



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ



Михајло Маринковић

Графичка корисничка спрега за приказ података на контролној табли возила заснована на Андроид платформи

ДИПЛОМСКИ РАД
- Основне академске студије -

Нови Сад, 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:	Михајло Маринковић
Ментор, МН:	др Небојша Пјевалица
Наслов рада, НР:	Графичка корисничка спрега за приказ података на контролној табли возила заснована на Андроид платформи
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/страна/ цитата/табела/слика/графика/прилога)	7/35/0/3/23/0/0
Научна област, НО:	Електротехника и рачунарство
Научна дисциплина, НД:	Рачунарска техника
Предметна одредница/Кључне речи, ПО:	Контролна табла возила, кориснички спрега
УДК	
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, ВН:	
Извод, ИЗ:	У овом раду је представљен један од начина реализације графичке корисничке спреге за приказ података на контролној табли возила. Обухваћене су информације о брзини возила, броју обртаја у мотору, нивоу горива у резервоару, нивоу уља, температури мотора као и позицији папучице на гасу. Имплементирани су тродимензионални ефекти након корисникове интеракције са одређеним индикатором.
Датум прихватања теме, ДП:	
Датум одбране, ДО:	
Чланови комисије, КО:	Председник: др Јелена Ковачевић
	Члан: др Иван Каштелан
	Члан, ментор: др Небојша Пјевалица
	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 :	Mihajlo Marinković
Mentor, MN :	Nebojša Pjevalica, PhD
Title, TI :	Graphical user interface for data display on the vehicle dashboard based on Android platform.
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>	7/35/0/3/23/0/0
Scientific field, SF :	Electrical Engineering
Scientific discipline, SD :	Computer Engineering, Engineering of Computer Based Systems
Subject/Key words, S/KW :	Vehicle dashboard, user interface
UC	
Holding data, HD :	The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, N :	
Abstract, AB :	This paper presents one of the ways of realization of graphical user interface for data display on the vehicle dashboard. Information that are included are vehicle speed, engine revs (rpm), fuel level, oil level, engine temperature and the position of the throttle pedal. 3D effects are realized upon the user's interaction with a specific indicator.
Accepted by the Scientific Board on, ASB :	
Defended on, DE :	
Defended Board, DB :	President: Jelena Kovačević, PhD
	Member: Ivan Kaštelan, PhD
	Member, Mentor: Nebojša Pjevalica, PhD
	Mentor's sign

Zahvalnost

Zahvaljujem se institutu RT-RK na pruženoj mogućnosti za realizaciju ovog rada.

Takođe, zahvaljujem se mentoru dr Nebojši Pjevalici, stručnim saradnicima Tomislavu Maruni, Branimiru Kovačeviću, Marku Kovačeviću kao i celokupnom timu *Android4Auto* na stručnoj pomoći i savetima prilikom izrade ovog rada.

Na kraju, zahvaljujem se svojoj porodici na pruženoj podršci tokom mog školovanja.

SADRŽAJ

1. Uvod.....	1
2. Teorijske osnove	3
2.1 Android.....	3
2.2 Automobilska industrija	6
2.3 Kontrolna jedinica motora.....	7
2.4 OBD II.....	8
2.4.1 Prolaz za komunikaciju.....	8
2.4.2 Protokoli za komunikaciju	9
2.4.3 OBD II identifikatori poruka	9
2.4.4 OBD II ELM327 Bluetooth adapter	10
3. Koncept rešenja.....	11
3.1 Car Info Service	11
3.2 Modul za iscrtavanje indikatora na ekranu	13
3.3 Flip View Indicator	14
3.4 3D Efekat.....	15
4. Programsko rešenje.....	16
4.1 Paketi: Comfort, Sport, Eco	17
4.2 Paket Flipping	22
4.3 Paket Effect	23
5. Rezultati	24
6. Zaključak	26
7. Literatura.....	27

SPISAK SLIKA

Slika 1. Arhitektura Android platforme	4
Slika 2. Prikaz digitalne instrument table	6
Slika 3. Kontrolna jedinica motora	7
Slika 4. Prolaz za komunikaciju	8
Slika 5. OBD adapterska kutija	9
Slika 6. OBD II Adapter	10
Slika 7. Arhitektura rešenja	11
Slika 8. MSC dijagram razmene podataka	12
Slika 9. Prikaz izbora načina prikazivanja informacija	13
Slika 10. Skup indikatora	13
Slika 11. Šematski prikaz rotacije	14
Slika 12. Prikaz promene indikatora	14
Slika 13. Šematski prikaz 3D efekta	15
Slika 14. Promena prikaza informacija nakon pritiska na centralni indikator	15
Slika 15. Prikaz animacija <i>zoomin</i> , <i>leftAlignment</i> I <i>rightAlignment</i>	17
Slika 16. Prikaz animacija <i>rightSpeedMove</i> i <i>moveDownImg</i>	18
Slika 17. Prikaz animacije <i>moveUpImg</i>	18
Slika 18. Prikaz kružnog progresa	20
Slika 19. Realizacija kružnog progresa za indikator nivoa goriva u rezervoaru	20
Slika 20. Rotacija strelice	21
Slika 21. Prikaz rezultata funkcije <i>updateBitmap()</i>	21
Slika 22. Prikaz promene indikatora	22
Slika 23. Primer 3D Efekta	23

SPISAK TABELA

Tabela 1. Način rada	10
Tabela 2. Sadržaj osnovnih paketa programskog rešenja	16
Tabela 3. Opis animacija u <i>XML-u</i>	19

SKRAĆENICE

ABS – *Anti-lock Braking System*, sistem protiv blokiranja točkova

Android SDK – *Software Development Kit*, Android razvojni paket

AOT - *Ahead-of time*

API – *Application programming interface*, aplikativna programska sprega

ARM – Porodica arhitektura skupa instrukcija za procesore zasnovane na RISC arhitekturi

ART - *Android Runtime*

CAN – *Car Area Network / Controller Area Network*, računarska mreža koja povezuje sve podsisteme u automobilu

ECU – *Engine Control Unit*, kontrolna jedinica motora

ESP – *Electronic Stability Program*, sistem protiv proklizavanja

GM – *General Motors*, Grupacija proizvođača automobila (Opel, Ševrolet, Bjuik, Kadilak, GMC, Sab automobili)

GUI – *Graphical User Interface*, grafička korisnička sprega

JIT – *Just in time*

JNI – *Java Native Interface*, programskim okvir koji omogućava komunikaciju, spregu, između koda napisanog u programskom jeziku Java i koda napisanog u drugim jezicima, npr. C, C++ i assembler

LCD - *Liquid Crystal Display*, ekrani zasnovani na tehnologiji tečnih kristala

MSC – *Message sequence chart*, dijagram prenosa podataka u sistemu

OBD II – *On-board diagnostics*, sistem za dijagnostiku kvara na vozilu uz pomoć računara

OpenGL - *Open Graphics Library*, višeplatformska programska sprega za pisanje programa koji rade sa dvodimenzionalnom i trodimenzionalnom računarskom grafikom.

XML - *Extensible Markup Language*, jezik koji opisuje izgled aktivnosti

1. Uvod

U ovom radu je predstavljen jedan od načina realizacije grafičke korisničke sprege, aplikacije (engl. *Graphical User Interface - GUI*) koja prikazuje informacije od strane kontrolne jedinice motora (engl. *Engine Control Unit*) putem OBD II (engl. *On-board diagnostics*) protokola. Informacije koje su obuhvaćene: brzina vozila, broj obrtaja u minutu, nivo goriva u rezervoaru, nivo ulja, temperatura u motoru i pozicija papučice na gasu. Rešenje je realizovano za platforme sa Android programskom podrškom [1] verzije 4.1 (engl. *Jelly Bean*) i novije. Ispitan je i verifikovan rad na uređajima Lenovo A5500F i HTC Nexus 9, kao i na *Automotive* platformi zasnovanoj na ARM fizičkoj arhitekturi sa Android operativnim sistemom. Navedeno rešenje je prilagođeno ciljnim platformama. *GUI* je optimizovan kako bi trošio što manje resursa, što utiče na smanjenje potrošnje električne energije, zagrevanje samog uređaja, kao i na kvalitetnu i pouzdanu uslugu.

Prilikom projektovanja grafičke korisničke sprege akcenat je stavljen na naredne elemente:

- Sam uređaj treba da oponaša prikaz informacija kontrolne table.
- Interakcija sa korisnikom treba da bude što prirodnija, a ujedno i interesantna.
- Upotreba novog radnog okruženja (Android Studio), kao i novih tehnika dostupnih sa Android programskom podrškom kako bi se dobilo na celokupnom korisničkom utisku.
- Realizacija trodimenzionalnih efekata (engl. *Three-dimensional Effects – 3D Effects*) korišćenjem standardnog dvodimenzionalnog radnog okruženja.

