НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

ДО ЗАХИСТУ ДОПУЩЕНО
В.о. завідувача кафедри
_____Олександр КОВАЛЬ
«_____»_____2023р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій»
спеціальності 121 Інженерія програмного забезпечення
на тему: «Програмний застосунок розпізнавання маневрів транспортних засобів»

Виконав:
студент IV курсу, групи ТІ-91
Лола Назар Олегович _____

Керівник:
Доцент к.е.н.
Гусєва Ірина Ігорівна _____

Рецензент:
_____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Олександр КОВАЛЬ
«____»_____2023р.

ЗАВДАННЯ
на дипломну роботу студенту

Лолі Назару Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Програмний застосунок розпізнавання маневрів транспортних засобів»
керівник роботи: Гусєва Ірина Ігорівна к.е.н.
затверджені наказом по університету від "29" травня 2023 року №2039
2. Строк подання студентом роботи 12.06.2023 р.
3. Вихідні дані до роботи: програмний застосунок розпізнавання маневрів транспортних засобів за допомогою сенсорів мобільного пристрою, розроблений на мові програмування Java з використанням системи збірки Gradle, сформований датасет у фораті JSON
4. Зміст (дипломної роботи) пояснювальної записки (перелік завдань, які потрібно розробити) Розробка програмного застосунку розпізнавання маневрів транспортнихзасобів
5. Перелік ілюстративного матеріалу блок-схеми алгоритмів, схеми бази даних, інтерфейс розробленої системи,

6. Дата видачі завдання «_30_» _вересня_ 2022р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Строки виконання етапів роботи	Примітка
1	Отримання завдання	30.09.2022	виконано
2	Дослідження предметної області	02.10.2022 – 16.04.2023	виконано
3	Постановка вимог до проєктування системи	17.04.2023 – 20.05.2023	виконано
4	Дослідження існуючих рішень	21.04.2023-25.04.2023	виконано
5	Розробка програмного продукту	26.04.2023 – 11.05.2023	виконано
6	Тестування	12.05.2023 – 21.05.2023	виконано
7	Захист програмного продукту	15.05.2023 – 19.05.2023	виконано
8	Оформлення дипломної роботи	22.05.2023 – 04.06.2023	виконано
9	Передзахист	5.06.2023 – 11.06.2023	виконано
10	Захист	19.06.2023 – 23.06.2023	виконано

Студент _____ Лола Назар

Керівник роботи _____ Ірина Гусєва

РЕФЕРАТ

Структура та обсяг дипломної роботи. Робота містить 49 сторінок, 13 рисунків, 2 додатки та 15 посилань.

Метою роботи є розробка програмного застосунку для розпізнавання маневрів транспортного засобу в рамках поїздки та надання інформації про використання палива.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1) проаналізувати існуючі алгоритми розпізнавання маневрів транспортних засобів. Сформувати вимоги для подальшої реалізації програмного застосунку;

2) визначити необхідні дані для алгоритму розпізнавання маневрів;

3) дослідити наявні технології та інструменти розробки мобільних застосунків. Обрати засоби розробки;

4) розробити програмний застосунок для розпізнавання маневрів транспортних засобів з подальшим виведенням їх списку та аналітичних даних.

Практичним значенням є створений програмний застосунок розпізнавання маневрів транспортних засобів, що надає аналітичну інформацію щодо виконаних маневрів та може допомогти водієві підвищити енергоефективність керування транспортним засобом.

Апробація результатів дипломної роботи

Основні положення роботи доповідалися та обговорювалися на конференції:

1. Сучасні проблеми наукового забезпечення енергетики. У 2-х т. :Матеріали XX Міжнар. наук.-практ. конф. молод. вчених і студ. (присвячена 125-річчю КПІ ім. Ігоря Сікорського та 90-річчю НН ІАТЕ (ТЕФ)), м. Київ,25–28 квіт.2023 р. – Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2023. – Т. 2. – 269 с.

Ключові слова: програмний застосунок, розпізнавання маневрів.

ABSTRACT

Structure and scope of the thesis. The work contains 49 pages, 13 figures, 2 appendices and 15 references.

The purpose of the work is to develop a software application for recognizing vehicle maneuvers during a trip and providing information on fuel consumption.

To achieve the goal, the following tasks must be solved:

1) analyze the existing algorithms for recognizing vehicle maneuvers. Form requirements for further implementation of the software application;

2) determine the necessary data for the maneuver recognition algorithm;

3) explore the available technologies and tools for developing mobile applications. Choose development tools;

4) develop a software application for recognizing maneuvers of vehicles with subsequent output of their list and analytical data.

The practical value is the created software application for recognition of vehicle maneuvers, which provides analytical information about the performed maneuvers and can help the driver to improve the energy efficiency of driving the vehicle.

Approbation of the results of the thesis

The main provisions of the work were reported and discussed at the conference:

1. Modern problems of scientific energy supply. In 2 vols.: Materials of XX International. science and practice conf. young scientists and students. (dedicated to the 125th anniversary of Igor Sikorskyi KPI and the 90th anniversary of NN IATE (TEF)), Kyiv, April 25–28, 2023 – Kyiv: KPI named after Igor Sikorskyi, Polytechnic Publishing House, 2023. - Vol. 2. - 269 p.

Keywords: software application, recognition of maneuvers.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

GPS – Global Positioning System. Глобальна система позиціонування, що складається з супутників, приймачів і земних станцій, яка дозволяє визначити точне місцезнаходження об'єкта на землі.

Акселерометр, пристрій, який використовується для вимірювання прискорення об'єкта в трьох основних напрямках.

Гіроскоп, пристрій, призначений для вимірювання або засвідчення кутової швидкості обертання тіла навколо осі. В основі роботи гіроскопа лежить принцип збереження моменту імпульсу, згідно з яким тіло, яке обертається, зберігає свою кутову швидкість, якщо на нього не діють зовнішні моменти сили.

Клас, тип даних, який визначає об'єкти з певними характеристиками та здатностями. Об'єкти, створені з одного класу, мають спільні атрибути та методи, але можуть мати різні значення атрибутів.

БД – база даних. Організованою колекція даних, яка зберігається в комп'ютерній системі. Вона служить для ефективного зберігання, організації, керування та доступу до великого обсягу даних.

ОС – операційна система. Програмне забезпечення, яке керує апаратними та програмними ресурсами комп'ютерної системи та забезпечує взаємодію між користувачем і комп'ютером.

Бібліотека, збірка попередньо написаних кодових ресурсів, які розробники можуть використовувати для виконання конкретних завдань. Бібліотеки містять підпрограми, функції, класи, методи та інші компоненти, які можуть бути використані для розширення функціональності програм.

IDE – Integrated Development Environment. Програмний інструмент, який надає розробникам усе необхідне для написання, відлагодження та тестування програмного коду в одному інтерфейсі.

ВСТУП

Одним із перспективних напрямків розвитку цифрових технологій є їх впровадження у транспортний сектор. Завдяки розробці різноманітних мобільних додатків, забезпечуються зручність і покращення якості обслуговування водіїв та пасажирів.

Враховуючи енергетичні кризи та потребу у підвищенні ефективності використання енергії, особливо в транспортному секторі, дослідники активно займаються вирішенням проблеми оптимізації споживання палива або електроенергії під час руху. Одним з підходів до вирішення цієї проблеми є аналіз послідовності маневрів, виконаних транспортним засобом під час подорожі.

Кожен маневр, виконаний транспортним засобом, споживає певну кількість палива або заряду батареї. З цього можна зробити висновок, що рух від точки А до точки Б може мати різні послідовності маневрів, які вимагають різної кількості енергії. Оптимальним варіантом буде той, в якому послідовність маневрів мінімізує споживання енергії.

З метою розробки програмного застосунку для розпізнавання маневрів транспортних засобів і аналізу витрат палива під час виконання кожного маневру, використовуються вихідні дані роботи. Це дає можливість підвищити економічну ефективність використання енергії під час поїздок, що в свою чергу сприятиме зниженню витрат на паливо та зменшенню негативного впливу на навколишнє середовище.

Результати даної дипломної роботи можуть бути використані в подальшому дослідженні і розвитку систем оптимізації енергоефективності транспорту, впровадженні автоматизованих систем керування транспортними засобами та поліпшенні загальної безпеки на дорогах.

Додатковими перевагами використання програмного застосунку для розпізнавання маневрів транспортних засобів і аналізу витрат палива є можливість збору даних про стиль водіння водіїв. Ця інформація може бути використана для

навчання систем штучного інтелекту та машинного навчання з метою поліпшення алгоритмів прогнозування та оптимізації споживання енергії.

Дані, отримані з програмного застосунку, також можуть бути використані для аналізу та оцінки технічного стану транспортних засобів. Наприклад, на основі інформації про споживання палива або електроенергії можна виявити можливі несправності або неправильну роботу окремих компонентів транспортного засобу. Це дозволяє вчасно виявляти проблеми та здійснювати технічне обслуговування, що сприяє підвищенню безпеки і тривалості експлуатації автотранспорту.

Узагальнюючи, впровадження програмного застосунку для розпізнавання маневрів транспортних засобів та аналізу витрат палива має потенціал для значного покращення енергоефективності водіння, зниження витрат на паливо та покращення загальної екологічної стійкості транспортного сектору. Такі розвинені цифрові рішення можуть сприяти досягненню більш сталого та ефективного транспортного майбутнього.

1 ПОСТАНОВКА ЗАДАЧІ РОЗПІЗНАВАННЯ МАНЕВРІВ ТРАНСПОРТНИХ ЗАСОБІВ

Метою даної роботи є розробка програмного застосунку для розпізнавання маневрів транспортного засобу в рамках поїздки та надання інформації про використання палива.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1) проаналізувати існуючі алгоритми розпізнавання маневрів транспортних засобів. Сформувати вимоги для подальшої реалізації програмного застосунку;

2) визначити необхідні дані для алгоритму розпізнавання маневрів;

3) дослідити наявні технології та інструменти розробки мобільних застосунків. Обрати засоби розробки;

4) розробити архітектуру програмного застосунку;

5) розробити алгоритм розпізнавання маневрів транспортних засобів;

6) розробити програмний застосунок для розпізнавання маневрів транспортних засобів з подальшим виведенням їх списку та аналітичних даних.

Вхідні дані: показники кількості палива/заряду батареї з OBD порту, дані акселерометра та гіроскопу мобільного телефону, а також дані про швидкість переміщення з GPS.

Вихідні дані: список виконаних маневрів за поїздку у хронологічній послідовності разом з інформацією про використання палива та середню швидкість.

Користувачами розробленої системи є водії транспортних засобів.

Висновки до розділу 1

В цьому розділі сформульована задача розпізнавання маневрів транспортних засобів, визначені основні сенсори та маневри, які розпізнаватимуться. Визначені взідні та вихідні дані. Визначені користувачі системи. Визначена мета роботи.

2 АНАЛІЗ АНАЛОГІЧНИХ СИСТЕМ РОЗПІЗНАВАННЯ МАНЕВРІВ ТРАНСПОРТНИХ ЗАСОБІВ

2.1 Виявлення маневрів автомобіля в реальному часі на основі датчиків смартфонів

Для розпізнавання маневрів транспортних засобів в реальному часі використовуються датчиків смартфона. По-перше, метод зміщення системи координат використовується для визначення місцезнаходження смартфона. По-друге, як вхідні дані використовуються відфільтровані дані гіроскопа та акселерометра. По-третє, використовується техніка надмірної вибірки синтетичної меншості задля передискретизації, щоб вирішити проблему незбалансованих даних. Модель довгострокова нейронна мережа короткочасної пам'яті створена для виявлення маневрів автомобіля. Модель налаштована на основі реальних даних водіння, зібраних під час інтенсивних випробувань у різних місцях. Базуючись на експериментальних результатах, можна зробити наступні висновки: По-перше, замість використання різних датчиків і вилучення додаткових статистичних ознак, у цій статті використовуються дані трьох відфільтрованих датчиків, що зменшує обчислювальну складність і заощаджує споживання батареї. По-друге, запропонована довгострокова нейронна мережа короткочасної пам'яті ідентифікує декілька маневрів автомобіля із середньою точністю 0,97, запам’ятовуванням 0,98 і оцінкою F1 0,98, забезпечуючи найсучаснішу продуктивність порівняно з іншими моделями. По-третє, модель забезпечує стійку продуктивність на основі даних про перехрестя, ділянки доріг і місцеві дороги в різних місцях. По-четверте, модель можна легко перенести на різні контролери або місця.

Легко маршрутизовані та відфільтровані дані датчиків є бажаними деталями для розгортання моделі. Однак, даний підхід має певні обмеження. По-перше, продуктивність довгострокової нейронної мережі короткочасної пам'яті можна покращити, додавши більше шарів або спробувавши більше комбінацій різних

параметрів. По-друге, різні маневри транспортного засобу можуть вказувати на стан безпеки водія, і майбутні дослідження можуть ввести різницю між нормальним, агресивним і небезпечним водінням. По-третє, модель була розроблена для відносно обмеженого набору даних, що може вплинути на узагальнення моделі при застосуванні до інших об’єктів. З розвитком технології зв’язаних транспортних засобів і мобільного датчика в майбутньому з'явиться велика кількість реальних даних про водіння, і модель може бути вдосконалена на основі цих нових даних [1].

Пропонуються два алгоритми для виявлення подій за кермом, використовуючи дані, зібрані зі смартфонових датчиків, таких як GPS, акселерометр і магнітометр. Особливо важливою є здатність цих алгоритмів класифікувати ці події як агресивні або неагресивні.

Перший алгоритм використовує пороговий підхід для виявлення та класифікації подій водіння на основі GPS-даних. З іншого боку, другий алгоритм використовує метод зіставлення шаблонів, аналізуючи часові ряди з датчиків акселерометра та магнітометра. Первинні експериментальні результати свідчать про те, що алгоритм зіставлення шаблонів перевершує правило-базовий алгоритм для класифікації подій як у поперечних, так і у поздовжніх рухах.

