A framework for collection, contextual enrichment and advanced analytics of automotive data

Vdović, Hrvoje

Doctoral thesis / Disertacija

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:168:109289

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-25**

Repository / Repozitorij:

FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repozitory

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Hrvoje Vdović

A FRAMEWORK FOR COLLECTION, CONTEXTUAL ENRICHMENT AND ADVANCED ANALYTICS OF AUTOMOTIVE DATA

DOCTORAL THESIS

Zagreb, 2022

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Hrvoje Vdović

A FRAMEWORK FOR COLLECTION, CONTEXTUAL ENRICHMENT AND ADVANCED ANALYTICS OF AUTOMOTIVE DATA

DOCTORAL THESIS

Supervisor: Assistant Professor Jurica Babić, PhD

Zagreb, 2022

Sveučilište u Zagrebu

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Hrvoje Vdović

RADNI OKVIR ZA PRIKUPLJANJE, KONTEKSTNO OBOGAĆIVANJE I NAPREDNU ANALITIKU PODATAKA IZ VOZILA

DOKTORSKI RAD

Mentor: doc. dr. sc. Jurica Babić

Zagreb, 2022.

The doctoral thesis has been completed at the University of Zagreb Faculty of Electrical Engineering and Computing, Department of Telecommunications.

Supervisor: Assistant Professor Jurica Babić, PhD

The thesis has: 132 pages

Thesis number: _____

About the Supervisor

Jurica Babić was born in Varaždin in 1989. He received B.Sc., M.Sc. and Ph.D. degrees in computing from the University of Zagreb, Faculty of Electrical Engineering and Computing (FER), Zagreb, Croatia, in 2010, 2012 and 2018, respectively.

From September 2012 he is working at the Department of Telecommunications FER. In April 2019 he was promoted to Assistant Professor. He participated in three scientific and research projects financed by European Regional Development Fund and four EU Erasmus+ projects. Currently he is a project coordinator of the project entitled "Innovative solutions based on emerging technologies for improving social inclusion of people with disabilities" which is financed by Agency for mobility and EU programmes through the Erasmus+ program. He published more than 30 publications in scientific journals and conference proceedings in the area of multi-agent systems, electric vehicles and electric vehicle charging infrastructure.

Prof. Babić is a member of IEEE, a researcher at Laboratory for social networking and social computing (SocialLAB) and Laboratory for assistive technologies and augmentative and alternative communication (ICT-AAC), and an ambassador for social inclusion and diversity at Agency for mobility and EU programmes. He participates as a reviewer in multiple international journals and scientific conferences. In 2015, together with other ICT-AAC team members, he received a state award „Ivan Filipović“ for significant results in education.

O mentoru

Jurica Babić rođen je u Varaždinu 1989. godine. Diplomirao je, magistrirao i doktorirao u polju računarstva na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva (FER), 2010., 2012. odnosno 2018. godine.

Od rujna 2012. godine radi na Zavodu za telekomunikacije FER-a. U travnju 2019. godine izabran je u zvanje docenta. Sudjelovao je na tri znanstveno-istraživačkih projekata financiranih iz Europskog fonda za regionalni razvoj i četiri EU Erasmus+ projekta. Trenutno je koordinator projekta: "Innovative solutions based on emerging technologies for improving social inclusion of people with disabilities" koji financira Agencija za mobilnost i programe EU kroz program Erasmus+. Objavio je više od 30 radova u časopisima i zbornicima konferencija u području višeagentskih sustava, električnih vozila i infrastrukture punionica za električna vozila.

Prof. Babić član je stručne udruge IEEE, istraživač u laboratorijima „Laboratorij za društveno umrežavanje i društveno umrežavanje“ (SocialLAB) i „Laboratorij za asistivne tehnologije i potpomognutu komunikaciju“ (ICT-AAC), te ambasador za uključivanje i raznolikost pri Agenciji za mobilnost i programe EU. Sudjeluje kao recenzent u većem broju inozemnih časopisa i znanstvenih konferencija. 2015. godine zajedno je s drugim članovima tima ICT-AAC primio državnu nagradu „Ivan Filipović“ za značajna ostvarenja u odgojno-obrazovnoj djelatnosti.

Ovaj rad je posvećen mojoj obitelji. Hvala vam na ljubavi, brizi i podršci!

Abstract

The thesis addresses the challenges related to collection, contextual enrichment and analytics of automotive data. Current trends that focus on vehicle connectivity, i.e. vehicle-2-infrastructure and vehicle-2-vehicle, are seen as beneficial for data collection and analysis. However, collecting data from connected vehicles is not a simple task because connected vehicle hardware and software are a closed system meant to be used by vehicle manufacturers only. Additionally, when automotive data is combined with various heterogeneous data sources, it can provide valuable additional insights in data analysis. The key component for impactful research in this domain is a high-quality source of contextually enriched automotive data, enabling interdisciplinary study from environmental sustainability, automotive engineering, behavioural science, telecommunications and transportation science perspectives. To the best of the author's knowledge, there is no open source data set of contextually enriched automotive data available.

To facilitate the creation of such a data set, this thesis presents a framework for the collection and contextual enrichment of automotive data. The framework utilizes smartphones to collect the vehicle's On-Board Diagnostics (OBD) or Controller Area Network (CAN) data and enriches it with Internet-based and built-in smartphone sensor data. The framework was evaluated on a case study of a data collection experiment. In this experiment, 9 drivers collected more than 90 hours of driving data, forming a contextually enriched automotive data set. A special emphasis in the analysis of the collected data was placed on deriving and calculating metrics for ranking the drivers and trips according to their eco-efficient driving patterns, which is a valuable insight that can be used to improve the transportation sustainability.

Keywords: automotive data, data collection, contextual enrichment, data analytics

Prošireni sažetak

Radni okvir za prikupljanje, kontekstno obogaćivanje i naprednu analitiku podataka iz vozila

Doktorski rad se bavi izazovima vezanim uz prikupljanje, kontekstno obogaćivanje i analitiku podataka iz vozila. Trenutni trendovi koji se fokusiraju prema umrežavanju vozila kao što su povezivanje vozila s infrastrukturom i povezivanje vozila s drugim vozilima, smatraju se korisnima za prikupljanje i analizu podataka. Međutim, prikupljanje podataka iz umreženih vozila nije jednostavan zadatak, jer su hardver i softver umreženih vozila zatvoreni sustav koji je namijenjen da mu pristupaju samo proizvođači vozila. Osim toga, kada se podaci iz vozila kombiniraju s različitim heterogenim izvorima podataka, moguće je pružiti vrijedne dodatne uvide tijekom analize podataka. Ključna komponenta za utjecajna istraživanja u ovoj domeni je visokokvalitetan izvor kontekstno obogaćenih podataka iz vozila, koji bi omogućio interdisciplinarna istraživanja iz perspektive okolišne održivosti, automobilskog inženjerstva, bihevioralne znanosti, telekomunikacija i prometnih znanosti. Koliko je autoru poznato, ne postoji javno dostupan skup kontekstno obogaćenih podataka iz vozila.

Kako bi se olakšalo stvaranje takvog skupa podataka, doktorski rad predstavlja radni okvir za prikupljanje, kontekstno obogaćivanje i naprednu analitiku podataka iz vozila. Okvir koristi pametne telefone za prikupljanje podataka s OBD dijagnostičkog priključka ili CAN sabirnice i obogaćuje ih s podacima prikupljenih s internetskih izvora i podacima iz senzora ugrađenih unutar pametnog telefona. Radni okvir je evaluiran na studijskom slučaju eksperimenta prikupljanja podataka. U tom eksperimentu je 9 vozača prikupilo više od 90 sati podataka o vožnji, koji su zajedno formirali kontekstno obogaćen skup podataka iz vozila. U sklopu doktorskog rada je prikupljeni skup podataka analiziran te je poseban naglasak u analizi podataka stavljen na izračun metrike za rangiranje vozača i putovanja prema ekološkim obrascima vožnje koja se može iskoristiti za poboljšanje održivosti prijevoza.

Uvodno poglavlje ("1. Introduction") započinje rad pružajući sažeti opis istraživačkog područja i ciljeva istraživanja. U ovom poglavlju se definira problem u obliku sljedećih istraživačkih pitanja:

1.Kako prikupiti podatke iz vozila u stvarnom vremenu, pokrivaju ći širok spektar različitih

vozila?

2.Kako obogatiti podatke iz vozila kontekstom iz više heterogenih izvora?

3.Kako skladištiti prikupljene kontekstno oboga ćene podatke iz vozila?

4.Kako stvoriti novo znanje analizom kontekstno oboga ćenih podataka iz vozila?

5.Kako pružiti pristup prikupljenim podacima iz vozila uz primjenu zadanih politika sigurnosti i prava pristupa?

Prvo istraživačko pitanje proučava način prikupljanja podataka iz vozila. Kako bi vozilo moglo prikupiti podatke, mora biti povezano na Internet. Umrežena vozila su idealan kandidat za prikupljanje podataka, ali su njihovi hardver i softver zatvoren sustav te su potrebne administratorske ovlasti kako bi mu se pristupilo. Vozila umrežena pomoću pametnih telefona su skalabilno, nadogradivo i jeftino rješenje tog problema te su zbog tog razloga odabrana kao rješenje za prikupljanje podataka. Kako bi se još više proširio spektar vozila s kojih se podaci mogu prikupljati, odabrane su dvije automobilske komunikacijske tehnologije s kojih će se prikupljati podaci: OBD-II dijagnostički utor i CAN sabirnica. Drugo istraživačko pitanje se bavi obogaćivanjem podataka iz vozila kontekstom. Pametni telefoni rješavaju ovaj problem, jer omogućuju istovremeno prikupljanje podataka iz vozila i iz internetskih izvora, kao i iz senzora ugrađenih u pametni telefon. Treće istraživačko pitanje preporučuje kako skladištiti prikupljene kontekstno obogaćene podatke iz vozila. Takvi podaci konvergiraju velikoj količini podataka (engl. *big data*) te je potrebno adresirati njihova svojstva 5V: količinu (*volume*), različitost (*variety*), brzinu (*velocity*), vjerodostojnost (*veracity*) i vrijednost (*value*). Platforma za skladištenje kontekstno obogaćenih podataka iz vozila mora biti dizajnirana i implementirana na način koji će adresirati sva navedena svojstva. Cilj četvrtog istraživačkog pitanja je proučiti kako stvoriti novo znanje analizom kontekstno obogaćenih podataka iz vozila. Znanost o podacima i metode koje koriste umjetno inteligenciju su korisne za postizanje tog cilja. Konačno, peto istraživačko pitanje demonstrira kako omogućiti nesmetan pristup prikupljenim i analiziranim podacima iz vozila pomoću aplikacijskog programskog sučelja koje služi kao posrednik u kojem su implementirane zadane politike sigurnosti i prava pristupa.

Drugo poglavlje ("2. State of the Art") pruža pregled relevantnih istraživanja u području automobilske informacijske i komunikacijske tehnologije (ICT) i u području manipulacije podataka iz vozila. Kako bi se stekao uvid u to kako su softver i podaci postali bitni za suvremena vozila, korisno je pogledati povijesne trendove u automobilskoj industriji te je zbog toga dat kratak pregled povijesti automobila i automobilskog softvera. U pregledu je prvo prikazana rana povijest automobila, gdje su opisani prvi automobili s motorom s unutarnjim izgaranjem i njihov razvoj. Zatim su opisani prvi električni automobili te prva vozila koja su u sebi imale komponente koje su izvršavale softver. Od tada do danas je automobilski softver preuzeo upravljanje gotovo svih funkcionalnosti u vozilima. S napretkom automobilskog softvera pojavila su se i umrežena vozila - vozila koja imaju mogućnost komunikacije s Internetom, drugim vozil-

ima i infrastrukturom. Konačno, u posljednjih desetak godina se intenzivno radi na razvoju autonomnih vozila koja se mogu kretati bez ljudskog upravljanja. Ovaj povijesni pregled poslužio je za kategorizaciju automobila i njihovih komponenti. Četiri svojstva povezana sa sve većom količinom softvera i podataka u automobilima odabrana su kao osnova za klasifikaciju:

- *električna* (tj. vozila koja pokreću jedan ili više elektromotora);
- *softverizirana* (tj. vozila s bilo kojom vrstom automobilskog softvera);
- *umrežena* (tj. vozila koja se mogu povezati na Internet i komunicirati s pametnim uređajima, drugim automobilima i infrastrukturom);
- *autonomna* (tj. vozila sposobna za vožnju bez ljudskog utjecaja).

Slijedeći spomenutu klasifikaciju, pružen je pregled ICT arhitekture suvremenih vozila. Nakon navedenog, dat je pregled literature koja se bavi temom manipulacije podataka iz vozila. U sklopu pregleda literature obrađeni su radovi koji se bave prikupljanjem, kontekstnim obogaćivanjem, pohranom i analitikom podataka iz vozila. Zaključeno je da ne postoji rješenje za prikupljanje podataka koje podržava automobilske komunikacijske protokole više i niže razine te da rješenja koja postoje rijetko podržavaju kontekstno obogaćivanje podataka. Osim toga, postoji potreba za visoko dostupnim i skalabilnim rješenjem za pohranu podataka i vozila te za specifikacijom aplikacijskog programskog sučelja za pristup takvim podacima.

Treće poglavlje ("3. Collection and Contextual Enrichment of Automotive Data") opisuje prikupljanje i kontekstno obogaćivanje podataka iz vozila implementirane u radnom okviru koji rješava probleme identificirane u prvom i drugom poglavlju. Kako bi se bolje razumio razvojni proces korišten prilikom uspostavljanja radnog okvira, prvo se opisuje korištena metodologija. Metodologija se sastoji od sljedeća četiri koraka: razumijevanje domene, prikupljanje i razumijevanje podataka, modeliranje i evaluacija. Nakon metodologije su opisane svaka od pet različitih međusobno povezanih komponenti radnog okvira. Prva komponenta su *izvori podataka* iz kojih se prikupljaju podaci. To uključuje automobilske izvore podataka - OBD-II priključak i CAN sabirnicu, izvore s mobilnih uređaja - informacije o lokaciji, signalu i senzorima te izvore podataka s Interneta - informacije o vremenu i prometu. S obzirom da CAN sabirnica nije lako dostupna u putničkim vozilima, sljedeća komponenta uključena kao dio okvira je *simulator CAN sabirnice*, koji služi kao zamjena stvarnim vozilima. Pomoću simulatora se može kreirati scenarij za simulaciju jedne ili više komponenti vozila. Kako bi se stvorio scenarij, korisnik mora unijeti podatke o komponentama i podacima koji se simuliraju. Simulator te podatke obrađuje i pretvara u kontinuirani niz CAN poruka koje se preko USB-CAN pretvarača odašilju na CAN izlaz. Moduli za *prikupljanje i kontekstno obogaćivanje* treća su komponenta radnog okvira. Implementirani su u obliku aplikacije za pametne telefone i prikupljaju podatke s OBD-II priključka ili CAN sabirnice te ih obogaćuju informacijama prikupljenih s Interneta i mobilnog uređaja. Aplikacija komunicira s OBD-II priključkom i CAN sabirnicom putem Bluetooth ili Wi-Fi veze. Pri prvom pokretanju, skupljaju se podaci o vozaču i vozilu

koje prikuplja podatke. Na svaki prikupljeni podataka iz vozila dodaju se kontekstni podaci o lokaciji, prometu, vremenu, mobilnom signalu i pozicijskim senzorima pametnog telefona te se šalju na platformu za pohranu podataka. *Distribuirana platforma za pohranu podataka* koja se koristi za pohranu kontekstno obogaćenih podataka iz vozila je četvrta komponenta radnog okvira. Ona prima podatke iz modula za prikupljanje i sigurno ih pohranjuje za daljnju upotrebu. Platforma je implementirana na način da zadovoljava zahtjeve pouzdanosti, skalabilnosti i robusnosti koji proizlaze iz domene te koristi nerelacijsku bazu podataka za spremanje podataka. Posljednja komponenta je *aplikacijsko programsko sučelje* koje se koristi za pristup i analizu pohranjenih automobilskih podataka. Sastoji se od modula za dohvat podataka, modula za statistiku i modula za naprednu analitiku s kojima korisnici mogu komunicirati. Konačno, na kraju poglavlju je dan pregled politika sigurnosti i prava pristupa unutar radnog okvira između svih njegovih komponenti koje komuniciraju međusobno i s okolinom.

Nakon što je radni okvir uspostavljen i nakon što su sve komponente u njemu implementirane i validirane, mogao se iskoristiti za prikupljanje kontekstno obogaćenog skupa podataka iz vozila. Taj skup podataka je opisan u četvrtom poglavlju ("4. Contextually Enriched Automotive Data Set"). Kako bi se pružilo bolje razumijevanje skupa podataka koji se prikuplja, detaljno je opisan model kontekstno obogaćenih podataka. Model se sastoji od jedanaest međusobno povezanih podatkovnih entiteta koji opisuju vozača, vozilo, putovanja i podatke prikupljene tijekom putovanja. U ovom poglavlju je također opisano kako je osmišljen i izveden eksperiment prikupljanja podataka. Eksperiment je trajao 7 tjedana te je u njemu sudjelovalo 9 vozača različitog spola (5 muškaraca, 4 žene), različite dobi (25-53 godine) i različitog vozačkog iskustva (2-36 godina) koji su vozili 8 različitih vozila (2 vozača su vozili isto vozilo). Tijekom eksperimenta je prikupljeno više od 300,000 podatkovnih točaka te je nakon obavljenog čišćenja podataka formiran konačan skup podataka koji se sastoji od 287,882 podatkovnih točaka prikupljenih tijekom 212 putovanja, što iznosi više od 94 sata snimljene vožnje. Nakon što su podaci prikupljeni, izračunate su statističke informacije o skupu podataka te su prikazane zajedno s primjerima vizualizacije koji služe kao uvod u moguće slučajeve korištenja takvog skupa podataka.

Peto poglavlje ("5. Advanced Analytics of the Contextually Enriched Automotive Data Set") opisuje analitičke postupke izvedene na prikupljenim kontekstno obogaćenim podacima iz vozila. Prikazana su tri primjera analize nad skupom podataka. U sklopu prvog primjera je izvedena i izračunata metrika *eko indeks* za rangiranje vozača prema njihovim ekološki učinkovitim obrascima vožnje. Metrika se sastoji od četiri savjeta za ekološku vožnju: izbjegavati naglo ubrzavanje, izbjegavati naglo kočenje, voziti na niskom broju okretaja motora i voziti brzinom jednakom ili nižom brzini okolnog prometa. Svaki od savjeta je izražen matematičkom mjerom čija je vrijednost u rasponu od 0 do 1, gdje je 0 najbolji, a 1 najlošiji rezultat. Eko indeks je zatim izražen kao kombinacija tih matematičkih mjera te je izračunat za svakog od

vozača koristeći prikupljeni skup podataka. Drugi primjer pokazuje korištenje algoritma učenja bez nadzora k-means nad skupom putovanja sadržanim u skupu podataka kako bi se putovanja podijelila u skupine slične po svojim karakteristikama vožnje. Karakteristike odabrane za ovu podjelu uključuju podatke o brzini i ubrzanju vozila, brzini okolnog prometa, prijeđenoj udaljenosti i trajanju putovanja. Algoritam je podijelio putovanja u tri skupine od kojih prva odgovara putovanjima na međugradskim prometnicama na velikim brzinama, druga putovanjima na gradskim prometnicama s visokim vrijednostima ubrzanja i usporavanja (agresivna vožnja) i treća putovanjima na gradskim prometnicama s niskim vrijednostima ubrzanja i usporavanja (defenzivna vožnja). Konačno, kao treći primjer analize demonstriran je sustav za interaktivnu vizualnu usporedbu ruta vožnje na temelju kontekstnih informacija. Sustav omogućuje odabir dvije lokacije na temelju kojih se pretražuju sva putovanja koja su u cijelosti ili djelomično prolazila kroz odabrane lokacije. Pronađena putovanja mogu, između ostalog, biti uspoređena na temelju rute, brzine, vremena i vidljivosti.

Šesto poglavlje ("6. Conclusions and Future Work") zaključuje disertaciju diskusijom o znanstvenom i poslovnom odjeku koje imaju radni okvir i skup podataka opisani u ovom radu. Također, prezentirano je nekoliko ideja o budućim istraživačkim smjerovima. Doktorski rad ima i tri dodatka: i) prvi dodatak ("ELM327 Bluetooth-to-OBD-II Device Installation Manual") opisuje instalaciju ELM327 uređaja unutar vozila; ii) drugi dodatak ("ELM327 Bluetooth-to-OBD-II Device Smartphone Connection Manual") opisuje način povezivanja ELM327 uređaja s pametnim telefonom; te iii) treći dodatak ("Data Collection Application Manual") u kojem su opisane upute za korištenje aplikacije za prikupljanje podataka.

Konačno, očekivani znanstveni doprinos ove doktorske disertacije se sastoji od tri dijela. Prvi dio je metoda za prikupljanje i kontekstno obogaćivanje podataka iz vozila s podrškom za automobilske komunikacijske protokole više i niže razine. Ova metoda je opisana u trećem poglavlju disertacije, zajedno s korištenim izvorima podataka, komunikacijskim protokolima i razvojnim tehnologijama. Drugi dio je referentni skup kontekstno-obogaćenih podataka iz vozila, izgrađen koristeći načela otvorenih podataka i privatnosti podataka. Skup podataka je prikupljen koristeći metodu definiranu prvim dijelom znanstvenog doprinosa te je detaljno opisan u četvrtom poglavlju. Treći dio znanstvenog doprinosa je specifikacija aplikacijskog programskog sučelja za naprednu analitiku nad referentnim skupom podataka uz primjenu zadanih politika sigurnosti i prava pristupa. Specifikacija sučelja i politike sigurnosti i prava pristupa korištene u radnom okviru su opisane u trećem poglavlju, dok su primjeri napredne analitike generirani koristeći implementirano sučelje demonstrirani u petom poglavlju.

Ključne riječi: podaci iz vozila, prikupljanje podataka, kontekstno obogaćivanje, analitika podataka

Contents

Chapter 1

Introduction

Over the last decades, software has become an essential part of daily life. Working, shopping, travelling, communicating and playing are permeated and mediated by software [1]. Just as steam was the driver of the industrial age in the 19th century, software is the driver of today's information age. There is no single industry that is not at least partially controlled by software, and the automotive industry is no different. The global automotive software market is worth more than $13 billion and is growing steadily at about $100 million per year [2]. The majority of new vehicle requirements are software-related, making vehicle software an indispensable part of the modern automotive industry, especially for new technologies such as electric vehicles [3].

It is estimated that by 2023, approximately 70% of vehicles worldwide will be equipped with the necessary hardware to connect to the Internet [4], enabling automated links to all other connected objects such as smartphones, tracking devices, traffic lights and other motor vehicles [5]. The interaction with other devices and information systems creates an information-rich travel environment for the passengers and enhances the situational awareness of the vehicle [6]. Software plays a prominent part in the development of connected cars, because the information collected from other vehicles and the grid has to be processed and displayed to the driver as quickly as possible. Connected vehicles enable the collection of automotive data in real time, and for that reason they are one of the main reasons for the increasing amount of data in the automotive industry [7].

The data generated by vehicles is extremely valuable for applications such as fleet management and predictive maintenance, as it can provide insights into driver behaviour and efficiency as well as the current state of the vehicle. Collected and stored automotive data can provide valuable insights [8] and help in applications such as optimising resource planning for next-generation vehicular networks, reducing traffic density [9], improving vehicle routing, optimising electric vehicle charging station placement [10], and profiling driver behaviour [11] and driving style [12]. In addition, the answer to the growing concern about global warming and sustainability is being explored in various areas such as energy efficiency and energy consump-

tion measurement and monitoring [13].

A way of providing even better insights from the collected automotive data is to enrich it with context. Contextual enrichment encompasses the identification of additional information sources which can be used to enrich the available data and contribute to its interpretation [14]. Location and weather condition information can facilitate the generation of new insights when analysing the existing automotive data [11]. With the addition of other location-based information sources such as traffic data, researchers can come to even more useful conclusions about the vehicle and driver performance. A contextually enriched automotive data set can be useful in conducting sustainability-related research in the automotive domain. For example, some studies show that driving patterns are one of the most important factors influencing the fuel consumption of vehicles [15] and that these patterns can be evaluated when vehicle data is collected [16]. Solutions that model driver behaviour and determine the driver's eco-efficiency can be developed and validated using such a contextually enriched automotive data set.

1.1Problem Statement

Current trends that focus on vehicle connectivity, i.e. vehicle-2-infrastructure and vehicle-2-vehicle, are seen as beneficial for data collection and analysis [17]. Vehicle telematics, an interdisciplinary field combining telecommunications and vehicle informatics, is making this possible. However, collecting data from connected vehicles is not a simple task because connected vehicle hardware and software are a closed system meant to be used by vehicle manufacturers only. This means that standard connected vehicles cannot be used for real-time data acquisition without elevated access to the vehicle software and knowledge of vehicle bus and communication systems specifications. Statista researchers state that only 31% of households in the US have had at least one connected car in 2018, while that number was around 20% for European countries [18]. This is why targeting only connected cars severely limits the number of vehicles from which data can be connected.

Combining automotive data with various heterogeneous data sources can provide valuable additional insights in data analysis [19]. These sources can answer questions such as where the vehicle is moving and why it is moving in such a manner (because of traffic, weather or something else). If connected vehicles are used for data collection, elevated access to vehicle software and hardware is once again needed to add contextual data to the automotive data. Also, the number of data sources which can be used to provide context is limited as it is predefined by the automotive manufacturers.

Another issue is that collecting contextually enriched automotive data ultimately indicates the convergence to what is commonly known as big data [9]. In general, the term big data is used to define exceptionally large amounts of different data sets, or data records within these

data sets [20]. It is usually characterised by five main properties, sometimes referred to as 5Vs: volume, variety, velocity, veracity and value [21]. To create a storage platform for automotive big data, these properties must be taken into account to provide a robust, fault-tolerant and reliable solution.

With the collected and contextually enriched automotive data usually being stored in big data storage platforms [22], a middleware solution is needed to provide an abstraction layer for the stored data. This middleware solution is usually in the form of an application programming interface (API). The API should be implemented in a way that facilitates data analytics and should apply the security and access policies required to keep the privacy of the drivers involved in the data collection process.

To summarise, the automotive industry is becoming increasingly data-intensive. The key component for impactful research in this area is a high-quality source of contextually enriched automotive data. Such data can be applied in various interdisciplinary research areas, such as automotive engineering (analysis of raw automotive data), behavioural science (classification of driver behaviour and style), telecommunications (analysis of existing network infrastructure and planning of next-generation vehicular networks), and transportation science (analysis of traffic congestion). The conducted survey shows that an open source data set of contextually enriched automotive data is currently not available.

The main challenge introduced in the scope of this thesis is the collection of a contextually enriched automotive data set. To solve this challenge, a framework for the collection, contextual enrichment and advanced analytics of automotive data is proposed. Several research questions have to be answered to be able to develop such a framework. The first one is the one that the whole framework is based on, and can be formulated as:

RQ1: *How to collect automotive data in real-time, covering a broad range of diverse vehicles?*

To be able to collect data in real-time, Internet access needs to be provided to vehicles. Connected vehicles would be ideal candidates for data collection, if not for the fact that connected vehicle hardware and software are a closed system meant to be used by vehicle manufacturers only. Also, according to [18], only 31% of households in the US have had at least one connected car in 2018, while that number was around 20% for European countries. This is why targeting only connected cars does not satisfy the requirement of automotive data collection from a broad range of vehicles. Thanks to the ever-growing worldwide smartphone penetration, smartphone-based connected vehicles offer a scalable, upgradeable and cost-effective solution to this problem [23].

Smartphone-based connected vehicles are vehicles that are connected to the Internet via a smartphone device. There are several advantages of smartphone-based vehicle telematics over

implementations that use conventional vehicle telematics to collect automotive data [24]. Smartphones are cheaper and have much shorter development cycles than vehicles, which makes them a shortcut to new technologies. The use of smartphones in vehicle telematics enables data collection not only from connected vehicles but virtually any vehicle with the addition of a hardware communication module.

Once smartphones were determined as a data collection tool, two automotive communication technologies have been identified to further maximize the spectrum of vehicles from which data can be collected: a higher-level diagnostics protocol – On-board diagnostics II (OBD-II), and a lower-level bus system – Controller Area Network (CAN) bus. OBD-II diagnostic interface is mandatory for all passenger vehicles manufactured in the EU since 2003 and since 1996 in the US, making it ideal as a collection source for such vehicles. However, electric vehicles and specialized vehicles (e.g. agricultural, mining, municipal vehicles) do not have to conform to such a standard. Therefore, the most common underlying bus protocol – CAN bus was chosen as an alternative for automotive data collection.

After the automotive data collection process is defined, the second research question deals with contextual enrichment of automotive data:

RQ2: *How to enrich automotive data with context from multiple heterogeneous sources?*

Contextual enrichment is closely connected to the previous problem, as most of the contextual information can be acquired only through an Internet connection. The solution is also similar, as smartphones enable contextual enrichment of automotive data by enabling simultaneous data acquisition from vehicles and several other Internet-based data sources, as well as from the built-in smartphone sensors [25]. Location, traffic and weather data are only a fraction of the information that can be stored together with automotive data using a smartphone device.

As contextually enriched automotive data converges to big data, the third research question deals with the problem of big automotive data storage:

RQ3: *How to store the collected contextually enriched automotive data?*

The five main challenging properties defined through the big data concept, also known as **5V**, are as follows: *volume*, *variety*, *velocity*, *veracity* and *value* [26]. *Volume* is one of the most emphasized big data challenges, since the amount of data that is being generated can not be stored on a finite physical medium. The common approach to this specific challenge is to *aggregate data* or to employ *real-time data-stream analysis* which substantially lowers the amount of data stored on various physical mediums. The second challenge is defined through *velocity*, i.e., the speed of data generating process. The estimated rate of global traffic by 2018

is 50,000 GB/second [27] which is an enormous data rate to process. The sampling of the data and the scalability of the system are considered to be good practices in addressing this challenge. Finally, the last three challenge properties, i.e., *variety*, *veracity*, and *value* are all addressing certain aspect of differences in the data. The most prominent challenge in the area of data difference is handling data from a wide *variety* of sources, since they generate data in various forms, i.e., structured, unstructured, and semi-structured [28]. The *veracity* [28] in the data has become a notable challenge with the increase in the number of data generating entities. Ensuring the data reliability, authenticity and protecting the data from unauthorized access and modification become much bigger challenges when the data is collected from many different sources [29]. The last challenge in the *data difference* aspect of *big data* is handling data records with no useful *value*, i.e., information [29]. This can be tackled with advanced methods of data pre-processing where they are either discarded or transformed into a format appropriate for further analysis.

The data platform for the storage of automotive big data must be designed in a way to address all the aforementioned challenges. Specific technologies must be targeted in order to ensure that the platform is scalable, robust and fault tolerant. The most common approach is to distribute the data platform components over multiple machines, with each of them being responsible for a particular functionality such as load balancing, data ingestion, reliability insurance and finally, data storage.

Once the contextually enriched automotive data is successfully stored, it can be analysed to generate new insights. Therefore, the fourth research question tackles the analysis of the contextually enriched data:

RQ4: *How to generate insights from contextually enriched automotive data?*

By adding contextual information to the automotive data, interesting possibilities are opened up for data analysis. Contextual information can provide useful in several different areas such as transportation sustainability, behavioral science and telecommunications. The goal of the analysis is to identify the methods and techniques which can be performed over the existing data to create new knowledge. Data science can be extremely useful when performing such analysis. To be able to successfully evaluate the identified analysis methods and techniques, it is paramount to understand the domain and potential use cases of the data set. This is where the performed survey of the related research can be of value.

Finally, after the data has been collected, enriched, stored and analysed, it can be offered to consumers to access. The fifth and final research question is oriented towards providing access to the stored and analysed automotive data:

RQ5: *How to provide access to the collected automotive data, applying the required security and access policies?*

APIs are a common solution for providing access to stored data. There are multiple benefits of employing an API for accessing the data instead of accessing the data directly from the database. Firstly, using an API means that the database software can be changed at any time, and all of the possible application consuming the data through the API will not have to change, only the API will need to be modified. Secondly, the API can be used by multiple clients who do not have to write their own queries for fetching the data. Next, the API allows the control over which data collections can be accessed through it and in what format. Finally, the API can be designed in a way which facilitates statistical and analytical calculations. This way insight generation can become a part of the API.

1.2Thesis Outline

After the introductory Chapter 1, which explains the context of the research and introduces the research problem through research question, Chapter 2 provides a detailed review of the *state-of-the-art* research regarding selected topics on automotive information and communications technology (ICT) which are relevant to this thesis and a review of papers dealing with automotive data manipulation. At the end of the Chapter 2, a reflection on the relevant reviewed studies is provided and a proposed solution is stated.

Chapter 3 explains in detail the collection and contextual enrichment of automotive data implemented in the framework, which is the focal point of this thesis. In this chapter, each major module of the framework is explained in detail. Chapter 4 describes the contextually enriched automotive data set collected using the framework, and which is then used as a referent data set for data analysis. In Chapter 5, several examples of data analytics are applied on the collected data set, and the new insights generated by the analytical procedures are explained.

Finally, Chapter 6 concludes the thesis and demonstrates ideas for future work that would improve the proposed framework's functionalities, performance and tackle the identified limitations.

Chapter 2

State of the Art

This chapter presents an overview of the topics considered most relevant to this thesis. First, an introduction to some selected topics on automotive ICT is provided, explaining the context behind the research domain of this thesis. Afterwards, literature dealing with manipulation of automotive data is examined to demonstrate the current state of research regarding the collection, contextual enrichment, storage and analytics of automotive data. Finally, the presented state of the art is synthesized with the aim of deriving a solution for the previously raised research questions.

2.1Selected Topics on Automotive ICT

This section provides an introduction to the topic of automotive ICT. Historical evolution of the automotive software development is presented, leading up to the vehicles of today in which software is crucial for their operation and the generation of automotive data. Afterwards, the ICT architecture of the modern vehicle is described, with the emphasis on softwarized, connected and autonomous vehicle components.

2.1.1History of Automotive Software

To get a sense of how software and data became essential to modern vehicles, it is beneficial to take a look at the historical trends in the automotive industry. For that reason, a brief overview of the history of the automobile and automotive software is provided in this section. Although automobiles can be categorized in many different ways, four properties related to the increasing amount of software and data in automobiles have been selected to be used as a basis for classification:

- *electric* (i.e. vehicles propelled by one or more electric motors);
- *softwarized* (i.e. vehicles running any kind of automotive software);

- *connected* (i.e. vehicles capable of connecting to the Internet and communicating with smart devices, other cars and infrastructure);
- *autonomous* (i.e. vehicles capable of driving without human input).

Each of the selected automobile properties will be examined more closely, and some of the most recognizable and important cars that fall into the properties' category will be described.

Early history of the automobile

At the turn of the 20th century, cars propelled by internal combustion engines (ICEs) pioneered by Karl Benz and Gottlieb Daimler became prevalent over electric and steam cars. The popularity of electric cars was hampered by a lack of battery charging infrastructure [30], while steam cars needed around 45 minutes to start [31] and lost their popularity after the invention of the ICE's electric starter. In 1908, Henry Ford launched the Ford Model T, the first automobile to be mass-produced on a moving assembly line. The Model T was an instant success due to its general utility, fine performance and price. It was produced from October 1908 until May 1927 for an overall production of 15 million cars [32], making it one of the best selling cars in history. In the years after the Model T, automotive advances such as automatic transmission, air conditioning, electronic fuel injection, seat belts and airbags were introduced [33]. One of the most important models of the era which succeeded Ford T is Toyota Corolla. Introduced in 1966 and still produced today, it is the best selling nameplate of all time because of its reasonable price and reliability, with an overall production of more than 40 million cars [34]. It is worth pointing out that both Ford Model T and the 1966 Toyota Corolla were not softwarized or electric; they were completely mechanical and propelled by an ICE.

Electric cars

Although being pushed out of the market by ICEs at the start of the 20th century, electric cars started making a comeback after the Arab oil embargo in 1973-74 [35]. The most prolific electric car of that era was the CitiCar, produced in around 2,600 units from 1974 to 1976 by Sebring Vanguard. It had a plastic, two-seater body with batteries supplying power to a 3.5-horsepower electric motor [30]. But again, with the falling price of oil the electric car sales plummeted. The next resurgence of electric cars came in the mid-90s, primarily driven by environmental concerns. In 1996, General Motors presented the EV1; an EV based on lead acid batteries propelled by an induction motor. It was the first mass-produced purpose-designed EV of the modern era from a major automaker [35]. Around the same time, hybrid electric vehicles (HEVs) were introduced as another alternative to ICEs. Today's best selling and best known HEV is Toyota Prius, launched in 1997 as the first commercial HEV [36]. HEVs combine a conventional ICE system with an electric propulsion system. They provide improvements over ICE vehicles in the form of regenerative braking, shutting off the ICE when the car is stopped,

allowing a smaller and more efficient engine, and not requiring the engine to follow the driving cycle closely. This results in improved fuel economy and lower greenhouse gas and tailpipe emissions [37], which is one of the main reasons behind the creation of HEVs and will be the main reason for the expected displacement of ICE vehicles with EVs.

Softwarized cars

The first softwarized car was the 1977 Oldsmobile Toronado. It featured the first production automotive microcomputer ECU - a single-function controller for electronic spark timing [38]. In 1978, General Motors offered a trip computer which displayed speed, fuel, trip and engine information as an option on the Cadillac Seville. It was also used to test how well a microprocessor could control multiple functions such as port fuel injection, electronic spark timing and cruise control. By 1981, GM was using microprocessor-based engine controls executing about 50,000 lines of code across the entirety of its domestic passenger car production with other car companies quickly following suit [38]. The hardware/software systems in the first softwarized cars were growing *bottom-up*, with local, isolated and unrelated software-based solutions. Cars had dedicated ECUs for different tasks as well as dedicated sensors and actuators. To optimize wiring, bus systems were introduced, starting with the CAN bus [39] in 1991 by which the ECUs became connected with sensors, actuators and one another [40]. Gradually, functions distributed over several ECUs were added, connected by bus systems as the underlying communications infrastructure [41]. Communication architectures of today's cars consist of multiple types of communication bus technologies providing different functionality, from advanced control to entertainment [42]. The amount of software in today's cars contains close to 100 million lines of software code on 70 to 100 ECUs networked throughout the body of the car [38].

Connected cars

The possibility of enabling communication between cars, the Internet and other connected devices, in order to access various data sources, has been of interest in the last several decades. To enable such communication, vehicle telematics, an interdisciplinary field combining telecommunications and vehicle informatics, is being used. Today, connected cars are integrating the data collected by internal car sensors with the information gathered from the web and the surroundings [43], with a goal of creating a cleaner, more secure, and more proficient driving experience. The first attempt at creating a connected car was the General Motors' OnStar Project. Introduced in 1996 as an option on some Cadillac models, it provided services limited to safety and security such as crash notification, in which the call center is notified if the vehicle crashes or an airbag is deployed [44]. Today's connected cars are capable of accessing the Internet anytime, interacting with other smart devices on the road or in mechanical shops, interacting with other vehicles and are equipped with a set of modern applications and dynamic contex-

tual functionalities offering advanced infotainment features to the driver and passengers [45]. Nissan Leaf, the world's all-time best-selling highway-capable electric car in history [46] was also the world's best connected car in 2010 according to UK's Car Magazine [47]. Since the start of its production, it had an 'Intelligent Transport' system which displayed nearby charging stations with directions and a smart phone app that could turn battery charging, air conditioning or heating on and off as well as show the current battery level.

Autonomous cars

The next major wave of automotive technology innovation are autonomous vehicles (AVs), vehicles capable of moving with little or no human input. Reliable communication and sharing vehicle information with other vehicles and the infrastructure provide a basis for the creation of AVs. Already, most new cars are equipped with sensors and connected to high-speed wireless networks, transmitting streams of valuable data and facilitating a wide range of digital services (e.g. mobility information, infotainment) [5]. AVs use that data to "sense" their surroundings and drive the vehicle without human input. One of the key technologies that enables autonomous driving is *drive-by-wire*, the control of a vehicle's motion systems through electronic means. AVs have the potential to alter transportation systems by averting deadly crashes, increasing road capacity, saving fuel, lowering emissions and turning vehicles from owned products to an on-demand service [48].

Although over 14 companies are currently developing autonomous and semi-autonomous vehicles [49], most of them are not yet offering the vehicles for purchase. Tesla has been taking a more aggressive approach, rolling out advanced automation features in its commercially available vehicles, led by their Model S. Tesla Model S is currently a semi-autonomous vehicle, equipped with hardware capable of "full self-driving". The fully-autonomous Model S coast-to-coast demonstration drive accross the U.S. was scheduled for the end of 2018 [50], but is still not realized as of the time of writing this thesis. Current automation software features that ship with the Tesla Model S are adaptive cruise control, autosteer, auto lane change, navigation system and a summon mode which enables parking while outside of the vehicle, using the key fob or a smart phone [49].

Another company developing autonomous vehicles is Waymo, a subsidiary of Alphabet Inc, the parent company of Google. Unlike Tesla, they focus on deploying autonomous vehicles in a ride-sharing service. In 2018, Waymo One became the first autonomous ride-sharing service worldwide operating without any safety drivers in the car [51]. Waymo cars are built for full autonomy with sensors that give 360-degree views and lasers that detect objects up to 300 meters away. Short-range lasers detect and focus on objects near the vehicle, while radar is used to see around vehicles and track objects in motion. However, Waymo still operates in only several American cities, and needs detailed mapping of the territory (from lane markers to stop

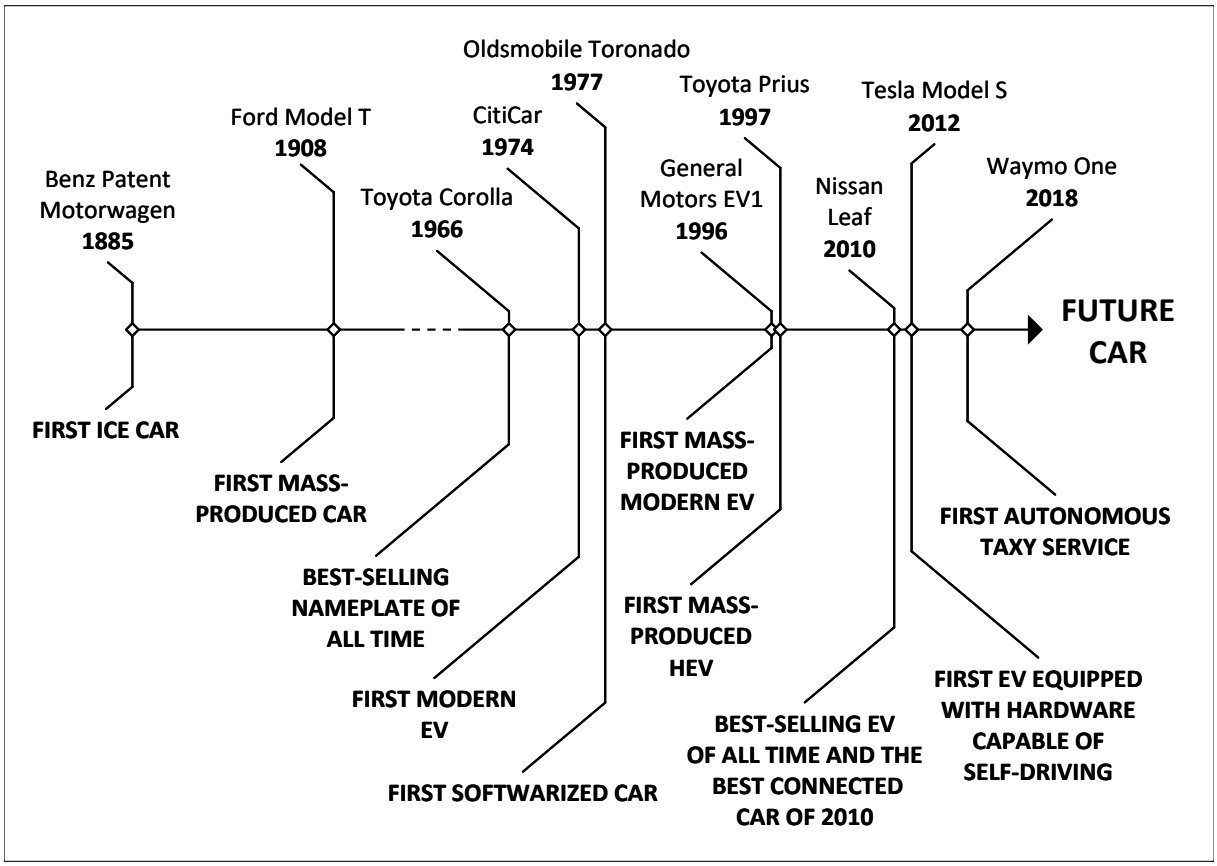

Figure 2.1: The timeline of the automotive evolution: from the first ICE car to connected, autonomous, electric and softwarized cars of the future

signs to curbs and crosswalks), before it starts operating in a new area [52].

