УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Урош Ковачевић

Једно решење програмске подршке за парсирање и визуелизацију сигнала садржаних у ТТЛ датотеци

ДИПЛОМСКИ РАД
- Основне академске студије -

Нови Сад, 2013

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**:		Урош Ковачевић	
Ментор, **МН**:		доц. Иштван Пап	
Наслов рада, **НР**:		Једно решење програмске подршке за парсирање и визуелизацију сигнала садржаних у ТТЛ датотеци	
Језик публикације, **ЈП**:		Српски / латиница	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		2013.	
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)		7/49/0/2/22/0/0	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кључне речи, **ПО**:		CAN magistrala, FlexRay magistrala, LIN magistrala, TTL datoteka, XML konfiguraciona datoteka	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		Cilj zadatka je realizacija programske podrške za parsiranje TTL i XML datoteka kao i grafičko prikazivanje signala sadržanih u njima. TTL datoteka je dnevnik koji sakuplja informacije koje se razmenjuju u okviru automobila na različitim tipovima automobilskih sprežnih sistema poput CAN, FlexRay, LIN i sličnih. Informacije se sakupljaju preko Data Logger-a, uređaja sa podrškom za veliki broj različitih magistrala, koji prikuplja informacije sa vozila u toku razmene podataka između različitih kontrolnih jedinica (ECU) samog vozila. Podatke čuva u TTL datoteci.	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	доц. Јелена Ковачевић	
	Члан:	доц. Милан Бјелица	Потпис ментора
	Члан, ментор:	доц. Иштван Пап	

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

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

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**:		Урош Ковачевић
Mentor, **MN**:		doc. Иштван Пап
Title, **TI**:		One solution softwer support for parsing and visualization of signals contained in TTL files
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**:		2013.
Publisher, **PB**:		Author's reprint
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		7/49/0/2/22/0/0
Scientific field, **SF**:		Electrical Engineering
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems
Subject/Key words, **S/KW**:		CAN bus, FlexRay bus, LIN bus, TTL file, XML configuration file
UC		
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, **N**:		
Abstract, **AB**:		The task is the realization the program for parsing XML files and TTL as well as graphical representation of the signal contained therein. TTL is the log file which collects information exchanged in the car on different types of car system such as CAN, FlexRay, LIN, and similar. Information collected through the Data Logger, the devices with support for a wide variety of buses, which collects information from the vehicle during the exchange of data between different control units (ECU) of the vehicle. The data stored in the TTL file.
Accepted by the Scientific Board on, **ASB**:		
Defended on, **DE**:		
Defended Board, **DB**:	President:	doc. Jelena Kovačević
	Member:	doc. Milan Bjelica — Menthor's sign
	Member, Mentor:	doc. Ištvan Pap

Zahvalnost

Zahvaljujem se Danijelu Spasojeviću i Veliboru Iliću kao i celom TTTech timu koji su mi pomogli pri izradi rada.

SADRŽAJ

SPISAK SLIKA

SPISAK TABELA

SKRAĆENICE

TTL TTX Logger
XML Extensible Markup Language
WEB World Wide Web
WPF Windows Presentation Foundation
ID Indentifier
SAE Society of Automotive Engineers
PCM Powertrain control module
OEM Original equipment manufacturer
ECU Electronic control unit

1. Uvod

Automobilska industrija (eng. *automotive*) je izraz koji pokriva širok spektar kompanija i organizacija u koje su uključene dizajn, razvoj, proizvodnja, marketing i prodaja motornih vozila, motocikala i teretnih vozila. To je jedan od najvažnijih ekonomskih sektora po prihodu u svetu.

Termin automobilska industrija ne uključuje industrije posvećene održavanju automobila kao što su radionice za popravku kvarova ili benzinske stanice.

Naziv *automotive* je nastao od Grčke reči *autos* (samostalno) i latinske reči *motivus* (kretanje), da predstavlja sve oblike vozila sa sopstvenim pogonom. Predložio ga je član SAE (*Society of Automotive Engineers*), Elmer Sperry.

Istorija industrije automobila počinje 1890. godine sa stotinama proizvođača koji razvijaju kočije. Više decenija Amerika je bila vodeća u svetu po proizvodnji automobila. 1929. godine svet je imao oko 32.000.000 automobila u upotrebi, od čega je Američka industrija proizvodila preko 90%. U to vreme je bilo jedno vozilo na svaku petu osobu. Posle Drugog svetskog rata, Amerika je pokrivala 75% ukupne svetske proizvodnje. 1980. godine vodeću ulogu preuzima Japan, da bi ga 2009. Kina prestigla sa 13.8 miliona proizvedenih jedinica. Danas se u svetu proizvede preko 80.000.000 jedinica godišnje. [1]

Jedna od glavnih stvari koja je pomogla automobilima da omoguće više sigurnosti i pogodnosti je elektronika. Napretkom u tehnologiji i elektronici, proizvođači automobila su bili u mogućnosti da ponude širok spektar servisa i pogodnosti koje je većina novih vlasnika znala da ceni. U bitnije novine koje je elektronika uvela u vozila je *Electronic Fuel Injection* (EFI), koji obezbeđuje tačno potrebnu količinu goriva za motor. Zamenio je standardni karburator zbog boljih osobina, pre svega jer sprečava preopterećenje motora, njegov ekonomičniji rad, manje zagađenja itd. Tu je i računarska dijagnostika koja je omogućila

vozačima da uoče probleme motora ili druge vrste problema pre nego što se desi kvar. Pre računarske dijagnostike, većina vozača nije znala da sa vozilom nije sve u redu dok se nešto drastično ne desi, kao pregrevanje ili nestanak goriva.

Zatim tu je inovacija u vidu *all-wheel drive* (AWD), nastala 1900. godine , koja podrazumeva da sva četiri točka primaju snagu od motora a ne samo dva od njih. Ovim je obezbeđena bolja kontrola vozila kao i bolja vuča po ledu i mokrom putu. Od bitnije elektronike tu su još *airbag* od 1952. godine, *global positioning system* (GPS), *hybrid cars* i druge [2].

Kako je elektronika postajala kompleksnija tako se pojavila i potreba za komunikacijom i upravljanjem odnosno programskom podrškom za istu.

1.1 Zadatak rada

Cilj zadatka je realizacija programske podrške za parsiranje *TTL* i konfiguracionih datoteka kao i prikazivanje signala sadržanih u njima. *TTL* datoteka je dnevnik koji sakuplja informacije koje se razmenjuju u okviru automobila na različitim tipovima automobilskih sprežnih sistema poput *CAN*, *Flexray*, *LIN* i sličnih. Informacije se sakupljaju preko *Data Logger*-a, (slika 1-1) uređaja sa podrškom za veliki broj različitih magistrala, koji prikuplja informacije sa vozila u toku razmene podataka između različitih kontrolnih jedinica (ECU) samog vozila. Podatke čuva u *TTL* datoteci. [3]

Slika 1-1.*Data Logger*

XML je jezik koji definiše skup pravila za kodiranje dokumenata u format koji je čitljiv i ljudima i mašinama. Projektni ciljevi *XML*-a naglašavaju jednostavnost, opštost i

upotrebljivost preko interneta. Iako je *XML* dizajniran sa fokusom na dokumente, široko se koristi i za predstavljanje proizvoljne strukture podataka, na primer za konfigurisanje različitih alata [4].

Cilj ovog rada bio je pisanje *C# WPF* programa koji parsira *TTL* datoteku na osnovu zadatih kriterijuma (u *XML* konfiguraciji) i grafički prikazuje signale u vidu grafova i sličnih vizuelnih komponenti.

2. Teorijske osnove

Najveći broj funkcija koje se odvijaju u vozilu pored klasičnih analognih i digitalnih signala pokrivaju tri sprežna sistema: *CAN, FlexRay* i *LIN*. Ove magistrale su pre svega zadužene za prenos podataka odgovornih za sigurnost u vozilu.

2.1 *CAN* magistrala

CAN je magistrala koju je razvio Robert Bosch 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 više-žične kablove. Omogućava komunikaciju između automobilskih uređaja preko jednosmerne ili dvosmerne linije podataka, brzinom prenosa podataka od 1Mbps. [6]

2.1.1 Pre pojave *CAN*-a

Od ranih četrdesetih, proizvođači automobila su konstantno poboljšavali tehnologiju svojih vozila dopunjavanjem i povećanjem broja elektronskih komponenti (slika 2-1). Kako je tehnologija napredovala, vozila su postala više kompleksna jer su elektronske komponente zamenile mehaničke sisteme i pružile dodatnu udobnost, pogodnosti i sigurnosne karakteristike.

Sve do pojave *CAN* magistrale vozila su sadržala ogroman broj električnih vodova, što je bilo neophodno za povezivanje svih raznolikih elektronskih komponenti. Zbog ogromne količine ožičenja, dodatna instalacija je zahtevala od proizvođača ne samo da razumeju kako integrisani sistemi komuniciraju već je zahtevala da brojne veze u vozilu budu ostvarene.

