



**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: Горан Пеић Бибић
Број индекса: РА 167-2011**

Тема рада: Једно решење реализације силуматора догађаја у аутомобилу

Ментор рада: Проф. др Небојша Пјевалица

Нови Сад, мај, 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:	Горан Пеић Бибић
Ментор, МН:	Проф. др Небојша Пјевалица
Наслов рада, НР:	Једно решење реализације симулатора догађаја у аутомобилу.
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)	7/21/0/3/9/0/0
Научна област, НО:	Електротехника и рачунарство
Научна дисциплина, НД:	Рачунарска техника
Предметна одредница/Кључне речи, ПО:	Симулатор, аутомобил, ОВД-Псистем
УДК	
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, ВН:	
Извод, ИЗ:	У овом раду реализован је симулатор који комуницира са клијентском апликацијом путем Bluetooth протокола. Симулатор симулира догађаје аутомобила као што су брзина, позиција папучице итд.
Датум прихватања теме, ДП:	
Датум одбране, ДО:	
Чланови комисије, КО:	Председник:
	Члан:
	Члан, ментор:
	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 :	Goran Peić Bibić
Mentor, MN :	Prof. dr Nebojša Pjevalica
Title, TI :	One solution for implementing simulator events in the car.
Language of text, LT :	Serbian
Language of abstract, LA :	Serbian
Country of publication, CP :	Republic of Serbia
Locality of publication, LP :	Vojvodina
Publication year, PY :	2015
Publisher, PB :	Author's reprint
Publication place, PP :	Novi Sad, Dositeja Obradovica sq. 6
Physical description, PD : <small>(chapters/pages/ref./tables/pictures/graphs/appendixes)</small>	7/21/0/3/9/0/0
Scientific field, SF :	Electrical Engineering
Scientific discipline, SD :	Computer Engineering, Engineering of Computer Based Systems
Subject/Key words, S/KW :	simulator, auto, OBD 2 system
UC	
Holding data, HD :	The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, N :	
Abstract, AB :	In this work was realized simulator that communicates with client application via Bluetooth protocol. Simulator simulate events such as car speed, pedal position, etc.
Accepted by the Scientific Board on, ASB :	
Defended on, DE :	
Defended Board, DB :	President:
	Member:
	Member, Mentor:
	Mentor's sign

Zahvalnost

Svom mentoru prof. dr Nebojši Pjevalici, želeo bih odati zahvalnost na izboru interesantne teme, znanju i iskustvu koje sam stekao tokom izrade diplomskog rada.

Za nesebičnu posvećenost i pomoć prilikom realizacije rada, posebnu zahvalnost dugujem Tomislavu Maruni, Mladenu Kovačevu, Branimiru Kovačeviću, Marku Kovačeviću i Nenadu Jovanoviću.

SADRŽAJ

1. Uvod.....	6
2. Teorijske osnove	8
2.1 Microsoft Visual Studio	8
2.2 OBD-II	9
2.2.1 OBD-II i režimi rada.....	10
2.3 ELM 327	10
2.4 CAN magistrala.....	11
2.5 Elektronska kontrolna jedinica.....	12
3. Koncept rešenja.....	14
3.1 Analiza problema	14
3.2 Algoritam za rešavanje problema.....	15
3.3 Problemi pri projektovanju.....	16
3.4 Modularnosti	16
3.5 Skalabilnost	16
3.6 Prednost u odnosu na postojeća rešenja	16
4. Programsko rešenje.....	17
4.1 Grafički prikaz i način rada simulatora	18
4.2 Metode.....	20
5. Rezultati	23
5.1 Ispitivanje realizovanog rešenja	23
6. Zaključak	26
7. Literatura.....	27

SPISAK SLIKA

Slika 1 OBD-II priključak.....	9
Slika 2 ELM 327 priključak.....	11
Slika 3 Pozicije ECU-a u automobilu	13
Slika 4 Komunikacija simulatora i korisničke aplikacije.....	15
Slika 5 Pozicijarealizovanog simulatora u sistemu.....	17
Slika 6 Izgled Simulatora	18
Slika 7 Torque Pro	24
Slika 8 Torque Pro sa prepoznatim greškama.....	24
Slika 9 AplikacijaCarService	25

SPISAK TABELA

Tabela 1 Podržani standardi od strane ELM mikrokontrolera.....	11
Tabela 2 Podržane komande simulatora	20
Tabela 3 Brzina slanja odgovora.....	23

SKRAĆENICE

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

BCE – **B**rake **C**ontrol **M**odul, Kontrolni modul motorakočnica

ECM – **E**ngine **C**ontrol **M**odul, Kontrolni modul motora

ECU – **E**lectronic **C**ontrol **U**nit, Elektronska kontrolna jedinica

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

GEM – **G**eneral **E**lectric **M**odul, Opšti električni modul

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

PCM – **P**owertrain **C**ontrol **M**odul, Kontrolni modul pogona

1. Uvod

U ovom radu realizovan je simulator koji simulira događaje u automobilu. Simulator i klijentske aplikacije komuniciraju putem Bluetooth protokola. Zadatak simulatora je da što realnije simulira događaje u automobilu.

Za dobavljanje informacija u realnom vremenu [1] od vozila ili u slučaju laboratorijskih ispitivanja simulatora, korišćen je OBD-II sistem koji se po standardu koristi u svim vozilima. Kako su svi proizvođači vozila shvatili da je neophodno uspostaviti način komunikacije automobila sa spoljašnošću radi lakšeg servisiranja i unapređivanja karakteristika vozila osmišljen je OBD-II sistem koji definiše način komunikacije između elektronske kontrolne jedinice automobila i spoljašnjih uređaja za bilo koju marku vozila. Svaki proizvođač ima nešto specifično za svaki model vozila, kao na primer kodove grešaka, ali sam način njihovog dobavljanja za svaku marku vozila je isti što je suština ovog sistema. OBD-II se ugrađuje u sve automobile od 1996 godine.

