прогрес завантаження, а також загальне та поточне значення аналізу.

Закінчивши обробку, користувачу представляється звіт про результати на новому екрані, зображеному на рисунку 3.3.3. Тут можна ознайомитися зі статистикою по кількості запитів, як загалом, так і в статичному режимі. Додатково, екран надає докладну інформацію про навантаження на пристрій під час збору даних за допомогою статичного та динамічного методів. Ця інформація представлена у вигляді двох графіків: перший, відображений червоним кольором, показує результати без використання динамічного збору, тоді як другий графік демонструє результати з динамічним збором.

На цьому ж екрані розташована кнопка, яка дозволяє повернутися до екрану налаштувань і, за бажанням, повторно запустити процес збору з іншими параметрами.

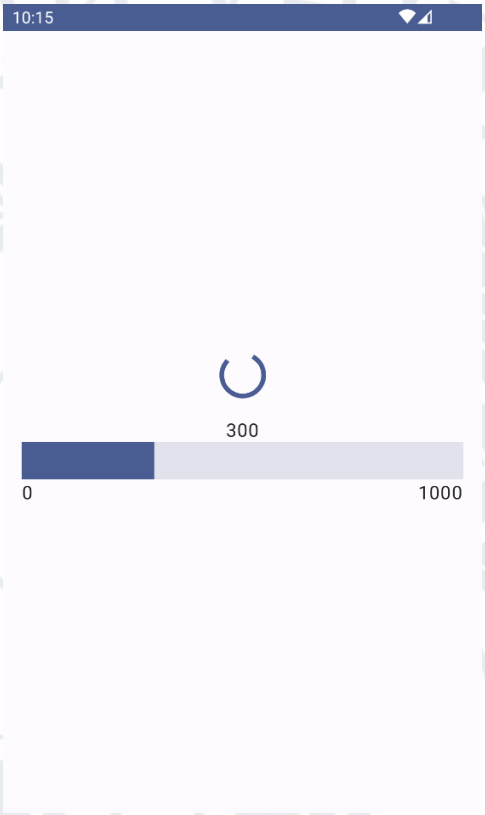

Рис. 3.3.2 Екран процесу збору даних

Рис. 3.3.3. Екран результату алгоритму

При детальному розгляді кожного графіка можна виявити очікуванні характеристики завантаженості системи.

На графіку завантаженості без динамічного збору даних, як показано на рисунку 3.3.4, спостерігається стабільна активність. Це свідчить про те, що ресурси пристрою використовуються безперервно, незалежно від дій користувача чи стану системи. Тобто, в періоди, коли користувач або система не активні, дані продовжують реєструватися на кожному етапі збору.

Рис. 3.3.4. Графік завантаженості без використання динамічного методу збору даних

З іншого боку, графік завантаженості, який базується на пропонованому алгоритмі, демонструє відмінності в патернах завантаженості. Яскраво виділяються моменти, коли навантаження на систему знижується. Це відбувається тоді, коли користувач або система перебувають у пасивному стані. В такі моменти алгоритм призупиняє збір даних, зменшуючи тим самим навантаження на пристрій. Такий підхід оптимізує використання ресурсів та забезпечує більш ефективну роботу додатку.

Рис. 3.3.5 Графік завантаженості з використанням динамічного методу збору даних

Висновки

Даний розділ комплексно підходить до розробки додатку, включаючи теоретичні, технічні та практичні аспекти. Він наглядно демонструє процес роботи додатку, його структурні та інтерфейсні особливості, а також підтверджує ефективність використання динамічного збору даних для оптимізації ресурсів мобільного пристрою.

На початку розділу було визначено основні концепції та теоретичні засади, на яких базується розробка додатку. Обговорено методологію, цілі та завдання, які стояли перед розробниками.

Далі розглянуто основні технології, мови програмування та інструменти, які були використані під час розробки. Описано особливості кожного інструменту та його роль у процесі створення додатку.

На кінець, у розділі було розглянуто структурну та інтерфейсну частини розробленого додатку. Визначено основні етапи роботи програми, від налаштувань параметрів до відображення результатів збору даних. Особлива увага приділена аналізу завантаженості системи з використанням та без використання динамічного збору даних. Виявлено, що динамічний збір даних дозволяє оптимізувати використання ресурсів пристрою, зменшуючи навантаження в періоди не активності користувача або системи.

ВИСНОВКИ

В першому розділі було проведено огляд і дослідження існуючих типів мобільних сенсорів. Проаналізовано різноманітні типи датчиків та методи їх збору, які можуть бути використані для збору даних з мобільних пристроїв.

У другому розділі було виявлено, що сучасні системи збору даних з мобільних сенсорів стикаються з високими енергетичними витратами та обчислювальними обмеженнями. Основні проблеми існуючих методів включають високі затрати ресурсів, недостатню точність та питання конфіденційності. Було наведено нові методи та підходи, зокрема алгоритми для динамічного управління частотою збору даних, які дозволяють оптимізувати енергетичні витрати та забезпечити точність зібраної інформації.

