



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

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

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

ЗАВРШНИ (BACHELOR)РАД

Кандидат: Александар Ђургуз
Број индекса: РА177/2011

Тема рада: Једно решење родитељске контроле аутомобила

Ментор рада: проф.др Небојша Пјевалица

Нови Сад, јун, 2015



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:	Александар Ђургиз		
Ментор, МН:	др. Небојша Пјевалица		
Наслов рада, НР:	Једно решење родитељске контроле 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)	7/28/0/5/18/0/0		
Научна област, НО:	Електротехника и рачунарство		
Научна дисциплина, НД:	Рачунарска техника		
Предметна одредница/Кјучне речи, ПО:	сервис, аутомобил, „OBD II system”,родитељска контрола,андроид		
УДК			
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад		
Важна напомена, ВН:			
Извод, ИЗ:	У овом раду реализован је сервис за слање и пријем електронске поште у случају да возило прекорачи брзину кретања. Реализована је и корисничка апликација која омогућава промену корисничких подешавања везана за електронску адресу пошиљаоца, примаоца и ограничења брзине за одређену деоницу пута. Целокупан рад се заснива на познавању OBD II система, раду са „JavaMail API“ и коришћењу „Java“ програмског језика.		
Датум прихватања теме, ДП:			
Датум одбране, ДО:			
Чланови комисије, КО:	Председник:	др. Јелена Ковачевић	
	Члан:	др. Иван Каштелан	Потпис ментора
	Члан, ментор:	др. Небојша Пјевалица	



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:	Aleksandar Curguz	
Mentor, MN:	Nebojša Pjevalica, PhD	
Title, TI:	One solution for parental control in the car	
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: (chapters/pages/ref./tables/pictures/graphs/appendices)	7/28/0/5/18/0/0	
Scientific field, SF:	Electrical Engineering	
Scientific discipline, SD:	Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, S/KW:	service, auto, „OBDII system”, parental control, android	
UC		
Holding data, HD:	The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, N:		
Abstract, AB:	<p>This paper explains Android service for sending and receiving e-mail in case speeding has occurred. Client app has been made for changing e-mail sender and receiver also made for changing speed limits for particular section of road. Project is based on OBD II system, JavaMail API and Java programming knowledge.</p>	
Accepted by the Scientific Board on, ASB:		
Defended on, DE:		
Defended Board, DB:	President:	Jelena Kovacevic, PhD
	Member:	Ivan Kastelan, PhD
	Member, Mentor:	Nebojsa Pjevalica, PhD
		Menthor's sign

Zahvalnost

Zahvaljujem se institutu RT-RK, pre svega Automotive timu: Tomislavu Maruni, Nebojši Pjevalici, Marku Kovačeviću, Branimiru Kovačeviću, Nenadu Jovanoviću i Mladenu Kovačevu na stručnoj i nesebičnoj pomoći u toku izrade rada.

Na kraju se zahvaljujem porodici, prijateljima i kolegama na pruženoj moralnoj podršci.

SADRŽAJ

1.	Uvod.....	1
2.	Teorijske osnove	4
2.1	Android operativni sistem	4
2.1.1	Android servisi.....	6
2.1.2	Android Studio.....	9
2.2	OBD II.....	10
2.2.1	OBD II poruke	12
2.3	OBD II simulator.....	13
2.4	JavaMail biblioteka za slanje i primanje elektronske pošte	13
3.	Koncept rešenja.....	15
3.1	Analiza problema	15
3.2	Algoritam za rešenje problema	16
3.3	Problemi pri projektovanju	17
3.4	Realizacija rešenja	17
3.5	Modularnost.....	18
3.6	Prednosti u odnosu na postojeća rešenja	18
4.	Programsko rešenje	19
4.1	Moduli servisa	19
4.1.1	Modul aktivnosti.....	20
4.1.2	Modul za slanje elektronske pošte.....	20
4.1.3	Modul za prijem elektronske pošte	21
4.1.4	Programska nit za slanje zahteva o stanju brzine	22
4.1.5	Modul servisa za slanje i prijem elektronske pošte	23

5.Rezultati i verifikacija	24
6.Zaključak	27
7.Literatura	29

SPISAK SLIKA

<i>Slika1 Komunikacija OBDII sistema sa automobilom.....</i>	3
<i>Slika 2 Programski stek Android operativnog sistema</i>	5
<i>Slika3Android maskote.....</i>	6
<i>Slika4 Sve verzije Android operativnog sistema</i>	6
<i>Slika5 Životni vek Android servisa.....</i>	8
<i>Slika6 Slikovit prikaz “ Logcat-a ”</i>	9
<i>Slika7 Uključivanje režima za razvoj aplikacije na uređaju.....</i>	10
<i>Slika 8 OBD II konektor</i>	11
<i>Slika9 OBD II simulator</i>	13
<i>Slika10 JavaMail API dijagram</i>	14
<i>Slika11 Koncept rešenja po glavnim modulima</i>	16
<i>Slika12 Dobijanje pristupa Gmail nalogu</i>	18
<i>Slika13 Komunikacija servisa za razmenu elektronske pošte sa simulatorom/motorom vozila.....</i>	19
<i>Slika14 Dijagram klasa servisa za slanje i prijem elektronske pošte</i>	20
<i>Slika15 Pokretanje servisa i prikazivanje servisa u rukovaocu Android aplikacijama</i>	25
<i>Slika16 Korisničke opcije korisničke aplikacije</i>	25
<i>Slika17 Poslata elektronska pošta o prekoračenju brzine</i>	26
<i>Slika18 Prikaz poruke od roditelja na ekranu računara u vozilu</i>	26

SPISAK TABELA

<i>Tabela 1 Stanja android servisa</i>	7
<i>Tabela 2 Režimi poruka</i>	12
<i>Tabela 3 Funkcije modula za slanje elektronske pošte</i>	21
<i>Tabela 4 Funkcije modula za prijem elektronske pošte</i>	22
<i>Tabela5 API funkcije iz servisa za prihvatanje informacija(CarInfoService) sa ECU/simulator.....</i>	23

SKRAĆENICE

ADB - Android Debug Bridge, Android alat za ispravljanje grešaka

API – Application Programming Interface, Programski prilagodni sloj

C2DM - Android Cloud to Device Messaging, slanje podataka sa servera na aplikacije na Android uređajima

CAN – Controller Area Network, mreža (magistrala) za razmenu poruka u automobilskom sistemu

ECU – Engine Control Unit, Kontrolna jedinica motora

IMAP – Internet Email Access Protocol, protokol za pristupanje porukama na Internetu

JNI – Java Native Interface, Programski okvir koji omogućava spregu Java programskog jezika sa drugim programskim jezicima

OBD – On-Board Diagnostic, Dijagnostika na uređaju (automobilu)

PID - Performance Information Data, informacije o karakteristikama automobila

SAE – Society of Automotive Engineers, udruženje automobilskih inženjera

SMTP – Simple Mail Transfer Protocol, jednostavan protokol za slanje elektronske pošte

SSL – Secure Sockets Layer, protokol bezbednog prenosa podataka

TTL – Transport Layer Security, bezbednost transportnog sloja

1. Uvod

U ovom radu realizovan je Android servis kao i korisnička aplikacija za očitavanje brzine kretanja vozila, kao i slanje i primanje elektronske pošte u slučaju prekoračenja brzine.

Zbog sve većeg nepoštovanja zakona i razvijanja nepropisnih brzina, kao i sve boljih perfomansi automobila potrebno je nekako ući u trag nesavesnim vozačima, pre svega mlađoj populaciji. Jedno rešenje, realizovano u ovom radu, predstavlja roditeljsku kontrolu nad automobilom. Ona podrazumeva da roditelji u svakom trenutku mogu imati uvid kojom brzinom se kreće automobil kojim upravlja njihovo dete, a da pritom vozač (dete), nema predstavu da se informacije iz automobila šalju roditelju. Roditeljska kontrola podrazumeva kontrolu brzine na različitim deonicama puta, jer postoje različita ograničenja brzine (npr. na autoputu - 120 km/h, kroz naseljeno mesto -50 km/h, itd.). Kada brzina odstupi od predviđenog ograničenja, šalje se elektronska pošta roditelju (engl. *E-mail*), a da vozač (tinejdžer) to ne zna. Po prijemu elektronske pošte roditelj može poslati elektronsku poštu čiji se sadržaj automatski prikazuje na ekranu računara u automobilu, u vidu upozoravajućeg dijaloga.

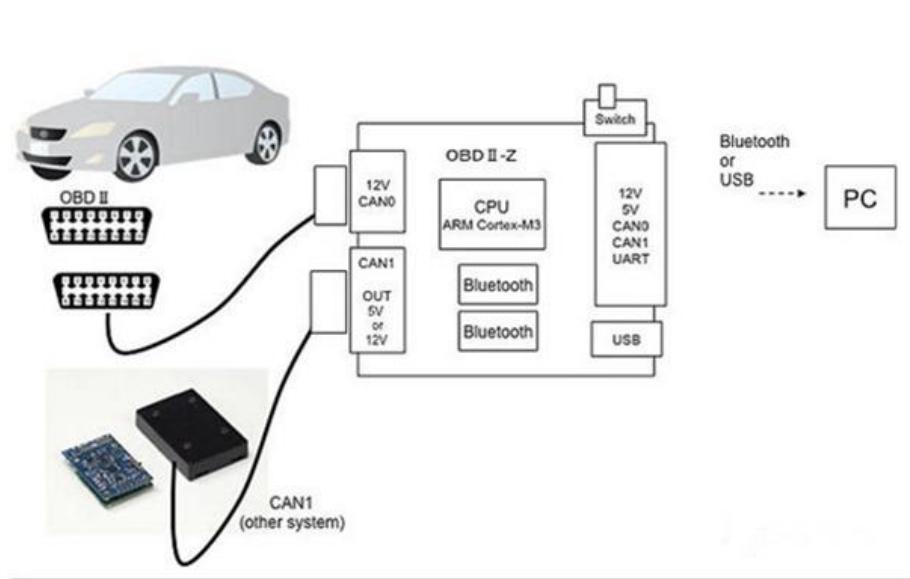
Ovo treba da probudi svest kod mlađe populacije kako bi se pridržavali predviđenih ograničenja na različitim deonicama puta. Ako bi se kojim slučajem desila saobraćajna nesreća, ustanovilo bi se kolikom je brzinom vozilo išlo i da li se pridržavalo propisanih ograničenja na toj deonici puta. Svakako da je ovo bitna stavka u daljoj istrazi o krivcu i načinu izazivanja saobraćajne nesreće.

Za korišćenje ovog rešenja Android servisa kao i korisničke aplikacije neophodno je posedovati uređaj sa Android operativnim sistemom i podrškom za „Bluetooth“ komunikaciju. Razlog za ovakav izbor je sve veća prisutnost Android operativnog sistema u uređajima potrošačke elektronike. Android je najzastupljeniji u oblasti mobilnih telefona, tableta, pametnih kuća, pametnih televizora, a uskoro i u računarima u vozilima novije generacije. Ovi računari u

vozilu predstavljaju budućnost automobila. Android operativni sistem je otvorenog koda. Proizvođači uređaja mogu ga lako prilagoditi svojim potrebama, Dostupan je svima i nudi raznovrsne API-je.

