



# УНИВЕРЗИТЕТ У НОВОМ САДУ

## ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



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: Ивана Николић

Број индекса: РА101/2012

Тема рада: Апликациона програмска спрега за руковање системима у возилу у апликативним окружењима заснованим на HTML5

Ментор рада: доц. др Милан Ђелица

Нови Сад, avgust, 2016



## 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:	Ивана Николић		
Ментор, МН:	доц. др Милан Ђелица		
Наслов рада, НР:	Апликациона програмска спрела за руковање системима у возилу у апликативним окружењима заснованим на HTML5		
Језик публикације, ЈП:	Српски / латиница		
Језик извода, ЈИ:	Српски		
Земља публиковања, ЗП:	Република Србија		
Уже географско подручје, УГП:	Војводина		
Година, ГО:	2016		
Издавач, ИЗ:	Ауторски репринг		
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6		
Физички опис рада, ФО: (поглавља/страница/цитата/табела/слика/графика/прилога)	7/28/ 0/2/16/0/0		
Научна област, НО:	Електротехника и рачунарство		
Научна дисциплина, НД:	Рачунарска техника		
Предметна одредница/Кључне речи, ПО:			
УДК			
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад		
Важна напомена, ВН:			
Извод, ИЗ:	У раду је представљено једно рјешење апликационе програмске спреле за руковање системима у возилу у апликативним окружењима заснованим на HTML5. Описана је реализација JavaScript библиотеке са апликационом програмском спрелом којом се омогућује HTML5 апликацијама, које се извршавају у склопу апликативног окружења у возилу, да приступају системима возила, као и реализација додатка за Chromium прегледач, који омогућава повезивање HTML5 апликација са GENIVI средњим слојем и одговарајућим руковаоцима. Уз објашњење рјешења такође је дато објашњење појмова неопходних за схватање рада, проширено slikama. Ово рјешење испуњава услове задате поставком задатка.		
Датум прихватања теме, ДП:			
Датум одбране, ДО:			
Чланови комисије, КО:	Председник:	доц. др Јелена Ковачевић	
	Члан:	доц. др Иштван Пап	Потпис ментора
	Члан, ментор:	доц. др Милан Ђелица	



## KEY WORDS DOCUMENTATION

Accession number, <b>ANO:</b>			
Identification number, <b>INO:</b>			
Document type, <b>DT:</b>	Monographic publication		
Type of record, <b>TR:</b>	Textual printed material		
Contents code, <b>CC:</b>	Bachelor Thesis		
Author, <b>AU:</b>	Ivana Nikolić		
Mentor, <b>MN:</b>	Milan Bjelica PhD		
Title, <b>TI:</b>	Application Programming Interface for Systems Manipulation in Vehicles in Application Development Environments Based on HTML5		
Language of text, <b>LT:</b>	Serbian		
Language of abstract, <b>LA:</b>	Serbian		
Country of publication, <b>CP:</b>	Republic of Serbia		
Locality of publication, <b>LP:</b>	Vojvodina		
Publication year, <b>PY:</b>	2016		
Publisher, <b>PB:</b>	Author's reprint		
Publication place, <b>PP:</b>	Novi Sad, Dositeja Obradovica sq. 6		
Physical description, <b>PD:</b> <small>(chapters/pages/ref./tables/pictures/graphs/appendices)</small>	7/28/ 0/2/16/0/0		
Scientific field, <b>SF:</b>	Electrical Engineering		
Scientific discipline, <b>SD:</b>	Computer Engineering, Engineering of Computer Based Systems		
Subject/Key words, <b>S/KW:</b>			
<b>UC</b>			
Holding data, <b>HD:</b>	The Library of Faculty of Technical Sciences, Novi Sad, Serbia		
Note, <b>N:</b>			
Abstract, <b>AB:</b>	This paper presents one solution of application programming interface for in-vehicle infotainment systems handling. The paper describes the implementation of <i>JavaScript</i> library with application programming interface that allows <i>HTML5</i> applications, which run within the application environment of the vehicle, to access the vehicle systems, and the implementation of a plugin for <i>Chromium</i> browser, which allows connecting <i>HTML5</i> applications with <i>GENIVI</i> middleware and the corresponding drivers.		
Accepted by the Scientific Board on, <b>ASB:</b>			
Defended on, <b>DE:</b>			
Defended Board, <b>DB:</b>	President:	Jelena Kovačević PhD	
	Member:	Ištván Pap PhD	Menthor's sign
	Member, Mentor:	Milan Bjelica PhD	

## **Zahvalnost**

Zahvaljujem se mentoru doc. dr Milanu Bjelici i Mileni Milošević na stručnoj pomoći i savjetima tokom izrade ovog rada.



# УНИВЕРЗИТЕТ У НОВОМ САДУ

## ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



### SADRŽAJ

1.	Uvod .....	1
2.	Teorijske osnove.....	3
2.1	In-vehicle infotainment .....	3
2.2	Web tehnologije kao osnova za razvoj IVI GUI.....	7
2.3	Aplikativne sprege u vozilima .....	8
2.3.1	Vlasničke tehnologije .....	8
2.3.2	Tehnologije otvorenog koda.....	12
3.	Koncept rješenja .....	18
3.1	Proširenje <i>Chromium</i> pregledača .....	19
3.2	Opis rješenja.....	19
4.	Programsko rješenje .....	22
4.1	IVI JS plugin .....	22
4.2	IVI JS API.....	24
5.	Ispitivanje .....	31
5.1	Opis okruženja za ispitivanje .....	31
5.2	Ključni slučajevi upotrebe .....	32
5.3	Ispitivanje performansi .....	34
6.	Zaključak .....	37
7.	Literatura .....	38

---

## **SPISAK SLIKA**

Slika 1: Primjer IVI sistema.....	5
Slika 2: Izgled <i>Apple CarPlay</i> aplikacije na displeju <i>IVI</i> sistema .....	9
Slika 3: Izgled <i>Android Auto</i> aplikacije na displeju <i>IVI</i> sistema u automobilu .....	10
Slika 4: Izgled <i>Android Auto</i> aplikacije .....	11
Slika 5: Arhitektura <i>Android Auto</i> softvera .....	12
Slika 6: Arhitektura softvera koji je u skladu sa <i>GENIVI</i> specifikacijom .....	13
Slika 7: Izgled <i>AGL</i> aplikacije .....	14
Slika 8: Primjer <i>AGA</i> aplikacije .....	15
Slika 9: Prikaz arhitekture rješenja .....	18
Slika 10: Postupak ostvarivanja TCP veze sa HMI agentom .....	23
Slika 11: Okruženje za testiranje izrađeno tako da predstavlja unutrašnjost automobila	32
Slika 12: Početna brzina je 0km/h – automobil stoji .....	33
Slika 13: Brzina se povećala na 100km/h kao posledica pritiska pedale gasa .....	33
Slika 14: Prikaz trenutne potrošnje goriva i broja preostalih kilometara .....	34

## SPISAK TABELA

Tabela 1: Primjer primjene <i>LIN</i> magistrale u <i>IVI</i> aplikaciji .....	6
Tabela 2: Pregled API grupa i informacija koje im pripadaju .....	21
Tabela 3: Rezultati testnih slučajeva .....	35

## SKRAĆENICE

- ADAS – *Advanced Driver Assistance Systems*, Sistemi za pomoć vozačima  
API – *Application Programming Interface*, Interfejs za programiranje aplikacija  
CSS – *Cascading Sheet Style*  
HTTP – *Hypertext Transfer Protocol*, Komunikacioni protokol za razmjenu podataka  
IVI – *In-vehicle Infotainment*, Integriran skup sistema u automobilu

## 1. Uvod

U poslednjih nekoliko godina, integriran skup sistema u automobilu koji vozačima i putnicima pruža sadržaj zabavnog i prije svega informacionog karaktera, poznat pod nazivom *In-Vehicle Infotainment* (IVI), postaje jednako važan za kupce kao i 'ono što se nalazi ispod haube'. Međutim, iako je razvoj IVI sistema danas u velikom napretku, još uvijek ne postoji jedinstven interfejs za programiranje aplikacija (eng. *Application Programming Interface*, API) kao ni fleksibilna rješenja za integraciju novih aplikacija. Potrebno je definisati API sloj koji će na standardizovan način omogućavati aplikacijama da pristupaju funkcionalnostima vozila.

Rješenje opisano u ovom radu predstavlja jedan pristup koji omogućava integraciju novih aplikacija u različitim IVI sistemima – interfejs za programiranje aplikacija zasnovan na *JavaScript (JS)* programskom jeziku, koji omogućava pristup specijalizovanoj internoj komunikacionoj mreži, koja povezuje komponente unutar vozila (eng. *vehicle bus*), iz standardnog pretraživača na uniforman način. Rješenje je realizovano u okviru Chromium pretraživača koji se izvršava na *Yocto GENIVI* srednjem sloju (eng. *middleware*). Pretraživač je proširen IVI dodatkom (eng. *plugin*) za *Chromium* pregledač, koji omogućava povezivanje *HTML5* aplikacija sa *GENIVI* srednjim slojem i odgovarajućim rukovaocima, kako bi podaci o vozilu bili dostupni u JS.

Ovaj rad je sačinjen od 7 poglavlja.

U prvom poglavlju vrši se pozicioniranje rada u oblast – objašnjava se tema rada, i ukratko se opisuje njegov sadržaj.

Drugo poglavlje opisuje pojmove *In-Vehicle Infotainment* i *Web* tehnologija u kontekstu *IVI* grafičkih korisničkih okruženja. Dat je i osvrt na aplikativne tehnologije u vozilima.

U trećem poglavlju je detaljno opisano aplikativno okruženje za vozila zasnovano na *HTML5*. Dat je dijagram rješenja kao i opis arhitekture. Takođe je opisan način na koji je implementiran dodatak za *Chromium* pretraživač i dato je objašnjenje komunikacije među slojevima arhitekture.

Četvrto poglavlje sadrži detaljan opis programske realizacije dodatka pretraživaču, kao i *JS* biblioteke sa aplikacionom programskom spregom.

U petom poglavlju je dat opis okruženja za ispitivanje rješenja, navedeni su ključni slučajevi upotrebe i izvršena je verifikacija i ispitivanje performansi rješenja što u ovom slučaju podrazumijeva računanje vremena propagacije komande kroz dodatak.

U šestom poglavlju je dat osvrt na ono što je do sada urađeno, kao i dalji pravci kretanja razvoja.

U sedmom poglavlju je naveden spisak literature korištene prilikom pisanja rada.

