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

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Інтернет-речей система автопілот для Smart city»

Виконав:
студент IV курсу, групи ІО-63
Левченко Костянтин Володимирович _____

Керівник:
Доцент кафедри ОТ, к.т.н.
Ткаченко Валентина Василіївна _____

Консультант з нормоконтролю:
Професор кафедри ОТ, д.т.н.
Сімоненко Валерій Павлович _____

Рецензент:

_____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій СТІРЕНКО

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Левченко Костянтину Володимировичу

1. Тема проєкту «Інтернет-речей система автопілот для Smart city», керівник проєкту Ткаченко Валентина Василівна, Доцент, кандидат технічних наук, затверджені наказом по університету від «07» травня 2020 р. № 1081-с

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту див. технічне завдання

4. Зміст пояснювальної записки дослідження предметної області, огляд існуючих рішень, визначення вимог і завдань для програмного продукту, вибір платформи та технології, обґрунтування оптимальності використання обраних інструментів для розробки, реалізація проєкту.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Сімоненко В.П.		

7. Дата видачі завдання _____

* Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломного проєкту.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Відмітки про виконання
1	*Затвердження теми роботи*	01.09.2019	виконано
2	*Вивчення та аналіз завдання*	02.09.2019-01.02.2020	виконано
3	*Розробка архітектури та загальної структури систем*	02.02.2020-04.03.2020	виконано
4	*Розробка структур окремих Підсистем*	05.03.2020-13.03.2020	виконано
5	*Програмна реалізація системи*	14.03.2020-13.04.2020	виконано
6	*Оформлення пояснювальної записки*	14.04.2020-17.05.2020	виконано
7	*Захист програмного продукту*		
8	*Передзахист*	26.05.2020	
9	*Захист*		

Студент Костянтин ЛЕВЧЕНКО

Керівник Валентина ТКАЧЕНКО

Анотація

У бакалаврській дипломній роботі реалізовано систему автопілот для Smart city, призначеної для забезпечення безпеки руху транспортних засобів без втручання пілота.

Програма дозволяє прокладати траєкторію руху транспортного засобу підєднаного до системи Smart city, а також надає можливість спілкуватися з іншими автомобілями в системі для можливості уникнення аварійних ситуацій. Програмний продукт був реалізований на мові C++ за допомогою бібліотеки Qt Automotive Suite у візуальному середовищі розробки Qt Creator.

Аннотация

В бакалаврской дипломной работе реализована система автопилот для Smart city, предназначенной для обеспечения безопасности движения транспортных средств без вмешательства пилота.

Программа позволяет прокладывать траекторию движения транспортного средства подключенного к системе Smart city, а также предоставляет возможность общаться с другими автомобилями в системе для возможности предотвращения аварийных ситуаций. Программный продукт был реализован на языке C ++ с помощью библиотеки Qt Automotive Suite в визуальной среде разработки Qt Creator.

Annotation

In this work for a Bachelor's Degree, implemented an autopilot system for Smart city, designed to ensure the safety of vehicles without pilot intervention.

The program allows you to pave the trajectory of the vehicle connected to the Smart city system, and also provides the opportunity to communicate with other cars in the system to avoid accidents. The software product was implemented in C ++ using the Qt Automotive Suite library in Qt Creator visual development environment.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	ІАЛЦ.467100.002 ТЗ	Система автопілот для Smart city.	4	
			Технічне завдання		
3	А4	ІАЛЦ.467100.003 ПЗ	Система автопілот для Smart city.	64	
			Пояснювальна записка		
4	А4	ІАЛЦ.467100.004 А1	Система автопілот для Smart city.		
			Лістинг програми		

Технічне завдання
до дипломного проекту

на тему: «Інтернет-речей система автопілот для Smart city»

Київ – 2020

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання розповсюджується на розробку системи автопілот для Smart city.

Область застосування: використання системи в навчальному процесі.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки служить завдання на виконання розробки системи автопілот для Smart city, затвердженою кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний Інститут ім. Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка системи автопілот для Smart city.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами для розробки служать науково-технічна література з комп’ютерних технологій, публікації в періодичних виданнях, публікації в Інтернеті за даним питанням.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- Можливість розгортання додатка на популярних операційних системах;
- Можливість налаштування швидкості руху;
- Задання маршруту руху транспортного засобу;
- Можливість додавання нового функціоналу;

5.2. Вимоги до програмного забезпечення

- Операційна система Ubuntu 20.04, Ubuntu 18.04, Ubuntu 16.04, Windows 10

5.3. Вимоги до апаратного забезпечення

- Комп’ютер на базі процесору Intel Pentium 3 і вище
- Оперативної пам’яті не менше 512 Мбайт
- Підключення до Інтернету

6. ЕТАПИ РОЗРОБКИ

	Дата
Затвердження теми роботи	01.09.2019
Вивчення та аналіз завдання	02.09.2019-01.02.2020
Розробка архітектури та загальної структури систем	02.02.2020-04.03.2020
Розробка структур окремих підсистем	05.03.2020-13.03.2020
Програмна реалізація системи	14.03.2020-13.04.2020
Оформлення пояснювальної записки	14.04.2020-17.05.2020
Захист програмного продукту	
Передзахист	26.05.2020
Захист	

Пояснювальна записка дипломного проекту

на тему: «Інтернет-речей система автопілот для Smart city»

Київ - 2020 року

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ ТА СКОРОЧЕНЬ

GPS	Система глобального позиціювання (англ. Global Positioning System)
OCX	Сегмент оперативного управління (англ. Operational Control Segment)
API	Прикладний програмний інтерфейс(англ. Application Programming Interface, API)
OS	Операційна система (англ. operating system)
VM	Віртуальна машина (англ.Virtual Machine)
JVM	Віртуальна машина (англ. JAVA Java virtual Machine)
LTS	Довгострокова підтримка системи (англ. Long-term support)
SNAT	Статичне перетворення мережових адрес (англ.Static Network Address Translation)
GKE	Google Kubernetes двигун (англ. Google Kubernetes Engine)
NAT	Перетворення мережевих адрес (англ. Network Address Translation)
BSM	
WLAN	Локальна мережа (англ. Wireless Local Area Network)
CAM	Повідомлення про кооперативну обізнаність (англ. Cooperative Awareness Message)
DENM	Децентралізоване повідомлення про навколишнє середовище (англ. Decentralized Environmental Notification Message)
VPC	Віртуальна приватна хмара (англ. Virtual Private Cloud)
JOSM	Редактор мап OpenStreetMap на мові Java (англ. Java OpenStreetMap)
OSM	Сервіс мап (англ.OpenStreetMap)

ВСТУП

На сьогодні розробляється досить багато систем для автопілотування апаратів. Кожна з яких відрізняється одна від одної, але мають багато спільного, системи автопілоту розробляються переважно для зменшення впливу людського фактору під час керування автомобіля, що призводить до збільшення ДТП та забруднення навколишнього середовища та фінансових збитків.[1]

Актуальність теми

Через втрати та травмування населення під час ДТП та фінансові збитки в економіці країн, автопілот дозволить значно знизити ці показники. Також більшість компаній перевізників бажають розробити автопілот для транспорту який перевозить вантажі, щоб зменшити витрати на водіїв та їх збиткові помилки.

Мета і задачі дослідження

Мета роботи – розробка системи автопілоту автомобіля який буде базуватися на QT Automotive Suite.

Щоб досягти поставленої мети було сформовано наступні задачі:

- Проаналізувати вже існуючі автомобільні автопілоти;
- Розробити модель роботи автопілоту та реалізувати його програмну частину;
- Розробити систему відображення перешкод на мапі;
- Провести віртуальне тестування на різних моделях;
- Отримані результати тестування проаналізувати та дослідити;
- В разі отримання задовільних результатів провести тестування в умовах доріг загального користування з допомогою водія;

П*рактичне значення*

Автопілот дозволить більш надійно та безпечно переміщатися на автомобілі, також зменшить кількість порушень правил дорожнього руху, що сприятиме зменшенню кількості ДТП та травмувань на дорозі, а також зменшить вплив людського фактору під час керування автомобіля що призведе до зменшення забруднення навколишнього середовища.

РОЗДІЛ 1
ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ СИСТЕМ АВТОПІЛОТУ

Автопілот автомобіля – це система, яка може керувати автомобілем за допомогою водія або автономно. Автопілот має системи керування, які здатні аналізувати сенсорні дані та розпізнавати транспортні засоби, що робить планування маршруту більш безпечним.

Програмне забезпечення керує всіма системами автомобіля а саме :

- Повертання керма;
- Зміна передачі;
- Прискоренням та сповільненням;

Сенсори збирають інформацію про оточуюче середовище, яка лягає в основу дій автопілоту.

Зазвичай встановлювані датчики :

- Камери;
- Системи глобального позиціонування (GPS);
- Датчики одометра;
- Гіростабілізатори;
- Радари;

1.1.Рівні автономності

Автопілот поділяється на рівні автономності:

– *Рівень 0.* Автономність відсутня

Водій виконую всю роботу [2];

– *Рівень 1.* «Helping out», «Допомога водію»

Система допомагає водію в керуванні автомобіля. Приклад: водій керує, а система зберігає задану швидкість або регулює потужність двигуна та управляє гальмом [2];

– *Рівень 2.* «Building confidence», «Часткова автономність»

Система повністю керує автомобілем, здійснює прискорення, гальмування та керування. Водій пильнує за роботою автопілоту та готовий втрутитися в будь який момент, якщо система не може правильно відреагувати [2];

– *Рівень 3.* «Taking control», «Умовна автономність»

Від водія не потрібна негайне втручання. Система сама реагує на екстреній ситуації, такі як екстрене гальмування. Водію потрібно втручатися на протязі обмеженого часу, який визначений виробником [2];

– *Рівень 4.* «Mind off», «Широка автономність»

Відрізняється від 3-го рівня автономності тим, що від водія не потрібно постійної уваги. Повне автоматичне керування здійснюється тільки в деяких просторових областях або деяких ситуацій таких, як затори [2];

– *Рівень 5.* « Adulthood», «Повна автономність»

Людського втручання не потребує [2].

1.2. Переваги та недоліки автопілоту

Плюси поділяються на декілька категорій:

Економічні

Значна мінімізація ДТП та людських жертв, з цього слідує зменшення на страхову медицину та медицину швидкого реагування. Зниження ціни на транспортування вантажів та населення. Зниження потреби в індивідуальному авто за рахунок автопілотних таксі. Підвищення ефективності користування доріг за рахунок дотримання правил дорожнього руху. Зміна на ринку праці, поступове зникнення професій: водій таксі, далекобійник і т.д. та поява нових, пов'язаних з розробкою та обслуговуванням автономного транспорту;

Соціальні

Економія часу, який водій витрачає на керування авто, дозволить зайняти більш важливими справами. Можливість самостійно пересуватися на власному автономному автомобілі для людей з обмеженими можливостями та неповнолітнім;

Інші переваги

Транспортування вантажів в небезпечних зонах, під час природних або техногенних катастроф або військових дій.

Недоліками є:

Людський фактор

Відсутність навичок самостійного керування в екстрених ситуаціях;

Загальна стандартизація

Також не можна залишати без уваги проблеми з впровадженням в масси. Одна зі складностей є стардантизація. Оскільки різни виробники мають різні технології автопілоту, необхідні загальноприйняті стандартию;

Законодавчі регулювання

Технологія автопілоту дуже швидко розвивається, але суперечливі правила відносно стандартів та законодавчого регулювання призупиняють цей процес. Конфлікти між стандартами та законами різних країн є тією перешкодою, яке потрібно уладнати, тільки тоді нові технології будуть сумісні не тільки локально, а й глобально.

1.3. Автопілоти провідних компаній

1.3.1. Waymo

Перед тим, як автомобілі Waymo виїжджають в будь яке місце, команда цієї компанії створює власний пробні тривимірні мапи, на яких виділяється така інформація, як дрожні профілі, бордюри, тротуарні, вказівники смуг, пішохідних переходів, світлофори, дорожні знаки та інші дорожні елементи. Датчики та програмне забезпечення постійно сканують об’єкти навколо автомобіля – пішоходів, велосипедистів, автомобілі,перешкоди – і постійно зчитують керування дорожнім рухом. Програмне забезпечення передбачає переміщення всього навколо автомобіля. Після програмне забезпечення використовує інформацію для прогнозування можливих траєкторій усіх учасників дорожнього руху, завдяки цьому програмне забезпечення визначає всі потрібні показники для безпечного просування по заданому маршруту [6];

1.3.2. Tesla

Сучасні автомобілі Tesla оснащені сучасним обладнанням, здатним сьогодні надати функції автопілоту, і повноцінні можливості самостійного водіння в майбутньому - завдяки оновленням програмного забезпечення, розробленим для покращення функціональності з часом. Вісім камер об'ємного звуку забезпечують 360 градусів видимості навколо автомобіля на відстані до 250 метрів. рис. 1.1.

Радар, спрямований вперед, з покращеною обробкою надає додаткові дані про світ на надмірній довжині хвилі, яку можна побачити через сильний дощ, туман, пил і навіть автомобіль попереду. Щоб використовувати набір камер таким потужним, нове обладнання представляє абсолютно новий і потужний набір інструментів для обробки зору, розроблений Tesla.