Za izradu diplomskog rada u programskom jeziku C# korišćeno je okruženje Microsoft Visual Studio 2012.

Za korišćenje je potrebno pokrenuti simulator na računaru sa Bluetooth podrškom kako bi se uspostavila komunikacija sa klijentskim aplikacijama.

Ovaj rad se sastoji od sedam poglavlja:

U drugom poglavlju se govori o teorijskim osnovama koje su potrebne za razumevanje ovog rada, uključujući korišćene tehnologije.

U trećem poglavlju je opisan koncept na kome je zasnovana realizacija ovog simulatora.

U četvrtom poglavlju je dat 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 programskog paketa Microsoft Visual Studio.

U petom poglavlju su prikazani rezultati ispitivanja koje je sprovedeno radi verifikacije simulatora.

U šestom poglavlju je sumirano šta je urađeno u okviru datog rešenja i datje predlog mogućih proširenja.

U sedmom poglavlju je navedena korišćena literatura.

2. Teorijske osnove

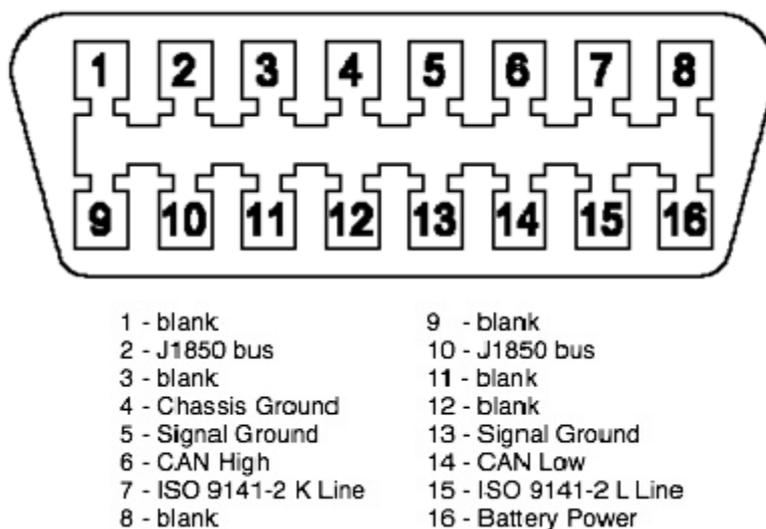
U ovom poglavlju su izložene osnovne informacije o razvojnom okruženju, dat je opis protokola i sistema koji se koriste u automobilske industriji i alata koji se koriste za simuliranje rada automobila.

2.1 Microsoft Visual Studio

Microsoft Visual Studio je integrisano razvojno okruženje kompanije Microsoft. Koristi se za razvoj računarskih programa za Windows operativne sisteme, web stranice, aplikacije i usluge. Takođe se koriste Microsoft-ove platforme za razvoj poput aplikativnih programskih sprega (engl. *Application Program Interface*) za Windows: Windows Forms, Windows Presentation Foundation, Windows Store i Microsoft Silverlight. Program takođe sadrži alate poput dizajnera oblika koji se koriste za pravljenje aplikacija sa grafičkom korisničkom spregom, web dizajnera, dizajnera klasa i dizajnera šema baza podataka. Visual Studio podržava različite programske jezike kao i uređivače koda. Prihvata proširenja koja poboljšavaju funkcionalost na skoro svakom nivou dodajući podršku sistema za upravljanje izvornim kodom i dodajući nove skupove alata poput tekstualnih urednika i vizualnih dizajnera za jezik specifičnih domena ili za druge delove procesa razvoja softvera. Podržani jezici su C, C++ i C++/CLI (preko Visual C++), VB.NET (preko Visual Basic .NET), C# (preko Visual C#) i F# (počevši od programa Visual Studio 2010). Podrška za ostale programske jezike poput M, Python, Ruby-ja, kao i ostalih je dostupna instalacijom odgovarajućih dodatnih jezičkih servisa. Takođe podržava XML/XSLT, HTML/XHTML, JavaScript i CSS.

2.2 OBD-II

On-Board Diagnostic [2] je automobilski pojam koji se odnosi samo na dijagnostiku i izveštaj o sposobnosti vozila. OBD-II je danas zastupljen u većini automobila kao i manjim kamionima. Tokom 70-ih i ranih 80-ih godina proizvođači vozila su počeli koristiti elektronska sredstva za kontrolu funkcija motora i dijagnosticiranje problema motora. Tokom godina On-Board Diagnostic sistemi postaju sve sofisticiraniji. OBD-II, novi standard, uveden sredinom 90-ih godina, daje gotovo potpunu kontrolu motora i prati delove šasije, tela i pomoćnih uređaja, kao i dijagnostičku kontrolnu mrežu automobila. U svim modernim automobila, počevši od 2008 godine, se koristi CAN (ISO 15765-4) [3] protokol za komunikaciju sa klijentskim aplikacijama. OBD-II je poboljšanje u odnosu na OBD-I u smislu sposobnosti i standardizacije. OBD-II standard određuje vrstu dijagnostičkog priključka i njegovih nožica, kao i format poruka. Rane verzije OBD-a pružaju indikacije o kvarovima u vozilu, ali ne mogu pružiti bilo kakvu informaciju u vezi sa prirodom problema. Moderne OBD realizacije koriste standardizovani niz digitalnih kodova greške, koji omogućavaju brzu identifikaciju problema u automobilu. Slika 1 predstavlja OBD-II priključak.