## 2. Teorijske osnove

Prvo poglavlje sadrži detaljnije objašnjenje pojma *In-Vehicle Infotainment*, kao i opis razvoja ovih sistema i njihov značaj. U sledećem poglavlju je dat opis nekih *Web* tehnologija koje čine osnovu za razvoj *IVI* grafičkog korisničkog okruženja (eng. *Graphical User Interface*), kao i njihov značaj i prednosti. U narednom poglavlju su navedene i predstavljene neke aplikativne sprege u vozilima.

### 2.1 In-vehicle infotainment

*In-Vehicle Infotainment* ili *IVI* je termin iz automobilske industrije koji se odnosi na integrисани skup hardvera i softvera u automobilu koji vozačima i putnicima dostavlja informacije i pruža sadržaj zabavnog karaktera [1].

Prije početka razvoja *IVI* sistema, jedini izvor zabavnog sadržaja u automobilima su bili audio sistemi koji su se sastojali od FM radija, kaseta i CD plejera. Međutim, zahvaljujući novom digitalizovanom načinu života, sve većem razvoju 'uvijek povezani' (eng. *always connected* – u smislu pristupa internetu) zajednica i sve višim zahtjevima korisnika, *IVI* sistemi su postali jedna od ključnih i najbrže rastućih tehnologija u automobilskoj industriji [2]. U skladu sa tim, mogućnosti *IVI* sistema kao i broj funkcionalnosti koje nude stalno rastu u cilju obezbjeđivanja jedinstvenog iskustva vožnje i poboljšanja bezbjednosti, pa tako danas za pružanje zabavnog i informacionog sadržaja vozačima i putnicima, *IVI* sistemi koriste razne audio/video (A/V) interfejse, ekrane osjetljive na dodir, tastature i mnoge druge vrste uređaja, zahvaljujući kojima mogu da ponude sledeće:

- A/V funkcije i alate za ostvarivanje dvosmjerne komunikacije kao što su standardni radio i CD plejeri, glasovne komande vozila, obavljanje *hands-free* telefonskog razgovora (korisnik obavlja razgovor pomoću mikrofona i spikerfona) i mnoge druge vrste interaktivnog audia ili videa;
- Ekrane na zadnjim sjedištima koji omogućavaju putnicima gledanje filmova ili drugih vizuelnih medija;
- Mogućnost povezivanja mobilnog uređaja - novija vozila imaju čitav niz sistema koji omogućavaju uređajima poput pametnih telefona ili laptop računara povezivanje sa vozilom;
- Automobilske navigacione sisteme;
- Bluetooth i USB vezu;
- Video plejere;
- Karpjutere (računar specijalizovan za automobile – u vidu kompaktne dimenzije, male potrošnje energije i nekih drugih prilagođenih komponenti);
- Internet, WiFi;
- Mnogi IVI sistemi takođe imaju bezbjednosne funkcije koje sprječavaju vozače da koriste bilo koju video uslugu ili druge elemente sistema koji bi mogli uticati na njihovu pažnju;
- *Head-up* displej – HUD (eng. *head-up display*) – transparentni displej koji prikazuje podatke vozaču bez potrebe skretanja pogleda sa puta;
- Sistem za pomoć vozačima (eng. *Advanced Driver Assistance Systems*, ADAS);
- Personalizaciju unutrašnjosti.

Na Slici 1 je prikazan izgled jednog IVI sistema. Ovaj IVI sistem je pozicioniran na sredini kontrolne table automobila, između vozača i suvozača. Sastoji se od dva ekrana raspoređena po vertikali. Na donjem ekranu je prikazan izbornik aplikacije kojim se upravlja pomoću dugmića osjetljivih na dodir, koji su vidljivi sa lijeve i desne strane, i dugmića koji se takođe nalaze sa lijeve i desne strane, ali i ispod ekrana. Na ekranu iznad se vidi prikaz navigacije.



Slika 1: Primjer IVI sistema

## 2.2 Komunikacija IVI sistema sa komponentama automobila

U ovom poglavlju će biti ukratko predstavljene specijalizovane interne komunikacijske mreže koje povezuju komponente unutar vozila, a preko kojih *IVI* sistemi mogu dobiti sve potrebne podatke o automobilu.

Kotrolna mreža (eng. *Controller Area Network bus*, CAN) ili CAN magistrala [2] je standardna asinhrona serijska magistrala u vozilima dizajnirana sa ciljem da omogući elektronskim kontrolnim jedinicama (eng. *Electronic Control Units*, ECUs; kočnica, motor, automatski mjenjač, elektronsko ubrizgavanje goriva, itd.) poznatim kao čvorovi (u daljem tekstu čvor), da međusobno komuniciraju u vozilu bez centralnog računara.

Uredaji koji su povezani na CAN mreže su obično senzori, pogoni i drugi kontrolni uređaji. Ovi uređaji su povezani na CAN magistralu preko centralnog računara, CAN kontrolera i CAN primopredajnika.

*LIN* magistrala (*Local Interconnect Network Bus*) [3] je serijski mrežni protokol koji se koristi za komunikaciju komponenti vozila. Ova magistrala je implementirana da bi dopunila postojeću *CAN* magistralu što je dovelo do stvaranja hierarhije među magistralama. Trenutno se *LIN* magistrala kombinuje sa jednostavnim senzorima kako bi se kreirale male komunikacione mreže, koje se mogu povezati sa većim mrežama, kao što je na primjer *CAN*.

*LIN* ne može u potpunosti zamijeniti *CAN* magistralu, ali se može iskoristiti kao dobra alternativa ukoliko su niski troškovi od suštinskog značaja i ukoliko brzina protoka nije važna.

U Tabeli 1 su prikazane informacije koje *LIN* magistrala može obezbijediti *IVI* aplikacijama.

Segmenti aplikacije	Primjeri primjene <i>LIN</i> magistrale
Krov	Senzor, svjetlosni senzor, kontrola svjetla, krovni prozor
Volan	Tempomat, brisači, rotirajuće svjetlo, klima, radio
Sjedište	Položaj sjedišta, senzori za putnike
Motor	Senzori, mali motori, ventilatori
Klima	Kontrola klime, mali motor
Vrata	Retrovizori, podizanje/spuštanje prozora, prekidač za kontrolu sjedišta, brava
Osvjetljenje	Jačina osvjetljenja

Tabela 1: Primjer primjene *LIN* magistrale u *IVI* aplikaciji

*MOST (Media Oriented Systems Transport)* [4] je multimedijalna mrežna tehnologija velike brzine optimizovana za automobilsku industriju. Može se koristiti za aplikacije unutar ili izvan automobila. Ova magistrala koristi prsten topologiju i sinhromu komunikaciju podataka za prenos zvuka, videa, glasa i podataka o vozilu preko plastičnog optičkog vlakna ili električnog provodnika.

*FlexRay* [5] je još jedan automobilski komunikacijski mrežni protokol koji je otporan na greske i razvijen da bude brži od *CAN* magistrale.

## 2.3 Web tehnologije kao osnova za razvoj IVI GUI

Pod *Web* (internet) tehnologijom [6] se podrazumijeva uspostavljanje i korištenje mehanizama koji omogućavaju međusobnu komunikaciju i dijeljenje resursa među računarima. Drugim riječima, internet tehnologija pruža platformu za efikasnu komunikaciju korisnika i uređaja na računarskoj mreži.

Neki od primjera *Web* tehnologije su:

- *HTML, CSS, JavaScript*;
- Tehnologije i programski jezici pomoću kojih se kreiraju aplikacije za internet. Neki od programske jezike su *Perl, C#, Java i Visual Basic .Net*;
- Internet serveri i serverske tehnologije koje olakšavaju rukovanje zahtjevima na mreži gdje različiti korisnici dijele iste resurse i komuniciraju jedni sa drugima;
- Baze podataka koje su od ključnog značaja za skladištenje podataka i informacija na računarskoj mreži;
- Poslovne aplikacije prilagođene izvršavanju specifičnih zadataka na mreži.

Posebno treba izdvojiti tehnologije otvorenog koda (eng. *open source*), navedene u prvoj tački, koje pružaju širok spektar mogućnosti razvoja *IVI* grafičkog korisničkog okruženja koje je zasnovano na internet tehnologiji. Neke od tih tehnologija su *JavaScript* [7] – skriptni programski jezik, dinamičan, slabo tipiziran, standardizovan u specifikaciji *ECMAScript* jezika, *CSS (Cascading Style Sheets)* [8] – stilski jezik koji se koristi za opisivanje prikaza dokumenta napisanog u programskom jeziku, *HTML (Hyper Text Markup Language)* [9] – opisni jezik namijenjen opisu internet stranica, pomoću kog se jednostavno mogu izdvojiti elementi kao što su naslovi, paragrafi, citati i sl.

*JavaScript* je skriptni programski jezik, čija je osnovna uloga definisanje funkcionalnosti internet stranica sa strane klijenta. Zajedno sa *HTML* i *CSS* jezicima, *JS* predstavlja jednu od ključnih tehnologija za proizvodnju internet sadržaja. Osnovne karakteristike *JavaScript* programskog jezika su da je to slabo tipiziran jezik sa skromnom podrškom za objektno-orientisano programiranje. Ovaj jezik je najpoznatiji po programiranju klijentske funkcionalnosti internet stranica, ali se može koristiti i kao skriptni jezik za pristup objektima koji se nalaze u drugim aplikacijama. Uloga *JS* programskog jezika u razvijanju grafičkog korisničkog iskustva za *IVI* sisteme u automobilskoj industriji ogleda se u kontroli stanja automata koji na osnovu primljenih podataka iz dodatka internet pretraživaču, koji je takođe realizovan u *JS* programskom jeziku, upravlja samom grafikom.

*HTML* je standardni jezik za kreiranje internet stranica i *Web* aplikacija, koji semantički opisuje strukturu i izgled internet stranica.

*CSS* – još jedan stilski jezik koji se koristi za opisivanje izgleda internet stranice. Namjena *CSS* je da omogući razdvajanje sadržaja dokumenta od njegovog izgleda, uključujući boje, fontove i pozadinu.

Ova tri programska jezika predstavljaju osnovnu tehnologiju koju koristi većina internet stranica za kreiranje vizuelno privlačnog sadržaja, korisničkih interfejsa za *Web* aplikacije, kao i korisničke interfejsa za mnoge aplikacije za mobilne uređaje. Prednosti ovih tehnologija u odnosu na ostale tehnologije za razvoj grafičkog korisničkog okruženja se ogledaju u tome što je razvijanje softvera mnogo lakše, sintaksa je jednostavna, a i ove tri tehnologije su najbolje dokumentovane tehnologije za razvoj softvera.

