



УНИВЕРЗИТЕТ У НОВОМ САДУ ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департаман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ЗАВРШНИ (BACHELOR) РАД

Кандидат: Никола Пуача

Број индекса: РА58-2011

Тема рада: Једна реализација Андроид сервиса за комуникацију са јединицом за контролу аутомобила

Ментор рада: Др. Иван Каштелан

Нови Сад, Јун, 2015.



УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:	
Идентификациони број, ИБР:	
Тип документације, ТД:	Монографска документација
Тип записа, ТЗ:	Текстуални штампани материјал
Врста рада, ВР:	Завршни (Bachelor) рад
Аутор, АУ:	Никола Пуача
Ментор, МН:	Др. Иван Каштелан
Наслов рада, НР:	Једна реализација Андроид сервиса за комуникацију са јединицом за контролу аутомобила
Језик публикације, ЈП:	Српски / латиница
Језик извода, ЈИ:	Српски
Земља публиковања, ЗП:	Република Србија
Уже географско подручје, УГП:	Војводина
Година, ГО:	2015
Издавач, ИЗ:	Ауторски репринт
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, ФО: (поглавља/страница/ цитата/табела/слика/графика/прилога)	7/25/0/4/10/0/0
Научна област, НО:	Електротехника и рачунарство
Научна дисциплина, НД:	Рачунарска техника
Предметна одредница/Кључне речи, ПО:	сервис, аутомобил, ОБД 2 систем, сакупљач параметара, андроид
УДК	
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, ВН:	
Извод, ИЗ:	У овом раду је реализован сервис који добавља податке од симулатора или аутомобила, као што су брзина, број обртаја итд. Ти подаци ће се добављати из сервиса у клијентску апликацију уз помоћ апи-ја функција који је доступан клијенту и користити се по жељи.
Датум прихватања теме, ДП:	
Датум одбране, ДО:	
Чланови комисије, КО:	Председник: Др. Јелена Ковачевић
	Члан: Др. Небојша Пјевалица
	Члан, ментор: Др. Иван Каштелан
	Потпис ментора



KEY WORDS DOCUMENTATION

Accession number, ANO :	
Identification number, INO :	
Document type, DT :	Monographic publication
Type of record, TR :	Textual printed material
Contents code, CC :	Bachelor Thesis
Author, AU :	Nikola Puača
Mentor, MN :	Ivan Kaštelan, Phd
Title, TI :	One solution of Android service for communication with control unit in Vehicle Infotainment device
Language of text, LT :	Serbian
Language of abstract, LA :	Serbian
Country of publication, CP :	Republic of Serbia
Locality of publication, LP :	Vojvodina
Publication year, PY :	2015
Publisher, PB :	Author's reprint
Publication place, PP :	Novi Sad, Dositeja Obradovica sq. 6
Physical description, PD : <small>(chapters/pages/ref./tables/pictures/graphs/appendixes)</small>	7/25/0/4/10/0/0
Scientific field, SF :	Electrical Engineering
Scientific discipline, SD :	Computer Engineering, Engineering of Computer Based Systems
Subject/Key words, S/KW :	servis, auto, OBD 2 system, acquisition of parameters, android
UC	
Holding data, HD :	The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, N :	
Abstract, AB :	In this paper service has been made for gathering data from simulator or car like, car speed, RPM etc. This data will be forward from service to a client with api functions, which are available to client.
Accepted by the Scientific Board on, ASB :	
Defended on, DE :	
Defended Board, DB :	President: Jelena Kovačević, PhD
	Member: Nebojša Pjevalica, PhD
	Member, Mentor: Ivan Kaštelan, PhD
	Mentor's sign

Zahvalnost

Zahvaljujem se Tomislavu Maruni, Branimiru Kovačeviću, Marku Kovačeviću, Nenadu Jovanoviću, Ivanu Kaštelanu i Mladenu Kovačevu na stručnoj pomoći prilikom izrade ovog rada.

SADRŽAJ

1. Uvod.....	1
2. Teorijske osnove	3
2.1 OBD II.....	3
2.1.1 OBD II poruke	4
2.2 OBD II simulator.....	5
2.3 Android operativni sistem	6
2.3.1 Aktivnosti u Androidu	6
2.3.2 Servisi u Androidu	7
2.4 Android Studio	9
3. Koncept rešenja.....	10
3.1 Analiza problema i opis rešenja	10
3.2 Način rada sistema	11
3.3 Problemi pri projektovanju.....	12
3.4 Zahtevi koje rešenje treba da ispunjava	13
3.4.1 Zahtevi od strane radne memorije	13
3.4.2 Modularnost.....	13
4. Programsko rešenje.....	14
4.1 Moduli servisa.....	15
4.1.1 Modul za pokretanje servisa prilikom pokretanja operativnog sistema	15
4.1.2 Modul aktivnosti.....	16
4.1.3 Glavni modul servisa	16
4.1.4 Modul za ponovno pokretanje servisa	16
4.1.5 Modul za inicijalizaciju konekcije.....	16
4.1.6 Modul za realizaciju funkcija API-ja.....	17

4.1.7 Modul za obradu zahteva.....	19
4.1.8 Modul sa funkcijama za komunikaciju sa automobilom	19
4.1.9 Moduli za obradu zahteva prilikom slanja jednog parametra.....	20
4.2 Modul za servis-klijent komunikaciju.....	20
4.3 Modul klijenta	20
5. Verifikacija i rezultati	21
6. Zaključak	24
7. Literatura.....	25

SPISAK SLIKA

Slika 1 OBD II konektor	4
Slika 2 OBD II simulator	5
Slika 3 Sve verzije Android operativnog sistema	6
Slika 4 Životni vek aktivnosti	7
Slika 5 Životni vek servisa	8
Slika 6 Delovi računara u vozilu	11
Slika 7 Komunikacija novog rešenja sistema	14
Slika 8 Arhitektura servisa	15
Slika 9 Konačan izgled servisa i aplikacije za proveru	22
Slika 10 Servis iz automobila korišćen u aplikaciji za vizualizaciju podataka	23

SPISAK TABELA

Tabela 1 Opis režima rada OBD II sistema	5
Tabela 2 API funkcije	18
Tabela 3 Funkcije za komunikaciju servisa i automobila	20
Tabela 4 Brzina dobavljanja parametara.....	22

SKRAĆENICE

API – **A**pplication **P**rogramming **I**nterface, Programski prilagodni sloj

ECU – **E**ngine **C**ontrol **U**nit, Kontrolna jedinica motora

GUI – **G**raphical **U**ser **I**nterface, Grafičko korisničko okruženje

OBD – **O**n-**B**oard **D**iagnostic, Dijagnostika na uređaju (automobilu)

PID - **P**erformance **I**nformation **D**ata, Informacije o karakteristikama automobila

SAE – **S**ociety of **A**utomotive **E**ngineers, Udruženje automobilskih inženjera

1. Uvod

Razvojem elektronike razvijaju se opcije u automobilu koje čine sve da olakšaju vožnju i učine je sigurnijom. Tako je nastao računar u vozilu koji obezbeđuje ogromne pogodnosti vozaču. Svaki računar u vozilu ima operativni sistem koji obezbeđuje najrazličitije funkcije kao što su prikaz navigacije, kontrolisanje sistema glasovnim komandama, telefoniranje itd.

U ovom radu je opisano jedno rešenje Android servisa koji služi za komunikaciju sa kontrolnim jedinicama automobila, kao i Android aplikacije za ispitivanje pravilnog očitavanja podataka o dešavanjima u vozilu i njihovog predstavljanja u računaru vozila. Ova dešavanja uključuju pre svega poruke o greškama (kvarovima na vozilu), ali i ostale informacije o vozilu kao što su: brzina, nivo goriva, poziciju papučice za gas, broj obrtaja motora itd. Za korišćenje ovog rešenja servisa kao i aplikacije za proveru neophodno je posedovati Android operativni sistem i podršku za bluetooth. Razlog za ovakav izbor je sve veća prisutnost Android operativnog sistema u uređajima koji se koriste u vozilima. Android je najzastupljeniji u oblasti mobilnih telefona, tableta, i računara za vozila (engl. Infotainment u daljem tekstu računara), koja predstavljaju budućnost automobila.

