UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Peter Lauko

Android-OBD integracija

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Mentor: doc. dr. Dejan Lavbič

Ljubljana, 2012

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: 00257/2012
Datum: 10.04.2012

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

Kandidat: **PETER LAUKO**

Naslov: **ANDROID-OBD INTEGRACIJA**
ANDROID-OBD INTEGRATION

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Število naprav, priključenih v globalno omrežje, se močno povečuje. Poleg računalnikov in mobilnih telefonov je vedno več senzorjev, ki v svetovni splet posredujejo različne podatke (npr. vreme, stanje na cesti ipd.). Eden izmed bogatih virov različnih podatkov je tudi osebni avtomobil (npr. stil vožnje, poraba goriva glede na različne kriterije, anomalije delovanja ipd.).

V okviru diplomske naloge bi bilo potrebno raziskati možnosti integracijo podatkov iz različnih virov, kjer bi eden izmed teh virov bil tudi avtomobilski računalnik. Dostop do podatkov avtomobila lahko izvedemo s pomočjo OBDII vmesnika, nato pa je potrebno te podatke smiselno integrirati s podatki, ki so na voljo na svetovnem spletu. Omenjeno idejo realizirajte na dejanskem primeru, kjer uporabite podatke iz avtomobilskega računalnika preko OBDII vmesnika in mobilni telefon. Slednji naj vam omogoča povezavo v svetovni splet in uporabo številnih senzorjev (npr. GPS, orientacijski senzor ipd.).

Mentor:
doc. dr. Dejan Lavbič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

Zahvala

Zahvaljujem se mentorju doc. dr. Dejanu Lavbiču za vse predloge, pomoč in strokovne nasvete v času izdelave diplomskega dela.

Posebna zahvala gre tudi mojim staršem in sestri za vso podporo in spodbude v času študija.

Kazalo vsebine

Kazalo slik

Seznam uporabljenih kratic

OBD – On Board Diagnostics (standard za komunikacijo z elektroniko v avtomobilu)

OBDII oz. OBD2 – novejša verzija OBD standarda (ponuja višje hitrosti prenosov in je bolj standardna)

CAN-BUS – omrežje senzorjev in procesorjev v avtomobilu

GPS – Global Positioning System (sistem za pozicioniranje)

OS – operacijski sistem

ALDL – Assembly Line Diagnostic Link (predhodnik OBD standarda)

AVD – Android Virtual Device (androidni emulator, ki ga lahko poganjamo na osebnem računalniku)

MIL – Malfunction Indicator Lamp (signalna lučka na armaturni plošči vozila, ki opozarja na napako pri delovanju motorja)

PID – Parameter Identification (identifikacijska oznaka senzorja v vozilu)

API – Application Programming Interface (specifikacija o komunikaciji med programskimi komponentami)

XML – Extensible Markup Language (jezik oz. pravila formatiranja, ki podatke prikazuje tako, da jih lahko interpretirajo tako ljudje kot računalniki)

POVZETEK

Uporaba OBD komunikacijskega standarda je v osebnih vozilih že nekaj časa prisotna, v Evropski uniji pa celo zakonsko določena od leta 2001. Glavna prednost, ki jo omenjeni standard prinaša, je lažje diagnosticiranje napak. Poleg tega omogoča tudi spremljanje večine parametrov v realnem času, pri nekaterih vozilih pa celo upravljanje komponent, kot npr. pomika stekel, klimatske naprave itd.

Namen tega diplomskega dela je bil raziskati morebitne dodatne vrednosti, ki jih lahko pridobimo s kombiniranjem podatkov, pridobljenih iz vozila prek OBD vmesnika, s podatki, pridobljenimi prek internetne povezave mobilnega telefona. Iz vozila smo pridobivali podatke o trenutni hitrosti, trenutnem številu vrtljajev motorja vozila ter nivoju goriva v rezervoarju. Pridobljene podatke smo obdelali ter prikazovali na zaslonu mobilne naprave skupaj s podatki o vremenski napovedi, lokacijah bencinskih servisov itd. Končni produkt je bila aplikacija, ki vozniku lajša delo, tako da ga sproti obvešča o spremembah pomembnih dejavnikov v vozilu in okolici.

Kot platformo za razvoj aplikacije smo uporabili OS Android. Glavni razlogi za to odločitev so v razširjenosti androidnih mobilnih telefonov, programsekga jezika Java ter pestri izbiri orodij za programiranje. Orodje, ki smo ga uporabljali med razovojem, je IDE Eclipse.

KLJUČNE BESEDE

OBD, on board diagnostics, CAN BUS, Android, Bluetooth, Java, mobilna aplikacija

ABSTRACT

OBD communication standard in cars is used for more then a decade, and it is obligatory by law for the manufacturers to support this standard from the year 2001. The main purpose of this standard is for easier diagnostic of troubles on the engine, although it can be used to monitor engine parameters in real-time. Some vehicles even support controlling some of the components like air conditioning, windows opening/closing etc.

The purpose of this task was to discover new added value which could be gained from combining data from the vehicle via the OBD connector and data, gained from the mobile phones internet connection. We were receiving data about current speed, engine rpm and current fuel level from engine control unit. After we processed all the data in real-time, we viewed it on the mobile device screen together with the current weather report, gas station locations, etc. Our product was an application capable of helping the driver with informing him with key changes in the car and also in the surrounding.

The platform used was Android OS. The reasons for this selection is in popularity of mobile phones with Android OS and Java programming language, and also because of the big number of good Java programming environments. For our development we used Eclipse IDE.

KEY WORDS

OBD, on board diagnostics, CAN BUS, Android, Bluetooth, Java, mobile application

1. Uvod

Kot pomoč pri lažjem diagnosticiranju napak na motorjih osebnih avtomobilov so proizvajalci določili standard za komunikacijo z diagnostičnimi orodji. V osemdesetih letih je bil v ta namen najprej razvit standard ALDL (Assembly Line Diagnostic Link), ki pa ga je kmalu nadomestil modernejši in bolj razširjeni standard OBD [1].

OBD Standard so neprestano dopolnjevali; prve različice so podpirale namreč le prikaz t.i. »MIL« (Malfunction Indicator Lamp) – signalne luči, ki opozarja na napako na motorju, sčasoma pa se je standard razvijal naprej in podpiral vse več funkcij.

V Evropski uniji je za osebne avtomobile z bencinskim motorjem uporaba OBD komunikacijskega priključka zakonsko obvezna od leta 2001, od leta 2004 pa tudi za vozila z dizelskim motorjem. To določa Evropski standard za emisije vozil, ki razvršča vozila po emisijah v EURO razrede. Nekateri proizvajalci avtomobilov so OBD priključek vgrajevali v svoja vozila tudi že pred sprejetjem zakona. V združenih državah Amerike pa je bil OBD standard obvezen že leta 1996 [2][3].

Na tržišču se je v zadnjih letih pojavilo zelo veliko naprav, ki podpirajo OBD standard. Kupci jih lahko uporabljajo za odkrivanje/brisanje napak na elektroniki vozila ter za spremljanje parametrov motorne elektronike. Ker so tovrstne samostojne naprave z lastnim zaslonom in napajanjem sorazmerno drage, obstajajo tudi takšne, ki le posredujejo podatke iz vozila prek USB kabla, WiFi ali Bluetooth povezave.

V tej diplomski nalogi smo uporabili modul, ki nam je nudil podatke prek Bluetooth povezave. Nanj smo se nato povezali z mobilim telefonom, ki je imel nameščen operacijski sistem Android, za katerega smo razvili ustrezno aplikacijo. Ker je razvoj potekal za operacijski sistem Android je to pomenilo razvoj programa v programskem jeziku Java, omembe vreden pa je tudi simulator vozila, ki smo ga razvili za lažje diagnosticiranje Bluetooth povezave ter sprotno testiranje aplikacije.

Da bi voznikom čim bolj olajšali vsakodnevno vožnjo, smo razvili program, ki uporabniku pomaga pri spremljanju parametrov vozila. Program je spisan za operacijski sistem Android, deluje pa v povezavi z Bluetooth OBD modulom. Na ta način ima uporabnik na svojem telefonu ves čas podatke o nivoju goriva, hitrosti, številu obratov motorja itd. Možno bi bilo

dodati prikaz podatkov še z drugih senzorjev, ki pa jih nismo podprli, saj so za povprečnega uporabnika irelevantni.

Cilj je bil obveščati uporabnika o hitrosti, (pre)nizkem nivoju goriva, ter mu pomagati pri odkrivanju napak na motorju vozila. Poleg tega prikazuje aplikacija tudi zemljevid trenutne lokacije vozila in vremensko stanje za trenutno lokacijo. Bistvena pa je dodana vrednost, ki jo dobimo s kombiniranjem teh podatkov. Tako lahko uporabnikov telefon samodejno zazna kritični nivo goriva, avtomatično prek internetne povezave naloži seznam najbližnjih bencinskih črpalk ter uporabnika usmerja proti najbližji. To je le eden izmed možnih primerov uporabe te oz. podobne opreme v vozilu.

V nadaljevanju si bomo najprej ogledali uporabljena orodja oz. tehnologije. Sem spadajo Bluetooth modul za povezavo na vozilo in programska orodja. V tretjem poglavju sledi razlaga povezovanja med uporabljenimi napravami, kjer bo poudarek predvsem na Bluetooth komunikaciji med androidno mobilno napravo ter uporabljenim ELM327 krmilnikom. Za tem sledi opis simulatorja vozila in podrobnejši opis razvite Android aplikacije, na koncu pa še predlogi za izbojlšave ter nadaljnji razvoj.

2. Opis uporabljenih orodij, naprav in tehnologij

V tem poglavju bomo opisali orodja, naprave in različne tehnologije, ki smo jih uporabljali med razvojem programske rešitve. V grobem jih delimo na:

- naprave: krmilnik ELM327, mobilna naprava Android,
- programska orodja: Eclipse IDE, Android AVD.

Podrobnejši opis sledi v naslednjih podpoglavjih.

2.1 Mikrokrmilnik ELM327

Za branje podatkov z OBD konektorja v vozilu smo uporabili Bluetooth modul, osnovan na krmilniku ELM327. ELM327 je verjetno najbolj razširjen mikrokrmilnik za povezovanje na osebna vozila. Podpira mnogo komunikacijskih protokolov, kar je pomembno, kajti praktično vsak proizvajalec vozil uporablja svoj standard. To so standardi, prek katerih komunicirajo naprave v vozilu, npr. motorni računalnik, senzorji itd. ELM327 podpira sledeče komunikacijske standarde [5]:

- SAE J1850 PWM (41.6 kbaud)
- SAE J1850 VPW (10.4 kbaud)
- ISO 9141-2 (5 baud init, 10.4 kbaud)
- ISO 14230-4 KWP (5 baud init, 10.4 kbaud)
- ISO 14230-4 KWP (fast init, 10.4 kbaud)
- ISO 15765-4 CAN (11 bit ID, 500 kbaud)
- ISO 15765-4 CAN (29 bit ID, 500 kbaud)
- ISO 15765-4 CAN (11 bit ID, 250 kbaud)
- ISO 15765-4 CAN (29 bit ID, 250 kbaud)

Krmilnik ELM327 s podporo za Bluetooth povezavo, kakršnega smo uporabili, je prikazan na spodnji sliki (slika 1).

Slika 1: ELM327 Bluetooth modul.