Рис.1.1 Покриття датчиків Tesla [4]

Побудований у глибокій нейромережі, Tesla Vision реконструює довкілля автомобіля з більшим рівнем надійності, ніж ті, які досягаються класичними методами обробки зору. Всі нові автомобілі Tesla мають необхідне в майбутньому обладнання для повноцінного керування

автомобілем практично за будь-яких обставин. Система розроблена таким чином, щоб можна було здійснювати поїздки на короткі та міжміські відстані, не вимагаючи дій особі на водійському місці [4].

1.3.3.General motors

В центрі можливостей самостійного водіння автомобіля - комп'ютери, які виконують функції, необхідні для розуміння світу навколо транспортного засобу та прийняття рішень щодо водіння, які безпечно перевозять пасажирів. Жодна технологія не робить роботу «мозку». Натомість комп’ютери використовують комбінацію систем, які безпечно працюють разом, включаючи рис.1.2. :

Рис.1.2 Схема зв'язку систем автопілоту General Motors [3]

Розглянемо три елемента з цієї схеми - сприйняття, планування та управління - щоб показати, як Cruise AV сприймає оточення та приймає рішення щодо водіння. У автомобілі, що керує автотранспортом, сприйняття «sees», використовуючи датчики для спостереження за його оточенням. Датчики подають інформацію на комп'ютер, який поєднує дані датчика з

картографічними даними високої чіткості для локалізації транспортного засобу. Сприйняття виявляє та класифікує об’єкти, визначає їх розташування та забезпечує їх швидкість та напрямок. Він будує тривимірну модель світу, яка відслідковує важливі об’єкти. Сприйняття також прогнозує майбутній рух об’єктів - пішоходи та вантажівки мають різні передбачувані рухи. Використовуючи тривимірну модель та дані карти, Сприйняття визначає вільний проїзний простір навколо транспортного засобу. Сприйняття визначає інші екологічні невизначеності. Наприклад, знаючи своє місцезнаходження, Сприйняття знає, де він повинен шукати рухомі об’єкти. Якщо його погляд заблоковано, сприйняття позначить цю область як невідому.

Якщо об'єкт важко помітити через дощу чи туману або тому, що він схований за вантажівкою, мозок комп'ютера це знає і відповідно підлаштовує прийняття рішень та ефективність. Це дає розумне прийняття рішень та функціонування, засноване як на тому, що датчики "sees" бачать , так і на тому, що може бути приховано від зору. Для виконання функцій сприйняття автомобіль має п'ять лідарів, 16 камер та 21 радар. Їх комбіновані дані забезпечують різноманітність сенсорів, що дозволяє сприйняттю бачити складні середовища. Поєднуючи усі дані це допомагає, наприклад, визначити пішоходів, типи транспортних засобів та деталі доріг, такі як лінії смуги руху, будівельні зони та вивіски. Додатковий набір датчиків дальньої дальності відслідковує швидкісні об'єкти, такі як зустрічні транспортні засоби, і датчики короткої дальності надають детальну інформацію про рухомі предмети поблизу транспортного засобу, такі як пішоходи та велосипеди [3].

1.3.4.Volvo

Volvo разом з Veoneer керує компанією Zenuity, котра розробляє програмне забезпечення для систем допомоги водію так і для автономних автомобілів. Технологія підключеного дорожнього перегляду (CRV) забезпечує розширений огляд дороги, попередньо виявляючи критичні події заздалегідь у здатності водія, що підвищує безпеку, зручність та досвід

водіння. CRV забезпечує корисний зворотний зв'язок з-за горизонту, пов'язані функції, що працюють над переповненою та часто оновленою інформацією про подальшу дорогу - все це підтримується функцією пошуку зондів, що постійно забезпечує оновлення карт на основі максимально можливого рівня довіри. Використовуючи технології лідар з камерами та датчиками навколо автомобіля, а також програмного забезпечення для автономного водіння для обробки даних з усіх датчиків, ми можемо отримати повністю безпечний та автономний автомобіль. На новому рівні, автономному автомобілю не потрібна буде допомога водія, це дозволить підвищити комфорт пересування та безпеку пасажирів [5].

ВИСНОВКИ ДО РОЗДІЛУ 1

1. Аналіз існуючих систем автопілоту показав, що всі розробники використовують однакове обладнання, але різних виробників для зчитування середовища, що знаходиться навколо авто. Програмне забезпечення у кожної компанії різне та різні підходи:

- Waymo, робить натиск на попередньо отриману інформацію з місць, які будуть проїжджати автомобіль, та попередньо побудовану тривимірну мапу з усіма перешкодами та дорожніми знаками;
- Tesla, робить натиск на далекозорість свого обладнання, а саме камер об'ємного звуку, які можуть зчитувати інформацію на дистанції 250 метрів;
- General Motors, робить натиск на програмне забезпечення;
- Volvo, робить натиск на два напрямки, програмного забезпечення та обладнання. Використовуючи зонди для отримання максимально достовірної інформації про стан дороги.

РОЗДІЛ 2
ПРОЕКТУВАННЯ

2.1. Опис предметної області

Автопілот - це система, що використовується для управління траєкторією повітряного судна, морського судна або космічного корабля без необхідності постійного ручного управління з боку оператора людини. Основною задачею цієї системи є зменшення впливу людського фактору під час керування автомобіля. Сюди входять: зменшення кількості ДТП та забруднення навколишнього середовища а також зменшення витрат компаній перевізників за рахунок повного контролю електронікою над транспортом під час руху.[1]

Система складається з сервера для розрахунку траєкторії і вхідних даних, клієнта, сервера віртуалізації мап а також сервера для їх взаємодії. Для організації ефективної роботи системи існують спеціальні архітектури, що дозволяють можливість подальшого розширення функціоналу.

2.1.1.V2V

Автомобільні спеціальні мережі V2V - створюються, застосовуючи принципи мобільних спеціальних мереж (MANET) - спонтанного створення бездротової мережі мобільних пристроїв - до сфери транспортних засобів. Про VANET вперше згадували та впроваджували у 2001 р. У додатках "Автомобільний спеціальний мобільний зв'язок та мережа", де можна формувати мережі та передавати інформацію серед автомобілів. Було показано, що в VANET будуть співіснувати архітектури комунікацій між автомобілями та автомобілями та на дорогах, щоб забезпечити безпеку дорожнього руху, навігацію та інші послуги на дорогах. VANET - це ключова частина системи інтелектуальних транспортних систем (ITC). Іноді VANET називають інтелектуальними транспортними мережами [7].

Рис. 2.1 Приклад V2V взаємодії [8]

2.1.2. V2V приклади застосування

VANET підтримують широкий спектр застосувань - від простого одноразового поширення інформації, наприклад, спільних повідомлень про обізнаність (CAM) до багаторазового поширення повідомлень на величезних відстанях. Більшість проблем, що цікавлять мобільні спеціальні мережі (MANET), цікавлять VANET, але деталі відрізняються. Замість того, щоб рухатися навмання, транспортні засоби, як правило, рухаються організовано. Взаємодія з придорожньою технікою також може бути охарактеризована досить точно. І, нарешті, більшість транспортних засобів обмежені у своєму діапазоні руху, наприклад, обмежуючись рухатись по брукованій дорозі.

Приклади застосувань VANET:

Електронні гальмівні вогні, які дозволяють водієві (або автономному автомобілю чи вантажівці) реагувати на розбиття транспортних засобів, навіть якщо вони можуть бути затемненими (наприклад, іншими транспортними засобами).

Взвод, який дозволяє транспортним засобам уважно (до декількох дюймів) слідувати за провідним транспортним засобом, бездротово отримуючи інформацію про прискорення та рульове управління, утворюючи таким чином електронні сполучені "дорожні поїзди".

Інформаційні системи про дорожній рух, які використовують зв'язок VANET для надання поточних звітів про перешкоди для супутникової навігаційної системи транспортного засобу

Служби надзвичайних ситуацій на автомобільному транспорті - де комунікації VANET, мережі VANET, попередження про безпеку дорожнього руху та поширення інформації про стан використовуються для зменшення затримок та пришвидшення аварійно-рятувальних операцій для врятування життя постраждалих.

Служби на дорозі - передбачається також, що майбутня транспортна магістраль буде "керованою інформацією" або "бездротовою підтримкою". VANET можуть допомогти рекламувати послуги (магазини, автозаправні станції, ресторани тощо) для водія і навіть надсилати сповіщення про будь-який продаж, який відбувається в цей момент [7].

2.1.3. V2V технологія та стандарти

Технологія:

VANET можуть використовувати будь-яку технологію бездротової мережі. Найбільш відомими є радіотехнології короткого діапазону , такі як WLAN (або стандартний Wi-Fi, або ZigBee). Крім того, стільникові технології або LTE можуть використовуватися для VANET. Найновішою технологією цієї бездротової мережі є світлодіодне спілкування [VLC] (інфрачервона передача та прийом).

Стандарти:

Основна стандартизація пакетів протоколів VANET відбувається в США, Європі та Японії, що відповідає їхньому домінуванню в автомобільній промисловості.

Стек протоколу WAVE призначений для забезпечення багатоканальної роботи (навіть для транспортних засобів, оснащених лише одним радіо), безпеки та легких протоколів прикладного рівня. В межах комунікаційного товариства IEEE існує Технічний підкомітет з питань транспортних мереж та

додатків телематики (VNTA). Статут цього комітету полягає в активному сприянні технічній діяльності в галузі автомобільних мереж, комунікацій V2V, V2R та V2I, стандартів, безпеки дорожнього руху та транспортних засобів, моніторингу руху в режимі реального часу, технологій управління перехрестями, майбутніх програм телематики та Послуги на базі ITC [7].

2.1.4. Технологія V2D

Зв'язок між автомобілем та пристроєм (V2D) - це особливий тип автомобільної комунікаційної системи, який полягає в обміні інформацією між транспортним засобом та будь-яким електронним пристроєм, який може бути підключений до самого транспортного засобу.

Зростаюча тенденція розвитку мобільних додатків для нашого повсякденного використання в кінцевому рахунку увійшла і в автомобільний сектор. Підключення автомобіля до мобільних додатків має великий потенціал запропонувати кращий досвід водіння, надаючи інформацію про навколишні транспортні засоби та інфраструктуру та значно спрощуючи взаємодію між автомобілем та його водієм. Той факт, що додатки можуть значно підвищити безпеку водіння, привернув увагу користувачів автомобілів і спричинив збільшення кількості нових додатків, розроблених спеціально для автомобільної галузі. Ця тенденція має такий великий вплив, що зараз виробники починають розробляти автомобілі, дбаючи про їх взаємодію з мобільними телефонами. Наприклад, починаючи з 2017 року Volvo збирається продавати автомобілі без ключів, завдяки додатку, який дає можливість відкривати та запускати транспортний засіб дистанційно. Іншим сектором, який міг би отримати вигоду від цієї технології, є спільне використання автомобілів [9].

2.1.5. Технологія V2G

Автомобіль до сітки (V2G) описує систему, в якій є електричні транспортні засоби, так і електричні машини для акумуляторів (BEV), гібриди

підключення (PHEV) або електричні транспортні засоби з водневими паливними елементами (FCEV), які спілкуються з електромережою шляхом повернення електроенергії в мережу. Можливості зберігання V2G можуть дозволити електромережам зберігати та передавати електроенергію, вироблену з відновлюваних джерел енергії, таких як сонячна та вітряна, з виходом, який коливається в залежності від погоди та часу доби [10].

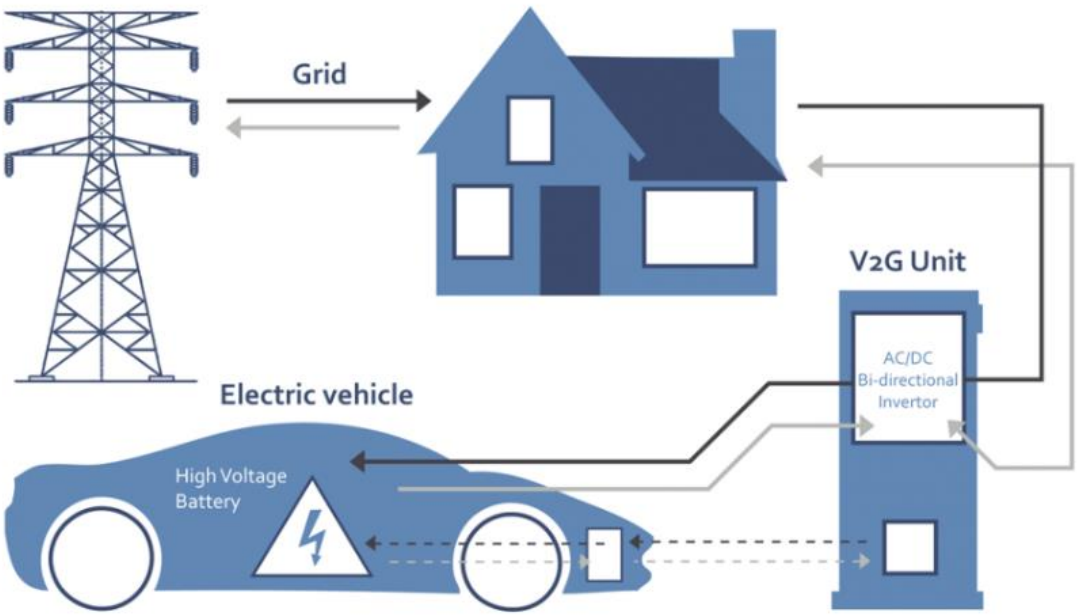


Рис. 2.2 Приклад V2G взаємодії [11]

V2G можна застосовувати для різних транспортних засобів, тобто електричних транспортних засобів, що підключаються до (BEV і PHEV) сітки. Оскільки в будь-який момент часу 95 відсотків автомобілів припарковані, акумулятори в електромобілях можуть використовуватися для передачі електроенергії від машини до електророзподільної мережі та назад. У звіті 2015 року про потенційний заробіток, пов'язаний з V2G, встановлено, що за належної регуляторної підтримки власники транспортних засобів можуть заробляти $ 454, 394 та 318 доларів на рік залежно від того, чи був їх середній щоденний проїзд 32, 64 або 97 км.