У третьому розділі було розглянуто комплексний підхід до розробки додатку, включаючи його структурні та інтерфейсні особливості. Додаток базується на сучасних технологіях та інструментах, і його робота демонструє ефективність використання динамічного збору даних для оптимізації ресурсів мобільного пристрою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ

1. Android Developers. Sensors overview. URL: https://developer.android.com/guide/topics/sensors/sensors_overview

2. Research Methods Knowledge Base by Prof William M.K. Trochim. URL: https://conjointly.com/kb/table-of-contents/

3. Charith Pereral , Arkady Zaslavsky, Peter Christen, Ali Salehi and Dimitrios Georgakopoulos. Capturing Sensor Data from Mobile Phones using Global Sensor NetworkMiddleware URL: https://www.researchgate.net/publication/234017923_Capturing_Sensor_Data_from_Mobile_Phones_using_Global_Sensor_NetworkMiddleware

4. Types Of Sensors In a Smartphone. URL: https://iasetraining.org/types-of-sensors-in-smartphone/

5. P. Delgado-Santos, G. Stragapede, R. Tolosana, R. Guest, F. Deravi, R. Vera-Rodriguez (2022). A Survey of Privacy Vulnerabilities of Mobile Device Sensors, URL: https://dl.acm.org/doi/10.1145/3510579

6. Sensor stack. URL: https://source.android.com/docs/core/interaction/sensors/sensor-stack

7. George Lawton. data preprocessing. URL: https://www.techtarget.com/searchdatamanagement/definition/data-preprocessing

8. What Is Data Preprocessing & What Are The Steps Involved? URL: https://monkeylearn.com/blog/data-preprocessing/

9. Chernyak O.I., Zakharchenko P.V. (2019). ОСНОВНІ ПОНЯТТЯ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ. URL: https://moodle.znu.edu.ua/pluginfile.php/481676/mod_resource/content/2/%D0%9B%D0%B5%D0%BA%D1%86%D1%96%D1%8F%201.pdf

10. Li Canchen (2019). Preprocessing Methods and Pipelines of Data Mining: An Overview. URL: https://arxiv.org/pdf/1906.08510.pdf

11. Ihab F Ilyas, Xu Chu. Data Cleaning (2019)

12. Guide To Data Cleaning: Definition, Benefits, Components, And How To Clean Your Data. URL: https://www.tableau.com/learn/articles/what-is-data-cleaning

13. M. H. Alsharif, A. H. Kelechi, K. Yahya, S. A. Chaudhry (2020). Machine Learning Algorithms for Smart Data Analysis in Internet of Things Environment: Taxonomies and Research Trends. URL: https://www.mdpi.com/2073-8994/12/1/88

14. Kishor H. Walse, Rajiv V. Dharaskar and Vilas M. Thakare.(2016). PCA Based Optimal ANN Classifiersfor Human Activity Recognition UsingMobile Sensors Data. URL: https://www.researchgate.net/publication/304621634_PCA_Based_Optimal_ANN_Classifiers_for_Human_Activity_Recognition_Using_Mobile_Sensors_Data

15. Charanraj Shetty (2020). Time Series Models. URL: https://towardsdatascience.com/time-series-models-d9266f8ac7b0

16. Dinesh Asanka (2021). Internals of Physical Join Operators. URL: https://www.sqlshack.com/internals-of-physical-join-operators-nested-loops-join-hash-match-join-merge-join-in-sql-server/

17. Simon Tavasoli (2023). Top 10 Machine Learning Algorithms For Beginners: Supervised, and More. URL: https://www.simplilearn.com/10-algorithms-machine-learning-engineers-need-to-know-article

18. Dheeraj Vaidya (2023). Statistical Analysis. URL: https://www.wallstreetmojo.com/statistical-analysis/

19. Time Series Analysis and Forecasting: Examples, Approaches, and Tools. URL: altexsoft.com/blog/business/time-series-analysis-and-forecasting-novel-business-perspectives/

20. Geospatial data definition (2020). URL: https://www.ibm.com/topics/geospatial-data

21. C. Morikawa, M. Kobayashi, M. Satoh, Y. Kuroda, T. Inomata, H. Matsuo, T. Miura, M. Hilaga. Image and video processing on mobile devices: a survey. URL: https://link.springer.com/article/10.1007/s00371-021-02200-8

22. The Data Visualisation Catalogue. URL: https://datavizcatalogue.com/

23. What Is Data Visualization? Definition, Examples, And Learning Resources . URL: https://www.tableau.com/learn/articles/data-visualization

24. What is data visualization? URL: https://www.ibm.com/topics/data-visualization

25. Yogita Deepak Mane (2019). Protection concern in Mobile Cloud Computing-A Survey. URL: https://www.researchgate.net/publication/330753437_Protection_concern_in_Mobile_Cloud_Computing-A_Survey

26. Sunil Choenni, Mortaza S. Bargh, Carmelita Roepan, and Ronald Meijer (2015). Privacy and security in data collection by citizens. URL: https://www.researchgate.net/publication/282646818_Privacy_and_security_in_data_collection_by_citizens.