Kot smo že omenili so na trgu različne izvedbe modulov oz. krmilnikov za branje podatkov prek OBD konektorja – bistvena razlika med njimi je v tem, kako se lahko nanje povežemo. Na trgu lahko trenutno kupimo module z različnimi vmesniki, med najbolj razširjenimi pa so:

- moduli s COM vmesnikom,
- moduli z USB vmesnikom,
- moduli s podporo za Bluetooth,
- WiFi moduli.

Prve različice modulov, ki so bazirale na ELM327, so delovale prek COM kabla, a kmalu so jih nadomestile izvedbe z vgrajenim pretvornikom na USB konektor, ki je veliko bolj razširjen. V današnjem času mobilnih naprav in brezžičnih povezav pa so najbolj popularni moduli, na katere se lahko povežemo kar brezžično prek Bluetooth ali WiFi povezave. Primer modula z vgrajenim kablom (USB vmesnik) je prikazan na sliki 2 na naslednji strani.

Slika 2: ELM327 modul - USB izvedba.

V našem primeru smo uporabili različico Bluetooth OBD modula, ki bazira na ELM327, na tržišču pa je dosegljiv že za manj kot 20 €. Ker je to cenovno veliko ugodnejša rešitev kot samostojne enote z lastnim ekranom in napajanjem, so postali ti preprosti moduli zelo priljubljeni med tehnično navdušenimi lastniki avtomobilov. Ker pa te enote nimajo lastnega ekrana, potrebujemo za njihovo uporabo še napravo z ustrezno aplikacijo in podporo povezavi na ELM327 modul; to je bodisi USB priključek ali pa WiFi/Bluetooth kompatibilnost.

2.2 Android

Sprotna testiranja aplikacije so potekala na mobilni napravi Samsung Galaxy X-Cover S5690 (prikazana na sliki 3), ki ima nameščen operacijski sistem Android verzije 2.3.6 Gingerbread. Poleg omenjene verzije operacijskega sistema, ki je namenjen mobilnim telefonom srednjega cenovnega razreda, bi lahko z nekaj (predvsem kozmetičnimi) popravki aplikacijo poganjali tudi na operacijskih sistemih Android 3.x (tablični računalniki) in 4.x (tablični računalniki ter mobilni telefoni višjega cenovnega razreda), a tega v trenutni verziji aplikacije še nismo implementirali.

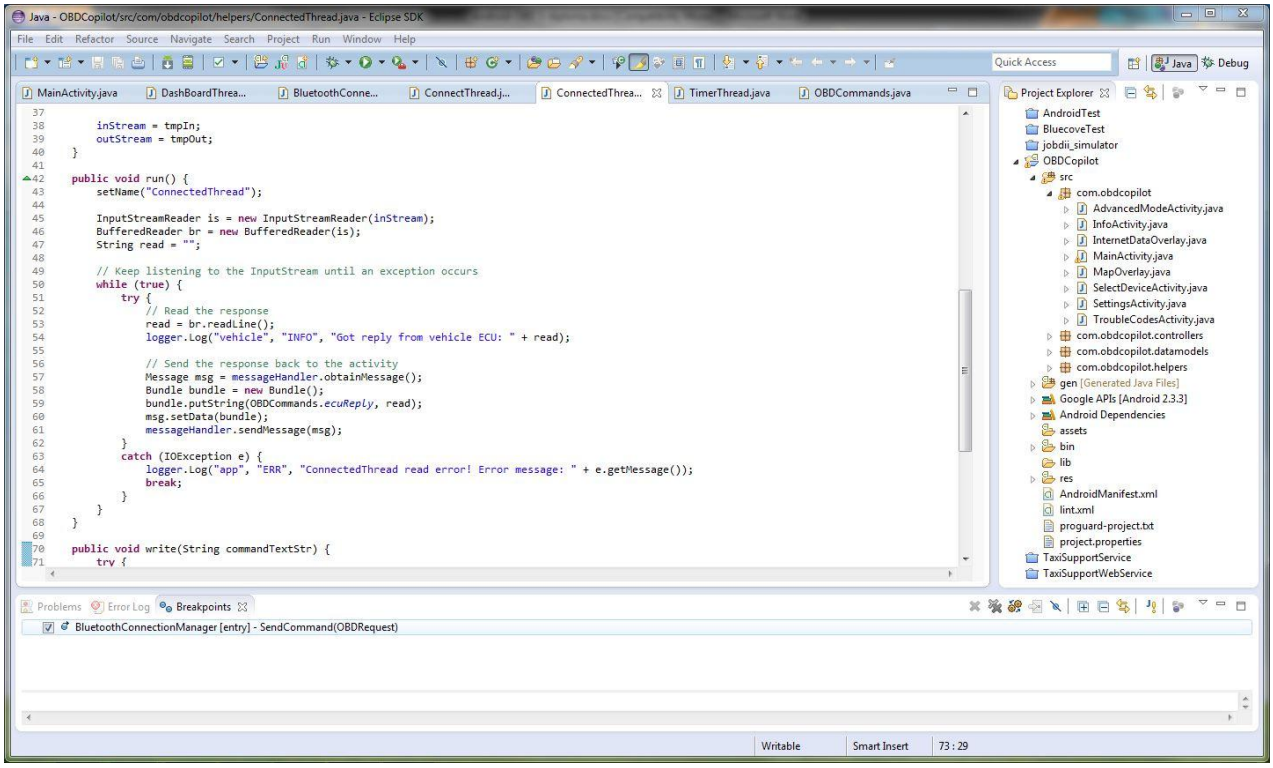
Android naprava za poganjanje našega programa mora imeti Bluetooth modul ter podporo za prenos podatkov. Pri telefonih je to samoumevno, a nekateri tablični računalniki nimajo reže za SIM kartico, kar bi omogočalo prenos podatkov tudi med vožnjo.

Slika 3: Samsung Galaxy Xcover S5690.

2.3 Eclipse IDE

Po tem ko smo pridobili vse potrebne naprave za razvoj aplikacij na platformi Android, smo začeli s kodiranjem programske rešitve in ta aplikacija je osrednja tema našega diplomskega dela. Razvoj je potekal v programskem jeziku Java, kot orodje za razvoj pa smo uporabili Ecplise IDE.

Eclipse je razvojno okolje z odlično podporo za Java programiranje, poleg tega pa nudi še veliko dodatkov za lažje delo za Android platformo. To vključuje tudi AVD (Android Virtual Device) Manager, v katerem lahko poganjamo virtualne Android naprave. Izgled Eclipse razvojnega orodja je prikazan na sliki 4.



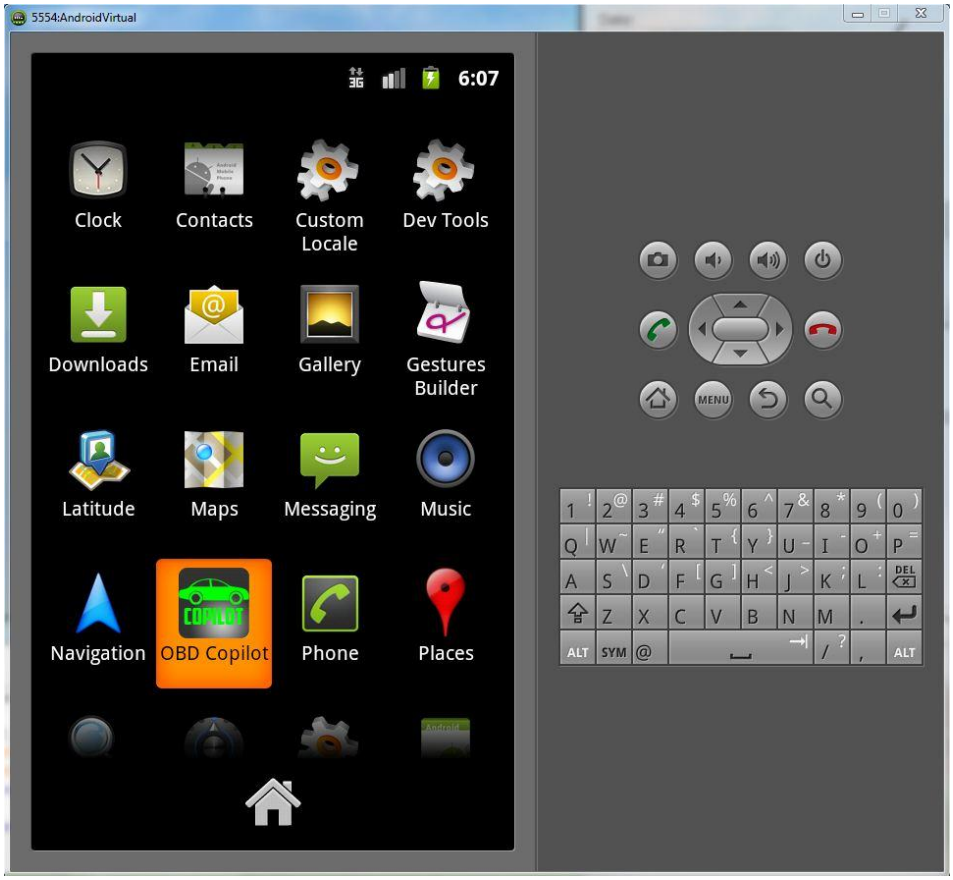
Slika 4: Eclipse IDE razvojno orodje.

V našem primeru je uporaba virtualne naprave Android AVD močno olajšala začetke razvoja aplikacije, saj smo lahko vsako dodano funkcionalnost takoj preskusili na tej navidezni napravi. Za nadaljnji razvoj pa se je AVD izkazal za neuporabnega, saj ne nudi podpore za Bluetooth standard, na katerem je bilo osnovano praktično vse nadaljnje delo. K sreči Eclipse podpira debugiranje/razhroščevanje tudi na mobilnih napravah, ki jih povežemo na razvojni računalnik prek USB povezave. Na ta način smo lahko spremljali razvoj in razhroščevanje tudi na delih aplikacije, ki so vsebovale Bluetooth funkcionalnosti.

2.4 Android AVD

AVD ali Android Virutal Device je programski emulator mobilne naprave Android. S spleta ga lahko prenesemo kot dodatek oz. vtičnik za Eclipse IDE. Integracija AVD in razvojnega orodja Eclipse je zelo dobra in nam je močno olajšala nadaljnje delo.

Emulator je v osnovi kopija mobilne naprave, ki ima implementirano večino lastnosti in funkcionalnosti prave Android naprave. Ima pa seveda tudi pomanjkljivosti; največja med njimi je vsekakor manjkajoča podpora Bluetooth standardu. Izgled AVD emulatorja je prikazan na sliki 5 na naslednji strani. Emulator na sliki prikazuje glavni meni naprave, v seznamu aplikacij pa je označena naša aplikacija.