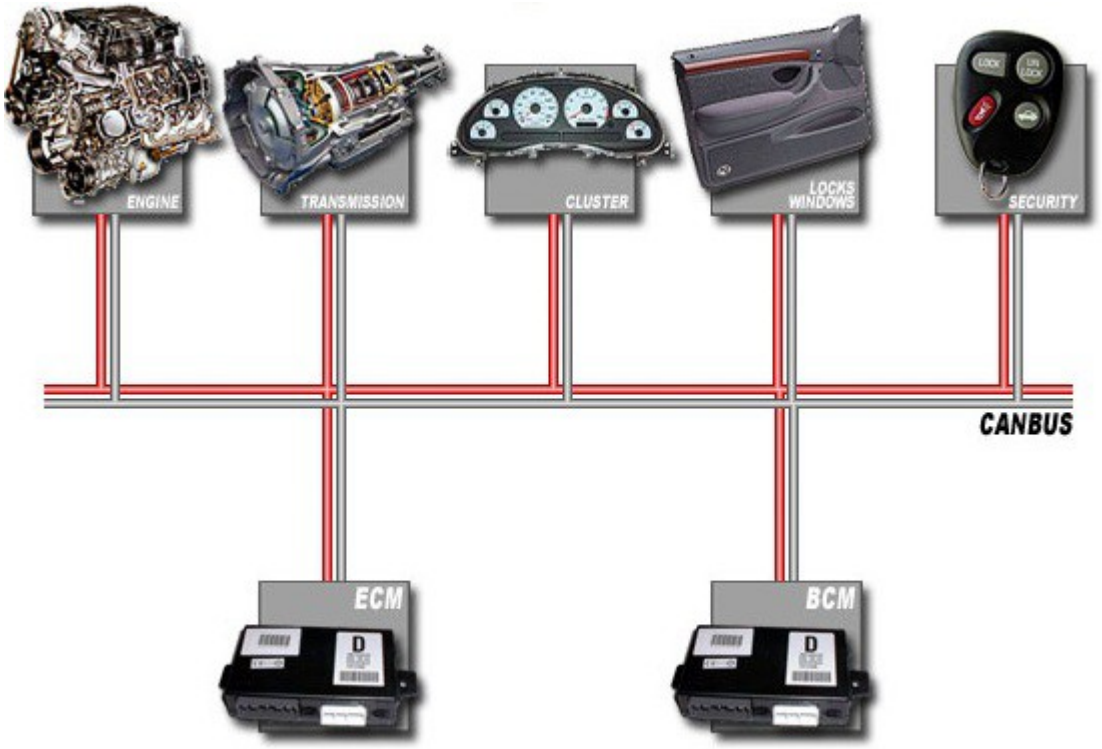
Da stvar bude gora, kako su prolazile godine ožičenje se razlikovalo između vozila. Rezultat toga je da proizvođač mora biti visoko edukovan i mora ulagati intezivan napor za većinu jednostavne dodatne opreme, ili će prodavnice iskusiti nebrojene sate izgubljenog vremena na rešavanje problema pa čak i skupih reklamacija za oštećenu opremu. Tokom ovog razvoja, specijalizovane radionice su imale sve teži zadatak pronalaska kvalifikovanog osoblja koje je sposobno da izvede svakodnevne instalacije i kao rezultat, morali su ili povećati cenu kao naknadu za zahtevano stručno usavršavanje, ili jednostavno okrenuti se od mušterija koje poseduju zahtevna vozila. [5]

Slika 2-2.Mreža u automobilu pre pojave *CAN*-a [5]

2.1.2 Pojava *CAN*-a

Prvo vozilo sa *CAN* magistralom, *BMW* 850, se pojavio na tržištu 1986. Redukovanjem 2 km ožičenja, ukupna težina vozila značajno je opala za nekih 50 kg, gde je takođe korišćeno samo polovina konektora. Po prvi put svaki od sistema i senzora u vozilu su bili u mogućnosti da komuniciraju sa velikim brzinama (25kbps- 1Mbps) na jednostranoj ili dvostranoj liniji podataka za razliku od prethodnih više-žičnih kablova. Slika 2-2 prikazuje *CAN* magistralu.

2006. godine, preko 70% svih prodatih automobila u Severnoj Americi koristi tehnologiju *CAN* magistrale. 2008. godine *Society of Automotive Engineers* (SAE) zahtevalo je da 100% prodatih vozila u Americi koriste *CAN* komunikacioni protokol, gde Evropska unija ima sličan zakon. [5]



Slika 2-3.Mreža *CAN* magistrale [5]

2.1.3 *CAN* u industriji automobila

Moderan automobil može sadržati i do 70 elektronskih kontrolnih jedinica za različite podsisteme. Tipično najveći procesor je kontrolna jedinica motora (PCM), ostali služe za kontrolu transmisije, *airbag*, sigurnosnu bravu, automatski kontroler brzine, *audio* sisteme, prozore, vrata, mala podešavanja, baterije i punjenje sistema. Neki od ovih oblika su nezavisni podsistemi, ali komunikacija između ostalih je ključna. Podsistemi kontrolišu aktuatore ili primaju informacije od senzora. *CAN* standard je osmišljen da ispuni ove potrebe. *CAN* magistrala može koristiti u vozilu da poveže kontrolnu jedinicu motora i transmisiju ili da poveže kontroler zaključavanja vrata, kontrolu klime, kontrolu sedišta itd. [6]

2.1.4 Osobine *CAN*-a

Svaki čvor je u stanju da pošalje i primi poruku ali ne istovremeno. Poruka se sastoji prvenstveno od identifikatora (*ID*), koji ujedno predstavlja prioritet poruke i sekcije podataka od osam bajtova. Poboljšana verzija *CAN-a* (*CAN FD*) produžuje sekciju podataka i do 64 bita po poruci. Signal se dekodira preko *NRZ* (*non return to zero*) i svi čvorovi ga uočavaju.

Uređaji koji su konektovani *CAN* mrežom tipično su senzori, aktuatori i drugi kontrolni uređaji. Ovi uređaji nisu povezani direktno na magistralu već preko procesora i *CAN* kontrolera.

Ukoliko je magistrala neaktivna (u stanju *idle*), što je predstavljeno naponom od 5V, bilo koji čvor može početi prenos. Ako dva ili više čvorova počnu slati poruke u isto vreme, poruka sa većim *ID*-om će preći preko poruka sa slabijim *ID*-om drugih čvorova, tako da konačno ostaju samo dominantne poruke koje primaju svi čvorovi. Ovaj mehanizam predstavlja primarnu arbitražu magistrale. Poruke sa manjim brojevima *ID*-ova imaju veći prioritet i biće poslate prve.

Svaki čvor zahteva:

- Procesor domaćina – procesor određuje šta primljene poruke znače i koje poruke žele da se šalju. Senzori, aktuatori i kontrolni uređaji mogu biti povezani sa procesorom.
- *CAN* kontroler
    1. Pri primanju – *CAN* kontroler čuva serijski primljene bite sa magistrale dok cela poruka ne bude dostupna, koju onda može preuzeti procesor.
    2. Pri slanju – Procesor čuva poslate poruke od kontrolera, koje dalje šalje serijski po bitima na magistralu.
- Transiver
    1. Pri primanju – prilagođava nivo signala sa magistrale na nivo koji *CAN* kontroler očekuje.
    2. Pri slanju – pretvara poslati signal primljen sa *CAN* kontrolera u signal za slanje na magstralu

Brzina prenosa je 1Mbit/s za razdaljinu do 40m. Za duže razdaljine brzina prenosa opada npr. za 500m prenos je 125kbit/s.

CAN koristi standardizovan *ISO* 11898-1 protokol prenosnog nivoa. [6]

2.2 *FlexRay* magistrala

Flexray je magistralni sistem za komunikaciju velike brzine, otporan na greške.

Za sigurni napredak automobila, mora doći do: povećanja učinka, brzine, količine i pouzdanosti podataka koji se prenose između jedinica elektronskih kontrolera.

Napredne kontrole i sigurnosni sistemi (kombinovani senzori, aktuatori i elektronske kontrolne jedinice) počele su da zahtevaju sinhronizaciju i učinak koji prevazilazi postojeći *CAN* standard. Nakon godina partnerstva sa *OEM* (*Original Equipment Manufacturer*), dobavljačima alata i krajnjim korisnicima, pojavio se *FlexRay* standard kao komunikaciona magistrala da odgovori novim izazovima u sledećim generacijama vozila. Potrebno je bilo vremena automobilima da se naviknu na novi mrežni standard. Dok *FlexRay* bude rešavao buduće glavne i zahtevne mrežne zahteve, on neće baciti u drugi plan druga dva standarda, *CAN* i *LIN*. Da bi došlo do optimizacije cene i olakšanja u prenosu podataka, sledeće generacije će sadržati *FlexRay* za zahtevne aplikacije, *CAN* za osnovne pokretače a *LIN* za jeftiniju elektronsku opremu. *FlexRay* je u dosta aspekata stvoren da smanji cenu dok daje dobre preformanse. [7]

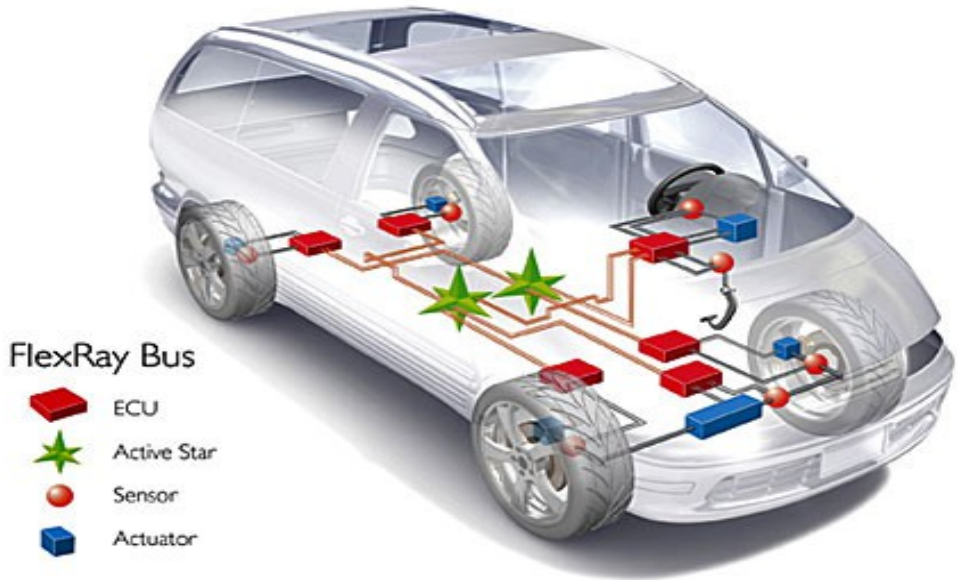
2.2.1 Osobine *FlexRay*-a

Podržava jednosmenu ili dvosmernu komunikaciju pa može sadržati jedan ili dva para provodnika. Diferencijalni signali na parovima smanjuju efekat šuma na mreži bez skupe zaštite.