Future cars

Since the end of the 1970s until today, software has become an essential part of automobile manufacturing to the point that modern cars cannot function without it. EVs are already being considered as the future of transportation, as the need to reduce greenhouse gas emissions is becoming more urgent [53]. Connected and autonomous cars are expected to be the next step in the automotive evolution, as they are the key to creating next generation intelligent transportation systems [6]. Vehicle manufacturers such as Mercedes-Benz [54], Renault [55] and Volvo [56] are already preparing for the future with their concept vehicles, and as the trends in automotive evolution (Figure 2.1) - from the first mass-produced car in Ford Model T, to the electric, semi-autonomous Tesla Model S - all indicate, that vehicle will be electric, connected and autonomous (Figure 2.2), with software as the key driver behind the advancements.

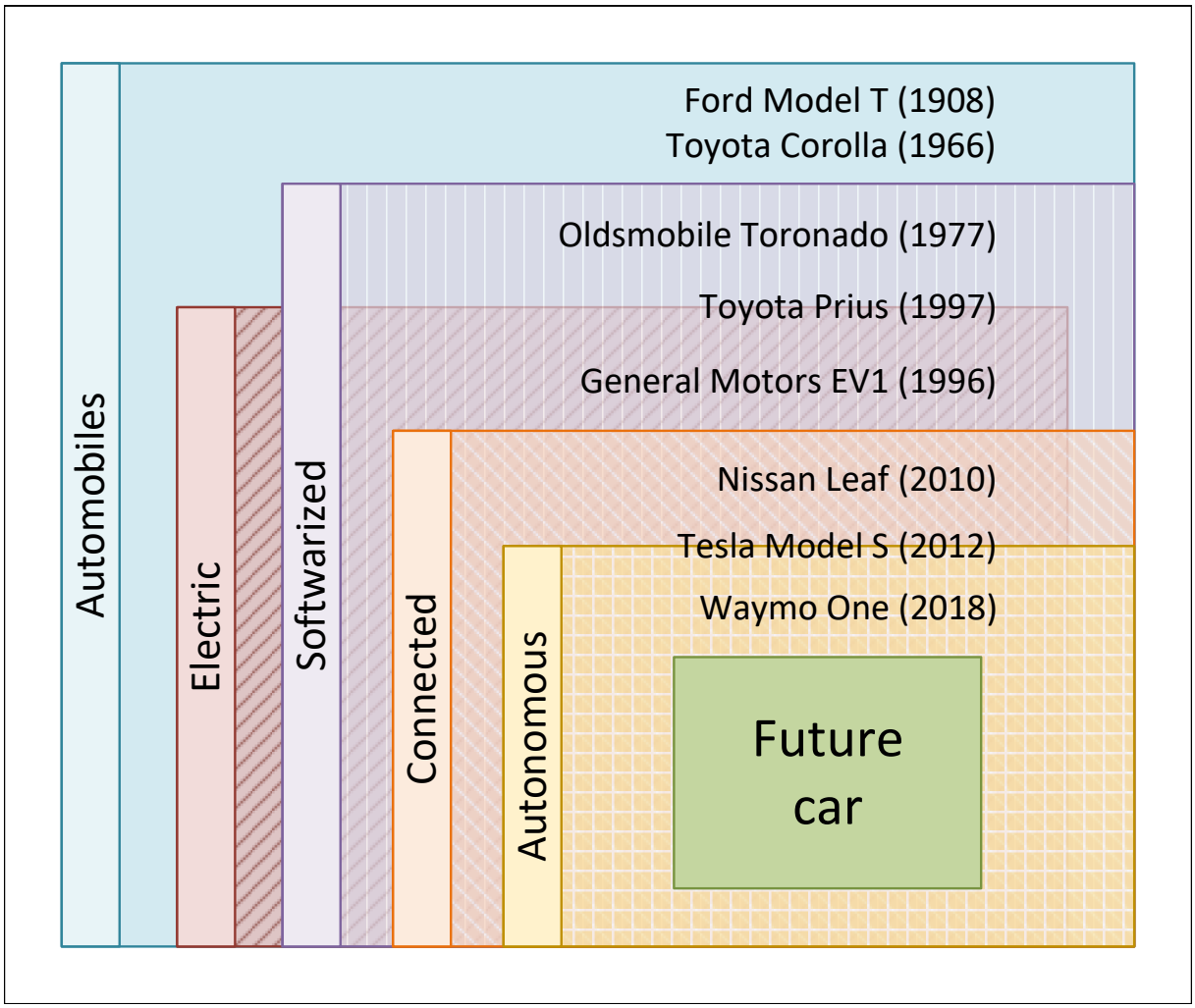

Figure 2.2: Automotive evolution viewed through four properties: electric, softwarized, connected, autonomous

2.1.2ICT Architecture of Modern Vehicles

The vast majority of innovations in vehicles is driven by ICT in the form of electronics and software, and in case of modern vehicles, ICT becomes the backbone of all relevant system functions. In the following, an overview of the software and ICT architecture found in modern vehicles will be provided, with a special emphasis on vehicle components running automotive software. The vehicle architecture will be divided into three groups related to the research area of this thesis: *softwarized*, *connected* and *autonomous*.

Softwarized Architecture

In modern vehicles, automotive software is the foundation of all functions. It would be impossible to count all the vehicles components running automotive software, so a high-level approach will be taken, providing an overview of automotive electronics, bus systems and software architecture. Well-defined software architecture is especially important in automotive software

design, as it helps to understand and predict the performance of the system by examining all of its subsystems, components and modules. Software and electronics in automotive systems go hand in hand, which is why software architecture cannot be viewed as stand-alone, without being connected to vehicle electronic systems.

Electronic Control Units (ECUs) are embedded systems on which software that controls one or more electronic systems or subsystems in a vehicle is executed. Each ECU is responsible for performing a certain vehicle function. Assembly language or the C programming language can be used to program the ECUs [57], which, along with the software running on them, form the electrical and software system of the car. When electronics was first started being deployed in vehicles, the operation of ECUs was largely autonomous with no interaction among different ECUs.

The networking of different ECUs was made possible with the introduction of *automotive bus systems*. Automotive bus systems interconnect components inside a vehicle, allowing multiple access to various sensors and enabling higher-level software functions that run on several subsystems. They can be divided into four different communication groups based on their application area and technical properties:

•event-triggered bus systems (e.g. CAN);

•time-triggered bus systems (e.g. FlexRay);

•subnetworks (e.g. LIN);

•multimedia bus systems (e.g. MOST).

The comparison of the mentioned representatives of each automotive bus system communication group can be seen in Table 2.1.

Table 2.1: Comparison of automotive bus systems

Bus System	**CAN**	**FlexRay**	**LIN**	**MOST**
Type	event-triggered	time-triggered	subbus	multimedia
Application example	ABS, battery management system, engine control	drive-by-wire systems	door locking, powered windows	entertainment, navigation
Bandwidth	up to 1 Mbit/s	up to 10 MBit/s	up to 20 kbit/s	up to 24.8 Mbit/s
Access control	CMSA/CD	TDMA	Polling	TDM, CSMA/CA
Error protection	CRC, parity bits	CRC, bus guardian	checksum, parity bits	CRC, checksum, parity bits

Control Area Network (CAN) is an event-triggered bus system mainly used for soft real-time in-car serial communication between ECUs, networking for example the antilock braking system (ABS) [58] or the battery management system in electric vehicles [59]. It enables data

rates up to 1 Mbit/s and broadcasts messages to all connected nodes which independently decide whether to process the message or not. CAN ensures the quickest transmission of top priority messages using CSMA/CD (Carrier Sense Multiple Access / Collision Detection) access control method and it detects transfer errors using cyclic redundancy check (CRC) and parity bits, and initiates retransmission of affected messages [58].

FlexRay is a time-triggered bus system developed in 1999 for future safety-relevant high-speed automotive networks, targeting highly safety relevant applications such as drive-by-wire and powertrain [58]. It supports data rates up to 10 Mbit/s and up to 64 connected nodes. It uses the cyclic TDMA (Time Division Multiple Access) that provides fixed time slots during which devices can access the bus and mini-slots which can enlarge if devices have additional data to be sent that doesn't fit in their designated time slot [60]. The error tolerance is achieved by redundant channels, CRC and a bus guardian that detects and handles logical errors [58].

Local Interconnect Network (LIN) is a local sub network usually connected as a sub-bus to a more powerful bus, such as CAN. It has been developed for less complex networks where the bandwidth and versatility of CAN bus is not required, such as, automatic door locking mechanisms, power-windows and temperature and rain sensors [58]. LIN supports data transmission of up to 20 kbit/s and error protection in the form of checksums and parity bits.

Multimedia bus systems such as *Media Oriented Systems Transport (MOST)* are responsible for the transmission of high-quality audio, voice and video data streams used for in-vehicle entertainment. MOST features a data rate of up to 24.8 Mbit/s in synchronous and 14.4 Mbit/s in asynchronous transmission mode with a 700 kbit/s control channel [60]. Access control in synchronous mode is realized by TDM (Time Division Multiplex) and by CSMA/CA in asynchronous mode. An internal MOST system service is in charge of error management, detecting errors over parity bits, CRC and checksums.

Vehicle electronics and software architecture can be viewed as multiple subsystems comprised of different ECUs that communicate over bus systems. Although each subsystem is responsible for a certain part of the vehicle, they communicate with one another, distributing some of the more complex functions over several subsystems and ECUs. The main subsystems found in vehicles are:

- powertrain subsystem;
- chassis subsystem;
- body subsystem;
- multimedia subsystem.

The *powertrain subsystem* is in charge of the components that generate power and deliver it to the road surface; e.g. motor, motor controller, and in the case of an electric vehicle, battery pack and battery management system. It has a relatively small number of user interfaces; the start/stop switch for starting and shutting off the vehicle and the accelerator pedal for increasing

the speed of the vehicle. Along with controlling the powertrain according to user inputs, software functions of the powertrain subsystem include collecting information about the working parameters of the motor, and notifying the driver via information systems if anything is out of order. The throttle-by-wire system, needed for the control of autonomous vehicles without human input, is also implemented in the powertrain subsystem of the vehicle.

The *chassis subsystem* encompasses vehicle's axles and wheels, brakes, steering system, suspension and shock absorbers. It includes systems such as the antilock braking system (ABS), electronic stability program (ESP), parking brake, tire pressure monitoring system, suspension, power steering, brake-by-wire and steer-by-wire systems [61]. Functional failure of any of the aforementioned systems can put the passengers' lives at risk, which is why the safety requirements for the chassis subsystem are very stringent, with sensitive monitoring mechanisms in place for each of the safety critical systems. Similar to the powertrain system, the chassis subsystem also has a small number of user interfaces; the brake pedal, the steering wheel and the parking brake.

The vehicle software and electronics responsible for passive safety and comfort and convenience are included in the *body subsystem* [61]. The *passive safety systems* are all onboard systems in charge of protecting the safety of vehicle's passengers (e.g. airbags, seat belt sensors and tighteners). The *comfort and convenience systems* include the central locking system, powered windows, wipers, heating and air conditioning, mirror adjusters, headlamp controls, parking aid features and other similar auxiliary systems which improve the passengers' travel comfort. Practically all of the comfort and convenience systems provide some kind of user interface, while passive safety systems usually provide none. The body subsystem contains the largest number of independent software functions due to the amount of different functionalities implemented in the subsystem.

The *multimedia subsystem* includes the instrument cluster and the infotainment system which provides added comfort value to the passengers. The instrument cluster usually displays vehicle information such as current speed, engine RPM and indicators about vehicle status, and it must comply with regulations and laws [62] to ensure that the information is properly displayed to the driver. To be able to display this information and to provide controls to elements from other subsystems, the ECUs of the multimedia subsystem are networked with the ECUs from the powertrain, chassis and body subsystems [61]. Infotainment system providing audio and video entertainment, voice control and smart phone integration is also a part of the multimedia subsystem. Examples of modern vehicle infotainment platforms are the QNX Car platform by Blackberry, Windows Embedded Automotive by Microsoft, GENIVI by GENIVI Alliance, Tizen IVI and Automotive Grade Linux by the Linux Foundation, and Android by Google [45].

Connected Architecture

Connected vehicle is a vehicle capable of communicating with other vehicles, the infrastructure, smart devices and, most importantly, the Internet. A connected vehicle can exchange information by several different types of communication - vehicle to vehicle, vehicle to road infrastructure, vehicle to internet and vehicle to smart device.

Vehicle To Vehicle (V2V) communication is the direct transmission of data between two or more vehicles. V2V communication can be used for accident avoidance, route optimization, multimedia information sharing and social interaction [45]. Communicating vehicles form a vehicular ad hoc network (VANET) which changes constantly and quickly. VANET enables communication between moving vehicles using dedicated short range communication (DSRC) [63], a short-range wireless communication channel designed for automotive use, standardized by the IEEE 802.11p standard, also known as WAVE (Wireless Access in Vehicular Environments) [64].

The WAVE system is also used in the *Vehicle To Road Infrastructure (V2I)* communication. The main system components enabling V2I communication are onboard units (OBUs) and roadside units (RSUs). An OBU is a device mounted on a vehicle used for exchanging information with RSUs (or with other OBUs in case of V2V communication) [63]. The RSUs are usually fixed along the side of the road or on light poles, traffic lights and road signs. RSU's main functions are extending the range of VANET by redistributing information from OBUs to other OBUs, running safety applications such as accident or work zone warnings, and providing Internet connectivity to OBUs [63].

Internet communication is the fundamental requirement for a connected vehicle. There are three different solutions by which a vehicle can achieve an Internet connection: an embedded solution, a tethered solution and an integrated solution [45]. In the embedded solution, a SIM card is permanently installed in the vehicle during the manufacturing process. The module carrying the SIM card in the embedded solution is called the Telematics Control Unit (TCU). The tethered solution offers a modem that allows the insertion of a personal SIM decoupled from the vehicle, while keeping the application intelligence inside the vehicle. In the integrated approach, both intelligence and Internet connectivity are provided by a user's smart device, with the applications running on the smartphone displayed on the vehicles interface.

Today, vehicle-to-Internet connection is most commonly achieved through 4G mobile communication. Connected vehicles generate high amounts of data, and with more and more vehicles becoming connected every year, the amount of generated data is set to increase. Vehicle manufacturers and telecommunications companies such as Audi, BWM, Deutsche Telekom, Ericsson and Vodafone have already started to prepare for the increase in vehicle generated data by forming the Fifth-Generation Automotive Association (5GAA) [65]. 5G mobile networks have reduced latency, massive device connectivity and are able to handle much more data, and

the 5GAA aims to use this technology to address society's connected mobility needs.

Finally, connected vehicles need to offer a way to communicate with the user's personal smart devices. The *smart device integration* is usually achieved by a Bluetooth or Wi-Fi connection to the infotainment system. The infotainment system provides a user interface (UI) and the radio/entertainment/media functionality [66]. Current trends in infotainment systems steer away from embedded applications capable of running on only one system, and towards multiplatform applications capable of running on systems adopted by more than just one car manufacturer. The most widespread solutions for smart device integration are MirrorLink by Car Connectivity Consortium, AppLink by Ford, Apple CarPlay by Apple and Android Auto by the Open Automotive Alliance [45].

Autonomous Architecture

An autonomous or a self-driving vehicle is a vehicle capable of perceiving its surroundings and navigating itself without human intervention. For autonomous driving to be made possible, a vehicle must be equipped with electronic control systems and surrounding detection sensors, along with complex autonomous driving algorithms, including:

- •perception - sensing the surrounding environment using various types of sensor techniques such as RADAR, LIDAR and computer vision;
- •localization - finding the car's position using the Global Positioning System (GPS);
- •planning - determining the behaviour and motion of the vehicle based on information from perception and localization;
- •control - executing the planned function by steering, accelerating and braking;
- •system management - supervising the driving system with functions such as fault management and logging [67].

The information on surrounding environment is collected using several types of sensor components. *RADAR* system uses radio waves to determine the range, angle and velocity of objects. The *LIDAR* system creates a digital 3D representation of the surrounding objects by illuminating them with pulsed laser light and measuring the reflected pulses with a sensor. *Cameras* and *computer vision* are then used to recognize and distinguish various visual objects [68]. Localization is achieved by a *GPS system* using additional information such as vehicle motion sensors, environment perception data and digital maps to account for the errors that can occur in bad GPS satellite signal conditions. Planning algorithms are usually embedded in the vehicle's autopilot functionality and achieved entirely by software functions. Autopilot algorithms are divided in three stages: global routing (i.e. finding the fastest and safest way from the initial position to the goal position), behaviour reasoning (i.e. assessing the driving situation and determining the behaviour of the autonomous vehicle based on the global route and perception information), and local motion planning (i.e. avoiding static and dynamic obstacle collision)

[68]. Autopilot software in autonomous vehicles also has some interesting *ethical requirements* as it needs to replicate the human decision making process [69].

The key technology enabling the control of an autonomous vehicle is *drive-by-wire*. Drive-by-wire is referring to the replacement of mechanical or hydraulic systems, such as acceleration (throttle-by-wire), braking (brake-by-wire) or steering (steer-by-wire), by electronic ones [70]. Additional ECUs are added so that the software in autonomous vehicles can perform all driving functions. Naturally, drive-by-wire is a highly safety-critical system which is why it must be designed in a fault-tolerant fashion, with control system redundancies. For a critical drive-by-wire system, a management system must ensure that a system failure does not lead to endangerment of human lives or the vehicle environment, and that a failure of a single component does not lead to the failure of the whole drive-by-wire system [70].

Holistic Overview of the ICT Architecture

After a detailed analysis of each of the four architecture groups, it is useful to take a look at the ICT architecture as a whole to understand how various components interact with one another and how they fit in the complete system.

The complete ICT architecture of modern vehicles can be seen on Figure 2.3. All three

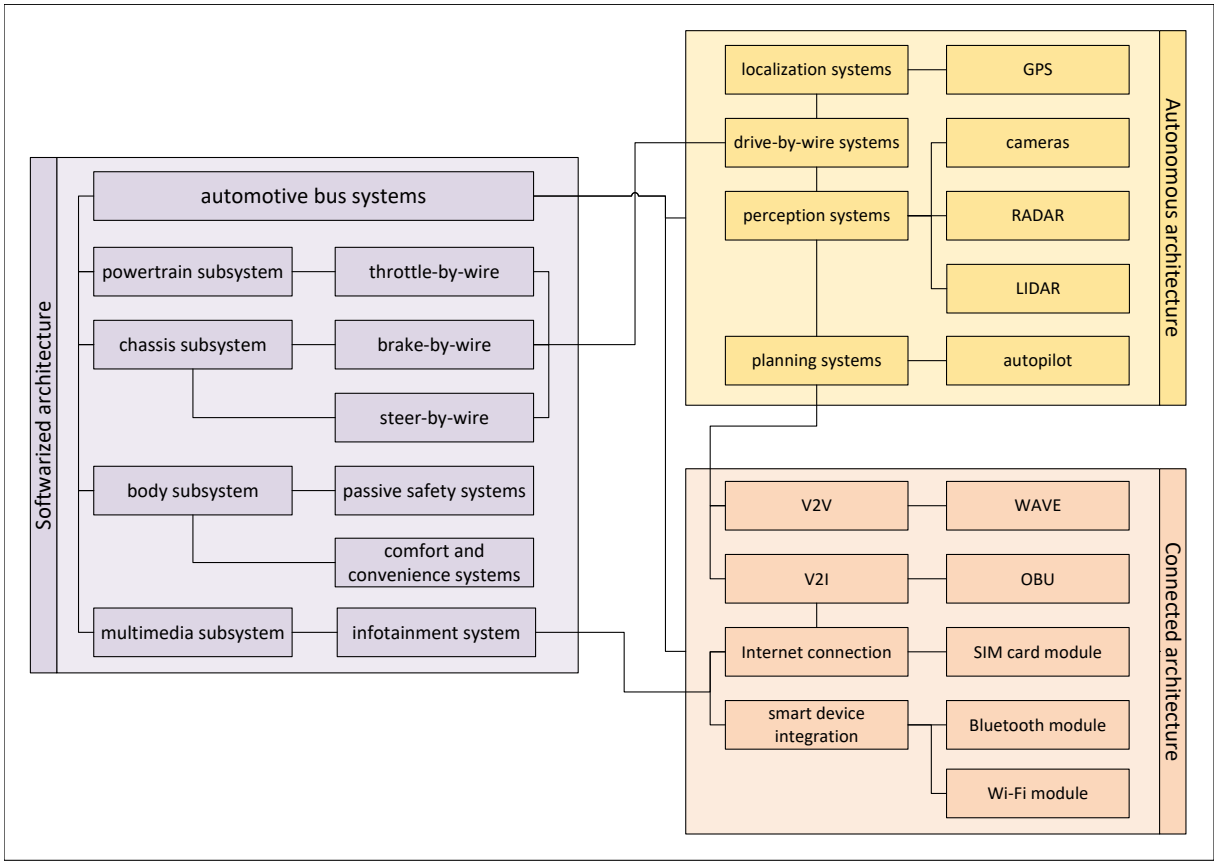

Figure 2.3: ICT architecture of a modern vehicle

groups and components within them are interconnected and exchange information via automotive bus systems. Subsystems inside the *sofwarized architecture* group receive user input from the driver and the passengers, and control the components which are under their responsibility in line with the user input. The infotainment systems found in modern vehicles are dependent on the communication components installed in the vehicle which are a part of the *connected architecture* group. The *connected architecture* group is responsible for communication with the Internet, smart devices, other vehicles and the infrastructure. V2V and V2I technology is closely connected to the planning systems inside of the *autonomous architecture* group that uses the input from the surrounding vehicles, infrastructure, on-board cameras and sensors to determine vehicle behaviour. After a suitable behaviour is calculated, the vehicle is controlled via drive-by-wire systems found in the powertrain and chassis subsystems of the *softwarized architecture* group.

The ICT architecture of modern vehicles is a complex system in which components must work in unison, but at the same time be independent enough to not allow the entire system to fail if a single component fails. Not to compromise the vehicle's safety, the historical evolutionary development of vehicle architecture avoided dramatic changes for as long as possible. Today's automotive manufacturers are leaving legacy concepts behind and taking a holistic approach for handling all major activities in vehicle development. The advance in vehicle ICT is also the main reason behind the increased amount of automotive data. Automotive data is beneficial for multiple different applications including fleet management and predictive maintenance. However, to be able to utilize it, several problems need to be solved dealing with the collection, storage and analytics of automotive data.

2.2Automotive Data Manipulation in Literature

To understand the relevance of the thesis topic, an overview of the literature dealing with the subject of automotive data manipulation is provided in the following sections. More precisely, papers dealing with the collection, contextual enrichment, storage and analytics of automotive data are examined as a part of this literature review.

2.2.1Automotive Data Collection

The research papers dealing with automotive data collection usually provide support for automotive data collection from higher-level automotive communication protocols such as On-Board Diagnostics (OBD). Reininger et al. [71] described a prototype smartphone application that collects vehicle data from OBD-II through an OBD-II-to-Wi-Fi module. The data is stored in a non-relational database on the smartphone and sent to the server when the Wi-Fi module is disconnected. Since the network connection is busy receiving automotive data, this solution

cannot simultaneously forward the automotive data to a remote server without caching it. To mitigate this problem, Albertengo et al. [72] used a Bluetooth module to transmit OBD-II data to the smartphone, eliminating the need to cache the data on the device before sending it to the cloud. Their work has shown that it is possible to build a platform for intelligent transport services, based on low-cost, easy-to-deploy devices connected to existing cellular networks.

Bluetooth-enabled OBD-II devices are a de facto standard for automotive data collection via smartphone based telematics. Zaldivar et al. [73] used such a device to collect data such as vehicle speed, RPM and fuel level and send it to an emergency services database or to other third parties defined by the user if an accident is detected. The application also offers general purpose information to the driver, including gas levels, detection of failures in mechanical elements and extensive engine feedback data. In a similar research, Sathyanarayana et al. [74] collected vehicle speed, steering wheel angle, engine RPM, and gas/brake pedal pressure information from the OBD port and used it to perform maneuver recognition. They conclude that portable devices could be effectively used for the maneuver recognition, as their research yielded a high accuracy of 89%.

Handel et al. [75] explored the use of OBD data collection in insurance telematics. Insurance telematics is is a type of telematics which calculates an insurance premium based not only on static measures like the drivers age, occupation or place of residence, car model and configuration, or expected mileage over the policy period, but also on dynamic measures like actual mileage, time spent on the road or the time of day when the trip is being made, location, and the driver's actual style of driving. Insurance companies can offer their customers a measurement probe for the OBD outlet which captures vehicle data and sends it to their servers for processing, eventually reducing premium costs for safe drivers.

On the other hand, collection of automotive data from lower level automotive communication protocols such as can Controller Area Network (CAN) is much rarer. Marx et al. [76] compared different hardware/software solutions and conversion methods for CAN bus data collection. They collected data from agricultural vehicles such as tractors to quantify differences between the collection methods and to show potential data accumulation rates. The data they collected was stored in log files and was not available for real-time analysis. They concluded that automotive data could indeed be collected in this way, with the error rate being less than 0.07%, which does not exceed any scientific field data analysis criteria. Wang et al. [77] collected CAN data to identify the Big Five personality traits (i.e. openness to experience, conscientiousness, extraversion, agreeableness and neuroticism) of the drivers through driving signals, comparing their findings to previously collected demographics and questionnaire scores. All participants in the study drove the same vehicle, and the data was collected via a data logger installed in the vehicle. As in work described above, the collected data was not available for real-time analysis. Among the papers examined, no data collection solutions that support both

higher- and lower-level automotive communication protocols were found, which is a problem that the framework presented in this thesis aims to solve.

2.2.2Automotive Data Contextual Enrichment

Not many examples of contextual enrichment of automotive data were encountered among the surveyed papers. There are some research papers that use contextual enrichment of mobility data sets that do not necessarily consist of automotive data. Sila et al. [8] enriched GPS data describing human mobility with contextual spatial information to identify types of places where people spend time to socialise on the transport routes they use. They proposed a framework for the identification of dynamic (travel modes) and static (significant locations) behaviour using trajectory segmentation, data mining, and spatio-temporal analysis, and used the framework to create a contextually enriched data set that supports new ways of modelling human movement behaviour. Some researchers employ contextual enrichment methods on automotive data set, but they do it with specialized equipment. In a research performed by Sathyanarayana et al. [74], automotive data was enriched with information from a GPS receiver, digital accelerometer, gyroscope and compass, ambient light and proximity sensors in order to perform automatic driving maneuver recognition.

Zaldivar et al. [73] enriched automotive data collected via the OBD-II port with location information using an Android device and used it for real-time accident detection. Once the accident is detected, the solution executes an automatic call to an emergency operator, which sends an ambulance or other rescue services to the accident location. Castignani et al. [11] presented an evaluation study using smartphone sensor data contextually enriched with weather information to profile driver behaviour. Their mobile application was able to accurately detect risky driving events and to distinguish between aggressive and calm drivers based on the contextually enriched data set. Albertengo et al. [72] and Reininger et al. [71] described a connected vehicle architecture using smartphones for data collection and contextual enrichment. Both pieces of research enriched OBD-II data with location information collected by the smartphone device and demonstrated that it is possible to build a cost-effective, easy-to-use platform for connected vehicles using smartphones.

2.2.3Automotive Data Storage

Since the framework presented in this thesis aims to collect and store automotive data, solutions for big data storage are also examined, with special emphasis put on solutions for automotive data storage.

Being aware of many possible technologies and solutions for dealing with the main challenges in big data, each offering different characteristics related to scalability, flexibility, reli-

ability and other nonfunctional requirements, Oussous at al. [20] presented a paper in which they discussed and compared some of the recent technologies concerning big data. For the purpose of clear understanding of the connections between technologies, solutions are categorized according to both their different layers of usage (i.e. data storage layer, data processing layer, data querying layer, data access layer and management layer), and their features. On the other hand, Gandomi and Haider [78], discussed appropriate and efficient analytic methods to deal with large volumes of heterogeneous data, pointing out several dimensions of big data other than size that are equally important in defining big data. In the paper, the emphasis is put on dealing with unstructured text, audio and video formats, with the reflection on predictive analytics for structured data. Both of the papers emphasize wide application of big data concepts and agree on the importance of choosing the most appropriate methods, taking into account parameters such as efficiency, scalability, robustness, performance, reliability, cost, security and many others, depending on the specific application.

For the purpose of better understanding and more detailed implementation description, it is interesting to examine the papers describing specific data platforms (i.e., with the emphasis on collecting data or describing an approach for storing big data). Kiran et al. [79] presented an implementation of the lambda architecture which enables both online and batch data processing for immense volumes of sensor data within a single framework. The paper presents an overview of the lambda architecture and its usage with Apache Storm and Hadoop, as well as implementation challenges. The processing of the router sensor data on the ESnet network data has been presented as a working example of the proposed solution. It is mentioned in the paper that many approaches are limited by capability, expense and resources so they put emphasis on designing the solution characterized by the high throughput, cost effectiveness, minimum requirements on network maintenance and responding to requests of dense and intense data demand in the form of a services. As far as storing big data is concerned, Membrey et al. [80] proposed an interesting stream oriented, disk based storage solution designed for storing high throughput data from low-latency sources. The paper focuses on a use case study from a financial company, hence it tackles requirements for such database (i.e. accepting data at high speed, deterministic insertion of the data, coping with burst traffic, executing queries in a reasonable time, not blocking the client, having low impact on the client). What they showed with their research is that a specialized database can outperform a general purpose database.

Johanson et al. [7] stated that one of the key factors to be thriving in the very competitive consumer vehicle market is knowledge-driven product development. That implies understanding customer needs and vehicle performance through, among other things, large sets of data collected from connected vehicles. The volume of collected data can be very high [7], entities can be heterogeneous and under varying network conditions [81] so it is important to design systems that are highly scalable, flexible and efficient. Johanson et al. described the problem

as a big data challenge in automotive context and in the paper they explore opportunities for knowledge driven product development, presenting a technological framework for capturing and online analysis of data from connected vehicles including integrated telematics (i.e. 2G, 3G or 4G, GPS) and analytics (i.e. CAN bus signals, MDF files, Apache Spark's MLlib) services [7]. Luckow et al. [22] also described the problem in a similar manner and introduced Hadoop as a scalable platform suitable for many use cases and applications in the automotive industry as it enables handling the different V's of big data, especially requirements related to data volume and velocity. In their paper, Hadoop is defined as an ecosystem of various tools such as e.g. Flume, Kafka for data ingestion, Spark, Flink for data processing, and MLlib, Mahout for advanced data analytics. Woźniak et al. [82] describe how they have identified and integrated data sources to build a big data service for the automotive industry. They emphasise the importance of domain knowledge in building big data services and the usefulness of creating future databases with big data in mind.

2.2.4Automotive Data Analytics

To be able to access and analyse any kind of data, middleware solutions are usually deployed, most commonly in the form of an API. Due to various possible variables and constraints a data set can contain, providing a general application programming interface solution for data processing and analytics is impracticable. Each API must be developed with a specific use case in mind, but it can follow a previously defined design methodology. Pinheiro et al. [83] developed one such methodology to enable the development of efficient APIs to handle optimisation problems data. The methodology they proposed involves the design of a data parser and a definition of efficient data structures to hold the data in memory. They point out that the proposed API development methodology can help researchers to design and implement APIs tailored for specific problems. Wagner et al. [84] present the design of a higher level API for use in sensor networks that is powerful, convenient to use, and compact. The API they described is designed in a way that enables algorithm developers to quickly and easily implement their designs without having to handle low-level networking details themselves. Alongside development methodology definition and data abstraction, API design and usability is also of interest to researchers as stated by Ofoeada et al. [85]. They performed a review of research pertaining to APIs and concluded that the value of an API is in its design and usability and that a good API should be easy to learn and use which in turn translates into the productivity of developers.

With the increase of the amount of data present in today's automotive industry, researchers in this field also turned to the development of APIs for efficient data acquisition, processing and analysis. As smart urban transportation management relies heavily on the information collected from multiple heterogeneous data sources, as well as on the ability to extract actionable insights from them, Fiore et al. [86] have proposed an integrated platform for big data analytics. The

solution allows an easy, rapid and transparent development of smart cities applications using a flexible and dynamic cloud infrastructure and a Python-based API. Zhu et al. [87] introduced a navigation application programming interface for estimating fuel advantages of alternative routes based on large-scale, real-world travel data for conventional vehicles and hybrid-electric vehicles. While standard navigation APIs, such as Google Directions API, integrate traffic conditions and provide feasible alternative routes for origin–destination pairs, the API described in this paper calculates fuel-consumption models and identifies alternate routes that save fuel. Boaz and Tarico [88] have provided an overview of the Advanced Transportation Controller (ATC) API standards used in the development of intelligent transportation systems. This API aims to bridge the technology gap created by underpowered and outdated traffic controllers through providing powerful general purpose on-street computing platform that is designed to grow with technological advancements. Mobility as a Service (MaaS) is another transportation science use case where APIs can offer benefits in data processing. Garcia et al. [89] have described the Transport Operator to MaaS Provider (TOMP) API, a standardized and technical interface between MaaS providers and transport operators. The API allows all participating companies to communicate about planning, booking, execution, support, general information and payments of multimodal, end user specific trips.

Once exposed through an API, automotive data can provide various insights into the condition and performance of the vehicle and behaviour and style of the driver. Fugiglando et al. [90] proposed a new method for the analysis and classification of driver behaviour using a selected subset of CAN bus signals. They collected the data in an uncontrolled experiment and introduced an unsupervised learning technique that clusters drivers in different groups according to their driving style. Uvarov and Ponomarev [91] performed a similar research using automotive data collected via OBD-II vehicle port. They calculated additional features describing acceleration and jerk vehicle speed and added those features to the main data set. Driver identification was carried out using several machine learning methods such as KNN, decision tree, random forest, gradient boosting and support vector machine.

Puchalski and Komorska [92] analysed driving style and performed driver classification using OBD-II data of a hybrid electric vehicle. To characterise drivers, they correlated acceleration and speed of the vehicle to fuel consumption. They used cluster analysis of driving styles based on the car acceleration to show the impact of driving style on average fuel consumption. Three driving styles were demonstrated as a part of their research: eco-driving, eco-neutral and not eco-driving. According to their model, the driver's individual characteristics are best described by the mean square value of car acceleration with mean positive acceleration values. Similarly, Martinelli et al. [93] performed cluster analysis to identify driver aggressiveness. They computed two different aggressiveness indexes depending on the road on which the driver was driving (urban or highway roads). The road identification task does not take any additional

input other than the provided automotive data, showing that automotive data can be used not only to describe the driving style and behaviour, but to classify the roads being driven on.

Choi et al. [94] employed previously obtained CAN bus data to model driving behaviour in three distinct classification tasks: action detection, distraction detection and driver identification. Their detection success varied from 30-70%, depending on the number of unique conditions based on CAN bus signals. On the other hand, Castignani et al. [11] analysed how smartphone sensors and OBD-II data can be used to identify driving manoeuvres. They proposed SenseFleet, a driver profile platform that is able to detect risky driving events independently from the mobile device and vehicle. The developed platform is able to accurately detect risky driving events and provide a representative score for each individual driver. The authors state that this kind of profiling system can be useful in use cases such as insurance premium adjustment, fuel consumption optimisation or CO2 emission reduction. Kwak et al. [95] explored the possibility of car theft detection by driver profiling using CAN bus data. They enriched the data set with statistical features such as mean, median and standard deviation and used those features along with the collected data to detect the driver characteristics in real time and inform the vehicle owners of the theft events as soon as possible.

Following these claims, Alessandrini et al. [96] stated that driving style has an influence on vehicle consumption and emissions, and used their research to confirm this by calculating vehicle consumption using OBD-II data. Their method of calculation has shown a difference between measurements and models never greater than 4%, indicating that the methodology is accurate enough to calculate the power and consumption of vehicles during their real use. Another example of automotive data analysis with a goal of improving transportation sustainability is the research by Hilpert et al. [97], who proposed a system for calculating the product carbon footprint (PCF) of individual products' transportation processes. The system combined OBD-II data with location information and gathered it in real-time via a smartphone, creating a high-quality data set for PCF calculation. The developed PCF calculation module was integrated in an industry-used Environmental Management Information System (EMIS).

2.3Reflection and Proposed Solution

The review of related research has clearly shown that there is a rising interest in topics dealing with automotive data manipulation. This is perhaps not surprising, especially when taking into account the findings presented in Section 2.1. The historical developments in the automotive industry indicate that future vehicles will be electric, connected and autonomous, and software will serve as the driving force behind all major advancements. The overview of the ICT architecture present in modern vehicles has once again shown the importance of software in enabling advanced functionalities such as intercomponent communication, connectivity and autonomous

driving. The growing amount of software in modern vehicles is in obvious correlation with the rising amount of automotive data.

Automotive data collection, contextual enrichment, storage and analytics are explored in various research papers and several conclusions can be made from the literature review performed in Section 2.2. In the surveyed papers dealing with automotive data collection, no data collection solution was found that supports both higher- and lower-level automotive communication protocols. As for the papers dealing with contextual enrichment, researchers tend to use contextual enrichment on mobility data sets, but only a small number of them use enrichment on automotive data, and those that do focus mainly on enrichment with location data only. Researchers examining automotive data storage agree on the importance of choosing the most appropriate methods, taking into account parameters such as efficiency, scalability, robustness, performance, reliability, cost, security and many others, depending on the specific application. Additionally, they emphasize the importance of domain knowledge when building big data services along with the usefulness of creating future databases with big data in mind. In order to access and analyse the collected automotive data a middleware solution in the form of an API is required. The performed literature review has shown that when implementing an API for data analytics, one must pay attention to its design, usability and the methodology of the development. The review has also demonstrated that APIs are useful when managing big data in the automotive domain and can be utilized in various use cases where advanced analytics is effective, such as transportation sustainability, intelligent transportation systems and mobility as a service. Finally, to perform impactful research in the area of automotive data analytics, a high-quality source of contextually enriched automotive data is required. The conducted survey has shown that an open source data set of contextually enriched automotive data is currently not available.

To summarize, the framework presented in this thesis aims to provide solutions to the main problems identified in the state-of-the-art review:

- *higher- and lower-level automotive communication protocol support* - the framework allows automotive data collection over OBD-II port and CAN bus;
- *contextual enrichment of automotive data from multiple heterogeneous data sources* - the framework enriches the automotive data with data collected from Internet-based sources (weather, traffic) and built-in smartphone hardware (location, signal, motion sensors);
- *highly available, scalable and fault-tolerant storage of automotive data* - the framework includes a big data storage solution tailored for the storage of automotive data;
- *specification of an application programming interface for accessing and analysing automotive data* - the framework offers an application programming interface with modules for acquiring and performing statistical and analytical calculations of automotive data;
- *an open source contextually enriched automotive data set* - the framework is used in a

data collection experiment to create a contextually enriched automotive data set.

Taking all this into consideration, the research goal of the thesis is to design, develop and implement a framework for collection, contextual enrichment and advanced analytics of automotive data which solves the aforementioned research challenges. The challenges related to framework's functionalities regarding automotive communication protocol support, automotive data contextual enrichment, storage and analytics were solved during the implementation of the framework (see Chapter 3). Once operational, the framework was used to collect a contextually enriched automotive data set in a data collection experiment (see Chapter 4), and the data set is used in various data analysis use cases (see Chapter 5) to prove the usefulness of such data in the field of automotive data analytics.

Chapter 3

Collection and Contextual Enrichment of Automotive Data

This chapter explains in detail the framework for collection, contextual enrichment and advanced analytics of automotive data. First, to better understand the development process used while establishing the framework, the used methodology will be described (Section 3.1). After that, each module of the framework, which can be seen in its entirety in Figure 3.1, will be examined more closely. The framework consists of five interconnected components. The first component are the *data sources* from which the data is acquired (Section 3.2). This includes automotive data sources - OBD-II and CAN bus, mobile device sources - location, signal and sensor information, and Internet data sources - weather and traffic information. Seeing as CAN bus data is difficult to acquire, the next component included as a part of the framework is the *CAN bus simulator* (Section 3.3). The modelling, implementation and validation process used in the development of the simulator will also be described. The *collection and contextual enrichment modules* are the third major component (Section 3.4). They are implemented in the form of a smartphone application and ingest OBD or CAN bus data, enriching it with Internet-based and mobile sensor information. The *distributed data storage platform* used to store automotive data is the fourth framework component (Section 3.5). It receives the data from the collection modules and safely stores it for further use. The final component is the *application programming interface* used to access and analyse the stored automotive data (Section 3.6). It consists of data acquisition, statistics and advanced analytics modules with which data consumers can interact.

3.1Methodology

Since the research described in this thesis is data-driven, the methodology uses a process model that describes a common approach used by data science experts. The methodology used in this

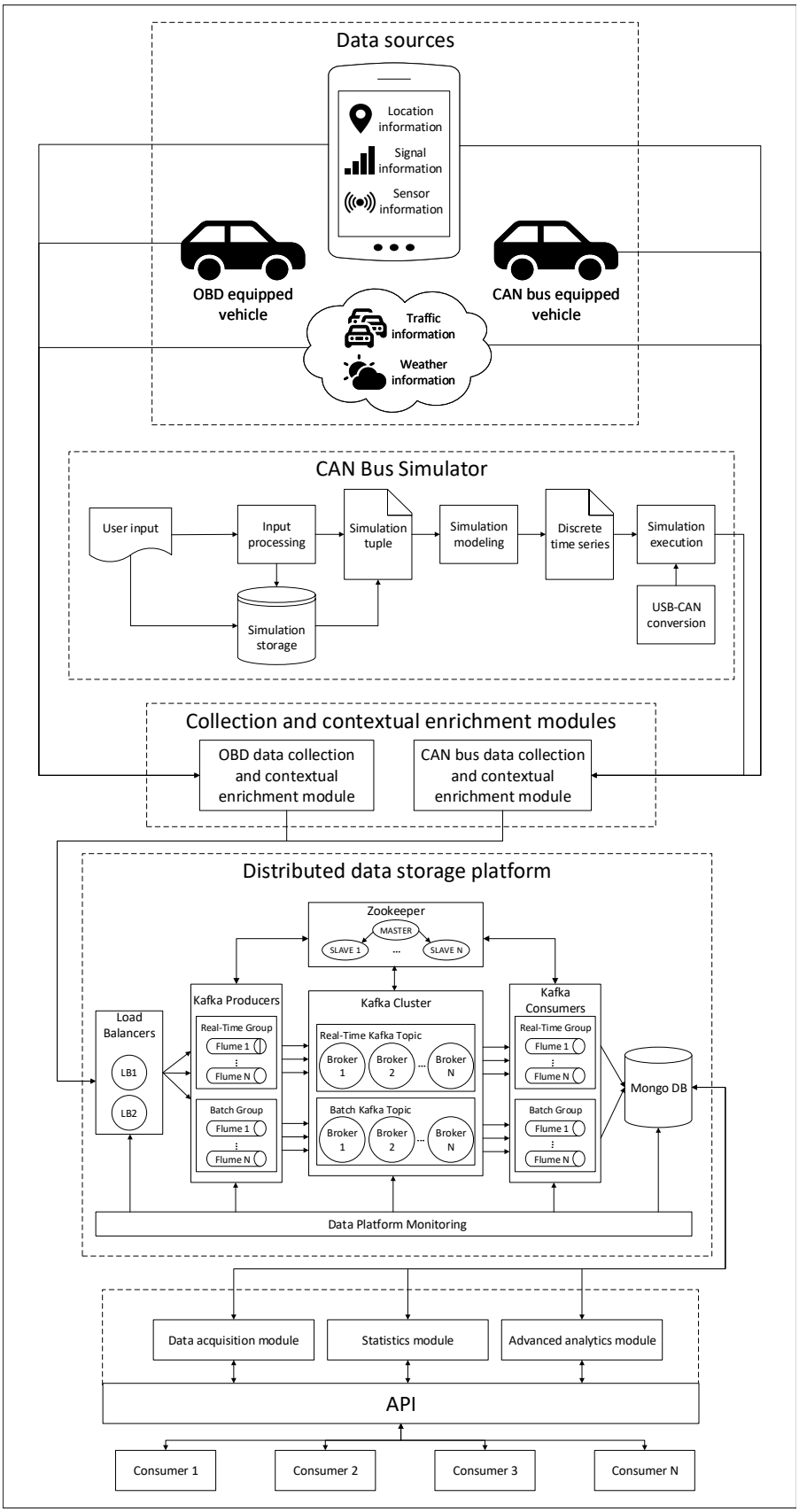

Figure 3.1: Framework for Collection, Contextual Enrichment and Advanced Analytics of Automotive Data

research is mainly based on the Team Data Science Process (TDSP) [98], an agile and iterative data science methodology for delivering predictive analytics solutions and intelligent applications efficiently. However, as TDSP is mainly business-oriented, elements describing business understanding and deployment have been replaced by domain understanding and evaluation elements taken from the Cross-industry Standard Process for Data Mining (CRISP-DM) [99], another widely used data science methodology, to adjust it to the use case of this research. The methodology of this research can be seen in Figure 3.2 and consists of the following elements: (i) domain understanding, (ii) data collection and understanding, (iii) modelling, (iv) evaluation.

