Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти - другий (магістерський)

Дослідження методів та програмних систем курсової стабілізації руху автомобіля
(тема)

Виконав: студент 2 курсу, групи ПЗСм-18-1
Гавриш О.В.
(прізвище, ініціали)
спеціальності 121 - Інженерія програмного забезпечення
(код і повна назва спеціальності)
Освітньо-професійної програми
(тип програми)
Програмне забезпечення систем
(повна назва освітньої програми)
Керівник проф. Білоус Н.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2019 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 - Інженерія програмного забезпечення
(код і повна назва)

Спеціалізація Програмне забезпечення систем
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Гавришу Олегу Владиславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів та програмних систем курсової стабілізації руху автомобіля»
затверджена наказом по університету від ___ _____2019 р. № _____
2. Термін подання студентом роботи до екзаменаційної комісії
17 грудня 2019 р.
3. Вихідні дані до роботи *електронні ресурси за обраною тематикою, мінімальні вимоги до функціональності програми, загальні вимоги до архітектури системи*
4. Перелік питань, що потрібно опрацювати в роботі *аналіз предметної області і постановка задачі, аналіз вимог до програмної системи, UML-моделювання предметної області, дослідження алгоритмів стабілізації руху, розробка математичної моделі руху автомобіля з урахуванням зовнішніх чинників, розробка математичної моделі підбору параметрів стабілізації, розробка бази даних, розробка алгоритмів, програмна реалізація системи, тестування.*

5 Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	проф. Білоус Н.В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз проблемної області та постановка задачі	02.09.19-18.09.19	виконано
2	Аналіз існуючих методів стабілізації	18.09.19-21.09.19	виконано
3	Аналіз та моделювання предметної області	21.09.19-30.09.19	виконано
4	Розробка математичної моделі руху автомобіля	01.10.19-14.10.19	виконано
5	Аналіз методів пошуку компенсаційних значень	14.10.19-24.10.19	виконано
6	Розробка схеми бази даних	24.10.19-26.10.19	виконано
7	Розробка алгоритму стабілізації	26.10.19-13.11.19	виконано
8	Створення коду програми	14.11.19-24.11.19	виконано
9	Тестування і налагодження програми	24.11.19-03.12.19	виконано
10	Підготовка пояснювальної записки	03.12.19-08.12.19	виконано
11	Підготовка презентації та доповіді	09.12.19	виконано
12	Попередній захист	10.12.19	виконано
13	Нормоконтроль, рецензування		
14	Занесення диплома в електронний архів		
15	Допуск до захисту у зав. кафедри		

Дата видачі завдання __2___ вересня_____2019 р.

Студент _____
(підпис)

Керівник роботи _____ __проф. Білоус Н.В._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Атестаційна робота магістра містить: 60 с., 12 рис., 11 джер.

ANDROID, FIREBASE, FIRESTORE, JSON, NOSQL, OKHTTP, БАЗА ДАНИХ, РОБОТ, СТАБІЛІЗАЦІЯ.

Об’єктом дослідження є методи та алгоритми стабілізації руху колісних наземних безпілотних транспортних засобів.

Метою роботи є покращення ефективності використання роботизованих платформ в системі стабілізації руху.

Методи розробки базуються на таких технологіях, як Android SDK, Kotlin, Firebase Firestore, ESP IDS, Gradle.

В результаті роботи було досліджено методи теорії стабілізації руху автомобіля для підтримки повного контролю над траєкторією та швидкістю пересування і програмно реалізовано систему стабілізації, яка представляє собою ПЗ для автомобіля та мобільний додаток для операційної системи Android.

ANDROID, DATABASE, FIREBASE, FIRESTORE, JSON, NOSQL, OKHTTP, ROBOT, STABILIZATION.

The object of the study is the methods and algorythms of motion stabilization for radiocontrolled wheeled ground vehicles.

The purpose of the work is to create a software system for vehicle stabilization.

Development methods are based on such technologies as Android SDK, Kotlin, Firebase Firestore, ESP IDS, Gradle.

As a result, the methods of stabilization theory in the direction of their use to support full control above vehicle were investigated, and the system for stabilization, which is a firmware for vehicle and a mobile application for the Android operating system, were implemented.

ЗМІСТ

ВСТУП

В наш час, з розвитком технічного прогресу, все більше галузей промисловості, медицини, військові та навіть цивільні люди в побуті використовують різноманітні пристрої для автоматизації чи полегшення певних процесів та завдань. Це дозволяє виконувати збільшити безпеку, швидкість точність, потужність виконання процесів, що в свою чергу дає ще більший поштовх для розвитку індустрії. Адже певні процеси є небезпечними чи навіть невиконуваними за тими чи іншими причинами для людини і надзвичайно важливим являється автоматизація саме таких видів діяльності.

Одним із напрямків такої автоматизації являються безпілотні транспортні машини, які можуть бути використані для доставки вантажів, тактичної розвідки, спортивних змагань, розміщення спеціалізованих пристроїв тощо. Виділяють декілька найпопулярніших типів: повітряні, водні/підводні та наземні.

Найпростішими та доступними у використанні являються саме наземні. Вони зазвичай мають колісне або гусеничне шасі, рідше - шнекового чи шарнірного типу. Вони є відносно дешевими та простими у створенні не потребують спеціально підготовленої поверхні для пересування, як правило ними можна дистанційно керувати за допомогою радіоканалу та певного протоколу комунікації.

Та на практиці виникають певні труднощі, що залежать від впливу оточуючого середовища, поверхні пересування, кінематичних та динамічних характеристик шасі, геометричної трансформації шин, різного коефіцієнта щеплення, різного роду нерівності тощо. Це досить сильно ускладнює прогнозованість керування такими засобами, потребує деяких ручних коригувань з боку оператора, знижує продуктивність то точність роботи такого пристрою.

Об’єктом даного дослідження являється проблема стабілізації руху колісних роботів по пересіченій місцевості чи в умовах недостатнього щеплення з поверхнею. Предметом дослідження являються методи стабілізації і прийняття раціональних рішень для підтримки прямолінійного руху робота або за певною бажаною траєкторією. Метою роботи є дослідження методів для створення

програмної системи стабілізації руху колісних роботів, що використовуються для пересування по пересіченій місцевості.

Під час виконання атестаційної роботи було проведено збір даних, аналіз і моделювання предметної області стабілізації колісних роботів, проаналізовано різноманітні методи стабілізації, розроблено математичну модель руху колісного робота, алгоритм коригування вхідних параметрів, не реляційну базу даних для зберігання статистичних даних та розроблено алгоритми для підтримки руху за заданою траєкторією, алгоритми для підтримки зв’язку в реальному часі між роботом та оператором. Виконано тестування та апробацію отриманих результатів на практиці, а також модульне тестування окремих частин програмної системи.

За результатами атестаційної роботи було створено презентацію (див. додаток А). Також була написана та подана до опублікування стаття «Дослідження методів та програмних систем курсової стабілізації руху автомобіля» в матеріалах IV міжнародної науково-практичної конференції в Ліверпулі «Scientific Achievements of Modern Society» (див. дод. Б). Найцікавіші уривки програмного коду клієнта, сервера та пристрою наведено в додатку В.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз проблемної області систем стабілізації

Якщо розглянути галузь робототехніки, то під стабілізацією розуміють процес коригування вхідних та внутрішніх параметрів системи таким чином, щоб вихідні дані мали мінімальне відхилення від норми чи від очікуваних результатів, яких прагне досягти оператор.

На сьогоднішній день використання колісних роботів породжує певні незручності та потребує деяких зусиль від оператора при використанні на пересіченій місцевості. Це стається через неоднорідність поверхні, непередбачуваність поведінки шасі, та інші чинники, що унікальні для кожного середовища використання.

У простих випадках за відсутності автоматизованої програмної чи апаратної системи стабілізації, оператор повинен приймати міри для збереження контролю над траєкторією руху робота. Це може бути як зменшення швидкості пересування для збереження щеплення з дорогою, так і зворотне коригування вхідних параметрів для компенсації зовнішніх чинників. Обидва варіанти є дієвими, але все таки накладають обмеження на ефективність використання такої машини.

Також сама суть стабілізації, нехай то автоматизована, чи ручна, залежить від типу середовища, поверхні, характеристик шасі, габаритів та ваги вантажу який перевозиться, обладнання, тощо. Наприклад в умовах низьких температур коли поверхня може бути покрита снігом чи льодом, а шини робота втрачають свої ходові властивості, виникають проблеми із щеплення з дорожнім покриттям. Це можна компенсувати як зворотнім підрулюванням у випадку заносу, так і зменшенням потужності, яка подається на колеса.

У випадку руху по кам’янистій поверхні, важливий вплив мають динамічні властивості шасі, те, як апарат проходить через велике каміння, як при цьому змінюються його напрям руху, наскільки ефективне щеплення з дорогою мають колеса. В такому випадку також може бути дієвим зменшення швидкості пересування, проте все одно можуть спостерігатися несподівані зміни траєкторії руху внаслідок фізичного зміщення шасі відносно попереднього напрямку.