Dupli kanal često povećava otpornost na greške ili povećava propusni opseg. U početku je *FlexRay* koristio samo jedan kanal ali kako su aplikacije postajale zahtevnije počelo je korišćenje dvostrukih kanala.

FlexRay magistrala zahteva terminator na krajevima, u formi otpornika između parova provodnika. Previše ili premalo otpora može da prekine *FlexRay* mrežu, tipična *Flexray* mreža ima otpor između 80 i 100 Ω. [7]

Ono što razlikuje *FlexRay*, *CAN* i *LIN* od tradicionalnih mreža kao npr. *Ethernet* je njihova topologija odnosno izgled mreže. *FlexRay* podržava jednostavnu *multi-drop* vezu (jedan provodnik povezuje više jedinica), kao i topologiju zvezde za komplikovanije mreže. Odabir prave topologije pomaže u optimizaciji cene, preformansi i pouzdanosti. Magistrala unutar automobila je prikazana na slici 2-3.



Slika 2-4.*FlexRay* unutar vozila [8]

2.3 *LIN* magistrala

LIN je mrežni serijski protokol koji se koristi za komunikaciju između sastavnih delova vozila. Kako je napredovala tehnologija i implementacija u vozilima, potreba za jeftinijim serijskim mrežama je porasla dok je *CAN* magistrala bila isuviše skupa za sve delove u automobilu. Evropski proizvđači vozila počeli su koristiti različite serijske topologije, što je dovelo do problema (ne)podudarnosti.

Proizvođači u koje spadaju *Volkswagen*, *Audi*, *BMW*, *DaimlerChrysler* i *Volvo* zajedno sa stručnjacima u komunikaciji *Volcano* i *Motorola*, uveli su *LIN* 2000. godine. Prva u potpunosti izvršena verzija nove *LIN* specifikacije (*LIN* verzija 1.3) bila je objavljena u novembru 2002. godine. U septembru 2003. godine, uvedena je verzija 2.0 da proširi mogućnosti i dodatne osobine dijagnostike. Glavni zadatak *LIN*-a je da ponudi optimizaciju cene a najviše se koristi za povezivanje senzora i kontrolnih sistema za vrata, točkove i sedišta. [7]

2.3.1 Osobine *LIN*-a

LIN je serijska mreža koja obuhvata jedan glavni i tipično do šesnaest podređenih čvorova. Sve poruke pokreće glavni, sa najmanje jednim podređenim čvorom, koji odgovara na identifikator poruke. Glavni čvor se takođe može ponašati i kao podređeni, odgovarajući na sopstvene poruke. Detekcija kolizije nije neophodna jer *glavni čvor* započinje svu komunikaciju.

Još neke osobine *LIN* magistrale:

- Jednosmerna komunikacija sa brzinom prenosa do 19.2 kbit/s.
- Zagarantovano vreme kašnjenja.
- Promenljive dužine podataka (dva, četiri i osam bajtova).
- Konfiguracijska fleksibilnost.
- Višestruki odziv sa sinhronizacijom.
- Provera podataka i otkrivanje greške.
- Otkrivanje defektnih čvorova.
- Silikonska implementacija, niže cene osnovane na standardu *UART/SCI.*
- Radni napon od 12V.

Podaci se šalju preko mreže u izabranim dužinama blokova. Zadatak glavnog čvora je prenos zaglavlja koji se sastoji od prekidnih signala praćeni sinhronizacijom i polja identifikacije. Podređeni čvor odgovara blokom podataka koji sadrže između dva, četiri i osam bajtova podataka sa još tri bajta kontrolnih informacija [9]. Neke od ključnih razlika u osobinama glavnih magistrala prikazani su na slici 2-4. [7]

Bus	LIN	CAN	FlexRay
Speed	40 kbit/s	1 Mbit/s	10 Mbit/s
Cost	$	$$	$$$
Wires	1	2	2 or 4
Typical Applications	Body Electronics (Mirrors, Power Seats, Accesories)	Powertrain (Engine, Transmission, ABS)	High-Performance Powertrain, Safety (Drive-by-wire, active suspension, adaptive cruise control)

Slika 2-5.Razlike između glavnih magistrala [7]

2.4 TTL datoteka

Podaci generisani sa vozila treba da budu preneseni kroz *Data Logger* i upisani u datoteku koja služi kao prostor za njihovo skladištenje. Nakon završetka rada, korisnik preuzima datoteke na računar za pregled i analizu. *TTL* je binarna datoteka koja se sastoji od dve sekcije:

- Sekcija zaglavlja
- Sekcija podataka

2.4.1 Sekcija zaglavlja

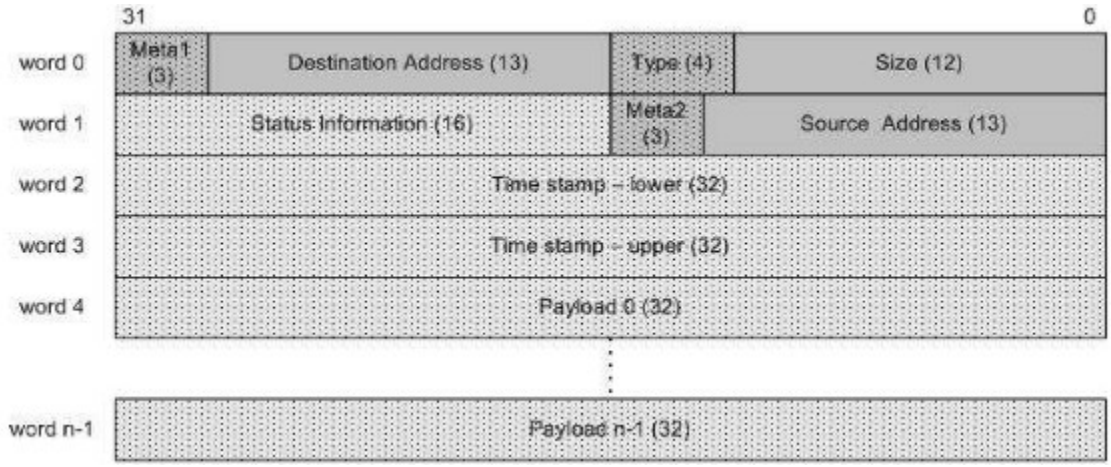
Sastoji se od polja:

- Niz znakova *TTL*
- Verzije datoteke
- Veličine bloka
- Veličine zaglavlja
- Informacije o datoteci
- *XML* konfiguracija

Najznačajnija informacija za ovaj zadatak je veličina zaglavlja (eng. *header size*) koji zauzima četiri bajta. Preskakanjem dobijene veličine stiže se do sekcije podataka.

2.4.2 Sekcija podataka

Slika 2-5 prikazuje polja sekcije podataka tipične *TTL* datoteke.



