UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matic Končan

Aplikacija za optimizacijo sistema varčne vožnje vozil na motorni pogon

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Dejan Lavbič

Ljubljana, 2013

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00322/2012
Datum: 03.09.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MATIC KONČAN**

Naslov: **APLIKACIJA ZA OPTIMIZACIJO SISTEMA VARČNE VOŽNJE VOZIL NA MOTORNI POGON**

APPLICATION FOR ECO-DRIVING OPTIMIZATION OF MOTOR-DRIVEN CARS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Omejenih naravnih virov se vedno bolj zavedamo in to zavedanje od nas zahteva predvsem preudarno in varčno uporabo le-teh. Še posebej je ta problem prisoten v avtomobilski industriji, saj zmanjševanje zaloge fosilnih goriv vpliva na večanje cene goriva. Vozniki lahko z varčnim načinom vožnje veliko prihranijo in pozitivno vplivajo na kakovost zraka. V okviru diplomske naloge naj kandidat razvije inteligentnega pomočnika v obliki mobilne aplikacije, ki voznika opozarja ob neekonomični vožnji. Metoda ugotavljanja neekonomičnosti vožnje naj temelji na senzorjih, do katerih je moč dostopati preko sistema OBD. Kandidat naj izbere ustrezne senzorje in pripravi modele, ki uporabniku pomagajo pri varčni vožnji.

Mentor:

doc. dr. Dejan Lavbič

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Matic Končan, z vpisno številko **63020082**, sem avtor diplomskega dela z naslovom:

Aplikacija za optimizacijo sistema varčne vožnje vozil na motorni pogon

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Dejana Lavbiča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 21. februarja 2013 Podpis avtorja:

Iskreno se zahvaljujem mentorju doc. dr. Dejanu Lavbiču za vso pomoč in strokovne nasvete ob izdelavi diplomskega dela.

Posebno zahvalo izrekam tudi Nataliji Volmajer in Žigi Šveglju. Za vajino pomoč in podporo sem vama zelo hvaležen.

Za potrpežljivost in podporo pri študiju se zahvaljujem tudi svojim staršem.

Kazalo

Kazalo slik in tabel

Seznam uporabljenih kratic

ADT (Android Development Tools): dodatek za razvojno okolje Eclipse IDE, ki omogoča razvoj aplikacij za operacijski sistem Android

ALDL (Assembly Line Diagnostic Link): predhodnik protokola OBD

API (Application Programming Interface): nabor ukazov, funkcij, protokolov in orodij za razvoj aplikacij

CARB (California Air Resources Board): agencija, ki skrbi za kvaliteto zraka v Kaliforniji

CSV (Comma Separated Values): vrednosti, ločene z vejicami

DLC (Diagnostic Link Connector): diagnostični priključek

DTCs (Diagnostic Trouble Codes): oznake, ki omogočajo določitev težave v vozilu

ECU (Electronic Control Unit): krmilna enota sistema OBD

EOBD (European On Board Diagnostics): evropska različica sistema OBD

EPA (Environmental Protection Agency): agencija za varovanje okolja v Združenih državah Amerike

GPS (Global Positioning System): satelitski navigacijski sistem

HTTP (HyperText Transfer Protocol): protokol, ki se uporablja za prenos podatkov po svetovnem spletu

IDE (Integrated Development Environment): programska oprema, ki omogoča celovit razvoj aplikacij

MIL (Malfunction Indicator Light): opozorilna lučka na armaturni plošči vozila, ki opozarja na napako v vozilu

OBD (On-Board Diagnostics): standard, ki zagotavlja računalniško podprt sistem v vozilih

PHP (Hypertext Preprocessor): odprtokodni programski jezik, ki omogoča razvoj dinamičnih spletnih strani ali aplikacij

PID (Parameter Identification Numbers): število, ki v izbranem načinu označuje senzor v vozilu

REST (REpresentational State Transfer): arhitekturni način, ki omogoča izmenjavo podatkov porazdeljenih sistemov

SAE (Society of Automotive Engineers): organizacija iz Združenih držav Amerike, ki skrbi za standardizacijo v različnih gospodarskih panogah; *danes SAE International*

SDK (Software Development Kit): nabor programske opreme, ki omogoča razvoj aplikacij za določeno napravo ali operacijski sistem

SQL (Structured Query Language): programski jezik, namenjen upravljanju s podatki v sistemih za upravljanje s podatkovnimi bazami

Povzetek

Zmanjševanje zaloge fosilnih goriv vpliva na večanje cen goriva. Vpliv uporabe goriva se odraža tudi v slabši kakovosti zraka. Namen diplomskega dela je izdelava mobilne aplikacije za pomoč voznikom pri optimizaciji varčne vožnje. Aplikacija je razvita za operacijski sistem Android. Slednja preko sistema OBD, ki je od leta 2004 vgrajen v vseh novih vozilih v Evropski uniji, pridobi podatke o porabi goriva. Voznika vozila ob nevarčni vožnji z zvočnim opozorilom opozori na preveliko porabo goriva. V diplomski nalogi je predstavljen standard OBD in njegova zgodovina. Predstavljen je vmesnik ELM327, ki ga aplikacija uporablja za povezavo s sistemom OBD. Predstavljeni so nekateri ukazi OBD, med katerimi je tudi ukaz, ki sporoča trenutno porabo goriva v vozilu. Ker v testiranih vozilih slednji ni podprt, se vrednost izračunava s pomočjo vrednosti senzorja masnega pretoka zraka. Ugotovljeno je bilo, da tudi senzor masnega pretoka zraka v vseh vozilih ni podprt. Poleg trenutne porabe goriva aplikacija izračunava tudi povprečno porabo goriva. Za razvoj aplikacije je bil razvit vmesnik za testiranje ukazov OBD in funkcionalnost, ki omogoča analizo dobljenih podatkov.

Ključne besede

OBD, on board diagnostics, ELM327, mobilna aplikacija, Android, vozilo, okolje, poraba, gorivo

Abstract

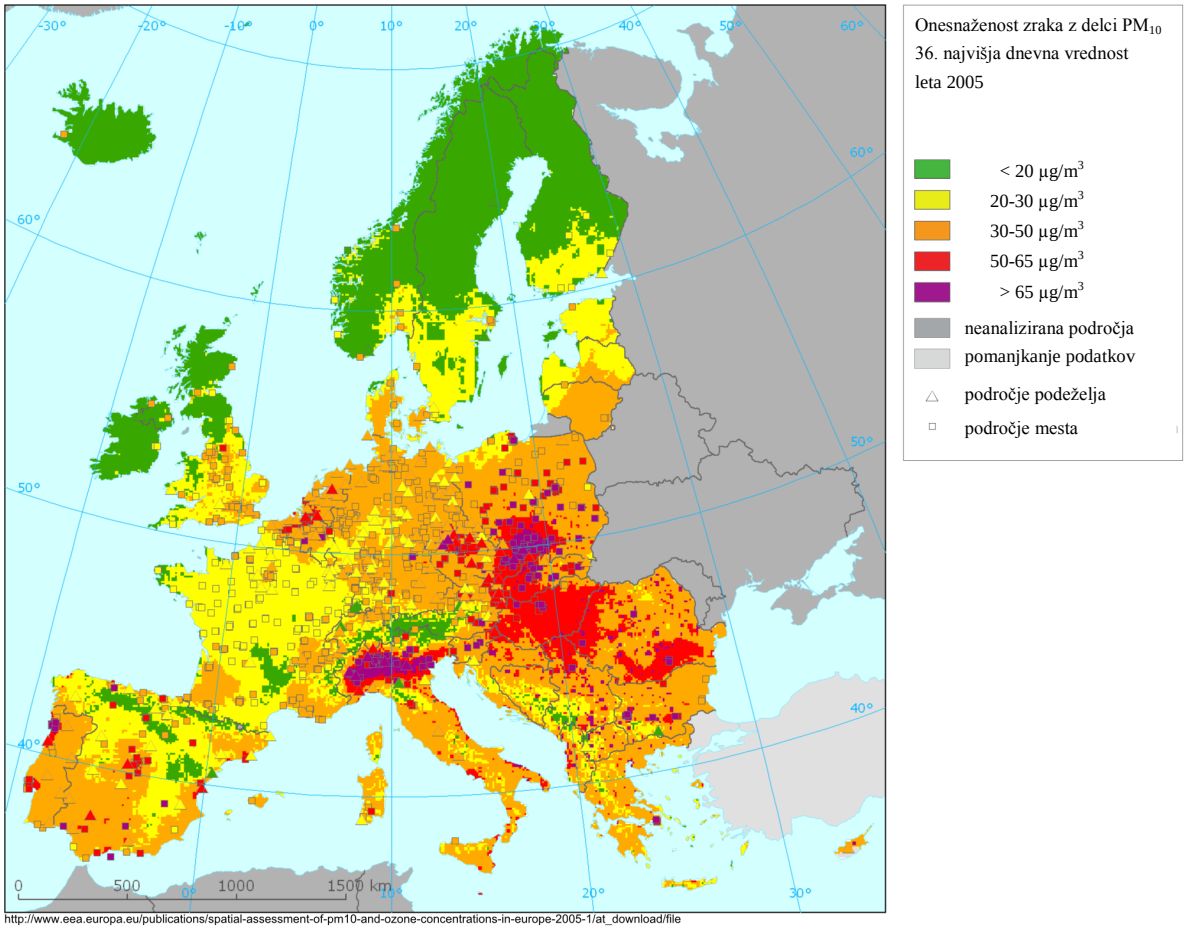
The declining fossil fuel reserves are causing fuel prices to rise. Fuel consumption is also causing the quality of air to drop. The aim of this diploma thesis was to develop a mobile application to help drivers optimise their driving in terms of fuel efficiency. The application has been developed for Android. It obtains fuel consumption data using the OBD system, which has been built into every new vehicle in the EU since 2004. It emits an audible warning about excessive fuel consumption whenever the driver is not driving in a fuel-efficient manner. The diploma thesis presents the OBD standard and its history. It also showcases the ELM327 interface, which the application uses to connect to the OBD system. Some of the OBD commands are outlined, including the command to display the vehicle's current fuel consumption. Since the tested vehicles do not support this, the value was calculated using data obtained from a mass air flow sensor. However, it was found that the mass air flow sensor is also not supported by all the vehicles. In addition to the current fuel consumption, the application also calculates the average fuel consumption. Developing the application involved developing an interface for testing OBD commands and functionalities that enable the acquired data to be analysed.