Щоб дізнатися, що потрібно для автоматизованої стабілізації, треба в першу чергу виділити основні методі та метрики, якими керується людина в ролі оператора при ручній стабілізації руху. Перш за все, це візуальний контроль напрямку руху та фізичного положення в просторі самого робота. Це дозволяє помічати факти відхилення від траєкторії та приймати міри по її відновленню або попереджати такі моменти заздалегідь.

Іншим способом є протидія негативним чинникам, які скоріше за все вплинуть на траєкторію руху. Наприклад оператор може помітити, що на траєкторії руху безпілотної машини знаходиться складна поверхня і для більш контрольованого її проходження, необхідно зменшити чи навпаки - збільшити швидкість.

Варто звернути увагу й на такий особливий спосіб стабілізації, коли оператор має мету максимально швидко та ефективно пройти певну ділянку середовища та має умовно нескінченну кількість тренувальних спроб. Такий особливий випадок зустрічається в спортивних перегонах, які хоча й проводяться на складних трасах, але оператори мають можливість спробувати різні стратегії та стилі проходження траси та підібрати найефективніші емпіричним шляхом.

Можна зробити висновок, що програмний продукт, що створюється, має поєднувати в собі всі перелічені методі стабілізації для досягнення найбільш ефективного пересування та роботи.

Дана програмна система буде представляти собою сукупність мобільного додатку, програмного забезпечення, яке безпосередньо керує апартною частиною робота та серверну частину, яка здебільшого використовується для обробки даних. Мобільний додаток дає можливість користуватися програмним застосунком у будь-який зручний для користувача час, володіє певною автономністю, власне, мобільністю, тож може бути використаним, практично в будь-який час і в будь-якому місці. В наші часи найбільш популярними мобільними операційними системами стали Android від компанії Google та iOS від компанії Apple. Передову сходинку серед пристроїв середнього класу вже не перший рік займає саме операційна система під назвою Android, і через свою

популярність та відкритість, саме ця система була обрана в якості платформи для реалізації користувацької частини програмної системи стабілізації руху колісних роботів.

Варто зауважити, що програмний продукт, що створюється, буде включати в себе функціонал складання оптимальної послідовності керуючого сигналу для апаратної частини для максимально ефективного та стабільного пересування автомобіля чи роботизованої платформи. В наш час для того, щоб отримати найбільш ефективне та дієве рішення у більшості сфер використання таких апаратів, існує необхідність саме у науковому підході, адже матеріальні та часові витрати, пов'язані з неточностями, помилками чи похибками, можуть бути дуже великими. Раціональні шляхи вирішення проблем та рішення дозволяють отримувати прийнятний результат та виконувати поставлені задачі при мінімальних людських, ресурсних, грошових і матеріальних зусиллях. Також слід відзначити, що попереднє моделювання системи в цілому, а також окремих її компонентів дозволяє використовувати сучасні системи автоматизованого проектування для попередньої перевірки майбутнього пристрою чи програми, моделювання різноманітних ситуацій ще до початку безпосереднього процесу розробки.

Як висновок з усього переліченого, можна сказати, що для створення такого складного продукту стабілізації руху колісних роботів випливає необхідність в попередньому математичному моделюванні системи, розробці схеми бази даних для зберігання створеної математичної моделі та статистичних даних, аналізі різноманітних підходів та методів, що можуть бути застосованими для створення раціонального та ефективного руху колісних роботів по пересіченій місцевості. Це є особливо важливим навіть для етапу розробки апаратної та механічної платформи. Адже алгоритм руху напряму впливає на фактичну поведінку роботизованого транспортного засобу, що може викликати пошкодження та сповільнити розробку у разі виникнення помилкової ситуації.

1.2 Аналіз аналогів

Щоб зрозуміти поточний стан ринку та знайти базові функціональні можливості і не функціональні властивості програмного продукту, правильно та раціонально розставити пріоритети в ході розробки та запобігти накраплянню на недоліки, допущені організаціями, які вже колись до цього створювали подібні системи, треба провести детальний аналіз подібних програмних систем. В наш час такі колісні транспортні засоби набули неабиякої популярності у різних сферах та аспектах нашого життя, тож існує багато готових програмних та апаратних рішень, які дозволяють реалізувати стабілізацію руху радіокерованого автомобіля.

Одним із найпопулярніших таких пристроїв є SkyRC GC301 (див. рис. 1.1), що є системою для стабілізації руху радіокерованого автомобіля шляхом коригування поворотного приводу. Система використовує гіроскоп для визначення поточного напрямку руху та фізичного положення автомобіля в просторі. На основі цих даних програмна частина виконує коригування вхідного параметру повороту передніх коліс відповідно до отриманих результатів. Із переваг можна відмітити ручне коригування межі впливу шляхом зміни значення апаратного регулятора на лицевій панелі і також аналогічного регулятора, який може керуватись за радіоканалом. Недоліком являється те, що система не бере до уваги поточну лінійну швидкість пересування автомобіля та факт прослизування коліс.

Внаслідок цього, він видає неточні коригування та є недостатньо ефективним на малих швидкостях, так як використовує лише гіроскопічний ефект в якості метрики стабілізації.

Іншим аналогом додатку, що розробляється, може слугувати розробка компанії Spektrum – AVC (Active Vehicle Control). Це ціле сімейство прийчачів для радіокерованих автомобілів, які включають в себе комплексну систему стабілізації (див. рис. 1.2).

Рисунок 1.1 – Зовнішній вигляд SkyRC GC301

Рисунок 1.2 – Зовнішній вигляд Spektrum AVC SRS4220

Основною перевагою серед конкурентів являється те, що коригування виконується як для поворотного механізму, так і для тягового. Це дозволяє значно розширити діапазон роботи системи стабілізації, різноманіття поверхонь пересування та інших зовнішніх умов середовища. Як і попередній приклад, дана система аналізує положення автомобіля на основі даних гіроскопа.

Проте така прийомна система обов’язково потребує спеціального передавача, щоб повноцінно використовувати можливості стабілізації, виконувати калібрацію системи, проводити налаштування в реальному часі. Одним із недоліків являється те, що не береться до уваги поточна швидкість

транспортного засобу, що може досить сильно впливати на характеристики стабілізації.

Також аналогом є продукт компанії HPI Racing - Dbox2 (див. рис. 1.3). Серед його особливостей слід відмітити лише малі розміри та простоту підключення. За принципом роботи він мало чим відрізняється від першого аналогу SkyRC.

Рисунок 1.3 – Зовнішній вигляд HPI Racing Dbox2

Даний модуль має також фізичні перемикачі для вибору режимів роботи та аналоговий регулятор чутливості. Серед мінусів також можна відмітити відсутність контролю швидкості та вплив на тяговий агрегат.

Всі вище описані системи виконані у вигляді окремого апаратного модуля із встановленим програмним забезпеченням. Це накладає певні обмеження до інтерфейсу калібрації та налаштування стабілізації. Такі системи не мають зворотного зв’язку чи можливості збереження різних наборів налаштувань. Також жоден із пристроїв не враховує поточну швидкість пересування транспортного засобу та коефіцієнт пробуксовування коліс на слизькій поверхні. Використання апаратних елементів інтерфейсу, механічних потенціометрів, кнопок накладає певні обмеження на подальше розширення функціоналу.

1.3 Постановка задачі

В ході виконання атестаційної роботи магістра необхідно дослідити різноманітні методи для аналізу метрик, створення алгоритму, його налаштування та калібрацію.

Засновуючись на результатах дослідження необхідно змоделювати та розробити програмну систему, в якій буде реалізована стабілізація руху колісного безпілотного транспортного засобу з використанням акселерометра, гіроскопа, датчиків швидкостей передніх та задніх коліс, поточного кута повороту коліс. Ця система має стати зручним інструментом для розробників та користувачів різноманітних колісних роботів, радіокерованих автомобілів, військової та промислової техніки. Створена система повинна поєднувати у собі всі переваги та виключати недоліки вже існуючих систем.

В першу чергу в роботі потрібно спроектувати та розробити модель радіокерованого автомобіля з усіма необхідними датчиками та сенсорами. Програмне та апаратне забезпечення повинно підтримувати протокол спілкування з мобільним додатком, включати в себе базовий алгоритм стабілізації на основі заданих параметрів, збір, тимчасове зберігання та пакетну передачу отриманих даних. Для реалізації контролера було обрано модуль Espresif ESP-32, який поєднуж в собі двоядерний програмований процесор, Wi-Fi модуль для зв’язку з іншими пристроями, та необхідну периферію для підключення зовнішніх сенсорів та керуючих агрегатів.