Slika 1 OBD-II priključak

Najzastupljeniji OBD-II protokoli:

- J1850 PWM (Ford automobili)
- J1850 VPW (GM automobili)
- ISO9141-2 (Azija, Evropa, Chrysler automobili)
- ISO14230-4 (Protocol 2000)
- ISO15765-4 (CAN)

2.2.1 OBD-II i režimi rada

Postoji deset načina rada opisanih u OBD-II standardu [4].

Režimi rada su:

- 01 – Trenutni podaci.
- 02 – Ovaj režim rada daje trenutne podatke (engl. freeze frame) o smetnji. Kad se otkrije greška kontrolna jedinica motora pribavlja sve trenutne podatke iz automobila, tako da korisnik ima uvid o stanju automobila u trenutku otkrivanja greške.
- 03 – Ovaj režim rada pokazuje greške automobila.
- 04 – Ovaj režim rada se koristi za brisanje snimljenih kodova kvara i isključivanje indikatora koji otkriva greške motora.
- 05 – Test rezultat, nadgleda senzor za kiseonik.
- 06 – Režim koji daje rezultate na sistemima koji ne podležu stalnim nadzorima.
- 07 – Ovaj režim daje nepotvrđene kodove greške.
- 08 – Ovaj režim daje rezultate samo-dijagnostike na drugim sistemima.
- 09 – Daje informacije o vozilu kao što su: identifikacijski broj vozila (engl. *vehicle identification number*) i vrednosti kalibracije.
- 0A – Režim koji daje trajne kodove greške.

2.3 ELM 327

ELM 327 [5] je programirani mikrokontroler čija je uloga prevođenje OBD sprege koja se koristi u većini modernih automobila. ELM 327 je jedna od prevodioca OBD-a iz ELM Electronics. Automobili novijih generacija moraju biti opremljeni sa spregom za povezivanje dijagnostičke opreme za ispitivanje rada automobila. Prenos podataka na ove sprege podržava nekoliko standarda, ali nijedan od njih nije direktno upotrebljiv za PC ili pametne uređaje. ELM327 je dizajniran da deluje kao most između On-Board Diagnostics (OBD) prolaza i standardne RS232 serijske sprege. Pored toga što može da automatski prepozna i tumači devet OBD protokola, ELM327 pruža i podršku za velike brzine komunikacije. Slika 2 predstavlja ELM priključak.



Slika 2 ELM 327 priključak

ELM 327 za komunikaciju sa korisnikom koristi ATkomande. AT komande služe za definisanje načina komuniciranja sa automobilom, pre potvrde protokola. ELM 327 je proizvod koji jedini podržava sve standarde OBD protokola.

Spisak dostupnih ELM mikrokontrolera i podržanih protokola daje se u nastavku na Tabela 1:

	ELM320	ELM322	ELM323	ELM325	ELM327	ELM328	ELM329
SAE J1850-PWM	X				X		
SAE J1850-VPW		X			X		
ISO 9141-2			X		X		
ISO 14230-4 (slow)			X		X		
ISO 14230-4 (fast)			X		X		
ISO 15765-4 (CAN)					X		X
SAE J2411 (SWCAN)					X		X
KW1281 (SAE J2818)							
SAE J1939 (250kbps)					X		X
SAE J1939 (500kbps)					X		X
SAE J1708 (J1587)				X			
SAE J1708 (J1922)				X			

Tabela 1 Podržani standardi od strane ELM mikrokontrolera

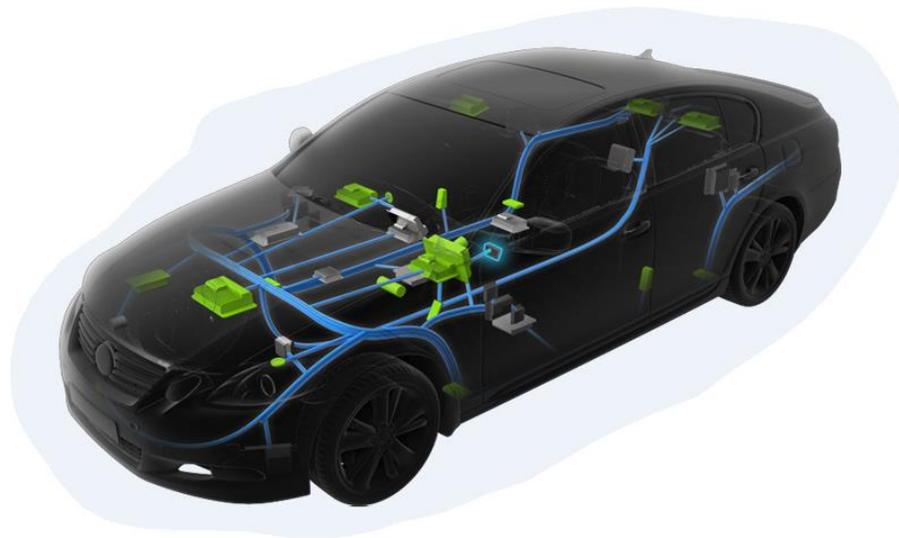
2.4 CAN magistrala

CAN je magistrala podataka koju je razvio Robert Bosch 1983 godine i koja je ubrzo prihvaćena u automobilskoj i vazdušno-kosmičkoj industriji. To je magistrala serijskog protokola

čiji je cilj povezivanje samostalnih sistema i senzora kao alternativno rešenje za uobičajne višezične kablove. Omogućava komunikaciju između automobilskih uređaja preko jednosmerne ili dvosmerne linije podataka, brzinom prenosa podataka od 1Mbps. Moderan automobil može sadržati i do 80 elektronskih kontrolnih jedinica za različite podsisteme. Tipično, najveći procesor je kontrolna jedinica motora (engl. Engine Control Unit). Ostale kontrolne jedinice služe za kontrolu transmisije, vazdušnih jastuka, sigurnosnu bravu, automatski kontroler brzine, zvučne sisteme, prozore, vrata, mala podešavanja, baterije i punjenje sistema. Podsystemi kontrolišu aktuatorne ili primaju informacije od senzora. CAN standard je osmišljen da ispuni ove potrebe. CAN magistrala se može koristiti u vozilu da poveže kontrolnu jedinicu motora i prenos ili da poveže kontroler zaključavanja vrata, kontrolu klime, kontrolu sedišta itd.