Keywords

OBD, on board diagnostics, ELM327, mobile application, Android, vehicle, environment, consumption, fuel

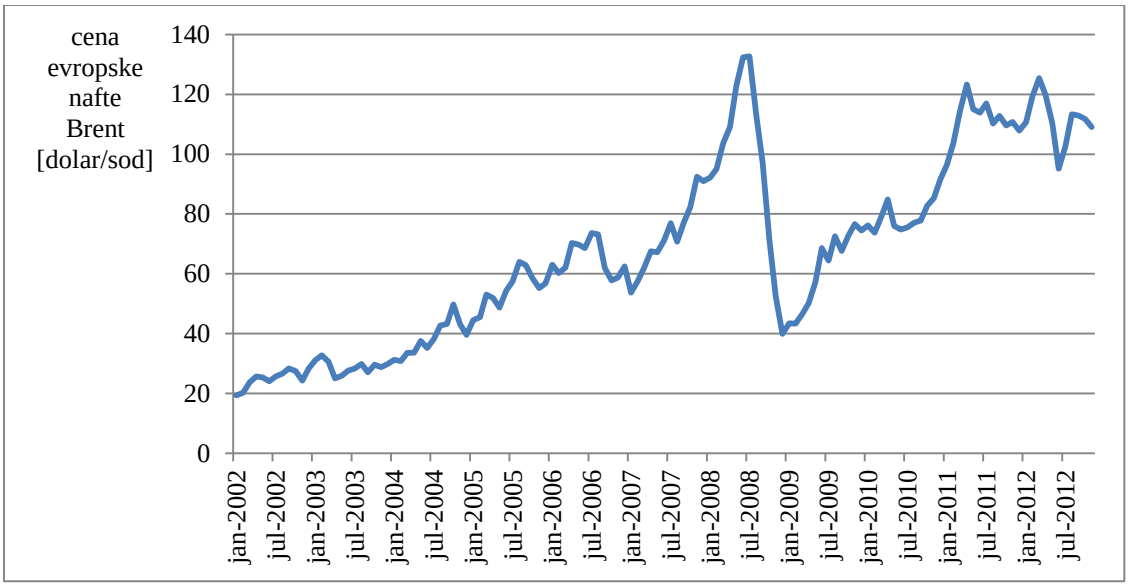
1 Uvod

Zrak je eden najpomembnejših elementov, od katerih je odvisno življenje na Zemlji. Kakovost zraka za življenje se slabša. Še posebej v mestih, kjer sta poseljenost prebivalstva in razvoj industrije večja. Na slabšo kakovost zraka vplivajo tudi stranski produkti izgorevanja fosilnih goriv in nekaterih obnovljivih virov energije. Le te uporabljamo za vožnjo z različnimi prevoznimi sredstvi, med katere sodijo tudi vozila cestnega prometa. Onesnaženost zraka v Evropi prikazuje slika 1.



Slika 1: Onesnaženost zraka z delci PM_{10} leta 2005.[1]
V Sloveniji je za varovanje zdravja ljudi z zakonom določena dnevna mejna koncentracija 50 µg/m^3, ki v koledarskem letu ne sme biti presežena več kot 35-krat. Letna mejna koncentracija ne sme presegati 40 µg/m^3.[2]

Znano je, da je omejena zaloga fosilnih goriv. Kljub temu dejstvu pa se še vedno najpogosteje uporabljajo kot gorivo za prevozna sredstva cestnega prometa. Ker se število vozil veča, se veča tudi potreba po gorivu. To ponudnikom naftnih derivatov omogoča prilagajanje cen goriva. Na dvigovanje cen goriva vpliva tudi gibanje cen evropske nafte Brent (slika 2).



Slika 2: Graf gibanja cene evropske nafte Brent v dolarjih na sod v obdobju od leta 2002 do 2012.[3]

Količina porabe goriva v vozilih je odvisna od tehničnih lastnosti in načina uporabe vozil. Na prvo lahko vplivamo z izborom vozila, na drugo pa z njegovim vzdrževanjem in načinom vožnje. K izboljšanju slednjega lahko pripomorejo različna izobraževanja. V pomoč so lahko tudi pripomočki, ki jih vgradimo v vozilo ali uporabljamo med vožnjo.

Namen diplomskega dela je izdelati mobilno aplikacijo, ki bi pripomogla k nemoteni varčni vožnji. Aplikacija bi s pomočjo sistema OBD voznika vozila opozarjala na preveliko porabo goriva, hkrati pa s svojim delovanjem ne bi vplivala na varnost udeležencev v cestnem prometu.

V nadaljevanju bomo najprej predstavili sistem OBD. Na kratko bomo opisali vzroke za njegov razvoj, njegovo veliko razširjenost in navedli osnovne sestavne dele.

V tretjem poglavju bomo predstavili orodja, naprave in tehnologije, ki smo jih uporabili. Opisali bomo vmesnik OBD II, predstavili način izmenjave podatkov ter opisali uporabljene ukaze OBD. Predstavili bomo simulator sistema OBD II, ki nam je močno olajšal razvoj in podali vzroke za izbor platforme Android.

V četrtem poglavju sledi opis razvoja mobilne aplikacije in v zadnjem poglavju še sklepne ugotovitve.

2 Sistem OBD

Sistem OBD je računalniško podprt sistem, ki omogoča diagnostiko vozil v cestnem prometu.[6, 7]

Prvi pojavi njegovih predhodnikov segajo v konec sedemdesetih in začetek osemdesetih let prejšnjega stoletja, ko so nekateri proizvajalci avtomobilov razvili sisteme za spremljanje delovanja glavnih komponent motorja. S tem so začeli izpolnjevati zahteve po zmanjšanju emisij vozil, ki jih je zahtevala ameriška agencija za varovanje okolja EPA.[5]

Razvijali so različne priključke, strojne vmesnike in protokole. Eden prvih protokolov je bil ALDL, ki ga je uvedel General Motors za potrebe testiranja na lastni proizvodnji liniji leta 1980. Poleg navedenega je lastnikom vozil v primeru težav z motorjem omogočil prikaz dvomestnega števila, s katerim so ugotovili mesto težave v vozilu. Do tedaj je na okvaro zgolj opozarjala lučka »Check Engine«, poznana tudi kot MIL.[4, 7]

V tem obdobju so proizvajalci v svoja vozila uvajali različne sisteme in signale. Zaradi potrebe po njihovi standardizaciji je SAE leta 1988 z lastnim standardom priporočila vrsto diagnostičnega priključka in množico diagnostičnih oznak DTCs. Večino teh standardov in priporočil je kasneje sprejela tudi vladna organizacija CARB, ki zakonsko zagotavlja kvaliteto zraka v Kaliforniji.[4, 5]

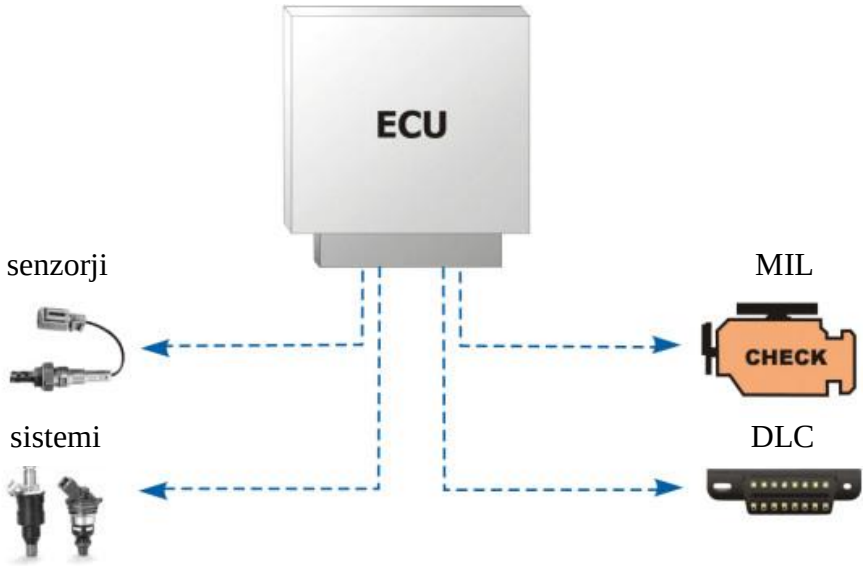
CARB je z začetkom leta 1991 zahtevala uvedbo osnovnih funkcionalnosti OBD v vseh novih vozilih, prodanih v Kaliforniji. V tem času vrsta priključka, njegova lokacija in podatkovni protokol še niso bili standardizirani. Z željo širitve uporabe sistema OBD po celotnih Združenih državah Amerike, so razvili razširjeno specifikacijo standarda, poimenovanega OBD II. V standard so vključili tudi omenjena priporočila SAE iz leta 1988. Od 1.1.1996 je uporaba standarda postala obvezna v vseh prodanih vozilih v Kaliforniji.[4]

Dovršenost standarda OBD II je omogočila, da ga je prevzela tudi državna agencija EPA in ga z istim datumom uveljavila po celotnih Združenih državah Amerike.[4]

Uporabi enakovrednega in povsem podobnega standarda je sledila tudi Evropska unija. Poimenovali so ga EOBD. Uporabo slednjega je z letom 2001 zahtevala v vseh novo prodanih bencinskih vozilih, v vozilih na dizelski pogon pa z letom 2004. Kasneje so standard OBD prevzele tudi nekatere druge države.[4]

Danes uporaba standarda OBD v vozilih omogoča podrobnejši nadzor nad njihovim delovanjem in nadzor nad izpolnjevanjem strogih okoljevarstvenih standardov. Poleg nadzora delovanja motorja omogoča tudi spremljanje delovanja podvozja, karoserije in nekaterih drugih naprav.[5]

Osnovni sistem OBD prikazuje slika 3. Sestavljen je iz krmilne enote ECU, ki podatke iz različnih senzorjev uporablja za krmiljenje različnih sistemov v vozilu. S tem zagotavlja željeno uspešnost delovanja vozila. Lučka MIL skrbi za pravočasno obveščanje voznika o napakah vozila. Sodobna vozila podpirajo več sto parametrov PID, do katerih lahko z uporabo orodja, imenovanega »scan tool«, dostopamo preko diagnostičnega priključka DLC.[7]



Slika 3: Osnovni sistem OBD.[7]

Standard OBD dovoljuje uporabo petih signalnih protokolov OBD[4]:

- SAE J1850 PWM,
- SAE J1850 VPW,
- ISO 9141-2,
- ISO 14230 KWP2000 ali
- ISO 15765 CAN.

Običajno v posameznem vozilu proizvajalci uporabijo samo enega od naštetih signalnih protokolov OBD.

3 Opis uporabljenih orodij, naprav in tehnologij

3.1 Vmesnik OBD II

Izdelave rešitve smo se lotili z izborom vmesnika OBD II, ki omogoča povezavo s sistemom OBD vozila.

Ob pregledu ponudbe vmesnikov na trgu smo opazili, da jih obstaja več vrst, ki se razlikujejo predvsem po naslednjih lastnostih:

- podprtih signalnih protokolih OBD,
- načinu povezave vmesnika OBD II z računalnikom oz. drugo primerno napravo,
- naprednejših ukazih vmesnika,
- velikosti in
- ceni.

Z vmesnikom OBD II se je možno povezati na več načinov. Najpogosteje se uporablja naslednje standarde ali tehnologije[8]:

- RS232,
- USB,
- Bluetooth ali
- Wi-Fi.

Odločili smo se za nakup zelo razširjenega vmesnika ELM327 v1.5, ki omogoča povezavo z uporabo brezžične tehnologije Bluetooth. Vmesnik je kopija vmesnika ELM327[9], ki omogoča povezavo RS232 s sistemom OBD in ga izdeluje družba Elm Electronics iz Kanade. Na nakup so vplivali popolna podpora signalnih protokolov OBD, uporaba povezave Bluetooth in nizka cena. Vmesnik ELM327 prikazuje slika 4.

Slika 4: Vmesnik ELM327 v1.5, ki smo ga uporabljali.

3.2 Komunikacija z vmesnikom OBD II in krmilno enoto ECU

Vmesnik ELM327 smo priključili na diagnostični priključek OBD vozila, ki se standardno nahaja v bližini volanskega obroča. Na prenosnem računalniku smo z vmesnikom vzpostavili povezavo Bluetooth. S programom, ki omogoča terminalni način dela smo začeli z uporabo vmesnika. Primer uporabe vmesnika ELM327 prikazuje slika 5.

Slika 5: Odgovor vmesnika ELM327 na ukaz ATZ.

S pomočjo uporabniških navodil[9] smo ugotovili, da se ukazi delijo v dve skupini:

- ukazi AT za krmiljenje in uporabo vmesnika ELM327, ki so sestavljeni iz ASCII znakov in
- ukazi OBD za komunikacijo in izmenjavo podatkov s sistemom OBD vozila, ki lahko vsebujejo ASCII znake šestnajstiških števil (znaki 0-9 in A-F).