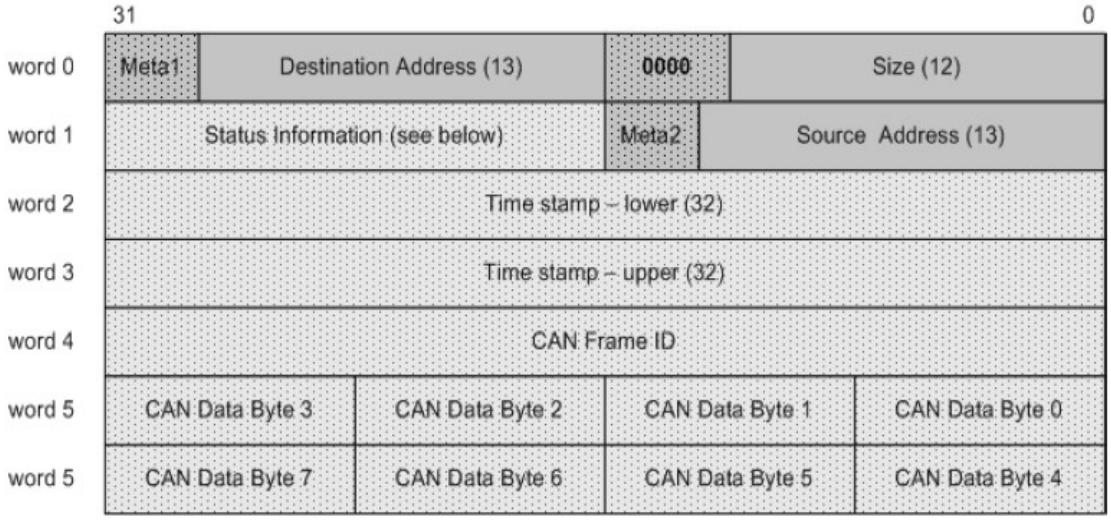
Slika 2-6.Polja sekcije podataka

Sastoji se od polja:

- *Meta_*1
- *Destination address*
- *Type*
- *Size*

- *Status information*
- *Meta_2*
- *Source address*
- *Time stamp*
- *Payload*

TTL sadrži poruke koje su sa logera pristizale u različitim vremenskim trenucima. Vreme pristizanja svake poruke je sačuvano u polju *time stamp* koje zauzima osam bajtova i predstavljeno je u mikrosekundama (us). Svaka poruka može da sadrži informaciju jednog ili više signala različitih veličina predstavljenim u bitima. Veličina jedne poruke smeštena je u polje *size.* Polje *payload* varira u odnosu na vrstu signala (*CAN*, *FlexRay, LIN*). Za tip signala *CAN*, koji je potreban za prikazivanje u zadatku, *payload* sadrži ID poruke, preko koje se poruka identifikuje, od četiri bajta i vrednost signala od maksimalno osam bajtova (slika 2-6). Tip poruke detektuje sa koje magistrale je poruka. U zadatku se parsiraju isključivo poruke sa magistrale podataka (eng. *Bus*).



Slika 2-7.Polja sekcije podataka *CAN* poruke

2.5 Konfiguraciona datoteka

Konfiguraciona datoteka sadrži dodatne informacije o svakom pojedinačnom signalu prisutnom u *TTL* datoteci. Slika 2-7 prikazuje opis signala u okviru konfiguracione datoteke.

```
<Signal>
    <Id>9</Id>
    <Tag>MO_Kuehlerluefter_2</Tag>
    <Interface>CAN</Interface>
    <Channel>1</Channel>
    <SignalDescription>
        <MessageID>1088</MessageID>
        <Startbit>1</Startbit>
        <Length>7</Length>
        <QualityBit>
            <StartBit>0</StartBit>
            <Length>1</Length>
            <TrueVal>1</TrueVal>
        </QualityBit>
        <Endiannes>Little</Endiannes>
        <Factor>1</Factor>
        <Offset>0</Offset>
        <Minimum>0</Minimum>
        <Maximum>1</Maximum>
        <Unit>m</Unit>
        <Validity></Validity>
    </SignalDescription>
</Signal>
```

Slika 2-8.Sadržaj konfiguracione datoteke za jedan signal

Svi neophodni atributi vezani za signal nalaze se u ovoj datoteci. U njih spadaju *ID* signala preko kojeg se identifikuje isti, jedinstveni naziv signala, tip signala, kanal, *ID* poruke u koju je sadržan, startni bit u poruci gde je smešten, njegova dužina, minimum, maksimum.

Tu je takođe bit vrednosti koji određuje jedinstveni izlaz iz multipleksera u slučaju da postoji više ulaznih signala na istoj poziciji u poruci.

3. Koncept rešenja

3.1 Parsiranje *TTL* i konfiguracionih datoteka

Potrebno je anilizirati željene podatke iz dve vrste datoteka: *TTL* i konfiguracione.

Pristup binarnoj *TTL* datoteci podrazumeva čitanje i pozicioniranje kroz datoteku. Zaglavlje se zanemaruje čime se pozicionira na deo datoteke koji je suštinski za parsiranje. Prvi deo poruke podrazumeva polja koja su ista za sve vrste poruka. To su polja iz sekcija podataka bez *payload*-a. Kako sva polja zauzimaju određene bite date u specifikaciji, prikupljaju se i čuvaju njihove vrednosti.

Nakon toga prikupljaju se biti i čuvaju vrednosti identifikatora poruke i vrednosti signala. Ukoliko je zadovoljen uslov da je signal tipa *CAN*, sledi njihovo pamćenje u strukturu podataka. Najpogodnija je mapa, koja umesto pretrage vrednosti podatka preko indeksiranja, vrši pretragu preko jedinstvenog ključa (mapiranje).

Paralelno se vrši i prikupljanje atributa signala koji su smešteni u konfiguracionoj datoteci. Ključni parametri su početni bit i dužina preko kojih se detektuje pozicija signala u poruci. Za iščitavanje atributa iz konfiguracione datoteke se koristi ugrađena klasa *XmlReader*. Objekti koji će sadržati ove osobine takođe se čuvaju u mapi gde će kriterijum potražnje biti ključ tj. njihov naziv (eng. *Tag*).

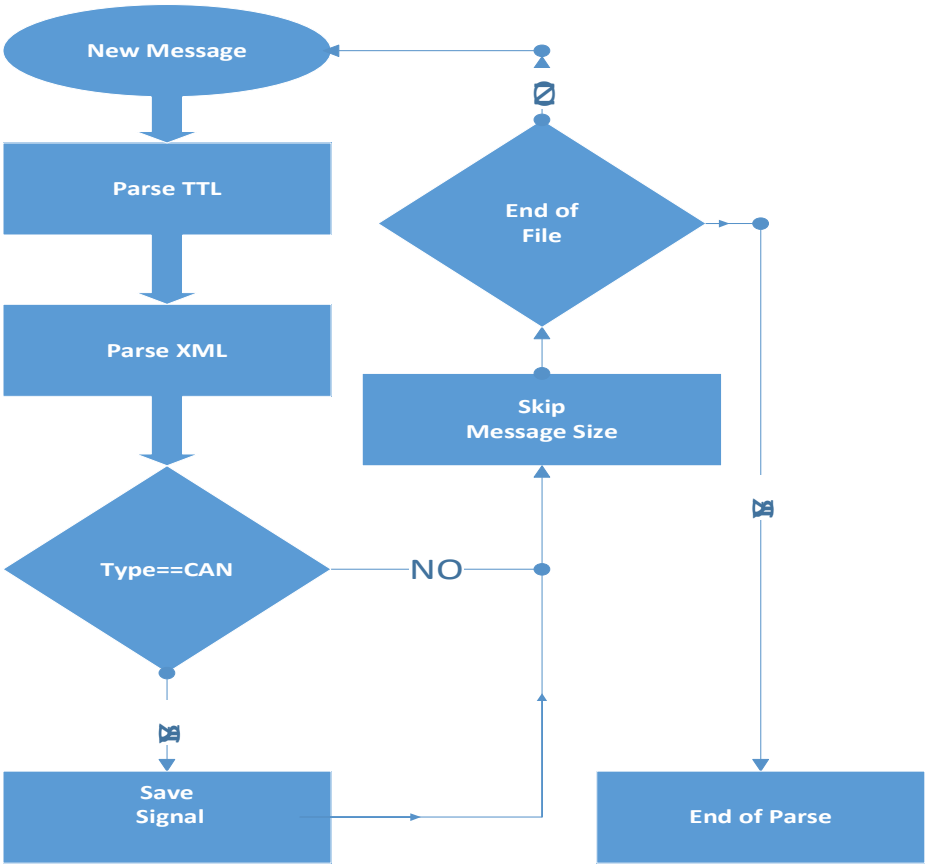
3.2 Izrada aplikacije

Sledeći korak je izrada aplikacije čija je funkcija iscrtavanje grafika i prikaz dobijenih signala. Na grafikonu se prikazuju rezultati dobavljeni iz odabrane datoteke, gde se vremena

pristiglih vrednosti postavljaju na apcisu a same vrednosti signala na ordinatu. Za izradu se koristi programski jezik *C# WPF* uz propratni alat kreiranja različitih formi grafika (eng. *toolkit*). Korisnička sprega aplikacije se sastoji od standardnih elemenata, kako bi korisnik u što kraćem vremenskom roku savladao njene funkcije. Osnovni elementi uz grafik su pomoćni alati koji daju funkcionalnost samoj aplikaciji. U njih spadaju polja za kontrolu izbora (eng. *check box*), sekcije za kontrolu izbora (eng. *combo box*),meni (eng. *menu*), dugmići (eng. *button*), prostor za iscrtavanje grafikona (eng. *canvas*) i labele (eng. *label*). Pored elemenata za funkcije programa, tu su i alati za vizuelno poboljšanje aplikacije poput slika (*images*), okvira (*border*), lista (*list box*) itd.

Radi bolje preglednosti koristi se linijski grafik. Aplikacija sadrži opcije za kontrolu grafika u koje spadaju: uvećanje za željeni iznos, pomeranje na početak i kraj grafikona, pomeranje za dva procenta u oba smera i prikaz sa svim elementima grafika ili samo određenih (linije i tačke).

Od dodatnih informacija program treba da ima nazive datoteka koje je korisnik odabrao za parsiranje i dodatne informacije o odabranom signalu gde spadaju minimum, maksimum, dužina, magistrala i kanal. Pomoćni alat za ove ispise su labele.



Slika 3-9.Algoritam koncepta rešenja

Slika 3-1 ilustruje blok dijagram koncepta rešenja. Pristizanjem nove poruke parsiraju se *TTL* i konfiguracione datoteke nakon čega se proverava tip magistrale podataka. U koliko je magistrala tipa *CAN* , podaci i vremena signala se pamte. U drugom slučaju očitava se nova poruka. Procedura se ponavlja sve dok se ne stigne do kraja datoteke.