Za dobavljanje informacija od vozila, korišćen je OBD-II (engl. On-board diagnostic) sistem koji se po standardu koristi u svim vozilima u SAD od 1996 godine. Iz razloga zaštite životne sredine i radi lakšeg servisiranja i poboljšanja karakteristika vozila nastao je OBD-II sistem. On tačno definiše način komunikacije između kontrolnih jedinica u vozilu (engl. Control Unit) i spoljašnjih uređaja (npr. Android aplikacija) za bilo koju marku vozila.

U današnje vreme ne postoji potreba da se prilikom kupovine polovnog vozila ide u servis na pregled automobila. Razvoj OBD II sistema je omogućio da se pomoću korisnicima raspoloživih aplikacija očitavaju parametri automobila. Tako da samo upotrebom telefona korisnik može očitati sve podatke, što je najbitnije, videti sve greške (kvarove u vozilu), ako postoje.

Rad je organizovan na sledeći način:

- Teorijske osnove – kratak opis OBD II sistema, njegovih PID-ova (engl. Performance Information Data) i Android sistema i njegovih komponenti.
- Koncept rešenja – predstavlja opis analize problema i kratak opis rešenja, način rada, opis problema pri realizaciji rešenja, kao i zahteve koje rešenje treba da ispuni
- Programsko rešenje – predstavlja detaljan opis celog sistema i svih modula sistema pojedinačno
- Rezultati i verifikacija – pregled rezultata ispitivanja, verifikacija funkcionalnosti kao i verifikacija ubrzanja koje je postignuto novim rešenjem
- Zaključak – osvrt na urađeno i mogućnosti za poboljšanja
- Literatura – spisak literature koja je korišćena

2. Teorijske osnove

U ovom poglavlju su izložene osnove Android-a, Android Studija, OBD II protokola i sistema koji se koriste u automobilskoj industriji, kao i alata koji se koriste za simuliranje rada automobila.

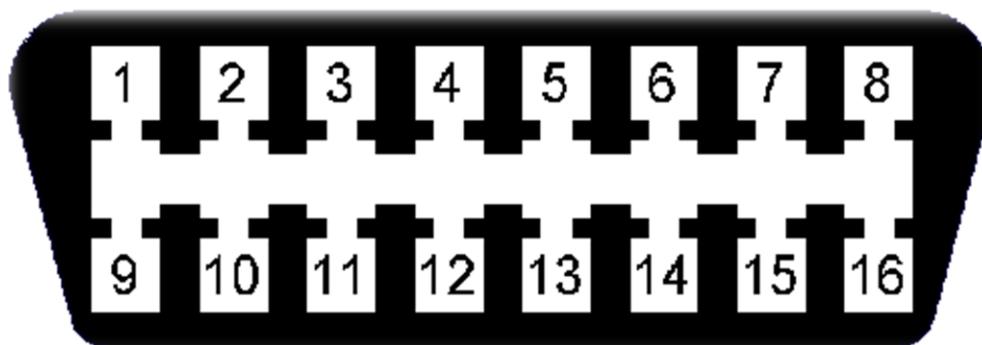
2.1 OBD II

OBD sistem se nalazi u većini automobila i manjih kamiona današnjice.

Američki kongres je 1970 izglasao zakon o čistijem vazduhu i osnovao agenciju za zaštitu životne sredine. Da bi ostali u okviru ovih standarda proizvođači automobila su se okrenuli sistemima u automobilu koje kontroliše elektronika. Tako su uspevali kontrolisati funkcije motora i dijagnostikovati njegove probleme. Senzori su merili performanse motora i prilagođavali sistem da zagađenje bude minimalno.

Na početku je bilo nekoliko proizvođača i svaki je imao svoj sistem i signale. Godine 1988 SAE (engl. Society of Automotive Engineers) je postavilo standard kakav se konektor koristi i skup dijagnostičkih ispitnih (engl. test) signala. OBD II je proširenje ovog početnog standarda SAE. OBD II omogućava očitavanje parametara motora, delova šasije, tela automobila itd. Korišćenje OBD II sistema u svim modelima vozila u SAD je moralo početi do 1. Januara 1996.

Većina automobila koji su proizvedeni od 1996. godine pa nadalje imaju OBD II sistem. Po standardu OBD II konektor, koji služi sa komunikaciju sa OBD II sistemom, mora da se nalazi najviše metar od vozača i izgleda kao Slika 1.



Slika 1 OBD II konektor

Za OBD II sistem postoji 5 mogućih protokola, najveća razlika među protokolima je u brzini komunikacije. Koji protokol se koristi može se utvrditi po priključcima na konektoru:

J1850 VPW – konektor mora imati metalne kontakte u priključcima 2, 4, 16, ali ne 10

ISO 9141-2 i KWP2000 – konektor mora imati metalne kontakte u priključcima 4, 5, 7, 15 i 16

J1850 PWM – konektor mora imati metalne kontakte u priključcima 2, 4, 5, 10, i 16

ISO 15765 CAN (u daljem tekstu CAN) – konektor mora imati metalne kontakte u priključcima 4, 5, 6, 14 i 16, on je najnoviji, od 2008 u SAD se samo on koristi

2.1.1 OBD II poruke

Komunikacija sa automobilom se obavlja preko OBD II poruka, tako što se na OBD II konektor u automobilu poveže odgovarajući alat npr. bluetooth uređaj. Preko alata šalje se poruke na OBD II sistem. Uređaj u sistemu prepoznaje poruku i vrati odgovor.

Kada se žele trenutne vrednosti parametara automobila ili vrednosti kada se desila neka greška šalje se poruka koja se sastoji iz 2 dela.

Prvi deo je željeni režim rada Tabela 1 (trenutni ili u trenutku greške). Drugi deo će biti PID za tačno željenu vrednost. Kada se želi trenutna brzina šalje se režim rada 01, a PID za brzinu je 0D i na kraju svake poruke ide “\r” tj novi red. Tako da, ako se želi dobiti trenutna brzina šalje se poruka “010D\r”. Istim sistemom kada se želi dobiti brzina u trenutku greške šalje se poruka “020D\r”. Za druge parametre kao što su broj obrtaja samo umesto OD se šalje OC ili za temperaturu se umesto OD šalje 05.

Režimi rada (03, 04, 07, 08) Tabela 1 se šalju bez dodatnih PID-ova. Npr. kada se žele kodovi greške poruka je “03\r” i odgovor će biti svi mogući kodovi greške bez obzira da li su vezani za brzinu, motor, transmisiju ili drugo [1].

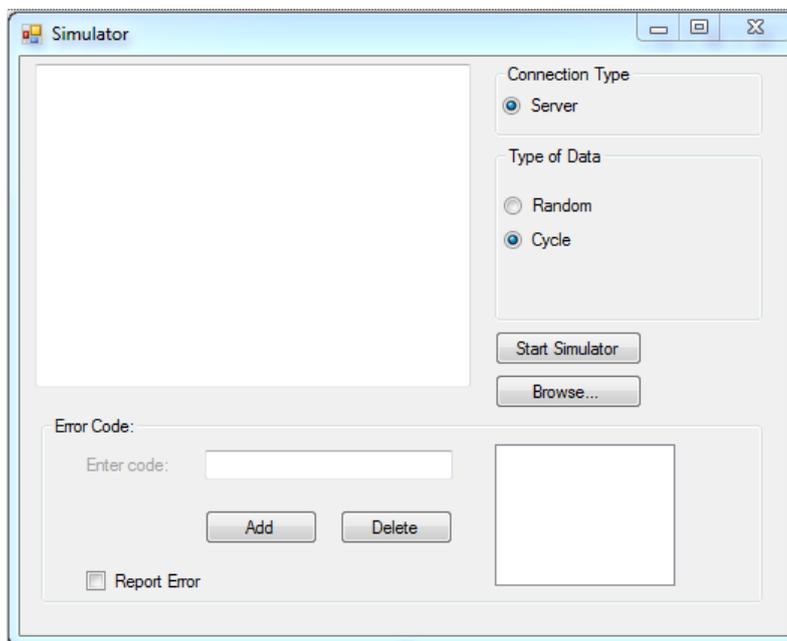
Režimi rada	Opis režima rada
01	Trenutni podaci

02	Dobavlja podatke nastale baš u trenutku kada je ustanovljena greška u sistemu
03	Kodovi greške
04	Uklanjanje svih kodova greške
05	Nadgleda senzor za kiseonik(podržavaju svi osim CAN protokol)
06	Nadgleda senzor za kiseonik(podržava samo CAN protokol)
07	Upozorenje na greške koje se mogu dogoditi
08	Kontrolne operacije nad OBD II sistemom

Tabela 1 Opis režima rada OBD II sistema

2.2 OBD II simulator

OBD II simulator simulira rad automobila (motora i ostalih komponenti). Da bi simulator verodostojnije predstavio rad automobila postoje različiti načini generisanja podataka. Npr. može se izabrati da se podaci za sve parametre nasumično generišu. Takođe postoji i opcija da se u grafičkom okruženju zada željena vrednost određenog parametra. Najbolja opcija je svakako to što simulator ima scenarije tj. generiše realne parametre koji su dobavljeni tokom vožnje. OBD II simulator predstavlja Slika 2.