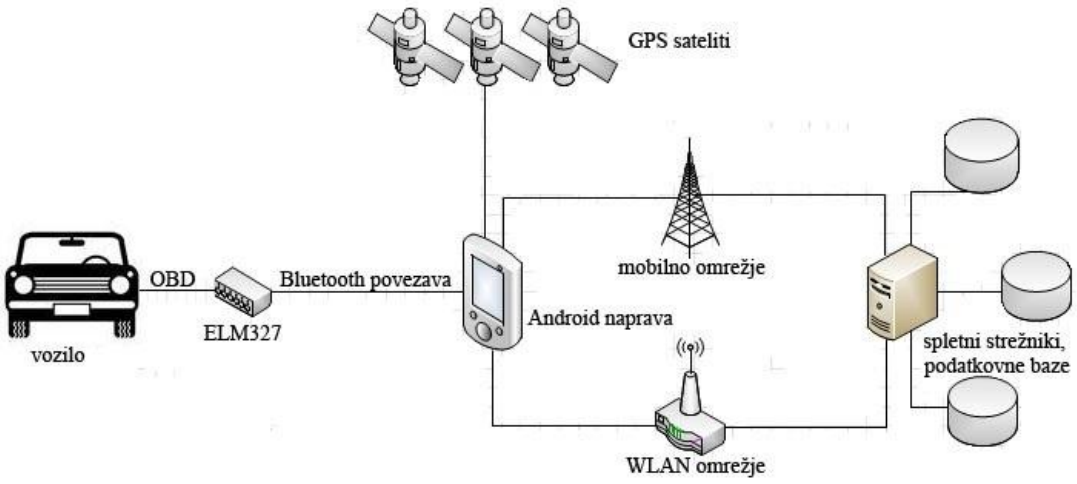
Slika 5: Android AVD emulator.

3. Android mobilna naprava kot pomočnik pri vožnji

V tem poglavju si bomo najprej natančneje pogledali, kako aplikacija dostopa do podatkov, nato pa sledi opis programskega simulatorja vozila. Drugi del tega poglavja bo osredotočen na aplikacijo ter njeno uporabo pri vožnji.

3.1 Dostopanje do podatkov iz različnih virov

Podatki iz avtomobila, pridobljeni preko Bluetooth povezave, so en del prikaza na ekranu mobilne naprave, za ostale podatke pa se mobilna naprava poveže na internet, od koder sproti nalaga zemljevid trenutne lokacije vozila oz. uporabnika, sezname benciniskih črpalk, servisov itd. Poleg tega aplikacija uporablja tudi GPS modul mobilne naprave ter na ta način posodablja trenutno lokacijo. Shema celotnega omrežja informacij oz. podatkovnih poti, uporabljenih med delovanjem aplikacije, je prikazana na sliki 6.



Slika 6: Shema podatkovnih poti.

3.1.1 Povezovanje na ELM327 krmilnik

Ko modul priključimo v OBD konektor v vozilu in obrnemo ključ do pozicije za kontakt, le-ta dobi napajanje in nanj se lahko povežemo z vsako napravo, ki podpira Bluetooth in ima primerno programsko opremo. Omenjeni konektor se v večini vozil nahaja nekje pod volanom oz. pod armaturno ploščo. Edina izjema, ki smo jo med razvojem našli, je Renault, kjer se je konektor nahajal pod sredinskim naslonom za roko med voznikovim in sovoznikovim sedežem [4].

Za prenosne računalnike z operacijskim sistemom Windows je kar precej programskih orodij za branje podatkov prek OBD vmesnika že prisotnih na tržišču, a večina je komercialnih in njihove licence so sorazmerno drage. Tudi za mobilne naprave z operacijskim sistemom Android je stanje podobno.

Po uspešni Bluetooth povezavi modul nato sprejema ukaze, jih obdela ter posreduje naprej prek OBD konektorja, nato pa počaka na odgovor motornega računalnika ter odgovor posreduje nazaj napravi prek Bluetooth povezave. Seznam ukazov, ki jih podpira ELM327 [6], je precej dolg, tako da bomo tu navedli le nekaj primerov.

3.1.1.1 Primeri ELM327 ukazov

Sporočila, namenjena ELM327 krmilniku, morajo biti ustrezno formatirana, da jih le-ta pravilno interpretira. Praviloma so to tri oz. štiri znakovna sporočila, ki jim sledi znak za novo vrstico (\n) [7].

Odgovori ELM327 krmilnika so lahko bodisi le potrditveni (»OK«) ali pa vsebujejo vrednost z zahtevanega PID-a. PID je okrajšava za Parameter ID, torej v odgovoru dobimo vrednost nekega parametra v vozilu.

V nadaljevanju je naštetih nekaj primerov.

Ukazi za vzpostavljanje komunikacije:

- ATD – All default – Nastavi vse nastavitve ELM krmilnika na privzete vrednosti.
- ATZ – Reset all – Ponovno zažene ELM krmilnik.
- ATE0/ATE1 – Echo off/on – Vklopi oz. izklopi ponavljanje prejetega ukaza pred vsakim poslanim odgovorom.
- ATI – Identify – Vrne model ELM krmilnika oz. Bluetooth modula.

Ukazi za branje podatkov iz vozila:

- 0101 – ECU PIDs – Vrne seznam motornih računalnikov, s katerih lahko beremo podatke.
- 010C – Engine RPM – Vrne trenutno število vrtljajev motorja.
- 010D – Speed – Vrne trenunto hitrost vozila v km/h.

3.1.2.2 Dekodiranje ELM327 odgovorov

Odgovori krmilnika so v večini primerov enkodirani. Razlog za to tiči predvsem v tem, da lahko na ta način zajamemo večji nabor vrednosti ter tudi negativne vrednosti parametrov. Ker je nabor znakov za odgovor krmilika omejen – odgovori lahko vsebujejo le alfanumerične znake – morajo biti tudi negativne vrednosti senzorjev primerno obravnavane.

Tu je prikazan primer dešifriranja odgovorov na ukaze 0101 in 010C, kot zanimivost pa je omenjen še odgovor na ukaz 0105.

> **0101** (pomen ukaza: pošlji število prisotnih napak – stanje o MIL)

> **41 01 00 04 80 80** (odgovor krmilnika)

Razlaga odgovora krmilnika je naslednja: prva znaka (41) pomenita, da gre za odgovor na ukaz. Druga dva znaka odgovora (01) povesta, na kateri ukaz krmilnik trenutno odgovarja (enaka sta drugemu paru znakov v ukazu). Nato sledita znaka, ki opisujeta število trenutno prisotnih napak na vozilu, ki je v našem primeru 00 oziroma 0. Pomen zadnih treh parov

znakov pa je odvisen od proizvajalca ter od tipa motorja v vozilu – ali gre za bencinski ali dizelski motor.

> **010C** (pomen ukaza: pošlji trenutno število vrtljajev motorja)

> **41 0C 1A EE** (odgovor krmilnika)

Podobno kot v zgornjem primeru lahko tu takoj razberemo, da gre za odgovor (41) na ukaz (0C). Trenutna vrednost je 1A EE, kar je, če pretvorimo iz šestnajstiškega v desetiški sistem, 6894. Ta vrednost še ni čisto prava, saj jo moramo deliti še s 4, da dobimo pravo število vrtljajev motorja, ki znaša 1723 obratov na minuto.

> **0105** (pomen ukaza: pošlji trenutno temperaturo hladilne tekočine)

> **41 05 7F** (odgovor krmilnika)

Vrednost tega odgovora je 7F, kar je v desetiškem sistemu 127. Ker pa morajo biti zajete tudi negativne vrednosti senzorja, moramo od zgornje vrednosti odšteti 40, da dobimo pravilno vrednost v stopinjah Celzija. Na ta način so zajete tudi vrednosti do -40°C. V našem primeru to torej pomeni 127 – 40 = 87°C.

3.1.2 Uporaba GPS modula v operacijskem sistemu Android

Operacijski sistem Android nudi zelo dobro podporo uporabi GPS modula [9], tako da je sama implementacija dokaj preprosta. Na prvi strani naše aplikacije imamo prikazan zemljevid Google Maps, na katerem je z rdečim križcem označena trenutna lokacija uporabnika.

Za prikaz tega križca mora biti izpolnjenih nekaj pogojev:

- GPS na napravi mora biti vključen,
- naprava mora imeti trenutno lokacijo (GPS povezava vzpostavljena).

Nato pa le ob vsaki zaznani spremembi lokacije križec prestavimo na ustrezno mesto ter ekran premaknemo do iste točke na zemljevidu. Tako je prikazani zemljevid ves čas osredotočen na uporabnikovo lokacijo. Trenutna lokacija se posodablja v določenem časovnem intervalu oz. po premiku čez določeno razdaljo. Privzeta nastavitev je 3 sekunde ali 5 metrov, kar se nastavi ob vklopu sledenja GPS (slika 7).

```
130         if (showMapBool) {
131             mapView.setVisibility(View.VISIBLE);
132             mapView.setBuiltInZoomControls(true);
133             mapView.setSatellite(false);
134             mapView.getOverlays().clear();
135 
136             // This will start the GPS module and start receiving location updates
137             locManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
138             locManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 3000L, 5.0f, this);
139 
140             GeoPoint defaultLocation = new GeoPoint((int)(46.044931 * 1E6), (int)(14.488188 * 1E6));
141 
142             // Move to default position
143             MapController mapController = mapView.getController();
144             mapController.setCenter(defaultLocation);
145             mapController.setZoom(18);
```

Slika 7: Odlomek kode, ki skrbi za označitev trenutne lokacije.

Da pridobimo trenutno lokacijo uporabnika, moramo počakati na dovolj dober signal z GPS satelita. Ob zagonu aplikacije se začne iskanje za GPS sateliti, kar ponavadi traja par minut. Po tem ko dobimo signal z vsaj treh satelitov, lahko GPS modul v Android napravi že določi natančen položaj uporabnika – za to poskrbi že GPS modul v sami napravi, tako da implementacije ni bilo potrebno na novo razvijati.

3.1.3 Branje podatkov z interneta

Ker se v današnjem času vse več podatkov nahaja na spletu, je dobra povezljivost ključnega pomena za uspeh in nadaljnji razvoj aplikacije. V našem primeru smo prek internetne

povezave dostopali do lokacij različnih ponudnikov storitev (bencinski servisi, avtomehanične delavnice) kot tudi do vremenske napovedi.

Naša androidna naprava ima tako kot večina tudi WiFi modul, s katerim se lahko povežemo na lokalna WLAN omrežja. To nam v avtomobilu sicer bolj malo koristi, saj imamo med vožnjo po cestah le redkokdaj možnost povezati se na tovrstno omrežje. Vendar pa je bila pri razvoju takšna možnost povezovanja na splet koristna predvsem zaradi stroškov prenosa podatkov, ki jih za uporabo mobilnega omrežja (GPRS/EDGE/3G) zaračunavajo operaterji.