27. Brij B. Gupta, Akshat Gaurav, Prabin Kumar Panigrahi (2023). Analysis of security and privacy issues of information management of big data in B2B based healthcare systems. URL: https://doi.org/10.1016/j.jbusres.2023.113859

28. N. Khan, S. Khusro, S. Ali, J. Ahmad (2016). Sensors are Power Hungry: An Investigation of Smartphone Sensors Impact on Battery Power from Lifelogging Perspective. URL: https://www.researchgate.net/publication/317494054_Sensors_are_Power_Hungry_An_Investigation_of_Smartphone_Sensors_Impact_on_Battery_Power_from_Lifelogging_Perspective.

29. Limits to computing: A computer scientist explains why even in the age of AI, some problems are just too difficult (2023). URL: https://theconversation.com/limits-to-computing-a-computer-scientist-explains-why-even-in-the-age-of-ai-some-problems-are-just-too-difficult-191930

30. Evaluating the Energy Consumption of Mobile Data Transfer—From Technology Development to Consumer Behaviour and Life Cycle Thinking (2018). URL: http://dx.doi.org/10.3390/su10072494

31. Natalia Vélez, Brian Christian, Mathew Hardy, Bill D. Thompson, and Thomas L. Griffiths. (2023). How do Humans Overcome Individual Computational Limitations by Working Together? URL: https://ncbi.nlm.nih.gov/pmc/articles/PMC10334258/

32. Ahmet Karamana, Doga Erisika, Ozlem Durmaz Incela, Gulfem Isiklar Alptekina (2016). Resource Usage Analysis of a Sensor-Based Mobile Augmented Reality Application. URL: https://doi.org/10.1016/j.procs.2016.04.129

33. .Xu, Xia, Jin Tang, H. Xiang (2022). Data Transmission Reliability Analysis of Wireless Sensor Networks for Social Network Optimization. URL: https://downloads.hindawi.com/journals/js/2022/3842722.pdf

34. Tang, Xianzhi, C. Ding (2021). Information Security Terminal Architecture of Power Transportation Mobile Internet of Things Based on Big Data Analysis. URL: https://downloads.hindawi.com/journals/wcmc/2021/5544716.pdf

35. Xiaolong Bai, Jie Yin and Yu-Ping Wang (2017). Sensor Guardian: prevent privacy inference on Android sensors. URL: https://jis-eurasipjournals.springeropen.com/articles/10.1186/s13635-017-0061-8

36. Rehman, A., K. Haseeb, T. Saba, Jaime Lloret, S. Sendra (2021). An Optimization Model with Network Edges for Multimedia Sensors Using Artificial Intelligence of Things. URL: https://downloads.hindawi.com/journals/wcmc/2021/5544716.pdf

37. Li, Xian, S. Bi, Z. Quan, Hui Wang (2021). Online Cognitive Data Sensing and Processing Optimization in Energy-harvesting Edge Computing Systems. URL: https://arxiv.org/pdf/2106.14113.pdf

38. A. Capponi, C. Fiandrino, D. Kliazovich, P. Bouvry, S. Giordano (2017). A Cost-Effective Distributed Framework for Data Collection in Cloud-Based Mobile Crowd Sensing Architectures. **DOI:** 10.1109/TSUSC.2017.2666043

39. I. M. Pires, N. M. Garcia, N. Pombo, F. Flórez-Revuelta (2016). URL: https://doi.org/10.3390/s16020184

40. K. Haseeb, I. Ud Din, A. Almogren, Z. Jan, N. Abbas, M. Adnan (2020). URL: https://ieeexplore.ieee.org/document/9220160

41. What is android (2023). URL: https://www.android.com/what-is-android/

42. Android Operating System (2021). URL: https://www.javatpoint.com/android-operating-system

43. Kotlin (2023). URL: https://kotlinlang.org/

44. Compose Overview (2023). URL: https://developer.android.com/jetpack/compose

ДОДАТКИ

ДОДАТОК А.

Класифікація можливих типів мобільних датчиків

Sensor Type	Sensor/Data Source	Measured/Logged Quantity	Scope/Purpose	Sensor Type
Motion	Accelerometer/ Linear Accelerometer	Acceleration Force	Device Translation	Hardware
	Gyroscope	Angular Velocity	Device Rotation	Hardware
	Rotation Vector	Angle	Device Orientation	Hardware, Software
	Gravity	Magnitude of Gravity	Device Orientation	Hardware, Software
	Significant Motion	Change of User Movement	Walking or Riding Vehicle	Software
	Step Counter	Number of Steps	Physical Activity Tracking	Software
	Step Detector	Step	Physical Activity Tracking	Software
Position	Geomagnetic Field	Earth's Magnetic Field	Device Orientation	Hardware

