Izrada sustava za online prikupljanje podataka s vozila opremljenih OBD uređajem

Vaiti, Tin

SVEUČILIŠTE U ZAGREBU

FAKULTET PROMETNIH ZNANOSTI

Tin Vaiti

IZRADA SUSTAVA ZA ONLINE PRIKUPLJANJE PODATAKA S VOZILA OPREMLJENIH OBD UREĐAJEM

ZAVRŠNI RAD

Zagreb, 2020.

SVEUŠILIŠTE U ZAGREBU
FAKULTET PROMETNIH ZNANOSTI
ODBOR ZA ZAVRŠNI RAD

Zagreb, 31. ožujka 2020.

Zavod: **Zavod za inteligentne transportne sustave**
Predmet: **Baze podataka**

ZAVRŠNI ZADATAK br. 5620

Pristupnik: **Tin Vaiti (0135250937)**
Studij: Inteligentni transportni sustavi i logistika
Smjer: Inteligentni transportni sustavi

Zadatak: **Izrada sustava za online prikupljanje podataka s vozila opremljenih OBD urebajem**

Opis zadatka:

U radu je potrebno opisati podatke koji se mogu prikupljati s OBD urebaja. Potom je potrebno dizajnirati sustav za prikupljanje podataka iz OBD urebaja ugrabenog u vozilo preko pametnog telefona i slanja podataka na poslužitelj. Potrebno je dizajnirati i implementirati relacijsku bazu podataka na poslužitelju u koju ce se navedeni podaci spremiti, te izraditi mobilnu aplikaciju koja ce slati podatke na poslužitelj i služiti za prikaz podataka s OBD urebaja u realnom vremenu. Kao rezultat rada potrebno je prikazati i objasniti način rada dizajniranog sustava te prikazati i analizirati prikupljene podatke.

Mentor:

Tomislav Erdelic, mag. ing. el. techn. inf.

Predsjednik povjerenstva za završni ispit:

SVEUČILIŠTE U ZAGREBU

FAKULTET PROMETNIH ZNANOSTI

ZAVRŠNI RAD

IZRADA SUSTAVA ZA ONLINE PRIKUPLJANJE PODATAKA S VOZILA OPREMLJENIH OBD UREĐAJEM

DESIGN OF A SYSTEM FOR ONLINE DATA COLLECTION FROM VEHICLES EQUIPPED WITH AN OBD DEVICE

Mentor: Tomislav Erdelić, mag. ing. el. techn. inf.

Student: Tin Vaiti

JMBAG: 0135250937

Zagreb, 2020.

Sažetak

U ovom završnom radu izrađeno je grafičko korisničko sučelje (eng. Graphical User Interface, GUI) na bazi Android sustava pomoću kojega se prikazuju prikupljeni podaci s OBD uređaja ugrađenog u automobil. Prije izrade grafičkog sučelja osposobljen je OBD uređaj čija je glavna zadaća očitavanje parametara s automobila, ali i uspostava komunikacijske veze između razvijene aplikacije i automobila. Parametri koji se prikupljaju su: brzina, potrošnja goriva, okretaji motora, pozicija papučice gasa, napon motora, razina goriva, temperatura ulja u motoru itd. Prikupljeni podaci spremaju se na poslužitelj putem mobilnog uređaja koji ima pristup internetu. Za tu potrebu na poslužitelju je implementiran web servis na koji se šalju podaci i spremaju u relacijsku bazu podataka. Dobivene informacije mogu biti korištene za dobivanje navika i ponašanje vozača u prometu te u mnoge druge svrhe.

Ključne riječi: OBD, Java, Android Studio, XML, obrada i pohrana podataka, GPS

Summary

In this paper, a graphical user interface (GUI) based on the Android system was created, which is used to display collected data from an OBD device mounted in a car. Before creating the graphical interface, the OBD device was enabeled. OBD's main task is to read the parameters from the car, but also to establish a communication between the developed application and the car. The collected parameters are: vehicle speed, fuel consumption rate, engine rpm, throttle position, control module power, fuel level, engine oil temperature, etc. The collected data is stored on the server via a mobile device that has internet access. For this purpose, a web service is implemented on the server to which data is sent and stored in a relational database. The information obtained can be used to observe habits and behavior of drivers in traffic and for many other purposes.

Key words: OBD, Java, Android Studio, XML, data analysis, GPS

Sadržaj

1. Uvod

Tijekom 1970-ih i ranih 1980-ih proizvođači automobila počeli su koristiti elektroničke uređaje za kontrolu funkcija motora i dijagnosticiranje problema motora. Tijekom godina ugrađeni dijagnostički sustavi postajali su sve sofisticiraniji te je sredinom 1990-ih OBD-II (eng. "On-board diagnostics") postao novi standard, koji omogućava potpunu kontrolu motora, nadziranje dijelova šasije, karoserije i dijagnostičku upravljačku mrežu automobila [1]. Razvojem tehnologije OBD-a olakšan je pristup podacima koje prikupljaju automobili. Neki od najčešće očitanih podataka su brzina vozila, okretaji motora, temperatura zraka izvan vozila i sl. OBD-II je najnovija verzija OBD-a koji se zakonski morao implementirati u sva vozila proizvedenim u Americi poslije 1996. godine, a u Europi poslije 2001. godine. Prednosti prikupljanja i analiziranja podataka iz vozila omogućuju dijagnostiku različitih vrsta kvarova vozila, prikazivanje prikupljenih podataka na lako čitljivi način (originalni zapis podataka u vozilu je u heksadekadskom zapisu), detekcija prekoračenja propisane brzine, anomalija u vožnji, kvarova senzora, velike potrošnje goriva i dr.

Ovaj završni rad opisuje postupak izrade Android aplikacije i način prikupljanja podataka iz automobila pomoću OBD-II uređaja, te slanje takvih podataka na poslužitelj. Rad je podijeljen u 6 cjelina:

1. Uvod

2. Opis i način prikupljanja podataka

3. Pohranjivanje podataka na web poslužitelj

4. Grafičko sučelje

5. Rezultati

6. Zaključak

U drugom poglavlju opisan je postupak izrade Android aplikacije, princip rada OBD-II uređaja i način povezivanja aplikacije s OBD-II uređajem.

Treće poglavlje opisuje kako se očitani podaci šalju na web poslužitelj.

Četvrto poglavlje je prezentacija grafičkog sučelja i objašnjavanje svrhe nekih elementa sučelja.

Peto poglavlje sadrži rezultate dobivene prikupljanjem podataka koji se nalaze na web poslužitelju.

U šestom poglavlju dan je zaključak rada.

2. Prikupljanje podataka

Prikupljeni podaci trebaju biti točni i korisni stoga je od velike važnosti izbor sustava s kojim se žele prikupiti podaci, a ujedno je i od velike važnosti odabir odgovarajuće tehnologije za obradu podataka. Tehnologija korištena za prikupljanje podataka u ovom radu je OBD-II, a za slanje podataka na web poslužitelj koristila se razvijena Android aplikacija na pametnom uređaju povezanom na internet.

2.1. Povijest razvoja OBD-II standarda

Zbog velike količine smoga u Los Angelesu, država Kalifornija počela je zahtijevati sustave za kontrolu emisije plinova na modelima automobila koji su proizvedeni 1966. godine. Savezna vlada Sjedinjenih Američkih Država (SAD) proširila je takve sustave kontrole širom zemlje 1968. godine.

Kongres SAD-a je 1970. donio Zakon o čistom zraku i osnovao Agenciju za zaštitu okoliša (eng. Environmental Protection Agency, EPA). Time je započet niz emisijskih standarda i zahtjeva za održavanje vozila kroz duže vremensko razdoblje. Kako bi udovoljili ovim standardima, proizvođači su se okrenuli elektronski kontroliranim sustavima za dovod goriva i paljenje motora. U takvom sustavu, senzori mjere rad motora te na temelju takvih mjerenja centralno računalo upravlja ostalim sustavima u automobilu, a sve s ciljem minimizacije zagađenja. Rana dijagnostika se također vršila na takvim senzorima.

U početku je bilo malo standarda te je i svaki proizvođač imao vlastiti sustav i ispitne signale. 1988. godine Društvo automobilskih inženjera (eng. Society of Automotive Engineers, SAE) postavilo je standardni priključni utikač i postavilo dijagnostičke ispitne signale. EPA je prilagodila većinu svojih standarda iz dijagnostičkih programa i preporuka donesenih od strane SAE. OBD-II je prošireni skup standarda koje je razvio SAE, a usvojili su ga EPA i CARB (eng. California Air Resources Board) s primjenom do 1. siječnja 1996. godine.

Dijagnostički sustavi danas se nalaze u većini automobila i lakim teretnim vozilima. Tijekom godina, ugrađeni dijagnostički sustavi postali su sve sofisticiraniji. OBD-II, novi

standard predstavljen sredinom 90-ih, omogućuje gotovo potpunu kontrolu motora, a nadzire i dijelove šasije, karoserije i dodatnih uređaja, kao i dijagnostičku upravljačku mrežu automobila, [1].

Na slici 1. prikazan je jedan jednostavan OBD-II uređaj korišten u ovome radu koji ima mogućnost spajanja preko Bluetooth veze.

Slika 1. Prikaz OBD-II uređaja, Izvor: [12]

2.1.1. Princip rada OBD-II uređaja

U automobilu postoje razni senzori: senzori za mjerenje vanjske temperature zraka, lambda senzor, senzori tlaka u razvodniku itd. Svaki od ovih senzora šalje signal na centralno računalo automobila, Upravljačkoj jedinici motora (eng. Engine Control Unit, ECU). ECU koristi te informacije za prilagodbu različitih elemenata rada motora, npr. ubrizgavanje goriva u motor.

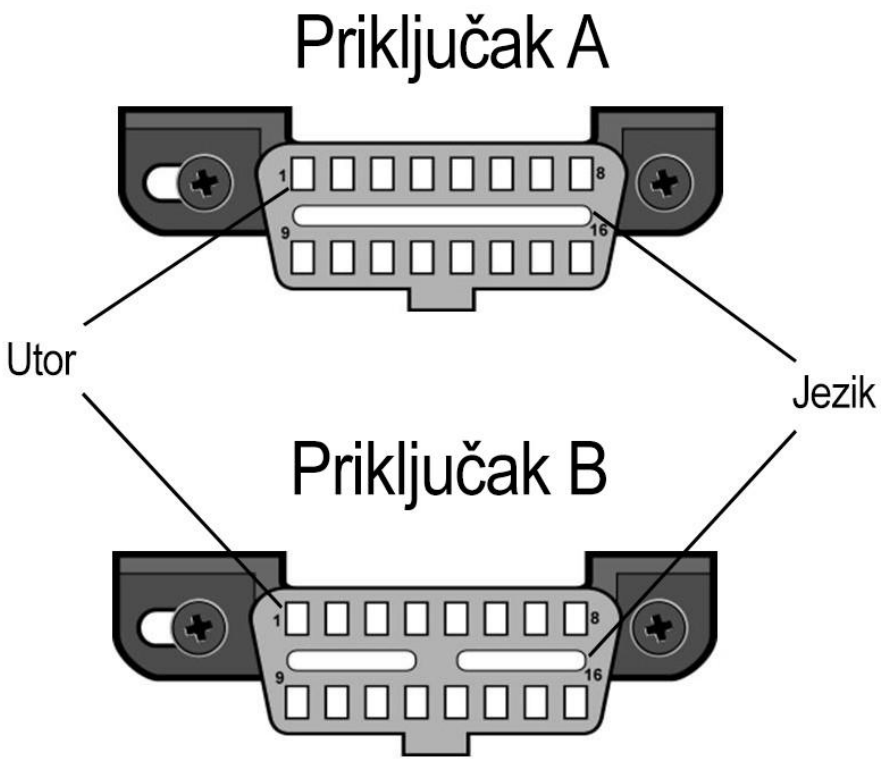
Ako je informacija koju ECU dobije od jednog od svojih senzora izvan standardnog intervala točnosti, sprema se kôd koji se zove Dijagnostički kôd kvara (eng. Diagnostic Trouble Code). Također ECU šalje signal i vozaču kao treptajuće svjetlo odgovarajuće oznake na instrument ploči, [2].

2.1.2. Protokoli OBD-II uređaja

OBD-II sustav općenito ima 5 protokola i dve vrste priključka. Različiti modeli koriste različite protokole. Moguće je da automobil ima priključak tipa A ili priključak tipa B. Oba imaju fizičku razliku u svojim utorima.

Konektori tipa A imaju 16 utora raspoređenih u dva reda. Svaki red ima 8 utora, a po sredini ih dijeli jedan veliki utor zvan „jezik“.

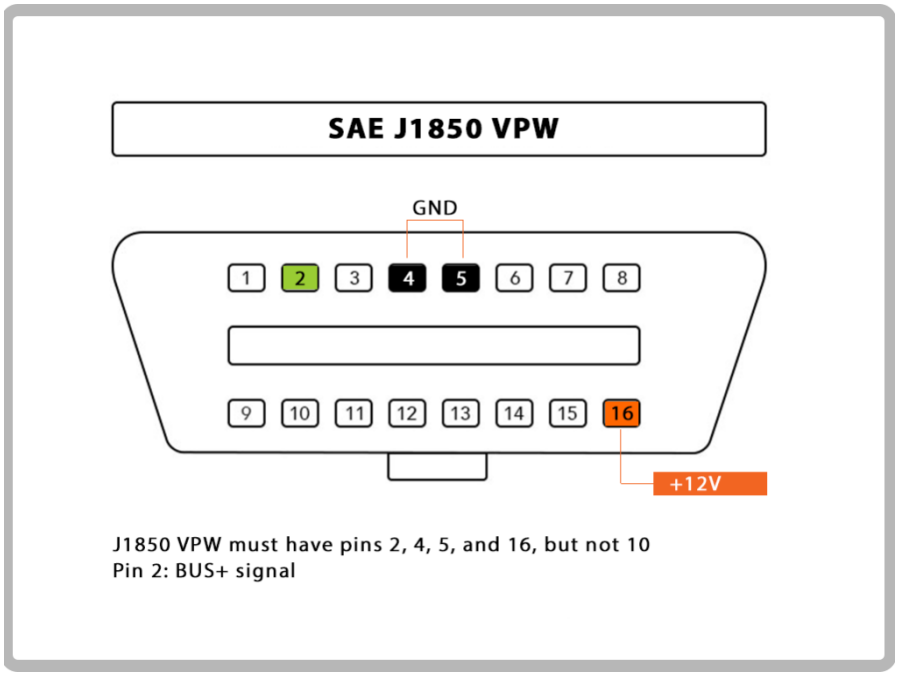
Konektori tipa B također imaju 16 utora, ali jezik im se dijeli na dva dijela, [3].