## 2.4 Aplikativne sprege u vozilima

Postoje različite tehnologije za razvoj *IVI* korisničkog interfejsa (eng. *Graphical User Interface* – vrsta korisničkog interfejsa koja omogućava korisnicima da komuniciraju sa elektronskim uređajima preko grafičkih ikonica i vizuelnih objekata). Tehnologije koje se bave izradom softvera za automobilsku industriju se dijele na vlasničke tehnologije i tehnologije otvorenog koda (eng. *open source technologies*).

### 2.4.1 Vlasničke tehnologije

*Apple CarPlay* [10] – jedan od primjera vlasničke tehnologije, koji proizvođačima automobila nudi mogućnost da komplikovane *IVI* sisteme zamijene displejem koji komunicira sa *iPhone* telefonom. Ovo rješenje nije sistem u automobilu na kom se pokreće *iOS* (operativni sistem) ili *iOS* aplikacije, ovo rješenje samo prebacuje *iOS* okruženje na displej *IVI* sistema, što omogućava korisnicima da kontrolišu ne samo aplikacije već i uređaj, kako preko displeja tako i pomoću glasovnih komandi.

Ideja *CarPlay* aplikacije jeste da se omogući da se poznato *iOS* korisničko okruženje prenese na *IVI* ekrane, kao i da se omogući njegova kontrola koristeći sve dostupne kontrole automobila, tako da korisnicima bude omogućeno da koriste sve *Apple* aplikacije (na primjer da slušaju *Apple iTune*, da koriste *Apple* navigaciju *Apple Maps*, sve funkcionalnosti telefona itd.) Ostale aplikacije, kao što su *Pandora*, *iHeartRadio*, *NPR News*, su takođe dostupne kroz *CarPlay* [11].

Među proizvođačima automobila koji podržavaju ovu tehnologiju su *Audi*, *Honda*, *Hundai*, *Kia*, *Volkswagen*, najnoviji modeli *Mercedes* automobila, itd.

Na Slici 2 je prikazan izgled *CarPlay* aplikacije u automobilu. Prikaz na ekranu kontrolne table automobila je prilagođen tipičnom *Apple* okruženju.



Slika 2: Izgled *Apple CarPlay* aplikacije na displeju *IVI* sistema

*Android Auto* [12] je još jedan od vodećih primjera vlasničkih tehnologija. Ova tehnologija, slično *Apple CarPlay* aplikaciji, omogućava pametnim telefonima sa *Android* operativnim sistemom (verzija 5.0 “Lollipop” i više) da se povežu sa automobilom, i prebacuje aplikacije i sav sadržaj sa telefona na ekrane na kontrolnoj tabli automobila, i optimizuje ih tako da budu čitljive u toku vožnje. Ovaj standard omogućava vozačima da kontrolišu *GPS* (eng. *Global Positioning System*; mape i navigaciju), reprodukciju muzike, *SMS* poruke (eng. *Short Message Service*), telefoniju, internet pretraživanje, itd. Cilj *Android Auto* tehnologije je proširivanje funkcionalnosti telefona na automobile, tako da glavni ekran u automobilu bude spoljni ekran (eng. *external display*) za telefon, s tim da prikaz sadržaja sa telefona bude prilagođen korisničkom okruženju tipičnom za automobile.

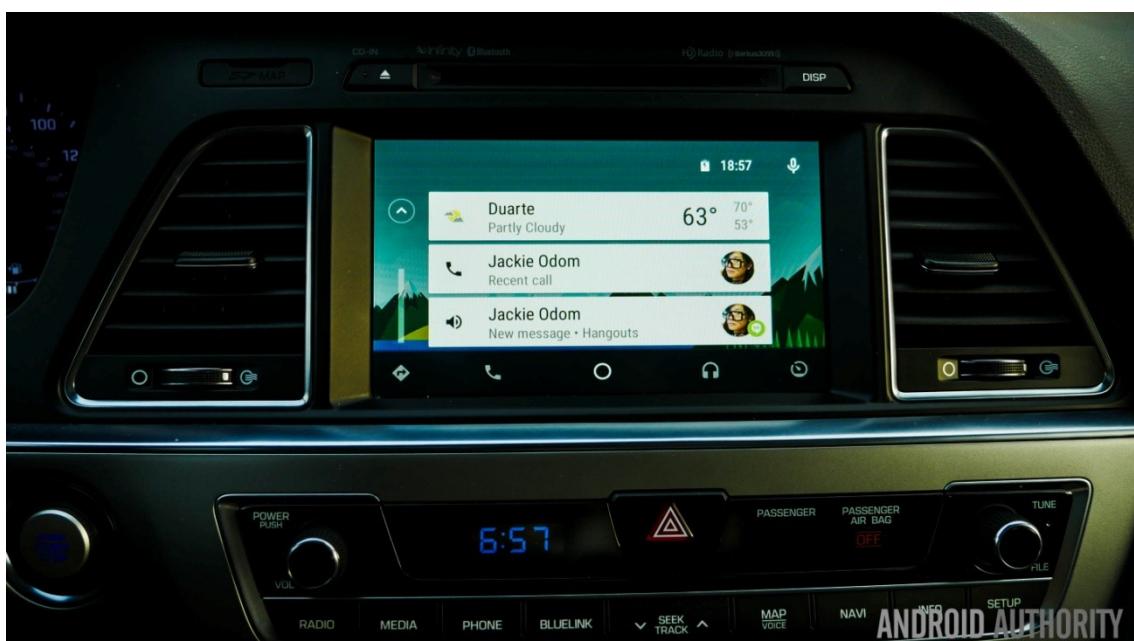
Sa *Android Auto* aplikacijom, mobilni uređaj vozača može pristupiti sledećim automobilskim ulazima i senzorima:

- *GPS* senzorima i visoko kvalitetnim *GPS* antenama;
- Kontrolama na volanu;
- Ozvučenju;
- Zvučnicima;

- Mikrofonu;
- Brzini vozila;
- Kompasu;
- Mobilnim antenama;
- Podacima automobila (u procesu razvijanja).

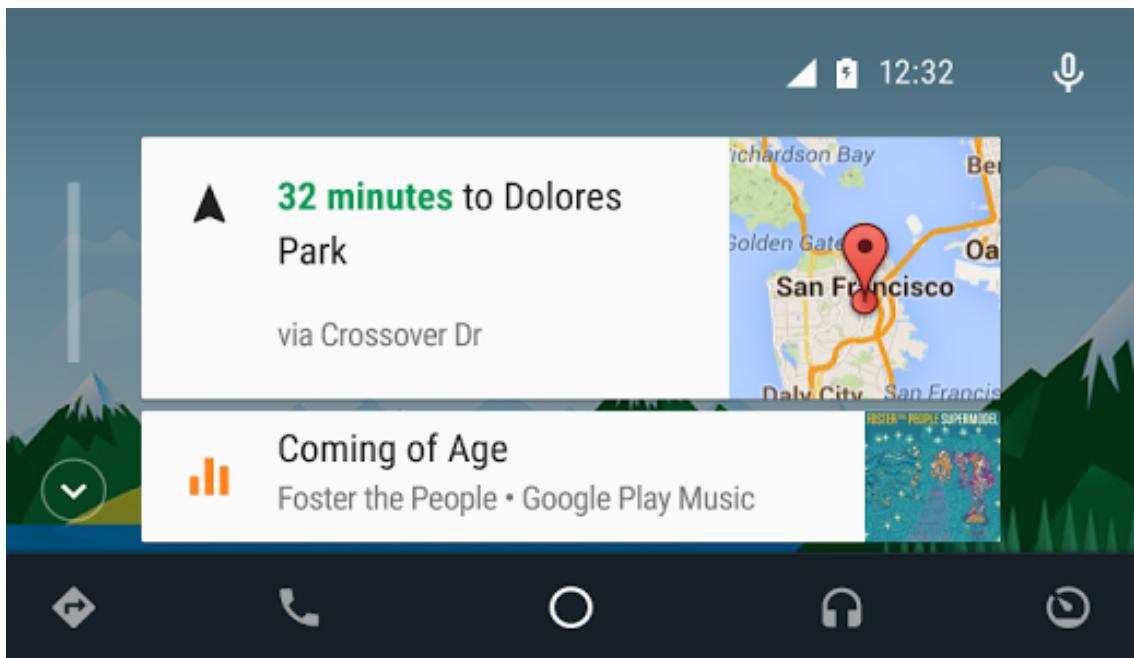
Proizvođači automobila koji podržavaju *Google Android Auto* su: *Audi, Volkswagen, Ford, Mercedes, Volvo*.

Na Slici 3 je prikazan izgled *Android Auto* aplikacije. Prikazana je univerzalna pozadina na kojoj se nalaze tri kartice koje prikazuju različit sadržaj.



Slika 3: Izgled *Android Auto* aplikacije na displeju *IVI* sistema u automobilu

Na Slici 4 je još jedan prikaz *Android Auto* aplikacije. Na prvoj kartici je prikaz navigacije, a na drugoj kartici je prikazana pjesma koja se trenutno sluša.



Slika 4: Izgled *Android Auto* aplikacije

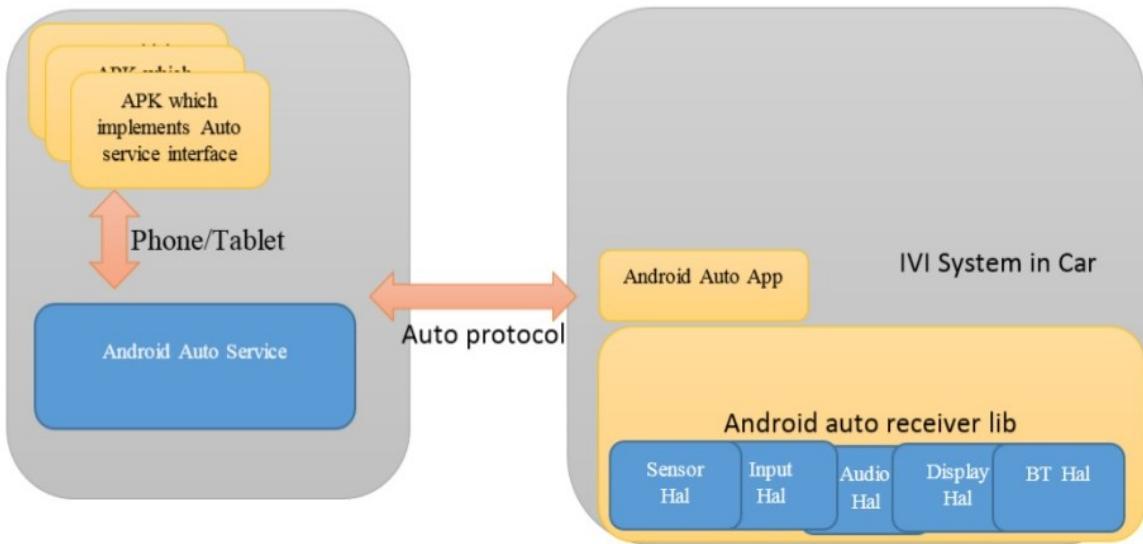
Na Slici 5 je prikazana arhitektura *Android Auto* softvera. Lijevi blok sa slike predstavlja *Android* telefon/tablet, na kome se nalaze:

- *Android Auto Service*;
- *Google Play Service* (nije prikazano na slici ali ulazi u sastav softvera);
- Aplikacije koje koriste funkcionalnosti *Android Auto Service* komponente;
- *Android Auto SDK* (takođe nije prikazano na slici).