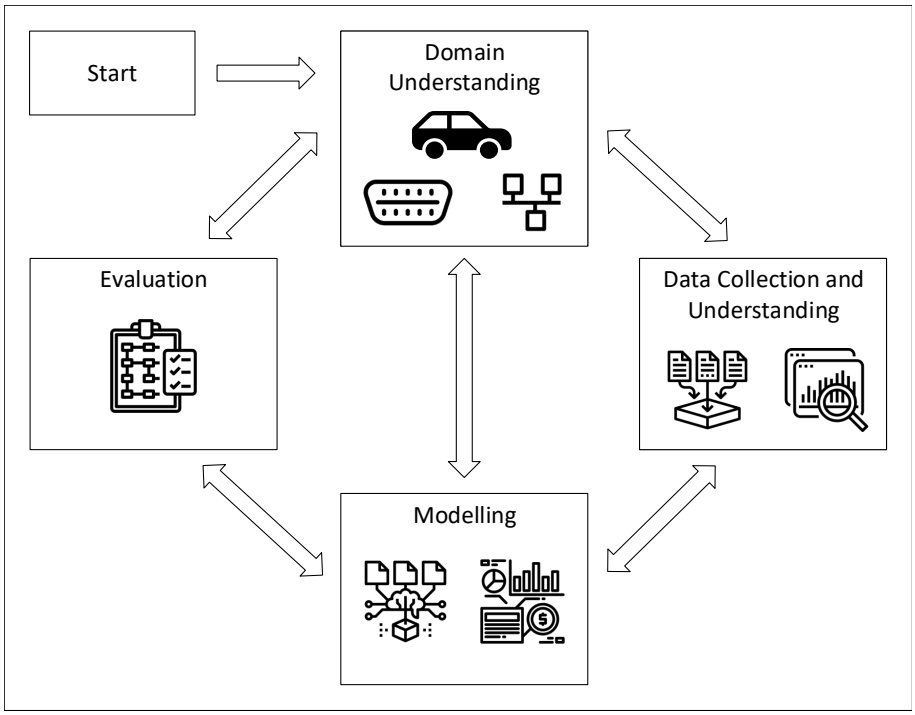

Figure 3.2: Research methodology

Domain understanding is the first step when conducting any kind of research. The domain of this research is primarily focused on automotive communication technologies, software and data science. A review of automotive software, connected vehicles and ICT technologies was done as a part of preliminary research, and can be seen in Section 2. In the same section, an overview of automotive data manipulation in literature is also provided.

A large part of the research presented in this thesis is focused on *data collection and understanding*, as the presented framework aims to enable the collection of automotive data that can be performed on the widest possible range of diverse vehicles. The automotive communication protocols needed to be researched to understand the data that can be collected from each of them and to select the most appropriate protocols for data collection. The collected data was to be enriched with data from several heterogeneous data sources. These sources also had to be explored in order to understand usefulness of information contained in their data sets. Vehicles can generate a considerable amount of data, and when this data is combined with data from

external sources, the challenge of big data storage arises. To solve this challenge, a distributed data platform for the storage of automotive data has been developed [100]. At the end of the data acquisition step, the goal was to have a contextually enriched, clean, high-quality automotive data set ready for analysis and modelling.

The data analysis carried out in the *modelling* phase is focused on quantifying the driving performance by observing the data from multiple perspectives: driver eco-efficiency, trip eco-efficiency and driving route comparison. Appropriate data-driven analysis techniques need to be identified and properly used to develop models which will be used to evaluate the driving performance. The ultimate goal of this step is to determine the optimal data features used for classification to provide the most accurate result.

Finally, *evaluation* has to be performed on all parts of the developed framework. The entire framework was validated during its usage in a real-world data collection experiment. As the framework was developed with this purpose in mind, if the data collection experiment resulted in the successful creation of a contextually enriched automotive data set, a conclusion can be made that the framework is functioning properly. Additionally, all of the framework's modules were validated individually during their development, and notes were taken on all of the errors that arose due to faults in the hardware and software and those errors were fixed.

3.2Data Sources

The framework intends to solve the challenge of real-time automotive data collection from the widest possible range of diverse vehicles. To be able to collect data in real-time, Internet access needs to be provided to vehicles. Connected vehicles would be ideal candidates for data collection, if not for the fact that connected vehicle hardware and software are a closed system meant to be used by vehicle manufacturers only. Also, according to [18], only 31% of households in the US have had at least one connected car in 2018, while that number was around 20% for European countries. This is why targeting only connected cars does not satisfy the requirement of automotive data collection from the widest possible range of vehicles. As previously stated, smartphone-based connected vehicles - vehicles that are connected to the Internet via a smartphone device, offer a scalable, upgradeable and cost-effective solution to this problem.

Once smartphones were determined as a data collection tool, two automotive communication technologies have been identified to further maximize the spectrum of vehicles from which data can be collected: a higher-level diagnostics protocol – OBD-II, and a lower-level bus system – CAN bus. OBD-II diagnostic interface is mandatory for all passenger vehicles manufactured in the EU since 2003 and since 1996 in the US, making it ideal as a collection source for such vehicles. However, electric vehicles and specialized vehicles (e.g. agricultural,

mining, municipal vehicles) do not have to conform to such a standard. Therefore, the most common underlying bus protocol – CAN bus was chosen as an alternative for automotive data collection.

3.2.1OBD-II

OBD-II is a standardized automotive interface providing vehicle diagnostic information. It also allows capturing current engine and system conditions in real-time [101]. The OBD-II port is usually located in the close vicinity of the vehicle's steering wheel and is easily accessible. All of this makes it a suitable candidate for automotive data collection. To relay the information from OBD-II to the collection module, an ELM327 OBD-II-to-Bluetooth programmed microcontroller was used (Figure 3.3).

Figure 3.3: ELM-327 OBD-II-to-Bluetooth device [102]

To request data from a vehicle via Bluetooth, the collection module must send a parameter ID (PID) code to the ELM327 Bluetooth device. There are more than 200 available PID codes for requesting various vehicle data (e.g. vehicle speed, engine speed, throttle position). However, not all vehicles support all PIDs, and there can be manufacturer-defined custom PIDs that are not defined in the OBD-II standard. Supported and available PIDs for a given vehicle can be requested via a separate PID code. OBD-II scanning rates differ from vehicle to vehicle, as some vehicles respond to requests slower, while some respond faster. Requests can only be sent sequentially, OBD-II does not allow parallel requests. After a request is sent, ELM327 waits for a response for up to 200 ms [103]. If a response is not received in that period of time, the message of "NODATA" will be sent back. It is important to avoid requests for unavailable data to avoid these timeouts and maximize the number of responses which can be received in a second. For all these reasons, it is impossible to request all available data from each vehicle and keep the scanning frequency at under 1 second, which is seen as the upper limit for the time period between two data points. To mitigate this, a subset of all defined PIDs that should be retrieved from the vehicle needed to be chosen. Not all PIDs selected for collection are available on all vehicles, so the intersection of PIDs selected for collection and PIDs available in a given vehicle

forms the final collected data set for each vehicle. In mathematical terms, if S_{all} is the set of all existing PIDs, $S_{selected}$ the set of PIDs selected for collection, $S_{available}$ the set of available PIDs for a vehicle, and $S_{collected}$ the set of collected PIDs for a vehicle, then:

$$S_{selected} \subset S_{all}, \quad S_{available} \subset S_{all} \tag{3.1}$$

$$S_{selected} \cap S_{available} = S_{collected} \tag{3.2}$$

The final set of 13 PIDs selected for collection ordered by their PID number can be seen in Table 3.1. *Calculated engine load* indicates the percent of peak available torque*. *Engine speed* provides information on how many full turns the engine does every minute. *Vehicle speed* is the actual speed of the vehicle measured in km/h. *Throttle position* represents the openness of the throttle valve expressed as a percentage. *Relative throttle position* is the throttle position compared to the learned closed position described by *absolute throttle position*. For example, if absolute throttle position is at 9% with the throttle fully closed, then the relative throttle position will read 0% in the same position. The same applies to the *relative accelerator pedal* position when compared to *accelerator pedal position* values. These values are implemented this way in the OBD-II standard to serve as redundancy to one another. The sensor values for absolute throttle position and accelerator pedal position finish with a letter as they are returned by multiple separate sensors which the ECU uses to validate the integrity of the signals. To ensure the accurate value is acquired in the case of a failure of one of these sensors, all of them were selected for collection. Finally, *engine fuel rate* indicates the vehicle's current fuel consumption.

To further improve the scanning frequency, some PIDs describing throttle and accelerator pedal position are collected only if the throttle position command is unavailable or if throttle position value is constant (and therefore invalid) throughout the duration of the measurement. Therefore, a primary ($S_{primary}$) and an alternative set (S_{alt}) of PIDs selected for collection ($S_{selected}$) is defined. Both the primary and the alternative set contain a set of shared values (S_{base}); however, they differ in the way throttle position values are collected. These sets are described in Equations 3.3 - 3.6.

$$S_{selected} = S_{primary} \,|\, S_{alt} \tag{3.3}$$

$$S_{base} = (P_{eng\_load}, P_{rpm}, P_{speed}, P_{fuel\_level}, P_{fuel\_rate}) \tag{3.4}$$

$$S_{primary} = S_{base} \cup (P_{throttle}) \tag{3.5}$$

$$S_{alt} = S_{base} \cup (P_{rel\_thr}, P_{abs\_thr\_B}, P_{abs\_thr\_C}, P_{acc\_D}, P_{acc\_E}, P_{acc\_F}, P_{rel\_acc}) \tag{3.6}$$

*engine's rotational force, the amount of work an engine can exert

Table 3.1: OBD-II PIDs selected for collection

PID (hex)	PID (dec)	Description	Units	Abbreviation
04	4	Calculated engine load	%	P_{eng_load}
0C	12	Engine speed	rpm	P_{rpm}
0D	13	Vehicle speed	km/h	P_{speed}
11	17	Throttle position	%	$P_{throttle}$
2F	47	Fuel tank level input	%	P_{fuel_level}
45	69	Relative throttle position	%	P_{rel_thr}
47	71	Absolute throttle position B	%	$P_{abs\_thr\_B}$
48	72	Absolute throttle position C	%	$P_{abs\_thr\_C}$
49	73	Accelerator pedal position D	%	P_{acc_D}
4A	74	Accelerator pedal position E	%	P_{acc_E}
4B	75	Accelerator pedal position F	%	P_{acc_F}
5A	90	Relative accelerator pedal position	%	P_{rel_acc}
5E	94	Engine fuel rate	l/h	P_{fuel_rate}

The primary set is collected if throttle position value is available and valid, while the alternative set is collected in all other instances. With these improvements, the final data scanning frequency of the OBD-II data collection module was at around 1 second for all vehicles which participated in the data collection experiment.

3.2.2CAN Bus

There are some major differences between CAN bus and OBD-II data collection. First of all, CAN message format is different from OBD-II - each CAN message has an identifier (11 bits or 29 bits for extended format), and can carry up to eight data bytes. Secondly, CAN is a bus system, so if a device is connected to the bus, it captures all data that is propagated on the bus, in contrast to OBD-II where data needs to be requested and is received in the response. This is why the module for CAN bus data collection filters through received messages to only store the wanted data. CAN message identifiers differ from vehicle to vehicle, so to be able to filter the data, knowledge of the vehicle's CAN message specification is needed. The final difference is in the way CAN bus data is collected. OBD-II data collection module receives the data via Bluetooth. Other solutions based on Wi-Fi or USB connections also exists, however OBD-to-Bluetooth hardware is usually small in size and simple to use, which is important as it causes minimal distractions for the driver. On the other hand, the CAN bus collection module receives

the data through a Wi-Fi connection with a CAN-to-Wireless gateway (Figure 3.4) as there is no major difference in size or simplicity of use when compared to it's Bluetooth counterparts and it has also proven to be more robust and offers more features than the tested CAN-to-Bluetooth gateways.

Figure 3.4: CAN-to-Wireless gateway by PEAK System [104] used for CAN data collection

The final data set which can be collected by the CAN bus data collection module is the same as the OBD-II data set seen in Table 3.1 to maintain database consistency for both data sources. Although data collection frequency of the CAN bus data collection module can be higher than in the OBD-II counterpart, it is kept at a value of 1 second to mirror the frequency of the OBD-II data collection module.

3.2.3Mobile Device

Once vehicle information is successfully collected from one of the two previously mentioned sources, automotive data can be enriched with context from multiple heterogeneous sources. Automotive data is contextually enriched with data acquired from the mobile device performing the collection and data acquired from the Internet. More information on this data will be given in Chapter 4. Three main types of information used for contextual enrichment are collected from the mobile device: location, signal and sensor information.

Location information, including latitude, longitude and altitude, is collected to represent not only the location of the mobile device, but the location of the vehicle. This information is refreshed each time the device location changes and is paramount for fetching Internet-based information which will be examined in the next section.

The next data set collected from the mobile device describes *signal information*. This includes information about the current phone signal strength and information about the network and cell the device is currently connected. For example, some fields included in this data set are the name of the network operator, network type, received signal strength indication (RSSI),

bandwidth and the list of neighbouring cells. Similarly to location data, signal information is refreshed each time the device signal strength changes. The main reason behind the collection of signal information alongside automotive data is the potential usage of this information in future research about analysis of existing network infrastructure and planning of next-generation vehicular networks. This data can help in making well-informed decisions about the placement of new infrastructure nodes, minimizing the cost and maximizing the system's efficiency, which in turn facilitates sustainability.

Finally, every automotive data point is enriched with mobile device *sensor information*. Information from up to seven different sensor types is collected, depending on their availability on the mobile device used for collection: accelerometer, gyroscope, uncalibrated gyroscope, magnetic field, uncalibrated magnetic field, rotation vector and game rotation vector. The current status of these sensors is sent alongside every automotive data point. This can be used to position the mobile device in three-dimensional space and detect sudden movements caused by driving events, as seen in research by [11].

3.2.4Internet

Using the location information collected from the mobile device, two new data sets are collected from the Internet: traffic information and weather information.

Traffic information is collected using the Traffic API provided by TomTom, a Dutch multinational developer and creator of location technology and consumer electronics [105]. The API offers data about current average traffic speed, free-flow speed (traffic speed expected under ideal conditions) and road closures at the provided location. TomTom divides its map into tiles defined by the zoom level defined in the API call. The zoom level which provides the tile side of 76 meters was used. When taking into account the average speed of the vehicle driving in urban areas, which is around 35km/h (9.7 m/s) [106], it is enough to poll the API every eight seconds as this offers the optimal balance between fetching redundant information to often, and not fetching it frequently enough, resulting in enriching automotive data with outdated traffic information. This period is configurable in the collection and contextual enrichment application, and is set to eight seconds by default.

To acquire *weather information*, the API offered by OpenWeather, an online service that provides global weather data [107] was used. Like traffic information, weather information is also location dependent. The information collected from this endpoint includes current temperature, atmospheric pressure, humidity, wind speed and rain and snow volume. Weather information is refreshed every 10 minutes, as this is stated as the update frequency of the current weather state by the OpenWeather API specification. Like in the case of traffic information, this period is also configurable in the collection and contextual enrichment application, and is set to 10 minutes by default.

3.3CAN Bus Simulator

The biggest hurdle in CAN bus data collection is the knowledge of the CAN message specification of the vehicle from which data is to be collected. Vehicle manufacturers normally do not publicly disclose these specifications, so they need to be either reverse-engineered from the captured data (as shown in [108]) or explicitly requested from the vehicle manufacturer. RASCO, a company manufacturing municipal equipment for road maintenance [109] has provided one such specification for one of their specialized vehicles which was used to create a CAN bus simulator [110]. This simulator was used while testing the CAN bus data collection module, replacing the real vehicle.

The simulator needs to satisfy several requirements which were identified in collaboration with industry specialists from RASCO [109], a company manufacturing municipal equipment for road maintenance. It must be able to simulate a single component as well as multiple components simultaneously, acting as a CAN bus system with multiple components connected to it. The simulator must use extended data frames with 29 identifier bits and 8 bytes of data, sent at a bit rate of 250 kBit/s as per vehicle specification. Messages can be repeated at a set interval, with the number of repeats and time between repeats being user defined. As the data bytes in repeated messages can change over time, the simulator must allow the user to specify how exactly they change (e.g. incrementing or decrementing linearly or exponentially). Generated messages must be sent to a hardware output, simulating an actual CAN bus on which other devices can connect to read the data on the bus. Finally, the simulator must provide persistent storage for the created simulations so they can be reused once they are created.

3.3.1Architecture

To show how the simulator converts various inputs into a CAN message data stream, its architecture was established through a formal model. The simulator's architecture based on the formal model can be seen in Figure 3.5. As displayed, the simulator has a single input and a single output type. The input of the simulator is entered by its users via a graphical user interface. The graphical user interface enables the user to create simulations, store them into different categories and combine existing simulations into a complex scenario. To create a new simulation, a user must provide a name by which a simulation will be identified, 29 identifier bits and 8 bytes of data that together form a CAN data frame, and optional information on how this data frame will act over time. Over time, a data frame can be sent repeatedly in arbitrary intervals defined in milliseconds, with changed or unchanged data bytes. A user can define which data bytes will change and in what way, e.g. whether some byte values will increase or decrease linearly or exponentially over time.

The *user input* data is processed in the *input processing* module and formed into a *simulation*

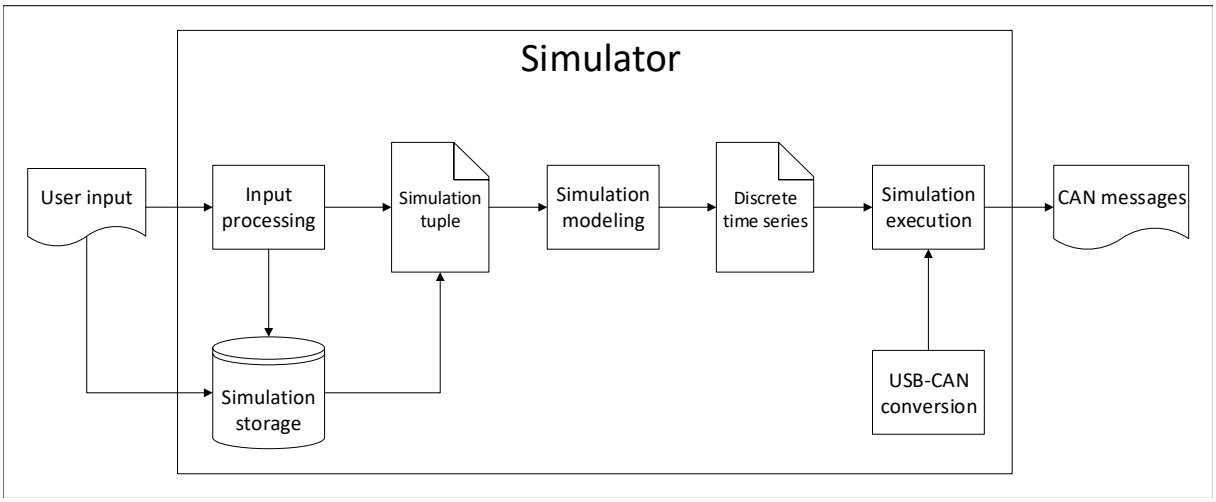

Figure 3.5: Proposed CAN bus simulator architecture

tuple. A simulation tuple, T_{sim}, describes the simulation of exactly one vehicle component, such as engine control unit or the accelerator pedal control unit, which generates CAN data that changes over time.

$$T_{sim} = (name, ID, data, t_s, t_d, f_{data}(t \in [t_s, t_s + t_d])) \tag{3.7}$$

The simulation tuple contains the following parameters (Equation 3.7):

- *name* - a custom name which identifies the simulation for subsequent reuse;
- *ID* - 29 identifier bits in a hexadecimal format;
- $data$ - 8 bytes of data in a hexadecimal format;
- t_s - the relative time in milliseconds from the beginning of a scenario when the simulation will start executing;
- t_d - the duration of the simulation in milliseconds;
- $f_{data}(t \in [t_s, t_s + t_d])$ - transformation function which indicates the change in data bytes over time (constant, linear or exponential increase or decrease).

The input processing module is also responsible for storing simulations in the database. The storage is used to allow a user to execute a previously created simulation or to incorporate a previously created simulation in a new simulation scenario.

A simulation scenario is a set of simulations defined by simulation tuples that are executed simultaneously. Users can create a simulation scenario using already existing simulations saved in the simulation storage and/or by using the simulations created by themselves. When a simulation scenario is determined, the simulation tuples of all simulations which need to be executed as a part of the scenario are passed on to the *simulation modelling* module. This module's function is to analyze the simulation tuples and create a discrete time series using the transformation

function provided in simulation tuples (Equation 3.8).

$$f_{sm}([T_{sim}]) = DTS = [T_{dts}] \tag{3.8}$$

$$T_{dts} = (ID, data, t_m) \tag{3.9}$$

The *discrete time series* (DTS) is a list of tuples with the following parameters (Equation 3.9):

- ID - 29 identifier bits;
- $data$ - 8 bytes of data;
- t_m - the relative time from the start of the simulation when the message needs to be sent.

The discrete time series is used by the *simulation execution* module to create a CAN message stream. Seeing as the simulator is running on a PC, additional hardware is needed to simulate the CAN bus. For this purpose, the simulation execution module uses a *USB-CAN converter*, to which it feeds raw CAN messages in times specified in the model, outputting a stream of *CAN messages*.

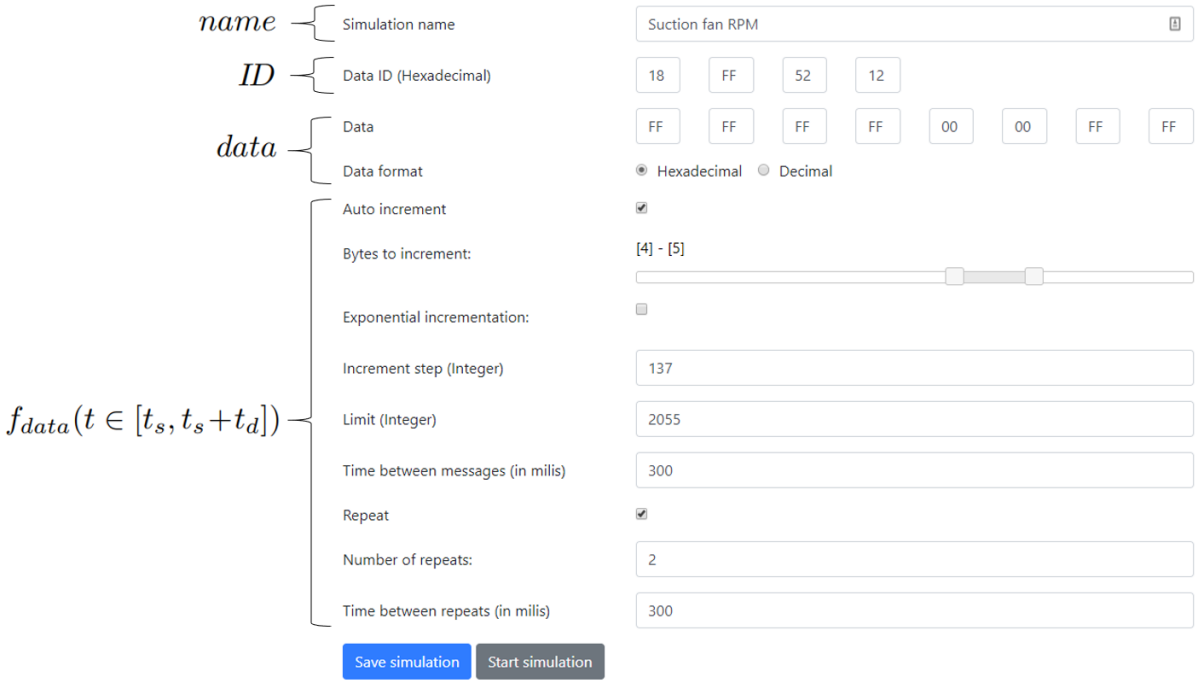

Figure 3.6: Simulator input collection form mapped on the simulation tuple

3.3.2Implementation

The simulator was implemented in Java using the model-view-controller (MVC) software design pattern and Spring application framework and it complies to the previously defined formal model and architecture (Figure 3.5). To collect user input, a graphical user interface designed using Javascript and the Bootstrap front-end framework is used. A user can create a new simula-

tion using the input form which can be seen in Figure 3.6. The input form contains all the fields needed to model the simulation tuple (Equation 3.7); including simulation name, identifier, data and optional fields describing the incrementation of chosen data bytes and number of message repeats over time.

Input processing is completed in the applications controllers. Controllers are also used to save simulations into the simulation storage. For the purpose of the simulation storage, a MySQL storage is used. Along with creating, storing and running single simulations, a user can create complex scenarios using multiple already created simulations. To create a scenario, a timeline is used on which users can add existing simulations and specify their start times relative to the start time of the scenario. An example of a complex scenario, which is explained in the next subsection in more detail, can be seen in Figure 3.7).

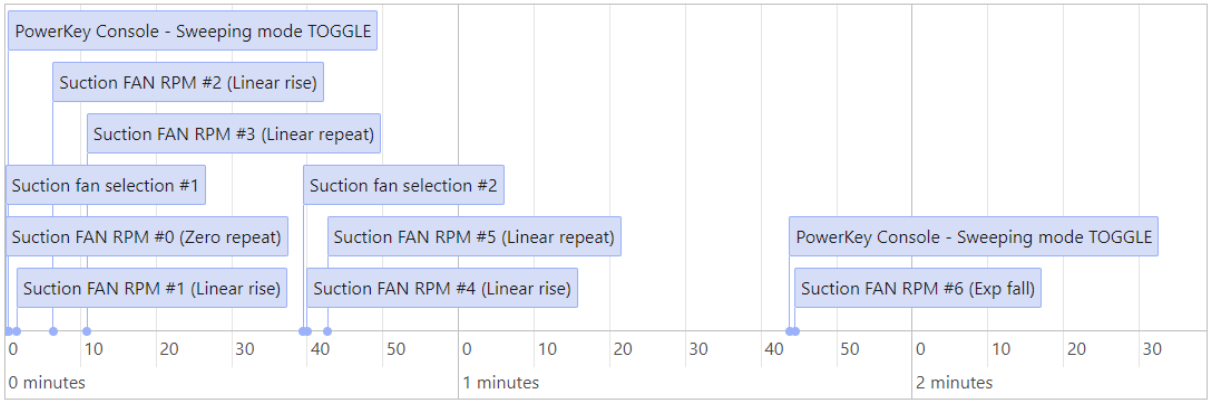

Figure 3.7: An example of a simulation scenario timeline

Simulations scheduled for execution are converted into a list of objects representing the discrete time series according to the formal model (Equation 3.8). The USB to CAN conversion is done using PEAK-system's PCAN-USB, a CAN hardware interface for USB. PCAN-USB device drivers must be installed on the computer for it to be able to run the simulator. PCAN-Basic, a programming interface to the PCAN system is used to communicate with the hardware and send all created CAN messages to the CAN bus interface.

3.3.3Validation

The simulator was validated using CAN traces captured on the Lynx compact sweeper vehicle [111] developed by the RASCO company. The trace files were analyzed and three mutually dependent parameters were selected for simulation.

The first parameter is the console button press indicating the cleaner entering or exiting the sweeping mode. The button acts as a toggle, so the first press of the button means that the cleaner is entering sweeping mode, while the second press means that the cleaner is exiting the sweeping mode. The CAN ID of the messages generated by the console is `0x18EFFF27`, while

the significant data is carried on the third byte - signifying the number of the key on the console which is pressed (`0x09` for the sweeping mode toggle), and the fourth byte - signifying the contact state. Contact state is one of two values; 1 (`0x01`) if the button is pressed, or 0 (`0x00`) if the button is not pressed. A single press of the console button generates two messages, one message with contact state value of 1 as the button is first pressed, and the second message with contact state value of 0 as the button is released. The second message is sent 300 ms after the first.

The second parameter is the suction fan selection which indicates the mode of operation of the cleaner's suction fan. The CAN ID of these messages is `0x18FF0712` and the significant data is carried on the sixth byte describing the suction fan selection, ranging from 0 (`0x00`) to 100 (`0x64`). The messages carrying this information are sent repeatedly every 130 ms.

The third parameter is the suction fan revolutions per minute (RPM). Messages carrying this variable have the ID `0x18FF5212` and the relevant data is on the fourth and fifth byte ranging from 0 (`0x0000`) to 4,000 (`0x0FA0`). The messages carrying this information are sent repeatedly every 300 ms.

The recorded values of the three parameters through the time span of a little more than two minutes can be seen in Figure 3.8. The figure clearly shows how the three parameters influence one another through time. Once the console button is pressed the first time (indicating the sweeping mode start), suction fan RPM begins to rise. It rises sharply and linearly until it reaches 2,055 and then continues to rise linearly with a smaller incline until it reaches 2,552. The RPM value hovers around the average of 2,536 for about 35 seconds when the suction fan selection changes from 60 to 80. Then the RPM linearly rises to 3,000 after which it stays at the average value of 2,943 for about a minute. At that time, the console button is pressed again, stopping the sweeping mode. The RPM then falls exponentially to 1,445 after which the recording stops.

This entire scenario was recreated in the simulator in order to validate it. As the console button press and suction fan selection parameters can only have one of two possible values during the course of the scenario, the only thing needed to simulate them are the moments at which their values change and the interval between messages. On the other hand, considering that the suction fan RPM can amount to any value between 0 and 4,000, the simulation needed to be broken down into multiple parts: linear rise at different angles, repetition of a constant value and the exponential fall. After the angles of the linear rises and the rate of the exponential fall were identified, the simulation scenario was ready to be executed (Figure 3.7). The output of the simulation can be seen in Figure 3.9.

When visually inspecting the two graphs, it can be seen that the most apparent difference is that the simulator cannot generate the oscillations in the continuous repeat of the RPM messages, which can happen in reality. This is circumvented by the simulator sending an average

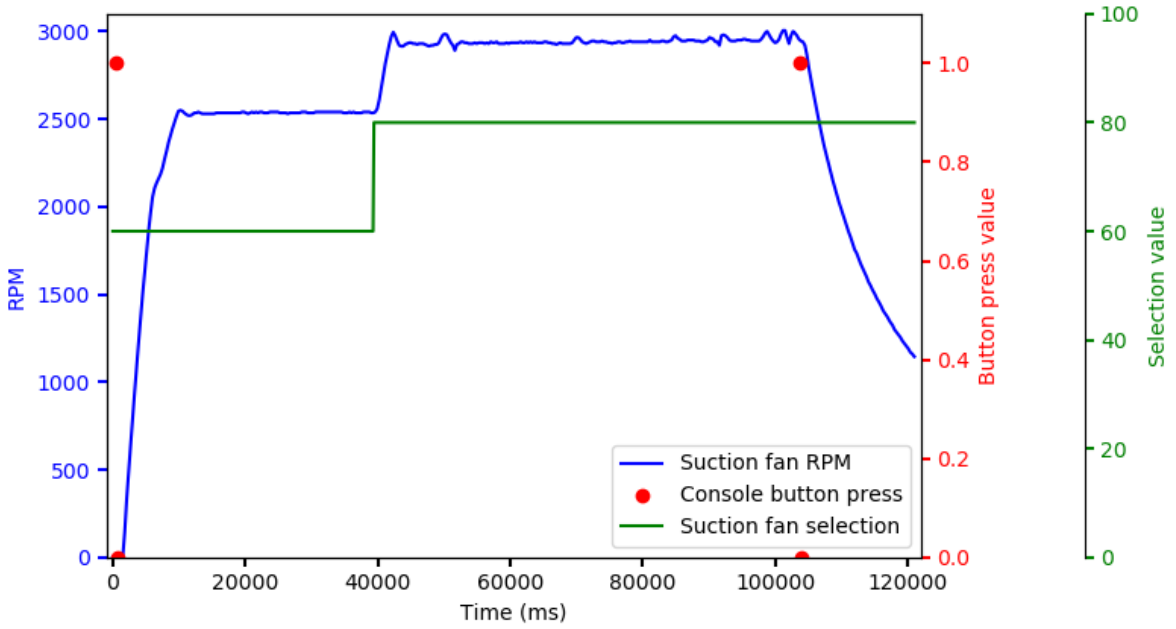

Figure 3.8: The actual values of the *suction fan RPM*, *console button press* and *suction fan selection* parameters through time

value calculated over the interval in which the value is nearly constant.

To quantify the precision of the simulator, several mathematical similarity measures were used. Four total messages are sent describing the console button press. The values are sent in the following order: 1, 0, 1, 0, which is trivial to replicate on the simulator. The only difference between the messages recorded on the vehicle and the messages generated by the simulator are in the message timestamps. The lists of message timestamps were compared using cosine similarity, a similarity measure between two non-zero vectors, that measures the cosine angle between them. This method returned a similarity score of 99.999973%. Another method we used to check the similarity is the dynamic time warping (DTW). DTW is an algorithm for measuring similarity between two time series which can vary in speed. The algorithm minimizes the effects of shifting and distortion in time, making it suitable for checking the similarity of recorded and generated messages with potentially different timestamps. The DTW algorithm calculates the distance $d \in \mathbb{R} \geq 0$ between the two sequences - the distance is smaller the more similar the sequences. To enable the comparison of the results of the DTW algorithm on different variables, the values and times were normalized to values between 0 and 1. The maximal DTW distance is the number of values in the time series, which is 4 for this variable. Distance metrics can be translated to a normalized similarity measure using the following equation:

$$S(x,y) = \frac{M - D(x,y)}{M}, \tag{3.10}$$

where $S(x,y)$ is the similarity between variables x and y, $D(x,y)$ is the distance, and M is the

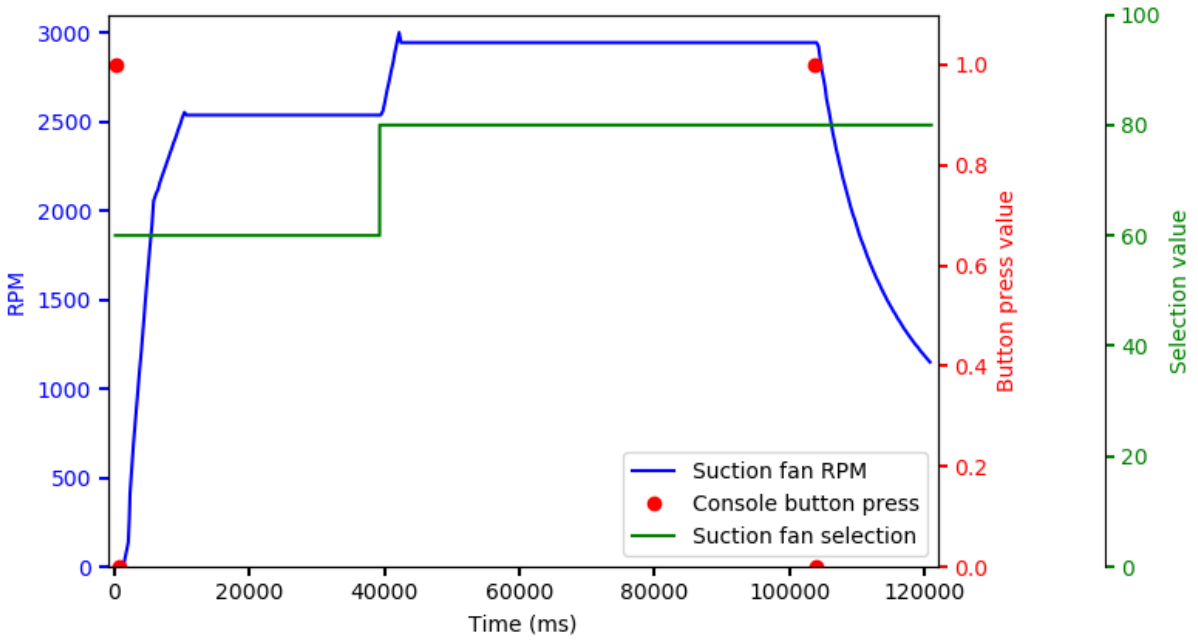

Figure 3.9: The simulated values of the *suction fan RPM*, *console button press* and *suction fan selection* parameters through time

maximum value that $D(x,y)$ could be. The DTW algorithm used to calculate the distance is described in [112] and returned the similarity value of 99.987192%. Frechet distance is another measure of similarity suitable for comparing the two sequences. It calculates the similarity between curves, taking into account the location and order of the points along the curves. On a normalized plane, with times and values between 0 and 1, maximal Frechet distance is $\sqrt{2} = 1.414214$, the length of the diagonal of the plane. The calculated Frechet similarity of the two curves describing the console button press is 99.970996%.

The same methods were used to measure the similarity of the suction fan selection recorded and simulated messages. Again, the simulated values are equal to recorded ones; 303 messages with the suction fan selection value of 60 (`0x3C`) and 628 messages with the value of 80 (`0x50`) - a total of 931 messages sent, one message every 130 ms. The cosine similarity score of the two time vectors is 99.999974%, the calculated DTW similarity score is 99.971393%, while the Frechet similarity amounts to 99.861604%.

The suction fan RPM value movement through time is much more complex than that of the two previously analyzed variables, as the range of values is much larger. Cosine similarity of the time vectors is 99.999935% and cosine similarity of the value vectors is 99.994672%. DTW similarity of the two normalized time series is 99.588656% and Frechet similarity is 98.190726%.

As the described implementation and validation of the simulator show, all of the previously identified requirements that are the consequence of the usage domain were successfully covered. Furthermore, the validation results seen on Table 3.2 show that the simulator can indeed

Variable	Console button press	Suction fan selection	Suction fan RPM
Cosine similarity of timestamps	99.999973%	99.999974%	99.999935%
Cosine similarity of values	100%	100%	99.994672%
Dynamic time warping similarity	99.987192%	99.971393%	99.588656%
Frechet similarity	99.970996%	99.861604%	98.190726%

Table 3.2: Validation results

simulate a CAN bus with a high degree of precision, as all but one similarity measures returned a score above 99.5%.

The vehicle components which output a small range of values that change regularly, such as the console button press and the suction fan selection, can be simulated with 100% precision as seen when validated using the cosine similarity method. However, as the intervals in which the messages generated by various components are sent on the bus are not completely regular in real vehicles, there is a slight difference between real and simulated message timestamps. This is evident in the cosine similarity of timestamps which is not 100%, but is above 99.9999% in the simulation of all three parameters, which is highly satisfactory for the simulator's use case. Dynamic time warping, which allows us to compare curves with similar shapes, but different magnitudes, lengths and phases, returns the similarity above 99.58% for all three parameters, with the parameters which have a smaller value range having a score greater than 99.97%. Finally, the Frechet similarity for the console button press and the suction fan selection parameter is above 99.86%, with the suction fan RPM variable Frechet similarity being the only score among all metrics that is below 99.5% at the value of 98.190726%. This can be attributed to the oscillations in values in the continuous repeat of messages in the data captured from the vehicle. The Frechet distance is the highest in parts where two curves diverge the most, and as the simulator cannot simulate these oscillations, the Frechet similarity score is negatively impacted.

3.4Collection and Contextual Enrichment Modules

Once the data sources from which data is to be collected in this research were identified, and the problem of acquiring automotive data over a CAN bus was solved by developing the CAN bus simulator, the collection and contextual enrichment modules could be designed. As previously stated, smartphone-based connected vehicles provide the most accessible and cost effective way to collect automotive data. For this reason, the collection and contextual enrichment modules are implemented as a part of a smartphone application, more precisely, an Android OS application. Android was chosen as the development platform as it is open source, offers easier access to functionalities needed to implement the smartphone-to-vehicle communication than its competitors and has the highest market share of all mobile operating systems on the market [113].

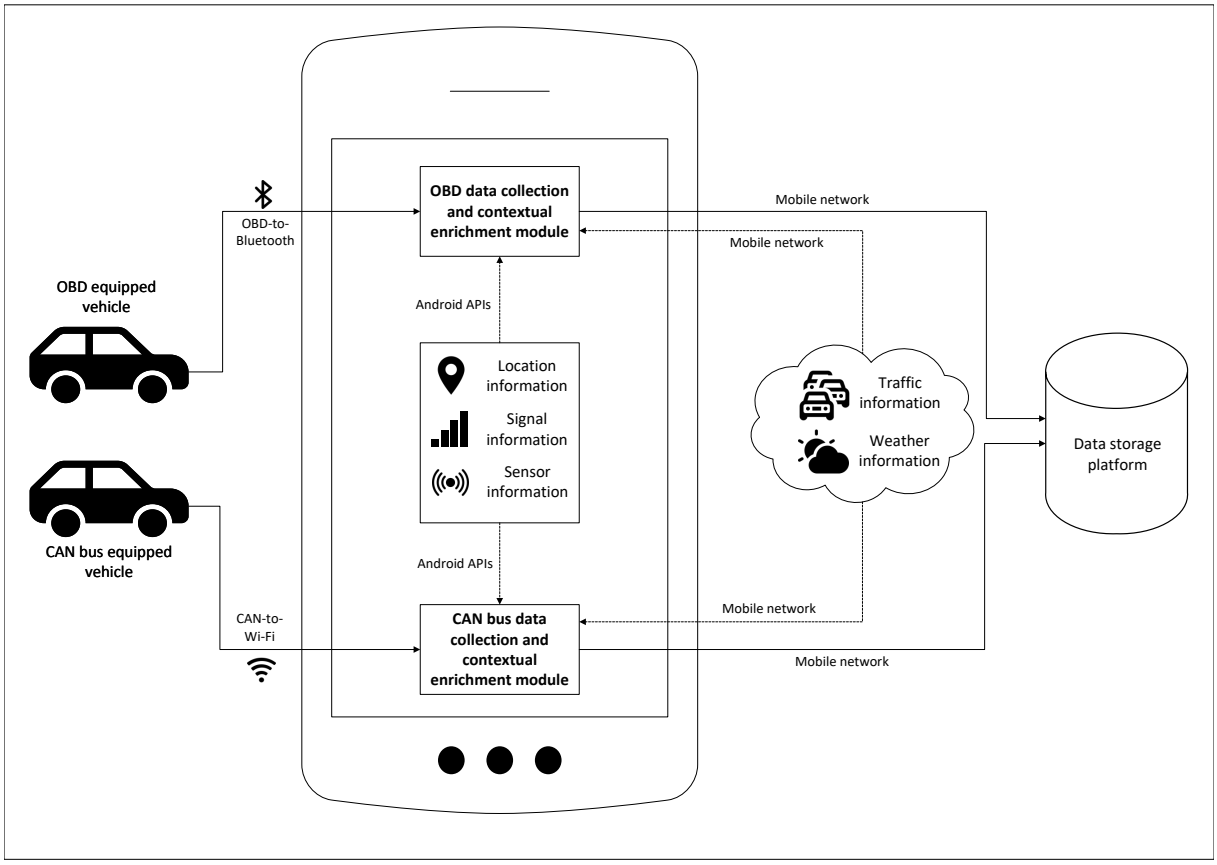

Figure 3.10: Collection and contextual enrichment modules as a part of the framework

3.4.1Architecture

The collection and contextual enrichment modules and their communication with the rest of the framework can be seen in Figure 3.10. To maximise the spectrum of vehicles from which data can be collected, two automotive communication technologies have been selected for automo-

tive data collection - OBD-II and CAN. As these two protocols differ significantly, two different data collection modules have to be utilised. For the communication of these modules with the vehicles, two wireless communication technologies are used - Bluetooth for the collection of OBD-II data, and Wi-Fi for the collection of CAN bus data. These communication technologies can be changed or replaced with other similar technologies with some adjustments being made to the data collection modules. The reasoning behind the usage of these specific communication technologies (Bluetooth and Wi-Fi) is their ease of use and the availability of the hardware communication modules while setting up the collection experiment.