	Proximity	Distance	Device Distance from Surface	Hardware
	Magnetometer	Earth's Magnetic Field	Device Orientation	Hardware
	Geomagnetic Rotation Vector	Earth's Magnetic Field	Device Orientation	Hardware, Software
	Game Rotation Vector	Angle	Device Rotation	Hardware, Software
Environmental	Light	Illuminance	Screen Luminosity Regulation	Hardware
	Pressure	Ambient Pressure	Contextual Information	Hardware
	Temperature	Ambient Temperature	Contextual Information	Hardware
	Humidity	Ambient Humidity	Contextual information	Hardware
Health	BPM	Number of Beats	Physical Activity Monitoring	Hardware
	ECG	Sinus Rhythm Graph	Physical Activity Monitoring	Hardware

	SpO22	Arterial Blood Oxygen Saturation Percentage Level	Physical Activity Monitoring	Hardware
	Blood Pressure	Systolic and Diastolic Average Pressure	Physical Activity Monitoring	Software
	Stress	Percentage Based on Heart Beat Variability	Physical Activity Monitoring	Software
	Sleep/Wake Amount	Time	Physical Activity Monitoring	Hardware, Software
	Sleep Phase Transitions	Time	Physical Activity Monitoring	Hardware, Software
	Caloric Consumption	Step Counter	Physical Activity Monitoring	Software
Touchscreen	Keystroke	Keys Presses and Releases	Key Input	Hardware
	Touch Data	Screen Coordinates, Pressure of Touch	Complex Touch Gestures	Hardware
Network, Location and Application	Wi-Fi	SSID, RSSI, Encryption Protocol, Frequency, Channel	Connectivity	Hardware

	Bluetooth	SSID, RSSI, Encryption Protocol, Frequency, Channel	Connectivity	Hardware
	Cell Tower	ID	Connectivity	Hardware
	GPS	Latitude, Longitude, Altitude, Bearing, Accuracy	Navigation	Hardware
	App Usage	Name and Time of Used Apps	System Log	Software

ДОДАТОК Б.

Класифікація сенсорів за категоріями

Category	External Sensors	Mobile Sensors
Magnetic/Mechanical sensors	Compass; Magnetometer; Strain sensors; Search-coil magnetometer; Fluxgate magnetometer; Superconductor magnetometers; Hall effect sensor; Magnetoresistive magnetometers; Spin-valve transistors; Giant magnetoimpedance magnetic sensors; Magnetodiode; Magnetotransistor; Magnetostrictive magnetometers; Magneto-optical sensor; MEMS Based Magnetometers; Ball/tilt/foot switch; Sole pressure switch; Pressure sensors; Contact sensors; Mechanical switches	Magnetometer; Compass
Environmental sensors	Barometer; Humidity; Light sensor; Thermal sensor	Ambient air temperature and pressure; Light Sensor; Humidity Barometer; Photometer; Thermal sensor
Location sensors	GPS receiver; Automatic Vehicle Identification (AVI) readers; Ubisense Real-Time Location Systems (RTLS); Wi-Fi Location-Based Services	GPS receiver; Wi-Fi Location-Based Services
Motion sensors	Accelerometer; Gyroscope; Pressure sensor; Gravity sensor; Inclinometer; Pedometer; Rotational sensor	Accelerometer; Gravity sensor; Gyroscope; Rotational vector sensor; Orientation sensor
Imaging/Video sensors	Digital camera; 3D camera; Optical sensor; Infrared sensor	Digital camera; Infrared sensor
Proximity sensors	Proximity sensor; Touch sensor; RFID; Tactile sensor; NFC	Proximity sensor; Touch sensor; RFID; NFC
Acoustic sensors	Microphone; Silicon microphones; Acoustic wave devices; Surface acoustic wave	Microphone
Medical sensors	EEG; ECG; EMG; EOG; EDA; Photoplethysmogram; Blood pressure and arterial	None

	tonometry; Respiration; Dosage control/detection; Stress sensors; Heart rate sensors; electrooculography; electrodermal activity sensors	
Chemical sensors	Oxygen saturation; Aroma sensors; Metal-oxide; Semi conductive polymers; Conductive electro active polymers; Electrochemical gas sensors; Actinometer	None
Optical sensors	Photoplethysmography sensors; Fiber optic sensors; Infrared sensors; Radio frequency sensors	Infrared sensors; Radio frequency sensors
Force sensors	Force sensitive resistor; Mass sensor; Fingerprint sensor	Fingerprint sensor
Photoelectric sensors	Oximeter	None

ДОДАТОК В.

