Optimizacija algoritma za manipulaciju paketima u uređajima za pristup i usmjeravanje u automobilskim komunikacijama

Janjić, Zlatko

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNALSTVA I INFORMACIJSKIH TEHNOLOGIJA

Optimizacija algoritma za manipulaciju paketima u uređajima za pristup i usmjeravanje u automobilskim komunikacijama

Diplomski rad

Zlatko Janjić

Osijek, 2017.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 21.09.2017.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Zlatko Janjić
Studij, smjer:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D 979, 12.10.2015.
OIB studenta:	94834835803
Mentor:	Izv. prof. dr. sc. Marijan Herceg
Sumentor:	
Sumentor iz tvrtke:	Josip Posavi
Predsjednik Povjerenstva:	Doc.dr.sc. Ratko Grbić
Član Povjerenstva:	Doc.dr.sc. Mario Vranješ
Naslov diplomskog rada:	Optimizacija algoritma za manipulaciju paketima u uređajima za pristup i usmjeravanje u automobilskim komunikacijama
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak diplomskog rada:	Današnji automobili opremljeni su nizom različitih upravljačkih jedinica koje u ovisnosti o namjeni i složenosti komuniciraju kroz različita komunikacijska sučelja. Zbog toga se pojavila potreba za spregom između raznorodnih uređaja kao i potreba za manipulacijom signala koji se na taj način razmjenjuju. Ova manipulacija unosi određenu nadogradnju i optimizacija istog je od ključnog značaja u modernim stvarno-vremenskim sustavima. Zadaci rada su: analiza i/ili optimizacija algoritma za manipulaciju; matematičko određivanje maksimalne
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	21.09.2017.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis: Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 25.09.2017.

Ime i prezime studenta:	Zlatko Janjić
Studij:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D 979, 12.10.2015.
Ephorus podudaranje [%]:	0

Ovom izjavom izjavljujem da je rad pod nazivom**: Optimizacija algoritma za manipulaciju paketima u uređajima za pristup i usmjeravanje u automobilskim komunikacijama**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Marijan Herceg

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.

Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Velik broj elektronskih uređaja u automobilu rezultirao je stvaranjem novog problema – problem povezivanja odnosno komunikacije istih. Proizvođači automobila su eksponencijalno povećavali broj potrebnih vodiča i releja koji su upravljali elektronskim uređajima. Primjerice; podizanjem/spuštanjem suvozačevog prozora može upravljati prekidač na suvozačevim vratim, prekidač na vozačevim vratima, gumb za blokadu prozora na vozačevim vratima i sustav za zaključavanje vozila (zatvaranje prozora pri zaključavanju). Realizacija navedene logike rezultirala bi velikim brojem vodiča i releja za samo jedan prozor čime se povećava masa vozila ali i zauzeti volumen.

Opće prihvaćeno rješenje problema je u korištenju komunikacijskih sabirnih vodova. Elektronski uređaji pojedine logičke cjeline automobila povezane su pojedinim sabirnim vodovima. Na razini sklopovlja, sabirni vodovi su, u osnovi električni vodiči (najčešće 2 vodiča) a na razini programa – složenije programske strukture koje zahtijevaju svoje mikro upravljače. Zbog sigurnosti ali i prioriteta u komunikaciji, automobili su podijeljeni u logičke komunikacijske cjeline gdje svaka cjelina predstavlja jednu sabirnicu. Najčešće korištene sabirnice su CAN (*eng. Controller Area Network*), FlexRay, MOST (*eng. Media Oriented Systems Transport*) i LIN (*eng. Local Interconnect Network*). Zbog velikih razlika pojedinih standarda, navedene sabirnice ne mogu komunicirati bez uređaja posrednika – Poveznik (*eng. Gateway*).

U ovom radu bit će opisano ispravljanje grešaka, mjerenje performansi i optimizacija poveznika u vlasništvu instituta RT-RK Osijek d.o.o. Poveznik ima mogućnost rada sa CAN, CAN FD (*eng. CAN with Flexible Data - rate*) i FlexRay standarda, postavke mu se zadaju TTX-Connexion standardom postavki usmjeravanja i izvršava se na Linux platformi.

1.1. Zadatak

Zadatak diplomskog rada je napraviti analizu i optimizaciju algoritma za manipulaciju; matematičko određivanje maksimalne dopuštene složenosti manipulacije kako bi se paket proslijedio u zadanom vremenskom intervalu; mjerenje i usporedba performansi uređaja opterećenog velikim brojem paketa bez manipulacija i s manipulacijama.

2. KOMUNIKACIJA U AUTOINDUSTRIJI

Pred autoindustriju se svakodnevno postavljaju novi izazovi u vidu tehnologije. Brojni moduli od kojih se sastoji automobil moraju međusobno komunicirati. Zbog brojnih zahtjeva, razvoj takve komunikacije je zahtijevao velike resurse kojima su raspolagala rijetka poduzeća (npr. Bosch). Visoke razine sigurnosti, pouzdanosti, robusnosti i otpornosti na pogreške su odlike takvih komunikacijskih sustava.