Slika 2 OBD II simulator

2.3 Android operativni sistem

Android operativni sistem je proizvela kompanija „Android“ 2003. godine koju je 2005. godine kupila kompanija Google. Nakon njihovog preuzimanja operativni sistem je od 2007. ugrađivan u pametne mobilne telefone osjetljive na dodir. Posle sve većeg širenja na tržištu mobilnih telefona, kompanija Google je odlučila da Android učini dostupnim i na drugim platformama. To su televizori, tableti i slični uređaji potrošačke elektronike te je Android postao vodeći na ovom tržištu. Android je zasnovan na Linux jezgri i programiran je u C,C++ i Java programskim jezicima. Dodatna pristupačnost je u tome što je Android sistem otvorenog koda. To znači da je dopuštena slobodna izmena i distribucija softvera od strane proizvođača uređaja ili pojedinaca. Takođe i alati koje ova kompanija nudi su besplatni i ne predstavljaju dodatne troškove inženjerima koji razvijaju softver za ovaj operativni sistem. Stoga je veoma popularan i zastupljen. Aplikacije se mogu instalirati sa Google Play prodavnice i većina aplikacija je besplatna. Trenutan broj aplikacija na ovom servisu je oko milion i po. Slika 3 predstavlja sve verzije Android operativnog sistema od nastanka pa do danas [2].



Slika 3 Sve verzije Android operativnog sistema

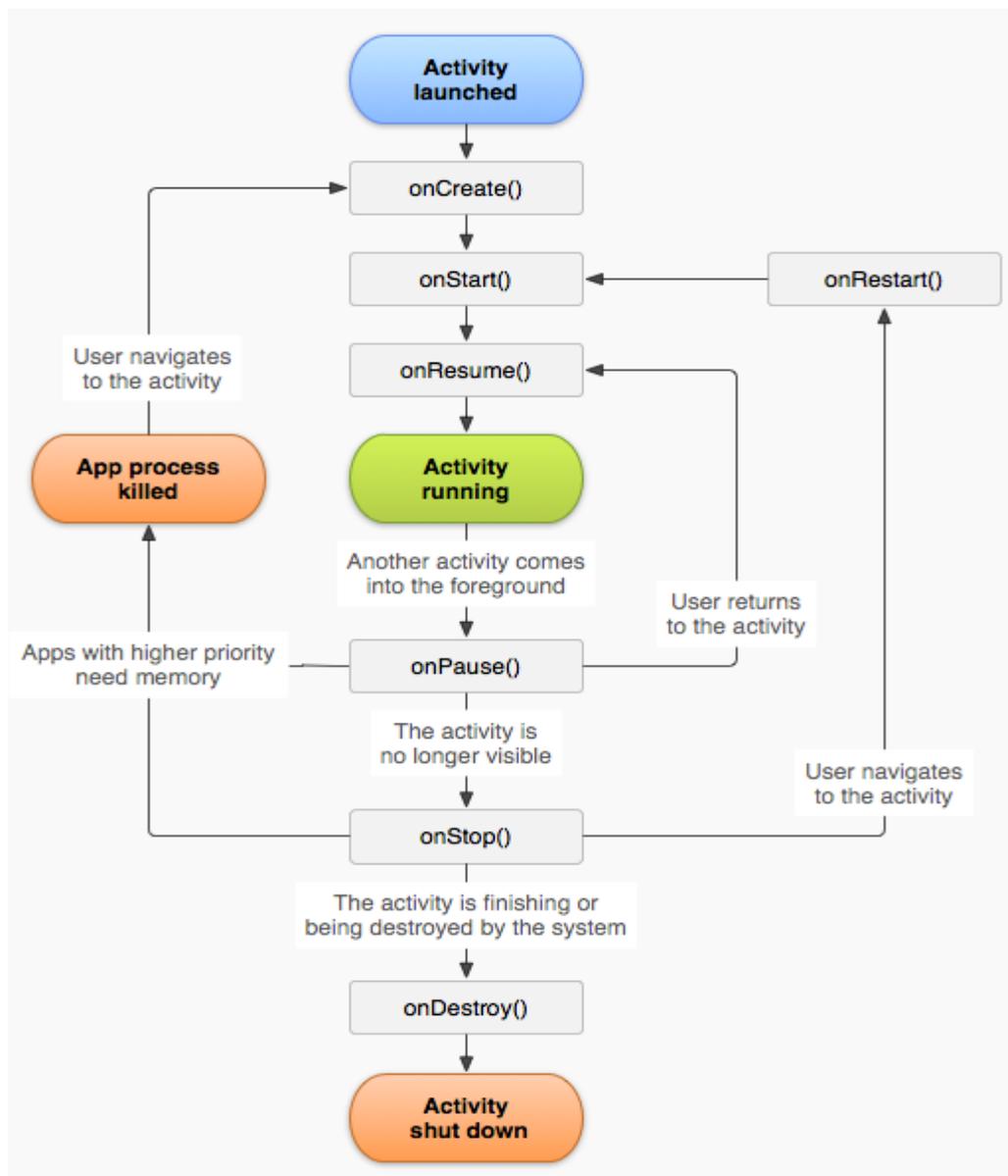
2.3.1 Aktivnosti u Androidu

Aktivnost je komponenta aplikacije koja dobija ekran i sa kojim korisnik može da vrši interakciju. To je npr.pozivanje, slikanje, slanje elektronske pošte ili slično. Svaka aktivnost dobija svoj poseban ekran u kojem može da se iscrtava grafičko okruženje.

Jedna aplikacija se sastoji od više aktivnosti koje su povezane. Jedna aktivnost je uvek “glavna” i ona se prva pojavi kada se aplikacija prvi put pokrene. Svaka aktivnost može

pokrenuti novu aktivnost da bi neke nove radnje bile odrađene. Svaki put kada se pokrene nova aktivnost, stanje prethodne sistem sačuva na steku aktivnosti(engl. back stack).

Kada se nova aktivnost pokrene prethodna prelazi u stanje pauze, promena stanja aktivnosti je registrovana kroz povratni poziv (engl. callback) životnog ciklusa. Ima nekoliko povratnih poziva koje aktivnost može da primi u odnosu na promenu stanja. Svaka aktivnost ima stanja kroz koja prolazi [3]. Ta stanja i čitav životni ciklus jedne aktivnosti predstavlja Slika 4.