Ukaze AT uporabljamo za nastavitev delovanja vmesnika ELM327. Delimo jih na splošne ukaze, splošne ukaze signalnih protokolov OBD in ukaze, vezane na posamezen signalni protokol OBD. Med testiranjem in izdelavo rešitve smo uporabljali predvsem naslednje splošne ukaze:

- ATD (uveljavi privzete (tovarniške) nastavitve),
- ATE0 (omogoči izpis podatkov sistema OBD),
- ATE1 (onemogoči izpis podatkov sistema OBD),
- ATI (izpiše različico vmesnika),
- ATIGN (prepozna, če je vozilo prižgano),
- ATL0 (omogoči izpis v eni vrstici),
- ATL1 (omogoči izpis v več vrsticah),
- ATWS (ponovni zagon – hitra izvedba),
- ATZ (ponovni zagon)

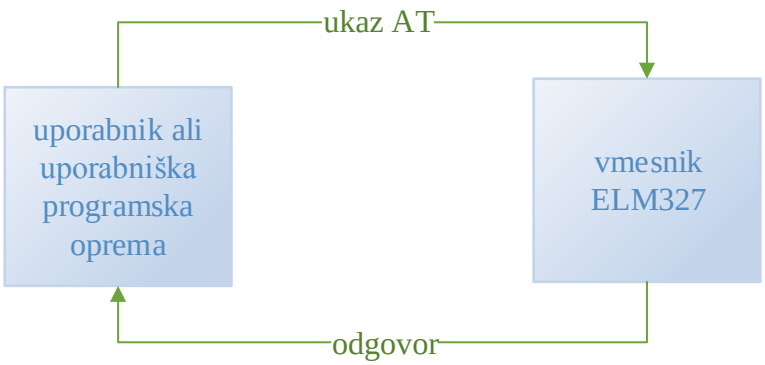
in splošne ukaze signalnih protokolov OBD:

- ATDP (izpiše naziv trenutno uporabljenega signalnega protokola OBD),
- ATDPN (izpiše številko trenutno uporabljenega signalnega protokola OBD),
- ATSP h (uporabi signalni protokol h in ga shrani v nastavitve),
- ATSP Ah (uporabi signalni protokol h in ga shrani v nastavitve; v primeru neuspeha ga prepozna in shrani v nastavitve),
- ATST hh (uporabi časovno omejitev za prejem podatkov),
- ATTP h (uporabi signalni protokol h),
- ATTP Ah (uporabi signalni protokol h; v primeru neuspeha ga prepozna).

Komunikacija med računalnikom ali drugo napravo in vmesnikom OBD II z ukazi AT je dvosmerna. Poteka v smeri proti vmesniku in od njega, in sicer po naslednjih pravilih:

1. uporabnik ali uporabniška programska oprema pošlje zahtevo (npr. ATZ) preko izbrane povezave (npr. Bluetooth) vmesniku,
2. vmesnik sprejme zahtevo, se nanjo odzove s podatki (npr. ELM327 v1.5) in jih posreduje uporabniku ali uporabniški programski opremi.

Komunikacijo prikazuje tudi naslednja slika 6.

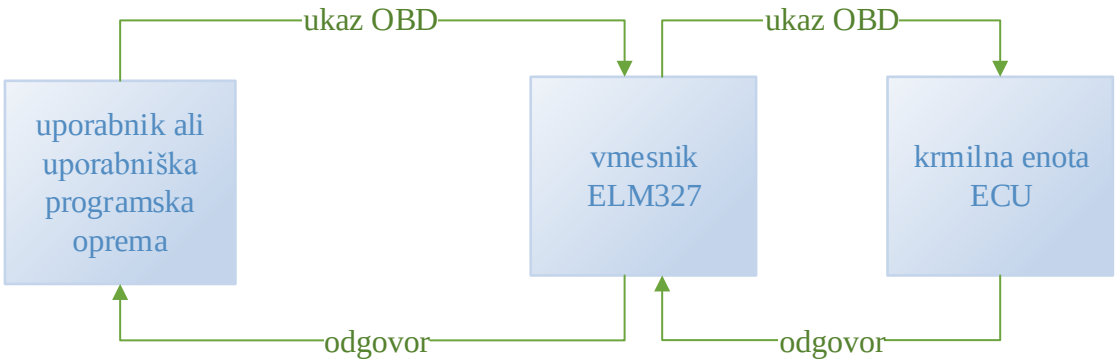


Slika 6: Komunikacija z ukazi AT.

Podobno poteka komunikacija z vmesnikom OBD II, z uporabo ukazov OBD:

1. uporabnik ali uporabniška programska oprema pošlje zahtevo (npr. 010C) preko izbrane povezave (npr. Bluetooth) vmesniku,
2. vmesnik sprejme zahtevo in jo s pomočjo izbranega signalnega protokola OBD (npr. ISO 9141-2) prenese krmilni enoti ECU vozila,
3. krmilna enota ECU se odzove s podatki (npr. 41 0C 2F 80) in jih preko vmesnika posreduje uporabniku ali uporabniški programski opremi.

Komunikacijo prikazuje tudi naslednja slika 7.



Slika 7: Komunikacija z ukazi OBD.

Za uspešno oddajo ukaza OBD vmesniku OBD II se mora vsak ukaz zaključiti s t.i. »carriage return« znakom (šestnajstiška vrednost znaka je 0D). V veliko programskih jezikih (npr. C, Java, C# ipd.) se znak označuje s kombinacijo znakov »\r«. V primeru, da se ukaz ne zaključi z omenjenim znakom, vmesnik po 20 sekundah odgovori z znakom »?«. Enako odgovori tudi na vse neveljavne ukaze.

V večini primerov na veljavne ukaze AT vmesnik OBD II odgovarja s podatki v nam razumljivi obliki. Za razliko na ukaze OBD vmesnik OBD II odgovarja z znaki v šestnajstiški obliki. Le-te je za razumevanje skladno s standardom OBD potrebo primerno uporabiti ali pretvoriti. Za uporabo ali pretvorbo lahko poskrbi uporabniška programska oprema, ki podatke prikaže uporabniku (npr. Obrati motorja: 3040,25 rpm).

Vsak odgovor vmesnika OBD II se zaključi z znakom »>« (šestnajstiška vrednost znaka je 3E). To pomeni, da je vmesnik v mirovanju in da čaka na sprejem novega ukaza.

Vmesnik OBD II na vsak odgovor krmilne enote ECU čaka 200 ms (privzeto nastavljena vrednost, ki jo je možno spremeniti z ukazom AT SP hh) in vrne vse podatke, ki jih dobi v tem času. Če ne dobi odgovora, vrne sporočilo »NO DATA«. Ker posamezno vozilo ne podpira vseh ukazov OBD, vmesnik enako odgovori tudi na ukaze, ki niso podprti.

Od verzije 1.3 dalje vmesnik ELM327 omogoča nastavitev števila vrstic podatkov, ki jih pričakuje od krmilne enote ECU. Po prejemi določenega številka vrstic, vmesnik ne čaka 200 ms za sprejem novega ukaza, ampak podatke posreduje takoj.

Hitro smo opazili, da ponujena rešitev omogoča hitrejšo komunikacijo, zato smo jo uporabili tudi v naši rešitvi. S testiranjem smo ugotovili, da z njeno uporabo lahko prejmemo tudi preko 12 odgovorov na sekundo, kar v povprečju pomeni en odgovor na vsakih 83,33 ms. Ugotovili smo tudi, da se hitrost sistema OBD med vozili lahko močno razlikuje. Ugotovitve prikazuje tabela 1.

vozilo	**št. ukazov / sekundo**
Toyota Corolla, letnik 2003	3,87
Peugeot 207, letnik 2008	7,60
Ford Focus, letnik 2008	12,72
Volvo V60, letnik 2012	13,40

Tabela 1: Povprečno število ukazov sistema OBD v sekundi.

3.3 Opis ukazov OBD

Ukazi OBD so razdeljeni v deset načinov delovanja[10], med katerimi proizvajalcem vozil ni potrebno podpreti vseh:

- 01: prikaz trenutnih podatkov,
- 02: prikaz podatkov v času pojava napake v vozilu,
- 03: prikaz diagnostičnih oznak napak v vozilu,
- 04: izbris podatkov v času pojava napake v vozilu (način 02) in diagnostičnih oznak napak (način 03),
- 05: rezultati testiranj, spremljanje senzorja za kisik,
- 06: rezultati testiranj, spremljanje ostalih komponent in sistemov,
- 07: prikaz diagnostičnih oznak napak v vozilu, ugotovljenih med trenutno in predhodno vožnjo,
- 08: nadzor delovanja komponent ali sistemov v vozilu,
- 09: prikaz informacij o vozilu,
- 0A: prikaz diagnostičnih oznak napak v vozilu (tudi izbrisanih).

Pri spoznavanju ukazov OBD smo se omejili za ukaze v načinu 01, ki prikazujejo trenutne podatke iz sistema OBD. Nekatere ukaze smo uporabili tudi pri razvoju aplikacije.

Vsak željen ukaz v načinu 01, ki je podprt s standardom OBD, ima med drugim določen parameter PID, število bajtov podatka, enačbo za preračun podatka v vrednost, njegove mejne vrednosti in mersko enoto. Nekateri ukazi imajo namesto enačbe določeno zahtevnejše pravilo, po katerem se podatek uporabi ali pretvori.

V nadaljevanju bomo predstavili nekatere ukaze OBD in način izračunavanja vrednosti podatkov. Povzetek ukazov prikazuje tabela 2.

način	PID	opis	število bajtov	sp. meja	zg. meja	merska enota	enačba
01	00	*podprti parametri PID od 01 do 20*	4	/	/	/	bitno kodiran podatek
01	0B	*senzor razdelilnika absolutnega tlaka*	1	0	255	kPa	A

način	PID	opis	število bajtov	sp. meja	zg. meja	merska enota	enačba
01	0C	*obrati motorja*	2	0	16.383,75	rpm	$\frac{(A \cdot 256) + B}{4}$
01	0D	*hitrost vozila*	1	0	255	km/h	A
01	0F	*temperatura vsesanega zraka*	1	-40	215	°C	$A - 40$
01	10	*senzor masnega pretoka zraka*	2	0	655,35	g/s	$\frac{(A \cdot 256) + B}{100}$
01	51	*vrsta goriva*	1	/	/	/	glede na tabelo
01	5E	*pretok goriva v motorju*	2	0	3212,75	l/h	$((A \cdot 256) + B) \cdot 0,05$

Tabela 2: Ukazi OBD, ki smo jih uporabili za spoznavanje s sistemom OBD.[10]

3.3.1 Podprti parametri PID od 01 do 20

Podatek predstavlja informacijo o naslednjih (od ukaza 0100 dalje) 32-ih parametrih PID (šestnajstiško od 01 do 20), ki jih sistem OBD vozila podpira.

Primer:
ukaz: *0100*
odgovor vmesnika: *41 00 FE 1B 30 13*

Odgovor vmesnika je sestavljen iz 6 bajtov, od katerih zadnji štirje predstavljajo podatek, ki ga je potrebno pretvoriti. Odgovor je sestavljen iz:

- prvi znak (4) pomeni, da gre za odgovor na ukaz,
- drugi znak (1) pomeni, da gre za odgovor na ukaz v načinu 01,
- tretji in četrti znak pomenita (00), da gre za odgovor na parameter PID 00,
- preostalih osem znakov – štirje bajti (FE 1B 30 13) predstavljajo podatek v šestnajstiški obliki.

Za pridobitev vrednosti je potrebno podatek iz šestnajstiške oblike pretvoriti v dvojiško. Ukaz OBD je podprt, če posamezno mesto vsebuje vrednost 1.