Також роботі потрібно спроектувати та розробити мобільний додаток для безпосереднього керування автомобілем, отримання та аналізу даних з сенсорів, встановлення контролюючих параметрів стабілізації та їх адаптації в реальному часі. Додаток буде мати два режими роботи: режим збору даних та калібрації і режим керування транспортним засобом. Користувачі будуть мати можливість авторизуватися та зареєструватися у додатку. У режимі калібрації користувачі зможуть скористатися наступними функціями:

- перегляд поточного статусу датчиків в реальному часі;
- збір пакетів з історією значень датчиків за період часу;

– запуск аналізу даних з датчиків;
– можливість ручної зміни параметрів стабілізації;
– додавання та збереження користувацьких наборів параметрів;
– можливість застосувати попередньо створений набір параметрів;
– редагування та видалення наборів параметрів.

В режимі керування можна скористатися наступними функціями:

– ручне керування поворотом та тяговим зусиллям автомобіля;
– перегляд актуальних даних про поточний заряд батареї автомобіля;
– увімкнення стабілізації з використанням одного із збережених наборів;
– перегляд статусу каналу зв’язку з радіокерованим автомобілем;
– налаштування чутливості, крайніх та центральних точок осей керування.

В ході виконання роботи необхідно виконати наступні завдання:

– провести аналіз предметної області та виконати UML-проектування;
– розробити математичну модель для стабілізації руху;
– розробити базу даних для зберігання інформації з математичної моделі;
– розробити алгоритми підтримки стабілізації;
– програмно реалізувати систему стабілізації;
– провести тестування розробленої системи на коректність роботи.

Для розробки та створення програмного забезпечення автомобіля було обрано середовище розробки VS Code ті плагін Platform IO, воно дозволяє легко підключати сторонні бібліотеки для роботи з пристроями, виконувати компіляцію, запуск та відладку програмного коду. Було прийнято рішення розробляти модуль стабілізації на основі платформи Espresif ESP-32, оскільки вона є однією з найбільш поширеною серед інтегрованих платформ для розробки пристроїв з підтримкою Wi-Fi та зовнішніх модулів. Те, що модуль використовує Wi-Fi канал для зв’язку дозволяє використовувати будь-який сучасний пристрій, який підтримує даний мережевий протокол.

Для розробки мобільного додатку було обрано середовище розробки Android Studio, що базується на IntelliJ Idea, оскільки воно надає значно більше

можливостей, ніж його конкуренти, і є більш зручним для використання. Було прийнято рішення розробляти мобільний додаток під платформу Android, оскільки вона є найбільш поширеною серед мобільних платформ. Те, що додаток є мобільним, дасть змогу користувачам використовувати його у будь-якому місці у зручний для них час. Для написання клієнтської частини буде використовуватися Android SDK з використанням додаткових бібліотек.

Для створення серверної частини обрано хмарну платформу Google Firebase. Вона задовольняє потреби зберігання та отримання даних, їх обробку з використанням NoSQL інструментів, а також пропонує вбудовані сервіси для хмарної обробки, аналізу та візуалізації даних, авторизації та реєстрації нових користувачів системи. В той же час, система є зручною завдяки тому, що не потребує окремого процесу розсортування, а надається як хмарний сервіс, який можна налаштувати під конкретну задачу і працювати з ним за допомогою клієнтського набору розробки програмного забезпечення.

Також ще однією із переваг обраного рішення є потенційна можливість використання інших суміжних хмарних сервісів не лише для зберігання, а й для обробки даних, в тому числі із використанням засобів глибокого машинного навчання. Це дозволить за необхідності удосконалити алгоритм, зробити поведінку стабілізації більш точною, адаптувати її до будь яких умов використання транспортного засобу.

2 ПЕРЕЛІК ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Зазвичай, в загальному вигляді, робота програмної системи або окремих її модулів заключається в тому, що вона приймає на вхід певний набір даних, виконує обробку та, власне виведення отриманих результатів для подальшого використання чи аналізу. Тож, для того, щоб створити таку систему чи її підсистему, перш за все, треба сформувати вимоги до правил виконання її функціональних елементів та обробки вхідних параметрів. Вцілому, під функціональними і не функціональними вимогами до програмного продукту мають на увазі той функціонал і властивості, які вона повинна мати для того, щоб виконувати поставлені задачі. В даній атестаційній роботі потрібно реалізувати систему стабілізації руху автомобіля спираючись на показання зовнішніх датчиків та сенсорів. Тож, перш за все, треба заздалегідь визначитись та продумати вимоги до програмного продукту задля досягнення найкращого результату.

2.1 Призначення розробки

Даний програмний продукт для стабілізації руху автомобіля призначений перш за все для користувачів різноманітних колісних роботизованих платформ, радіокерованих автомобілів, які потребують стабільного та передбачуваного пересування по поверхні з нерівностями, недостатнім щепленням тощо. Треба створити програмне забезпечення для самого автомобіля, яке буде проміжним шаром між агрегатами, датчиками та мобільним додатком. А також безпосередньо сам мобільний додаток, який давав би змогу проводити аналіз отриманих даних, які надійшли з автомобіля, налаштовувати та проводити калібрацію системи стабілізації. Для того, щоб користувач мав змогу зберігати декілька варіантів налаштувань, система повинна підтримувати збереження та застосування цих наборів на сервері. Зазвичай, досвідчені користувачі можуть самостійно коригувати рух керованого транспортного засобу для мінімізації впливу різних характеристик поверхні, проте це потребує певних зусиль, концентрації, не завжди є оптимальним, тому потребує автоматизації.

Тож, із всього вищесказаного можна зробити висновок, що є необхідність у створенні програмної системи, яка допоможе користувачам таких роботизованих платформ максимально ефективно використовувати можливості шасі для пересування по наземному маршруту.

2.2 Вимоги до програмного продукту

Так як даний програмний продукт включає в себе взаємодію різних пристроїв в реальному часі з критичними вимогами до часу відгуку, програмна система має бути побудована на багатошаровій клієнт-серверній архітектурі і реалізована за допомогою мов програмування C++ і Kotlin для радіокерованого автомобіля та мобільного додатку відповідно. Радіокерований автомобіль буде надсилати необроблені дані до мобільного додатку та приймати керуючі команди, за протоколом HTTP. Мобільний клієнт отримує ці дані, виконує аналіз, виводить користувачеві та надає можливість коригування. Згодом ці оброблені дані можуть бути завантажені на сервер для зберігання і подальшого перевикористання в стабілізації.

Програмне забезпечення для апаратної частини має бути реалізовано за допомогою платформи ESP IDF, вона включає в себе базовий функціонал для роботи з HTTP сервером, обробки та відповіді на команди, підтримує роботу із зовнішніми апаратними пристроями та датчиками. Використання мови C++ забезпечує високу швидкість роботи програми, більш делікатну та гнучку обробку даних в пам’яті, малий розмір скомпільованого коду.

Клієнтська частина повинна бути реалізована за допомогою Android SDK та деяких зовнішніх бібліотек для спрощення реалізації окремих можливостей системи. Мінімальною версію операційної системи було обрано API 17 Jelly Bean, вона є оптимальною за обсягом підтримуваного функціоналу системи та інструментів розробки, а також охоплює переважну частину всіх мобільних пристроїв під управлінням ОС Android. Для спрощення роботи з HTTP обрано бібліотеку OkHttp. Вона підтримує можливість легкої інтеграції наскрізної функціональності, наприклад логування, обробка помилок, серіалізація та

десеріалізація даних тощо. Так як додаток буде мати паралельно обробку декількох потоків даних, в якості допоміжного інструменту обрано Kotlin Coroutines, що дозволить надзвичайно ефективно запускати, контролювати численні асинхронні задачі, забезпечувати коректну відміну, зв’язок між залежними операціями тощо.

Серверна частина перевано складається із NoSql сховища даних, що є частиною інфраструктури Google Firebase Firestore. Це забезпечує простоту та швидкість роботи з великим обсягом даних в даній предметній області. Також дане рішення пропонує зручний інструмент для взаємодії з мобільного додатку і механізм авторизації та реєстрації користувачів.

Дані, що передаються з клієнта на сервер, мають бути захищені від зовнішнього втручання та сторонньої модифікації. Це досягається завдяки використанню шифрування даних на транспортному рівні (TLS).

Основна функціональність програмної системи полягає у наступному:

– перегляд поточного статусу датчиків в реальному часі;

– збір пакетів з історією значень датчиків за період часу;

– запуск аналізу даних з датчиків;

– ручне керування поворотом та тяговим зусиллям автомобіля;

– увімкнення стабілізації з використанням одного із збережених наборів параметрів.

Користувачам має бути доступна функція реєстрації у системі. Всі користувачі, які реєструються у системі, повинні автоматично належати до категорії. Після того як користувачем в режимі калібрації було створено набір параметрів, його можна використовувати для стабілізації і на інших пристроях.