Акумулятори мають обмежену кількість циклів зарядки, а також термін зберігання, тому використання транспортних засобів для зберігання сітки може вплинути на довговічність акумулятора. Дослідження, які циклічно

використовують батареї два та більше разів на день, показали значне зниження ємності та значно скоротили термін експлуатації. Однак ємність акумулятора - це комплексна функція таких факторів, як хімія акумулятора, швидкість заряду і розряду, температура, стан заряду та вік. Більшість досліджень із повільною швидкістю скидання показують лише кілька відсотків додаткової деградації, тоді як одне дослідження припустимо доказала, що використання транспортних засобів для зберігання в сітці може покращити довговічність.

Іноді модуляція зарядки парку електромобілів агрегатором для надання послуг до електромережі, але без фактичного електричного потоку від транспортних засобів до електромережі називається однонаправленную V2G, на відміну від двонаправленої V2G, що загалом обговорюється в цій статті [10].

2.1.6. Принцип роботи V2G

Вирівнювання пікового навантаження:

Концепція дозволяє автомобілям V2G забезпечувати потужність, щоб допомогти збалансувати навантаження за допомогою зарядки вночі, коли попит низький та відправлення потужності назад до сітки, коли попит великий. Пікове вирівнювання навантаження може дати нові можливості для комунальних служб надавати послуги регулювання (підтримуючи стабільність напруги та частоти) та задовольняти раптові вимоги до електроенергії. Ці послуги в пов’язані з "розумними лічильниками" вони дозволять автомобілям V2G віддавати потужність в електромережу, а взамін отримувати грошові винагороди залежно від того, яка кількість енергії буде віддана в електромережу. У своєму нинішньому розвитку було запропоновано, що таке використання електричних транспортних засобів може захищати відновлювані джерела енергії, наприклад, енергію вітру, наприклад, зберігаючи надлишки енергії, виробленої у вітряні періоди, і віддаючи її в електромережу під час періодів високого навантаження, таким чином ефективно стабілізуючи переривчастістю вітрової енергії. Деякі розглядають

це застосування технології «автомобіль до мережі» як підхід, щоб допомогти відновлюваній енергії стати базовою технологією електроенергії.

Запропоновано, що комунальним підприємствам не доведеться будувати стільки електростанцій природного газу і вугілля для задоволення пікового попиту або як страховий поліс від відключення електроенергії. Оскільки попит можна виміряти локально за допомогою простого вимірювання частоти, при необхідності можна забезпечити динамічне вирівнювання навантаження [10].

Резервне живлення:

Сучасні електромобілі, як правило, можуть зберігати в своїх акумуляторах більше, ніж середня добова потреба в енергії. Навіть без можливостей вироблення газу PHEV такий транспортний засіб може використовуватися для аварійного живлення протягом декількох днів (наприклад, освітлення, побутова техніка тощо). Це був би приклад передачі автомобіля до дому (V2H). Як такі вони можуть розглядатися як додаткова технологія для переривчастих відновлюваних джерел енергії, таких як вітрові і сонячні електричні. Транспортні засоби з водневими паливними елементами (FCV) з цистернами, що містять до 5,6 кг водню, можуть доставити електроенергії більше ніж 90 кВт на рік [10].

2.1.7. Типи V2G

- Односпрямований V2G або V1G

Багато переваг V2G в масштабі сітки можна досягти за допомогою однонаправленого V2G, також відомого як V1G або "розумна зарядка". Каліфорнійський незалежний системний оператор (CAISO) визначає V1G як "однонаправлене керування послугами зарядки" та визначає чотири рівні інтерфейсу транспортного засобу та сітки (VGI). який охоплює всі способи надання електромережами таких мереж:

1. Односпрямований потік потужності (V1G) з одним ресурсом та об'єднаними діючими особами.

2. V1G із сукупними ресурсами.

- Двонаправлений потік потужності (V2G)

V1G передбачає зміну часу або швидкості, з якої заряджається електромобіль, щоб забезпечити допоміжні послуги до електромережі, тоді як V2G також включає зворотний потік електроенергії. V1G включає такі додатки, як транспортні засоби з тимчасовим зарядом, які потрібно заряджати в середині дня для поглинання надлишкової генерації сонячної енергії, або зміна швидкості заряду електричних транспортних засобів для надання послуг з частотного реагування або послуг з балансування навантаження.

V1G може бути найкращим варіантом почати інтегрувати електромережі як керовані навантаження в електричну мережу через технічні проблеми, що існують зараз щодо доцільності V2G. V2G вимагає спеціалізованого обладнання (особливо двонаправлених інверторів), має досить високі втрати та обмежену ефективність у зворотному напрямку, і може сприяти деградації батареї ЕВ через збільшення пропускної здатності енергії. Крім того, доходи від V2G в пілотному проекті SCE були меншими, ніж витрати на адміністрування проекту [16], що вказує на те, що V2G все ще має шляхи пройти, перш ніж бути економічно здійсненним [10].

- Двосторонній локальний V2G (V2H, V2B, V2X)

Автомобіль до будівництва (V2B) або автомобіль до всього (V2X), як правило, не впливають безпосередньо на роботу сітки, але створюють баланс у локальному середовищі. Електричний транспортний засіб використовується в якості житлового резервного джерела живлення в періоди відключення електроенергії або для збільшення власного споживання енергії, виробленої на місці (уникнення оплати за потреби).

На відміну від більш зрілих рішень V1G, V2X ще не досяг розгортання на ринку, крім Японії, де комерційні рішення V2H були доступні з 2012 року як резервне рішення у разі відключення електроенергії.

- Двонаправлений V2G

За допомогою V2G електромобілі можуть бути обладнані, щоб фактично подавати електроенергію в мережу. Оператор комунальної або передавальної системи може бути готовий купувати енергію у споживачів у періоди пікового попиту або використовувати ємність батареї EV для надання додаткових послуг, таких як балансування та регулювання частоти, включаючи первинне регулювання частоти та вторинний резерв. Таким чином, вважається, що V2G у більшості застосувань має більш високу потенційну комерційну цінність, ніж V2B або V2H [10].

2.1.8. Технологія V2X

Комунікація автомобіля до всього (V2X) - це передача інформації з транспортного засобу до будь-якої системи, яка може вплинути на транспортний засіб, і навпаки. Це автомобільна система зв'язку, яка включає інші більш специфічні типи зв'язку, як V2I (транспортна-інфраструктура), V2N (транспортний засіб до мережі), V2V (від транспортного засобу до транспортного засобу), V2P (автомобіль-пішохід) , V2D (автомобіль до пристрою) та V2G (автомобіль до мережі).

Існує два типи технології зв'язку V2X, залежно від основної технології, яка використовується на основі WLAN та на клітинній основі [12].

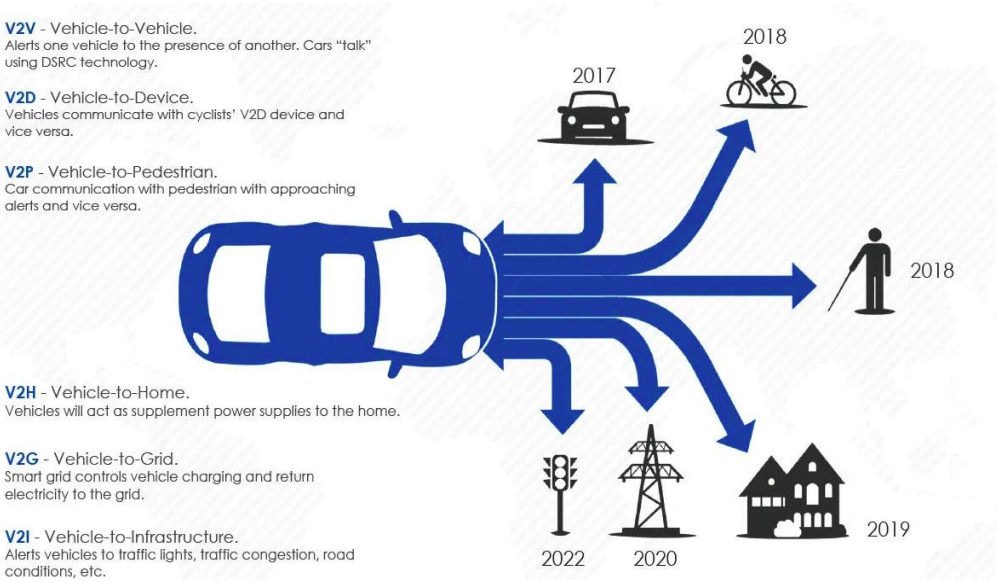


Рис. 2.3 Приклад V2X взаємодії [13]

Стандартизація V2X на основі WLAN замінює стандарти V2X на стільникових системах. Ця технологія називається виділеною короткою дальністю зв'язку (DSRC). DSRC використовує базовий радіозв'язок, передбачений 802.11p.

2.1.9. V2X принцип роботи та використання

- **802.11p (DSRC):**

Оригінальний зв'язок V2X використовує технологію WLAN і працює безпосередньо між транспортними засобами та транспортними засобами (V2V) та інфраструктурою дорожнього руху (V2I), які утворюють автомобільну спеціальну мережу, оскільки два передавача V2X потрапляють в діапазон один одного. Отже, для зв'язку транспортних засобів не потрібна жодна інфраструктура зв'язку, що є ключовим для забезпечення безпеки у віддалених або малорозвинених районах. WLAN особливо добре підходить для зв'язку V2X через низьку затримку. Він передає повідомлення, відомі як Кооперативні повідомлення про поінформованість (CAM) або Основне повідомлення про безпеку (BSM), і децентралізовані повідомлення про навколишнє середовище (DENM). Інші повідомлення, пов’язані з дорожньою інфраструктурою, - це повідомлення про фазу та час (SPAT), повідомлення про інформацію про транспортний засіб (IVI) та повідомлення про запит на обслуговування (SRM). Обсяг даних цих повідомлень дуже низький. Радіотехнологія є частиною сімейства стандартів WLAN IEEE 802.11 і в США відома як бездротовий доступ у транспортних умовах (WAVE), а в Європі як ITS-G5. Для доповнення режиму прямого зв'язку транспортні засоби можуть бути обладнані традиційними технологіями стільникового зв'язку, що підтримують послуги на базі V2N. Це розширення з V2N було досягнуто в Європі за допомогою платформи C-ITS із стільниковими системами та системами мовлення (TMC / DAB +).[12]

- **3GPP (C-V2X)**

Більш недавній зв'язок V2X використовує стільникові мережі і називається стільниковим V2X (або C-V2X) для диференціації його від V2X на базі WLAN. Існувало декілька галузевих організацій, таких як Автомобільна асоціація 5G (5GAA), яка рекламує C-V2X завдяки його перевагам перед V2X на базі WLAN (не враховуючи при цьому недоліки). C-V2X спочатку визначається як LTE у версії 3GPP випуску 14 і призначений для роботи в кількох режимах:

- Пристрій-пристрій (V2V або V2I) та
- Пристрій до мережі (V2N).

У версії 15 версії 3GPP функціональність V2X розширена для підтримки 5G. C-V2X включає підтримку як прямого зв'язку між транспортними засобами (V2V), так і традиційного зв'язку на основі стільникової мережі. Крім того, C-V2X забезпечує шлях міграції до систем та послуг на базі 5G, що передбачає несумісність та більш високі витрати порівняно з рішеннями на базі 4G [12].

Прямий зв'язок між автомобілем та іншими пристроями (V2V, V2I) використовує так званий інтерфейс PC5. PC5 посилається на опорну точку, де користувальницьке обладнання (UE), тобто мобільний телефон, безпосередньо спілкується з іншим UE по прямому каналу. У цьому випадку зв’язок із базовою станцією не потрібен. На системному архітектурному рівні служба близькості (ProSe) - це особливість, яка визначає архітектуру прямого зв'язку між UE. У специфікаціях 3GPP RAN "побічна посилання" - це термінологія для позначення прямого зв'язку через PC5. Інтерфейс PC5 спочатку був визначений для задоволення потреб у критичній комунікації для спільноти громадської безпеки (Public Safety-LTE або PS-LTE) у випуску 13.

Мотивація критично важливої комунікації полягала в тому, щоб дозволити правоохоронним органам у разі надзвичайні ситуації використовувати зв'язок LTE, навіть якщо інфраструктура недоступна, наприклад, сценарій стихійного лиха. У випуску 14 далі використання інтерфейсу PC5 було розширено для задоволення різних потреб на ринку,

таких як комунікація за допомогою таких пристроїв, як smartwatch. У C-V2X інтерфейс PC5 повторно застосовується для прямого зв'язку в V2V та V2I.