For the collection of OBD-II data, the ELM327 OBD-II-to-Bluetooth programmed microcontroller was used. The sequence diagram describing the communication between the smartphone, the ELM327 and the vehicle is shown in Figure 3.11. For each data point that is to be collected, the smartphone needs to send a request via Bluetooth to the ELM327. The ELM327 then communicates with the vehicle through the OBD interface and propagates the request it received from the smartphone. After sending the request, it waits up to 200 ms for a reply from the vehicle. If none were to come, an internal timer stops the waiting, and the ELM327 responds with a 'NO DATA' message. Parallel requests are not possible because of the way the communication protocol functions, message identification does not exist so the smartphone must wait for a response to arrive before sending a new request to the ELM327. Vehicles typically have a response time of 100 ms, which means that around 10 data points can be collected in one second. There are more than 200 data points available for collection through the OBD interface, and to keep the scanning frequency at around 1 second, we have chosen to collect information described previously in Section 3.2.1 and displayed in Table 3.1.

On the other hand, CAN bus data is collected via a CAN-to-Wi-Fi module, more specifically,

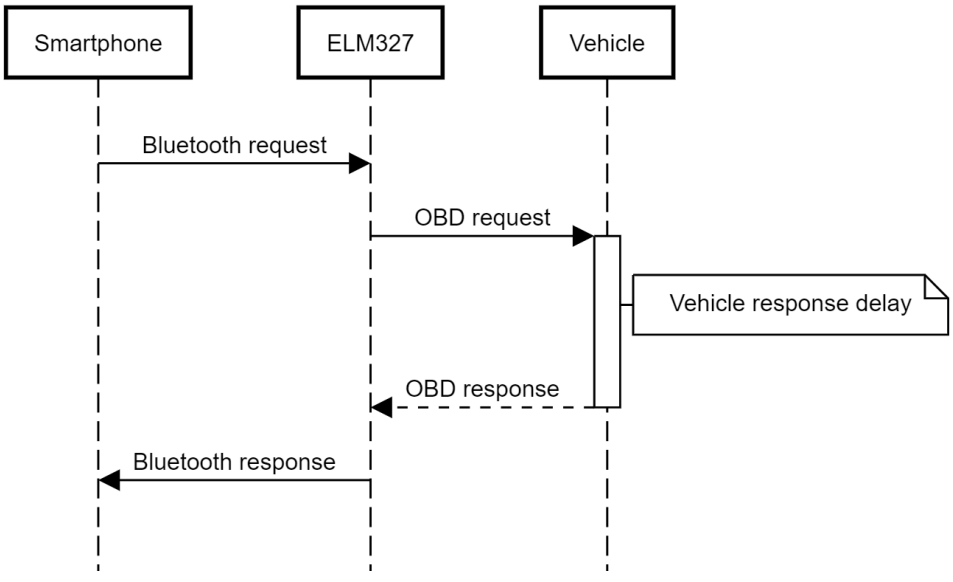

Figure 3.11: Sequence diagram of communication between a smartphone, ELM327 and a vehicle

PEAK Systems' PCAN Wireless Gateway [104]. The CAN bus connects to the gateway by a 9-pin D-Sub connector. The gateway propagates the messages received through the CAN bus to all devices which are connected to its Wi-Fi network by wrapping the CAN frames in TCP or UDP messages. The data structure of one such frame can be seen in Table 3.3. The values of the frame are stored in Network Byte order and the CAN data is stored as single bytes in ascending order. The CAN ID is used to identify the valuable vehicle information whose CAN data is then collected and contextually enriched.

Contextual enrichment of automotive data is done using the data acquired from the smartphone and the Internet, as described in Section 3.2. Location, signal and sensor information are collected from the mobile device using the APIs provided by Android OS, while traffic and weather information is collected from Internet-based APIs via the smartphones mobile network connection. Finally, each time that a single set of automotive data selected for collection (Ta-

Length	**Field name**	**Meaning**	
2 Byte	Length	Total length of the packet (maximum length of a classic CAN frame is 0x24, decimal 36)	
2 Byte	Message Type	Type of the message, value 0x80 represents a classic CAN frame	
8 Byte	Tag	Not used	
4 Byte	Timestamp Low	Timestamp of CAN messages in μs	
4 Byte	Timestamp High		
1 Byte	Channel	Not used	
1 Byte	DLC	Data Length Count (DLC) indicates the length of the CAN data in bytes	
2 Byte	Flags	Value can be: 0x01 - Message is a Remote Transmission Request 0x02 - Message has an Extended ID	
4 Byte	CAN ID	Bit 0-28 Bit 29 Bit 30 Bit 31	ID Fixed value 0 RTR 1 for Extended frame, 0 for Standard frame
8 Byte	CAN Data	8 bytes containing data	

Table 3.3: CAN data frame structure

ble 3.1) is collected from a vehicle and enriched with context by the collection and contextual enrichment module, it is sent to the distributed data storage platform.

3.4.2Implementation

Both the OBD-II and CAN collection and contextual enrichment modules are implemented in a smartphone application for Android OS devices. The application is based on the Android OBD Reader open-source project [114] which offers built-in ELM327 communication capabilities (the application communicating with the ELM327 device connected to a vehicle is shown in Figure 3.12). For the purpose of communication with the CAN-to-Wi-Fi gateway, the application uses server sockets provided in Java programming language.

The UI of the application can be seen in Figure 3.13. The top of the application displays information regarding speed (current vehicle speed collected from CAN/OBD-II, and current and average traffic speed collected from the TomTom traffic API) and RPM. Information about

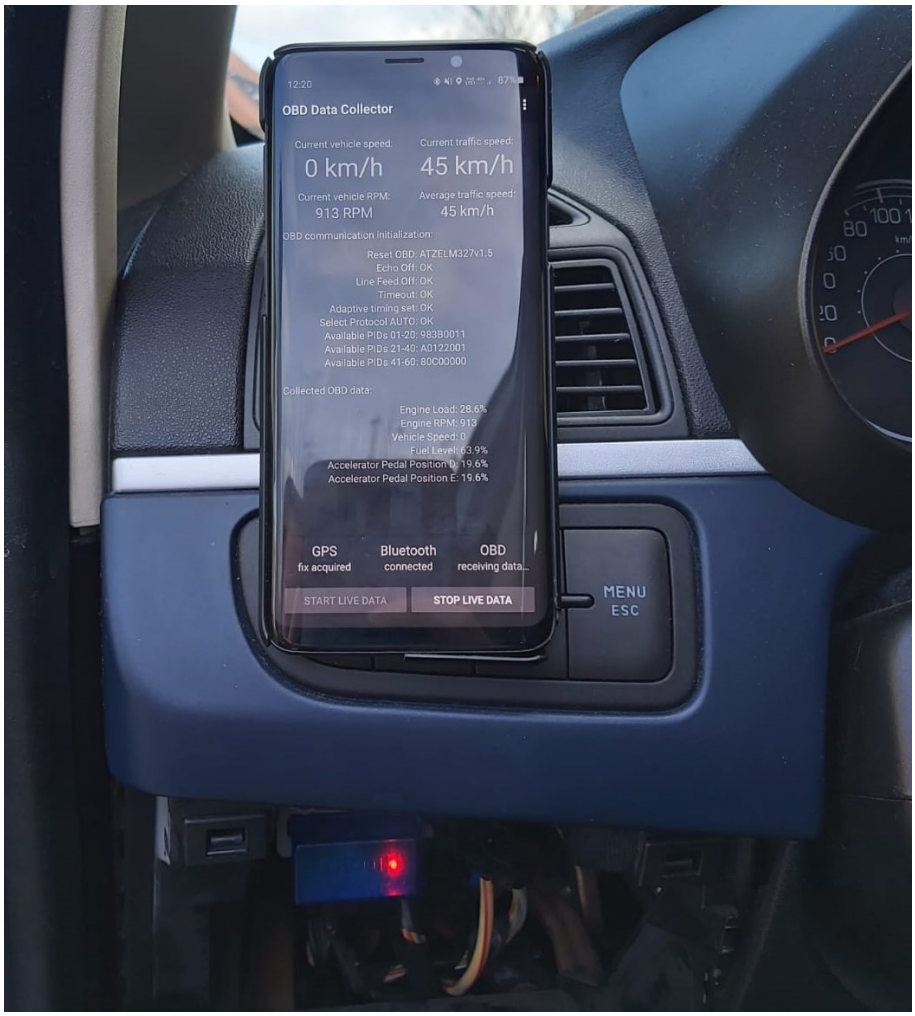

Figure 3.12: Automotive data collector application collecting automotive data via ELM-327 Bluetooth device

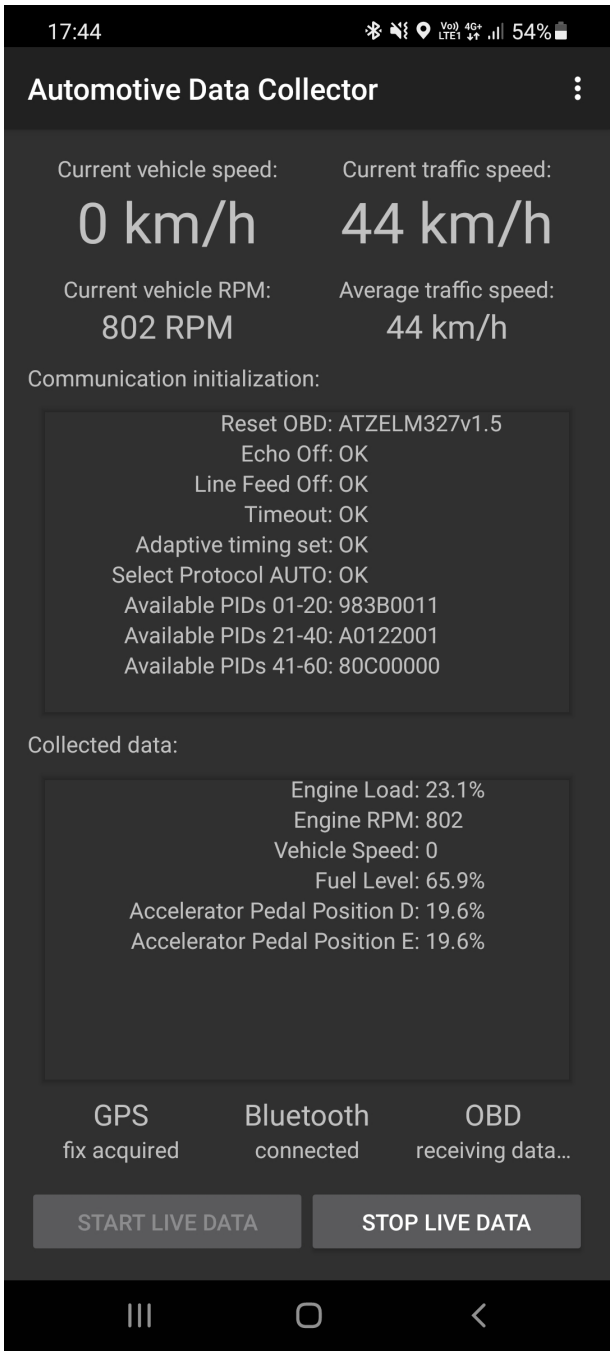

Figure 3.13: Automotive Data Collector main screen while collecting automotive data

communication initialization is printed out in the box below the speed information. Protocol messages and various other communication data is printed here in order to provide an easier way to troubleshoot potential problems in communication between the smartphone and gateways. Under the communication initialization box is the collected data box. In this box, the data collected from the vehicle is displayed and refreshed in real time as the smartphone receives it. On the bottom of the screen, information about GPS location status and Bluetooth and OBD-II/CAN connection status is visible along with the buttons for starting and stopping live automotive data collection.

To provide additional context to the collected data, vehicle and driver meta information is also collected. Vehicle and driver meta information can be described as follows:

$$V_{info} = (man, mod, year, fuel) \tag{3.11}$$

$$D_{info} = (gen, age, exp) \tag{3.12}$$

where vehicle meta information (V_{info}) contains information about vehicle manufacturer (man), model (mod), year of production ($year$) and fuel type ($fuel$), and driver meta information (D_{info}) describes the driver's gender (gen), age (age) and years of driving experience (exp). This

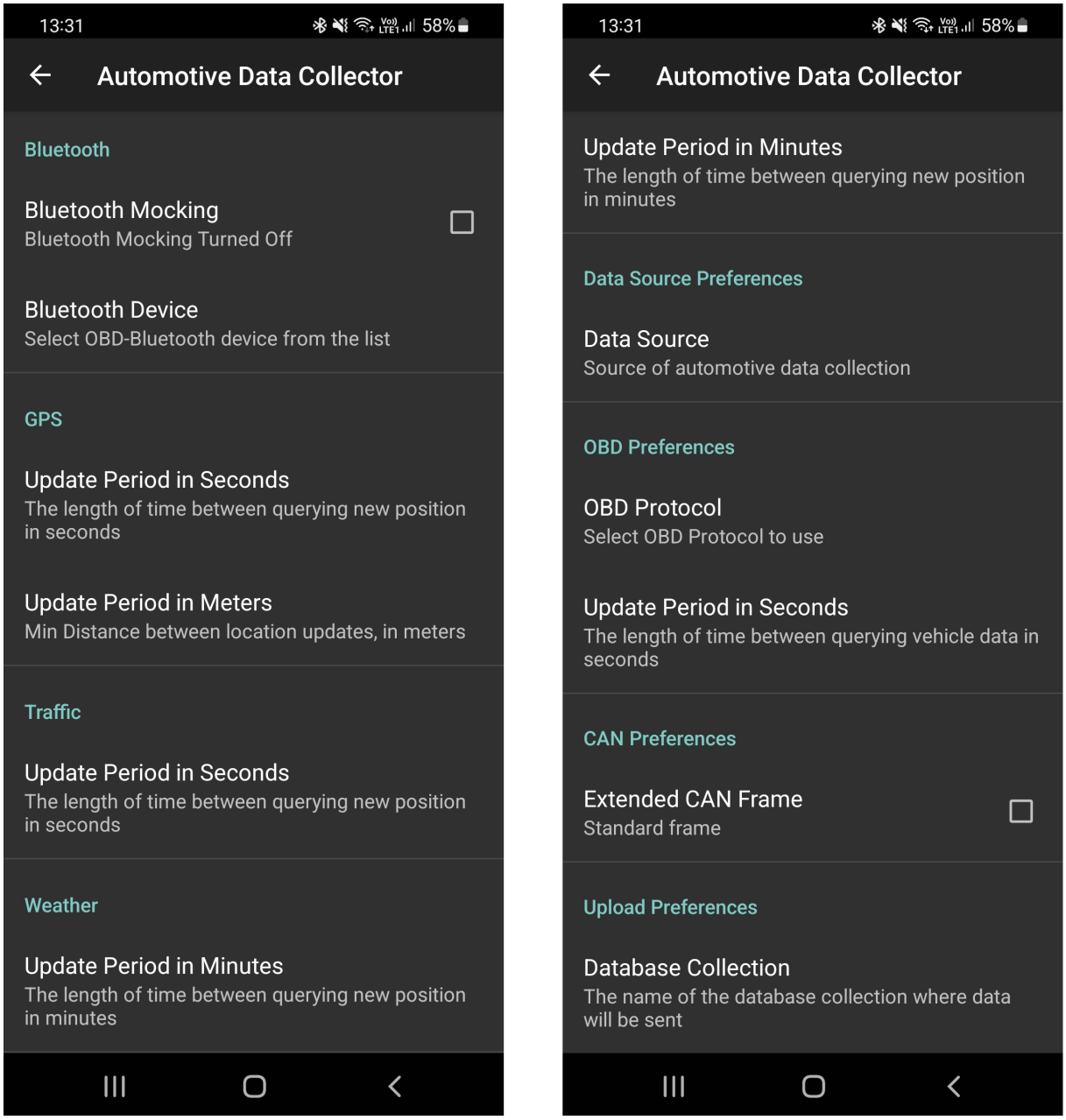

Figure 3.14: Automotive Data Collector settings screen

is not something that can be collected from the OBD port or the CAN bus, which is why it was implemented as a part of the application. The user is prompted by the application to enter vehicle and driver meta information before the first time the data collection is initialized on a vehicle.

To allow the user to change the properties of data collection, the application also offers a settings screen (Figure 3.14). The available settings can be defined as the following sets:

$$S_{app} = (S_{blu}, S_{gps}, S_{trf}, S_{wtr}, S_{src}, S_{obd}, S_{can}, S_{upl}) \tag{3.13}$$

$$S_{blu} = (mock \,|\, \neg mock, dev) \tag{3.14}$$

$$S_{gps} = (T_s, D_m) \tag{3.15}$$

$$S_{trf} = T_s \tag{3.16}$$

$$S_{wtr} = T_{min} \tag{3.17}$$

$$S_{src} = CAN \,|\, OBD \tag{3.18}$$

$$S_{obd} = (prot, T_s) \tag{3.19}$$

$$S_{can} = (ext \,|\, \neg ext) \tag{3.20}$$

$$S_{upl} = db \tag{3.21}$$

The application settings (S_{app}) include Bluetooth (S_{blu}), GPS (S_{gps}), traffic data (S_{trf}), weather data (S_{wtr}), automotive data source (S_{src}), OBD (S_{obd}), CAN (S_{can}) and upload settings (S_{upl}). Bluetooth settings (S_{blu}) offer the option to select whether the data will be mocked or not ($mock \,|\, \neg mock$) and if the data is not mocked, from which Bluetooth device it will be collected (dev). The user can also change GPS data collection settings (S_{gps}), more specifically, the update period in seconds (T_s) and update distance in meters (D_m) which have the default values of 1 second and 5 meters. Traffic settings (S_{trf}) allow changing of the update period in seconds (T_s) which is set to 8 seconds by default. Similarly, using the weather settings (S_{wtr}), the user can change the weather information update period (T_{min}) which is set to 10 minutes by default. The source of automotive data collection (S_{src}) can be set to OBD or CAN. In OBD data collection settings (S_{obd}), OBD protocol which will be used can be set ($prot$). The OBD-II protocols available for selection in the application can be seen in Table 3.4. The default OBD protocol option is "AUTO", using which the ELM-327 device automatically determines and selects the compatible protocol. The update period of OBD data collection can also be set (T_s) and amounts to 1 second by default. If CAN bus is chosen as a data source, the CAN frame size can be provided (S_{can}) and can be set to either extended (ext) or standard ($\neg ext$). Finally, the upload settings (S_{upl}) offer the option to select the database collection (db) in which the contextually

Table 3.4: OBD-II protocols available for selection in the collection application

OBD-II Protocol	Bus System	Speed (kbps)	Vehicle Manufacturers
SAE J1850 PWM	J1850	41.6	Ford
SAE J1850 VPW	J1850	10.4	GM
ISO 9141-2	K-line	10.4	Chrysler, European, Asian
ISO 14230-4 KWP	K-line (5-baud init)	10.4	Chrysler, European, Asian
ISO 14230-4 KWP FAST	K-line (fast init)	10.4	Chrysler, European, Asian
ISO 15765-4 CAN	CAN (11 bit ID)	500	European (2003 or later), American (2008 or later)
ISO 15765-4 CAN B	CAN (29 bit ID)	500	European (2003 or later), American (2008 or later)
ISO 15765-4 CAN C	CAN (11 bit ID)	250	European (2003 or later), American (2008 or later)
ISO 15765-4 CAN D	CAN (29 bit ID)	250	European (2003 or later), American (2008 or later)
SAE J1939 CAN	CAN (29 bit ID)	500	heavy duty vehicles

enriched automotive data set will be stored.

3.4.3Validation

The application running collection and contextual modules was validated by testing it through several phases. Once the first version of the application was complete, alpha testing was performed. Alpha testing served as acceptance testing, and it was performed by several drivers running the application on their smartphones which communicated with their vehicles. To identify all possible issues and bugs before releasing the application to its intended users which would participate in the data collection experiment, the application was equipped with Firebase Crashlytics support. Firebase Crashlytics [115] is a real-time crash report tool which intelligently groups crashes and highlights the circumstances that lead up to them. During this phase of the testing process, multiple issues were identified and fixed. Several application crashes occurred and the causes were captured by Crashlytics which facilitated the fixing process. Additionally, an a single vehicle, the Bluetooth module could not retrieve the vehicle data due to the incorrect OBD protocol chosen automatically by the application. In this case, the user has to manually find and select the appropriate OBD protocol in the application settings. To avoid this happening to the drivers participating in the data collection experiment, detailed instructions on identifying and choosing the appropriate OBD protocol were created and provided to the

drivers.

Once alpha testing was complete, the application was provided to the end users and beta testing was performed. All of the nine drivers which were to be participants in the data collection experiment were using the application to collect data from their vehicles. Due to the difference in vehicle and smartphone manufacturers and their combinations, several more bugs were noticed and fixed. During beta testing, the data was visualized to identify potential problems and inconsistencies in the collected data. For example, location API could sometimes return incorrect location information which is visible only when the trip route is overlaid on a map. A process for identifying and fixing these inconsistencies was established during the beta testing phase. After a week of error-free beta testing the application was ready to be used in the data collection experiment.

3.5Distributed Data Storage Platform

The next part of the framework is the storage platform in which the data collected by the collection and contextual enrichment modules is stored. In this section, the data platform that supports both, data streaming and storage developed for the purposes of automotive data collection will be described. First, the architecture is presented through the conceptual model of the data platform which states the basic requirements, along with the challenges that need to be solved, and through the queuing theory perspective which focuses on the data flow and specific elements necessary to ensure minimal data loss. Afterwards, the details concerning the implementation of the data platform which is based on the conceptual and queuing network model are provided. Finally, the validation of the data platform functionalities is described.

3.5.1Architecture

As evident from the literature and feedback from the real-life projects, the main requirements for the data platform are reliability, scalability and robustness. In the case of this data platform, *reliability* refers to high availability and data loss minimization. Figure 3.15 displays the platform's conceptual model and reveals that the data platform is expected to have a single *entry point*, as well as a single *exit point*, suggesting that the first challenge is to ensure its high availability. The second requirement is *scalability*, which means that the data platform needs to be adaptable to the potential data increases, be it due to the higher number of messages (i.e., higher bandwidth), or to the size of the messages being larger in size [116]. The aforementioned is depicted in the Figure 3.15 as multiple data flows. The feature of the data platform, that emerges regarding scalability is how to support adding more resources to existing nodes within the data platform, or to add more nodes to handle the incoming data when the number of data producing entities increases. The third, and the most important requirement, is *robustness* [117] or *fault-*

tolerance. Fault-tolerance is of great importance for the data platform, since it ensures that the platform is still functional even in the case of some nodes shutting down as a consequence of hardware or software failure, which is known to be a challenging task to achieve [118].

After assuring that all the requirements stated above are met, a way to monitor the data platform and ensure its compatibility with different data sources and data structures needs to be established. Such a challenge is solved within the data storage component of the platform. Concerning the functionality, the data platform should ensure both real-time data streaming and access to historical data, as one of the functionality, will be real-time vehicle fleet management, and prediction of vehicle failures based on the historical data. Therefore, the data platform architecture is based on *lambda architecture* [79] designed for stream and batch processing.

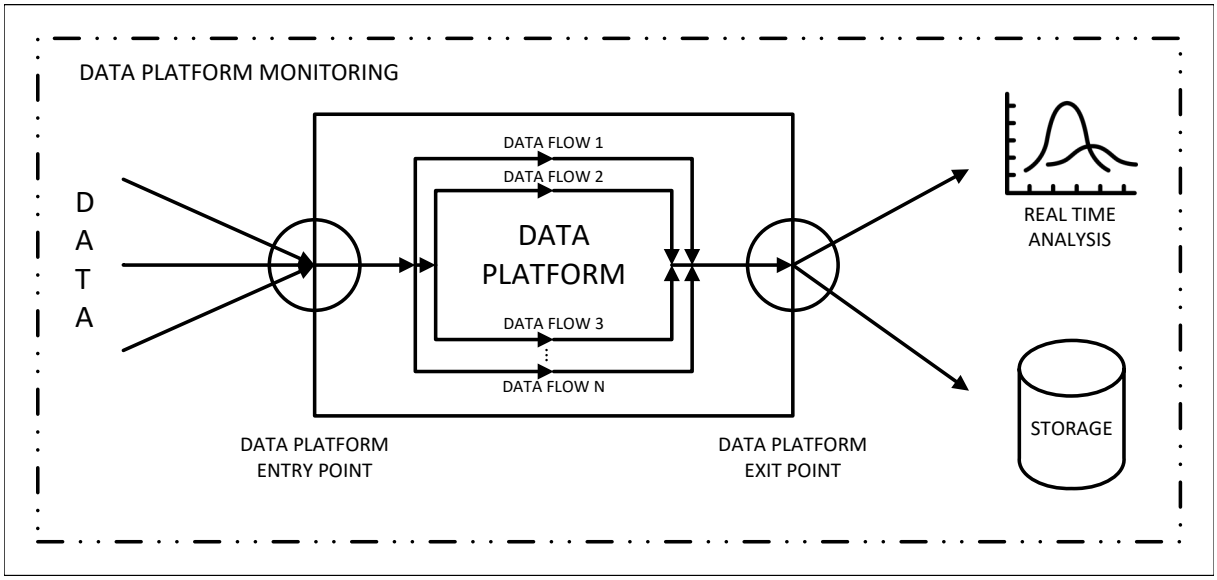

Figure 3.15: A conceptual model of the data platform

Following the requirements outlined in the conceptual model, a queueing network model was designed, presented in Figure 3.16. The following variables are introduced to the model: λ - indicating the data arrival rate to the platform, and μ - indicating the data service rate of the platform. The ultimate goal of the platform is to propagate all received messages, with minimal loss. In all further queueing network calculations, the hyphotesis of the data loss being zero will be applied, meaning that the data service rate is greater than the arrival rate $\mu > \lambda$.

To create a single, high availability entry point, multiple queues must share a single address, with a single queue processing all requests until failure, when another queue steps in its place. These queues also serve as *load balancers*, distributing the data evenly to the queues in the data collection part of the queueing network:

$$\lambda_{c1} = \lambda_{c2} = ... = \lambda_{cn} \tag{3.22}$$

$$\lambda_{c1} + \lambda_{c2} + ... + \lambda_{cn} = \lambda \tag{3.23}$$

Each of the *data collection* queues is expected to have maximal throughput, $\lambda_{cn} = \mu_{cn}$, and in the case of a failure of a queue, load balancers distribute its load on other operating queues. To ensure that data is not lost in the case of a failure of a queue containing data, a *data replication* cluster is in the middle of the platform. In the data replication cluster, all queues synchronize with each other until each queue contains all the data which is currently in the data platform.

The final part of the queueing network consists of *data propagation* queues. They take the data independently from the data replication cluster and propagate it to the exit of the platform. Here, the data does not have to be distributed evenly over the queues, but the queues have to propagate all the data that entered the platform to the exit point:

$$\lambda_{p1} \neq \lambda_{p2} \neq ... \neq \lambda_{pn} \tag{3.24}$$

$$\lambda_{p1} + \lambda_{p2} + ... + \lambda_{pn} = \lambda < \mu_{p1} + \mu_{p2} + ... + \mu_{pn} = \mu \tag{3.25}$$

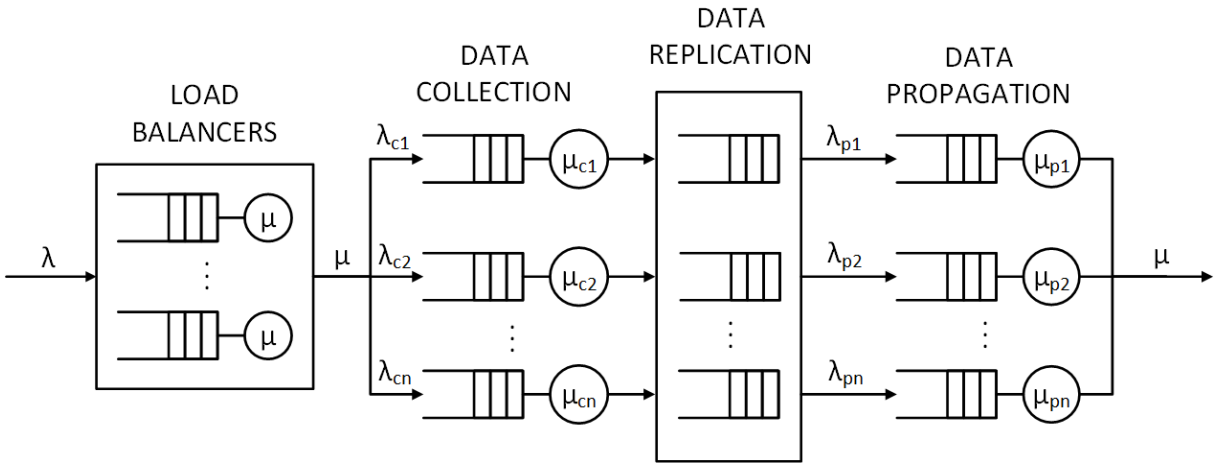

Figure 3.16: A queueing network model of the data platform

3.5.2Implementation

For the implementation of the data platform, specific technologies were targeted to address all the challenges, i.e., scalability, robustness, and fault tolerance. Figure 3.17 depicts the final components used in the data platform, as well as the *end-to-end* data flow. Components are realized on separate virtual machines with the same specifications. All machines are running the latest stable Debian distribution, i.e., Debian 9, code name *"Stretch"*. Since the data platform is implemented on virtual machines, hardware specifications can be fine-tuned considering the demands of the use-case. However, the data platform was implemented on an identical setup hardware-wise; 16 GB of RAM memory, 8 CPU cores (4 CPU's with 2 cores) with 2.2 GHz frequency per core. The following paragraphs describe components used in the realization of the data platform.

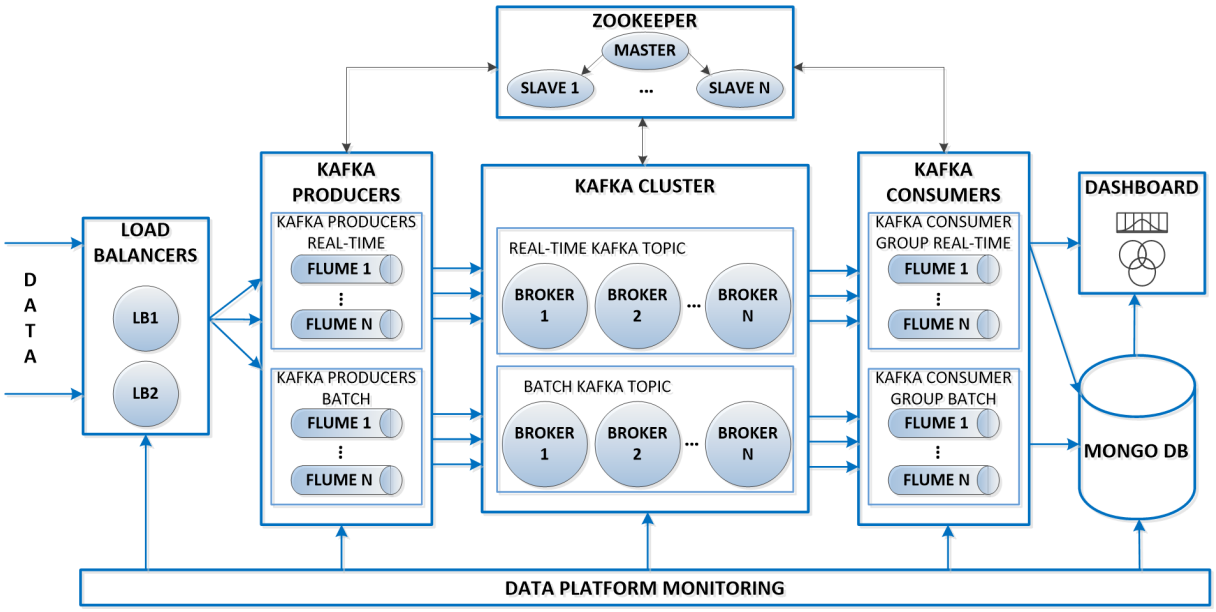

Figure 3.17: The proposed data platform with implemented modules

Load balancers

As stated in Section 3.5.1, the most prominent challenge relates to the single point of failure at the data platform entry point (see Figure 3.15). To achieve *one-point access* to the data platform, two *load balancers* were used, a master and a slave, using the *HAProxy and Keepalived* [†] solutions. The two load balancers share a *single floating IP address* on which the data can be sent to the data platform. The floating IP address is assigned to the master balancer by default. All data traffic travels through the master load balancer and, in the case of its failure or unexpected shut down, the slave balancer takes over the role of the master balancer (and the floating IP address) until the master balancer goes back online. The load balancer forwards the data to the Apache Flume entry points in a round-robin technique, while periodically checking their availability. Such a design choice ensures that the system is fault tolerant from the perspective of the single point of failure, which without the load balancers would be the data platform entry point.

Kafka producers and consumers

The entry and exit points of the platform are implemented using multiple *Apache Flume agents* on multiple data platform nodes with different addresses. Apache Flume [‡] is a Java based application that serves as a reliable channel between a wide variety of different source and sink types. The data platform utilizes two different types of Flume agents; one type for the data platform entry point and the other for the exit point. Each Flume agent consists of three

[†]http://www.haproxy.org
[‡]https://flume.apache.org

components: *source*, *channel*, and *sink*. On the data platform entry point, the Flume agent has an Avro source type [119], i.e., plain-text data serialization mechanism, which is used to receive data over the Internet. It also implements the Kafka sink to send the data *to* the Kafka cluster, as depicted on Figure 3.17. On the data platform exit point, the Flume agent has a Kafka source type to receive the data flow *from* the Kafka cluster. Since a MongoDB database is used as the platform's data storage solution, the exit point Flume agent implements a MongoDB sink type, with an interceptor that defines the MongoDB collection in which each data fragment should be stored. All Flume agents implement the file type channel. The file type channel is chosen because of its robustness (it is not implemented in RAM memory, as opposed to the Memory type channel), and persistence in the case of an agent failure. Although data events are not stored in the *fast* RAM memory, file type channel is able to withstand momentous throughput, i.e., tens of millions of events per minute [120].

Zookeeper and Kafka cluster

Together with aforementioned Flume, Apache Zookeeper [§] and Kafka [¶] are the core elements of the data platform. Kafka is the main insurance of reliability while streaming real-time generated data between different systems [121]. Kafka's functionality is similar to publish-subscribe based systems; Kafka producers define the topic for which they are producing, while Kafka consumers define the topic from which they will consume the data. The data platform consists of multiple Kafka nodes and all of those nodes are responsible for a specific topic - a real-time data streaming topic or a batch data streaming topic (see Figure 3.17). To achieve concurrency in Kafka based data platforms, each topic is split into data partitions and specific nodes are responsible for only a specific number of partitions. Besides concurrency, Kafka ensures fault-tolerance with user defined number of brokers and replicas per topic, i.e., if one of the brokers becomes unavailable, his replica broker will resume his task. Data that is streamed through the platform stays in the internal memory of Kafka brokers for a user specified amount of time, even after the data is consumed. Therefore, the data can be retrieved even if the storage destination is not. Zookeeper's task is to coordinate the different Kafka brokers and their corresponding topic partitions, as well as to ensure that one of the Zookeeper nodes is always elected as a leader by other alive nodes, which guarantees robustness from the perspective of Kafka node management [122].

MongoDB

For the purpose of data storage, the data platform implements an open source MongoDB database [123]. MongoDB provides an *easy-to-use* mechanic for scalability and flexibility, which is one

[§] https://zookeeper.apache.org
[¶] https://kafka.apache.org

of the reasons why it is commonly used in cloud computing architectures. MongoDB is also characterized by high writing and reading speed, which is especially favorable for big data analysis [124], and therefore MongoDB is a perfect fit for the data platform.

Data platform monitor

The monitoring dashboard is an important part of the data platform, as it is used to monitor the data platform health and workflow. For this reason, the Cockpit dashboard [125], an open source software to monitor the workflow and health of each node in the data platform, is used. The Cockpit dashboard connects to all the entities in the data platform using the *SSH* protocol and enables its users to remotely manage all entities through a virtual system terminal, as well as to monitor all active processes and resource usage. The HAProxy load balancing solution periodically ensures that all other entities in the data platform are available and offers a Web interface where their status is visualized, i.e., green - available, yellow - starting, and red - not available (Figure 3.18). Besides visualizing the data platform entities status, HAProxy can send alerts to the administrator email if one or more components are not available.

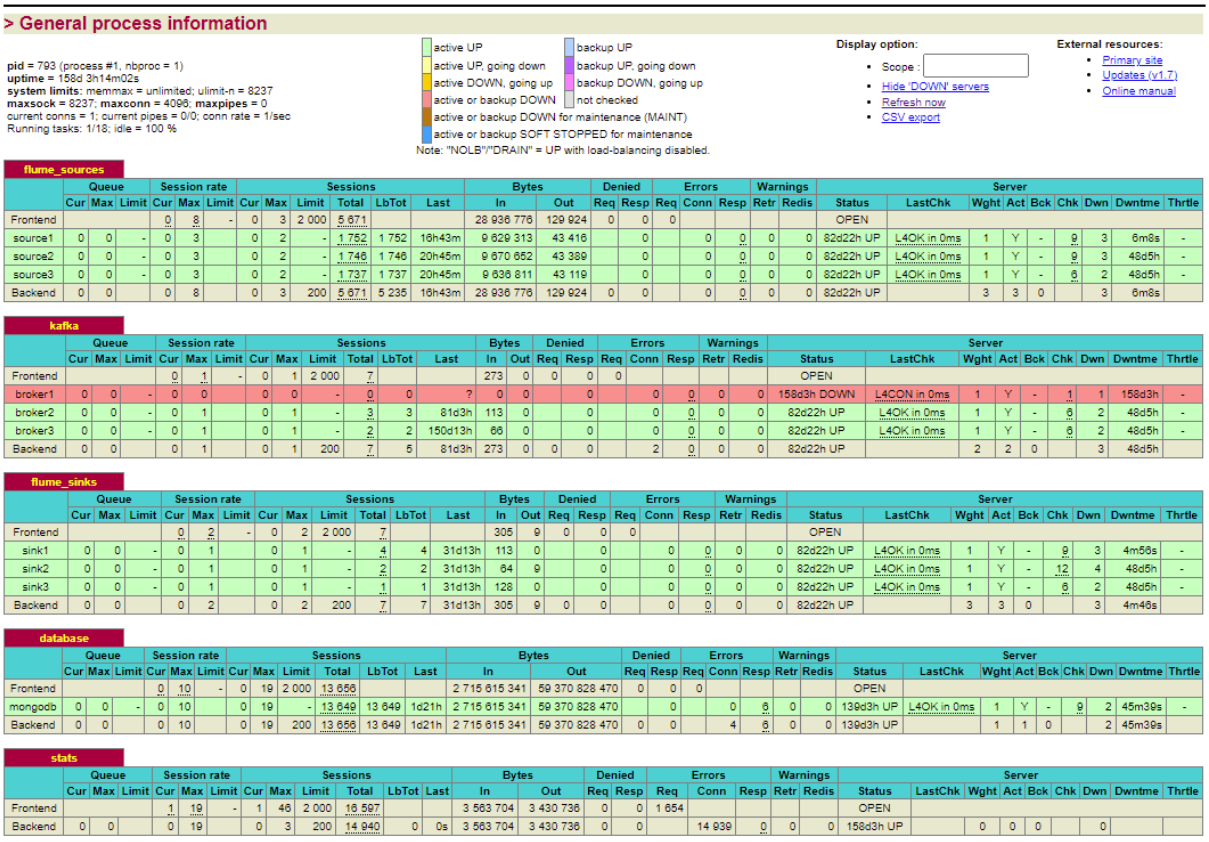

Figure 3.18: The monitoring dashboard displaying the availability of data platform nodes (all nodes are available except Kafka broker #1, which is marked by red color)

3.5.3Validation

To validate the platform, it was tested on three different testing levels: unit testing, integration testing and system testing. Firstly, each component of the system was unit-tested to ensure that it is functioning properly. While designing the system, the first thing that was developed was the Kafka cluster. After the Kafka cluster, each component added into the system was integration-tested verifying the interfaces between components. Finally, system testing was executed to verify that the system meets its requirements: reliability, scalability and robustness. To test reliability and robustness, a continuous stream of data was sent to the platform and the components were switched off one by one, and in different combinations, and it was observed if all of the sent data arrived at the other end of the platform. The test was completed successfully when all the data sent to the entry point of the platform was received at the exit point. The expected maximum load of the platform is around 30,000 messages per second of a size not bigger than 128 bytes. To test the reliability of the platform under these conditions a stress test was executed with the doubled amount of messages with double size, resulting in 60,000 messages per second with a size of 256 bytes. The platform passed the stress test without issues. To test scalability, another stress test was performed which sent increasingly higher amounts of data until the platform could not ingest it anymore. After that, new nodes were added to the platform architecture, increasing its throughput and the test was executed once again. After adding additional resources to the platform, it passed the test, indicating that the scalability requirement is satisfied.

3.6Application Programming Interface for Data Acquirement and Analytics

With the collected and contextually enriched automotive data being stored in big data storage platforms [22], a middleware solution is needed to provide an abstraction layer for the stored data, and this is where application programming interfaces (APIs) come into play. There are multiple benefits of employing an API for accessing the data instead of accessing the data directly from the database. Firstly, using an API means that the database software can be changed at any time, and all of the possible application consuming the data through the API will not have to change, only the API will need to be modified. Secondly, the API can be used by multiple clients who do not have to write their own queries for fetching the data. Next, the API allows the control over which data collections can be accessed through it and in what format. Finally, statistical and analytical calculations can be included as a part of the API, making it generate and process the data before exposing it to the end-user.

3.6.1Architecture

The API is achieved as a representational state transfer (REST) system, meaning, among other things, that specific data resources are uniquely identified by uniform resource locators (URLs). The API is implemented in a way that satisfies the OpenAPI Initiative's (OAI) Open API Specification. The OpenAPI Specification (OAS) defines a standard, programming language-agnostic interface description for REST APIs [126]. By conforming to the OpenAPI specification we ensure that both humans and computers are able to discover and understand the capabilities of our service without requiring access to source code, additional documentation, or inspection of network traffic.

The API is designed to provide an abstraction layer over the stored contextually enriched automotive data. This is reflected in the API's architecture (Figure 3.19). Once the automotive data is collected on a vehicle, relayed over Bluetooth to the smartphone and contextually enriched, it is sent to the distributed data storage platform. The first functionality of the API is to provide structured access to the collected data. This is achieved by exposing the collected data stored in the data platform via the operations offered by the API's *data acquisition module*, which will be explored later.