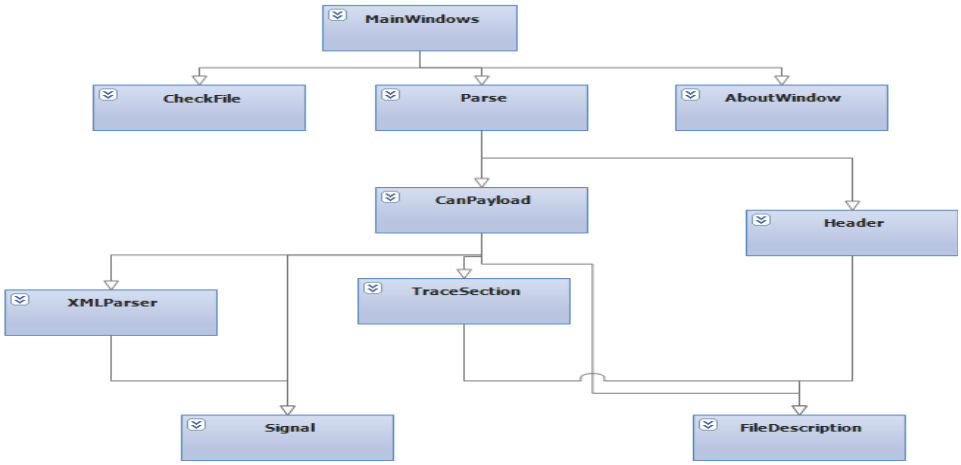
4. Programsko rešenje

Programsko rešenje je realizovano u programskom jeziku C#. Program je prezentovan pomoću WPF-a, koji predstavlja grafički podsistem za opisivanje korisničke sprege aplikacija koje su predviđene za *Windows* operativni sistem. Razvojno okruženje je *Visual Studio* 2010.

Projekat se sastoji od sledećih klasa:

- *File Description*
- *Header*
- *Trace Section*
- *Can Payload*
- *Parse*
- *XML Parse*
- *Signal*
- *Check File*
- *Main Window*

Zavisnost između klasa je prikazana na slici 4-1. Ilustrovan je odnos koristi različitih metoda između navedenih klasa.



Slika 4-10.Dijagram hijerarhije klasa

4.1 Klasa FileDescription

Modul sadrži razne funkcionalnosti vezane za čitanje binarne datoteke:

`void openFile()`	Metoda za otvaranje datoteke koja koristi binarni čitač iz biblioteke *System.IO*
`void setFileName(string fileName)`	Metoda za postavljanje naziva otvorene datoteke
`int readByte()`	Čitanje jednog bajta
`void readBytesBuff(int num, byte[] buffer)`	Čitanje više bajtova gde je taj broj predstavljen prvim parametrom uz njihovo smeštanje u deo memorije (drugi parametar)
`long position()`	Vraća vrednost trenutne pozicije
`void seekFromBeg(int offset)`	Pozicioniranje od početka datoteke sa dodatnom razdaljinom koja je zadata parametrom
`long length()`	Dužina datoteke
`void closeFile()`	Zatvaranje datoteke

4.2 Klasa Header

Sadrži polje *headerSize* koje predstavlja veličinu zaglavlja *TTL* datoteke. Uloga modula je računanje zaglavlja *TTL* datoteke (zaglavlje se tokom parsiranja podataka preskače).

`void` **`parseHeader()`**

Računa veličinu zaglavlja tako što preskače prvih dvanaest bajtova preko metode `seekFromBeg()` i smešta četiri bajta, koji predstavljaju veličinu zaglavlja, u polje *headerSize*. Deo memorije će dobiti četiri bajta i polju će se dodavati po osam bita dok ne prođe petlja za poslednji bajt. Pri dodavanju bita vrši se i pomeranje u levu stranu radi premeštanja bita na niže pozicije.

4.3 Klasa TraceSection

U ovom modulu su definisana polja koja su vezana za sekciju podataka *TTL* datoteke: polja *meta_*1*, destiantionAddress, type, size, statusInformation, meta_*2*, sourceAddress, timeStamp, sourceAddrInterface* i polje za proveru završetka datoteke *endOfFile*. Zadatak modula je dodela vrednosti iz sekcije podataka.

`void` `parseTraceSection(int headerSize, int signalSize)`

Sadrži niz od dva memorijska prostora koji će ponaosob definisati po jednu reč iz sekcije podataka i imati njihove bajte (*word_array*[]). Niz *word_array*[0] sadrži bite sledećih polja: *meta1* (tri bita), *destination address* (trinaest bita), *type* (četiri bita), *size* (dvanaest bita) (slika 4-2). Niz *word_array*[1] sadrži bite sledećih polja: *status information* (šesnaest bita), *meta2* (tri bita), *source address* (trinaest bita) (slika 4-3).

Slika 4-11.*Word_array*[0]

Slika 4-12.*Word_array*[1]

Tu je još i memorijski prostor za *time stamp*. Nakon dodavanja bita pojedinačno u elemente niza, vrši se njihovo pomeranje ka nižim pozicijama za broj bita koji zauzimaju polja. Nakon pomeranja biti se maskiraju, čije su maske navedene u nabrajanju (*enumeration*), i konačna vrednost se dodaje istoimenim poljima. Isti postupak se radi i za polje *time stamp*.

4.4 Klasa CanPayload

U modulu su definisana polja *messageID*, *data*, dve mape *objectSignalMap* i *signalMap* i dva reda *objectSignalQueue* i *signalInformationQueue***.** Uloga modula je pronalazak i čuvanje identifikatora poruke kao i izdvajanje signala iz poruka.

void `fillObjectSignalMap()` Za svaki objekat u redu koji je isparsiran u *XML* modulu vrši se provera da li u mapi sa objektima iz modula *Signal* već postoji elemenat sa istim ključem (*ID* poruke). Ukoliko ne postoji sačuvaj u mapu *ID* poruku kao ključ a u vrednost sačuvaj red sa svim objektima signala (*objectSignalQueue*) koji se nalaze u toj poruci.
void `fillSignalMap(Signal obj, long parsedData)` Puni mapu *signalMap* koja u ključu sadrži naziv signala izdvojen iz modula *XML* a za vrednost sadrži red koji ima u sebi *time stamp* i podatak (*signalInformationQueue*) koji je u tom trenutku pristigao. Ukoliko ne sadrži signal sa traženim ključem, sačuvaj ga u mapu.
long `getMask(int length)` Pravi masku u zavisnosti od vrednosti parametra. Poslednji bit se prebacuje na jedinicu i pomera za jedno mesto u levo. Postupak se ponavlja sve dok se ne dobije broj jedinica jednak broju prosleđenom parametrom.
long `parseData(Signal obj, long data)` Vraća vrednost signala na osnovu njegovog startnog bita i dužine u poruci. Biti celokupne poruke se pomeraju u desno za broj dobijen formulom *messageLength* – (*startBit* - 1). Kada je vrednost konkretnog signala pomerena na najniže bite, radi se maskiranje preko prethodne metode. Povratna vrednost je vrednost signala.
void `canPayload(int headerSize, int messageSize)`

Dodeljuje vrednosti poljima *messageID* i *data* na isti način kao i metoda *parseTraceSection()* iz modula *TraceSection*. U nastavku sledi provera da li u mapi sa objektima signala postoje signali koji daju ulaze na isti multiplekser. Tu informaciju sadrži *quality* bit koji je parsiran iz konfiguracione datoteke. Kontrolu postojanja će javiti promenljiva *getIsQualityBit()* koja ako detektuje postojanje multipleksera vraća vrednost na *true.*

Potom se proverava koji od signala daje izlaz na multiplekseru. Vrši se provera najnižih bita i vrednosti *trueVal* koja predstavlja atribut signala isparsiran u *XML* modulu. Ukoliko su vrednosti iste, trenutni signal će u tom vremenskom trenutku dati vrednost, ukoliko nisu vrši se provera sledećeg signala.

4.5 Klasa Parse

Polje *newSignalOffset* nosi informaciju o udaljenosti od početka datoteke u svrhu parsiranja sledeće poruke. Modul ujedinjuje *TTL* i *XML* parsiranje.

```
void skipSignal()
```

Iterativno povećava promenljivu *newSignalOffset* za trenutnu dužinu poruke koja je dobijena u postupku parsiranja iz metode *TraceSection*. Tim postupkom automatski se prebacuje na sledeću poruku.

```
void parseSignal()
```

Glavna metoda, ujedinjuje *XML* i *TTL* parsiranje. Na početku otvara datoteku i zanemaruje zaglavlje preko modula *FileDescritption* i *Header*. Potom poziva metodu parsiranja *XML* datoteke i popunjava mapu objektima signala. Konačno parsira *TTL* datoteku pozivajući ranije navedene metode iz modula *TraceSection* i *CanPayload* uz uslove da nije pretražena cela datoteka i da tip poruke bude *CAN*.

4.6 Klasa XML Parse

Modul za parsiranje konfiguracione datoteke. Koristi ugrađenu klasu *XmlReader* koji pripada biblioteci *System.Xml.*

```
void xmlParse()
```

Preko objekta *reader* klase *XmlReader* spušta se na različite hijerarhijske nivoe i skuplja atribute signala.