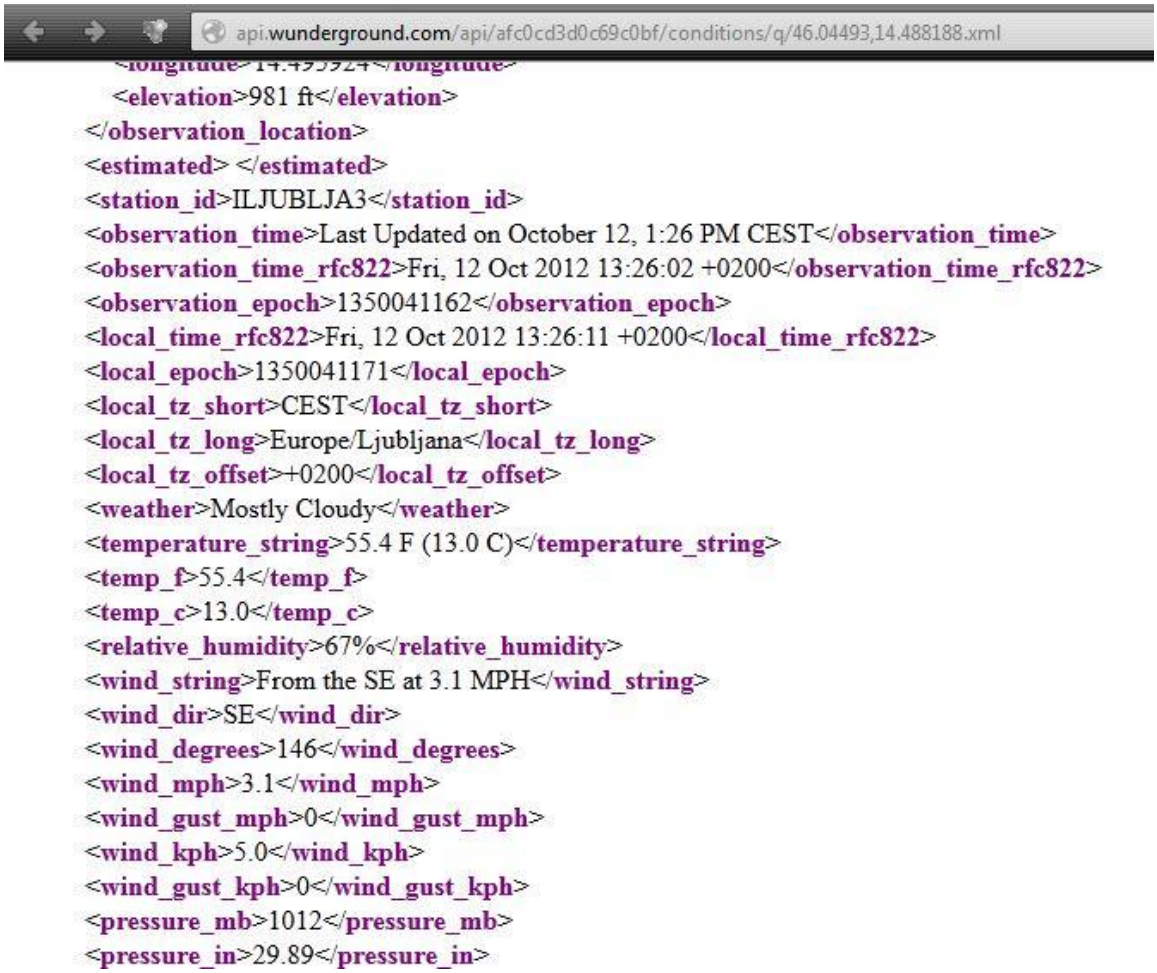
Izbiro načina povezovanja na splet, ali prek WiFi ali prek mobilnega omrežja, opravi že sam operacijski sistem Android, ki avtomatično zazna delujočo WiFi povezavo ter v takem primeru ne nalaga podatkov s spleta prek mobilnega omrežja.

3.1.3.1 Dostopanje do vremenske napovedi

Na spletu je dostopnih kar nekaj API-jev, ki nudijo vremensko sliko oz. vremensko napoved v XML formatu. XML format je zelo priročen za nadaljnjo uporabo v aplikacijah, saj omogoča lažjo manipulacijo z vrednostmi, programer pa lahko na ta način hitreje razvije aplikacijo za prikaz pridobljenih informacij oz. podatkov.

Naša aplikacija je registrirana na spletnem API-ju s podatki o vremenu, imenovanem Wunderground. Ta spletni API omogoča pridobivanje napovedi in trenutnega stanja iz podatnih koordinat, kar je zelo priročno v našem primeru, saj lahko koordinate trenutne lokacije uporabnika sprotno osvežujemo preko GPS.

Na sliki na naslednji strani (slika 8) je prikazana struktura zahtevka, ki ga pošljemo spletnemu API-ju ter XML odgovor, ki ga dobimo z vremensko napovedjo. V naslovni vrstici je viden zahtevek s podano geografsko širino in dolžino, v spodnjem delu okna pa je odlomek odgovora, formatiranega v XML.



Slika 8: Zahtevek in odlomek XML odgovora z vremensko napovedjo.

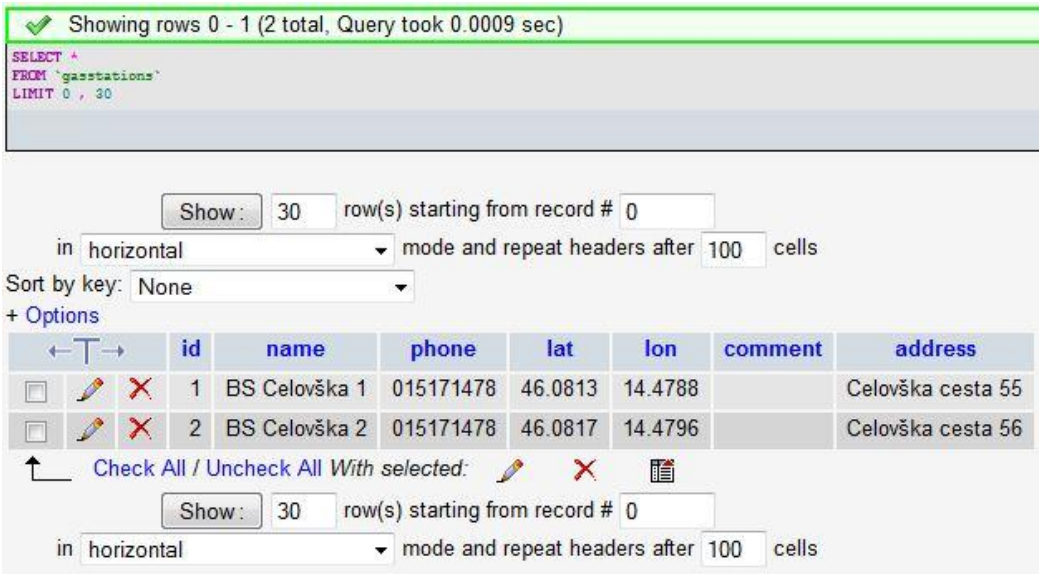
3.1.3.2 Dostopanje do lokacij bencinskih servisov in mehaničnih delavnic

Za prikaz bencinskih servisov in avtomehaničnih delavnic smo potrbovali seznam lokacij z opisi, telefonskimi številkami in koordinatami le-teh. Ker podobnega seznama, kakršnega smo potrebovali, ni bilo nikjer na spletu, smo postavili podatkovno bazo MySQL na testnem strežniku. Baza je vsebovala dve tabeli, v eni smo hranili bencinske servise, v drugi pa avtomehanične delavnice.

3.1.3.3 Podatkovna baza

Uporaba MySQL podatkovne baze se je izkazala za dobro odločitev, saj lahko na ta način posodabljamo sezname lokacij, ne da bi nam bilo treba na mobilno napravo nameščati nove verzije aplikacije. Aplikacija dostopa do spletnega mesta, kjer se nahaja PHP skripta, katere naloga je, da pošlje SELECT poizvedbo na podatkovno bazo. Podatkovna baza sestavi seznam rezultatov poizvedbe, ki ga nato vrne naši mobilni napravi oz. aplikaciji.

Tu smo pri večjem številu rezultatov že lahko opazili krajši zamik v delovanju aplikacije, saj je prenos podatkov na napravo trajal nekoliko dlje. To smo popravili s podajanjem parametrov pri poizvedbi. Na ta način smo lahko npr. prenesli le 10 servisov, ki so bili najbližji naši trenutni lokaciji; za takšno poizvedbo smo morali prenesti podatke o trenutni geografski širini in dolžini uporabnika ter radiju območja, v katerem iščemo želen servis. Primer vrednosti v podatkovni bazi je prikazan na spodnji sliki (slika 9). V poglavju s predstavitvijo aplikacije pa bomo prikazali še prikaz teh podatkov na strani aplikacije.



Slika 9: Primer vrednosti iz testne podatkovne baze.

Za ustrezno interpretacijo podanih parametrov med podatkovno bazo in našo aplikacijo skrbi PHP skripta. Na ta način smo razbremenili podatkovno pot med androidno napravo in spletom, poleg tega pa smo prestavili vso procesiranje z mobilne naprave na stran internetnega strežnika. Za tovrstne operacije računanja razdalj med točkami je SQL jezik zelo učinkovit. Odlomek PHP skripte, ki prejete parametre poda MySQL bazi, je prikazan na spodnji sliki (slika 10).

```
while($row = mysql_fetch_array($result))
{
  if($curLat != null && $curLon != null && $radius != null)
  {
    $noparams = 0;

    $distLat = $row['lat'] * $pi;
    $distLon = $row['lon'] * $pi;

    $r = 6372.797; // Earth radius
    $dlat = $distLat - $curLat;
    $dlon = $distLon - $curLon;

    $a = sin($dlat / 2) * sin($dlat / 2) + cos($curLat) * cos($distLat) * sin($dlon / 2) * sin($dlon / 2);
    $c = 2 * atan2(sqrt($a), sqrt(1 - $a));
    $d = $c * $r;
  }
  else
  {
    $noparams = 1;
  }

  if ($d < $radius || $noparams == 1)
  {
    echo $row['name'] . ';';
    echo $row['phone'] . ';';
    echo $row['lat'] . ';';
    echo $row['lon'] . ';';
    echo $row['comment'] . ";";
    echo $row['address'] . ";";
    echo '<br/>';
    $i++;
  }
}
```

Slika 10: Odlomek PHP skripte s formulo za računanje razdalj.

3.1.3.4 Kombiniranje podatkov pridobljenih iz različnih virov

Ker bi podatkom, pridobljenih z različnih virov, radi dodali vrednost, jih kombiniramo med seboj in s tem uporabniku olajšamo odločanje v različnih situacijah.

Najpreprostejši primer tega je zemljevid s trenutno lokacijo uporabnika. Koordinate lokacije pridobimo prek GPS satelitov, nato jih poiščemo na zemljevidu Google Maps in na ustreznem mestu narišemo indikator trenutne lokacije.

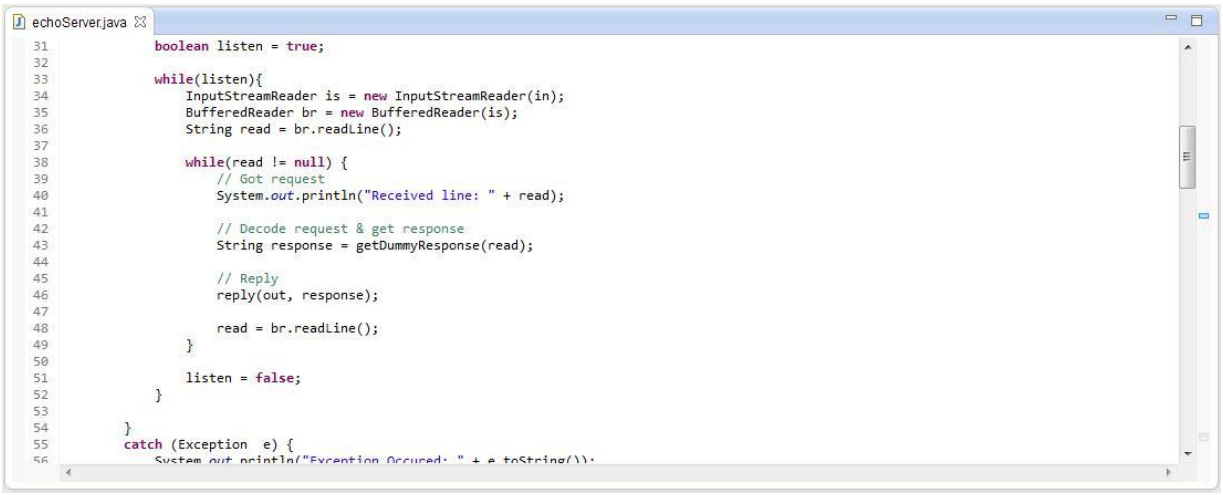
Podobno je z kombiniranjem podatkov, pridobljenih prek Bluetooth povezave (iz vozila); ko namreč zaznamo napako, to prikažemo uporabniku kot obvestilo na ekranu naprave, nato pa prek internetne povezave s spleta naložimo seznam bližnjih servisov. Uporabnik lahko na zemljevidu označi poljuben servis in ga po želji nato tudi pokliče.