F	**E**	**1**	**B**	**3**	**0**	**1**	**3**
0000	0001	1111	1100	0011	0110	0110	0000
1,2,3,4,	5,6,7,8	9,A,B,C,	D,E,F,10,	11,12,13,14	15,16,17,18	19,1A,1B,1C,	1D,1E,1F,20

Iz zgornjega primera je razvidno, da sistem OBD podpira ukaze 0108, 0109, 010A, 010B, 010C, 010D, 010E, 0113, 0114, 0116, 0117, 011A in 011B.

3.3.2 Senzor razdelilnika absolutnega tlaka

Senzor razdelilnika absolutnega tlaka zagotavlja takojšnje informacije senzorja tlaka krmilni enoti ECU. Podatki se uporabljajo za izračun gostote zraka in za določitev masnega pretoka zraka v motorju. To vpliva na določitev potrebne porabe goriva za optimalno izgorevanje in na hitrost vžiga motorja.[11]

Primer:
ukaz: *010B*
odgovor vmesnika: *41 0B 1B*

Za določitev vrednosti je potrebno 1 bajt velik podatek 1B iz šestnajstiške oblike pretvoriti v desetiško. Ugotovimo, da je vrednost senzorja razdelilnika absolutnega tlaka 27 kPa.

3.3.3 Obrati motorja

Podatek predstavlja število obratov motorja v minuti.

Primer:
ukaz: *010C*
odgovor vmesnika: *41 0C 2F 81*

Vsak bajt podatka posebej iz šestnajstiške oblike pretvorimo v desetiško: $2F_{(16)} = 47_{(10)}$ in $81_{(16)} = 129_{(10)}$. Po enakem vrstnem redu vrednosti v desetiški obliki vstavimo v enačbo 1.

$$\frac{(\mathrm{A} \cdot 256) + \mathrm{B}}{4} \qquad (1)$$

Z izračunom (enačba 2) ugotovimo, da je število obratov motorja 3040,25 rpm.

$$\frac{(47 \cdot 256) + 129}{4} = 3040,25 \qquad (2)$$

Iz primera je razvidno, da enačba omogoča izračun vrednosti podatka v decimalnem številu. V nadaljevanju bomo predstavili tudi primer enačbe ukaza, ki omogoča uporabo negativnih števil.

3.3.4 Hitrost vozila

Podatek predstavlja hitrost vozila v km/h.

Primer:
ukaz: *010D*
odgovor vmesnika: *41 0D 21*

Način izračuna vrednosti je podoben izračunu vrednosti senzorja razdelilnika absolutnega tlaka. S preračunom podatka ugotovimo, da je hitrost vozila 33 km/h.

3.3.5 Temperatura vsesanega zraka

Podatek predstavlja temperaturo vsesanega zraka v motorju v °C.

Primer:
ukaz: *010F*
odgovor vmesnika: *41 0F 2D*

Podatek 2D iz šestnajstiške vrednosti pretvorimo v dvojiško: $2D_{(16)} = 45_{(10)}$. Dobljeno vrednost vstavimo v enačbo 3.

$$A - 40 \qquad (3)$$

Z izračunom (enačba 4) ugotovimo, da je temperatura vsesanega zraka v motorju 5°C.

$$45 - 40 = 5 \qquad (4)$$

Iz enačbe je razvidno, da je temperatura vsesanega zraka lahko tudi negativno število, vendar ne manjše od -40°C.

3.3.6 Senzor masnega pretoka zraka

Senzor masnega pretoka zraka se uporablja za ugotovitev pretoka zraka, ki vstopa v motor z notranjim izgorevanjem. Podatek je pomemben, ker krmilna enota ECU z njim uravnava in določa točen podatek o masi goriva, ki ga motor potrebuje. Zrak ob tem zaradi vpliva temperature in pritiska spreminja svojo gostoto.

Primer:
ukaz: *0110*
odgovor vmesnika: *41 10 3C 80*

Način izračuna vrednosti je podoben izračunu obratov motorja. S preračunom podatka ugotovimo, da je pretok zraka, ki vstopa v motor 154,88 g/s.

3.3.7 Vrsta goriva

Podatek predstavlja vrsto goriva, ki poganja motor vozila.

Vrsto goriva izvemo tako, da dobljeno šestnajstiško vrednost primerjamo s podatki v tabeli 3.

vrednost	vrsta goriva
01	bencin
02	metanol
03	etanol
04	dizel
05	LPG (utekočinjen naftni plin)
06	CNG (stisnjen zemeljski plin)
07	propan
08	elektrika
09	bencin (uporaba v vozilu z dvojnim rezervoarjem)
0A	metanol (uporaba v vozilu z dvojnim rezervoarjem)
0B	etanol (uporaba v vozilu z dvojnim rezervoarjem)
0C	LPG (uporaba v vozilu z dvojnim rezervoarjem)
0D	CNG (uporaba v vozilu z dvojnim rezervoarjem)
0E	propan (uporaba v vozilu z dvojnim rezervoarjem)
0F	elektrika (uporaba v vozilu z dvojnim rezervoarjem)
10	mešanica plina in elektrike (uporaba v vozilu z dvojnim rezervoarjem)
11	bencin (hibridno vozilo)
12	etanol (hibridno vozilo)
13	dizel (hibridno vozilo)
14	elektrika (hibridno vozilo)
15	mešanica goriva (hibridno vozilo)
16	regenerativno (hibridno vozilo)

Tabela 3: Vrsta goriva.[10]

Primer:
ukaz: *0151*
odgovor vmesnika: *41 51 04*

Ugotovimo, da vozilo poganja motor z dizelskim gorivom.

3.3.8 Pretok goriva v motorju

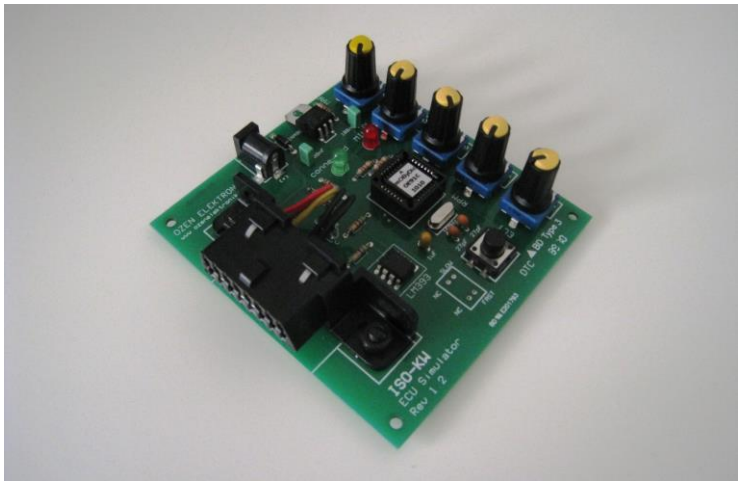
Podatek predstavlja pretok goriva v motorju v l/h.

Primer:
ukaz: *015E*
odgovor vmesnika: *41 5E 02 3C*

Način izračuna vrednosti je podoben izračunu obratov motorja. S preračunom podatka ugotovimo, da je pretok goriva v motorju 28,60 l/h.

3.4 Simulator OBD II

Rešitve iz praktičnih razlogov nismo razvijali ob priklopu vmesnika OBD II na diagnostični priključek DLC sistema OBD vozila. Odločili smo se za nakup simulatorja sistema OBD, ki ga prikazuje slika 8.



Slika 8: Simulator Özen Elektronik mOByDic 1010.

Simulator Özen Elektronik mOByDic 1010[12] omogoča priklop vmesnika OBD II z uporabo signalnega protokola ISO 9141-2. Med širokim naborom funkcij[13], ki jih omogoča, smo med spoznavanjem sistema OBD in razvojem rešitve uporabljali naslednje parametre PID:

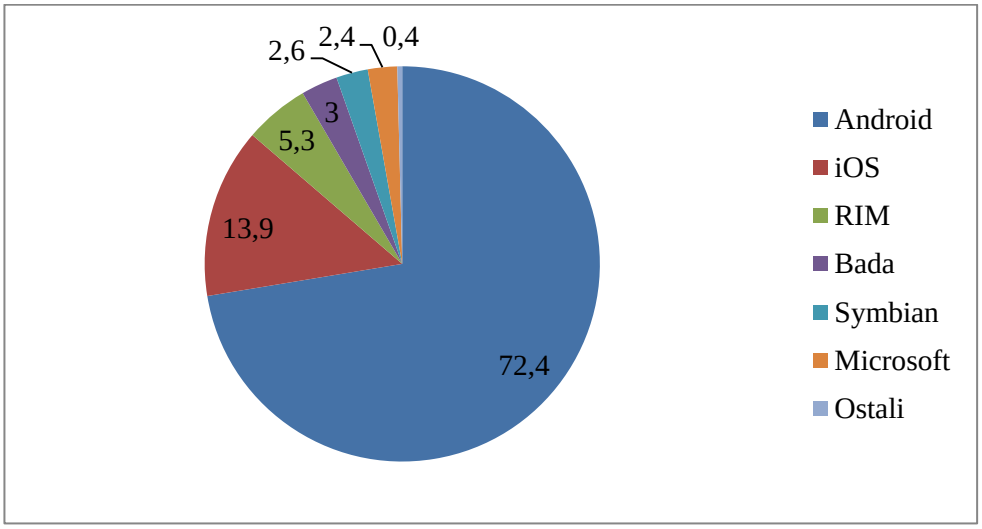
- 00: podprti parametri PID od 01 do 20,
- 0C: obrati motorja,
- 0D: hitrost vozila,
- 0F: temperatura vsesanega zraka in
- 10: senzor masnega pretoka zraka.

Žal simulator ne podpira parametrov PID 0B (senzor razdelilnika absolutnega tlaka), 51 (vrsta goriva) in 5E (pretok goriva v motorju), zato smo njihovo testiranje opravili v vozilih.