Slika 2. Prikaz A i B konektora, Izvor: [16]

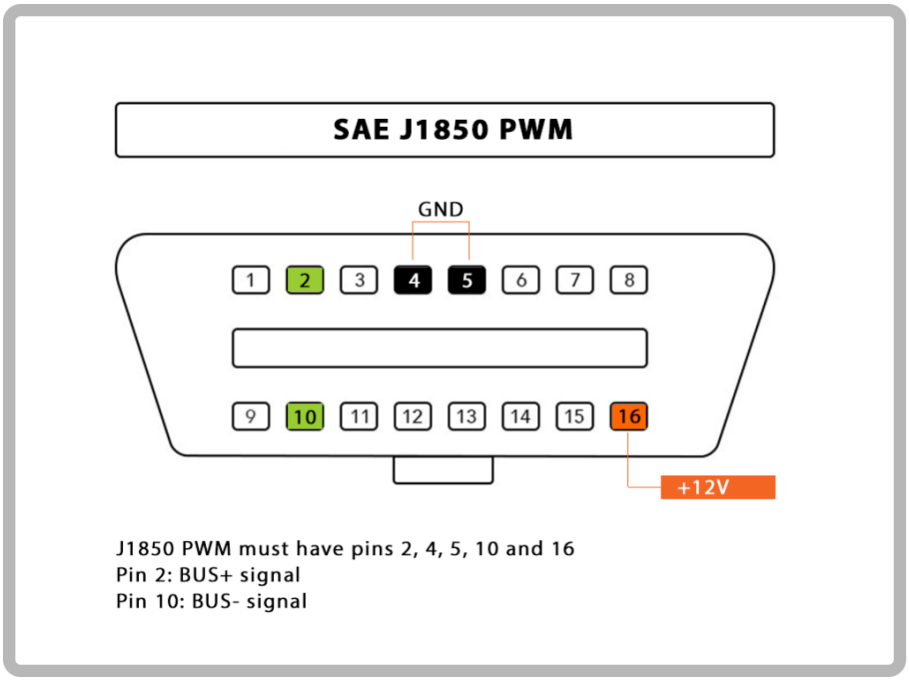
Postoji 5 vrsta OBD-II protokola:

1. SAE J1850 VPW – sadrži utore 2, 4, 5 i 16



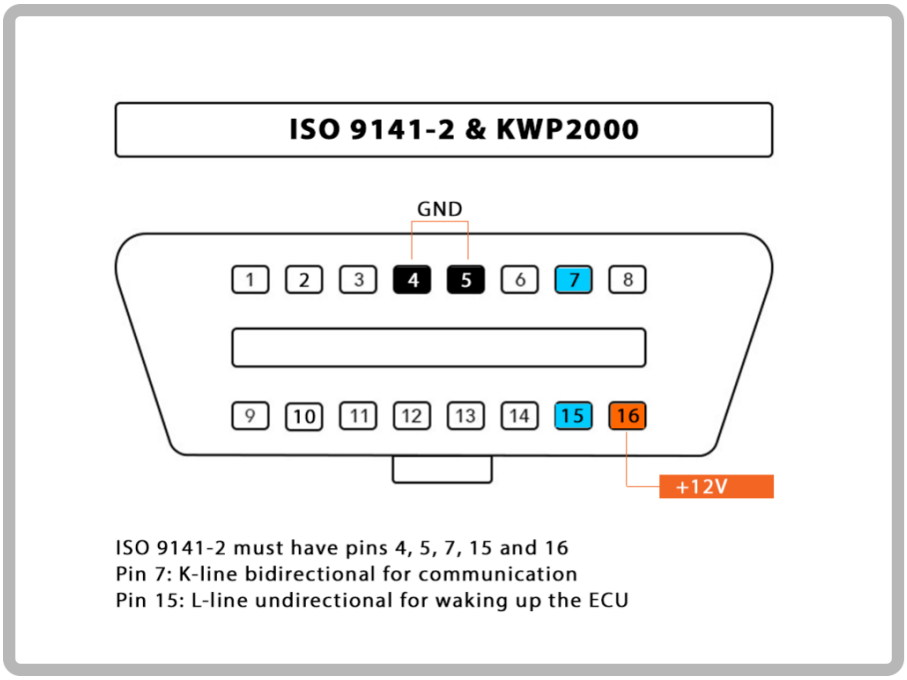
Slika 3. Protokol SAE J1850 VPW, Izvor: [3]

2. SAE J1850 PWM – sadrži utore 2, 4, 5, 10 i 16



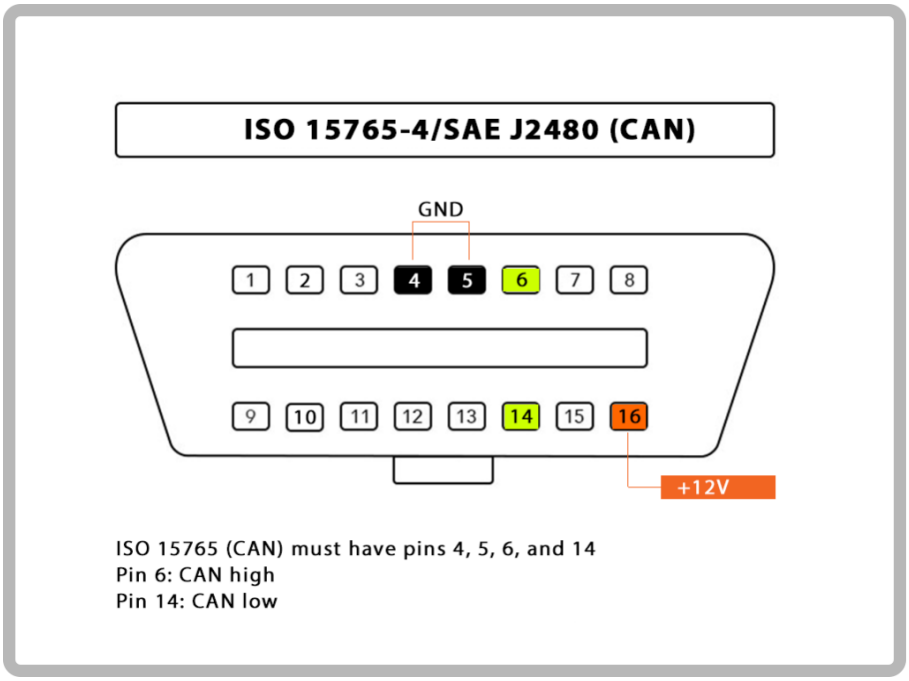
Slika 4. Protokol SAE J1850 PWM, Izvor: [3]

3. ISO 9141-2 – sadrži utore 4, 5, 7, 15 i 16



Slika 5. Protokol ISO 9141-2 i KWP2000, Izvor: [3]

4. ISO 14230 KWP2000 – sadrži utore 7 i 15, te se nalazi u Azijskim automobilima
5. ISO 15765-4/SAE J2480 (CAN) – sadrži utore 4, 5, 6, 14 i 16



Slika 6. Protokol ISO 15765-4/SAE J2480 (CAN), Izvor: [3]

2.1.3. Podaci prikupljeni OBD-II uređajem

OBD-II uređaj može očitati više od 200 podataka iz automobila, pri čemu se upit šalje kao heksadekadska kombinacija dijagnostičke usluge i PID-a (eng. Parameter IDs). Postoji 10 dijagnostičkih usluga opisanih u najnovijem OBD-II standardu SAE J1979, prikazanih u tablici 1.

PID-ovi su kôdovi koji se koriste za traženje podataka s vozila, a koriste se kao dijagnostički alat. SAE standard J1979 definira raznovrsne PID-ove koji se koriste u svakom automobilu, ali proizvođači automobila također mogu definirati dodatne PID-ove specifične za njihova vozila, [3].

Tablica 1. Dijagnostičke usluge

Usluga	Opis
01	Prikaz trenutnih podataka
02	Prikaz zamrznutih podataka
03	Prikaz pohranjenih dijagnostičkih kôdova s greškama
04	Brisanje pohranjenih dijagnostičkih kôdova
05	Rezultati ispitivanja praćenja senzora za kisik
06	Rezultati ispitivanja ostalih komponenti sustava
07	Prikaz dijagnostičkih kôdova za probleme koji su na čekanju
08	Kontrola za upravljanje on-board sustavom
09	Prikaz podataka o vozilu
0A	Trajni dijagnostički kôdovi s problemima

U ovom radu koristili su se podaci očitani sa OBD-II uređaja prikazani tablicom 2.

Tablica 2. Komande i mjerne jedinice korištenih podataka

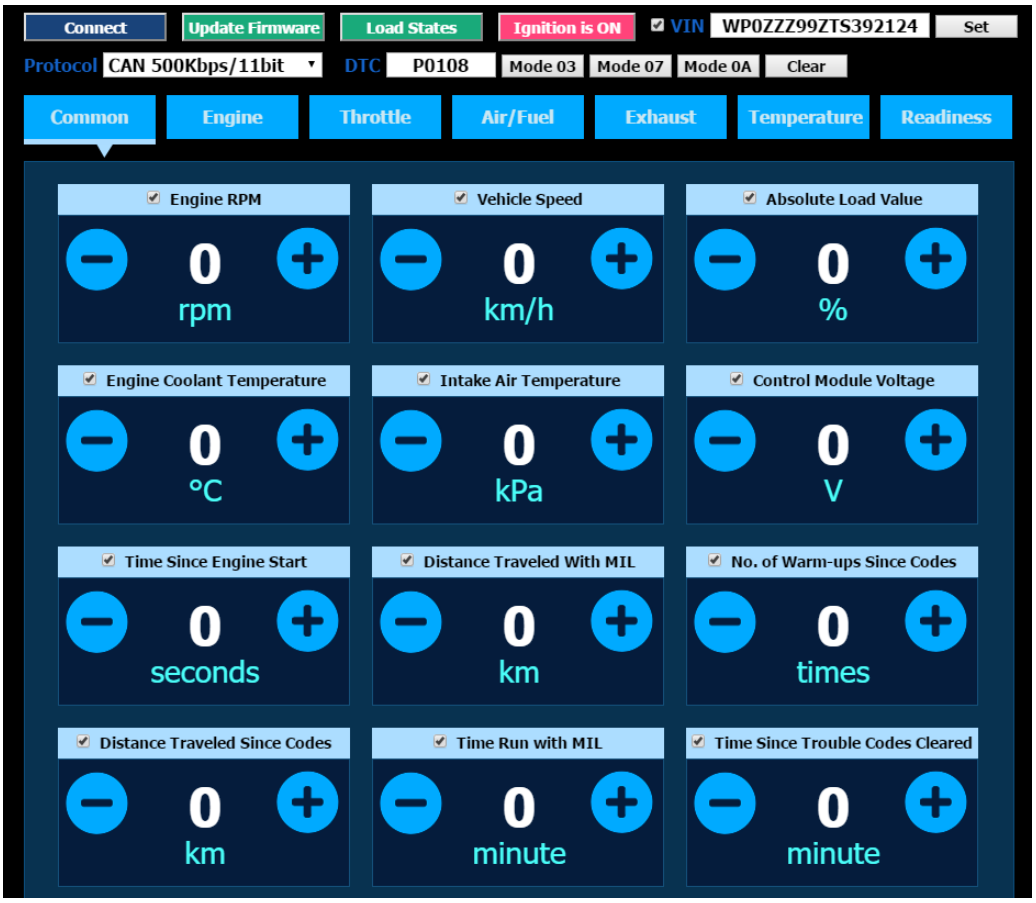
Ime Komande	Mjerna jedinica
Brzina vozila	km/h
Vrijeme proteklo od paljenja motora	s
Intenzitet potrošnje goriva	kPa
Razina goriva u spremniku	%
Praćenje paljenja motora	-
Voltaža upravljačkog modula	V
Pozicija papučice gasa	%
Okretaji motora u minuti	rpm
Temperatura ulja u motoru	°C
Maseni protok zraka	g/s
Apsolutno opterećenje	%
Pritisak goriva	kPa
Barometarski pritisak	kPa
Tlak u šini za gorivo	kPa
Tlak usisnog goriva	kPa
Temperatura rashladne tekućine u motoru	°C
Temperatura zraka	°C

2.2. Testiranje OBD-II uređaja

Za testiranje OBD-II uređaja korišten je Freematics OBD-II emulator (slika 7.) koji podržava simulaciju KWP2000, ISO9141 i CAN protokola. Sastoji se od utora za OBD-II uređaj sa 16 pinova koji je identičan stvarnim vozilima, USB A utor putem kojeg se povezuje na računalo i utor za napajanje. Freematics OBD-II emulator emulira do 6 kôda greške prilikom dijagnostike kao da je stvarno vozilo u kvaru (primjerice upaljena lampica upozorenja za pregled motora na vozilu) i podržava zapis VIN-a (eng. Vehicle Identification Number, VIN).

Slika 7. Prikaz Freematics OBD-II emulatora

Emulator se povezuje s računalom putem USB A kabela i upravlja putem grafičkog korisničkog sučelja prikazanom na slici 8., na kojem se postavljaju varijable vozila, primjerice brzina vozila, okretaji motora ili da li je vozilo upaljeno ili ugašeno. Na emulatoru je moguće postaviti greške koje emuliraju kvar vozila što OBD-II uređaj može detektirati i obavijestiti vozača o kvarovima.