Desni blok sa slike predstavlja *IVI* sistem u automobilima, u kom se nalazi *Android Auto Service* biblioteka u kojoj su implementirane sledeće hal (*Hardware Abstraction Layer*) komponente:

- Ulaz (eng. *input*);
- *Audio*;
- Senzor podaci;
- *Bluetooth*;
- Ekran.

Ova dva bloka povezuje *Auto protocol* koji je sačinjen od pet različitih kanala preko kojih telefon komunicira sa gore navedenim aspektima automobila.

Slika 5: Arhitektura *Android Auto* softvera

#### 2.4.2 Tehnologije otvorenog koda

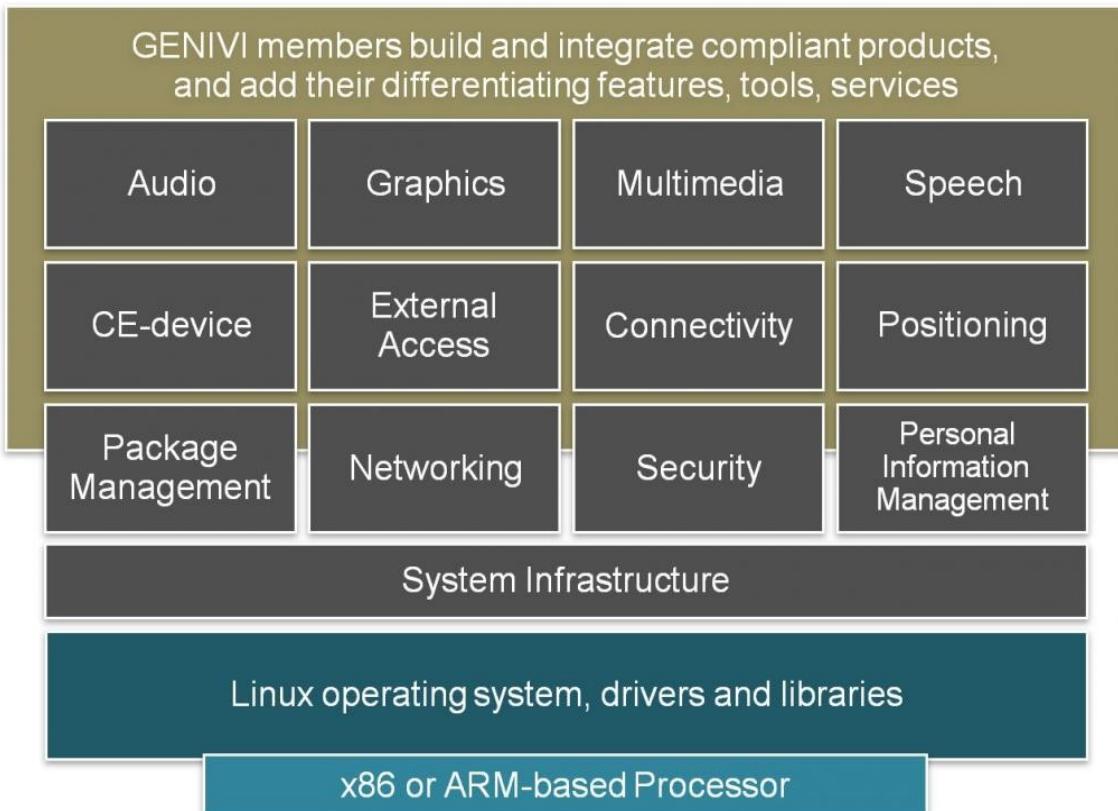
U ovom poglavlju će biti predstavljena tri projekta otvorenog koda: *GENIVI*, *Automotive Grade Linux (AGL)* i *Automotive Grade Android (AGA)*.

*GENIVI Alliance* [13] je neprofitni savez, koji se sastoji od preko 180 automobilskih kompanija, i koji razvija otvorenu i globalno konzistentnu platformu, zasnovanu na *Linux* operativnom sistemu, koju će koristiti cijela automobilska industrija za razvoj *IVI* sistema. Alijansa zapravo radi na usklađivanju automobilske industrije sa potrebama potrošača.

*GENIVI* nije kompletno *IVI* okruženje, to je platforma koja se sastoji od operativnog sistema (*Linux*) i *middleware* komponenti sa funkcionalnostima koje bi svi *IVI* sistemi trebalo da zahtijevaju. Ovo rješenje nudi proizvođačima sredstva koja im omogućavaju da se fokusiraju samo na razvoj viših slojeva arhitekture *IVI* sistema.

Što se tiče arhitekture softvera, *GENIVI* alijansa je koristila takozvani *V*-model pristupa razvoju softvera. Ovaj model razvoja podrazumijeva da se počinje od funkcionalnih zahtjeva, a zatim se vrši odabir ili razvoj komponenti kako bi se ispunili ti zahtjevi. Dobijena softverska arhitektura, nazvana specifikacija, je glavni 'proizvod' *GENIVI* saveza, koji se objavljuje na svakih 6 mjeseci. *GENIVI* softver se sastoji uglavnom od postojećih softverskih komponenti otvorenog koda, sa dodatkom softvera namijenjenom isključivo automobilskoj industriji.

Slika 6 prikazuje arhitekturu softvera otvorenog koda, koji je zasnovan na *Linux* operativnom sistemu, i u skladu je sa *GENIVI* specifikacijom [14]. Ovaj softver je usklađen sa *GENIVI* specifikacijom, a na njega su nadograđene različite alatke, usluge i opcije.



Slika 6: Arhitektura softvera koji je u skladu sa *GENIVI* specifikacijom

*Automotive Grade Linux (AGL)* [15] je radna grupa fokusirana na razvoj softverskog rješenja, koje će biti otvorenog koda, za razvoj aplikacija za automobile. Ciljevi *AGL* grupe su sledeći:

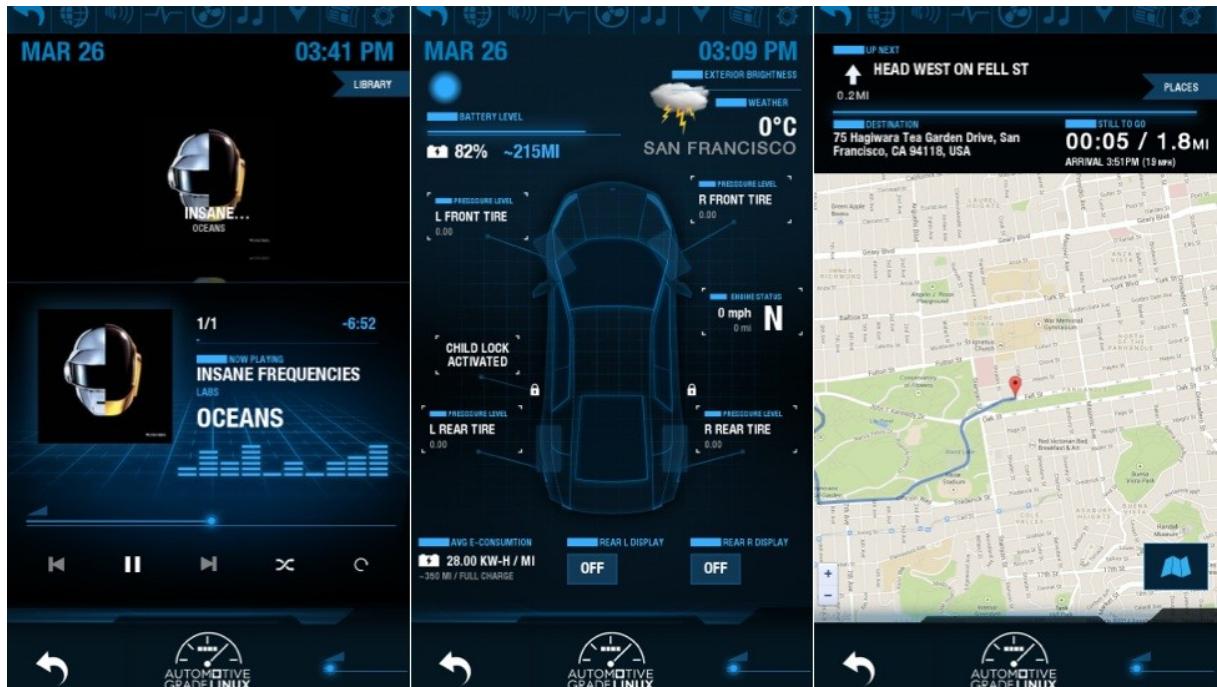
- Uvođenje *Linux* operativnog sistema u automobilsku industriju;
- Obezbeđivanje okruženja za razvoj *IVI* sistema;
- Omogućavanje vrlo brzog razvoja softvera za automobilsku industriju kako bi se održao korak sa zahtjevima potrošača za boljim *IVI* iskustvom.

*AGL* specifikacija definije arhitekturu *AGL* platforme. Glavni cilj je definisanje softverske platforme na kojoj se mogu graditi aplikacije. Kao takav, ovaj dokument ne definije zahtjeve za aplikacije, osim u slučaju početnog ekrana (eng. *Home Screen*). Specifikacije aplikacija razvijaju različiti projekti koji koriste *AGL* platformu za njihov razvoj, a koje će se kasnije koristiti za dodavanje novih ili prepravljanje već postojećih zahtjeva *AGL* platforme.

*AGL* specifikacija zahtjeva 1.0 predstavlja važnu prekretnicu u industriji na putu ka usvajanju *open source* metodologije razvoja, kako bi se ubrzale inovacije i stvaranje sigurnog i pouzdanog iskustva u automobilu. Specifikacija omogućava proizvođačima originalne

opreme (eng. *Original Equipment Manufacturers*) i dobavljačima da identifikuju praznine između specifikacije i koda i daju direktni doprinos programerskoj zajednici.

