МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

Катерина НЕСТЕРЕНКО
“____”_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСНИКА ОСВІТНЬОГО СТУПЕНЯ
“МАГІСТР”

Тема: «Мобільний застосунок для створення мап за допомогою процесу Dead reckoning»

Виконавець: Михайленко Денис Олександрович

Керівник: к.т.н. с.н.с. доцент Клюєв Євген Іванович

Нормоконтролер: асистент Кравченко Ольга Сергіївна

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення
Освітній ступінь магістр
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Катерина НЕСТЕРЕНКО
"___" _____ 2023 р

ЗАВДАННЯ
на виконання дипломного проекту студента
Михайленка Дениса Олександровича

1. Тема проекту: «Мобільний застосунок для створення мап за допомогою процесу Dead Reckoning»
затверджена наказом ректора від 29.09.2023 р. № 1994/ст.

2. Термін виконання проекту: з 02.10.2023 р. до 31.12.2023 р.

3. Вихідні дані до проекту: Android застосунок для створення мап за допомогою процесу Dead reckoning.

4. Зміст пояснювальної записки:
   1. Аналіз проблеми та предметної області
   2. Специфікація вимог до системи
   3. Проектування системи
   4. Розробка, демонстрація і тестування системи

5. Перелік обов'язкових слайдів презентації:
   1. Постановка задачі
   2. Актуальність теми
   3.Функціональні вимоги – діаграма прецедентів
   4. Блок схема роботи основного застосунку
   5. Методи обрахування кроків і напряму
   6. Архітектура застосунку – діаграма класів
   7. Демонстрація роботи ПЗ

6. Календарний план-графік

№ пор	Завдання	Термін виконання	Відмітка про виконання
1.	Розробка графіка роботи. Оформлення 1 розділу ДП на основі досліджень під час переддипломної практики	25.10.23 – 31.10.23	
2.	Представлення керівнику зробленої роботи	01.11.23 – 01.11.23	
3.	Написання 2 розділу, представлення керівнику	02.11.23 – 15.11.23	
4.	Перший нормо-контроль 1-2 розділів	16.11.23 – 23.11.23	
5.	Написання 3 розділу, представлення керівнику	18.11.23 –27.11.23	
6.	Написання 4 розділу, представлення керівнику	28.11.23– 02.12.23	
7.	Загальне редагування та друк пояснювальної записки, графічного матеріалу	06.12.23 – 09.12.23	
8.	Проходження нормо-контролю, перевірка на антиплагіат, перепліт пояснювальної записки.	10.12.23 – 15.12.23	
9.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації	13.12.23 – 16.12.23	
10.	Отримання відгуку керівника, рецензії.	17.12.23 – 18.12.23	
11.	Підготовка матеріалів для передачі секретарю ДЕК (ПЗ, CD-R з електронними копіями ПЗ, презентації, відгук керівника, рецензія) в папці	19.12.23 – 22.12.23	
12.	Захист дипломної роботи перед ЕК	28.12.23-28.12.23	

7. Дата видачі завдання 02.10.2023

Керівник: к.т.н. с.н.с. доцент Євген КЛЮЄВ

Завдання прийняв до виконання: Денис МИХАЙЛЕНКО

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Мобільний застосунок для створення мап за допомогою процесу Dead reckoning»: 91 с., 20 рис., 4 табл., 23 інформаційних джерел.

СТВОРЕННЯ МАП, DEAD RECKONING, МОБІЛЬНИЙ ЗАСТОСУНОК, ІНЕРЦІЙНА НАВІГАЦІЯ

Предмет дослідження – механізм інерційної навігації за допомогою процесу dead reckoning.

Об’єкт розробки – застосунок, що допомагає автоматизовано створювати мапу і впроваджує навігацію по ній з використанням dead reckoning.

Мета роботи – полегшити створення мап і навігацію по ним в ситуаціях відсутності супутникового та мобільного зв’язку.

Результати дослідження можна використовувати як засіб автоматизованого створення мап підземних та підводних споруд та навігацію по ним, що поліпшить умови праці комунальних служб, військових, рятувальників і т. д. Також частини проекту можна використовувати в інших дослідженнях суміжних областей, які потребують інерційної навігації (аерокосмічна галузь, геодезична галузь і т. п.)

ABSTRACT

Explanatory note for the diploma project "Mobile application for creating maps using the Dead reckoning process": 91 p., 20 figures, 4 table, 23 information sources.

MAP CREATION, DEAD RECKONING, MOBILE APPLICATION, INERTIAL NAVIGATION

The subject of the study is the mechanism of inertial navigation using the dead reckoning process.

The object of development is an application that helps to automatically create a map and implements navigation on it using dead reckoning.

The goal of the work is to facilitate the creation of maps and navigation on them in situations of absence of satellite and mobile communication.

The results of the study can be used as a means of automated creation of maps of underground and underwater structures and navigation on them, which will improve the working conditions of public utilities, military personnel, rescuers, etc. Also, parts of the project can be used in other research in related areas that require inertial navigation (aerospace industry, geodesy, etc.)..

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

SOLID – single responsibility, open–closed, Liskov substitution, interface segregation, dependency inversion

ООП – Об'єкто-орієнтоване програмування

ІПЗ – Інженерія Програмного Забезпечення

ПЗ – Програмне забезпечення

ГІС – геоінформаційні системи

GNSS – Global Navigation Satellite System (Супутникова система навігації)

GPS – Global Positioning System (Система глобального позиціювання)

RFID – Radio-Frequency Identification

PDR – Pedestrian Dead Reckoning

ІСВ – Інерційна система відліку

БД – база даних

ВСТУП

Створення мап є важливою задачею для багатьох сфер діяльності, таких як навігація, картографія, ГІС та інші. Традиційно для створення мап використовується супутникова навігація, яка забезпечує високоточну позицію та орієнтацію. Однак у деяких випадках, наприклад, усередині будівель або в умовах поганої видимості, супутникова навігація не може бути використана.

В таких випадках може бути використаний процес Dead reckoning, який дозволяє визначати поточну позицію та орієнтацію пристрою, використовуючи лише інформацію про його попереднє положення, швидкість та напрямок руху.

Мобільні застосунки для створення мап за допомогою процесу Dead reckoning мають низку переваг перед застосунками, що використовують супутникову навігацію. По-перше, вони не потребують доступу до супутникових сигналів, що робить їх незалежними від умов навколишнього середовища. По-друге, вони можуть бути використані для створення мап у приміщеннях та інших місцях, де супутникова навігація недоступна.

Мета даної роботи - полегшити і вдосконалити процес створення мап для об’єктів, де будь які види навігації крім dead reckoning малозастосовні.

Об’єктом розробки є мобільний застосунок, що дозволяє користувачам створювати мапи за допомогою процесу Dead reckoning.

Предметом розробки є процес створення мап за допомогою процесу Dead reckoning.

Наукова новизна отриманих результатів:

- Уперше створено працюючий мобільний застосунок з використанням dead reckoning з низькою похибкою праці.
- Доповнено існуючі уявлення, щодо застосування dead reckoning в сучасній картографії.

- Доповнена інформація, щодо застосування даного підходу та застосунку для створення мап для наукової діяльності в геодезії та спелеології.

Практичне значення одержаних результатів:

- Можливість застосування програмного коду модуля dead reckoning в інших ПЗ.
- Застосування розробленого ПЗ для створення мап багатьох штучних підземних структур, наприклад метро, дренажні, каналізаційні системи.

Ключові слова: створення мап, dead reckoning, спелеологія, мапи.

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМИ ТА ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Існуючі методи та системи позиціонування. Їх недоліки та переваги

Позиціонування - це процес визначення точного положення об'єкта в просторі або відстеження його руху. Цей процес став надзвичайно важливим у багатьох галузях, включаючи геодезію, навігацію, транспорт, маркетинг та інші.

Супутникова система навігації (GNSS — Global Navigation Satellite System) — комплексна електронно-технічна система, що складається з сукупності наземного та космічного обладнання та призначена для позиціонування в просторі (місцезнаходження в географічній системі координат) і в часі, а також визначення параметрів руху (швидкості, напрямку та ін.) для наземних, водних та повітряних об'єктів[1].

Глобальна система позиціонування (GPS)

GPS[2] - це система супутникового навігаційного позиціонування, розроблена США. Вона використовує супутники, розташовані в навколоземній орбіті, для визначення точного положення об'єкта на земній поверхні.

Переваги:

- Висока точність - GPS може забезпечити точність на рівні кількох метрів.
- Глобальність - GPS працює в будь-якому місці на Землі.
- Широкий спектр застосувань

Недоліки:

- Вплив погодних умов - сильний дощ або сніг може спричинити втрату сигналу GPS.
- Затримка - GPS може мати деяку затримку в оновленні позиції.

- Залежність від супутників - без доступу до супутників GPS система не працює.

Глонасс (Глобальна навігаційна супутникова система)

Це російська система глобального навігаційного позиціонування, аналогічна до GPS. Вона включає супутники, які надають інформацію про місцезнаходження і час у будь-якій точці на Землі.

Переваги:

- Глонасс працює поблизу Росії та у сусідніх регіонах і забезпечує точне позиціонування.
- Ця система може бути корисною для громадянської авіації, армії, сільського господарства та інших сфер.

Недоліки:

- Глонасс не так поширений, як GPS, і тому може бути менше сумісним зі супутниковим обладнанням.
- Є питання щодо надійності і доступності в районах, де сигнали Глонасс можуть бути блоковані або впливати на них інші чинники.

Галілео

Це європейська система глобального навігаційного позиціонування, розроблена Європейським космічним агентством та Європейською комісією. Система базується на супутниках, які забезпечують точні дані про позицію.

Переваги:

- Галілео має великий потенціал для цивільних та комерційних застосувань.
- Висока точність і доступність в різних частинах світу.

Недоліки:

- Реалізація Галілео вимагає великих інвестицій, і система все ще розвивається.
- Сумісність із системами GPS та Глонасс може бути проблемою.

Бейдоу

Бейдоу - це китайська система глобального навігаційного позиціонування, яка складається з групи супутників. Система призначена для використання в різних галузях, включаючи транспорт, комунікації і військові цілі.

Переваги:

- Забезпечує незалежність Китаю в глобальних системах навігації.
- Підтримує високу точність і широкий спектр застосувань.

Недоліки:

- Система ще розвивається і не має такої глобальної покриття, як GPS.

IRNSS/NavIC

Індійська регіональна система навігації (IRNSS), також відома як NavIC, призначена для навігації в регіоні Південної Азії. Вона включає сім геостаціонарних супутників.[3]

Переваги:

- Забезпечує точне позиціонування в регіоні Південної Азії.
- Використовується в авіації, сільському господарстві та інших галузях.

Недоліки:

- Має обмежену географічну покриття.

Інерціальні системи навігації

Використовують акселерометри та гіроскопи для вимірювання руху об'єкта і визначення його позиції на основі власної інерції.

Переваги:

- Незалежність від супутників - ІНС працює навіть без доступу до супутників.
- Висока швидкість оновлення - вимірювання відбуваються в реальному часі.

- Висока точність на коротких відстанях - на коротких відстанях вони можуть бути дуже точними.

Недоліки:

- Накопичення помилок - з часом інерціальні системи можуть накопичувати помилки у вимірюваннях.
- Висока вартість - ІНС зазвичай дорожчі порівняно з іншими системами.
- Потребує калібрування - систему потрібно періодично калібрувати для зменшення помилок.

Локальні системи позиціонування - такі як системи RFID (Radio-Frequency Identification) або Bluetooth, використовують радіочастотний сигнал для визначення положення об'єкта на невеликих відстанях.

Radio-Frequency Identification (RFID)

RFID - це технологія, що використовує радіочастотні сигнали для ідентифікації та відстеження об'єктів. Система включає мікрочіпи (RFID-теги) і читачі (RFID-ридери). Теги зберігають унікальний ідентифікаційний номер, який може бути зчитаний за допомогою RFID-ридера.[4]

Переваги:

- Висока точність ідентифікації та відстеження об'єктів.
- Можливість відстеження в режимі реального часу.
- Застосовується в логістиці, управлінні запасами, безпеці та інших сферах.

Недоліки:

- Обмежена дальність зчитування - зазвичай RFID працює на відстані до кількох метрів.
- Вплив металу та води на сигнали RFID.
- Залежність від інфраструктури (читачів).

Bluetooth Location Services

Bluetooth Location Services використовує технологію Bluetooth для визначення положення пристроїв або об'єктів у короткому діапазоні. Вона може використовуватися для внутрішнього позиціонування в приміщеннях, таких як торгові центри або аеропорти.[5]

Переваги:

- Висока точність в приміщеннях, де GPS може бути недоступним.
- Застосовується для створення інтерактивних мап та навігаційних додатків.
- Зручно для маркетингу та реклами в приміщеннях.

Недоліки:

- Обмежена дальність дії Bluetooth.
- Потребує підтримки від пристроїв та інфраструктури в приміщенні.
- Питання конфіденційності і безпеки даних.

Ultra-Wideband (UWB) Technology

Технологія Ultra-Wideband використовує велику ширину смуги для вимірювання відстаней між пристроями. Вона забезпечує дуже точне визначення місцезнаходження об'єктів в просторі.[6]

Переваги:

- Надзвичайно висока точність позиціонування (до сантиметрів).
- Можливість використання в аугментованій реальності (AR) та інших передових додатках.

Недоліки:

- Обмежена дальність дії UWB пристроїв.
- Не так поширена, як інші технології.

Wi-Fi-based Positioning Systems

Wi-Fi-based Positioning Systems використовують мережі Wi-Fi для визначення місцезнаходження пристроїв. Вони вимірюють сигнали Wi-Fi точок доступу та використовують їх для визначення положення.

Переваги:

- Висока точність в межах місць з встановленими Wi-Fi точками доступу.
- Можливість використовувати існуючу інфраструктуру Wi-Fi.

Недоліки:

- Обмежена дальність дії Wi-Fi сигналів.
- Потребує доступу до мережі Wi-Fi.

Методи навігації з використанням графічних даних

1. Навігація за зорями (Астронавігація)

Навігація за зорями використовує зірки, сонце, місяць і інші небесні об'єкти для визначення місцезнаходження та курсу. Цей метод був використовуваний мореплавцями та астронавтами для визначення свого положення в океані або в космосі.

Переваги:

- Незалежність від інфраструктури або супутників.
- Висока точність, якщо користувач володіє необхідними навичками.

Недоліки:

- Потребує високої експертизи та спеціалізованого обладнання.
- Залежність від погодних умов та видимості небесних об'єктів.

Візуальна навігація в пейзажі

Візуальна навігація в пейзажі використовує зовнішні візуальні орієнтири, такі як гори, водойми, будівлі і природні об'єкти, для визначення місцезнаходження та навігації.

Переваги:

- Доступність орієнтирів у великих містах та природних середовищах.
- Використання прикмет і знаків у пейзажі для орієнтації.

Недоліки:

- Залежність від видимості орієнтирів та зміни в пейзажі.
- Можливість помилок у визначенні місцезнаходження без великого досвіду.

Використання фотограмметрії

Фотограмметрія - це метод визначення місцезнаходження та побудови 3D-моделей на основі аналізу фотографій та зображень. Цей метод використовується у геодезії, картографії та археології.

Переваги:

- Висока точність при створенні мап та моделей.
- Здатність працювати з аерофотозйомкою і супутниковими зображеннями.

Недоліки:

- Вимагає спеціального обладнання для обробки та аналізу фотографій.
- Великі витрати часу на обробку та аналіз.

Таблиця 1.1.

Порівняльний аналіз методів навігації

Характеристики	Супутникова навігація	Інерціальна навігація	Локальні системи позиціонування	Методи навігації з використанням графічних даних
Інфраструктура	Супутники	Не потрібна	Інфраструктура в приміщенні	Не потрібна
Точність	Висока (до декількох метрів)	Висока на коротких відстанях / Середня на дальніх (до декількох десятків метрів)	Висока в приміщеннях (до декількох метрів)	Висока (до декількох сантиметрів)
Швидкість оновлення	Середня	Висока	Середня	Середня
Залежність від погодних умов	Частково залежить	Не залежить	Не залежить	Не залежить
Залежність від видимості	Не залежить	Не залежить	Не залежить	Може залежати
Застосування	Автомобільна навігація, геодезія, авіація, морська навігація	Внутрішнє позиціонування, візуальна навігація, аерофотозйомка	Внутрішнє позиціонування, маркетинг, безпека	Візуалізація, фотограмметрія

1.2. Процес Dead Reckoning

Пристрою не завжди вдається отримати сигнал від супутника, іноді його можуть намірено блокувати або втрачати через перешкоди в міських районах, відсутність супутникового покриття в певній області (наприклад під землею та під водою).

Pedestrian Dead Reckoning (PDR) [7] відноситься до класу завдань інерціальної навігації . Загальним рішенням у мобільних телефонах є використання супутникової навігації (GPS, ГЛОНАСС, Галілео тощо). Проте сигнал GPS також може бути приглушений, і проблема PDR цікава для локалізації користувача в приміщеннях, таких як великі магазини, торговельні центри в містах тощо, де сигнал може бути приглушений.

Усі сучасні смартфони мають внутрішні апаратні засоби на основі мікроелектромеханічних сенсорів (MEMS), які включають стандартний набір інерційних вимірювальних пристроїв (ІВП): акселерометри, гіроскопи, магнітометри та датчик тиску (опційно).

Акселерометри можуть бути використані для виявлення кроків і подальшого обчислення довжини. Проте вони чутливі до швидкості ходьби та нахилу дороги, що має враховуватися при розрахунках. З цієї причини будь-який метод інерційного визначення страждає від накопичувальної похибки, тобто методи PDR накопичують похибку з часом, оскільки місце розташування обчислюється на основі попереднього результату[22].

Процес Dead reckoning можна описати як послідовність таких кроків:

1. Визначення початкової позиції об’єкта (це робиться або вручну або за допомогою інших систем навігації)
2. Вимірювання поточної швидкості та напряму руху об'єкта. Це може бути зроблено за допомогою таких датчиків, як акселерометр, гіроскоп та магнітометр.
3. Обчислення пройденої відстані за допомогою виміряної швидкості та часу.

4. Обчислення поточної позиції об'єкта за допомогою пройденої відстані та початкового положення.
5. Обчислення поточної орієнтації об'єкта за допомогою виміряного напряму руху.

1.3. Обмеження використання Dead Reckoning

Навігація з визначенням положення за допомогою dead reckoning (PDR) - це технологія, яка використовує сенсори смартфона, такі як акселерометр, гіроскоп та магнітометр, для визначення поточної позиції та напрямку руху користувача. PDR є відносно простим і дешевим способом навігації, але він має ряд обмежень, які необхідно враховувати при його використанні.