У даному дослідженні розглядаються три датчики на смартфоні. По-перше, 3-осьовий акселерометр вимірює силу прискорення, що виникає внаслідок руху телефону або гравітації. Кожна з трьох осей відповідає бічному, поздовжньому та вертикальному прискоренню. У цьому дослідженні нас цікавлять лише рухи вздовж бічної та поздовжньої осей, які відповідають руху вбік і руху вперед-назад відповідно.

У реальних ситуаціях поперечне прискорення або рух вбік відображають такі події водіння, як повороти вліво і вправо або зміна смуги руху, тоді як поздовжнє прискорення відповідає гальмуванню та прискоренню автомобіля. Другий датчик - магнітометр - вимірює силу магнітного поля, яка дозволяє нам визначити

напрямок, в якому спрямований смартфон відносно магнітного північного полюса. Цей датчик зазвичай використовується в компасах.

Необроблені дані з магнітометра використовуються як ознака для виявлення рухових подій в бічній площині. Нарешті, GPS-приймач надає дані про місцезнаходження та швидкість транспортного засобу, до якого підключений смартфон. Загалом дані з акселерометра та магнітометра збираються з частотою 5 Гц, що означає, що кожен зразок записується кожні 200 мс. Дані з GPS-приймача дискретизуються з частотою 1 Гц [4].

2.2 Маневри транспортних засобів

Маневри транспортних засобів - це зміни в русі або руховій поведінці транспортного засобу. Вони виконуються водіями або операторами транспортних засобів залежно від їх потреб і умов дорожнього руху. Маневри можуть бути контрольованими та безпечними, якщо вони виконуються з дотриманням правил дорожнього руху та урахуванням інших учасників дорожнього руху.

Серед можливих маневрів транспортного засобу на дорозі було обрано чотири, а саме:

- прискорення;
- сповільнення;
- поворот наліво;
- поворот направо.

Розпізнавання маневрів здійснюється за допомогою показників акселерометра та гіроскопа.

Акселерометр вимірює довжину вектора прискорення по трьом осям. На рисунку 2.1 зображена орієнтація осей OX, OY та OZ акселерометра у просторі відповідно мобільному пристрою, в який його вбудовано.

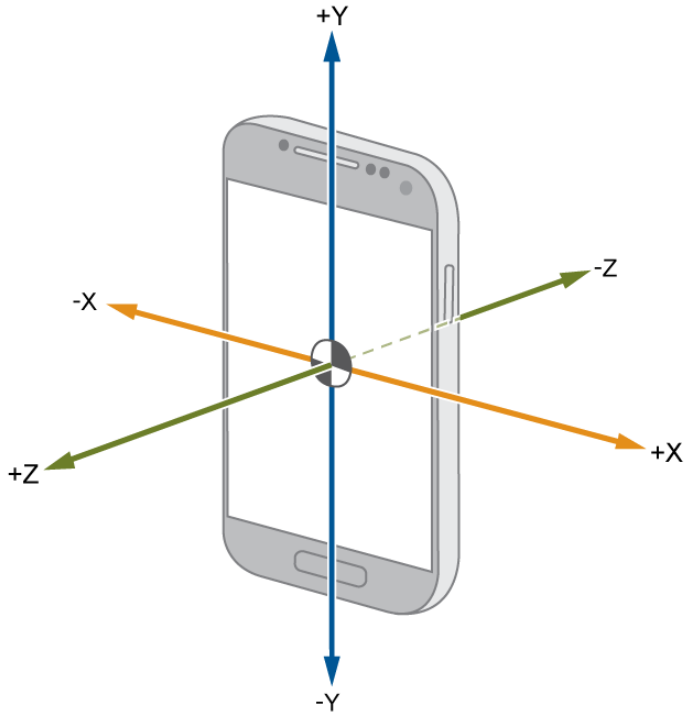

Рисунок 2.1 — Відносне розташування осей акселерометра у просторі

Під час прискорення чи сповільнення транспортного засобу, значення довжини вектора прискорення всіх осей, що вираховується за формулою довжини вектора по трьом координатам, зростає. Отже, якщо довжина вищезазначеного вектора перевищує певну межу, визначену внаслідок аналізу показників сенсора під час виконання маневрів, можна зробити висновок, що транспоортний засіб виконує маневр прискорення або сповільнення. Розрізнити ці два маневра можливо перевіривши значення прискорення узяте з осі, що розташована уздовж автомобіля. В залежності від розташування пристрою відносно напрямку руху автомобіля, додатнє значення буде вказувати на один маневр, а від'ємне — на інший.

Для розпізнавання маневрів повороту використаємо гіроскоп. Робота даного сенсору полягає у вимірювання швидкості повороту пристрою у просторі навколо трьох осей. На рисунку 2.2 зображене відносне розташування осей гіроскопа у просторі.

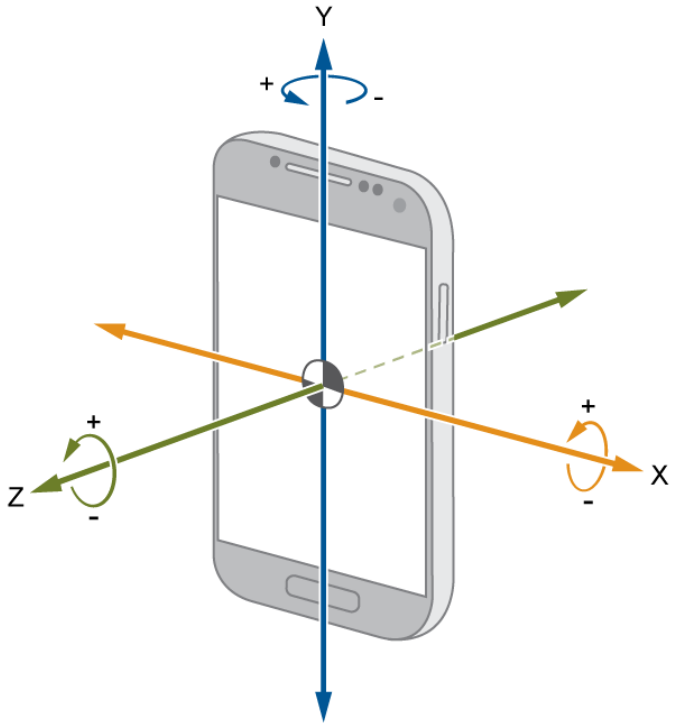

Рисунок 2.2 — Відносне розташування осей гіроскопу у просторі

Під час виконання маневру повороту, пристрій виконує обертання у просторі, яке можливо визначити внаслідок знаходження довжини вектора обертання за трьома осями та його порівняння з граничним занченням, що було сформовано внаслідок дослідження даних вищезазначеного сенсора під час вик1онання маневрів повороту транспортного засобу. Для того, щоб відрізнити поворот наліво від повороту направо, слід перевірити значення вектора обертання за віссю, що розташована вертикально відносно транспортного засобу. У залежності від розташування пристрою з гіроскопом відносно транспортного засобу додатнє значення та від'ємне значення будуть вказувати на перший або другий маневр повороту.

Тип і якість доріг, а також організація дорожнього руху, мають значний вплив на маневри, які здійснюють водії. Можна розглянути чотири типові ділянки доріг: міська дорога, однопроїзна дорога з двома смугами руху в незабудованій місцевості, швидкісна двопроїзна дорога та шосе. Отримані результати демонструють значні різниці у стилях водіння на цих дорогах. Аналіз процесів

розгону, гальмування та керування підтверджує висновок, що тип дороги сильно впливає на спосіб виконання цих маневрів та їх частоту. У дослідженні були сформульовані умовні критерії для визначення цих маневрів на основі зареєстрованих значень прискорень. Ці критерії були аналізовані та перевірені експериментально та за допомогою комп'ютерного моделювання.

Аналіз структури маневрів, які виконує водій, заснований на двох параметрах: екстремальному значенні прискорення під час виконання конкретного маневру (максимальне значення для позитивних прискорень і мінімальне значення для негативних прискорень) і частоті виконання маневрів під час проходження заданого маршруту. Для обох цих маневрів найвищі (за абсолютним значенням) прискорення спостерігаються під час міської їзди — до 3 м/с² при розгоні автомобіля і до 4–5 м/с² при гальмуванні. На подальших ділянках доріг, де швидкість збільшується, екстремальні значення прискорення для цих маневрів зменшуються, а на шосе вони не перевищують 1 м/с². Частота виконання маневрів прискорення та гальмування також сильно залежить від типу дороги. В середньому, у місті маневри прискорення та гальмування відбувалися кожні 13 секунд, поза містом — кожні 36 секунд, на швидкісній трасі — кожні 7 хвилин і на шосе — кожні 14 хвилин. Це означає, що маневри прискорення та гальмування в умовах міського руху відбувалися приблизно в 65 разів частіше, ніж на шосе.

Для правого та лівого повороту найвищі значення прискорення спостерігаються в міському русі на заміських дорогах - до 3-4 м/с². На швидкісній дорозі максимальні значення для поворотних маневрів становлять до 2 м/с², тоді як на шосе вони складають близько 1 м/с². На відміну від маневрів прискорення та гальмування, для яких існує сильна залежність від частоти їх виконання від типу дороги, частота маневрів повороту для всіх чотирьох типів доріг дуже схожа і навіть майже однакова для трьох типів доріг: міської дороги - кожні 26 секунд, приміської дороги та шосе - кожні 25 секунд відповідно. Є відмінність лише в частоті маневрів повороту на швидкісній дорозі, оскільки вони відбуваються кожні 11 секунд. Це більше ніж вдвічі частіше, ніж на трьох інших маршрутах, але слід

зауважити, що у випадку маневрів прискорення та гальмування варіація є значно більшою. Проте варто враховувати, що з урахуванням різної швидкості маршрутів, маневри здійснюються на різних інтервалах доріг [2].

2.3 Профілювання поведінки водія

Проведена кількісна оцінка продуктивності чотирьох алгоритмів машинного навчання з різними конфігураціями для виявлення семи типів подій під час водіння. Для цього викоистовувалися дані, зібрані з чотирьох датчиків смартфона з операційною системою Android (акселерометр, лінійне прискорення, магнітометр і гіроскоп). У реальному експерименті з двома водіями зібрані 69 зразків цих подій, що записувалися час початку та закінчення кожної події. Також порівняна продуктивність застосування різних розмірів ковзного вікна.

Проведені 15 виконань з різними випадковими початковими числами для оцінки 3865 вузлів. В результаті виявлено п'ять найефективніших вузлів для кожного типу події. Загалом, результати показують, що:

1) більші розміри вікна працюють краще;

2) гіроскоп і акселерометр є найкращими датчиками для виявлення подій під час керування транспортним засобом;

3) використання всіх осей датчика зазвичай працює краще, ніж використання лише однієї осі, за винятком агресивних поворотів ліворуч;

4) на сьогоднішній день алгоритм випадкового лісу є найбільш продуктивним алгоритмом машинного навчання, за ним йде багатошаровий перцептрон [3].

Висновки до розділу 2

Роглянуто іноваційні методи вирішення задачі розпізнавання маневрів транспортних засобів, такі як використання машинного навчання та алгоритми, засновані на правилах. Проведені порівняння ефективності різноманітних підходів.

3 ЗАСОБИ РОЗРОБКИ СИСТЕМИ

3.1 Мова програмування Java

Java є високорівневою об'єктно-орієнтованою мовою програмування, яка має мінімальні залежності від конкретної реалізації. Ця мова загального призначення розроблена з метою забезпечити можливість написання програм один раз і запуску їх на будь-якій платформі (WORA - Write Once, Run Anywhere). Код Java компілюється до байт-коду, який може виконуватись на будь-якій віртуальній машині Java (JVM), незалежно від архітектури комп'ютера.

Синтаксис Java подібний до C і C++, але він має менше можливостей низькорівневого програмування. Java надає динамічні можливості в середовищі виконання, такі як відображення та модифікація коду, які зазвичай відсутні у традиційних скомпільованих мовах.

Java є однією з найпопулярніших мов програмування і знаходить широке застосування у розробці клієнт-серверних веб-додатків. За даними GitHub на 2019 рік, Java має велику кількість розробників (9 мільйонів). Перша версія Java була розроблена Джеймсом Гослінгом у Sun Microsystems та була випущена у травні 1995 року.

Початково Java поставлялась разом із власними ліцензіями компанії Sun. Проте, з часом більшість технологій Java було переліцензовано за ліцензією GPL-2.0 в рамках Java Community Process. Сьогодні Oracle пропонує власну віртуальну машину Java HotSpot, але еталонною реалізацією є OpenJDK JVM, яка є безкоштовним програмним забезпеченням з відкритим вихідним кодом і використовується більшістю розробників. Також OpenJDK є вбудованою в JVM більшості дистрибутивів Linux [5].

3.2 Середовище розробки Android Studio

Android Studio — це інтегроване середовище розробки (IDE), призначене для розробки мобільних застосунків на платформі Android. Його розробляє компанія Google і він є основним інструментом для розробки додатків для платформи Android.

Основні переваги використання Android Studio для розробки мобільних додатків на платформі Android включають пункти описані нижче.

1. Інтегроване середовище розробки: Android Studio забезпечує розробникам повний набір інструментів, необхідних для розробки мобільних додатків на платформі Android. Це включає редактор коду, інструменти для налагодження та профілювання додатків, інструменти для дизайну інтерфейсу користувача та інші.

2. Підтримка різних мов програмування: Android Studio підтримує різні мови програмування, включаючи Java, Kotlin та C++. Це дає розробникам можливість вибрати мову програмування.

3. Автоматична перевірка коду: Android Studio має вбудовані інструменти для автоматичної перевірки коду, що дозволяє виявляти можливі помилки та пропонувати відповідні виправлення.

4. Швидкість розробки: Android Studio забезпечує швидку розробку застосунків на платформі Android завдяки вбудованим шаблонам та бібліотекам. Це дозволяє розробникам зосередитися на створенні корисної функціональності, а не на деталях роботи зі структурою додатку.

5. Підтримка спільноти розробників: Android Studio є дуже популярним середовищем розробки для мобільних застосунків на платформі Android, тому на ньому розробляється багато бібліотек та інших розширень, які дозволяють розробникам виконувати свої завдання більш ефективно та швидко.