U ovom rešenju realizovan je Android servis kao deo korisničke aplikacije. Ovo rešenje je prihvatljivo iz razloga što u okviru servisa postoji potreba za korisničkim podešavanjima u kojima korisnik (roditelj) podešava adresu pošiljaoca i primaoca elektronske pošte, kao i ograničenje brzine. Samo zbog ovog detalja servis je realizovan na ovaj način. Sve ostale odlike servisa su zadržane, a to su pre svega dobavljanje brzine direktno iz automobila (ili putem simulatora događaja iz automobila) kao i slanje i primanje elektronske pošte u pozadini. Ovim je omogućeno da se i dalje neometano koristi operativni sistem u vozilu, jer će se sve odvijati u pozadini daleko od očiju onoga koji kontroliše uređaj u automobilu, jer on ne treba ni da zna da ovako nešto postoji.

Za dobavljanje informacija od vozila ili u slučaju laboratorijskih ispitivanja simulatora, korišćen je OBD-II (engl. *On-Board diagnostic*) sistem koji se po standardu koristi u svim vozilima. Kako su svi proizvođači vozila shvatili da je neophodno uspostaviti način komunikacije automobila sa spoljašnjosti, radi lakšeg servisiranja i poboljšanja karakteristika vozila osmišljen je OBD-II sistem koji povezuje računar u automobilu sa njegovim motorom kao i mnogim drugim spoljašnjim jedinicama u jednu celinu. Takođe definiše način komunikacije i sinhronizaciju između kontrolne jedinice motora i spoljašnjih uređaja za bilo koju marku automobila. Svaki proizvođač ima nešto specifično za svoje modele vozila. Na primer, greške u vozilu se razlikuju od proizvođača do proizvođača, ali je način njihovog dobavljanja za svaku marku vozila isti što je suština ovog sistema. Sa razvojem OBD-II sistema i aplikacija koje koriste ovaj sistem, vlasnici automobila mogu se sa Bluetooth modulom (engl. *Bluetooth dongle*) priključiti na OBD II konektor u vozilu. U današnje vreme ne postoji više potreba da se pri kupovini polovnog vozila ide kod majstora na autodijagnostiku.



Slika1 Komunikacija OBDII sistema sa automobilom

Na *Slika1* prikazana je komunikacija OBD II sistema sa automobilom. Uređaj za autodijagnostiku se preko OBD II konektora povezuje sa automobilom, a uređaj za autodijagnostiku sa personalnim računarom ili laptopom preko usb kabla ili bluetooth protokola na kojem postoji program za dobavljanje i prikaz željenih parametara (brzine, grešaka na motoru automobila, itd). Komunikacija teče tako što korisnik na laptopu zatraži željeni parameter i prosledi ga uređaju za autodijagnostiku koji zahtev prosledi na CAN magistralu (engl. *Controller Area Network*) i čeka da kontrolna jedinica motora obradi zahtev i vrati odgovor na zahtev. Po dobijanju odgovora, odgovarajući program na laptopu prikazuje vrednost zahtevanog parametra.

2. Teorijske osnove

U ovom poglavlju su objašnjeni svi potrebni moduli koji su korišteni kako bi se realizovao dati problem.U te module spadaju:

- Android operativni sistem
- Android servisi
- OBD II sistem
- JavaMail Api za slanje i primanje elektronske pošte
- Simulator za dobavljanje određenih parametara od automobila

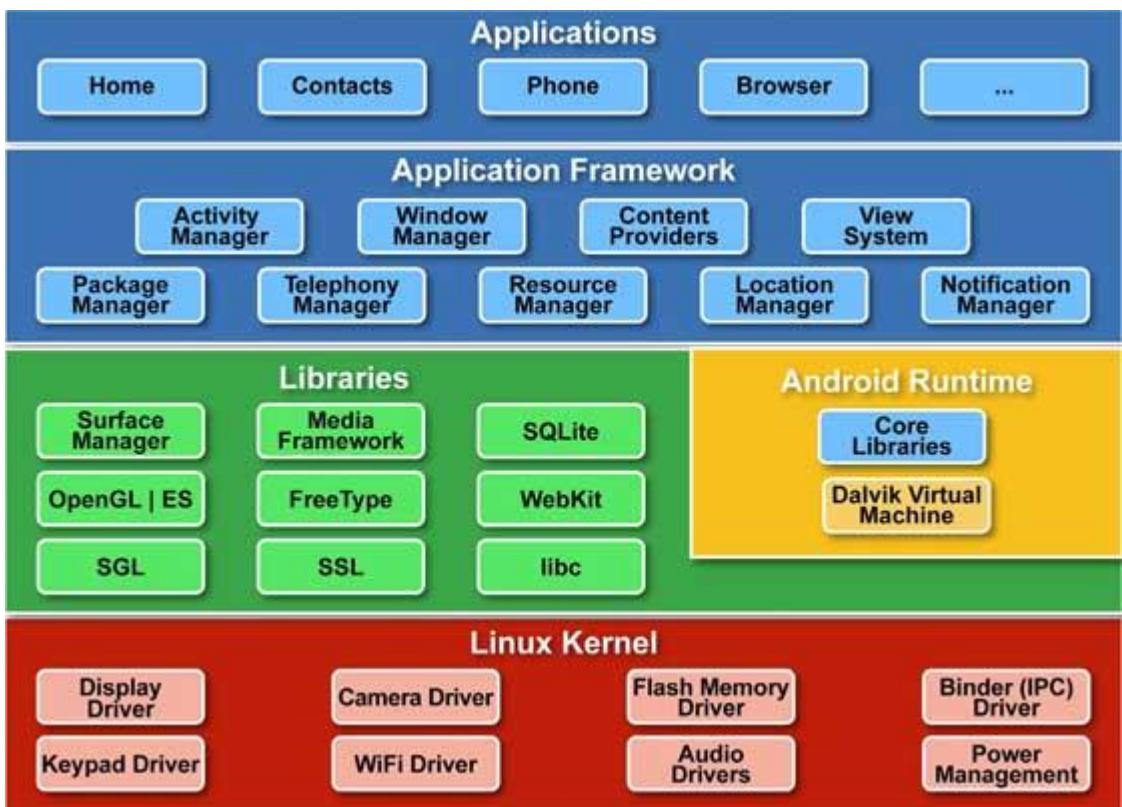
2.1 Android operativni sistem

Android operativni sistem [1][2] je trenutno najrasprostranjeniji operativni sistem za mobilne telefone. Zasnovan je na Linux jezgru i prilagođen je tako da se može koristiti na većini mobilnih uređaja, uključujući mobilne telefone, tablet uređaje, televizore, satove, naočare. Sa tehničke strane Android predstavlja Linux operativni sistem razvijen za ARM i x86 arhitekturu i sastoji se od modifikovanog monolitnog Linux jezgra zaduženog za podršku fizičke arhitekture funkcija niskog nivoa.

Jedna od najvećih prednosti koju imaju Android korisnici su mnogobrojne Android aplikacije. Android nudi ogromnu selekciju aplikacija koju je moguće dobaviti putem aplikacije Google prodavnice (engl.*Google Store*). Ona predstavlja centralno mesto koje sadrži aplikacije za različite namene. Korisnici mogu naći aplikacije za produktivnost, za gledanje video materijala, čitanje knjiga, igre, itd. Google prodavnica sadrži preko jedan milion aplikacija. Cena preuzimanja aplikacija varira. Većina aplikacija su besplatne, ali dobar broj aplikacija se plaća po nekoliko dolara. Prema podacima iz maja 2013.godine, 48 milijardi aplikacija je instalirano sa Google prodavnice. Google Play prodavnica je za razliku od sličnih aplikacija za druge

operativne sisteme otvorena i nema striktna pravila. Ovo se sa jedne strane pokazalo kao dobra stvar, međutim lažne aplikacije i aplikacije sa sadržajem za odrasle su uspele da se nađu ovde. Google se dosta bavi uklanjanjem ovakvih aplikacija, ali ne toliko dobro kao što je to slučaj u prodavnicama drugih operativnih sistema.

Android operativni sistem ima slojevitu programsku strukturu. Sledеća slika prikazuje programske slojeve počev od najnižeg (Linux jezgro) do najvišeg (Aplikativnog sloja). Android programski stek je grubo podeljen na 5 celina i 4 glavna sloja, kao sto je prikazano na *Slika 2*.



Slika 2 Programski stek Android operativnog sistema

Na *Slika 2* vidimo slikovit prikaz programskog steka Android operativnog sistema. Kao što se može videti, na dnu steka nalazi se Linux jezgro (engl.*Linux kernel*) koji predstavlja srce Android operativnog sistema. Linux jezgro obezbeđuje osnovne sistemske funkcionalnosti poput upravljaljanja procesima, upravljanja memorijom, upravljanje uređajima poput kamere, tastature, ekrana itd. Na vrhu Linux jezgra nalazi se grupa biblioteka koje imaju razne namene. Na primer, SQLite baza podataka je korisna biblioteka za čuvanje i deljenje aplikacijskih podataka. SSL biblioteke su odgovorne za internet bezbednost. Neke druge biblioteke se koriste za puštanje muzike, snimanje glasa i reprodukciju video sadržaja. U Android RunTime sekciji, nalazi se Dalvik virtualna mašina koja je vrsta Java virtualne maštine, specijalno dizajnirana i optimizovana za Android. Dalvik virtualna mašina omogućava svakoj aplikaciji da se pokrene iz svog procesa sa vlastitim objektom Dalvik virtualne maštine. Ostala dva najviša nivoa su aplikacioni programski okvir (engl.*Application Framework*) i aplikacije. U aplikacionom

programskom okviru se nalaze sistemski servisi odgovorni za rukovanje pozivima, obaveštenjima, alarmom, itd. Njih koriste aplikacije sa vrha programskog steka Android operativnog sistema.

Android je široko rasprostranjen operativni sistem i prednjači u odnosu na druge sisteme po broju prodatih uređaja. Jedna od anegdota vezana za Android je ta da su imena različitih verzija Androida [3] davana po slatkišima *Slika3*, po abecednom redosledu respektivno. Na primer, Froyo(zamrznuti jogurt), Gingerbread (medenjak), Ice Cream Sandwich (sladoled u sendviču) *Slika4*.