Лістинг програмної реалізації

MainActivity.kt

```
package com.hromovych.dynamicdatacollection
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.lifecycle.viewmodel.compose.viewModel
import com.hromovych.dynamicdatacollection.ui.screens.processing.ProcessingScreen
import com.hromovych.dynamicdatacollection.ui.screens.result.ProcessingResultScreen
import com.hromovych.dynamicdatacollection.ui.screens.settings.SettingsScreen
import com.hromovych.dynamicdatacollection.ui.theme.DynamicDataCollectionTheme
// Активність - головний елемент інтерфейсної системи Android
class MainActivity : ComponentActivity() {
    // визначаємо що потрібно зробити при створенні цієї активності
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // Призначаємо їй інтерфейсний контент, з налаштованою темою, та поверхньою для
всього доступного вікна телефону
        setContent {
            DynamicDataCollectionTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    MainScreen()
                }
            }
        }
    }
}
// Головна composable функція нашого екрану. Де залежно від поточного етапу
відображається відповідний екран
// Детальніше про кожний екран в середині відповідної compose функції
@Composable
private fun MainScreen(
    // Логіка алгоритму і зберігання даних відбувається в окремому класі MainViewModel
    viewModel: MainViewModel = viewModel()
) {
    when (viewModel.stage) {
        MainViewModel.Stage.Settings -> SettingsScreen(
            totalAmount = viewModel.totalValue,
```

```
            updateTotalAmount = { viewModel.totalValue = it },
            stepForTimeUnit = viewModel.stepForTimeUnit,
            updateStepForTimeUnit = { viewModel.stepForTimeUnit = it },
            randomHoldKey = viewModel.randomHoldKey,
            updateRandomStep = { viewModel.randomHoldKey = it },
            startDataProcessing = viewModel::startProcessing,
        )
        MainViewModel.Stage.Processing -> ProcessingScreen(
            totalValue = viewModel.totalValue,
            currentValue = viewModel.processedValue,
        )
        MainViewModel.Stage.Result -> viewModel.analyzedResult?.let { result ->
            ProcessingResultScreen(
                analyzedResult = result,
                openSettings = viewModel::backToSettings
            )
        }
    }
}
```

MainViewModel.kt

```
package com.hromovych.dynamicdatacollection
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.setValue
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.hromovych.dynamicdatacollection.domain.AnalyzedResult
import com.hromovych.dynamicdatacollection.domain.LoadCondition
import kotlinx.coroutines.Job
import kotlinx.coroutines.delay
import kotlinx.coroutines.launch
import kotlin.random.Random
class MainViewModel : ViewModel() {
    // етап по якому відрізняємо вікна
    var stage by mutableStateOf(Stage.Settings)
        private set
    // Загальна значення для аналізу. Визначає загальну кількість процесу до якого потрібно
дійти за визначений крок
    var totalValue by mutableStateOf(1000)
    // Кількість яку буде проходити алгоритм за одиницю часу щоб досягти загального
значення
    var stepForTimeUnit by mutableStateOf(30)
    // Шанс, що об'єкт перебуває в статичному стані. Можливість вираховується по умові 1/N.
Де N задане число в цьому полі
    var randomHoldKey by mutableStateOf(5)
    // Значення опрацьованої частини, в межах в 0 до значення totalValue. Використовується
для екрану завантаження
    var processedValue by mutableStateOf(0)
        private set
    // результат алгоритму аналізу, містить інформацію про навантаження статичними та
```

```
динамічним методом
    var analyzedResult by mutableStateOf<AnalyzedResult?>(null)
        private set
    // Допоміжна зміна для процесу обробки даних, зберігає стан процесу
    private var processingTask: Job? = null
    // Команда для запуску процесу збору даних
    fun startProcessing() {
        // переходими в етап обробки
        stage = Stage.Processing
        // обнуляємо значення опрацьованої частини
        processedValue = 0
        // запускаємо завдання опрацювання
        runProcessingTask()
    }
    private fun runProcessingTask() {
        // зупиняємо минулий таймер якщо такий був
        processingTask?.cancel()
        // Визначаємо списки для збору обробленого завантаження алгоритмами
        // Список даних для статичного збору
        val constantLoad = mutableListOf(LoadCondition.Hold)
        // Список даних для динамічного збору
        val dynamicLoad = mutableListOf(LoadCondition.Hold)
        // Кількість запитів в режимі 'простою'
        var holdRequests = 0
        // створюємо завдання для роботі у фоні
        processingTask = viewModelScope.launch {
            // Обробляти дані доки не пройшли всю відстань
            while (processedValue <= totalValue) {
                // Емулювати обробку даних кожну секунду
                delay(1_000)
                // Статичний метод завжди завантажений під час обробки
                constantLoad.add(LoadCondition.Load)
                // Генеруємо подію 'простою'
                if (isHoldRequest()) {
                    // оновлюємо нове значення кількості запитів в режимі 'простою'
                    holdRequests++
                    // Динамічний метод в випадку 'простою' нічого не обробляє, тому він не
навантажується
                    dynamicLoad.add(LoadCondition.Hold)
                } else {
                    // Перебуваємо в динамічному стані, тому оновлюємо успішне пересування на
визначений крок
                    processedValue += stepForTimeUnit
                    // Динамічний метод обробляє дані, тому він навантажений
                    dynamicLoad.add(LoadCondition.Load)
                }
            }
            // Генеруємо результат роботи навантаження під час збору даних
            analyzedResult = AnalyzedResult(
                totalRequests = constantLoad.size,
                holdRequests = holdRequests,
```

```
                constantLoad = constantLoad.toList(),
                dynamicLoad = dynamicLoad.toList(),
            )
            processingTask = null
            // переходимо на етап показу результатів
            stage = Stage.Result
        }
    }
    // Повертаємося назад до етапу налаштувань, для можливості повторного запуску
    fun backToSettings() {
        stage = Stage.Settings
    }
    // Генерація експерементальної зупинки, на основі заданого ключа randomHoldKey
    private fun isHoldRequest(): Boolean {
        return Random.nextInt(randomHoldKey) == 0
    }
    // Допоміжний клас для обробки поточного екрану
    enum class Stage {
        Settings, Processing, Result
    }
}
```