Зв'язок Cellular V2X в режимі 4 покладається на розподілену схему розподілу ресурсів, а саме на основі зондування напівстійкого планування, яке планує радіо ресурси в автономному режимі на кожному користувальницькому обладнанні (UE).

Додаток до прямого зв'язку через PC5, C-V2X також дозволяє пристрою C-V2X використовувати мережеве мережеве з'єднання традиційним чином через інтерфейс Uu. Uu посилається на логічний інтерфейс між UE та базовою станцією. Зазвичай це називається автомобілем до мережі (V2N). V2N є унікальним випадком використання для C-V2X і не існує в V2X на базі 802.11p, враховуючи, що останній підтримує лише прямий зв'язок. Однак, подібно до V2X на базі WLAN, також у випадку C-V2X потрібні два радіозв'язку, щоб мати змогу одночасно спілкуватися через інтерфейс PC5 з сусідніми станціями та через інтерфейс UU з мережею.

У той час як 3GPP визначає функції транспортування даних, які дозволяють V2X, він не включає V2X семантичний контент, але пропонує використовувати стандарти ITS-G5, такі як CAM, DENM, BSM тощо для функцій транспортування даних 3GPP V2X.

Використання:

Завдяки миттєвому спілкуванню V2X дозволяє застосовувати такі додатки щодо безпеки дорожнього руху, як:

- Попередження про зіткнення
- Попередження про зміну смуги руху / попередження про сліпе місце
- Попередження про аварійне гальмування
- Перетин перехрестя

2.2. Визначення вимог і завдань

Основними функціями системи є:

1. Зміна маршрут руху;
2. Притримування автомобілем заданого маршруту руху;
3. Прискорення та сповільнення руху;
4. Візуалізація панелі приладів;
5. Візуалізація мап;

Основними вимогами до сервісу є:

1. Можливість розгортання додатка на популярних операційних системах;
2. Можливість налаштування швидкості руху;
3. Задання маршруту руху транспортного засобу;
4. Можливість додавання нового функціоналу;

2.3. Опис функціоналу додатку

Система орієнтована для використання у системі освіти для ознайомлення студентів із напрямком Embedded розробки.

Функціонал для користувачів:

1. Можливість швидко розгорнути середу розробки;
2. Можливість розгортання додатка на популярних операційних системах;
3. Доступ до документації по розробці;
4. Можливість швидкого запуску;
5. Можливість мануально компілювання системи;
6. Низькі вимогу до аппаратного обладнення;
7. Можливість вести розробку без доступу до інтернета;
8. Організація розробки группами.

2.4. Розробка підходу для реалізації додатку

Головна задача системи - можливість розгортання системи без залежності від потужності апаратної системи та операційної системи. Тому для розробки системи використовується бібліотеки Cmake, Qt Automotive Suite та такі технології як V2V, V2G, V2X, а також системи з відкритим кодом Docker-engine та OpenStreetMap.

2.5. Docker

Docker Engine - це технологія контейнерної роботи з відкритим кодом для створення та контейнеризації ваших програм. Docker Engine діє як програма клієнт-сервер із:

- Сервер із тривалим daemon-процесом dockerd.
- API, які задають інтерфейси, які програми можуть використовувати для спілкування та інструктажу демона Docker.
- Док-сервер клієнтського інтерфейсу командного рядка (CLI).

CLI використовує API Docker для управління або взаємодії з демоном Docker за допомогою сценаріїв або прямих команд CLI. Багато інших програм Docker використовують основні API та CLI. Демон створює та керує об'єктами Docker, такими як зображення, контейнери, мережі та томи [14].

2.5.1. Docker архітектура

Docker використовує архітектуру клієнт-сервер. Клієнт Docker спілкується з демоном Docker, який відповідальний за запуск та розповсюдження ваших контейнерів Docker. Клієнт Docker і демон можуть працювати в одній і тій же системі, або ви можете підключити клієнт Docker до віддаленого демона Docker. Клієнт Docker і демон демонструють зв'язок за допомогою REST API, через сокети UNIX або мережевий інтерфейс [14].

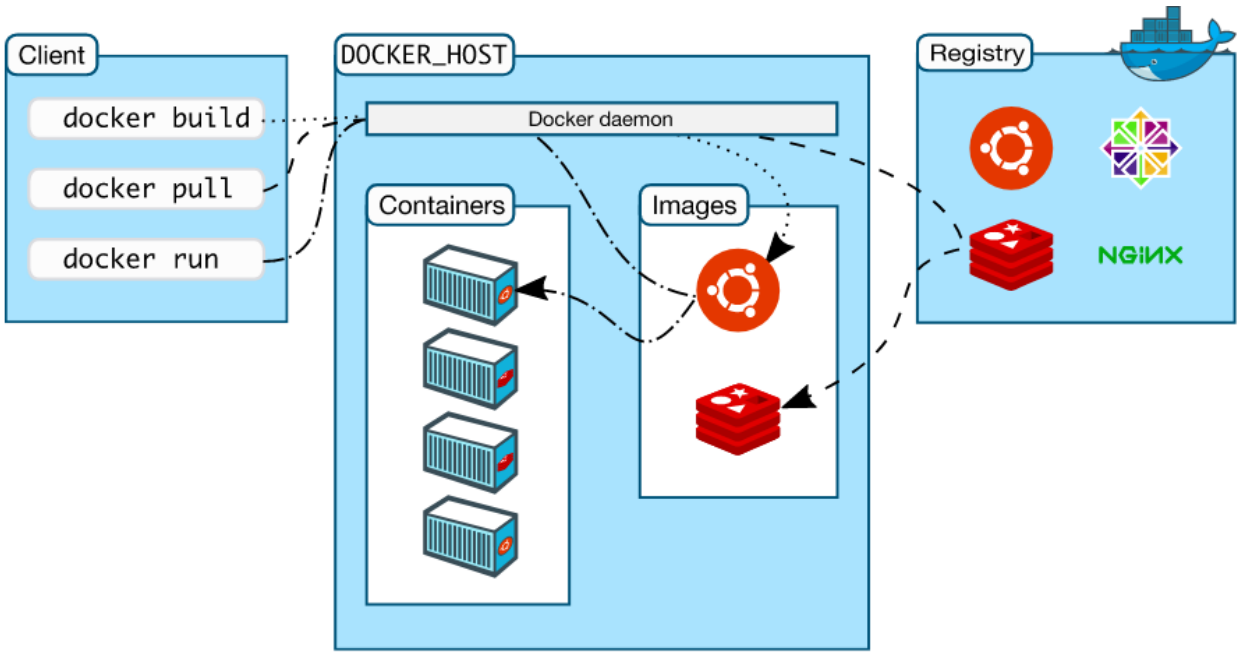


Рис. 2.4 Архітектура Докера [15]

- **Демон Докер (Docker daemon):**

Демон Docker (dockerd) слухає запити Docker API та керує об'єктами Docker, такими як зображення, контейнери, мережі та томи. Демон може також спілкуватися з іншими демонами, щоб керувати Docker [14].

- **Клієнт Докер (Docker client):**

Клієнт Docker (докер) є основним способом взаємодії багатьох користувачів Docker з Docker. Під час використання таких команд, як docker run, клієнт відправляє ці команди dockerd, який виконує їх. Команда docker використовує API Docker. Клієнт Docker може спілкуватися з більш ніж одним демоном [14].

- **Докерні регістри (Docker registries):**

Реєстр Docker зберігає Docker image. Docker Hub - це публічний реєстр, який може використовувати кожен, і Docker налаштований шукати зображення на Docker Hub за замовчуванням. Ви навіть можете запустити власний приватний реєстр. Якщо ви використовуєте Докер-центр обробки даних (DDC), він включає в себе Докер-довірений реєстр (DTR) [14].

Коли ви використовуєте команди "docker" або "docker run", потрібні зображення витягуються з налаштованого реєстру. Коли ви використовуєте команду push docker, ваше зображення висувається в налаштований реєстр.

- **Зображення (Images)**

Зображення - шаблон, доступний лише для читання, з інструкціями щодо створення контейнера Docker. Часто зображення базується на іншому зображенні, з деякими додатковими налаштуваннями. Наприклад, ви можете створити зображення, яке базується на зображенні ubuntu, але встановлює веб-сервер Apache та ваше додаток, а також деталі конфігурації, необхідні для запуску програми [14].

Ви можете створювати власні зображення або використовувати лише ті, які створені іншими та опубліковані в реєстрі. Щоб створити власний образ, ви створите Dockerfile з простим синтаксисом для визначення кроків, необхідних для створення зображення та запуску його. Кожна інструкція в Dockerfile створює шар у зображенні. Коли ви змінюєте Dockerfile і відновлюють зображення, відновлюються лише ті шари, які змінилися. Це частина того, що робить зображення настільки легкими, маленькими та швидкими в порівнянні з іншими технологіями віртуалізації [14].

- **Контейнери (Containers):**

Контейнер - це запущений екземпляр зображення. Ви можете створити, запустити, зупинити, перемістити або видалити контейнер за допомогою API Docker або CLI. Ви можете підключити контейнер до однієї або декількох мереж, приєднати до нього сховище або навіть створити нове зображення залежно від його поточного стану [14].

За замовчуванням контейнер відносно добре відокремлений від інших контейнерів та його хост-машини. Ви можете керувати тим, наскільки ізольованою є мережа, сховище чи інші базові підсистеми контейнера від інших контейнерів або від хост-машини.

Контейнер визначається його зображенням, а також будь-якими параметрами конфігурації, які ви надаєте йому під час створення або запуску.

Коли контейнер видалений, будь-які зміни його стану, які не зберігаються в постійному сховищі, зникають [14].

```
klevchenko@klevchenko:~$ docker run -it ubuntu bash
```

Рис. 2.5 Приклад виконання команди докера

2.6. Docker-compose

Compose - це інструмент для визначення та запуску багато контейнерних програм Docker. За допомогою Compose ви використовуєте файл YAML для налаштування служб своєї програми. Потім за допомогою однієї команди ви створюєте та запускаєте всі служби зі своєї конфігурації. Compose призначений для роботи у будь-яких середовищах: виробництво, постановка, розробка, тестування, а також робочі процеси CI.

Використання Compose - це трьохетапний процес:

Визначте середовище програми за допомогою Dockerfile, щоб воно могло бути відтворене в будь-якому місці.

- Визначте сервіси, які складають вашу програму в docker-compose.yml, щоб вони могли працювати разом в ізольованому середовищі.
- Запустіть докер-композицію і Compose запускає весь ваш додаток.

```
version: '3.7'

services:

# kubehub db service
  db:
    image: mariadb:10
    ports:
      - '3306:3306'
# kubehub backend service
  kubehub-back:
    build:
      context: ./
      dockerfile: Dockerfile
    volumes:
      - .:/app
    ports:
      - "8000:8000"
    depends_on:
      - "db
```

Рис.2.6 Приклад docker-compose.yml

2.7. Qt Automotive Suite

Qt Automotive Suite - це сукупність програмних компонентів та інструментів, які дають змогу розробляти системи внутрішньо-транспортних розваг (IVI). Побудований поверх пропозиції Qt для створення пристроїв, Qt Automotive Suite включає вбудоване середовище розробки з додатковими інструментами для швидкого розвитку інтерфейсу, а також інтроспекцією та налагодженням програми. Модуль Qt GENIVI Extras допомагає зробити ваш продукт на базі Qt Automotive Suite сумісним з архітектурою автомобільної платформи GENIVI. Qt Automotive Suite складається з декількох компонентів, побудованих на Qt і Qt Creator [16].