Android Studio підтримує ті ж самі мови програмування, що й IntelliJ такі як Java, C++ і інші, включаючи Go з використанням розширень. Починаючи з версії

Android Studio 3.0, він також підтримує Kotlin, а також "всі функції мови Java 7 і певну підмножину функцій мови Java 8, які залежать від версії платформи". Деякі функції Java 9 також були портовані в зовнішні проекти. Хоча IntelliJ стверджує, що Android Studio підтримує всі випущені версії Java і Java 12, рівень підтримки версій Java до Java 12 у Android Studio не зовсім зрозумілий (документація згадує про часткову підтримку Java 8). Проте деякі нові функції мови до Java 12 можуть бути використані в Android.

Після компіляції програми з використанням Android Studio її можна опублікувати в Google Play Store. При цьому програма повинна відповідати політиці Google Play Store щодо вмісту, що стосується розробників [6].

3.3 Мова розмітки XML

XML (Extensible Markup Language) є мовою розмітки, призначеною для обміну та збереження даних між різними системами та платформами. Вона дозволяє створювати структуровані дані зі семантичним змістом, спрощуючи опис інформації в форматі, легкому для розуміння та обробки.

XML-документи складаються з тегів, які вказують на початок та кінець елемента, а також змісту елемента всередині. Крім того, використовуються атрибути, що надають додаткові властивості елементам. XML-документи широко використовуються для збереження конфігурацій, опису інтерфейсів, передачі даних між серверами та клієнтами, а також для створення довідників та іншої інформації.

У розробці Android-додатків XML використовується для опису різних елементів інтерфейсу користувача, таких як макети (layouts), стилі (styles), ресурси (resources) та інші. Використання XML для створення інтерфейсів користувача дозволяє легко модифікувати та розширювати їх, не втручаючись в код додатку.

Узагальнюючи, XML є потужним та гнучким інструментом, який допомагає розробникам створювати структуровані та організовані документи, що легко

обробляються програмним забезпеченням. У контексті розробки Android-додатків, XML відіграє важливу роль у створенні інтерфейсів користувача та управлінні ресурсами додатків [7].

3.4 База даних SQLite

SQLite — це легка вбудовувана реляційна база даних, яка є частиною більшості операційних систем, включаючи Android. Вона працює на основі файлів і зберігає дані в реляційному вигляді з допомогою SQL-запитів.

SQLite дуже ефективна у використанні на мобільних пристроях, оскільки його розмір невеликий, а функціональність повноцінна. SQLite підтримує багато типів даних, таких як цілі числа, рядки, дати, булеві значення та інші. Також вона має можливості для створення індексів, транзакцій, обмежень і багато іншого.

В Android SQLite використовується для збереження даних у різних додатках, включаючи контакти, повідомлення, історію перегляду та інші. Для взаємодії з SQLite використовуються спеціальні класи, які надають методи для створення, збереження, видалення та оновлення таблиць та записів [8].

3.5 Операційна система Android

Android, розроблений компанією Google, є мобільною операційною системою, яка призначена для різних пристроїв, таких як смартфони, планшети, телевізори та інші. Ця платформа відкрита, що дозволяє розробникам створювати різноманітні додатки, використовуючи різні мови програмування.

Операційна система Android має багато функцій, серед яких багатозадачність, мультимедійний фреймворк та підтримку багатьох мов. Інтерфейс Android може бути налаштований користувачем, а також підтримує різні типи підключення до мережі, такі як Wi-Fi, Bluetooth та мобільний інтернет.

Для розробки додатків для Android зазвичай використовуються мови програмування Java або Kotlin. XML-файли використовуються для створення інтерфейсу користувача, а SQLite використовується для збереження даних. Для поліпшення продуктивності та швидкодії додатків на платформі Android розробники використовують різні бібліотеки та фреймворки, такі як Android SDK, Retrofit, OkHttp та інші.

Android - це мобільна операційна система, заснована на модифікованій версії ядра Linux та інше програмне забезпечення, що забезпечується відкритим вихідним кодом. Вона була розроблена переважно для сенсорних мобільних пристроїв, таких як смартфони та планшети. Розробкою Android займається консорціум розробників, відомий як Open Handset Alliance, але основну роботу над ним виконує Google. Операційна система була представлена в листопаді 2007 року, а перший комерційний пристрій, що працює на Android - HTC Dream, був випущений у вересні 2008 року.

Android Open Source Project (AOSP) - це безкоштовне програмне забезпечення з відкритим вихідним кодом, яке є основою операційної системи. Він переважно ліцензується за ліцензією Apache. Інші пристрої працюють на фірмовій версії Android, розробленій Google, яка постачається з поточним закритим програмним забезпеченням, включаючи Google Mobile Services (GMS). GMS включає такі програми, як Google Chrome, Google Play Store і пов'язані сервіси. Незважаючи на те, що AOSP є безкоштовним, торгова марка "Android" належить Google, і вона обмежує використання цієї назви "несертифікованими" пристроями поза своєю екосистемою.

Більшість смартфонів на базі Android Open Source Project працює в екосистемі Google, повідомляє просто як Android. Деякі виробники, такі як Samsung і HTC, мають власні інтерфейси користувачів і програми, такі як TouchWiz і One UI. Існують також конкурентні екосистеми, такі як Fire OS від Amazon, ColorOS від OPPO, OriginOS від Vivo та MagicUI від Honor, а також спеціальні ПЗУ, такі як LineageOS.

Вихідний код Android використовується для розробки різних варіацій операційної системи для інших пристроїв, включаючи телевізори (Android TV) та носії пристрої (Wear OS). Є також платформи, які вибирають формат APK для розповсюдження програмного забезпечення для Android, такі як Google Play Store, Amazon Appstore, Samsung Galaxy Store, Huawei AppGallery та інші.

Android є найпопулярнішою мобільною операційною системою у світі з 2011 року на смартфонах та з 2013 року на планшетах. У травні 2021 року вона мала понад три мільярди активних користувачів, що містяться. Google Play Store містить понад 3 мільйони програм. Останньою версією Android є Android 13, яка була випущена у вересні 2022 року, а також нещодавно була випущена версія Android 12.1/12L, яка має покращення, спрямоване на роботу на складних пристроях з великими екранами, такими як телефони, планшети та Chromebook [9].

3.6 Формат даних JSON

JSON (JavaScript Object Notation) - це формат обміну даними, що використовується для передачі структурованої інформації між різними програмами. Він є легким, простим у використанні і широко використовується в багатьох веб-додатках як стандартний формат.

JSON-об'єкти можуть містити інші JSON-об'єкти і вкладені ключ-значення. Це дозволяє гнучко будувати структуру даних та використовувати складні об'єкти для збереження інформації.

Одним з найпоширеніших використань JSON є обмін даними між клієнтом та сервером у веб-додатках. JSON-дані можуть передаватися через HTTP-запити та відповіді, що робить їх зручними для розробки програмного забезпечення, що працює з мережевими даними.

Основні типи даних JSON описані нижче.

1. Число. Може бути десятковим числом зі знаком, яке може містити дробову частину та використовувати експоненціальний запис E. Проте воно не

може представляти нечісельні значення, такі як NaN. Формат чисел не робить розрізнення між цілими числами та числами з плаваючою комою. У деяких випадках JavaScript використовує формат підвищеної точності з рухомою версією IEEE-754 для представлення числових значень (також підтримується BigInt). Інші мови, які реалізують JSON, можуть мати власний спосіб кодування чисел.

2. Рядок. Послідовність символів Unicode, яка може містити нуль або більше символів. Рядки в JSON вказуються у подвійних лапках і підтримують синтаксис екранування зворотною косою.

3. Логічне значення. Може бути або значення "true", або значення "false".

4. Масив. Впорядкований список, що складається з нуля або більше елементів. Кожен елемент може бути будь-якого типу. У JSON масиви відображаються в квадратних дужках і елементи розділяються комами.

5. Об'єкт. Набір пар ключ-значення, де ключі є рядками. JSON не накладає ніяких обмежень на рядки, що використовуються як ключі, і не вимагає, щоб рядки були унікальними або непорядкованими. Об’єкти в JSON відображаються у фігурних дужках, а кожна пара ключ-визначення розділяється комою, а двокрапка відокремлює ключ від значення.

6. Null. Спеціальне значення, що позначає порожнє значення.

Записи, які знаходяться перед або після синтаксичних елементів в JSON, та ігноруються. У JSON можна включити пробіл, горизонтальну табуляцію, переведення рядка та повернення каретки. Стандарт JSON не має синтаксису для коментарів.

У ранніх версіях JSON було обмежено, оскільки текст JSON міг би зберігатися лише з об’єкта чи масиву, які у своїй версії могли б надрукувати інші типи даних усередині них. Протест це обмеження було скасовано у пізніших версіях.

Стосовно чисел в JSON, вони агностичні для способу представлення в різних мовах програмування. Це може створювати проблеми з переносимістю, оскільки деякі реалізації можуть розглядати різні записи чисел як одне і те ж число, тоді як

інші можуть їх розрізняти. Стандарт JSON не висуває вимог щодо деталей реалізації чисел, таких як обробка переповнення, недоповнення, втрати точності або округлення. Однак очікуйте не більше двійкової точності IEEE 754 для забезпечення хорошої сумісності.

JSON не дозволяє коментувати у своєму синтаксисі. Це зроблено з метою уникнення використання коментарів для збереження директиви синтаксичного аналізу, що може порушити взаємодію. Також в JSON заборонено використання "кінцевих ком" (комі після останнього значення всередині структури даних) [10].

3.7 Система збірки Gradle

Gradle є потужною системою збирання та автоматизації розробки програмного забезпечення. Вона забезпечує керування залежностями проекту, компіляцію коду, створення пакетів, тестування, розгортання та багато інших задач, пов'язаних з розробкою програмного забезпечення.

Gradle використовує декларативний підхід до збирання проектів. Ви описуєте структуру проекту та його залежності в конфігураційному файлі (зазвичай build.gradle), а Gradle використовує ці вказівки для виконання відповідних дій. Для опису завдань використовуються мови скриптів Groovy або Kotlin, що дозволяє легко налаштовувати та розширювати процес збирання.

Основні переваги використання Gradle включають гнучкість, ефективність, масштабованість та інтеграцію з іншими інструментами розробки. Завдяки гнучкості, ви можете налаштувати процес збирання відповідно до ваших потреб, додавати власні завдання, плагіни та розширення. Ефективність Gradle досягається за рахунок системи кешування, інкрементальної збірки та оптимізованої моделі залежностей. Підтримка масштабованості дозволяє працювати з великими проектами з багатьма модулями та підпроектами, налаштовувати залежності та керувати різними конфігураціями збирання. Інтеграція Gradle з різними інструментами та сервісами, такими як Android Studio, Jenkins, SonarQube та

системи контролю версій, дозволяє зручно використовувати їх разом для розробки та розгортання проекту.

У розробці програмного забезпечення для Android, Gradle використовується як система збирання та керування залежностями. За допомогою файлів конфігурації проекту build.gradle та settings.gradle, ви можете налаштовувати версії бібліотек, додавати залежності, визначати властивості додатку та виконувати багато інших налаштувань, що допомагають вам створити ефективний та функціональний програмний застосунок для платформи Android [11].

3.8 Графічна бібліотека GraphView

Бібліотека GraphView є інструментом, що допомагає розробникам Android-додатків візуалізувати дані у формі графіків та діаграм. Вона є компактною, легкою у використанні та забезпечує розробникам швидкі й зручні можливості для створення різноманітних графічних представлень даних.

Завдяки бібліотеці GraphView розробники можуть створювати різні типи графіків, такі як лінійні, стовпчикові та кругові діаграми. Бібліотека пропонує широкі можливості налаштування вигляду графіків, включаючи кольори, шрифти, легенди, маркери точок та осі координат.

Однією з ключових особливостей GraphView є підтримка взаємодії з користувачем. Розробники можуть налаштовувати обробники подій, наприклад, обробку натискання на точки графіка або прокрутку, що дозволяє створювати інтерактивні графічні елементи для користувачів.

Бібліотека також підтримує анімацію, що дозволяє створювати переходи між різними станами графіків. Це сприяє покращенню візуального ефекту й динамічності графічних представлень даних.

Загалом, бібліотека GraphView є потужним інструментом для візуалізації даних у формі графіків та діаграм у Android-застосунках. Вона надає розробникам

зручні й гнучкі можливості для створення привабливих графічних представлень даних з широким спектром настройок і можливостей взаємодії з користувачем [12].

3.9 Архітектура Model-view-controller

Шаблон проектування програмного забезпечення, відомий як Model-view-controller (MVC), широко використовується для розробки інтерфейсів користувача. Він дозволяє розділити програмну логіку на три взаємопов'язані елементи, щоб відокремити внутрішнє представлення інформації від того, як інформація відображається та взаємодіє з користувачем.

Початково шаблон MVC був розроблений для настільних графічних інтерфейсів користувача (GUI), але з часом став дуже популярним у розробці веб-додатків. Багато популярних мов програмування мають фреймворки, що спрощують реалізацію шаблону MVC.

Модель є ключовим компонентом узагальненого візерунка програми. Ця динамічна структура даних функціонує незалежно від інтерфейсу користувача і безпосередньо керує даними, логікою та правилами програми. У деяких платформах, наприклад у Smalltalk-80, програміст повністю контролює проектування моделі [13]. У фреймворках WebObjects, Rails та Django тип моделі часто відображається як таблиця у базі даних програми [14].

Перегляд відноситься до будь-якого способу представлення інформації, такого як діаграма, графік або таблиця. Він може мати кілька варіантів, наприклад, стовпчикова діаграма для управління і табличний вигляд для бухгалтерів.

У Smalltalk-80 перегляд є лише візуальним представленням моделі і не обробляє введення користувача [13]. У WebObjects перегляд представляє собою повний елемент інтерфейсу користувача, такий як меню або кнопка, і отримує вхідні дані від користувача [14]. Проте як у Smalltalk-80, так і у WebObjects перегляд призначений для загального використання та може бути комбінований [13][14].