Rad je sačinjen od nekoliko celina:

- Teorijske osnove – kratak opis konkretnog problema, uvid u automobilsku industriju kao i novije tehnologije u ovoj oblasti, kontrolna jedinica motora (engl. *Engine Control Unit*), OBD II protokol i opis Android operativnog sistema.
- Koncept rešenja – generalna slika rešenja i opis samog koncepta.

- Programsko rešenje – konkretizacija koncepta i detaljan opis realizacije.
- Rezultati – pregled postignutih rezultata.
- Zaključak – predstavljen značaj postignutih rezultata i mogućnosti daljeg unapređenja rada.
- Literatura – predstavlja spisak korišćene literature prilikom izrade rada.

2. Teorijske osnove

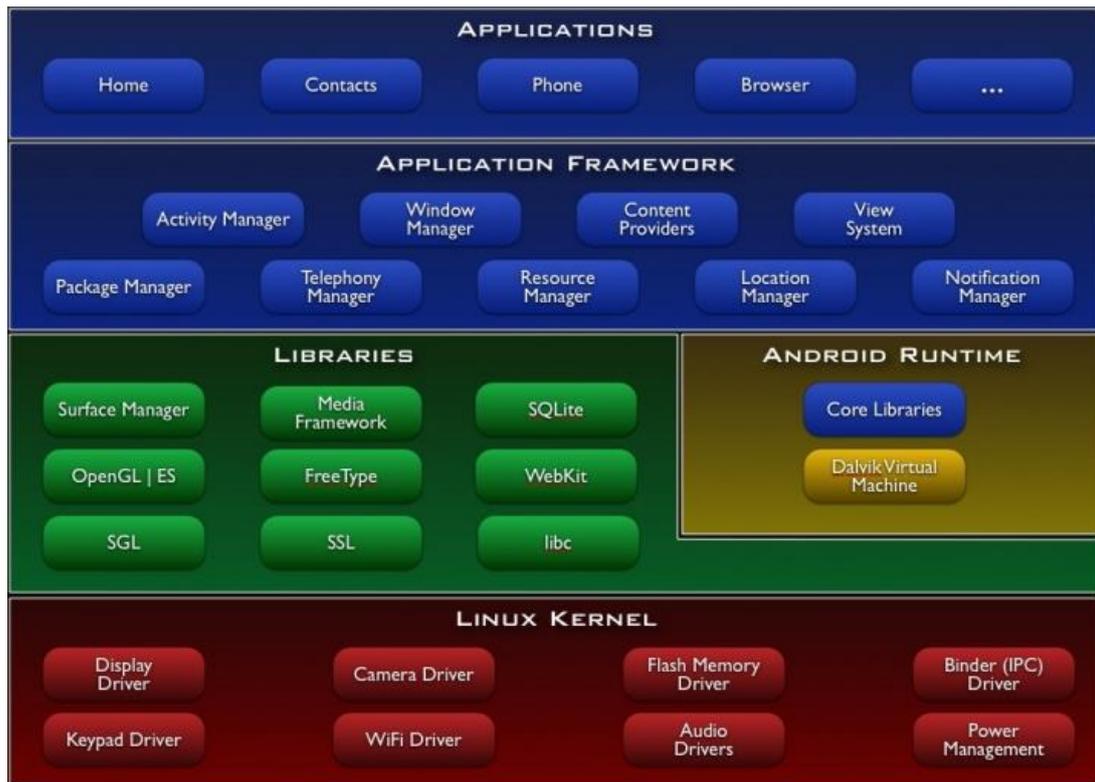
U datom poglavlju su opisane teorijske osnove na kojima je rad zasnovan. Izložene su osnovne informacije o samoj Android platformi i njenim aplikacijama. Obuhvaćen je opis komunikacionog protokola za realizaciju ovog rešenja, kao i opis kontrolne jedinice motora. Opisano je trenutno stanje u automobilskoj industriji i dat je prikaz novih tehnologija koje se primenjuju.

2.1 Android

Android je operativni sistem zasnovan na Linux jezgru, prvenstveno namenjen za mobilne telefone i tablet računare. Ono što razlikuje ovaj sistem od drugih, jeste da je Android sistem otvorenog koda što omogućava korisniku određeni stepen slobode u prilagođavanju samog sistema svojim potrebama. Prvobitno je Android platforma razvijena za ARM procesorsku arhitekturu, a danas nalazi primenu na velikom broju različitih uređaja koje karakterišu različite fizičke arhitekture. Android aplikacije su nezavisne od same fizičke arhitekture, tj. mogu se izvršavati na različitim platformama. I pored toga što je Android trenutno najpopularnija i najrasprostranjenija platforma za mobilne uređaje pre svega, danas nalazi primenu kod digitalnih televizijskih prijemnika, igračkih konzola, digitalnih kamera, druge potrošačke elektronike i na kraju u vozilima što je i tema ovog rada.

Na Slika 1 prikazana je arhitektura Android platforme, tj. Android stek koji se može podeliti na nekoliko slojeva:

- Jezgro operativnog sistema (engl. *Linux Kernel*)
- Srednji sloj (engl. *Middleware*)
- Aplikativni sloj



Slika 1. Arhitektura Android platforme

Android operativni sistem se oslanja na Linux 2.6 jezgro. Linux jezgro je zaduženo za komunikaciju sa samom fizičkom arhitekturom. Ovaj sloj se takođe ponaša kao apstrakcioni sloj između fizičke arhitekture i ostatka programske podrške. Linux jezgro je zaduženo i za upravljanje memorijom, procesima, mrežnim operacijama i sa bezbednošću sistema.

Naredni sloj obuhvata nativni skup biblioteka, C/C++ biblioteka. Funkcionalnosti koje pružaju nativne biblioteke izložene su programeru kroz Android aplikativni okvir (engl. *Android Application Framework*).

U nastavku je dat pregled najčešće korišćenih biblioteka:

- Sistemske C biblioteke – implementacija standardne C sistemske biblioteke, koja je prilagođena za namenske Linux zasnovane uređaje.
- Biblioteke za rukovanje audio/video podacima – podrška za reprodukciju i snimanje raznih audio i video formata. Neki od podržanih formata: MPEG4, X.264, MP3, AAC, PNG, JPG.
- Biblioteke Web pretraživača (engl. *LibWebCore*) – web pretraživač.
- SGL – predstavlja osnovnu dvodimenzionalnu (2D) grafičku podršku.
- 3D biblioteke – u osnovi se zasnivaju na OpenGL ES aplikativnoj programskoj sprezi.

- SQLite – alat namenjen za relacione baze podataka.

Pored nativnih biblioteka Android sadrži i skup osnovnih biblioteka koje su realizovane u Java programskom jeziku (*java.io, java.lang, java.util, java.math*).

Najviši sloj programske podrške predstavlja aplikativni sloj, koji se sastoji od aplikativnog okvira (engl. *Application Framework*) i samih aplikacija. Aplikativni okvir sadrži programe koji upravljaju resursima, u toku izvršavanja aplikacije koja koristi ove programe.

Kao što je prethodno navedeno u okviru ovog sloja nalaze se Android aplikacije, koje se izvršavaju na Dalvik virtualnoj mašini. Dalvik virtualna mašina predstavlja određen tip Java virtualne mašine koja je prilagođena i optimizovana kako bi trošila što manje procesorskog vremena i memorije. Takođe, na datoj virtualnoj mašini izvršavaju se datoteke (.dex) formata, a ne (.class) formata. Virtualna mašina pokreće klase koje su prevedene Java programskim prevodiocem u dex format pomoću “dx” alata. Svaki proces ima svoju virtualnu mašinu, i po pravilu svaka aplikacija se pokreće u nezavisnom procesu. Android uništava proces kada nije više potreban ili kada je potrebno osloboditi memoriju za druge aplikacije. Od Android verzije 5.0 “Lollipop”, Dalvik je zamenjen sa ART (engl. *Android Runtime*). Za razliku od Dalvik virtualne mašine, koja prevodi aplikaciju pri svakom pokretanju (engl. *Just in time, JIT*), ART to radi po sistemu pre vremena (engl. *Ahead-of time, AOT*). Pri postavljanju aplikacija (engl. *Install*), vrši se automatsko pevođenje. Ovo zauzima više memorijskog mesta na samom uređaju i zahteva duže postavljanje u poređenju sa Dalvik-om. Međutim, aplikacije se pokreću mnogo brže i sam rad aplikacije je mnogo optimizovaniji kada se koristi ART u poređenju sa Dalvik-om.

Android aplikacije se pišu u programskom jeziku Java. Napisan kod se prevodi pomoću Android SDK (engl. *Software Development Kit*) alata sa svim uključenim podacima i potrebnim datotekama u jedinstven Android paket. Dati Android paket predstavlja arhivsku datoteku sa ekstenzijom (.apk). Datoteka se prosleđuje na određeni Android uređaj, gde se postavlja (engl. *Install*) i omogućava se pristup aplikaciji od strane samih korisnika. Aplikacije napisane u programskom jeziku Java imaju isključivo pristup samo aplikativnom sloju Android arhitekture. Kako bi se rešio ovaj nedostatak povezivanja Android aplikacija sa određenim kodom koji nije napisan u programskom jeziku Java, uveden je JNI (engl. *Java Nativ Interface*) modul. JNI predstavlja programskim okvir koji omogućava komunikaciju, spregu, između koda napisanog u programskom jeziku Java i koda napisanog u drugim jezicima, npr. C, C++ i assembler. Na ovaj način je omogućen pristup sistemskom prostoru iz aplikativnog sloja.