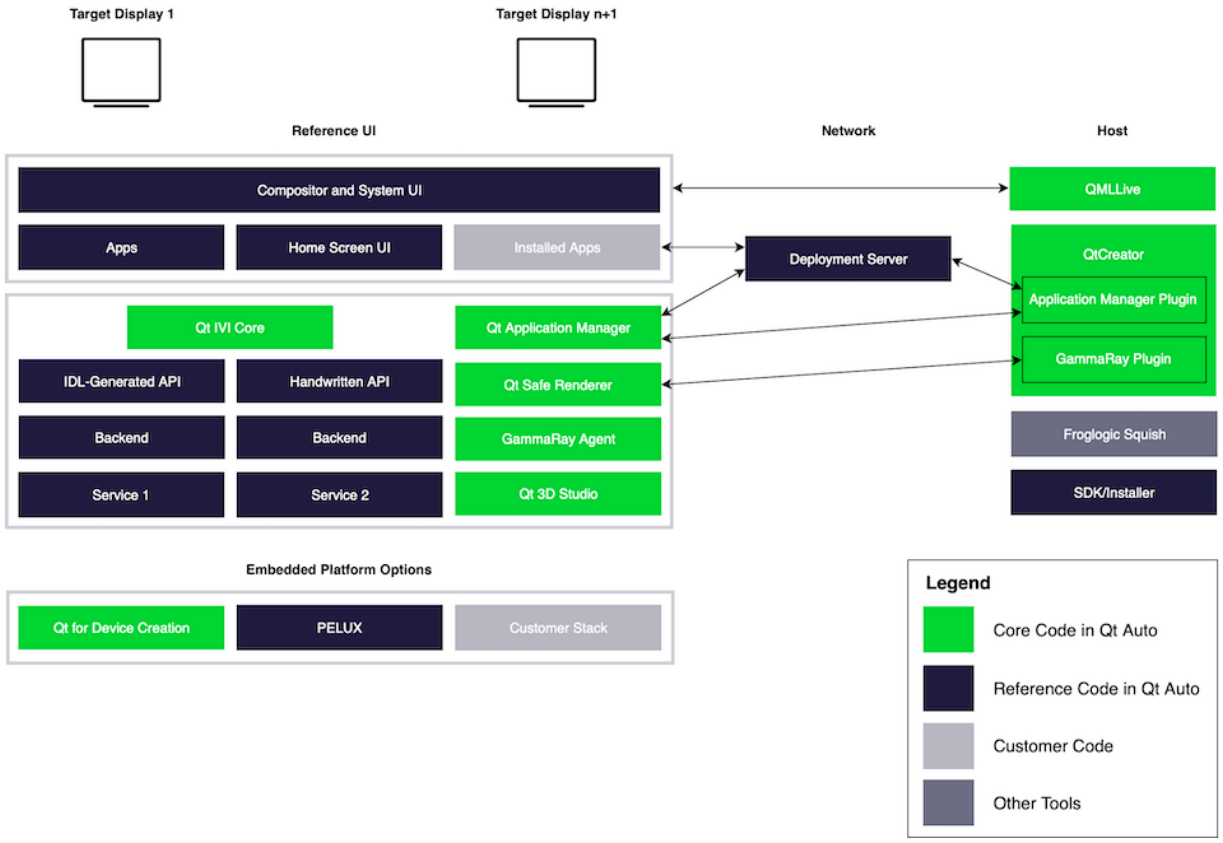


Рис.2.7 Структура Qt Automotive Suite [16]

2.7.1. Qt Automotive Suite компоненти та інструменти

Компоненти:

- **Qt Manager Manager** забезпечує основу для вбудованої системи з декількома додатками. Цей модуль забезпечує керування життєвим

циклом додатків. Він надає API для окремого впровадження інтерфейсу користувача та його логіки застосування. Цей модуль також забезпечує API для компонування на високому рівні для спрощення розробки системного інтерфейсу та використання цих функцій у додатках [16].

- **Qt IVI** надає інтерфейси C ++ та QML для доступу до функцій автомобіля та проміжного програмного забезпечення для інформаційних розваг.

  Також дозволяє автоматично генерувати код для реалізації нових функцій IVI. Довідковий користувальницький інтерфейс Забезпечує реалізацію базового інтерфейсу інтерфейсу для Qt в системах інформаційних розваг автомобіля (IVI) [16].
- **Qt GENIVI Extras** Розкриває попередньо визначені інтерфейси альянсу GENIVI [17].
- **Qt Safe Renderer** Подає компонент візуалізації інтерфейсу користувача, який може використовуватися для надання критично важливих для безпеки елементів, таких як попереджувальні індикатори, у функціональних системах безпеки, які потребують сертифікації, не змінюючи бібліотеки Qt. Цей модуль відокремлює критично важливе значення для безпеки від інших частин системи, гарантуючи, що модуль може надавати критичні для безпеки елементи інтерфейсу, навіть якщо в основному інтерфейсі є збої [16].

2.8. Cmake

CMake - це розширювана система з відкритим кодом, яка керує процесом збирання в операційній системі та незалежно від компілятора. На відміну від багатьох платформних систем, CMake розроблений для використання у поєднанні з різними середовищами побудови. Прості файли конфігурації, розміщені у кожному вихідному каталозі (звані файлами CMakeLists.txt), використовуються для створення стандартних файлів збірки

(наприклад, файлів у Unix та проектів / робочих просторів у Windows MSVC), які використовуються звичайним способом [18].

CMake може генерувати рідне середовище збирання, яке буде компілювати вихідний код, створювати бібліотеки, генерувати обгортки та створювати виконувані файли у довільних комбінаціях [18].

CMake підтримує збірки на місці, тому може підтримувати декілька збірок з одного дерева-джерела.

CMake також підтримує статичну та динамічну збірку бібліотек. Ще одна приємна особливість CMake - це те, що він генерує кеш-файл, призначений для використання з графічним редактором. Наприклад, коли CMake запускається, він знаходить файли, бібліотеки та виконувані файли, і може зустріти необов’язкові директиви збірки. Ця інформація збирається в кеш-пам'ять, яку користувач може змінити до створення власних файлів збірки.

CMake призначений для підтримки складних ієрархій каталогів та додатків, залежних від декількох бібліотек. Наприклад, CMake підтримує проекти, що складаються з декількох наборів інструментів (тобто бібліотек), де кожен набір інструментів може містити кілька каталогів, а додаток залежить від наборів інструментів плюс додаткового коду [18].

CMake також може обробляти ситуації, коли виконувані файли повинні бути побудовані для генерування коду, який потім компілюється та пов'язується в остаточну програму. Оскільки CMake є відкритим кодом і має просту, розширювану конструкцію, CMake можна розширювати за необхідності для підтримки нових функцій. Використовувати CMake просто. Процес збирання контролюється шляхом створення одного або декількох файлів CMakeLists.txt у кожному каталозі (включаючи підкаталоги), що складають проект. Кожен CMakeLists.txt складається з однієї або декількох команд. Кожна команда має вигляд COMMAND (args…), де COMMAND - це ім'я команди, а args - список аргументів, розділених пробілом. CMake надає багато заздалегідь визначених команд, але якщо вам потрібно, ви можете

додати свої власні команди. Крім того, досвідчений користувач може додати інші генератори makefile для певної компіляції / компіляції ОС. (У той час як Unix та MSVC ++ підтримуються в даний час, інші розробники додають іншу підтримку компілятора / ОС.) [18].

2.9. OpenStreetMap

OpenStreetMap (OSM) - спільний проект зі створення безкоштовної карти світу з можливістю редагування. Геодані, що лежать в основі карти, вважаються основним результатом проекту. Створення та зростання OSM мотивовано обмеженнями у використанні чи доступності даних карт у більшості країн світу та появою недорогих портативних супутникових навігаційних пристроїв [19].

Користувачі можуть збирати дані за допомогою ручного опитування, GPS-пристроїв, аерофотозйомки та інших безкоштовних джерел. Ці переповнені дані потім стають доступними під ліцензією на відкриту базу даних. Сайт підтримується Фондом OpenStreetMap, некомерційною організацією, зареєстрованою в Англії та Уельсі [19].

Дані з OSM можуть використовуватися різними способами, включаючи виготовлення паперових карт та електронних карт (подібних, наприклад, до Карт Google), геокодування адрес та назв місць та планування маршруту. Визначні користувачі включають Facebook, Craigslist, Lista, OsmAnd, Geocaching, MapQuest Open, статистичне програмне забезпечення JMP та Foursquare. Багато користувачів GPS-пристроїв використовують дані OSM для заміни вбудованих даних карт на своїх пристроях [19].

2.9.1. Програмне забезпечення для роботи OpenStreetMap

Редагування карт можна здійснити за допомогою:

- Редактора веб-браузера за замовчуванням під назвою iD, додатка HTML5 за допомогою D3.js та написаного Mapbox. Більш рання програма на основі Flash Potlatch зберігається для користувачів

середнього рівня. JOSM і Merkaartor - це більш потужні програми для редагування настільних ПК, які краще підходять для досвідчених користувачів [19].

- Vespucci - повнофункціональний редактор для Android [19.]
- StreetComplete - це новий, простий додаток для Android, який дозволяє користувачам без будь-яких знань OpenStreetMap відповідати на прості запити наявних даних у OpenStreetMap і таким чином надавати дані.
- Maps.me - це мобільний додаток (який працює як на Android, так і на iOS), який пропонує офлайн-карти, які також містять обмежений редактор даних, це програма для iOS, яка дозволяє користувачам створювати та редагувати інформацію в OpenStreetMap. Pushpin - ще одна програма для iOS, яка дозволяє додавати POI в дорозі [19].

2.10. Проектування графічного інтерфейсу

Графічний інтерфейс системи представлений у вигляді додаток для ПК. Додаток буде містити наступні інтерфейси візуалізації.

2.10.1. Візуалізація панелі приладів

Візуалізація панелі приладів повинна містити інформацію про швидкість руху, кількість оборотів в хвилину двигуна, індикатори заряду акумулятора, кількості палива та масла, попередження про активоване аварійне гальмо, попередження про не пристібнутий пасок безпеки, попередження про проблеми з двигуном, попередження про пошкодження колес, попередження про низький рівень палива та масла, попередження низького заряду акумулятора, а також сигнал про включені ходові вогні і ввімкнені повороти.

Рис. 2.8 Панель приладів гібридного авто [17]

Рис. 2.9 Панель приладів спортивного авто [17]

2.10.2. Візуалізація мап

Візуалізація мап повинна містити інформацію про розташування доріг, будівель а також транспортних засобів. На рис. 2.10. зображено візуалізація мап.

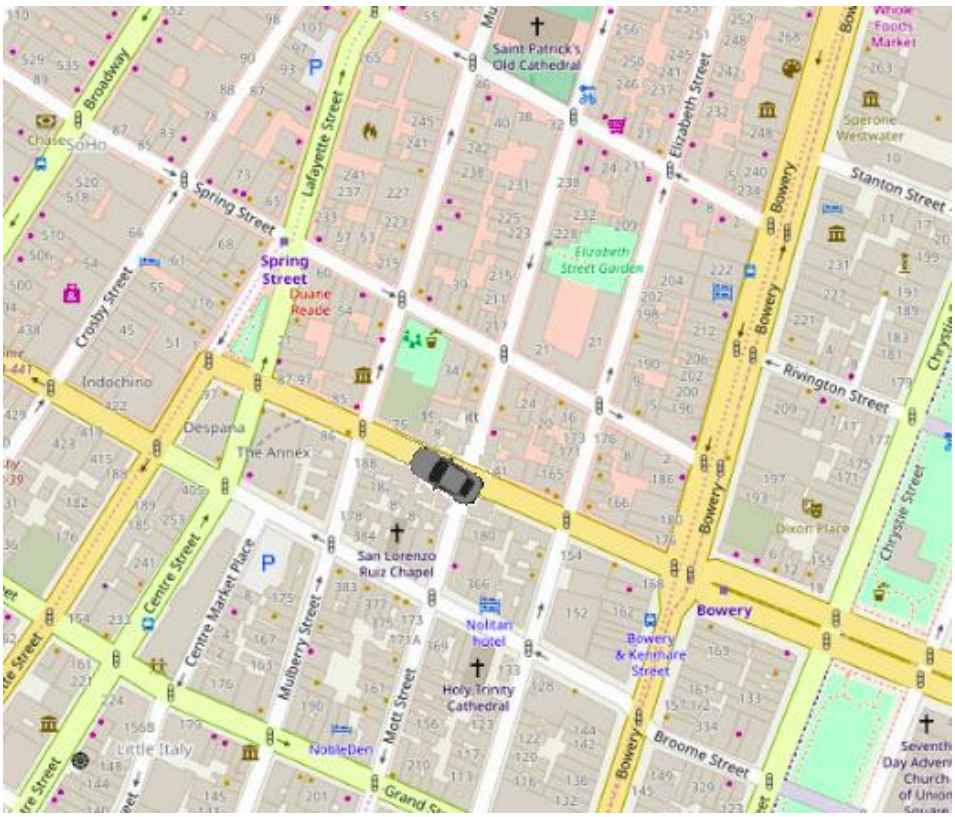


Рис. 2.10 Візуалізація мап [20]

ВИСНОВОК ДО РОЗДІЛУ 2

У розділі 2 було проведено опис предметної області проекту, складені основні функції та вимоги до функціоналу додатку. Було проведено розбір основних технологій на яких будуються системи автопілотування таких як V2V, V2G, V2X проведено аналіз існуючих бібліотек та систем для розробки даного додатку. На основі цього аналізу були обрано бібліотеки Cmake, Qt Automotive Suite та системи Docker-engine і OpenStreetMap для для реалізації програмного та графічного інтерфейсів системи.

Було побудовано діаграму прецедентів для користування основних функцій сервісу:

1. Можливість швидко розгорнути середу розробки;
2. Можливість розгортання додатка на популярних операційних системах;
3. Доступ до документації по розробці;
4. Можливість швидкого запуску;
5. Можливість мануально компілювання системи;
6. Низькі вимогу до аппаратного обладнення;
7. Можливість вести розробку без доступу до інтернета;
8. Організація розробки группами;

Було розроблено графічний інтерфейс додатку за допомогою Qt Automotive Suite та ReactJS та OpenStreetMap.

РОЗДІЛ 3
РОЗРОБКА ДОДАТКУ

3.1. Вибір технологій та їх обґрунтування

3.1.1. Вибір платформи для додатку

У розділах 1 та 2 були визначені вимоги щодо проектування системи автопілот для Smart city та були поставлені такі завдання:

1. Можливість розгортання додатка на популярних операційних системах;
2. Можливість налаштування швидкості руху;
3. Задання маршруту руху транспортного засобу;
4. Можливість додавання нового функціоналу;

Для того, щоб усі вимоги були дотримані, необхідно вибрати платформу для розробки. Вона повинна бути зручна та швидка, щоб зробити систему якомога якіснішою. Серед запропонованих систем було виявлено такі найбільш популярні комп’ютерні платформи:

- Windows;
- Linux;
- macOS.