3.2 Predstavitev simulatorja vozila

3.2.1 Potreba po simulatorju vozila

Za testiranje Bluetooth povezljivosti smo na začetku uporabili vozilo z OBD modulom, a ker ti moduli nimajo zaslona, je težko vedeti, kaj se s povezavo dogaja. Zaradi tega smo spisali ločeno aplikacijo, ki je delovala kot simulator vozila. Podpira praktično celoten nabor ukazov, ki jih podpira tudi ELM327 modul, le da vrača naključne vrednosti za podatke s senzorjev. Bistvena prednost pa je v tem, da jo lahko poganjamo na osebnem računalniku, tako pa lahko spremljamo na ekranu vse prejete ukaze ter poslane odgovore. To nam je zelo pomagalo pri razvoju aplikacije, saj bi brez tega le težko ugotovili, kaj se dogaja z Bluetooth povezavo.

Osnovna zanka programa oz. simulatorja je prikazana na sliki na naslednji strani (slika 11).



Slika 11: Osnovna zanka simulatorja vozila.

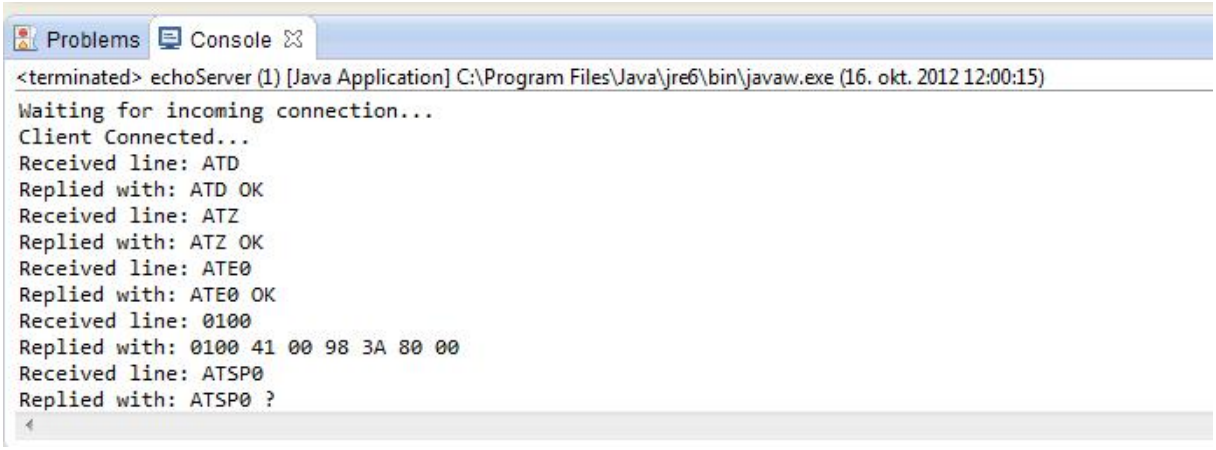
Prikazana zanka programa se ponavlja dokler Android naprave ne prekine Bluetooth povezave. Do takrat pa sledi vrstnemu redu:

- čaka na prejeti ukaz,
- prejme ukaz,
- dekodira ukaz in sestavi primeren odgovor,
- pravilno formatira odgovor,
- pošlje odgovor.

3.2.2 Delovanje simulatorja

Simulator vozila je preprosto orodje, ki na napravi, na kateri ga poženemo, odpre Bluetooth vtičnik (socket) in čaka, da se nanj poveže druga Bluetooth naprava. Po uspešni povezavi naprave preide v stanje odgovarjanja na ukaze. Ko sprejme sporočilo, ga najprej izpiše na ekranu, nato pa prevede v tekstovno obliko, na podlagi katere poišče primeren odgovor. Odgovori so lahko le potrditveni – na te naprava odgovori le z »OK« - ali pa vsebujejo podatke z enega izmed senzorjev. V kolikor naprava ne razume prejetega sporočila, oz. je leto neveljavno, odgovori z vprašajem (?). Odgovori so za lažje razhroščevanje tudi prikazani

na ekranu. Primer povezave mobilne naprave na simulator je prikazan na sliki 12 – simulator smo poganjali v konzoli Eclipse IDE.



Slika 12: Prikaz komunikacije na strani simulatorja.

3.2.3 Prednosti uporabe simulatorja

Poleg omenjene prednosti sprotnega pregledovanja prejetih ukazov in poslanih odgovorov ima simulator še eno veliko prednost pred dejanskim vozilom – aplikacijo smo lahko razvijali brez tega, da bi imeli stalno v bližini vozilo s ključem, obrnjenim do položaja za kontakt.

Med razvojem pa smo tudi ugotovili, da je testiranje oz. odkrivanje težav na aplikaciji z uporabo vozila skoraj nemogoča, saj ne moremo spremljati dogajanja na Bluetooth komunikaciji.

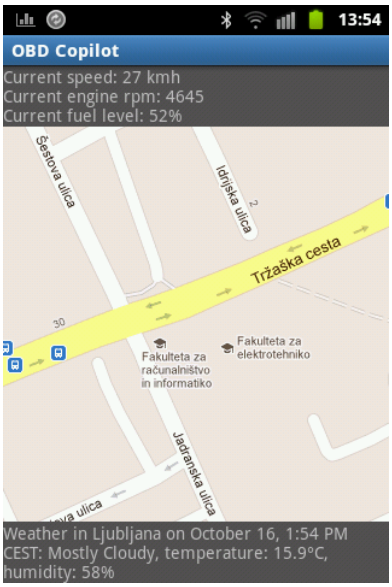
3.3 Predstavitev aplikacije

3.3.1 Prvi zagon aplikacije

Po namestitvi aplikacije na uporabnikovo mobilno napravo ter zagonu le-te, aplikacija najprej ustvari novo mapo »OBDCopilot« na spominski kartici mobilne naprave. V tej mapi se ustvari tudi nova datoteka z nastavitvami, v katero se takoj tudi zapišejo privzete nastavitve. Poleg nastavitev se v isti mapi hranijo tudi datoteke z logi delovanja aplikacije in OBD komunikacije. Vse našteto se zgodi avtomatično, uporabnik mora le še nastaviti pravo Bluetooth napravo kot vir podatkov iz vozila.

3.3.2 Prvi zaslon aplikacije

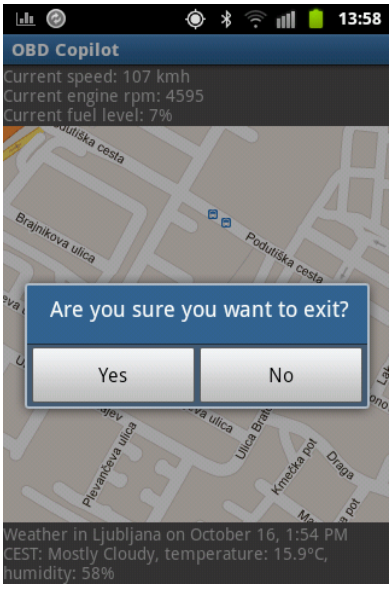
Prvi zaslon aplikacije oz. stran, ki se prikaže, ko poženemo aplikacijo, je sestavljena iz različnih modulov, ki jih lahko izberemo v nastavitvah aplikacije (slika 13).



Slika 13: Izgled začetnega zaslona aplikacije z vključenimi vsemi možnostmi.

Po želji lahko aplikacija prikazuje zemljevid trenutne lokacije vozila oz. uporabnika, podatke iz vozila ter podatke z interneta. Med podatke z interneta spadajo vremenska napoved ter seznam servisov in bencinskih črpalk.

Na prvem zaslonu se prikazujejo tudi t.i. »Toast« obvestila – to so kratka tekstovna sporočila, ki se prikažejo v ospredju za nekaj sekund, ponavadi pa vsebujejo pomembnejša obvestila o trenutnem stanju oz. delovanju aplikacije. Nekatera obvestila dajejo uporabniku tudi možnost izbire – kot primer je spodaj prikazano potrdilo za izhod iz aplikacije (slika 14).

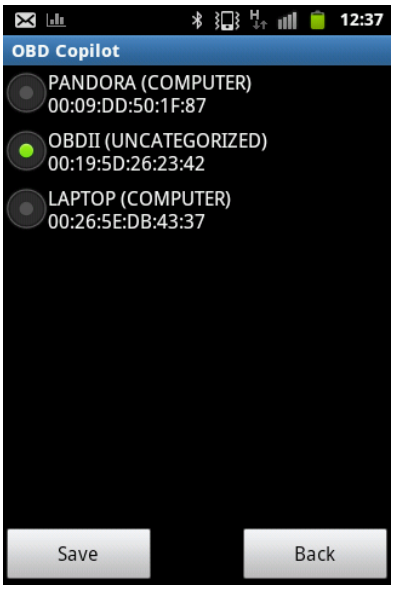


Slika 14: Potrditev izhoda iz aplikacije.

3.3.3 Nastavitve

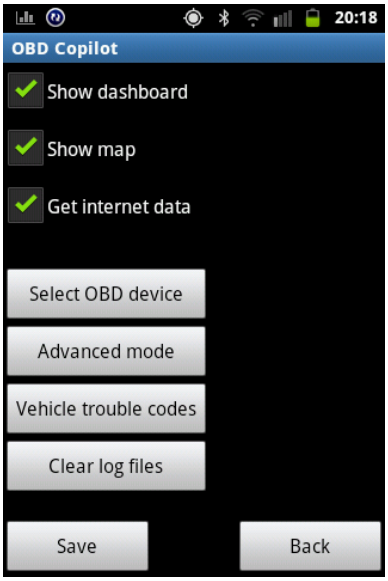
Pred prvo povezavo na Bluetooth OBD modul moramo le-tega nastaviti v nastavitvah. Sam OBD Bluetooth modul moramo najprej spariti z našo mobilno napravo. To naredimo v meniju operacijskega sistema Android – ob tem moramo tudi vnesti geslo oz. PIN Bluetooth modula [8]. Po uspešni sparitvi bo nova naprava vidna na seznamu v nastavitvah naše aplikacije. Ko shranimo novo izbrano napravo in se vnremo na prvo stran aplikacije, bi se že moralo začeti

povezovanje na novo izbrano napravo. Meni za izbiro Bluetooth naprave s seznamom vseh Bluetooth naprav je prikazan na sliki 15.



Slika 15: Meni za izbiro Bluetooth naprave.