2.2 Automobilska industrija

Ako se za neku granu industrije može reći da prati savremene trendove i da ulaže u nove tehnologije, onda je to sigurno automobilska industrija. Modernizacija automobilske industrije, doprinela je razvijanju tehnologija koje pružaju veću sigurnost vozaču, udobnost i efikasnost u saobraćaju. Ono čemu danas industrija teži, jeste da omogući korisniku potpuni ugođaj u vožnji. Računari se odavno koriste za projektovanje i proizvodnju automobila, sve više postaju neophodni deo automobila, jednako važan kao i sam motor, ali odnedavno pronalaze svoje mesto uz vozača, pružajući mu veću sigurnost, informacije i zabavu.

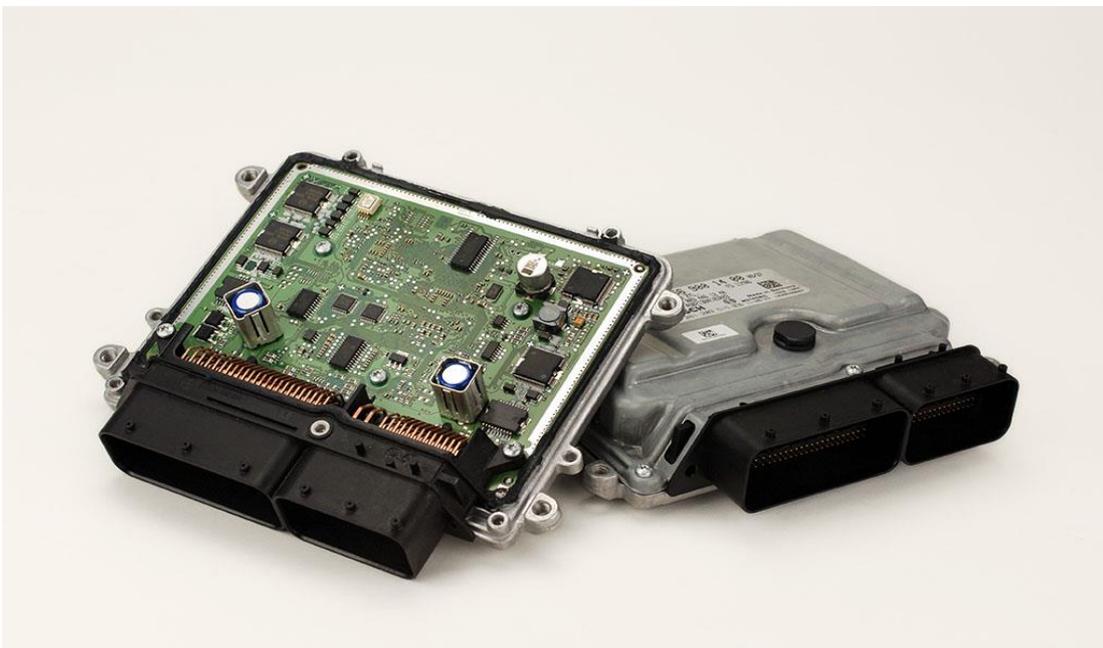
Tako je npr. jedan od proizvođača razvio potpuno novu platformu koja zamenjuje standardnu instrument tablu. Umesto uobičajnih indikatora, nalazi se LCD ekran od 12.3 inča koji je pozicioniran ispred samog vozača. Takođe, novi izgled je doneo nove promene u unutrašnjosti automobila, pa je tako i tradicionalna kontrolna tabla zamenjena ovim sistemom. Sistem se kontroliše pritiskom kontrola na volanu ili unapređenog rotacionog kontrolera. Prikazuje sve bitne informacije kao i bitna podešavanja automobila. Na slici ispod prikazana je digitalna instrument tabla.



Slika 2. Prikaz digitalne instrument table

2.3 Kontrolna jedinica motora

Automobili novije generacije poseduju sistem koji je neophodan za prikupljanje informacija o radu motora, kao i priključak za dijagnostiku. S obzirom da proizvođači automobila imaju slične oznake za ovaj sistem, usvojena je oznaka na međunarodnom nivo (engl. *Engine Control Unit - ECU*). ECU je mikroprocesor [2] koji upravlja pripremom gorivne smeše, paljenjem smeše, nadgledanjem sastava izduvnih gasova, takođe obavlja brojne druge funkcije nad sistemima kao što su: sistem protiv blokiranja točkova (engl. *Anti-lock Braking System – ABS*), sistem protiv proklizavanja (engl. *Electronic Stability Program – ESP*), vazdušni jastuci i slično.



Slika 3. Kontrolna jedinica motora

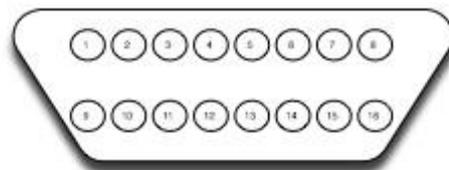
ECU neprestano prati parametre motora kao što su: temperature motora, brzina vozila, količina usisnog vazduha, sastav izduvnih gasova, položaj papučice na gasu, a u nekim slučajevima atmosferski pritisak i nadmorsku visinu. Na osnovu tih informacija podešava rad motora nekoliko desetina puta u sekundi kako bi se obezbedile optimalne performanse. Današnje kontrolne jedinice poseduju i OBD (engl. *On-board diagnostics*) priključak, koji omogućava povezivanje dijagnostičkih uređaja.

2.4 OBD II

OBD II (engl. *On Board Diagnostics II*) [3], sistem za dijagnostiku kvara na vozilu uz pomoć računara, predstavlja drugu generaciju sistema za dijagnostiku. Ukoliko se pojavi nepravilnost u radu vozila, OBD II sistem će uključiti kontrolnu lampicu na instrument tabli kako bi upozorio vozača da postoji neka nepravilnost. Sistem će zapamtiti sve informacije o kvarovima u svoju memoriju, kako bi serviser mogao na što efikasniji način otkloniti problem. OBD II je razvijen kako bi se otklonili nedostaci prethodne generacije sistema za dijagnostiku i kako bi se napravio pouzdan sistem koji će pružiti serviserima veći broj korisnih informacija. Jedan od glavnih nedostataka prve generacije bio je mali broj informacija koje su se mogle dobiti od sistema za dijagnostiku.

2.4.1 Prolaz za komunikaciju

Na Sliku 4 prikazan je 16-polni OBD II priključak za dijagnostiku.



1 - blank	9 - blank
2 - J1850 bus	10 - J1850 bus
3 - blank	11 - blank
4 - Chassis Ground	12 - blank
5 - Signal Ground	13 - Signal Ground
6 - CAN High	14 - CAN Low
7 - ISO 9141-2 K Line	15 - ISO 9141-2 L Line
8 - blank	16 - Battery Power

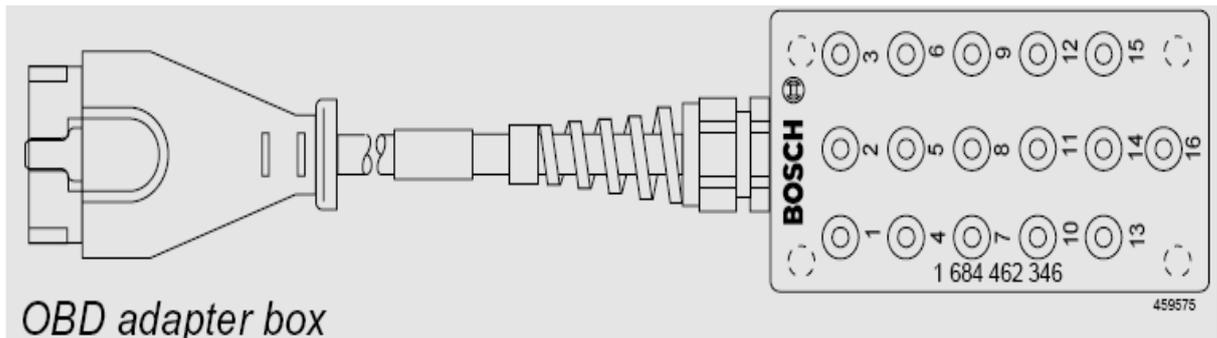
Slika 4. Prolaz za komunikaciju

U nastavku je dat pregled pinova OBD II priključka:

- Pin 7 i 15 – K i L vod
- Pin 2 i 10 – BUS + i BUS -
- Pin 4 – masa vozila / baterije
- Pin 5 – masa elektronike / signala
- Pin 16 – stalni +
- Pin 6 i 14 – CAN-High i CAN-Low, CAN (engl. *Car Area Network/Controller Area Network*), predstavlja računarsku mrežu koja povezuje sve podsisteme u automobilu.