ProcessingScreen.kt

```
package com.hromovych.dynamicdatacollection.ui.screens.processing
import androidx.compose.animation.core.LinearOutSlowInEasing
import androidx.compose.animation.core.animateFloatAsState
import androidx.compose.animation.core.tween
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.CircularProgressIndicator
import androidx.compose.material3.LinearProgressIndicator
import androidx.compose.material3.ProgressIndicatorDefaults
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import com.hromovych.dynamicdatacollection.ui.theme.DynamicDataCollectionTheme
// Екран процесу завантаження, відображаю користувачу поточний процес, загальну
кількість, інформує про роботу алгоритму
@Composable
fun ProcessingScreen(
    totalValue: Int,
```

```
    currentValue: Int,
    modifier: Modifier = Modifier,
) {
    val progress by animateFloatAsState(
        targetValue = currentValue / totalValue.toFloat(),
        tween(
            durationMillis = 200,
            delayMillis = 1,
            easing = LinearOutSlowInEasing
        ), label = "progress"
    )
    Column(
        modifier = modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        CircularProgressIndicator()
        Spacer(Modifier.height(16.dp))
        Text(text = currentValue.toString())
        LinearProgressIndicator(
            progress = progress,
            strokeCap = ProgressIndicatorDefaults.LinearStrokeCap,
            modifier = Modifier
                .fillMaxWidth()
                .height(32.dp)
        )
        Row(
            modifier = Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.SpaceBetween
        ) {
            Text("0")
            Text(totalValue.toString())
        }
    }
}
@Preview(showBackground = true)
@Composable
private fun ProcessingDialogPreview() {
    DynamicDataCollectionTheme {
        ProcessingScreen(totalValue = 100, currentValue = 25, modifier = Modifier.fillMaxSize())
    }
}
```

SettingsScreen.kt

```
package com.hromovych.dynamicdatacollection.ui.screens.settings
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.defaultMinSize
```

```
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material3.FilledTonalButton
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import com.hromovych.dynamicdatacollection.R
import com.hromovych.dynamicdatacollection.ui.screens.components.AmountInputField
import com.hromovych.dynamicdatacollection.ui.theme.DynamicDataCollectionTheme
// Початковий екран налаштувань, де користувач самостійно може встановити допустимі
параметри для експерементального запуска алгоритму
// Визначаються поля для редагування значень, а також кнопка для запуску початку процесу
startDataProcessing
@Composable
fun SettingsScreen(
    totalAmount: Int,
    updateTotalAmount: (Int) -> Unit,
    stepForTimeUnit: Int,
    updateStepForTimeUnit: (Int) -> Unit,
    randomHoldKey: Int,
    updateRandomStep: (Int) -> Unit,
    startDataProcessing: () -> Unit,
    modifier: Modifier = Modifier
) {
    Column(
        modifier = modifier
            .fillMaxSize()
            .verticalScroll(rememberScrollState())
            .padding(16.dp),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.spacedBy(16.dp)
    ) {
        AmountInputField(
            value = totalAmount,
            updateValue = updateTotalAmount,
            title = stringResource(R.string.total_amount),
            unitStr = stringResource(R.string.condition_unit_short),
            supportingStr = stringResource(R.string.total_amount_supporting_hint),
            modifier = Modifier.fillMaxWidth()
        )
        AmountInputField(
```

```
            value = stepForTimeUnit,
            updateValue = updateStepForTimeUnit,
            title = stringResource(R.string.step_by_time_unit),
            unitStr = stringResource(R.string.condition_unit_short),
            supportingStr = stringResource(R.string.step_by_unit_time_supporting_hint),
            modifier = Modifier.fillMaxWidth()
        )
        AmountInputField(
            value = randomHoldKey,
            updateValue = updateRandomStep,
            title = stringResource(R.string.possibility_of_random_hold),
            supportingStr = stringResource(R.string.possibility_of_random_hold_hint),
            modifier = Modifier.fillMaxWidth()
        )
        Spacer(modifier = Modifier.height(16.dp))
        Column(
            horizontalAlignment = Alignment.CenterHorizontally,
        ) {
            Text(text = stringResource(R.string.pay_attention), style =
MaterialTheme.typography.titleSmall)
            Text(
                text = stringResource(
                    R.string.approximate_analyse_time_message,
                    totalAmount / stepForTimeUnit * (1.0 / randomHoldKey + 1)
                ),
                style = MaterialTheme.typography.bodySmall,
                textAlign = TextAlign.Center
            )
        }
        Spacer(
            modifier = Modifier
                .weight(1f)
                .defaultMinSize(minWidth = 8.dp)
        )
        FilledTonalButton(onClick = { startDataProcessing() }, modifier = Modifier.fillMaxWidth()) {
            Text(text = stringResource(R.string.start_data_processing))
        }
    }
}
@Preview(showBackground = true)
@Composable
private fun SettingsScreenPreview() {
    DynamicDataCollectionTheme {
        SettingsScreen(
            totalAmount = 100,
            updateTotalAmount = {},
            stepForTimeUnit = 20,
            updateStepForTimeUnit = {},
            randomHoldKey = 5,
            updateRandomStep = {},
            startDataProcessing = {},
```

```
        )
    }
}
```