4.7 Klasa Signal

Modul za stvaranje objekata signala sa svim svojim atributima:

Id	Brojčani identifikator signala
bus	Naziv magistrale sa koje pristiže signal
channel	Broj kanala
startBit	Početni bit vrednosti signala u poruci
length	Dužina u bitima vrednosti signala
messageID	Brojčani identifikator poruke
tag	Naziv signala
factor	Faktor optimizacije vrednosti u svrhu smeštanja vrednosti u određen broj bita
minimum	Minimalna vrednost signala
maximum	Maksimalna vrednost signala
qualityStartBit	Početni bit stanja izlaza na multiplekseru
qualityLength	Dužina u bitima vrednosti stanja izlaza na multiplekseru
trueVal	Bit(ovi) stanja signala na izlazu multipleksera (u slučaju da je ova vrednost jednaka bitima koji su pomoću *qualityStartBit* i *qualityLength* izdvojeni iz poruke, trenutni signal je na izlazu iz multipleksera)

Svaki atribut poseduje odgovarajuće metode za postavljanje i očitavanje

4.8 Klasa Check File

Modul za postavljanje i proveru prethodne i trenutno izabrane datoteke.

void setOldFile(string oldF)	Pamti trenutno aktiviranu *TTL* datoteku za parsiranje pred promenu na sledeću
void setNewFile(string newF)	Pamti naziv novo izabrane *TTL* datoteke
void setIsCreatedObject(bool isObjC)	Ukoliko su kreirana polja za kontrolu izbora signala na osnovu prethodno odabranog TTL datoteke, izbriši ih iz liste (lista je prazna za polja sledeće datoteke)
void getOldFile()	
void getNewFile()	
void getIsCreatedObject()	
bool compareFiles()	provera da li je došlo do promene izbora na drugu *TTL* datoteku, tj poredi *oldFile* i *newFile*

4.9 Klasa Main Window

Modul koji pokriva vizuelni prikaz korisničke aplikacije uz sve njene funkcionalnosti.

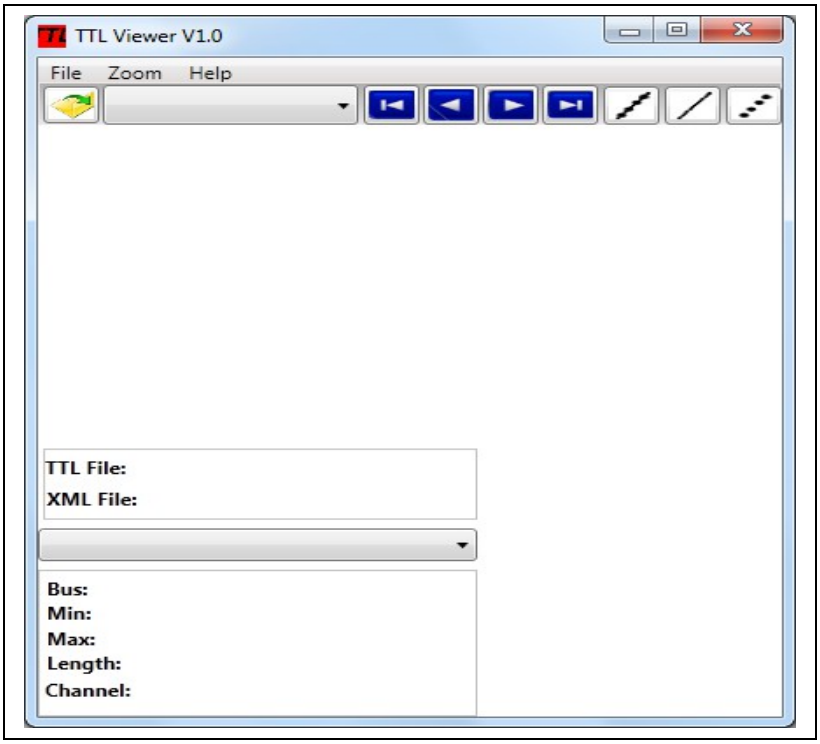
void change SizeofWindow() Kreira nit koja na jednu sekundu proverava visinu i dužinu prostora za isrctavanje. Ukoliko korisnik smanji veličinu prozora, automatski se smanjuje i prostor za iscrtavanje grafikona pa će doći do ažuriranja i veličine grafika.
void createCheckBox() Dinamički kreira polja za kontrolu izbora signala (eng. *check box*) srazmerno broju nađenih signala u parsiranim datotekama. Njihova uloga je iscrtavanje grafika na prostoru za iscrtavanje (eng. *canvas*). U zavisnosti od broja označenih polja, broj grafika koji prikazuje aplikacija je promenljiv. Svaki od njih nosi naziv jednog od signala kojeg reprezentuje.
void checkbox_Click(object sender, EventArgs e) Procedura se aktivira na klik jednog od polja za kontrolu izbora signala i sadržaj koji on

predstavlja tj. signal ubacuje u mapu za iscrtavanje (*drawMap()*). Sledi provera koliko je grafik/grafici uvećan i njegovo konačno iscrtavanje.
void `openFile_Click(object sender, EventArgs e)` Prikazuje na ekranu dijalog za odabir jedne od ponuđenih datoteka za obradu podataka. Prikazuje ime trenutne datoteke i poziva navedenu funkciju za dinamičko kreiranje polja za kontrolu izbora signala (`createCheckBox()`).
void `drawOnCanvas(object sender, EventArgs e)` Glavna procedura odgovorna za kreiranje grafika. Pored podešavanja izgleda grafika tu je i algoritam za iscrtavanje više od jednog grafika. Prostor za iscrtavanje se deli srazmerno broju označenih signala. Ukoliko su označena tri, prostor za iscrtavanje se deli na tri dela, gde će svaki deo zauzimati po jedan grafik itd. Podržava i opcije za različite varijante prikaza grafika kao što su linijski sa tačkama koje predstavljaju vrednosti, samo linijski ili samo tačke. Pri svakom pozivu se briše stari prostor za iscrtavanje i vrši se njegovo ažuriranje.
void `checkZoomValue()` Podešava vrednost promenljive *zoom* na osnovu akcije klika na jednu od opcija sekcije za kontrolu izbora uvećanja (eng. *combo box*)
void `zoomMenuItem_Click(object sender, EventArgs e)` Predstavlja stavku u listi *zoom* koja na odabir jedne od opcija korisnik menja vrednost sekcije za izbor uvećanja, zaduženu za istu funkcionalnost. Automatski dolazi do uvećanja grafika.
void `comboBox_Zoom(object sender, EventArgs e)` Uvećava grafik za izabrani procenat od strane korisnika. Uzima se vrednost početne i krajnje tačke čija će razlika predstavljati razdaljinu. Zatim se razdaljina množi sa promenljivom *zoom* koja dobija vrednost izabranu od strane korisnika preko sekcije za kontrolu izbora uvećanja. Ukoliko je razlika trenutne tačke i početne tačke tog signala manja od razdaljine, ona ulazi u opseg i biće iscrtana u novoj mapi (*drowMapZoomedAndMooved*). Sledeći korak je poziv procedure za iscrtavanje ažuriranog grafika.
void `moveRightLeft_Click(object sender, EventArgs e)` Procedura pomera grafik u željenom smeru (u zavisnosti od odabira dugmeta na koji se klikne, levo ili desno). Pomeraj se obavlja za dva procenta od ukupne dužine grafika,a zatim se računa razdaljina koja je jednaka razlici početne i krajnje tačke i za razliku od procedure za uvećanje grafika, ova procedura sadrži i pomeraj *delta*. Pomeraj je jednak proizvodu razdaljine i početnog procenta koji se povećava za 0.02 na svaki sledeći klik na dugme za

pomeranje u istom smeru. Zbir početne i krajnje tačke sa pomerajem daje nove granice, gde ukoliko vrednosti pripadaju tom opsegu, ubacuju se u mapu i konačno bivaju iscrtane.
void `moveToEnd(object sender, EventArgs e)` Pomera grafik na sam kraj ili početak u zavisnosti od odabira dugmeta, levo ili desno. Princip je sličan kao i pomeranje za dva procenta osim što se pamti razdaljina od uvećanog dela (ukoliko je uvećan) i razdaljina celokupnog grafika. Njihova razlika će dati pomeraj koji se dodaje na početnu i krajnju tačku signala što će činiti opseg za proveru pripadnosti vrednosti signala.
void `showName()` Ispisuje nazive odabranih datoteka za parsiranje (*TTL* i *XML*). Odbacuje nepotrebne delove iz putanja koje vode do datoteka i čuva samo njihova imena.
void `onSelectionChanged(object snder, EventArgs e)` Ispisuje atribute u zavisnosti od odabranog signala iz sekcije za kontrolu izbora signala.
void `aboutMenuItem_Click()` Poziva prozor koji prikazuje dodatne informacije o programu (modul *About Window*).

4.10 Opis korisničke sprege

U opisu korisničke sprege prikazani su izgled aplikacije, njene komponente i funkcije koje obavljaju. Slika 4-4 predstavlja izlgled aplikacije pri njenom pokretanju.