3.5 Mobilna aplikacija za operacijski sistem Android

Odločili smo se za razvoj aplikacije za operacijski sistem Android[14]. Razlogi za odločitev so bili naslednji:

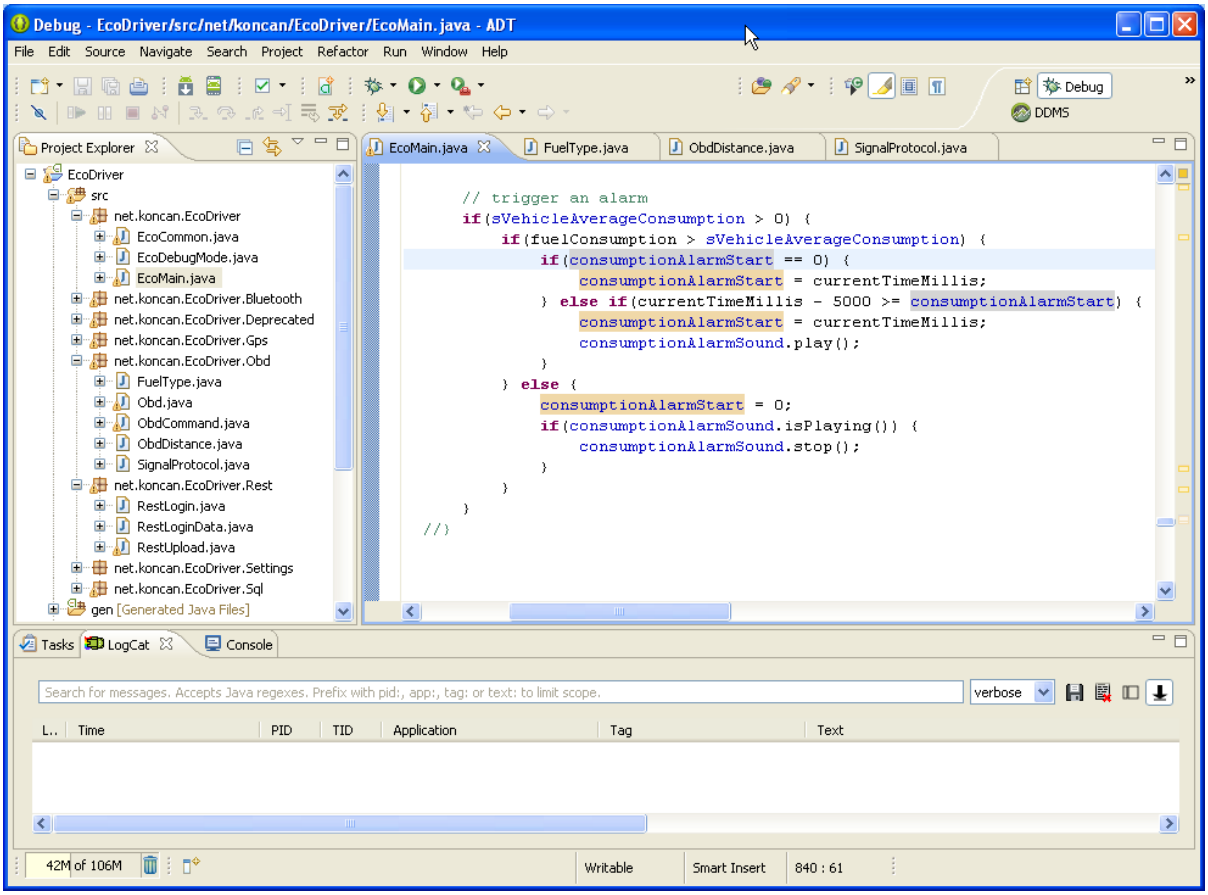
- velika razširjenost operacijskega sistema (slika 9),
- poznavanje programskega jezika Java,
- možnost razvoja v operacijskem sistemu Microsoft Windows,
- možnost uporabe razvojnega okolja Eclipse IDE z vtičnikom ADT (slika 10) in
- možnost testiranja na mobilnih telefonih Google Nexus S in HTC One V.



Slika 9: Delež prodanih mobilnih naprav glede na operacijski sistem v 3. kvartalu leta 2012.[15]

Za razvojno okolje smo uporabili paket Android SDK[16], ki vsebuje vse potrebne komponente za razvoj aplikacije:

- Eclipse IDE z ADT (slika 10),
- Android »SDK tools«,
- Android »Platform-tools«,
- zadnjo različico platforme Android,
- zadnjo različico sistemske slike Android za emulator.



Slika 10: Razvojno okolje Eclipse IDE z vtičnikom ADT.

V začetni fazi smo si za testiranje razvoja aplikacij za operacijski sistem Android pomagali z emulatorjem (slika 11). Ker emulator zaradi omejitev[17] ne omogoča uporabe povezave Bluetooth, smo razvoj aplikacije nadaljevali s testiranjem na mobilnem telefonu. Možna rešitev bi bila tudi namestitev operacijskega sistema Android na osebni računalnik, s pomočjo projekta Android-x86.[18]

Slika 11: Emulator operacijskega sistema Android 4.0.[17]

Pomemben dejavnik pred začetkom razvoja aplikacije za operacijski sistem Android je bil tudi izbor najnižjega nivoja API, ki ga bo aplikacija podpirala. Nivo API je številčna vrednosti, ki enolično določa različico ogrodja API, ki jo omogoča posamezna različica platforme Android. Vsaka naslednja različica platforme Android vključuje nadgradnjo predhodne različice in hkrati zagotavlja združljivost z vsemi predhodnimi različicami.[19]

Ker smo želeli razviti aplikacijo za čim večje število uporabnikov, smo se odločili, da uporabimo nivo API 10 in s tem podpremo delovanje aplikacije v vseh napravah z različico operacijskega sistema Android 2.3.3 ali novejšo. Tabela 4 potrjuje, da smo s tem podprli 88,20 % naprav.

različica	kodno ime	API	delež naprav [%]
1.6	Donut	4	0,20
2.1	Eclair	7	2,40
2.2	Froyo	8	9,00
2.3 - 2.3.2	Gingerbread	9	0,20
2.3.3 - 2.3.7		10	47,40
3.1	Honeycomb	12	0,40
3.2		13	1,10
4.0.3 - 4.0.4	Ice Cream Sandwich	15	29,10
4.1	Jelly Bean	16	9,00
4.2		17	1,20

Tabela 4: Pregled deleža naprav, ki podpira posamezni nivo API.[20]

Z določitvijo nivoja API smo imeli zagotovljene vse pogoje za razvoj aplikacije.

4 Razvoj aplikacije

4.1 Povezava Bluetooth in komunikacijski protokol

Na začetku izdelave aplikacije smo podprli povezavo Bluetooth in protokol za komunikacijo z vmesnikom OBD II.

Omogočili smo naslednje funkcionalnosti:

- vklop adapterja Bluetooth na napravi,
- iskanje vmesnikov OBD II s povezavo Bluetooth in možnost seznanitve naprav,
- prikaz z napravo seznanjenih vmesnikov OBD II,
- vzpostavitev povezave Bluetooth z najdenimi ali s seznanjenimi vmesniki OBD II,
- prekinitev povezave z vmesnikom OBD II,
- samodejna vzpostavitev povezave z zadnje povezanim vmesnikom OBD II ob zagonu aplikacije,
- samodejna vzpostavitev povezave z zadnje povezanim vmesnikom OBD II v primeru nepredvidene prekinitve povezave,
- pošiljanje ukaza vmesniku OBD II (slika 12),
- branje odgovora vmesnika OBD II (slika 13).

```
public void write(String m) {
     m += "\r";
     try {
          if(!m.isEmpty()) {
               mOutputStream.write(m.getBytes());
               mOutputStream.flush();
          }
     } catch (IOException e) {
     }
}
```

Slika 12: Del programske kode, ki skrbi za pošiljanje ukaza vmesniku OBD II. Ukaz se zaključi s kombinacijo znakov »\r«.

```
public void run() {
     byte b;
     StringBuilder m;
     while (true) {
          b = 0;
          m = new StringBuilder();
          try {
               while ((char)(b = (byte)mInputStream.read()) != '>') {
                    m.append((char)b);
               }
               sendToUI(m.toString());
          } catch (Exception e) {
          }
     }
}
```

Slika 13: Del programske kode, ki skrbi za branje podatkov vmesnika OBD II. Branje se zaključi s pojavom znaka »>«.

4.2 Uporaba ukazov OBD in njihovo testiranje

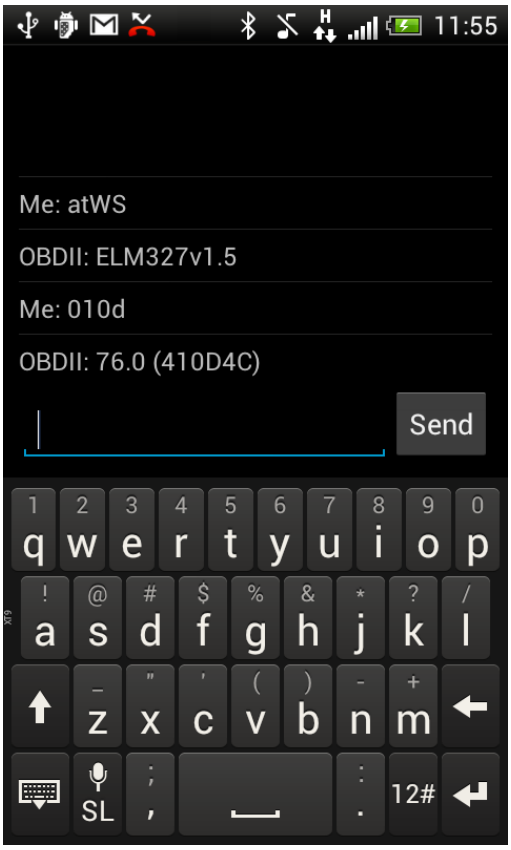
Razvili smo funkcionalnost, ki omogoča enostavno dodajanje in uporabo večine ukazov OBD. Za posamezni ukaz OBD smo omogočili nastavitev:

- aktivnosti ukaza,
- opis ukaza,
- števila bajtov podatka,
- števila vrstic odgovora,
- merske enote,
- vrste številčnega sistema podatka (dvojiški, desetiški, šestnajstiški) in
- formule za preračun vrednosti.

Z aktivnostjo ukaza smo določili ali se ukaz med povezavo z vmesnikom OBD II izvaja ali ne. Posebna ukaza sta ukaza 0100 (podprti parametri PID od 01 do 20) in 0151 (vrsta goriva), ki se ne glede na nastavitev aktivnosti izvedeta vsakič, ko se vzpostavi povezava z vmesnikom OBD II. Ostali aktivni ukazi se izvedejo le v primeru, da jih sistem OBD vozila podpira, s čimer smo pohitrili komunikacijo naprave z vmesnikom OBD II.

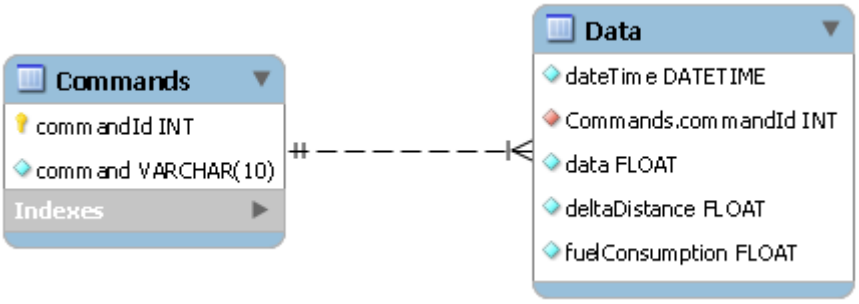
Ukaza 0100 in 0151 sta posebna ukaza, saj vrednosti podatkov nista podani z enačbo. Razvili smo dodatni funkcionalnosti, ki omogočata njuno uporabo. Načina razumevanja podatkov sta opisana v poglavjih 3.3.1 in 3.3.7.

Hkrati smo za potrebe testiranja ukazov OBD razvili funkcionalnost, ki omogoča izpis podatka vmesnika OBD II in vrednosti, izračunane glede na nastavitev ukaza. Slednjo vrednost izpiše le v primeru, da je ukaz OBD v aplikaciji podprt. Primer uporabe vmesnika za testiranje ukazov OBD prikazuje slika 14.



Slika 14: Vmesnik za testiranje ukazov in vrednosti, izračunanih glede na nastavitev ukaza. V primeru ukaza 010D (hitrost vozila) dobimo podatek v vrednosti 76,00 (km/h), preračunani iz šestnajstiške vrednosti 4C. Iz slike je razvidno tudi, da vmesnik OBD II pri sprejemu ukazov ne ločuje med malimi in velikimi črkami.

Ob izvajanju ukazov aplikacija podatke shranjuje v podatkovno bazo SQLite (fizični model prikazuje slika 15), ki jo platforma Android podpira. Podatke smo kasneje uporabili za njihovo analizo in izračun porabe goriva.



Slika 15: Fizični model podatkovne baze aplikacije.