Перелік обмежень[7]:

- **Точність**. PDR не так точний, як GPS. Його точність зазвичай становить кілька метрів, що може бути достатнім для навігації в міських умовах та на коротких відстанях, але недостатнім для навігації в сільській місцевості або в місцях з великою кількістю перешкод.
- **Вплив навколишнього середовища.** PDR може бути спотворено статичними факторами, такими як нерівність поверхні, вітер або тряска. Це може призвести до того, що система буде відображати неправильну позицію або напрямок руху. Також цей метод навігації може бути спотворений магнітними полями, створюваними навколишніми об'єктами, такими як будівлі, машини та електромагнітні поля.
- **Накопичення помилки**. PDR схильний до накопичення помилки, що означає, що його показання поступово відхиляються від фактичної позиції. Це може бути викликано багатьма факторами, такими як неточність датчиків, шум і похибки програмного забезпечення. Але помилками можна знехтувати на коротких відстанях та у випадку, коли періодично, через певний проміжок

відстані, можна позицію синхронізувати з іншими методами навігації.

- **Положення смартфона в руках**. Якщо розглядати dead reckoning в реалізації в мобільному застосунку, то на жаль, щоб цей метод працював правильно – людина повинна тримати пристрій в руках. Тремтіння рук та неправильне тримання смартфона можуть вплинути на точність показань датчиків.
- **Розташування осей зчитування даних у датчиках смартфона**: Різне положення осей датчиків у різних моделях смартфонів може впливати на точність компаса.

Способи подолання обмежень PDR:

- **Калібрування**: Калібрування датчиків може допомогти покращити точність PDR.
- **Інтеграція з іншими датчиками і системами навігації**: Інтеграція з іншими датчиками і системами навігації, такими як GPS, може допомогти покращити точність і стабільність PDR.
- **Використовування алгоритмів коригування**. Алгоритми коригування можуть допомогти компенсувати дрейф і інші спотворення PDR.

1.4. Актуальність теми

Сучасний світ відомий своєю надзвичайною мобільністю та потребою в постійному доступі до інформації про місцезнаходження, навігації та створення мап. Споживачі та професіонали з різних галузей, такі як туризм, спорт, дослідження, геологія та спелеологія, все більше прагнуть відслідковувати свій рух, створювати маршрути та робити записи про їхню діяльність у важкодоступних або позаземних областях, де сигнал супутникової навігації (наприклад, GPS) може бути обмежений або недоступний. У зв'язку з цим виникає актуальна потреба в розробці

мобільного застосунку, який би дозволяв користувачам створювати мапи та навігуватися в умовах відсутності супутникового сигналу.

Створення ж мап підземних/підводних споруд є важливою задачею для багатьох сфер діяльності, таких як навігація, рятувальні операції, дослідження та ін. Однак, створення мап вручну може бути трудомістким і тривалим процесом.

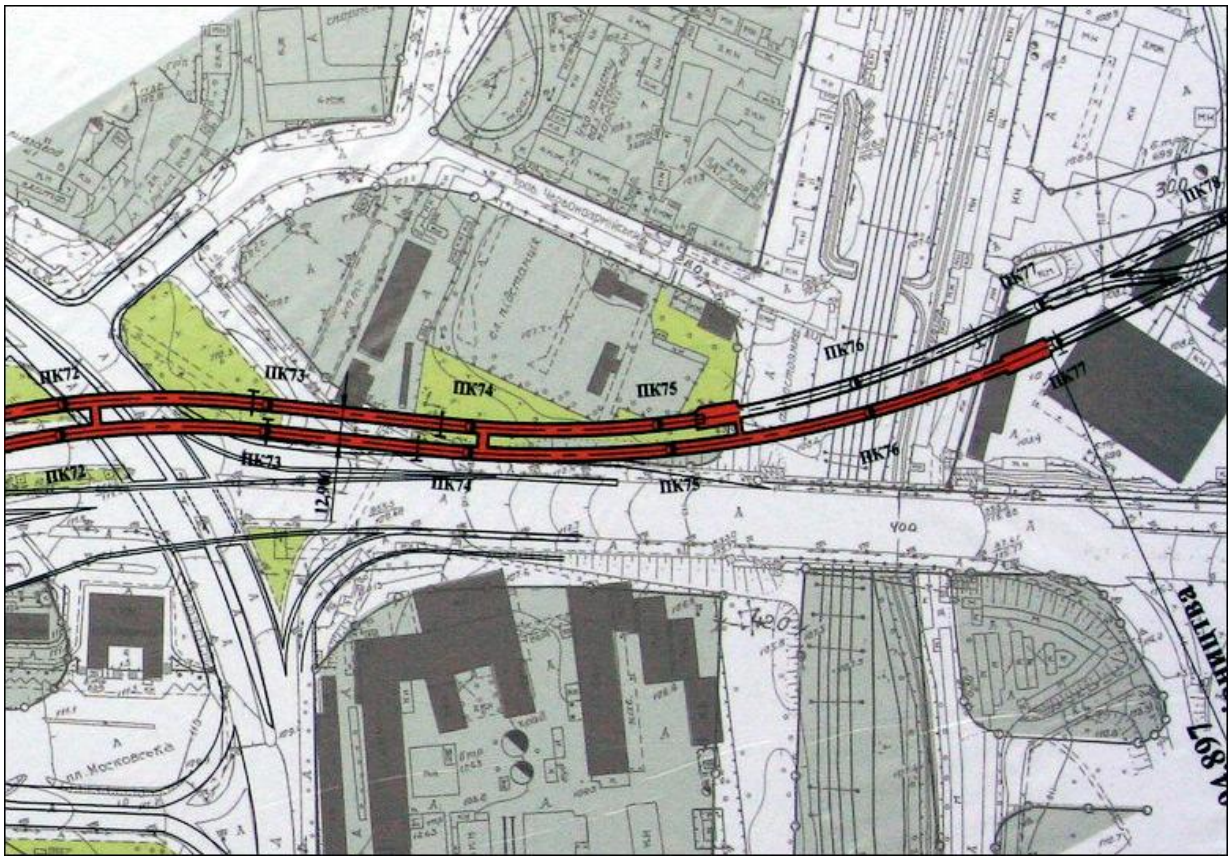

Рис. 1.1. Частина мапи синьої гілки Київського метро

На схемі[9] (див. Рис. 1.1) представлена професійна мапа ділянки метрополітену. Створена професійними картографами з спеціальною нотацією, але на створення було втрачено багато часу, бо створювалася вручну.

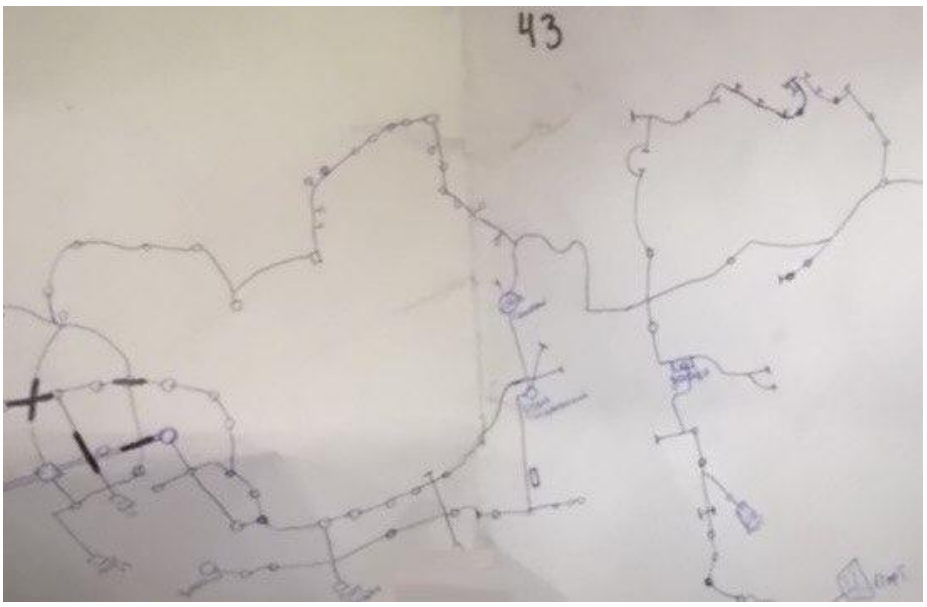

Рис. 1.2. Фрагмент мапи Київської ДШС 16,27,48,43

На мапі (див. Рис. 1.2) представлений приклад мапи дігера-дослідника. Створена на папері ручкою. Не зберігає відстані тунелів та висотність, лише приблизний напрям. На жаль подібні мапи дуже часто зустрічаються навіть в професійному середовищі через складності з створенням детальних та професійних мап.

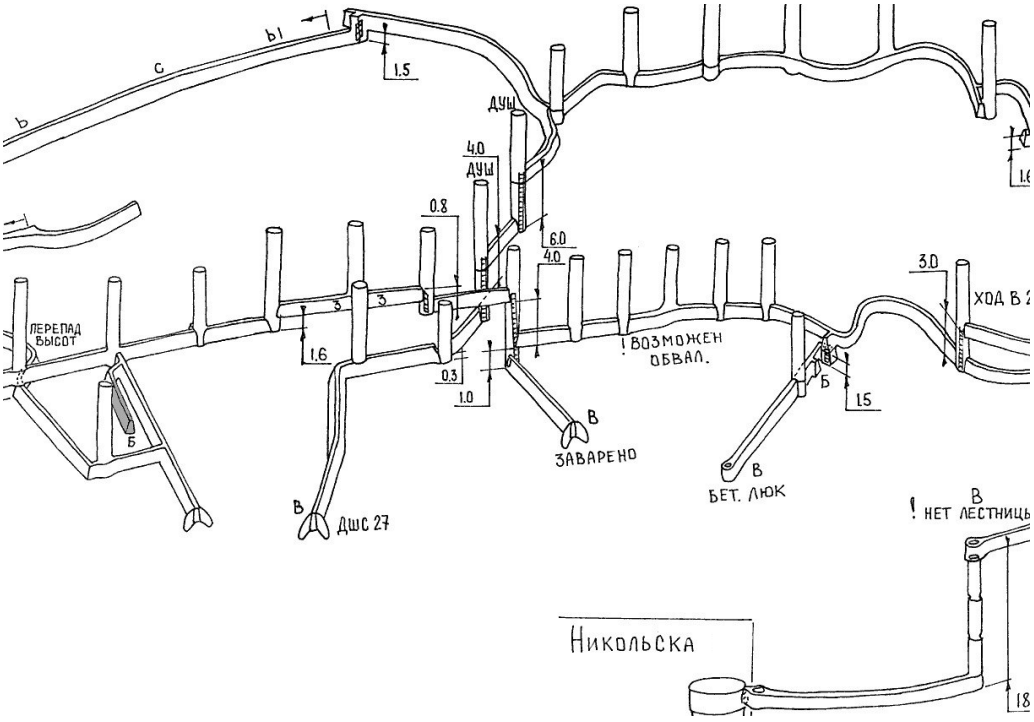

Рис. 1.3. Фрагмент мапи Київської ДШС 27

На мапі (див. Рис. 1.3) представлений приклад мапи спеліолога / дігера-дослідника. Також створена на папері олівцем, але досить професійно з позначками висот, відстаней та 3д ефектом. Однак, на створення такої мапи також потрібно витратити багато часу та зусиль.

Метод "Dead Reckoning" стає важливим рішенням у випадках, коли супутникова навігація стає недоступною. Цей метод базується на використанні датчиків, що вбудовані в смартфони, таких як акселерометри, гіроскопи та магнітометри, для визначення позиції користувача та руху в просторі. Він дозволяє створювати мапи, фіксувати маршрути та відстежувати рух без залежності від зовнішнього супутникового сигналу.

Розроблений мобільний застосунок дозволить зекономити час та зручніше створювати мапи підземних/підводних споруд. Застосунок буде автоматично створювати мапу під час руху користувача по певній структурі. Для цього застосунок буде використовувати датчики пристрою, такі як акселерометр, гіроскоп та магнітометр.[8]

Крім того, застосунок буде мати такі переваги:

- Точність. Застосунок буде використовувати алгоритми, які дозволять зменшити похибку в визначенні поточної позиції та орієнтації пристрою.
- Надійність. Застосунок буде використовувати алгоритми, які дозволять уникнути помилок у створенні мапи.
- Простота використання. Застосунок буде мати простий та зрозумілий інтерфейс, який дозволить користувачам легко створювати мапи.

Застосунок може бути використаний для таких цілей:

- Навігація. Застосунок може бути використаний для навігації в підземних спорудах.
- Дослідження. Застосунок може бути використаний для дослідження природніх печер, катакомб і штучних споруд.

- Рятувальні операції. Застосунок може бути використаний для рятувальних операцій у підземних спорудах.
- Туризм. Туристи та альпіністи зможуть створювати маршрути під час альпіністських походів або в дикій природі.
- Екстремальні види спорту. Любителі екстремальних видів спорту, таких як гірськолижний спорт чи велосипедизм, зможуть створювати маршрути та аналізувати свою активність без супутникового зв'язку

Таким чином, використання системи є ефективним інструментом для наведених вище цілей. Застосунок дозволить зекономити час, підвищити точність та надійність створення мап, а також зробити цей процес більш зручним і автоматичним.

1.5. Аналіз існуючих розробок

Процес Dead Reckoning активно застосовується в навігації та прикладних науках. Частіш за все він програмно реалізований в складних спеціалізованих застосунках, драйверах і т. п., до яких не має доступ публічний користувач.

На тему розробки мобільних застосунків з використанням цієї технології написано безліч статей, але щодо реалізації, зможемо дізнатися – реалізовано мало, а те що є досить недосконале з великою похибкою в роботі.

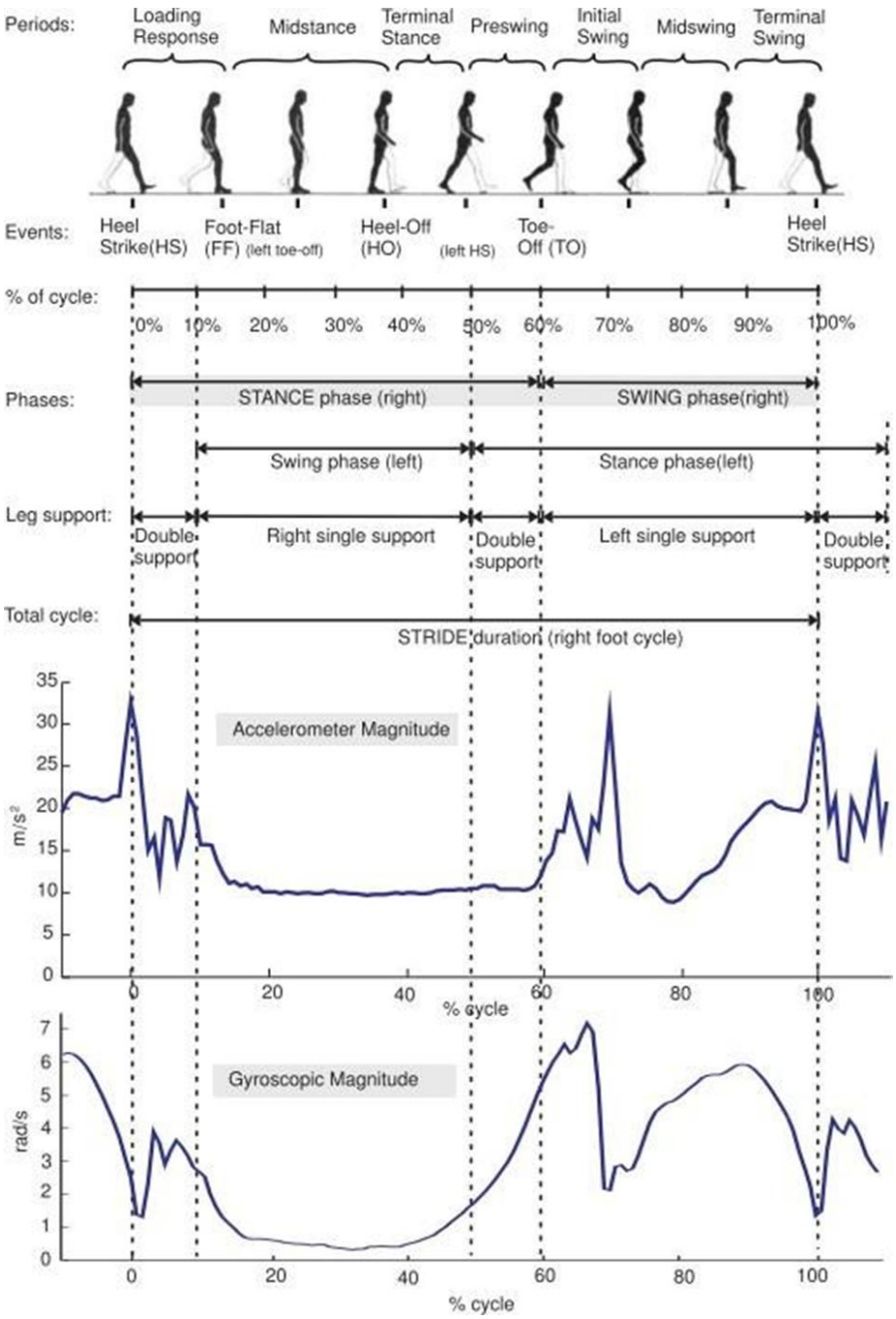

Рис. 1.4. Реєстрація фаз руху датчиками телефону

Крім того, реалізації і підходи дуже сильно відрізняються, як математичним апаратом обробки даних, так і переліком використаних датчиків. Різницю між датчиками гарно зображено[10] вище (див. Рис. 1.4).

DeadReckoning

Багато сучасних зручностей покладаються на нашу здатність точно визначати місцезнаходження. GPS значно полегшує цю роботу, але система не ідеальна. Є багато місць, де сигнал GPS обмежений, від підземних тунелів до внутрішніх будівель і навіть на вулиці, якщо поблизу великих споруд.

Dead Reckoning – це програма для Android, яка пропонує рішення. Використовуючи вбудовані датчики смартфона для відстеження місцезнаходження, додаток не покладається на GPS і робить процес відстеження самодостатнім і офлайн[11].

ОС: Android

Код: відкритий

Застосунок має багато недоліків. По перше не має можливості вибрати та налаштувати датчики. По друге, створена мапа не накладається на супутникові знімки гугл. По третє точність не дуже велика і похибка накопичується дуже швидко.

Рис. 1.5. Скріншот програми Dead reckoning

PDR

Примітивний застосунок, єдиний, який реалізує dead reckoning в iOS. Має дуже багато недоліків: мапи не зберігаються, дуже велика похибка, яку вже видно на 10 метрах маршруту. Мова застосунку китайська і налаштувань майже не містить[12].

ОС: iOS

Код: закритий

Рис. 1.6. Скріншот програми PDR

Strava

Strava — це мобільний застосунок для спортсменів, який дозволяє відстежувати спортивну активність за допомогою GPS. Сервіс заснований у Сан-Франциско, США, і найпопулярнішими видами активності в ньому є біг і їзда на велосипеді[13].