У Rails і Django роль перегляду виконується HTML-шаблонами, тому в їхній схемі перегляд визначає користувацький інтерфейс, що відображається у веб-браузері, а не безпосередньо представляє віджет інтерфейсу користувача [15]. Цей підхід ставить меншу акцент на малі, перевикористовувані перегляди, оскільки типовий перегляд в Rails відповідає певному дії контролера.

У переглядах Smalltalk-80 взаємодіють як з моделлю, так і з контролером [13], тоді як у WebObjects перегляд спілкується лише з контролером, який в свою чергу спілкується з моделлю [14]. У Rails і Django перегляд або шаблон використовується контролером або переглядом для підготовки відповіді для клієнта.

Приймаючи вхідні дані, контролер перетворює їх на команди для моделі або перегляду.

Кожен контролер має одне пов'язане представлення та модель, і один об'єкт моделі може бути на слуху у різних контролерів. Тільки один контролер, відомий як "активний" контролер, отримує дані користувача в певний момент часу, і глобальний об'єкт диспетчера вікон визначає активний контролер. Якщо введення користувача призводить до зміни моделі, контролер сигналізує про це, але модель відповідає за оновлення своїх переглядів [13].

У WebObjects представлення обробляють введені користувачем дані, а контролер виступає посередником між представленнями та моделями. Може бути лише один контролер на програму або один контролер на вікно. Значна частина логіки конкретної програми знаходиться в контролері [14].

У Rails запити, що надходять на сервер від клієнта, проходять через "маршрутизатор", який визначає метод конкретного контролера для обробки запиту. У цьому методі контролер взаємодіє з даними запиту та відповідними об'єктами моделі, підготовлює відповідь за допомогою перегляду. Зазвичай, кожен тип моделі має свій власний контролер, наприклад, клієнтська модель має відповідний клієнтський контролер. Однак розробники можуть створювати інші типи контролерів за своїм бажанням [15].

Висновки до розділу 3

В даному розділі описані інструменти, обрані для реалізації програмного застосунку.

Мовою програмування була обрана Java, адже вона вважається однією з оптимальних при створені мобільних застосунків. Для реалізації інтерфейсу обрано XML.

Середовищем розробки обрано Android Studio, через функції, які підвищують зручність розробки мобільних застосунків для операційної системи Android, такі як вбудований емулятор та зручний інтерфейс керування проектом.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Структура програмного застосунку

Архітектура програмного застосунку представлена у вигляді MVC, що розшифровується як Model, View, Controller. Контролер виконує обробку даних з датасету та запису і читання їх з бази даних та маршрутизацію додатку, View використовується для відображення користувацького інтерфейсу, в той час як модель виконує роль збереження даних про розпізнані маневри. Схема архітектури системи по класам представлена на рисунку 4.1.

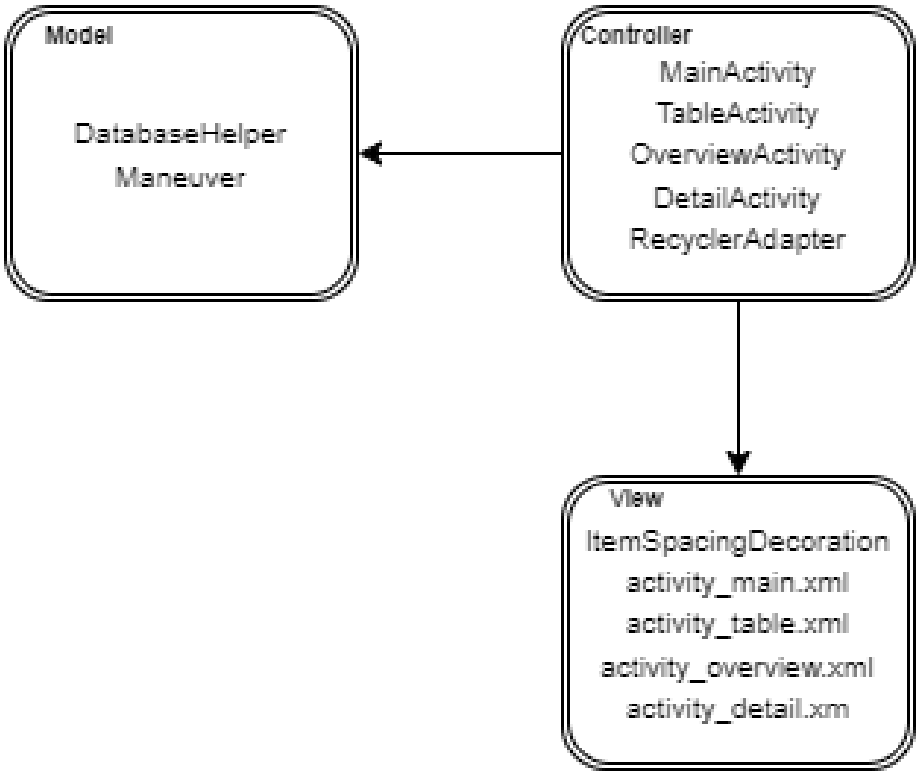

Рисунок 4.1 — Архітектура системи

База даних складається з однієї таблиці, яка містить дані про розпізнаний маневр, а саме його тип, середню швидкість під час його виконання та кількість палива, що була використана, що можна побачити на рисунку 4.2.

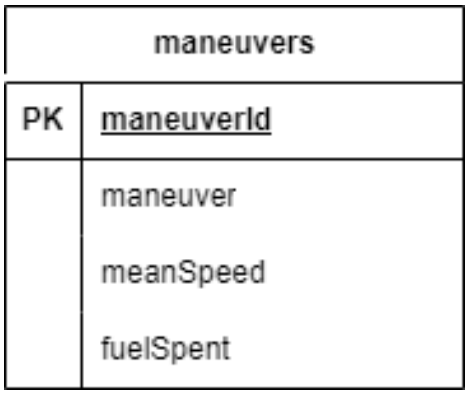

Рисунок 4.2 — Таблиця маневрів у базі даних

Саме ці дані будуть використовуватись для подальшого відображення їх користувачеві.

Мобільний застосунок написаний для операційної системи Android складається з екранів, які мають назву acticity. Кожен екран має свій клас-контролер та XML файл, у якому описана розмітка графічного інтерфейсу користувача.

Застосунок, створений у даній роботі складається з чотирьох екранів, які перераховані нижче:

1) головний екран застосунку;

2) екран з таблицею маневрів, виконаних під час поїздки;

3) екран з переглядом аналітики споживання палива та середньої швидкості одного маневру, обраного у таблиці;

4) екран з загальною інформацією про споживання палива та середню швидкість за час поїздки.

На рисунку 4.3 зображена схема екранів застосунку.

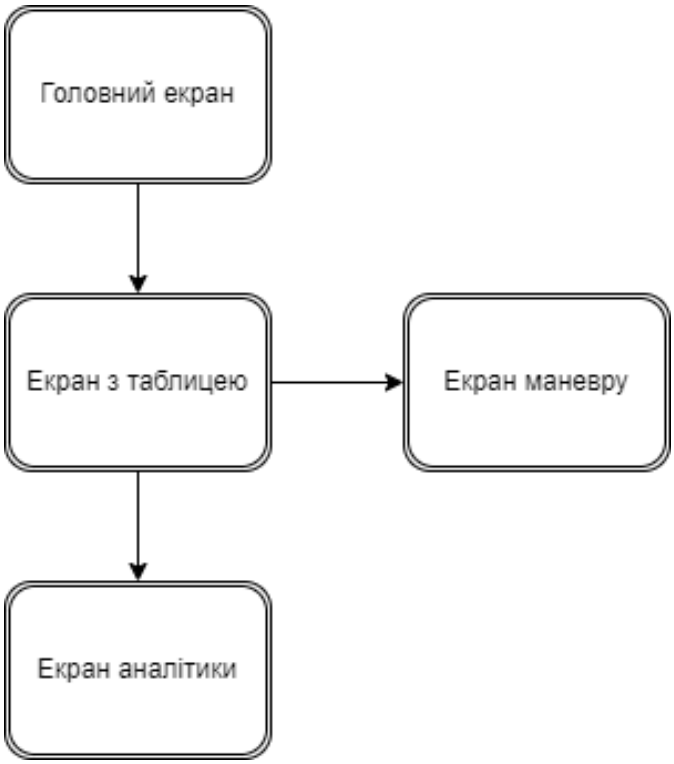

Рисунок 4.3 — Екрани застосунку

Клас-контролер основного екрану застосунку називається MainActivity, та містить реалізацію алгоритму розпізнавання маневрів та запис інформації про маневри до бази даних.

Контролер екрану з таблицею має назву TableActivity та містить у собі реалізацію списку виконаних маневрів за допомогою RecyclerView та процесу зчитування записыв з таблицы у базы даних для заповнення цього списку.

У кожного елемента списку є ідентифікатор, який дорівнює id відповідного запису про маневр у базі даних, для того, щоб при переході до екрану з аналітикою про обраний маневр, проводилося відображення даних стосовно саме цього запису.

Контролер екрану з аналітичними даними про обраний маневр має назву DetailActivity. У ньому реалізоване зчитування запису з бази даних за допомогою id обраного маневру, що передається з TableActivity та представлення даних про маневр відносно інших маневрів у поїздці у візуальному форматі за допомогою бібліотеки GraphView.

Клас-контролер екрану з загальними даними про поїздку має назву OverviewActivity, та містить у собі реалізацію представлення даних про використання палива та середню швидкість під час поїздки у вигляді графіків, та процесу зчитування значень з бази даних для їх заповнення.

4.2 Опис структури датасету

Датасет, обраний для симуляції вхідних даних з сенсорів мобільного пристрою має розширення JSON. Для зчитування даних датасету було обрано бібліотеку GSON, що дозволила ітеративно зчитувати записи з файлу у потоці, замість попереднього завантаження усього файлу до оперативної пам’яті, через що вдалося запобігти помилці out of Memory, яка свідчиться про те, що кількість памєяті, яка була виділена у оперативній пам'яті є надто великою.

Для реалізації процесу зчитування даних необхідно створити клас для роботи із записом у JSON файлі у вигляді об'єкту.

Створений клас має назву SensorData, і використовується для роботи з даними, які зчитуються з датасету.

Клас містить поля для кожної сутності у JSON записі, і має оголошення відповідних класів, таких як:

1) Acceleration, з полями для показників акселерометра для трьох осей;
2) RotationRate, з полями для показників гіроскопа для трьох осей;
3) Location, для отримання даних про поточну швидкість руху із даних про геолокацію;
4) Odb, з полем для зберігання даних про поточну кількість палива у баці.

4.3 Алгоритм обробки даних

Оскільки під час реальної поїздки через дефекти дороги або через особливості роботи елктроніки в мобільному пристрої сигнали з сенсорів у

необроблему вигляді можуть містити сторонній шум, що може призвести до некоректної роботи алгоритму визначення маневрів, перед реалізацією алгоритму необхідно було впровадити фільтрацію сигналів задля вирівнювання можливих скачків.

Для фільтрації вводу було обрано фільтр низьких частот. Робота такого фільтра полягає в тому, що він пропускає сигнали з частотою нижче певного обмеження, що дозволяє запобігти різким змінам сигналу показників сенсорів, через те, що в такому випадку їхня частота буде перевищувати обмеження.

Програмна реалізація даного фільтру у роботі застосунку наступна.

Для значення кожної з трьох осей акселерометра та гіроскопа створено буферні змінні у тілі класу головного екрану додатку, в якому і виконується аналіз вхідних значень. Також створені константні змінні ALPHA та BETA. Схема алгоритму зображена на рисунку 4.4.

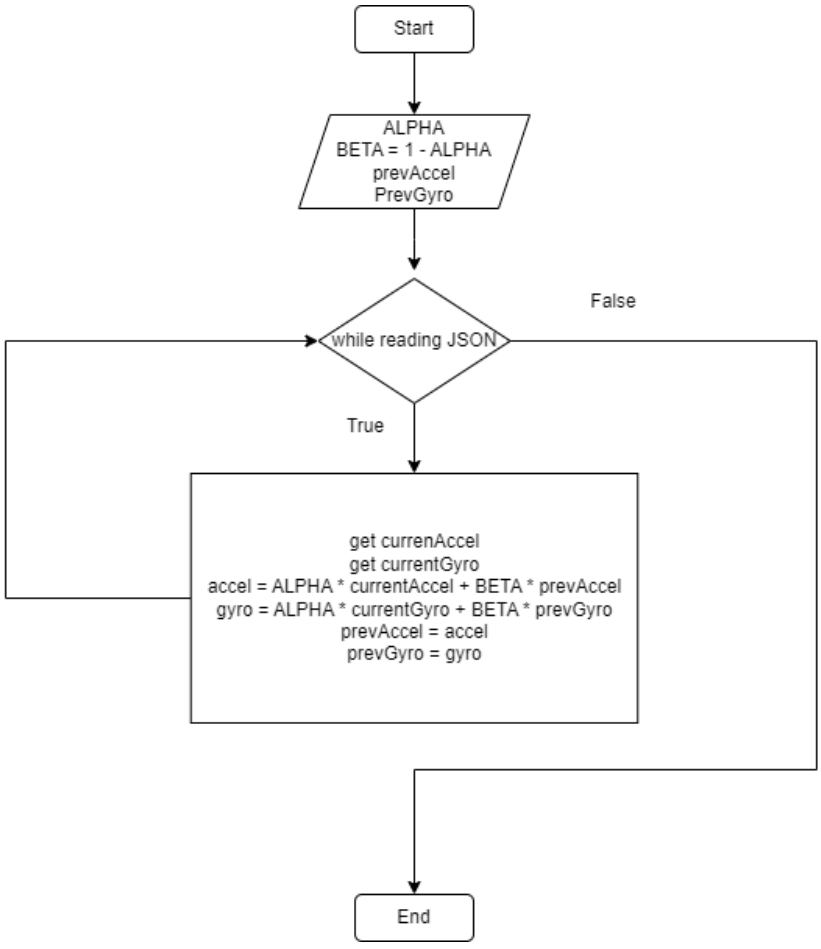

Рисунок 4.4 — Алгоритм обробки даних

Безпосередньо в тілі циклу проходження по записам з JSON файлу, перед передачею показників сенсорів до алгоритму аналізу проводиться фільтрація для показника кожної вісі за формулою 4.1.

$$x = A \times c + B \times b \quad (4.1)$$

де A — значення змінної ALPHA;

B — значення змінної BETA;

c — поточні дані узяті з сенсору;

b — дані сенсору з попередньої ітерації.