Оскільки додаток орієнтований на операторів, що керують транспортним засобом в реальному часі, то він повинен мати максимально простий та зрозумілий інтерфейс користувача, не перевантажений графічними елементами, мати можливість користування без візуального зв’язку. Так оператор зможе сконцентруватися на автомобілі, яким він керує. До інтерфейсу користувача висуваються наступні вимоги:

– уніфікація розміщення керуючих елементів;
– адаптивність розмітки для пристроїв з різними розмірами екранів;
– підтвердження виконання критичних дій у системі;
– мінімізація кількості полів, що заповнюються;
– сповіщення користувача про виконання ним некоректних дій;
– точність налаштування параметрів;
– мінімально можливий час відгуку системи [6].

Так як дана програмна система напряму взаємодіє з реальним фізичним пристроєм, то надзвичайно важливим аспектом постає безпека програмної системи, яка має забезпечуватися такими засобами, як формування різних рівнів доступу із переліком допустимих прав та авторизацією користувачів.

Передача команд із мобільного пристрою до радіокерованого автомобіля має вестися в реальному часі і мати щонайменші затримки за повний цикл. Тому вкрай необхідно мінімізувати сторонні чинники, які сповільнюють передачу, максимально ефективно використати програмні та апаратні можливості обох підсистем, врахувати можливість виникнення транспортних помилок в каналі передачі даних та надати для користувача безшовний спосіб обробки та відновлення процесу передачі після виникнення будь-яких помилок.

Тож під час створення даного програмного продукту, треба провести аналіз предметної області стабілізації руху автомобіля, спроектувати та розробити за допомогою обраних технологій та мов програмування комбіновану систему для стабілізації. Створений програмний продукт має бути зручним у використанні, надавати користувачу необхідну швидкість роботи та передачі даних.

3 ОПИС ПРИЙНЯТИХ ПРОЕКТНИХ РІШЕНЬ

3.1 Аналіз та UML-моделювання предметної області

Для ефективної розробки системи стабілізації треба провести аналіз того, які методи стабілізації взагалі існують і який функціонал допоможе втілити дані методи в програмній реалізації. Одним із таких способів є коригування керуючих сигналів, спираючись на дані, що були отримані з сенсорів за певний проміжок часу. Це дозволяє динамічно адаптувати систему до поточного стану дорожнього покриття, коефіцієнту зчеплення з дорогою та інших умов навколишнього середовища.

Складність такої системи полягає в тому, що важливо саме аналізувати не тільки поточне значення датчиків, а мати дані й про історію. Це дозволяє покращити коректність та ефективність роботи в цілому. Тобто алгоритм має неперервно працювати з пакетами даних, що перетинаються із використанням динамічного програмування для пришвидшення загального часу обробки даних в системі.

Сама по собі стабілізація – це дуже складний процес, який потребує дуже точного та детального налаштування параметрів, які мають широкий діапазон допустимих значень. Тобто спочатку треба поставити задачу, для яких умов необхідно вивести оптимальні параметри стабілізації, потім провести ряд випробовувань і на основі отриманих даних сформувати необхідний набір вхідних параметрів для системи. Тож для того, щоб максимально ефективно використовувати готові відкалібровані параметри системи, було створено декілька категорій користувачів. Це дозволило оптимізувати процес роботи з системою, провести більш детальне розподілення ролей та обов’язків при використанні системи на практиці. Всі дані будуть зберігатися в хмарному сховищі, тож будуть доступні на будь-якому пристрої якщо користувач має необхідні привілегії для користування цими даними.

Користувачі в режимі калібрації та збору даних мають змогу вручну керувати автомобілем через радіоканал, водночас зберігаючи в системі дані з усіх датчиків. В подальшому вони будуть аналізуватися алгоритмом і готовий

результат може бути збережено для практичного використання в системі стабілізації (див. рис. 3.1).

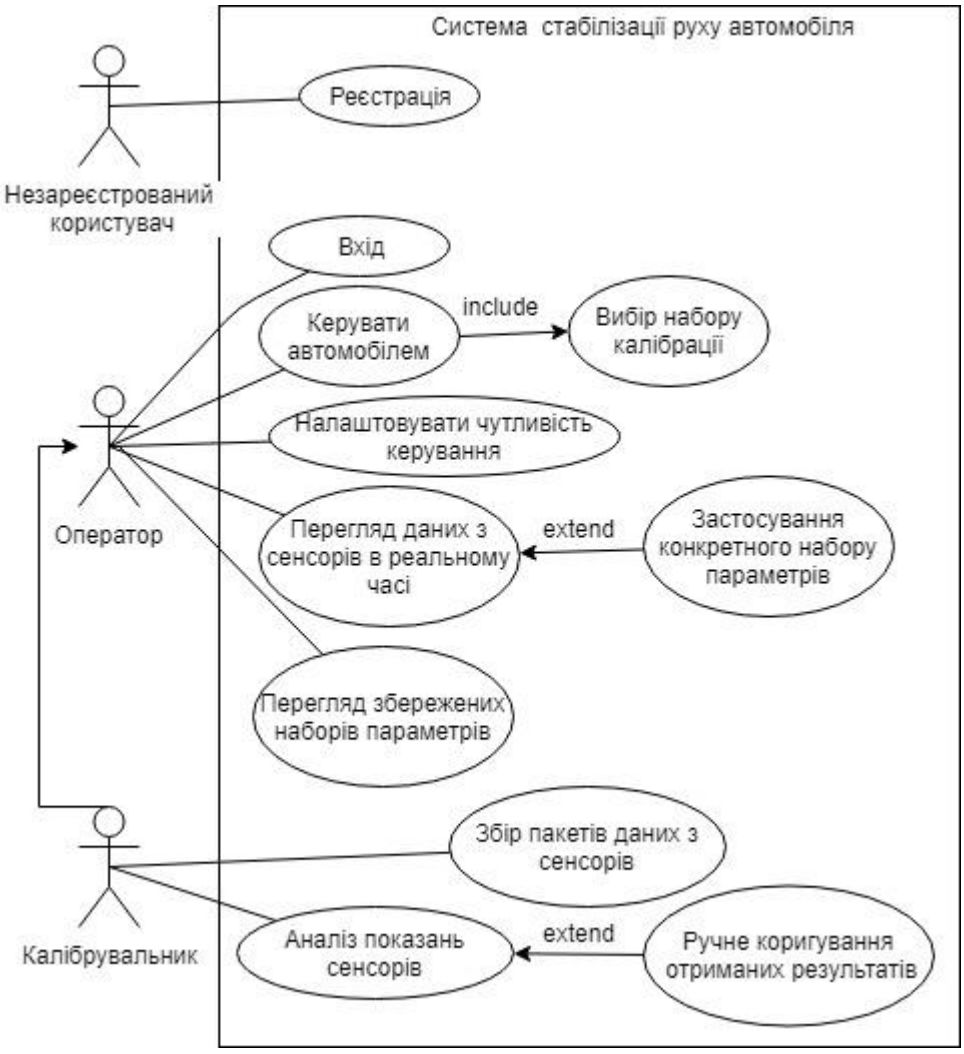

Рисунок 3.1 – Діаграма прецедентів

Створена діаграма прецедентів надає можливість виконати аналіз основних функціональних вимог до системи, візуалізувати її з точки зору різних категорій користувачів (ролей) та побачити функціонал, який доступний для тих чи інших рівнів користувачів. З діаграми можна зробити висновок, що незареєстрований користувач може лише створити собі новий акаунт, тобто зареєструватися. Після входу він має доступ до ряду функцій керування, збору та аналізу даних, налаштувань додатка.

3.2 Аналіз існуючих методів стабілізації

Стабілізація, нехай то автоматизована, чи ручна, залежить від типу середовища, поверхні, характеристик шасі, габаритів та ваги вантажу який перевозиться, обладнання, тощо. Зазвичай системи для стабілізації руху радіокерованого автомобіля працюють шляхом коригування поворотного приводу. Система використовує гіроскоп для визначення поточного напрямку руху та фізичного положення автомобіля в просторі. На основі цих даних програмна частина виконує коригування вхідного параметру повороту передніх коліс відповідно до отриманих результатів. Цей підхід є дієвим, проте він не враховує факт різниці кутової швидкості різних колісних осей [2].

Поворотність автомобіля – це властивість збільшувати, зберігати чи зменшувати кривизну траєкторії встановленого руху при збільшенні бокового прискорення. Розрізняють надмірну, недостатню та нейтральну поворотність [3]. Дані типи можна математично описати на основі кута уведення передньої осі δ_F, задньої δ_R та бокового прискорення j_Y.

Надмірна поворотність характеризує автомобіль, який збільшує кривизну траєкторії зі збільшенням бокового прискорення і описується наступною формулою:

$$\frac{\partial(\delta_F - \delta_R)}{\partial j_Y} < 0,$$

де δ_F, δ_R – зміщення осей автомобіля;

j_Y – бокове прискорення.