ОС: Android, iOS

Код: закритий

Додаток дозволяє користувачам знаходити маршрути та інших спортсменів у базі даних. Якщо два користувачі рухаються одночасно та в тому ж місці (наприклад, беруть участь в одному забігу), їхні дані будуть автоматично об'єднані. Користувачі можуть висловлювати свою підтримку один одному, коментувати активність.

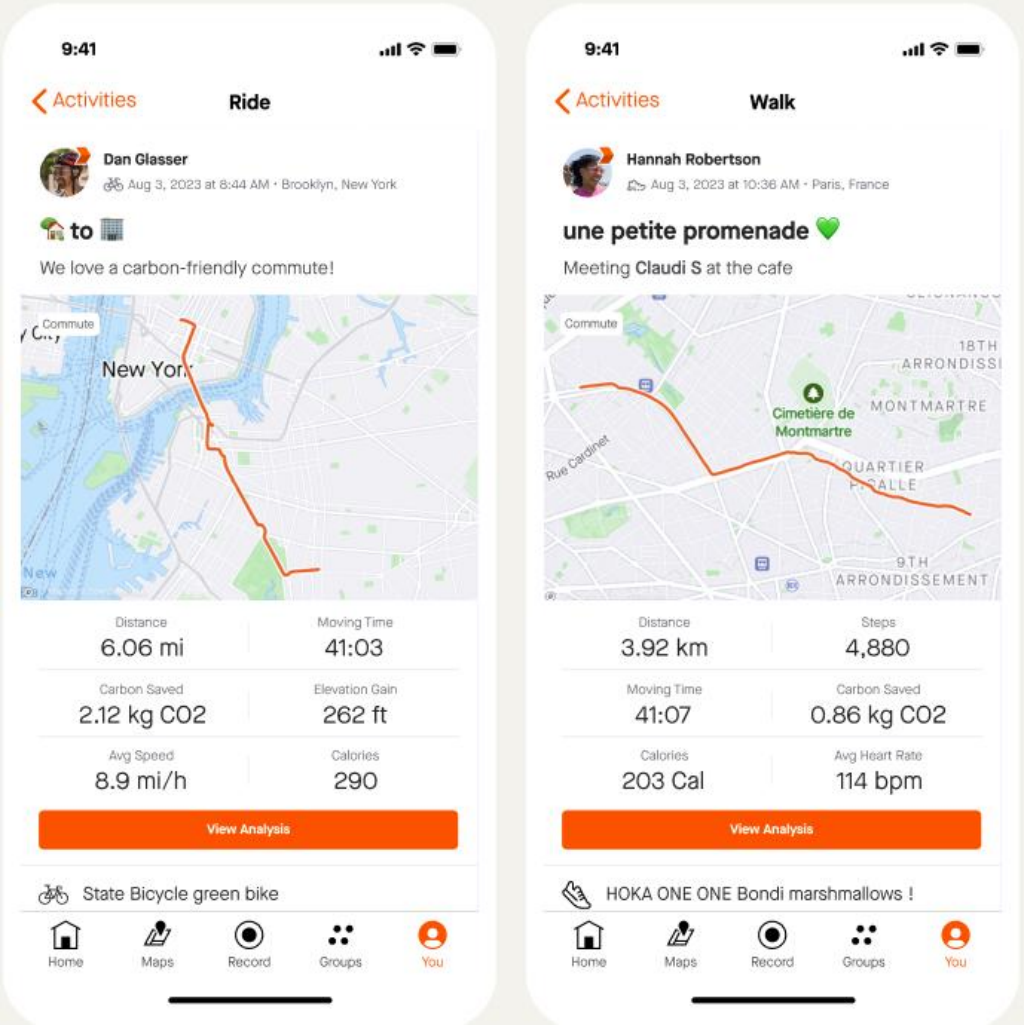

Рис. 1.7. Скріншот програми Strava

Python\Mathlab скрипти

Python\Mathlab скрипти – дуже багато рішень в цій сфері зроблено саме в такому виді як прототипи.

В загальному, ці застосунки надалі залишаються приватними і являються власністю компаній, чи окремих фізичних осіб або ж публічними з відкритим кодом, але, як правило на стадії погано працюючих прототипів.

В такому випадку – python займається пасивним збором даних з датчиків пристрою, а MathLab для візуалізації даних у вигляді графіку або траєкторії, що може бути накладена на мапу[10].

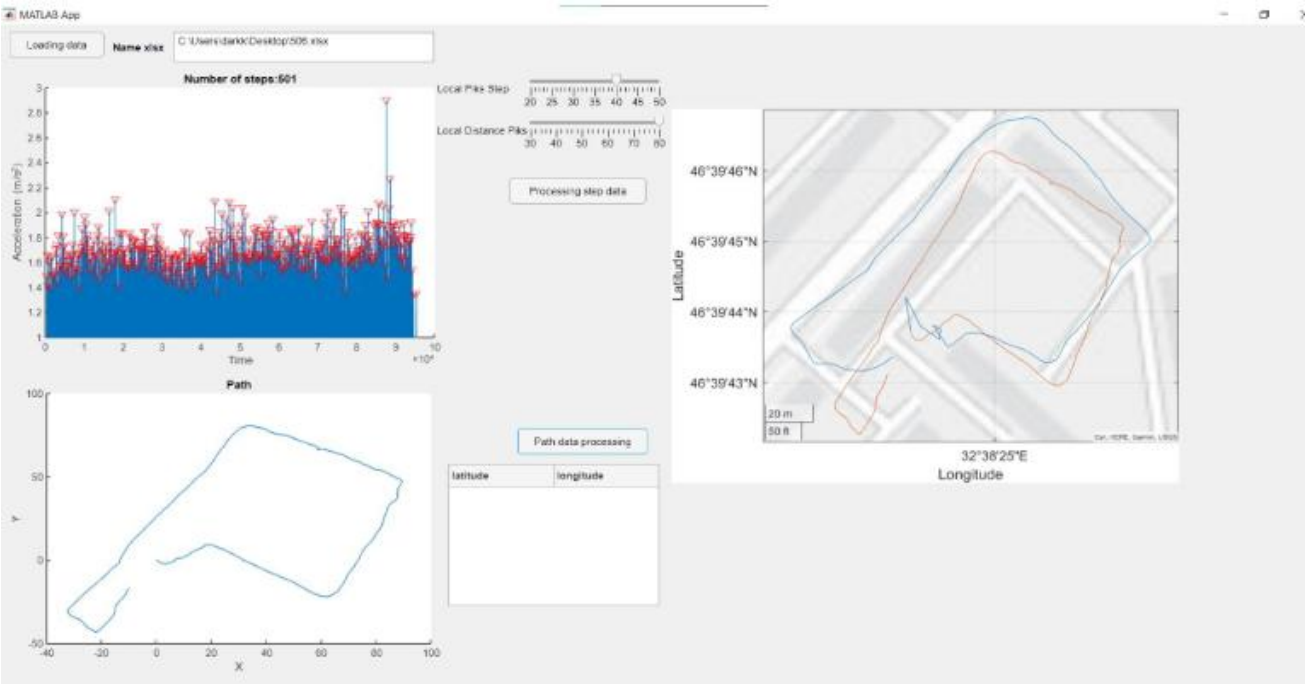

Рис. 1.8. Скріншот вікна MathLab, де координати візуалізовано у вигляді траєкторії

Як можемо бачити, в даній області спостерігається дефіцит зручних засобів для автоматизованого створення мап без використання GPS.

Під час розробки моєї роботи, я намагався не допустити недоліки існуючих засобів і використати новітні методи.

Таблиця 1.2.

Порівняльний аналіз аналогів

Характеристики	DeadReckoning App	PDR	Strava	Python / Mathlab скрипти
Ліцензування	Безкоштовне	Безкоштовне	Безкоштовне	Умовно безкоштовне / платне
OpenSource	+	-	-	+/-
Чи використовується Dead reckoning	+	+	-	+
Чи використовується GPS	+	-	+	-/+
ОС	Android	iOS	Android, iOS	Не обмежено
Точність	Невисока, похибка накопичується швидко	Дуже низька	Середня	Залежить від алгоритму та налаштувань
Можливість збереження мап	+	-	+	+
Можливість накладення мап на супутникові знімки	+	-	+	-/+
Зручний GUI	+	-	+	-
Наявна документація	+	-	+	-
Наявність підказок	-	-	+	-

1.6. Постановка задачі

Мета дипломної роботи – покращення механізму створення мап за допомогою Dead Reckoning, який буде мати високу точність і буде доступний для використання користувачами різних категорій.

Для досягнення мети необхідно вирішити наступні задачі:

- Розробити математичну модель для обчислення пройденої відстані, поточної позиції та поточної орієнтації. Модель повинна бути точна та повинна враховувати всі фактори, які можуть впливати на

точність Dead Reckoning, такі як шум датчиків, нерівності поверхні тощо.

- Розробити алгоритм для обробки даних з датчиків. Алгоритм повинен бути ефективним і повинен забезпечувати високу точність обчислення пройденої відстані, поточної позиції та поточної орієнтації.
- Розробити інтерфейс користувача для вибору початкової позиції та перегляду результатів. Інтерфейс повинен бути зручним та інтуїтивно зрозумілим для користувачів різних категорій.

Висновки

У цьому розділі дипломної роботи були розглянуті такі питання:

1. Огляд існуючих методів позиціонування, включаючи супутникову навігацію, інерціальну навігацію, локальні системи позиціонування та методи навігації з використанням графічних даних.

2. Опис процесу Dead Reckoning та його застосування для створення мап підземних/підводних споруд.

3. Обґрунтування необхідності розробки мобільного застосунку для створення мап за допомогою Dead Reckoning та аналіз існуючих розробок у цій області.

Наразі існує дефіцит зручних засобів для автоматизованого створення мап без використання GPS. Існуючі застосунки мають низку недоліків, таких як низька точність, відсутність можливості збереження мап, відсутність можливості накладення мап на супутникові знімки тощо.

Використання методу Dead Reckoning для створення мап підземних/підводних споруд є перспективним напрямком. Цей метод дозволяє створювати мапи в умовах, де використання GPS неможливо або утруднене.

Розробка мобільного застосунку для створення мап за допомогою Dead Reckoning є актуальним завданням. Такий застосунок може бути

використаний для різних цілей, таких як картографування підземних комунікацій, створення мап підводних затонулих об'єктів, відстеження переміщення людей у складних умовах тощо.

Для досягнення цієї мети необхідно вирішити наступні задачі:

- Проаналізувати існуючи математичні моделі для обчислення пройденої відстані, поточної позиції та поточної орієнтації. На основі цих даних зробити власні реалізації досліджених методів. Реалізації повинні бути точними та повинні враховувати всі фактори, які можуть впливати на точність Dead Reckoning.
- Розробити алгоритм для обробки даних з датчиків. Алгоритм повинен бути ефективним і повинен забезпечувати високу точність обчислення пройденої відстані, поточної позиції та поточної орієнтації.
- Розробити інтерфейс користувача для вибору початкової позиції та перегляду результатів. Інтерфейс повинен бути зручним та інтуїтивно зрозумілим для користувачів різних категорій.

Розв'язання цих задач буде розглянуто в наступних розділах дипломної роботи.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО СИСТЕМИ

2.1. Вибір методології розробки ПЗ

Для розробки дипломного проекту вибрана об’єктно орієнтована методологія.

Ця методологія прийшла на змінну процедурній структурі програмного коду, коли стало очевидно, що класичні методи процедурного програмування не здатні впоратися ні зі зростаючою складністю програм та їх розробки, ні з підвищенням їх надійності.

Ця методологія розглядає програму як множину об'єктів, які взаємодіють між собою. Об'єкт - це сутнісна одиниця програми, яка має свої дані (поля) та методи (функції).

Переваги цієї методології:

- Чіткість і логічність структури програм. Методологія дозволяє розробляти програму, яка відповідає реальним об'єктам та процесам. Це робить програму більш зрозумілою та легкою у супроводі.
- Зменшення повторення коду. Наслідування дозволяє повторно використовувати код, що призводить до зменшення помилок і покращення продуктивності.
- ООП-системи можна легко модернізувати від малих до великих.
- Модульність. ООП дозволяє легко розбивати програму на самостійні модулі, що полегшує її розробку та тестування.

Принцип інкапсуляції допомагає програмісту створювати безпечні програми, в яких не може зробити зміни код з інших частин програми.

Як і всі інші методології ООП має і недоліки:

- Довжина програм, розроблених з використанням мови ООП більша, ніж структурне програмування.

- Ми не можемо застосовувати ООП скрізь, оскільки це не універсальний підхід. Застосовується лише тоді, коли це необхідно.
- Складність. ООП може бути складною для розуміння та застосування, особливо для початківців.
- Навантаження на пам'ять. ООП може вимагати більше пам'яті, ніж інші парадигми програмування.
- Сповільнення виконання. ООП може призвести до деякого сповільнення виконання програми, особливо в разі неефективного використання наслідування.

Зважаючи на переваги і недоліки, ця методологія ідеально підійде для розробки мобільного застосунку для створення мап.

Найголовніше це те, що залишається можливість легкого повторного використовування коду. Функціонал розроблюваного дипломного ПЗ можна буде використати і в інших проектах без особливих труднощів або покращувати ПС як проект з відкритим кодом.

2.2. Функціональні вимоги

Загальні функціональні вимоги розроблюваного ПЗ:

- Застосунок повинен дозволяти користувачам вибирати початкову позицію для створення мапи.
- Застосунок повинен дозволяти користувачам отримувати дані з вибраного типу датчиків пристрою, таких як акселерометр, гіроскоп і магнітометр.
- Застосунок повинен використовувати дані з датчиків для обчислення пройденої відстані, поточної позиції та поточної орієнтації і відображати це в графічному вигляді (траєкторією на мапі).
- Створити механізм збереження/завантаження мап.

- Застосунок повинен дозволяти користувачам налаштовувати параметри обчислення, такі як чутливість датчиків і алгоритм обробки даних.
- Застосунок може дозволяти користувачам ділитися створеними мапами з іншими користувачами.

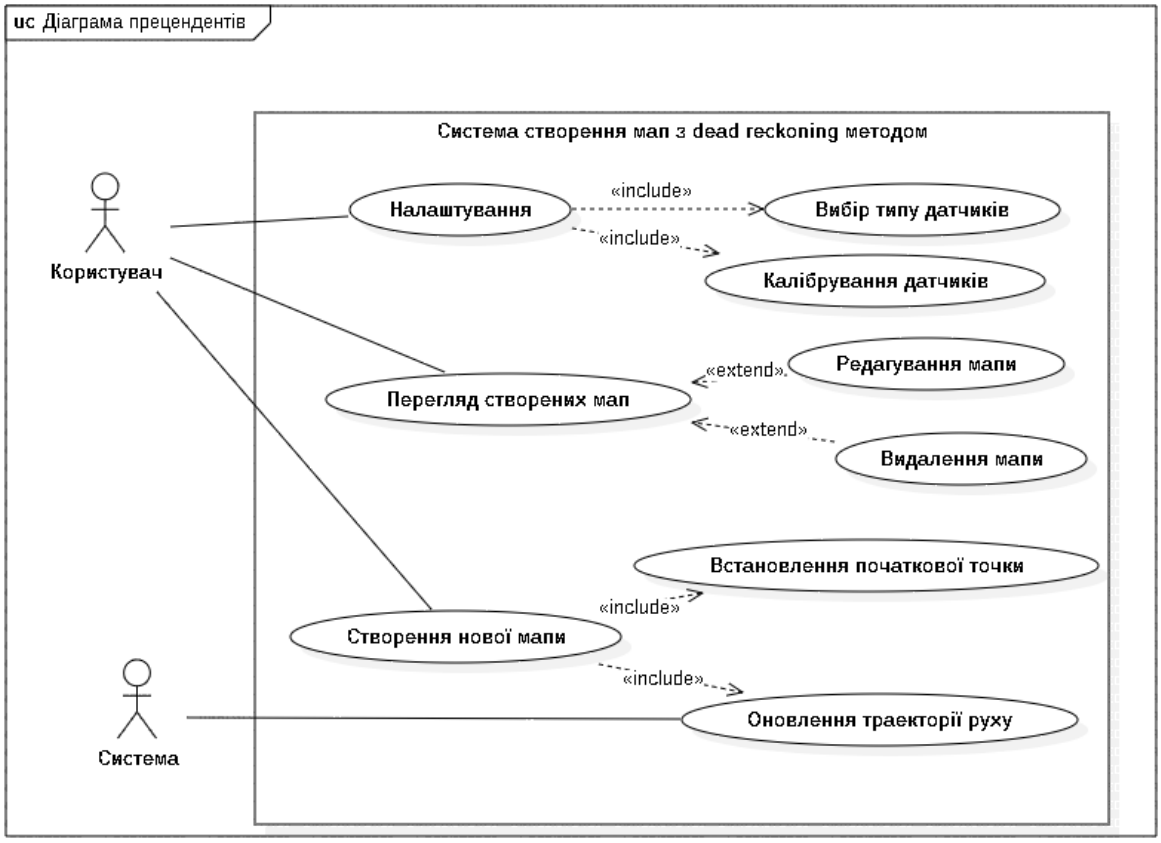

Рис. 2.1. Діаграма прецедентів

Загалом дана система передбачає використання лише одним користувачем.

Розглянемо варіанти користування:

Після запуску застосунка у користувача є меню з діями: переглянути мапи, створити нову мапу або налаштування.

Перегляд мап

Цей прецедент дозволяє користувачеві переглядати існуючі мапи.

Застосунок відображає список існуючих мап. Користувач може вибрати мапу з списку, і застосунок відображає мапу на екрані, а саме траєкторія руху буде накладена на віджет мап.

Крім перегляду мап тут є можливість редагувати та видаляти мапи.

Створення нової мапи

Цей прецедент дозволяє користувачеві створювати нові мапи, вибираючи початкову позицію і отримуючи дані з датчиків пристрою.

Застосунок запитує у користувача початкову позицію для мапи.

Користувач вибирає початкову позицію, і застосунок починає отримувати дані з датчиків пристрою.

Застосунок використовує ці дані для обчислення пройденої відстані, поточної позиції та поточної орієнтації. Застосунок зберігає створену мапу.

Налаштування

Цей прецедент дозволяє користувачеві налаштувати параметри обчислення, такі як чутливість датчиків і алгоритм обробки даних.

Застосунок відображає список параметрів, які можна налаштувати. Користувач може змінити значення параметрів, і застосунок зберігає нові значення.

Щоб відкалібрувати датчики, користувач повинен вибрати калібрування з налаштувань. Застосунок запускає процес калібрування, який може включати в себе ряд дій.

Після завершення процесу калібрування застосунок зберігає нові значення датчиків.

2.3. Нефункціональні та апаратні вимоги

Доступність:

- Застосунок повинен бути доступним в будь яких умовах, незалежно від того, чи є покриття мережею

Надійність:

- Застосунок повинен бути стійким до збоїв і помилок.