*Slika3*Android maskote



Slika4 Sve verzije Android operativnog sistema

2.1.1 Android servisi

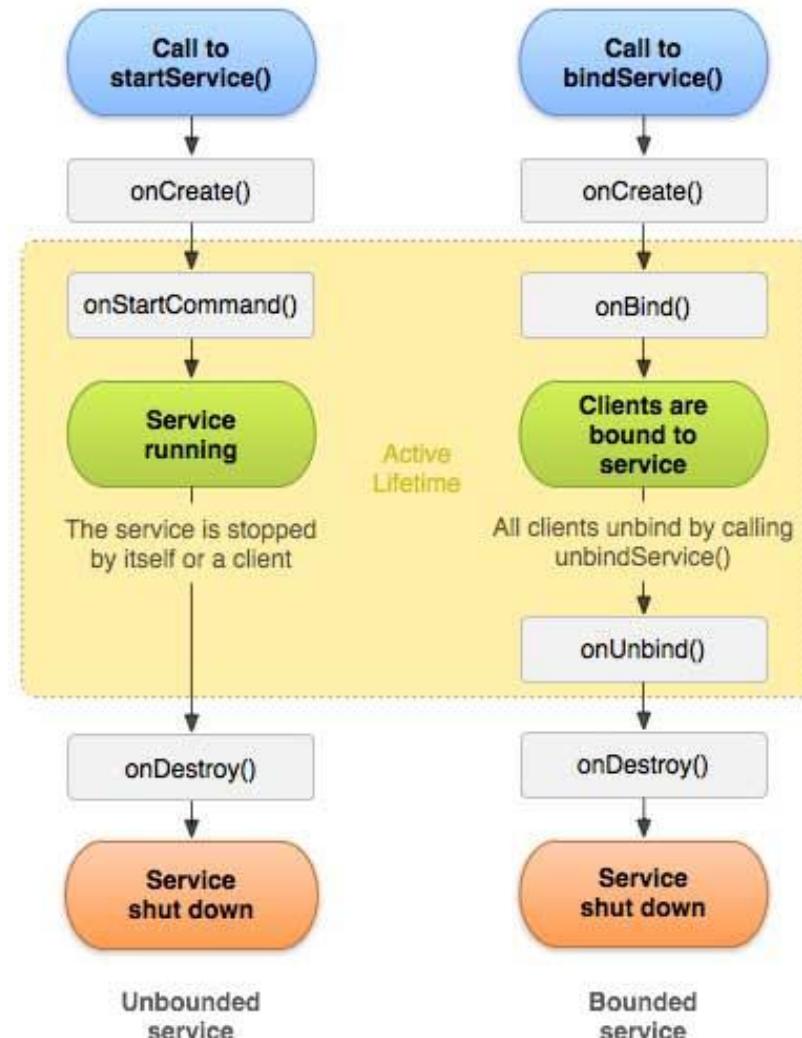
Android servis [4] je programska komponenta koja se izvršava u pozadini i obavlja dugotrajne operacije, bez potrebe za interakcijom sa korisnikom. Na primer, servis može da

pušta muziku u pozadini, dok je korisnik u drugoj aplikaciji ili može da skida podatke preko interneta bez blokiranja korisnikove interakcije sa aktivnošću neke druge aplikacije. Android servis u suštini ima dva stanja koja su prikazana u *Tabela 1*:

Stanje	Opis
Pokrenut	Servis je pokrenut kada ga aplikaciona komponenta pokrene pozivom funkcije <code>startService()</code> . Jednom pokrenut, servis se izvršava pozadini neodređeno dugo, čak i ako se zaustavi komponenta koja ga je pokrenula.
Povezan	Servis je vezan kada se aplikaciona komponenta veže za njega pozivanjem funkcije <code>bindService()</code> . Vezani servis nudi korisnik-server spregu koja dozvoljava komponentama da komuniciraju sa servisom, šalju zahteve, dobijaju rezultate od servisa, itd.

Tabela 1 Stanja android servisa

Dijagrami prikazani na *Slika 5* ilustruju životni vek Android servisa. Dijagram na levoj strani prikazuje životni vek servisa, kada je on pokrenut funkcijom `startService()`, a dijagram na desnoj strani prikazuje životni vek servisa, kada je on pokrenut funkcijom `bindService()`.



Slika5 Životni vek Android servisa

Postoje dve vrste Android servisa:

- Udaljeni (engl. *Remote Service*)
- Lokalni (engl. *Local Service*)

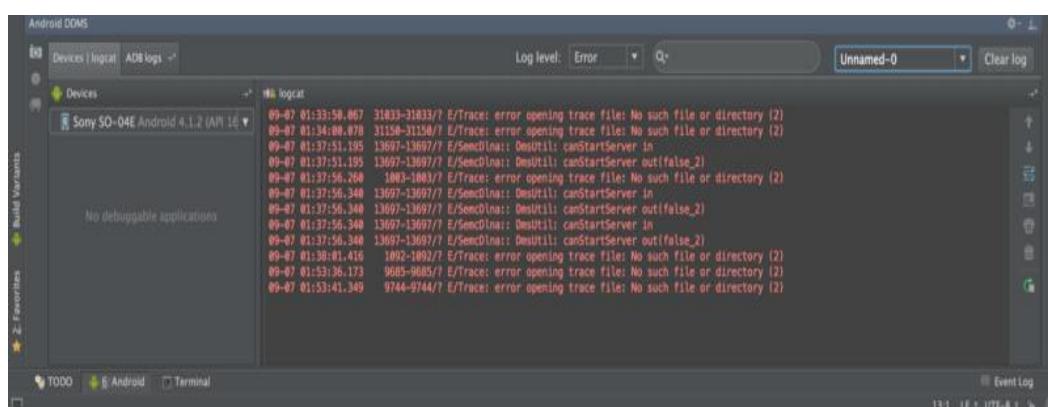
Udaljeni servisi se izvršavaju u posebnom memorijskom prostoru i sa njima se komunicira preko među-procesne komunikacije (engl. *InterProcess Communication*), dok se lokalni servis izvršava u istom memorijskom prostoru kao i aplikacija koja ga je pokrenula.

Za potrebe ovog projekta korišćen je lokalni servis čiji je zadatak da dobavlja brzinu kretanja sa motora (simulatora) i da u odnosu na ograničenje brzine koje je definisano u podešavanjima korisničke aplikacije, šalje elektronsku poštu na adresu podešenu u podešavanjima, u slučaju da je dobavljena brzina kretanja veća od ograničenja. Servis se pokreće iz korisničke aplikacije. Kada se servis pokrene, korisnik može izaći iz korisničke aplikacije i neometano koristiti uređaj za svoje potrebe. Na sličan način se vrši zaustavljanje servisa. Ponovnim ulaskom u korisničku aplikaciju i pritiskom na određeno dugme trajno se zaustavlja servis do ponovnog pokretanja. Takođe ulaskom u korisničku aplikaciju mogu se promeniti

korisnička podešavanja vezana za slanje i prijem elektronske pošte, kao i podešavanja vezana za ograničenje brzine.

2.1.2 Android Studio

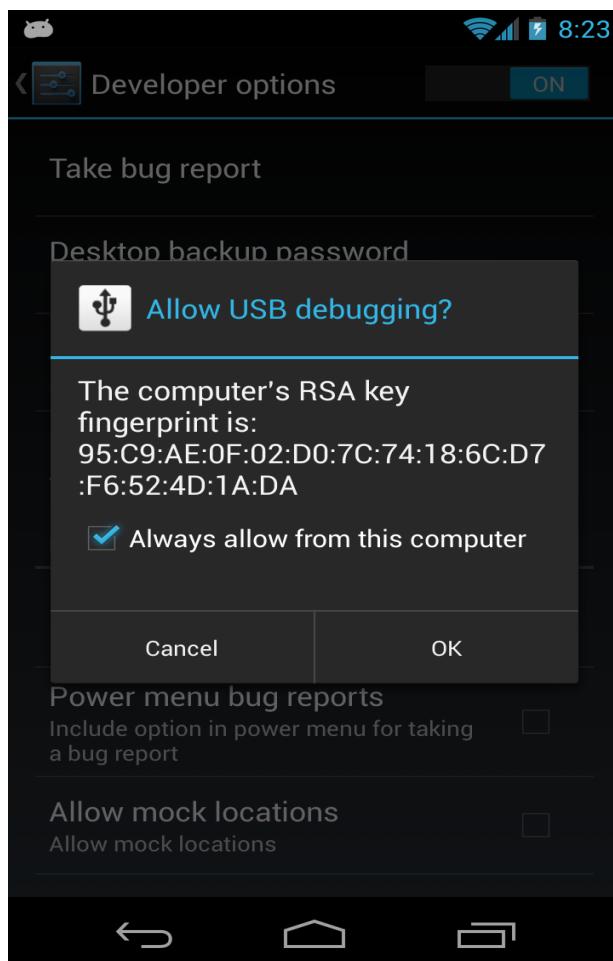
Android Studio predstavlja razvojno okruženje za izradu Android aplikacija. Napravljen je od strane kompanije Google 2013.godine. Omogućava pisanje programskog koda, praćenje ponašanja aplikacija zahvaljujući izlaznim porukama (engl.*Logcat*) i praćenje izvršavanje programa koji umnogome olakšava otkrivanje i otklanjanje grešaka. Prednost ovog programa je što se može nabaviti besplatno,zahvaljujući njegovoj “Apache 2.0” licenci. Da bi mogli ispitati ispravnost Android aplikacije na nekom ciljnog uređaju, potrebno je instalirati ADB upravljački programa (engl. *Android Debug Bridge driver*) i preko univerzalne serijske magistrale (engl. *Universal Serial Bus*) izvršiti povezivanje ciljnog uređaja i računara na kojem je pokrenut Android Studio. Kako bi se ostvarila komunikacija, potrebno je na uređaju na kojem se testira aplikacija omogućiti režim za razvojno okruženje (engl. *Developer options*).



Slika6 Slikovit prikaz “Logcat-a”

Android logcat sistem obezbeđuje mehanizam za kolekciju i prikaz sistemskih izlaznih poruka kao i poruka koje programer u toku razvijanja aplikacije napravi. Ima dva dela koji se sastoje iz znakovnih nizova (engl.*String*). Prvi znakovni niz predstavlja uglavnom ime, obeležje neke funkcije ili klase u kojoj će biti ispisana poruka, dok drugi znakovni niz predstavlja tekst poruke. Prikaz “Logcat-a” prikazan je na *Slika6*.

Android Studio se uglavnom koristi za pisanje programa u Java programskom jeziku,ali sem Java programske podrške gde se koristi programski jezik C naziva se sopstveni kod (engl.*Native*), a mehanizam koji predstavlja spregu između Java programskog koda i C programske podrške naziva se JNI (engl. *Java Native Interface*).



Slika7 Uključivanje režima za razvoj aplikacije na uređaju

Kako bi se razvijale aplikacije na mobilnom uređaju, potrebno je instalirati određene upravljačke programe (engl. *driver*) kako bi se mogao koristiti režim za razvojno okruženje na mobilnom uređaju. Na slici Slika7 prikazano je uključivanje režima rada za razvijanje aplikacija na Android operativnom sistemu.