Нейтральна поворотність виникає коли автомобіль не змінює кривизну своєї траєкторії руху при збільшенні бокового прискорення (див. рис. 3.3). Такий тип поворотності виникає зазвичай при русі на малій швидкості і вважається ідеальним, так як траєкторія руху являється повністю передбачуваною і не вимагає жодної корекції, описується наступною формулою:

$$\frac{\partial(\delta_F - \delta_R)}{\partial j_Y} = 0,$$

де δ_F, δ_R – зміщення осей автомобіля;

j_Y – бокове прискорення.

Надмірна поворотність характеризує автомобіль, який зменшує кривизну траєкторії зі збільшенням бокового прискорення і описується наступною формулою:

$$\frac{\partial(\delta_F - \delta_R)}{\partial j_Y} > 0,$$

де δ_F, δ_R – зміщення осей автомобіля;

j_Y – бокове прискорення.

Визначивши поточний стан руху автомобіля, відносну швидкість v_r та його положення на основі даних з гіроскопа G_y, можна вирахувати компенсаційний кут повороту коліс S_c за наступною формулою:

$$S_c = \frac{G_y * k}{v_r},$$

де S_c – компенсаційний кут;

G_y – значення гіроскопа по осі y;

k – коефіцієнт впливу;

v_r – відносна лінійна швидкість.

Для налаштування та калібрації системи під реальні умови використання, введено коефіцієнт k, що дозволяє змінювати інтенсивність компенсації. Також слід відзначити, що чим більша лінійна швидкість пересування, тим меншим має

бути кут коригування. Даний підхід задовольняє вимогам системи адже дозволяє в реальному часі проводити корекцію керуючих параметрів системи.

3.3 Розробка архітектури застосунку

Найскладнішою частиною даної системи з точки зору внутрішніх міжкомпонентних зв’язків являється мобільний додаток для керування. Він повинен мати змогу комунікації з хмарним сервісом, отримання вхідних даних керування від користувача, обробки цих даних та надсилання команд до радіокерованого автомобіля в реальному часі. Тому надзвичайно важливо побудувати архітектуру таким чином, щоб зменшити зв’язність компонентів між собою. Одним із варіантів вирішення даної задачі є використання архітектурного підходу VIPER (див. рис. 3.2).

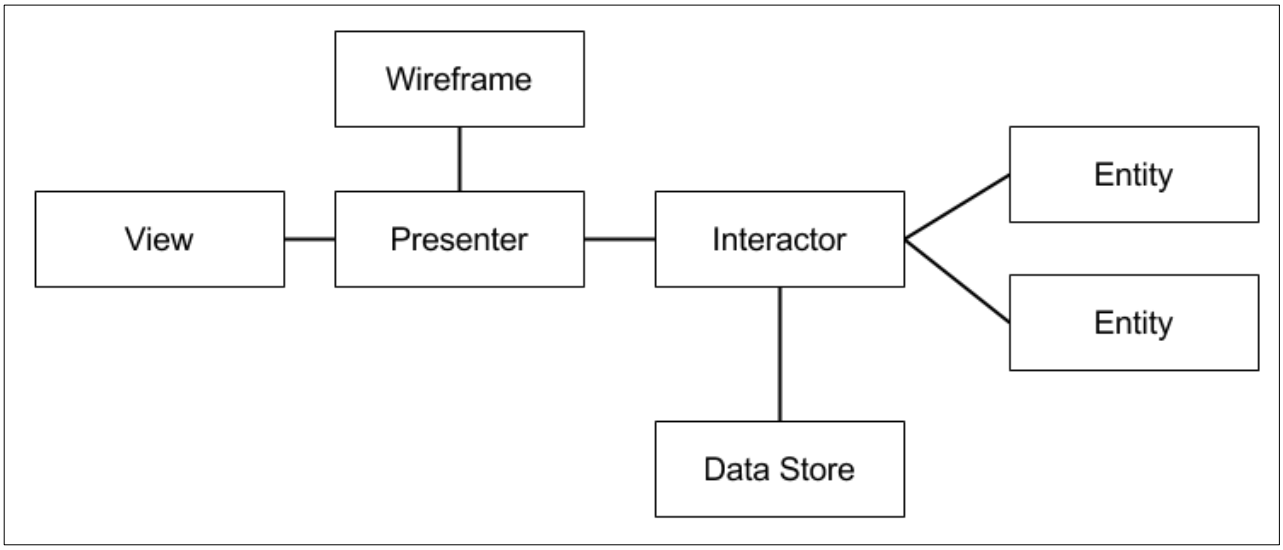

Рисунок 3.2 – діаграма архітектури мобільного додатку

Саме такий підхід має ряд переваг у порівнянні з іншими архітектурними схемами. Він надає змогу максимально ефективно розділяти зони відповідальності всередині системи, як на різних шарах, так і всередині одного. Наприклад, у кожного екрану є своє представлення, яке відповідає за відображення елементів на дисплеї мобільного пристрою, обробку подій взаємодії з користувачем.

Потім ці дані та події надсилаються до представника. Він являється проміжним організатором роботи між представленням та бізнес логікою додатка. За допомогою каркасу виконується навігація між різними екранами, передача необхідних вхідних параметрів та обробка отриманого результату на попередньому екрані.

Взаємодійники ж в свою чергу, являються місцем, де описана бізнес логіка застосунку. Вони не мають прив’язки до конкретної платформи, тож у разі використання кросплатформеної мови програмування, можуть бути повторно використані в іншому додатку, наприклад для мобільної операційної системи iOS.

Сховища даних виконують більш низькорівневі операції, які зазвичай специфічні для конкретної роботи. Наприклад, робота з локальною базою даних, відправка та обробка мережевих запитів, робота з файлами, доступ до системних функцій платформи тощо. Саме завдяки тому, що платформо залежна логіка винесена до сховищ даних, вони ізолюють шар бізнес логіки і роблять його повністю незалежним, надають можливість повторного використання окремих компонентів в інших подібних додатках.

Також важливою перевагою саме такої архітектури являється можливість модульного тестування окремих компонентів системи, так як можна провести ізоляцію логіки і використати заглушки замість компонентів, від яких залежить перший.

Діаграма розгортування — це діаграма UML, на якій зображені окремі компоненти складної системи, відображені зв’язки, напрямки та протоколи комунікації між ними. Це дозволяє у короткий проміжок часу визначити залежності між компонентами, асоціювати програмні артефакти із фактичними місцями розсортування та виконання програмного коду (див. рис. 3.3).

Дивлячись на цю діаграму, можна одразу визначити, що система має 3 основних складових: контролер автомобіля, мобільний пристрій для керування та хмарний сервіс.

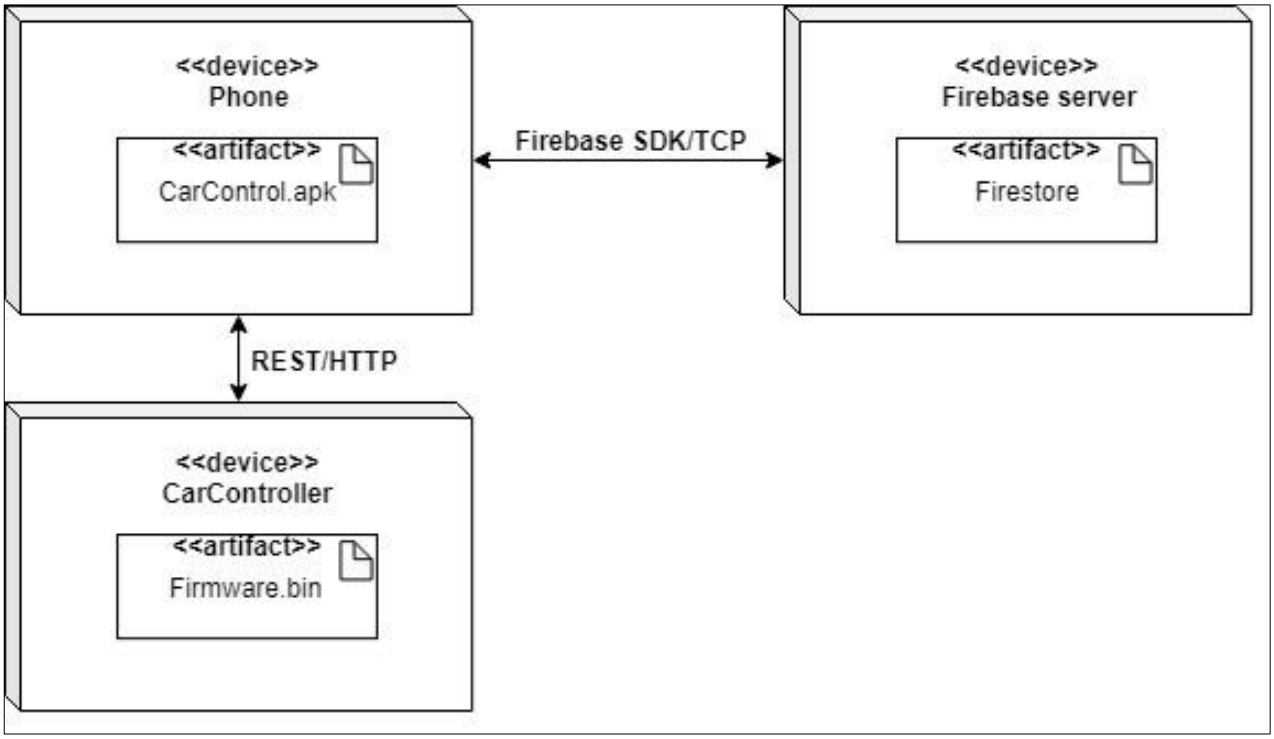

Рисунок 3.3 – Діаграма розгортання

Контролер автомобіля напряму не взаємодіє з хмарним сервісом, а вся комунікація та зв’язок виконується виключно завдяки мобільному пристрою, так як саме він має фізичну можливість спілкуватися з обома іншими компонентами одночасно. Також він виконує роль адаптаційного перетворювача даних. Це дозволяє зменшити зв’язність між окремими високорівневими компонентами даної програмної системи і забезпечити незалежність їхнього функціонування.