2.5 Elektronska kontrolna jedinica

Automobili novije generacije poseduju sistem koji je neophodan za prikupljanje informacija o radu motora. Kontrolna jedinica motora (ECU) je elektronska kontrolna jedinica koja kontroliše čitav niz pogona na motoru s unutarašnjim sagorevanjem kako bi se osigurala optimalne karakteristike vozila. To čini čitajući vrednosti iz mnoštvo senzora unutar motora, tumačenjem podataka pomoću multidimenzionalne mape karakteristika (tzv. Lookup tablice), te u skladu s tim podešava pogon motora. Pre ECU-a, mešavine vazduha i goriva, vreme paljenja i praznog hoda su mehanički postavljali i dinamički kontrolisali pneumatski mehanički sistemi. Postoji veliki broj različitih vrsta ECU-a, uključujući i Engine Control Modul (ECM), Powertrain Control Module (PCM), Brake Control Module (BCM), General Electric Module (GEM) i druge. Noviji modeli automobila mogu imati i do 80 ECU-a, zbog velike složenosti sistema. Programiranje uključeno u razvoj ECU-a postaje sve složenije za održavanje. 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 priključak, koji omogućava povezivanje dijagnostičkih uređaja. Slika 3 predstavlja poziciju ECU-a u automobilu.



Slika 3 Pozicije ECU-a u automobilu

3. Koncept rešenja

U okviru ovog poglavlja data je analiza problema, opis algoritma po kom rešenje funkcioniše, kao i diskusija problema, koje je neophodno rešiti.

3.1 Analiza problema

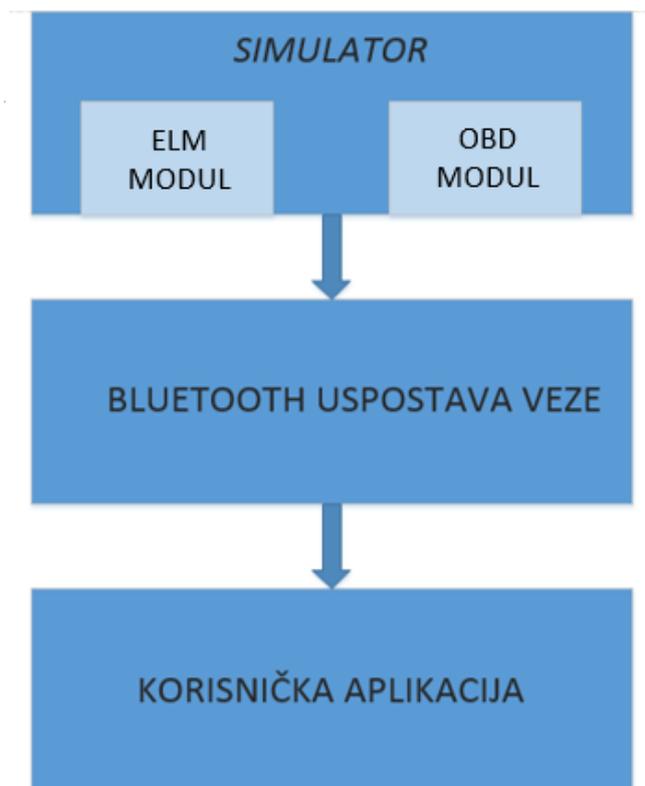
Nakon što je američki kongres 70-ih godina izglasao zakon o čistijem vazduhu i osnovao agenciju za zaštitu životne sredine, proizvođači automobila su počeli da se okreću sistemima koji se kontrolišu putem elektronike. Tako su uspeli da kontrolišu funkcije motora i prilagođavaju sisteme kako bi smanjili emisiju izduvnih gasova.

Kako ne bi koristili prave automobile za laboratorijska ispitivanja, počela je proizvodnja aplikacija koje simuliraju događaje automobila. Korišćenjem simulatora, čija je glavna uloga što realnije simuliranje događaja automobila, moguće je ispitivanje korisničkih aplikacija koje se bave dijagnostikom istih. Da bi verifikacija bila verodostojnija postoje različiti generatori podataka simulatora. Npr. može se izabrati da se podaci za sve parametre nasumično generišu. Tokom godina usavršavala se funkcionalnost kako korisničkih aplikacija tako i simulatora automobila.

Posredstvom servisa podaci se prikazuju korisniku na putni računar. Glavna namena putnog računara je konstantno informisanje vozača o svim dešavanjima u automobilu.

3.2 Algoritam za rešavanje problema

Slika 4 predstavlja komunikaciju simulatora i korisničke aplikacije.



Slika 4 Komunikacija simulatora i korisničke aplikacije

Na početku je potrebno uspostaviti vezu sa korisničkom aplikacijom putem Bluetooth veze. Korisnička aplikacija može biti bilo koja aplikacija koja je namenjena za dijagnostiku automobila. Nakon uspostave komunikacije, simulator odgovara na komande ELM mikrokontrolera pomoću AT komandi. AT komande su isključivo vezane za ELM mikrokontroler. AT komande služe da bi se definisao način komuniciranja pre uspostave veze sa automobilom.