ProcessingResultScreen.kt

```
package com.hromovych.dynamicdatacollection.ui.screens.result
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.defaultMinSize
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material3.Divider
import androidx.compose.material3.FilledTonalButton
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import co.yml.charts.ui.linechart.model.LineStyle
import com.hromovych.dynamicdatacollection.R
import com.hromovych.dynamicdatacollection.domain.AnalyzedResult
import com.hromovych.dynamicdatacollection.domain.LoadCondition
import com.hromovych.dynamicdatacollection.ui.screens.components.RequestsChartItem
import com.hromovych.dynamicdatacollection.ui.theme.DynamicDataCollectionTheme
// Екран результату роботи алгоритму. Дозволяє переглянути загальну інформацію по
виконаному збору, кількість запитів в статичному режимі
// Відображає графіки роботи без використання динамічного збору даних та з його
використанням
@Composable
fun ProcessingResultScreen(
    analyzedResult: AnalyzedResult,
    openSettings: () -> Unit,
    modifier: Modifier = Modifier
) {
    Column(
        modifier = modifier
            .fillMaxSize()
            .padding(16.dp)
            .verticalScroll(rememberScrollState()),
        verticalArrangement = Arrangement.spacedBy(16.dp)
    ) {
        Text(text = stringResource(R.string.total_requests_count, analyzedResult.totalRequests))
        Text(text = stringResource(
            R.string.request_count_in_static_mode,
            analyzedResult.holdRequests
```

```
        ))
        Divider()
        RequestsChartItem(
            title = stringResource(R.string.load_without_dynamic_collection),
            data = analyzedResult.constantLoad,
            lineStyle = LineStyle(color = Color.Red)
        )
        Divider()
        RequestsChartItem(
            title = stringResource(R.string.load_with_dynamic_collection),
            data = analyzedResult.dynamicLoad,
            lineStyle = LineStyle(color = Color.Green)
        )
        Spacer(
            modifier = Modifier
                .weight(1f)
                .defaultMinSize(minWidth = 24.dp)
        )
        FilledTonalButton(onClick = { openSettings() }, modifier = Modifier.fillMaxWidth()) {
            Text(text = stringResource(R.string.back_to_settings))
        }
    }
}
@Preview(showBackground = true)
@Composable
private fun ProcessingResultScreenPreview() {
    DynamicDataCollectionTheme {
        val fakeData =
            listOf(LoadCondition.Hold) + List(30) { LoadCondition.Load }
        val holdRequests = fakeData.size / 5
        ProcessingResultScreen(
            AnalyzedResult(
                totalRequests = fakeData.size,
                holdRequests = holdRequests,
                constantLoad = fakeData.toList(),
                dynamicLoad = fakeData.mapIndexed { index, condition ->
                    if (index % 5 == 0) LoadCondition.Hold else condition
                }
            ),
            openSettings = {},
            modifier = Modifier.fillMaxSize()
        )
    }
}
```

AmountInputField.kt

```
package com.hromovych.dynamicdatacollection.ui.screens.components
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material3.OutlinedTextField
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
```

```
import androidx.compose.ui.Modifier
import androidx.compose.ui.text.input.KeyboardType
// Допоміжний Ui елемент для відображення поля введення кількості
@Composable
fun AmountInputField(
    value: Int,
    updateValue: (Int) -> Unit,
    title: String,
    modifier: Modifier = Modifier,
    unitStr: String = "",
    supportingStr: String = "",
    minValue: Int = 1
) {
    OutlinedTextField(
        value = value.toString(),
        onValueChange = {
            val intValue = it.toIntOrNull()
            val resultAmount = when {
                it.isEmpty() -> minValue
                intValue == null -> value
                intValue < minValue -> minValue
                else -> intValue
            }
            if (resultAmount != value) {
                updateValue(resultAmount)
            }
        },
        keyboardOptions = KeyboardOptions.Default.copy(keyboardType = KeyboardType.Decimal),
        label = {
            Text(text = title)
        },
        singleLine = true,
        trailingIcon = {
            Text(text = unitStr)
        },
        supportingText = {
            Text(text = supportingStr)
        },
        modifier = modifier
    )
}
```