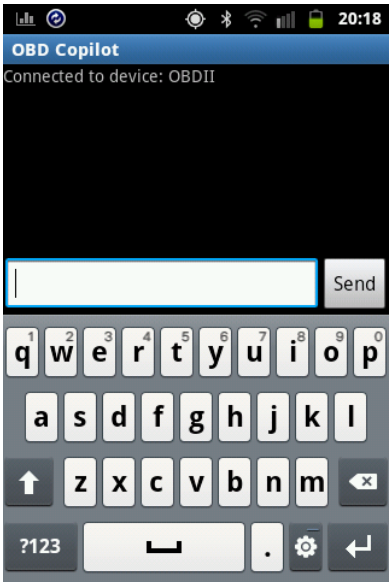
Poleg izbire Bluetooth naprave lahko v nastavitvah določamo tudi elemente prikaza na prvi strani aplikacije. Kombiniramo lahko med prikazom zemljevida, prikazom vremenske napovedi ter prikazom podatkov iz vozila – podrobnosti so prikazane na sliki na naslednji strani (slika 16).



Slika 16: Nastavitve aplikacije.

3.3.4 Meni z naprednimi funkcijami

Meni z naprednimi funkcijami je bil v začetku razvit iz testnih razlogov, toda izkazalo se je, da je zaradi možnosti ročnega vnašanja ukazov lahko zelo koristen za zahtevnejšega uporabnika. Omenjeni meni oz. okno vsebuje tekstovno polje s seznamom že poslanih ukazov in prejetih odgovorov, poleg tega pa še eno polje za vnos novih ukazov in gumb za pošiljanje vnešenega ukaza. Za lažjo predstavo je na spodnji sliki prikazana sestava tega okna (slika 17).



Slika 17: Okno z naprednimi možnostmi.

3.3.5 Meni za branje in brisanje napak na vozilu

Pomembna funkcionalnost spisane androidne aplikacije je tudi možnost detekcije oz. branja napak na vozilu, kot tudi možnost brisanj shranjenih napak (reset).

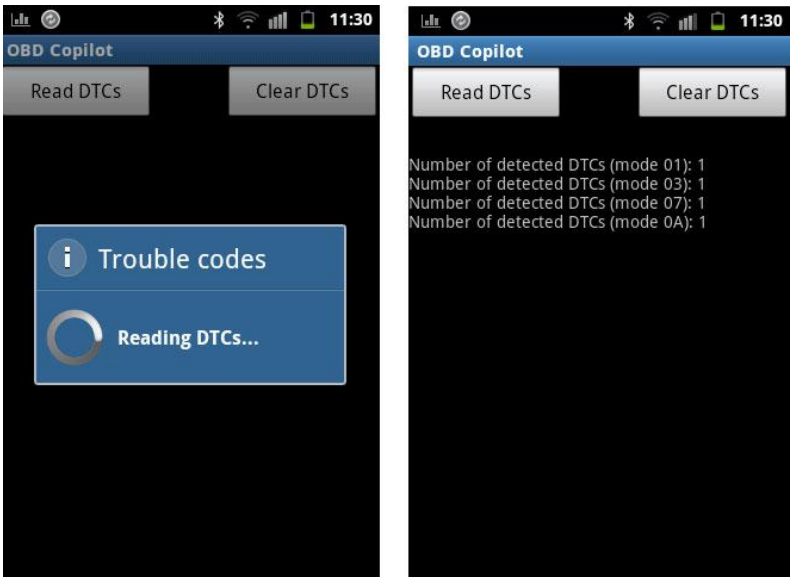
Moderna vozila imajo že zelo dobro razvit samodiagnostični sistem, ki je v nekaterih primerih zelo občutljiv in zazna že rahla odstopanja v vrednostih, prebranih z različnih senzorjev [10]. Kot vozniki lahko tovrstne težave hitro opazimo ob vklopu kontrolne lučke za zaznane težave na motorju (tipičen izgled te lučke je prikazan na sliki 18 na naslednji strani).

V primeru resnejših težav ali v primerih, ko lahko zaradi zaznane napake nastane še večja škoda na motorju vozila, se ponavadi vključi t.i. zasilni program (»limp mode«). Omenjeni program delovanja ponavadi ne deluje več na podlagi podatkov s senzorjev, ampak za delovanje motorja uporablja prednastavljene vrednosti različnih parametrov. Poleg tega ponavadi omeji tudi maksimalno število vrtljajev motorja na neko nizko vrednost.

Slika 18: Tipičen izgled kontrolne lučke za zaznane težave na motorju vozila.

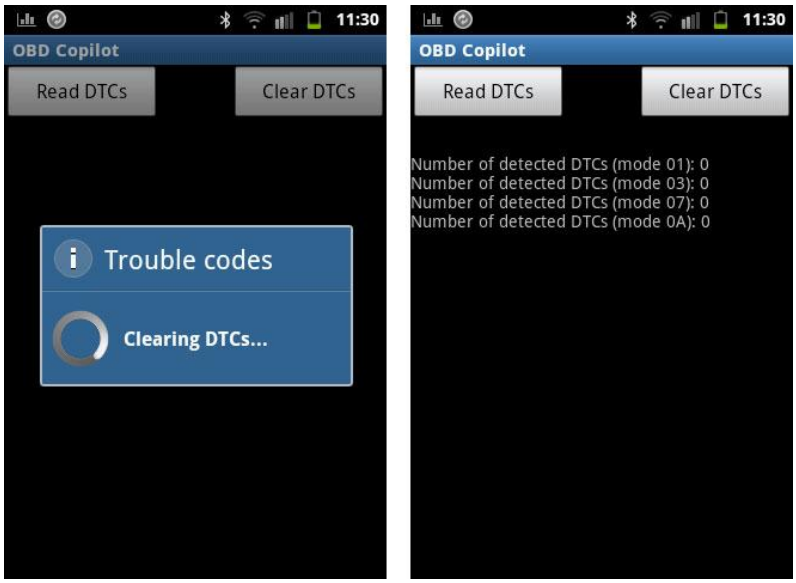
Ker omenjena signalna lučka ne pove praktično ničesar o sami napaki, preostane uporabniku le to, da se odpelje do najbližjega servisa z ustrezno opremo. Naša aplikacija pa nudi možnost, da v primeru detekcije napake to tudi izpiše.

Poleg branja napak je dodana tudi možnost brisanja napak iz spomina motornega računalnika – to je seveda smiselno narediti šele takrat, ko smo napako na motorju že odpravili. Izgled menija ter sam proces branja napak je prikazan na naslednjih slikah (slika 19).



Slika 19: Proces branja napak.

Po končanem popravilu lahko z brisanjem napake v spominu motornega računalnika ugasnemo signalno lučko na armaturni plošči. Proces za brisanje napake prikazujeta spodnji sliki (slika 20).



Slika 20: Proces brisanja napak.

Sledi kratka razlaga različnih tipov napak na vozilu. Napake na vozilu razvrščamo v štiri kategorije (»modes«):

- mode 01 (»Count«): Vrača status kontrolne lučke za napako ter število vseh napak.
- mode 03 (»All«): Vrača seznam vseh zaznanih napak.
- mode 07 (»Pending«): Vrača seznam napak, zaznanih na trenutni oz. zadnji vožnji.
- mode 0A (»Historic/Permanent«): Vrača seznam starih oz. pobrisanih napak.

Med razvojem tega dela aplikacije smo naleteli tudi na manjši problem, ki je posledica tega, da proizvajalci vozil niso zavezani k podpori vseh tipov oz. kategorij napak. Na nekaterih vozilih so tako podprte samo napake tipa 01.

3.3.6 Logiranje

Aplikacija ima implementirano tudi podporo dvonivojskemu logiranju dogodkov. Ta funkionalnost je bila dodana predvsem zaradi lažjega odpravljanja napak pri delovanju, lažjega odkrivanja vzrokov za obstoječe napake ter bolj stabilnega delovanja aplikacije.

Napake ter glavni dogodki v delovanju se beležijo v log, imenovan application.log, podatki o Bluetooth komunikaciji z ELM modulom pa se hranijo v log, imenovan vehicle.log. Na ta način je ločeno beleženje napak ter beleženje ukazov in odgovorov, s tem pa je odpravljanje težav veliko hitrejše in lažje.

Vsak vnos v log je dodan v novo vrstico, vsebuje pa podatke o datumu ter uri dogodka, tipu dogodka (ali gre za napako ali le za obvestilo) ter krajši opis dogodka. Primer vsebine loga s podatki o Bluetooth komunikaciji je prikazan na sliki 21, na sliki 22 pa je prikazana vsebina loga s podatki o aplikaciji. V zadnjem logu je prikazana tudi napaka v delovanju aplikacije – v tem primeru je prišlo do prekinitve Bluetooth komunikacije med mobilno napravo ter ELM327 modulom.

```
11  2012-09-27 16:24:36 INFO Got reply from vehicle ECU: ELM327 v1.5
12  2012-09-27 16:24:36 INFO Got reply from vehicle ECU:
13  2012-09-27 16:24:40 INFO Sending request to ECU: ATE0
14  2012-09-27 16:24:40 INFO Got reply from vehicle ECU: >ATE0
15  2012-09-27 16:24:40 INFO Got reply from vehicle ECU: OK
16  2012-09-27 16:24:40 INFO Got reply from vehicle ECU:
17  2012-09-27 16:24:45 INFO Sending request to ECU: 0100
18  2012-09-27 16:24:48 INFO Got reply from vehicle ECU: >41 00 98 3A 80 00
19  2012-09-27 16:24:48 INFO Got reply from vehicle ECU:
20  2012-09-27 16:24:50 INFO Sending request to ECU: ATSP0
21  2012-09-27 16:24:50 INFO Got reply from vehicle ECU: >OK
22  2012-09-27 16:24:50 INFO Got reply from vehicle ECU:
23  2012-09-27 16:24:56 INFO Sending request to ECU: 010D
24  2012-09-27 16:25:01 INFO Sending request to ECU: 010C
25  2012-09-27 16:25:01 INFO Got reply from vehicle ECU: >SEARCHING...
26  2012-09-27 16:25:01 INFO Got reply from vehicle ECU: STOPPED
27  2012-09-27 16:25:01 INFO Got reply from vehicle ECU:
28  2012-09-27 16:25:06 INFO Sending request to ECU: 012F
29  2012-09-27 16:25:06 INFO Got reply from vehicle ECU: >NO DATA
```

Slika 21: Izsek loga z zapisi o Bluetooth komunikaciji.

```
1  2012-09-27 16:29:28 INFO Application has started...
2  2012-09-27 16:29:38 INFO Starting initial sequence...
3  2012-09-27 16:29:38 ERR ConnectedThread read error! Error message: Socket disconnected
4  2012-09-27 16:29:43 INFO Application shutting down...
```

Slika 22: Izsek loga z zapisi o delovanju aplikacije.

Poleg samega logiranja je v aplikaciji dodana tudi možnost brisanja logov – v trenutni verziji se to še ne proži avtomatsko, ampak le, če uporabnik želi sprostiti nekaj prostora na spominski kartici naprave.

3.4 Uporaba aplikacije

Kot tipičen scenarij za uporabo naše aplikacije smo si med razvojem predstavljali sledeče: nek voznik, ki ima v vozilu nameščen Bluetooth različico ELM327 modula ter Android mobilno napravo, bi rad le-to pred začetkom vožnje povezal z vozilom. Svojo Android napravo (mobilni telefon oz. tablični računalnik) bi nato namestil na neko vidno mesto, kjer bi med vožnjo lahko nemoteno spremljal morebitna opozorila, ne da bi mu to preusmerjalo pozornost od dogajanja na cesti.