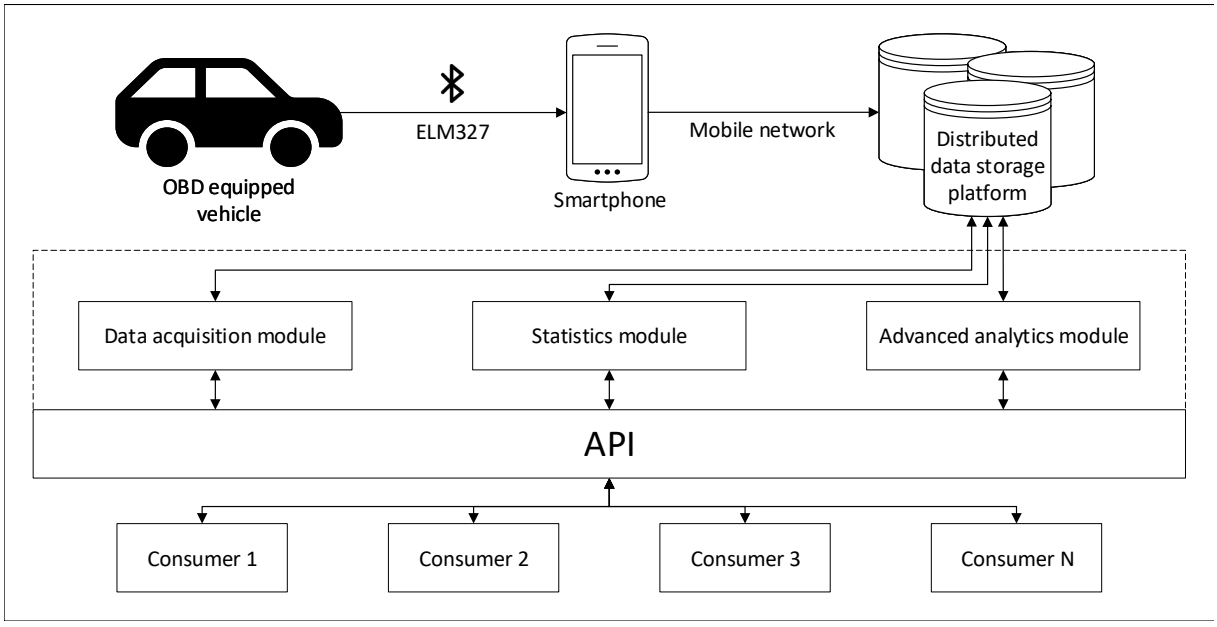

Figure 3.19: The architecture of the API for advanced analytics of contextually enriched automotive data

The API offers additional functionalities through the other two of its modules. *Statistics module* is used for asynchronous generation of statistical information which describes the collected data set. To start the generation process, a consumer can send a request to the API. The process of statistics generation is performed in the background and the consumer can communicate with the statistics module to check the progress. Once the statistics generation process

is complete, it saves the statistical data to the data storage platform and this data is available for access through the API. Similarly, *advanced analytics module* is used to generate analytical information. Generation of analytical information is started by the consumer, it is executed as a background process, and its progress can also be tracked via the API. The difference between the two modules is in the techniques used to generate new data. While the statistics module uses statistical and mathematical methods to calculate statistical information, advanced analytics module can use artificial intelligence methods to generate analytics results. Both modules can be easily replaced or upgraded while the API is operational, as they function separately from the API as is depicted on Figure 3.19.

3.6.2Implementation

The entirety of the API is developed in Python using Flask micro web framework. As the database used for data storage is MongoDB, PyMongo distribution which contains tools for interacting with MongoDB database was utilized for the communication with the data storage platform. All of the elements described in the data model are returned as JSON objects by the API. To achieve the desired modularity, statistics and advanced analytics modules were implemented as standalone threads which can be called upon to execute when desired. The API uses basic authentication for access to all of its available operations, with the users who are allowed to access the API being provisioned by the API's administrator. Monitoring is achieved using the Flask Monitoring Dashboard, providing information such as median request duration, number of hits, hourly and daily API utilization, which are used to quantify the API's performance and troubleshoot if a problem occurs.

The operations that are available as a part of the API can be seen in Table 3.5. Each element which is a part of the final data model is exposed for access via a corresponding operation offered by the API. All data points and elements will be described in detail in Section 4.1. To ensure the API's robustness, each operation has a rate limit of one request per second. There are two root URL elements for acquiring stored data: */trip* and */driver*. The URLs with */trip* as their root element are used for fetching trip data. *Trip* elements can be acquired as a list of all stored trips, or a single *Trip* element can be returned using the trip's ID. Trip ID can also be used to fetch a list of the trip's *DataPoints*, *OBDData*, *LocationData*, *TrafficData*, *WeatherData*, *SignalData* or *SensorData*. To fetch driver data, URLs with */driver* root element can be used. Similarly to trip related operations, a list of all *Driver* elements can be acquired, or a single *Driver* element can be returned using the driver's ID. In addition to this, the driver ID can be used to fetch a list of driver's *Trips*, the driver's *MobileDevice* or the driver's *Vehicle*. All of the previously mentioned URLs are accessed via HTTP GET method.

To access the previously generated data set statistics, */statistics* URL is used. If statistics generation has not been performed, this URL will return an empty list. To run the statistics

Table 3.5: Available Operations for Acquiring Contextually Enriched Automotive Data

Data type	URL	Method	Function
List<Trip>	/trip/list	GET	Returns a list of all recorded Trips
Trip	/trip/{id}	GET	Returns a Trip with the given id
List<DataPoint>	/trip/{id}/data	GET	Returns a list of the Trip's DataPoints
List<OBDData>	/trip/{id}/obd-data	GET	Returns a list of the Trip's OBDData
List<LocationData>	/trip/{id}/location-data	GET	Returns a list of the Trip's Location-Data
List<TrafficData>	/trip/{id}/traffic-data	GET	Returns a list of the Trip's Traffic-Data
List<WeatherData>	/trip/{id}/weather-data	GET	Returns a list of the Trip's Weather-Data
List<SignalData>	/trip/{id}/signal-data	GET	Returns a list of the Trip's SignalData
List<SensorData>	/trip/{id}/sensor-data	GET	Returns a list of the Trip's Sensor-Data
List<Driver>	/driver/list	GET	Returns a list of all Drivers
Driver	/driver/{id}	GET	Returns a Driver with the given id
List<Trip>	/driver/{id}/trips	GET	Returns a list of Driver's Trips
MobileDevice	/driver/{id}/mobile-device	GET	Returns a Driver's MobileDevice
Vehicle	/driver/{id}/vehicle	GET	Returns a Driver's Vehicle
List<Object>	/statistics	GET	Returns generated statistics
String	/statistics/generate	POST	Runs statistics generation
String	/statistics/generate	GET	Returns statistics generation process status
List<Object>	/analytics	GET	Returns generated analytics results
String	/analytics/run	POST	Runs analytics
String	/analytics/run	GET	Returns analytics process status

generation, a POST request must be performed to the */statistics/generate* URL with the body of the message describing the type of statistics to be generated (e.g. *{"stats_type": "speed"}* needs to be sent as body of the POST request for the generation of speed-related statistical information). By sending a GET request to the same URL, the current status of the statistics generation process is acquired (*"in progress"*, *"not running"*). Currently available types of statistics which can be generated are *"cumulative"* (calculating the statistics of the entire data set such as total/average trip duration and total/average trip distance), *"speed"* (calculating the average speed values by trip and driver), *"acceleration"* (calculating the average acceleration

values by trip and driver) and *"all"* (calculating all available statistics). Due to the modularity of the API architecture, new operations can be added by upgrading the statistics module with the desired operation and by specifying the corresponding "stats_type" value to be received in the request.

Working with the analytics module is similar to the described statistics generation process. Analytics results can be accessed using the */analytics* URL. If the analytics has not been run before accessing this URL, an empty list will be returned. The analytics process is started by sending a POST request to */analytics/run* URL containing a *"analytics_type"* field in the body of the message desribing the type of analytics process that should be run. A GET request to the same URL, returns the status of the analytics process (*"in progress"*, *"not running"*). The currently supported *"analytics_type"* is *"eco_index"*, a calculated measurement which evaluates eco-efficient driving patterns of a group of drivers, described in more detail in [127]. The method for adding new analytics operations is analogous to adding statistics operation, the analytics module needs to be upgraded and a new *"analytics_type"* needs to be added.

3.6.3Validation

The API was validated using the Inspector tool provided by the Swagger Project [128], which presents a set of open-source software tools to design, build, document, and use RESTful web services. The Inspector tool allows easy testing and validation of API operations direct from the web browser. Each of the available operations was tested for the following:

- invalid response codes,
- invalid response headers,
- API timeouts,
- slow API response (with respect to response data bytes),
- incorrect required data in JSON response.

The found inconsistencies and problems were fixed during the API development.

Additionally, to monitor the API performance, the Flask Monitoring Dashboard [129] was utilized. The Flask Monitoring Dashboard offers several useful functionalities for automatic monitoring and performance evaluation of Flask-based APIs. The dashboard presents the information on the speed of API request processing (seen in Figure 3.20) and on the amount of request being process by each operation. Also, it tracks the execution path of every request, allowing the identification of the functions in the code which tak the most time to execute. The dashboard also automatically detects outlying requests which take much longer to process than regular requests. All of these functionalities were employed to develop the most stable API version possible, which could be used in the data analysis use cases.

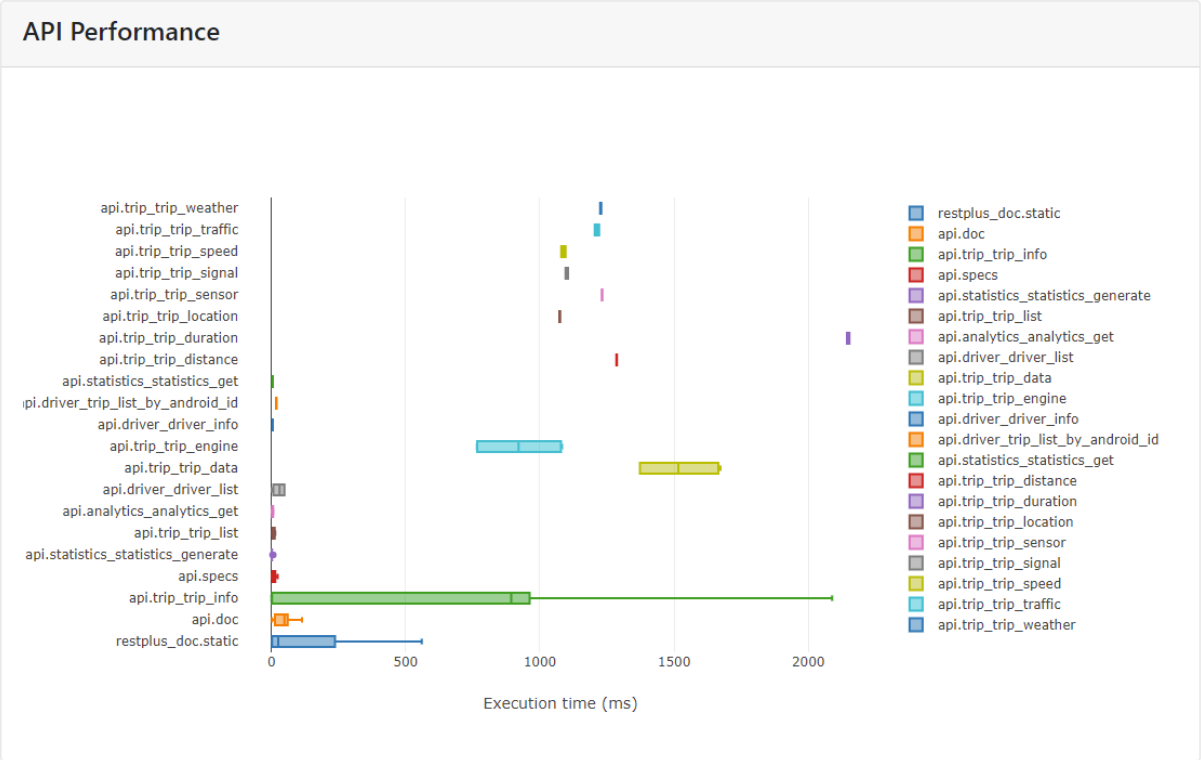

Figure 3.20: The API performance measured in execution time (ms) of API route calls in the Flask Monitoring Dashboard

3.7Framework Data Security and Access Policies

As the framework collects and stores sensitive data about vehicle performance and movement, special emphasis was put on ensuring data security and having proper access policies in place. Data security refers to the process of protecting data from unauthorized access and data corruption throughout its lifecycle. This section describes the security properties and access policies of the communication between the framework and it's users, as well as within the framework between it's modules.

3.7.1OBD-II to Collection Modules

OBD-II collection is done using the ELM327 microcontroller which connects to the collection module on the smartphone via bluetooth. To connect to the ELM327 through the smartphone, the user needs to know the ELM327 PIN number (known only to the owner of the ELM327 device), which provides the first security layer. High-end ELM327 devices use 128-bit encryption which prevents man-in-the-middle attacks and Bluetooth sniffing which solves the problem of unauthorized access to the vehicle data being transmitted to the collection module. Breaking this kind of encryption is only theoretically possible, as even with a supercomputer, it would take 1 billion billion years to crack the 128-bit AES key using a brute force attack. Also, the

OBD-II port does not offer any options to change vehicle settings other than resetting fault codes, so the driver and passengers are never in danger if OBD-II connection is exposed.

3.7.2CAN to Collection Modules

CAN bus data collection is done using the PCAN Wireless Gateway to which the smartphone collection module is connected through a Wi-Fi connection. The CAN bus must be physically connected to the gateway, so without no intrusion can be made without accessing the vehicle interior. The gateway works as any other network router; to connect to it, a password must be entered. It would take more than 2000 years using a high-end machine to brute force a 12 character WPA2 password, so this provides a reasonably good protection against unauthorized wireless access. Even if this protection is to be overriden, to intercept the data, the attacker would still need to access the administrator interface of the gateway which is protected with a username and password combination.

3.7.3Collection Modules to Data Platform

The collection modules send the data to the secured port on the data platform. The communication between the smartphones and the data platform is secured using Transport Layer Security (TLS), a cryptographic protocol designed to provide communications security over a computer network. On the data platform entry point, the Apache Flume agent uses an Avro source type [119] which is a plain-text data serialization mechanism supporting TLS for secure communication. To be able to send data to the data platform, the correct credentials must be set on the collection modules, preventing unwanted data ingestion on the data platform. TLS also prevents unauthorized access to the sent data by any kind of packet sniffers.

3.7.4Data Platform to Application Programming Interface

The application programming interface communicates directly with the data platform's database. To connect to the MongoDB database, authentication is required. The communication between the API and the database is encrypted and can only been established once the API is properly authenticated. The database is located in a private network which provides an extra layer of security, as any kind of attacker would first need to gain access to the private network to get a chance to tamper with the database or execute packet sniffing.

3.7.5Application Programming Interface to Consumers

The final communication segment in the framework is between the API and its consumers. To access the API's operations, consumers need to have valid credentials for the operation

they are trying to access and they need to authenticate with the API using those credentials. The authentication mechanism used by the API is Basic Auth which is a widely used protocol for simple username/password authentication. Each API consumers is provided with its own credentials, and additional credentials can be provisioned by the API's administrator.

Chapter 4

Contextually Enriched Automotive Data Set

Once the framework was established and all the modules in it were implemented and validated, it could be used to acquire a contextually enriched automotive data set. To provide a better understanding of the data set which is to be collected, the data model will first be examined in detail in Section 4.1. Afterwards, the data collection experiment was devised and performed in order to collect the data set. This experiment and its execution will be described in Section 4.2. Finally, in Section 4.4, data set descriptive statistics will be demonstrated, along with some visualisation examples which serve as an introduction into the possible use cases of such a data set.

4.1Data Model

The data model consists of 11 data entities which are filled out with the information collected by the framework. Figure 4.1 shows how these entities are collected via the framework. *AutomotiveData* is collected from the vehicles OBD port or CAN bus using Bluetooth or Wi-Fi gateways. The collection and contextual enrichment modules collect *Driver* and *Vehicle* data through user input. *MobileDevice* information is collected from the smartphone performing the collection. A Trip is formed when data collection is started and it consists of multiple *DataPoint* items which contain *LocationData*, *SignalData* and *SensorData* information collected from the smartphone, and *TrafficData* and *WeatherData* information collected from online APIs.

The entire model of the collected data stored in the data storage platform is displayed as a UML diagram in Figure 4.2. As previously mentioned, before initializing the data collection, drivers have to input their personal and vehicle information on the smartphone application. These values are stored in the *Driver* and *Vehicle* collections in the database. The *Driver* collection contains the following information (information type is disclosed in the brackets next to

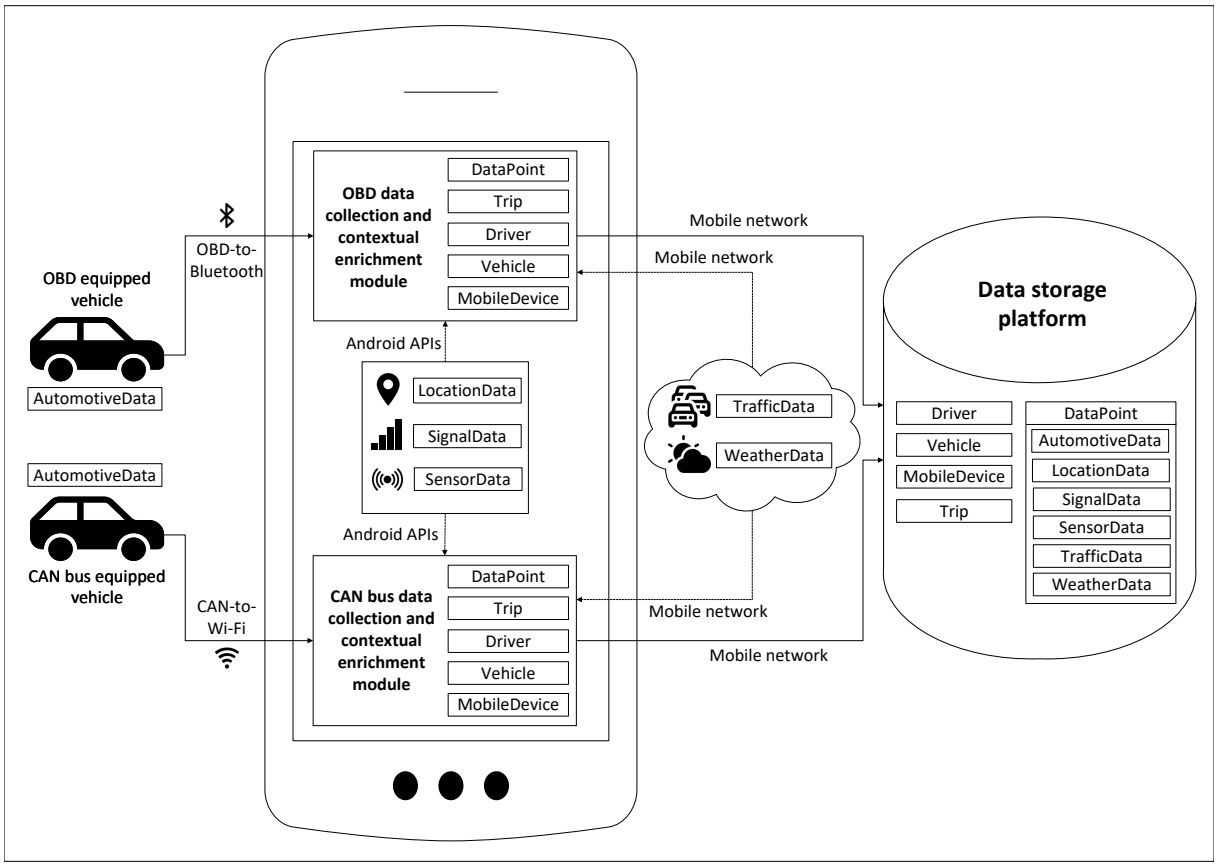

Figure 4.1: Data entities contained in the data model overlaid on the framework

each entry):

- *id* - the identifier of the driver (string);
- *gender* - the driver's gender, possible values are "male", "female" and "other" (string);
- *age* - the driver's age (integer);
- *drivingExperience* - years of driving experience of the driver (integer);
- *vehicleId* - the identifier of the driver's vehicle (string);
- *mobileDeviceId* - the identifier of the driver's mobile device (string).

The *Vehicle* collection is comprised of the following information:

- *id* - the identifier of the vehicle (string);
- *manufacturer* - the manufacturer of the vehicle, e.g. "Volkswagen" (string);
- *model* - the model of the vehicle, e.g. "Touran" (string);
- *yearOfProduction* - the year of the vehicle production (integer);
- *fuelType* - the type of the fuel which the vehicle consumes, possible values are "diesel", "petrol (benzine)", "LPG", "hybrid (petrol)", "hybrid (diesel)" and "electric" (string).

MobileDevice data is also collected, mainly for debugging purposes, containing information about the smartphone device on which the data collection app is running. This includes information about:

- *id* - the identifier of the mobile device (string);

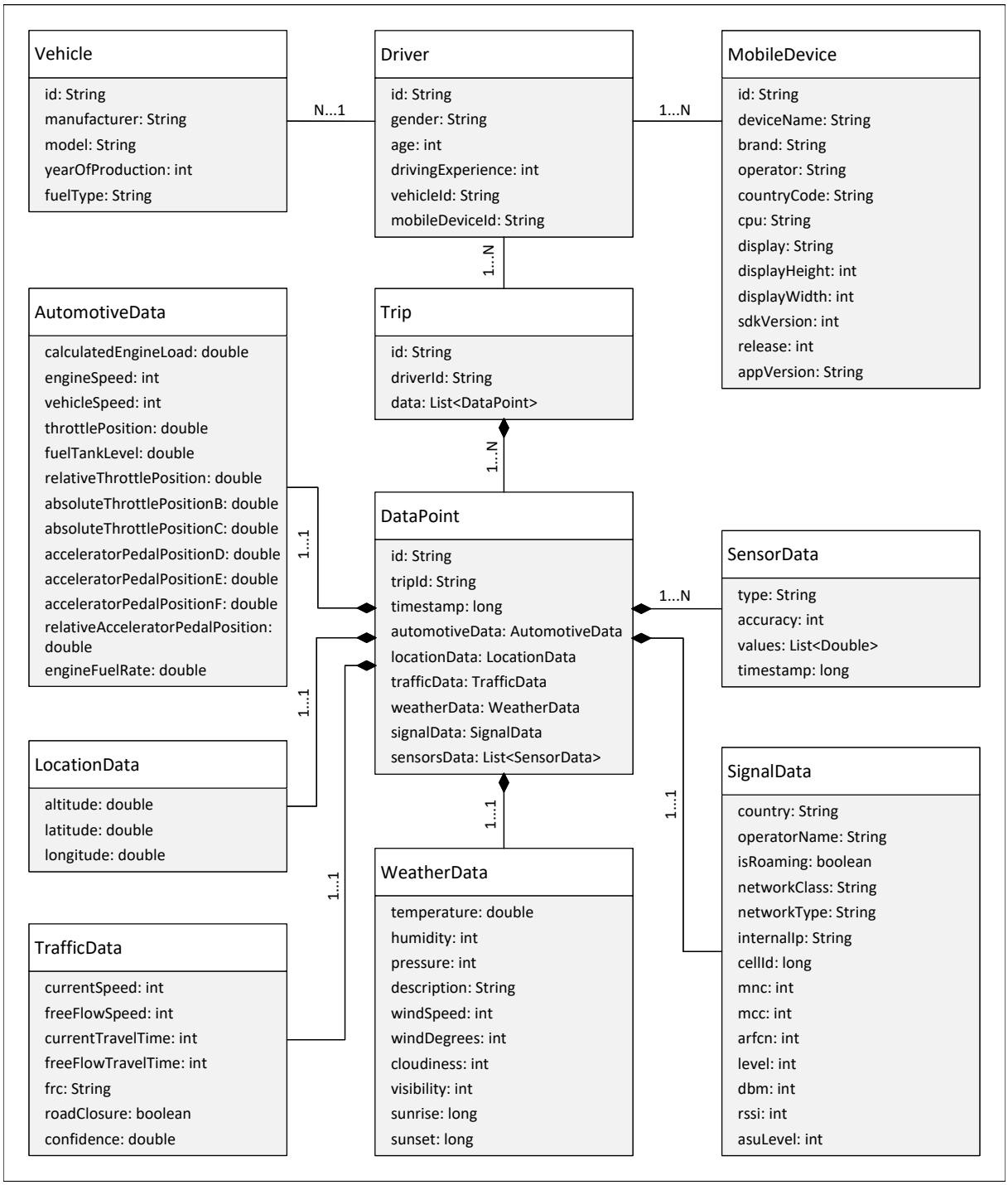

Figure 4.2: Contextually enriched automotive data model

- *deviceName* - the name of the device on which the application is running, e.g. "Samsung SM-G996B" (string);
- *brand* - the device's brand, e.g. "Samsung" (string);
- *operator* - the operator of the device's mobile network connection, e.g. "Telemach" (string);
- *countryCode* - the country code of the operator, e.g. "HR" (string);

- *cpu* - the name of the instruction set (CPU type + ABI convention) the device uses, e.g. "arm64-v8a" (string);
- *display* - the model of the display used by the device (string);
- *displayHeight* - the height of the display in pixels (integer);
- *displayWidth* - the width of the display in pixels (integer);
- *sdkVersion* - the version of the SDK which the device is running (integer);
- *release* - the release version of the operating system the device is running (integer);
- *appVersion* - the version of the collection and contextual enrichment smartphone application (string).

Once the driver initializes the data collection, a *Trip* entry is formed which encompasses all the data collected until the driver stops the data collection:

- *id* - the identifier of the trip (string);
- *driverId* - the identifier of the driver which performed the trip (string);
- *data* - the data points recorded during the trip (list of *DataPoint* items).

A single *DataPoint* describes the vehicle and contextual information at a point in time, and is composed of the following information:

- *id* - the identifier of the data point (string);
- *tripId* - the identifier of the trip to which the data point belongs (string);
- *timestamp* - the timestamp at which the data point was recorded (long);
- *automotiveData* - automotive information, collected via OBD-II/CAN (AutomotiveData);
- *locationData* - information describing the current location of the device/vehicle (LocationData);
- *trafficData* - traffic information about the area surrounding the device/vehicle (TrafficData);
- *weatherData* - weather information about the area surrounding the device/vehicle (WeatherData);
- *signalData* - information about the device's mobile signal (SignalData);
- *sensorsData* - information collected from the device's sensors (list of SensorData).

AutomotiveData is the collection in which automotive data is stored. In the case of the data collection experiment, this data that is collected from the vehicles OBD-II port via the ELM327. It contains the information previously described when talking about OBD-II and CAN bus data sources in Section 3.2, more specifically:

- *calculatedEngineLoad* - peak available torque, expressed as a percentage (double);
- *engineSpeed* - RPM, number of full turns the engine does every minute (integer);
- *vehicleSpeed* - actual speed of the vehicle, measured in km/h (integer);
- *throttlePosition* - openness of the throttle valve, expressed as a percentage (double);
- *fuelTankLevel* - the level of fuel in the vehicle's fuel tank, expressed as a percentage

(double);

- *relativeThrottlePosition* - throttle position compared to the learned closed position described by *throttlePosition*, expressed as a percentage (double);
- *absoluteThrottlePositionB* - redundant information used to validate the integrity of the throttlePosition value, expressed as a percentage (double);
- *absoluteThrottlePositionC* - redundant information used to validate the integrity of the throttlePosition value, expressed as a percentage (double);
- *acceleratorPedalPositionD* - position of the accelerator pedal, sensor D, expressed as a percentage (double);
- *acceleratorPedalPositionE* - position of the accelerator pedal, sensor E, expressed as a percentage (double);
- *acceleratorPedalPositionF* - position of the accelerator pedal, sensor F, expressed as a percentage (double);
- *relativeacceleratorPedalPosition* - accelerator pedal position compared to the learned minimal position of the accelerator pedal, expressed as a percentage (double);
- *engineFuelRate* - the fuel consumption rate of the vehicle, expressed in l/h (double).

LocationData is the positional information collected from the smartphones built-in GPS receiver. It contains information about the device's:

- *altitude* - height above sea level, in meters (double);
- *latitude* - geographic coordinate that specifies the north–south position (double);
- *longitude* - geographic coordinate that specifies the east–west position (double).

TrafficData and *WeatherData* are both collected from Internet-based APIs (TomTom API and OpenWeather API), and they contain information about the current traffic and weather conditions around the vehicle's location. *TrafficData* describes:

- *currentSpeed* - current average traffic speed at the device/vehicle location, measured in km/h (integer);
- *freeFlowSpeed* - free flow traffic speed expected under ideal conditions, measured in km/h (integer);
- *currentTravelTime* - the time required to pass through the returned location segment, measured in seconds (integer);
- *freeFlowTravelTime* - the time required to pass through the returned location segment under ideal conditions, measured in seconds (integer);
- *frc* - Functional Road Class (string), possible values are:
  - "FRC0": Motorway, freeway or other major road;
  - "FRC1": Major road, less important than a motorway;
  - "FRC2": Other major road;
  - "FRC3": Secondary road;

  - "FRC4": Local connecting road;
  - "FRC5": Local road of high importance;
  - "FRC6": Local road;
- *roadClosure* - indicates if the road is closed to traffic or not (boolean);
- *confidence* - a measure of the quality of the provided travel time and speed, the value is in range between 0 and 1 where 1 means full confidence, meaning that the response contains the highest quality data.

On the other hand, *WeatherData* describes:

- *temperature* - temperature in Celsius (double);
- *humidity* - humidity in percentage (integer);
- *pressure* - atmospheric pressure in hPa (integer);
- *description* - weather description (string);
- *windSpeed* - wind speed in m/s (integer);
- *windDegrees* - wind direction in degrees (integer);
- *cloudiness* - cloudiness in percentage (integer);
- *visibility* - visibility in meters (integer);
- *sunrise* - time of sunrise (long);
- *sunset* - time of sunset (long).

SignalData describes the smartphone's mobile network information:

- *country* - country code of the mobile network operator (string);
- *operatorName* - the name of the mobile network operator (string);
- *isRoaming* - indicates if the mobile device is in roaming mode (boolean);
- *networkClass* - the class of the network to which the device is connected, possible values are "2G", "3G", "4G" and "5G" (string);
- *networkType* - the type of the network to which the device is connected, e.g. "UMTS", "HSPA", "LTE" (string);
- *internalIp* - IP address of the device (string);
- *cellId* - identifier of the mobile network cell to which the device is connected (long);
- *mnc* - Mobile Network Code, contains 2 or 3 digits (integer);
- *mcc* - Mobile Country Code, contains 3 digits (integer);
- *arfcn* - Absolute Radio Frequency Channel Number, unique number given to the used radio channel (integer);
- *level* - abstract level value for the overall signal quality (integer);
- *dbm* - signal strength as dBm (integer);
- *rssi* - Received Signal Strength Indicator, a measurement of the power present in a received radio signal (integer);
- *asuLevel* - Arbitrary Strength Unit, an integer value proportional to the received signal

strength measured by the mobile phone (integer).

Finally, *SensorData* is used for storing information collected from the smartphones motion and position sensors:

- *type* - type of the sensor, possible values are "accelerometer", "gyroscope", "uncalibrated gyroscope", "magnetic field", "uncalibrated magnetic field", "rotation vector" and "game rotation vector" (string);
- *accuracy* - the accuracy of the sensor reading, possible values are 1, 2 and 3, 1 being the lowest accuracy and 3 being the highest (integer);
- *values* - a list of sensor reading values (list of doubles);
- *timestamp* - timestamp of the sensor reading (long).

4.2Data Collection Experiment

The data was collected during a period of 7 weeks (1 November - 19 December 2020) by 9 drivers of different genders (5 male, 4 female), different age groups (25 - 53) and different years of driving experience (2 - 36), driving 8 different vehicles (2 drivers shared the same car) in northwest Croatia in and around Zagreb. Detailed information about experiment participants can be seen in Table 4.1.

Table 4.1: Data collection experiment participants information

Driver information				**Vehicle information**		
ID	**Gender**	**Age**	**Driving experience**	**Make and model**	**Year of production**	**Fuel type**
A	female	25	2 years	Opel Corsa	2011	Diesel
B	female	26	2 years	Fiat Grande Punto	2007	Diesel
C	male	25	7 years	Alfa Romeo Giulietta	2012	Diesel
D	female	25	7 years	Fiat Punto Evo	2011	Diesel
E	male	26	8 years	Fiat Grande Punto	2007	Diesel
F	male	29	10 years	Škoda Fabia	2005	Gasoline
G	male	31	12 years	Renault Megane 2 Sedan	2006	Diesel
H	female	52	31 years	Citroen C4 Picasso	2015	Diesel
I	male	53	35 years	Volkswagen Touran	2020	Diesel

Every driver was asked to provide information about the vehicle they will be driving during the course of the experiment, such as make and model, year of production and fuel type. All data collection was done over the OBD-II ports using the ELM-327 Bluetooth device. Each time data collection was initiated on the driver's smartphone device, it would instantiate a new trip with

a unique identifier. The drivers were not instructed to drive on any predetermined routes, but to collect the data at any time they please. Most of the trips were recorded during daily commutes. Four of the nine drivers drove almost exclusively in urban areas at lower speeds, while the others were driving on inter-city roads at higher speeds. To comply with the privacy and data security policies, all of the drivers signed a consent form before starting the data collection. Over the course of the experiment, 307,652 data points were collected, amounting to a total exceeding 100 hours of recorded trip data.

4.3Data Cleansing

To produce the final data set, data cleansing was performed in order to remove corrupt or inaccurate records. This process was done in seven steps.

Step 1 checks if *AutomotiveData* is collected properly. Incorrect records can occur due to errors in communication between the smartphone and the wireless gateway device, or errors in communication between the gateway and the underlying bus or diagnostic system. This is indicated in unrealistic values in the data set such as speed being over 300 km/h, RPM over 10,000 or percentage values over 100% or the bus returning "NODATA" or "BUSERROR" value. Such records are removed from the data set.

Step 2 checks for records with invalid *LocationData*. As contextual enrichment from traffic and weather data sources depends on location, if the smartphone location provider returned an inaccurate location, this data point could contain inaccurate traffic or weather data. Therefore, the data set was inspected for location outliers which were then removed.

Step 3 identifies records with missing *TrafficData*. As step 2 already deals with records with invalid *LocationData* which can cause invalid *TrafficData*, this step ensures that there are no records with *TrafficData* values missing.

Step 4 is similar to step 3, as it removes records with missing *WeatherData*. This can happen due to errors in communication with the online weather API service.

Step 5 removes records with invalid *SignalData*. As this information is collected from the smartphone performing the collection, it depends on the smartphone device to return correct mobile signal information. In rare occurrences, this information can be either empty or filled with unrealistic values such as maximal integer value. These records are also removed from the data set.

Step 6 is similar to step 5, as it identifies invalid *SensorData* records, which is also collected from the smartphone device. Records with empty or unrealistic SensorData values are removed.

Step 7 is the final data cleansing step, and it removes *Trip* records and their corresponding *DataPoint* records. There are several reasons to remove an entire trip from the data set. Sometimes due to poor network conditions, or errors in communication between the gateway and the

smartphone, the sampling frequency of data points is too low. These trips cannot be salvaged as they don't contain the expected amount of information. Another reason is due to human error. The drivers performing the collection would sometimes stop the collection straight after starting it, which would result in a trip containing only several data points. These records were also removed

Table 4.2 provides a summary of the data cleansing process, showing results of each data cleansing step. Most records were removed in step 7, as trip deletion removes the highest amount of records from the data set. Invalid *LocationData* values are cause for the second most number of deletions. The final cleaned data set consists of 287,882 data points collected during 212 trips made by 9 drivers. The final data set is 93.57% of the size of the originally collected data set.

Table 4.2: Data cleansing results

Step	Cleansed data entity	Records before (No.)	Records after (No.)	Records removed (No.)	Records left (%)
1	AutomotiveData	307,652	305,330	2,322	99.25
2	LocationData	305,330	301,659	3,671	98.05
3	TrafficData	301,659	300,014	1,645	97.52
4	WeatherData	300,014	299,537	477	97.36
5	SignalData	299,537	298,932	605	97.17
6	SensorData	298,932	298,421	511	97.00
7	Trip	298,421	287,882	10,539	93.57

4.4Data Set Statistics and Visualization

After the data set was collected and cleaned, statistical information was calculated to provide more context to the collected data. As already mentioned, there is a total of 212 trips recorded in the data set containing 94 hours, 25 minutes and 44 seconds of contextually enriched automotive data. This results in an average trip duration of 26 minutes and 44 seconds. The total trip distance of the recorded data is 5,208.71 km, with an average trip distance being 24.57 km. In Table 4.3, cumulative statistics per driver are presented, along with the previously mentioned summarized values.

When observing the driver statistics, several things can be noticed to which attention must be paid when performing data analysis. It can be noticed that some drivers contribute to the total recorded data more significantly than others. This indicates that special care must be taken not to allow certain drivers to over-influence the results of any kind of analytical procedure that

Table 4.3: Cumulative statistics of the collected data set grouped by driver

Driver ID	Number of trips	Total trip duration	Avg. trip duration	Total trip distance (km)	Avg. trip distance (km)	Avg. speed (km/h)	Avg. traffic speed (km/h)
A	27	14:03:01	00:31:13	982.94	36.41	69.96	74.22
B	37	11:39:21	00:18:54	402.23	10.87	34.51	42.89
C	8	03:48:03	00:28:30	246.52	30.82	64.86	67.22
D	17	06:24:58	00:22:39	356.80	20.99	55.61	63.53
E	15	09:08:59	00:36:36	578.35	38.56	63.21	66.09
F	13	04:41:52	00:21:41	101.64	7.82	21.64	30.78
G	34	08:27:32	00:14:56	274.80	8.08	32.49	41.64
H	27	03:58:56	00:08:51	124.71	4.62	31.32	48.25
I	34	32:13:02	00:56:51	2,140.73	62.96	66.45	69.24
Total	**212**	**94:25:44**	**00:26:44**	**5,208.71**	**24.57**	**55.16**	**61.47**

is performed over the data. Another thing that can be noticed is that the participating drivers can be classified into two groups according to their average speed. The first group consists of the drivers with an average recorded speed lower than 35 km/h and average traffic speed lower than 50 km/h, while the second group of drivers have an average speed higher than 55 km/h and average traffic speed higher than 60 km/h. This corresponds to the type of the road the drivers were mostly driving on. As mentioned earlier, four drivers with the lower average speed mostly drove in urban areas, while the five drivers with the higher average speed mostly drove on faster roads connecting multiple different urban areas.

Another useful way of observing the data is data visualization. Data trends, patterns and outliers can be understood and noticed more easily by graphically representing the data. To demonstrate this, a single trip was chosen from the collected contextually enriched automotive data set and visualized using Plotly, a graphing, analytics and statistics tool. The trip which was chosen for visualization was driven by driver E (male, 26, 8 years of driving experience, driving a Fiat Grande Punto 2007, diesel), on the 4th of December 2020, starting at 21:05 CET. The trip started in Zagreb, the capital city of Croatia and it's biggest urban area, and ended in Oroslavje, a small city (population 7000) 40 km north of Zagreb, as can be seen in Figure 4.3. The trip lasted for 34 minutes and 42 seconds, and in that time 40.83 km of distance was driven. During the trip, 1646 contextually enriched data points were collected. Average speed of the trip calculated from the collected automotive data is 67.71 km/h.

Figure 4.4 shows the change in vehicle and surrounding traffic and free flow speed through time. Several things can be noticed when visually inspecting this graph. It can be observed

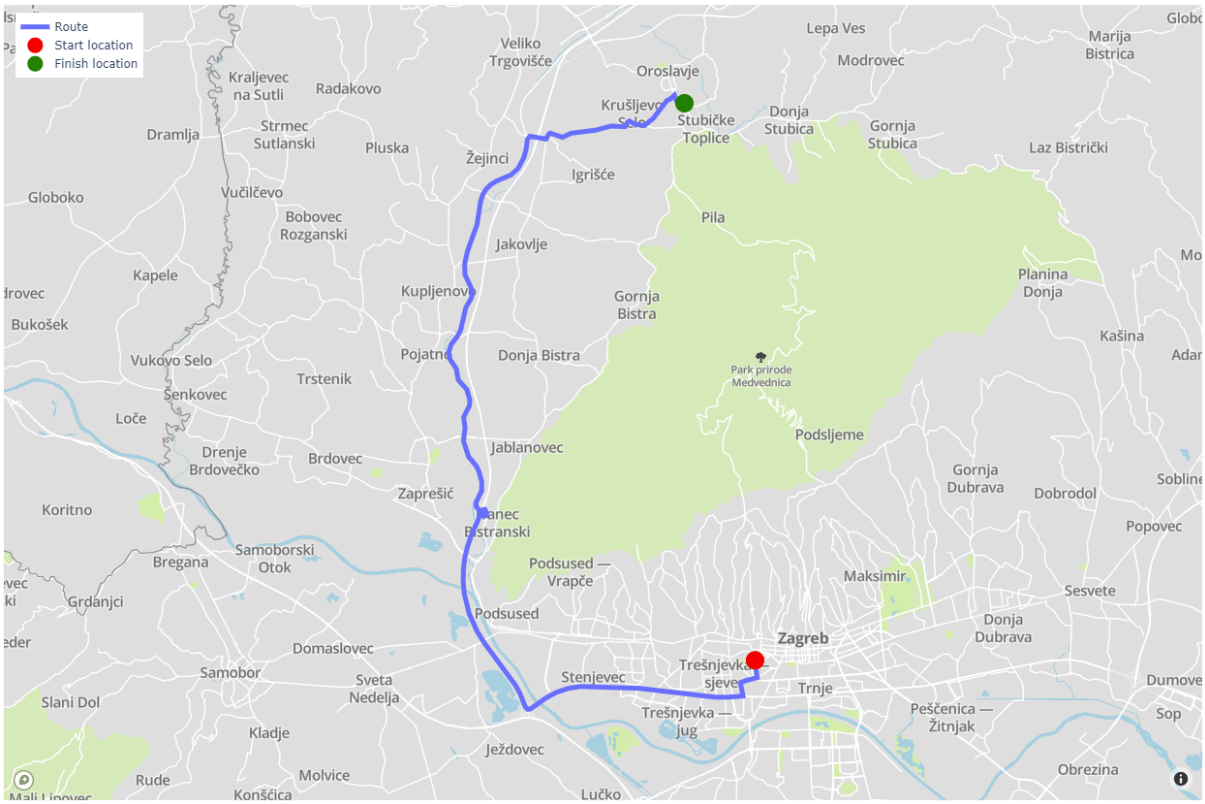

Figure 4.3: A single trip's route from start (red circle) to finish (green circle) location

that vehicle speed follows the traffic speeds pretty closely, with a calculated Pearson correlation coefficient of 0.81. This confirms that the collected traffic data is sensical, providing the driver abided by the traffic laws. Another thing that can be seen is that apart from several minutes of the trip, the traffic speed matches free flow speed. This indicates that there were almost no traffic congestions during the entirety of the trip, which is not surprising when taking into account the time of the trip which was in the evening when no daily commutes are happening. It is also noticeable from that the trip started and ended in urban areas which are connected with highways/fast roads, as in the beginning and ending of the trip there are lower speeds with relatively frequent stops, while the middle of the trip contains speeds averaging at 90 km/h and higher.

In Figure 4.5, the data describing revolutions per minute of the engine (RPM), engine load

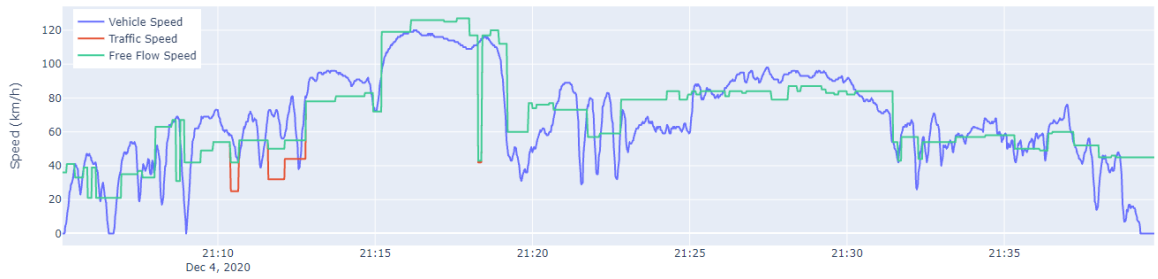

Figure 4.4: A single trip's vehicle, traffic and free flow speed through time

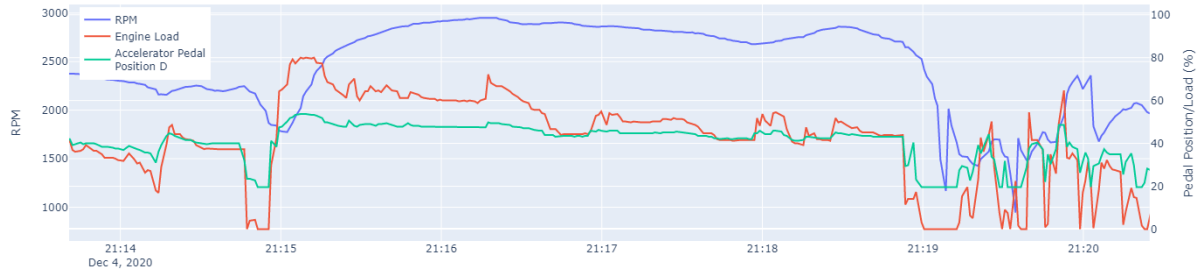

Figure 4.5: A single trip's RPM, engine load and accelerator pedal positions through time

and accelerator pedal position through time is displayed. On this figure, only around 6 minutes of the trip are displayed to make it easier to distinguish the variables from one another, as they change more frequently then the variables shown on the Figure 4.4. This excerpt shows the values on variables during the section of the trip when the vehicle speed was highest. When observing the shape of the RPM values, it can be noticed that it follows the shape of the vehicle speed values seen in Figure 4.4. The calculated Pearson correlation coefficient of these two variables is 0.85. It can also be noticed that the engine load and the accelerator pedal position follow a similar pattern, which indicates that there could be correlation between them. This is confirmed with their Pearson correlation coefficient which amounts to 0.77. This can be explained by the fact that the harder the accelerator pedal is pressed, the more power must be produced by the engine to accelerate the vehicle. There is normally a lot of oscillations in these two signals which is seen on the final minute displayed on the graph, as accelerator pedal is usually not pressed constantly, but it changes the position frequently due to gear shifting, stopping, slowing down and accelerating.