- При виняткових ситуаціях помилках програми система повинна мати змогу відновити свій попередній стан виконавши автоматичний перезапуск.
- Під час роботи дані повинні регулярно резервуватися.

Вимоги до часу зберігання даних:

- Дані повинні зберігатися без обмежень по часу.

Масштабованість:

- Вертикальна: передбачити можливість підтримки мап великого розміру з великою кількістю об’єктів.
- Горизонтальна: не передбачена.

Вимоги до зручності використання:

- Застосунок повинен бути легким у використанні та інтуїтивно зрозумілим.
- На освоєння всіх функцій системи у не підготовленого користувача не повинно йти більше декількох годин.
- Документація по використанню повинна бути доступною.
- Необхідно зробити максимальну кількість зрозумілих повідомлень для користувача під час випадкових ситуацій.
- Використання зручного шрифту та налаштуванням для людей з обмеженими можливостями.

Вимоги до безпеки:

- Потрібно передбачити всі виняткові ситуації для впровадження безпеки користуванням застосунку.

Вимоги до конфігурованості системи:

- Застосунок повинен дозволяти налаштувати будь-які властивості функціоналу.

Вимоги до продуктивності системи та обмеження:

- Застосунок не повинен використовувати більше ніж половину наявної ОЗУ пристрою.

Вимоги до клієнту:

- Операційна система: Android
- Об’єм оперативної пам’яті: 1 Gb або більше;
- Наявність 200 Mb вільної дискової пам’яті
- Процесор Intel I5 або краще;

Можливість багаторазового використання:

- Код та компоненти повинні використовуватися повторно там, де це можливо.

Розширюваність:

- Система повинна бути реалізована базово з функціями які в подальшому повинні задовольняти вимоги розширюваності системи.

Портативність:

- Використовувати Java та принципи ООП для забезпечення портативності для інших ОС.

Модульність:

- Обов’язкові модулі: модуль математичних обчислень, модуль завантаження віджету мапи, модуль обробки даних датчиків, модуль інтерфейсу.

Тестування:

- Unit тести.
- Емпіричне тестування GUI та алгоритмів ПЗ

Локалізація:

- Програма повинна бути мові: українська.

2.4. Бізнес вимоги

Описаний застосунок призначений для створення мап у місцях, де немає доступу до мобільного інтернету та GPS.

Підставою для розробки системи є наказ ректора “Про затвердження тем та призначення керівників дипломних робіт” № 1994/ст, затверджений 29.09.2023 р.

Аудиторія проекту складає з себе людей, які потребують створення мап у мобільних умовах, наприклад, туристи, мандрівники, рятувальники, військові тощо.

В перших версіях ніякого прибутку від проекту не планується, а в подальшому можна створити тарифні плани з різними можливостями та обмеженнями.

2.5 Вимоги до інформаційної і програмної сумісності та обґрунтування вибору програмних засобів

Система повинна бути створена на **Java**.

За рейтингом PYPL (PopularitY of Programming Language)[14] Java посідає 2 місце. Це свідчить про те, що ця мова досить популярна на світовому рівні.

Java має ряд переваг, які роблять її хорошим вибором для розробки мобільних застосунків, зокрема:

Java є кросплатформеною платформою, що означає, що її можна використовувати для розробки застосунків для різних мобільних пристроїв, таких як смартфони та планшети. Це дозволяє розробникам створювати застосунки, які будуть доступні для широкого кола користувачів.

Java є безпечною мовою програмування, яка має вбудовані механізми для захисту даних користувачів. Це важливо для мобільних застосунків, які часто містять конфіденційну інформацію, наприклад, особисті дані користувачів.

Java є економічною мовою програмування, яка не вимагає дорогих інструментів та ресурсів. Це важливо для невеликих команд розробників, які працюють над обмеженим бюджетом.

Ось кілька конкретних причин, чому Java варто застосувати для розробки мобільного застосунку для створення мап за допомогою процесу Dead reckoning:

Java + Android SDK має вбудовані функції для роботи з географічними даними. Це дозволяє розробникам легко створювати застосунки, які можуть відстежувати поточну геопозицію користувача та створювати мапи на основі цих даних.

Java має велику спільноту розробників і велику кількість бібліотек та фреймворків, які можна використовувати для розробки мобільних застосунків. Це дає розробникам доступ до широкого спектру ресурсів, які можуть допомогти їм у розробці застосунку.

Звичайно, існують й інші мови програмування, які можна використовувати для розробки мобільних застосунків. Однак Java є хорошим вибором для цього проекту, оскільки вона має ряд переваг, які роблять її придатною для цієї мети.

Для внутрішньої БД буде використана **SQLite.**

SQLite - це відкрита, автономна, компактна та високопродуктивна база даних, яка часто використовується в мобільних застосунках. Вона має ряд переваг, які роблять її ідеальним вибором для цього типу застосунків.

SQLite є внутрішньою базою даних, що означає, що вона зберігається в одному файлі. Це робить її ідеальним вибором для мобільних застосунків, які повинні зберігати дані локально.

Переваги:

Компактність – SQLite є дуже компактною базою даних, що робить її ідеальним вибором для мобільних пристроїв з обмеженим обсягом пам'яті.

Автономність – SQLite є автономною базою даних, що означає, що вона не залежить від будь-якого іншого програмного забезпечення або сервера. Це робить її ідеальним вибором для мобільних застосунків, які повинні працювати автономно.

Висока продуктивність – SQLite є високопродуктивною базою даних, що робить її ідеальним вибором для мобільних застосунків, які вимагають швидкого доступу до даних.

Легкість використання – SQLite є відносно легкою в використанні базою даних, що робить її ідеальним вибором для розробників мобільних застосунків.

Для віджету мап в Android застосунках є багато різних альтернатив.

Найпопулярніші це Google Maps та Open Street Map.

В даній роботі використано **Google Maps**. Чому я обираю роботу з віджетом Google Maps, а не OpenStreetMaps?

Якщо я розробляю застосунок, який потребує точних даних про розташування, широкого охоплення території та широкого набору функцій, я оберу роботу з віджетом Google Maps. Google Maps пропонує всі ці переваги, що робить його ідеальним вибором для таких застосунків, як: застосунки для навігації, застосунки для відстеження фізичної активності, застосунки для бізнесу, які потребують точних даних про розташування.

Переваги Google Maps у порівнянні з OpenStreetMaps:

Точність роботи – Google Maps має більш точні дані про розташування, ніж OpenStreetMaps. Це завдяки тому, що Google Maps використовує дані з супутників, авіаліній та інших джерел.

Дані про об’єкти – Google Maps охоплює більшу кількість об’єктів на мапах, ніж OpenStreetMaps. Це означає, що ви з більшою ймовірністю знайдете потрібну інформацію в Google Maps.

Функціональність – Google Maps пропонує більше функцій API, ніж OpenStreetMaps. Це включає в себе встроєні функції, як навігація, пошук, відстеження руху в реальному часі тощо. Але, звичайно, можно за допомогою поліморфізму впроваджувати свої алгоритми також.

Інтеграція – Google Maps легко інтегрувати з іншими продуктами Google, такими як Google Workspace та Google Cloud Platform. Це може бути корисно для розробників, які хочуть створити комплексні рішення.

Для розробки буду використовувати **Intellij IDEA** та систему контролю версій **Git.**

Висновки

У даному розділі була проведена специфікація вимог.

Вибір об'єктно-орієнтованої методології для розробки застосунку є обґрунтованим, оскільки вона дозволяє розробляти програму, яка відповідає реальним об'єктам та процесам. Це робить програму більш зрозумілою та легкою у супроводі.

Визначені функціональні вимоги дозволяють розробити застосунок, який буде виконувати всі необхідні функції для створення мап у місцях, де немає доступу до мобільного інтернету та GPS.

Визначені нефункціональні вимоги забезпечують надійність, доступність, зручність використання, безпеку та інші важливі характеристики застосунку.

Визначені бізнес-вимоги забезпечують відповідність застосунку потребам цільової аудиторії.

Вибір мови програмування Java є обґрунтованим, оскільки вона має ряд переваг, які роблять її придатною для розробки мобільних застосунків.

Вибір середовища розробки Intellij IDEA та системи контролю версій Git також є доречним, оскільки вони дозволяють розробникам ефективно створювати та підтримувати застосунок.

Також обгрунтовано вибір системи мап (Google maps) та внутрішньої БД (SQLLite).

РОЗДІЛ 3
ПРОЕКТУВАННЯ СИСТЕМИ

3.1. Загальна математична модель Dead reckoning

Математична модель PDR побудована на нелінійному фільтрі частинок.

$$S_k = [x^T V^T q^T \varepsilon_a^T \varepsilon_\omega^T]_k^T \quad (3.1)$$

Рух користувача має вектор стану S_k, процес рівняння стану описано в такій нелінійній формі, що оцінює параметри, що входять до вектора стану.

Де x - вектор координат, V - вектор швидкості, q - кватерніон повороту,

ε - вектор зміщення відліків акселерометра та гіроскопа.

Ідея[10] полягає в тому, що при розгляді рівномірного руху в ньому можна виявити основну гармонійну характеристику координат усіх точок людського тіла, які мають кореляцію з кроками. Додаткова інформація для покращення точності систем обмежень, що накладаються на рух частин тіла, як приклад - утримання вертикального положення.

Зазвичай перехід між фазами ходьби супроводжується відповідною зміною прискорення. Таким чином більшість методів виявлення і підрахунку кроків засновані на обробці величини вектору прискорення.

Для точної оцінки координат людини можна використовувати апроксимацію закону їх зміни. Ця апроксимація складається з двох складових

- Середнього значення за період спостереження, яке називають «трендом».
- Гармонічного коливання з періодом, рівним тривалості одного або двох кроків.

Для горизонтальної координати тренд є лінійною функцією, оскільки швидкість ходьби приймається за постійну.

Для вертикальної координати тренд є константою, оскільки ходьба здійснюється на горизонтальній площині. Період гармонічних складових вважається постійним.

Апроксимація дозволяє точно оцінити координати пішохода, навіть якщо вони змінюються нерівномірно. Гармонічне коливання з періодом, рівним тривалості кроку, відображає характерну для ходьби періодичність. Константа періоду гармонічних складових дозволяє враховувати індивідуальні особливості людини, такі як довжина кроку та швидкість ходьби.

3.2. Системи координат

В даній роботі використовуються 2 системи координат.

Інерційна система відліку

Інерційна система відліку — це система відліку, в якій тіло, на яке не діють жодні сили, рухається рівномірно й прямолінійно, або це система відліку, в якій прискорення тіла зумовлене тільки дією на нього сил. Існування інерційних систем відліку постулюється в сучасному формулюванні законів Ньютонах[15].

В інерційній системі відліку тіло, на яке не діють жодні сили, рухається рівномірно й прямолінійно. В інерційній системі відліку прискорення тіла зумовлене тільки дією на нього сил.

Для спрощення розрахунків у навігації часто використовують Землю як ІСВ, хоча вона і не є ідеальною. При цьому застосовують спеціальну систему координат, прив'язану до центра Землі та екваторіальної площини. Однак, обертання Землі не враховується у цій системі.

- Центр Землі обирається як точка O – початок координат відповідно до прийнятої моделі.
- Вісь z співпадає з віссю обертання Землі.
- Осі x та y знаходяться в екваторіальній площині.

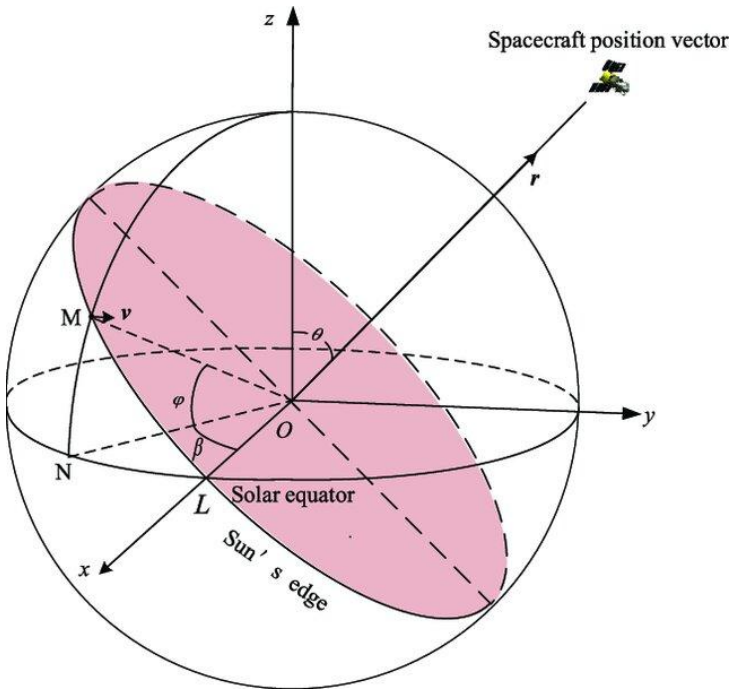

Рис. 3.1. Геліоцентрична інерціальна прямокутна система координат

Географічні координати

Географічні координати - це система координат, яка використовується для визначення положення точок на поверхні Землі. Ця система складається з двох координат: широти і довготи[16].

Широта - це кут між поверхнею Землі та площиною екватора. Широта вимірюється в градусах від 0° до 90°. 0° широти відповідає екватору, 90° північної широти відповідає Північному полюсу, а 90° південної широти відповідає Південному полюсу.

Довгота - це кут між початковим меридіаном і меридіаном, що проходить через точку, яку потрібно визначити. Довгота вимірюється в градусах від 0° до 180°. 0° довготи відповідає Гринвіцькому меридіану, 180° східної довготи відповідає 180° західної довготи.

При визначенні географічних координат Земля приймається за сферу, а не за еліпсоїд обертання. Географічні координати визначають положення точки на поверхні Землі або, ширше, в географічній оболонці. Географічні координати базуються на сферичному принципі. Подібні координати використовуються для інших планет, а також на небесній сфері.

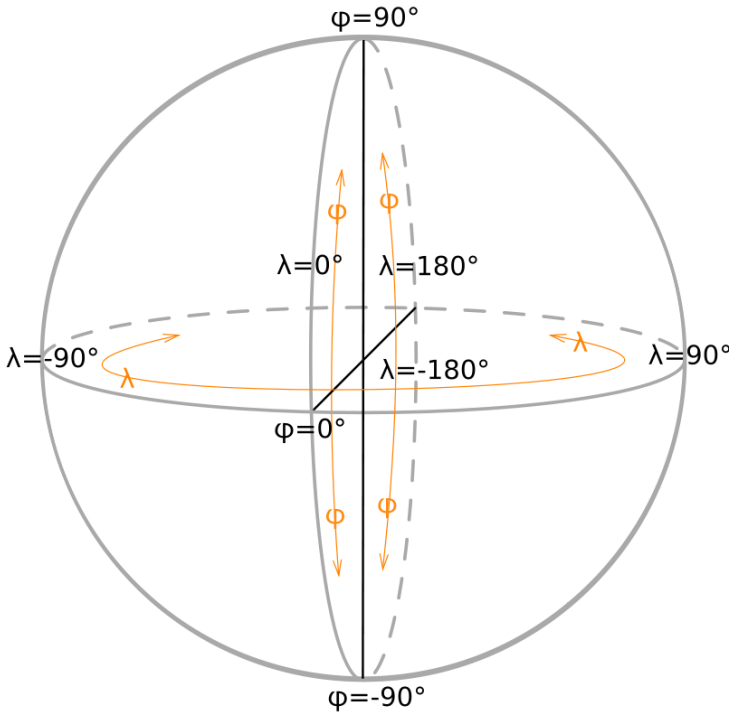

Рис. 3.2. Географічна сфера з позначками географічної системи координат

Між географічною і інерційною системами є взаємозв'язок, тому можна проводити конвертацію із однієї системи в іншу.

В навігації за початок координатної системи обирається центр мас об'єкта. Перехід початку координат з інерційної системи координат до географічної на основі значень широти та довготи. Координати центру географічної системи координат O_g в інерційних значеннях набувають значень :

$$\begin{aligned} X_{og} &= (R+h)\cos(\varphi)\sin(Ut+\lambda) \\ Y_{og} &= (R+h)\cos(\varphi)\sin(Ut+\lambda) \\ Z_{og} &= (R+h)\sin(\varphi) \end{aligned} \qquad (3.2)$$

де:

R — радіус Землі;

U — кутова швидкість обертання Землі;

h — висота над рівнем моря;

φ — широта;

λ — довгота;

t — час.

Географічна система координат має значний недолік для навігації на високих широтах – її кутова швидкість стає дуже великою, а на полюсі вона й зовсім стає нескінченною. Тому в таких випадках використовують напівавтономну систему координат в азимуті. Ця система зручніша для розрахунків, а для виведення інформації координати перетворюються в географічну.

Приведення координат між цими системами відбувається за формулами:

$$N = Y_w \cos(\varepsilon) + X_w \sin(\varepsilon)$$
$$E = -Y_w \sin(\varepsilon) + X_w \cos(\varepsilon) \qquad (3.3)$$

де:

N – вісь, що вказує на північ, координата в географічній системі

E – вісь, спрямована на схід, координата в географічній системі

X_w – координата x в описаній системі координат

Y_w – координата y в описаній системі координат

ε – кут нахилу між системами координат

3.3. Вибір і опис методів підрахування кроків

Існує безліч математичного-програмних методів для підрахунку кроків. В цьому розділі описані декілька відомих, які можна застосувати з датчиками, які містить телефон.

Кожен метод має свою спрямованість, недоліки, переваги і тому вони описані в розділі проектування. Бо обмежуватися одним методом і датчиком не дуже доречно через те, що в різних умовах та на різних пристроях ефективність методів може відрізнятися.

3.3.1. Метод підрахунку кроків на основі порогового виявлення

Метод підрахунку кроків на основі порогового виявлення є одним з найпростіших і найпоширеніших методів. Він використовує датчики руху, такі як акселерометри, для виявлення коливань, які відповідають крокам[17].

Метод працює так:

- Датчики руху збирають дані про прискорення пристрою.
- Ці дані обробляються за допомогою порогового детектора.
- Якщо прискорення пристрою перевищує певний пороговий рівень, то це вважається кроком.

Пороговий рівень зазвичай встановлюється експериментально. Він повинен бути достатньо високим, щоб не реєструвати помилкові кроки, але не занадто високим, щоб не пропускати справжні кроки.

Найбільша складність в цьому методі пов’язана з пороговим рівнем.

В різних ситуаціях бажано мати різні порогові рівні. Наприклад, порогове значення залежить від того, чи хтось піднімається, спускається, біжить чи просто йде з різною швидкістю.

Також телефон може бути в різних положеннях: у руках, у кишені штанів або шортів або прикріплений до поясу. Це дає багато зайвих даних і мало інформації. Оскільки виробники використовують різні версії акселерометрів, то якщо програмне забезпечення працює на одному телефоні, немає гарантії, що воно працюватиме на іншому пристрої, навіть тієї ж моделі.