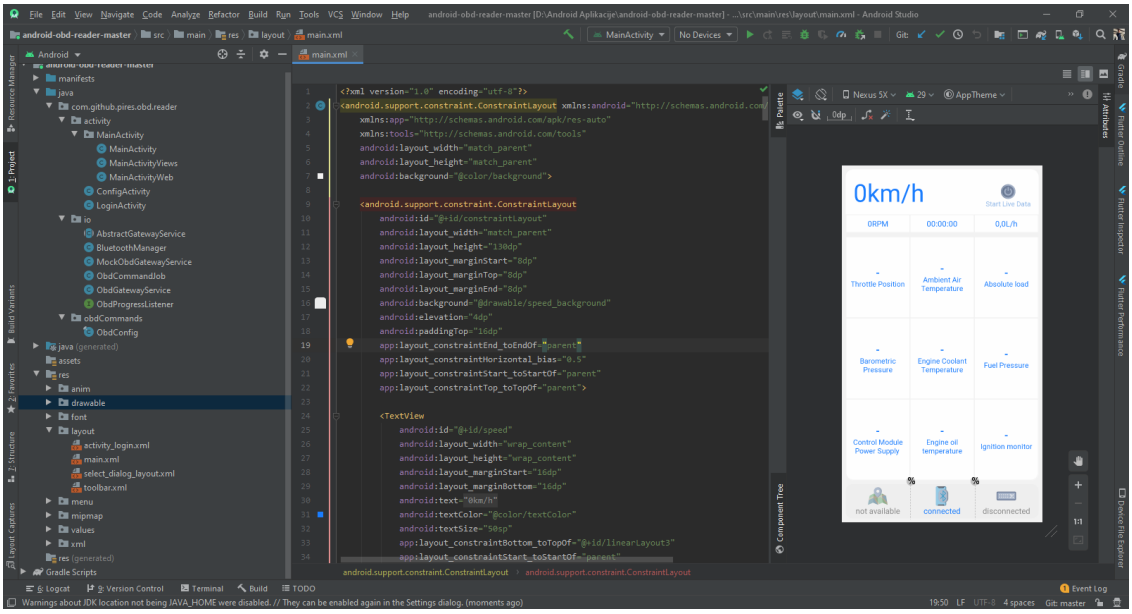
Slika 8. Prikaz najčešćih varijabli u Freematis grafičkom sučelju

2.3. Android Studio

Aplikacija korištena za komuniciranje s OBD-II uređajem napravljena je pomoću alata Android Studio koje je službeno integrirano razvojno okruženje (eng. Integrated Development Environment, IDE) za izradu Android aplikacija koje se temelji na IntelliJ IDEA. Imajući kao temelj IntelliJ uređivač kôda te razne razvojne alate Android studio nudi mnoge mogućnosti koje poboljšavaju produktivnost prilikom izgradnje Android aplikacija primjerice:

- Inteligentni uređivač kôda koji pomaže pri pisanju boljeg kôda, ubrzava rad i povećava produktivnost nudeći napredno dovršavanje i analizu kôda;
- Brzi emulator, bogat značajkama, koji omogućuje pokretanje aplikacija brže nego na stvarnom uređaju i omogućuje prototipiranje i testiranje aplikacije na različitim uređajima

- Podrška za C++ i set razvojnih alata (eng. Native Development Kit, NDK);
- Integracija Firebasea i Google Cloud Platforme;
- Layout Editor baziran na XML (eng. Xtensible Markup Language) jeziku, no Android Studio nudi i „drag and drop“ urednik za pojednostavljenu izradu izgleda aplikacije;
- Analizator android paketa (eng. Android Package, APK) za jednostavnu provjeru sadržaja APK-a;
- Ugrađen JDK (eng. Java Development Kit) koji omogućuje pisanje Java kôda unutar Android Studija;
- Raznovrsni predlošci kôda i integracija s GitHub-om;
- Opsežni alati za testiranje;
- Lint alat čija je glavna funkcija uhvatiti greške prilikom izvođenja, [4].



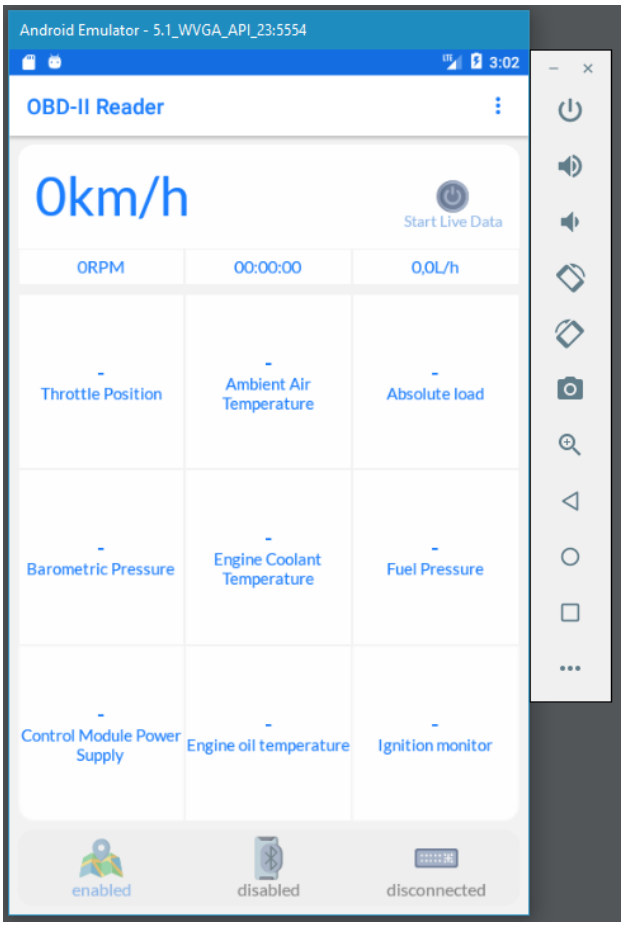
Slika 9. Prikaz Android Studija verzije 3.6.1

Na slici 9. prikazano je sučelje Android Studija verzije 3.6.1 gdje se na lijevoj strani prozora može pronaći struktura projekta u kojem se nalaze klase, skripte, XML datoteke koje omogućuju izradu sučelja aplikacije, animacije koje se također rade u XML datoteci i sl. Na vrhu ispod alatne trake nalaze se svi otvoreni prozori za laku manipulaciju između datoteka. Nadalje na desnoj strani je prikazano sučelje koje je strukturirano po kôdu u XML datoteci i koji se može vidjeti na sredini slike. Na donjoj strani Android Studija nalazi se prozori Build, Terminal, TODO i ostalo, koji se mogu

otvoriti ili zatvoriti te dodati na traku. Svakom prozoru je moguće promijeniti veličinu po želji korisnika te se isto odnosi na temu, veličinu slova, strukturu projekta i ostalo. Izrada Android aplikacija nije nužna u Android Studiju nego ih je moguće također izraditi u drugim razvojnim alatima kao što je Eclipse ili Visual Studio. Android Studio radi na svim operativnim sustavima te je za optimalno korištenje potrebno na računalu imati minimalno 2 GB RAM memorije, 500 MB slobodnog prostora na disku i najmanje 1 GB slobodnog prostora za instalaciju Android SDK-a. Također kako bi Android Studio normalno funkcionirao potrebno je instalirati Java Development Kit (JDK) verzije 7 ili više. Glavne prednosti koje Android Studio pruža su niz gotovih, kvalitetno izrađenih predložaka, lako uređivanje korisničkog sučelja te podrška za izradu aplikacija na svim uređajima koji podržavaju Android operacijski sustav.

2.3.1. Emulator u Android Studiju

Emulator je virtualni uređaj kojemu je glavna uloga testiranje aplikacije bez korištenja fizičkog uređaja, a bazira se na strojnom emulatoru otvorenog kôda QEMU (engl. Open source processor emulator). Emulator omogućava testiranje aplikacije na različitim virtualnim uređajima koji mogu posjedovati različite postavke s različitim verzijama operativnog sustava, memorijom, veličinom ekrana i slično. Glavni nedostatci emulatora su nepodržavanje funkcionalnosti kao što su WiFi, Bluetooth, GPS i sl., [5].



Slika 10. Prikaz emulatora u Android Studiju

Na slici 10. prikazan je izgled emulatora Android uređaja i izgled testne aplikacije kada se pokrene u AVD-u (engl. Android Virtual Device). Desno od emulatora nalazi se alatna traka koja omogućuje funkcionalnosti normalnog uređaja kao što je rotacija ekrana, podešavanje glasnoće, slikanje zaslona i slično. Zbog nedostatka komponenti kao što su Bluetooth i GPS, većina aplikacija se testira na fizičkim uređajima, a kako bi se to omogućilo sve što je potrebno je spojiti kabelom željeni Android uređaj i računalo te instalirati softver potreban za komunikaciju.

2.3.2. Aplikacijske komponente

Glavni elementi od kojih se sastoje Android aplikacije su njene komponente, gdje svaka komponenta na Android platformi ima svoj životni ciklus (slika 11.) koji omogućava rad s aplikacijom na zaseban način. Postoji pet osnovnih komponenti u Android sustavu, a to su aktivnosti, fragmenti, namjere, usluge i pružatelji sadržaja.

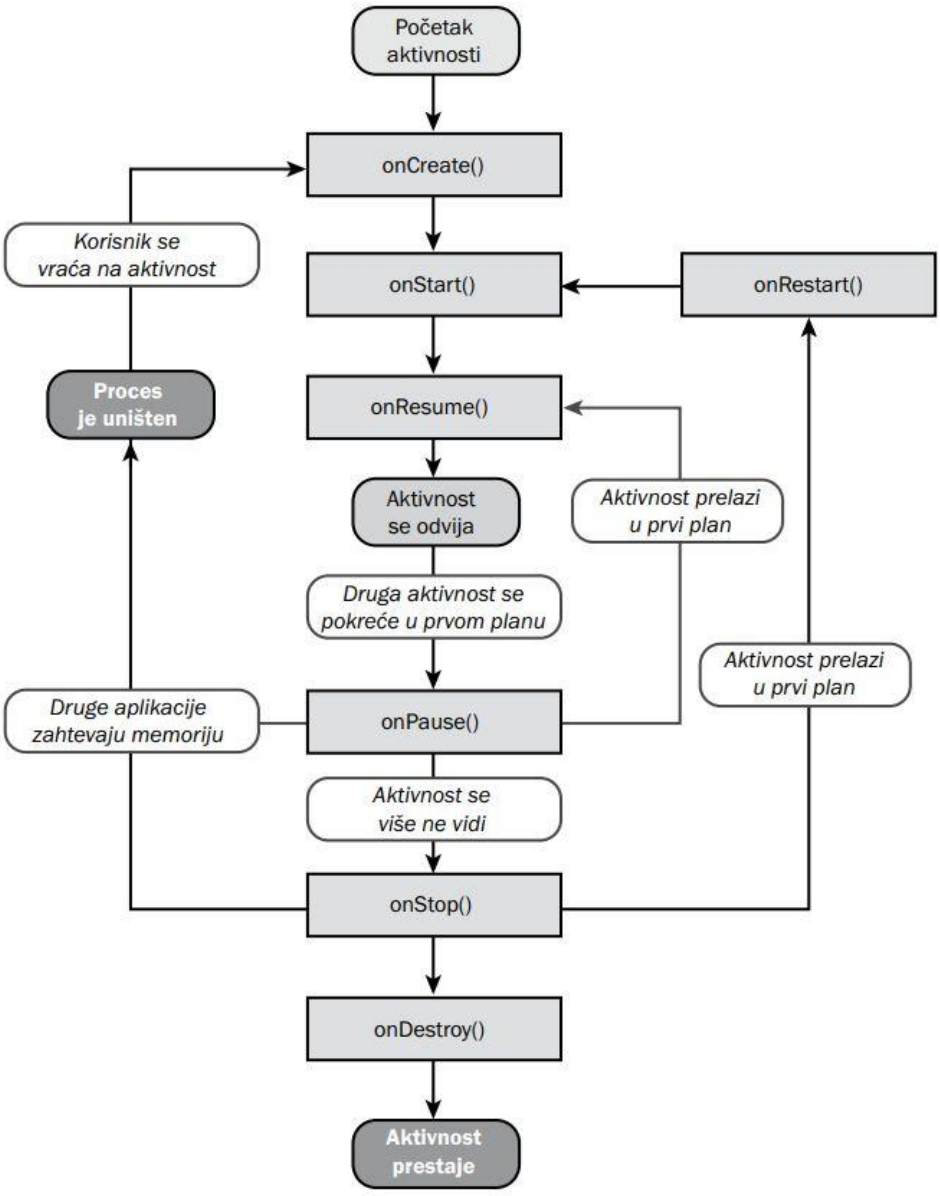
Aktivnost (engl. Activity) je jedan od zaslona korisničkog sučelja na Android aplikaciji. Aplikacije se sastoje od više aktivnosti gdje korisnik može posjećivati druge aktivnosti pritiskom na gumb ili drugim gestama koje je programer aplikacije odredio. Važno je napomenuti da je svaka aktivnost nezavisan dio aplikacije iako rade zajedno i dio su jedne aplikacije. Tipično se u Android aplikacijama nalazi jedna glavna aktivnost kao što je početni zaslon i nekoliko aktivnosti koje su povezane na nju. Aktivnost se definira Java klasom, a korisničko sučelje se dohvaća iz XML datoteke.

Fragment (engl. Fragment) predstavlja dio korisničkog sučelja koje se nalazi u aktivnosti, a može se opisati kao manja aktivnost koja ima svoj vlastiti životni ciklus, ali se za razliku od klasične aktivnosti ne može prikazati zasebno. Glavne značajke fragmenta su da se sučelje može lakše prilagoditi uređaju, može se dinamički uklanjati i dodavati, ima svoj vlastiti životni ciklus, može se jednostavno ponovno koristiti i mogu se inkapsulirati dijelovi korisničkog sučelja. Kada se aktivnost pauzira ili uništi tada će se isto desiti i fragmentu. Fragment je kao i aktivnost također definiran određenom java klasom, a korisničko sučelje se učitava iz određene XML datoteke.

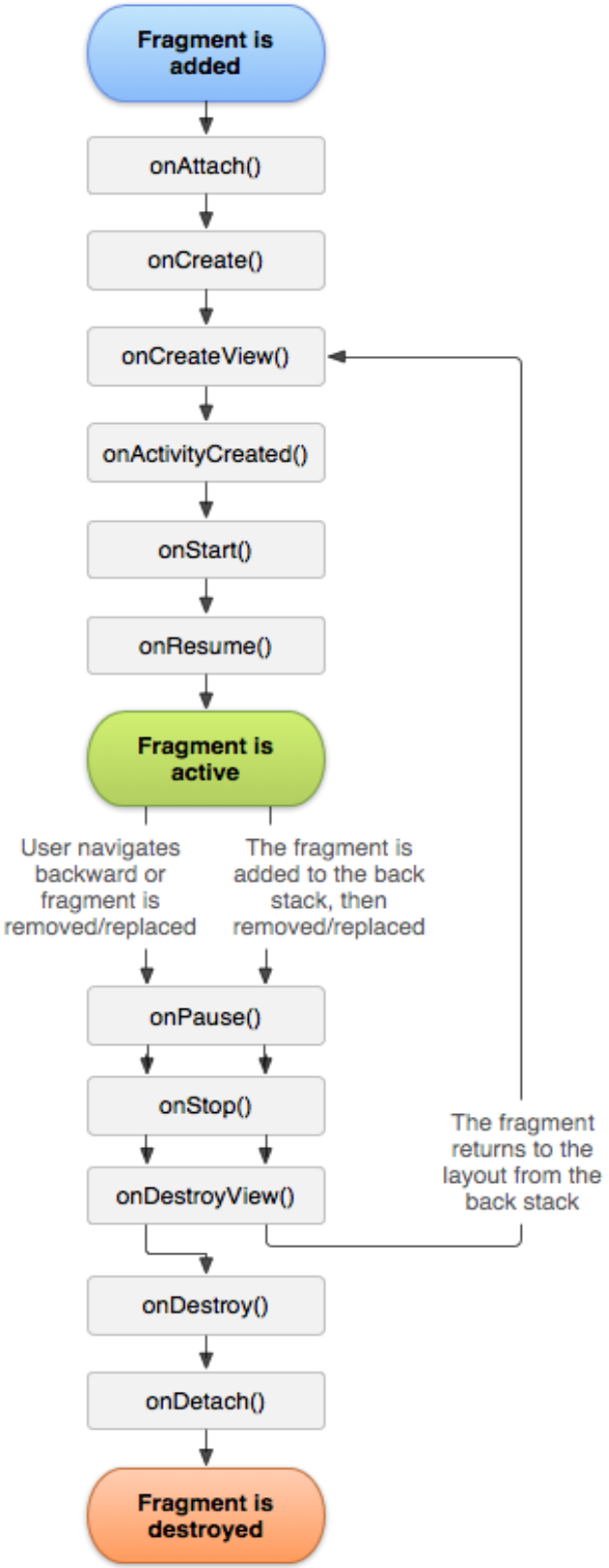
Namjere (engl. Intents) su paketi poruka koje se šalju između glavnih blokova aplikacija. Glavna funkcija im je pokretanje aktivnosti, te navođenje uslugama da se pokrenu ili zaustave. Rad namjeri je asinkron što znači da rade u pozadini i nesmetano šalju kôd s porukama. Namjere se dijele na eksplicitne kod kojih pošiljatelj jasno govori koja komponenta prima poruku i implicitne kod kojih pošiljatelj zadaje vrstu primatelja.