Slika 4 Životni vek aktivnosti

2.3.2 Servisi u Androidu

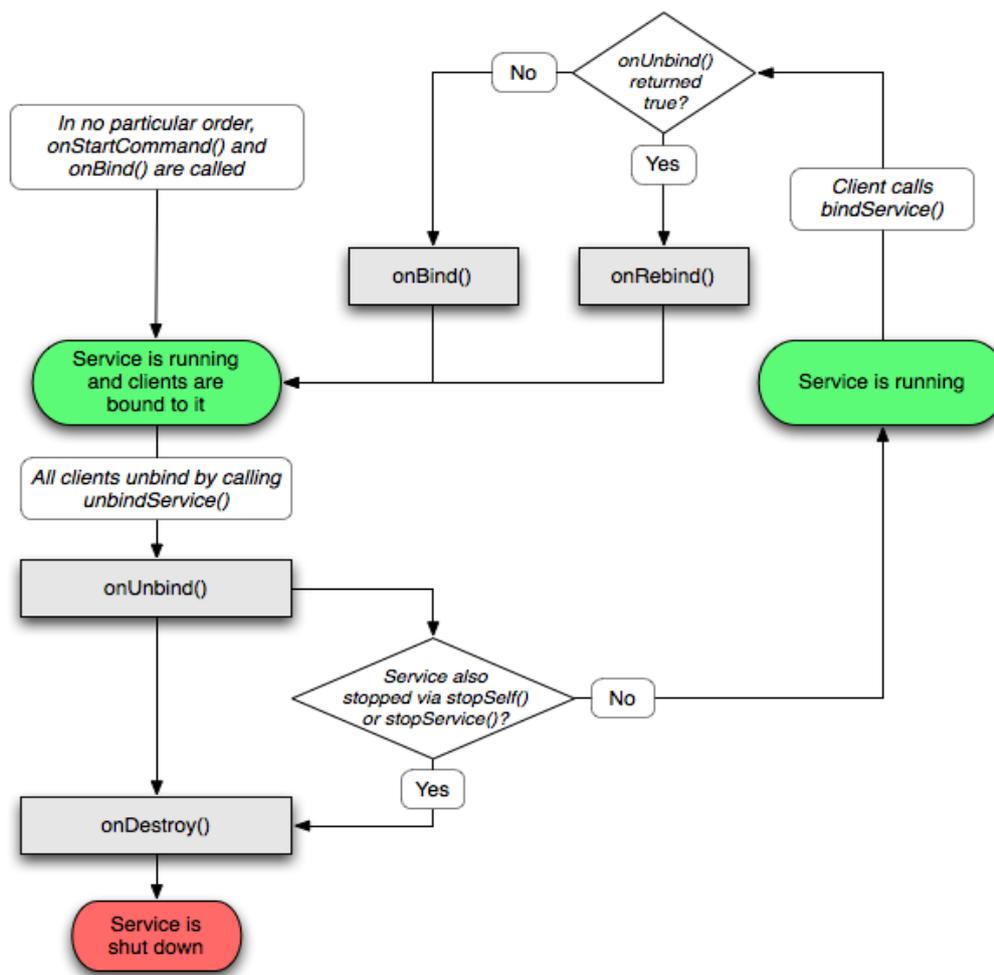
Servis je aplikaciona komponenta koja je prisutna u pozadini i može da živi bez aplikacije. Servis opslužuje više klijenata istovremeno istim podacima. Servisi nemaju grafičko korisničko

okruženje. Aplikacija može da pokrene servis i on nastavlja da postoji u pozadini iako se aplikacija zaustavi. Postoje dve vrste servisa: lokalni (engl. local) i udaljeni (engl. remote).

Lokalni servisi su komponente aplikacije, postoje u istom procesu kao i aplikacija. Uglavnom lokalni servisi odrade jednu operaciju koja ne treba da vrati rezultat. To je npr. preuzimanje ili otpremanje datoteke preko interneta. Kada se zadata operacija završi servis se može zaustaviti, programski pozivanjem funkcije (stopSelf) ili aplikacijski i u tom slučaju se poziva (stopService).

Udaljeni servisi su složeniji - prvo se treba povezati na udaljeni servis sa aplikacijom ili drugim servisom, a tek zatim se može vršiti komunikacija. Komunikacija može biti u vidu slanja zahteva, dobijanja rezultata i među-procesne komunikacije. Među-procesna komunikacija se vrši preko sprega (engl. interface) koje su u Android-u predstavljene AIDL (engl. Android Interface Definition Language) datotekama. Udaljeni servis je pokrenut sve dok ima neko ko je na njega povezan, kada nema nikoga povezanog, on se uništi.

Način pokretanja Android servisa, povezivanje klijentskih aplikacija i servisa kao i kompletan životni ciklus Android servisa predstavlja Slika 5.



Slika 5 Životni vek servisa

Kada se lokalni servis pokrene tada se na njega mogu povezati udaljeni klijenti. On će raditi sve dok svi klijenti ne prekinu vezu i dok se lokalni servis ne zaustavi.

Kada servis nije pokrenut, a na njega se povežu klijenti (aplikacije ili drugi servisi) on će se pokrenuti i klijenti će biti povezani na njega. Sve dok svi klijenti ne prekinu vezu sa servisom on neće biti zaustavljen [4].

2.4 Android Studio

Android Studio predstavlja razvojno okruženje namenjeno za pisanje Android aplikacija. Google je koristeći IntelliJ platformu razvio Android Studio i objavio 2013. godine. On je zamenio Eclipse koji se do njegove pojave koristio za programiranje Android aplikacija. Dostupan je besplatno. Može se koristiti na svim operativnim sistemima. Nalazi pod Apache 2.0 licencom.

3. Koncept rešenja

U ovom poglavlju data je analiza problema i kratak opis rešenja, opisan je način rada sistema, izneti problemi pri realizaciji rešenja, kao i zahtevi koje novo rešenje treba da ispunjava.

3.1 Analiza problema i opis rešenja

Pošto se većina danas dostupnih uređaja potrošačke elektronike kao i računara zasniva na Linux operativnom sistemu, ovaj projekat predstavlja deo većeg projekta u kojem se pokušava napraviti računar zasnovan na Android-u.

Glavna namena računara je konstantno informisanje vozača o svim dešavanjima u automobilu. Stalna dostupnost informacija i brzo osvežavanje grafičkog korisničkog okruženja, realizovano je korišćenjem Android servisa. Android servis je uzet kao najbolje rešenje za dati problem zato što može da više klijenata opslužuje istim podacima. Korisnik na svom računaru želi informacije o brzini vozila, koliko kilometara mu je ostalo do sledećeg servisa, egzaktno nivo goriva, koliko još kilometara može preći sa trenutnim nivoom goriva i slično.

Sve ove informacije dobavlja postojeće rešenje servisa (CarInfoService) koje ima mnoštvo nedostataka npr. u brzini dobavljanja parametara, nemogućnosti rada sa više klijenata itd. Svi ovi nedostaci u postojećem rešenju servisa će u novom rešenju biti ispravljani.

Jedna od najbitnijih odlika novog rešenja servisa jeste mogućnost rada sa više klijenata zbog toga što u vozilu postoji više aplikacija koje preuzimaju podatke istovremeno. Servis treba sa pokretanjem Android sistema (paljenjem vozila) da se pokrene i uvek radi u pozadini tako da se podaci mogu preuzimati. Da bi se omogućila komunikacija između klijenta i servisa mora da postoji određen API (engl. Application Programming Interface) koji klijentu dozvoljava da preuzima informacije. Brzina dobavljanja podataka mora biti maksimalna moguća kako bi klijent

na vreme bio obavešten o tome šta se dešava u automobilu. Svakom klijentu se daje mogućnost izbora parametara koje želi da preuzme i koliko često želi zahtevati željeni parametar tj. u kom intervalu poziva funkciju za dobavljanje parametra. Npr. parametar kao što je brzina koji se češće menja se treba dobavljati češće nego nivo goriva.



Slika 6 Delovi računara u vozilu

Slika 6 predstavlja Android servis koji je deo računara, opslužuje 2 različite aplikacije na računaru, a radi i sa samim korisničkim okruženjem računara.

3.2 Način rada sistema

Na početku treba uspostaviti vezu između servisa i klijenta koja će biti ostvarena preko sprege u kojoj će se nalaziti sve funkcije API-ja. Sprega će biti definisan u Android biblioteci koja je napravljena i uključena u servis.

Pre svakog početka razmene podataka treba odraditi slanje komandi na ELM mikrontroler (koji služi kao sprega između OBD II sistema i servisa) [5]. Te komande će odraditi inicijalizaciju mikrontrolera i nakon toga možemo vršiti razmenu podataka.