Після цього значення для кожної з осей, отримані формулою передаються до методу з реалізацією алгоритму аналізу та записуються до відповідних буферних змінних, що містять значення попередньої ітерації.

4.4 Алгоритм розпізнавання маневрів

Для визначення маневрів було вирішено використовувати алгоритм, що керується певними правилами, з якими порівнюються вхідні дані, після чого виноситься певне «судження».

Після отримання даних про показники сенсорів в даний момент з датасету та попередньої фільтрації, вони передаються до алгоритму, який обчислюючи довжину векторів прискорення та обертання зрівнює їх з граничними значеннями для визначення, чи це був маневр повороту або зміни швидкості, після чого вже остаточно визначається, що це був за маневр після ще одного зрівняння з граничними значеннями. На рисунку 4.5 представлена схема алгоритму.

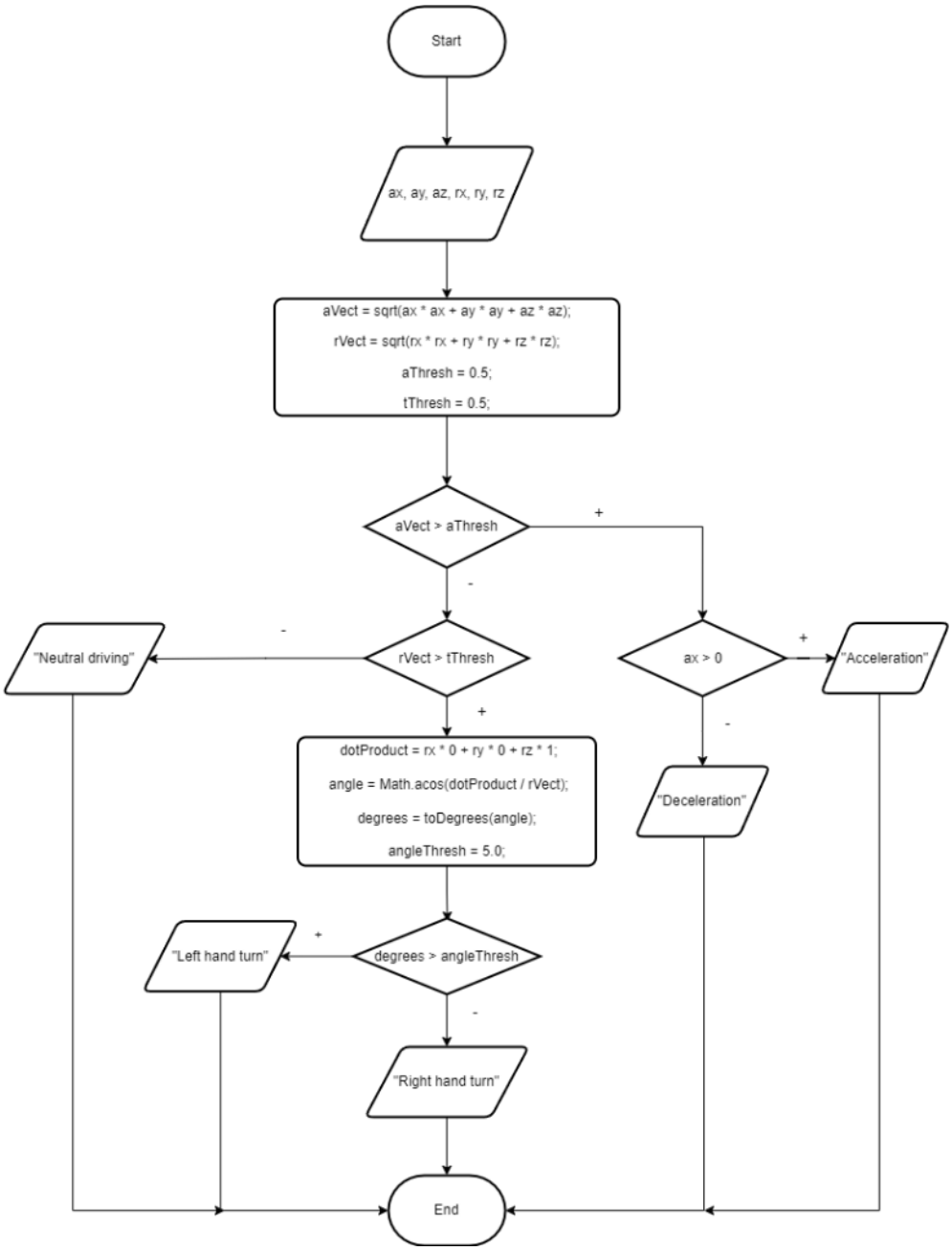

Рисунок 4.5 — Алгоритм розпізнавання маневру

Отриманий висновок зберігається як поточний стан, після чого робота алгоритму продовжується вже з наступним записом із датасета, і в той момент, коли стан змінюється, маневр, який щойно закінчився записується до бази даних разом з вирахуваною середньою швидкістю методом біжучого середнього, та дельтою палива на початку та в його кінці.

4.5 Алгоритм зберігання даних

Після визначення маневру, описаного у попередньому розділі. Відбувається запис інформації про маневр до бази даних. Як було зазначено вище, база даних складається х однієї таблиц з наступними даними:

- назва маневра, що зберігається у рядковому типі даних varchar;
- середня швидкість під час виконання маневру;
- кількість палива, витраченого за маневр.

Схема алгоритму зберыгання даних представлена на рисунку 4.6.

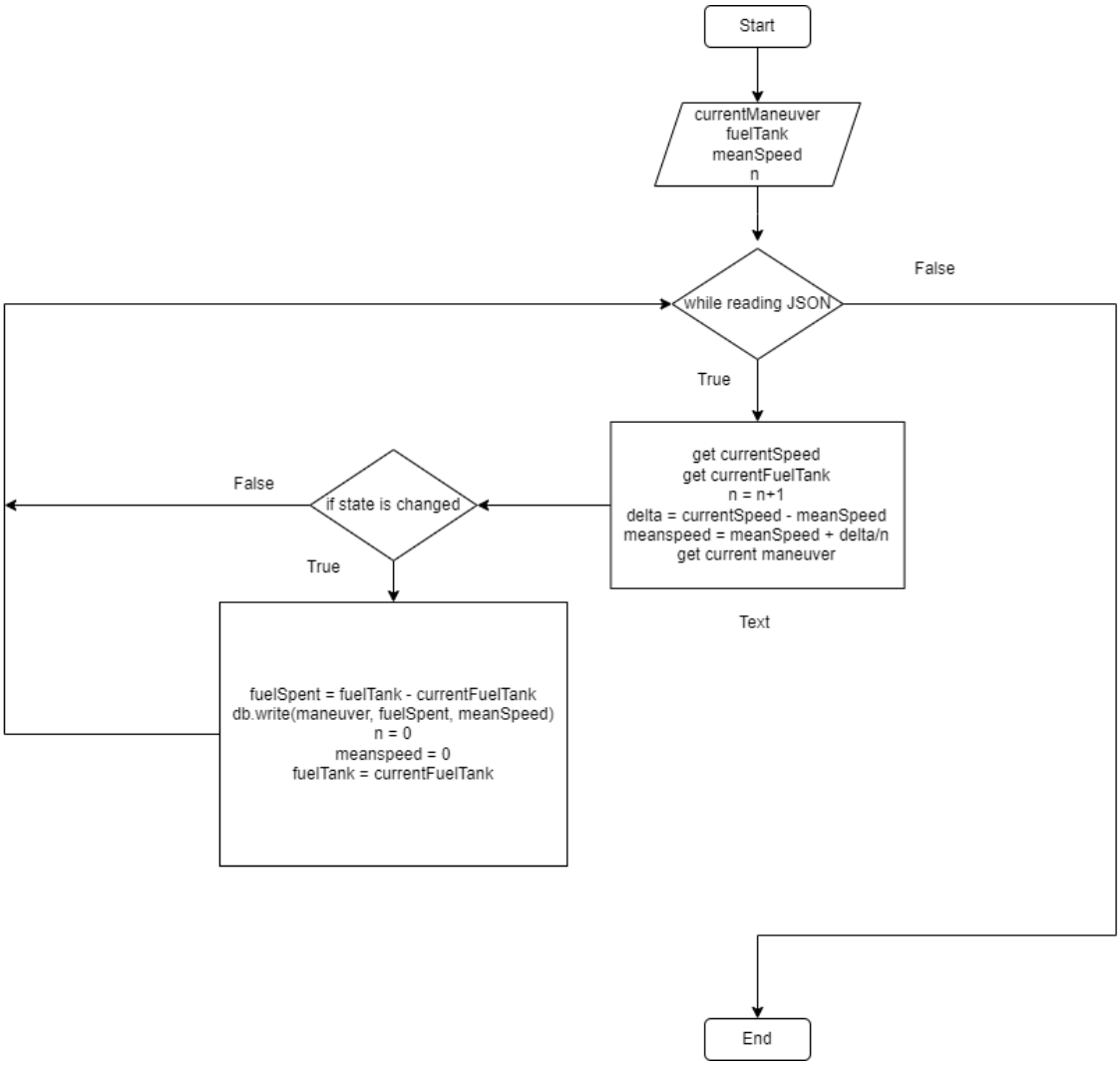

Рисунок 4.6 — Алгоритм зберігання даних

Кількість палива, використаного під виконання маневру, вираховується наступним чином.

У головній частині класу основного екрану застосунку оголошена змінна, що використовується для зберігання значення поточної кількості паливав баці.

Під час зміни стану поточного маневру, який повертається методом з реалізацією алгоритму описаному в попередньому розділі, шляхом віднімання знаходиться різниця поточної кількості палива, та кількості палива під час попередньої зміни стану, що зберігається у вищезазначеній змінній. Після цього, поточна кількість палива записується до змінної з попереднім значення, та робота алгоритму продовжується до наступної зміни станів.

Тепер варто описати процес вирахування середньої щвидкості транспортного засобу під час виконання маневру.

У JSON файлі, окрім даних про показники акселерометра, гіроскоппу та огб'єму палива в баці також міститься показник поточної відносної швидкості переміщення мобільного пристрою у просторі, отриманий за допомогою GPS.

Обчислення середньої швидкості відбувається в рамках кожної ітерації за формулою біжучого середнього, яка вказана у формулі 4.2.

$$\text{avgs} = \frac{(\text{S}' - \text{S})}{\text{n}} \tag{4.2}$$

де S’ — поточна швидкість;

S — попередня середня швидкість;

n — поточна кількість ітерацій;

avgs — нова середня швидкість.

Після зміни стану маневру, значення середньої швидкості разом з типом маневру да використаним паливом записується до бази даних, а значення кількості ітерацій n та поточної середньої швидкості стираються, щоб почати заново вираховування середньої швидкості наступного маневру.

Для запису параметрів до бази даних створено клас DatabaseHelper, який містить в собі реалізацію відповідних запитів, для запису та зчитування даних про

маневри із БД. У класі описана назви БД, назва таблиці разом з колонками для записів, та реалізовані методи описані нижче:

- addManeuver, що використовуєтьсядля додання запису про маневр до таблиці після зміни стану поточного маневру у алгоритмі розпізнавання, разом з даними про використання палива та середню швидкість;
- getAllManeuvers, який використовується для отримання усіх записів таблиці для заповнення графіків та таблиць у графічному інтерфейсі користувача;
- getManeuver використовується для отримання запису з таблиці за id, для відображення даних про обраний маневр на екрані DetailActivity.

Також для роботи з сутністю маневру у таблиці було створено клас Maneuver, який містить атрибути полів у таблиці. Перелік відповідних полів наведений нижче.

1. Параметр id, у якому зберігається ідентифікаційний номер запису.
2. Параметр maneuver, у якому зберігається назва маневру.
3. Параметр meanSpeed, у якому зберігається середня швидкість під час виконання манверу.
4. Параметр fuelSpent, у якому зберігається кількість палива, витраченого за час виконання маневру.

Висновки до розділу 4

У розділі розглянуто архітектуру системи та створено відповідні схеми, зокрема розбір екранів застосунку разом з процесами, що були реалізовані у відповідних класах-контролерах.

Створений детальний опис роботи алгоритму фільтрації даних про показники сенсорів, таких як акселерометр та гіроскоп.

Описано алгоритм розпізнавання маневрів танспортного засобу та описано реалізацію вирахування аналітичних даних зі спожититого палива та середньої швидкості за час виконання манвевру.

Створено детальні схеми роботи алгоритмів, що використовуються в роботі системи.

5 РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ

5.1 Системні вимоги

Мінімальні системні вимоги для встановлення та роботи застосунку описані нижче.

1. Операційна система Android не нижче версії 7.0.
2. 100 мб оперативної пам’яті.
3. Процесор з частотою від 0.5 ГГц.

5.2 Встановлення системи

Як зображено на рисунку 5.1, користувач повинен встановити застосунок на свій мобільний пристрій за допомогою файлу інсталяції з розширенням .apk.