Na Slici 7 je prikazan izgled *AGL* aplikacije.



Slika 7: Izgled *AGL* aplikacije

*Automotive Grade Android (AGA)* [16] je softverska platforma otvorenog koda koja omogućava integraciju *Android* aplikacija u *IVI* sisteme. *AGA* pored kreiranja novih, omogućava prilagođavanje već postojećih aplikacija *IVI* sistemima, sa ciljem omogućavanja pristupa podacima o automobilu i bezbjedne komunikacije sa vozačima.

Na Slici 8 je prikazana jedna *AGA* aplikacija.



Slika 8: Primjer *AGA* aplikacije

### 2.4.3 Tehnologije u istraživanjima

Objavljeno je nekoliko istraživačkih radova u kojima je predstavljeno jedinstveno okruženje za programiranje aplikacija za *IVI* sisteme. U ovom poglavlju će biti opisana dva rješenja, *Webinos* i *Android4Auto*. *Webinos* [17] je predstavio i razvio, u okviru projekta, okruženje za programiranje aplikacija koje se zasniva na internetu, i sastoji se od tri glavne grupe:

- Podaci o vozilu;
- Navigacija;

- 
- Lociranje.

Integracija *Androida* i *IVI* sistema je predstavljena u *Android4Auto* radu [18]. U ovom radu je razvijeno okruženje koje omogućava aplikacijama da koriste sledeće:

- Radio;
- Navigaciju;
- Povezivanje;
- Klimu;
- Informacije o vozilu;
- Prepoznavanje glasa.

U *Webinos* projektu su razvijene dvije *JavaScript* biblioteke za interakciju sa vozilom, *Vehicle API*, za preuzimanje informacija o automobilu preko prenosnih sistema, i *Navigation API*, za interakciju sa navigacionim sistemom.

U okviru *Vehicle* biblioteke je omogućen pristup podacima vozila koji se često mijenjaju, i to pomoću registrovanja na događaje. Programeri aplikacija mogu registrovati takozvane slušaoce (eng. *listeners*) određenih događaja, ili mogu zahtijevati podatke o vozilu. Statične informacije o vozilu, kao što su brend, model, itd. su obezbijeđene kao odvojeni atributi objekta vozila. Sledećim podacima se može pristupiti iz *Vehicle* biblioteke, i to pomoću *addEventListener()* i *get()* funkcija: parking senzori (eng. *Parking Sensor Data*), podaci o poslednjoj vožnji (eng. *Tripcomputer Data*), podaci o brzinama (eng. *Gear Data*), osvjetljenje (eng. *Light Data*), ulje (eng. *Engine Oil Data*), brisači (eng. *Wiper Data*), prozori (eng. *Window Data*), vrata (eng. *Door Data*), pritisak u gumama (eng. *Tire Pressure Data*), klima (eng. *Climate Data*), sjedišta (eng. *Seat Data*). Ovi podaci se prosleđuju definisanoj *callback* funkciji (*callback* je dio izvršnog koda/funkcija koja se prosleđuje kao argument druge funkcije, od koje se očekuje da pozove taj argument u nekom pogodnom trenutku) koja se aktivira svaki put kada se neki od podataka promijene.

Kada aplikacija proslijedi odredište (*Point-of-Interest – POI*) navigacionom sistemu, može se registrovati *callback*. *Callback* rukovalac podržava mehanizam kojim obavještava aplikaciju o tome da li je destinacija uspješno proslijedena navigacionom sistemu, da je smjernica otkazana ili da je tražena destinacija dostignuta.

Kako bi omogućili pristup podacima vozila, kao i interakciju sa navigacionim sistemom, u ovom projektu se pristupa dvjema različitim magistralama:

- *MOST* magistrala (za medijski orijentisane transportne sisteme, eng. *Media Oriented Transport Systems*);

- *CAN* magistrala (standardna magistrala koja omogućava komunikaciju između mikrokontrolera i uređaja u aplikacijama, bez centralnog računara, eng. *Controller Area Network*).

Kada se pokrene *Web* aplikacija, internet pretraživač pokreće *JavaScript* biblioteku kako bi obezbijedio pristup podacima o vozilu. Biblioteka uspostavlja vezu sa *Vehicle Data Provider* slojem, koji sadrži *Vehicle Access Manager* dodatak za pristup magistralama, koristeći *WebSocket*.

U *Android4Auto* radu je predstavljena integracija *Android* operativnog sistema u *IVI* sisteme. *Android4Auto* sistem se sastoji od sledećih modula:

- *UI Application*;
- *Notification Service*;
- *Business Library*;
- *Framework Library*.

*UI Application* predstavlja *Android* aplikaciju koja koristi *3D* grafički okvir (eng. *framework*) za prikazivanje grafike. Ova aplikacija je zadužena za vizualizaciju korisnih informacija o osnovnim funkcionalnostima *IVI* sistema, i pokreće se kada se završi inicijalizacija (eng. *boot*) *Android* operativnog sistema.

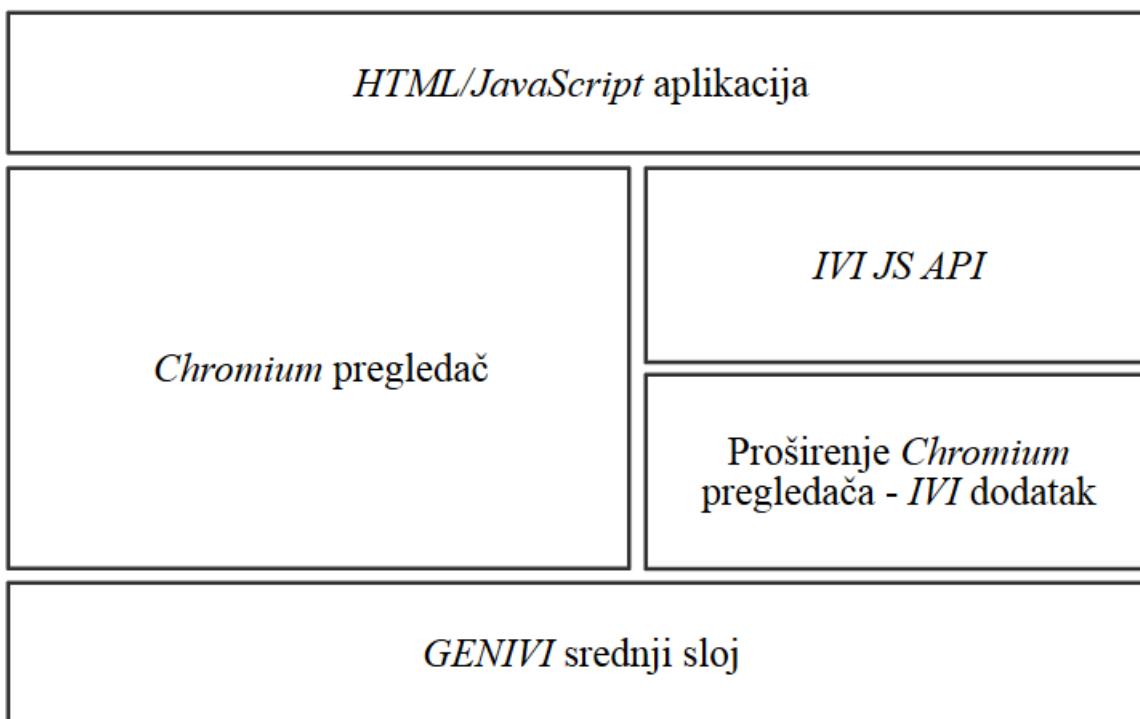
*Notification Service* je zadužen za prikazivanje obavještenja iz svih slojeva, koristeći *3D* grafički okvir.

*Business Library* predstavlja *Java* biblioteku koja sadrži logiku za aplikativni sloj. Na primjer, rukuje obavještenjima, prati aktivnosti sistema (na primjer ako dođe do telefonskog poziva dok je radio aktivan, ona će isključiti radio ili smanjiti zvuk), itd. Ova biblioteka omogućava vezu između aplikacije i *Framework Library* modula.

*Framework Library* predstavlja biblioteku koja omogućava gornjim slojevima arhitekture da se povežu sa svim raspoloživim *IVI* uslugama (miltimedija, radio, navigacija, itd.). Sve funkcije i klase se nalaze u ovoj biblioteci.

### 3. Koncept rješenja

Na Slici 9 je prikazana arhitektura rješenja, a detaljnije objašnjenje je dato u nastavku poglavlja.



Slika 9: Prikaz arhitekture rješenja

### 3.1 Proširenje *Chromium* pregledača

Plugin ili dodatak je softverska komponenta koja dodaje određenu funkcionalnost nekom programu [19]. Na primjer, internet pretraživači omogućavaju korisnicima da instaliraju dodatke kako bi im dodali funkcionalnosti koje se ne nalaze u njihovoј podrazumijevanoj instalaciji. Nakon što se plugin instalira, pretraživač ga automatski prepoznaće i njegova funkcionalnost se integriše u *HTML* fajl koji se prikazuje.

Aplikacije podržavaju dodatke iz više razloga:

1. Omogućavanje programerima da kreiraju funkcionalnosti kako bi proširili aplikaciju;
2. Lako dodavanje novih funkcionalnosti;
3. Smanjenje veličine aplikacije;
4. Razdvajanje izvornog koda od aplikacije zbog nekompatibilnih softverskih licenci.

Za proširenje *Chromium* internet pretraživača, u ovom rješenju je iskorišten dodatak (*plugin*), jer on predstavlja jedini način komunikacije JS koda sa nižim slojevima ove arhitekture (*HMI* agent koji šalje podatke dodatku). U nastavku je detaljnije opisana arhitektura ovog rješenja.

### 3.2 Opis rješenja

Rješenje predstavljeno u ovom radu zasniva se na internet pretraživaču kao aplikativnom okruženju. Koristi se *Chromium* pretraživač koji se izvršava na *GENIVI* srednjem sloju (eng. *middleware*). U sledećem pasusu je objašnjeno zašto se u ovom rješenju koristi baš *Chromium*.

Postojeće *GENIVI* rješenje koristi *QTWebKit* mašinu za internet pretraživač (eng. *web browser engine*), ali ta tehnologija je zastarjela i potrebno ju je zamijeniti. Pošto se trendovi na tržištu u poslednje vrijeme oslanjaju na *Chromium*, nekoliko rješenja, koja ga koriste, je analizirano na *GENIVI AMM* sastanku u aprilu 2015.godine. *Chromium*, *Crosswalk* (biblioteka za *HTML* aplikacije, zasnovana na *Chromium* kodu), *QTWebEngine* i *CEF* (jednostavan okvir za ugradnju pretraživača zasnovanih na *Chromium* internet pretraživaču u druge aplikacije) su međusobno upoređeni, a rezultati pokazuju da je *Chromium* najkompletnije rješenje u pogledu funkcionalnosti internet pretraživača [20].