- Pinovi 1, 3, 8, 9, 11, 12 i 13 su slobodni za specifične namene proizvođača. Npr. na njih obično nemački i ostali evropski proizvođači definišu vodove na koje se može priključiti više upravljačkih uređaja.

Takođe, treba napomenuti da K i L vodovi upravljačkih uređaja (npr. *ABS*, automatski menjač, *Airbag*) u zavisnosti od proizvođača, mogu biti priključeni na slobodne pinove 16-polnog priključka (npr. pinovi 1, 8, 9 ili 13). Takvi sistemi se ispituju pomoću OBD adapterske kutije, koja je prikazana na Slika 5.



Slika 5. OBD adapterska kutija

2.4.2 Protokoli za komunikaciju

Za komunikaciju se koriste tri protokola. Razlike se odnose najviše na način komunikacije između kontrolne jedinice motora i alata za dijagnostiku.

Tipovi protokola:

- J1850 VPW (GM) koriste pinove 2,4,5 i 16
- ISO 9141-2 (Evropa, Azija) koriste pinove 4,5,7,15 i 16
- J1850 PWM (Ford) koristi pinove 2,4,5,10 i 16

I pored toga što postoje ova tri protokola koji definišu komunikaciju sa fizičkog nivoa, podaci, odnosno komande koje se prenose su definisane samo jednim standardom SAE J1979.

2.4.3 OBD II identifikatori poruka

Kako bi se ostvarila komunikacija između vozila i uređaja za dijagnostiku, tj. kako bi se dobavili podaci iz vozila mora postojati neki skup simbola po kojima će se potraživati sami podaci. U ovom slučaju to su OBD II identifikatori poruka (engl. *On-board diagnostics Parameter IDs*). Za dobaljanje različitih podataka koriste se različiti identifikatori, kao što je prikazano u Tabela 1.

Identifikatori	Opis identifikatora
01	Trenutni podaci
02	Dobavljanje podataka u trenutku pojave greške u sistemu
03	Kodovi greške
04	Uklanjanje kodova grešaka
05	Nadgledanje senzora kiseonika (ne CAN protokol)
06	Nadgledanje senzora kiseonika (samo CAN protokol)
07	Kodovi greške koji su se dogodili tokom trenutne ili poslednje vožnje tj. ne potvrđeni kodovi greške
08	Kontrolne operacije nad OBD II sistemom

Tabela 1. Način rada

Pored navedenih identifikatora, ukoliko korisnik želi da dobavi podatke o trenutnoj brzini na identifikator 01 dodaje se sufiks 0D. Analogno tome, za broj obrtaja dodaje se sufiks 0C, a za temperaturu 05 itd. Takođe, treba napomenuti da se svaki od ovih identifikatora mora završiti znakom za novi red “\r”, po protokolu.

2.4.4 OBD II ELM327 Bluetooth adapter

Za potrebe ovog rada kao način komunikacije korišćen je *OBD II ELM327* adapter. Komunikacija se ostvaruje Bluetooth vezom.

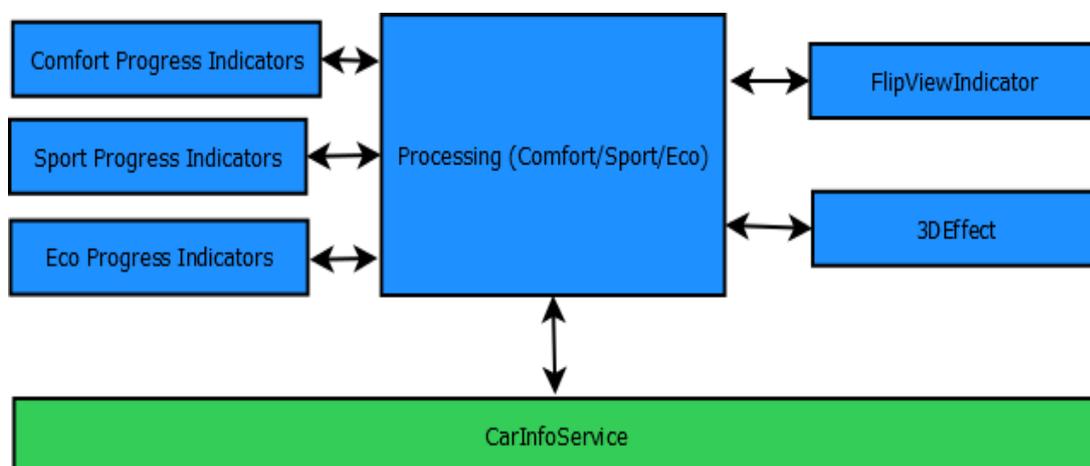


Slika 6. OBD II Adapter

3. Koncept rešenja

Koncept rešenja Android aplikacije pod nazivom “Dashboard” zasniva se na korišćenju Android sistemskih klasa, više-nitnog programiranja i realizaciji pojedinih modula. U ovom poglavlju je dat opis programskih modula, koji predstavljaju najbitnije elemente aplikacije. Projekat se sastoji od sledećih modula:

- Modul koji predstavlja spregu, način komunikacije između aplikacije i vozila pod nazivom Car Info Service.
- Modul za iscrtavanje određenih indikatora na ekranu.
- Moduli za 3D animaciju.



Slika 7. Arhitektura rešenja

3.1 Car Info Service

Modul je realizovan kao Android servis. Android servis ima mogućnost opsluživanja većeg broja klijenata. Kako bi se omogućila komunikacija između klijenta i servisa, mora postojati specificiran skup funkcija koje će klijent pozivati i preko kojih će potraživati informacije. Takođe, jedan od bitnih faktora koji se mora zadovoljiti, jeste brzina preuzimanja podataka.

Brzina preuzimanja podataka mora biti maksimalna kako bi se klijent pravovremeno obavestio o promenama u vozilu.

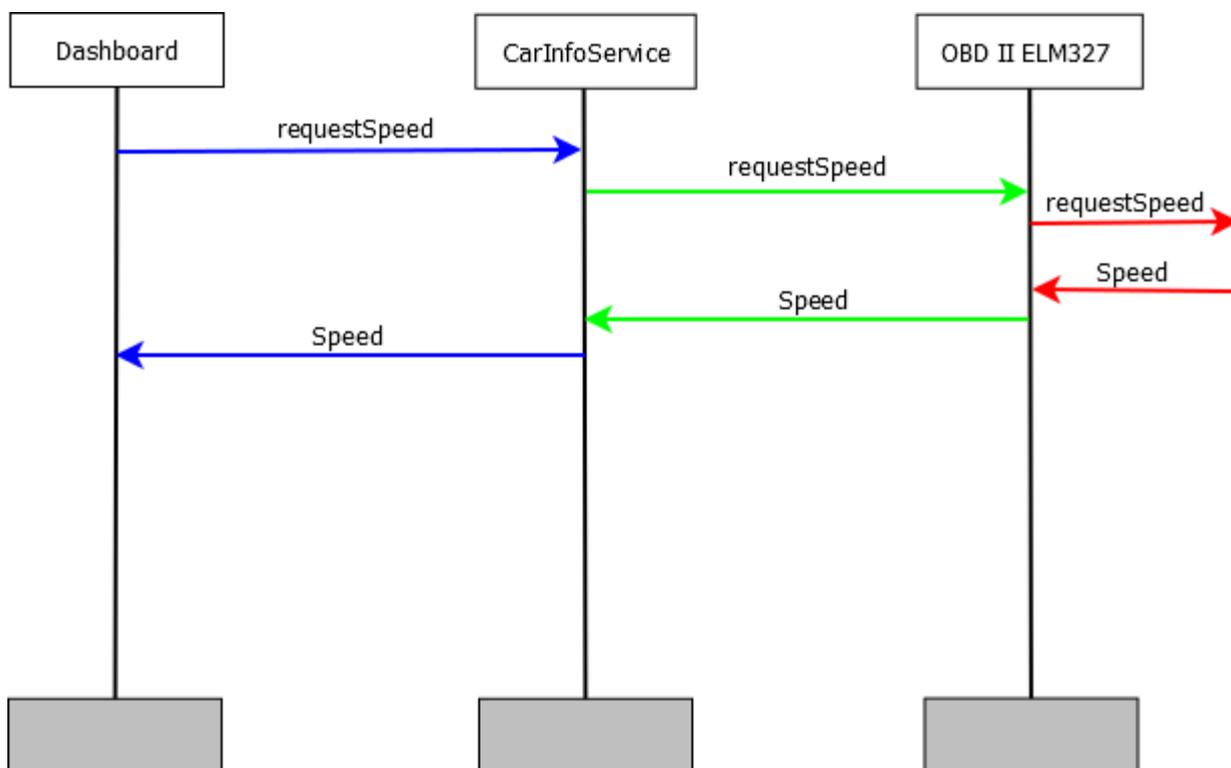
Komunikacija između servisa i vozila ostvarena je *Bluetooth* vezom. U okviru opcija aplikacije izlistavaju se svi dostupni Bluetooth uređaji ili oni uređaji sa kojima je dati prenosni uređaj uparen i na osnovu njih bira se željeni uređaj i povezuje se sa njim.