Для розробки додатку була обрана Ubuntu 20.04 LTS яка є Unix подібною тобто відноситься до сімейства Linux. Платформа на даний час є найновішою також даний реліз є LTS тобто вона буде підтримуватися довгий час що дасть змогу вести довгострокову розробку додатку не побоюючись зміни стандартних пакетів які впливають на розробку системи. Також платформа підходить для роботи з обраними у другому розділі технологіями та бібліотеками.

3.1.2. Вибір мови програмування

Для розробки системи автопілот для Smart city була обрана мова C++. Переваги C ++:

- **Переносимість** - С ++ пропонує функцію портативності або незалежності платформи, що дозволяє користувачеві легко запускати ту саму програму на різних операційних системах або інтерфейсах.

Припустимо, ви пишете програму в ОС LINUX і з явної причини переходите на ОС Windows, ви зможете запускати ту саму програму і в Windows без будь-яких помилок. Ця функція є дуже зручною для програміста [21].

- **Об’єктно-орієнтованість** - Однією з найбільших переваг С ++ є особливість об'єктно-орієнтованого програмування, яка включає такі поняття, як класи, успадкування, поліморфізм, абстрагування даних та інкапсуляція, що дозволяє повторно використовувати код і робить програму ще більш надійною [21].

Ця особливість породила численні перспективи роботи та технології. С++ був створений комбінуванням функцій не лише С, але Simula 67, першої об'єктно-орієнтованої мови програмування [21].

- **Мульти Парадигма** - С ++ - мова програмування має багато парадигми. Термін «парадигма» відноситься до стилю програмування. Вона включає логіку, структуру та процедуру програми. Родова, імперативна та об’єктно-орієнтована - це три парадигми С ++ [21].
- **Маніпуляція низького рівня** - оскільки С ++ тісно пов'язаний з С, що є процедурною мовою, і тісно пов’язаною з машинною мовою, С ++ дозволяє здійснювати низький рівень маніпулювання даними на певному рівні. Вбудовані системи та компілятор створюються за допомогою С ++ [21].
- **Управління пам'яттю** - С ++ надає програмісту загальний контроль над управлінням пам'яттю. Це може розглядатися як як актив, так і як зобов'язання, оскільки це збільшує відповідальність користувача за управління пам’яттю, а не тим, що ним керує збирач сміття. Ця концепція реалізована за допомогою DMA (динамічного розподілу пам'яті) за допомогою покажчиків [21].

- **Сумісність із C** - C ++ в значній мірі сумісний з C. Практично кожна програма без помилок C є дійсною програмою C ++. Залежно від використовуваного компілятора, кожна програма C ++ може працювати на файлі з розширенням .cpp [21].
- **Масштабованість** - Масштабованість означає здатність програми до масштабування. Це означає, що програма C ++ здатна працювати як в малому масштабі, так і у великому масштабі даних. Ми також можемо створювати додатки, що потребують великої ресурсів [21].

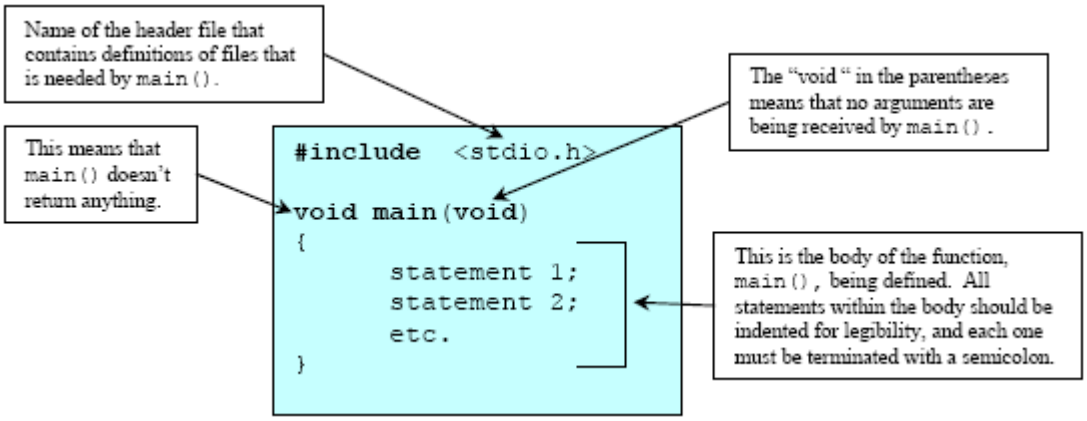


Рис. 3.1 Приклад структури C/C++ програми [22]

3.1.3. Вибір допоміжних бібліотек

C++ – сучасна мова програмування. Для більш зручної роботи можна використовувати допоміжні фреймворки та бібліотеки, які полегшують роботу при розробці додатку. Такими фреймворками є: Boost, Qt Automotive Suite.

Boost - це набір бібліотек для мови програмування C ++, яка забезпечує підтримку операцій та структур, таких як лінійна алгебра, генерація псевдовипадкових чисел, багатопоточність, обробка зображень, регулярні вирази та тестування одиниць. Він містить 162 окремих бібліотеки (станом на версію 1.73) [23].

При розробці додатку використовувались такі концепції Boost:

- **Boost.Algorithm Boost.Algorithm** - надає різні алгоритми, що доповнюють алгоритми зі стандартної бібліотеки [24].
- **Boost.Any Boost.Any** - надає тип, що називається boost :: any, який може зберігати об'єкти довільних типів [24].
- **Boost.Array TR1, C ++ 11 Boost.Array** - дозволяє обробляти масиви C ++, як контейнери зі стандартної бібліотеки [24].
- **Boost.Asio Boost.Asio** - дозволяє розробляти додатки, такі як мережеві програми, які обробляють дані асинхронно [24].
- **Boost.Assign Boost.Assign** - надає допоміжні функції для додавання декількох значень до контейнера без необхідності повторного виклику функцій-членів, таких як push_back () [24].
- **Boost.Atomic C ++ 11 Boost.Atomic** - визначає клас boost :: atomic для виконання атомних операцій на інтегральних значеннях. Бібліотека використовується в багатопотокових програмах, яким потрібно ділити цілісні значення між потоками [24].
- **Boost.Bimap Boost.Bimap** - надає клас під назвою boost :: bimap, який схожий на std :: map. Важлива відмінність полягає в тому, що boost :: bimap дозволяє шукати як ключ, так і значення [24].
- **Boost.Bind TR1, C ++ 11 Boost.Bind** - це адаптер для передачі функцій як параметрів шаблону, навіть якщо підпис функції несумісний із очікуваним параметром шаблону [24].
- **Boost.Chrono C ++ 11 Boost.Chrono** - визначає численні годинники для отримання значень, таких як поточний час або час процесора [24].
- **Boost.Circular_Buffer Boost.Circular_Buffer** - пропонує круговий контейнер з постійним розміром пам'яті [24].
- **Boost.CompressionPair Boost.CompressionPair** - забезпечує структуру даних boost :: compression_pair, яка схожа на std :: pair, але потребує менше пам’яті, коли параметри шаблону порожні класи [24].

- **Boost.Container Boost.Container -** визначає всі контейнери зі стандартної бібліотеки, а також додаткові контейнери, такі як boost :: container :: slist [24].
- **Boost.Conversion Boost.Conversion -** надає двом операторам режиму виконувати пониження та перехресні операції [24].
- **Boost.Coroutine Boost.Coroutine -** дає можливість використовувати підпрограми в C ++. Супроводи часто використовуються в інших мовах програмування, використовуючи ключове слово coroutine [24].
- **Boost.DateTime Boost.DateTime -** може використовуватися для обробки, читання та запису значень дати та часу [24].
- **Boost.DynamicBitset Boost.DynamicBitset** - забезпечує структуру, схожу на std :: bitset, за винятком того, що вона налаштована під час виконання [24].
- **Boost.EnableIf C ++ 11 Boost.EnableIf** - дозволяє перевантажувати функції на основі властивостей типу [24].
- **Boost.Exception Boost.Exception -** дозволяє додавати додаткові дані до винятків, щоб ви могли надати більше даних для вибору обробників [24].
- **Boost.Filesystem Boost.Filesystem** - забезпечує клас для обробки шляхів і декількох функцій для доступу до файлів і каталогів [24].
- **Boost.Flyweight Boost.Flyweight -** полегшує використання однойменної моделі дизайну [24].
- **Boost.Foreach Boost.Foreach** - надає макрос, подібний до діапазону для циклу, що вводиться з C ++ 11 [24].
- **Boost.Format Boost.Format** - замінює функцію std :: printf () на безпечний і розширюваний клас, формат boost :: [24].
- **Boost.Function TR1, C ++ 11 Boost.Function -** спрощує визначення вказівників функції [24].
- **Boost.Fusion Boost.Fusion** - дозволяє створювати неоднорідні контейнери - контейнери, в яких можна зберігати елементи різних типів [24].

- **Boost.Graph Boost.Graph** - надає алгоритми для таких операцій, як пошук найкоротшого шляху між двома точками в графіку [24].
- **Boost.Heap Boost.Heap** - надає багато варіантів класу std :: prior_queue зі стандартної бібліотеки [24].
- **Boost.Integer C ++ 11 Boost.Integer** - визначає спеціалізовані типи для цілих чисел, які були доступні розробникам C з моменту виходу стандарту C99 у 1999 році [24].
- **Boost.Interprocess Boost.Interprocess -** використовує спільну пам'ять, щоб допомогти програмам швидко та ефективно спілкуватися [24].

3.2. Основні рішення з реалізації додатку та його компонентів

Розробку додатку можна поділити на такі пункти:

1. Реалізація серверної частини.
2. Візуалізації панелі приладів.
3. Візуалізації map.
4. Реалізація можливості мануального компілювання.
5. Реалізація можливості запуску додатку за допомогою технологій Docker та Docker-Compose.

3.2.1. Реалізація серверної частини

Для зручного ведення розробки та можливості масштабування додатку у майбутньому було вирішено поділити систему на три основні компоненти:

- **VWS (Virtual World Server)** - даний компонент відповідає за обробку сигналів які надсилають у систему IoT девайси. Тобто веде обрахунок оброблених даних та подій які можуть вплинути на загальну систему Smart city. Дана система відповідальна за візуалізацію даних у іншому компоненті системи - OpenStreetMap. Також він дозволяє можливість спілкування між девайсами які підключені до системи Smart city.
- **SCS (Smart City Server)** - компонент SCS відповідає за обробку сигналів які надсилають у систему автомобілі. Дозволяє транспортним засобам

спілкуватися між собою. Також система відповідальна за візуалізацію даних у іншому компоненті системи - OpenStreetMap.

- **OpenStreetMap** - відповідальний за обробку даних які поступають на нього з VWS і SCS, після якої вносить зміни на мапі.

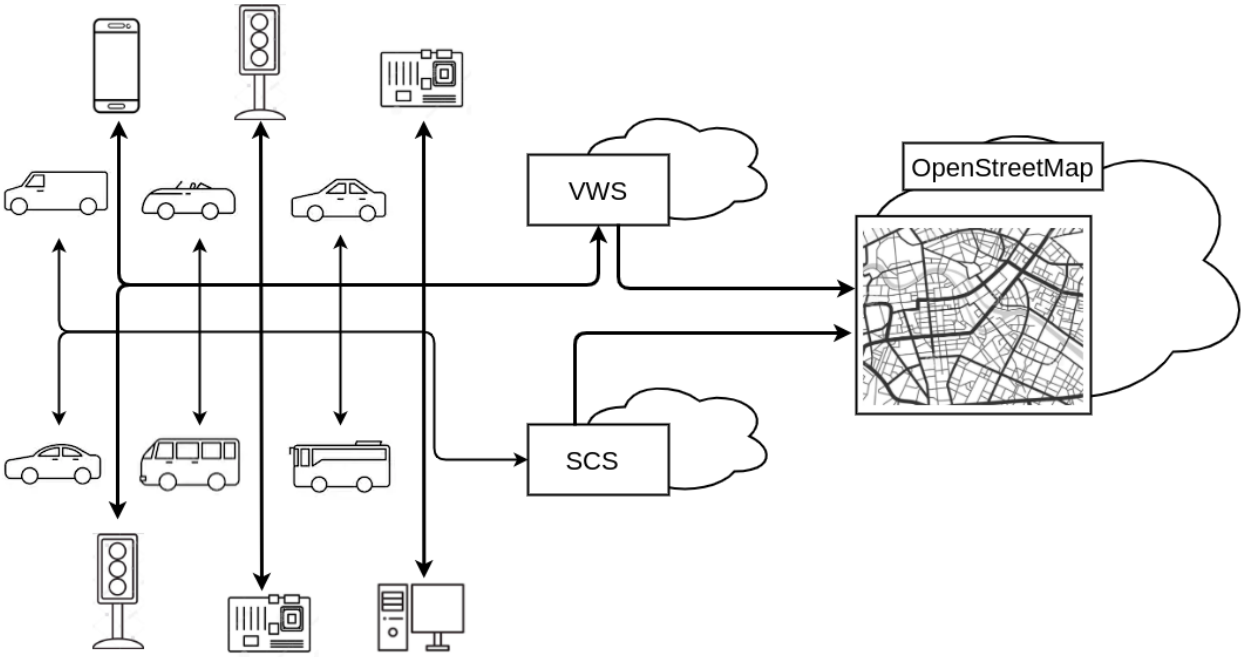


Рис. 3.2 Архітектура роботи системи Smart City

3.2.2. Візуалізації панелі приладів

Для реалізації даного компонента системи було обрано фреймворк з відкритим кодом Qt Automotive Suite, який вже має заготовки для реалізації анімацій панелі приладів. В ході розробки було створено два типи панелей приладів:

- панель гібридного авто див. рис. 3.3.
- панель приладів спортивного авто див. рис. 2.9.