Якщо тестувати цей метод кількома користувачами – пороги, налаштовані індивідуально будуть сильно відрізняються між пристроями, тому частота вимірювань також матиме великий вплив. При використанні постійної частоти, яка виявляє ходьбу однієї людини нормального темпу, немає гарантії, що ходьба іншої людини буде виявлена з використанням цих же налаштувань з цим же приладом.

Переваги методу:

- Він простий у реалізації і не вимагає складного обладнання.

- Він може бути реалізований на широкому спектрі пристроїв, включаючи смартфони, планшети та смарт-годинники.
- Висока швидкість роботи.

Недоліки методу:

- Він може бути не дуже точним, особливо при ходьбі по нерівній поверхні.
- Він може реєструвати помилкові кроки, наприклад, при трясці пристрою.
- Береться багато даних, які постійно оновлюються – що використовує ресурси пристрою.

3.3.2. Метод підрахунку кроків на основі динамічного порогового виявлення

Метод підрахунку кроків на основі динамічного порогового виявлення є модифікацією методу порогового виявлення, який використовує змінний поріг для виявлення кроків. Цей метод покращує точність підрахунку кроків, особливо при ходьбі по нерівній поверхні або при трясці пристрою[18].

Метод працює так:

- Датчики руху збирають дані про прискорення пристрою.
- Відбувається зглажування даних
- Дані обробляються за допомогою динамічного порогового детектора.
- Якщо прискорення пристрою перевищує поріг, то це вважається кроком.

Динамічний поріг змінюється в залежності від умов ходьби. Цей метод більш точний, ніж статичний поріг, але він також більш складний у реалізації.

Існує кілька методів для встановлення динамічного порога. Один із методів полягає в використанні фільтра середнього значення. Цей фільтр

використовує середнє значення прискорення пристрою за певний період часу. Поріг встановлюється на рівні середнього значення прискорення, плюс деякий допуск.

Інший метод для встановлення динамічного порога полягає в використанні фільтра на основі рівня шуму. Цей фільтр використовує дані про шум, який присутній у даних акселерометра. Поріг встановлюється на рівні шуму, плюс деякий допуск.

Сигнал не завжди виглядає гладким, іноді має додаткові коливання та шум. Це пов'язано з неточністю збору даних, а також оцифруванням сигналу. Згладжування сигналу допоможе вирішити цю проблему.

Гладкий сигнал: це простий спосіб усереднити сусідні значення, щоб видалити частину шуму. Наприклад, можна замінити кожен зразок середнім значенням поточного зразка, попереднього зразка та зразка після нього. Таким чином, видаляються надмірно зашумлені ділянки сигналу.

Щоб побачити різницю в ефекті, необхідно використовувати різні кроки згладжування. Зі збільшенням кроку згладжування сигнал виглядатиме чистішим і приємнішим візуально, але слід бути обережним при використанні занадто великого вікна, оскільки це може згладити кроки, які потрібно спочатку виявити.

Переваги методу:

- Він покращує точність підрахунку кроків, особливо при ходьбі по нерівній поверхні або при трясці пристрою.
- Він може бути реалізований на широкому спектрі пристроїв, включаючи смартфони, планшети та смарт-годинники.
- Можливість змінювати тем ходьбі.
- Більш універсальний через визначення порогу динамічним шляхом.

Недоліки методу:

- Він може бути більш складним у реалізації, ніж статичний поріг.
- Потрібне згладжування, можлива втрата даних сигналу.

- Використовує більше часу для роботи.

3.3.3. Метод підрахунку кроків на основі виявлення піків

Метод підрахунку кроків на основі виявлення піків - це метод, який використовує датчики руху для виявлення кроків. Цей метод працює, визначаючи точки, в яких сигнал прискорення досягає свого максимального значення. Ці точки і будуть рахуватися за кроки[17].

Метод працює так:

- Датчики руху збирають дані про прискорення пристрою.
- Відбувається фільтрування даних.
- Знаходяться точки, в яких сигнал прискорення досягає свого максимального значення.
- Кожна точка піку вважається кроком.

Підхід виявлення піків шукає періоди, притаманні циклічній природі ходьби, використовуючи значення прискорення або кутові швидкості. Для цього можна використовувати різні методи, такі як:

- Пошук локальних максимумів: цей метод шукає точки, в яких сигнал прискорення досягає свого максимального значення протягом певного періоду часу.
- Пошук динамічних максимумів: цей метод шукає точки, в яких сигнал прискорення досягає свого максимального значення протягом певного періоду часу, з урахуванням динаміки руху.
- Пошук узагальнених максимумів: цей метод шукає точки, в яких сигнал прискорення досягає свого максимального значення протягом певного періоду часу, з урахуванням динаміки руху та шуму.

Вибір методу виявлення піків залежить від конкретних умов використання. Наприклад, метод пошуку локальних максимумів може бути достатнім для використання в умовах, де шум не є великою проблемою. Однак у складніших умовах, наприклад, при ходьбі по нерівній поверхні або

при трясці пристрою, може бути необхідний більш складний метод, такий як метод пошуку динамічних максимумів або метод пошуку узагальнених максимумів.

Підхід виявлення піків оцінює кроки на основі кількості піків, заданих отриманою послідовністю даних, і не покладається на заздалегідь визначені пороги, але страждає від піків перешкод через навколишній шум і випадкові перешкоди.

Щоб позбутися шуму і перешкод з навколишнього середовища, необхідно використовувати фільтр нижніх частот для видалення перешкод.

А також можна обмежити проміжки часу між двома піками відповідно до моделі людської поведінки: необхідно подивитися на проміжок часу між будь-якими двома кроками. Порівнюючи, наскільки далеко піки знаходяться один від одного, вважається, що при подоланні проміжку це вже два або більше піків, які підлягають подальшій обробці, але якщо в часовому інтервалі є кілька піків, то виділяється головний, критерієм виділення головного є модуль вектора прискорення, тобто величина піку.

Завдяки використанню методу виявлення піків, отримується більш точна інформація про крок людини. На відміну від методу нульового перетину, де як межа, так і сам діапазон розташовані поблизу областей з підвищеним шумом. Метод виявлення піків уникає більшості проблем інших методів.

Переваги методу:

- Можливіть фіксувати напрям в 3х осях
- Відносно гарна якість роботи
- Згладжування не призводить до втрати даних
- Більш точний, ніж інші наведені методи

Недоліки методу:

- Необхідна фільтрація шумів і помилкових піків.
- Потрібна додаткова обробка даних.

- Швидкість роботи повільна.

3.3.4. Метод підрахунку кроків на основі нульового перетину

Метод нульового перетину[7] - це метод підрахунку кроків, який визначає кількість нульових точок в даних, які надіслані від датчиків. Ці дані схильні до спотворення через різні причини (в тому числі й через недосконаліть самих датчиків), тому зазвичай потрібна попередня фільтрація та антизгладжування оригінальних даних[19].

Є одна критична проблема при використанні методу нульового перетину.

Як згадувалося раніше, необроблені дані прискорення містять шум, навіть після використання техніки шумозаглушення. Такий шум може перетнути нуль у регістрі сигналу, навіть коли користувач перебуває в стані спокою.

Граничний нульовий перехід використовується для вирішення цієї проблеми. Граничний нульовий перехід можна розглядати як механізм фільтрації/обмеження, який допомагає системі враховувати лише ті нульові переходи, які є результатом фактичного руху тіла.

Це досягається шляхом використання штучних границь по обидва боки нульової точки, що називається діапазоном. Нульовий перехід враховується лише в тому випадку, якщо сигнал прийшов ззовні цього діапазону, інакше він ігнорується.

Підхід нульового переходу шукає періоди, притаманні циклічній природі ходьби, використовуючи значення прискорення або кутові швидкості, і може забезпечувати кращі характеристики, наприклад, з вертикальними прискореннями, але погіршуються, якщо смартфон нещільно прикріплений до людського тіла.

Цей метод найкраще використовувати в поєднанні з іншими методами, щоб підвищити їх точність. Незалежне його використання призводить до великої похибки у вимірюванні.

Переваги методу:

- Можливість зміни темпу
- Може застосуватися в комбінації з іншими методами
- Можливість зміни частоти запису даних

Недоліки методу:

- Потрібне згладжування, можлива втрата даних сигналу
- Статична межа виявлення
- Потрібна додаткова обробка даних
- Метод сам по собі, без використання в парі з іншими методами працює погано

3.3.5. Метод визначення напрямку на основі вертикальної складової кутової швидкості

Метод визначення напрямку на основі вертикальної складової кутової швидкості - це метод, який використовує дані про кутову швидкість, щоб визначити напрямок руху пристрою. Цей метод працює, вимірюючи величину вертикальної складової кутової швидкості, на основі даних гіроскопа. Коли пристрій повертає, вертикальна складова кутової швидкості буде змінюватися[20].

Позначаючи вертикальну складову кутової швидкості через ω_v, її обчислення в можна виконати:

$$\omega_v(t) = \hat{\gamma}(t)^T \omega(t) \qquad (3.4)$$

Де $\omega(t)$, тривимірний вектор кутових швидкостей має бути отриманий після відповідної фільтрації вихідних вимірювань гіроскопа в порядку зменшити вплив випадкових шумів.

$\hat{\gamma}(t)^T$ - одиничний вектор, що вказує на напрямок сили тяжіння в момент часу t;

На основі правила правої руки ω_v вимірює швидкість, з якою людина повертає в горизонтальній площині, де позитивні швидкість вказує на поворот праворуч, а негативна – ліворуч.

Якщо взяти інтеграл, то можна розраховувати зміну курсу.

$$\psi(t) - \psi_s = \int_{t_s}^{t_f} \omega_v(\tau) d\tau \qquad (3.5)$$

Де $[t_f; t_s]$ – інтервал запису даних х датчиків

$\psi(t) - \psi_s$ – кут курсу в радіанах у часі відносно його початкового значення (а саме зміна курсу)

Переваги методу:

- Простота в реалізації
- Швидко обраховує (найшвидший із методів для визначення напрямку)
- Можливість зміни частоти запису даних

Недоліки методу:

- Потрібна додаткова обробка даних
- Потрібна гарна фіксація пристрою(телефону) щоб уникнути зайвих коливань.
- Не дуже велика точність і ймовірність похибки
- Накопичувальна похибка

3.3.6. Метод визначення напрямку на основі розрахунку даних з магнітометра

Магнітометр - це 3-осьовий датчик, який вимірює зовнішнє магнітне поле, що є комбінацією геомагнітного поля Землі та локальних магнітних збурень (наприклад, від феромагнітних матеріалів).

Магнітометр характеризується жорсткими та м'якими залізними спотвореннями.

Жорстке залізне спотворення - це адитивний ефект, коли до виміряного поля додається постійна складова. Це може бути пов'язано, наприклад, з дією постійного магніту або власним нульовим зміщенням датчика.

М'яке залізне спотворення - це мультиплікативний ефект, який відображає зміну напрямку та/або ослаблення вектора магнітної індукції. Цей ефект може бути викликаний наявністю металевого предмета в безпосередній близькості від магнітометра або внутрішніми спотвореннями датчика - помилкою коефіцієнта масштабу.

Для визначення кута напрямку, зібраних даних про силу магнітного поля Землі в 3-х осях, можна розрахувати напрямок за допомогою квадрантного арктангенса з даних з датчика, перетворивши силу в кут в радіанах, що відповідає координатам напрямку ходьби людини. Потім перетворити на градусну міру[21].

Переваги методу:

- Простий у реалізації і не вимагає складного обладнання.
- Може вимірювати напрям в 3х напрямках
- Швидко працюючий метод
- Не має накопичувальної похибки
- Значна точність роботи

Недоліки методу:

- Може бути неточним при наявності металевих предметів поблизу.
- Може бути неточним в разі сильних магнітних збурень.

3.3.7. Метод визначення напрямку на основі модуля кутової швидкості з урахуванням знака

Цей метод визначення напрямку використовує модуль кутової швидкості, тобто її абсолютне значення, а також знак кутової швидкості.

Якщо не враховувати помилки вимірювання, величина векторів не залежить від систем координат, у яких вони вимірюються, тому ωm не

залежить від орієнтації датчика. Однак на практиці систематичні помилки, такі як шум, призводять до того, що вимірюване значення ωm матиме деяку залежність від орієнтації[20].

Модуль кутової швидкості можна визначити за формулою:

$$\omega_m(t) = \sqrt{\omega_x^2(t) + \omega_y^2(t) + \omega_z^2(t)} \qquad (3.6)$$

Де:

$\omega_x(t)$, $\omega_y(t)$, $\omega_z(t)$- компоненти кутової швидкості в напрямках x, y, z

Щоб розрізнити повороти ліворуч і праворуч, необхідно визначити знак $\omega_m(t)$ для кожного моменту часу. Це можна зробити, визначивши, яка вісь (x, y або z) акселерометра є найбільш домінантною, і використовуючи знак гіроскопа на цій осі для визначення напрямку обертання. Цей метод може бути ненадійним у ситуаціях, коли дві різні осі є однаково домінуючими (наприклад, коли смартфон нахилений на 45°, сила тяжіння проявляється однаково як на осі x та на осі z). Інший підхід допоможе уникнути цього, щоб відрізнити поворот ліворуч і праворуч, спостерігаючи за знаком $\omega_v(t)$.

$$\omega_{sm}(t) = sgn(\omega_v(t)) * \omega_m(t) \qquad (3.7)$$

Де:

$\omega_{sm}(t)$ – модуль кутової швидкості зі знаком в момент часу t.

$sgn(\omega_v(t))$ – знак кутової швидкості в момент часу t.

$\omega_m(t)$ – величина кутової швидкості в момент часу t.

Переваги методу:

- Може вимірювати напрям в 3х напрямках

Недоліки методу:

- Необхідна фільтрація.
- Накоплення помилки.

3.3.8. Порівняльний аналіз описаних методів

Таблиця 3.1.

Порівняльний аналіз методів обрахування кроків

Характеристики	Метод порогового виявлення	Метод динамічного порогового виявлення	Метод виявлення піків	Метод нульового перетину
Точність	Низька	Середня	Висока	Середня
Швидкість роботи	Висока	Висока	Низька	Середня
Складність реалізації	Проста	Середня	Висока	Проста
Залежність від положення пристрою	Висока	Висока	Середня	Середня
Можливість застосування в комбінації з іншими методами	-	-	+	+

Таблиця 3.2.

Порівняльний аналіз методів визначення напрямку

Характеристики	Метод визначення напрямку на основі вертикальної складової кутової швидкості	Метод визначення напрямку на основі розрахунку даних з магнітометра	Метод визначення напрямку на основі модуля кутової швидкості з урахуванням знака
Точність	Низька	Висока	Низька
Швидкість роботи	Висока	Низька	Швидка
Складність реалізації	Проста	Середня	Середня
Залежність від наявності металевих предметів поблизу	Висока	Низька	Низька
Залежність від наявності сильних магнітних збурень	Висока	Низька	Низька
Можливість визначення напрямку в 3-х осях	-	+	+

На основі аналізу існуючих методів було виявлено, що універсальних підходів немає. Кожен містить переваги і недоліки. В цій роботі було обрано реалізацію методів:

1) **Метод порогового виявлення**

   Метод порогового виявлення для підрахунку кроків. Цей метод є простим і швидким, що важливо для застосунків, які повинні відображати карту в реальному часі.

2) **Метод динамічного порогового виявлення**

   Можна використовувати для підрахунку кроків, щоб визначити відстань, яку пройшов користувач. Ця відстань використовується для масштабування карти або для відстеження прогресу користувача. Працює повільніше попереднього, але більш точно.

3) **Метод визначення напрямку на основі вертикальної складової кутової швидкості**

   Цей метод також є простим і швидким, що також важливо для застосунків, які повинні відображати карту в реальному часі.

4) **Метод визначення напрямку на основі розрахунку даних з магнітометра**

   Можна використовувати для визначення напрямку з великою точністю, в якому рухається користувач. Ця інформація можна використовувати для повороту карти відповідно до напрямку руху користувача.

В даній сфері – чим більше методів і налаштувань доступно користувачу – тим краще, бо світ мобільних пристроїв наповнений різним апаратним обладнанням всередині телефонів. І універсальні рішення під Android поки що залишаються лише в теоріях.

3.4. Загальна блок схема роботи застосунку

В попередніх розділах були оглянуті методи фіксації кроків, визначення напрямку, математична модель процесу Dead reckoning. Було розроблено ТЗ та проведений детальний аналіз аналогів.

За цими даними розроблена схема (див. Рис. 3.3) того, як повинна працювати головна функція застосунку – створення мапи за допомогою інерціальних методів.

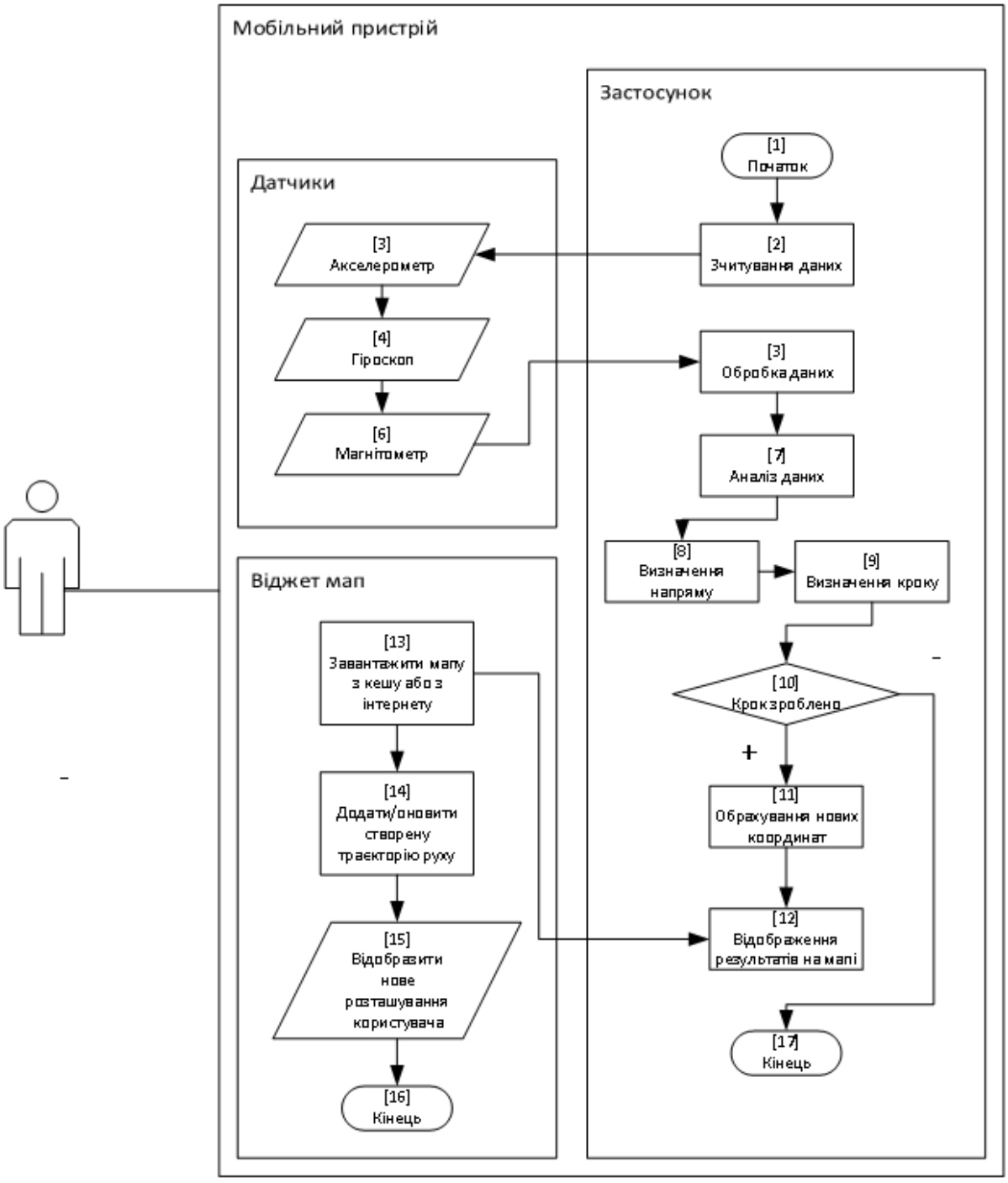

Рис. 3.3. Блок схема ітерації головного процесу застосунку

На блок схемі (див. Рис. 3.3) зображена ітерація процесу створення мапи в режимі реального часу за допомогою мобільного застосунку, який використовує користувач.