RequestsChartItem.kt

```
package com.hromovych.dynamicdatacollection.ui.screens.components
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
```

```
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import co.yml.charts.axis.AxisData
import co.yml.charts.common.model.AccessibilityConfig
import co.yml.charts.common.model.Point
import co.yml.charts.ui.linechart.LineChart
import co.yml.charts.ui.linechart.model.GridLines
import co.yml.charts.ui.linechart.model.IntersectionPoint
import co.yml.charts.ui.linechart.model.Line
import co.yml.charts.ui.linechart.model.LineChartData
import co.yml.charts.ui.linechart.model.LinePlotData
import co.yml.charts.ui.linechart.model.LineStyle
import co.yml.charts.ui.linechart.model.ShadowUnderLine
import com.hromovych.dynamicdatacollection.R
import com.hromovych.dynamicdatacollection.domain.LoadCondition
import com.hromovych.dynamicdatacollection.ui.theme.DynamicDataCollectionTheme
// Допоміжний елемент для відображення графікува отриманих данних
@Composable
fun RequestsChartItem(
    title: String,
    data: List<LoadCondition>,
    modifier: Modifier = Modifier,
    lineStyle: LineStyle = LineStyle()
) {
    Column(modifier) {
        Text(text = title)
        Spacer(modifier = Modifier.height(4.dp))
        RequestsChart(data = data, lineStyle = lineStyle)
    }
}
@Composable
private fun RequestsChart(
    data: List<LoadCondition>,
    lineStyle: LineStyle = LineStyle()
) {
    val context = LocalContext.current
    val pointsData = data.mapIndexed { index, condition ->
        Point(x = index.toFloat(), y = condition.ordinal.toFloat())
    }
    LineChart(
        modifier = Modifier
            .fillMaxWidth()
            .height(300.dp),
        lineChartData = LineChartData(
            linePlotData = LinePlotData(
                lines = listOf(
                    Line(
                        dataPoints = pointsData,
                        shadowUnderLine = ShadowUnderLine(color = lineStyle.color.copy(alpha = .3f)),
```

```
                intersectionPoint = IntersectionPoint(color = lineStyle.color.copy(alpha = 0.8f)),
                lineStyle = lineStyle
            )
        )
    ),
    xAxisData = AxisData.Builder()
        .axisStepSize(25.dp)
        .steps(pointsData.size - 1)
        .labelData { it.toString() }
        .labelAndAxisLinePadding(10.dp)
        .build(),
    yAxisData = AxisData.Builder()
        .steps(LoadCondition.values().size - 1)
        .labelAndAxisLinePadding(10.dp)
        .topPadding(4.dp)
        .labelData {
            when (LoadCondition.values()[it]) {
                LoadCondition.Load -> context.getString(R.string.load_condition_title)
                LoadCondition.Hold -> context.getString(R.string.hold_condition_title)
            }
        }
        .build(),
    gridLines = GridLines()
)
)
}
@Preview
@Composable
private fun RequestsChartPreview() {
    DynamicDataCollectionTheme {
        RequestsChart(
            data = (1..20).map {
                if (it % 5 == 0) LoadCondition.Hold else LoadCondition.Load
            }
        )
    }
}
```

AnalyzedResult.kt

```
package com.hromovych.dynamicdatacollection.domain
// Допоміжний клас для визначення результату роботи алгоритму
data class AnalyzedResult(
    val totalRequests: Int,
    val holdRequests: Int,
    val constantLoad: List<LoadCondition>,
    val dynamicLoad: List<LoadCondition>
)
```

LoadCondition.kt

```
package com.hromovych.dynamicdatacollection.domain
// Допоміжний клас для визначення стану навантаження
```

```
enum class LoadCondition {
    Hold, Load
}
```

Theme.kt

```
package com.hromovych.dynamicdatacollection.ui.theme
import android.app.Activity
import android.os.Build
import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.darkColorScheme
import androidx.compose.material3.dynamicDarkColorScheme
import androidx.compose.material3.dynamicLightColorScheme
import androidx.compose.material3.lightColorScheme
import androidx.compose.runtime.Composable
import androidx.compose.runtime.SideEffect
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.graphics.toArgb
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.platform.LocalView
import androidx.core.view.WindowCompat
private val DarkColorScheme = darkColorScheme(
    primary = Purple80,
    secondary = PurpleGrey80,
    tertiary = Pink80
)
private val LightColorScheme = lightColorScheme(
    primary = Purple40,
    secondary = PurpleGrey40,
    tertiary = Pink40,
    background = Color(0xFFFFFBFE),
    surface = Color(0xFFFFFBFE),
)
@Composable
fun DynamicDataCollectionTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    // Dynamic color is available on Android 12+
    dynamicColor: Boolean = true,
    content: @Composable () -> Unit
) {
    val colorScheme = when {
        dynamicColor && Build.VERSION.SDK_INT >= Build.VERSION_CODES.S -> {
            val context = LocalContext.current
            if (darkTheme) dynamicDarkColorScheme(context) else
dynamicLightColorScheme(context)
        }
        darkTheme -> DarkColorScheme
        else -> LightColorScheme
    }
    val view = LocalView.current
    if (!view.isInEditMode) {
```

```
        SideEffect {
            val window = (view.context as Activity).window
            window.statusBarColor = colorScheme.primary.toArgb()
            WindowCompat.getInsetsController(window, view).isAppearanceLightStatusBars =
darkTheme
        }
    }
```