При проектуванні загальної архітектури програмного продукту треба також враховувати подальший розвиток після першої ітерації розробки а також можливість тестування окремих компонентів. Так, наприклад, при розробці застосунку для керування автомобілем, корисно мати можливість випробувати його функціонал без реального автомобіля. Програмний код на різних стадіях розробки, а особливо на початку, може бути дещо нестабільним, мати помилки в логіці роботи. Тож ці помилки повинні мати мінімальний вплив на реальні пристрої.

Також з розширенням системи треба залишати зручність у доданні нового функціоналу, підтримувати код у належному стані. Цьому можуть допомогти

використання певної структури модулів, загальні правила створення нових компонентів, їх структура та взаємозв’язки.

Завдяки використанню загальних протоколів спілкування HTTP, клієнти системи не являються прив’язаними до платформи, забезпечується зручний перегляд реальних запитів, є можливості для задання власної структури запитів та відповідей. Всередині тіла запиту можна передавати довільні дані, наприклад відформатовані у JSON форматі.

Тож, як висновок можна сказати, що такі обрані програмні рішення дозволяють зменшити зв’язність компонентів як всередині підсистем, так і між окремими підсистемами, проте водночас забезпечують надійні канали спілкування між ними. Це збільшує гнучкість системи, робить її подальший розвиток більш простим за рахунок використання незалежної між компонентної архітектури.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Для збереження всієї інформації системи стабілізації руху автомобіля була використана NoSQL система Firebase Firestore, яка є однією із найбільш популярних систем для ефективного збереження даних в довільному форматі. Дане рішення має наступні переваги:

- швидкість роботи;
- простота використання;
- підтримка довільної структури даних;
- підтримка трігерів для мобільного клієнта;
- можливість зберігання великої кількості даних;
- можливість одночасного підключення до бази багатьох користувачів.

Firebase Firestore призначений для роботи в мережі і може бути доступна через Internet. Таким чином, до сервера Firestore можна підключатися з будь-якої точки земної кулі через програму-клієнт, встановлену на пристрої користувача. Firebase Firestore не підтримує реляційні бази даних, тобто дані не представлені у вигляді таблиць, розбитих на рядки і стовпці, на перетині яких знаходяться дані. Замість цього, всі дані зберігаються у вигляді колекції, елементом яких може бути інша колекція, або документ з набором полів.

На відміну від реляційних моделей зберігання та обробки даних, таке рішення не має класичних таблиць та зв’язків між ними, база даних для системи стабілізації має деревовидну ієрархічну структуру (див. рис. 4.1). Такий підхід дозволяє обійти одне з обмежень сховищ такого типу – складність використання комплексних запитів. Тож, для отримання інформації про користувача, достатньо знайти його ідентифікатор в загальному списку користувачів і потім перейти до необхідного запису. Також обране рішення дозволяє оптимізувати процес отримання даних шляхом явного вказування або виключення даних які необхідно отримати в рамках конкретної операції доступу.

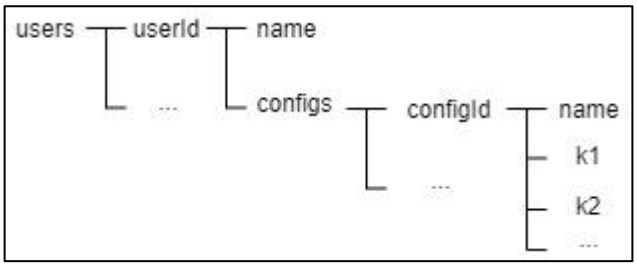

Рисунок 4.1 – Загальний вигляд структури даних NoSQL сховища

Таким чином, для системи стабілізації руху автомобіля було розроблено базу даних з використанням Firebase Firestore, яка забезпечує компроміс між широкими функціональними можливостями та простотою використання. Використання саме NoSQL рішень дозволило зробити структуру зберігання даних більш гнучкою, а доступ до даних швидшим. Проте так як даний підхід накладає деякі обмеження на операції, це було враховано в загальній архітектурі системи та при реалізації алгоритмів, що взаємодіють з даними. Зокрема, було використано індексування бази даних для оптимізації пошуку елементів за ідентифікатором та формування загального списку об’єктів.

4.1 Реалізація мобільного додатку

Реалізація мобільного додатку виконувалась в інтегрованому середовищі розробки Android Studio 3.5. Даний інструментарій надає змогу виконувати повний цикл розробки інтерфейсу, логіки програми, запуску та тестування. Така можливість є надзвичайно корисною в процесі розробки, адже дозволяє налаштовувати процес розробки таким чином, що статичний та динамічний аналіз коду проводиться постійно і своєчасно та заздалегідь виправляються потенційні помилки в коді. Також дане середовище розробки дозволяє швидко застосовувати внесені зміни як до логіки роботи, так і зовнішнього вигляду екранів.

Для створення зовнішнього вигляду екранів було використано верстку в форматі xml. Цей підхід являється загальноприйнятим в області розробки мобільних додатків для операційної системи Android. Фрагмент лістингу розмітки для головного екрану наведено на рисунку 4.2.

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView...>

    <TextView...>

    <View...>

    <de.nitri.gauge.Gauge...>

    <com.hakito.netcar.widget.RelativeTouchView...>

    <com.hakito.netcar.widget.RelativeTouchView...>

    <com.jjoe64.graphview.GraphView...>

    <ImageButton...>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Рисунок 4.2 – приклад розмітки головного екрану

Використання такого підходу дозволило відокремити структуру та налаштування компонентів на екрані від логіки застосунку. Також перевагою даного способу являється те, що складові компоненти представлені в ієрархічному вигляді, тож при використанні більш складних схем розміщення, компоненти не втрачають своєї семантики та належності до тих чи інших частин екрану.

Основними екранами являються екран налаштувань параметрів та головний екран керування роботизовано платформою. Екран налаштувань виконано в вигляді діалогового вікна, тож ці два екрани можуть бути одночасно активними і мати синхронізацію змін. Завдяки цьому покращиться досвід користування додатком та простота. Вони є тісно пов’язані, так як використовують одне й те ж сховище параметрів, проте для запобігання дуплікації коду та зменшення зв’язності було введено деякі абстракції, які можна побачити на діаграмі класів (див. рис. 4.3).

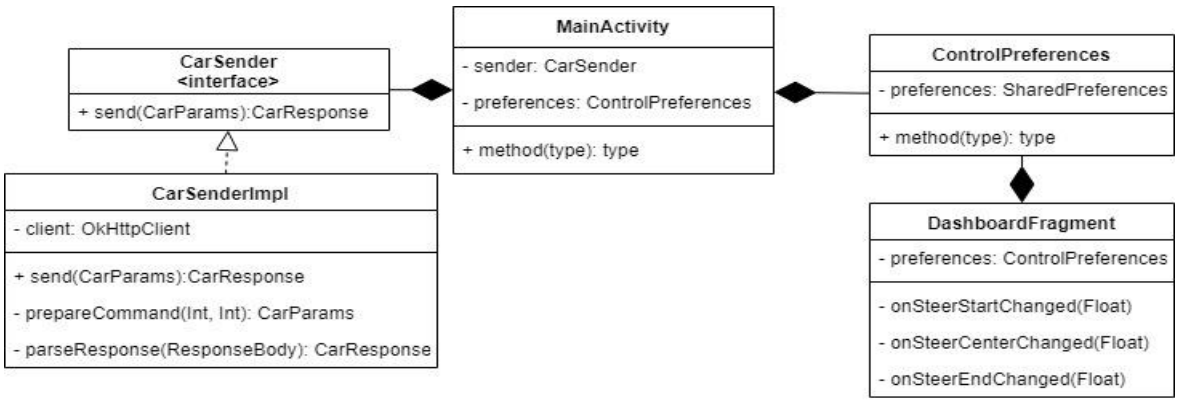

Рисунок 4.3 – діаграма класів

З даної діаграми видно, що основний клас CarSenderImpl, який відповідає за комунікацію з радіокерованим автомобілем використовується не напряму, а через абстрактний інтерфейс CarSender. Це дозволило, по-перше, надати можливість тестування окремих компонентів системи завдяки заміні на заглушки і, по-друге, зменшило зв’язність окремих компонентів всередині системи.