Usluge ili servisi (engl. Services) obavljaju funkcije u pozadini aplikacije te nemaju svoje korisničko sučelje, ali imaju mogućnost obavljanja istih poslova kao i aktivnosti no bez prikazivanja toga što su napravili. Korisne su za radnje koje se obavljaju u određeno vrijeme neovisno o tome što je prikazano na zaslonu npr. slušanje glazbe u pozadini dok je otvorena druga aplikacija, [5].

Pružatelji sadržaja (engl. Content Providers) izvršavaju funkciju izmjene podataka između aplikacija. Android sustav radi na principu da svaku aplikaciju izvršava u posebnom i izoliranom okruženju gdje su podaci tih aplikacija potpuno izolirani od podataka drugih. Takvo odvajanje korisničkog sučelja i pohrane podataka nudi veću fleksibilnost sustava, [5].



Slika 11. Životni ciklus aktivnosti u Android okruženju, Izvor: [14]



Slika 12. Životni ciklus fragmenta u Android okruženju, Izvor: [13]

3. Pohranjivanje podataka na web poslužitelj

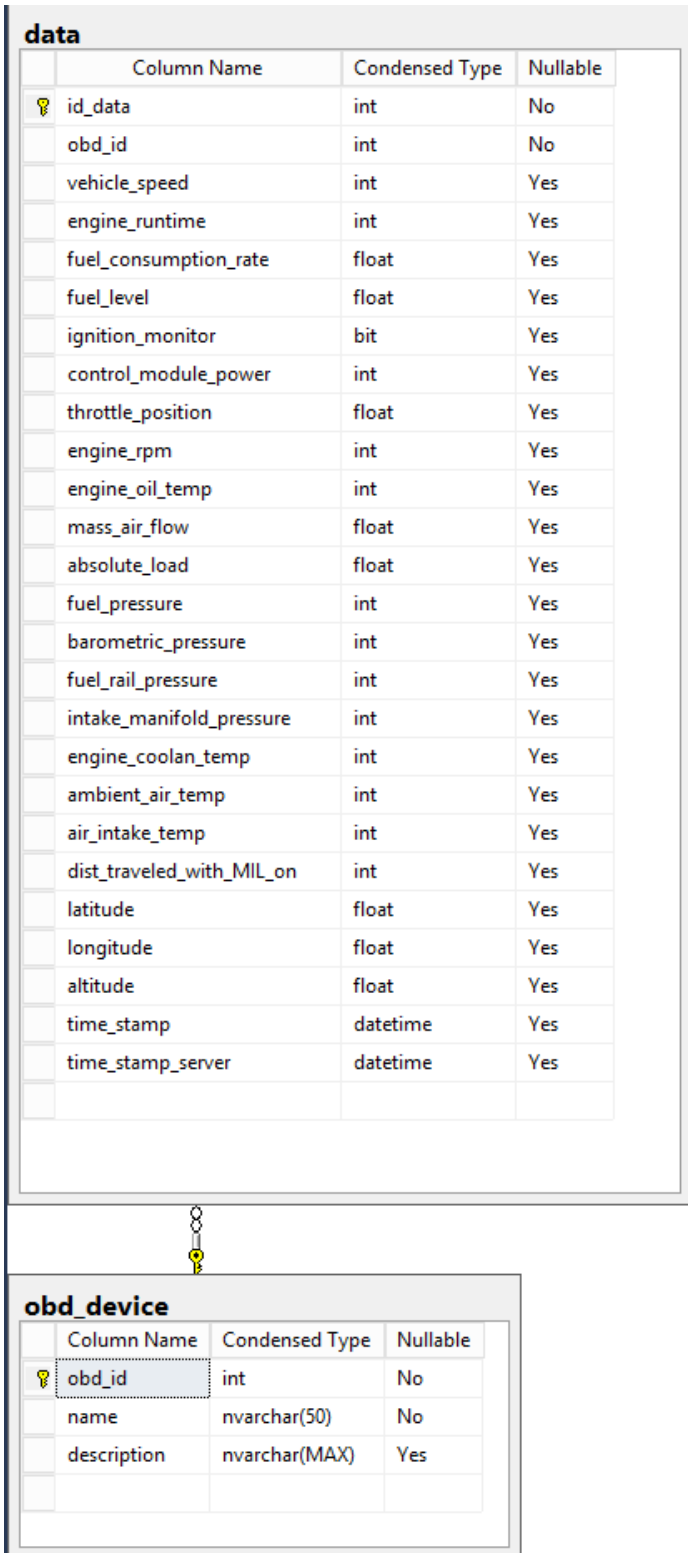
Prikupljanje podataka je jedan od najvažnijih dijelova u realizaciji projekta ili u istraživanju jer podaci trebaju biti točni, korisni i konzistentni stoga je od velike važnosti odabir sustava ili senzora kojima će se prikupljati podaci. Odabir takvog sustava ovisi o tipu podataka koji će biti koristan za određenu vrstu istraživanja. Web servis koji je korišten u ovom projektu izrađen je pomoću C# programskog jezika i podaci su spremljeni na računalu.

3.1. Web poslužitelj

Na web poslužitelj šalju se samo određeni podaci prikupljeni OBD-II uređajem te dodano trenutna lokacija koja se dobiva putem mobilnog uređaja, vremenski trenutak kada je podatak prikupljen na mobilnom uređaju i vrijeme kada je podatak zaprimljen na web poslužitelj. Kako bi se podaci uspješno spremili na web poslužitelj potrebno je najprije odrediti naziv za svaki prikupljeni podatak, zatim je za svaki prikupljeni podatak odrediti kakav je tip podataka (integer, bit, float i dr.). Na kraju se treba odrediti da li prikupljeni podatak može biti prazan (null). Kako bi se moglo prikupljati više podataka s različitih OBD-II uređaja uvodi se dodatna komponenta koja označuje svaki OBD-II uređaj zasebnim identifikatorom, nazivom i opisom, gdje opis nije obavezan.

Na poslužitelju je napravljena relacijska baza podataka pod nazivom *obd_data*, koja sadrži dvije tablice: *data* i *obd_device.* Iz dijagrama baze podataka na slici 13. vidljivo je da su tablica *data* sadrži mnoštvo podataka s OBD uređaja te dodatno sadrži latitude, longitude, altitude, time_stamp i time_stamp_server što su vrijednosti koje ne sakuplja OBD-II uređaj. Za prikupljanje geografske širine, dužine i visine koristio se GPS prijemnik u mobilnom uređaju. Podatak time_stamp je vremenski period kada je podatak prikupljen na mobilnom uređaju, a time_stamp_server je vremenski period kada je podatak zaprimljen na web poslužitelj. Tipovi podataka u tablici određeni na temelju pregleda sirovih podataka s OBD uređaja i standarda. Tablica *obd_device* sadrži samo tri stupca: *obd_id*, *name* i *description* koja suže za spremanje podataka o svakom pojedinom OBD uređaju koji se koristi, budući da se istovremeno mogu primati

podaci s velikog broja različitih OBD uređaja. Vrijednost *obd_id* je primarni ključ i jedinstvena, dok je *name* dodijeljeni naziv tome uređaju. Također, veza između tablica je 1:N, što znači da je 1 obd uređaj može imati više zapisa podatka, i prema tome u tablici *data* pojavljuje se strani ključ *obd_id*.



Slika 13. Dijagram baze podataka

Baza podataka pisana je u SQL-u (eng. Structured Query Language, SQL). Svaki očitani podatak koji je poslan putem Android aplikacije umetnut je u bazu podataka INSERT (slika 14.) naredbom koja se sastoji od imena tablica gdje će podaci biti pohranjeni, imena stupca i vrijednosti svakog stupca koji se zapisuju pomoću VALUES naredbe.

```
INSERT [dbo].[data] ([id_data], [obd_id], [vehicle_speed], [engine_runtime], [fuel_consumption_rate], [fuel_level],
[ignition_monitor], [control_module_power], [throttle_position], [engine_rpm], [engine_oil_temp], [mass_air_flow],
[absolute_load], [fuel_pressure], [barometric_pressure], [fuel_rail_pressure], [intake_manifold_pressure],
[engine_coolan_temp], [ambient_air_temp], [air_intake_temp], [dist_traveled_with_MIL_on], [latitude], [longitude],
[altitude], [time_stamp], [time_stamp_server])
VALUES
   (
      11725, 2, 0, 591, NULL, 43.92157, N'ON', 13, 22.745098, 1322, 54, 23.91, NULL, NULL, 99, 35340, 84, 83, 37, 48,
0, 16.0643, 45.8095, 167.1921292245388, CAST(N'2019-07-01 15:24:23.000' AS DateTime), CAST(N'2019-07-01 15:23:56.597'
AS DateTime)
   )
```

Slika 14. Prikaz umetanja podataka pomoću SQL-a

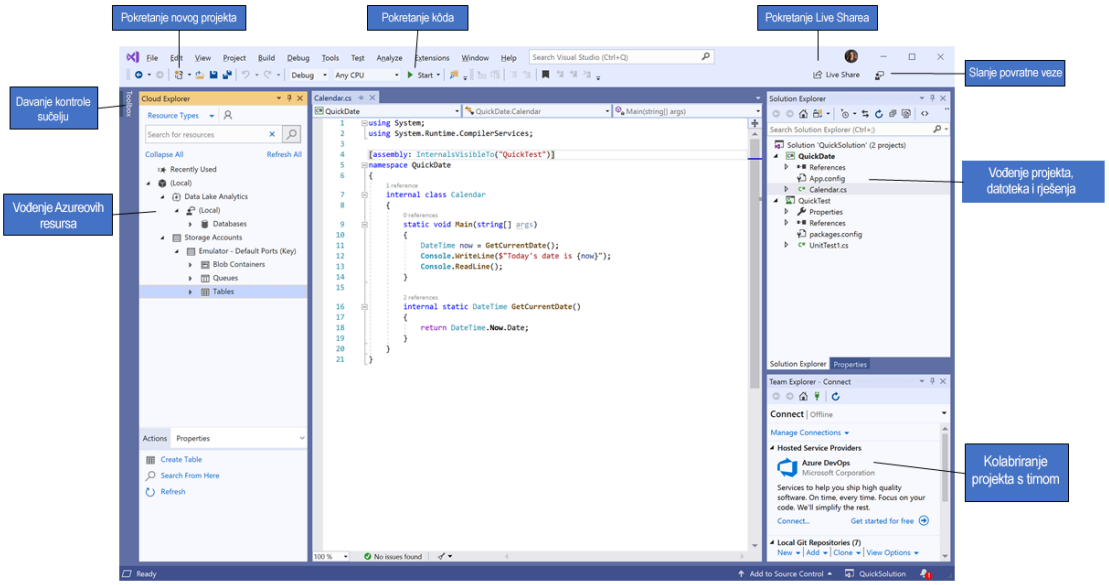
3.2. Web servis

Web servis rađen je pomoću Visual Studija (slika 15.) što je integrirano razvojno okruženje (eng. Integrated Development Environment, IDE) koje se koristi za uređivanje, uklanjanje pogrešaka i izgradnju kôda i izradu raznovrsnih aplikacija.

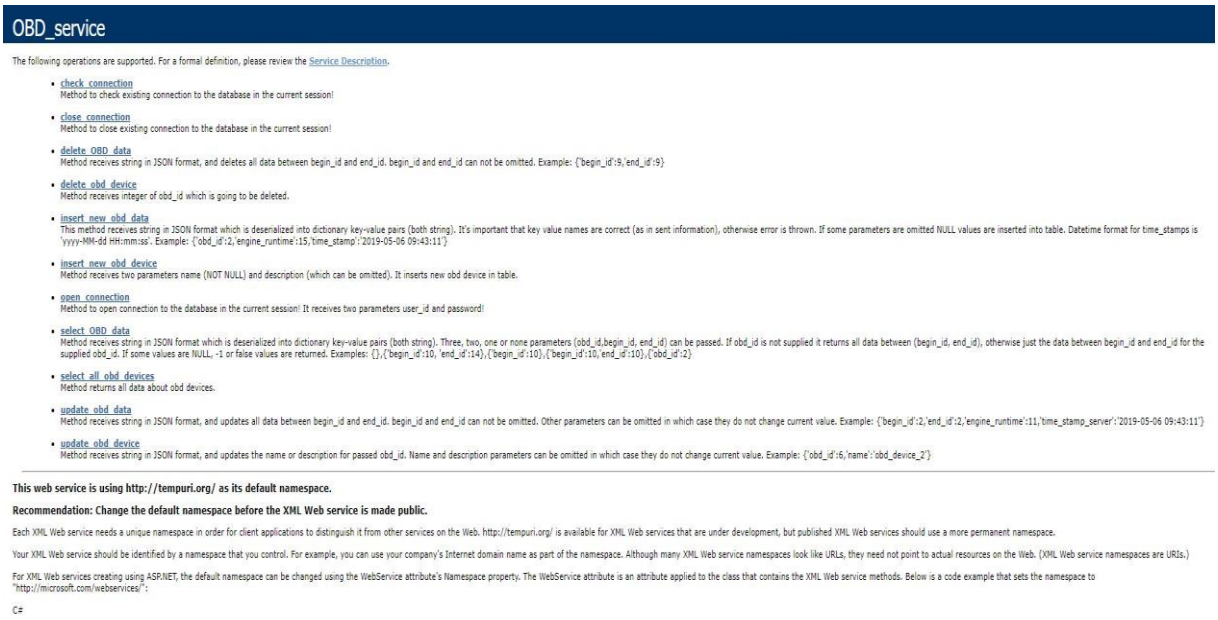
Na web servisu prikazanom na slici 16, omogućene su sljedeće funkcije gdje svaka obavlja drugačiju radnju:

- check connection – provjerava da li je uspješno uspostavljena konekcija u trenutnoj sesiji;
- close connection – zatvara konekciju u trenutnoj sesiji;
- delete OBD data – briše sve prikupljene podatke u rasponu koji je određen;
- delete OBD device – briše određeni uređaj sa web poslužitelja;
- insert new OBD data – umeće nove podatke u obliku JSON-a;
- insert new OBD device – stvara novi uređaj na web poslužitelju;
- open connection – otvara novu konekciju u trenutnoj sesiji;

- select OBD data – dohvaća sve podatke u rasponu koji je određen;
- select all OBD devices – dohvaća sve OBD-II uređaje koji se nalaze na web poslužitelju;
- update OBD data – ažurira željene podatke u željenom rasponu;
- update OBD device – ažurira podatke vezane za OBD-II uređaj, [6].