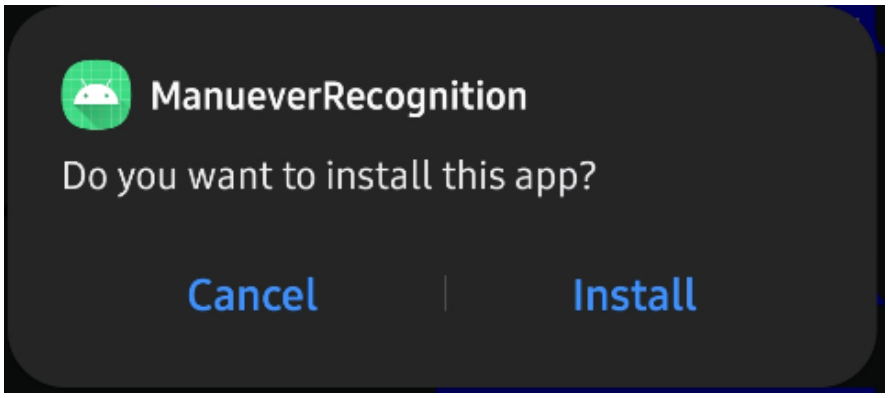

Рисунок 5.1 — Процес встановлення програмного застосунку

Після натискання користувачем кнопки Install, буде ропочато процес встановлення застосунку. З завершенням цього процесу, користувач зможе запустити та використовувати застосунок.

5.3 Робота користувача з системою

Після встановлення програмного забезпечення користувач може його запустити, та потрапити на основний екран застосунку, зображени на рисунку 5.2. Скориставшись кнопкою Start, можливо почати процес аналізу вхідних даних для розпізнавання маневрів та формування аналітичних даних відповідно до кожного.

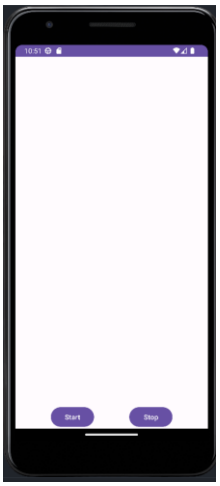

Рисунок 5.2 — Вигляд основного екрану програмного застосунку

Після завершення аналізу даних користувач потрапляє на сторінку з результатами поїздки. На цій сторінці, яка зображена на рисунку 5.3, представлені дані про кожний маневр поїздки разом з середньою швидкістю у кілометрах за годину та обсягом палива витраченого під час його виконання вимірюваного у літрах. Для представлення даних було використано клас RecyclerView.

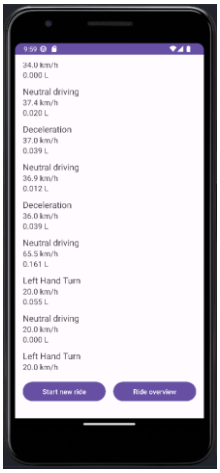

Рисунок 5.3 — Вигляд екрану з представленням результатів роботи алгоритму

За допомогою кнопки Start new ride можливо повернутися на головний екран застосунку. При натисканні кнопки Ride overview, користувач потрапляє на екран із загальними даними про поїздку, який зображено на рисунку 5.4. На ньому демонструються такі показники як:

- загальна кількість палива, витрачена за час поїздки;
- середня кількість витраченого палива на маневр;
- середня швидкість для усіх маневрів;
- відсоток маневрів які можливо могли бути виконані з перевищенням максимально дозволеної швидкості.

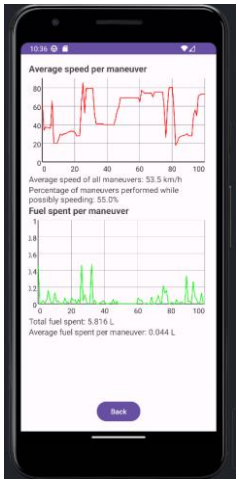

Рисунок 5.4 — Вигляд екрану з загальними даними про поїздку

Також користувач може побачити графіки використання палива та швидкості маневрів для візуалізації даних в динаміці

При натисканні кнопки Back користувач може повернутися на попередню сторінку з таблицею.

Також при натисканні на елемент таблиці коистувач потряпляє на екран аналізом даних конкретного маневра, зображений на рисунку 5.5.

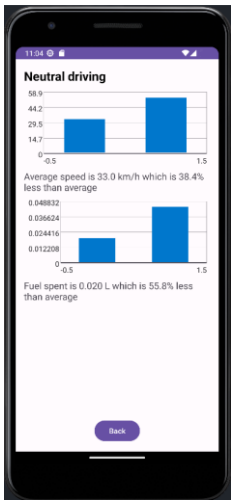

Рисунок 5.5 — Вигляд екрану з аналізом даних про маневр

Вираховується відношення у відсотках між середніми значеннями за поїздку та відповідними значеннями обраного маневра. Для візуалізації даних створено Bar Chart, який демонструє відношення візуально.

нтерфейсі, але допомагає зменшити кількість неактуальної інформації.

Висновки до розділу 5

Показано роботу додатку, включаючи його функціонал та дизайн інтерфейсу, які були належним чином налаштовані для відповідності потребам та ролі користувача.

ВИСНОВКИ

У ході виконання розроблено мобільний застосунок для розпізнавання маневрів транспортних засобів та надання аналітичної інформації про споживання палива та середню швидкість. На основі поставлених задач, було проведено роботу описану нижче.

1. Проаналізовано існуючі алгоритми розпізнавання маневрів транспортних засобів. Сформувано вимоги для реалізації програмного застосунку, а саме необхідність обробки даних сенсорів, та створення графічного інтерфейсу користувача з представленням аналітики про подорож.

2. Визначено необхідні дані для алгоритму розпізнавання маневрів, а саме дані сенсорів акселерометра та гіроскопа мобільного пристрою.

3. Досліджено наявні технології та інструменти розробки мобільних застосунків. Обрано засоби розробки, такі як мова програмування Java, SQLite у якості системи керування базами даних та інтегроване середовище розробки Android Studio.

4. Розроблено архітектуру програмного застосунку, яка містить такі складові як модель, представлення та контролер.

5. Розроблено алгоритм розпізнавання маневрів транспортних засобів., який визначає зміну стану виконуваного маневру, шляхом порівняння показників акселерометра зі сформованими правилами.

6. Розроблено програмний застосунок для розпізнавання маневрів транспортних засобів з подальшим виведенням їх списку та аналітичних даних.

7. Дипломна робота виконана в рамках проекту Driver’s behavior cognition розпізнавання поведінки водія (спільно із Політехнічним інститутом м. Томар, Португалія).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A Deep Learning Approach to Detect Real-Time Vehicle Maneuvers Based on Smartphone Sensors / P. Li et al. IEEE Transactions on Intelligent Transportation Systems. 2020. P. 1–10. URL: https://doi.org/10.1109/tits.2020.3032055 (date of access: 02.06.2023)..

2. Jurecki R. S., Stańczyk T. L., Ziubiński M. Analysis of the Structure of Driver Maneuvers in Different Road Conditions. Energies. 2022. Vol. 15, no. 19. P. 7073. URL: https://doi.org/10.3390/en15197073 (date of access: 02.06.2023).

3. Driver behavior profiling: An investigation with different smartphone sensors and machine learning / J. Ferreira et al. PLOS ONE. 2017. Vol. 12, no. 4. P. e0174959. URL: https://doi.org/10.1371/journal.pone.0174959 (date of access: 02.06.2023).

4. Saiprasert C., Pholprasit T., Thajchayapong S. Detection of Driving Events using Sensory Data on Smartphone. International Journal of Intelligent Transportation Systems Research. 2015. Vol. 15, no. 1. P. 17–28. URL: https://doi.org/10.1007/s13177-015-0116-5 (date of access: 02.06.2023).

5. Офіційна документація Java. URL: https://docs.oracle.com/en/java/ (дата звернення: 19.05.2023).

6. Офіційна документація Android Studio. URL: https://developer.android.com/studio (дата звернення: 26.05.2023).

7. XML Introduction. URL: https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction (date of access: 26.05.2023).

8. SQLite Documentation. SQLite Home Page. URL: https://www.sqlite.org/docs.html (date of access: 02.06.2023).

9. Source documentation of Android OS. Android Open Source Project. URL: https://source.android.com/docs?hl=ru (date of access: 02.06.2023).

10. Офіційна документація JSON. URL: https://www.json.org/json-en.html (дата звернення: 02.06.2023).

11. Gradle Documentation. Gradle User Manual. URL:

https://docs.gradle.org/current/userguide/userguide.html (date of access: 02.06.2023).

12. GitHub - jjoe64/GraphView: Android Graph Library for creating zoomable and scrollable line and bar graphs. GitHub. URL: https://github.com/jjoe64/GraphView (date of access: 02.06.2023).

13. LaLonde W. R., Pugh J. M., Lalonde W. Inside Smalltalk. Pearson Education, Limited, 1990. 500 p.

14. Library D. Inside WebObjects: WebObjects Desktop Applications. Fatbrain, 2003.

15. Action View Overview Ruby on Rails Guides. Ruby on Rails Guides. URL: https://guides.rubyonrails.org/action_view_overview.html (date of access: 02.06.2023).

ДОДАТОК А

Програмний застосунок розпізнавання маневрів транспортних засобів

Текст програми

Аркушів 24

Київ — 2023

MainActivity.java

```
package com.example.manueverrecognition;

import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;

import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import com.example.manueverrecognition.database.DatabaseHelper;
import com.example.manueverrecognition.sensors.SensorData;
import com.google.gson.Gson;
import com.google.gson.stream.JsonReader;

import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

@RequiresApi(api = Build.VERSION_CODES.O)
public class MainActivity extends AppCompatActivity {

    private static final String TAG = "MainActivity";

    private DatabaseHelper dbHelper;
    private static final double ALPHA = 0.2;
    private static final double BETA = 1 - ALPHA;

    private double prevAx = 0;
```

```
private double prevAy = 0;
private double prevAz = 0;
private double prevRx = 0;
private double prevRy = 0;
private double prevRz = 0;

private double fuelTank = 0;

private double totalFuelSpent;

private double meanSpeed = 0;
private Button startButton;
private Button stopButton;
private boolean readingJson = false;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    dbHelper = new DatabaseHelper(this);

    startButton = findViewById(R.id.start_button);
    stopButton = findViewById(R.id.stop_button);

    startButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            dbHelper.clearTable();
            readingJson = true;
            readJson();
        }
});
```

```
    stopButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            readingJson = false;
        }
    });
}

private void readJson() {
    try {
        InputStream is = getResources().openRawResource(R.raw.output);
        JsonReader jsonReader = new JsonReader(new InputStreamReader(is));
        jsonReader.setLenient(true);
        jsonReader.beginArray();

        String maneuver = "Neutral driving";
        int n = 0;

        while (jsonReader.hasNext() && readingJson) {
            SensorData sensorData = new Gson().fromJson(jsonReader, SensorData.class);

            double ax = sensorData.getAcceleration().getX();
            double ay = sensorData.getAcceleration().getY();
            double az = sensorData.getAcceleration().getZ();

            double rx = sensorData.getRotationRate().getX();
            double ry = sensorData.getRotationRate().getY();
            double rz = sensorData.getRotationRate().getZ();

            double currentSpeed = sensorData.getLocation().getSpeed();

            double currentFuelTank = sensorData.getOdb().getFuel_tank();

            ax = ALPHA * ax + BETA * prevAx;
```

```
    ay = ALPHA * ay + BETA * prevAy;
    az = ALPHA * az + BETA * prevAz;
    prevAx = ax;
    prevAy = ay;
    prevAz = az;

    rx = ALPHA * rx + BETA * prevRx;
    ry = ALPHA * ry + BETA * prevRy;
    rz = ALPHA * rz + BETA * prevRz;
    prevRx = rx;
    prevRy = ry;
    prevRz = rz;

    n++;
    double delta = currentSpeed - meanSpeed;
    meanSpeed += delta / n;


    String currentManeuver = determineManeuver(ax, ay, az, rx, ry, rz);

    if (!maneuver.equals(currentManeuver)){
      Log.d(TAG, "Maneuver: " + maneuver);

      double fuelSpent = Math.abs(currentFuelTank - fuelTank)/100;

      dbHelper.addManeuver(maneuver, meanSpeed, fuelSpent);

      totalFuelSpent += Math.abs(currentFuelTank - fuelTank);
      fuelTank = currentFuelTank;
      n = 0;
      meanSpeed = 0;
    }
    maneuver = currentManeuver;
}
```

```
            jsonReader.endArray();
            jsonReader.close();
            showTableActivity();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void showTableActivity() {
        Intent intent = new Intent(this, TableActivity.class);
        startActivity(intent);
    }

    public String determineManeuver(double ax, double ay, double az, double rx, double ry, double rz) {
        double accelerationMagnitude = Math.sqrt(ax * ax + ay * ay + az * az);

        double accelerationThreshold = 0.5;
        if (accelerationMagnitude > accelerationThreshold) {
            if (ax > 0) {
                return "Acceleration";
            } else {
                return "Deceleration";
            }
        }

        double gyroscopeMagnitude = Math.sqrt(rx * rx + ry * ry + rz * rz);

        double turnThreshold = 0.5;
        if (gyroscopeMagnitude > turnThreshold) {
            double dotProduct = rx * 0 + ry * 0 + rz * 1;

            double angle = Math.acos(dotProduct / gyroscopeMagnitude);
```

```
        double angleDegrees = Math.toDegrees(angle);

        double turnAngleThreshold = 5.0;
        if (angleDegrees > turnAngleThreshold) {
            if (ry > 0) {
                return "Left Hand Turn";
            } else {
                return "Right Hand Turn";
            }
        }
    }

    return "Neutral driving";
}

@Override
protected void onResume() {
    super.onResume();
}

@Override
protected void onPause() {
    super.onPause();
}

@Override
protected void onDestroy() {
    super.onDestroy();
    dbHelper.clearTable();
}

@Override
public void onLowMemory() {
    super.onLowMemory();
```

```
        }
    }
```