Veza između klijenta i servisa ostvarena je preko sprege (engl. *Interface*) u okviru koje su realizovane sve bitnije funkcije aplikativne programske sprege (engl. *Application programming interface - API*). Sprega je implementirana kao zasebna biblioteka.

U nastavku je dat pregled najbitnijih funkcija programske sprege:

- *requestSpeed* – funkcija prosleđuje informaciju o trenutnoj brzini vozila.
- *requestThrottlePosition* – funkcija prosleđuje informaciju o poziciji papučice na gasu.
- *requestEngineRpm* – funkcija prosleđuje informaciju o broju obrtaja motora u minutu.
- *requestFuelLevel* – funkcija prosleđuje informaciju o količini goriva u rezervoaru.
- *requestEngineTemperature* – funkcija prosleđuje informaciju o trenutnoj temperaturi motora.

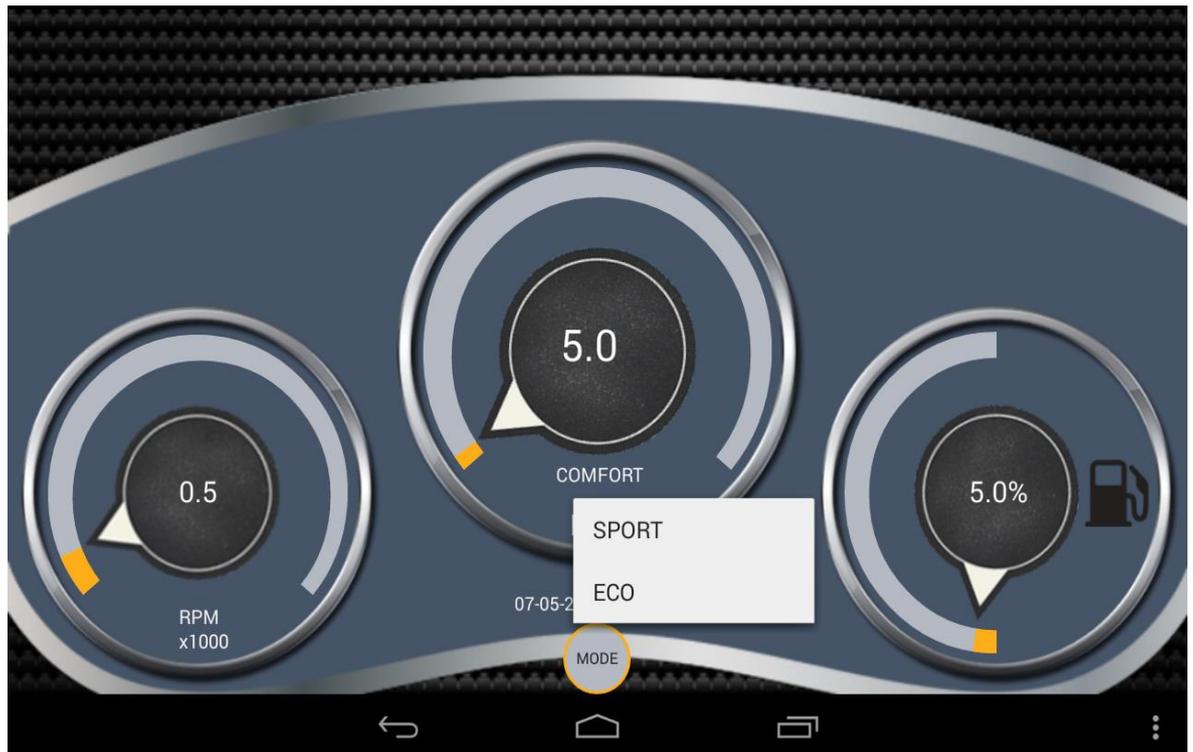
Primer razmene podataka prikazan je na Slika 8.



Slika 8. MSC dijagram razmene podataka

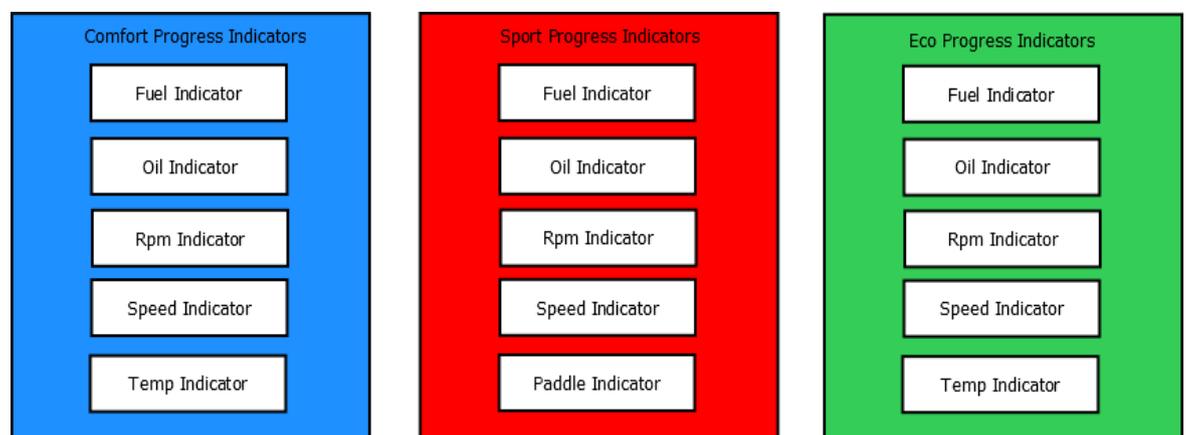
3.2 Modul za iscrtavanje indikatora na ekranu

Modul predstavlja grafičku spregu prema korisniku, odnosno omogućava grafički prikaz određenih indikatora. Pri pokretanju aplikacije pokreće se "Comfort", režim prikazivanja informacija. Korisnik ima mogućnost da bira način prikazivanja između tri ponuđena: *Comfort*, *Sport* i *Eco*. Izbor prikazivanja informacija, korisnik ostvaruje pritiskom na kontrolno dugme, nakon čega se otvara padajući meni sa ponuđenim opcijama. Pritiskom na željenu opciju prelazi se na novu aktivnost sa odgovarajućim podešavanjima za prikazivaje datih informacija.



Slika 9. Prikaz izbora načina prikazivanja informacija

Na Slika 10 može se videti da svaki od načina prikazivanja poseduje skup indikatora koje iscrtava.



Slika 10. Skup indikatora

Svaki od indikatora se sastoji od pet celina:

- Okvira.
- Strelice.
- Teksta.
- Kružnog progressa, koji u zavisnosti od indikatora ima drugačiji ugao iscrtavanja.
- Ikonica, koje pojedini indikatori sadrže.

3.3 Flip View Indicator

Kao što je prikazano na Slika 9, centralni deo ekrana rezervisan je za indikator brzine kretanja vozila, levi deo za indikator broja obrtaja motora u minutu i desni deo za nivo goriva u rezervoaru. Ono što omogućava ovaj modul, jeste da korisnik ima mogućnost promene prikazivanja trenutnih indikatora. Korisnik pritiskom na levi ili desni indikator vrši njegovu zamenu za svoj par. Data promena je realizovana rotacijom indikatora oko Y-ose. Bitno je napomenuti da nakon završetka animacije dolazi do sakrivanja izvornog indikatora i prikazivanja odredišnog indikatora. Takođe, potrebno je promeniti ugao za 180 stepeni kako ne bi odredišni indikator bio okrenut naopako. Ponovnim pritiskom vraća se prikaz početnog indikatora.