Цей застосунок зчитує дані з акселерометра, гіроскопа, магнітометра.

Займається попередньою обробкою даних (фільтрація, видалення зайвих шумів і т. п.).

Потім проводить аналіз даних – за методами, які були описані в цьому розділі.

На основі проведеного аналізу – визначається напрямок руху та чи зробив користувач крок.

Якщо крок був зроблений – обчислюються нові координати позиції користувача і відбувається відображення змін на мапі візуально.

Використовується сторонній сервіс для мап. Сам вигляд мапи залежить від того, чи встиг застосунок завантажити/закешувати потрібний фрагмент мапи.

3.5. Проектування архітектури

В даному проекті використано об’єктно-орієнтовану методологію з багаторівневою архітектурою.

Трирівнева архітектура - це тип архітектури програмного забезпечення, яка розділяє програму на три рівні:

Рівень представлення - відповідає за інтерфейс користувача. Він відповідає за те, як користувач взаємодіє з програмою. В даному рівні буде графічний інтерфейс з контролерами-обробникам подій.

Рівень бізнес логіки - відповідає за бізнес-логіку програми. Він відповідає за те, як програма виконує свої завдання. На ньому знаходиться вся логіка і алгоритми застосунку.

Рівень даних - відповідає за зберігання даних програми. Він відповідає за те, де і як зберігаються дані програми. В нашому випадку на цьому рівні буде відбуватися запис до файлової системи.

Трирівнева архітектура має ряд переваг, зокрема:

Гнучкість - рівні можуть розроблятися і підтримуватися незалежно один від одного. Це дозволяє легко додавати нові функції або змінювати існуючі.

Надійність - якщо один рівень виходить з ладу, інші рівні можуть продовжувати працювати.

Безпека - рівні можуть бути оточені своїми власними межами безпеки. Це допомагає захистити дані від несанкціонованого доступу.

3.5.1. Пакетний рівень

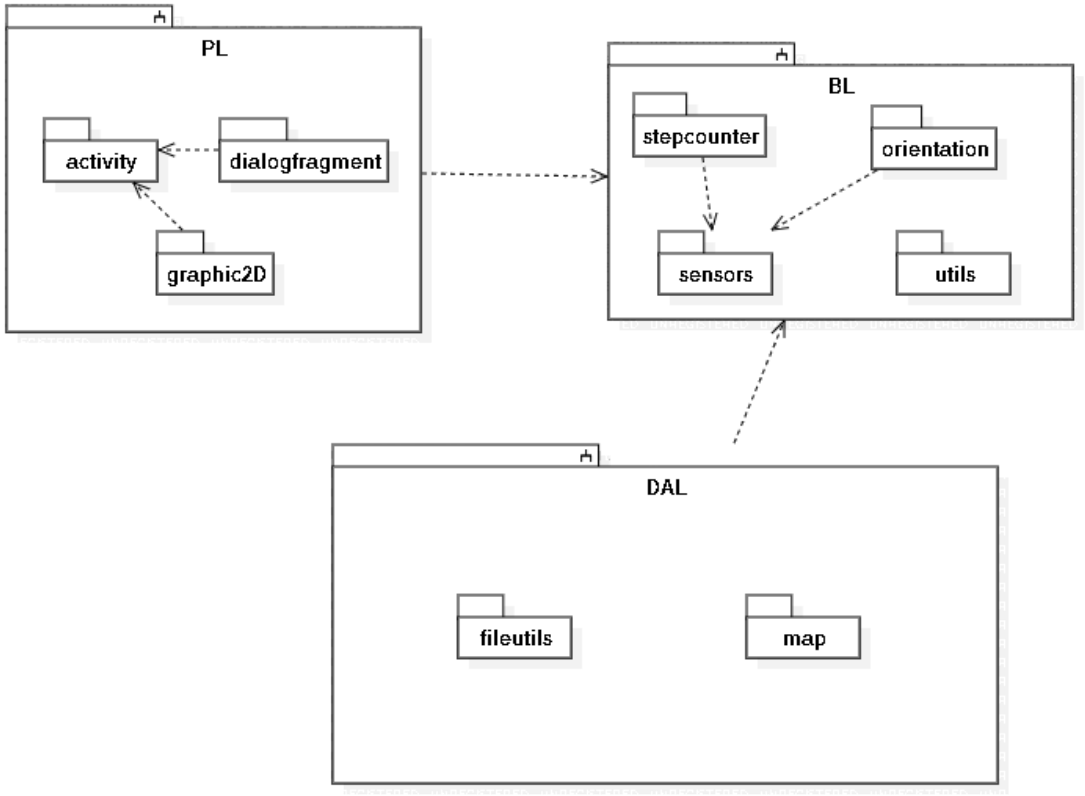

Рис. 3.4. UML діаграма пакетів

Згідно з діаграмою пакетів (див. Рис. 3.4) маємо

Діаграма пакетів показує структуру застосунку, що складається з трьох модулів:

- **PL (Presentation Layer)** - відповідає за відображення інформації на екрані пристрою.
- **BLL (Business Logic Layer)** - відповідає за реалізацію бізнес-логіки застосунку.
- **DAL (Data Access Layer)** - відповідає за доступ до даних.

PL містить три пакети:

- **activity** - містить класи, які відповідають за створення та управління активностями застосунку.
- **dialogfragment** - містить класи, які відповідають за створення та управління діалоговими фрагментами застосунку.
- **graphic2d** - містить класи, які відповідають за відображення графічних об'єктів на екрані пристрою. В даному випадку потрібен для малювання траєкторії руху.

BLL містить чотири пакети:

- **stepcounter** - містить класи, які відповідають за підрахунок кроків.
- **sensor** - містить класи, які відповідають за взаємодію з датчиками.
- **orientation** - містить класи, які відповідають за визначення напрямку.
- **utils** - містить загальніфункціональні класи, які використовуються різними пакетами.

DAL містить два пакети:

- **fileutils** - містить класи, які відповідають за роботу з файлами.
- **database** – містить класи для роботи з базою даних
- **map** - містить класи, які відповідають за дані мапи.

Взаємодія між пакетами:

PL взаємодіє з BLL для отримання доступу до бізнес-логіки застосунку.

BLL взаємодіє з DAL для доступу до даних.

3.5.2. Рівень компонентів

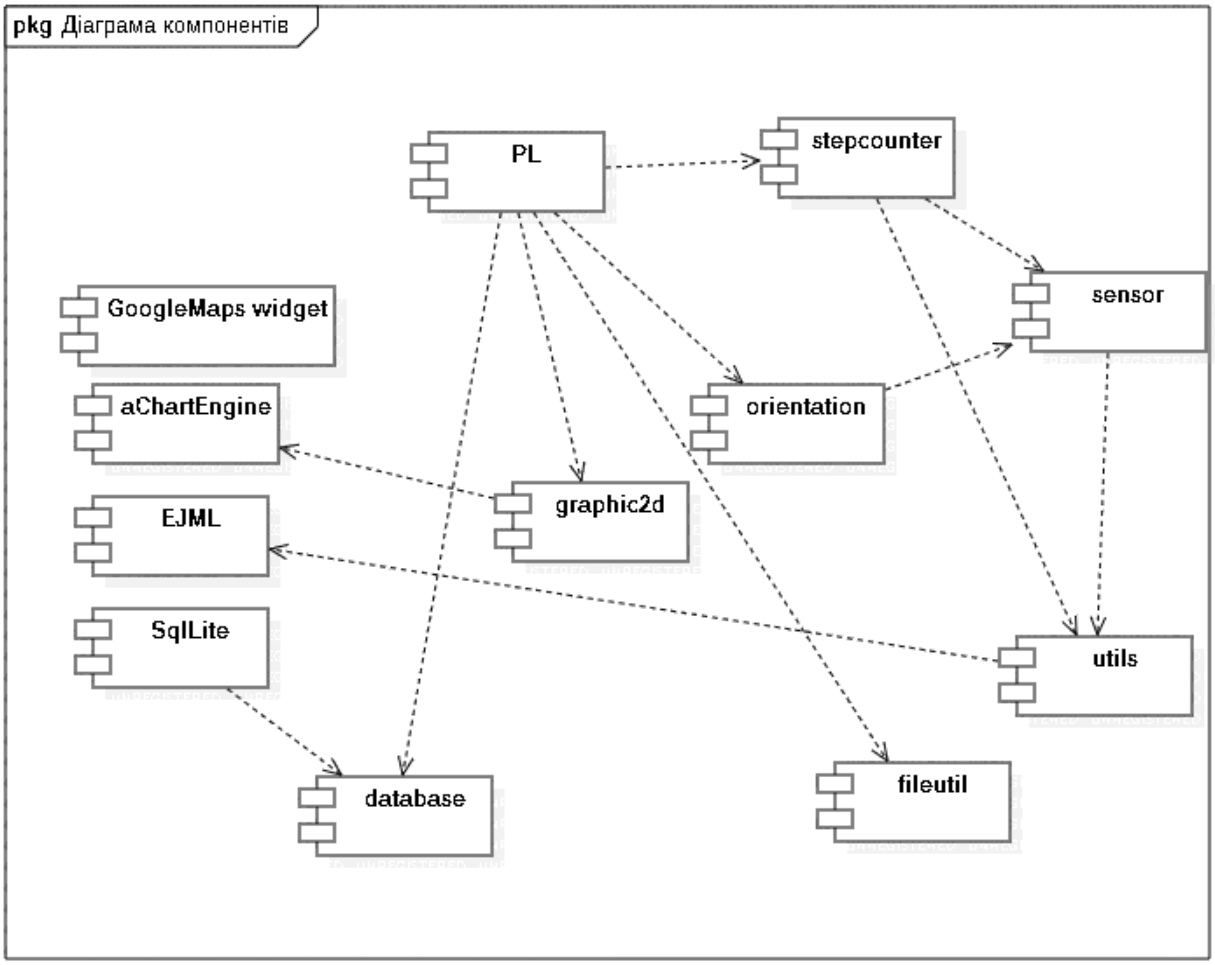

Рис. 3.5. UML діаграма компонентів

Згідно діаграми (див. Рис. 3.5) система міститиме наступні компоненти:

PL – GUI користувача і обробники подій графічного інтерфейсу

Graphic2D - відповідає за відображення графічних об'єктів на екрані пристрою.

StepCounter - відповідає за підрахунок кроків.

Sensor - відповідає за взаємодію з датчиками.

Orientation - відповідає за визначення напрямку.

Utils - містить загальні функціональні класи, які використовуються різними компонентами.

FileUtil - відповідає за роботу з файлами.

Database - відповідає за роботу з базою даних.

Сторонні компоненти:

Google Maps Widget - використовує сторонню бібліотеку Google Maps API для відображення карти.

aChartEngine - використовується для відображення графіків.

EJML - використовується для обробки матриць.

SQLite - вбудована SQLite база даних.

3.5.3. Рівень класів

В даному підрозділі було спроектовано діаграму класів застосунку. До неї не були включені всі існуючі класи, поля, методи. Лише ті, які виконують провідну роль в процесах виконання застосунку.

Опис діаграми(див. Рис. 3.6):

Головний клас DeadReckoningApp

Головний клас логіки застосунку.

- orientationStrategy: OrientationStrategy – поле методу визначення напрямку за допомогою датчиків
- stepCountStrategy: StepCountStrategy – поле методу підрахування кроків за допомогою датчиків
- getInstance() : DeadReckoningApp – дозволяє створити або отримати єдиний екземпляр об’єкту цього класу.
- getDatabaseHelper() : DatabaseHelper – отримати об’єкт для роботи з БД
- getFileHelper() : FileHelper – отримати об’єкт для роботи з файлами(мапами)
- getTrajectoryPlot() : TrajectoryPlot – отримати полотно для малювання траекторії руху

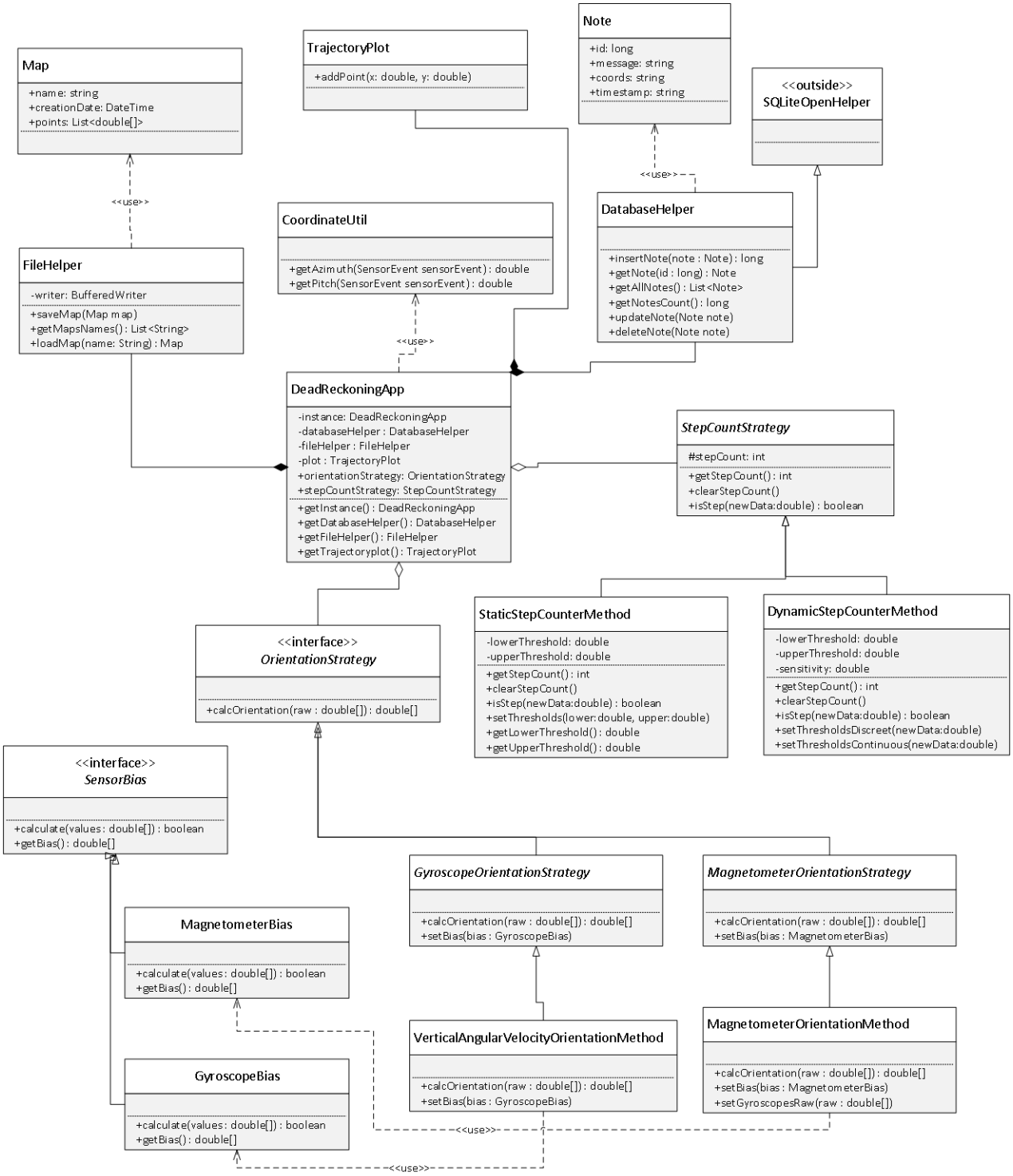

Рис. 3.6. UML діаграма класів

Клас Note

Клас запису даних до БД. Використовується для запису налаштувань або метаданих

- id: long – унікальний ідентифікатор запису

- message: string – значення запису
- type: int – тип запису
- timestamp: string – дата останньої зміни запису

Клас DatabaseHelper

Клас для роботи з локальною базою даною SQLLite. Дозовляє записувати і зчитувати об’єкти Note з БД.

- insertNote(note : Note) : long – додати запис до БД
- getNote(id : long) : Note – отримати запис по його ідентифікатору
- getAllNotes() : List<Note> - отримати список всіх записів
- getNotesCount() : long – отримати кількість записів в БД
- updateNote(Note note) – оновити дані певного запису в БД
- deleteNote(Note note) – видалити запис з БД

Статичний клас CoordinateUtil

Клас, який відповідає за роботою з системами координат.

- getAzimuth(SensorEvent sensorEvent) : double – отримати азимут
- getPitch(SensorEvent sensorEvent) : double – отримати відстань

Клас TrajectoryPlot

Клас для роботи з полотном для малювання траєкторії руху поверх гугл мапи.

- addPoint(x: double, y: double) – додати певну точку до траєкторії руху

Клас Map

Клас даних метадат мапи

- name: string – назва мапи
- creationDate: DateTime – дати створення

- points: List<double[]> – список всіх записаних точок траєкторії

Клас FileHelper

Клас для запису / считування мап з файлової системи

- saveMap(Map map) – зберегти мапу до файлу. Якщо файл з такою назвою існує – то він просто перепишеться.
- getMapsNames() : List<String> – отримати список назв мап, які збережені.
- loadMap(name: String) : Map – завантажити мапу з файлової системи.