Slika 15. Prikaz Visual Studija s otvorenim projektom i nekoliko ključnih alata, Izvor: [15]



Slika 16. Prikaz sučelja gdje se izvršavaju radnje web servisa

3.3. Komunikacija aplikacije i web poslužitelja

Za povezivanje i slanje podataka između aplikacije i web poslužitelj koristio se HTTP (Hypertext Transfer Protocol) koji omogućava razgovaranje sa poslužiteljem pomoću sljedećih metoda:

- GET – zahtjeva dohvat podataka;
- HEAD – dohvaća samo headere zahtjeva;
- POST – zahtjeva promjenu ili umetanje podataka;
- PUT – ima istu funkciju kao i POST samo pozivanjem uzastopno više puta nema nikakvih posljedica;
- DELETE – briše željene podatke;
- CONNECT – uspostavlja konekciju sa web poslužiteljem;
- OPTIONS – opisuje način povezivanja sa željenim podacima;
- TRACE – testira put do željenog podatka;
- PATCH – koristi se za parcijalne modifikacije podataka, [7].

U aplikaciji za uspostavu konekcije se koristio OkHttp library koji olakšava cjelokupan proces komunikacije.

Prvi korak za uspješno povezivanje je prijava na web poslužitelj putem posebnog korisničkog imena i zaporke kako bi se uspostavila konekcija i otvorila sesija.

```
OkHttpClient client = new OkHttpClient.Builder()
        .cookieJar(new JavaNetCookieJar(cookieManager))
        .build();

String url =
"http://sordito.fpz.hr/OBD_service/OBD_service.asmx/open_connection?user_id=" +
username + "&password=" + password;
Request request = new Request.Builder()
        .url(url)
        .build();

client.newCall(request).enqueue(new Callback() {
    @Override
    public void onFailure(@NotNull Call call, @NotNull IOException e) {
        e.printStackTrace();
    }

    @Override
    public void onResponse(@NotNull Call call, @NotNull Response response) throws
IOException {
        String myResponse = Objects.requireNonNull(response.body()).string();
        cookie = response.headers().get("Set-Cookie");
        System.out.println("Cookie: " + cookie);
        XmlToJson jsonResponse = new XmlToJson.Builder(myResponse).build();
        System.out.println("Open Connection:\n" +
jsonResponse.toFormattedString());

        if (jsonResponse.toString().contains("true")) {
            checkHttpConnection();
        } else {
            try {
                JSONObject mainObject = new
JSONObject(jsonResponse.toJson().toString());
                JSONObject webResponseObject =
mainObject.getJSONObject("WebResponse");
                String errorMessage = webResponseObject.getString("ErrorMessage");
                LoginActivity.this.runOnUiThread(() -> callToast(errorMessage,
true));
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }
});
```

Slika 17. Prikaz kôda za prijavu korisnika na web poslužitelj

Programski kôd na slici 17. uz pomoć OkHttp-a otvara klijent putem kojeg će se obaviti poziv na web poslužitelj i otvara se mjesto gdje će se spremiti kolačići. Priprema se String zvan url u kojeg se stavlja korisničko ime i zaporka te se upotpunjen sprema u zahtjev za poziv gdje će se dobiti povratna informacija je li je veza uspješno uspostavljena ili ne. Ako je veza uspješno uspostavljena u onResponse metodi

moguće je dobiti podatke o kolačićima koji će poslije služiti za održavanje sesije. Nakon što se uspješno korisnik prijavi i otvori se sesija omogućeno je nesmetano slanje podataka.

```
private void sendDataToWeb() {
    SharedPreferences pref = PreferenceManager
            .getDefaultSharedPreferences(MainActivity.this);
    obdID = pref.getString("web_service_obd_device", "");

    if (cookiePreference != null) {
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        String timeStamp = ",'time_stamp':'" + dateFormat.format(new Date()) +
"'";
        String finalUrl = httpUrl + resultString + timeStamp + "}";

        OkHttpClient client = new OkHttpClient();
        Request request = new Request.Builder()
                .header("Cookie", cookiePreference)
                .url(finalUrl)
                .build();

        client.newCall(request).enqueue(new Callback() {
            @Override
            public void onFailure(@NotNull Call call, @NotNull IOException e) {
                e.printStackTrace();
            }

            @Override
            public void onResponse(@NotNull Call call, @NotNull Response response)
throws IOException {
                String myResponse = Objects.requireNonNull(res-
ponse.body()).string();
                XmlToJson jsonResponse = new XmlToJson.Builder(myRes-
ponse).build();

                System.out.println("Sent data: " + finalUrl);
                sendData = true;

                if (jsonResponse.toString().contains("false")) {
                    try {
                        JSONObject mainObject = new JSONObject(jsonResponse.toJ-
son().toString());
                        JSONObject webResponseObject = mainObject.getJSONOb-
ject("WebResponse");
                        String errorMessage = webResponseObject.getString("Error-
Message");
                        System.out.println("Sent data error: " + errorMessage);
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
            }
        });
    } else MainActivity.this.runOnUiThread(() -> callToast("Error", true));
}
```

Slika 18. Prikaz kôda za slanje prikupljenih podataka na web poslužitelj

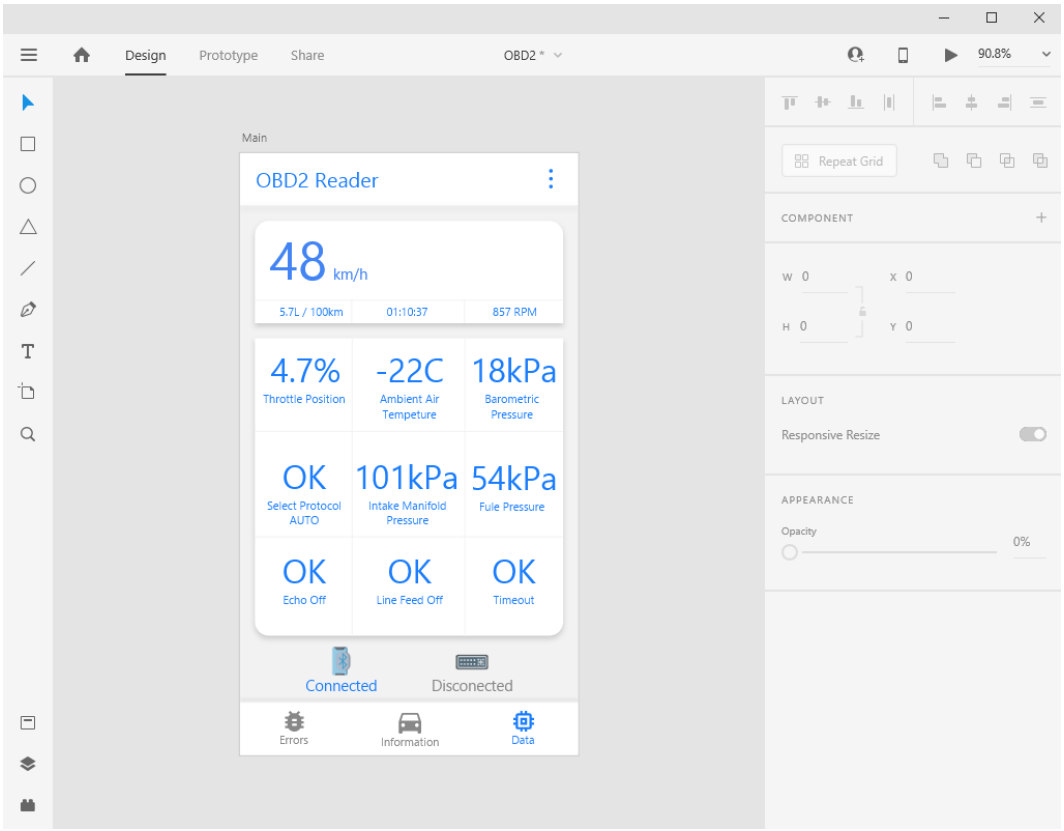
Programski kôd na slici 18. služi za slanje podataka koje mobilni uređaj prikupi na web poslužitelj. Za uspješno poslane podatke potrebno je imati točne podatke OBD-II identifikatora, kolačiće kako bi sesija bila održana i prikupljeni podaci sa OBD-II uređaja koji moraju biti korektno formatirani. U kôdu putem varijable SharedPreferences iz druge se klase dobivaju spremljene informacije o trenutnom OBD-II identifikatoru koji se koristi. Nadalje formatira se lokalno vrijeme koje se šalje za usporedbu sa vremenom na web poslužitelju te se sve to šalje na web poslužitelj gdje se podaci pohranjuju i dobiva se povratna informacija, da li se dogodila greška tijekom slanja ili su podaci uspješno poslani.

4. Grafičko sučelje

Izgled aplikacije napravljen je u adobe xd-u gdje su se realizirali prikazi podataka i struktura aplikacije. Koristeći realizaciju u adobe XD-u izrađuje se dizajn aplikacije u Android studiju putem XML programskog jezika.

4.1. Izrada dizajna u Adobe XD-u

Adobe XD je vektorski alat za oblikovanje korisničkog sučelja namijenjen izradi web i mobilnih aplikacija, razvijen od strane Adobe Inc. Dostupan je za macOS i Windows, te postoje verzije za iOS i Android koje pomažu u pregledavanju rezultata rada izravno na mobilnom uređaju [8].



Slika 19. Prikaz sučelja Adobe XD alata

Na slici 19. prikazano je sučelje Adobe XD alata gdje se na lijevoj strani može pronaći alatna traka s komponentama za crtanje osnovnih oblika kao što su kvadrati, kružnice, ravne crte, tekst i slično. U centru se nalazi prostor gdje se crta dizajn tako da se odabere veličina ekrana na kojem će biti izrađen dizajn i u tim okvirima se stavljaju komponente i oblici. Na desnoj strani sučelja nalaze se dodatne opcije za oblikovanje pojedinih komponenti, odabir boje, zaobljenost rubova, veličina i sl.

Glavna zadaća izrade dizajna u takvom programu je jednostavnije predočenje povezanosti funkcija i funkcionalnosti aplikacije. Uz to izrada dizajna stvara olakšanu sliku kako će se koja funkcija aplikacije izraditi.

4.1. Komponente za izradu dizajna aplikacije u Android Studiju

Izrada sučelja aplikacije u Android studiju razlikuje se od vektorskih alata tako da se dizajn izrađuje putem XML programskog jezika. XML je jezik za označavanje podataka, a stvoren je kao standardni način kôdiranja podataka u internetskim aplikacijama te je osjetljiv na veliko i malo slovo, zahtjeva da se svaka oznaka pravilno zatvori i da se pazi na razmake, [9].

4.1.1. Korištene komponente za izradu aplikacije

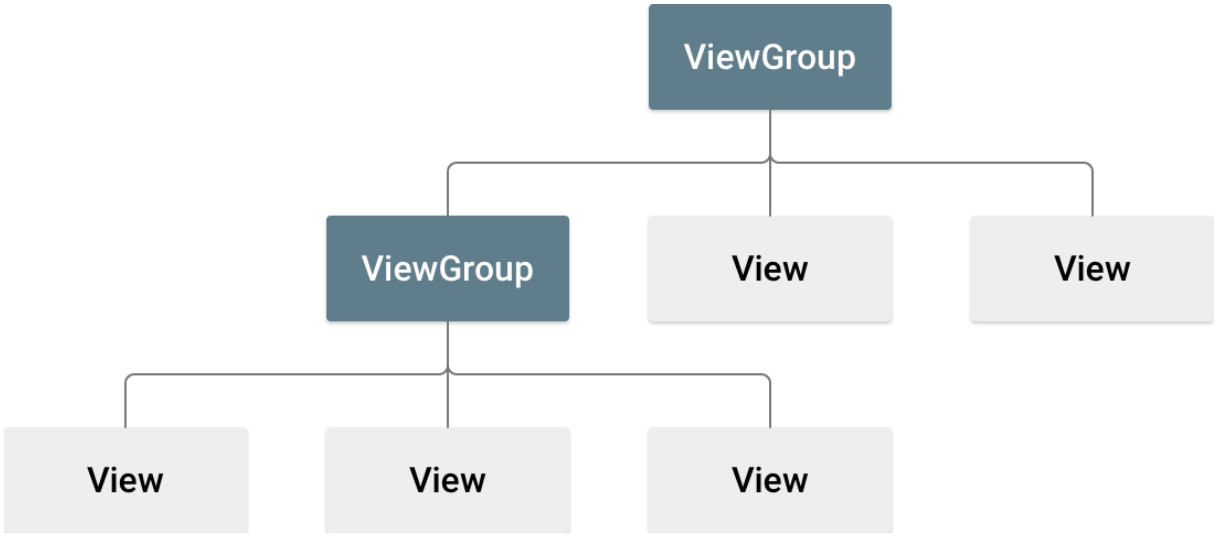
Postoje raznovrsne komponente koje je moguće koristiti za izradu aplikacije u Android Studiju, ali za izradu aplikacije u ovome radu korištene su:

- TextView – služi za prikaz teksta korisniku koji se programskim putem može promijeniti, nadograditi ili urediti;
- Button – element korisničkog sučelja koji korisnik može dodirnuti ili pritisnuti kako bi napravio nekakvu radnju i sastoji se od teksta ili ikone (ili teksta i ikone) koja priopćuje što se događa kada korisnik pritisne gumb;
- ImageView – služi za prikaz resursa slike, na primjer Bitmap ili Drawable resursa, a ImageView se također često koristi za primjenu nijansi na slici i upravljanje omjerom slike;

- TextInputLayout – raspored koji omotava TextInputEditText, EditText ili prikaz plutajuće naljepnice kada je savjet sakriven dok korisnik unosi tekst; omogućuje i prikaz grešaka prilikom krivog unosa, pomoćnog teksta, brojača karaktera, prikaz ili sakrivanje zaporki i slično;
- TextInputEditText – podsustav TextInputLayouta koji ima istu funkciju kao EditText čija je funkcija uređivanje i unos teksta;
- Toolbar – alatna traka na vrhu aplikacije.