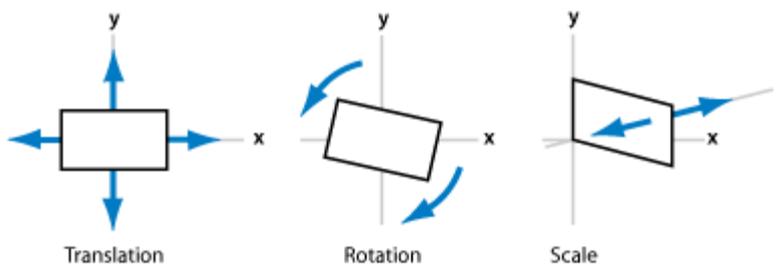
Slika 11. Šematski prikaz rotacije



Slika 12. Prikaz promene indikatora

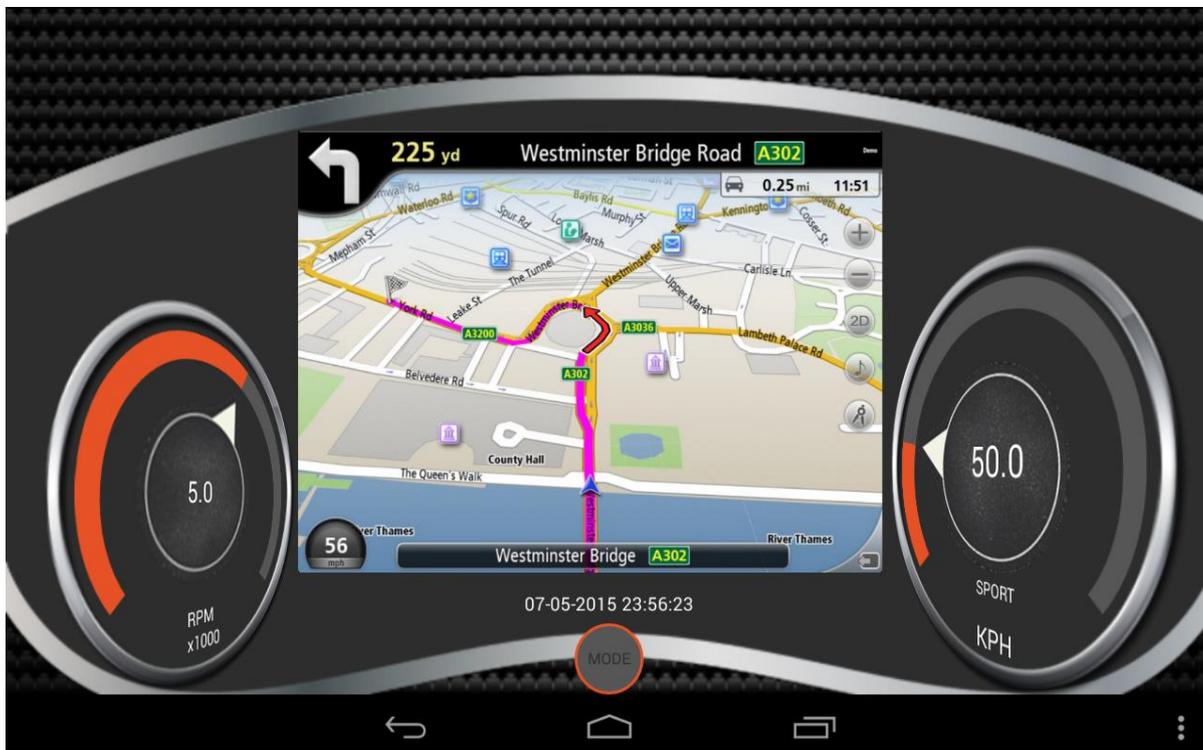
3.4 3D Efekat

Zadatak ovog modula je da omogući korisniku interakciju sa centralnim indikatorom. Pritiskom na centralni indikator, tj. indikator brzine pokreće se animacija, koja pomera indikator brzine u desni ugao i zamenjuje indikator koji se u tom trenutku tu nalazio. Nakon završetka date animacije pokreće se efekat pomeraja na Z-osi. S obzirom da je implementacija ovog rešenja realizovana u 2D prostoru, efekat pomeranja na Z-osi, tj. efekat dubine prostora ostvaren je upotrebom sledećih transformacija koje se mogu videti na Slika 13.



Slika 13. Šematski prikaz 3D efekta

U osnovi, zadatak ovog efekta je da napravi prostor navigacionoj mapi koja će zauzeti centralni deo ekrana. Ulazak mape u vidno polje realizovano je animacijom sa gornje strane ekrana. Pritiskom na mapu vraća se početni prikaz informacija.



Slika 14. Promena prikaza informacija nakon pritiska na centralni indikator

4. Programsko rešenje

Programska realizacija ovog rada napisana je u Java programskom jeziku [4]. Za izradu rada korišćen je Android Studio, razvojno okruženje. Za opis izgleda Android aktivnosti korišćene su XML datoteke (engl. *Extensible Markup Language*). Osnovni paketi koji čine dato programsko rešenje predstavljeni su u Tabela 2.

Naziv paketa	Opis paketa	Sadržaj paketa	
Comfort	Sadrži skup klasa koje služe za pokretanje Comfort aktivnosti i za isrctavanje indikatora	ComfortProcessing.java	
		ProgressIndicatorFuelDash.java	D
		ProgressIndicatorOilDash.java	
		ProgressIndicatorRpmDash.java	R
		ProgressIndicatorSpeedDash.java	A
		ProgressIndicatorTempDash.java	W
		RotatingIndicatorsOnZ.java	
Sport	Sadrži skup klasa koje služe za pokretanje Sport aktivnosti i za isrctavanje indikatora	SportProcessing.java	
		SportFuelIndicator.java	D
		SportOilIndicator.java	
		SportPadlleIndicator.java	R
		SportRpmIndicator.java	A
		SportSpeedIndicator.java	W
		RotatingIndicatorsOnZ.java	
Eco	Sadrži skup klasa koje služe za pokretanje Eco aktivnosti i za isrctavanje indikatora	EcoProcessing.java	
		EcoFuelIndicator.java	D
		EcoOilIndicator.java	
		EcoPadlleIndicator.java	R
		EcoRpmIndicator.java	A
		EcoSpeedIndicator.java	W
		RotatingIndicatorsOnZ.java	
Flipping	Sadrži klasu koja vrši promenu indikatora, rotacijom oko Y-ose	FlipViewAnimator.java	
Effect	Sadrži klasu koja vrši pomeranje indikatora na Z-osi, tj. stvara efekat dubine prostora	Effect3D.java	

Tabela 2. Sadržaj osnovnih paketa programskog rešenja

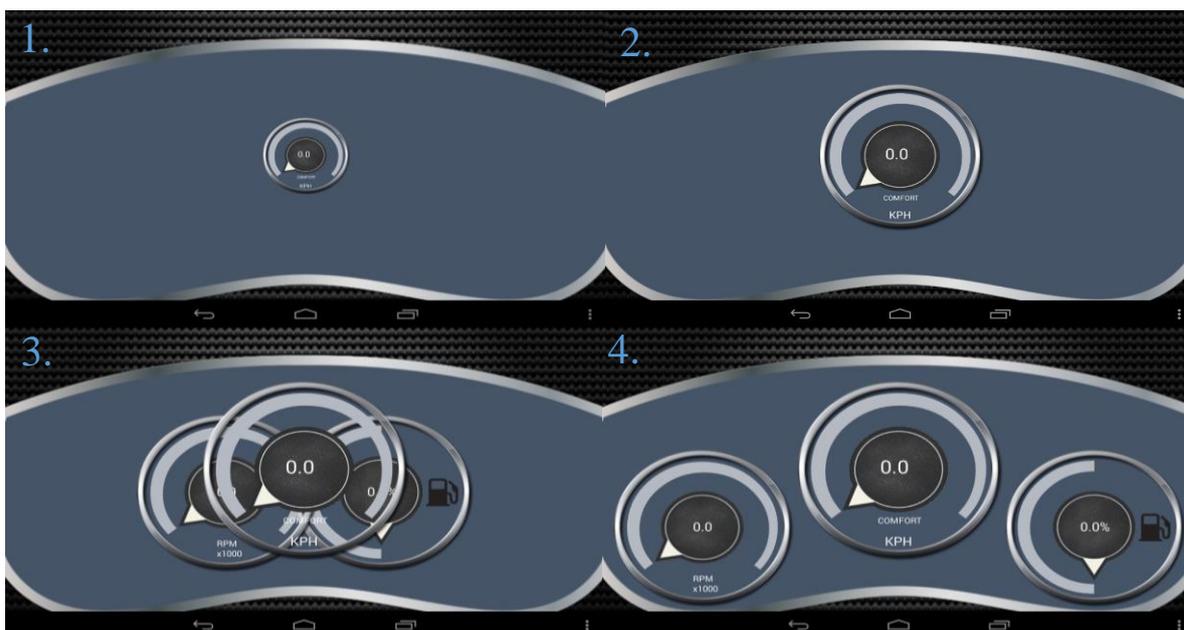
4.1 Paketi: Comfort, Sport, Eco

Navedeni paketi sadrže skup klasa koje služe za pokretanje određene aktivnosti (*ComfortProcessing*, *SportProcessing* ili *EcoProcessing*).

Sve tri klase nasleđuju Android sistemsku klasu *Activity*. Prvenstveno ove klase instanciraju izgled određene aktivnosti. Tačnije indikatore koji su realizovani u okviru paketa, dugme “MODE” za promenu načina prikazivanja indikatora, tekst koji predstavlja datum i vreme, kao i pozadinsku sliku koja u osnovi predstavlja okvir instrument table. Takođe, ove klase implementiraju *AnimationListener* Androidovu sistemsku spregu, na osnovu čega se dobijaju obaveštenja o trenutnoj animaciji. U okviru ovih klasa implementirane su sledeće animacije:

- zoomin
- leftAlignment
- rightAlignment
- moveUpImg
- moveDownImg
- rightSpeedMove

Pri pokretanju aplikacije pokreće se “zoomin” animacija, koja se primenjuje nad centralnim indikatorom tj. indikatorom brzine. Data animacija stvara efekat pojavljivanja indikatora brzine iz pozadine u prvi plan. Nakon završetka pomenute animacije pokreću se “*rightAlignment*” i “*leftAlignment*” animacije. Zadatak ovih animacija, jeste da iz centralne tačke indikatora brzine pomeri indikator obrtaja i indikator nivoa goriva u levi odnosno desni deo ekrana.