TableActivity.java

```
package com.example.manueverrecognition;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.manueverrecognition.adapters.RecyclerAdapter;
import com.example.manueverrecognition.database.DatabaseHelper;
import com.example.manueverrecognition.database.Maneuver;
import com.example.manueverrecognition.decoration.ItemSpacingDecoration;

import java.util.ArrayList;
import java.util.List;

public class TableActivity extends AppCompatActivity {
    private DatabaseHelper dbHelper;
    private RecyclerView recyclerView;
    private RecyclerAdapter adapter;
    private Button newRideButton;
    private Button overviewButton;

    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_table);

        dbHelper = new DatabaseHelper(this);

        recyclerView = findViewById(R.id.recycler_view);
        newRideButton = findViewById(R.id.start_new_ride_button);
        overviewButton = findViewById(R.id.overview_button);

        LinearLayoutManager layoutManager = new LinearLayoutManager(this);
        recyclerView.setLayoutManager(layoutManager);
```

```
        List<Maneuver> maneuvers = getManeuversFromDatabase();

        adapter = new RecyclerAdapter(this, maneuvers);
        recyclerView.setAdapter(adapter);

        int spacingInPixels = 16;
        recyclerView.addItemDecoration(new ItemSpacingDecoration(spacingInPixels));


        newRideButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showMainActivity();
            }
        });

        overviewButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showOverviewActivity();
            }
        });

        dbHelper.close();
    }

    private List<Maneuver> getManeuversFromDatabase() {
        List<Maneuver> maneuvers = new ArrayList<>();

        Cursor cursor = dbHelper.getAllManeuvers();

        if (cursor.moveToFirst()) {
            do {
                @SuppressLint("Range") int id = cursor.getInt(cursor.getColumnIndex("_id"));
                @SuppressLint("Range") String maneuverName =
cursor.getString(cursor.getColumnIndex("maneuver"));
                @SuppressLint("Range") double meanSpeed =
cursor.getDouble(cursor.getColumnIndex("meanSpeed"));
                @SuppressLint("Range") double fuelSpent =
cursor.getDouble(cursor.getColumnIndex("fuelSpent"));

                Maneuver maneuver = new Maneuver(id, maneuverName, meanSpeed, fuelSpent);
                maneuvers.add(maneuver);
            } while (cursor.moveToNext());
        }

        cursor.close();

        return maneuvers;
    }
```

```
    private void showMainActivity(){
        Intent intent = new Intent(TableActivity.this, MainActivity.class);
        startActivity(intent);
    }

    private void showOverviewActivity(){
        Intent intent = new Intent(TableActivity.this, OverviewActivity.class);
        startActivity(intent);
    }
}
```

OverviewActivity.java

```
package com.example.manueverrecognition;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.database.Cursor;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import com.example.manueverrecognition.database.DatabaseHelper;
import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.series.DataPoint;
import com.jjoe64.graphview.series.LineGraphSeries;

public class OverviewActivity extends AppCompatActivity {

    private TextView totalFuelSpentTextView;
    private TextView averageFuelSpentTextView;
    private TextView averageSpeedTextView;
    private TextView percentageFastManeuversTextView;
    private Button showTableButton;
    private DatabaseHelper databaseHelper;

    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_overview);
        databaseHelper = new DatabaseHelper(this);
        totalFuelSpentTextView = findViewById(R.id.total_fuel_spent_text_view);
        averageFuelSpentTextView = findViewById(R.id.average_fuel_spent_text_view);
        averageSpeedTextView = findViewById(R.id.average_speed_text_view);
```

```
        percentageFastManeuversTextView =
findViewById(R.id.percentage_fast_maneuvers_text_view);
        showTableButton = findViewById(R.id.show_table_button);

        showTableButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(OverviewActivity.this, TableActivity.class);
                startActivity(intent);
            }
        });

        GraphView speedGraphView = findViewById(R.id.graph_speed);
        GraphView fuelGraphView = findViewById(R.id.graph_fuel);

        LineGraphSeries<DataPoint> speedSeries = new
LineGraphSeries<>(generateSpeedDataPoints());

        speedSeries.setColor(Color.RED);
        speedSeries.setThickness(4);

        speedGraphView.addSeries(speedSeries);

        speedGraphView.getViewport().setXAxisBoundsManual(true);
        speedGraphView.getViewport().setMinX(0);
        speedGraphView.getViewport().setMaxX(100);
        speedGraphView.getViewport().setYAxisBoundsManual(true);
        speedGraphView.getViewport().setMinY(0);
        speedGraphView.getViewport().setMaxY(90);

        Cursor cursor = databaseHelper.getAllManeuvers();

        double totalFuelSpent = 0;
        int totalManeuvers = 0;
        double totalMeanSpeed = 0;
        int fastManeuvers = 0;

        if (cursor.moveToFirst()) {
            do {
                @SuppressLint("Range") double fuelSpent =
cursor.getDouble(cursor.getColumnIndex("fuelSpent"));
                @SuppressLint("Range") double meanSpeed =
cursor.getDouble(cursor.getColumnIndex("meanSpeed"));
                totalFuelSpent += fuelSpent;
                totalManeuvers++;
                totalMeanSpeed += meanSpeed;
                if (meanSpeed > 50) {
                    fastManeuvers++;
                }
            } while (cursor.moveToNext());
```

```
        }

        double averageFuelSpent = totalFuelSpent / totalManeuvers;
        double averageSpeed = totalMeanSpeed / totalManeuvers;
        double percentageFastManeuvers = ((double) fastManeuvers / totalManeuvers) * 100;

        totalFuelSpentTextView.setText("Total fuel spent: " + String.format("%.3f", totalFuelSpent) + "
L");
        averageFuelSpentTextView.setText("Average fuel spent per maneuver: " + String.format("%.3f",
averageFuelSpent) + " L");
        averageSpeedTextView.setText("Average speed of all maneuvers: " + String.format("%.1f",
averageSpeed) + " km/h");
        percentageFastManeuversTextView.setText("Percentage of maneuvers performed while possibly
speeding: " + String.format("%.1f", percentageFastManeuvers) + "%");

        LineGraphSeries<DataPoint> fuelSeries = new LineGraphSeries<>(generateFuelDataPoints());

        fuelSeries.setColor(Color.GREEN);
        fuelSeries.setThickness(4);

        fuelGraphView.addSeries(fuelSeries);

        fuelGraphView.getViewport().setXAxisBoundsManual(true);
        fuelGraphView.getViewport().setMinX(0);
        fuelGraphView.getViewport().setMaxX(100);
        fuelGraphView.getViewport().setYAxisBoundsManual(true);
        fuelGraphView.getViewport().setMinY(0);
        fuelGraphView.getViewport().setMaxY(1);

    }

    private DataPoint[] generateSpeedDataPoints() {
        Cursor cursor = databaseHelper.getAllManeuvers();

        int count = cursor.getCount();

        DataPoint[] dataPoints = new DataPoint[count];

        if (cursor.moveToFirst()) {
            int maneuverIndex = cursor.getColumnIndex("maneuver");
            int meanSpeedIndex = cursor.getColumnIndex("meanSpeed");
            for (int i = 0; i < count; i++) {
                String maneuver = cursor.getString(maneuverIndex);
                double meanSpeed = cursor.getDouble(meanSpeedIndex);
                dataPoints[i] = new DataPoint(i, meanSpeed);
                cursor.moveToNext();
            }
        }

        cursor.close();
```

```
        return dataPoints;
    }

    private DataPoint[] generateFuelDataPoints() {
        Cursor cursor = databaseHelper.getAllManeuvers();

        int count = cursor.getCount();

        DataPoint[] dataPoints = new DataPoint[count];

        if (cursor.moveToFirst()) {
            int maneuverIndex = cursor.getColumnIndex("maneuver");
            int fuelSpentIndex = cursor.getColumnIndex("fuelSpent");
            for (int i = 0; i < count; i++) {
                String maneuver = cursor.getString(maneuverIndex);
                double fuelSpent = cursor.getDouble(fuelSpentIndex);
                dataPoints[i] = new DataPoint(i, fuelSpent);
                cursor.moveToNext();
            }
        }

        cursor.close();

        return dataPoints;
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        databaseHelper.close();
    }
}
```

DetailActivity.java

```
package com.example.manueverrecognition;

import android.annotation.SuppressLint;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import com.example.manueverrecognition.database.DatabaseHelper;
import com.example.manueverrecognition.database.Maneuver;
import com.jjoe64.graphview.DefaultLabelFormatter;
```

```
import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.series.BarGraphSeries;
import com.jjoe64.graphview.series.DataPoint;

import java.util.ArrayList;
import java.util.List;

public class DetailActivity extends AppCompatActivity {
    private DatabaseHelper dbHelper;

    private TextView maneuverTextView;
    private TextView meanSpeedTextView;
    private TextView fuelSpentTextView;
    private GraphView fuelSpentGraph;
    private GraphView meanSpeedGraph;

    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detail);

        dbHelper = new DatabaseHelper(this);

        maneuverTextView = findViewById(R.id.maneuver_textview);
        meanSpeedTextView = findViewById(R.id.mean_speed_textview);
        fuelSpentTextView = findViewById(R.id.fuel_spent_textview);
        fuelSpentGraph = findViewById(R.id.fuel_spent_graph);
        meanSpeedGraph = findViewById(R.id.mean_speed_graph);

        int maneuverId = getIntent().getIntExtra("maneuverId", -1);
        Maneuver maneuver = dbHelper.getManeuver(maneuverId);

        if (maneuver != null) {
            maneuverTextView.setText(maneuver.getManeuver());
            meanSpeedTextView.setText("Average speed is " + String.format("%.1f",
maneuver.getMeanSpeed()) + " km/h");
            fuelSpentTextView.setText("Fuel spent is " + String.format("%.3f", maneuver.getFuelSpent())
+ " L");

            double averageFuelSpent = calculateAverageFuelSpent();
            double averageMeanSpeed = calculateAverageMeanSpeed();

            compareAndDisplayMessage(maneuver.getFuelSpent(), averageFuelSpent,
fuelSpentTextView);
            compareAndDisplayMessage(maneuver.getMeanSpeed(), averageMeanSpeed,
meanSpeedTextView);

            createBarChart(fuelSpentGraph, maneuver.getFuelSpent(), averageFuelSpent, "Fuel Spent");
```

```
            createBarChart(meanSpeedGraph, maneuver.getMeanSpeed(), averageMeanSpeed, "Mean
Speed");
        }

        dbHelper.close();

        Button returnButton = findViewById(R.id.return_button);
        returnButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
    }

    private double calculateAverageFuelSpent() {
        double totalFuelSpent = 0;
        int totalManeuvers = 0;
        dbHelper = new DatabaseHelper(this);
        Cursor cursor = dbHelper.getAllManeuvers();
        if (cursor != null && cursor.moveToFirst()) {
            do {
                @SuppressLint("Range") double fuelSpent =
cursor.getDouble(cursor.getColumnIndex("fuelSpent"));
                totalFuelSpent += fuelSpent;
                totalManeuvers++;
            } while (cursor.moveToNext());
            cursor.close();
        }
        dbHelper.close();
        return totalFuelSpent / totalManeuvers;
    }

    private double calculateAverageMeanSpeed() {
        double totalMeanSpeed = 0;
        int totalManeuvers = 0;
        dbHelper = new DatabaseHelper(this);
        Cursor cursor = dbHelper.getAllManeuvers();
        if (cursor != null && cursor.moveToFirst()) {
            do {
                @SuppressLint("Range") double meanSpeed =
cursor.getDouble(cursor.getColumnIndex("meanSpeed"));
                totalMeanSpeed += meanSpeed;
                totalManeuvers++;
            } while (cursor.moveToNext());
            cursor.close();
        }
        dbHelper.close();
        return totalMeanSpeed / totalManeuvers;
    }
```

```
    private void compareAndDisplayMessage(double currentValue, double averageValue, TextView
textView) {
        double percentageDifference = ((currentValue - averageValue) / averageValue) * 100;
        String message;
        if (percentageDifference < 0) {
            message = " which is " + String.format("%.1f", Math.abs(percentageDifference)) + "% less
than average";
        } else if (percentageDifference > 0) {
            message = " which is " + String.format("%.1f", Math.abs(percentageDifference)) + "% higher
than average";
        } else {
            message = " which is equal to the average";
        }
        String originalText = textView.getText().toString();
        textView.setText(originalText + message);
    }

    private void createBarChart(GraphView graph, double currentValue, double averageValue, String
label) {
        List<DataPoint> dataPoints = new ArrayList<>();
        dataPoints.add(new DataPoint(0, currentValue));
        dataPoints.add(new DataPoint(1, averageValue));

        BarGraphSeries<DataPoint> series = new BarGraphSeries<>(dataPoints.toArray(new
DataPoint[0]));
        series.setSpacing(50);

        graph.addSeries(series);
        graph.getViewport().setMinX(-0.5);
        graph.getViewport().setMaxX(1.5);
        graph.getViewport().setXAxisBoundsManual(true);
        graph.getGridLabelRenderer().setNumHorizontalLabels(2);
        graph.getGridLabelRenderer().setHorizontalLabelsVisible(true);
        graph.getGridLabelRenderer().setHumanRounding(false);
        graph.getGridLabelRenderer().setLabelFormatter(new DefaultLabelFormatter() {
            @Override
            public String formatLabel(double value, boolean isValueX) {
                if (isValueX) {
                    if (value == 0) {
                        return "Current";
                    } else if (value == 1) {
                        return "Average";
                    }
                }
                return super.formatLabel(value, isValueX);
            }
        });

        graph.getViewport().setYAxisBoundsManual(true);
```

```
        graph.getViewport().setMinY(0);
        graph.getViewport().setMaxY(Math.max(currentValue, averageValue) * 1.1);
    }
}
```