4.1.2. Vrste rasporeda i pogleda za izradu dizajna

Raspored definira strukturu korisničkog sučelja u aplikaciji kao što je to aktivnost. Svi elementi u rasporedu izgrađeni su korištenjem hijerarhije objekata pogleda i grupe pogleda. Pogled obično crta nešto što korisnik može vidjeti i vršiti nekakvu interakciju s njim. Dok je grupa pogleda nevidljivi spremnik koji definira strukturu izgleda za pogled i ostale objekte u grupi pogleda, kao što je prikazano na slici 20, [10].



Slika 20. Hijerarhija pogleda što definira dizajn rasporeda, Izvor: [10]

Objekti pogleda obično se nazivaju "widgeti" i mogu biti jedan od mnogih podrazreda, kao što su gumb ili polje za upis teksta. Objekti grupe pogleda obično se nazivaju "rasporedi" te mogu biti jedan od mnogih tipova koji pružaju drugačiju strukturu rasporeda. Postoje raznovrsni rasporedi no glavni i najkorišteniji su:

- Linear Layout – raspored koji podsustave organizira u jedan vodoravni ili okomiti red, te stvara pomičnu traku ako duljina prozora prelazi duljinu zaslona;

Slika 21. Slikovni prikaz Linear Layouta, Izvor: [10]

- Relative Layout – omogućuje određivanje mjesta podređenih objekata u odnosu jedno na drugo (podsustav A lijevo od podsustava B) ili nad sustavom (poravnano prema vrhu sustava);

Slika 22. Slikovni prikaz Relative Layouta, Izvor: [10]

- Web View – glavna zadaća je prikazati web preglednik.

Slika 23. Slikovni prikaz Web Viewa, Izvor: [10]

4.1.3. Raspored pomoću ograničenja

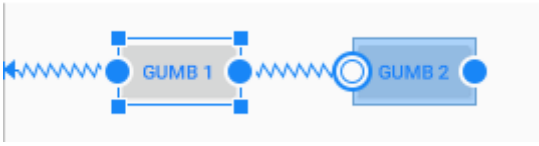
Constraint Layout je najnoviji tip rasporeda koji je korišten za izradu dizajna aplikacije u ovome radu. On omogućuje stvaranje velikih i složenih rasporeda s hijerarhijom ravnog pogleda (bez ugniježđenih grupa pogleda). Sličan je RelativeLayout-u po tome što su svi prikazi postavljeni prema odnosima između pogleda podsustava i sustava izgleda, ali je fleksibilniji od RelativeLayout-a i lakši za upotrebu s uređivačem izgleda za Android Studio. Jednostavnost uređivanja rasporeda koji nudi ConstraintLayout pomoću „drag and drop“ funkcija znatno olakšava izradu dizajna jer se ne treba u potpunosti raditi pomoću XML naredbi, [11].

Jedna od raznih pogodnosti koje nudi ConstraintLayout za razliku od drugih rasporeda je automatska prilagodba za različite veličine zaslona što se uspijeva pomoću ograničenja koja se postavljaju na svim komponentama koje se koriste kao što su gumb ili polje za pisanje. Postoje tri različita tipa ograničenja:

- ograničenje na sustav – postavlja se ograničenje na bočni dio sustava, kao primjer na slici 24. gdje je desna strana gumba je povezana sa desnim rubom sustava u kojem se nalazi gumb, te se udaljenost od ruba može mijenjati marginama;

Slika 24. Horizontalno ograničenje sustava

- ograničenje na podsustav – moguće je povezivanje i određivanje ograničenja podsustava kao što je prikazano na slici 25. s dva gumba koja su spojena horizontalnim ograničenjima i udaljenost se također određuje marginama između elemenata;

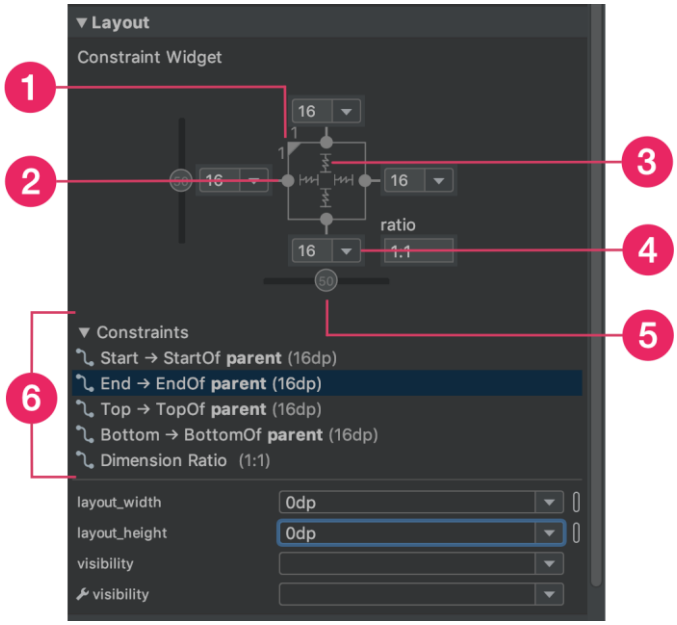


Slika 25. Horizontalno ograničenje podsustava

- ograničenje po osnovnoj liniji – poravnaju se osnovne linije teksta jednog pogleda s drugim kao što je prikazano na slici 26. gdje se poravnala osnovna linija gumba 1 sa osnovnom linijom gumba 2.

Slika 26. Horizontalno ograničenje komponenti

Ograničenja zasebnih elemenata i njihove margine se mogu se izmjenjivati putem grafičkog sučelja (slika 27.) koji olakšava cjelokupan proces bez korištenja XML kôda.



Slika 27. Sučelje za promjenu osnovnih komponenti elementa, Izvor: [11]

Opis opcija sučelja na slici 27 prikazan je u nastavku:

1. Promjena omjera veličine elementa
2. Pritiskom na taj dio briše se ograničenje
3. Odabir tipa za visinu ili dužinu
4. Promjena veličina margina
5. Promjena omjera ograničenja
6. Uređivač za zasebna ograničenja

4.2. Izrada i opis dizajna aplikacije u Android studiju

Aplikacija izrađena za ovaj rad sastoji se od tri aktivnosti, a to su početni zaslon, zaslon za prijavu korisnika i zaslon za podešavanje postavki.

4.2.1. Početni zaslon

Početni zaslon sastoji se od nekoliko osnovnih komponenti, glavnog rasporeda u kojem su smještene sve komponente i pozadine jer se proteže preko cijelog zaslona. Na glavnom rasporedu nalaze se tri dijela koji svaki ima svoju svrhu.

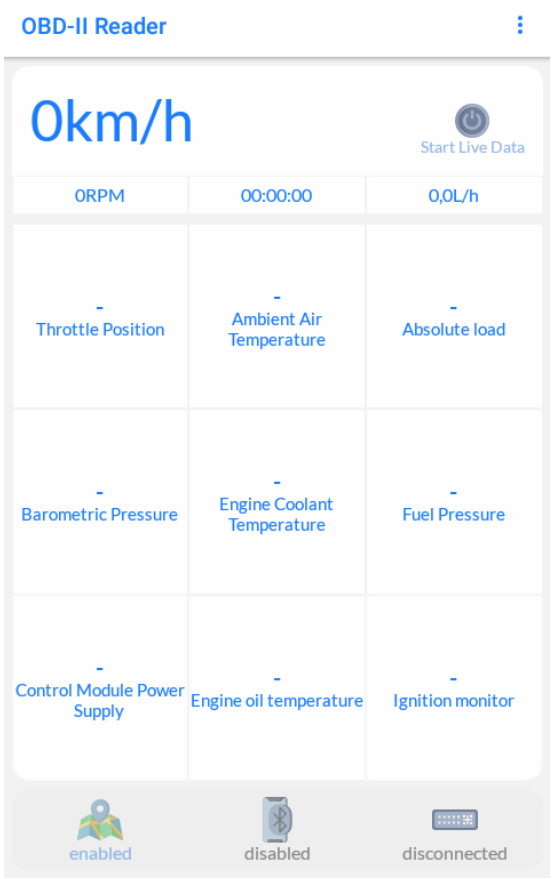
Prvi dio jest alatna traka koja sadrži ime aplikacije i gumb koji otvara prozor za prijavu, odjavu, pristup postavkama i provjeru konekcije.

Slika 28. Prikaz funkcije alatne trake

Drugi dio na gornjem dijelu zaslona (slika 29.) ispod alatne trake služi za prikaz najvažnijih podataka koji se ne mogu promijeniti, a to su brzina kojom se vozilo kreće, okretaji motora, vrijeme trajanja prikupljanja podataka i potrošnja goriva automobila. Svi prikazani podaci se automatski i stvarno-vremenski ažuriraju. Također se još i na desnoj strani nalazi gumb čija je svrha pokretanje prikupljanja podataka i slanja na poslužitelj.

Treći dio u sredini zaslona je sastavljen od tablice koja sadrži devet podataka koji se također prikazuju i ažuriraju stvarno-vremenski. Pritiskom na jednu od ćelija otvara se prozor sa svim podacima koji se mogu očitati s OBD-II uređaja i korisnik može odabrati koje podatke želi prikazati u kojoj ćeliji.

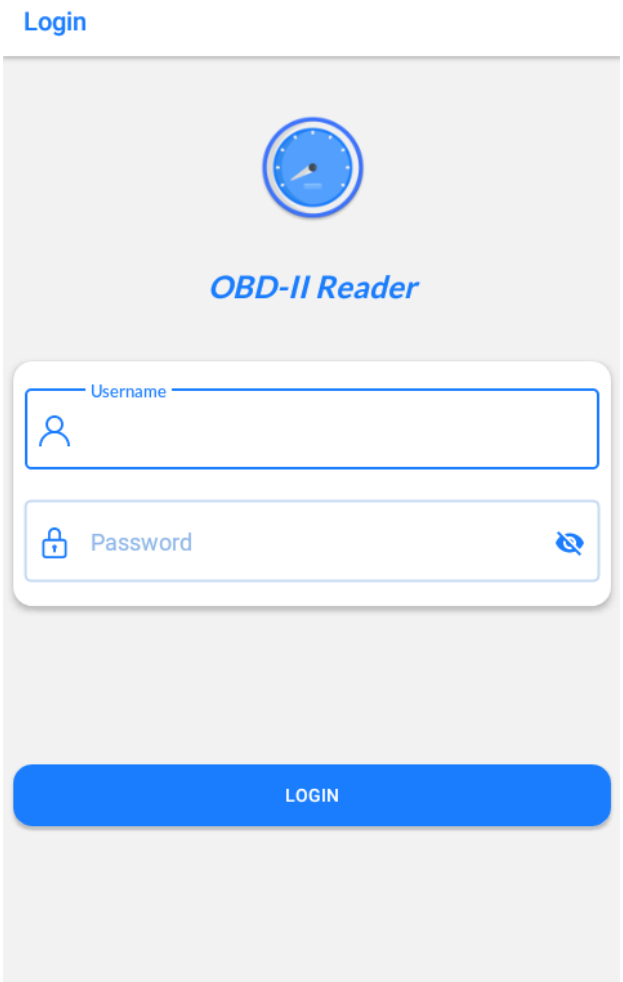
Četvrti dio služi kako bi se obavijestilo korisnika o tome da li je mobilni uređaj spojen i prikuplja li GPS podatke, da li je spojen s OBD-II uređajem putem Bluetootha i da li mobilni uređaj prikuplja podatke sa OBD-II uređaja.



Slika 29. Prikaz početnog zaslona aplikacije

4.2.2. Zaslon za prijavu

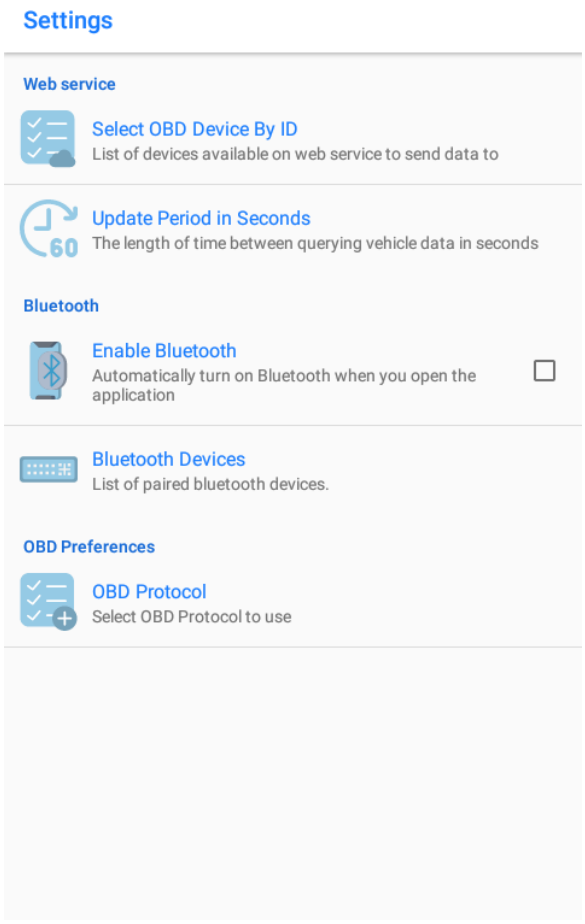
Zaslon za prijavu (slika 30.) je jednostavan zaslon koji je od velike važnosti. Sastoji se od alatne trake pri vrhu zaslona, logo i ime aplikacije za popunjavanje praznog prostora, prostora gdje korisnik upisuje korisničko ime i zaporku kako bi se uspješno povezao sa poslužiteljem i kako bi se uspješno slali podaci na njega, te se na dnu zaslona još nalazi gumb koji ako su podaci ispravni prijavljuje korisnika i otvara početni zaslon.



Slika 30. Prikaz zaslona za prijavu

4.2.3. Zaslon za podešavanje postavka

Zaslon za podešavanje postavki (slika 31.) na vrhu također ima alatnu traku na kojoj piše ime zaslona kako bi korisnik znao na kojem dijelu aplikacije se nalazi. Nadalje su postavke podijeljene u tri dijela, prvi dio su postavke vezane za poslužitelj, drugi dio za Bluetooth i zadnji dio za reference OBD-II uređaja. U postavkama se može odabrati na koji ID na poslužitelju se podaci šalju, koliko brzo da se ti prikupljeni podaci šalju na poslužitelj, automatsko spajanje Bluetootha na OBD-II uređaj, lista pogodnih OBD-II uređaja za spajanje s mobilnim uređajem te odabir protokola.



Slika 31. Prikaz zaslona za podešavanje postavki

5. Rezultati

Prilikom testiranja aplikacije na stvarnom vozilu prikupljeno je sveukupno 11231 podataka koji su uspješno bili očitani i poslani na web poslužitelj. Za primjer prikaza funkcionalnosti podataka odabrana su mjerenja od identifikatora 5130 do 7130 gdje su iscrtane pozicije kretanja vozila na karti pomoću GPS podataka (slika 34.). Podaci su bili prikupljeni u osobnom vozilu Opel Insignia na području Zagreba 25.6.2019. godine u vremenskom periodu od 13 sati i 35 minuta do 18 sati i 12 minuta.