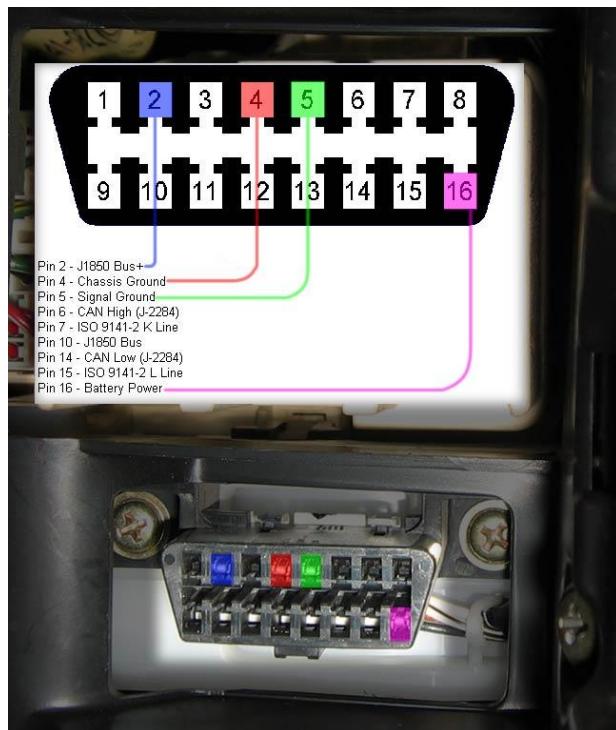
2.2 OBD II

OBD (engl. *On-Board Diagnostic*) [5] je sistem koji se nalazi u većini automobila današnjice. Američki kongres je 1970. godine izglasao zakon o čistijem vazduhu i osnovao agenciju za zaštitu životne sredine. Da bi uspeli ostati u okviru ovih standarda proizvođači automobila su se okrenuli sistemima u automobilu koje kontroliše elektronika. Tako su uspevali da kontrolišu funkcije motora i dijagnostikuju njegove probleme. Senzori su merili performanse motora i prilagođavali sistem da zagadenje bude minimalno. Tokom godina OBD sistem se usavršavao.

Na početku je postojalo nekoliko standarda i svaki je imao svoj sistem i signale. Godine 1988. SAE (eng. *Society of Automotive Engineers*) je postavio standard kakav se konektor koristi i set dijagnostičkih ispitnih (engl. *Test*) signala. OBD II predstavlja proširenje ovog početnog

standarda. OBD II omogućava gotovo kompletну kontrolu nad motorom i takođe nadgleda delove šasije i tela automobila. Njegovo korišćenje je počelo od 1. Januara 1996.godine.

Svi automobili koji su proizvedeni od 1996. godine pa nadalje imaju OBD II sistem. Najbolji način da se to potvrdi jeste potražiti OBD II konektor [6] u vozilu koji se po standardu nalazi najviše jedan metar od vozača i izgleda kao na *Slika 8*. Označeni pinovi na slici su pinovi koji se koriste u OBD II sistemu. Pin 2 predstavlja J1850 magistralu, pin 4 masu šasije automobila, pin 5 signal uzemljenja, pin 16 napajanje sa akumulatora.



Slika 8OBD II konektor

Za OBD II sistem postoji 5 mogućih protokola. Protokoli se razlikuju po načinu komunikacije. Koji protokol se koristi može se utvrditi po pinovima na konektoru:

- J1850 VPW – ovakav konektor mora imati metalne kontakte u priključcima 2, 4, 16, ali ne u 10
- ISO 9141-2/KWP2000 – ovakav konektor mora imati metalne kontakte u priključcima 4, 5, 7, 15 i 16
- J1850 PWM – ovakav konektor mora imati metalne kontakte u priključcima 2, 4, 5, 10, i 16
- CAN – ovakav konektor mora imati metalne kontakte u priključcima 4, 5, 6, 14 i 16 (od 2008. godine samo je on u upotrebi)

2.2.1 OBD II poruke

Komunikacija OBD II sistema sa automobilom se obavlja preko OBD II poruka tako što se na OBD II konektor prikači odgovarajući alat (npr. Bluetooth dongle). Preko alata šalju se poruke na CAN magistralu. Kada kontrolna jedinica motora prepozna poruke, vraća odgovor na CAN magistralu. Sa CAN magistrale se odgovor šalje preko Bluetooth dongle na uređaj, aplikaciju koja je tražila zahtev za određeni parametar. Kada je parametar primljen onda se iz aplikacije mogu očitati željeni parametri. Na ovaj način se radi autodijagnostika u automobilima. Ako postoji uređaj kao što je mobilni telefon sa ugrađenim bluetooth uređajem, može se na njega povezati bluetooth-a koja je na OBD II konektoru i uz pomoć određene aplikacije na telefonu dobaviti željene parameter (brzina, broj obrtaja, nivo goriva, itd.).

Za dobavljanje različitih podataka koriste se različite poruke kao što je predstavljeno u *Tabela 2.*

Režimi rada	Opis režima rada
01	Trenutni podaci
02	Sa ovim se mogu dobaviti podaci nastali u trenutku kada je ustanovljena greška u sistemu
03	Kodovi greške
04	Uklanjanje svih kodova greške i sačuvanih vrednosti
05	Nadgleda senzor za kiseonik(ne CAN protokol)
06	Nadgleda senzor za kiseonik(samo CAN protokol)
07	Kodovi greške koji suse dogodili tokom trenutne ili poslednje vožnje, tj. nepotvrđeni kodovi greške
08	Kontrolne operacije nad OBD II sistemom

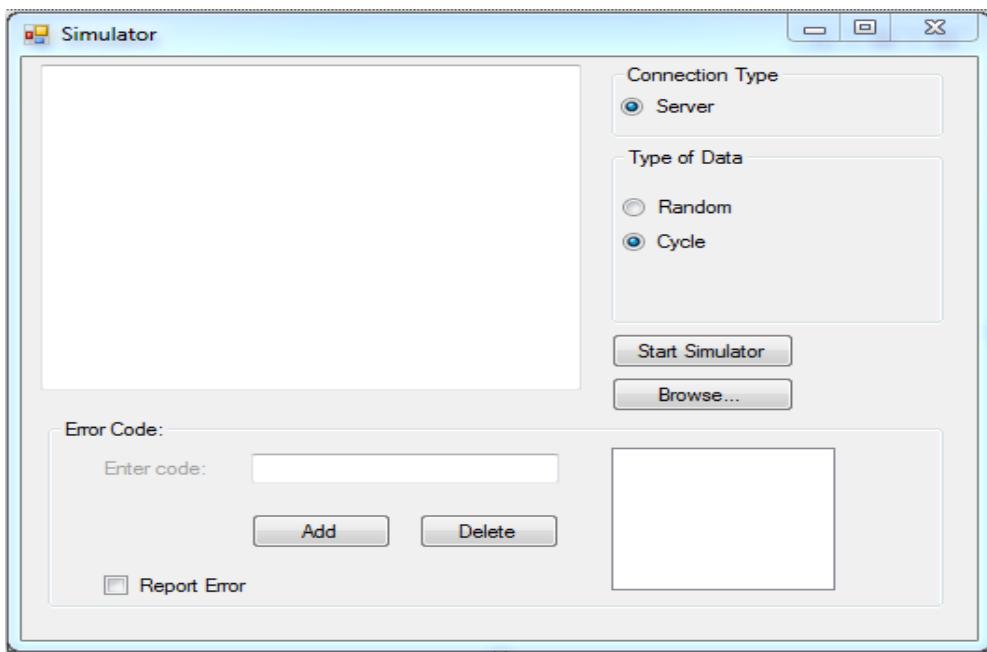
Tabela 2 Režimi poruka

Osim režima prikazanih u tabeli, u upotrebi su i režimi [7] za dobavljanje određenih vrednosti. Na režime 01 i 02 se još dodaje posebna grupa poruka za tačno određeni parametar. Npr. ako se želi dobaviti trenutna brzina automobila na vrednost 01 iz tabele 2 se dodaje poruka 0D, ako se želi dobiti brzina kada je nastala neka greška na vrednost 02 se dodaje poruka 0D. Za broj obrtaja se dodaje poruka 0C, za temperaturu poruka 05, itd.

Kada se ove poruke šalju na simulator događaja u vozilu ili CAN magistralu svaka poruka se mora završiti sa ‘\r’ tj. novim redom, zato što tako nalaže protokol.

2.3 OBD II simulator

Ovaj simulator simulira rad automobila (motora i ostalih komponenti). Korišćen je za realizaciju projekta. Simulator ima više režima generisanja vrednosti. Npr. može se izabratи da se podaci za sve parametre nasumično generišu. Takođe postoji i opcija da se u grafičkom okruženju zada tačna vrednost određenog podatka. OBD II simulator je predstavljen na *Slika9*.



Slika9 OBD II simulator

2.4 JavaMail biblioteka za slanje i primanje elektronske pošte

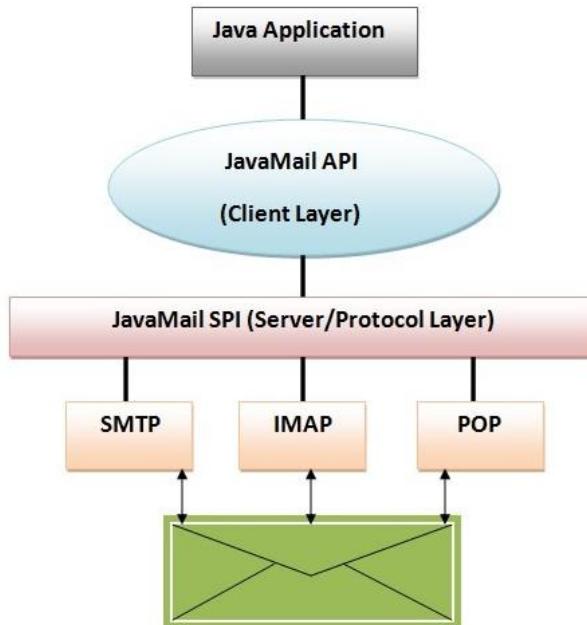
JavaMail API (engl. *Java Mail Application Programming Interface*) [9] je standardna Java ekstenzija koja obezbeđuje biblioteku klasa za korisnike elektronske pošte. Programi koriste JavaMail API kako bi komunicirali sa SMTP (engl. *Simple Email Transfer Protocol*), POP3 (engl. *Post Office Protocol*) i IMAP (engl. *Internet Email Access Protocol*) serverima i slali i primali elektronsku poštu. Ovaj API štiti aplikativne programere od detalja niskog nivoa protokola. JavaMail je reprezentacija visokog nivoa osnovnih komponenata proizvoljnog elektronskog sistema. Komponente su predstavljene apstraktnim klasama paketa `javax.mail`.

Npr. apstraktna klasа `javax.mail.Message` predstavlja elektronsku poruku. Ona deklariše apstraktne metode za dobavljanje i postavljanje raznih informacija o poruci, kao što su posiljalac i primalac, datum slanja poruke, naslov poruke, predmet poruke.