Pretraživač je proširen *IVI* dodatkom, kojim se ostvaruje komunikacija između nižih (*HMI* agent u ovom slučaju) i viših (*JS* sloj) slojeva arhitekture, kako bi informacije o vozilu bile dostupne iz *JavaScript* programske jezike. Put koji informacije pređu da bi stigle do korisničke aplikacije je sledeći:

- Najprije *HMI* agent proslijedi određenu informaciju *IVI* dodatku;
- *IVI* dodatak tu informaciju dalje, funkcijom *PostMessage*, prosleđuje *IVI JS* sloju;
- *IVI JS* sloj, funkcijom *HandleMessage*, prima poruku od dodatka, a zatim primljeni podatak prosleđuje korisničkoj aplikaciji;

*IVI* dodatak je realizovan kao *PPAPI* dodatak, zbog toga što se koristi *Chromium* pretraživač kod koga su *NPAPI* dodaci izbačeni iz upotrebe, jer su zastarjeli i predstavljaju glavni izvor krahova i sigurnosnih problema, a takođe utiču i na kompleksnost koda.

Dodatni *JS* sloj između dodatka i korisničke *HTML/JS* aplikacije je *IVI JS API*, koji predstavlja *JS* biblioteku preko koje korisnička aplikacija komunicira sa dodatkom. Ovaj dodatak je realizovan jer se na taj način olakšava razvoj aplikacija koje koriste API opisan u ovom radu što je i jedan od glavnih ciljeva ovog rješenja. Dakle, da bi napisali aplikaciju za *IVI* sistem, programeri će za dobijanje potrebnih informacija o automobilu koristiti *IVI JS API* koji je definisan tako da se uklapa u standardni način upotrebe *JavaScripta*.

Treba napomenuti da je moguć pristup podacima vozila bez *IVI JS* biblioteke, koristeći samo dodatak, ali sistem razmjene poruka koje koristi dodatak nije nešto što je intuitivno za *JS* programere.

U tabeli 1 je dat pregled svih *API* grupa i informacija koje im pripadaju.

Naziv klase dogadaja	Reprezentativne informacije
PEDALS	Status papučice za gas Status papučice za kočnicu
STEERING WHEEL	Dugme pušteno Dugme zadržano Dugme pritisnuto
IVI	Telefonski poziv Muzika Trajanje telefonskog poziva
CAR_SIMULATION	Brzina [kmh/mph] Broj obrtaja Brzina (1, 2, P, R, N, D) Temperatura motora [°C] Nivo goriva Trenutna potrošnja goriva [l] Spoljašnja temperatura [°C] Broj km/mi koje je moguće preći u odnosu na trenutnu potrošnju Ukupan broj pređenih km/mi Broj pređenih km/mi tokom trenutne vožnje Nadmorska visina [km/mi]
APP	Profil Upozorenje na nizak nivo goriva Upozorenje da nešto ne funkcioniše
Ostalo	
Average fuel consumption	
Fuel info	Nivo goriva Trenutna potrošnja Broj km/mi koje je moguće preći u odnosu na trenutnu potrošnju

Tabela 2: Pregled API grupa i informacija koje im pripadaju

## 4. Programsko rješenje

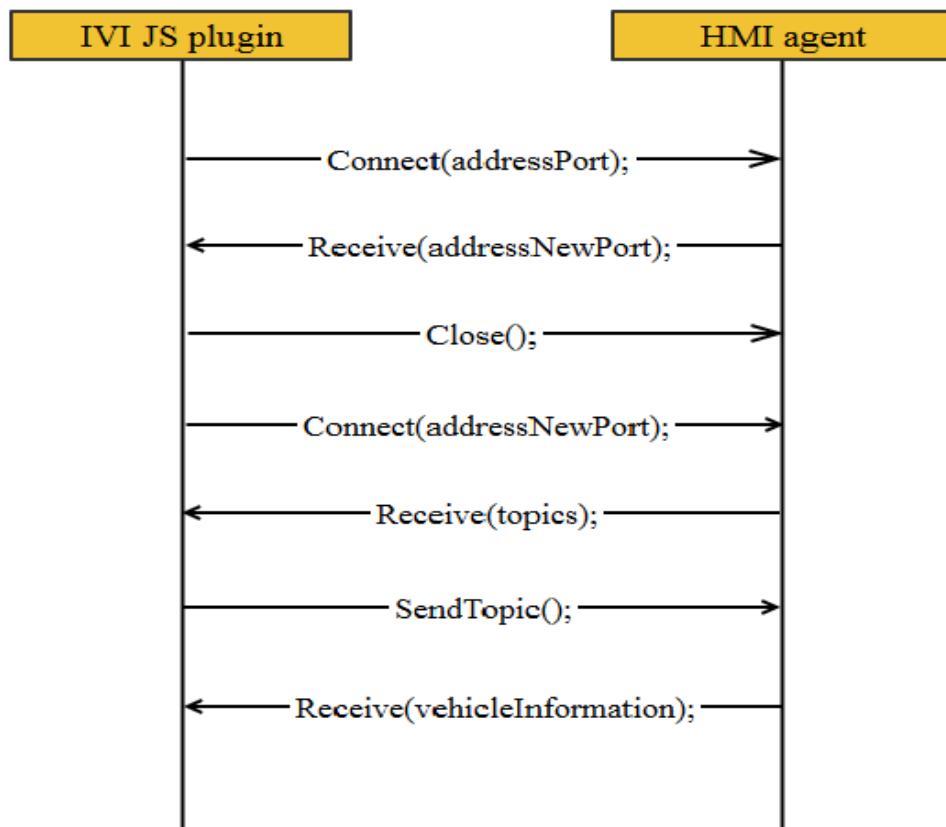
U ovom poglavlju će biti detaljno objašnjeni svi dijelovi arhitekture prikazane na Slici 9. Ovo rješenje se sastoji od JS biblioteke sa aplikacionom programskom spregom kojom se omogućuje HTML5 aplikacijama, koje se izvršavaju u sklopu aplikativnog okruženja u vozilu, da pristupaju sistemima vozila. Kao podrška biblioteci realizovan je dodatak za Chromium pregledač.

### 4.1 IVI JS plugin

U ovom modulu se postavlja inicijalna konfiguracija/stanje vozila, koja se šalje *IVI JS* biblioteci, i ostvaruje se *TCP* veza sa *HMI* agentom od kog dodatak dobija sve potrebne informacije, koje dalje prosljeđuje *IVI JS* biblioteci. Redoslijed operacija je sledeći:

- Ostvaruje se povezivanje na broj porta na kom se nalazi *HMI* agent (broj porta je unaprijed zadat);
- Kada je veza ostvarena, dodatak prima od agenta novi broj porta na koji će se povezati, i koji je samo njemu poznat;
- Zatvara se veza;
- Uspostavlja se veza na novom portu;
- Dodatak prima od agenta raspoložive događaje (u daljem tekstu topik) na koje se može registrovati;
- Dodatak šalje broj topika na koji se želi registrovati (u ovom slučaju se registruje na sve topike jer želi da dobija sve raspoložive informacije). Nakon ovog koraka dodatak kontinualno prima događaje od *HMI* agenta i prosljeđuje ih *IVI JS API*.

Opisani redoslijed operacija je prikazan na Slici 10.



Slika 10: Postupak ostvarivanja TCP veze sa HMI agentom

Ono što je takođe specifično za dodatak je način komunikacije sa IVI JS API slojem. Komunikacija između *JS* programskog jezika i *C/C++* koda se ostvaruje pomoću sistema za razmjenjivanje poruka. Sledeće dvije funkcije omogućavaju prijem i slanje poruka između *IVI* dodatka i *IVI JS API* sloja:

***HandleMessage*** – ovo je funkcija za čitanje primljene poruke. Kada se iz *JS* biblioteke pošalje poruka dodatku (i obrnuto) u njemu se automatski poziva ova funkcija. Primjer upotrebe funkcije:

*HandleMessage (message\_event);* – *message\_event* je istog tipa kao primljena poruka.

**PostMessage** – na sličan način poruka poslata JS sloju od strane dodatka, pomoću ove funkcije, aktivira događaj koji čeka poruke (eng. *event listener*). Primjer upotrebe ove funkcije:

`PostMessage(config);` – pri čemu je promjenljiva config tipa *VarDictionary*. Na ovaj način se šalje poruka iz dodatka JS sloju.

Osnovni oblik poruke je string, ali su podržani mnogi JS tipovi među kojima su cijeli brojevi, nizovi, baferi i takozvani rječnici (eng. *VarDictionary*).

## 4.2 IVI JS API

U JS biblioteci je realizovan IVI interfejs za dobijanje podataka o vozilu.

Biblioteka je tako definisana da odgovara standardnoj upotrebi JS programskog jezika, kako bi razvoj aplikacija koje koriste ovaj API bio vrlo jednostavan. Podacima o vozilu je moguće pristupiti bez ove biblioteke, samo preko dodatka, ali sistem razmjene poruka između dodatka (C/C++ koda) i JS koda nije intuitivan za JS programere.

U ovom modulu su definisane dvije grupe funkcija:

- Funkcije koje, kada modul od dodatka dobije nove vrijednosti za određene promjenljive, vraćaju te ažurirane vrijednosti promjenljivih, kako bi one u svakom trenutku bile dostupne JS aplikaciji. U aplikaciji je potrebno samo se registrovati na željene događaje (eng. *event*), odnosno postaviti *eventListener* na svaki od tih događaja;
- Funkcije koje vraćaju trenutnu vrijednost tražene promjenljive. Potrebno je samo pozvati funkciju iz aplikacije kako bi se dobili određeni podaci.

**onSpeedChanged** – vraća brzinu JS aplikaciji, ukoliko je aplikacija registrovana na čekanje ovog događaja, svaki put kada se desi promjena. Brzina se izražava i u km/h i u mph. Primjer upotrebe funkcije:

`ivy.onSpeedChanged(speedChanged);` – pri čemu je *speedChanged* promjenljiva tipa *var*.

**onRpmChanged** – ažurira rpm vrijednost – broj obrtaja u minuti. Primjer upotrebe funkcije:

---

`ivy.onRpmChanged(rpmChanged);` – pri čemu je `rpmChanged` promjenljiva tipa `var`.