Інтерфейс SensorBias

Абстракція для класів, що зберігають і обраховують дані калібрування датчиків

- calculate(values : double[]) : boolean – обчислює зміщення датчика на основі даних, що містять значення трьох осей датчику (x, y, z). Метод повертає значення True, якщо обрахування завершено, інакше False.
- getBias() : double[] - повертає розраховане зміщення датчка(яке перед цим було обчислено за допомогою calculate) як масив з трьох значень (x, y, z).

Клас MagnetometerBias

Клас, що зберігає і обраховує дані калібрування датчика магнітометру

- calculate(values : double[]) : boolean – обчислює зміщення датчика на основі даних, що містять значення трьох осей датчику (x, y, z). Метод повертає значення True, якщо обрахування завершено, інакше False.

- getBias() : double[] - повертає розраховане зміщення датчка(яке перед цим було обчислено за допомогою calculate) як масив з трьох значень (x, y, z).

Клас GyroscopeBias

Клас, що зберігає і обраховує дані калібрування датчика гіроскопу

- calculate(values : double[]) : boolean – обчислює зміщення датчика на основі даних, що містять значення трьох осей датчику (x, y, z). Метод повертає значення True, якщо обрахування завершено, інакше False.
- getBias() : double[] - повертає розраховане зміщення датчка(яке перед цим було обчислено за допомогою calculate) як масив з трьох значень (x, y, z).

Абстрактний клас StepCountStrategy

Абстракція для алгоритмів підрахунку кроків на основі даних датчиків.

- getStepCount() : int – загальна кількість кроків, що вже підрахована
- clearStepCount() – видалити дані про кількість підрахованих кроків
- isStep(newData:double) : boolean – дізнатися, чи нові дані, отримані з датчиків є кроком чи ні

Клас StaticStepCounterMethod

Реалізацію алгоритму підрахунку кроків методом на основі статичного порогового виявлення даних датчиків.

- getStepCount() : int – загальна кількість кроків, що вже підрахована
- clearStepCount() – видалити дані про кількість підрахованих кроків
- isStep(newData:double) : boolean – дізнатися, чи нові дані, отримані з датчиків є кроком чи ні за допмогою методу порогового виявлення

- setThresholds(lower:double, upper:double) – встановити нижній та верхній пороги виявлення кроку
- getLowerThreshold() : double – отримати нижній встановлений поріг виявлення кроку
- getUpperThreshold() : double – отримати верхній встановлений поріг виявлення кроку

Клас DynamicStepCounterMethod

Реалізацію алгоритму підрахунку кроків методом на основі динамічного порогового виявлення даних датчиків.

- getStepCount() : int – загальна кількість кроків, що вже підрахована
- clearStepCount() – видалити дані про кількість підрахованих кроків
- isStep(newData:double) : boolean – дізнатися, чи нові дані, отримані з датчиків є кроком чи ні за допмогою методу порогового виявлення
- setThresholds(newData:double) - встановлює поточні порогові значення за динамічним методом на основі значень вимірювань датчику

Інтерфейс OrientationStrategy

Абстракція для алгоритмів визначення напрямку на основі даних датчиків.

calcOrientation(raw : double[]) : double[] – обраховує напрям руху у вигляді вектора для 3х осей

Абстрактний клас GyroscopeOrientationStrategy

Абстракція для алгоритмів визначення напрямку на основі даних датчиків (де основна робота виконується за допомогою гіроскопа).

- calcOrientation(raw : double[]) : double[] – обраховує напрям руху у вигляді вектора для 3х осей

- setBias(bias : GyroscopeBias) – додає налаштування калібрування датчику

Клас VerticalAngularVelocityOrientationMethod

Реалізацію алгоритму визначення напрямку методом на основі вертикальної складової кутової швидкості.

- calcOrientation(raw : double[]) : double[] – обраховує напрям руху методом на основі вертикальної складової кутової швидкості у вигляді вектора для 3х осей
- setBias(bias : GyroscopeBias) – додає налаштування калібрування датчику

Абстрактний клас MagnetometerOrientationStrategy

Абстракція для алгоритмів визначення напрямку на основі даних датчиків (де основна робота виконується за допомогою магнітометра).

calcOrientation(raw : double[]) : double[] – обраховує напрям руху у вигляді вектора для 3х осей

setBias(bias : MagnetometerBias) – додає налаштування калібрування датчику

Клас MagnetometerOrientationMethod

Реалізацію алгоритму визначення напрямку методом на основі розрахунку даних з магнітометра.

calcOrientation(raw : double[]) : double[] – обраховує напрям руху у вигляді вектора для 3х осей на основі розрахунку даних з магнітометра.

setBias(bias : MagnetometerBias) – додає налаштування калібрування датчику магнітометру

setGyroscopeBias(bias : GyroscopeBias) – додає налаштування калібрування датчику гіроскоп

setGyroscopeRaw(raw : double[]) – додає дані гіроскопа для обчислення

Висновки

В цьому розділі дипломної роботи було розглянуто проектування застосунку для створення мапи в режимі реального часу за допомогою інерціальних методів.

Були розглянуті математичні моделі/методи для того, щоб визначити напрям руху та чи зробив користувач крок.

Методи для визначення кроків: на основі порогового виявлення, на основі динамічного порогового виявлення , на основі виявлення піків,на основі нульового перетину, на основі вертикальної складової кутової швидкості, на основі розрахунку даних з магнітометра, на основі модуля кутової швидкості з урахуванням знака.

Методи для визначення напрямку: на основі розрахунку даних з магнітометра, на основі модуля кутової швидкості з урахуванням знака.

Були створені наступні діаграми: діаграма пакетів, діаграма компонентів, діаграма класів, блок схема ітерації головного процесу застосунку.

В результаті проведеного проектування була розроблена застосунку для створення мапи в режимі реального часу за допомогою інерціальних методів. Ця архітектура відповідає вимогам застосунку і має переваги, які важливі для цього типу застосунків.

РОЗДІЛ 4
РОЗРОБКА, ДЕМОНСТРАЦІЯ І ТЕСТУВАННЯ СИСТЕМИ

4.1. Використання шаблонів GoF під час розробки

В попередньому розділі, на діаграмі класів (див. Рис. 3.6) був закладений фундамент для впровадження паттернів ООП GoF.

Перевага застосування впровадження паттернів в сучасних застосунках полягає в тому, що вони дозволяють розробляти більш гнучкі, надійне та ефективні програми.

Паттерни ООП GoF дозволяють легко змінювати або розширювати функціональність застосунку без необхідності вносити серйозні зміни в його код. Це досягається за рахунок того, що вони визначають загальні шаблони поведінки, які можна використовувати в різних контекстах – це дуже впливає на гнучкість і масштабованість системи.

Крім того, шаблони допомагають зробити застосунок більш надійним, оскільки вони допомагають розділити код на логічні модулі, які взаємодіють один з одним за допомогою добре відомих інтерфейсів. Це полегшує відстеження та усунення помилок.

Паттерни можуть допомогти покращити ефективність застосунку, оскільки вони дозволяють повторно використовувати код і уникати дублювання. Це може призвести до зменшення розміру коду, а також покращення його продуктивності.

Таким чином – роль паттернів в процесах інженерії ПЗ важлива.

В дипломному проекті, в зображених класах діаграми (див. Рис. 3.6) наведені місця для застосування 2х паттернів.

Перший – Singleton. В головному класі (DeadReckoningApp).

Другий – Strategy – для алгоритмів визначення кроку і напрямку.

Singleton для класу DeadReckoningApp дозволяє спростити взаємодію між різними компонентами застосунку. Наприклад, з будь якого місця проекту можна отримати доступ до БД, мап, алгоритмів роботи з датчиками і т. п.

Це дозволяє додатково не зберігати посилання на DeadReckoningApp через конструктори інших класів.

Крім того, переваги Singleton:

- Полегшує тестування. Тестувальники з будь якого тесту можуть отримати доступ до DeadReckoningApp.
- Уникнення створення декількох копій одного і того ж ресурсу – в нашому випадку, так як об’єкт БД SqlLite створюється в DeadReckoningApp то це гарантує те, що він буде створений один раз (якщо звісно не створювати спеціально в інших місцях коду).

Паттерн Strategy застосовується для класів, що реалізовувають методи(алгоритми) визначення напрямку та кроків.

Кожен метод/алгоритм міститься в своєму окремому класі. А зберігає поточний вибраний алгоритм – головний клас программи DeadReckoningApp.

Це дозволяє легко змінювати алгоритми без необхідності змінювати код застосунку. Наприклад, якщо потрібно користувач експерементує з алгоритмами в налаштуваннях – йому достатньо це зробити в декілька кліків в меню, в цей час на рівні коду - просто заміниться в застосунку об'єкт реалізації інтерфейсу OrientationStrategy чи StepCountStrategy.

4.2. Реалізація методів обрахування кроків/напрямку

Було реалізовано 4 методи визначення напрямку та кроків, що є основою інерційної навігації та процесу Dead reckoning.

Алгоритм обрахунку кроків на основі порогового виявлення – використовує значення 2х порогів (upperThreshold і lowerThreshold), які користувач методом спроб і помилок підбирає експериментально для свого пристрою, під свої особливості ходьби.

Лістинг 4.1.

Реалізація методу обрахунку кроків на основі порогового виявлення

```
public boolean isStep(double value)
{
    if (!peakFound)
    {
        if (value > upperThreshold)
        {
            peakFound = true;
            stepCount++;
        }
    } else
    {
        if (value < lowerThreshold) {
            peakFound = false;
        }
    }

    return peakFound;
}
```

peakFound - boolean змінна, яка вказує, чи був знайдений попередній пік.

Якщо peakFound false, то значить ми шукаємо пік і далі виконується наступне:

Якщо value(нове значення, отримане від датчика) більше за upperThreshold(верхній поріг, значення, яке потрібно перевищити, щоб вважати його піком), то ми знайшли пік:

peakFound стає true і збільшується stepCount (лічильник кроків).

Якщо peakFound true (тобто попередній пік знайдено), то ми перевіряємо чи опустилися нижче нижнього порога:

Якщо value менше за lowerThreshold, то вважаємо, що крок завершився: peakFound стає false - пошук нового піку починається знову.

В кінці методу повертається значення peakFound.

Якщо true, то поточне значення вважається піком і потенційно кроком.

Якщо false, то поточне значення не вважається піком.

Інший алгоритм - **обрахунку кроків на основі порогового виявлення** містить функцію isStep, що має майже ідентичний код з порівняннями значення датчика з нижнім і верхнім порогом.

Різниця лише в тому, що isStep при виконанні викликає setThresholds, код якої більш складний. В попередньому алгоритмі – setThresholds звичайний сетер значень, без логіки і обрахунків.

Перед тим, як викликаються методи нижче, слід памя'тати, що в конструкторі об'єкта змінні runCount = sumUpperValue = sumLowerValue = runCount = sumValue = avgValue = upperCount = lowerCount = 0.

Лістинг 4.2.

Реалізація алгоритму встановлення порогів динамічним шляхом

```
public void setThresholds(double value)
{
    runCount++;

    if (firstRun) {
        firstRun = false;
        lowerThreshold = value - sensitivity;
        upperThreshold = value + sensitivity;
        avgValue = value;
    }

    sumValue += value;

    if (value < avgValue) {
        lowerCount++;
        sumLowerValue += value;
    } else if (value > avgValue) {
        upperCount++;
        sumUpperValue += value;
    }

    if (runCount >= SENSOR_HZ) {
        runCount = 0;

        avgValue = sumValue / SENSOR_HZ;
        sumValue = 0;
```

```
            lowerThreshold = (sumLowerValue / lowerCount) - sensitivity;
            upperThreshold = (sumUpperValue / upperCount) + sensitivity;

            upperCount = 0;
            sumUpperValue = 0;
            lowerCount = 0;
            sumLowerValue = 0;
        }
}
```

Ініціалізація:

- Метод тепер отримує value – нове значення з датчику.
- Змінна runCount – відстежує кількість викликів методу.
- Змінна firstRun дозволяє перевірити, чи це перший виклик.

Під час першого виклику:

- firstRun встановлюється на false.
- Нижній поріг (lowerThreshold) та верхній поріг (upperThreshold) обчислюються як різниця value і sensitivity та їх сума відповідно.
- Середнє значення (avgValue) встановлюється на value.

Акумуляція значень:

- Змінна sumValue акумулює загальну суму всіх значень value.
- Змінна sumLowerValue акумулює значення value, які менші за avgValue.
- Змінна sumUpperValue акумулює значення value, які більші за avgValue.
- Змінні lowerCount та upperCount відстежують кількість відповідних значень.

Обновлення порогів (відбувається, коли runCount досягає значення SENSOR_HZ, яке встановлюється користувачем в налаштуваннях).

- runCount обнулюється.
- Середнє значення (avgValue) обчислюється як сума sumValue поділена на SENSOR_HZ.
- Нижній поріг (lowerThreshold) обчислюється як середня величина sumLowerValue поділена на lowerCount, з відніманням sensitivity.

- Верхній поріг (upperThreshold) обчислюється як середня величина sumUpperValue поділена на upperCount, з додаванням sensitivity.
- Змінні upperCount, sumUpperValue, lowerCount, та sumLowerValue обнулюються, готуючи метод до наступної порції даних.

Лістинг 4.3.

Реалізація алгоритму визначення напрямку на основі вертикальної складової кутової швидкості

```
public double[] calcOrientation(double[] values)
{
    long timestamp = Instant.now().toEpochMilli();

    //Зберегти першу часову мітку
    if (isFirstRun)
    {
        isFirstRun = false;
        lastTimestamp = timestamp;
        return new double[3];
    }

    //Застосувати калібровані значення
    double[] prepValues = new float[3];
    double[] biasValues = bias.getBias();
    for (int i = 0; i < 3; i++)
    {
        prepValues[i] = values[i] - biasValues[i];
        prepValues[i] = (Math.abs(prepValues[i]) > sensitivity) ?
prepValues[i] : 0;
    }

    return integrate(prepValues, timestamp);
}

private float[] integrate(double[] values, long timestamp)
{
    long currentTime = timestamp;
    long deltaTime = currentTime - lastTimestamp;

    double[] dOrientation = new double[3];

    //Інтегрування кутової швидкості за часом
    for (int i = 0; i < 3; i++)
        dOrientation[i] = values[i] * (double) deltaTime;

    lastTimestamp = currentTime;

    return dOrientation;
}
```

Фрагмент коду (див. Лістинг 4.3) використовується для обчислення кутових змін у тривимірному просторі, що дозволяє дізнатися напрямок. Це є реалізацією метода визначення напрямку на основі вертикальної складової кутової швидкості

Функція приймає в якості аргументу масив значень кутової швидкості, а потім повертає масив значень кутових змін.

Функція працює наступним чином:

Спочатку функція отримує поточну часову мітку. Якщо це перший виклик функції, поточна часова мітка зберігається як початкова часова мітка.

Потім функція застосовує калібровані значення до значень кутової швидкості. Калібровані значення використовуються для компенсації будь-яких похибок у датчиках.

Функція (Math.abs(prepValues[i]) > sensitivity) ? prepValues[i] : 0 перевіряє, чи абсолютне значення будь-якого з каліброваних значень prepValues[i] перевищує задане значення чутливості. Якщо так, значення prepValues[i] встановлюється на 0. Це робиться для того, щоб уникнути помилок, які можуть статися через шум у датчиках.

Нарешті, функція інтегрує калібровані значення кутової швидкості за часом. Інтегрування дає значення кутових змін.

Лістинг 4.4.

Реалізація методу визначення напрямку на основі розрахунку даних з магнітометра

```
public static double[] getOrientationMatrix(double[] values) {
    //Застосувати калібровані значення
    double[] prepValues = new double[3];
    for (int i = 0; i < 3; i++)
        prepValues[i] = values[i] - bias.getBias()[i];

    //Створити початкові матриці орієнтації з каліброваних значень
магнітометра та ґіроскопа
    SimpleMatrix magnInitMatrix = new
SimpleMatrix(vectorToMatrix(prepValues));
    SimpleMatrix gyroInitMatrix = new
SimpleMatrix(vectorToMatrix(gyroscopeValues));
```

```
    //Повернути початкові матриці орієнтації за допомогою матриці обертання
    magnInitMatrix = ROTATION_MATRIX.mult(magnInitMatrix);
    gyroInitMatrix = ROTATION_MATRIX.mult(gyroInitMatrix);

    //Витягнути значення з обробленої матриці гіроскопа для подальшого
розрахунку кутів
    double[][] tmpValues = denseMatrixToArray(gyroInitMatrix.getMatrix());

    //Обчислити крен та тангаж на основі витягнутих значень за допомогою
арктангенса двох аргументів
    double roll = Math.atan2(tmpValues[1][0], tmpValues[2][0]);
    double pitch = Math.atan2(-tmpValues[0][0], tmpValues[1][0] *
Math.sin(roll) + tmpValues[2][0] * Math.cos(roll));

    //Створити матрицю обертання, що представляє розраховані крен та тангаж
    double[][] rotationArray = {{Math.cos(pitch), Math.sin(pitch) *
Math.sin(roll), Math.sin(pitch) * Math.cos(roll)},
            {0, Math.cos(roll), -Math.sin(roll)},
            {-Math.sin(pitch), Math.cos(pitch) * Math.sin(roll),
Math.cos(pitch) * Math.cos(roll)}};

    //Повернути значення магнітного поля відповідно до показань крену та
тангажу за допомогою множення на матрицю обертання
    SimpleMatrix rotationMatrix = new SimpleMatrix(rotationArray);
    SimpleMatrix tmpR = rotationMatrix.mult(magnInitMatrix);

    // Обчислити напрямок (в радіанах) на основі поворотного значення
магнітного поля
    double heading = -1 * (Math.atan2(-tmpR.get(1), tmpR.get(0)) + 11.0 *
Math.PI / 180.0);

    //Створити матрицю обертання, що представляє обчислений напрямок.
    double[][] rotationMatrix2 = {{Math.cos(heading), -Math.sin(heading), 0},
            {Math.sin(heading), Math.cos(heading), 0},
            {0, 0, 1}};

    //Обчислити повну (початкову) матрицю обертання шляхом множення матриці
крену/тангажу на матрицю напрямку
    SimpleMatrix rotationMatrix3 = new SimpleMatrix(rotationMatrix2);
    SimpleMatrix result = rotationMatrix.mult(rotationMatrix3);

    // Витягнути значення з остаточної матриці орієнтації та перетворити їх
на масив кутів орієнтації
    double[][] orientationMatrix = denseMatrixToArray(result.getMatrix());

    //Повернути масив орієнтації з трьома значеннями для кутів орієнтації по
осях XY, XZ та YZ.
    double[] headings = new float[3];
    headings[0] = (float) Math.atan2(orientationMatrix[1][0],
orientationMatrix[0][0]);
    headings[1] = (float) Math.atan2(orientationMatrix[2][0],
orientationMatrix[0][0]);
    headings[2] = (float) Math.atan2(orientationMatrix[2][1],
orientationMatrix[1][1]);

    return headings;
}
```

Функція (див. Лістинг 4.4) отримує вхідні значення від датчиків магнітометра та гіроскопа. Далі функція обробляє ці значення для отримання початкових матриць орієнтації.