The final example of contextually enriched automotive data visualization can be seen in Figure 4.6 which describes the trip's mobile signal information including ASU level, dBm and network class. The network class is displayed as the background color; in this trip the mobile device was connected most of the time to a 4G network which is indicated by the green background color, while in the final 5 minutes of the trip the device was connected to a 3G network indicated by the yellow background color. As previously mentioned, ASU level is pro-

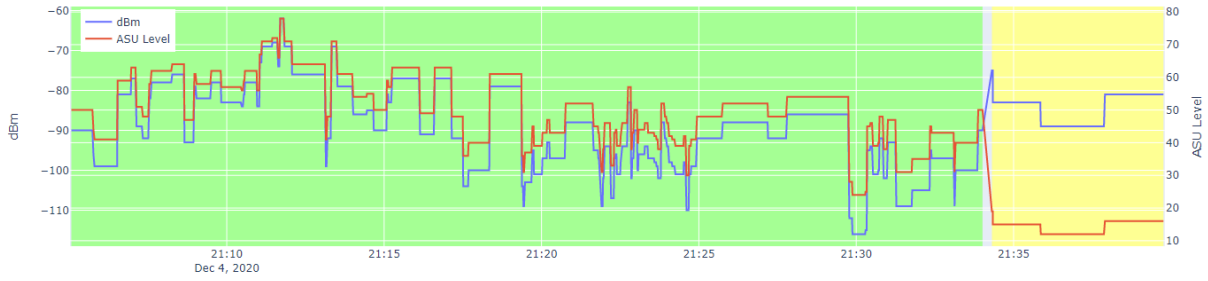

Figure 4.6: A single trip's mobile signal information including ASU level, dBm and network class (green background is for 4G, yellow for 3G) through time

portional to the received signal strength measured by the mobile phone while dBm represents signal strength in dBm. They follow the same pattern during the entire trip, but the difference between these values is different in 4G and 3G sections of the trip as the ASU level is lower in 3G section and dBM is higher in 3G section. This implies that the ASU level is a more accurate representation of the quality of the Internet connection, as signal strength can be perceived as higher in 3G than in 4G networks, even though the connection is usually faster in 4G.

The displayed visualization examples prove that a mere graphical representation of the data can reveal insights into the collected data set. Using nothing but the graph in Figure 4.4, the types of road the driver was driving on and what was the traffic like during the ride could be discerned. Figures 4.5 and 4.6 have shown some correlation between different parameters and conclusions could be made on why they are happening. However, even more insights can be generated using data analytics procedures powered by artificial intelligence techniques which will be demonstrated further in the next chapter.

Chapter 5

Advanced Analytics of the Contextually Enriched Automotive Data Set

After the data collection experiment was executed and the contextually enriched data set was collected, analysis was performed on the collected data. This chapter demonstrates several examples of analytical procedures which can be made using such a data set. In Section 5.1, a process of deriving and calculating a metric for ranking the drivers according to their eco-efficient driving patterns is described. Section 5.2, demonstrates the use of an unsupervised learning algorithm (k-means clustering) in order to identify driving patterns on a trip level. Finally, Section 5.3 depicts a system for interactive visual driving route comparison based on contextual information.

5.1Eco-Efficient Driving Pattern Evaluation

The aim of the data analysis described in this section is to show that a contextually enriched automotive data set can be useful in research pertaining to sustainability, more specifically, fuel efficiency. Efficient driving is part of the solution to reducing greenhouse gas emissions from surface transportation, but it is a highly complex task that involves hundreds of separate tasks [130]. Studies usually simplify the eco-efficient way to drive with simple advice easily understood by drivers. A similar approach was followed here by proposing a metric which can be used to evaluate eco-efficient driving patterns of a group of drivers. This kind of metric can be utilized in several different ways. For example, vehicle fleet managers can use it to identify drivers with low fuel efficiency. It can also be used to provide real-time feedback to drivers who can change their driving style accordingly to optimize fuel efficiency.

The proposed metric consists of four eco-efficient driving advice. The first advice is to *accelerate moderately* - high acceleration consumes more fuel [131]. The second advice is similar - *decelerate smoothly*. A driver should slow down gently as this reduces exhaust emissions

and increases traffic safety and flow [132]. The third advice is to *drive with low engine RPM*. Fuel usage is higher while driving with high RPM, which is why drivers should strive to drive with RPM below 2,500 for gasoline cars and below 2,000 for diesel cars [133]. The fourth and final advice is to *drive at or below surrounding traffic speed*. Driving at speeds higher than the surrounding traffic flow speed is considered aggressive driving and aggressive driving causes higher fuel consumption [134].

To analyse the drivers according to these four eco-efficient driving advice, the advice need to be quantified into mathematical measurements. All four advice are displayed as percentages of driving events in the driver's entire collected data which contradict the advice. For *accelerate moderately* and *decelerate smoothly* advice, all acceleration/deceleration values between two collected data points for all drivers were calculated using vehicle speed and timestamp difference, and these values were used to calculate the average acceleration and deceleration for each driver. As all drivers are not equally represented in the data set, to calculate the final average acceleration and deceleration, the average of driver's average values were calculated. The average acceleration value calculated in this manner equals 0.6827 m/s^2, and the average deceleration value equals -0.6492 m/s^2. To put these values into perspective, a vehicle accelerating with an acceleration of 0.6827 m/s^2 would take 40.69 seconds to go from 0 to 100 km/h or 42.79 seconds to go from 100 to 0 km/h if decelerating with a deceleration of -0.6492 m/s^2. These values are in line with values identified by [132] in their research regarding the effects of eco-driving training on driving behavior, and to the values identified by [131] who explored the effect of acceleration and deceleration on vehicle fuel economy. The previously mentioned calculated average acceleration and deceleration values are therefore used as a reference for baseline acceleration and deceleration for the set of drivers participating in the data collection experiment. After the acceleration and deceleration averages were calculated, all acceleration/deceleration values were compared with the average for all drivers and the percentage of acceleration/deceleration values higher/lower than average was calculated to serve as a mathematical measurement of compliance to the eco-efficient driving advice. For the *drive with low engine RPM* advice, the percentage of engine RPM values higher than 2,500 for gasoline cars and 2,000 for diesel cars was calculated for each driver. Not to discriminate drivers who spent more time driving on higher speeds on faster roads where RPM values are naturally higher than the proposed eco-efficient driving values, only the data points collected while driving on speeds below 80 km/h were taken into account, as this is the highest possible allowed speed for driving within inhabited places in Croatia [135], where the experiment took place. And finally, for the *Drive at or below surrounding traffic speed* advice, the percentage of vehicle speed values higher than the surrounding traffic speed was calculated. To measure this, the actual vehicle speed values and the surrounding traffic speed collected from the TomTom API in the contextual enrichment module were used. The summary of all eco-efficient driving advice and their

measurements can be seen in Table 5.1.

Table 5.1: Eco-efficient driving advice and measurements

Eco-efficient driving advice	Measurement	Data set values used for calculation	Abbreviation
Accelerate moderately	Percentage of acceleration values higher than average acceleration	vehicle speed, average acceleration (calculated over data set)	M_{acc}
Decelerate smoothly	Percentage of deceleration values lower than average deceleration	vehicle speed, average deceleration (calculated over data set)	M_{dec}
Drive with low engine RPM	Percentage of engine RPM values higher than 2,500 for gasoline cars and 2,000 for diesel cars during vehicle speeds below 80 km/h	engine RPM, vehicle speed	M_{rpm}
Drive at or below surrounding traffic speed	Percentage of vehicle speed values higher than the surrounding traffic speed	vehicle speed, current traffic speed	M_{speed}

Once the measurements representing the advice were established, they were calculated for all of the drivers. Additionally, a metric which incorporates the four measurements in a single value named *eco index* was established. As the measurements represent the percentage of driving events negatively impacting the vehicle's fuel economy, the eco index (I_{eco}) is calculated as the sum of the established measurements subtracted from the total number of measurements (Equation 5.1). Detailed information regarding the variables M_{acc}, M_{dec}, M_{rpm} and M_{speed} can be found in Table 5.1.

$$I_{eco} = |M| - \sum_{i=1}^{|M|} M = 4 - (M_{acc} + M_{dec} + M_{rpm} + M_{speed}) \tag{5.1}$$

In other words, eco index amounts to the combined percentages of time that each eco-efficient driving advice is adhered by the driver. The value of the eco index is a number between 0 and 4 (0 being the driver who never adheres to the advice and 4 being the driver who always adheres to the advice). Eco index was calculated for each driver and drivers were ranked from best to worst so a comparison could be made on their eco-efficient driving performance. The calculated results of the measurements and eco index for each driver sorted by their eco index values can be seen in Table 5.2.

To provide additional context to the results, the average speed of the drivers is displayed along with the results as it is relevant to show whether the driver was mostly driving in urban

Table 5.2: Eco-efficient driving analysis results

Driver ID	Average speed (km/h)	M_{acc} (%)	M_{dec} (%)	M_{rpm} (%)	M_{speed} (%)	I_{eco}
D	55.61	20.44	23.43	5.32	29.32	3.2149
H	31.32	27.74	29.10	7.70	33.79	3.0167
I	66.45	22.33	21.79	4.71	51.69	2.9948
A	69.96	23.80	21.35	11.91	43.75	2.9919
E	63.21	24.83	27.30	19.45	38.51	2.8991
C	64.86	33.36	28.51	13.68	45.34	2.7911
G	32.49	36.39	32.07	10.94	45.88	2.7472
B	34.51	32.90	33.46	31.09	38.56	2.6399
F	21.64	43.58	41.88	6.49	58.54	2.4951

areas (lower average speeds) or inter-city roads (higher average speeds). As can be expected, the drivers with the lowest eco index values were drivers who were mostly driving in urban areas (drivers F, B and G), as this kind of driving implies frequent starting and stopping, which is why these drivers have higher M_{acc} and M_{dec} values. Driver H is an exception to this, as this driver has one of the best eco index scores, but was driving in urban areas. High eco index while driving in urban areas is a characteristic of defensive drivers as they tend to maintain a safe distance and drive slower, both of which are beneficial to the final score. Drivers who were driving on inter-city roads with higher average speeds finished with a higher eco index score. This can be attributed to maintaining a more steady speed on faster roads, meaning that their M_{acc} and M_{dec} is lower. Drivers with lower M_{rpm} performed better in the final eco index score. These values are closely connected to the M_{acc} and M_{dec} values, as the drivers who drive on lower RPM tend to accelerate and decelerate more moderately. The M_{speed} measurement has no apparent connection to the final eco index score, but it can potentially show which driver tends to drive more aggressively, trying to beat the traffic flow.

When analysing the driver information (Table 4.1) along with the eco index results (Table 5.2), several things can be noticed. First of all, gender seems to have no connection to the eco-efficient driving patterns of a driver. Another interesting thing to note is that two of the three best-performing drivers were the ones with the most driving experience (more than 30 years). Finally, it can be seen that the three drivers with the worst eco index results drove the oldest vehicles. This shows us that the vehicle's year of production potentially plays a part in some of the calculated measurements.

5.2Trip-Based Feature Evaluation

The next step after measuring the eco-efficiency per driver was to identify the driving patterns on the level of a single trip. To do this, several features of interest were identified in the contextually enriched data which were then used for establishing a measurement. The list of these features can be seen in Table 5.3. Each feature belongs to a certain context and contains several different values which describe a trip instead of a set of the trip's data points. The values in question which were calculated for each feature are average, median and standard deviation.

Once the features were identified and their values calculated, clustering could be used to

Table 5.3: Trip features used for clustering

Context	Feature	Description	Abbreviation
Time	Time of day	Morning (06:00-11:59) Afternoon (12:00-17:59) Evening (18:00-21:59) Nighttime (22:00-05:59)	T_{day}
	Day in the week	Weekend (Sunday/Saturday/Holiday) Workday (Monday-Friday)	T_{week}
	Duration	Trip duration in seconds	T_{dur}
Automotive	Distance	Total distance in kilometers traversed during the trip	A_{dist}
	Vehicle speed	Vehicle speed in km/h	A_{speed}
	Engine RPM	Engine revolutions per minute	A_{rpm}
	Acceleration	Acceleration between two data points in m/s^2	A_{acc}
	Deceleration	Deceleration between two data points in m/s^2	A_{dec}
Traffic	Vehicle speed higher than traffic speed	The difference between the current vehicle speed and the traffic speed when vehicle speed is higher than the traffic speed (in km/h)	TR_{high}
	Vehicle speed lower than traffic speed	The difference between the traffic speed and the current vehicle speed when vehicle speed is lower than the traffic speed and non-zero (in km/h)	TR_{low}
Road	Vehicle speed higher than free flow speed	The difference between the current vehicle speed and the free flow speed when vehicle speed is higher than the free flow speed (in km/h)	R_{high}
	Vehicle speed lower than free flow speed	The difference between the free flow speed and the current vehicle speed when vehicle speed is lower than the free flow speed and non-zero (in km/h)	R_{low}

identify patterns in the trips and categorise them according to the driving style. Clustering is a machine learning method using unsupervised learning to group data into clusters according to their similarity. The clustering method chosen for the advanced analysis is the K-means partitional clustering algorithm, one of the most popular, fastest and widely used unsupervised learning algorithms [136]. K-means divides the data into disjoint subsets, with each data point belonging to a specific set. It uses Euclidean distance to determine which cluster a particular point belongs to. Euclidean distance is a measure of the distance between a pair of points in n-dimensional space. All identified features were used as input for the clustering process, except T_{day} and T_{week}, as they are categorical variables with a fixed number of possible values, and k-means algorithm is not applicable to categorical data. Before execution, the algorithm requires a number of clusters (k), which represent the number of categories into which the data should be classified. A commonly used method for determining an appropriate number of clusters is the elbow method. This method consists of plotting the explained variation as a function of the number of clusters, and picking the elbow of the curve as the number of clusters to use. In this case, using the elbow method, the determined number of clusters is three as seen in Figure 5.1.

All of the 212 trips contained in the contextually enriched automotive data set were clustered into 3 groups of driving style. The clustering algorithm has divided the trips into the groups as follows:

•Group 1 - 96 trips, 45.3%

•Group 2 - 79 trips, 37.3%

•Group 3 - 37 trips, 17.4%

Fig. 5.2 shows a polar diagram in which the identified 3 groups are shown in different

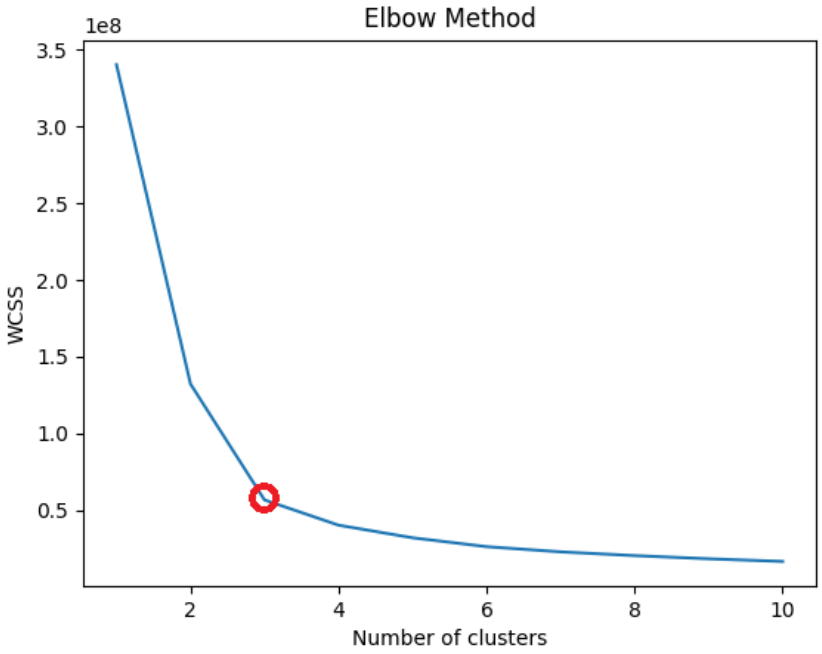

Figure 5.1: The optimal number of clusters determined using elbow method for k-means clustering

colours - Group 1 values are depicted with a blue line, Group 2 with a red line and Group 3 with a green line. Each branch on the diagram is depicted as a white line running outward from the centre of the circle and represents an average value of a feature from Table 5.3. The values on the branches are the mean values of the features normalised by z-score normalisation. These branches are also grouped according to the context to which they belong.

Several conclusions can be made when interpreting the polar diagram according to the four defined contexts. From the values T_{dur}, A_{dist}, A_{speed} and A_{rpm}, it can be noticed that Group 1 consists of trips which have on average the longest durations and distances, with high speed and RPM, which is a feature indicative of inter-city driving on fast roads or highways. The low changes in acceleration and deceleration further prove this point. The road and traffic context values of Group 1 are around average for the observed set of trips (zero value on the polar

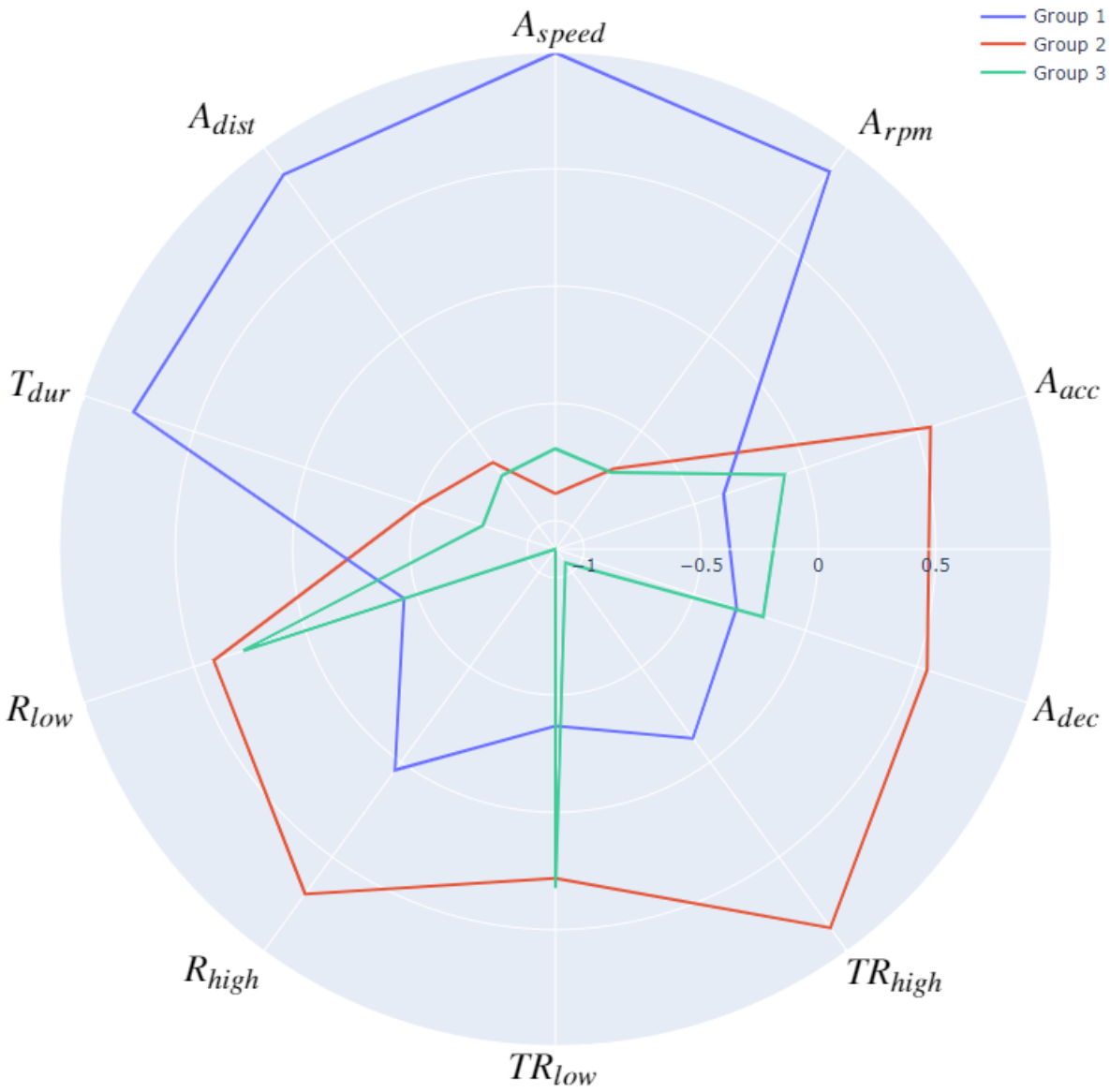

Figure 5.2: Normalised mean values of trip features (Table 5.3) by groups of driving style resulting from k-means clustering

diagram), which shows that on fast roads and highways, drivers usually drive around the speed limit. Group 2 has high acceleration and deceleration values with low speed and RPM over shorter distances and durations. This points to the fact that the trips in this group are driven in urban areas with a lot of starting and stopping. Group 2 also has the highest traffic and road context values, which signifies aggressive driving style. Group 3 is similar to Group 2 in their low speeds, RPM values, distances and duration, but it has a much lower values describing vehicle speed higher than traffic and road speed, which indicates defensive driving in urban areas.

To validate the set hypothesis concerning the trips which belong to each group defined by the clustering process, additional analysis can be made. When taking the automotive context features into consideration, the analysis can be divided into two parts: the speed to RPM correlation and the acceleration to deceleration correlation. Figure 5.3 shows the position of each trip as a function of average values of speed and RPM. The trip colors indicate to which cluster a certain trip belongs. This graph shows the actual values of trip features, when compared to the polar diagram which displays z-score normalized averages of each feature per group.

The three clusters are clearly visible on this graph. Once again, as noticeable from the polar graph, the trips belonging to Group 1 have the highest engine RPM and speed. Groups 2 and 3 are similarly positioned in the graph, as they both contain trips which have lower engine RPM and speed. The difference is that in Group 3 the average speed values trips are slightly higher and RPM values are slightly lower than those from Group 2.

Figure 5.4 displays the distribution of trips according to the average values of acceleration and deceleration. The first thing that can be noticed here is that the majority of the trips have acceleration and deceleration values below 1 m/s^2. The biggest outliers belong to Group 2,

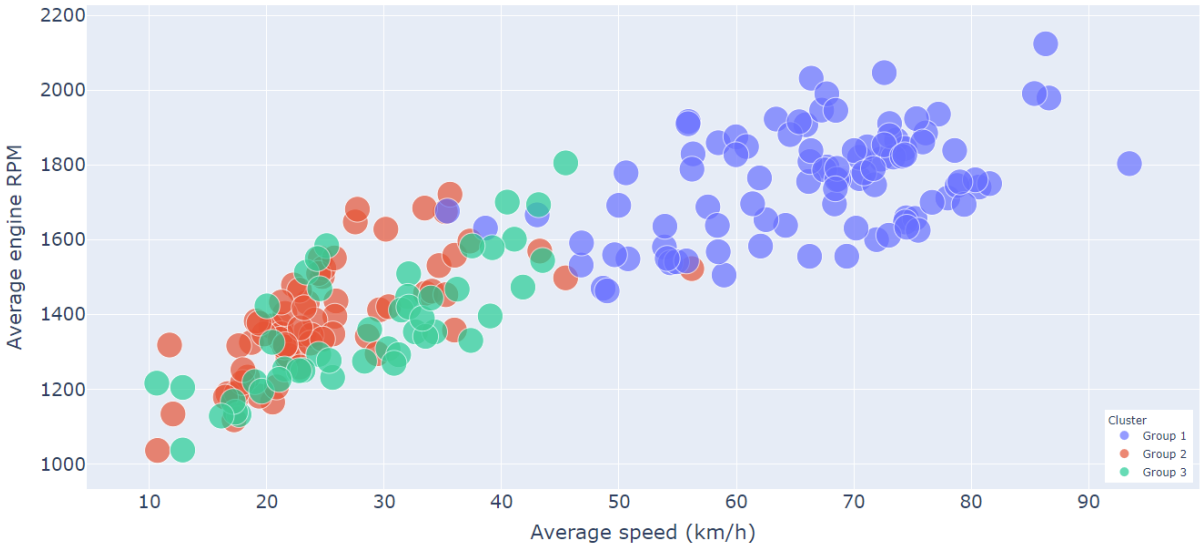

Figure 5.3: Average vehicle speed and engine RPM for each trip in the data set (different groups are colored by different colors)

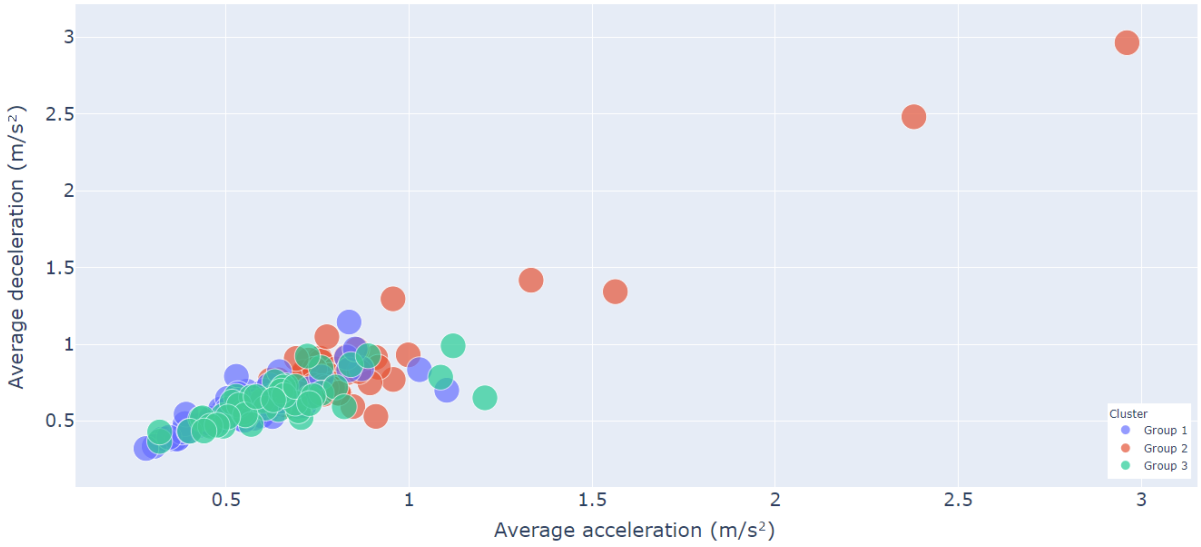

Figure 5.4: Average acceleration and deceleration for each trip in the data set (different groups are colored by different colors)

with some of the trips having both acceleration and deceleration values above 2.5 m/s^2. These trips presumably skew the total average values of Group 2, but it is worth to note that the trips belonging to this group also fill out the high point of the most populated area of the graph (below 1 m/s^2). Group 1 and Group 3 have similar layouts of trip values on the graph, with Group 1 populating containing trips with the lowest acceleration and deceleration values.

The other two contexts which can be additionally observed are related to traffic and roads. Figure 5.5 shows traffic context values of TR_{low} (x-axis) and TR_{high} (y-axis) for each trip, while road context values R_{low} (x-axis) and R_{high} (y-axis) are shown in Figure 5.6. The two graphs are similar, since both position the trip characteristics with respect to the external traffic flow,

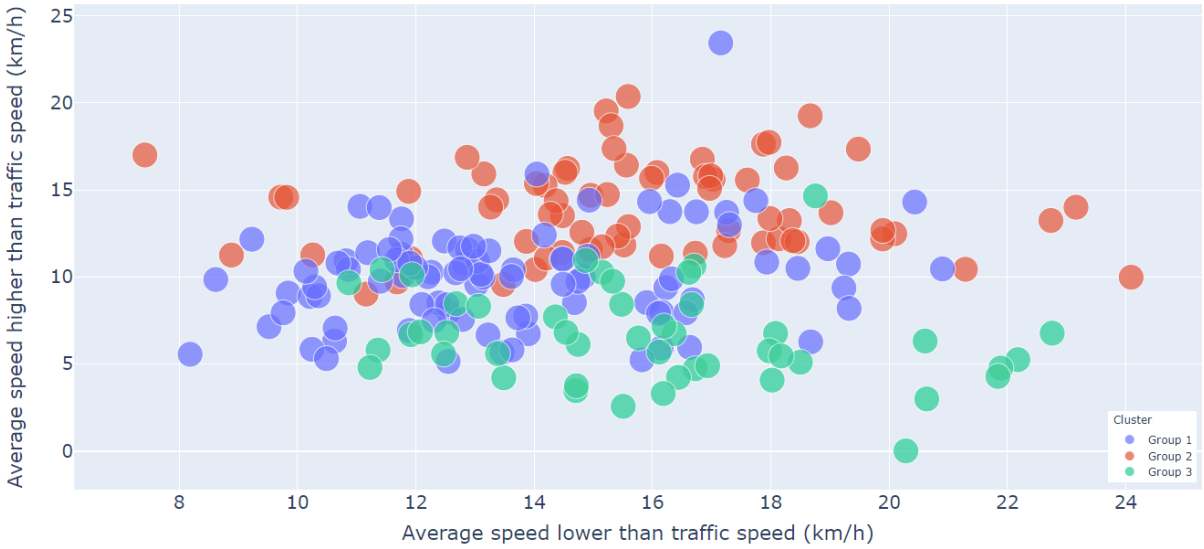

Figure 5.5: Average difference of vehicle speed and traffic speed for each trip in the data set (different groups are colored by different colors)

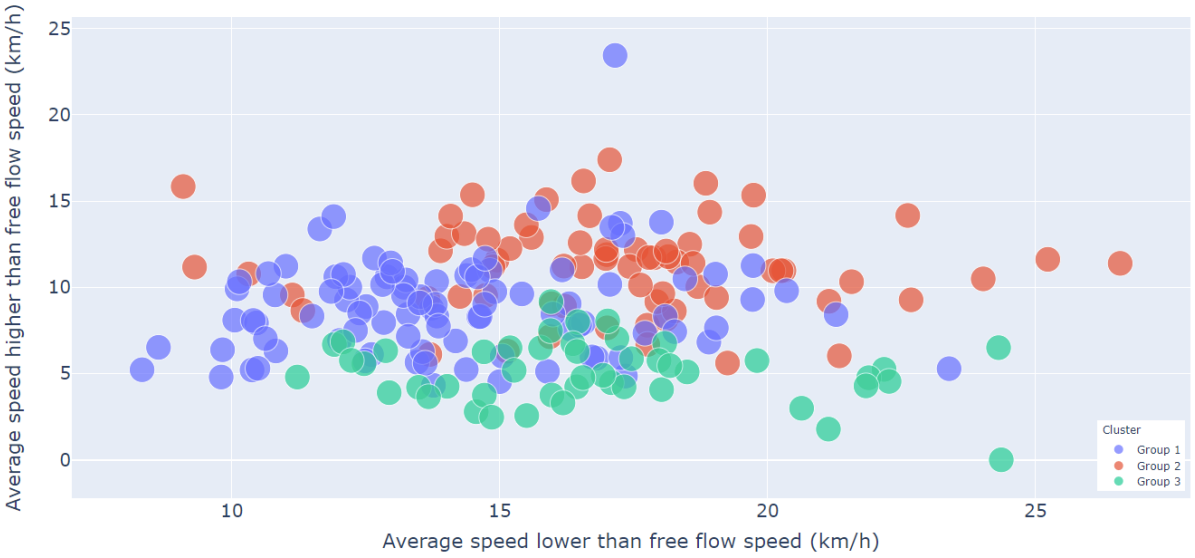

Figure 5.6: Average difference of vehicle speed and free flow speed for each trip in the data set (different groups are colored by different colors)

the difference being that road context is identical to the traffic context under ideal condition. Confirming the polar graph information, trips belonging to Group 2 have on average the highest positive difference between vehicle speed and the surrounding traffic and free flow speed, while Group 3 has the lowest. As for the values of speed being lower than traffic and free flow speed, the graphs differ slightly, but the Groups 1 and 3 have higher values than Group 2 in both cases.

After examining the properties of the clusters and trips from several different viewpoints, the final interpretation of groups can be summarised as following:

- **Group 1** - Driving style of this group is characterised with high speeds and high RPM, with vehicles covering longer distances over a longer period of time. The driving speed is in line with the traffic speed and usual speeds on the road. This indicates that the trips belonging to this group take place on fast roads or highways, where there are no frequent stops.
- **Group 2** - Driving style of this group is characterised with low speeds, low RPM, high acceleration and high deceleration, over short distances. Due to the high acceleration and deceleration values, the driving speed in the trips has high differences when compared to the traffic speed and usual road speeds. The properties of the trips indicate that they take place in urban areas.
- **Group 3** - Driving style of this group is characterised with low speeds, low RPM, moderate acceleration and moderate deceleration, over short distances. The driving speed is usually in line or lower than the rest of the traffic. As with Group 2, the properties of the trips indicate that they take place in urban areas, but lower acceleration, deceleration and difference with traffic speed point to a more defensive style of driving.

Additional observations can be made when taking a look at the distribution of the trips per

Table 5.4: Data collection participants information and number of trips per group

Driver ID	Group		
	1	**2**	**3**
A	**27**	-	-
B	7	**21**	9
C	**6**	2	-
D	**9**	1	7
E	**14**	1	-
F	-	**13**	-
G	3	**28**	3
H	2	-	**25**
I	**28**	1	5

group by driver. Table 5.4 summarizes the number of trips per group for each driver coupled with corresponding driver IDs (as seen in Table 4.1). The table shows that each driver has a dominant group of trips in a group, which can be interpreted as their driving style. Therefore, by categorizing the trips, the driver's driving styles can also be categorized.

5.3Driving Route Comparison

The final example of the contextually enriched automotive data set analytics deals with the comparison of driving route properties. The route comparison is achieved using a developed software system for interactive visual analysis of the collected data set. The idea behind the solution is to enable a way to detect routes (or route segments) with identical starting and stopping points and provide a visual graphical comparison.

To determine related routes, an algorithm was established for detecting similar route segments. The route segment which is being searched for is defined by its starting and ending location coordinates and the search radius. The search radius describes the radius around the starting and ending coordinates in which the search will be performed. These variables are defined by the user performing the search. To obtain a set of data that corresponds to the user's request, it is necessary to first find the trips which were driven through the defined start and end search radius. For the purpose of explaining the used algorithm, all trips contained in the data set will be denoted by T_{all}, the trips with routes which pass through the start radius by T_{start} and the trips with routes which pass through the end radius by T_{end}. Their relation can be described as:

$$T_{start} \subset T_{all}, \quad T_{end} \subset T_{all} \tag{5.2}$$

The trip T belongs to a given search query if it belongs to the set T_{start} and the set T_{end}, with the condition that the timestamp of the location contained in the start radius has a timestamp smaller than the location contained in the end radius. If the locations contained in the trip which belong to the start search radius are denoted by R_{start} and the locations contained in the trip which belong to the end search radius by R_{end}, the trips resulting from the search (T_{search}) can be described as:

$$R_{start} \in T \in T_{start}, \quad R_{end} \in T \in T_{end} \tag{5.4}$$

$$t(R_{start}) < t(R_{end}) \implies T \in T_{search} \subset T_{start} \cap T_{end} \tag{5.5}$$

However, only trip segments which match the selected start and end locations are taken into account for comparison. In other words, for each trip T which belongs to the set of searched trips T_{search}, only a subset of data points is taken into account, starting with the data point belonging to R_{start}, and ending with the data point belonging to R_{end}. If n is the number of data points in the segment, it can be defined as:

$$S \in T \in T_{search} \tag{5.6}$$

$$S_1 \in R_{start}, \quad S_n \in R_{end} \tag{5.7}$$

Finally, if m is the number of trips matching the search criteria, the set of trip segments for route comparison resulting from the search can be defined as:

$$S_{search} = S_{1..m} \in T_{1..m} \in T_{search} \tag{5.8}$$

For the previously described search process to be time efficient over the MongoDB database, a new index had to be added. When querying a specific MongoDB database without a pre-set index, MongoDB first scans the collections or scans each document in the collection to select the documents that match the query. Unlike the classic search method, index-based search limits the number of documents that MongoDB must view. Indexes are special data structures within NoSQL databases that store a subset of collection data in a custom MongoDB format to make iteration as short as possible. The subset that the index stores is the value of a specific field or set of fields that automatically sorts when stored. MongoDB supports geospatial data as a data type and offers two special indexes that make queries over geospatial data efficient. These are the indexes *2d* and *2dsphere*. The *2d* index uses Euclidean geometry (also called plane geometry) and is mostly used in art and architecture, while the *2dsphere* index uses non-Euclidean

Figure 5.7: Start (green) and end (red) location of a search chosen on a map

geometry (also call spherical geometry), and is mostly used in navigation and astronomy, making it more suitable for this use case. A prerequisite for searching through the MongoDB index *2dsphere* is to modify documents in the data collections so that location data is converted to GeoJSON objects. GeoJSON objects will then save the specified index as a separate sorted collection and thus reduce the size of the documents being searched, i.e. remove the initial scan of all documents in the collection from the search process, providing a more time-efficient solution.

To initiate search, a user must provide a start and end location, and the search radius. This can be done using the input form, where the address or coordinates can be entered, along with the radius. A visual way of choosing the locations is also provided, and can be seen in Figure 5.7. Using this method, the user can click on the map to choose start and end location. The start location is indicated with a green circle in the pin, while the red circle indicates the end location. However, the search radius must still be entered via the input form. If the search radius is not entered, a default value of 50 meters is used.

The result of a search with the defined start and end locations seen in Figure 5.7 is displayed in Figure 5.8. The figure shows the trips which are drawn in different colors to make them easier to spot and differentiate. It can be noticed that the trips do not have the same route, but what connects them are the geographical positions of the start and end point. It is important to note that the presented results do not contain entire trips but a subset of data belonging to a given trip as previously explained, as this provides a quality insight into the efficiency of certain routes for a particular segment and the characteristics of drivers in certain driving conditions in that segment. In other words, the user focuses only on the desired part of the trips.

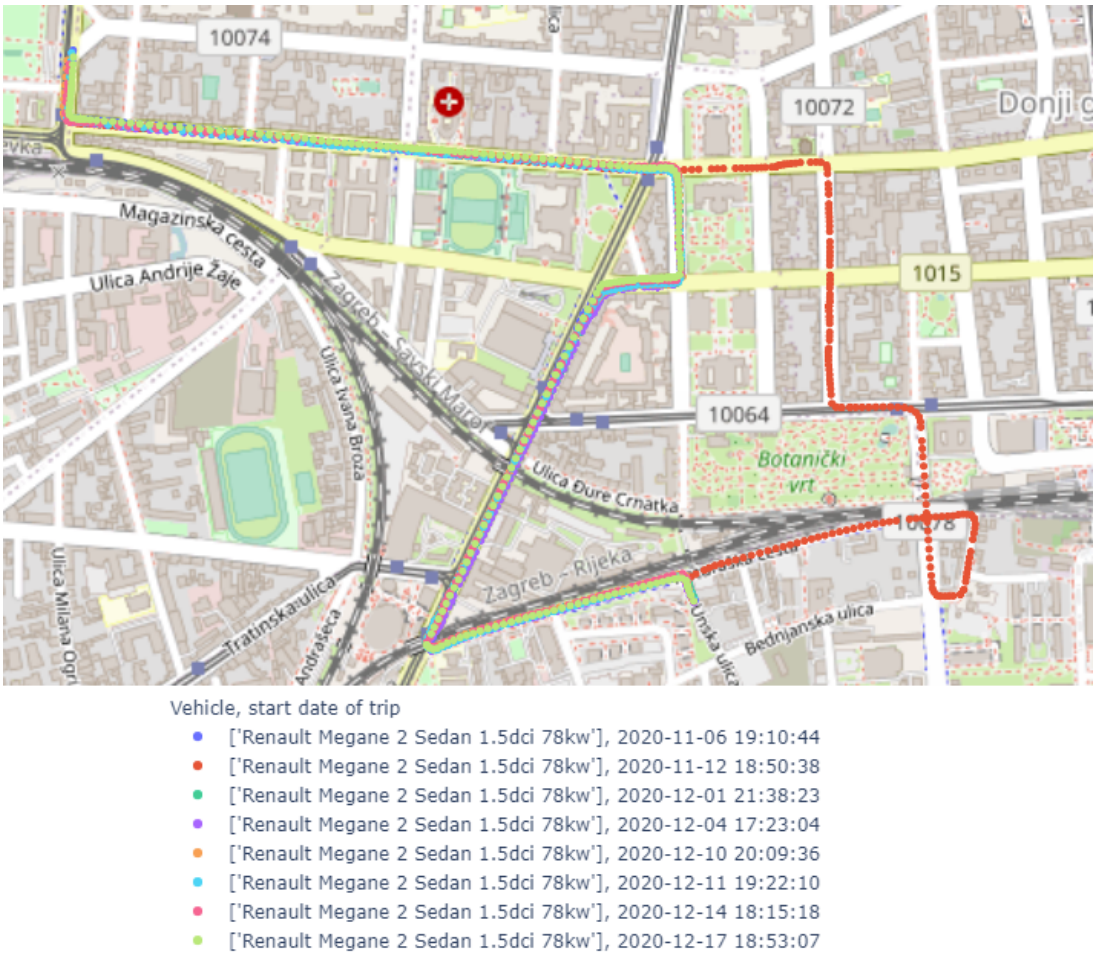

Figure 5.8: Routes which resulted from a search defined by the start and end locations displayed in Figure 5.7

Once the user defines the search locations, he is presented with several different visual analysis tools for the chosen trip segments. Some of the graphical comparison representations offered to the user will be demonstrated through two different case studies. For the first case study, the focus will be on the comparison of trip durations and speed, while the second case study comparison will focus on average speed during different weather conditions.

The first case study is a segment defined by the addresses Unska ul. 1, 10000, Zagreb and Ul. Republike Austrije 25, 10000, Zagreb, with the search radius set at 50 meters. The search results for this query were previously seen in Figure 5.8. Both addresses are located in the center of the city of Zagreb and all trips take place in city driving conditions. The first graphical representation in which the journeys of the defined segment are compared with respect to their duration is visible in Figure 5.9.

The bar chart shows the duration of the trip in minutes only in the displayed segment, not on the entire trip. All of the trips shown on this segment are made by the same driver during his daily commute. With this graphical display, the most time-efficient journey on the segment can easily be noticed, and closer analysis can show which routes have better duration results.