**`onEngineTemperatureChanged`** – ova funkcija javlja aplikaciji da se promijenila vrijednost temperature motora. Vraća vrijednost u stepenima Celzijusovim. Primjer upotrebe funkcije u JS aplikaciji je sledeća:

`ivy.onEngineTemperatureChanged(engineTemperatureChanged);` –  
engineTemperatureChanged je promjenljiva tipa `var`.

**`onThrottleStatusChanged`** – javlja JS aplikaciji kada se promijeni pozicija pedale gasa. Ukoliko je pedala pritisnuta ova funkcija vraća broj procenata u skladu sa jačinom pritiska pedale, a kada je pedala oslobođena vraća nulu. Upotreba ove funkcije kao i funkcija ispod, dok se ne navede suprotno, je ista kao u prethodna tri slučaja, pa neće biti navođena u daljem tekstu rada.

**`onBrakeStatusChanged`** – radi isto kao prethodna funkcija, ali za pedalu kočnice. Primjer registrovanja na ovaj događaj iz JS aplikacije:

`ivy.onBrakeStatusChanged(brakeStatus);` – promjenljiva `brakeStatus` je tipa `var`.

**`onGearIndicatorChanged`** – funkcija objavljuje događaj svaki put kada se promijeni režim. Vrijednosti koje funkcija vraća, odnosno režimi u kojima automobil može biti, su sledeće:

- Broj brzine ukoliko su u pitanju prva i druga brzina;
- Slovo P (eng. *parking*) ukoliko je automobil u parking režimu;
- Slovo R (eng. *reverse* – rikverc) kada automobil ide unazad;
- Slovo N (eng. *neutral*) kada je automobil u neutralnom režimu i
- Slovo D (eng. *drive mode*) kada je automobil u režimu vožnje.

**`onFuelInfoChanged`** – daje informacije o stanju goriva. Ova funkcija obezbjeđuje informacije o nivou goriva u rezervoaru, trenutnoj potrošnji goriva i broju kilometara (i milja) koje automobil može preći sa trenutnom količinom goriva.

**`onFuelTankLevelChanged`** – funkcija prosleđuje aplikaciji informaciju o nivou goriva u rezervoaru, svaki put kada dođe do promjene.

***onCurrentFuelConsumption*** – ukoliko se JS aplikacija registruje za čekanje događaja promjene trenutne potrošnje goriva, tu informaciju će dobiti od ove funkcije. Ova informacija je izražena u litrima.

***onRangeChanged*** – registrovanjem na čekanje događaja promjene broja kilometara/milja koje automobil može preći sa trenutnom potrošnjom goriva, JS aplikacija će dobiti tu informaciju pomoću ove funkcije.

***onAverageFuelConsumption*** – ova funkcija ažurira vrijednost pri svakoj promjeni prosječne potrošnje goriva. Ova vrijednost se izražava u litrima na 100km.

***onOutsideTemperatureChanged*** – funkcija vraća JS aplikaciji vrijednost temperature u stepenima Celzijusovim.

***onMileageChanged*** – funkcija ažurira ukupan broj kilometara koje je automobil prešao.

***onKeyDown*** – ukoliko je pritisnuto dugme na ručnom kontroleru (eng. *jog shuttle*) IVI JS API će objaviti događaj kojim obavještava JS aplikaciju da je dugme pritisnuto, i prosleđuje joj broj koji označava koje dugme je pritisnuto (eng. *key code*).

***onKeyUp*** – IVI JS API objavljuje događaj kojim obavještava JS aplikaciju da je pritisnuto dugme pušteno. Ova funkcija takođe prosleđuje i broj kojim je dugme označeno.

***onKeyPressed*** – kada je dugme na ručnom kontroleru pritisnuto, JS API objavljuje događaje dok god je dugme pritisnuto. Kao i u prethodna dva slučaja, JS API uz informaciju da je dugme pritisnuto, prosleđuje broj kojim je označeno pritisnuto dugme.

***onDriverProfileChanged*** – prilikom svake promjene profila vozača korisnička aplikacija dobija informaciju o tome. JS API prosleđuje identifikacioni broj profila i ime vozača.

***onMusicChanged*** – pri promjeni pjesme ili radio stanice, JS API obavještava korisničku aplikaciju o tome, i prosleđuje joj naziv nove pjesme ili radio stanice.

---

**onPhoneCall** – registrovanjem na čekanje događaja vezanih za telefon korisnička aplikacija će biti obavještena o svakom dolaznom i odlaznom pozivu, što podrazumijeva informaciju o pozivaocu, i trajanju poziva.

**onMalfunction** – ova funkcija obezbjeđuje informacije o određenim kvarovima i nepravilnostima, prosleđujući broj koji ih predstavlja. Tipičan primjer situacije koja će prouzrokovati aktiviranje ove funkcije, tj. objavljivanje događaja, je nizak nivo goriva u rezervoaru.

**onLowFuel** – kada nivo goriva padne ispod 1/6 rezervoara JS API objavljuje događaj kojim obavještava korisničku aplikaciju o tome.

**onTripMileageChange** – funkcija objavljuje broj pređenih kilometara od trenutka kada je vozilo upaljeno.

**onAltitudeChange** – registrovanjem na čekanje ovog događaja, korisnička aplikacija dobija od *JS API* informaciju o nadmorskoj visini na kojoj se automobil trenutno nalazi.

**getSpeed** – funkcija vraća brzinu automobila, u kilometrima i miljama, u trenutku kada je pozvana. Povratna vrijednost je objekat koji sadrži u sebi brzinu u kilometrima i miljama. Primjer poziva ove funkcije iz korisničke aplikacije:

```
speed = ivy.getSpeed();
```

Primjer inicijalizacije brzine u *JS* biblioteci je sledeći:

```
ivy.speed = {speed.kmh: 0, speed.mph: 0};
```

**getRpm** – pozivom ove funkcije dobija se trenutan broj obrtaja u minuti. Vrijednost koju funkcija vraća je cijeli broj (*integer*). Primjer poziva iz korisničke aplikacije:

```
rpm = ivy.getRpm();
```

**getEngineTemperature** – ova funkcija vraća trenutnu temperaturu motora izraženu u stepenima Celzijusovim. Povratna vrijednost je cjelobrojna. Primjer poziva je sledeći:

---

```
engineTemperature = ivy.getEngineTemperature;
```

**getThrottlePosition** – funkcija vraća 1 ukoliko je pedala gasa trenutno pritisnuta, a 0 ukoliko nije. Primjer poziva ove funkcije iz korisničke aplikacije:

```
throttlePosition = ivy.getThrottlePosition();
```

**getPedalStatus** – funkcija je identična prethodnoj, s tim da se sada radi o pedali koja može biti kočnica ili gas. Primjer korištenja ove funkcije je sledeći:

```
pedalStatus = ivy.getPedalStatus();
```

**getGearInfo** – vraća trenutnu brzinu u kojoj se nalazi automobil. Povratna vrijednost je cjelobrojna. Primjer poziva:

```
gear = ivy.getGearInfo();
```

**getFuelInfo** – vraća trenutne informacije o nivou goriva u rezervoaru, trenutnoj potrošnji i broju kilometara/milja koje automobil može preći u odnosu na nivo goriva i trenutnu potrošnju. Povratna vrijednost je objekat *fuelInfo* koji ima četiri polja:

- nivo goriva u rezervoaru (vrijednost izražena u litrima);
- trenutna potrošnja goriva (vrijednost izražena u litrima na 100km);
- broj preostalih kilometara koje automobil može preći;
- broj preostalih milja koje automobil može preći.

Inicijalizacija objekta je izvršena na sledeći način:

```
ivy.fuelInfo = {fuelTankLevel: 0, currentFuelConsumption: 0, range: 0, rangeMph: 0};
```

Poljima objekta se pristupa ovako:

```
ivy.fuelInfo.fuelTankLevel.
```

Primjer poziva funkcije iz korisničke aplikacije:

---

```
fuelInfo = ivy.getFuelInfo();
```

**getFuelTankLevel** – vraća informaciju o trenutnom nivou goriva u rezervoaru. Povratna vrijednost je cjelobrojna. S obzirom da se sve naredne funkcije pozivaju iz korisničke aplikacije na isti način, primjer poziva više neće biti navođen.

**getCurrentFuelConsumption** – funkcija vraća trenutnu potrošnju goriva izraženu u broju litara na 100km. Povratna vrijednost ove funkcije je cjelobrojna.

**getAverageFuelConsumption** – ova funkcija vraća trenutnu vrijednost prosječne potrošnje goriva. Povratna vrijednost funkcije je takođe cjelobrojna.

**getOutsideTemperature** – funkcija vraća trenutnu vrijednost temperature spoljašnjeg vazduha izraženu u stepenima Celzijusovim. Funkcija vraća cjelobrojnu vrijednost.

**getMileage** – pozivom ove funkcije dobija se vrijednost trenutnog broja pređenih kilometara. Povratna vrijednost funkcije je objekat koji ima dva polja:

- ukupan broj pređenih kilometara;
- ukupan broj pređenih milja.

Inicijalizacija objekta i pristupanje njegovim poljima se vrši na isti način kao u prethodnim slučajevima.

**getDriverProfile** – povratna vrijednost ove funkcije je objekat sa dva polja:

- ime vozača;
- identifikacioni broj vozača.

Objavljivanje događaja se vrši pomoću funkcije *dispatch*, koja prima naziv događaja i novu vrijednost određene promjenljive (ove vrijednosti modul prima od dodatka funkcijom HandleMessage). Primjer rukovanja porukama od dodatka u okviru kog se nalazi i poziv dispatch funkcije je sledeći:

```
case VehicleEvent.SPEED_CHANGED:
    if(DEBUG) {
        console.log("Current speed is " +
message_event.data.value + " KMH.");
    }
    ivy.speed.kmh = message_event.data.value;
```

```
ivy.speed.mph = ivy.speed.kmh * KMHtoMPH;  
ivy.dispatch(VehicleEvent.SPEED_CHANGED, ivy.speed);  
break;
```

## 5. Ispitivanje

Za potrebe ispitivanja napravljena je *JavaScript* aplikacija koja koristi funkcije *IVI JS* biblioteke. Na taj način je provjeren rad *IVI* dodatka i *IVI JS API*, kao i njihova međusobna komunikacija. Provjeren je rad svih funkcionalnosti koje obezbeđuje JS API.

### 5.1 Opis okruženja za ispitivanje

Za testiranje rješenja opisanog u ovom radu (u toku razvoja) je korištena jednostavna *HTML* aplikacija koja sadrži funkcije registrovane na čekanje događaja, kao i funkcije za dobavljanje trenutne vrijednosti tražene promjenljive kako bi se provjerilo da li su vrijednosti koje aplikacija dobija od biblioteke tačne. U poslednoj fazi razvoja ovo rješenje je korišteno, a samim tim i testirano, u *HTML* klaster aplikaciji.