Slika 15. Prikaz animacija *zoomin*, *leftAlignment* i *rightAlignment*

Korisnikovim pritiskom na centralni indikator, tj. indikator brzine prvo se pokreće *rightSpeedMove* animacija, koja pomera indikator brzine u desni deo ekrana i zauzima mesto indikatora koji se u tom trenutku tu nalazio. Nakon ove animacija pokreće se *moveDownImg* animacija, koja animira ulaz navigacione mape sa gornje strane ekrana i zauzima centralni deo ekrana. Pritiskom na navigacionu mapu pokreće se *moveUpImg* animacija koja vraća početni prikaza informacija.



Slika 16. Prikaz animacija *rightSpeedMove* i *moveDownImg*



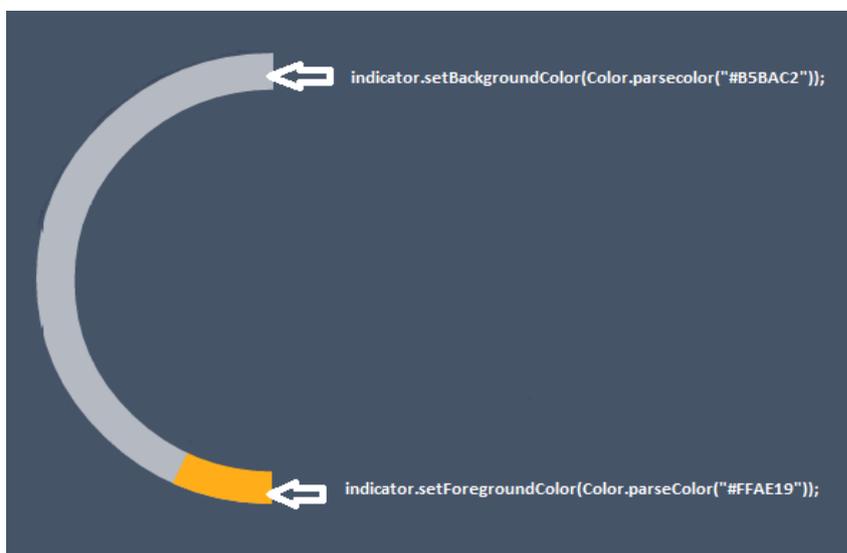
Slika 17. Prikaz animacije *moveUpImg*

Navedene animacije opisane su u *xml* datoteci, a detaljan opis predstavljen je u sledećoj tabeli.

Animacija	Opis animacije u XML-u
zoomin	<pre><?xml version="1.0" encoding="utf-8"?> <set xmlns:android="http://schemas.android.com/apk/res/android"> <scale android:fillAfter="false" xmlns:android="http://schemas.android.com/apk/res/android" android:fromXScale="0.3" android:fromYScale="0.3" android:pivotX="50%" android:pivotY="50%" android:toXScale="1" android:toYScale="1" android:duration="2000"> </scale> </set></pre>
leftAlignment	<pre><?xml version="1.0" encoding="utf-8"?> <set xmlns:android="http://schemas.android.com/apk/res/android"> android:interpolator="@android:anim/linear_interpolator"> <translate android:fromXDelta="0" android:toXDelta="-430" android:toYDelta="145" android:duration="2000"/> </set></pre>
rightAlignment	<pre><?xml version="1.0" encoding="utf-8"?> <set xmlns:android="http://schemas.android.com/apk/res/android"> android:interpolator="@android:anim/linear_interpolator"> <translate android:fromXDelta="0" android:toXDelta="430" android:toYDelta="145" android:duration="2000"/> </set></pre>
rightSpeedMove	<pre><?xml version="1.0" encoding="utf-8"?> <set xmlns:android="http://schemas.android.com/apk/res/android"> android:interpolator="@android:anim/linear_interpolator"> <translate android:fromXDelta="0" android:toXDelta="460" android:toYDelta="145" android:duration="900"/> </set></pre>
moveUpImg	<pre><?xml version="1.0" encoding="utf-8"?> <set xmlns:android="http://schemas.android.com/apk/res/android"> android:interpolator="@android:anim/linear_interpolator"> <translate android:fromXDelta="0" android:fromYDelta="0" android:toYDelta="-1280" android:duration="1000"/> </set></pre>
moveDownImg	<pre><?xml version="1.0" encoding="utf-8"?> <set xmlns:android="http://schemas.android.com/apk/res/android"> android:interpolator="@android:anim/linear_interpolator"> <translate android:fromXDelta="0" android:fromYDelta="-1280" android:toYDelta="0" android:duration="1500"/> </set></pre>

Tabela 3. Opis animacija u XML-u

Grafički prikaz indikatora, tj. njihovo iscrtavanje sastoji se od pet celina kao što je navedeno u konceptu rešenja. Klase koje implementiraju iscrtavanje ovih celina za svaki od načina prikazivanja informacija obeležene su u Tabela 2. Sadržaj osnovnih paketa programskog rešenja, ključnom reči “*DRAW*”. Pri instanciranju indikatora postavlja se boja pozadine (engl. *Background*) i boja prednje strane (engl. *Foreground*) kružnog progressa u zavisnosti o kom se načinu prikazivaja radi (*Comfort, Sport, Eco*).



Slika 18. Prikaz kružnog progressa

Takođe, ono što je spomenuto u konceptu rešenja jeste da se u zavisnosti od tipa indikatora menja i ugao iscrtavanja kružnog progressa. Tako je npr. kružni progres za nivo goriva u rezervoaru realizovan na sledeći način.



```

mBitmap = Bitmap.createBitmap((int) mRect.width(), (int)
    mRect.height(), Bitmap.Config.ARGB_8888);
Canvas canvas = new Canvas(mBitmap);
canvas.drawArc(mRect, 90, 180, true, mPaintBackground);
if (mValue < 0.01f) {
    canvas.drawLine(mRect.width() / 2, mRect.height() / 2,
        mRect.width() / 2, 0, mPaintForeground);
}
float angle = (mValue - minValue) / (maxValue - minValue) * 180;
degree = angle;
canvas.drawArc(mRect, 90, angle, true, mPaintForeground);
if (!mPieStyle) {
    canvas.drawArc(mRectInner, 90, 180, true, mPaintErase);
}

```

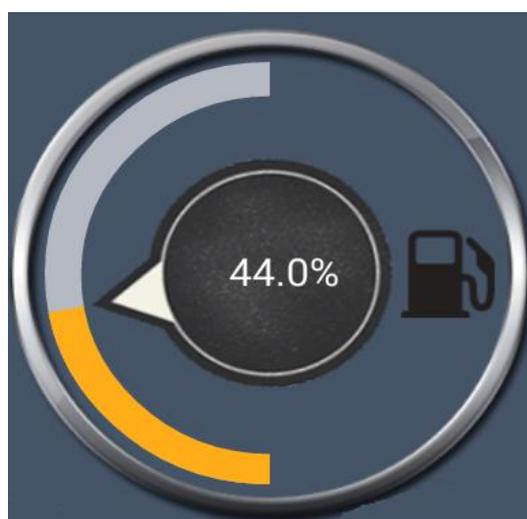
Slika 19. Realizacija kružnog progressa za indikator nivoa goriva u rezervoaru

Kao što se može videti na prethodnoj slici, kako bi se obojila pozadina kružnog progresa, tj. kako bi se osvežio kružni progres, potrebno je postaviti promenljive *mValue*, *minValue* i *maxValue* na određene vrednosti. Pozivom funkcije *setValue(float value, float max, float min)* u jednoj od tri aktivnosti (*ComfortProcessing*, *SportProcessing*, *EcoProcessing*) postavljaju se potrebne vrednosti datih promenljivih i na taj način se osvežava kružni progres, a ujedno se postavlja vrednost promenljive *degree*, koja predstavlja ugao rotacije strelice, i ispisuje se vrednost u tekstu, kao što je prikazano na Slika 20.



Slika 20. Rotacija strelice

Prilikom poziva funkcije *setValue*, u okviru iste poziva se funkcija *updateBitmap()*, koja vrši iscrtavanje svih pet celina (okvir, kružni progres, strelica, tekst i ikonica). Rezultat date funkcije predstavljen je na Slika 21.