Prilikom razmene podataka svaka od funkcija API-ja mora da šalje drugačiji PID na simulator ili kontrolnu jedinicu automobila da bi dobavila tačno tražene informacije. Kada se dobiju informacije one su u vidu znakovnog niza i zatim za svaki od parametara postoji različit način pretvaranja da bi dobili realnu vrednost parametra.

Da bi brzina razmene podataka bila maksimalna, realizovan je i način rada gde se umesto slanja samo jednog parametra po poruci, može slati zahtev za više parametara u jednoj poruci. Na taj način se brzina osvežavanja parametara kod klijenta povećava. Međutim ovaj način nije dostupan kod svih automobila i zato je ostavljena opcija u servisu da se bira način slanja zahteva.

3.3 Problemi pri projektovanju

Jedan od najvećih problema je bio to što OBD II simulator ili sistem u automobilu ne može da obradi i vrati dva zahteva istovremeno npr. ako se traže 2 parametra odjednom, svaki parametar u svom odgovoru ima ID koji mu se automatski dodeli i nije bitno koji se parametar prvi traži već parametar sa najvećim važnošću (ID-om) će biti prvi poslat. Kada bi se mnogo parametara tražilo u isto vreme došlo bi do zagušenja CAN magistrale, koja se koristi u novijim automobilima za komunikaciju, dok bi odgovor za parametre sa manjom važnošću stigao sa velikim zakašnjenjem, a u nekim slučajevima ne bi ni stigao.

Da bi se ovakav scenario izbegao, problem je rešen na taj način što kada se primi zahtev za parametrom od klijenta, ne šalje se odmah na obradu već se ubaci u red čekanja (engl. queue). Iz tog reda se stalno uzima jedan zahtev on se obradi i vrati se kroz povratni poziv (engl. callback) vrednost traženog parametra klijentu. Tek kada se sa tim završi može se uzeti sledeći zahtev iz reda, tako da očitavanje vrednosti parametara će ići redosledom kojim ih klijent zahteva.

Još jedan veliki problem u realizaciji je predstavljao odgovor koji simulator vrati kada mu se pošalje više zahteva odjednom (npr. kroz jednu poruku od simulatora se traži i brzina i broj obrtaja motora). Ovaj odgovor se dobija u vidu znakovnog niza iz kojega treba izvući vrednosti na određenim pozicijama. Zatim ih treba pretvarati tako da se dobiju realne (prave) vrednosti parametara. Npr. kada se pošalje 010D0C to je zahtev za brzinu i broj obrtaja i odgovor je 410D600C090A. Prvo se ovaj odgovor pretvori u niz znakova(engl. char array) i onda se pomera sve dok se ne dođe do PID-a koji je poznat, u ovom primeru to je 0D. Uzimaju se dva znaka posle njega i vrši se pretvaranje da bi se dobila realna brzina, u ovom primeru 96 km/h. Isti je način i za broj obrtaja s tim što se za njega uzimaju 4 znaka posle 0C.

3.4 Zahtevi koje rešenje treba da ispunjava

Da bi se rešenje zadatka smatralo uspešnim postoje određeni zahtevi koji moraju biti ispoštovani. Zahtevi uključuju potrebu da zauzeće radne memorije bude minimalno i kod modularan.

3.4.1 Zahtevi od strane radne memorije

Prilikom realizacije ustanovljeno je da se zauzeće radne memorije povećava sa povećanjem reda čekanja koji stalno prima nove zahteve za obradu. Nakon toga se metodom ispitivanja došlo do veličine reda od 30000 elemenata. Sa ovim će zauzeće u radnoj memoriji biti maksimalno 20 MB. Da bi tako i ostalo kada je red pun i ne može da primi više elemenata, funkcija koja upisuje zahtev u red vraća klijentu povratnu vrednost -1 (kod greške). Na osnovu toga klijent zna da mu taj zahtev neće biti obrađen, zato što nije upisan u red čekanja.

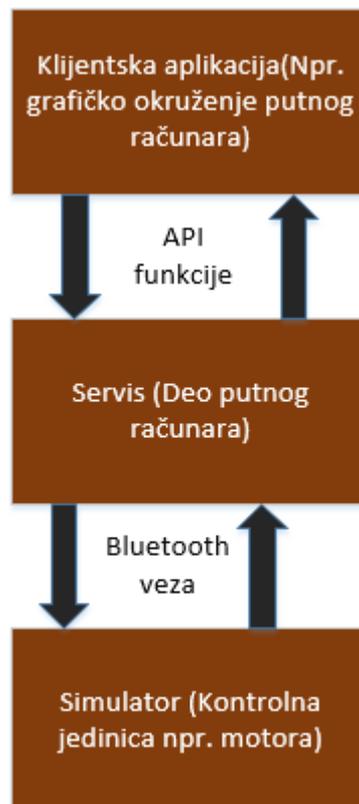
3.4.2 Modularnost

Kako bi servis bio održiv u smislu preglednosti rešenja čime se omogućuje laka nadogradnja i izmena postojećih funkcionalnosti potrebno je izvršiti modularizaciju na što funkcionalniji način. Ukoliko bi želeli dodati funkciju za novi parametar to se može uraditi vrlo jednostavno. To se radi samo dodavanjem novog koda sličnog postojećem u module za upis zahteva u red i modul za čitanje zahteva iz reda.

4. Programsko rešenje

U ovom poglavlju je dato objašnjenje i prikaz modularne strukture rešenja. Rešenje je realizovano u programskom jeziku Java na Android platformi [6], [7].

Način na koji se vrši komunikacija između svih delova računara i kontrolnih jedinica (npr. motora, transmisije, sigurnosne brave i drugih) prikazuje Slika 7.

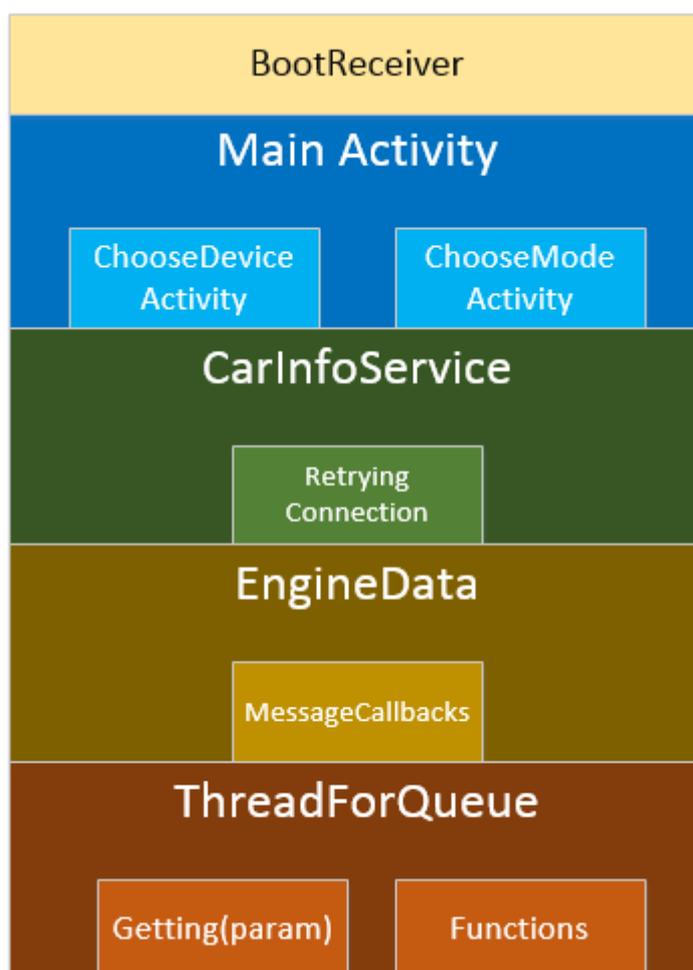


Slika 7 Komunikacija novog rešenja sistema

Slika 7 predstavlja komunikaciju u računaru zasnovanom na Android-u. Unutar računara se nalazi Android servis koji služi za komunikaciju između korisničkog okruženja računara i kontrolnih jedinica automobila. U daljem tekstu će svi moduli Android servisa biti objašnjeni ponaosob kao i klijent tj. aplikacija koja se koristi za testiranje ovog rešenja servisa.