4.3 Optimizacija povezave Bluetooth in komunikacijskega protokola

Veliko časa smo namenili vzpostavitvi hitre in stabilne povezave s sistemom OBD vozila. Težave smo imeli predvsem pri uporabi vmesnika OBD II v vozilu, ki podpira drugačen signalni protokol OBD od vozila, v katerem je bil vmesnik OBD II predhodno uporabljen. Za ta namen smo v aplikaciji podprli možnost izbire signalnega protokola OBD vozila z ukazom »ATSP Ah«. S tem smo poleg zanesljivosti povezave pohitrili tudi čas vzpostavitve komunikacije s sistemom OBD. Možno je uporabiti naslednje nastavitve:

- avtomatska izbira signalnega protokola OBD,
- SAE J1850 PWM (61,6 kb/s),
- SAE J1850 VPW (10,4 kb/s),
- ISO 9141-2 (5 b/s int, 10,4 kb/s),
- ISO 14230-4 KWP (5 b/s int, 10,4 kb/s),
- ISO 14230-4 KWP (fast int, 10,4 kb/s),
- ISO 15765-4 CAN (11 bitov ID, 500 kb/s),
- ISO 15765-4 CAN (29 bitov ID, 500 kb/s),
- ISO 15765-4 CAN (29 bitov ID, 250 kb/s) ali
- SAE J1939 CAN (29 bitov, 250 kb/s).

Za zanesljivo povezavo vmesnika OBD II s sistemom OBD vozila in željeno delovanje aplikacije smo ob vzpostavitvi povezave Bluetooth z vmesnikom OBD II uporabili naslednje ukaze:

- ATWS (ponovni zagon – hitra izvedba),
- ATSP Ah (uporabi signalni protokol h in ga shrani v nastavitve; v primeru neuspeha ga prepozna in shrani v nastavitve),

- 0100 (podprti parametri PID od 01 do 20) in
- 0151 (vrsta goriva).

Povezava Bluetooth z vmesnikom OBD II je mogoča vsakič, ko vmesnik OBD II priključimo na diagnostični priključek OBD vozila. Pri tem ni pomembno ali je vozilo prižgano ali ne. Dokler vozilo ni prižgano, povezava vmesnika OBD II s sistemom OBD vozila ni mogoča. Zato vmesnik OBD II za ukaza 0100, 0151 in ostale aktivne ukaze OBD ne more zagotoviti podatkov. Razvili smo algoritem, ki ugotovi stanje vklopa vozila in, ko je vozilo prižgano oz. se prižge, nadaljuje z izvajanjem ukazov. Algoritem zanesljivo deluje tudi ob večkratnem vklopu in izklopu vozila.

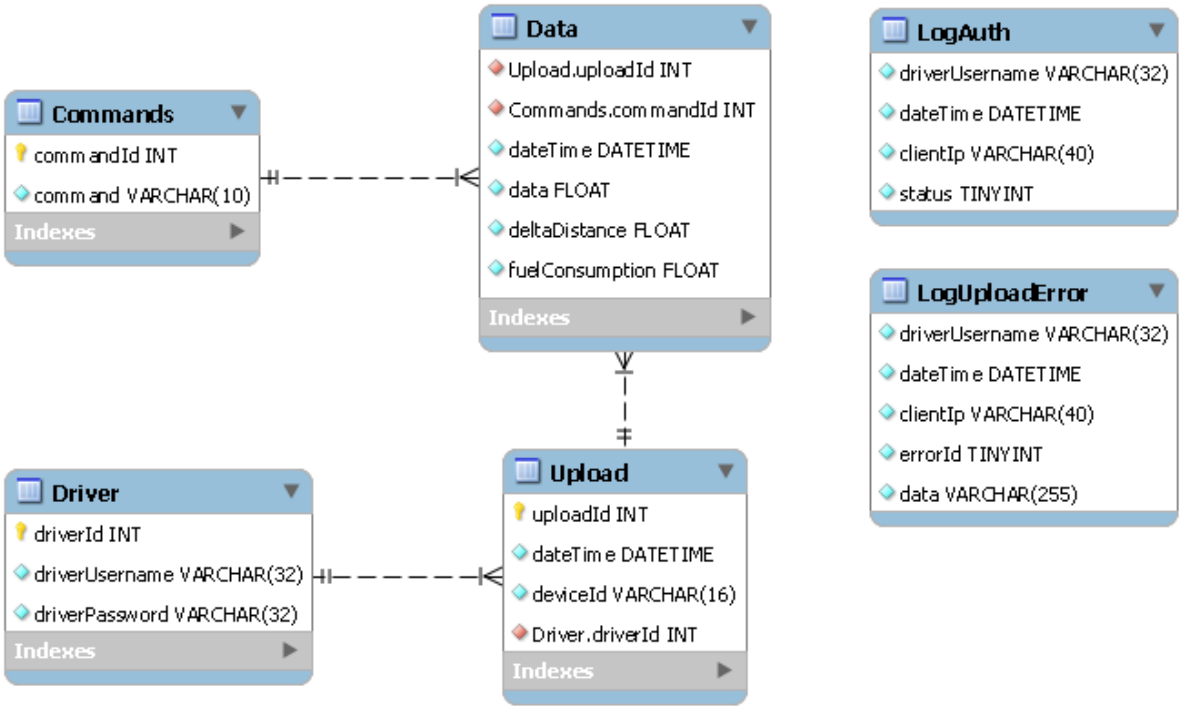
S temi izboljšavami smo zagotovili stabilno in zanesljivo delovanje aplikacije in izvajanje ukazov OBD.

4.4 Prenos podatkov na strežnik

Podatke, shranjene v podatkovni bazi SQLite, smo želeli analizirati na računalniku. Zato smo razvili možnost prenosa podatkov na strežnik s pomočjo arhitekturnega načina REST[21] in uporabo oblike podatkov CSV[22].

Na strežnik z operacijskim sistemom Microsoft Windows smo namestili sistem za upravljanje s podatkovnimi bazami Microsoft SQL Server 2012 Express in spletni strežnik Apache HTTP Server 2.4.2. Na spletnem strežniku smo omogočili uporabo skriptnega programskega jezika PHP 5.4.6. V programskem jeziku PHP smo omogočili gonilnike za povezavo s strežnikom Microsoft SQL Server.

Na strežniku SQL smo ustvarili podatkovno bazo (fizični model prikazuje slika 16) in v programskem jeziku PHP razvili spletno storitev, ki preko metode POST omogoča prenos podatkov aplikacije v podatkovno bazo na strežniku. Za testiranje razvoja spletne storitve smo si pomagali s spletnim brskalnikom Google Chrome 24.0.1312.56 m in njegovo razširitvijo Postman - REST Client 0.8.0.2. Tudi pri razvoju spletne storitve smo uporabili razvojno okolje Eclipse IDE.



Slika 16: Fizični model podatkovne baze na strežniku SQL.

Postopek prenosa podatkov iz aplikacije na strežnik je sledeč:

1. aplikacija preveri obstoj podatkov v podatkovni bazi SQLite in nadaljuje, če podatkovna baza vsebuje podatke,
2. aplikacija se preko mobilnega ali brezžičnega omrežja poveže s spletno storitvijo na strežniku in nadaljuje, če sta povezava in prijava z uporabniškim imenom in geslom uspela,
3. aplikacija v veljavni strukturi pripravi podatke za prenos in hkrati glede na podatke izračunava kontrolni ključ,
4. aplikacija podatke za prijavo skupaj s podatki in izračunanim kontrolnim ključem pošlje spletni storitvi,
5. spletna storitev sprejme podatke, če sta povezava in prijava z uporabniškim imenom in geslom uspela,
6. spletna storitev ob prejemu podatkov preverja njihovo veljavnost in hkrati glede na podatke, po enakem algoritmu kot aplikacija, izračunava kontrolni ključ,

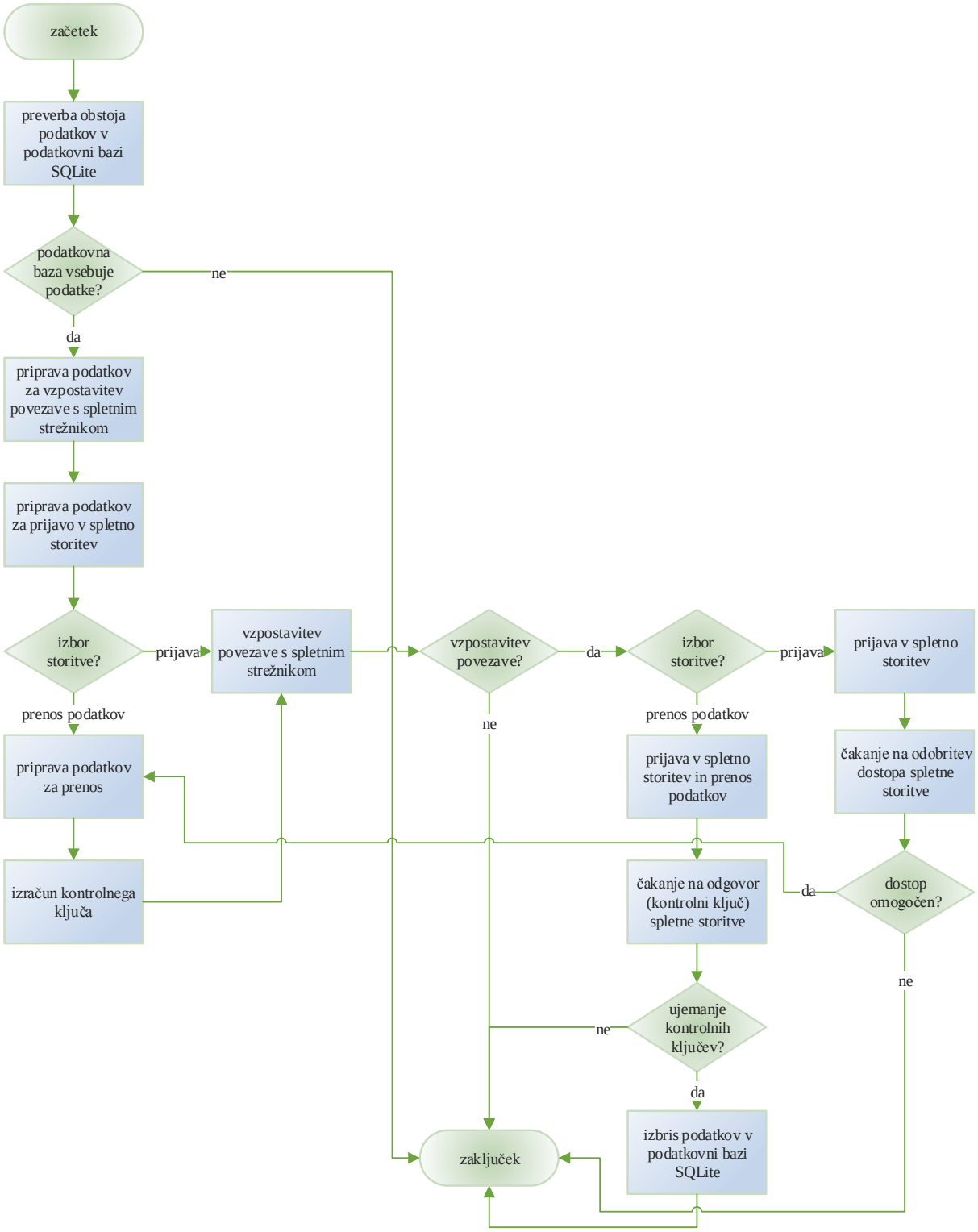
7. spletna storitev v primeru veljavnosti podatkov in ujemanja izračunanega kontrolnega ključa in kontrolnega ključa, izračunanega v aplikaciji, podatke shrani v podatkovno bazo SQL in hkrati aplikaciji odgovori z izračunanim kontrolnim ključem; v nasprotnem primeru spletna storitev prekliče izvajanje prenosa podatkov in vzroke za napako zapiše v podatkovno bazo (primer zapisa je prikazan v tabeli 5),

driverUsername	dateTime	clientIp	errorId	data
matic4	2012-12-12 12:56:44.097	192.168.1.130	3	
matic4	2012-12-12 12:56:44.099	192.168.1.130	6	0;012f;2012-
matic4	2012-12-12 12:56:45.001	192.168.1.130	6	0;0105;2012-12
matic4	2012-12-12 12:56:45.006	192.168.1.130	102	170
matic4	2012-12-12 12:59:56.046	192.168.1.130	7	0;010
matic4	2012-12-12 12:59:56.049	192.168.1.130	4	
matic	2013-01-16 14:37:31.090	80.95.239.1	10	0;AVG_SPEED4;2013 -01-16 14:37:23; 156.66864
matic	2013-01-16 14:37:31.093	80.95.239.1	102	58