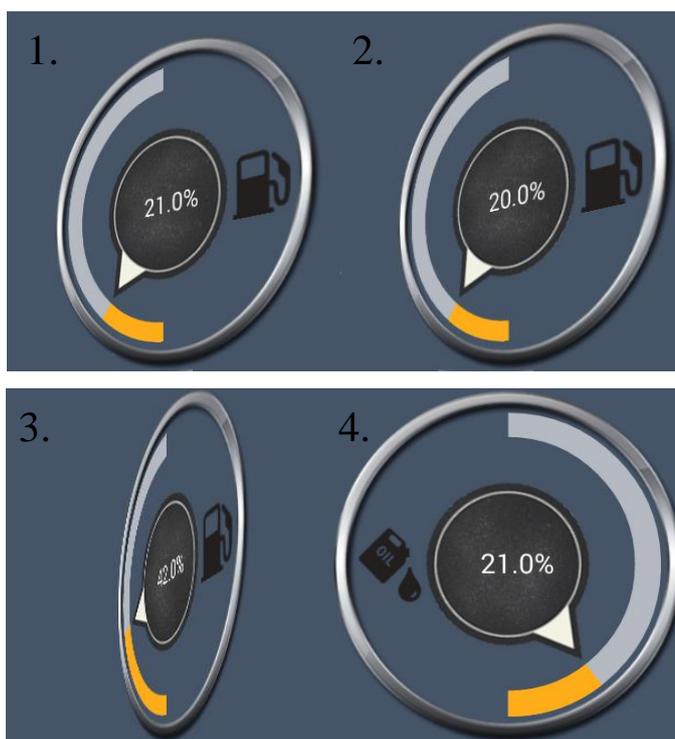
Slika 21. Prikaz rezultata funkcije *updateBitmap()*

4.2 Paket Flipping

U okviru ovog paketa realizovana je klasa *FlipViewAnimator*. Data klasa nasleđuje sistemsku klasu *Animator*, čiji je zadatak da animira pomeranje određenih objekata, u ovom slučaju da rotira indikator oko Y-ose. Kao što je navedeno u konceptu rešenja, na korisnikov pritisak na levi ili desni indikator vrši se njegova promena. Detaljan opis predstavljen je u konceptu rešenja. Za potrebe date animacije, korišćena je instanca klase *Camera*, koja se koristi za računanje 3D transformacija i za generisanje matrica [5]. *FlipViewAnimator(View fromView, View toView, int centerX, int centerY)* predstavlja konstruktor sa sledećim parametrima:

- *fromView* – polazni pogled u promeni.
- *toView* – odredišni pogled u promeni.
- *centerX* – centar pogleda na X-osi.
- *centerY* – centar pogleda na Y-osi.

Pokretanje animacije vrši se u jednoj od tri aktivnosti, u zavisnosti koja se trenutno prikazuje (*ComfortProcessing*, *SportProcessing* ili *EcoProcessing*). Na Slika 22 predstavljen je primer animacije.



Slika 22. Prikaz promene indikatora

4.3 Paket Effect

U datom paketu realizovana je klasa *3DEffect*. Kao i prethodna klasa, i ova klasa nasleđuje sistemsku klasu *Animator*. Zadatak ove animacije je da stvori efekat 3D rotacije na Y-osi. Rotacija je definisana početnim i krajnim uglom. Oba ugla su predstavljena u stepenima. Rotacija se odvija oko centralne tačke 2D prostora, definisana sa X i Y koordinatama. Pri pokretanju animacije, pokreće se pomeraj na Z-osi, tj. efekat dubine prostora koji je ključan faktor za datu animaciju. Detaljan opis predstavljen je u konceptu rešenja. Takođe, za potrebe date animacije korišćena je instanca klase *Camera*. *Effect3D(float fromDegrees, float toDegrees, float centerX, float centerY, float depthZ, boolean reverse)* predstavlja konstruktor sa sledećim parametrima:

- *fromDegrass* – početni ugao 3D rotacije.
- *toDegrass* – krajni ugao 3D rotacije.
- *centerX* – X centar 3D rotacije.
- *centerY* – Y centar 3D rotacije.
- *depthZ* – pomeraj na Z-osi.
- *reverse* – predstavlja promenljivu koja se postavlja na *true* ukoliko treba promeniti pravac pomeraja, u suprotnom postavlja se na *false*.

Cilj pomenute animacije jeste preraspodela dostupnog prostora tako da navigaciona mapa bude pozicionirana u središnjem delu ekrana. Pritiskom na centralni indikator, pokreće se animacija, koja je instancirana u klasi *RotatingIndicatorsOnZ*. Na Slika 23 predstavljen je primer 3D efekta.



Slika 23. Primer 3D Efekta

5. Rezultati

Relizacija rešenja je ispitana izvršavanjem aplikacije sa određenim brojem komponenti grafičke korisničke sprege (engl. *Graphical User Interface*). Konkretno ispitivanje je vršeno na pet indikatora i na navigacionoj mapi u zavisnosti od trenutnog načina prikazivanja informacija (*Comfort, Sport ili Eco*). Prilikom ispitivanja akcenat je stavljen na naredne elemente:

- Iscrtavanje indikatora
- Promena načina prikazivanja indikatora
- Uspešnost izvršavanja animacija
- Uspešnost izvršavanja 3D efekata

Platforme na kojima je vršeno ispitivanje su:

- Lenovo A5500-F (Chipset Mediatek MT8382, CPU Quad-core 1.3 GHz Cortex-A7, GPU Mali-400MP2)
- HTC Nexus 9 (Chipset Nvidia Tegra K1, CPU Dual-core 2.3 GHz Denver, GPU Kepler DX1)
- Automotive platforma zasnovana na ARM fizičkoj arhitekturi sa Android operativnim sistemom

Takođe, treba napomenuti da su za potrebe ispitivanja rešenja korišćena dva načina promene vrednosti indikatora. U prvom slučaju realizovana je logika promene vrednosti u određenom vremenskom intervalu u okviru jedne od tri aktivnosti (*ComfortProcessing, SportProcessing ili EcoProcessing*). Drugi slučaj podrazumeva korišćenje OBD II simulatora, na osnovu čega se menjaju vrednosti indikatora. Komunikacija sa simulatorom ostvarena je upotrebom *CarInfoService* servisa, *Bluetooth* vezom, koji je detaljno opisan u konceptu rešenja. Pored ova dva načina verifikacije, rešenje je verifikovano i u realnim uslovima korišćenjem OBD II adaptera u automobilu, gde se moglo videti da nema usporenja (engl. *lagging*) u odnosu na standardu instrument tablu.

Realizacijom trodimenzionalnih efekata stvoren je jedan koncept koji omogućava da interakcija sa samim korisnikom bude prirodija i interesantna.

Na kraju treba napomenuti, da je osnova ovog zadatka realizacija trodimenzionalnih efekata upotrebom standardnog Android radnog okruženja. Dalje unapređenje ovog rešenja bilo bi usmereno ka upotrebi OpenGL (engl. *Open Graphics Library*) biblioteke, koja bi omogućila iscrtavanje slika pomoću OpenGL tekstura umesto iscrtavanja slika u formatu bitmap datoteka preko standardne Android sprege. Ovaj način iscrtavanja slika omogućava korišćenje specijalizovane jedinice fizičke arhitekture u okviru grafičkog procesora (engl. *Graphics Processor Unit - GPU*), koja bi doprinela smanjenju opterećenja centralnog procesora.

6. Zaključak

U ovom radu je opisana realizacija Android aplikacije za prikazivanje informacija od strane kontrolne jedinice motora. Grafička korisnička sprema aplikacije je realizovana korišćenjem Android sistemskih biblioteka. Pored standardnih Android grafičkih elemenata uvedeni su i trodimenzionalni efekti i animacije. Izvršeno je ispitivanje na uređajima Lenovo A5500F i HTC Nexus 9 i na *Automotive* platformi. Prilikom projektovanja grafičke korisničke sprege za datu Android aplikaciju, vođeno je računa o jednostavnosti i minimalističkom izgledu aplikacije, kako bi se korisnicima olakšalo njeno svakodnevno korišćenje.

Zadatkom je realizovan osnovni koncept aplikacije. Ostavljen je prostor za dalja unapređenja, kao što su: prikaz lampice upozorenja, pokazivača pravca, pređene kilometraže, itd. Takođe, dalji razvoj bi mogao biti usmeren na realizaciji grafičke korisničke sprege upotrebom OpenGL biblioteke, kao i realizaciji trodimenzionalnih efekata u C++ programskom jeziku, radi boljeg iskorišćenja resursa.

7. Literatura

- [1] Vladimir Kovačević, Miroslav Popović: *Sistemska programska podrška u realnom vremenu*, Univerzitet u Novom Sadu, Fakultet Tehničkih Nauka, 2002
- [2] Vladimir Kovačević: *Logičko projektovanje računarskih sistema I – projektovanje digitalnih sistema*, Univerzitet u Novom Sadu, Fakultet Tehničkih Nauka, 2001
- [3] Web stranica Wikipedia, *The Free Encyclopedia*,
http://en.wikipedia.org/wiki/On_board_diagnostics
učitana 28.03.2015
- [4] Reto Meier: *Professional Android Application Development*, USA, 2008
- [5] Web stranica Android podrške za razvoj, *Android Developers*,
<http://developer.android.com>
korišćena od 28.03.2015 do 10.05.2015