4.1 Moduli servisa

Dijagram arhitekture Android servisa koji se nalazi u računara predstavlja Slika 8



Slika 8 Arhitektura servisa

Slika 8 predstavlja arhitekturu tj. sve module servisa i svaki će biti posebno objašnjen u daljem tekstu.

4.1.1 Modul za pokretanje servisa prilikom pokretanja operativnog sistema

Ovaj modul nasleđuje Android-ovu klasu za prijem neusmerenih poruka (engl. Broadcast Receiver) i kada se sistem pokrene desi se akcija koja pokrene servis. Na taj način svaki put kada se upali automobil, pali se sistem, a sa njim i servis.

4.1.2 Modul aktivnosti

Ovaj modul predstavlja Android aktivnost u kojoj se Android servis može pokrenuti. U slučaju da iz nekog razloga dođe do uništenja servisa u ovom modulu je omogućeno ponovno pokretanje servisa. Takođe iz ovog modula se prave 2 nove aktivnosti (modula). U prvoj se izlistavaju upareni bluetooth uređaji i vrši se izbor uređaja sa kojim se žele razmenjivati podaci. U drugoj aktivnosti se bira način rada tj. da li će kontrolnoj jedinici automobila biti slato više zahteva za parametrima u jednoj poruci ili jedan po poruci.

4.1.3 Glavni modul servisa

To je modul u kome je definisano šta servis radi kada se pokrene. Pri pokretanju servisa odradi se potrebna inicijalizacija, posle toga se klijent može povezati i preuzimati podatke. Takođe je definisano kako treba da se ponaša servis kad se na njega poveže jedan ili više klijenata. U tom slučaju ako nije izvršena inicijalizacija koja treba uvek da se izvrši pre razmene podataka, ona se izvrši i tek se onda mogu razmenjivati podaci. Realizovano je i kako se servis ponaša kada više nije povezan nijedan klijent i kada se servis opet pokrene. U ovom servisu nema stopiranja već se nakon oslobađanja zauzetih resursa, servis sam nakon 10 sekundi opet pokreće.

4.1.4 Modul za ponovno pokretanje servisa

Kada se proba odradi pokretanje servisa i iz nekog razloga ne uspe, nakon 10 sekundi će biti pozvan ovaj modul u kojem se opet proba izvršiti pokretanje servisa. Ovo će se iznova ponavljati sve dok se ne pokrene servis.

4.1.5 Modul za inicijalizaciju konekcije

Ovde se vrši inicijalizacija koja je potrebna pre svake razmene podataka između servisa i simulatora ili automobila. Šalju se komande ELM mikrokontroleru (koji služi kao sprega između OBD II sistema i servisa), kao što je resetovanje. Takođe i komanda koja radi uklanjanje zaglavlja sa odgovora da bi odgovor bilo lakše razumeti. Šalju se i komande za postavljanje prolaza (engl. port) na kojem se vrši komunikacija i vremensko odlaganje(engl. timeout) koje predstavlja vreme čekanja na odgovor od simulatora ili kontrolne jedinice automobila. Na kraju se šalje upit da li je dobro uspostavljena veza.

Ako dođe do greške prilikom inicijalizacije (npr. ne može se otvoriti bluetooth utičnica), nakon deset sekundi se opet pokuša odradi inicijalizacija konekcije i tako će iznova pokušavati sve dok inicijalizacija konekcija ne bude uspešna.

4.1.6 Modul za realizaciju funkcija API-ja

Kada klijent pozove funkciju zahtevajući parametar, u ovom modulu se svaki zahtev za parametrom ubaci u red čekanja [8]. Kada je red već popunjen funkcija će vraćati vrednost -1 (kod greške), kako bi klijent znao da ne očekuje odgovor na taj zahtev. Međutim, ako je povratna vrednost 0, onda je zahtev ubačen u red i očekuje se odgovor na njega preko povratnog poziva.

Tabela 2 predstavlja spisak funkcija API-ja gde je svaka funkcija ponaosob objašnjena.

<p>int requestSpeed(ICarInfoSpeedNotification speedNotification)</p> <p>Funkcija koja za parametar prihvata spregu pomoću koje će biti upućen povratni poziv (engl. callback). Brzina iz ove funkcije klijentu će biti prosleđena kroz povratni poziv. Kada se klijentu prosledi vrednost -1 znači da je došlo do greške u komunikaciji, a ako je -2 znači da OBD II sistem u automobilu ne podržava dati parametar (tj. vraća da nema traženog podatka). Kada je povratna vrednost funkcije 0 može se očekivati povratni poziv za brzinu.</p>
<p>int requestThrottlePosition(ICarInfoThrottleNotification throttleNotification)</p> <p>Funkcija koja za parametar prihvata spregu pomoću koje će biti upućen povratni poziv. Pozicija papučice iz ove funkcije klijentu će biti prosleđena kroz povratni poziv. Kada se klijentu prosledi vrednost -1 znači da je došlo do greške u komunikaciji, a ako je -2 znači da OBD II sistem u automobilu ne podržava dati parametar (tj. vraća da nema traženog podatka). Kada je povratna vrednost funkcije 0 može se očekivati povratni poziv za poziciju papučice.</p>
<p>int requestEngineRpm(ICarInfoEngineRpmNotification engineRpmNotification)</p> <p>Funkcija koja za parametar prihvata spregu pomoću koje će biti upućen povratni poziv. Broj obrtaja iz ove funkcije klijentu će biti prosleđen kroz povratni poziv. Kada se klijentu prosledi vrednost -1 znači da je došlo do greške u komunikaciji, a ako je -2 znači da OBD II sistem u automobilu ne podržava dati parametar (tj. vraća da nema traženog podatka). Kada je povratna vrednost funkcije 0 može se očekivati povratni poziv za broj obrtaja.</p>
<p>int requestFuelLevel(ICarInfoFuelLevelNotification fuelLevelNotification)</p> <p>Funkcija koja za parametar prihvata spregu pomoću koje će biti upućen povratni poziv. Nivo goriva iz ove funkcije klijentu će biti prosleđen kroz povratni poziv. Kada se klijentu prosledi vrednost -1 znači da je došlo do greške u komunikaciji, a ako je -2 znači da OBD II sistem u</p>

<p>automobilu ne podržava dati parametar (tj. vraća da nema traženog podatka). Kada je povratna vrednost funkcije 0 može se očekivati povratni poziv za nivo goriva.</p>
<pre>int requestTroubleCodes (ICarInfoTroubleCodesNotification troubleCodesNotification)</pre> <p>Funkcija koja za parametar prihvata spregu pomoću koje će biti upućen povratni poziv. Kodovi greške iz ove funkcije klijentu će biti prosleđena kroz povratni poziv. Kada se klijentu prosledi vrednost -1 znači da je došlo do greške u komunikaciji, a ako je -2 znači da u automobilu nema grešaka. Kada je povratna vrednost funkcije 0 može se očekivati povratni poziv za kodove greške.</p>
<pre>int requestEngineTemperature (IcarInfoEngineTemperatureNotification engineTemperatureNotification)</pre> <p>Funkcija koja za parametar prihvata spregu pomoću koje će biti upućen povratni poziv. Temperatura motora iz ove funkcije klijentu će biti prosleđena kroz povratni poziv. Kada se klijentu prosledi vrednost -1 znači da je došlo do greške u komunikaciji, a ako je -2 znači da OBD II sistem u automobilu ne podržava dati parametar (tj. vraća da nema traženog podatka). Kada je povratna vrednost funkcije 0 može se očekivati povratni poziv za temperaturu motora.</p>
<pre>int requestFuelPressure (ICarInfoFuelPressureNotification fuelPressureNotification)</pre> <p>Funkcija koja za parametar prihvata spregu pomoću koje će biti upućen povratni poziv. Pritisak goriva iz ove funkcije klijentu će biti prosleđena kroz povratni poziv. Kada se klijentu prosledi vrednost -1 znači da je došlo do greške u komunikaciji, a ako je -2 znači da OBD II sistem u automobilu ne podržava dati parametar (tj. vraća da nema traženog podatka). Kada je povratna vrednost funkcije 0 može se očekivati povratni poziv za pritisak goriva.</p>
<pre>int requestEngineAirTemperature (IcarInfoEngineAir TemperatureNotification engineAirTemperatureNotification)</pre> <p>Funkcija koja za parametar prihvata spregu pomoću koje će biti upućen povratni poziv. Temperatura vazduha koja ulazi u motor iz ove funkcije klijentu će biti prosleđena kroz povratni poziv. Kada se klijentu prosledi vrednost -1 znači da je došlo do greške u komunikaciji, a ako je -2 znači da OBD II sistem u automobilu ne podržava dati parametar Kada je povratna vrednost funkcije 0 može se očekivati povratni poziv za temperaturu vazduha koja ulazi u motor.</p>

Tabela 2 API funkcije

4.1.7 Modul za obradu zahteva

Modul za obradu zahteva iz reda čekanja se pokrene kada se prvi put prvi klijent poveže na servis. On stalno radi sve dok se servis ne uništi. U odnosu na to kakav način rada je odabran u parametrima servisa ovaj modul radi drugačije.