Дані панелі мають однаковий функціонал в який входять індикатори швидкість руху, кількості оборотів в хвилину двигуна, індикатор заряду акумулятора, індикатор кількості палива та масла, попередження про активоване аварійне гальмо, попередження про не пристібнутий пасок безпеки, попередження про проблеми з двигуном, попередження про пошкодження колес, попередження про низький рівень палива та масла,

попередження низького заряду акумулятора, а також сигнал про включені ходові вогні і ввімкнені повороти.

Рис. 3.3 Панель приладів гібридного авто [17]

3.2.3. Візуалізації мап

Для реалізації даного компонента системи було обрано систему з відкритим кодом OpenStreetMap, яка має велику кількість фреймворків для редагування та внесення змін в режимі реального часу. Основною задачею даної компоненту є відображення геолокації транспортного засобу який підключений до системи Smart City.

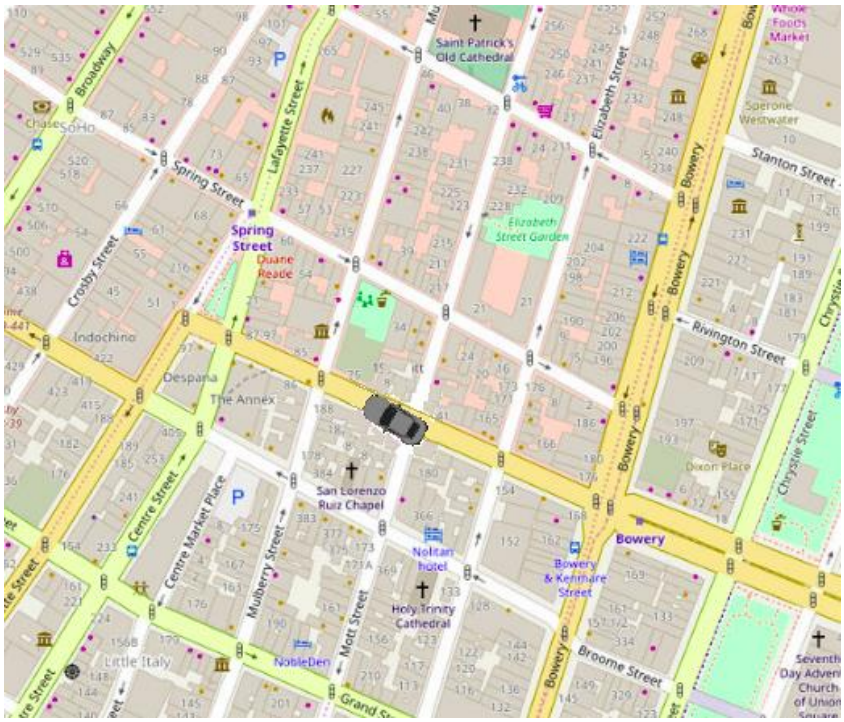


Рис. 3.5 Приклад візуалізації мап [20]

3.2.4 Реалізація можливості мануального компілювання

При розробці додатку було використано велику кількість бібліотек визначеної версії, що у майбутньому при відсутності мануалу по компілюванню може призвести до неможливості роботи із системою, через було вирішено створити детальний мануал по зборці проекту.

3.2.4.1. Підготовка віртуальної машини

Даний пункт зборки проекта є не обов'язковим якщо компілювання виконується не у віртуальному середовищі:

- VirtualBox 6.0, 20GB disk, 4GB mem
- Ubuntu ubuntu-16.04.6-desktop-amd64
- in the VirtualBox Devices menu, choose Insert Guest Additions CD image && run it && select Devices->Drag and drop->Bidirectional && reboot
- update ubuntu software, but do not upgrade
- Connect host to vpn with cisco client

3.2.4.2. Встановлення залежностей

Даний пункт є обов'язковим як для компілювання у не віртуальному середовищі і віртуальному середовищі.

- Відкрити командну строку і ввести команду:

sudo apt install autoconf automake autotools-dev build-essential cmake doxygen freeglut3 freeglut3-dev gcc git libagg-dev libcairo2-dev libfreetype6-dev libmarisa-dev libpangocairo-1.0-0 libpango1.0-dev libprotobuf-dev libqt5svg5-dev libtool libxml2-dev make pkg-config protobuf-compiler subversion libgoogle-glog-dev qtbase5-dev libqt5gui5 qttools5-dev-tools qttools5-dev qtdeclarative5-dev qtlocation5-dev qtpositioning5-dev qt5-default qml-module-qtquick-* qml-module-qtgraphicaleffects qml-module-qtpositioning libboost-thread-dev libboost-system-dev

3.2.4.3. Завантаження проекту

- ssh-keygen -t ecdsa
- cat ~/.ssh/id_ecdsa.pub
- add copied key to profile keys
- git config --global user.name "username"
- git config --global user.email "username@email.com"
- mkdir ~/SBC && cd ~/SBC
- git clone sbcx/sbc-platform.git
- cd sbc-platform && git checkout -b master_local remotes/origin/master
- git submodule update --init

3.2.4.4. Встановлення LIBOSMCOUT

- cd ~/Downloads
- wget master/lastSuccessfulBuild/artifact/package/libosmscout-0.1.20.deb
- sudo apt install ./libosmscout-0.1.20.deb

3.2.4.5. Компілювання SCS та VWS

SCS:

- cd ~/SBC/sbc-platform/src/SCS/mapviewer && mkdir build && cd build && qmake .. && make
- nano ~/SBC/sbc-platform/src/SCS/conf/config.json
- insert real path, for ex. (/home/username/SBC/sbc-platform/src/) instead <FULL_PATH_TO_REPOSITORY>

VWS:

- cd ~/SBC/sbc-platform/src/VWS && mkdir build && cd build && cmake .. && make

3.2.4.6. Компілювання та встановлення IC-LIB та IC_LINUX

IC-LIB:

- $ cd ~/SBC/sbc-platform/src/IC-Lib/ && mkdir build && cd build && cmake .. && make
- $ sudo make install && sudo ldconfig -v

IC_LINUX:

- $ cd ~/SBC/sbc-platform/src/IC-Linux/ && mkdir build && cd build && qmake .. && make

3.2.4.7. Запуск скомпільованої системи

Run everything in separate terminal windows:

- cd ~/SBC/sbc-platform/src/VWS/build/ && ./vw-server -p 40881 -s ../conf/demo-gps.json -t 10 -v
- cd ~/SBC/sbc-platform/src/SCS/ && ./mapviewer/build/SBCStat conf/config.json
- cd ~/SBC/sbc-platform/src/IC-Linux && ./build/SmartCity_IC conf/config.json

3.2.5. Реалізація можливості запуску додатку за допомогою технологій Docker та Docker-Compose

Для можливості запуску проекту без попереднього компілювання системи було вирішено використати Docker принцип роботи якого заключається шо у створеному файлі Dockerfile.txt описуються послідовність компілювання певного компонента системи. Після на основи даного файла створюється Docker image який можно багаторазово використовувати для запуску компонента.

Послідовність роботи з Docker:

- створити Dockerfile.txt і описати послідовність компілювання певного компонента системи див. рис. 3.6.
- створити Docker image із раніше написаного Docker файлу за допомогою команди docker build -t <image_name> path to Dockerfile

- запустити створений Docker image за допомогою команди docker run -it <image_name> bin/bash

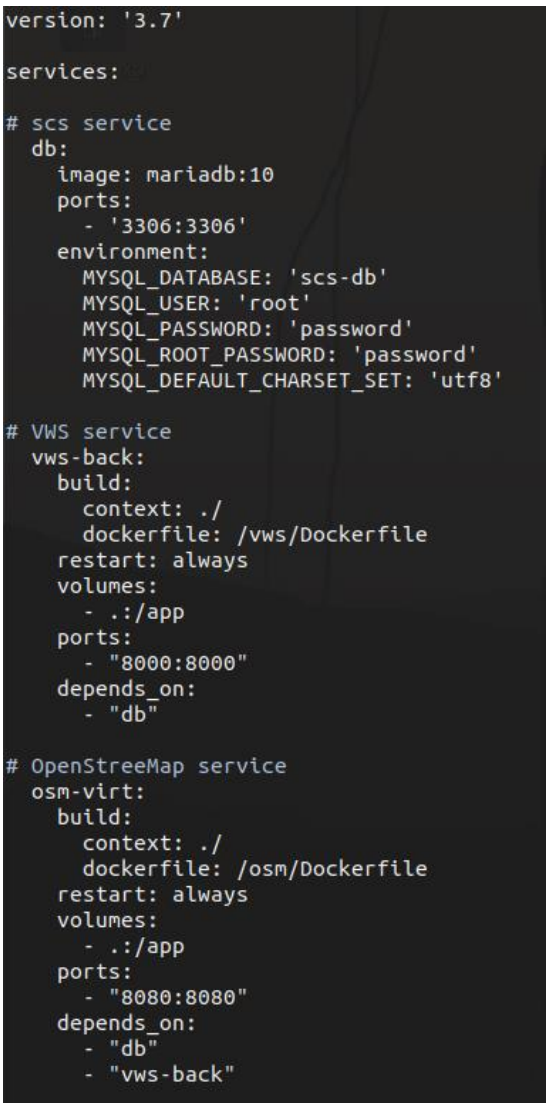
```
FROM ubuntu 20.04

ADD . /app

WORKDIR /app

RUN apt-get update && apt-get install -y gcc libmariadb-dev-compat mariadb-client
RUN pip install --upgrade pip && pip install --no-cache-dir -r requirements.txt

EXPOSE 8000

ENTRYPOINT [ "/bin/bash", "-c", "/app/scripts/start_server.sh" ]
```

Рис. 3.6 Приклад Dockerfile компоненту візуалізації мап

Так як кожен компонент описується окремим Doker файлом було вирішено використати технологію Docker-Compose за допомогою якого можна запустити всі контейнери з окремими компонентами додатку і налаштувати їх взаємодію між собою.

Послідовність роботи з Docker-Compose:

- створити docker_compose.yml файл і описати послідовність запуску системи див. рис. 3.7.
- створити Docker image із раніше написаного Docker файлу за допомогою команди docker-compose build path to docker_compose.yml
- запустити створений Docker image за допомогою команди docker-compose up



```
version: '3.7'

services:

# scs service
  db:
    image: mariadb:10
    ports:
      - '3306:3306'
    environment:
      MYSQL_DATABASE: 'scs-db'
      MYSQL_USER: 'root'
      MYSQL_PASSWORD: 'password'
      MYSQL_ROOT_PASSWORD: 'password'
      MYSQL_DEFAULT_CHARSET_SET: 'utf8'

# VWS service
  vws-back:
    build:
      context: ./
      dockerfile: /vws/Dockerfile
    restart: always
    volumes:
      - .:/app
    ports:
      - "8000:8000"
    depends_on:
      - "db"

# OpenStreeMap service
  osm-virt:
    build:
      context: ./
      dockerfile: /osm/Dockerfile
    restart: always
    volumes:
      - .:/app
    ports:
      - "8080:8080"
    depends_on:
      - "db"
      - "vws-back"
```

Рис. 3.7 Приклад docker compose файлу для запуску додатку

ВИСНОВОК ДО РОЗДІЛУ 3

У даному розділі був проведений аналіз технологій розробки додатку, а також огляд додаткових бібліотек, які можуть допомогти вирішити поставленні задачі. Враховуючи усі переваги та недоліки кожної технології, було обрано мову програмування та платформи, що реалізує побудову крос-платформних додатків – C++. Для побудови систем візуалізації було обрано Qt Automotive Suite і OpenStreetMaps, що має ряд переваг у порівнянні з іншими рішеннями.

Для можливості запуску додатку без залежності від платформи і версій основних бібліотек було обрано Docker і Docker-Compose - відповідно.

Реалізацію додатку та його компонентів було розбито на 5 основних етапів: реалізація серверної частини, візуалізації панелі приладів, візуалізації мап, реалізація можливості мануального компілювання, реалізація можливості запуску додатку за допомогою технологій Docker та Docker-Compose. Кожен етап реалізації виконувався згідно вимог, зазначених у технічному завданні. У результаті був створений додаток для запуску на системі Ubuntu 20.04.

Були наведені відповідні рисунки готових частин графічного інтерфейсу.

ВИСНОВКИ

Розробка даного дипломного проекту була присвячена дослідженню можливих варіантів рішень, спрямованих на вирішення задачі створення автопілоту адаптованого під систему Smart City.