Apstraktna klasа `javax.mail.Folder` predstavlja kontejner poruka. Ona deklariše apstraktne metode za dobavljanje poruke iz foldera, premeštanje poruka iz foldera u folder i brisanje poruka iz foldera. Sve ove klase su apstraktne jer ne prave nikakve prepostavke o tome kako se elektronska pošta čuva ili šalje između mašina. Konkretne potklase ovih klasa specijalizuju apstraktne klase za određene protokole i elektronske formate. Za korišćenje

standardne Internet elektronske pošte, može se koristiti `javax.mail.MimeMessage` umesto `javax.mail.Message`, `javax.mail.internet.Address` umesto `javax.mail.Address` i `com.sun.mail.pop3.POP3Store` umesto `javax.mail.Store`.

Za potrebe projekta korišćeni su SMTP server za slanje poruka kao i IMAP server za prijem poruka i otvaranje sesije za pristup poštanskom sandučetu određenog Gmail naloga. Dijagram JavaMail API –ja je prikazan na *Slika10*.



Slika10JavaMail API dijagram

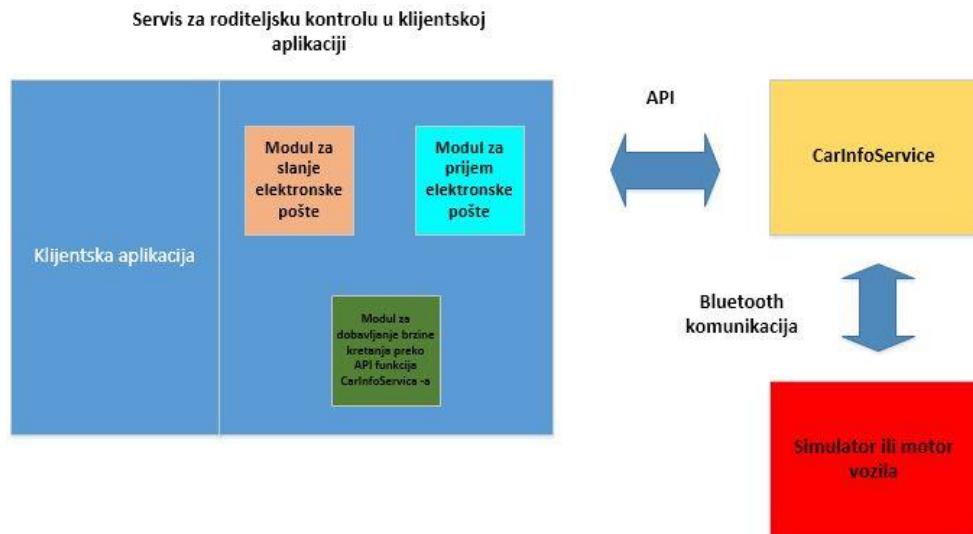
3. Koncept rešenja

U ovom poglavlju predstavljena je analiza problema i opisani su korišteni algoritmi.

3.1 Analiza problema

U radu je potrebno realizovati lokalni servis koji će omogućiti slanje elektronske pošte, kada vozač u automobilu prekorači ograničenje brzine koje je definisano u podešavanjima korisničke aplikacije, kao i prijem elektronske pošte od lica koje je zaduženo za kontrolu vozila (od roditelja). Kada roditelj dobije elektronsku poštu o prekoračenju brzine i odgovori na nju, na računaru u vozilu treba da se prikaže dijalog upozorenja sa sadržajem poruke koju je roditelj poslao. Cilj ovog servisa za slanje i prijem elektronske pošte je da se integriše u računar u vozilu, koji je baziran na Android operativnom sistemu. Servis je realizovan kroz korisničku aplikaciju koja poseduje korisnička podešavanja (engl. *Settings*) u kojima se podešava adresa pošiljaoca, adresa primaoca elektronske pošte i određeno ograničenje brzine shodno deonici puta kojom će se vozilo kretati. Takođe u korisničkoj aplikaciji može se pokrenuti ili zaustaviti servis. Jednom pokrenut, servis radi u pozadini, dok god ga korisnik ne zaustavi. Servis takođe u sebi ima modul koji na određen vremenski period zahteva brzinu od simulatora događaja motora ili motora automobila preko servisa za dobavljanje parametara u kojem je integrisan OBD II sistem. Slanje i prijem elektronske pošte je realizovan kroz servis prevashodno, jer vozač u automobilu ne treba da ima predstavu da kontrola brzine uopšte postoji kako bi mogao neometano koristiti Android sistem na računaru u vozilu. Servis se pokreće pri paljenju automobila i Android sistema. Za promene postavke korisničkih opcija, potrebno je ponovo ući u korisničku aplikaciju i načiniti izmene. Sve to vreme servis radi u pozadini.

3.2 Algoritam za rešenje problema



Slika11 Koncept rešenja po glavnim modulima

Na *Slika11* prikazan je koncept rešenja po glavnim modulima. Modul sa leve strane predstavlja servis za slanje, prijem elektronske pošte i dobavljanje brzine kretanja vozila. Modul za dobavljanje brzine iz servisa roditeljske kontrole ostvaruje komunikaciju sa servisom za prihvatanje informacija (CarInfoService) sa simulatora ili motora automobila. U servisu roditeljske kontrole pod modulom za dobavljanje brzine kretanja automobila sa simulatora definiše se koliko se često traži brzina i pokreću se moduli za slanje i prijem elektronske pošte. Moduli za slanje i prijem elektronske pošte su nezavisni jedan od drugog. Svaki od njih koristi različit protokol. Za slanje elektronske pošte se koristi SMTP protokol, dok za prijem IMAP protokol. Zajedničko za ova dva modula je to što moraju da izvrše autentifikaciju Gmail naloga za slanje i prijem elektronske pošte. Ova autentifikacija omogućava pristup Gmail nalogu bez potrebe unošenja lozinke ili eventualnog navođenja lozinke u programu, prilikom povezivanja određenog naloga na određen server kako bi se slale ili primale elektronske poruke. Zadatak servisa za prihvatanje informacija je da ostvari komunikaciju sa vozilom ili simulatorom. Modul servisa za roditeljsku kontrolu koji dobavlja brzinu kretanja vozila zahteva brzinu kretanja od servisa za prihvatanje informacija (CarInfoService). CarInfoService putem OBD II sistema očitava parametre brzine i vraća ih nazad modulu za dobavljanje brzine sa vozila koji je realizovan u okviru servisa za roditeljsku kontrolu. Na osnovu ovih zahteva brzine, izabranog pošiljaoca, izabranog primaoca elektronske pošte, kao i ograničenja brzine koja su podešena u opcijama korisničke aplikacije, servis šalje elektronsku poštu, ako je došlo do prekoračenja brzine. Treba napomenuti, da se uz određen algoritam, elektronska pošta šalje na neki određeni interval, a ne svaki put kada se brzina prekorači.

3.3 Problemi pri projektovanju

Prilikom izrade modula za slanje i prijem elektronskih poruka u okviru servisa za roditeljsku kontrolu, uočeni su problemi autentifikacije „Gmail“ naloga radi izbegavanja unošenja šifre za povezivanje na servere za slanje i prijem elektronske pošte. Ova autentifikacija ili bolje rečeno dobijanje sigurnosnih sertifikata za određeni nalog elektronske pošte je potrebna kako bi se pristupilo elektronskom sandučetu bez unošenja šifre korisnika. Kada se pristupi elektronskom sandučetu u ovom slučaju pošiljaoca, može se proveravati elektronsko sanduče na koje treba da stigne poruka od roditelja. Kada je poruka od roditelja primljena i pročitana, na ekranu računara u vozilu treba da se prikaže upozoravajući dijalog sa sadržajem koji je roditelj poslao putem elektronske pošte. Otvaranje i vršenje konstantnog učitavanja (engl. *polling*) svih poruka iz fascikle dolazećih poruka je proces koji zahteva puno resursa i vremena. Pod resursima podrazumevamo potrošnju memorije uređaja na kom se izvršava servis za roditeljsku kontrolu, kao i opterećenje Internet konekcije.

3.4 Realizacija rešenja

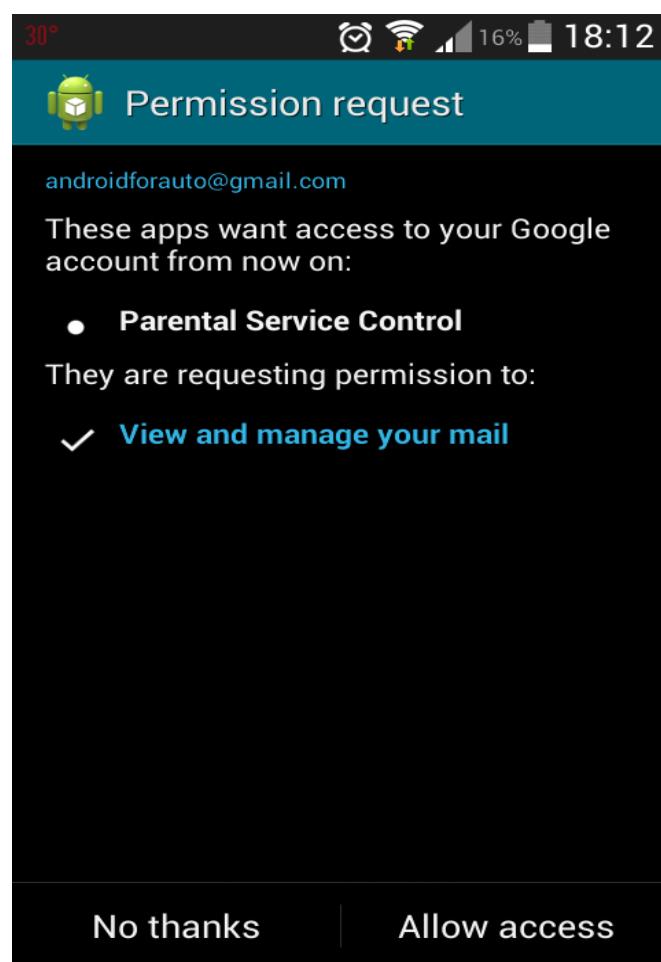
U poglavljiju 3.3 Problemi pri projektovanju navedeni su problemi pri projektovanju. Problem sa autentifikacijom određenog Gmail naloga je rešen korišćenjem modula Java Account Manager. On potražuje ključ u (engl. *token*) vidu bezbednosnih sertifikata od C2DM (engl. *Android Cloud to Device Messaging*) servera koji je zadužen za registraciju Gmail naloga na određenom Android uređaju i preuzimanje podataka od servera za prijem i slanje elektronske pošte. C2DM server preuzeće podatke, elektronsku poštu, šalje u određene aplikacije na Android uređaju koje obrađuju elektronsku poštu. Na taj način se pristupa Gmail elektronskoj pošti bez potrebe za unosom šifre. Sa dobijenim ključem servis za roditeljsku kontrolu može pristupiti želenom serveru elektronske pošte. Treba napomenuti da se za otvaranje sesije slanja i prijema elektronske pošte koristi SSL(engl. *Secure Socket Layer*) i TLS (engl. *Transport Layer Security*) bezbednosni (autentifikacioni) protokoli. Problem pristupanja fascikli pristiglih poruka željenog Gmail naloga je rešen korišćenjem funkcija iz JavaMail API-ja. API poseduje funkcije koje obaveštavaju korisnika o prispeću nove elektronske pošte. Na taj način nije potrebno samostalno proveravati prispeće novih poruka. Pri prijemu nove poruke, upozoravajući dijalog koji treba da se prikaže na ekranu i ispiše poruku od roditelja, nije moguće prikazati iz servisa roditeljske kontrole jer on nema grafičku korisničku spregu. Zbog toga je potrebno koristiti Java klasu, Window Manager, za prikaz upozoravajućeg dijaloga iz Android servisa.