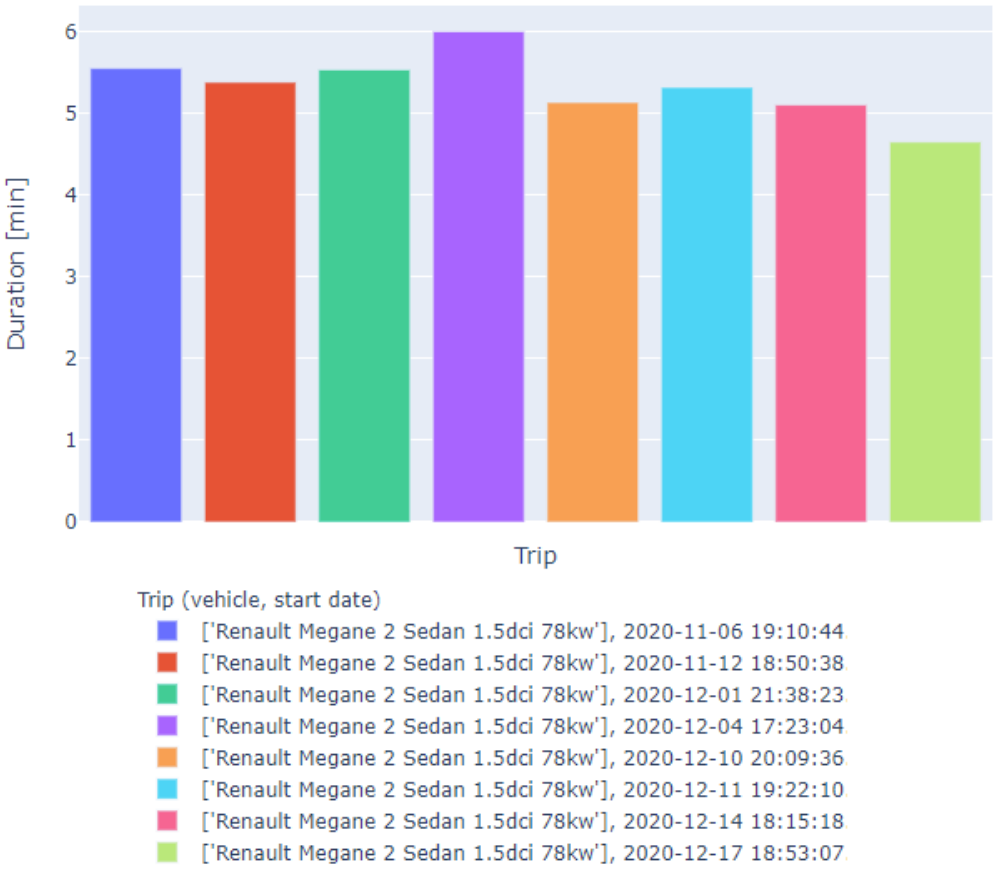

Figure 5.9: Graphical comparison of travel duration on the defined segment

Such an analysis can be of great benefit when solving routing optimization problems. For the observed scenario, it can be concluded that all trips except the one marked in red took place along the same route, and the trip durations on both routes have similar values.

The next graphical representation is the analysis of the average vehicle speed on the defined segment, and is visible in Figure 5.10. This analysis provides additional context to the previous analysis. The duration of an individual trip depends on the speed, but also the distance traveled. Due to the distance traveled, the journey may have a shorter duration and a lower average speed, which tells us about the importance of observing these two graphical representations in parallel. Specifically for a given scenario, it is easy to see that the two trips that achieved the highest results in terms of average speed are the trips marked in green and red. The trip marked in green had the best result in terms of time, so it is not surprising that it also had the highest average speed. In contrast, the result of a trip marked in red provides insight into the efficiency of this route. Observing the duration of the trip, it can be seen that the mentioned trip does not stand out in any way, in fact it occupies the fifth place out of eight analyzed. When the fact that the recorded time is reached at the second highest average speed is taken into account, it can be concluded that the route it uses is not efficient.

The second case study provides a comparison of trips over different weather conditions. For

Figure 5.10: Graphical comparison of average vehicle speed on the defined segment

the purposes of this analysis, a different segment will be used that contains data on the travel of different vehicles/drivers. The trip routes and the legend describing the drivers and trip times are displayed in Figure 5.11. The segment of the second case study is defined by the starting address Obrtnička 29, 10437, Rakitje and the ending address Ul. Milana Prpića 44-46, 49243, Oroslavje, with a search radius set at 300 meters. For the purpose of understanding the given segment, it can be said that both addresses are located in the vicinity of the city of Zagreb, i.e. the cities of Oroslavje and Sveta Nedelja, and all trips take place on fast roads or highways. On this segment, ten trips were made by three different drivers. Some trips on the displayed maps are covered and cannot be seen as multiple trips were made along the same route. This route is marked yellow on the map, and 9 out of 10 trips were completely or partially driven along it.

Before analyzing the remaining graphs, it is important to note that the trip marked in cyan has a different route than all others shown and achieves the best results in terms of travel duration and average speed in the segment defined in this case study. Consequently, it can be concluded that this route is most efficient of those shown on the map. The graphical representation that will be analyzed is shown in Figure 5.12 and provides a comparison of the average speed in different weather conditions grouped according to the gender of the driver. The aim of this analysis is to provide a comparison of driving characteristics in individual weather conditions

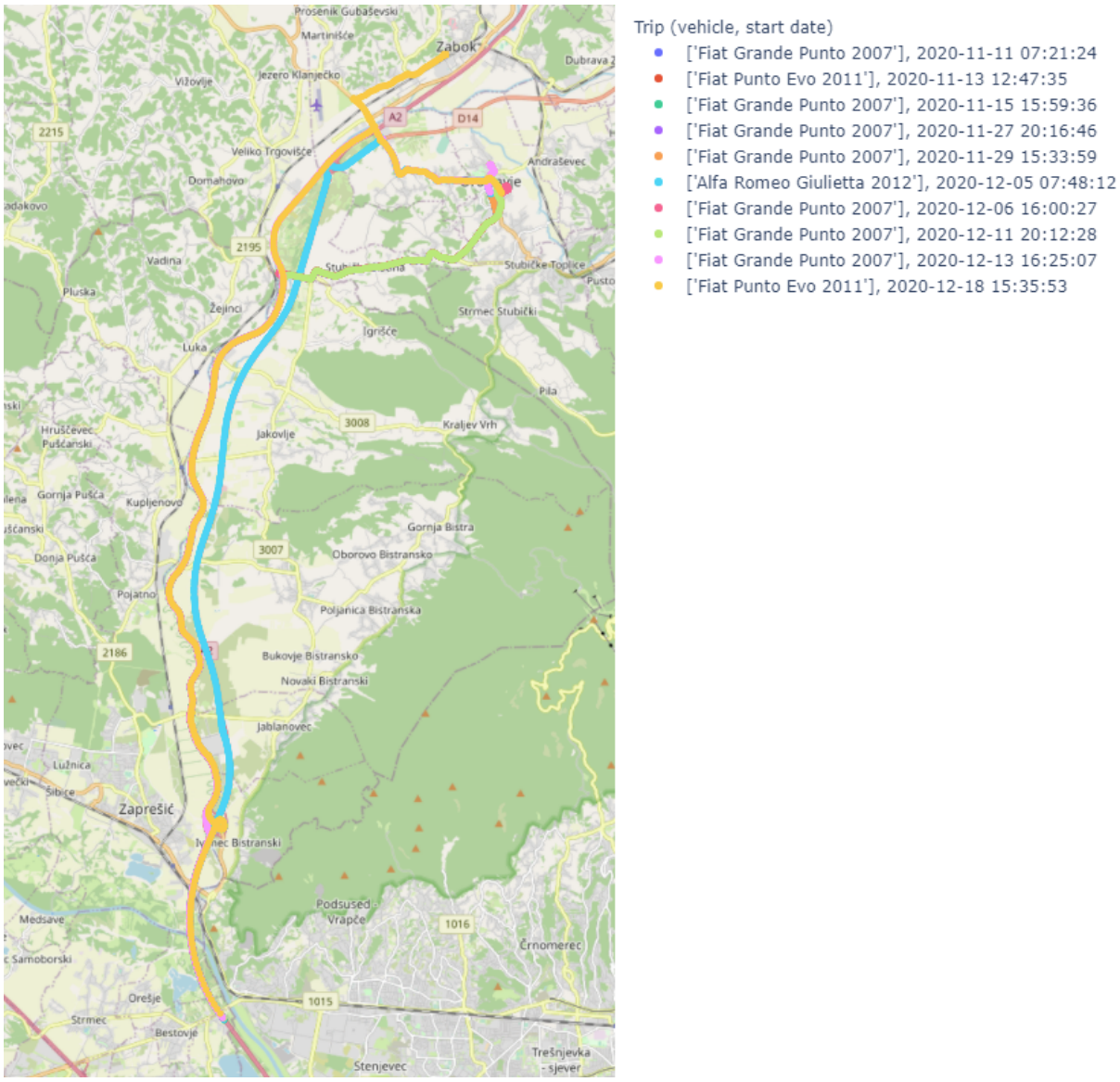

Figure 5.11: Search results for the route segments between the cities of Oroslavje and Sveta Nedelja

between men and women.

According to a study by Aldred et al. [137], women generally take much less risk in their driving and consequently are statistically better drivers, as evidenced by the lower number of traffic accidents. Although the collected data set has a small number of drivers, it can be observed if this hypothesis is valid for the data set. The bar chart in Figure 5.12 shows that it is difficult to define the behavior of men in different weather conditions, while for women there is an evident change in driving behavior under worse weather conditions. In cloudy conditions without precipitation, women achieved a higher average speed than men in the defined segment, while, for example, in foggy and drizzle conditions, they achieved significantly lower values of average speed.

The next graphical display provides insight into the correlation of the average travel speed and visibility for different vehicles. It is worth noting that the three vehicles shown in the

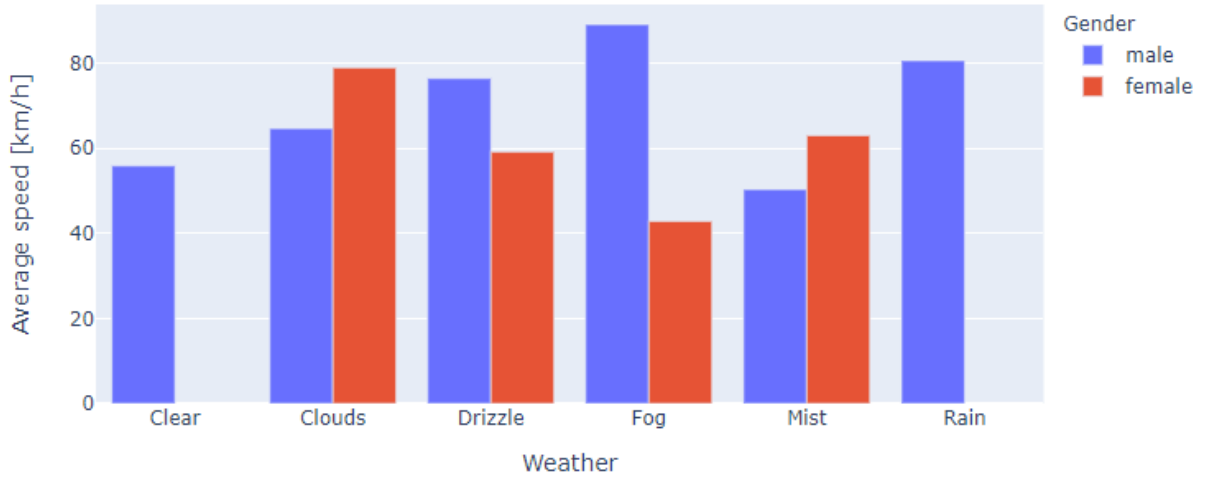

Figure 5.12: Graphical comparison of average vehicle speed in different weather conditions grouped by gender

bar chart, are driven by three distinct drivers, so each vehicle corresponds to a single driver. Visibility is defined as the distance expressed in meters at which an object can be clearly seen. In the first case observed, this data does not play a significant role due to the urban environment where visibility is already limited by a large number of buildings. The second observed segment contains trips that are driven on an open road where fog, precipitation and other factors that affect visibility have a significant impact on the characteristics of the trip. Figure 5.13 shows a comparison of the average speeds of men and women in visibilities ranging from zero to 10,000 meters, which is the highest visibility value returned by the OpenWeather API. By analyzing the graphic display, once again a confirmation can be made regarding the statement that female drivers commit to safer driving in unfavorable conditions.

The analysis of the correlation of the average speed with weather conditions and visibility is also grouped by the vehicle performing the trip. These graphical representations are visible in

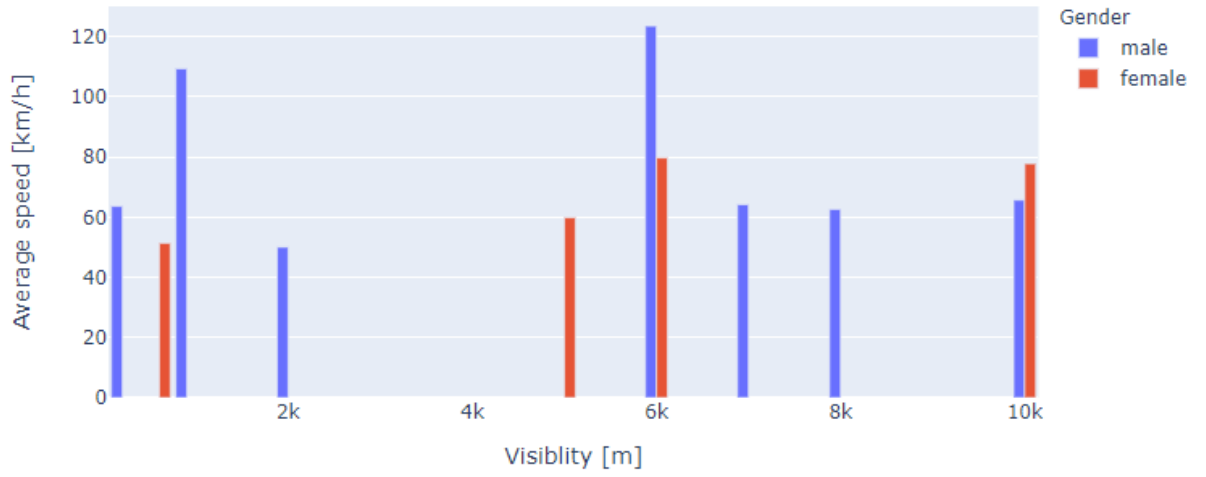

Figure 5.13: Graphical comparison of average vehicle speed in different visibility grouped by gender

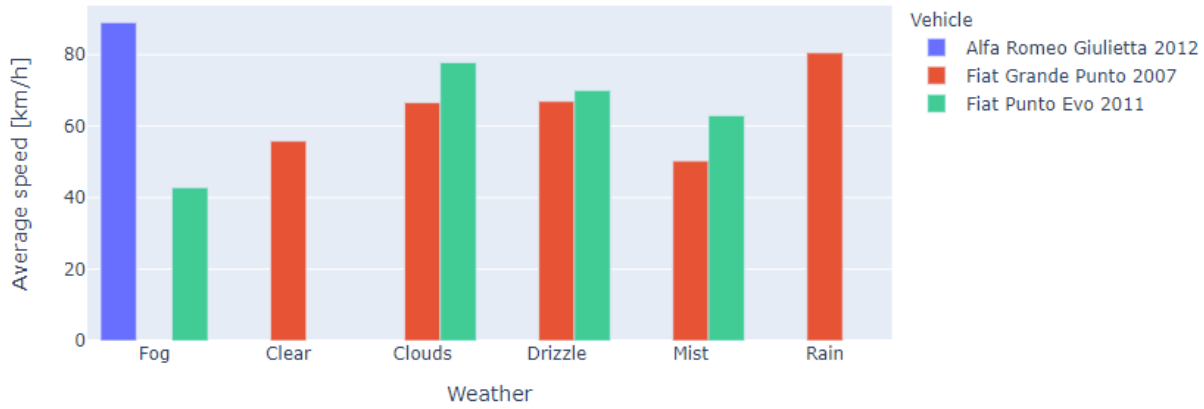

Figure 5.14: Graphical comparison of average vehicle speed in different weather conditions grouped by vehicle (each vehicle identifies a different driver)

Figures 5.14 and 5.15. Given that nine different drivers and eight different cars participated in the study, vehicles in all but one case uniquely define the driver. By analyzing the data recorded by different vehicles in certain segments, it can be determined whether there is a relationship between drivers driving cars with similar characteristics (such as engine power, year of manufacture, size). The two figures display the trips made by three different vehicles. As Alpha Romeo Giulietta 2012 is represented by only one trip, no conclusions can be made regarding the behaviour of the driver in different weather conditions. As for the other two vehicles/drivers, Fiat Punto Evo 2011 has lower average speeds in unfavorable weather conditions such as drizzle, mist and fog, while it's average speed shows no correlation to visibility. Similarly, the Fiat Grande Punto 2007 average speeds are lower in misty conditions, however, in this case rain and drizzle do not affect the average speed value. Also, like in the case of trips made by Fiat Punto Evo 2011, the average speed has no apparent correlation with visibility for the trips made by Fiat Grande Punto 2007.

The last displayed graph for this case study can be seen in Figure 5.16 and demonstrates

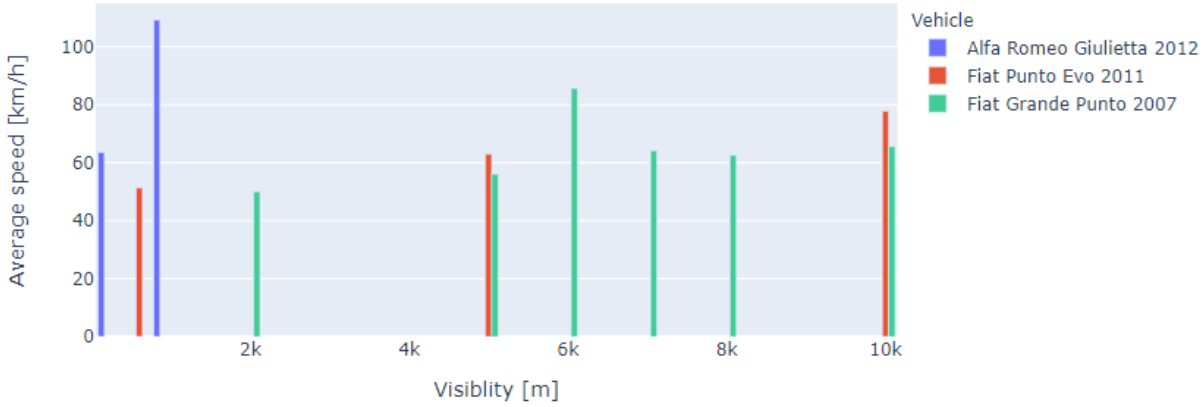

Figure 5.15: Graphical comparison of average vehicle speed in different visibility grouped by vehicle (each vehicle identifies a different driver)

Figure 5.16: Graphical comparison of average free flow, traffic and vehicle speed for the second defined trip segment

the comparison of the average value of free flow, traffic and vehicle speeds for the trips in the defined segment. First of all, it can be noticed that the average traffic speed values are usually close to the average free flow speed values, which means that most of the trips were made in conditions of light traffic. Another thing which can be observed is that none of the trips have a higher average speed than the average traffic speed, indicating that the drivers of these trips do not drive faster than the surrounding traffic. Finally, in trips which have a higher difference between the free flow and traffic speed (which indicates heavier traffic than usual), the difference between the traffic and vehicle speed is also higher. This shows that the drivers performing these trips are more careful and adjust their vehicle speed in heavier traffic conditions.

Chapter 6

Conclusions and Future Work

In the last several years, companies are increasingly using data processing and analysis to make informed engineering and business decisions. This can perhaps best be seen in the increase of revenue from big data and business analytics, which is projected to grow to $274 billion in 2022, while it was worth $122 billion in 2015 [138]. Software is the key driver for creating insights from previously collected data. Recently, there has been a significant increase in the amount of data collected in virtually all industries operating today, and the automotive industry is not different. This is mainly due to the rising adoption of connected cars which are capable of communicating with the Internet, and can therefore be utilized for data collection [3]. Vehicle generated data is valuable for use cases such as fleet management [139], predictive maintenance [140], driver behaviour profiling [141] and it is especially useful in applications dealing with transportation sustainability [142], as the vehicle's energy efficiency and energy consumption can be measured and monitored using automotive data.

The major obstacle for performing research in this area is that there is no open-source contextually enriched automotive data set available. This thesis aims to overcome this obstacle by providing a solution for collection of such a data set in the form of a framework for collection and contextual enrichment of automotive data. The proposed framework was modelled and developed in a way that satisfies the main research questions which were stated in the introduction of the paper. The data collection process was done by utilizing smartphones as the data collection tool to cover a broad range of diverse vehicles. Also, two automotive communication technologies - OBD-II and CAN are supported by the framework to further maximize the spectrum of vehicles from which the data can be collected. Additionally, to facilitate the development of CAN bus collection modules, a CAN bus simulator was modelled, developed and validated using multiple mathematical similarity measures. Smartphones also allow the contextual enrichment of automotive data using its built-in sensors and Internet connection. The challenge of contextually enriched automotive data storage was solved by implementing a custom distributed data storage platform. The data platform was constructed in a manner

that addresses the requirements of reliability, scalability, and fault tolerance, as well as requirements imposed by the big data complexity, and was validated on several electromobility use cases. The final part of the framework is the application programming interface which serves as a middleware solution between the end-users and the data storage platform and is used for accessing and analysing the stored data. Although the data collection process and data model used by the API were explained to provide a better understanding of the API's functionalities, the API is agnostic to the data collection process. Therefore, the API can also be used with a different storage solution behind it by changing the database connection library utilized by the API. In any case, the API described in this thesis can be used as a blueprint for the design and implementation of an API for advanced analytics intended to be used with automotive data.

The established and implemented framework was used in the data collection experiment to collect a contextually enriched automotive data set. The data set consists of 287,882 data points collected during 212 trips made by 9 drivers, resulting in a total of 94 hours, 25 minutes and 44 seconds of recorded driving time. This data set has been made publicly available for researchers to access and use. To further validate the framework and to demonstrate the research value of such a data set, the data set was analysed in several different use cases. It has been displayed how such a data set can be used in the field of transportation sustainability to evaluate the eco-efficient driving patterns of a group of drivers. For this reason, a metric named eco index was established which incorporates for eco-efficient driving advice and the drivers who participated in the data collection experiments were ranked accordingly. The established evaluation process is mathematically accurate and can be useful for applications such as vehicle fleet management, where cost and gas emission reduction can be achieved through improved fuel efficiency. Additionally, driving patterns were evaluated on the level of a single trip, and it was demonstrated how the trips can be divided into groups according to the similarity of their properties. Three groups were identified, which represent an interpretive classification of the trip. The results of the analysis clearly identified differentiating characteristics of each group. That said, Group 1 represents fast rides, long distances without major stops. Group 2 and 3 are both at lower speeds. Group 2 is characterised by a desire for accelerated movement and tendency to drive above the speed of the rest of the traffic. Group 3, on the other hand, includes trips with a smoother driving style that blends in with the surrounding traffic. The last data analytics use case demonstrated was a driving route comparison measurement which was established and implemented in a software system for interactive visual analysis of the collected data set. The system allows a user to detect the best route for the selected start and end location, analyse the driving style of each driver, compare driver behaviour in different weather conditions and provide insight on the driver's speed compared to the rest of the traffic on each trip which can be used to ultimately improve their driving style.

From a business perspective, the framework can impact many of the stakeholders involved

in mobility. Infrastructure owners are greatly influenced by the developments in the automotive industry. With the number of vehicles on the road continually growing, the fuel and energy infrastructure owners have to deploy more and more gas stations and EV charging stations and various research on the methodology of the deployment [10], [143] is already being made. The framework can help with these problem by providing the data for the analysis of driving patterns which can be helpful when determining the optimal placement positions. For the same reason, municipalities and governments responsible for the design, construction, management, safety and maintenance of the road infrastructure can also benefit from the data collected by the framework. As the framework is equipped with the ability to collect mobile network signal data along the travel routes, telecommunication companies can use this information to target the infrastructure upgrades to areas with low signal and areas with a higher amount of traffic. On the other hand, for most of the vehicle owners, vehicle efficiency is among the key concerns. As a part of this thesis it was demonstrated how the data collected by the framework can be used for the calculation of the driver's and vehicle's eco-efficiency. Also, smartphones are deeply integrated into today's lifestyle, and an average person today is more tech savvy than ever. This is why the framework's usage of smartphones as a data collection solution should not pose a significant issue for the drivers who opt for their driving data to be collected.

All of this makes the framework described in this thesis a valid blueprint for any kind of automotive data collection system where data needs to be collected, enriched and analysed and in real-time. However, there are several limitations of the framework which can be improved upon in future work. Improvements can be made to the framework in regard to supporting an additional mobile operating system along with Android as data collection solution to expand the potential user base. An obvious choice would be iOS as it has the second-highest market share of all mobile operating systems on the market. Additional contextual enrichment sources could also be added to the data collection process in future versions of the framework. For example, a new module can be added to the framework which would be used for evaluating the driving patterns of the drivers from the point of view of vehicle passengers. This would add a new dimension to the collected data and could be useful in research pertaining to mobility-as-a-service (MaaS) solutions. The collected data set could also be explored even further, as the focus in the analysis presented in this thesis was mainly on eco-efficiency, and that is not the only application which was planned when designing the proposed framework. Since automotive data is enriched not only with location, weather and traffic information, but with sensor and mobile network information, the data set can prove useful in various other areas of research. An analysis of the sensor data could show if the driving style can be recognized using nothing but information collected from smartphone sensors, while mobile network information can be used to analyse the existing network infrastructure.

Appendices

Appendix 1: ELM327 Bluetooth-to-OBD-II Device Installation Manual

To install the ELM327 Bluetooth-to-OBD-II device inside of the vehicle, the user must first locate the vehicle's OBD-II port. This port is usually found close to the dashboard or the steering wheel. Figures 6.1 and 6.2 show the location of such a port in Fiat Grande Punto 2007.

Figure 6.1: OBD-II port location in Fiat Grande Punto 2007

Figure 6.2: ELM 327 device next to the OBD-II port location in Fiat Grande Punto 2007

Once the OBD-II port is located the ELM327 device can be connected to the port. If the device is properly connected, the red light on its circuit board should light up as shown in Figure 6.3.

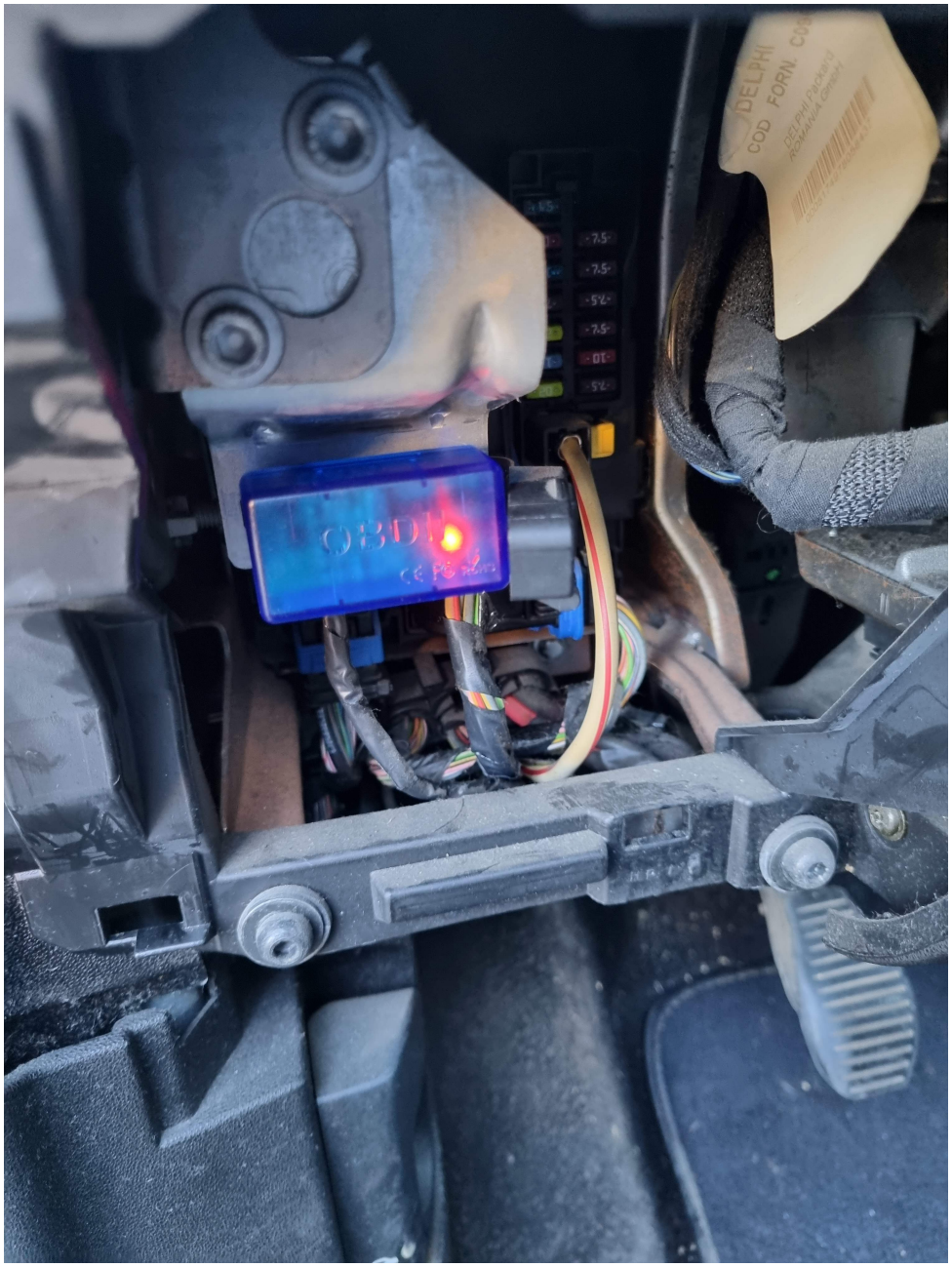

Figure 6.3: ELM 327 device connected to the OBD-II port location in Fiat Grande Punto 2007

Appendix 2: ELM327 Bluetooth-to-OBD-II Device Smartphone Connection Manual

After the ELM327 device is connected to the vehicle, the user can connect to it using the smartphone's Bluetooth connection. To connect to the device, the user must choose the device named "OBDII" inside the Bluetooth settings screen (as seen on the bottom of Figure 6.4).

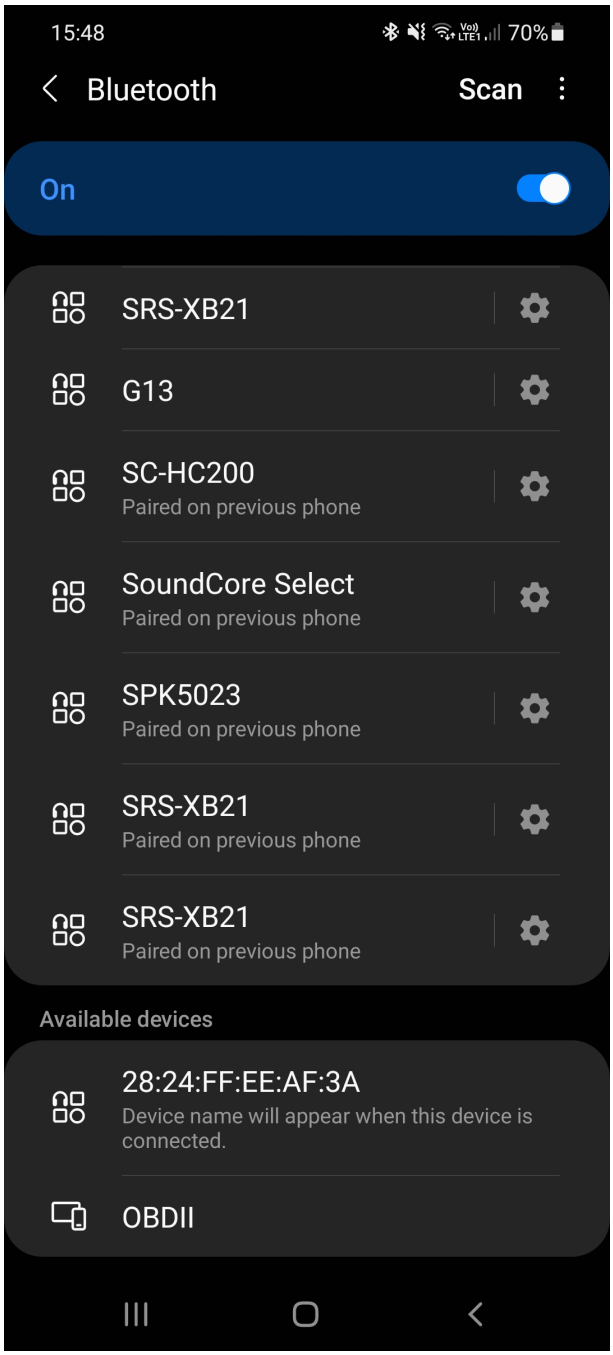

Figure 6.4: Bluetooth devices list with ELM327 device as "OBDII" displayed as the bottom item

Once the "OBDII" device is selected, the device PIN must be entered to complete the Bluetooth pairing process. After the correct PIN is entered (default value is "1234"), the connection is established.

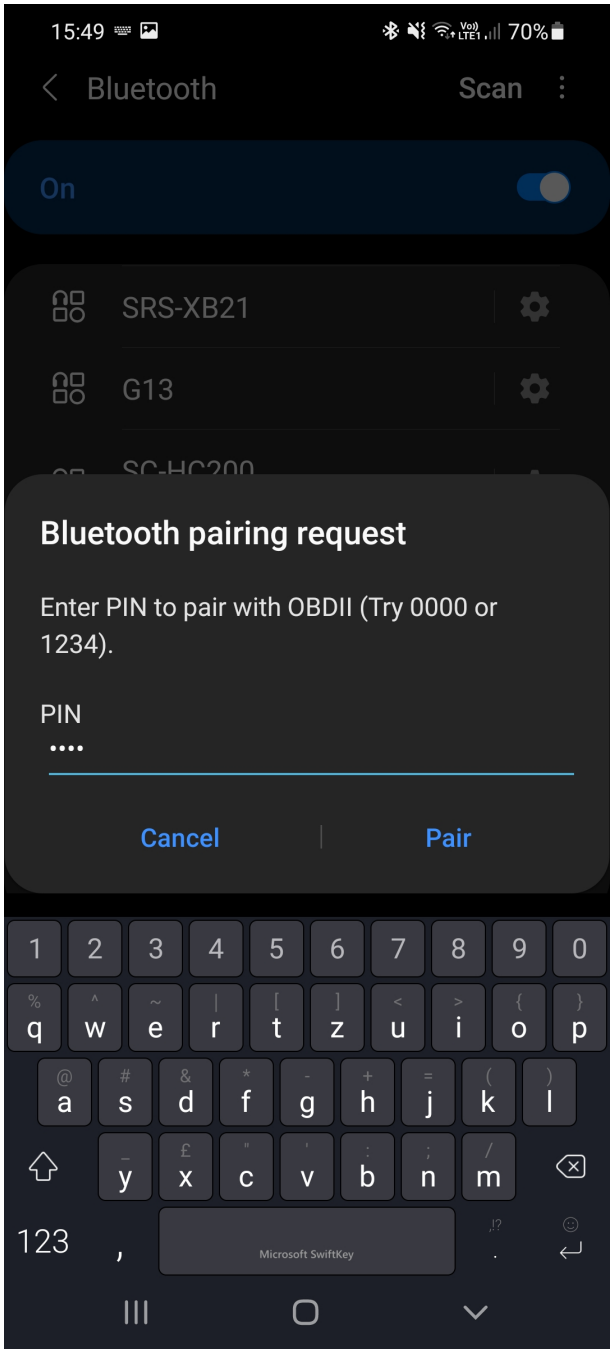

Figure 6.5: Bluetooth pairing request dialog where PIN must be entered to connect the smartphone to ELM327

Appendix 3: Data Collection Application Manual

When the application is started for the first time, three permissions must be allowed: location (Figure 6.6) - for the collection of location data, phone (Figure 6.7) - for the collection of mobile network data and storage (Figure 6.8) - for caching the collected data to device storage before sending it to the platform.

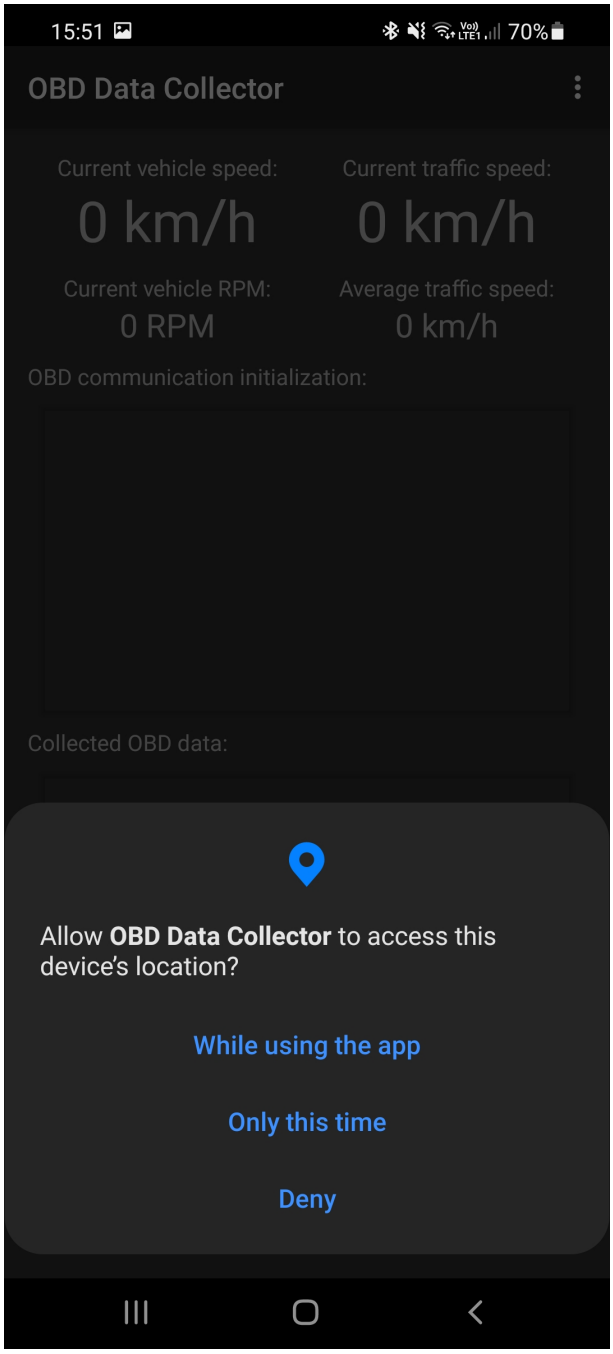

Figure 6.6: Location permission dialog presented the first time the collection application is started

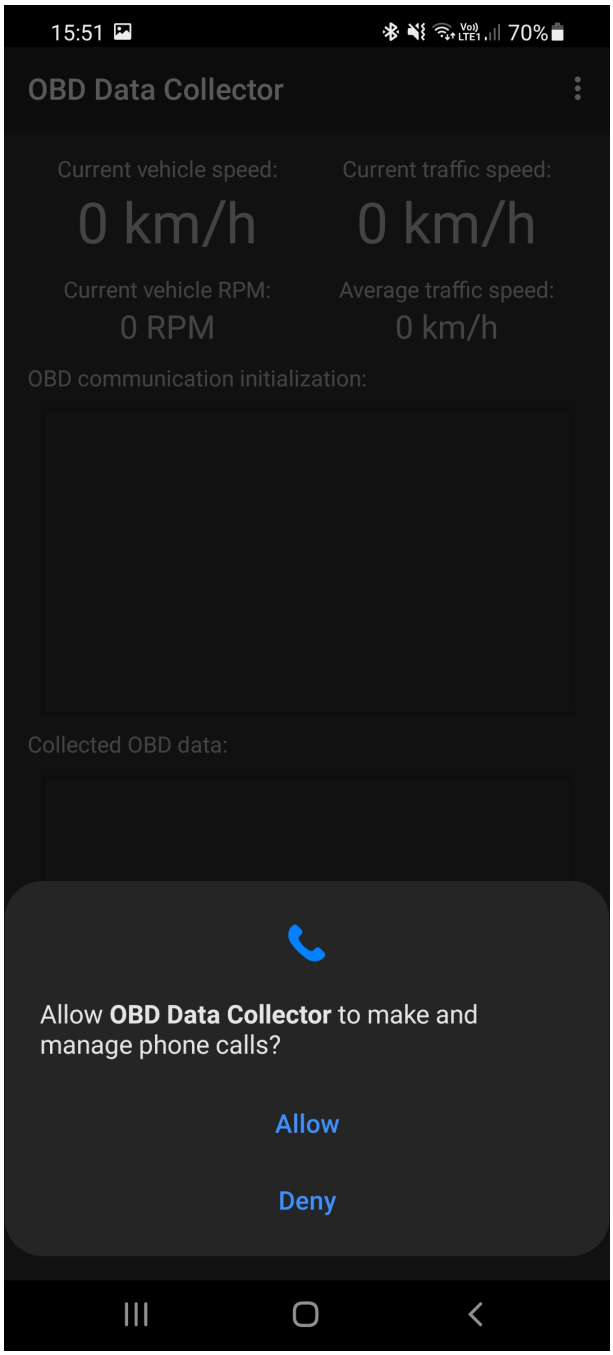

Figure 6.7: Phone permission dialog presented the first time the collection application is started

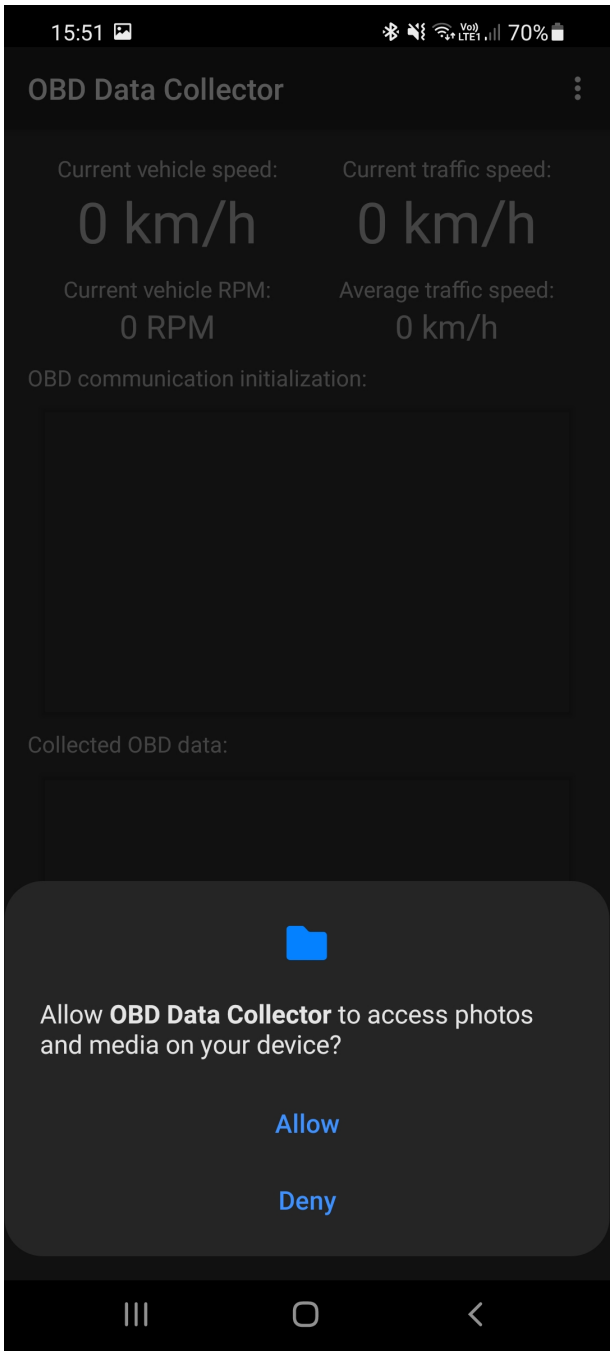

Figure 6.8: Storage permission dialog presented the first time the collection application is started

After the required permissions are accepted, the initial screen is presented to the user as seen in Figure 6.9.