Predstavnik tih komunikacijskih sustava je CAN standard koji omogućuje komunikaciju uređaja poput upravljača podizača stakala, zaključavanja, klima uređaja, svjetala i dr. Osim CAN-a koji dolazi i u verziji CAN FD, korišteniji sustavi komunikacije u automobilu su MOST sabirnica (omogućuje prijenos multimedijskog sadržaja) i FlexRay sabirnica (povezivanje i razmjena podataka između različitih sabirnica, komunikacija s glavnim računalom). U nastavku poglavlja će biti objašnjeni CAN (poglavlje 2.1) i FlexRay (poglavlje 2.2) standardi.

2.1. CAN standard

CAN je robusna serijska komunikacijska tehnologija koja je namijenjena komunikaciji elektroničkih uređaja u automobilskoj industriji. Neizostavan je dio svakog novijeg automobila ali i teretnog vozila (npr. kamion, traktor, bager i dr.).

Već se osamdesetih godina prošlog stoljeća pojavio problem individualne komunikacije svakog elektronskog uređaja što je rezultiralo povećanjem mase, zauzetim volumenom vozila i složenijom komunikacijskom strukturom. Tomu problemu na kraj je stao Bosch 1983. godine razvojem CAN standarda. Ubrzo velikani poluvodičke industrije poput Intela i Philipsa razvijaju prve CAN upravljačke čipove što rezultiralo pojavom prvog automobila s CAN sabirnicom – BMW serije 8, 1988. godine [1].

2.1.1. CAN i standardi

CAN je standardiziran 1994. godine i opisan je u četiri ISO dokumenta čija je usporedba s ISO-OSI modelom prikazana u tablici 2.1 ISO 11898-1 opisuje CAN protokol koji sadrži kontrolu srednjeg pristupa (*eng. Medium Access Control* - MAC), kontrolu logičkih veza (*eng. Logical Link Control* - LLS) i fizički signalizaciju (*eng. Physical Layer Signalization* - PLS). ISO 11898-2 i ISO 11898-3 se razlikuju u naponskim razinama i brzinama prijenosa podataka. ISO 11898-3 definira brzine prijenosa podataka do 125 kbit/s što se koristi u komunikaciji „manje bitnih“ uređaja poput kontrolora klime, svjetala, prozora i sl. ISO 11898-3 definira brzine

prijenosa podataka do 1 Mb/s što taj standard čini pogodnim za komunikaciju uređaja osnovne funkcionalnosti automobila (motor, kočnice, upravljač i sl.). MDI (*eng. Medium Depended Interface*) nije standardiziran ali postoji preporuka (CiA DS-102) za korištenje konektora SUB – D9 (Slika 2.1). Komunikacija u CAN mreži temeljena je na događajima (poruka se pokušava poslati odmah kad je i nastala) što u slučaju veće opterećenosti mreže može predstavljati problem za poruke manjeg prioriteta (poruke neće biti poslane) što ispravlja ISO 11898 – 4 standard dodavanjem vremensko – determinističkog uvjeta slanja [2].

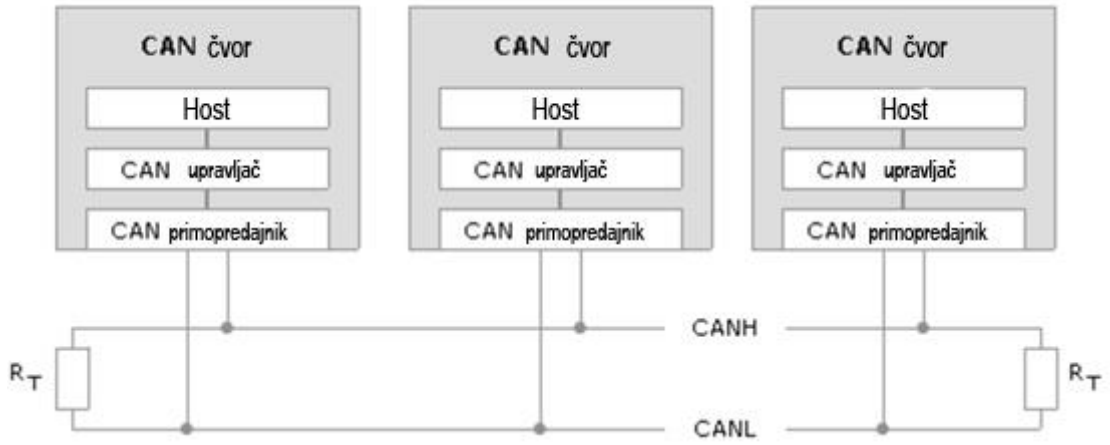
Tablica 2.1: Usporedba CAN standarda s ISO-OSI modelom [2].