Nakon uvodnih komandi korisnička aplikacija zahteva određeni protokol. Kada se detektuje tip protokola simulator simulira povezanost sa kontrolnom jedinicom motora. Tip protokola zavisi od modela automobila. Najzastupljeniji protokol je ISO 15765-4(CAN) [3]. Nakon određivanja protokola sledi komunikacija preko OBD komandi.

3.3 Problemi pri projektovanju

Najveći problem pri projektovanju aplikacije je sama komunikacija sa ELM mikrokontrolerom. Problem se sastojao u tome što se nije znao tačan format poruke, tj. kako treba da izgleda format odgovora na određenu komandu. Problem se rešio osluškivanjem komunikacije između OBD simulatora i komercijalno dostupne korisničke aplikacije Torque Pro [6].

3.4 Modularnosti

Svaki modul je nezavisan od ostalih tako da ispunjava jedan podskup specifičnih zahteva, pri čemu izmene u jednom modulu minimalno utiču na ostale module. Dodavanje novih komandi kao i dodavanje novih funkcija je veoma lako i ne utiče na ostatak modula.

3.5 Skalabilnost

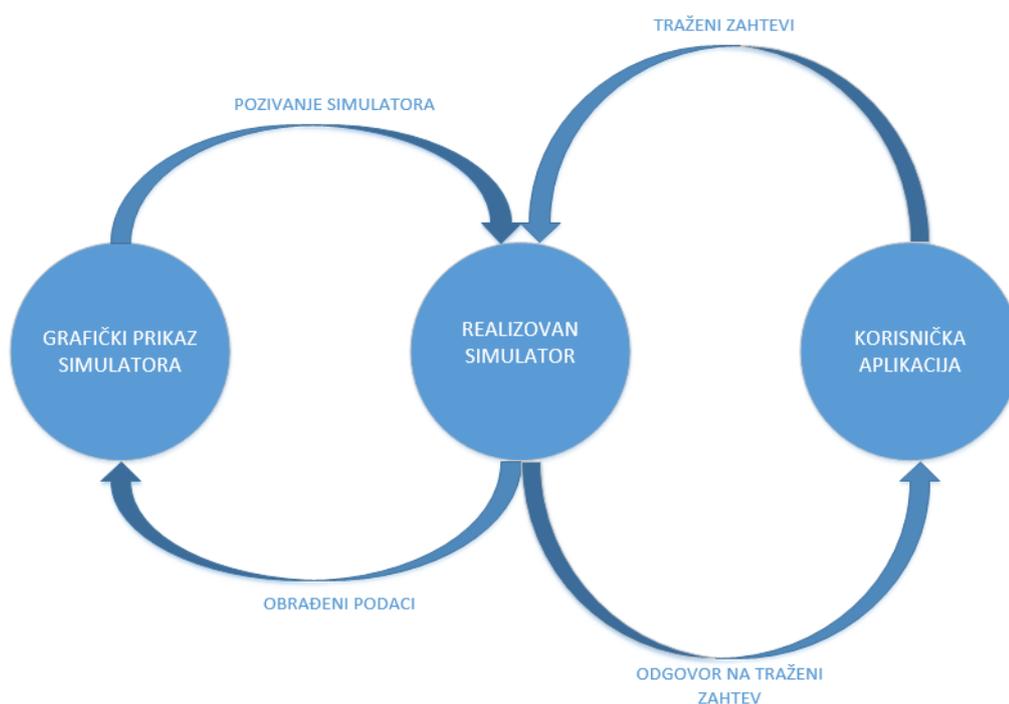
Aplikacija je skalabilna tako da ona lakše može podneti povećan protok podataka i može se koristiti za testiranje sa više aplikacija. Cilj kojim teže svi projektanti sistema jeste da se postigne linearnost u brzini odgovora na zahtev i količine podataka sa kojima se manipuliše.

3.6 Prednost u odnosu na postojeća rešenja

Srodna rešenja su retka. Samo poboljšanje u odnosu na postojeće rešenje simulatora [7] je u tome što simulator može da primi više komandi odjednom, kao i povećanje brzine odgovora na zahteve korisnika.