Za lažjo orientacijo ali kot pomoč pri iskanu določene lokacije bi mu pomagal zemljevid z označenim trenutnim položajem. Poleg tega bi imel ves čas na voljo tudi posodobljeno vremensko stanje s prikazom trenutne temperature. Pri vozilih, ki nimajo prikaza zunanje temperature, lahko aplikacija opozori voznika na nizko temperaturo oz. možnost poledice v zimskem času.

Podatki iz vozila so ključnega pomena v tej diplomski nalogi, saj je bistvena prednost aplikacije, da zazna kritične vrednosti različnih parametrov v vozilu. Recimo, da nivo goriva v vozilu pade pod določeno vrednost: uporabnik prejme obestilo, poleg tega pa mu aplikacija na zemljevidu pokaže tudi lokacije bližnjih bencinskih servisov. Tako ima uporabnik točne podatke o preostanku goriva, saj je naš prikaz natančnejši od merilnikov na armaturni plošči

vozila, poleg tega pa mu aplikacija prihrani skrb, kje je naslednji bencinski servis in kako daleč je do njega.

3.5 Scenariji uporabe aplikacije

V tem podpoglavju bomo opisali primer, s katerim želimo dokazati prednosti uporabe naše aplikacije, ki kombinira podatke iz različnih virov.

Med vožnjo z vklopljeno in povezano aplikacijo dobi voznik opozorilo: nivo goriva je padel pod 15%. Ta podatek je aplikacija prejela prek Bluetooth povezave z ELM327 modula, ki je podatek prebral iz motornega računalnika vozila. Podatek se tako kot ostali podatki iz vozila osveži približno enkrat na sekundo – toliko namreč traja, da pridejo na vrsto vse zahteve s strani aplikacije, med njimi pa so 200 ms zamiki. Med delovanjem namreč aplikacija pošilja zahteve po vrednosti PID-ov v naslednjem zaporedju:

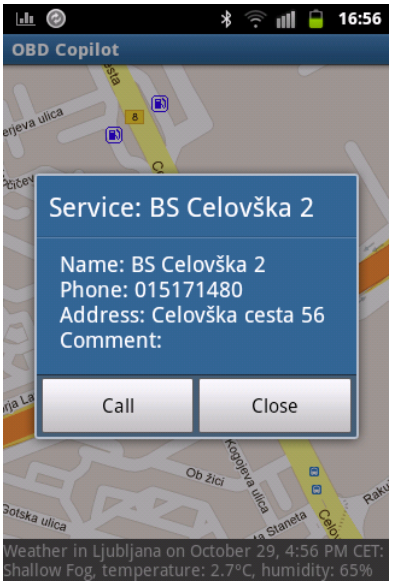
- hitrost,
- število vrtljajev motorja,
- nivo goriva,
- stanje MIL,
- hitrost,
- ...

Po opozorilu se s spleta samodejno naloži seznam bencinskih črpalk in servisov. Za to dejanje aplikacija najprej s pomočjo GPS modula androidne naprave določi natančno lokacijo uporabnika. Potem se poveže na spletni strežnik, kjer je pripravljena PHP skripta, ki vrne podatke iz baze. V primeru, da v tem trenutku lokacija uporabnika še ni bila znana oz. ni bila posredovana na strežnik, se iz baze vrnejo vsi zapisi o črpalkah oz. avtomehaničnih servisih. Na naslednjih slikah (slika 23) je primer, ki prikazuje simbole za lokacije omenjenih storitev (slika je povečava oz. izrez dejanskih zaslonskih posnetkov za boljšo vidljivost).

Slika 23: Simboli za različne tipe lokacij.

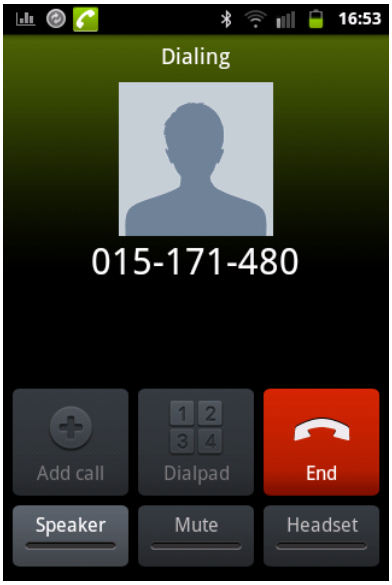
Uporabnik potem s klikom na simbol za črpalko ali avtomehanični servis odpre okno z nadaljnjimi možnostmi – v trenutni verziji aplikacije se tu pojavijo podrobnosti izbranega ponudnika storitev (slika 24).

Po želji lahko lahko ponudnika storitev tudi pokliče na podano telefonsko številko. Ta telefonska številka je del zapisa iz baze, ki je bil predstavljen v poglavju 3.1.3.3, in se je prenesel v napravo skupaj z geografsko lokacijo punudnika storitev.



Slika 24: Podrobnosti bencinskega servisa.

Po kliku na gumb za klic se samodejno vzpostavi telefonska zveza, tako da uporabniku ni treba pretipkavati ali prepisovati številke. Za lažjo predstavo je na spodnji sliki prikazano okno, ki se pojavi ob kliku gumba za klic (slika 25).



Slika 25: Klicanje shranjene številke.

Celoten postopek je sprožila sprememba vrednosti parametra iz vozila, ki je presegala oz. padla pod določeno kritično mejo. Za nadaljnji proces pa je aplikacija uporabila tako GPS modul naprave kot tudi povezavo na splet.

3.6 Problemi in možnosti izboljšav

Že med raziskovanjem o možnostih implementacij pomočnika pri vožnji smo se zavedali, da vseh idej ne bomo mogli dodelati v okviru te diplomske naloge. Tako je nekaj idej in zamisli ostalo neizdelanih in čakajo na naslednjo verzijo aplikacije.

Poleg neimplementiranih funkcionalnosti bi lahko na nekaj mestih izboljšali stabilnost ter omogočili hitrejše delovanje aplikacije. Med trenutnim testiranjem se sicer aplikacija obnaša stabilno, a se kljub vsemu občasno pojavi kakšna napaka. Tu gre predvsem za probleme na

Bluetooth komunikaciji, do katerih pride, ko se npr. povezava nepričakovano prekine (mobilna naprava se oddalji od vozila, ELM modul izgubi napajanje). Pri vseh teh napakah ponovni zagon aplikacije reši nastalo situacijo in nadaljnje delovanje poteka nemoteno.

Nadaljnje možnosti (predvsem vizualnih) izboljšav vidimo na področju prikaza podatkov iz vozila: namesto tekstovnega prikaza vrednosti bi bilo smiselno naredifti grafične prikazovalnike. Podobno velja za prikaz vremenske napovedi, kjer bi lahko uporabili grafično predstavitev trenutnega vremenskega stanja (simboli za sončno, oblačno, itd).

Smiselno bi bilo uporabiti še druge podatkovne baze, saj bi si na ta način prihranili trud sprotnega posodabljanja lastne baze servisov ter bencinskih črpalk. Storitev, dokaj podobno našim potrebam, nudi Google Places.

Z vidika integracije podatkov smo prišli do še ene zamisli, ki pa je žal ostala neimplementirana. Kadar motorni računalnik javi napako in se prenese seznam avtomehaničnih servisov, bi lahko dodali še opcijo za pošiljanje SMS oz. e-mail sporočila serviserju. To morda ne bi bila tako hitra rešitev kot klic, a v sporočilu bi lahko navedli več informacij. Tako bi aplikacija ob zaznavi napake prek OBD vmesnika v SMS sporočilu serviserju lahko poslala kodo napake, shranjene v motornem računalniku. Poleg tega pa bi aplikacija v sporočilo lahko dodala tudi podatek o trenutni lokaciji uporabnika in vlečna služba bi ob morebitni kritični napaki lahko odšla na teren z znano točno lokacijo stranke ter tudi krajšim opisom napake oz. kodo, ki jo je sporočil motorni računalnik.

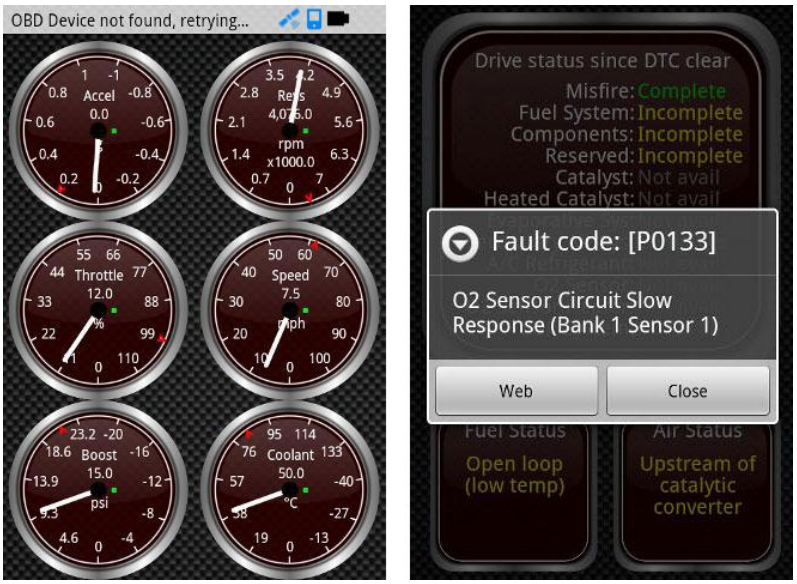
3.7 Primeri podobnih orodij oz. aplikacij

V sledečem poglavju si bomo ogledali sorodne aplikacje. Poleg aplikacij za operacijski sistem Android si bomo ogledali tudi primer aplikacije za operacijski sistem Windows [11]. Slednji morda ni tako priročen za vsakdanjo uporabo kot pomočnik pri vožnji, saj so naprave s tem operacijskim sistemom v povprečju nekoliko večje, a je dober primer razvoja sorodnih aplikacij za drug operacijski sistem.

Tu bi še omenili, da spodnje aplikacije uporabljajo le podatke, pridobljene prek Bluetooth povezave z ELM327 modula (podatki iz vozila). V naši aplikaciji smo dodali še podporo GPS sistemu za določanje lokacije ter uporabo spleta kot vira podatkov. Z izjemo aplikacije Torque, ki podpira sistem GPS, so ostale aplikacije osredotočene izključno na prikaz podatkov iz vozila.

3.7.1 Torque

Torque oz. Torque Lite, kot se imenuje verzija aplikacije, ki ni plačljiva, je glede na število prenosov z Google Play daleč najbolj popularna aplikacija za komunikacijo androidnih naprav z ELM327 moduli [12]. Do sedaj smo testirali le neplačljivo verzijo aplikacije, ki je na vseh vozilih delovala brez težav. Glavne prednosti te aplikacije so dobra preglednost in enostavna uporaba, saj je ogrodje aplikacije le veliko prazno polje, kamor lahko uporabnik sam razmesti želene števce s poljubnimi parametri iz vozila. Izgled aplikacije je prikazan na naslednji sliki (slika 26).