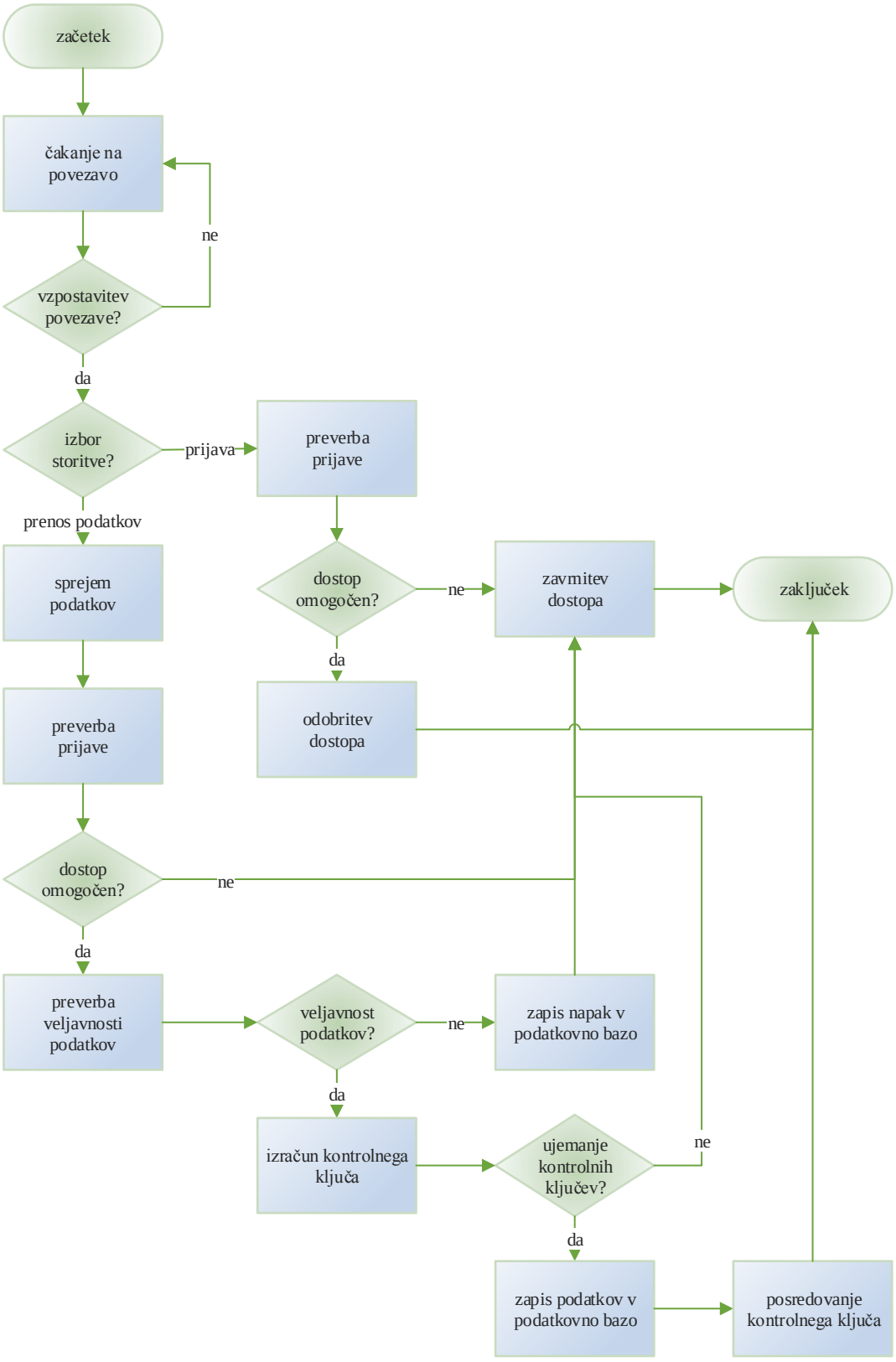
Tabela 5: Primer podatkov v tabeli LogUploadError na strežniku SQL.

8. aplikacija pobriše podatke v podatkovni bazi SQLite, če se kontrolni ključ spletne storitve ujema z lastnim izračunanim kontrolnim ključem. Pobriše le tiste podatke, ki so bili v podatkovni bazi SQLite do prenosa podatkov spletni storitvi.

Natančen postopek prenosa podatkov iz aplikacije na strežnik prikazujeta sliki 17 in 18.



Slika 17: Podatkovni tokovi aplikacije pri prijavi in prenosu podatkov na strežnik.



Slika 18: Podatkovni tokovi spletne storitve ob prijavi in prenosu podatkov aplikacije.

4.5 Trenutna in povprečna poraba goriva

Začeli smo z razvojem funkcionalnosti za izračun trenutne porabe vozila. Ugotovili smo, da ukaz 015E (pretok goriva v motorju v litrih na uro) standarda OBD v vozilih, s katerimi smo testirali aplikacijo, ni podprt. Uporabo ukaza smo v aplikaciji kljub temu podprli.

V vozilih, kjer sistem OBD omenjenega ukaza ne podpira, smo se odločili za izračun trenutne porabe goriva. Pomagali smo si z vrednostjo senzorja masnega pretoka zraka.[23]

S pomočjo enačbe 5 smo izračunali pretok goriva (Φ_g), Uporabili smo vrednost senzorja masnega pretoka zraka (MAF) in stehiometrično masno razmerje zraka in goriva (MR_{zg}). Pri tem smo za masno razmerje zraka in goriva uporabiti vrednost, vezano na gorivo, ki poganja motor vozila. Zaradi razpoložljivosti vozil smo v enačbi uporabili razmerje 14,64:1[24], kar je stehiometrično masno razmerje zraka in bencinskega goriva.

$$\phi_{\mathrm{g}}[\frac{\mathrm{g}}{\mathrm{s}}] = \frac{\mathrm{MAF}[\frac{\mathrm{g}}{\mathrm{s}}]}{\mathrm{MR_{zg}}} \qquad (5)$$

Rezultat enačbe 5 je pretok goriva, podan v gramih na sekundo, ki smo ga pretvorili (enačbe 6-8) v litre na uro. Pri izračunu smo za gostoto bencinskega goriva (ρ_g) uporabili vrednost 737,22 kg/m^3[25] (737.220,00 g/m^3), ki velja pri temperaturi goriva malo pod 16°C.

$$\phi'_{\mathrm{g}}[\frac{\mathrm{m}^3}{\mathrm{s}}] = \frac{\phi_{\mathrm{g}}[\frac{\mathrm{g}}{\mathrm{s}}]}{\rho_{\mathrm{g}}[\frac{\mathrm{g}}{\mathrm{m}^3}]} \qquad (6)$$

$$\phi''_{\mathrm{g}}[\frac{\mathrm{l}}{\mathrm{s}}] = \phi''_{\mathrm{g}}[\frac{\mathrm{dm}^3}{\mathrm{s}}] = \frac{\phi'_{\mathrm{g}}[\frac{\mathrm{m}^3}{\mathrm{s}}]}{1000} \qquad (7)$$

$$\phi'''_{\mathrm{g}}[\frac{\mathrm{l}}{\mathrm{h}}] = \phi''_{\mathrm{g}}[\frac{\mathrm{l}}{\mathrm{s}}] \cdot 3600 \qquad (8)$$

Iz enačbe 9 je razvidna povezava pretoka goriva (enačba 8) in trenutne hitrosti vozila (v_v). Ker je poraba goriva običajno izražena v litrih na 100 kilometrov, smo izpeljali enačbo 10. S tem smo posredno izrazili vrednost senzorja masnega pretoka zraka.

$$\mathrm{P_g}[\frac{\mathrm{l}}{\mathrm{km}}] = \frac{\phi_{\mathrm{g}}'''[\frac{\mathrm{l}}{\mathrm{h}}]}{\mathrm{v_v}[\frac{\mathrm{km}}{\mathrm{h}}]} \qquad (9)$$

$$\mathrm{P_g'}[\frac{\mathrm{l}}{100\mathrm{km}}] = \mathrm{P_g}[\frac{\mathrm{l}}{\mathrm{km}}] \cdot 100 \qquad (10)$$

Ker je vrednost senzorja masnega pretoka zraka iz standarda OBD podana v gramih na sekundo in hitrost vozila v kilometrih na uro, dodatni preračuni enačb 5-10 niso bili potrebni.

Ob vsakokratnem izračunu trenutne porabe goriva, smo podatek shranili v podatkovno bazo SQLite. Poleg porabe goriva smo shranili tudi prevoženo razdaljo vozila od predhodnega do trenutnega zapisa v podatkovno bazo.

Za izračun razdalje smo si pomagali z enačbo 11.

$$\mathrm{s[m]} = \mathrm{s_0[m]} + \mathrm{v_0}[\frac{\mathrm{m}}{\mathrm{s}}] \cdot \Delta\mathrm{t[s]} + \frac{\mathrm{a}[\frac{\mathrm{m}}{\mathrm{s^2}}] \cdot (\Delta\mathrm{t[s]})^2}{2} \qquad (11)$$

Z upoštevanjem enačb 12, 13 in 14 smo dobili enačbo 15, ki smo jo uporabili za izračun razdalje.

$$\mathrm{a}[\frac{\mathrm{m}}{\mathrm{s^2}}] = \frac{\Delta\mathrm{v}[\frac{\mathrm{m}}{\mathrm{s}}]}{\Delta\mathrm{t[s]}} \qquad (12)$$

$$\Delta\mathrm{v}[\frac{\mathrm{m}}{\mathrm{s}}] = \mathrm{v}[\frac{\mathrm{m}}{\mathrm{s}}] - \mathrm{v_0}[\frac{\mathrm{m}}{\mathrm{s}}] \qquad (13)$$

$$\Delta t[s] = t[s] - t_0[s] \qquad (14)$$

$$s[m] = s_0[m] + \frac{(t[s] + t_0[s]) \cdot (v[\frac{m}{s}] + v_0[\frac{m}{s}])}{2} \qquad (15)$$

V enačbah 11-15 uporabljeni simboli pomenijo:
t – čas trenutnega izračuna ($t > t_0$),
t_0 – čas predhodnega izračuna,
s – razdalja v času t,
s_0 – razdalja do časa t_0,
v – hitrost v času t,
v_0 – hitrost v času t_0,
a – pospešek v času t,
Δt – razlika med t_0 do t,
Δv – razlika v hitrosti med t_0 do t.

Da bi dobili povprečno porabo goriva v zadnjih 100 kilometrih, smo vrednosti vsakokratne trenutne porabe goriva v zadnjih 100 kilometrih povprečili.

5 Sklepne ugotovitve

Podatek o trenutni porabi goriva je osnova za razvoj mobilne aplikacije, ki bi voznika vozila opozarjala na preveliko porabo goriva.