4. Programsko rešenje

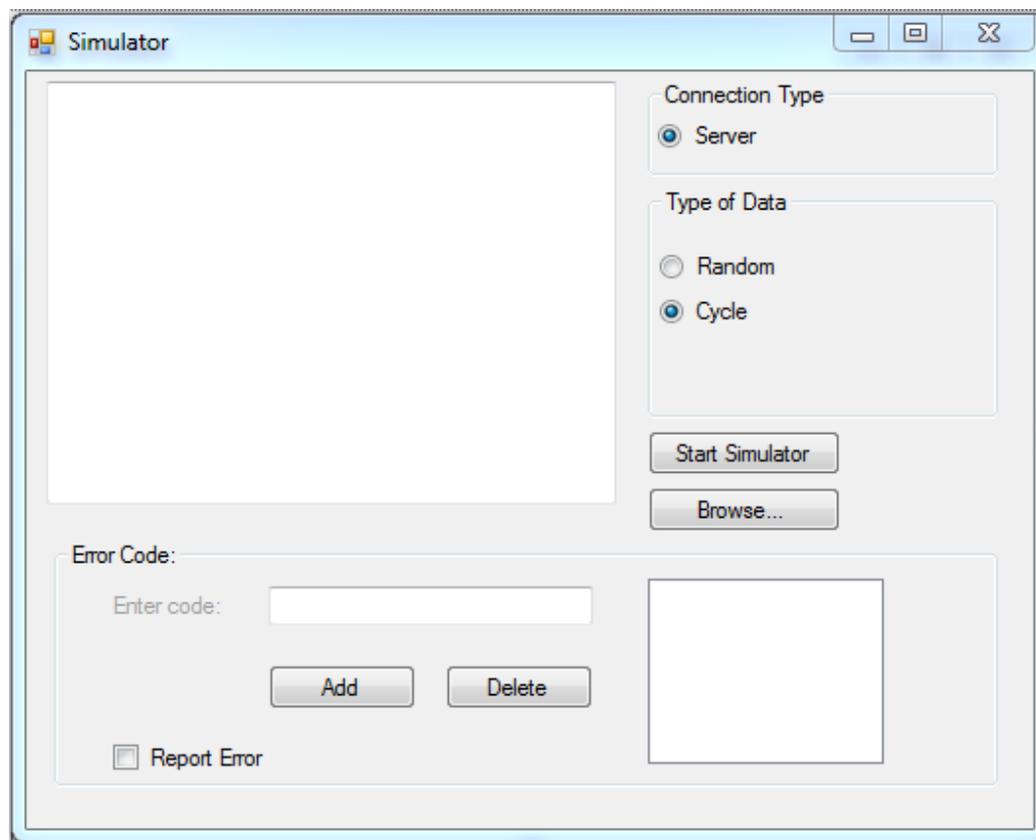
Ovo poglavlje podeljeno je u dve celine. Prva celina predstavlja grafički izgled simulatora i objašnjenje načina rada, dok druga predstavlja opis realizovanih metoda. Način na koji se vrši komunikacija između korisničke aplikacije i simulatora prikazan je na Slika 5.



Slika 5 Pozicijarealizovanog simulatora u sistemu

4.1 Grafički prikaz i način rada simulatora

U produžetku na Slika 6 prikazan je izgled simulatora:



Slika 6 Izgled Simulatora

Simulator (Slika 6 Izgled Simulatora) je realizovan tako što je potrebno prvo izabrati način generisanje podataka sa kojim želimo da simulator radi (Random ili Cycle). Opcija Random generiše nasumične vrednosti svih podataka.

Za opciju Random nije potrebna nikakava dodatnadataoteka, već samo da se aplikacija pokrene pritiskom na dugme Start Simulator. Nakon toga, potrebno je da se korisnikpoveže sa simulatorom da bi se komunikacija nastavila.

Drugi način generisanja podataka koji je podržan je opcija Cycle, za koju je potrebna dodatna datoteka. Do dodatne datoteka se dolazi pomoću Browse dugmeta koje otvara novi prozor u kojem treba da se izabere data datoteka. Kada se izabere datoteka simulator se pokreće na isti način kao i kada se odabere Random opcija. Data datoteka sadrži niz parametara koji predstavljaju vrednosti simulatora. U toku rada simulatora moguće je menjati način generisanja parametara (Cycle ili Random) kojim će raditi simulator. Simulator ima mogućnost provere koda greške, čiji meni se nalazi u donjem delu prozora aplikacije. Korišćenje koda greške je realizovano tako što se željena greška (definisana kodom greške) upisuje u tekstualni prozor.

Pritiskom na dugme Add greška se dodaje u listu željenih vrednosti samo ukoliko se upisana greška podudara sa nekom greškom izdatoteke grešaka. Datoteka grešaka se ubacuje u sistem na isti način kao i datoteka podataka. Da bi kodne greške bile dostupne korisniku potrebno je označiti Report error. U toku rada simulatora moguće je menjati dostupnost koda greške, kao i dodavanje dodatnih grešaka koje se mogu ispitivati. Kada se simulator i korisnička aplikacijapovežu preko Bluetooth veze i uspostave komunikaciju, simulator šaljeodgovore na tražene događaje od strane korisnika. Događaji na koje simulator može da odgovori prikazani su u Tabela 2 [8]:

PID	OPIS PID-OVA
“0D”	BRZINA
“11”	POZICIJA PAPUČICE
“0C”	OBRTAJI MOTORA
“0A”	PRITISAK GORIVA
“05”	TEMPERATURA RASHLADNE TEČNOSTI U MOTORA
“2F”	NIVO GORIVA U REZERVOARU
“0F”	TEMPERTURA VAZDUHA KOJA ULAZI U MOTOR
“10”	PROTOK VAZDUHA
“2D”	KODOVI GREŠKE
“33”	ATMOSFERSKI PRITISAK
“5C”	TEMPERATURA ULJA
“59”	PRITISAK U CEVIMA ZA GORIVO
“5E”	NIVO GORIVA U MOTORU
“04”	OPTEREĆENJE MOTORA
“5A”	RELATIVNI POLOŽAJ PEDALE GASA
“4A”	POLOŽAJ PEDALE GASA E

“49”	POLOŽAJ PEDALE GASA D
“06”	KRATKOROČNI REŽIM PODEŠAVANJE SMEŠE(PRVI NIZ CILINDARA)
“07”	DUGOROČNI REŽIM PODEŠAVANJE SMEŠE(PRVI NIZ CILINDARA)
“08”	KRATKOROČNI REŽIM PODEŠAVANJE SMEŠE(DRUGI NIZ CILINDARA)
“09”	DUGOROČNI REŽIM PODEŠAVANJE SMEŠE(DRUGI NIZ CILINDARA)

Tabela 2 Podržane komande simulatora

4.2 Metode

Simulator poseduje određeni skup metoda. Cilj izrade aplikacije da se prikaže što realniji simulator za ispitivanje korisničke aplikacije koja dijagnosticira događaje u automobilu.