Kada je izabran rad sa slanjem više zahteva u jednoj poruci ovde se u odnosu na popunjenost reda pravi znakovni niz koji će biti poslat simulatoru ili automobilu. Taj znakovni niz ne može da ima više od šest zahteva u sebi. Tako da, ako je red veći od šest uzima se prvih šest zahteva, a ostali će biti uzeti u sledećem prolazu. Ovako mogu da se obrađuju svi zahtevi osim kodova greške koje moraju posebno da se uzmu iz reda i obrade zato što dobavljaju različitim režimom rada (opis Tabela 1).

Kada je izabran rad sa slanjem jednog zahteva po poruci, ukloni se i obradi zahtev sa početka reda. Zatim se u zavisnosti koji je parametar u pitanju poziva funkcija koja pošalje tačno određen PID simulatoru, primi odgovor i pretvori vrednost parametra u čitljiv oblik (celobrojnu vrednost). Na kraju povratnim pozivom obavesti klijenta o vrednosti.

4.1.8 Modul sa funkcijama za komunikaciju sa automobilom

Tabela 3 predstavlja funkcije koje služe za komunikaciju servisa i OBD II sistema u automobilu.

```
String ReadAndWrite(InputStream m_input, OutputStream m_output,
String PID)
```

Funkcija koja radi slanje i prijem poruke (niz bajta) sa simulatorom ili kontrolnom jedinicom automobila.

```
int SpeedConversion(String hexSpeed)
```

Funkcija koja radi pretvaranje brzine tako što iz znakovnog niza koji stigne sa simulatora uzme zadnja dva znaka. Zatim ih treba pretvori iz heksadecimalne vrednosti u decimalnu i proslediti klijentu. Kao što za ovaj parameter postoji pretvaranje, tako mora postojati za svaki pošto se većina pretvaranja razlikuju.

```
String multipleCommandsCreatingString(String RES,
BlockingQueue<Object> m_queue)
```

U ovoj funkciji se pravi znakovni niz tako što se uzima iz reda zahtev jedan po jedan i dodaje se na znakovni niz. Ako već postoji u redu, zahtev neće biti dodat. Maksimalan broj zahteva u jednom znakovnom nizu je 6. Povratna vrednost je znakovni niz sa više zahteva koji će biti poslat simulatoru ili automobilu.

```
void multiCommandsConversion(String response)
```

U ovoj funkciji se uzme odgovor od simulatora ili automobila na više zahteva. Prolazi se kroz ceo znakovni niz (odgovor) i kada se naide na odgovor za neki parametar pretvori se njegova vrednost u čitljiv oblik i preko povratnog poziva pošalje se vrednost klijentu.

```
void callbackErrorReturn()
```

Ova funkcija biva pozivana, ako simulator ili automobil nisu ništa odgovorili, što znači da je došlo do neke greške. Tada će preko povratnog poziva klijentu svi parametri koje je zahtevao dobijati vrednost -1. Tako klijent zna da je došlo do greške.

Tabela 3 Funkcije za komunikaciju servisa i automobila

4.1.9 Moduli za obradu zahteva prilikom slanja jednog parametra

Svaki parametar ima poseban modul u kome se njegov PID pretvori iz znakovnog niza u niz bajtova i pošalje se simulatoru. Zatim se primi odgovor i odradi pretvaranje baš za njega. Preko povratnog poziva se vrati vrednost parametra klijentu.

4.2 Modul za servis-klijent komunikaciju

Ovaj modul predstavlja Android biblioteku koja je zasebno kreirana i uključena u postojeći servis. U biblioteci se nalazi sprega u kojoj su deklarirane funkcije koje će predstavljati zahteve za različitim parametrima. Ovo je tako realizovano jer se uz pomoć sprege može vršiti međuprocena komunikacija. Takođe za svaki zahtev postoji posebna sprega sa funkcijom povratnog poziva tako da čim se dobije vrednost, obavesti se klijent.

4.3 Modul klijenta

U klijentskoj aplikaciji se prvo odradi povezivanje sa servisom. Kada se klijent uspe povezati na servis sledi pozivanje funkcija iz API-ja koje dobavljaju tražene parametre. Kada servis dobije parametre od simulatora, parametri se kroz povratni poziv prosleđuju klijentu. Klijent zatim osvežava korisničko okruženje sa novim podacima. U odnosu na učestanost slanja zahteva za parametrima servis će sporije ili brže odgovarati klijentskim aplikacijama. U slučaju slanja zahteva velikom učestanošću red čekanja u servisu će biti zakrčen zahtevima jednog klijenta. U tom slučaju bi drugi klijenti dugo čekali da njihovi zahtevi dođu na red i budu obrađeni i odgovori bi sporije stizali.

5. Verifikacija i rezultati

Da bi se potvrdila stabilnost i ispravnost aplikacije ona mora biti podvrgnuta različitim vrstama provera. Nad rešenjem Android servisa (CarService) izvršena su ispitivanja koja proveravaju njegovu funkcionalnost kao i ispitivanja koja mere performanse samog servisa. To je odrađeno uz pomoć Android aplikacije za proveru servisa (TestingService), u nastavku ovog poglavlja klijent. Da bi se servis pravilno ispitao mora se predvideti što više mogućih scenarija za proveru njegove funkcionalnosti:

- Scenario 1: Kada se iz opcija servisa, ne izabere nijedan uređaj niti se izabere način rada
- Scenario 2: Kada je simulator pokrenut i servis uspostavi bluetooth konekciju sa simulatorom, a na servis se poveže klijent (slučaj kada sve funkcioniše očekivano)
- Scenario 3: Kada sve funkcioniše očekivano (klijent dobavlja podatke), zaustavi se klijent, a zatim simulator i onda se opet pokrene klijent
- Scenario 4: Kada sve funkcioniše očekivano (klijent dobavlja podatke), pa se ugasi bluetooth
- Scenario 5: Kada se simulatoru šalju zahtevi za neki parametar koji on ne podržava. Npr. simulator nema podršku za vrednost temperature motora, onda bi trebao da vraća da nema taj podatak

Nakon proverenih svih scenarija, može se zaključiti da je servis funkcionalno ispravan.