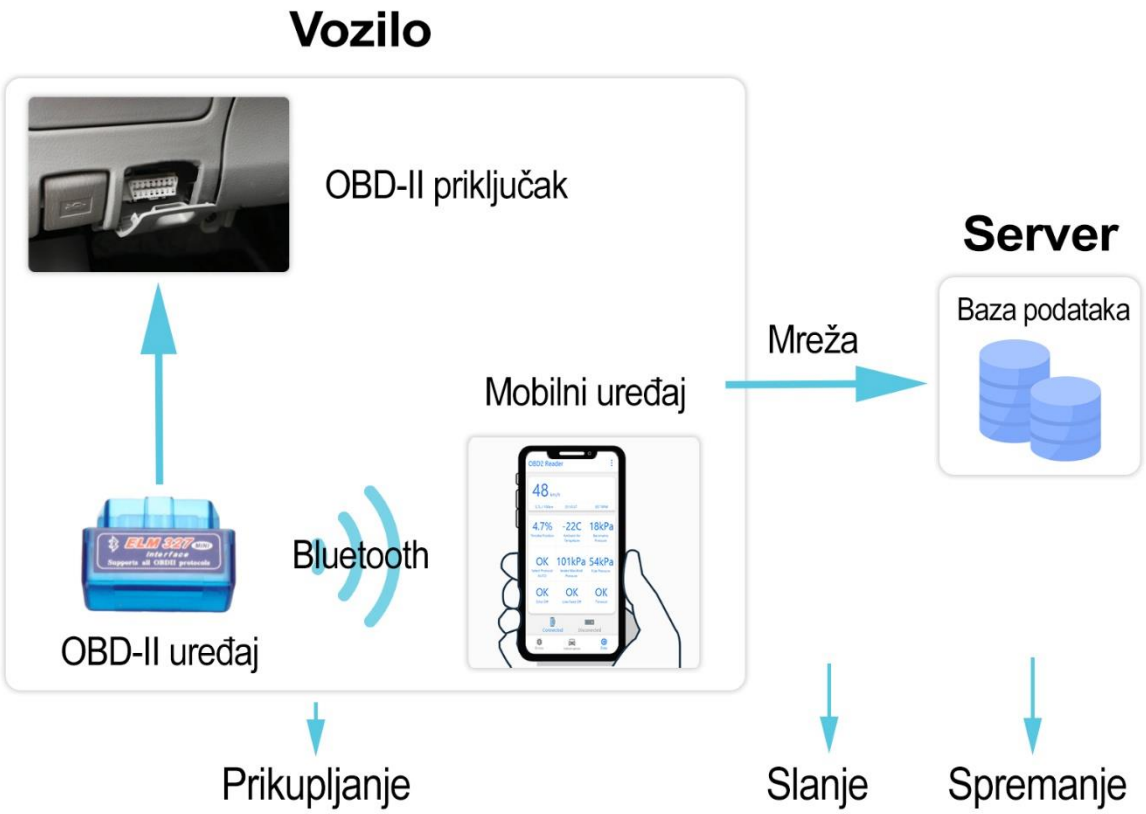
5.1. Implementacija sustava

Sustav implementacije se sastoji od tri dijela (slika 32).

Prvi dio je prikupljanje podataka s vozila. U vozilu se pronalazi OBD-II priključak koji se najčešće nalazi ispod volana u posebno označenom prostoru, te se priključuje OBD-II uređaj koji dobiva napajanje iz automobila. Nakon što se OBD-II uređaj uključi i uspostavi se veza s vozilom, OBD-II uređaj krene odašiljati Bluetooth signal koji drugi uređaji mogu detektirati i bežično se povezati na njega. U ovom radu za povezivanje se koristio mobilni uređaji za koje je izrađena mobilna aplikacija za prikaz i prikupljanje podataka, na način da se putem Bluetooth veze očitavaju podaci koje prikuplja OBD-II uređaj.

Drugi dio je slanje podataka, gdje se mobilni uređaj najprije treba prijaviti na web poslužitelj s već poznatim korisničkim imenom i zaporkom kako bi se uspostavila veza i kako bi se dobili kolačići s poslužitelja za održavanje međusobne veze. Tijekom procesa prikupljanja podataka aplikacija je izrađena na način da nakon što se očitaju sve potrebne komande koje su prikazane na slici 13. se spremaju u String varijablu koja se formatira za slanje na server HTTP-om putem POST funkcije.

Treći dio je spremanje uspješno očitanih podataka u bazu podataka na poslužitelju. Na slici 33. prikazan je oblik skupa podataka na web poslužitelju. Sastoji se od XML kôda koji je podijeljen u skupove OBD_data gdje su pohranjeni prikupljeni podaci pod različitim identifikatorom (id_data).



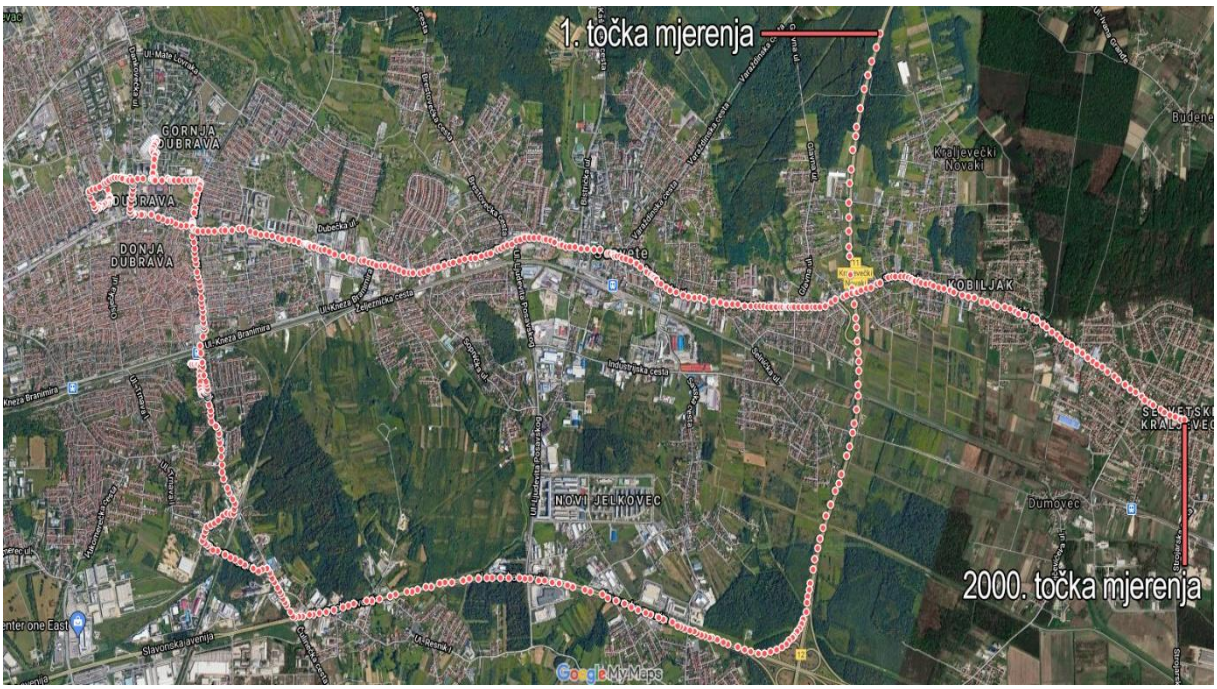
Slika 32. Dijagram sustava

```
▼<OBD_data>
   <id_data>1007</id_data>
   <obd_id>2</obd_id>
   <vehicle_speed>10</vehicle_speed>
   <engine_runtime>93</engine_runtime>
   <fuel_consumption_rate>-1</fuel_consumption_rate>
   <fuel_level>76.47059</fuel_level>
   <ignition_monitor>ON</ignition_monitor>
   <control_module_power>13</control_module_power>
   <throttle_position>81.56863</throttle_position>
   <engine_rpm>1176</engine_rpm>
   <engine_oil_temp>45</engine_oil_temp>
   <mass_air_flow>13.52</mass_air_flow>
   <absolute_load>-1</absolute_load>
   <fuel_pressure>-1</fuel_pressure>
   <barometric_pressure>99</barometric_pressure>
   <fuel_rail_pressure>54680</fuel_rail_pressure>
   <intake_manifold_pressure>107</intake_manifold_pressure>
   <engine_coolan_temp>58</engine_coolan_temp>
   <ambient_air_temp>29</ambient_air_temp>
   <air_intake_temp>38</air_intake_temp>
   <dist_traveled_with_MIL_on>0</dist_traveled_with_MIL_on>
   <longitude>45.8079</longitude>
   <latitude>16.0418</latitude>
   <altitude>172.05678564682603</altitude>
   <time_stamp>2019-06-21T11:11:27</time_stamp>
   <time_stamp_server>2019-06-21T11:11:10.187</time_stamp_server>
</OBD_data>
```

Slika 33. Prikaz prikupljenih podataka na web poslužitelju, Izvor: [6]

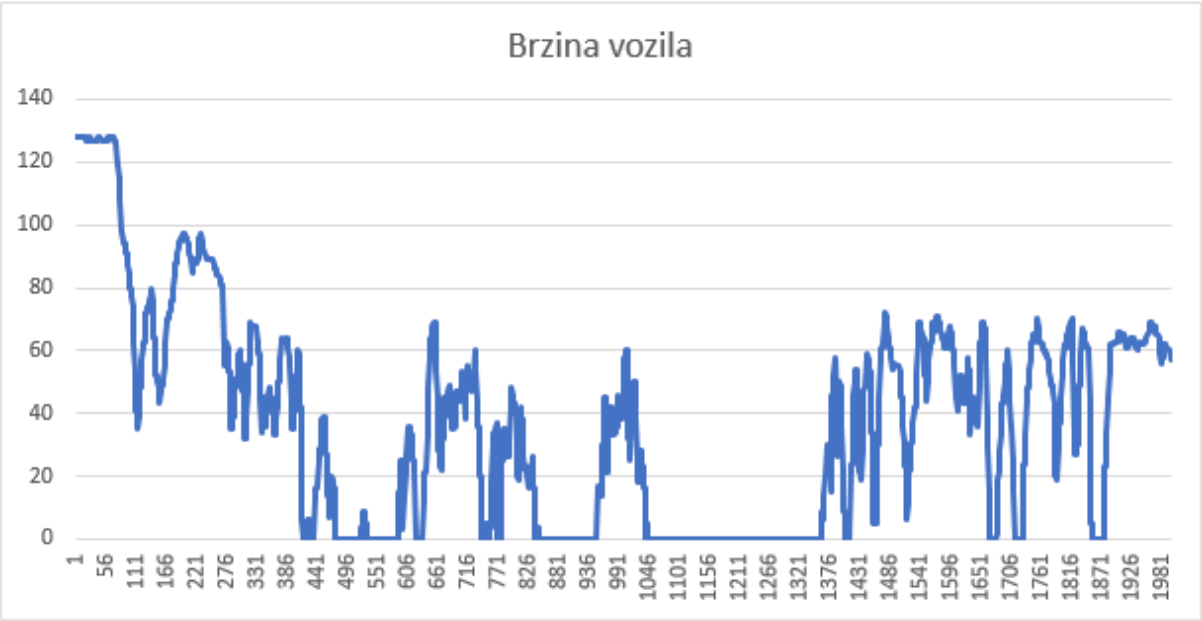
5.2. Analiza prikupljenih podataka

Iz podataka korištenih za primjer prikazane su vrijednosti očitane s OBD-II kao što su brzina, okretaji motora, razina papučice gasa, razina goriva u spremniku i dr. Za neke vrijednosti iscrtani su grafovi koji se mogu vidjeti u nastavku.



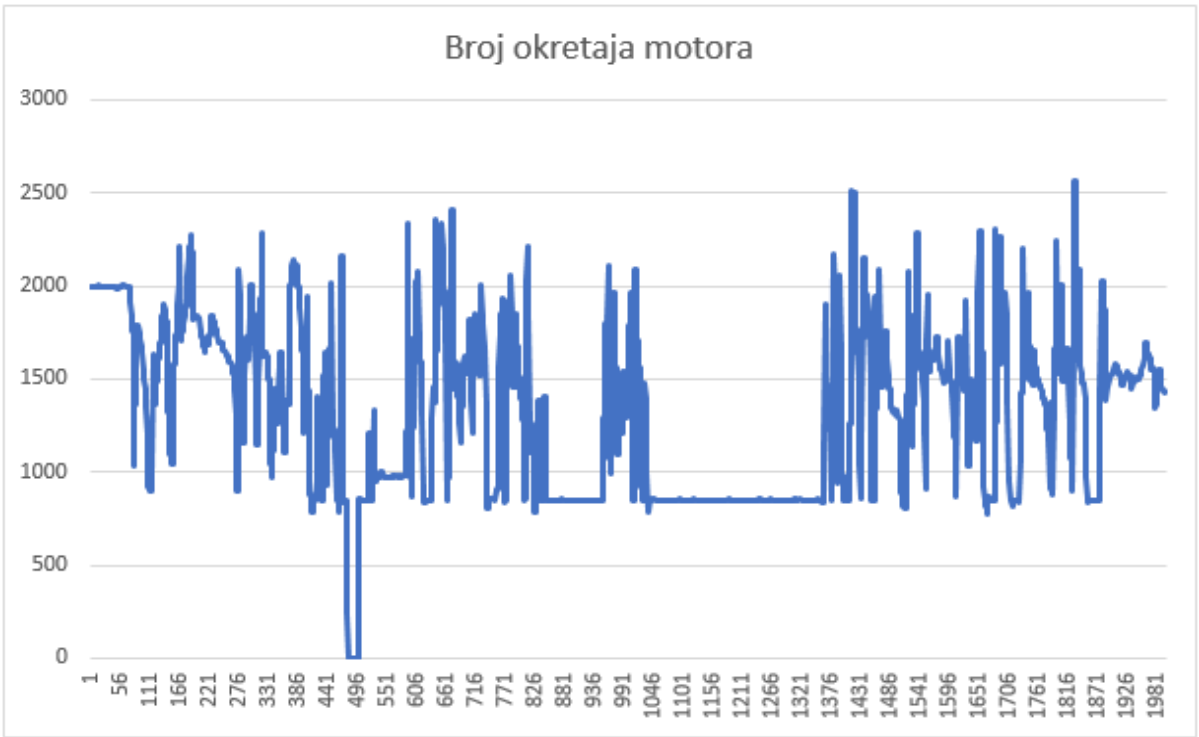
Slika 34. Prikaz lokacije vozila na Google karti

Na slici 34. prikazana su lokacijska mjerenja podataka pomoću geografske širine i dužine dobivene GPS prijemnikom na mobilnom uređaju. Mjerenja su iscrtana na Google karti gdje je lako vidljivo kretanje vozila u tom vremenskom periodu.