DatabaseHelper.java

```
package com.example.manueverrecognition.database;

import android.annotation.SuppressLint;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "mydatabase.db";
    private static final int DATABASE_VERSION = 2;
    private static final String TABLE_NAME = "maneuvers";
    private static final String COLUMN_ID = "_id";
    private static final String COLUMN_MANEUVER = "maneuver";
    private static final String COLUMN_MEAN_SPEED = "meanSpeed";
    private static final String COLUMN_FUEL_SPENT = "fuelSpent";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String maneuverTableSql = "CREATE TABLE " + TABLE_NAME + " (" +
                COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                COLUMN_MANEUVER + " TEXT, " +
                COLUMN_MEAN_SPEED + " REAL, " +
                COLUMN_FUEL_SPENT + " REAL)";
        db.execSQL(maneuverTableSql);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }

        public void addManeuver(String maneuver, double meanSpeed, double fuelSpent) {
            ContentValues values = new ContentValues();
            values.put(COLUMN_MANEUVER, maneuver);
            values.put(COLUMN_MEAN_SPEED, meanSpeed);
            values.put(COLUMN_FUEL_SPENT, fuelSpent);
```

```
        SQLiteDatabase db = getWritableDatabase();
        db.insert(TABLE_NAME, null, values);
    }

    public Cursor getAllManeuvers() {
        SQLiteDatabase db = getReadableDatabase();
        return db.rawQuery("SELECT * FROM " + TABLE_NAME, null);
    }

    public Maneuver getManeuver(int maneuverId) {
        SQLiteDatabase db = getReadableDatabase();
        String selection = COLUMN_ID + " = ?";
        String[] selectionArgs = { String.valueOf(maneuverId) };
        Cursor cursor = db.query(TABLE_NAME, null, selection, selectionArgs, null, null, null);

        Maneuver maneuver = null;
        if (cursor.moveToFirst()) {
            @SuppressLint("Range") String maneuverName =
cursor.getString(cursor.getColumnIndex(COLUMN_MANEUVER));
            @SuppressLint("Range") double meanSpeed =
cursor.getDouble(cursor.getColumnIndex(COLUMN_MEAN_SPEED));
            @SuppressLint("Range") double fuelSpent =
cursor.getDouble(cursor.getColumnIndex(COLUMN_FUEL_SPENT));

            maneuver = new Maneuver(maneuverId, maneuverName, meanSpeed, fuelSpent);
        }

        cursor.close();
        return maneuver;
    }



    public void clearTable() {
        SQLiteDatabase db = getWritableDatabase();
        db.execSQL("DELETE FROM " + TABLE_NAME);
    }
}
```

Maneuver.java

```
package com.example.manueverrecognition.database;

public class Maneuver {

    int id;
    private String maneuver;
    private double meanSpeed;
    private double fuelSpent;
```

```
    public Maneuver(int id, String maneuver, double meanSpeed, double fuelSpent) {
        this.id = id;
        this.maneuver = maneuver;
        this.meanSpeed = meanSpeed;
        this.fuelSpent = fuelSpent;
    }

    public int getId() {
        return id;
    }

    public String getManeuver() {
        return maneuver;
    }

    public double getMeanSpeed() {
        return meanSpeed;
    }

    public double getFuelSpent() {
        return fuelSpent;
    }
}
```

SensorData.java

```
package com.example.manueverrecognition.sensors;

public class SensorData {
    private Acceleration acceleration;
    private RotationRate rotation_rate;
    private Gravity gravity;
    private Location location;
    private Odb odb;

    public Acceleration getAcceleration() {
        return acceleration;
    }

    public void setAcceleration(Acceleration acceleration) {
        this.acceleration = acceleration;
    }

    public RotationRate getRotationRate() {
        return rotation_rate;
    }

    public void setRotationRate(RotationRate rotation_rate) {
        this.rotation_rate = rotation_rate;
```

```
}

public Gravity getGravity() {
   return gravity;
}

public void setGravity(Gravity gravity) {
   this.gravity = gravity;
}

public Location getLocation() {
   return location;
}

public void setLocation(Location location) {
   this.location = location;
}

public Odb getOdb() {
   return odb;
}

public void setOdb(Odb odb) {
   this.odb = odb;
}

public static class Acceleration {
   private double x;
   private double y;
   private double z;

   public double getX() {
      return x;
   }

   public void setX(double x) {
      this.x = x;
   }

   public double getY() {
      return y;
   }

   public void setY(double y) {
      this.y = y;
   }

   public double getZ() {
      return z;
   }
```

```
    public void setZ(double z) {
        this.z = z;
    }
}

public static class RotationRate {
    private double x;
    private double y;
    private double z;

    public double getX() {
        return x;
    }

    public void setX(double x) {
        this.x = x;
    }

    public double getY() {
        return y;
    }

    public void setY(double y) {
        this.y = y;
    }

    public double getZ() {
        return z;
    }

    public void setZ(double z) {
        this.z = z;
    }
}

public static class Gravity {
    private double x;
    private double y;
    private double z;

    public double getX() {
        return x;
    }

    public void setX(double x) {
        this.x = x;
    }

    public double getY() {
```

```
        return y;
    }

    public void setY(double y) {
        this.y = y;
    }

    public double getZ() {
        return z;
    }

    public void setZ(double z) {
        this.z = z;
    }
}

public static class Location {
    private String datetime;
    private int speed;
    private double horizontal_accuracy;
    private double longitude;
    private double latitude;
    private double course;

    public String getDatetime() {
        return datetime;
    }

    public void setDatetime(String datetime) {
        this.datetime = datetime;
    }

    public int getSpeed() {
        return speed;
    }

    public void setSpeed(int speed) {
        this.speed = speed;
    }

    public double getHorizontalAccuracy() {
        return horizontal_accuracy;
    }

    public void setHorizontalAccuracy(double horizontal_accuracy) {
        this.horizontal_accuracy = horizontal_accuracy;
    }

    public double getLongitude() {
        return longitude;
```

```
        }

        public void setLongitude(double longitude) {
            this.longitude = longitude;
        }

        public double getLatitude() {
            return latitude;
        }

        public void setLatitude(double latitude) {
            this.latitude = latitude;
        }

        public double getCourse() {
            return course;
        }

        public void setCourse(double course) {
            this.course = course;
        }
    }

    public static class Odb {
        private double fuel_tank;

        public double getFuel_tank() {
            return fuel_tank;
        }

        public void setFuel_tank(double fuel_tank) {
            this.fuel_tank = fuel_tank;
        }
    }
}
```

RecyclerAdapter.java

```
package com.example.manueverrecognition.adapters;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
```

```
import com.example.manueverrecognition.R;
import com.example.manueverrecognition.database.Maneuver;
import com.example.manueverrecognition.DetailActivity;

import java.util.List;

public class RecyclerAdapter extends
RecyclerView.Adapter<RecyclerAdapter.ManeuverViewHolder> {
   private Context context;
   private List<Maneuver> maneuvers;

   public RecyclerAdapter(Context context, List<Maneuver> maneuvers) {
      this.context = context;
      this.maneuvers = maneuvers;
   }

   @NonNull
   @Override
   public ManeuverViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
      View view = LayoutInflater.from(context).inflate(R.layout.grid_item, parent, false);
      return new ManeuverViewHolder(view);
   }

   @Override
   public void onBindViewHolder(@NonNull ManeuverViewHolder holder, int position) {
      Maneuver maneuver = maneuvers.get(position);
      holder.bind(maneuver);
   }

   @Override
   public int getItemCount() {
      return maneuvers.size();
   }

   class ManeuverViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener
{
      private TextView maneuverTextView;
      private TextView meanSpeedTextView;
      private TextView fuelSpentTextView;

      ManeuverViewHolder(@NonNull View itemView) {
         super(itemView);
         maneuverTextView = itemView.findViewById(R.id.maneuver_textview);
         meanSpeedTextView = itemView.findViewById(R.id.mean_speed_textview);
         fuelSpentTextView = itemView.findViewById(R.id.fuel_spent_textview);
         itemView.setOnClickListener(this);
      }

      void bind(Maneuver maneuver) {
```

```
            maneuverTextView.setText(maneuver.getManeuver());
            meanSpeedTextView.setText(String.format("%.1f", maneuver.getMeanSpeed()) + " km/h");
            fuelSpentTextView.setText(String.format("%.3f", maneuver.getFuelSpent()) + " L");
        }

        @Override
        public void onClick(View v) {
            int position = getAdapterPosition();
            if (position != RecyclerView.NO_POSITION) {
                Maneuver selectedManeuver = maneuvers.get(position);
                Intent intent = new Intent(context, DetailActivity.class);
                intent.putExtra("maneuverId", selectedManeuver.getId());
                context.startActivity(intent);
            }
        }
    }
}
```

ДОДАТОК Б

Програмний застосунок розпізнавання маневрів транспортних засобів

Апробація результатів роботи

Аркушів 3

Київ — 2023

ІНСТРУМЕНТАЛЬНІ ЗАСОБИ РОЗПІЗНАВАННЯ МАНЕВРІВ ТРАНСПОРТНИХ ЗАСОБІВ

Зі зростанням діджиталізації технічних та побутових процесів, все ширше має місце інтегрування цифрових технологій у ці сфери. Зокрема, створюється безмежна кількість мобільних додатків для виконання різноманітних задач від підрахунку калоражу їжі до створення впорядкованих списків або розкладів. Окремо розглянемо використання даного підходу у сфері енергоефективного водіння.

З кожним роком, враховуючи часті енергетичні кризи, людство почало відноситися до проблеми енергетики серйозніше, намагаючись збільшити ефективність генерації та споживання енергії, що також актуально і у розрізі сфери транспорту. Під час керування транспортним засобом, водій виконує маневри, кожен з яких, власне, споживає певний обсяг палива або заряду батареї. Подорож з пункту А до пункту Б можна уявити як послідовність маневрів, виконаних транспортним засобом, при чому ця послідовність не є фіксованою. Тобто, враховуючи це, ми можемо дійти до висновку, можуть існувати два еквівалентних маневра з точки зору переміщення з однієї точки до іншої, які при цьому використали різну кількість енергії. Отже це означає, що існує така послідовність маневрів при переміщенні з пункту А до пункту Б, при якому будуть витрачені найменші обсяги енергії.

Розглянемо таку задачу. Можливе створення мобільного додатку для допомоги водієві підвищити енергоефективність керування його транспогртним засобом.

Мобільний додаток повинен бути спроможним розпізнавати маневр, який виконується автомобілем та вираховувати кількість палива, що була на нього витрачена та об’єднувати ці маневри в рамках однієї поїздки.

Задля реалізації розпізнавання маневрів можливо скористатися алгоритмом на базі правил, що буде використовувати дані з акселерометру мобільного телефону.

Розглянемо 3-осьовий акселерометр, який вимірює силу прискорення та гравітацію смартфона. В даному випадку нас цікавить рух по горизонтальній і вертикальній осях, які відповідають бічному рухові, та рухові вперед і назад. В реальному світі бічне прискорення відповідає поворотам ліворуч і праворуч і зміні смуги руху. Прискорення вздовж відповідає гальмуванню і прискоренню автомобіля.

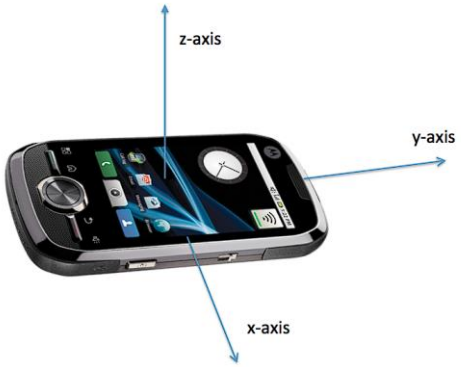

Рисунок 1 - 3-осьовий акселерометр

Інший датчик, магнітометр, вимірює силу магнітного поля. Він вказує, у якому напрямку ваш смартфон розташований відносно магнітної півночі. Наостанок маємо GPS приймач, який надає дані про місцезнаходження та швидкість автомобіля, до якого підключено ваш смартфон. Дані акселерометра та магнітометра знімаються з частотою 5 Гц, і вибірка записується кожні 200 мс.

Дані з приймача GPS дискретизуються з частотою 1 Гц. На малюнку 1 зображено акселерометр із відповідними осями. Бічний рух представлено віссю абсцисс, а вертикальний рух віссю ординат.

Також, у цей час мобільний телефон буде отримувати дані з OBD порту про запас палива оба заряду батареї та визначати, скільки саме було витрачено енергії для виконання маневру

Останні розробки в технології смартфонів і поширення смартфонів у всьому світі призвели до розробки багатьох нових додатків ITS. Пропонуються недорогі системи попередження про зміну смуги руху, основна ідея роботи полягає в застосуванні техніки обробки зображень камерою смартфона [1]. Крім того, запропонований алгоритм оптимізований для зображень низької якості та здатен виконуватись малопотужними смартфонами. Мохан та інші запропонували систему, яка використовує смартфони як пристрої моніторингу стану доріг і трафіку [2]. Результати були досягнуті шляхом використання вбудованих датчиків смартфона. Акселерометра та GPS виявляють вибоїни, ями, а також гальма та гудки автомобілів. Також був запропонований схожий метод, який використовує додаток для смартфона для аналізу дорожніх умов і збору даних з багатьох датчиків автомобіля для класифікації різних дорожніх перешкод з високою точністю [3]. Джонсон і Тріведі запропонували спосіб класифікації різних стилів керування за допомогою даних, зібраних зі смартфонів [4]. На їхню думку, стилі кервання поділяються на такі: помірний, агресивний, дуже агресивний. За результатами їхньої роботи моежмо зробити висновок, що різні датчики смартфона можуть бути чудовим джерелом точних вимірювань і оціночних даних про різні стилі водіння.

Перелік посилань:

1. M. Lan, M. Rofouei, S. Soatto, and M. Sarrafzadeh, ÒSmartLDWS: A Robust and Scalable Lane Departure Warning System for the Smartphones,Ó in Proceedings of the 12th International IEEE Conference on Intelligent Transportation Systems, 2009.
2. P. Mohan, V.N. Padmanabhan, and R. Ramjee, “Nericell: rich monitoring of road and traffic conditions using mobile smartphones", in Proceedings of the 6th ACM conference on Embedded network sensor systems, pp. 323-336, 2008.
3. M. Fazeen, B. Gozick, R. Dantu, M. Bhukhiya, and M. C. GonzGlez, “Safe Driving Using ˘ Mobile Phones," in IEEE Transaction on Intelligent Transportation Systems, vol. 13, no. 3, pp. 1462–1468, 2012.
4. D. A. Johnson and M. M. Trevidi, “Driving Style Recognition Using a Smartphone as a Sensor Platform," in Proceedings 14th International IEEE Conference on Intelligent Transportation Systems, pp. 1609–1615, 2011.