3.5 Modularnost

Kako bi servis za roditeljsku kontrolu bio pogodan za eventualnu nadogradnju potrebno je izvršiti sistematizaciju i jasno grupisati njegove module u manje celine kako bi bio što pogodniji i jasniji za eventualne izmene.

3.6 Prednosti u odnosu na postojeća rešenja

Najveća prednost u odnosu na postojeća rešenja [8] je što korisnik ne mora unositi šifru za prijavu na željeni Gmail nalog elektronske pošte. Gmail nalozi se dobavlja na osnovu svih prijavljenih Google naloga na Android uređaju. Dovoljno je samo izborom opcije, u korisničkom podešavanju korisničke aplikacije, izabrati željeni nalog i na taj način izvršiti autentifikaciju, putem ključa sa bezbednosnim sertifikatima.



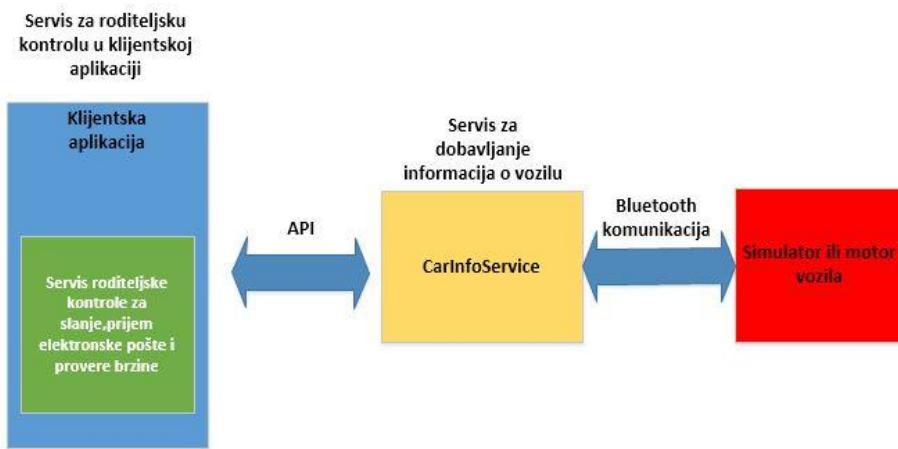
Slika12Dobijanje pristupa Gmail nalogu

Na Slika12prikazano je dobijanje bezbednosnih sertifikata za izabrani Gmail nalog elektronske pošte. Da bi određeni Android servis ili aplikacija mogla da koristi pristup nekom Gmail nalogu, potrebno je da prihvati bezbednosne sertifikate.

4. Programsко рење

U ovom poglavlju je predstavljeno programsko rešenje servisa roditeljske kontrole za slanje, prijem elektronske pošte i dobavljanje brzine kao i njegovih modula. Rešenje je realizovano u programskom jeziku Java na Android platformi. Radno okruženje korišćeno prilikom izrade zadatka je Android studio.

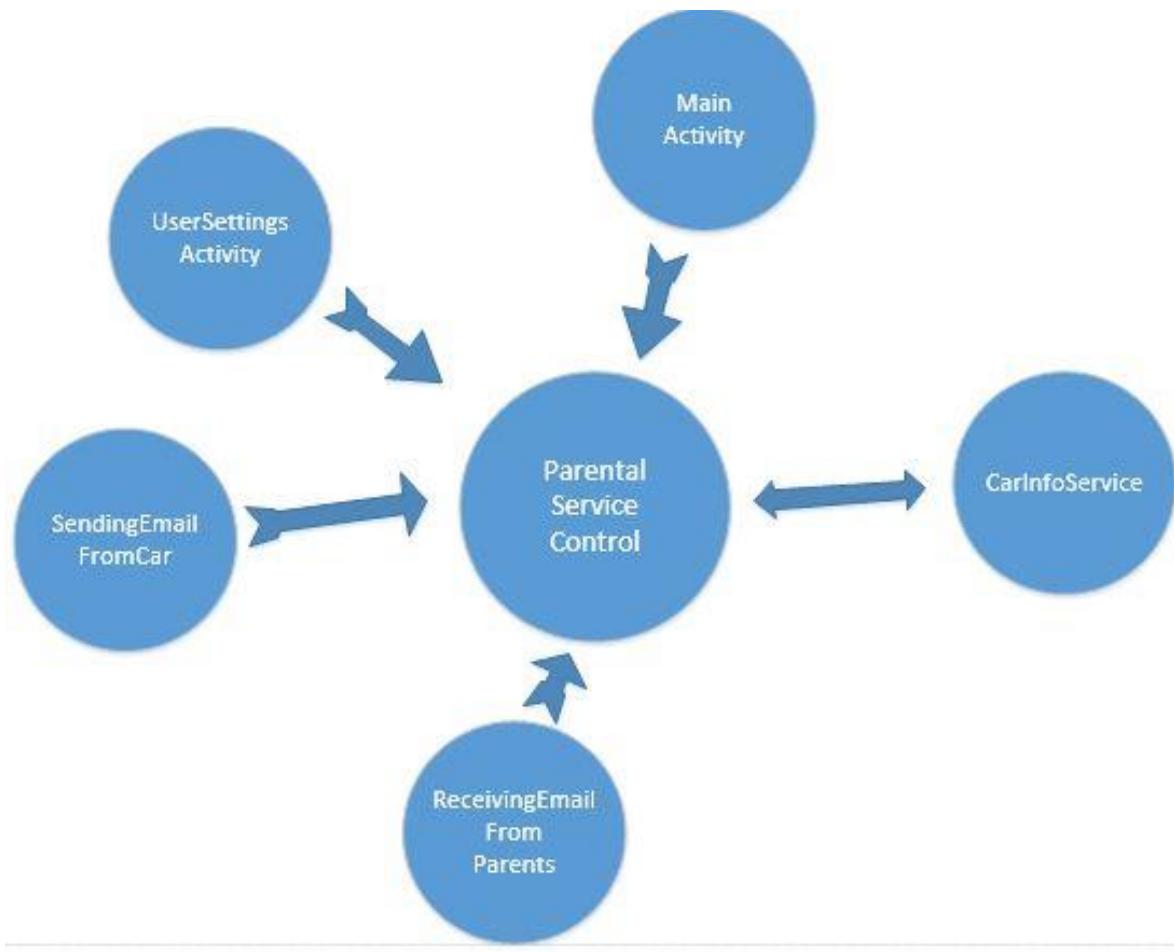
Servis roditeljske kontrole realizovan je kao lokalni Android servis i nalazi se u korisničkoj Java aplikaciji. Komunicira sa servisom za dobavljanje informacija o vozilu (CarInfoService) putem određenog API-ja, a CarInfoService sa simulatorom ili vozilom preko Bluetooth tehnologije za bežičnu komunikaciju. Način komunikacije modula prikazan je na *Slika13*.



Slika13 Komunikacija servisa za razmenu elektronske pošte sa simulatorom/motorom vozila

4.1 Moduli servisa

Dijagram svih klasa korisničke aplikacije i servisa roditeljske kontrole prikazan je na *Slika14*.



Slika14 Dijagram klasa servisa za slanje i prijem elektronske pošte

4.1.1 Modul aktivnosti

Ovaj modul predstavlja Android aktivnost u kojoj se Android korisnička aplikacija pokreće. Kada se prvi put pokrene korisnička aplikacija, otvaraju se korisnička podešavanja gde korisnik unosi sledeće parametere:

- adresu elektronske pošte primaoca,
- adresu elektronske pošte pošiljaoca,
- maksimalnu vrednost dozvoljene brzine kretanja vozila.

Takođe iz ovog modula se pokreće i zaustavlja servis za roditeljsku kontrolu.

4.1.2 Modul za slanje elektronske pošte

Na osnovu izabranog naloga elektronske pošte u podešavanjima korisničke aplikacije vrši se autentifikacija. Zatim se priprema sesija za otvaranje konekcije sa SMTP serverom. Ako je autentifikacija izvršena uspešno, i ako je dobijen ključ za pristup nalogu elektronske pošte bez unošenja šifre, izabran nalog je spreman za slanje elektronske poruke. Funkcija za slanje elektronske poruke će biti prozvana kada se desi prekoračenje brzine. Treba napomenuti da se

brzina dobavlja u programskoj niti, posredstvom API funkcija CarInfoService-a, koje služe za dobavljanje brzine vozila od simulatora događaja u vozilu ili od samog vozila putem Bluetooth konekcije. Kratak opis funkcija za slanje elektronske pošte dat je u *Tabela 3*.

```
public SendingEmail(Context ctx)
```

Konstruktor klase SendingEmail ima zadatak da podesi adresu elektronske pošte iz korisničkog podešavanja.

```
public void getToken()
```

Pozivom ove funkcija dobija se ključ za pristup Gmail nalogu i povezivanje na server za slanje elektronske pošte.

```
private SMTPTransport connectToSmtp(String host, int port, String userEmail, String oauthToken)
```

Funkcija koja omogućava povezivanje na SMTP server pomoću: imena domena(engl. *Host*) SMTP servera, ulaza (engl. *Port*) za SMTP server, Gmail naloga i dobijenog ključa bezbednosnih sertifikata. SMTP server je zadužen za slanje elektronske pošte.

```
public synchronized void sendMail(String subject, String body, String user, String oauthToken, String recipients)
```

Funkcija zadužena za slanje elektronske pošte.