Потім функція використовує ці матриці для розрахунку кутів крену та тангажу. Після цього функція використовує ці кути для розрахунку напряму.

Нарешті, функція об'єднує всі ці значення в одну матрицю орієнтації, яку і повертає[23].

4.3. Демонстрація роботи ПЗ

1. Головне вікно меню застосунку

При запуску застосунку відкривається вікно з меню (див. рис. 4.1)

Рис. 4.1. Головне меню застосунку

Це вікно складається з чотирьох кнопок, розташованих вертикально одна під одною. Кожна кнопка має однаковий розмір і текст.

Користувач може взаємодіяти з формою, натискаючи на кнопки.

Натискання на кнопку "Нова мапа" відкриває вікно для створення нової карти.

Натискання на кнопку "Перелік створених мап" відкриває вікно з списком створених карт.

Натискання на кнопку "Налаштування" відкриває вікно налаштувань.

Натискання на кнопку "Про застосунок" відкриває вікно з інформацією про застосунок.

2. Вікно створення мапи

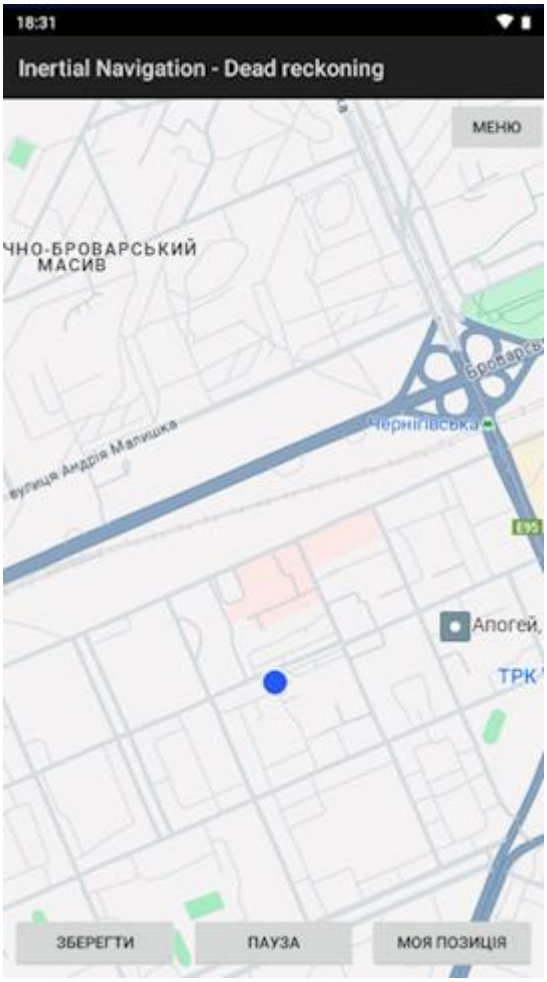

Рис. 4.2. Екран створення мапи

Це вікно складається з чотирьох кнопок:

Зберегти – зберегти мапу. Буде відкритий діалог, в якому застосунок попросить ввести назву мапи.

Пауза/Продовжити – дозволяє зупинити/продовжити запис траєкторії руху.

Моя позиція – при натисканні повертає екран до поточної позиції. Це дозволяє гортати мапу і після вивчення повертатися до поточного розташування.

При відкритті цього вікна застосунок намагається через gps дізнатися поточну позицію користувача. Якщо це неможливо, відкривається режим, де користувачу потрібно вибрати свою початкову позицію. Також цей режим можна відкрити власноруч – натиснувши і отримуючи кнопку «моя позиція».

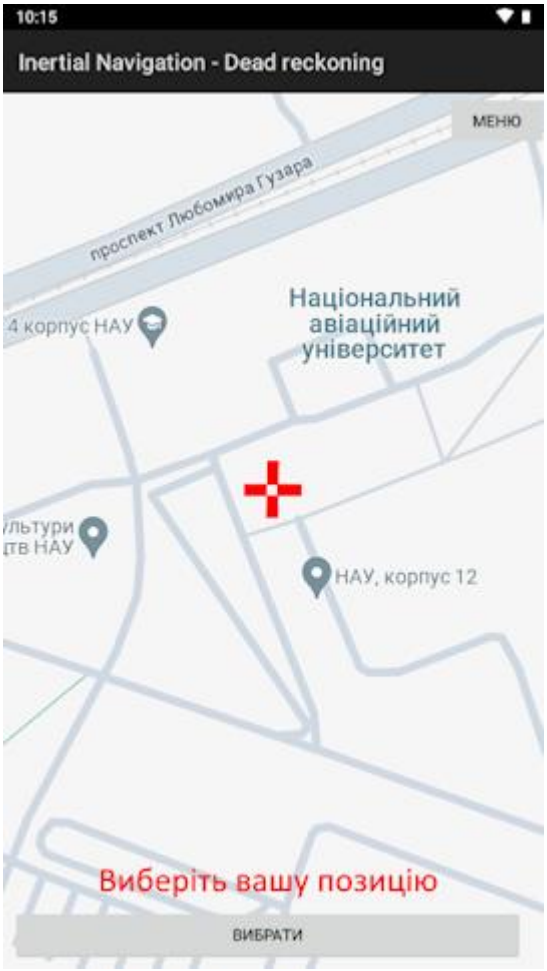

Рис. 4.3. Екран створення мапи

3. Процес створення мапи

Створення мапи відбувається автоматично під час запуску застосунку і знаходженні в вікні створення мапи (за умови, що пауза була не натиснута).

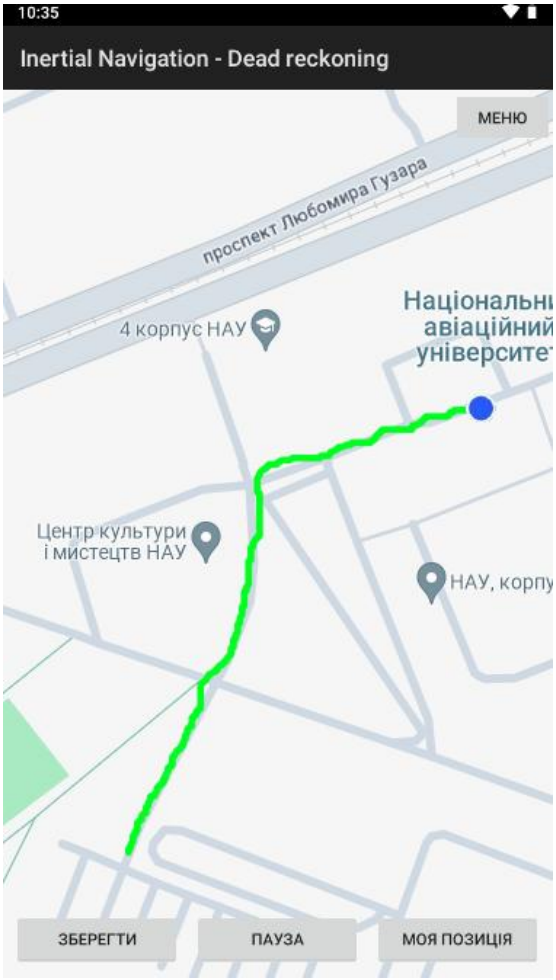

Рис. 4.4. Траєкторія руху по території НАУ

Створений застосунок дозволяє створювати мапи лабіринтів, тобто не лише лінійні маршрути. Як це відбувається:

Користувач проходить певний лінійний маршрут і робить запис траєкторії. Після цього він може зупинити запис (функція пауза), піти по створеній траєкторії руху і перенести позицію вручну – куди потрібно. І далі с тої точки знову почати запис і буде малюватися інша лінійна траєкторія, що дозволяє створювати мапи в структурах, що містять багато розвилок або маршрутів.

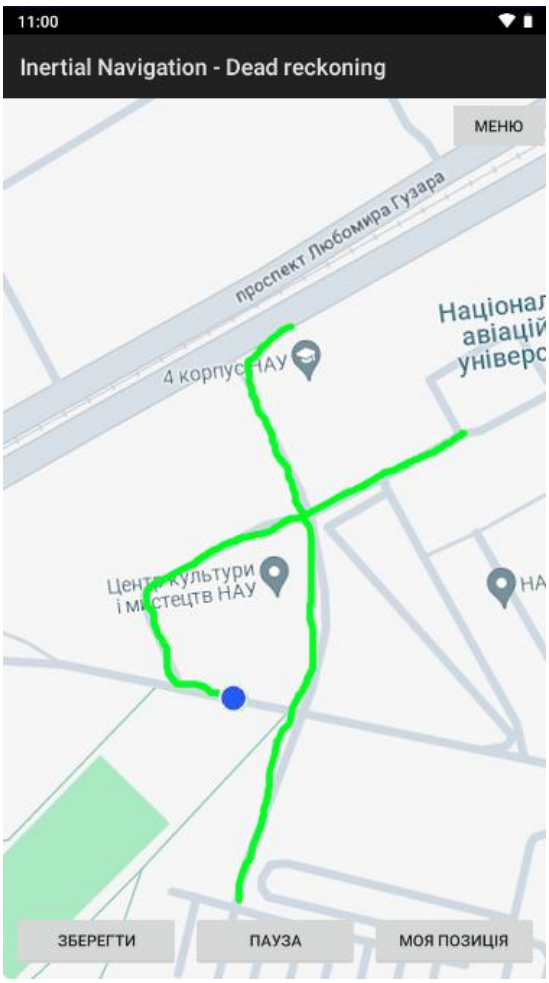

Рис. 4.5. Декілька траєкторій руху, які створюють більш складну мапу

Висновки

В цьому розділі був процес розробки і тестування застосунка. Реалізовано алгоритми визначення напрямку та кількості кроків на основі даних інерційної навігації, а саме:

Алгоритм визначення кроків на основі порогового виявлення

Алгоритм визначення кроків на основі динамічного порогового виявлення

Алгоритм визначення напрямку на основі вертикальної складової кутової швидкості

Алгоритм визначення кроків на основі розрахунку даних з магнітометра

Також у розділі було розроблено інтерфейс застосунку для створення та перегляду мап на основі. Інтерфейс дозволяє користувачеві створювати нові мапи, переглядати створені мапи та налаштовувати алгоритми, описані в цій роботі.

Демонстрація роботи застосунку показала, що він працює коректно і відповідає вимогам, що були поставлені в дипломному проекті.

ВИСНОВКИ

Метою даної дипломної роботи є полегшити і вдосконалити процес створення мап для об’єктів, де будь які види навігації крім dead reckoning малозастосовні.

Для досягнення мети був спроектований та розроблений застосунок.

Наразі існує дефіцит зручних засобів для автоматизованого створення мап без використання GPS. Існуючі застосунки мають низку недоліків, таких як низька точність, відсутність можливості збереження мап, відсутність можливості накладення мап на супутникові знімки тощо.

Використання методу Dead Reckoning для створення мап підземних/підводних споруд є перспективним напрямком. Цей метод дозволяє створювати мапи в умовах, де використання GPS неможливо або утруднене.

У дипломній роботі було розроблено мобільний застосунок для створення мап за допомогою процесу Dead Reckoning. Застосунок реалізовано на мові Java і використовує датчики гіроскопа, магнітометра та акселерометра для обчислення напрямку та кількості кроків.

Демонстрація роботи застосунку показала, що він працює коректно і відповідає вимогам, що були поставлені в дипломному проекті.

Основні результати роботи:

Зроблений аналіз підходів до обчислення кроків та напрямку, використовуючи можливості мобільних телефонів.

Програмно реалізовані описані в роботі алгоритми для обробки і аналізу даних з датчиків.

Розроблено інтерфейс користувача для створення мап за допомогою процеса dead reckoning. Інтерфейс є зручним та інтуїтивно зрозумілим для користувачів різних категорій.

Під час розробки було дотримано всіх етапів ітераційного життєвого циклу та принців методології ООП.

Підсумовуючи, можна сказати, що розроблений застосунок є ефективним інструментом для створення мап у місцях, де немає доступу до мобільного інтернету та GPS. Застосунок може бути використаний для різних цілей, таких як картографування підземних комунікацій, створення мап підводних затонулих об'єктів, відстеження переміщення людей у складних умовах тощо.

Рекомендації для подальших досліджень:

Додати підтримку інших датчиків, таких як барометр і підтримку GPS.

Програмно реалізувати всі найбільш популярні методи для обрахування кількості кроків та напряму.

Провести аналіз ефективності наведених в роботі методів в різних умовах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Косовець І. А. Алгоритм локального позиціонування об\'єкта на основі технології Bluetooth beacon [Електронний ресурс] / Іван Андрійович Косовець. – 2021. – Режим доступу до ресурсу: https://ela.kpi.ua/bitstream/123456789/49504/1/Kosovets_magistr.pdf.
2. Яременко Є. А. Система орієнтування на місцевості з використанням технології доповненої реальності [Електронний ресурс] / Євгеній Анатолійович Яременко. – 2018. – Режим доступу до ресурсу: https://ela.kpi.ua/bitstream/123456789/26411/1/Yaremenko_magistr.pdf.
3. Deepak Mishra. Evaluation of Intersystem Interference between NavIC, GPS and Galileo [Електронний ресурс] / Deepak Mishra, Sanghamitra Banik, Shalvi Bhatnagar. – 2018. – Режим доступу до ресурсу: https://www.caeaccess.org/archives/volume7/number12/mishra-2018-cae-652737.pdf.
4. Roza Dastres. Radio Frequency Identification (RFID) based wireless manufacturing systems, a review [Електронний ресурс] / Roza Dastres, Mohsen Soori, Mohammed Asamel. – 2022. – Режим доступу до ресурсу: https://hal.science/hal-03594902/document.
5. Enhancing Bluetooth Location Services with Direction Finding [Електронний ресурс] – Режим доступу до ресурсу: https://www.bluetooth.com/wp-content/uploads/2019/03/1901_Enhancing-Bluetooth-Location-Service_FINAL.pdf.
6. Chutao Zheng. Ultra-Wideband Technology: Characteristics, Applications and Challenges [Електронний ресурс] / Chutao Zheng, Yuchu Ge, Anfu Guo. – 2023. – Режим доступу до ресурсу: https://arxiv.org/pdf/2307.13066.pdf.
7. Cliff Randell. Personal Position Measurement Using Dead Reckoning [Електронний ресурс] / Cliff Randell, Chris Djiallis, Henk Muller. – 2003. – Режим доступу до ресурсу:

http://www1.cs.columbia.edu/~drexel/CandExam/Randell2003_ISWC_DeadReckoning.pdf.

8. Lazarevskiy O. A. Navigation dead reckoning system based on a mobile phone [Електронний ресурс] / O. A. Lazarevskiy. – 2021. – Режим доступу до ресурсу: https://er.nau.edu.ua/bitstream/NAU/50760/1/%D0%9B%D0%B0%D0%B7%D0%B0%D1%80%D0%B5%D0%B2%D1%81%D0%BA%D0%B8%D0%B9%20177-179.pdf.

9. Метро, якого немає: Складна дорога до Теремкам [Електронний ресурс]. – 2013. – Режим доступу до ресурсу: https://tov-tob.livejournal.com/97496.html.

10. Lazarevskiy O. A. Navigation dead reckoning system based on a mobile phone [Електронний ресурс] / Lazarevskiy. – 2021. – Режим доступу до ресурсу: https://er.nau.edu.ua/handle/NAU/51515.

11. DeadReckoning [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/nisargnp/DeadReckoning.

12. Apple Store - PDR [Електронний ресурс] – Режим доступу до ресурсу: https://apps.apple.com/ua/app/pdr%E6%B8%AC%E4%BD%8D/id644833467?l=ru.

13. Strava Offers New Tool to Calculate Carbon Savings on Platform [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://press.strava.com/articles/strava-offers-new-tool-to-calculate-carbon-savings-on-platform.

14. PYPL PopularitY of Programming Language [Електронний ресурс] – Режим доступу до ресурсу: https://pypl.github.io/PYPL.html.

15. Xiaolin Ning. Spacecraft Autonomous Navigation Using the Doppler Velocity Differences of Different Points on the Solar Disk [Електронний ресурс] / Xiaolin Ning, Wen Chao. – 2020. – Режим доступу до ресурсу: https://www.researchgate.net/publication/342062145_Spacecraft_Autonomous_

Navigation_Using_the_Doppler_Velocity_Differences_of_Different_Points_on_the_Solar_Disk.

16. Wikipedia: Geographic coordinate system [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Geographic_coordinate_system.

17. Xiaomin Kang. A Novel Walking Detection and Step Counting Algorithm Using Unconstrained Smartphones [Електронний ресурс] / Xiaomin Kang, Baoqi Huang, Guodong Qi. – 2018. – Режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5796454/

18. Designing a Pedometer and Calorie Counter [Електронний ресурс] – Режим доступу до ресурсу: http://web.cs.wpi.edu/~emmanuel/courses/cs528/F17/slides/papers/deepak_ganesan_pedometer.pdf.

19. Jungryul Seo. Step counting on smartphones using advanced zero-crossing and linear regression [Електронний ресурс] / Jungryul Seo, Yutsai Chiang, Teemu Laine. – 2015. – Режим доступу до ресурсу: https://www.researchgate.net/publication/282680705_Step_counting_on_smartphones_using_advanced_zero-crossing_and_linear_regression.

20. Adi Manos. Gravity-Based Methods for Heading Computation in Pedestrian Dead Reckoning [Електронний ресурс] / Adi Manos, Itzik Klein, Tamir Hazan. – 2019. – Режим доступу до ресурсу: https://www.researchgate.net/publication/331613480_Gravity-Based_Methods_for_Heading_Computation_in_Pedestrian_Dead_Reckoning.

21. MEMS accelerometers, magnetometers and orientation angles [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/articles/491476/.

22. Smartphone-Based Pedestrian Dead Reckoning for 3D Indoor Positioning [Електронний ресурс] / Jijun Geng, Linyuan Xia, Jingchao Xia та ін.]. – 2021. – Режим доступу до ресурсу:

https://www.researchgate.net/publication/356901887_Smartphone-Based_Pedestrian_Dead_Reckoning_for_3D_Indoor_Positioning.

23. Ying Guo. Multimode Pedestrian Dead Reckoning Gait Detection Algorithm Based on Identification of Pedestrian Phone Carrying Position [Електронний ресурс] / Ying Guo, Qinghua Liu, Xianlei Ji. – 2019. – Режим доступу до ресурсу: https://www.researchgate.net/publication/336950260_Multimode_Pedestrian_Dead_Reckoning_Gait_Detection_Algorithm_Based_on_Identification_of_Pedestrian_Phone_Carrying_Position.