4.2 Інтерфейс та функціонал системи

Так як створена програмна система має підтримку різних ролей, то робота мобільного застосунку починається з екрану авторизації. Завдяки використанню хмарної платформи Google Firebase, додаток підтримує три способи авторизації: через соціальні мережі Google, Facebook та класичний метод з використанням електронної пошти та паролю. Всі ці способи схожі, проте мають різні вимоги до зовнішнього вигляду та самого процесу авторизації, тому стартовий екран дозволяє швидко обрати потрібний спосіб авторизації і продовжити процес на спеціалізованому екрані, специфічному для кожної платформи окремо (див. рис. 4.4).

Після авторизації користувачеві доступний головний екран контролю транспортним засобом. Більшу його частину займають компоненти для безпосереднього керування. Ліва частина відповідає за кут повороту передніх коліс. Права частина відповідає за напрям та потужність роботи рушійного двигуна задньої осі (див. рис. 4.5).

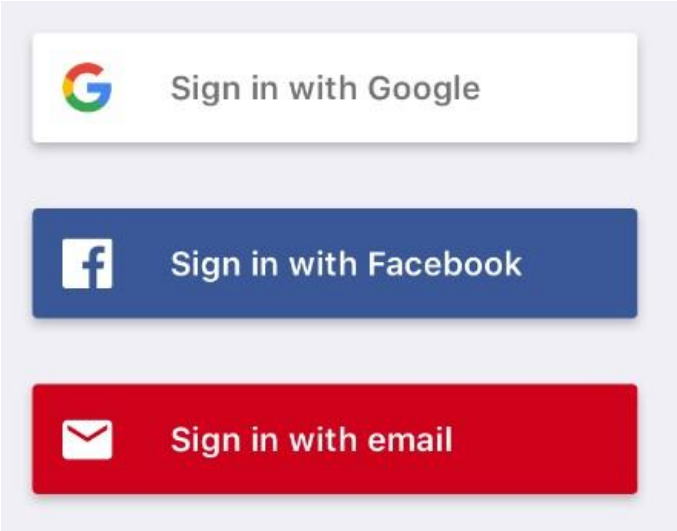

Рисунок 4.4 – Стартовий екран авторизації

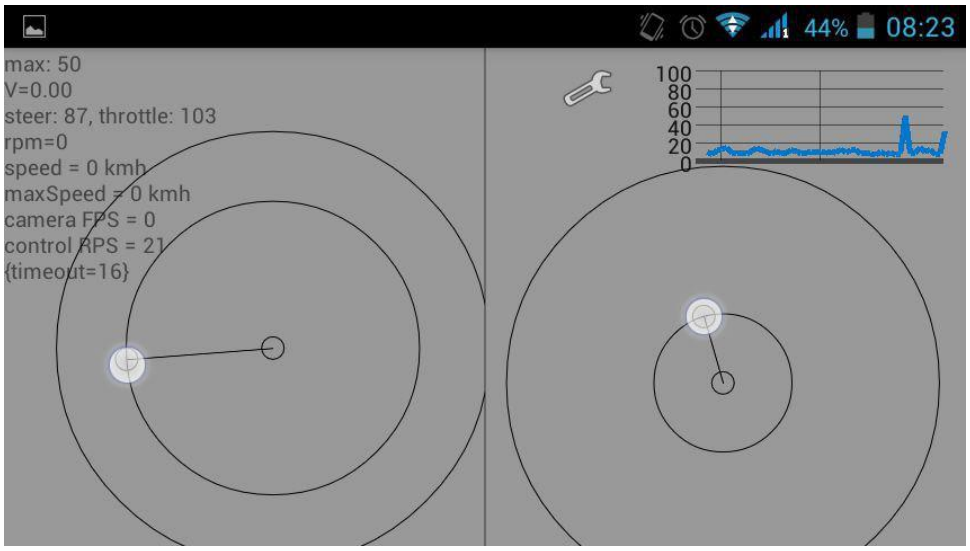

Рисунок 4.5 – головне вікно керування

Варто зауважити, що для гальмування інтерфейс не передбачує окремої кнопки. Дана функція виконана таким чином, що якщо користувач після пересування вперед за короткий проміжок часу намагається ввести двигун у положення реверсу, то спочатку виконається гальмування, і вже за наступної спроби увімкнеться реверс. Це було зроблено для того, щоб запобігти помилковому увімкненню реверсу на високій швидкості, а також спрощує процес взаємодії з інтерфейсом користувача.

Вцілому, даний екран був спроектований та реалізований таким чином, щоб користувач міг керувати транспортним засобом за відсутністю безпосереднього

зорового контакту з екраном: зони керування займають повний екран і нульовою точкою являється точка першого дотику до тієї чи іншої зони.

Також важливо бачити в реальному часі деякі параметри самого автомобіля, наприклад, поточну напругу батареї, лінійну швидкість пересування, середню частоту відправки команд. В правій частині екрану зображено графік для індикації повного часу відправки, застосування команди та отримання відповіді. Він слугує для того, щоб розуміти, на скільки стабільним є канал зв'язку. Також з даного екрану доступна кнопка налаштувань, яка надає можливість перейти на екран налаштувань керування (див. рис. 4.6).

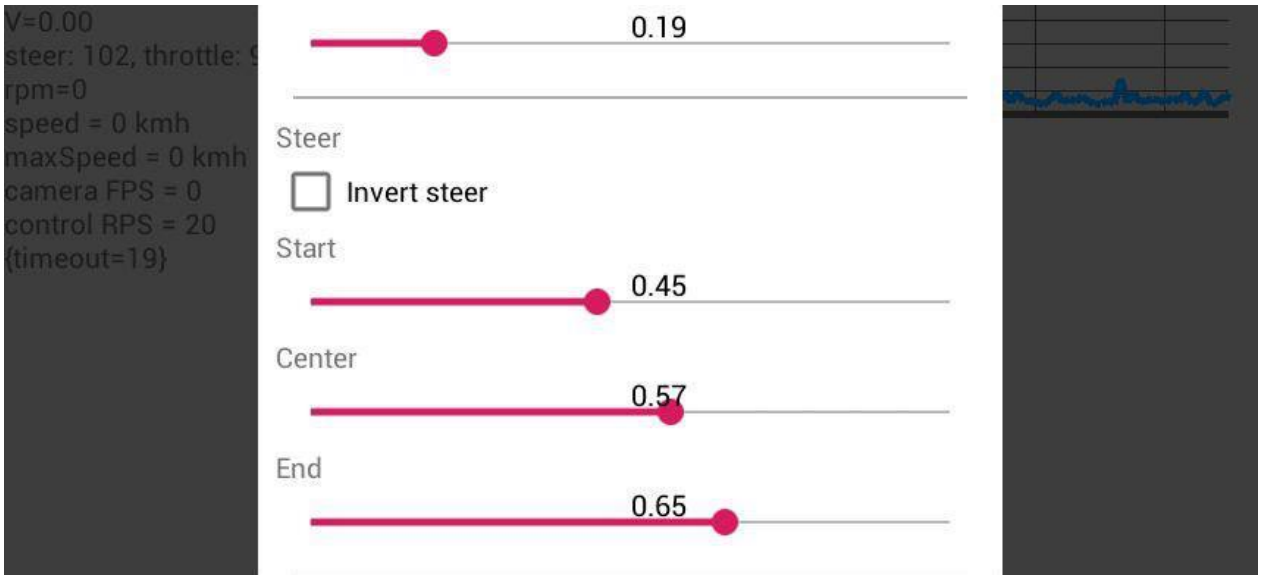

Рисунок 4.6 – вікно налаштувань

Дане вікно дозволяє детально виконати налаштування процесу керування автомобілем. Перш за все, за допомогою трьох повзунків можна виставити опорні точки. Центральний повзунок виставляє центральне положення коліс коли користувач не застосовує контроль повороту.

Інші два повзунки забезпечують обмеження крайніх точок повороту для лівої та правої сторони відповідно. Це дозволяє налаштувати діапазон та максимальний кут вивороту коліс в залежності від механічних та геометричних характеристик конкретного автомобіля.

За необхідності можна виконати інверсію значень керування. Це може бути необхідним у випадку якщо автомобіль має таку апаратну систему, що менше значення повертає колеса вправо, а більше – вліво. Впровадження такого параметра налаштувань допомагає зробити систему більш гнучкою та підтримувати більшу кількість різних транспортних платформ.