Tabela 3 Funkcije modula za slanje elektronske pošte

4.1.3 Modul za prijem elektronske pošte

Ovaj modul, kao i modul za slanje elektronske pošte treba na osnovu unete adrese elektronske pošte u podešavanjima korisničke aplikacije, da izvrši autentifikaciju Gmail naloga i spremi sesiju za otvaranje konekcije sa IMAP serverom. Ukoliko je autentifikacija uspešno izvršena i dobijen ključ sa bezbednosnim sertifikatom moguće je pristupiti željenom Gmail nalogu elektronske pošte, otvaranje fascikle pristiglih poruka i praćenje pristizanja novih elektronskih poruka. Treba napomenuti da sa promenom naloga elektronske pošte iz podešavanja korisničke aplikacije određeni Gmail nalog prolazi kroz istu proceduru dobijanja bezbednosnih sertifikata kao i otvaranja IMAP servera i fascikle pristiglih poruka. Pošto je ovaj modul pozvan u lokalnom servisu za prijem i slanje elektronske pošte, potrebno je proslediti promene Gmail naloga iz aktivnosti korisničkih podešavanja. Iz ovih razloga je korišćen modul Broadcast Receiver, koji se u aktivnosti korisničke aplikacije pokreće ukoliko se desila promena Gmail naloga i šalje akciju da se promena desila. U lokalnom servisu za roditeljsku kontrolu primi se ta akcija, odnosno registruje promena koja će prouzrokovati pristup fascikli pristiglih elektronskih poruka novo odabranog gmail naloga. Treba napomenuti da je za pristizanje novih poruka iskorišten osluškivač (engl. *Listener*) na nove poruke iz JavaMail API-ja. U prvoj verziji rešenja je korišteno čitanje čitave fascikle pristiglih poruka, sto je trošilo mnogo resursa, a u novijoj

verziji rešenja se samo proveravaju novo pristigle poruke i pretražuju po određenom kriterijumu. Kriterijumi su adresa elektronske pošte pošiljaoca i naslov poruke (engl. *Subject*). Ukoliko je elektronska poruka o prekoračenju brzine poslata roditelju, roditelj će nakon pregleda poruke odgovoriti na istu elektronsku adresu sa koga je i poslata poruka o prekoračenju brzine. U fasciklu pristiglih poruka istog Gmail naloga sa koga je poslata poruka će stići nova poruka, osluškivač na novo pristigle poruke će izvršiti obradu, i ukoliko je gore navedeni kriterijum zadovoljen, na ekranu računara u vozilu će se ispisati sadržaj poruke koju je roditelj poslao. Spisak značajnih funkcija ovog modula kao i kratak opis dat je u *Tabela 4*.

<pre>public ReceivedEmail(Context ctx, MailReceivedInterface listener)</pre>
--

Konstruktor klase ReceivedEmail ima zadatak da inicijalno podesi adresu elektronske pošte iz korisničkog podešavanja. Drugi parametar konstruktora predstavlja spregu koja služi za prikazavanje upozoravajućeg dijaloga iz servisa za roditeljsku kontrolu. On se koristi za preuzimanje sadržaja poruke od roditelja iz modula za prijem elektronske pošte.

<pre>void CheckEmails()</pre>

Funkcija za vršenje autentifikacijeGmail naloga,povezivanje na IMAP server i otvaranje fascikle pristiglih poruka određenog gmail naloga.

<pre>public IMAPStore connectToImap(String host, int port, String userEmail, String oauthToken, boolean debug)</pre>
--

Funkcija koja se povezuje na IMAP server pomoću imena domena IMAP servera, ulaza za IMAP server,gmail naloga i ključa za dobavljanje bezbednosnih sertifikata.IMAP server je zadužen za primanje poruka.

<pre>addMessageCountListener(new MessageCountAdapter())</pre>

Ova funkcija iz JavaMail API-ja ima zadatak da otvori fasciklu određenog Gmail naloga i da osluškuje dolazak novih elektronskih poruka. Kada dođe nova poruka ,osluškivač je obrađuje po određenom kriterijumu.

Tabela 4 Funkcije modula za prijem elektronske pošte

4.1.4 Programska nit za slanje zahteva o stanju brzine

Za potrebe servisa za roditeljsku kontrolu potrebno je od simulatora događaja u vozilu ili samog vozila zahtevati brzinu kretanja, kako bi servis mogao da šalje elektronsku poštu u slučaju prekoračenja brzine. Programska nit koristi API funkcije servisa za prihvatanje informacija o vozilu (CarInfoService). CarInfoService šalje zahtev za brzinu vozila simulatoru događaja u vozilu ili samom vozilu, a simulator/vozilo mu dostavlja traženu brzinu. Kada simulator vratí brzinu, CarInfoService je prihvati i prosleđuje je servisu za roditeljsku kontrolu. Konkretno modulu za proveravanje brzine u okviru servisa roditeljske kontrole. Na ovaj način servis roditeljske kontrole može pravilno da funkcioniše. U programskoj niti se definiše učestanost traženja informacija o brzini vozila od CarInfoService-a. Ta procedura se konstantno ponavlja

sve dok se servis za roditeljsku kontrolu ne zaustavi. API funkcije CarInfoService-a koje su korišćene u programskoj niti servisa roditeljske kontrole za slanje i prijem zahteva o stanju parametra brzine date su u *Tabela5*.

<code>int requestSpeed(IcarInfoSpeedNotification speedNotification)</code>	Funkcija iz API-ja CarInfoService-a koja je zadužena za slanje zahtevao informaciju brzini vozila koju treba da proslediti servisu za slanje i prijem elektronske pošte.
--	--

<code>public void onSpeedReceived(int speed)</code>	Povratna (engl. <i>Callback</i>)funkcija iz API-ja CarInfoService-a za prihvatanje informacija o brzini vozila.
---	--

Tabela5 API funkcije iz servisa za prihvatanje informacija(CarInfoService) sa ECU/simulator

4.1.5 Modul servisa za slanje i prijem elektronske pošte

Svi prethodno navedeni moduli pozivaju se u okviru modula servisa za slanje, prijem elektronske pošte i provere brzine (Servis roditeljske kontrole). Ovaj modul realizuje logiku slanja elektronske pošte. Pre svega interval slanja elektronske poruke. Brzina se dobavlja od vozila/simulatora na svakih 5 sekundi tako da postoji potencijalna mogućnost da se nova elektronska poruka šalje na svakih 5 sekundi. Da bi se ovaj scenario izbegao korišćen je algoritam skeniranja brzine na prva 3 minuta od kada je servis za roditeljsku kontrolu pokrenut. Ako je srednja vrednost te brzine veća od ograničenja brzine koje je definisano u podešavanjima korisničke aplikacije, elektronska poruka će se poslati i započeće novo praćenje brzine. Naredni ciklus praćenja brzine traje 30 minuta. U slučaju da se desi da je brzina manja od ograničenja, proces se ponovo ponavlja. Prilikom slanja elektronske poruke na 3 minuta, programska nit koja zahteva brzinu sa simulatora događaja u vozilu više ne šalje elektronsku poruku na 3 minuta, nego na svakih 30 minuta. Kada se elektronska poruka pošalje na 30 minuta u poruci je ceo izveštaj formiranja brzina kao i vreme njihovog nastanka. Brzine kao i vreme njihovog formiranja predstavljaju polja klase zadužene za kompletan izveštaj praćenja brzine vozila u intervalu od 30 minuta. Nakon isteka 30 minuta, ako postoji barem jedna srednja brzina veća od ograničenja, poslaće se elektronska poruka o prekoračenju brzine. Kada roditelj vidi poruku o prekoračenju i odgovori na nju, modul koji je zadužen za primanje elektronskih poruka primiće poruku i proveriće je po sledećem kriterijumu: elektronskoj adresi pošiljaoca i naslovu elektronske poruke. Ako je navedeni kriterijum zadovoljen na ekranu računara u vozilu će se prikazati poruka od roditelja u vidu upozoravajućeg dijaloga. Dok vozač ne pritisne dugme za uklanjanje poruke sa ekrana, na ekranu će upozoravajući dijalog biti prikazan.

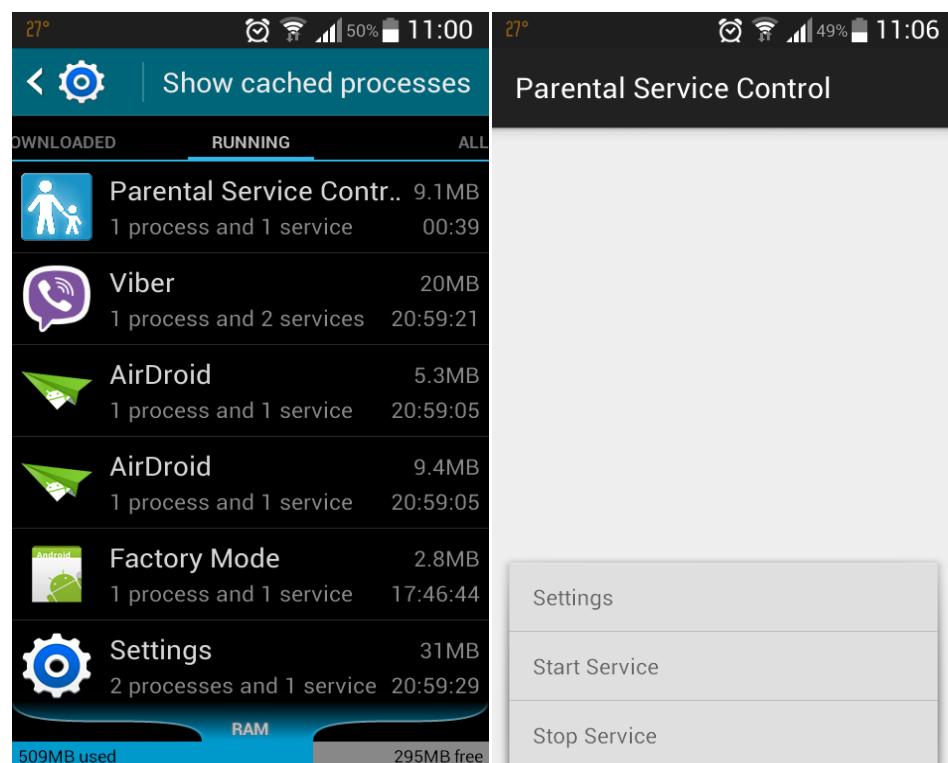
5. Rezultati i verifikacija

Da bi se potvrdila stabilnost i ispravnost korisničke aplikacije u kojoj je integriran servis za roditeljsku kontrolu, ona mora biti podvragnuta različitim vrstama provera. U cilju proveravanja korisničke aplikacije i servisa izvršeno je nekoliko scenarija kako bi se verifikovala ispravnost servisa za prijem, slanje elektronske pošte i dobavljanja brzine (Servis roditeljske kontrole).

- Scenario 1: Ponašanje aplikacije u slučaju promene postavki iz opcija korisničkih podešavanja.
- Scenario 2: Proces pristupa određenom elektronskom nalogu, povezivanje na server za slanje elektronskih poruka, kao i samo slanje elektronske poruke o prekoračenju brzine. Ovo slanje se vrši po unapred definisanoj elektronskoj adresi primaoca izabranoj u korisničkom podešavanju sa određenim predmetom poruke kao i tekstrom poruke. Ova elektronska pošta će biti poslata u slučaju prekoračenja brzine vozila.
- Scenario 3: Slanje odgovora na istu elektronsku adresu sa koje je poslata poruka o prekoračenju brzine kao i prikazivanje upozoravajućeg dijaloga na ekranu računaru u vozilu.