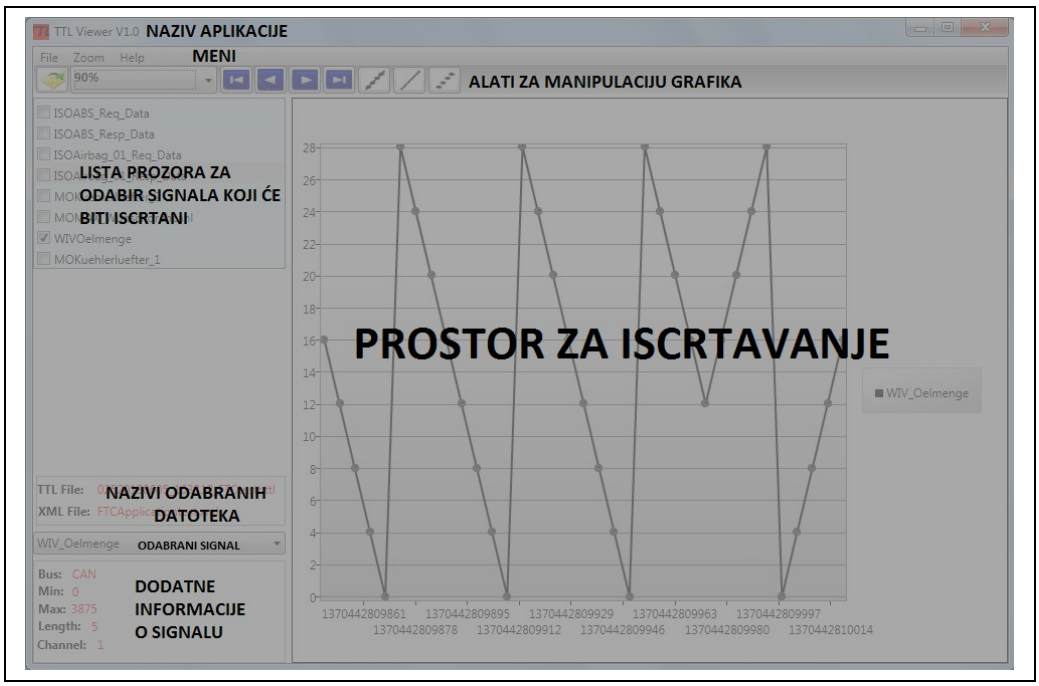


Slika 4-13.Izgled aplikacije nakon pokretanja

Pojedini elementi grafičke korisničke sprege će biti opisani u narednim sekcijama.

4.10.1 Vizuelni elementi

U aplikacione komponente spadaju: meni, alati za manipulaciju grafika, lista polja za kontrolu izbora signala koji će biti iscrtani, nazivi odabranih datoteka, dodatne informacije o signalu i prostor za iscrtavanje grafikona (slika 4-5).



Slika 4-14.Raspored pojedinih oblasti aplikacije

4.10.2 Meni

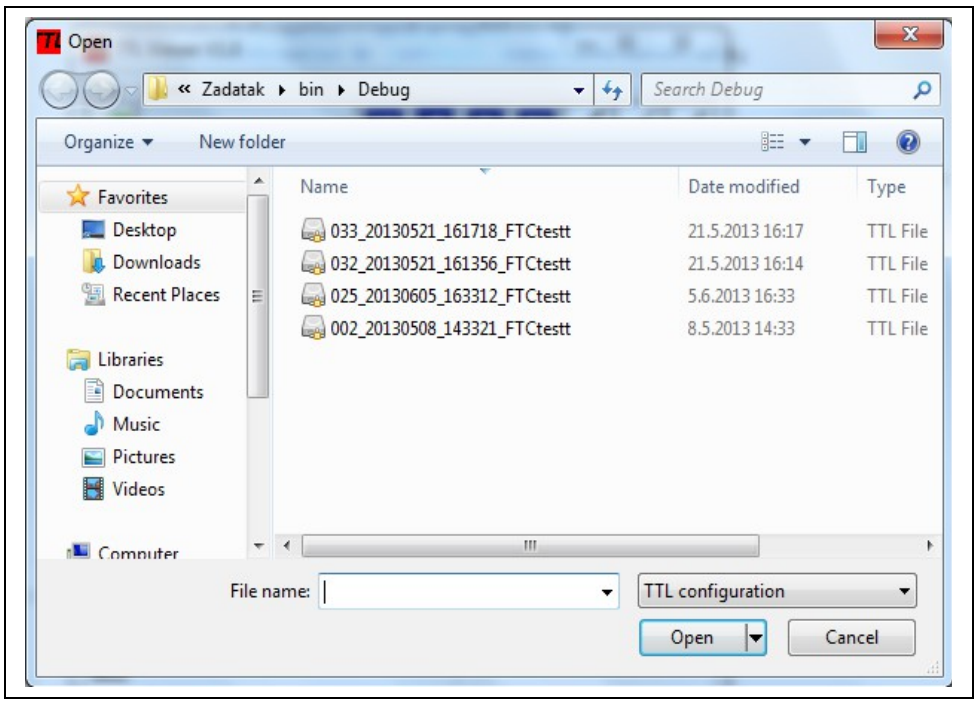
U meni alat spadaju tri kartice:

- ***File***
- ***Zoom***
- ***Help***

File - Nudi dve opcije:

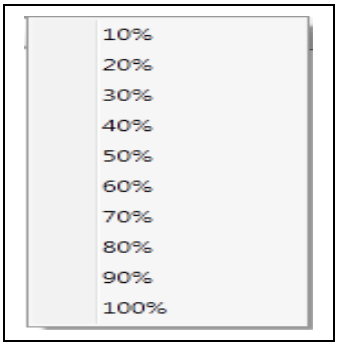
- ***Open*** - Otvara dva prozora gde korisnik bira *TTL* i konfiguracione datoteke za obradu podataka. Slika 4-6 prikazuje mogućnost odabira *TTL* datoteke nakon sto korisnik klikne na opciju *open*.

- ***Exit*** - Završetak rada i izlazak iz aplikacije.



Slika 4-15.Opcija *open* za odabir *TTL* i konfiguracionih datoteka

Zoom - Opcija odabira procenta za koliko će se uvećati grafik. Povezan je i sa sekcijom za odabir (eng. *combo box*) uvećanja koji ima istu ulogu (sekcija za odabir uvećanja automatski menja vrednost na onu odabranu u *zoom* kartici). Pomeranjem miša na opciju *zoom*, korisniku se ukazuje mogućnost odabira procenta uvećanja grafikona (slika 4-7). Klikom na neki od njih izabrano je željeno uvećanje.



Slika 4-16.Ponuđene opcije posle klika na *zoom*

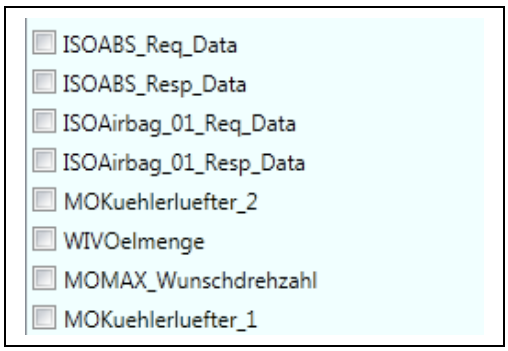
Help - Nudi opciju *about:*

- ***About*** **–** Prikazuje dodatne informacie o programu, slika 4-8.

Slika 4-17.Prikaz informacija o programu posle klika na opciju *about*

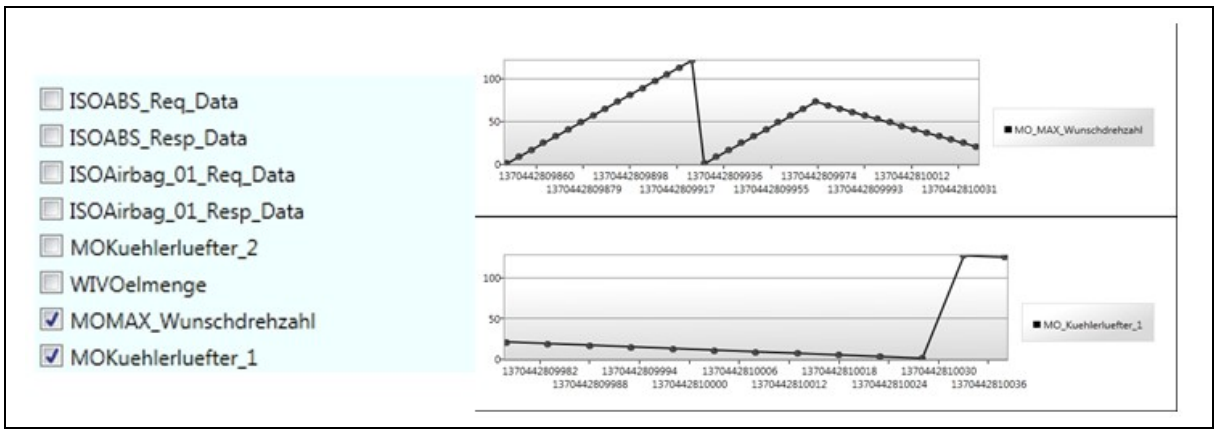
4.10.3 Lista prostora za odabir signala i prostor za iscrtavanje

Posle izbora datoteka automatski se prikazuju polja za kontrolu izbora (eng. *check box*) signala koji su pronađeni u datoteci. Korisnik mišem može selektovati signale koje želi da prikaže na grafikonu. Nakon što korisnik odabere željenu *TTL* datoteku, automatski se generišu polja za kontrolu izbora koji predstavljaju signale nađene u njoj (slika 4-9).