Realizovane su sledeće metode:

```
protected override void OnFormClosing (FormClosingEventArgs e)
```

Opis:

- Metoda za isključivanje aplikacije, kada se pritisne dugme za isključivanje otvori se novi prozor o potvrdi iste.

Parametri:

- Obezbeđuje podatke za isključivanje aplikacije.

Povratna vrednost:

- Nema je.

```
public void connectAsServer ()
```

Opis:

- Metoda za stvaranje programske niti i njeno pokretanje.

Parametri:

- Nema ih.

Povratna vrednost:

- Nema je.

```
public void serverConnectThread ()
```

Opis:

- Metoda za primanje i odgovaranje na AT komande.

Parametri:

- Nema ih.

Povratna vrednost:

- Nema je.

```
private void init ()
```

Opis:

- Metoda u kojoj dolazi do povezivanja između korisnika i simulatora, kao i početna inicijalizacija.

Parametri:

- Nema ih.

Povratna vrednost:

- Nema je.

```
private void closeForm()
```

Opis:

- Metoda za deinicijalizaciju nakon isključivanja aplikacije.

Parametri:

- Nema ih.

Povratna vrednost:

- Nema je.

```
private void deInit()
```

Opis:

- Metoda za deinicijalizaciju nakon raskidanja veze između korisnika i simulatora.

Parametri:

- Nema ih.

Povratna vrednost:

- Nema je.

```
private void updateUI(string message)
```

Opis:

- Metoda za ispis teksta na ekran.

Parametar:

message – poruka koja se ispisuje na ekran.

Povratna vrednost:

- Nema je.

```
public int parse(byte[] received, bool cyclebool, bool
randombool, List<string> errorlist, bool errorbool, string
fileData )
```

Opis:

- Metoda za raščlanjivanje režima rada.

Parametri:

received – prijem OBD komandi.

cyclebool – izabrana cycle opcija.

randombool – izabrana random opcija.

errorlist – lista sa kodom greške.

errorbool – izabrana Report error opcija.

fileData – datoteka sa parametrima za simulator.

Povratna vrednost:

- Odgovor simulatora na trženi zahtev klijenta.

```
private byte[] modeOne(byte[] received, bool cyclebool, bool
randombool, string fileData)
```

Opis:

- Metoda za raščlanjivanje OBD-a, kada korisnik pošalje jedna zahtev u jednoj poruci.

Parametri:

received –prijem OBD komandi.
 cyclebool –izabrana cycle opcija.
 randombool –izabrana random opcija.
 fileData –datoteka sa parametrima za simulator.

Povratna vrednost:

- Odgovor simulatora na trženi zahtev klijenta.

```
private byte[] modeThree (List<string> errorlist, bool errorbool)
```

Opis:

- Metoda za raščlanjivanje kodova greške.

Parametri:

errorlist –lista sa kodom greške.
 errorbool –izabrana Report error opcija.

Povratna vrednost:

- Odgovor simulatora na trženi zahtev klijenta.

```
private byte[] multiplePids (byte[] received, bool cyclebool, bool  

  randombool, string fileData)
```

Opis:

- Metoda za raščlanjivanje OBD-a, kada korisnik šalje više zahteva u jednoj poruci.

Parametri:

received –prijem OBD komandi.
 cyclebool –izabrana cycle opcija.
 randombool –izabrana random opcija.
 fileData –datoteka sa parametrima za simulator.

Povratna vrednost:

- Odgovor simulatora na trženi zahtev klijenta.

```
public void reset ()
```

Opis:

U ovoj metodi se brojači postavljaju na nulu.

Parametari:

Nema ih.

Povratna vrednost:

- Nema je.

5. Rezultati

U ovom poglavlju će biti predloženi rezultati ispitivanja i verifikacije simulatora. Sprovedeno je nekoliko vrsta ispitivanja.

5.1 Ispitivanje realizovanog rešenja

U prvom ispitivanju mereno je vreme slanja odgovora na primljenu komandu koju je poslala klijentska aplikacija. Brzina slanja odgovora prikazana je u Tabeli 3:

Rešenja	Brzina slanja odgovora po jednojkomandi
Postojeće rešenje simulatora [7]	230-250[ms]
Novo rešenje simulatora	170-200[ms]

Tabela 3 Brzina slanja odgovora

Iz tabele se vidi da se u novoj realizaciji simulatora dobija znatno ubrzanje nego u prethodnom rešenju.

Ispitivanje funkcionalnosti simulatora je realizovano sa više klijentskih aplikacija, da bi se potvrdila stabilnost i ispravnost iste.

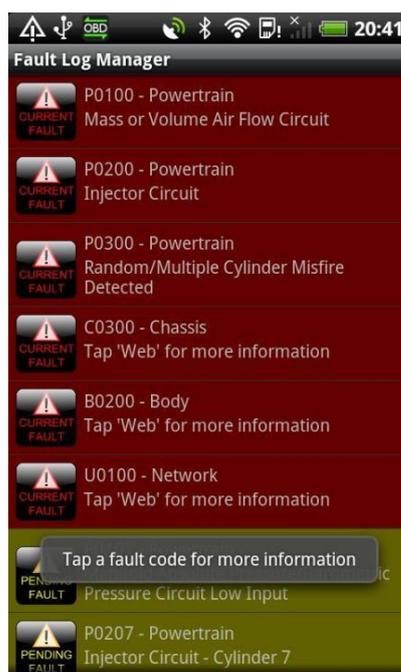
Detaljnije ispitivanje je rađeno sa komercijalno dostupnom korisničkom aplikacijom Torque Pro. Ispitivanje je realizovano u više faza. Prva faza ispitivanja bavila se uspostavljanjem veze između simulatora i klijentske aplikacije. Druga faza ispitivanja bavila se proverom podudarnosti dela simulatora sa ELM 327 kod koga se komunikacija bazira na AT komandama.