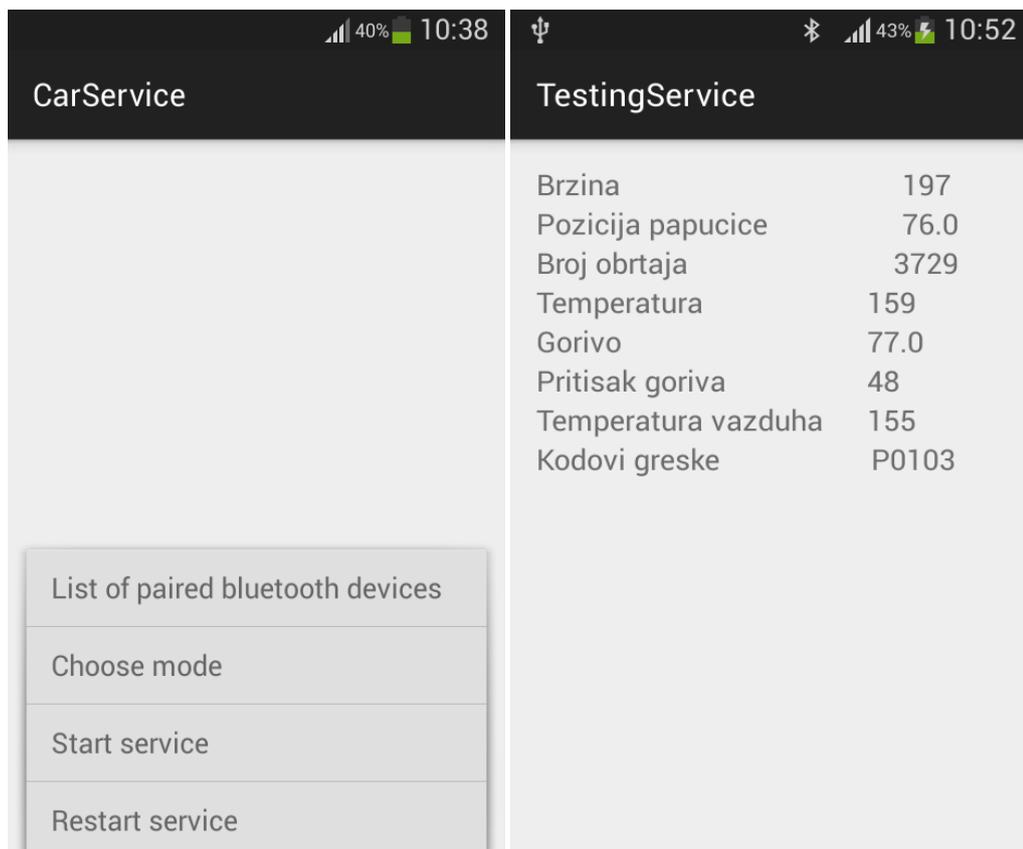
Druga provera predstavlja verifikaciju ubrzanja dobavljanja podataka sa simulatora u odnosu na postojeće rešenje servisa (koje je objašnjeno u Analizi problema). Tabela 4 predstavlja brzinu dobavljanja podataka:

Rešenja	Brzina dobavljanja jednog parametra [ms]
Postojeće rešenje servisa	200-250
Novo rešenje-Ovaj projekat(CarService)	120-150

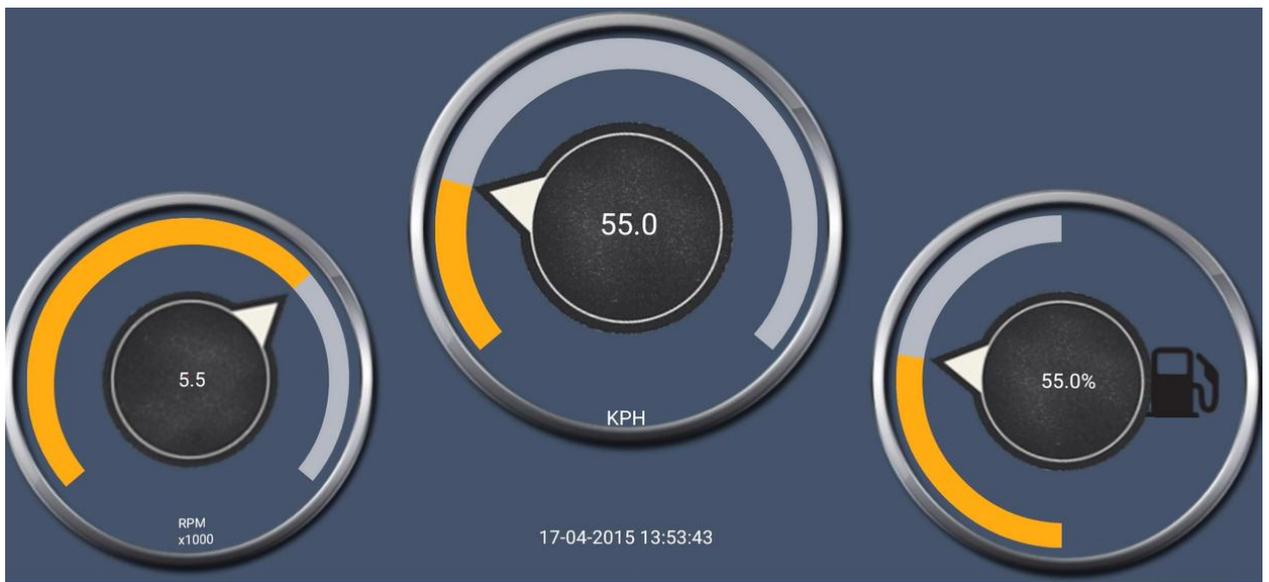
Tabela 4 Brzina dobavljanja parametara

Iz Tabela 4 se može zaključiti da je ubrzanje u nekim slučajevima i dvostruko, što predstavlja odličan napredak.

Android servis je proveren na telefonima zasnovanim na Android operativnom sistemu. Pored toga servis i aplikacija za proveru su ispitane na više različitih mobilnih uređaja (Samsung Galaxy Trend Plus, Samsung Galaxy S3) kao i na tabletu Google (HTC) Nexus 9 sa 5.0 Android Lollipop operativnim sistemom. Tokom verifikacije na svim uređajima, nije bilo nikakvih nepravilnosti. Konačan izgled aplikacije i servisa na telefonu Samsung Galaxy Trend Plus predstavlja Slika 9.



Slika 9 Konačan izgled servisa i aplikacije za proveru



Slika 10 Servis iz automobila korišćen u aplikaciji za vizualizaciju podataka

Slika 10 predstavlja aplikaciju koja je napravljena da vizualizuje podatke o automobilu koristeći realizovano novo rešenje servisa.

6. Zaključak

Zadatak je bio razvoj biblioteke za pristup podacima jedinice za kontrolu automobilu. Biblioteka je realizovana kao Android servis zajedno sa Android aplikacijom koja služi za dobavljanje podataka sa simulatora ili automobila (jedinice motora) preko servisa. Zbog reda čekanja unutar servisa, performanse rešenja su uslovljene brzinom odgovora koji stiže sa simulatora ili kontrolne jedinice automobila. Takođe dosta pažnje je posvećeno smanjenju potrošnje radne memorije sa što manjim uticajem na performanse.

Android servis i aplikacija su realizovani sa prenosivošću u vidu, tako da dodatne biblioteke nisu korišćene. Korišćene su samo funkcionalnosti koje obezbeđuje Android koje su u njemu već duže podržane.

Ovaj servis je napravljen da se koristi kao deo računara. On omogućava brzo dobavljanje i prikazivanje podataka korisniku i korišćenje podataka u drugim aplikacijama unutar računara.

Prostor za dalja poboljšanja može da ide u pravcu proširenja broja parametara koji se mogu dobavljati od simulatora ili kontrolne jedinice automobila. Dalje istraživanje CAN protokola i celog OBD II sistema može ići u pravcu daljeg poboljšanja brzine dobavljanja parametara.

7. Literatura

- [1] OBD Modes, dostupno na:
<http://www.outilsobdfacile.com/obd-mode-pid.php> učitano 12.04.2015.
- [2] Android history, dostupno na:
http://www.android.com/intl/en_us/history/ učitano 17.04.2015.
- [3] Android activity, dostupno na:
<http://developer.android.com/reference/android/app/Activity.html> učitano 17.04.2015.
- [4] Android service, dostupno na:
<http://developer.android.com/guide/components/services.html> učitano 17.04.2015.
- [5] ELM327 OBD to RS232 Interpreter, dostupno na:
<http://www.elmelectronics.com/DSheets/ELM327DSF.pdf> učitano 29.04.2015.
- [6] Professional Android Application Development, Reto Meier, May 1, 2008
- [7] Professional Android 2 Application Development, Reto Meier, Mar 1, 2010
- [8] Sistemska programska podrška u realnom vremenu 2, Miroslav Popović, Vladimir Kovačević, Univerzitet u Novom Sadu 2012