Slika 4-18.Izgled polja za kontrolu izbora nakon otvaranja datoteka

Nakon klika na nekog od signala vrši se njegovo prikazivanje na prostoru za iscrtavanje grafikona (eng. *canvas*) što je prikazano na slici 4-10. Korisnik može da odabere jedan ili više signala istovremeno.



Slika 4-19.Iscrtavanje grafika nakon klika na polja za kontrolu izbora

4.10.4 Alati za manipulaciju grafika

U alate spadaju opcije za odabir datoteka, uvećavanje grafikona kao i alati za njegovo pomeranje:

 - Ista funkcija kao i *File*, otvara dva prozora gde korisnik bira *TTL* i konfiguracione datoteke za obradu podataka.

- Ista uloga kao i *zoom*, uvećava grafik za izabrani procenat.

 - Pomeranje grafika na početnu poziciju ukoliko je došlo do uvećanja.

 - Pomeranje grafika u levu stranu za dva procenta na svaki klik.

 - Pomeranje grafika u levu stranu za dva procenta na svaki klik.

 - Pomeranje grafika na krajnju poziciju ukoliko je došlo do uvećanja.

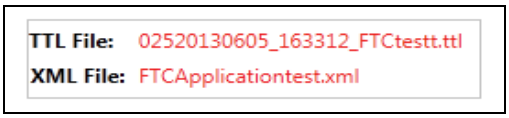
 - Prikazivanje grafika sa linijama i tačkama.

 - Prikazivanje grafika samo linijama.

 - Prikazivanje grafika u tačkama.

4.10.5 Nazivi odabranih datoteka

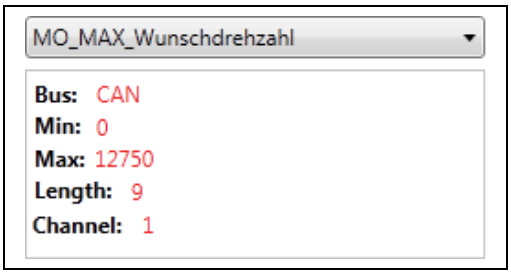
Prikazivanje naziva odabranih datoteka za parsiranje (konfiguracione i *TTL*). Prikaz naziva se ispisuje labelama koje se nalaze u donjem levom uglu aplikacije, slika 4-11.



Slika 4-20.Nazivi odabranih datoteka

4.10.6 Dodatne informacije o signalu

Obeležavanjem signala koji korisnik želi da prikaže pruža se mogućnost prikazivanja dodatnih informacija o signalu.



Slika 4-21.Dodatne informacije o signalu

Samo jedan od njih može da bude prikazan u jednom trenutku. Odabir signala se izvodi preko sekcije za kontrolu izbora (eng. *combo box*). Slika 4-12 prikazuje informacije jednog od odabranih signala koji su iscrtani na grafikonu.

5. Postupak ispitivanja

Za postupak verifikacije koristi se ugrađeni alati za proveru u okviru *Visual Studio* okruženja. Pravi se novi *test* projekat čije će metode ispitivati tačnost željenih metoda iz zadatka, na osnovu očekivanih rezultata. Izgled test projekta i glavne klase dat je na slici 5-1,

```
7  namespace TestProject1
8  {
9      [TestClass]
10     public class UnitTest1
11     {
12         [TestMethod]
13         public void TestMethod1()
14         {
15         }
16     }
17 }
```

Slika 5-22.Početna *test* klasa

U okviru *TestMethod* pišu se metode provere. Njihov broj je neograničen, *TestMethod* se navodi pre svake nove metode, potrebno je navesti očekivane rezultate i nakon toga dolazi do poređenja rezultata iz testiranog programa i postavljenih očekivanih.

Provera se izvodi pomoću klase *Assert* i njene metode *AreEqual()* koja će proveriti tačnost podudarnosti ove dve vrednosti. Ukoliko nije došlo do poklapanja, *Assert* izbacuje mesto nepodudarnosti.

Greške pri odabiru datoteka (*TTL* ili konfiguracione), koje korisnik želi da prikaže, su sprečene korišćenjem filtera ekstenzije. Ukoliko je potrebno odabrati *TTL* datoteku, dijalog će ponuditi samo datoteke sa tom ekstenzijom. Isto važi i za odabir konfiguracione datoteka, program prikazuje novi dijalog samo sa ekstenzijom *.xml*. U slučaju očitavanja neispravne datoteke dijalog će upozoriti korisnika o nastalom problemu.

Problem promene veličine grafikona nakon umanjivanja ili povećavanja prozora aplikacije je rešen sistemskom niti koja na svaki sekund proverava visinu i dužinu prostora za iscrtavanja i kada dodje do promene veličine prozora aplikacije, menja se i veličina grafikona.

Ime testa	**Opis koraka testiranja**	**Očekivani rezultati**
Pravilan odabir ekstenzije datoteke	Nakon otvaranja aplikacije i izbora opcije *open* pojavljuje se dijalog za odabir datoteke	Filter generise u dijalogu samo datoteke sa „.ttl" ekstenzijom (ili ".xml")
Test sa neispravnom datotekom	Otvori se dijalog za odabir datoteka i učita se neispravna datoteka	Aplikacija generiše dijalog sa upozorenjem
Test sa datotekom dužine 0	Otvori se dijalog za odabir datoteka i učita se nepostojeća datoteka	Aplikacija generiše dijalog sa upozorenjem
Metoda void parseTraceSection_test()	Provera poklapanja dobijenih i očekivanih rezultata u okviru test metoda, *Visual Studio*	Tačnost je ispunjena u koliko je došlo do poklapanja rezultata dobijene i očekivane vrednostu
Metoda void canPayload_test()	Provera poklapanja dobijenih i očekivanih rezultata u okviru test metoda, Visual Studio	Tačnost je ispunjena u koliko je došlo do poklapanja rezultata dobijene i očekivane vrednosti
Metoda void xmlParse_test()	Provera poklapanja dobijenih i očekivanih rezultata u okviru test metoda, Visual Studio	Tačnost je ispunjena u koliko je došlo do poklapanja rezultata dobijene i očekivane vrednosti

Tabela 5-1.Pregled testnih slučajeva

Ime testa	**Dobijeni rezultati**
Pravilan odabir ekstenzije datoteke	
Test sa neispravnom datotekom	

Test sa datotekom dužine 0	
Metoda void parseTraceSection_test()	
Metoda void canPayload_test()	
Metoda void xmlParse_test()	

Tabela 5-2.Rezultati ispitivanja

Na osnovu rezultata testiranja zaključuje se da realizovana aplikacija ispunjava osnovne slučajeve ispitivanja tačnosti rada.

6. Zaključak

U zadatku rada je potrebno bilo anilizirati željene podatke iz dve vrste datoteka, *TTL* i konfiguracione, a zatim i izrada aplikacije čija je funkcija iscrtavanje grafika i prikaz dobijenih signala. Na grafikonu se prikazuju rezultati dobavljeni iz odabrane datoteke, gde se vremena pristiglih vrednosti postavljaju na apcisu a same vrednosti signala na ordinatu. Za izradu se koristi *C# WPF* uz propratni alat kreiranja različitih formi grafika (eng. *toolkit*).

Ispravan rad progama je potvrđen nizom testova. *Unit test* je pokazao ispravnost rada parsiranja datoteka, odabir neispravnih i praznih datoteka je sprečen dijalogom upozorenja. Izbor datoteka željene ekstenzije je ostvaren preko filter dijaloga (generiše samo istoimene ekstenzije, „.ttl“ ili „.xml“).

Program na osnovu datoteka pamti i čuva željene informacije o signalima koje kasnije prikazuje na osnovu želja korisnika. Omogućava svim korisnicima *Data Loggera*-a prikaz zapisanih signala sa vozila bez korišćenja licenciranih i skupih aplikacija. Pojednostavljuje kontrolu različitih sigurnosnih uređaja u automobilu preko vizualne predstave njihovih signala.

U budućnosti je moguće dodati još novih alata koje će proširiti aplikaciju i dati joj nove mogućnosti za prikaz i manipulaciju grafika. Takođe potrebno je proširiti program u vidu parsiranja informacija pristiglih sa *FlexRay* i *LIN* magistrale. Signale je moguće prikazivati na različite načine, a implementiran je samo jedan od njih. Vizuelizacija signala u realnom vremenu takođe je jedan od pravaca za proširenje mogućnosti ove aplikacije.

7. Literatura

[1] "*Scientific and Tehnical Societies of the United States*" (Eighth ed.) Washington DC: National Academy of Sciences. 1968. p. 164.

[2] "*Changes To Electronics Found In Cars Over The Last 100 Years*", http://www.thepeoplehistory.com/carelectronics.html

[3] *A Robust and Comprehensive Automotive Data Logger*, http://www.tttech.com/products/, 2013 TTTech Computertechnik AG

[4] "*XML 1.0 Origin and Goals*", http://www.w3.org/TR/REC-xml/#sec-origin-goals, Retrieved July 2009.

[5] "*CAN BUS*", http://canbuskit.com/what.php

[6] "*CAN History*", http://www.can-cia.de/index.php?id=161

[7] "*FlexRay Automotive Communication Bus Overview*", http://www.ni.com/white-paper/3352/en/, Avgust 2009

[8] "*FlexRay Bus*", http://www.automotive-industry.nl/page/wb/ee-architectures.php

[9] "*Clemson Vehicular Electronics Laboratory: AUTOMOTIVE BUSES*", 090114 cvel.clemson.edu