Komunikacija sa ELM 327 predstavlja početni deo komunikacije sa simulatorom. Treća faza ispitivanja se bavi proverom podudarnosti simulatora sa elektronskom kontrolnom jedinicom, tj. da li simulator simulira događaje automobila na pravi način. Događaji koje simulator može da simulira su brzina, pozicija papučice gasa, temperatura rashladne tečnosti u motoru, obrtaji motora itd. Na Slika 7 je predstavljen izgled Torque Pro aplikacije koja je korišćena prilikom ispitivanja funkcionalnosti simulatora.



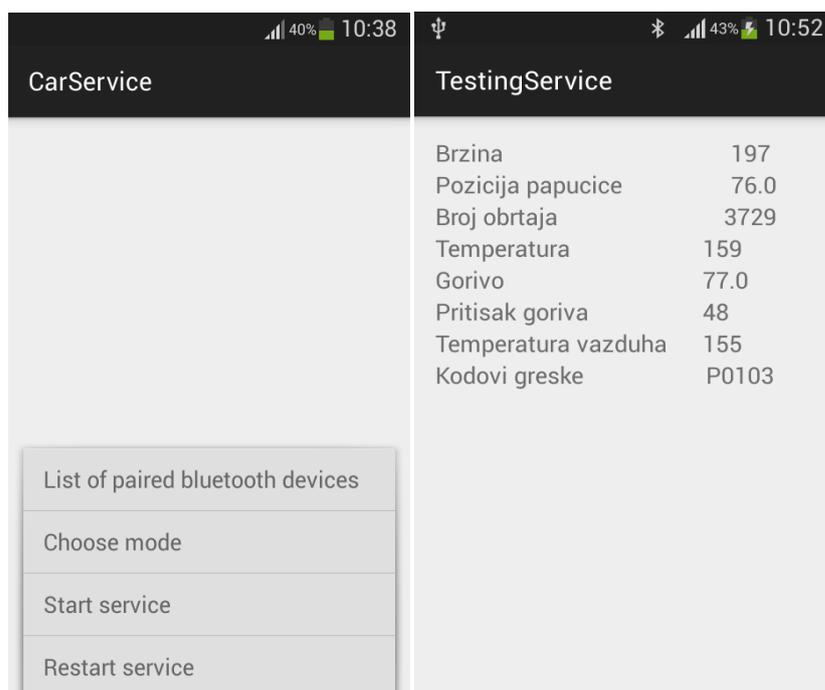
Slika 7 Torque Pro

U četvrtoj fazi ispitivane su greške koje simulator može da podrži. Greške su podeljene na grupe kao što su P - Powertrain, C – Chassis, B – Body i U – Network. Slika 8 prikazuje Torque Pro korisničku aplikaciju sa prepoznatim greškama.



Slika 8 Torque Pro sa prepoznatim greškama

U petoj fazi ispitana je mogućnost primanja i slanja više komandi, gde se u jednoj poruci primi više korisničkih zahteva, a zatim i odgovorina iste. Ova višekomandna mogućnost nije realizovana u dosadašnjim rešenjima simulatora. Za ispitivanje rešenja korišćena je CarService aplikacija, jer Torque Pro aplikacije ne podržava mogućnost slanja više komandi odjednom. Car Service je Android servis kojim se uspostavlja komunikacija sa OBD II sistemom automobila i dobavljaju traženi parametri vozila. Slika 9 predstavlja aplikaciju CarService.



Slika 9 AplikacijaCarService

6. Zaključak

U ovom radu je prikazana realizacija simulatora koji simulira događaje u automobilu. Komunikacija između simulatora i klijentske aplikacije se odvija putem Bluetooth protokola.

Cilj izrade aplikacije da se prikaže što realniji simulator za ispitivanje korisničke aplikacije koja dijagnosticira događaje u automobilu

U ovom rešenju simulatora realizovani su sledeći režimi rada: režim trenutnih vrednosti, režim kodova greške kao i režim nepotvrđenih kodova greške.

Mogući pravci daljeg rada na unapređenju rešenja mogući su unapređenjem postojećih funkcionalnosti ili dodavanjem novih. Modularna struktura rešenja omogućava integraciju izmena i dopuna, bez velikih poteškoća.

Simulator bi se u budućnost mogao proširiti sa dodatnim režimima kao što je režim trenutnih podatak u trenutku otkrivanja greške (engl. freeze frame), kao i povećanje broja komandi koje podržava isti.

7. Literatura

- [1]Sistemska programska podrška u realnom vremenu 2, Miroslav Popović, Vladimir Kovačević, Univerzitet u Novom Sadu 2012.
- [2]OBD dostupno na:<http://www.obdii.com/> učitano 17.04.2015
- [3]OBD CAN protocol dostupno na: <http://www.obdii.com/background.html> učitano 22.04.2015
- [4]OBD Modes, dostupno na:<http://www.outilsobdfacile.com/obd-mode-pid.php> učitano 30.04.2015
- [5]ELM OBD dostupno na:<http://www.elmelectronics.com/DSheets/ELM327DSF.pdf> učitano 12.05.2015
- [6]Torque Pro dostupno na:https://torque-bhp.com/wiki/Main_Page učitano 14.05.2015
- [7]OBDSim dostupno na: <http://icculus.org/obdgpslogger/obdsim.html> učitano 27.05.2015
- [8]OBD command dostupno na :http://www.bb-elec.com/Products/Manuals/Intelligent-OBDII-Gateway-Command-and-Response_251.pdf učitano 29.05.2015