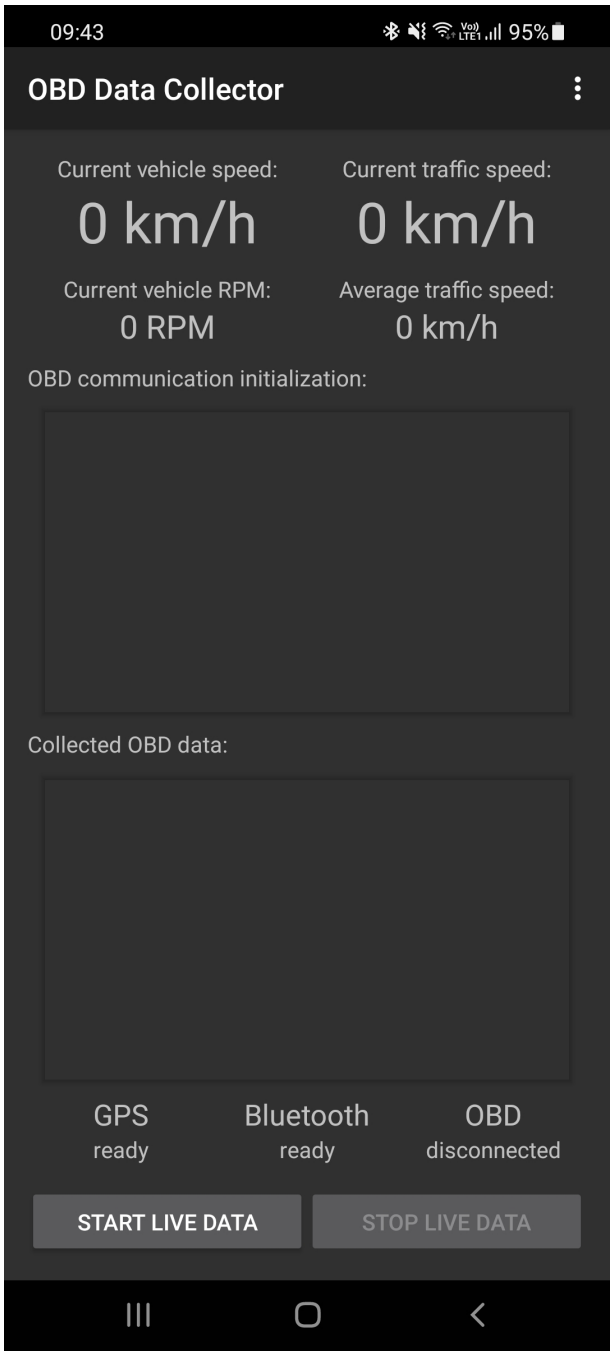

Figure 6.9: The initial screen of the data collection application

Once the user presses the "Start live data" button for the first time, a dialog is opened which requires the user to enter basic vehicle information, vehicle fuel type, and choose the ELM327 Bluetooth device from the list of connected Bluetooth devices (Figure 6.10).

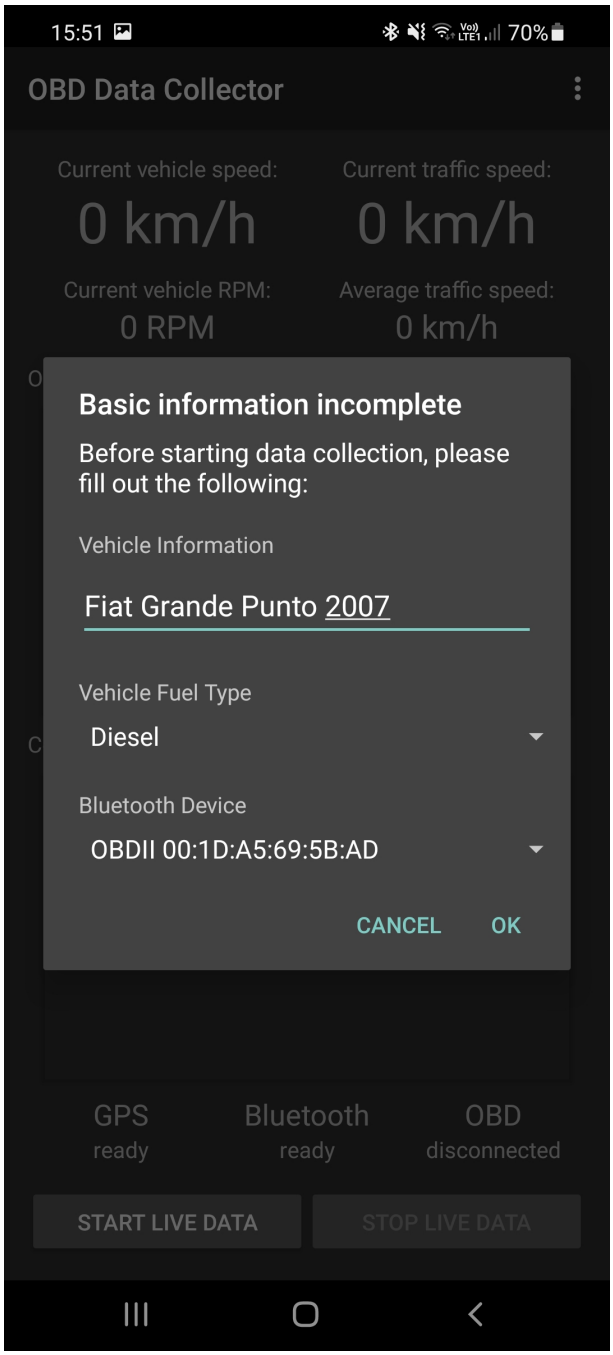

Figure 6.10: Basic vehicle information dialog

After the required data is entered, the data collection process starts. Figure 6.11 shows the application while data collection is active.

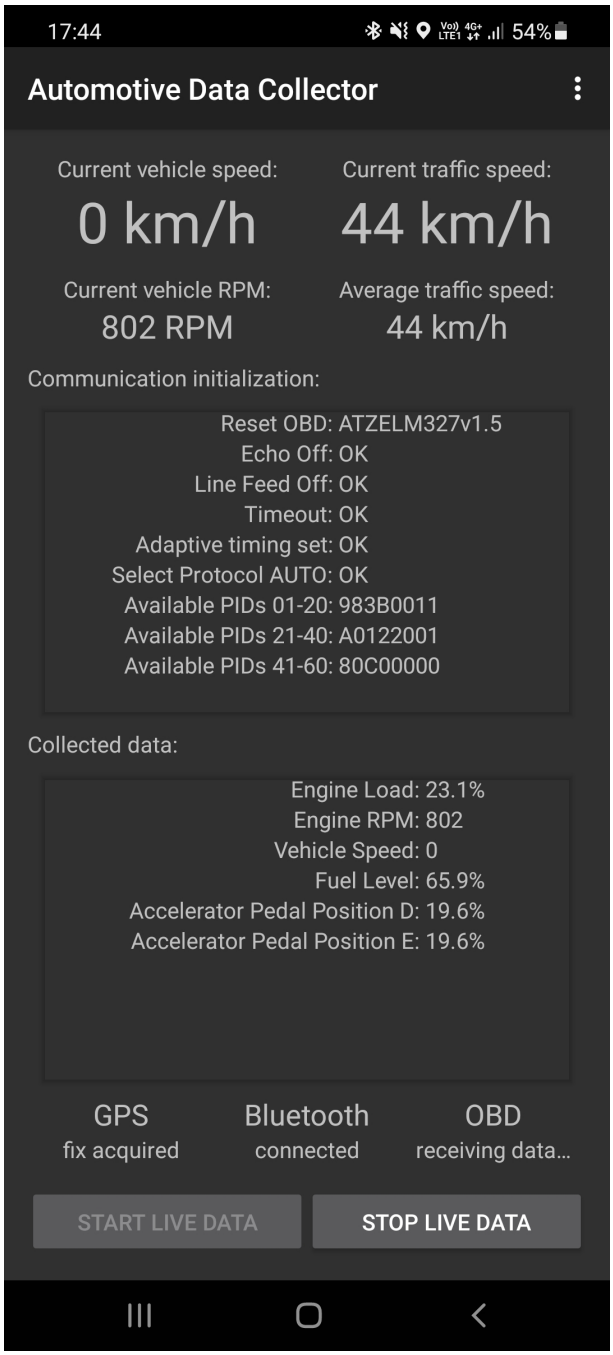

Figure 6.11: Basic vehicle information dialog

Bibliography

[1]Dodge, M., “Code/space and the challenge of software algorithms”, in Handbook on Geographies of Technology. Edward Elgar Publishing, 2017.

[2]Liu, S., “Automotive software market value 2018-2023.”, available at: www.statista.com/statistics/590055/worldwide-automotive-software-market-size/ 2019.

[3]Vdovic, H., Babic, J., Podobnik, V., “Automotive software in connected and autonomous electric vehicles: A review”, IEEE Access, Vol. 7, 2019, str. 166 365–166 379.

[4]Wagner, I., “Internet-connected vehicles - worldwide and united states 2023.”, available at: https://www.statista.com/statistics/275849/number-of-vehicles-connected-to-the-internet/ 2019.

[5]Viereckl, R., Ahlemann, D., Koster, A., Jursch, S., “Racing ahead with autonomous cars and digital innovation”, Auto Tech Review, Vol. 4, No. 12, 2015, str. 18–23.

[6]Lu, N., Cheng, N., Zhang, N., Shen, X., Mark, J. W., “Connected vehicles: Solutions and challenges”, IEEE internet of things journal, Vol. 1, No. 4, 2014, str. 289–299.

[7]Johanson, M., Belenki, S., Jalminger, J., Fant, M., Gjertz, M., “Big automotive data: Leveraging large volumes of data for knowledge-driven product development”, in 2014 IEEE International Conference on Big Data (Big Data). IEEE, 2014, str. 736–741.

[8]Siła-Nowicka, K., Vandrol, J., Oshan, T., Long, J. A., Demšar, U., Fotheringham, A. S., “Analysis of human mobility patterns from gps trajectories and contextual information”, International Journal of Geographical Information Science, Vol. 30, No. 5, 2016, str. 881–906.

[9]Mahmood, A., Zen, H., Hilles, S., “Big data and privacy issues for connected vehicles in intelligent transportation systems”, arXiv preprint arXiv:1806.02944, 2018.

[10]Pevec, D., Babic, J., Kayser, M. A., Carvalho, A., Ghiassi-Farrokhfal, Y., Podobnik, V., “A data-driven statistical approach for extending electric vehicle charging infrastructure”, International journal of energy research, Vol. 42, No. 9, 2018, str. 3102–3120.

[11]Castignani, G., Derrmann, T., Frank, R., Engel, T., “Driver behavior profiling using smartphones: A low-cost platform for driver monitoring”, IEEE Intelligent Transportation Systems Magazine, Vol. 7, No. 1, 2015, str. 91–102.

[12]Lasocki, J., Chłopek, Z., Godlewski, T., “Driving style analysis based on information from the vehicle’s obd system”, Combustion Engines, Vol. 58, 2019.

[13]Andria, G., Attivissimo, F., Di Nisio, A., Lanzolla, A., Pellegrino, A., “Development of an automotive data acquisition platform for analysis of driving behavior”, Measurement, Vol. 93, 2016, str. 278–287.

[14]Cavallo, G., Di Mauro, F., Pasteris, P., Sapino, M. L., Candan, K. S., “Contextually-enriched querying of integrated data sources”, in 2018 IEEE 34th International Conference on Data Engineering Workshops (ICDEW). IEEE, 2018, str. 9–16.

[15]Boggio-Marzet, A., Monzon, A., Rodriguez-Alloza, A. M., Wang, Y., “Combined influence of traffic conditions, driving behavior, and type of road on fuel consumption. real driving data from madrid area”, International Journal of Sustainable Transportation, 2021, str. 1–13.

[16]Skog, I., Händel, P., “Indirect instantaneous car-fuel consumption measurements”, IEEE Transactions on Instrumentation and Measurement, Vol. 63, No. 12, 2014, str. 3190–3198.

[17]Zhou, Y., Bridgelall, R., “Review of usage of real-world connected vehicle data”, Transportation Research Record, Vol. 2674, No. 10, 2020, str. 939–950.

[18]Statista Research Department, “Connected car household penetration country comparison in 2018”, available at: https://www.statista.com/statistics/523886/share-of-households-with-connected-car-by-country/ Jan 2019.

[19]Tselentis, D. I., Vlahogianni, E. I., Yannis, G., “Temporal analysis of driving efficiency using smartphone data”, Accident Analysis & Prevention, Vol. 154, 2021, str. 106081.

[20]Oussous, A., Benjelloun, F.-Z., Lahcen, A. A., Belfkih, S., “Big data technologies: A survey”, Journal of King Saud University-Computer and Information Sciences, Vol. 30, No. 4, 2018, str. 431–448.

[21]Anuradha, J. *et al.*, “A brief introduction on big data 5vs characteristics and hadoop technology”, Procedia computer science, Vol. 48, 2015, str. 319–324.

[22]Luckow, A., Kennedy, K., Manhardt, F., Djerekarov, E., Vorster, B., Apon, A., "Automotive big data: Applications, workloads and infrastructures", in 2015 IEEE International Conference on Big Data (Big Data). IEEE, 2015, str. 1201–1210.

[23]Grimberg, E., Botzer, A., Musicant, O., "Smartphones vs. in-vehicle data acquisition systems as tools for naturalistic driving studies: a comparative review", Safety science, Vol. 131, 2020, str. 104917.

[24]Wahlström, J., Skog, I., Händel, P., "Smartphone-based vehicle telematics: A ten-year anniversary", IEEE Transactions on Intelligent Transportation Systems, Vol. 18, No. 10, 2017, str. 2802–2825.

[25]Li, P., Abdel-Aty, M., Cai, Q., Islam, Z., "A deep learning approach to detect real-time vehicle maneuvers based on smartphone sensors", IEEE Transactions on Intelligent Transportation Systems, 2020.

[26]Yin, S., Kaynak, O., "Big data for modern industry: challenges and trends [point of view]", Proceedings of the IEEE, Vol. 103, No. 2, 2015, str. 143–146.

[27]IBM, "Extracting business value from the 4 v's of big data", available at: https://www.ibmbigdatahub.com/infographic/extracting-business-value-4-vs-big-data

[28]Kopanakis, I., Vassakis, K., Mastorakis, G., "Big data in data-driven innovation: the impact in enterprises' performance", in Proceedings of 11th Annual MIBES International Conference, 22nd of June-24th of June, 2016, str. 257–263.

[29]Demchenko, Y., Grosso, P., De Laat, C., Membrey, P., "Addressing big data issues in scientific data infrastructure", in 2013 International Conference on Collaboration Technologies and Systems (CTS). IEEE, 2013, str. 48–55.

[30]Encyclopedia Britannica. (2018) History of the automobile, available at: https://www.britannica.com/technology/automobile/History-of-the-automobile

[31]Arehart, M. (2016) Gas, Electric Or Steam? Car Shopping, 100 Years Ago, available at: https://www.npr.org/2016/09/12/492841796/gas-electric-or-steam-car-shopping-100-years-ago

[32]Alizon, F., Shooter, S. B., Simpson, T. W., "Henry Ford and the Model T: lessons for product platforming and mass customization", in ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers, 2008, str. 59–66.

[33]CEBOS. (2017) The biggest milestones in the history of automotive manufacturing, available at: https://www.cebos.com/blog/milestones-automotive-manufacturing/

[34]Ross, J. N. (2013) Toyota says you might have the 40-millionth Corolla ever built, available at: https://www.autoblog.com/2013/09/07/toyota-says-you-might-have-40-millionth-corolla-ever-built/

[35]Rajashekara, K., "Present status and future trends in electric vehicle propulsion technologies", IEEE Journal of Emerging and Selected Topics in Power Electronics, Vol. 1, No. 1, 2013, str. 3–10.

[36]Cobb, J. (2016) Americans buy their four-millionth hybrid car, available at: https://www.hybridcars.com/americans-buy-their-four-millionth-hybrid-car/

[37]Lave, L. B., MacLean, H. L., "An environmental-economic evaluation of hybrid electric vehicles: Toyota's prius vs. its conventional internal combustion engine corolla", Transportation Research Part D: Transport and Environment, Vol. 7, No. 2, 2002, str. 155–162.

[38]Charette, R. N., "This car runs on code", IEEE spectrum, Vol. 46, No. 3, 2009, str. 3.

[39]Bosch, R. *et al.*, "Can specification version 2.0", Rober Bousch GmbH, Postfach, Vol. 300240, 1991, str. 72.

[40]Broy, M., "Challenges in automotive software engineering", in Proceedings of the 28th international conference on Software engineering. ACM, 2006, str. 33–42.

[41]Broy, M., Kruger, I. H., Pretschner, A., Salzmann, C., "Engineering automotive software", Proceedings of the IEEE, Vol. 95, No. 2, 2007, str. 356–373.

[42]Axelsson, J., Fröberg, J., Hansson, H., Norström, C., Sandström, K., Villing, B., "A comparative case study of distributed network architectures for different automotive applications.", 2005.

[43]Summer, N., "The concept of connected car to lead the way of the future transportation", IRA-International Journal of Technology & Engineering (ISSN 2455-4480), Vol. 1, No. 2, 2016, available at: https://research-advances.org/index.php/IRAJTE/article/view/48

[44]Barabba, V., Huber, C., Cooke, F., Pudar, N., Smith, J., Paich, M., "A multimethod approach for creating new business models: The general motors onstar project", Interfaces, Vol. 32, No. 1, 2002, str. 20–34.

[45]Coppola, R., Morisio, M., "Connected car: technologies, issues, future trends", ACM Computing Surveys (CSUR), Vol. 49, No. 3, 2016, str. 46.

[46]Nissan. (2018) Nissan delivers 300,000th Nissan LEAF, available at: https://newsroom.nissan-global.com/releases/release-4a75570239bf1983b1e6a41b7d00d8f5-nissan-delivers-300000th-nissan-leaf/

[47]Green, G. (2010) Nissan Leaf (2010) electric prototype review, available at: https://www.carmagazine.co.uk/car-reviews/nissan/nissan-leaf-2010-electric-prototype-review/

[48]Fagnant, D. J., Kockelman, K., "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations", Transportation Research Part A: Policy and Practice, Vol. 77, 2015, str. 167–181.

[49]Endsley, M. R., "Autonomous driving systems: a preliminary naturalistic study of the Tesla Model S", Journal of Cognitive Engineering and Decision Making, Vol. 11, No. 3, 2017, str. 225–238.

[50]Lambert, F. (2018) Tesla promises 'series of new Autopilot features in 2018', still plans coast-to-coast drive, available at: https://electrek.co/2018/02/07/tesla-autopilot-features-2018-coast-to-coast-drive/

[51]Rice, D., "The driverless car and the legal system: Hopes and fears as the courts, regulatory agencies, waymo, tesla, and uber deal with this exciting and terrifying new technology", Journal of Strategic Innovation and Sustainability, Vol. 14, No. 1, 2019, str. 134–146.

[52]Waymo. (2021) Waymo Driver, available at: https://waymo.com/waymo-driver/

[53]Un-Noor, F., Padmanaban, S., Mihet-Popa, L., Mollah, M. N., Hossain, E., "A comprehensive study of key electric vehicle (EV) components, technologies, challenges, impacts, and future direction of development", Energies, Vol. 10, No. 8, 2017, str. 1217.

[54]Mercedes-Benz. (2018) The Mercedes-Benz F 015 Luxury in Motion, available at: https://www.mercedes-benz.com/en/mercedes-benz/innovation/research-vehicle-f-015-luxury-in-motion/

[55]De Beaumont, C. (2017) Renault Symbioz Demo Car: Innovative, People-focused Design, available at: https://group.renault.com/en/news/blog-renault/renault-symbioz-demo-car-an-innovative-people-focused-design/

[56]Moldrich, C. (2018) Volvo 360c: driverless concept to replace short-haul flights, available at: https://www.carmagazine.co.uk/car-news/first-official-pictures/volvo/360c-concept/

[57]Yoshida, M., Tada, O., Hashime, J., “Programming of the engine control unit by the c language”, SAE Technical Paper, Tech. Rep., 1996.

[58]Wolf, M., Weimerskirch, A., Paar, C., “Security in automotive bus systems”, in Workshop on Embedded Security in Cars, 2004.

[59]Rahimi-Eichi, H., Ojha, U., Baronti, F., Chow, M.-Y., “Battery management system: An overview of its application in the smart grid and electric vehicles”, IEEE Industrial Electronics Magazine, Vol. 7, No. 2, 2013, str. 4–16.

[60]Schmid, M., “Automotive bus systems”, Automotive Applications, December, 2004.

[61]Schäuffele, J., Zurawka, T., Carey, R., Automotive software engineering. Springer, 2005.

[62]Knirsch, A., Theis, A., Wietzke, J., Moore, R., “Compositing user interfaces in partitioned in-vehicle infotainment.”, in Mensch & Computer Workshopband, 2013, str. 63–70.

[63]Al-Sultan, S., Al-Doori, M. M., Al-Bayatti, A. H., Zedan, H., “A comprehensive survey on vehicular ad hoc network”, Journal of network and computer applications, Vol. 37, 2014, str. 380–392.

[64]Jiang, D., Delgrossi, L., “IEEE 802.11 p: Towards an international standard for wireless access in vehicular environments”, in Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE. IEEE, 2008, str. 2036–2040.

[65]Uhlemann, E., “Initial steps toward a cellular vehicle-to-everything standard [connected vehicles]”, IEEE Vehicular Technology Magazine, Vol. 12, No. 1, 2017, str. 14–19.

[66]Dakroub, H., Shaout, A., Awajan, A., “Connected car architecture and virtualization”, SAE International Journal of Passenger Cars-Electronic and Electrical Systems, Vol. 9, No. 2016-01-0081, 2016, str. 153–159.

[67]Jo, K., Kim, J., Kim, D., Jang, C., Sunwoo, M., “Development of autonomous car Part I: Distributed system architecture and development process”, IEEE Transactions on Industrial Electronics, Vol. 61, No. 12, 2014, str. 7131–7140.

[68]Jo, K., Kim, J., Kim, D., Jang, C., Sunwoo, M., “Development of autonomous car Part II: A case study on the implementation of an autonomous driving system based on distributed architecture”, IEEE Transactions on Industrial Electronics, Vol. 62, No. 8, 2015, str. 5119–5132.

[69]Lin, P., “Why ethics matters for autonomous cars”, in Autonomes fahren. Springer, 2015, str. 69–85.

[70]Wilwert, C., Navet, N., Song, Y.-Q., Simonot-Lion, F., “Design of automotive X-by-Wire systems”, 2005.

[71]Reininger, M., Miller, S., Zhuang, Y., Cappos, J., “A first look at vehicle data collection via smartphone sensors”, in 2015 IEEE Sensors Applications Symposium (SAS). IEEE, 2015, str. 1–6.

[72]Albertengo, G., Buttazzo, W., Tragno, A., Ricca, M., Bragagnini, A., Quasso, R., “Smartphone enabled connected vehicles pave the way to intelligent mobility”, in WTC 2014; World Telecommunications Congress 2014. VDE, 2014, str. 1–6.

[73]Zaldivar, J., Calafate, C. T., Cano, J. C., Manzoni, P., “Providing accident detection in vehicular networks through OBD-II devices and Android-based smartphones”, in 2011 IEEE 36th Conference on Local Computer Networks. IEEE, 2011, str. 813–819.

[74]Sathyanarayana, A., Sadjadi, S. O., Hansen, J. H., “Leveraging sensor information from portable devices towards automatic driving maneuver recognition”, in 2012 15th International IEEE Conference on Intelligent Transportation Systems. IEEE, 2012, str. 660–665.

[75]Handel, P., Skog, I., Wahlstrom, J., Bonawiede, F., Welch, R., Ohlsson, J., Ohlsson, M., “Insurance telematics: Opportunities and challenges with the smartphone solution”, IEEE Intelligent Transportation Systems Magazine, Vol. 6, No. 4, 2014, str. 57–70.

[76]Marx, S. E., Luck, J. D., Pitla, S. K., Hoy, R. M., “Comparing various hardware/software solutions and conversion methods for Controller Area Network (CAN) bus data collection”, Computers and Electronics in Agriculture, Vol. 128, 2016, str. 141–148.

[77]Wang, Y., Zhao, N., Liu, X., Karaburun, S., Chen, M., Zhu, T., “Identifying big five personality traits through controller area network bus data”, Journal of Advanced Transportation, Vol. 2020, 2020.

[78]Gandomi, A., Haider, M., “Beyond the hype: Big data concepts, methods, and analytics”, International Journal of Information Management, Vol. 35, No. 2, 2015, str. 137–144.

[79]Kiran, M., Murphy, P., Monga, I., Dugan, J., Baveja, S. S., “Lambda architecture for cost-effective batch and speed big data processing”, in Big Data (Big Data), 2015 IEEE International Conference on. IEEE, 2015, str. 2785–2792.

[80]Membrey, P., Chan, K. C., Demchenko, Y., “A disk based stream oriented approach for storing big data”, in 2013 International Conference on Collaboration Technologies and Systems (CTS). IEEE, 2013, str. 56–64.

[81]Amadeo, M., Campolo, C., Molinaro, A., "Information-centric networking for connected vehicles: a survey and future perspectives", IEEE Communications Magazine, Vol. 54, No. 2, 2016, str. 98–104.

[82]Woźniak, P., Valton, R., Fjeld, M., "Volvo single view of vehicle: Building a big data service from scratch in the automotive industry", in Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems, 2015, str. 671–678.

[83]Pinheiro, R. L., Landa-Silva, D., Qu, R., Constantino, A. A., Yanaga, E., "An application programming interface with increased performance for optimisation problems data", Journal of Management Analytics, Vol. 3, No. 4, 2016, str. 305–332.

[84]Wagner, R., Stinnett, J. R., Duarte, M., Baraniuk, R., Johnson, D. B., Ng, T. E., "A network application programming interface for data processing in sensor networks", Submitted to IPSN, 2007.

[85]Ofoeda, J., Boateng, R., Effah, J., "Application programming interface (api) research: A review of the past to inform the future", International Journal of Enterprise Information Systems (IJEIS), Vol. 15, No. 3, 2019, str. 76–95.

[86]Fiore, S., Elia, D., Pires, C. E., Mestre, D. G., Cappiello, C., Vitali, M., Andrade, N., Braz, T., Lezzi, D., Moraes, R. *et al.*, "An integrated big and fast data analytics platform for smart urban transportation management", IEEE Access, Vol. 7, 2019, str. 117 652–117 677.

[87]Zhu, L., Holden, J. R., Gonder, J. D., "Navigation application programming interface route fuel saving opportunity assessment on large-scale real-world travel data for conventional vehicles and hybrid electric vehicles", Transportation Research Record, Vol. 2672, No. 25, 2018, str. 139–149.

[88]Boaz, R. W., Tarico, D., "The atc application programming interface–closing the technology gap", ITE Journal, 2007.

[89]Garcia, J. R. R., van den Belt, E., Bakermans, B., Groen, T., "Blueprint for an application programming interface from transport operator to maas provider (tomp-api)–version dragonfly", 2020.

[90]Fugiglando, U., Massaro, E., Santi, P., Milardo, S., Abida, K., Stahlmann, R., Netter, F., Ratti, C., "Driving behavior analysis through can bus data in an uncontrolled environment", IEEE Transactions on Intelligent Transportation Systems, Vol. 20, No. 2, 2018, str. 737–748.

[91]Uvarov, K., Ponomarev, A., “Driver identification with OBD-II public data”, in 2021 28th Conference of Open Innovations Association (FRUCT). IEEE, Jan. 2021.

[92]Puchalski, A., Komorska, I., “Driving style analysis and driver classification using obd data of a hybrid electric vehicle”, Transport Problems, Vol. 15, 2020.

[93]Martinelli, F., Mercaldo, F., Nardone, V., Orlando, A., Santone, A., “Cluster analysis for driver aggressiveness identification.”, in ICISSP, 2018, str. 562–569.

[94]Choi, S., Kim, J., Kwak, D., Angkititrakul, P., Hansen, J., “Analysis and classification of driver behavior using in-vehicle CAN-bus information”, in Biennial workshop on DSP for in-vehicle and mobile systems, 2007, str. 17–19.

[95]Kwak, B. I., Woo, J., Kim, H. K., “Know your master: Driver profiling-based anti-theft method”, in 2016 14th Annual Conference on Privacy, Security and Trust (PST). IEEE, Dec. 2016.

[96]Alessandrini, A., Filippi, F., Ortenzi, F., “Consumption calculation of vehicles using OBD data”, in 20th International Emission Inventory Conference-" Emission Inventories-Meeting the Challenges Posed by Emerging Global, National, and Regional and Local Air Quality Issues, 2012.

[97]Hilpert, H., Thoroe, L., Schumann, M., “Real-time data collection for product carbon footprints in transportation processes based on OBD2 and smartphones”, in 2011 44th Hawaii International Conference on System Sciences. IEEE, 2011, str. 1–10.

[98]Severtson, W. R. B., Franks, L., Ericson, G., “What is the team data science process?”, Microsoft Azure, 2017.

[99]Wirth, R., Hipp, J., “CRISP-DM: Towards a standard process model for data mining”, in Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining. Springer-Verlag London, UK, 2000, str. 29–39.

[100]Pevec, D., Vdovic, H., Gace, I., Sabolic, M., Babic, J., Podobnik, V., “Distributed data platform for automotive industry: A robust solution for tackling big challenges of big data in transportation science”, in 2019 15th International Conference on Telecommunications (ConTEL). IEEE, 2019, str. 1–8.

[101]McCord, K., Automotive Diagnostic Systems: Understanding OBD I and OBD II. CarTech Inc, 2011.

[102]ELM Electronics. (2021) Elm327, available at: https://www.elmelectronics.com/ic/elm327/

[103]ELETRONICS, E., “ELM327 OBD to RS232 Interpreter”, ELM Electronics Datasheets, 2015.

[104]PEAK Systems. (2021) Pcan-wireless gateway, available at: https://www.peak-system.com/PCAN-Wireless-Gateway.331.0.html?&L=1

[105]TomTom, “TomTom”, available at: https://www.tomtom.com/en_us/ Dec 2020.

[106]Salvo, G., Caruso, L., Scordo, A., “Urban traffic analysis through an uav”, Procedia-Social and Behavioral Sciences, Vol. 111, 2014, str. 1083–1091.

[107]OpenWeather, “OpenWeather global services”, available at: https://openweathermap.org/ Dec 2020.

[108]Freiberger, S., Albrecht, M., Käufl, J., “Reverse engineering technologies for remanufacturing of automotive systems communicating via CAN bus”, Journal of remanufacturing, Vol. 1, No. 1, 2011, str. 6.

[109]RASCO, “RASCO Manufacturer Of Municipal Equipment For Road Maintenance”, available at: https://rasco.hr/en/ Nov 2020.

[110]Vdovic, H., Babic, J., Podobnik, V., “Specialized vehicle can bus simulator: From modelling to validation”, in 2020 5th International Conference on Smart and Sustainable Technologies (SpliTech). IEEE, 2020, str. 1–7.

[111]“Lynx - compact sweeper for the city of tomorrow”, available at: https://lynx.rasco.hr/en/

[112]Senin, P., “Dynamic time warping algorithm review”, Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA, Vol. 855, No. 1-23, 2008, str. 40.

[113]Statista, “Market share of leading mobile operating systems in europe from 2010 to 2019”, available at: https://www.statista.com/statistics/639928/market-share-mobile-operating-systems-eu/ Jan 2020.

[114]Pires, Paulo. (2021) Android OBD Reader, available at: https://github.com/pires/android-obd-reader

[115]Google. (2021) Firebase Crashlytics, available at: https://firebase.google.com/docs/crashlytics

[116]Jogalekar, P., Woodside, M., “Evaluating the scalability of distributed systems”, IEEE Transactions on parallel and distributed systems, Vol. 11, No. 6, 2000, str. 589–603.

[117]Eager, D. L., Sevcik, K. C., “Achieving robustness in distributed database systems”, ACM Transactions on Database Systems (TODS), Vol. 8, No. 3, 1983, str. 354–381.

[118]Jalote, P., Jalote, P., Fault tolerance in distributed systems. PTR Prentice Hall Englewood Cliffs, New Jersey, 1994.

[119]Khan, N., Yaqoob, I., Hashem, I. A. T., Inayat, Z., Ali, M., Kamaleldin, W., Alam, M., Shiraz, M., Gani, A., “Big data: survey, technologies, opportunities, and challenges”, The Scientific World Journal, 2014.

[120]Shreedharan, H., Using flume: flexible, scalable, and reliable data streaming. " O’Reilly Media, Inc.", 2014.

[121]Wang, G., Koshy, J., Subramanian, S., Paramasivam, K., Zadeh, M., Narkhede, N., Rao, J., Kreps, J., Stein, J., “Building a replicated logging system with apache kafka”, Proceedings of the VLDB Endowment, Vol. 8, No. 12, 2015, str. 1654–1655.

[122]Junqueira, F., Reed, B., ZooKeeper: distributed process coordination. " O’Reilly Media, Inc.", 2013.

[123]Boicea, A., Radulescu, F., Agapin, L. I., “Mongodb vs oracle–database comparison”, in 2012 third international conference on emerging intelligent data and web technologies. IEEE, 2012, str. 330–335.

[124]Moniruzzaman, A., Hossain, S. A., “Nosql database: New era of databases for big data analytics-classification, characteristics and comparison”, arXiv preprint arXiv:1307.0191, 2013.

[125]Cockpit, “Cockpit project”, available at: https://cockpit-project.org/

[126]Miller, D., Whitlock, J., Gardiner, M., Ralphson, M., Ratovsky, R., Sarid, U., “OpenAPI Specification”, available at: http://spec.openapis.org/oas/v3.1.0 2021.

[127]Vdovic, H., Babic, J., Podobnik, V., “Eco-Efficient Driving Pattern Evaluation for Sustainable Road Transport Based on Contextually Enriched Automotive Data”, 2021, Manuscript submitted for publication.

[128]Software, S. (2021) Swagger, available at: https://swagger.io/

[129]Vogel, P., Petre, B. (2021) Flask Monitoring Dashboard, available at: https://flask-monitoringdashboard.readthedocs.io/en/latest/

[130]Walker, G. H., Stanton, N. A., Young, M. S., “Hierarchical task analysis of driving: A new research tool”, Contemporary ergonomics, 2001, str. 435–440.

[131]Matsumoto, S., “The effect of acceleration and deceleration information of preceding vehicle group on fuel economy of the following vehicle.”, in VEHITS, 2015, str. 5–10.

[132]Andrieu, C., Saint Pierre, G., “Comparing effects of eco-driving training and simple advices on driving behavior”, Procedia-Social and Behavioral Sciences, Vol. 54, 2012, str. 211–220.

[133]Jakobsen, K., Mouritsen, S. C., Torp, K., “Evaluating eco-driving advice using GPS/CANBus data”, in Proceedings of the 21st ACM SIGSPATIAL international conference on advances in geographic information systems, 2013, str. 44–53.

[134]Tzirakis, E., Zannikos, F., Stournas, S., “Impact of driving style on fuel consumption and exhaust emissions: defensive and aggressive driving style”, in Proceedings of the 10th International Conference on Environmental Science and Technology. Global Network for Environmental Science and Technology (Global-NEST), 2007, str. 1497–1504.

[135]Paviši ć, B., Matiša, T., “Zakon o sigurnosti prometa na cestama”, 2013.

[136]Sammut, C., Webb, G. I., Encyclopedia of machine learning. Springer Science & Business Media, 2011.

[137]Aldred, R., Johnson, R., Jackson, C., Woodcock, J., “How does mode of travel affect risks posed to other road users? an analysis of english road fatality data, incorporating gender and road type”, Injury prevention, Vol. 27, No. 1, 2021, str. 71–76.

[138]IDC, “Revenue from big data and business analytics worldwide from 2015 to 2022”, available at: https://www.statista.com/statistics/551501/worldwide-big-data-business-analytics-revenue/ 2019.

[139]Kinnunen, S.-K., Hanski, J., Marttonen-Arola, S., Kärri, T., “A framework for creating value from fleet data at ecosystem level”, Management Systems in Production Engineering, Vol. 25, No. 3, 2017, str. 163–167.

[140]Killeen, P., Ding, B., Kiringa, I., Yeap, T., “Iot-based predictive maintenance for fleet management”, Procedia Computer Science, Vol. 151, 2019, str. 607–613.

[141]Kim, B., Baek, Y., “Sensor-based extraction approaches of in-vehicle information for driver behavior analysis”, Sensors, Vol. 20, No. 18, 2020, str. 5197.

[142]Yin, X., Li, Z., Shah, S. L., Zhang, L., Wang, C., “Fuel efficiency modeling and prediction for automotive vehicles: A data-driven approach”, in 2015 IEEE International Conference on Systems, Man, and Cybernetics. IEEE, 2015, str. 2527–2532.

[143]Babic, J., Carvalho, A., Ketter, W., Podobnik, V., “Evaluating policies for parking lots handling electric vehicles”, IEEE Access, Vol. 6, 2018, str. 944–961.

List of Abbreviations

ABS . . . . . .Antilock Braking System

API . . . . . .Application Programming Interface

AV . . . . . .Autonomous Vehicle

CAN . . . . . .Controller Area Network

CRISP-DM .Cross-industry Standard Process for Data Mining

CSMA/CD . .Carrier Sense Multiple Access / Collision Detection

CRC . . . . . .Cyclic Redundancy Check

DSRC . . . . . .Dedicated Short Range Communication

DTS . . . . . .Discrete Time Series

DTW . . . . . .Dynamic Time Warping

ECU . . . . . .Electronic Control Unit

EMIS . . . . . .Environmental Management Information System

ESP . . . . . .Electronic Stability Program

EV . . . . . .Electric Vehicle

GPS . . . . . .Global Positioning System

HEV . . . . . .Hybrid Electric Vehicle

ICE . . . . . .Internal Combustion Engine

ICT . . . . . .Information Communications Technology

LIN . . . . . .Local Interconnect Network

MaaS . . . . . .Mobility as a Service

MOST . . . . . .Media Oriented Systems Transport

MVC . . . . . .Model-View-Controller

OBD . . . . . .On-Board Diagnostics

OBU . . . . . .Onboard Unit

PCF . . . . . .Product Carbon Footprint

PID . . . . . .Parameter ID

RPM . . . . . .Revolutions Per Minute

RSSI . . . . . .Received Signal Strength Indication

RSU . . . . . .Roadside Unit

TDM . . . . . .Time Division Multiplex

TDSP . . . . . .Team Data Science Process

TCU . . . . . .Telematics Control Unit

TOMP . . . . . .Transport Operator to MaaS Provider

V2I . . . . . .Vehicle To Infrastructure

V2V . . . . . .Vehicle To Vehicle

VANET . . . . . .Vehicular Ad Hoc Network

WAVE . . . . . .Wireless Access in Vehicular Environments

Biography

Hrvoje Vdović was born in Zagreb in 1994. In 2015, he received his Bachelor's degree in Computing at the Faculty of Electrical Engineering and Computing (FER), University of Zagreb. He continued his education on the second cycle study programme "Information and Communication Technology", where he received his Master's degree in 2017.

In 2018, he joined the Department of Telecommunications at FER, where he works as a Research Associate within the "Social Networking and Computing laboratory (socialLAB)". As a doctoral researcher, he acquired his experience in the R&D of solutions for electric and connected vehicles working on projects "Compact city vacuum cleaner with universal platform for different types of engines and information communication system for operating parameters" and "Development of a multifunctional low-floor vehicle - MUNIVO". Since 2019, he takes part in the realization of the ERASMUS+ project „Innovative Solutions based on Emerging Technologies for Improving Social Inclusion of People with Disabilities (INNOSID)" as the lead developer of several intellectual outputs.

Vdović is a member of IEEE. He was awarded the first (2015) and second (2016) place awards of Ericsson Nikola Tesla for best student paper in the field of ICT. He published 7 conference papers as well as 4 journals papers. He carries out teaching activities in several courses at the Faculty of Electrical Engineering and Computing and the Croatian Defence Academy "Dr. Franjo Tuđman".

List of publications

Journal papers

1. **Vdović, H**., Babić, J., Podobnik, V., "Eco-efficient driving pattern evaluation for sustainable road transport based on contextually enriched automotive data", Journal of cleaner production, 311 (2021), 127564, 12 DOI: 10.1016/j.jclepro.2021.127564
2. **Vdović, H.**, Babić, J., Podobnik, V., "Automotive Software in Connected and Autonomous Electric Vehicles: A Review", IEEE access, 7 (2019), 166365-166379, DOI: 10.1109/access.2019.2953568

3.Dor čec, L., Pevec, D., **Vdović, H.**, Babić, J., Podobnik, V., "How do people value electric vehicle charging service? A gamified survey approach", Journal of cleaner production, 210 (2019), 887-897, DOI: 10.1016/j.jclepro.2018.11.032

4.Grgi ć, D., **Vdović, H.**, Babić, J., Podobnik, V., "CrocodileAgent 2018: Robust agent-based mechanisms for power trading in competitive environments", Computer science and information systems, 16 (2019), 1; 105-129, DOI: 10.2298/csis181010040g

Conference papers (international peer-reviewed and oral presentation)

1.Ga če, I., Pevec, D., **Vdović, H**., Babić, J., Podobnik, V., "Driving style Categorisation based on Unsupervised Learning: a Step towards Sustainable Transportation", Proceedings of 6th International Conference on Smart and Sustainable Technologies (SpliTech 2021), Split, IEEE, 2021. str. 1-6, DOI: 10.23919/SpliTech52315.2021.9566371

2. **Vdović, H**., Babić, J., Podobnik, V., "An Application Programming Interface for Advanced Analytics of Contextually Enriched Automotive Data", Proceedings of 16th International Conference on Telecommunications (ConTEL 2021), Zagreb, Hrvatska, 2021. str. 70-77, DOI: 10.23919/ConTEL52528.2021.9495964
3. **Vdović, H**., Babić, J., Podobnik, V., "Specialized Vehicle CAN Bus Simulator: From Modelling to Validation", Proceedings of the 5th International Conference on Smart and Sustainable Technologies (SpliTech 2020), Split, IEEE, 2020. str. 1-7, DOI: 10.23919/SpliTech49282.2020.9243843

4.Pevec, D., **Vdović, H**., Gače, I., Sabolić, M., Babić, J., Podobnik, V., "Distributed Data Platform for Automotive Industry: A Robust Solution for Tackling Big Challenges of Big Data in Transportation Science", Proceedings of 15th International Conference on Telecommunications (ConTEL 2019), Graz, IEEE, 2019. str. 1-8, DOI: 10.1109/ConTEL.2019.8848542

5.Dor čec, L., Pevec, D., **Vdović, H**., Babić, J., Podobnik, V., "Exploring Willingness to Pay for Electric Vehicle Charging with Gamified Survey", Proceedings of the 3rd International

Conference on Smart and Sustainable Technologies (SpliTech 2018), Split, IEEE, 2018. str. 1-8

6.Pirša, A., Roki ć, L., **Vdović, H**., Vertlberg, L., Žilak, M., Car, Ž., Podobnik, V., "Analysis of ICT-based Assistive Solutions for People with Disabilities", Proceedings of the 14th International Conference on Telecommunications (ConTEL 2017), Zagreb, IEEE, 2017. str. 13-18, DOI: 10.23919/ConTEL.2017.8000013

7.Pirša, A., Stani ć, B., Štracak, L., Todorović, Z., **Vdović, H**., Žilak, M., Vuković, M., Car, Ž., "Front-end solution for enhancing web sites accessibility", Proceedings of the 13th International Conference on Telecommunications, Graz, IEEE, 2015. str. 1-8, DOI: 10.1109/ConTEL.2015.7231202

Životopis

Hrvoje Vdović rođen je u Zagrebu 1994. godine. Svoju prvostupničku diplomu u području računarstva dobio je 2015. godine na Fakultetu elektrotehnike i računarstva (FER) Sveučilišta u Zagrebu. Svoje obrazovanje nastavio je na diplomskom studiju istog fakulteta te je 2017. stekao mag. ing. stupanj iz područja Informacijske i komunikacijske tehnologije (IKT).

Od 2018. radi na Zavodu za telekomunikacije FER-a kao znanstveni suradnik unutar „Laboratorija za društveno umrežavanje i društveno računarstvo (socialLAB)“. Kao doktorski istraživač je stekao iskustvo u istraživanju i razvoju rješenja za električna i umrežena vozila tijekom rada na projektima „Kompaktna gradska vakuumska čistilica s univerzalnom platformom za različite vrste pogona i informacijsko komunikacijskim sustavom upravljanja radnih parametara“ i „Razvoj multifunkcionalnog niskopodnog vozila – MUNIVO“. Od 2019. godine sudjeluje u realizaciji ERASMUS+ projekta „Innovative Solutions based on Emerging Technologies for Improving Social Inclusion of People with Disabilities (INNOSID)“ kao voditelj razvoja nekoliko intelektualnih rezultata.

Vdović je član strukovne udruge IEEE. Dobitnik je prve (2015.) i druge (2016.) nagrade Ericssona Nikole Tesle za najbolji studentski rad u području IKT-a. Objavio je 7 znanstvenih radova u zbornicima konferencija te 4 u časopisima. Provodi nastavne aktivnosti u sklopu nekoliko predmeta na Fakultetu elektrotehnike i računarstva i Hrvatskom vojnom učilištu „Dr. Franjo Tuđman“.