Da bi dosegli željeno, smo si pomagali s podatki, ki so dostopni v vozilih preko sistema OBD. Z uporabo sistema OBD v vozilih so začeli v Združenih državah Amerike. Kasneje je uporabo sistema OBD predpisala tudi Evropska unija in druge države.

Z mobilno aplikacijo, izdelano za operacijski sistem Android, smo se preko povezave Bluetooth povezali z vmesnikom OBD II, ki smo ga priključili na diagnostični priključek OBD vozila. Vmesnik ELM327, ki smo ga uporabili, je eden najbolj razširjenih vmesnikov za povezavo računalnika ali druge naprave s sistemom OBD. Naloga vmesnika ELM327 je zagotoviti povezljivost in prenos podatkov med računalnikom ali drugo napravo in sistemom OBD.

Sprva smo se s sistemom OBD seznanjali v vozilih. Za razvoj aplikacije pa smo uporabili simulator sistema OBD, na katerega smo priključili vmesnik ELM327.

V začetni fazi razvoja aplikacije smo omogočili povezavo Bluetooth aplikacije z vmesnikom ELM327 in komunikacijski protokol za izmenjavo podatkov. Razvili smo funkcionalnost, ki omogoča enostavno uporabo kateregakoli ukaza OBD, in vmesnik, s katerim smo lahko katerikoli ukaz testirali.

Posebej veliko pozornosti smo namenili zanesljivosti povezave s sistemom OBD, hitrosti izmenjave podatkov in odzivnosti aplikacije.

Ugotovili smo razliko med ukazoma ATZ in ATWS. Ukaz ATZ izvede ponovni zagon vmesnika ELM327. Pri testiranju slednjega ukaza se v enem izmed vozil med vožnjo resetira računalnik vozila in sproži večkratno zaklepanje vrat. Težavo smo odpravili z zamenjavo ukaza ATZ z ukazom ATWS in zagotovili zanesljivo delovanje aplikacije.

Za pohitritev povezave z vmesnikom OBD II smo v aplikaciji omogočili izbor signalnega protokola OBD, podprtega v vozilu. Pohitritev izmenjave podatkov s sistemom OBD vozila smo dosegli s funkcionalnostjo vmesnika ELM327. Ta omogoča določitev števila vrstic podatkov, ki jih posamezni ukaz pričakuje. Z mislijo na hitrost delovanja aplikacije in izmenjavo podatkov s sistemom OBD smo izvedli tudi optimizacijo programske kode.

Razvili smo algoritem, ki omogoča vzpostavitev povezave s sistemom OBD, ko je vozilo prižgano in zagotavlja nemoteno uporabo aplikacije tudi ob večkratnem vklopu in izklopu vozila.

V aplikaciji smo podprli uporabo podatkovne baze SQLite in omogočili zapis podatkov iz sistema OBD vozila. Ker smo želeli dobljene podatke analizirati, smo na strežnik namestili potrebno programsko opremo in omogočili prenos podatkov neposredno v podatkovno bazo. Prenos podatkov iz aplikacije na strežnik smo kasneje nadgradili in uporabili trinivojsko arhitekturo.[26]

Ugotovili smo, da s standardom OBD nekateri ukazi OBD niso podprti v vseh vozilih. Za izvedbo rešitve bi bilo najbolj enostavno uporabiti ukaz, ki poda trenutno vrednost pretoka goriva v motorju v litrih na uro. Žal ukaz v nobenem od vozil, kjer smo opravili testiranje, ni bil podprt.

Ugotovili smo, da je podatek možno izračunati s pomočjo vrednosti senzorja masnega pretoka zraka.[23] Na razdalji približno 300 kilometrov smo opravili testiranje in ugotovili, da se izračunana poraba od dejanske porabe razlikuje za manj kot 4%. Dejansko porabo vozila smo ugotovili tako, da smo na bencinski črpalki napolnili rezervoar vozila in po prevoženi razdalji na istem točilnem mestu ponovno napolnili rezervoar. Porabo smo izračunali na podlagi prevoženega števila kilometrov in količine natočenega goriva. Za primerjavo smo pred tem v aplikaciji podprli sedem različnih vrednosti porabe goriva in jih primerjali z izračunano dejansko porabo. Ugotovili smo, da je trenutna poraba goriva najbolj natančen približek dejanski porabi.

Po izvedbi izračuna trenutne porabe goriva s pomočjo vrednosti senzorja masnega pretoka zraka smo ugotovili, da ukaz prav tako ni podprt v vseh vozilih. Z raziskovanjem smo ugotovili, da je podatek možno izračunati.[23] Izračun je možen s pomočjo ukaza OBD, ki poda vrednost obremenitve motorja ali s pomočjo vrednosti senzorja razdelilnika absolutnega tlaka. Oba izračuna sta zahtevnejša in zahtevata veliko testiranj, zato ju omenjamo kot možno razširitev funkcionalnosti aplikacije.

Žal izračuna trenutne porabe goriva nismo uspeli testirati na vozilih, ki jih ne poganja motor z bencinskim gorivom. Delovanje smo podprli le v vozilih z motorjem na bencinsko gorivo. Podpora ostalim vozilom bi bila pomembna izboljšava aplikacije.

Dobrodošla izboljšava bi bila podpora anglosaškemu merskemu sistemu (imperialne enote). V aplikaciji smo uporabili najbolj razširjen Mednarodni sistem enot (SI), ki je v rabi tudi v Sloveniji.

Poleg trenutne porabe goriva smo v aplikaciji podprli tudi izračun povprečne porabe goriva zadnjih 100 kilometrov. Razdaljo smo izračunali s pomočjo podatka o hitrosti vozila (iz sistema OBD) in časa.

Testiranje izračuna razdalje smo opravili na daljši razdalji (približno 300 kilometrov). Ugotovili smo odstopanje približno 1,5 kilometra od dejanske prevožene poti. Ko smo delovanje testirali na krajših razdaljah v mestu, smo ugotovili, da se odstopanje pojavi ob ustavljanju in zavijanju.

Razdaljo bi bilo možno natančneje izračunati s pomočjo sprejemnika GPS, ki ga nekatere mobilne naprave vsebujejo. Pri izračunu bi bilo poleg spremembe geografske širine in višine potrebno upoštevati tudi spremembo nadmorske višine.

Na koncu smo v aplikaciji omogočili vnos vrednosti porabe goriva, ki jo navaja proizvajalec vozila. Vneseno vrednost aplikacija primerja s povprečjem trenutne porabe zadnjih nekaj sekund, ki jih dobimo iz sistema OBD. Če so odstopanja prevelika, aplikacija na to opozori s ponavljajočim zvočnim opozorilom. Namesto vnosa vrednosti porabe goriva, ki jo navaja proizvajalec, bi bila dobrodošla uporaba algoritma, ki bi prepoznal optimalno porabo goriva v vsakem trenutku.

V primerjavi porabe nismo upoštevali načina vožnje, kjer bi razlikovali glede na vožnjo po mestu, na obrobju ali na avtocesti. Poraba goriva se v teh primerih lahko bistveno razlikuje. S primerjavo povprečnih podatkov zadnjih nekaj sekund na enostaven način zgolj izločimo občasna odstopanja (npr. zaradi pospeševanja).

Uporaba aplikacije v vozilih je zgolj ena od možnosti, ki omogoča prihranek porabe goriva in manjši vpliv na okolje. Na to lahko še vedno največ vpliva voznik s svojim odnosom do vozila in narave.[27]

Viri

[1] (2013) European Environment Agency - Prostorska ocena delcev PM_{10} in koncentracije ozona v Evropi (2005). Dostopno na: http://www.eea.europa.eu/publications/spatial-assessment-of-pm10-and-ozone-concentrations-in-europe-2005-1.

[2] (2013) Agencija Republike Slovenije za okolje - Onesnaženost zraka z delci PM_{10} in $PM_{2.5}$. Dostopno na: http://kazalci.arso.gov.si/?data=indicator&ind_id=388#goal.

[3] (2013) U.S. Energy Information Administration – Cena surove nafte na trgu. Dostopno na: http://www.eia.gov/dnav/pet/hist/LeafHandler.ashx?n=pet&s=rbrte&f=m.

[4] (2013) Wikipedia – On-board diagnostics. Dostopno na: http://en.wikipedia.org/wiki/On-board_diagnostics.

[5] (2013) The OBD II Home Page – OBD-II Background. Dostopno na: http://www.obdii.com/background.html.

[6] (2013) United States Environmental Protection Agency – On-Board Diagnostics (OBD). Dostopno na: http://www.epa.gov/otaq/regs/im/obd/index.htm.

[7] (2013) OBD Solutions – What is OBD? Dostopno na: http://www.obdsol.com/articles/on-board-diagnostics/what-is-obd/.

[8] (2013) OBD Solutions – Choosing an OBD Interface. Dostopno na: http://www.obdsol.com/articles/on-board-diagnostics/choosing-an-obd-interface/.

[9] (2013) Elm Electronics – ELM327 (OBD to RS232 Interpreter). Dostopno na: http://elmelectronics.com/DSheets/ELM327DS.pdf.

[10] (2013) Wikipedia – OBD-II PIDs. Dostopno na: http://en.wikipedia.org/wiki/OBD-II_PIDs.

[11] (2013) Wikipedia – MAP sensor. Dostopno na: http://en.wikipedia.org/wiki/MAP_sensor.

[12] (2013) Özen elektronik – EOBD & OBDII ECU simulators. Dostopno na: http://www.ozenelektronik.com/?s=products2.

[13] (2013) Özen elektronik – ISO 9141-2 ECU Simulator v2.0. Dostopno na: http://www.ozenelektronik.com/downs/pdf/oe91c1010.pdf.

[14] (2013) Google – Android. Dostopno na: http://www.android.com/.

[15] (2013) Gartner Newsroom - Gartner Says Worldwide Sales of Mobile Phones Declined 3 Percent in Third Quarter of 2012; Smartphone Sales Increased 47 Percent. Dostopno na: http://www.gartner.com/newsroom/id/2237315.

[16] (2013) Android – Get the Android SDK. Dostopno na: http://developer.android.com/sdk/index.html.

[17] (2013) Android – Using the Android Emulator. Dostopno na: http://developer.android.com/tools/devices/emulator.html.

[18] (2013) Android-x86 – Porting Android to x86. Dostopno na: http://www.android-x86.org.

[19] (2013) Android – What is API Level? Dostopno na: http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels.

[20] (2013) Android – Platform Versions. Dostopno na: http://developer.android.com/about/dashboards/index.html.

[21] (2013) Wikipedia - Representational state transfer. Dostopno na: http://en.wikipedia.org/wiki/Representational_state_transfer.

[22] (2013) Wikipedia - Comma-separated values. Dostopno na: http://en.wikipedia.org/wiki/Comma-separated_values.

[23] (2013) Mp3car.com forum – Calculating MPG from VSS and MAF from OBD2. Dostopno na: http://www.mp3car.com/engine-management-obd-ii-engine-diagnostics-etc/75138-calculating-mpg-from-vss-and-maf-from-obd2.html.

[24] (2013) SVTgarage.com forum – Air Fuel Ratios for different fuel (used for tuning). Dostopno na: http://www.svtgarage.com/showthread.php?211-Air-Fuel-Ratios-for-different-fuel-(used-for-tuning).

[25] (2013) SImetric.co.uk – Specific gravity of liquids. Dostopno na: http://www.simetric.co.uk/si_liquids.htm.

[26] (2013) What Is 3-Tier(Multi-Tier) Architecture And Why Do You Need It? Dostopno na: http://blog.simcrest.com/what-is-3-tier-architecture-and-why-do-you-need-it/.

[27] (2013) U.S. Department Of Energy – Gas Mileage Tips. Dostopno na: http://www.fueleconomy.gov/feg/drive.shtml.