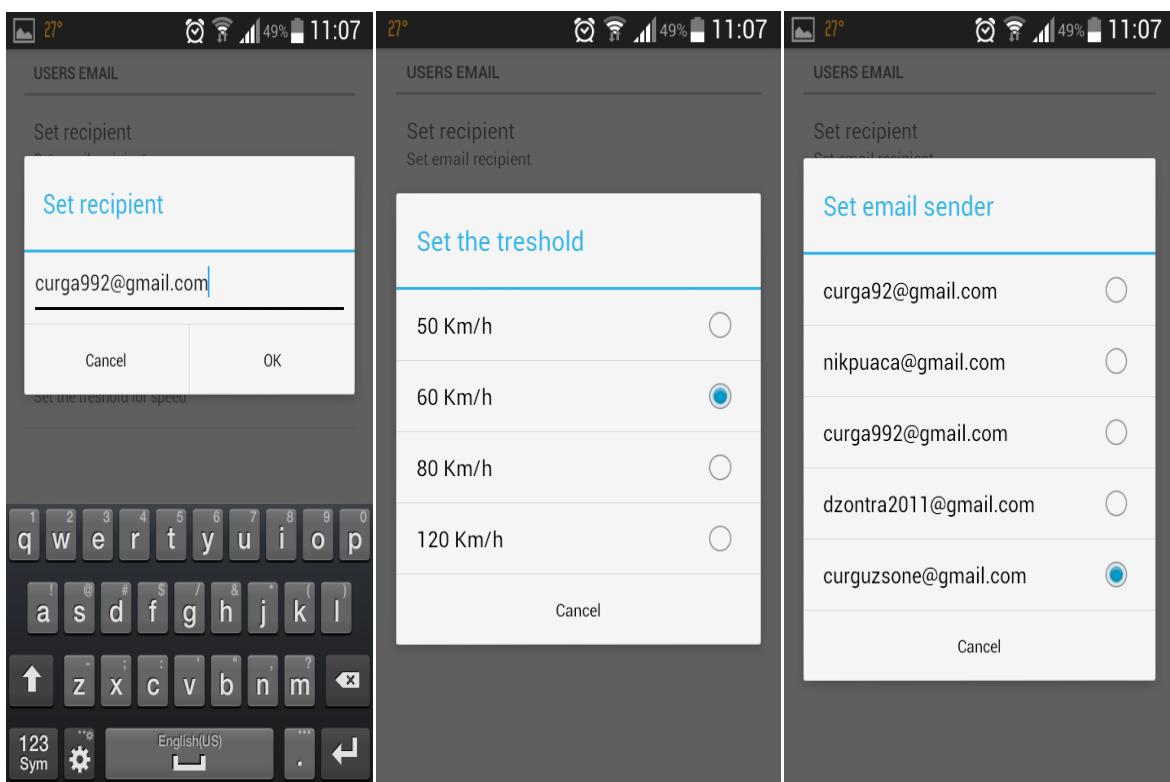
Svi prethodno navedeni scenariji su uspešno izvršeni i na taj način je potvrđena ispravnost funkcionisanja servisa roditeljske kontrole.

Android servis za roditeljsku kontrolu je proveren na uređajima koji koriste Android operativni sistem. Ispitivanja validnosti rada servisa za roditeljsku kontrolu i korisničke aplikacije vršena su na Samsung Galaxy SIII mini mobilnom uređaju, kao i na tabletu Google (HTC) Nexus 9, sa verzijom 5.0 Android operativnog sistema. Konačan izgled korisničke aplikacije i servisa za roditeljsku kontrolu na telefonu Samsung Galaxy SIII mini prikazani su *Slika 15.*



Slika15 Pokretanje servisa i prikazivanje servisa u rukovaocu Android aplikacijama

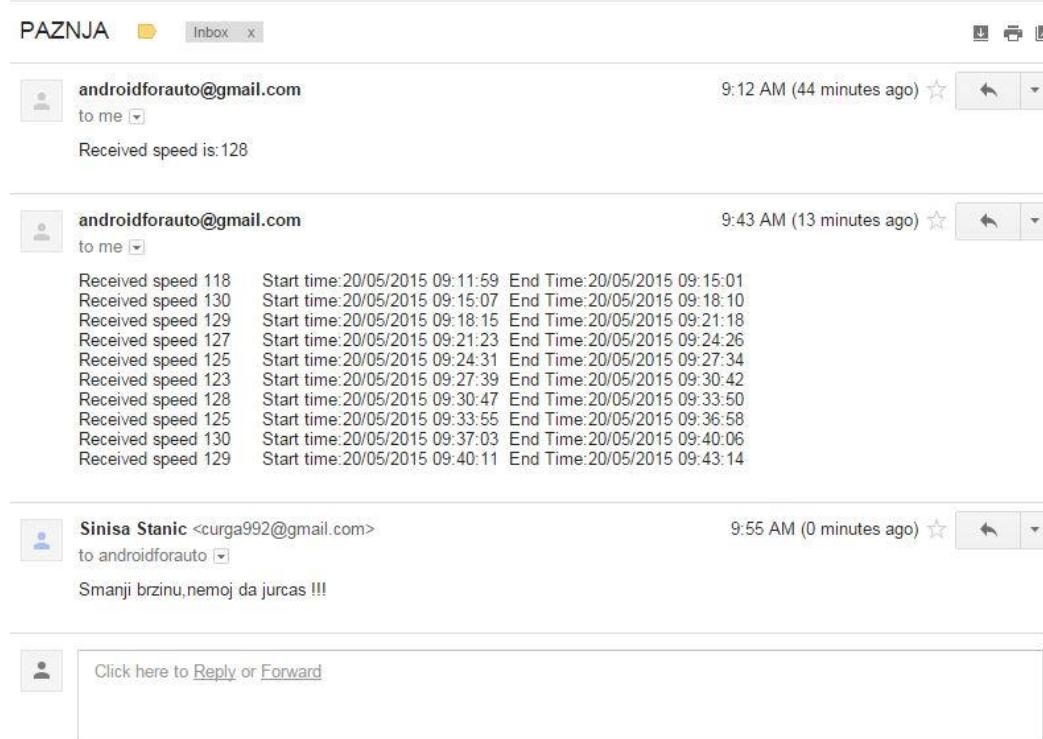
Na Slika16 prikazana su korisnička podešavanja (engl. *Settings*) u okviru korisničke aplikacije u kojoj korisnik (roditelj) može da promeni adresu elektronske pošte primaoca, pošiljaoca i ograničenje brzine. Treba napomenuti da za slanje elektronske poruke korisnik bira jednu adresu iz liste dostupnih elektronskih adresa.



Slika16 Korisničke opcije korisničke aplikacije

Na *Slika17* prikazan je prijem elektronske poruke o prekoračenju brzine prvi put kada se desilo prekoračenje brzine, kao i prikaz poruke nakon isteka intervala od 30 minuta, koja sadrži celokupan izveštaj brzina tokom 30 minuta kretanja vozila.

Na *Slika18* je prikazan odgovor roditelja koji se prikazuje u upozoravajućem dijalogu na ekranu računara u vozilu.



Slika17 Poslata elektronska pošta o prekoračenju brzine



Slika18 Prikaz poruke od roditelja na ekranu računara u vozilu

6.Zaključak

Prilikom izrade ovog zadatka korišćen je Java programski jezik, JavaMail aplikaciona sprega za slanje i prijem elektronske pošte, kao i protokoli pomoću kojih se dobijaju bezbednosni sertifikati za pristup Gmail nalozima. Realizacija rešenja je implementirana u razvojnom okruženju Android studija.

Cilj rada je razvoj Android servisa za roditeljsku kontrolu sa korisničkom aplikacijom. Zadatak servisa je da obavesti vlasnika vozila o eventualnom prekoračenju brzine dok njegova deca upravlju vozilom. Servis za roditeljsku kontrolu poseduje korisnička podešavanja u kojima je moguće izabrati željenu: elektronsku adresu pošiljaoca, kao i primaoca i željeno ograničenje brzine. Zbog korisničkih podešavanja servis roditeljske kontrole je realizovan kroz korisničku aplikaciju (lokalni servis), i zbog toga postoji potreba za grafičko korisničkom spregom. Takođe korisnička aplikacija omogućuje pokretanje i zaustavljanje servisa.

U cilju unapređenja postojećeg rešenja treba omogućiti da se servis za roditeljsku kontrolu pokrene sa pokretanjem operativnog sistema uređaja na kom se izvršava. Takođe u cilju daljeg unapređenja postojećeg rešenja treba realizovati praćenje kretanja vozila na mapi (engl. Geo-fencing). Ovo praćenje kretanja vozila na mapi treba da omogući da roditelj u okviru korisničkih podešavanja može da obeleži oblast na mapi po kojoj će se kretati automobil kojim upravlja njihovo dete. Ako automobil kojim upravlja njihovo dete odstupi od zadatih parametara, poslaće se elektronska pošta o napuštanju obeležene oblasti na mapi.

Jedan od nedostataka ovog rešenja roditeljske kontrole je što dati servis ne može da identificuje vozača. Može se desiti da roditelj nakon dužeg vremena odgovori na pristiglu poruku o prekoračenju brzine, a da se pritom promeni vozač. U tom slučaju poruka u računaru u vozilu će biti prikazana pogrešnom vozaču. Da bi se izbegli ovakvi slučajevi potrebno je razviti mehanizam identifikacije vozača u vozilu. Identifikacija bi se vršila svaki put pri pokretanju

operativnog sistema u računaru u vozilu (pri svakom pokretanju vozila). Vozač bi se na primer, mogao identifikovati otiskom prsta ili nekom drugom sličnom metodom. Nakon uspešnog završetka procedure identifikacije, servis roditeljske kontrole bi imao informaciju o vozaču i bilo bi moguće prikazati poruke koje su namenjene isključivo njemu.

7.Literatura

- [1] Professional Android Application Development, Reto Meier, May 1, 2008
- [2] Professional Android 2 Application Development, Reto Meier, Mar 1, 2010
- [3] Android develepor, Android 5.0 dostupno na:
<http://developer.android.com/about/versions/lollipop.html>, učitano 20.05.2015
- [4] Andoid services, dostupno na:
<http://developer.android.com/guide/components/services.html>, učitano 15.05.2015
- [5] OBD II Streamer family, dostupno na:
http://www.bb-elec.com/Products/Manuals/Intelligent-OBDII-Gateway-Command-and-Response_251.pdf, učitano 09.05.2015
- [6] ELM327 OBD to RS232 Interpreter, dostupno na:
<http://www.elmelectronics.com/DSheets/ELM327DSF.pdf>, učitano 11.05.2015
- [7] OBD Modes, dostupno na: <http://www.outilsobdfacile.com/obd-mode-pid.php>,
učitano 09.05.2015
- [8] Postojeća rešenja dostupno na:
www.javatpoint.com/example-of-sending-email-using-java-mail-api,učitano
13.05.2015
www.javatpoint.com/example-of-receiving-email-using-java-mail-api,
učitano 13.05.2015
- [9] JavaMail Api dostupno na:
<https://javamail.java.net/nonav/docs/api/>, učitano 13.05.2015