В ході виконання проекту були розглянуті вже існуючі на даний момент рішення, які реалізують поставлені задачі. На основі цих рішень було складено порівняльну характеристику з урахуванням всіх переваг та недоліків кожного. Було запропоновано варіанти модернізації даної системи, які включають в себе комплексне вирішення основних проблем існуючих додатків.

Також було проаналізовано предметну область даної роботи та визначено основні вимоги і функції додатку. На основі визначених вимог до системи побудовані схематичні рисунки та таблиці. Проведено проектування інтерфейсу та побудовано відповідні компоненти візуалізації системи.

Зважаючи на вище задані вимоги був проведений аналіз використання існуючих технологій та платформ для реалізації системи. Оптимальною платформою для реалізації було обрано Unix подібну систему Ubuntu 20.04 із сімейства Linux спираючись на економічну та функціональну вигідність використання. За допомогою технологій Docker і Docker-Compose додаток був реалізований з можливістю розгортання на більшості платформ Linux, Windows, Mac Os - відповідно. Також було обрано мову написання проекту, і проведений опис використаних додаткових бібліотек з обґрунтуванням доцільності їх використання.

Реалізація додатку відбувалася у 5 основних етапи з урахуванням зазначеного у технічному завданні функціоналу та вимог до розробки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Автопілот [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Autopilot
2. Рівні автономності автопілотів[Електронний ресурс] – Режим доступу до ресурсу: https://www.wired.com/2016/08/self-driving-car-levels-sae-nhtsa/
3. Автопілот General motors [Електронний ресурс] – Режим доступу до ресурсу: https://www.gm.com/our-stories/self-driving-cars.html
4. Автопілот Tesla[Електронний ресурс] – Режим доступу до ресурсу: https://www.tesla.com/autopilot
5. Автопілот Volvo[Електронний ресурс] – Режим доступу до ресурсу: https://www.zenuity.com
6. Автопілот Waymo[Електронний ресурс] – Режим доступу до ресурсу: https://waymo.com/tech/
7. V2V [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Vehicular_ad-hoc_network
8. Приклади взаємодії V2V [Електронний ресурс] – Режим доступу до ресурсу: https://doclecture.net/1-45687.html
9. V2D [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Vehicle-to-device
10. V2G [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Vehicle-to-grid
11. Приклади взаємодії V2G [Електронний ресурс] – Режим доступу до ресурсу: https://ignitisinnovation.com/2019/12/11/vehicle-to-grid/
12. V2X [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Vehicle-to-everything
13. Приклади взаємодії V2X [Електронний ресурс] – Режим доступу до ресурсу: http://mahbubulalam.com/what-is-vehicle-to-everything-and-how-will-it-help/

14. Docker [Електронний ресурс] – Режим доступу до ресурсу: https://docs.docker.com/engine/
15. Docker architecture [Електронний ресурс] – Режим доступу до ресурсу: https://docs.docker.com/get-started/overview/#docker-architecture
16. Qt Automotive Suite[Електронний ресурс] – Режим доступу до ресурсу: https://doc.qt.io/QtAutomotiveSuite/index.html
17. Qt Automotive Suite[Електронний ресурс] – Режим доступу до ресурсу: https://doc.qt.io/QtSafeRenderer/qtsaferenderer-saferenderer-qtcluster-example.html
18. Cmake [Електронний ресурс] – Режим доступу до ресурсу: https://cmake.org/overview/
19. OpenStreetMap [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/OpenStreetMap
20. OpenStreetMap [Електронний ресурс] – Режим доступу до ресурсу: https://www.hellojae.com/home/open-street-map
21. C++ [Електронний ресурс] – Режим доступу до ресурсу: https://data-flair.training/blogs/advantages-and-disadvantages-of-cpp/
22. Приклади структури C++ [Електронний ресурс] – Режим доступу до ресурсу: https://www.tenouk.com/Module1.html
23. Boost [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Boost_(C%2B%2B_libraries)
24. Boost концепції [Електронний ресурс] – Режим доступу до ресурсу: https://theboostcpplibraries.com/introduction-overview

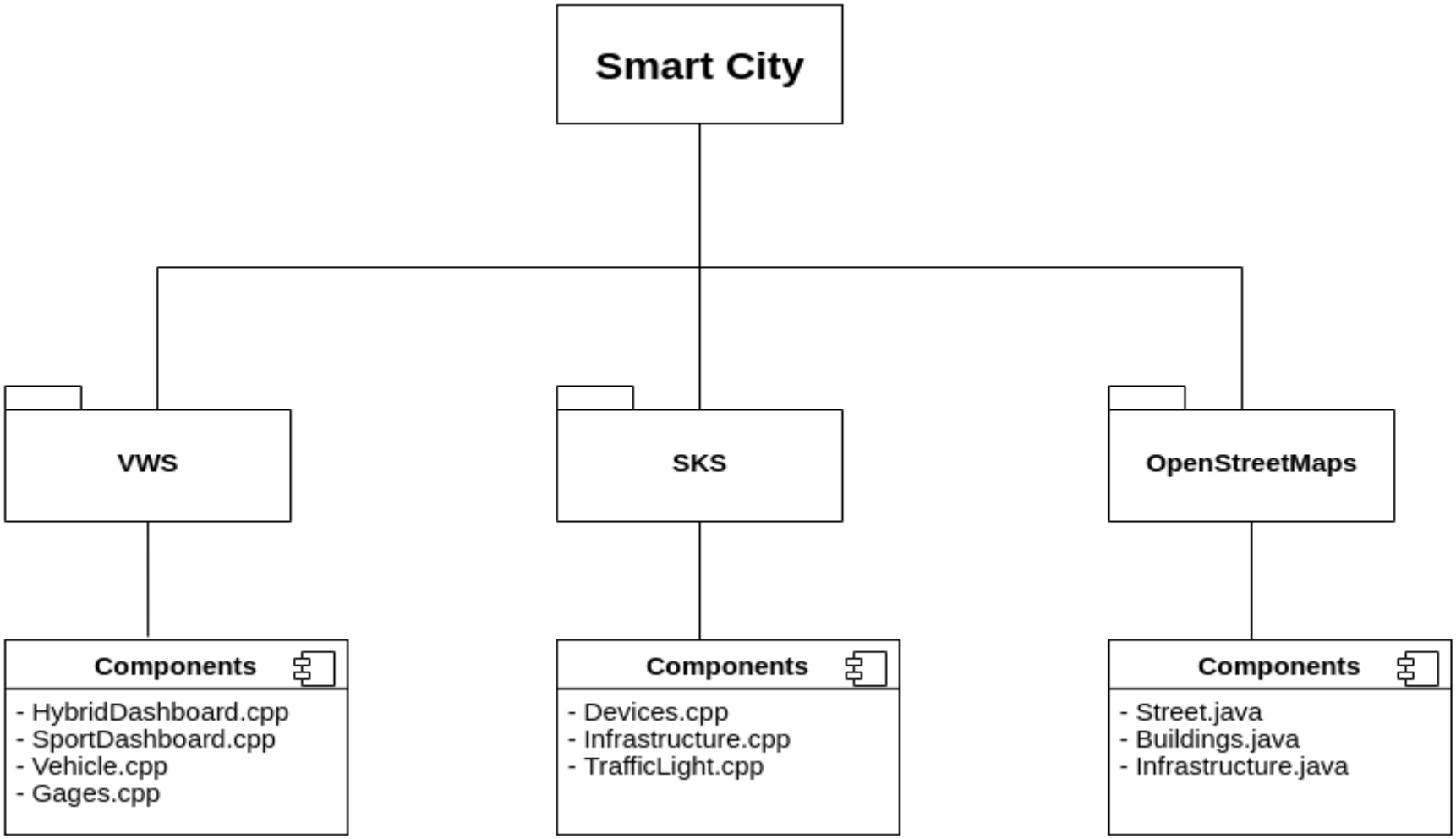
ДОДАТОК 1

Інтернет-речей система автопілот для «Smart city»

Схема структурна – структура програми
ІАЛЦ.467800.004 Д1

Аркушів 1

Київ – 2020



Зм.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467800.004 Д1		
Розробив		Левченко К.В.			Інтернет-речей система автопілот для «Smart city»	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Ткаченко В.В.					
Реценз.					Схема структурна	НТУУ «КПІ», ФІОТ, ІО-63	
Н. Контр.		Сімоненко В.П.					
Затв.		Стіренко С.Г.					

ДОДАТОК 2

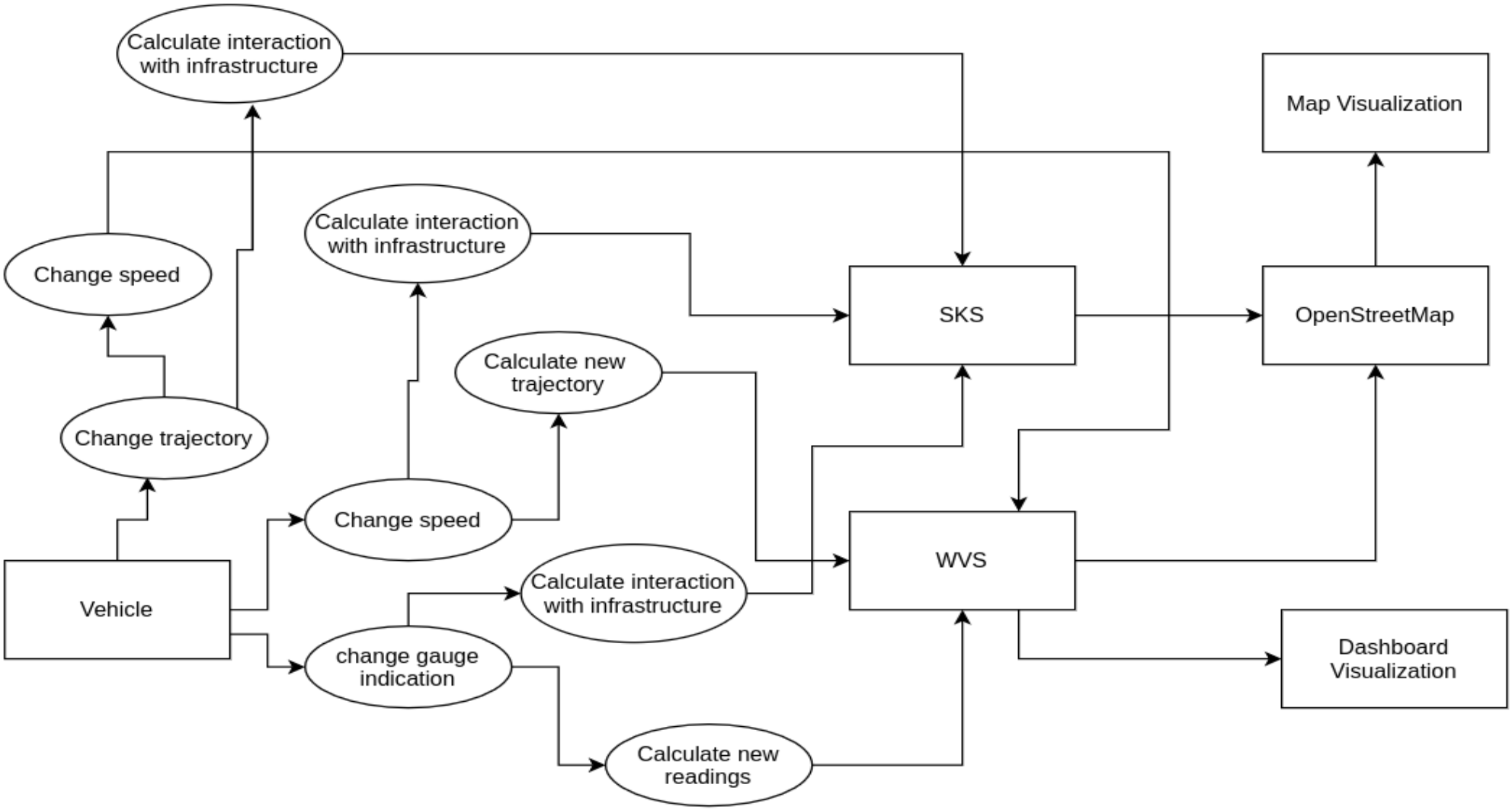
Інтернет-речей система автопілот для «Smart city»

Схема функціональна – схема прецедентів

ІАЛЦ.467800.005 Д2

Аркушів 1

Київ – 2020



Зм.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467800.005 Д2		
Розробив		Левченко К.В.			Інтернет-речей система автопілот для «Smart city»	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Ткаченко В.В.					
Реценз.					Схема функціональна	НТУУ «КПІ», ФІОТ, ІО-63	
Н. Контр.		Сімоненко В.П.					
Затв.		Стіренко С.Г.					

ДОДАТОК 3

Інтернет-речей система автопілот для «Smart city»

Схема принципова – схема алгоритму додавання нового пристрою

ІАЛЦ.467800.006 Д3

Аркушів 1

Київ – 2020

					ІАЛЦ.467800.006 Д3	
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розробив		*Левченко К.В.*			*Інтернет-речей система автопілот для «Smart city»*	*Літ.* / *Аркуш* / *Аркушів*
Перевірив		*Ткаченко В.В.*				*1* / *1*
Реценз.						
Н. Контр.		*Сімоненко В.П.*			*Схема принципова*	*НТУУ «КПІ», ФІОТ, ІО-63*
Затв.		*Стіренко С.Г.*				