Отже як висновок можна сказати, що розроблений мобільний додаток задовольняє вимоги до програмної системи, надає повний контроль над автомобілем, дозволяє в реальному часі переглядати поточний статус та інші характеристики. Налаштування дозволяють більш точно відкалібрувати контрольні параметри під конкретний автомобіль. Вікно параметрів, яке виконане у вигляді діалогу, дозволяє швидко змінити певні параметри і побачити зміни в реальному часі.

5 АНАЛІЗ ДОСЛІДНИЦЬКОЇ ЕКСПЛУАТАЦІЇ ТА МОЖЛИВИХ ЗАСТОСУВАНЬ

5.1 Аналіз можливих застосувань

Розроблена система дозволяє коригувати вхідні параметри для радіокерованого автомобіля. Основними вимогами для апаратної частини являються використання коліс для руху, задній привід та передні поворотні колеса. Якщо розглянути сучасні роботизовані, радіокеровані машини, то велика частина з них підходить під ці вимоги, тож може бути використана і там.

У військовій галузі можна застосувати дану систему для радіокерованих мобільних бойових чи транспортувальних транспортних засобів. В цій сфері надзвичайно важливою є мобільність, стабільність та здатність швидко пересуватися в складних умовах бездоріжжя. І зазвичай такі платформи перевозять певну вагу, озброєння чи багаж, тож надзвичайно важливо забезпечити їх рухомість та працездатність.

В аеропортах дана система може слугувати допоміжною ланкою для використання мініатюрних роботизованих машин. Так як в зоні польотів заборонено використання дронів, то саме такі наземні колісні машини можуть стати в нагоді для збору різноманітних метрик, контролю злітно-посадочних смуг тощо.

Також є деякі галузі, де швидкість пересування не є настільки критичною, проте така система може застосовуватись як аваріний захисний механізм, що доможе зменшити негативні наслідки чи збитки у разі виникнення надзвичайної непередбачуваної ситуації. Наприклад, коли спеціалізоване дорожнє покриття вкрите маслянистою плямою чи виникли непередбачувані погодні умови (гололід, сніг). Система стабілізації допоможе запобігти зіткненню або сходженню з траєкторії руху і залишить час для того, щоб вжити всі необхідні заходи.

5.2 Опис тестування системи

Кожна система має пройти певний процес тестування перед впровадженням. Існує декілька різних способів та методологій перевірки програмних продуктів на валідність та коректність роботи.

Окремі частини підсистем були спроектовані таким чином, щоб забезпечити можливість модульного тестування. Такі тести покривають дуже вузьку частину функціоналу і націлені на найбільш оперативне та точне виявлення проблем після змін у програмному коді створюваного продукту. Так як мобільний додаток створено з використанням мови програмування Kotlin, можна використати спеціалізований та гнучкий інструмент для модульного тестування JUnit. Початково він створювався для тестування Java програм, проте Kotlin також створений на основі цієї мови програмування, виконується на тій самій віртуальній машині і має повну зворотну сумісність. Приклад модульного тесту приведено на рисунку 5.1.

```
@Test
fun buildUrl() {
    val params = CarParams( steer: 75,  throttle: 150)

    val url = sender.buildUrl(params).toString()

    assertEquals( expected: "http://192.168.4.1:81/car?steer=75&throttle=150", url)
}
```

Рисунок 5.1 – приклад модульного тесту

Наведений метод не являється частиною кінцевого користувацького додатку, проте з точки зору розробки входить в один і той же проект, тож має доступ до всіх компонентів системи. Це реалізовано за допомогою різних наборів програмного коду та спеціальних анотацій, що вказують системі які з методів являються тестами та можуть налаштовувати специфічні параметри.

Так як дана програмна система являється комплексною і середовище виконання дуже сильно впливає на вихідні результати роботи, то неможливо абсолютно повністю ізолювати її для по компонентного тестування. Тому було

проведено кілька етапів мануального тестування системи вцілому, в реальних умовах. В ході тестування були застосовані евристичні метрики для оцінки коректності роботи системи [11]. Це дозволило провести суб’єктивну перевірку коректності системи. Недоліком даного підходу являється відсутність чіткої формалізації метрик коректності отриманих результатів роботи даної системи.

Як висновок можна сказати, що на практиці важко знайти ідеальний метод тестування, адже кожен з них має свої обмеження та специфіку. Тому краще комбінувати різні підходи для різних частин та аспектів додатка. При проектуванні архітектури треба починати задумуватись над подальшим тестуванням, так як деякі методи перевірки потребують зміни або взаємодії з програмним кодом системи.

ВИСНОВКИ

В ході виконання даної атестаційної роботи були досліджені основні методи і алгоритми систем стабілізації руху автомобіля, можливість їх застосування при розробці практичної системи керування та стабілізації. На основі цього аналізу були обрані методи, які будуть використовуватися під час програмної реалізації системи.

Було створено математичну модель руху колісного транспортного засобу з урахуванням зовнішніх чинників. На основі створеної математичної моделі було розроблено схему бази даних, яка буде зберігати всю необхідну інформацію системи. Також було розроблено алгоритми стабілізації та визначення оптимальних контрольних параметрів. Була розроблена UML-модель для візуалізації функціоналу системи з точки зору різних категорій користувачів.

В ході розробки було проведено аналіз вимог до продукту та проектування архітектури системи в цілому. Серед технологій для розробки системи було обрано Google Firebase, ESP IDF, Kotlin, Coroutines, Android SDK.

Так як система має підтримувати можливість комунікації різних компонентів в реальному часі та мати мінімальні затримки, було вирішено використовувати комплексний апаратний модуль ESP32. Його використання дозволило створити зручну у використанні систему, яка в той же час, має прийнятні показники швидкодії, надає можливість реагувати на команди за мінімально короткий проміжок часу. Проте, все ще залишилась проблема саме з обробкою даних із сенсорів для стабілізації в реальному часі.

Так як обробка фактично дискретизована, необхідно враховувати певні затримки від самих датчиків та контролера. Частково ця проблема була вирішена завдяки використанню двохядерного мікропроцесору та відповідного набору розробки програмного забезпечення ESP IDF. Також обрана система дозволила використати свої певні апаратні оптимізації. Наприклад такі як апаратні переривання. Вони надали змогу максимізувати ефективність зчитування даних з датчиків кутової швидкості коліс.

При тестуванні роботи створеної програмної системи було виявлено, що вона має гарну стабільність обробки даних, надає змогу відсліджувати помилки та коректно відновлювати свою подальшу роботу. Тестовий зразок керуючого пристрою виконано на базі сучасної мобільної операційної системи Android, тож дана підсистема має широкий набір пристроїв для використання.

Створений алгоритм надає можливість стабілізації руху колісних роботів в умовах недостатнього щеплення з дорогою, нерівностей поверхні, перевезення важких вантажів, швидкого пересування тощо. Це дозволяє застосовувати програмну систему в широкому спектрі можливих задач. Починаючи з військових цілей і закінчуючи цивільними допоміжними системами для оптимізації чи спрощення процесів виробництва, транспортування, комунікації, спостереження тощо.

В подальшому планується програмна реалізація системи, яка дає можливість повноцінно контролювати безпілотні наземні транспортні засоби із застосуванням розроблених алгоритмів стабілізації. Дана система повинна підтримувати функціонал складання оптимального алгоритму та визначення вхідних параметрів стабілізації, які найкраще підходять для поточного стану системи без участі оператора, на основі попередньо отриманих статистичних даних про різні типи поверхонь та оточуючого середовища використання системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Michael Barr, Programming Embedded Systems in C and C++. США: O'Reilly, 1999. 179 с.

2. Ashok Kumar S Mastering Firebase for Android Development. Великобританія: Packt, 2018.380 с.

3. Шилдт Г. Java. Повне керівництво. Київ: Діалектика, 2016. 1488 с.

4. Material design guidelines. URL: https://material.io/guidelines/ (дата звернення: 13.10.2019).

5. Буч Г. Мова UML. Керівництво користувача. Москва: ДМК Прес, 2006. 248 с.

6. Ларичев О. Теорія і методи прийняття рішень. Москва: Логос, 2002. 392 с.

7. Окулов С. Динамічне програмування. Москва: Біном, 2014. 296 с.

8. Кормен Т., Лейзерсон Ч., Рівест Р., Штайн К. Алгоритми. Побудова і аналіз. Москва: Вільямс, 2016. 1328 с.

9. Csaba Szepesv`ari, Algorithms for Reinforcement Learning. США: Morgan & Claypool Publishers, 2009. 98 с.

10. Пирогов В. Інформаційні системи і БД: організація і проектування. Санкт-Петербург: БХВ-Петербург, 2009. 528 с.

11. Белоус Н.В., Шергин В.Л., Мирошниченко Т.А., Артеменко Д.И. Использование обобщенного полярного преобразования координат при распознавании прямых//Современные информационные и электронные технологии. 2014. №15. с. 44-45