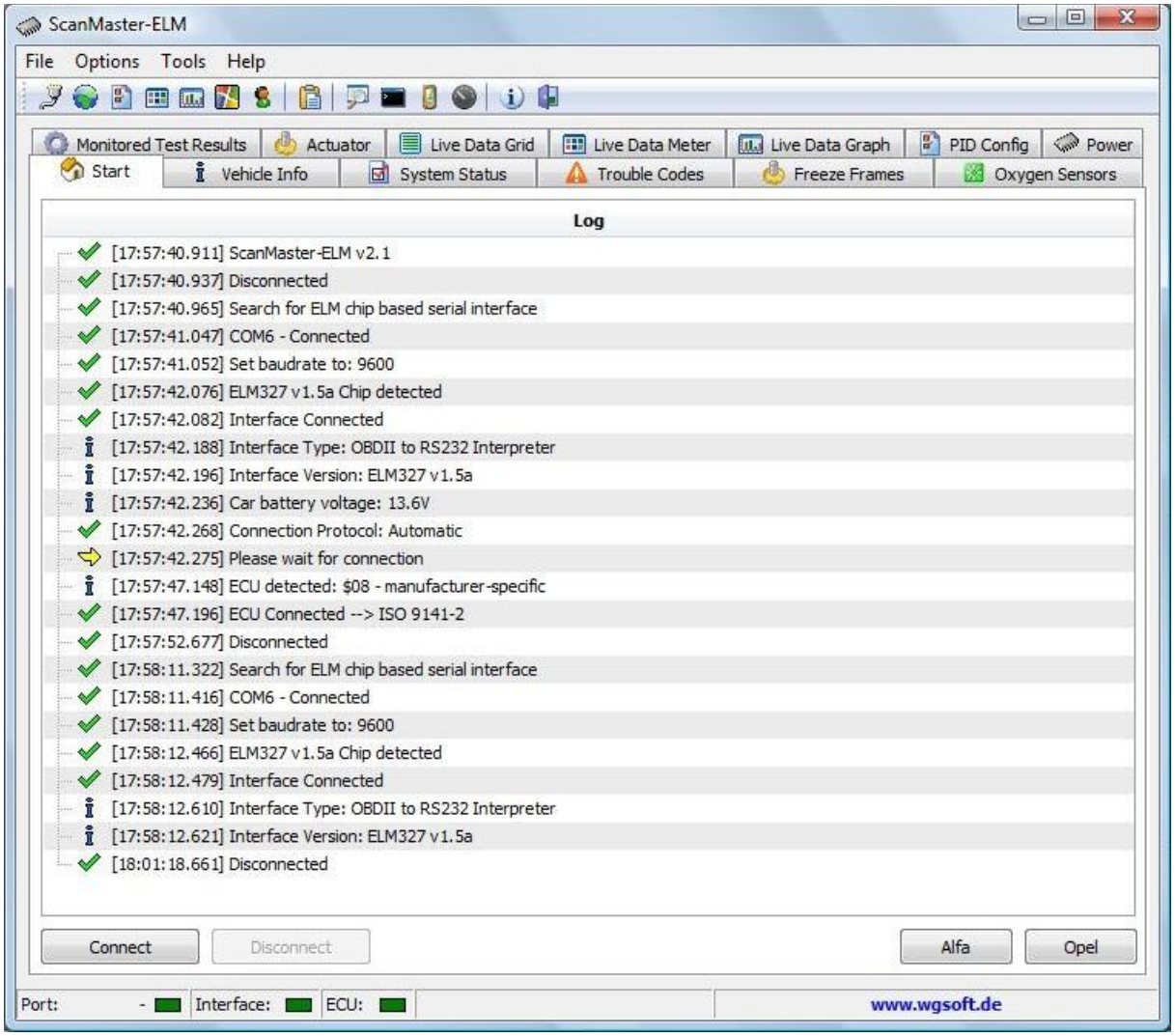
Slika 26: Izgled vmesnika aplikacije Torque.

3.7.2 OBD reader

Android OBD reader je odprtokodna aplikacija, dostopna na Google Code. Tako aplikacija kot tudi vsa izvorna koda sta objavljeni na spletu [13]. Zadnja verzija (1.3) je bila izdana maja 2012, kar je dokaj novo glede na večje število zastarelih odprtokodnih projektov, na katerih se je razvoj ustavil. Na žalost nam povezave do vozila z uporabo te aplikacije ni uspelo vzpostaviti, oz. se je le-ta nemudoma prekinila – vse je kazalo na problem z našim Android mobilnim telefonom oz. ELM327 modulom. Ne glede na to nam je aplikacija oz. njena izvorna koda pomagala pri razumevanju delovanja Bluetooth ter OBD komunikacije.

3.7.3 Scanmaster ELM

Scanmaster ELM je eno izmed plačljivih orodij za branje podatkov prek OBD modula [14]. Razvit je bil za operacijski sistem Windows XP/2000/Vista/7 in nudi pester izbor funkcionalnosti – podpira pregledovanje podatkov s senzorjev v realnem času, prikazovanje grafov, branje ter brisanje napak itd. Poleg tega ponuja tudi t.i. Dyno teste – to so testi pospeškov oz. moči motorja. S to funkcionalnostjo lahko uporabniki brez uporabe merilnih valjev izmerijo moč motorja ter čas za pospešitev do 100 km/h. Glavno okno aplikacije (v zavihkih je možen prikaz ostalih funkcionalnosti aplikacije) je prikazano na sliki 27.



Slika 27: Glavno okno aplikacije Scanmaster ELM.

4. Zaključek, sklepne ugotovitve

Namen te diplomske naloge je bil razvoj pametnega pomočnika pri vožnji, ki bi uporabniku olajšal vožnjo, tako da bi mu nudil pester izbor sprotno posodobljenih podatkov iz vozila in okolice.

Po postavitvi osnovnega načrta aplikacije in seznama opcij, ki jih bomo podprli, smo se lotili programske implementacije. Kasneje smo prvotni načrt rahlo spreminjali ter ga prilagajali potrebam ter novim idejam, a je bistvo ostalo nespremenjeno.

Izbira Android platforme se je izkazala za dobro odločitev, saj je na voljo veliko orodij in pripomočkov za lažji in hitrejši razvoj. Poleg tega pa je omenjena platforma tudi zelo razširjena med morebitnimi končnimi uporabniki. Kot razvojno orodje nam je bil Eclipse IDE v pomoč pri samem programiranju aplikacije, skupaj s programskih emulatorjem Android naprave (AVD) pa se je izkazal kot odlična kombinacija ter dobra izbira tudi za nadaljnji razvoj.

Z uporabo MySQL podatkovne baze smo bili zadovoljni, saj je bila postavitev le-te zelo hitra. Za nadaljnje delo smo potrebovali le še nekaj testnih podatkov, nato pa nismo imeli s podatkovno bazo nič več dela. Za nadaljnji razvoj aplikacije pa bi bilo smiselno opustiti uporabo trenutne baze, saj je bila že v začetku mišljena le kot testno orodje. Na spletu je namreč dostopnih kar precej portalov oz. baz z lokacijami različnih ponudnikov storitev – Google Places je le eden izmed mnogih primerov, ki bi bil smiselno nadomestilo naše trenutne podatkovne baze.

Vse naštete tehnologije nam je uspelo strniti v en izdelek oz. aplikacijo, ki koristi različne vire informacij, ki jih ima povprečna Android naprava na voljo. To je bilo tudi bistvo same naloge, saj so informacije po obdelavi oz. medsebojni povezavi ter pametnemu prikazu veliko bolj uporabne kot pred tem. Voznik ima tako boljši dostop do različnih informacij, ki mu lahko olajšajo vožnjo.

Implementacija podobnih pametnih pomočnikov pri vožnji postaja popularna tudi pri večjih prozvajalcih vozil. Trenutni trend je vpeljava t.i. centralnega vmesnika med voznikom oz. potniki ter vozilom. Proizvajalci imajo tu rahlo prednost pred nami, saj lahko izgled armaturne plošče oz. konzole prilagajajo po potrebi – dodajo LCD zaslon, zvočna opozorila

itd. Podobno velja za senzorje, saj jih lahko po potrebi integrirajo v vozilo že v fazi načrtovanja ter izdelave vozila. Še posebej se to opazi pri najnovejših električnih vozilih, saj imajo mnoga med njimi LCD zaslon, občutljiv na dotik, kot nadomestilo za celotno sredinsko konzolo (na sliki 28 je primer – Tesla Model S).

Slika 28: LCD zaslon, občutljiv na dotik kot nadomestilo sredinske konzole.

Prek vmesnika imajo voznik oz. potniki dostop do nastavitev klimatske naprave, radia, prikaza stanja baterij, navigacije itd. Iz tega je razvidno, da se tudi veliki proizvajalci trudijo povezati različne vire podatkov ter jih prikazovati skupaj, kot povezano celoto. Ob tem pa seveda ne smemo pozabiti na varnost, saj so se z razvojem tovrstnih sistemov razvili tudi novi načini kraje avtomobilov – primer je nova serija vozil proizvajalca BMW [15], pri katerem so tatovi iznašli način, kako odkleniti in zagnati avtomobile, ki imajo vgrajen sistem t.i. »keyless« ključev.

Kot smo lahko videli, ima razvoj tovrstnih aplikacij za pomoč pri vožnji svetlo in obetavno prihodnost. Znanje pa je pri razvoju takšnih aplikacij ključnega pomena. Mi smo med razvojem uporabili znanje z več različnih področij študija na fakulteti za računalništvo in informatiko – poleg splošnega znanja programiranja nam je delo močno olajšalo tudi poznavanje podatkovnih baz, spletnih tehnologij, grafičnih vmesnikov ter še mnogih drugih področij. Brez omenjenga predznanja bi bila izdelava podobne aplikacije veliko težja naloga.

5. Viri

[1] (2012) Wikipedia – OBD. Dostopno na:

http://en.wikipedia.org/wiki/On-board_diagnostics.

[2] (2012) Environmental protection agency (US). Dostopno na: http://epa.gov/obd/.

[3] (2012) Spletna stran OBD Solutions. Dostopno na:

http://www.obdsol.com/articles/on-board-diagnostics/what-is-obd/.

[4] (2102) Wikipedia – OBDII PIDs. Dostopno na:

http://en.wikipedia.org/wiki/OBD-II_PIDs.

[5] (2012) Wikipedia – ELM327. Dostopno na: http://en.wikipedia.org/wiki/ELM327.

[6] (2012) Obdii.com. Dostopno na: http://www.obdii.com/.

[7] (2012) ElmElectronics – seznam ukazov za ELM modul. Dostopno na:
http://elmelectronics.com/DSheets/ELM327DS.pdf.

[8] (2012) Android.com – o Bluetooth. Dostopno na:

http://developer.android.com/guide/topics/connectivity/bluetooth.html.

[9] (2012) Android.com – primeri razvoja. Dostopno na:

http://developer.android.com/tools/samples/index.html.

[10] (2012) About OBD by Kenny Lee. Dostopno na:

http://www.apsense.com/article/167805.html.

[11] (2012) MSDN – Project Detroit. Dostopno na:

http://channel9.msdn.com/coding4fun/detroit.

[12] (2012) Spletno mesto Google Play – Torque Lite. Dostopno na:

https://play.google.com/store/apps/details?id=org.prowl.torquefree.

[13] (2012) Android OBD reader – odprtokodni projekt. Dostopno na:

http://code.google.com/p/android-obd-reader.

[14] (2012) Spletna stran Scanmaster ELM. Dostopno na:

http://www.wgsoft.de/en/menu-products-scanmaster-elm-links.html.

[15] (2012) ZDNet – o varnosti OBD standarda pri proizvajalcu BMW. Dostopno na:

http://www.zdnet.com/hackers-steal-keyless-bmw-in-under-3-minutes-video-7000000507/.