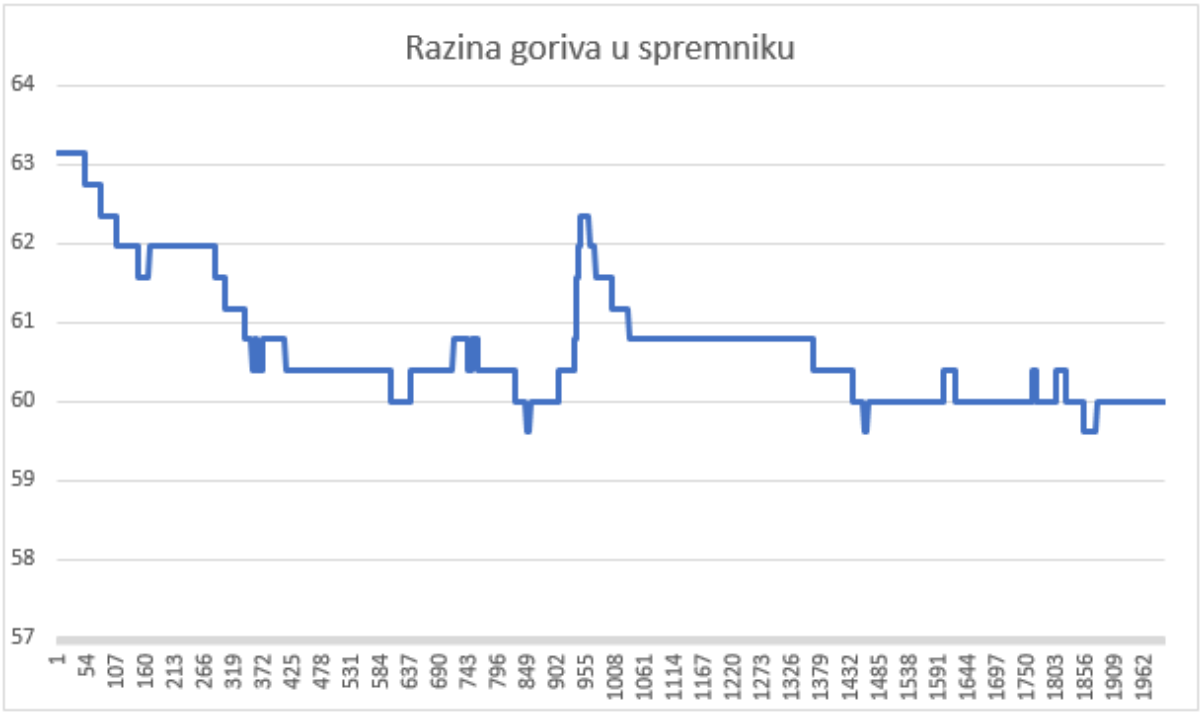
Slika 35. Prikaz brzine vozila

Na slici 35. grafički je prikazano mjerenje brzine tijekom te vožnje. Podaci brzine vozila očitani su s OBD-II uređaja gdje se podatak očita i pošalje na web poslužitelj u vremenskom periodu do 1 sekunde. Iz podataka o brzini vozila u kombinaciji s lokacijom može se primjerice očitati da li je vozač vozio preko propisanih ograničenja brzine.



Slika 36. Prikaz okretaja motora u vozilu

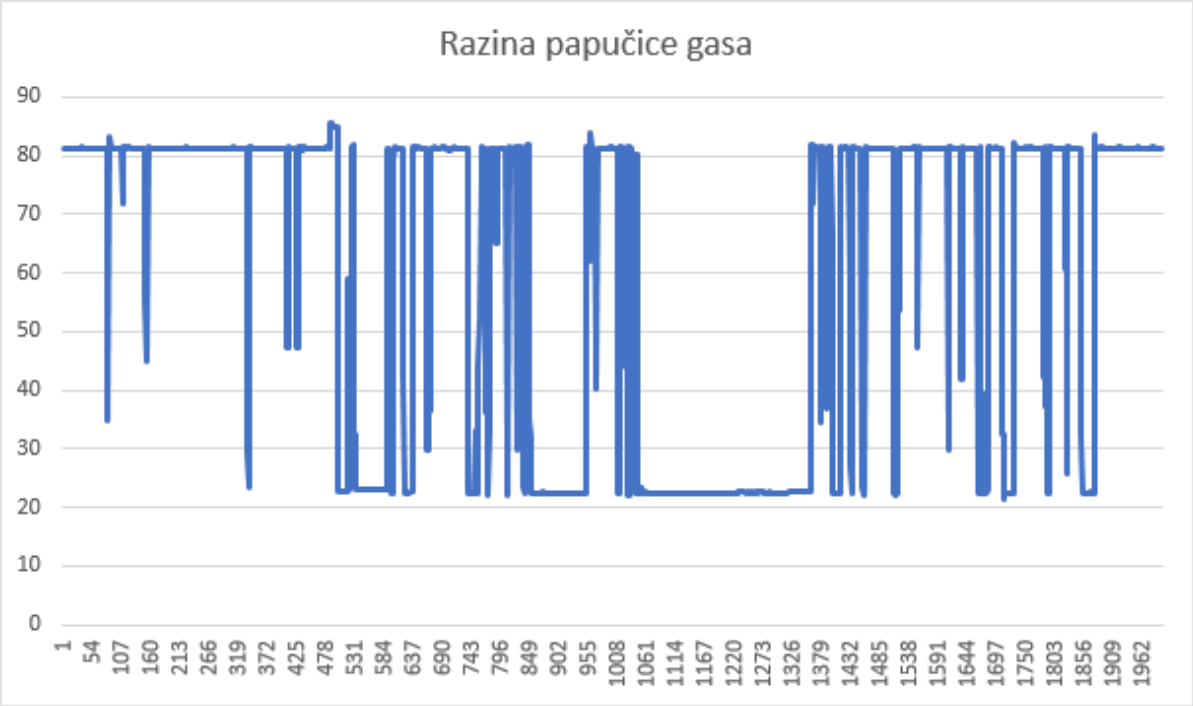
Na slici 36. grafički je prikazano mjerenje okretaja motora u vozilu iz čega se može očitati kada je vozilo ugašeno ili ako vozilo ima veću potrošnju goriva nego inače. Podaci o broju okretaja motora mogu pomoći pri pronalasku razloga veće potrošnje goriva, veći broj okretaja motora rezultira većoj potrošnji goriva.



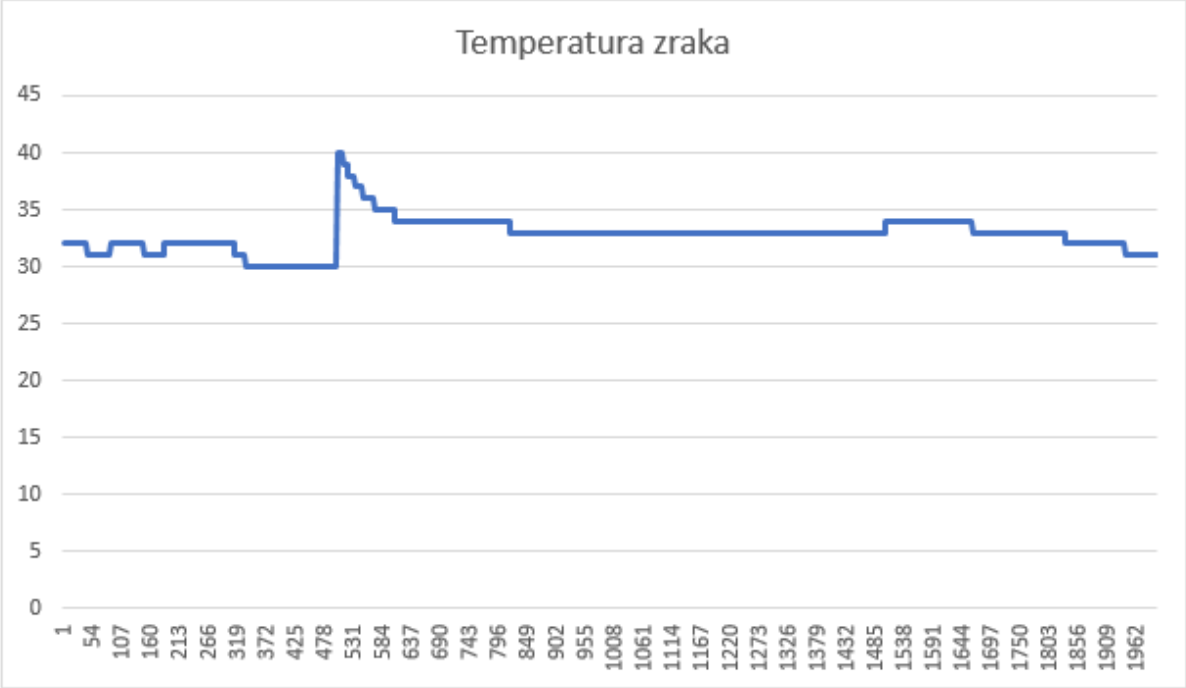
Slika 37. Prikaz razine goriva u spremniku

Na slici 37. grafički je prikazano mjerenje razine goriva u spremniku. Mjerenje razine goriva u spremniku omogućuje detaljniju analizu zbog mogućeg prikaza vremena i količine punjenja spremnika te također je moguće očitati stopu potrošnje goriva.

Na slici 38. grafički je prikazano mjerenje razine papučice gasa što je povezano sa okretajima motora i brzinom vozila, te se može očitati vremenski period ili koliko često vozilo mijenja brzinu što je vidljivo naglim skokovima na grafu.



Slika 38. Prikaz razine papučice gasa



Slika 39. Prikaz mjerenja temperature zraka

Na slici 39. grafički je prikazano mjerenje temperature zraka što može utjecati na potrošnju goriva.

6. Zaključak

U ovom završnom radu razvijen je demonstracijski model ITS aplikacije za prikupljanje podataka s vozila. Demonstracijski model ITS aplikacije zasniva se na mobilnoj aplikaciji izrađenoj na bazi Android operativnog sustava koju pokreće pametni telefon. Podaci se prikupljaju iz vozila preko OBD-II dijagnostičkog uređaja koji je spojen s pametnim telefonom preko Bluetooth bežične veze i koji pomoću internetske veze šalje prikupljene podatke na web poslužitelj. Prilikom izrade aplikacije za potrebe simuliranja rada sustava vozilo je zamijenjeno OBD-II emulatorom koji emulira stvarno vozilo, no podaci koji su slani na web poslužitelj su bili prikupljeni pomoću stvarnog vozila.

Ovaj završni rad i njegov praktični dio mogu se primijeniti u nastavi za demonstraciju rada sustava za prikupljanje stvarno-vremenskih podataka, za praćenje vozila putem GPS prijemnika ugrađenom u pametni telefon, za komuniciranje mobilne aplikacije s web poslužiteljem te kao edukacijski materijali budućim ITS stručnjacima pri razvoju ITS aplikacija zasnovanih na mobilnim platformama.

Popis literature

[1] B&B Electronics. Preuzeto sa: http://www.obdii.com/background.html [Pristupljeno: Srpanj 2020.]

[2] 1aauto. Preuzeto sa: https://www.1aauto.com/content/articles/what-is-obd2 [Pristupljeno: Srpanj 2020.]

[3] OBD Station: OBD2 protocols https://obdstation.com/obd2-protocols/ (Pristupljeno: Srpanj 2020.)

[4] Google Developers. Preuzeto sa: https://developer.android.com/studio/features [Pristupljeno: Srpanj 2020.]

[5] Google Developers. Preuzeto sa: https://developer.android.com/ [Pristupljeno: Srpanj 2020.]

[6] Sordito. Preuzeto sa: http://sordito.fpz.hr/OBD_service/OBD_service.asmx [Pristupljeno: Srpanj 2020.]

[7] MDN web docs. Preuzeto sa: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods [Pristupljeno: Srpanj 2020.]

[8] Adobe. Preuzeto sa: https://www.adobe.com/products/xd.html [Pristupljeno: Srpanj 2020.]

[9] Margaret Rouse, TechTarget. Preuzeto sa: https://whatis.techtarget.com/definition/

XML-Extensible-Markup-Language [Pristupljeno: Srpanj 2020.]

[10] Google Developers. Preuzeto sa: https://developer.android.com/guide/topics/ui/

declaring-layout [Pristupljeno: Srpanj 2020.]

[11] Google Developers. Preuzeto sa: https://developer.android.com/training/constrai

nt-layout [Pristupljeno: Srpanj 2020.]

[12] Mmobiel. Preuzeto sa: https://www.mmobiel.com/obd2-mini-elm327-bluetooth-adapter-car-compatible-with-android-windows [Pristupljeno: Srpanj 2020.]

[13] Webprogramiranje. Preuzeto sa: https://www.webprogramiranje.org/wp-content/uploads/lifecycle-srpski.jpg [Pristupljeno: Srpanj 2020.]

[14] Google Developers. Preuzeto sa: https://developer.android.com/guide/components/fragments [Pristupljeno: Srpanj 2020.]

[15] Microsoft Visual Studio. Preuzeto sa: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019 [Pristupljeno: Srpanj 2020.]

[16] Slide player. Preuzeto sa: https://slideplayer.com/slide/4502179/14/images/5/Connector+Types+%E2%80%93+A+%26+B+Img+source%3A+++_m%3Dknowledgebase%26_a%3Dviewarticle%26kbarticleid%3D3..jpg [Pristupljeno: Srpanj 2020.]

Popis slika

Popis tablica

Sveučilište u Zagrebu
Fakultet prometnih znanosti
10000 Zagreb
Vukelićeva 4

IZJAVA O AKADEMSKOJ ČESTITOSTI I SUGLASNOST

Izjavljujem i svojim potpisom potvrđujem kako je ovaj završni rad isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu što pokazuju korištene bilješke i bibliografija.

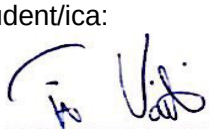
Izjavljujem kako nijedan dio rada nije napisan na nedozvoljen način, niti je prepisan iz necitiranog rada, te nijedan dio rada ne krši bilo čija autorska prava.

Izjavljujem također, kako nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Svojim potpisom potvrđujem i dajem suglasnost za javnu objavu završnog rada pod naslovom IZRADA SUSTAVA ZA ONLINE PRIKUPLJANJE PODATAKA S VOZILA OPREMLJENIH OBD UREĐAJEM na internetskim stranicama i repozitoriju Fakulteta prometnih znanosti, Digitalnom akademskom repozitoriju (DAR) pri Nacionalnoj i sveučilišnoj knjižnici u Zagrebu.

U Zagrebu, 07/09/2020

Student/ica:



(potpis)