Za testiranje klaster aplikacije je korišten ekran sa 1920x720 piksela sa adapterom od 12 volti (S123WU01 12.3 *HDMI*). Računar koji pokreće aplikaciju u *Chromium* internet pretraživaču, koji se izvršava na *Yocto* distribuciji *Linux* operativnog sistema, je *Fujitsu*, sa procesorom i7 i 8G *CPU/RAM*.

Demonstrator je sačinjen od volana, ručnog kontrolera (eng. *Jog shuttle*) i pedala. Volkswagen volan je povezan na Arduino mikrokontroler i preko njega šalje podatke HTTP agentu od kog dodatak dobija sve potrebne informacije, koje se dalje prosleđuju klaster aplikaciji. Pedale, marke Fanatec, u koje je takođe ugrađen Arduino mikrokontroler imaju princip funkcionisanja i prenosa podataka sličan volanu. Što se tiče ručnog kontrolera, on je takođe marke Volkswagen. Slično kao volan, i ručni kontroler ima svoj Arduino

mikrokontroler preko kog se vrši prenos događaja i podataka, preko HMI agenta, do IVI dodatka, i na kraju, preko dodatka do IVI ekrana.

Na Slici 11 je prikazano okruženje za testiranje.



Slika 11: Okruženje za testiranje izrađeno tako da predstavlja unutrašnjost automobila

Okruženje za testiranje je izrađeno tako da što realnije predstavi atmosferu unutar automobila. U lijevom dijelu kontrolne table, ispred vozača, se nalazi klaster ekran na kom je pokrenuta klaster aplikacija, a na sredini table, između vozača i suvozača, je IVI ekran koji je osjetljiv na dodir i kojim se uglavnom upravlja pomoću ručnog kontrolera. Volan je postavljen tako da ne smanjuje vidljivost klaster ekrana kada je vozač u sjedećem položaju, a ispod volana su pedale. Pomenuti računar i sva potrebna napajanja se nalaze ispod 'haube' demonstratora.

## 5.2 Ključni slučajevi upotrebe

U ovom poglavlju će biti predstavljeno nekoliko ključnih slučajeva upotrebe (eng. *use cases*) rješenja opisanog u ovom radu, a u narednom poglavlju je opisano ispitivanje performansi predstavljenih slučajeva upotrebe.

Jedan od zadataka koje je *IVI* dodatak trebalo da ispuni je ažuriranje vrijednosti brzine i broja obrtaja (*rpm*) srazmjerno jačini stiska pedale za gas i kočnice. Na slikama 12 i 13 je

prikazana promjena vrijednosti brzine na brzinometru klaster ekrana. Sa slike 12 se vidi da se brzina znatno povećala sa pritiskom pedale gasa.



Slika 12: Početna brzina je 0km/h – automobil stoji



Slika 13: Brzina se povećala na 100km/h kao posledica pritiska pedale gasa

Na Slici 14 su prikazane informacije o gorivu, što predstavlja zaseban slušaj upotrebe. Na slici se vidi da ovaj slučaj upotrebe podrazumijeva da *IVI* dodatak treba da obezbijedi informaciju o nivou goriva u rezervoaru automobila, informaciju o trenutnoj potrošnji goriva, kao i informaciju o broju kilometara koje automobil može da pređe sa trenutnim nivoom goriva. Dakle, u odnosu na trenutnu potrošnju goriva, na slici, koja iznosi 10 l/100km, broj preostalih kilometara koje je moguće preći je 450.



Slika 14: Prikaz trenutne potrošnje goriva i broja preostalih kilometara

### 5.3 Ispitivanje performansi

Performanse rješenja su ispitane mjerjenjem vremena koje protekne od pritiska papučice za gas do trenutka kada se na brzinometru promijeni brzina (test 1), i mjerjenjem vremena propagacije komande kroz dodatak, kako bi se znalo koliko rješenje predstavljeno u ovom radu produžava vrijeme čekanja (test 2). Ispitivanje vremena propagacije komande kroz dodatak je obavljeno pozivanjem nekoliko različitih funkcija koje vraćaju različite tipove podataka kao povratnu vrijednost. Osim mjerjenja propagacije komande kroz dodatak, ovim ispitivanjem (test 2) je provjereno da li složenost povratne vrijednosti utiče na brzinu. Izvršena su dva ispitivanja kako bi krajnji rezultat pokazao koliko procenata ukupnog vremena, što je izmjereno u prvom testu, uzme dodatak. Oba mjerena su ponovljena po 30 puta, a kao krajnji rezultat je uzeta srednja vrijednost dobijenih vremena.

Primjer poziva funkcija:

```
var rpm = ivy.getRpm();
console.log("RPM: " + ivy.rpm + "\n");

var speed = ivy.getSpeed();
console.log("Speed kmh: " + speed.kmh + " mph: " + speed.mph);

var fuelInfo = ivy.getFuelInfo();
console.log("Fuel tank level: " + ivy.fuelInfo.fuelTankLevel +
"\n" +
"Current fuel consumption: " + ivy.fuelInfo.currentConsumption +
"\n" +
"Range in KMH: " + ivy.fuelInfo.range + "\n" +
"Range in MPH: " + ivy.fuelInfo.rangeMph + "\n");
```

Do rezultata prvog ispitivanja se došlo tako što je telefonom sniman klaster, od trenutka pritiska pedale gasa, a zatim je snimak ubačen u program koji ga je izdijelio na frejmove. Frejmovi su pretvoreni u ms pomoću sledeće formule:

$$\text{broj\_frejmova} * 1/25 * 1000,$$

pri čemu  $1/25$  predstavlja broj frejmova u sekundi. Prosječna izmjerena vrijednost je 7 frejmova, odnosno 280ms.

Rezultati testnih slučajeva, testa u kom se mjerilo vrijeme propagacije komande kroz dodatak, su prikazani u tabeli 3.

Testni slučajevi		
Tip informacije koja se šalje iz JS-a	Tip informacije koja se šalje iz IVI dodatka	Prosječno izmjereno vrijeme [ms]
string	integer	7.5
string	Speed	7.5
string	fuelInfo	8

Tabela 3: Rezultati testnih slučajeva

Iz tabele 2 se vidi da se vrijeme odziva ne povećava značajno (vrijeme odziva varira između 7.5 i 8.5 ms) sa povećanjem kompleksnosti povratne informacije, tj. tabela pokazuje da vrijeme odziva ne zavisi od složenosti informacije koju vraća dodatak.

Izvršena testiranja pokazuju da je za propagaciju komande kroz dodatak potrebno 2.7% ukupnog vremena, što svjedoči o primjenljivosti rješenja.

## 6. Zaključak

U ovom radu je predstavljeno jedno rješenje realizacije aplikacione programske sprege za rukovanje sistemima u vozilu u aplikativnim okruženjima zasnovanim na *HTML5*. Ovo rješenje predstavlja jedinstven *API* sloj koji sve neophodne interfejse automobila izlaže *IVI* jedinicama.

Da bi se olakšao razvoj aplikacija *JS* programerima, rješenje je realizovano kao *JS* biblioteka čija je podrška dodatak za *Chromium* pretraživač.

Ispitane su performanse rješenja, a rezultati su pokazali da je ovo rješenje primjenljivo, te da ne daje vidljivo prekoračenje u smislu dodatnog vremena potrebnog za izvršavanje funkcija koje dobavljaju potrebne informacije o automobilu.

Dalji pravci razvoja podrazumijevaju proširenje funkcionalnosti *JS* biblioteke, dodavanjem različitih funkcionalnosti, koje će omogućiti dobavljanje informacija o automobilu, koje za sada nisu realizovane, kao na primjer pritisak u gumama, informacije o klima uređaju, poziciji sjedišta, vratima, poziciji prozora, brisača itd.

## 7. Literatura

- [1] <https://www.techopedia.com/definition/27778/in-vehicle-infotainment-ivi> avgust 2016.
- [2] [https://en.wikipedia.org/wiki/CAN\\_bus](https://en.wikipedia.org/wiki/CAN_bus) avgust 2016.
- [3] [https://en.wikipedia.org/wiki/Local\\_Interconnect\\_Network](https://en.wikipedia.org/wiki/Local_Interconnect_Network) avgust 2016.
- [4] [https://en.wikipedia.org/wiki/MOST\\_Bus](https://en.wikipedia.org/wiki/MOST_Bus) avgust 2016.
- [5] <http://www.ni.com/white-paper/3352/en/> avgust 2016.
- [6] <https://www.reference.com/technology/technology-eebf0fb1e023190a> avgust 2016.
- [7] <https://en.wikipedia.org/wiki/JavaScript> avgust 2016.
- [8] [https://en.wikipedia.org/wiki/Cascading\\_Style\\_Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) avgust 2016.
- [9] <https://sr.wikipedia.org/wiki/HTML> avgust 2016.
- [10] <http://www.techradar.com/news/car-tech/apple-carplay-everything-you-need-to-know-about-ios-in-the-car-1230381> avgust 2016.
- [11] <http://www.theglobeandmail.com/globe-drive/auto-shows/north-america/prepare-for-a-future-full-of-giant-screens-in-cars/article28180754/> avgust 2016.
- [12] [https://en.wikipedia.org/wiki/Android\\_Auto](https://en.wikipedia.org/wiki/Android_Auto) avgust 2016.
- [13] <https://www.genivi.org/about-genivi> avgust 2016.
- [14] <https://www.semiwiki.com/forum/content/4618-socs-new-context-look-beyond-ppa-%C2%96-part2.html> avgust 2016.
- [15] <https://www.automotivelinux.org/> avgust 2016.
- [16] <https://developer.lindholmen.se/redmine/projects/aga/wiki> avgust 2016.
- [17] S. Isenberg, W. Haberl, M. Goebel, M. Michel and U. Baumgarten, „Enabling Rich Web Applications for In-Vehicle Infotainment“, Web and automotive W3C workshop

- [18] B. Kovacevic, M. Kovacevic, T. Maruna, D. Rapic, “Android4Auto: A proposal for integration of Android in vehicle infotainment systems”, 2016 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 7-11 January, 2016.
- [19] [https://en.wikipedia.org/wiki/Plug-in\\_\(computing\)](https://en.wikipedia.org/wiki/Plug-in_(computing)) avgust 2016.
- [20] <https://at.projects.genivi.org/wiki/display/PROJ/Browser+Technology+Evaluation> avgust 2016.