<table>
<tr><th></th><th>ISO – OSI model</th><th>CAN model</th><th>CAN standard</th></tr>
<tr><td>7</td><td>Aplikacijski sloj</td><td rowspan="5">Nije definirano</td><td rowspan="5">-</td></tr>
<tr><td>6</td><td>Sloj prezentacije</td></tr>
<tr><td>5</td><td>Sloj sjednice</td></tr>
<tr><td>4</td><td>Transportni sloj</td></tr>
<tr><td>3</td><td>Mrežni sloj</td></tr>
<tr><td rowspan="2">2</td><td rowspan="2">Sloj podatkovne veze</td><td>LLC</td><td rowspan="3">ISO 11898 - 1</td></tr>
<tr><td>MAC</td></tr>
<tr><td rowspan="3">1</td><td rowspan="3">Fizički sloj</td><td>PLS</td></tr>
<tr><td>PMA</td><td>ISO 11898 – 2</td></tr>
<tr><td>PMS</td><td>ISO 11898 – 3</td></tr>
<tr><td></td><td></td><td>MDI</td><td>CiA DS – 102</td></tr>
</table>

Slika 2.1: Konektor SUB -D9 za spajanje uređaja na CAN sabirnicu.

2.1.2. Sklopovlje CAN-a

Najčešće se CAN mreža realizira linijskom topologijom po kojoj se svi uređaji u mreži spajaju paralelno na sabirnicu (Slika 2.2). Topologija zvijezde je moguća ali se rijetko koristi.

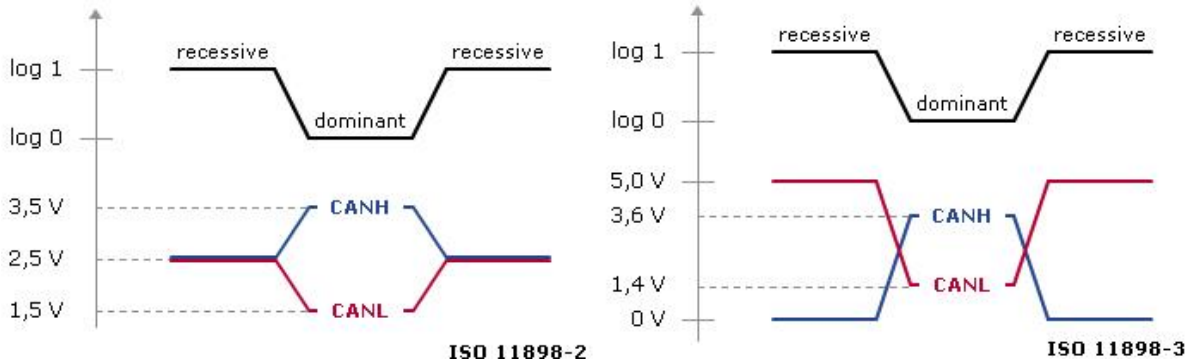


Slika 2.2: Shema spajanja CAN čvorova [2].

Sabirnica je sačinjena od nezaštićene uvrnute parice (*eng. Unshieled Twisted Pair* - UTP) površine poprečnog presjeka vodiča od 0.32 mm^2 do 0.6 mm^2 uz uvjet linijske otpornosti manje od 60 mΩ. Duljina sabirnog voda bi trebala biti ograničena na maksimalno 40 metara a na krajeve sabirnog voda se u svrhu smanjenja refleksije signala postavljaju omski otpornici R_T (tipično 120 Ω). ISO 11898 definira 32 uređaja kao maksimalni broj uređaja po jednoj CAN sabirnici [2].

CAN sabirnica se sastoji od 2 vodiča oznaka CANH (*eng. CAN high line*) i CANL (*eng. CAN low line*). Zbog velikih šumova koji se induciraju u automobilima, za detekciju logičkih stanja (0 ili 1) ne koriste se naponske razine u odnosu na minus pol automobila (potencijal šasije ili minus pol akumulatora) već razlike potencijala CANH i CANL vodiča (Slika 2.3). ISO 11898-2 definira logičku jedinicu (recesivni bit) kao razliku potencijala CANH i CANL vodiča od 0 V do 0.5V. Logička nula (dominantni bit) predstavljena je kao razlika potencijala veća od 0.9 V (tipično 2 V). Kod 125 kbit/s sabirnice (ISO 11898-3) naponske razine su nešto izmijenjene; 5 V potencijalne razlike za logičku jedinicu i 2 V za logičku nulu. U CAN standardu se često koriste nazivi recesivni (logička jedinica) i dominantni (logička nula) bit. Recesivni bit se može prepisati dominantnim bitom ali ne i obratno. Kada se na sabirnicu postavi recesivni bit CANH i CANL vodiči su na potencijalu 2,5 V. U tom trenutku neki drugi čvor može povećati napon CANH vodiča na 3,5 V a smanjiti CAHL na 1,5 V. Na sabirnici se tada

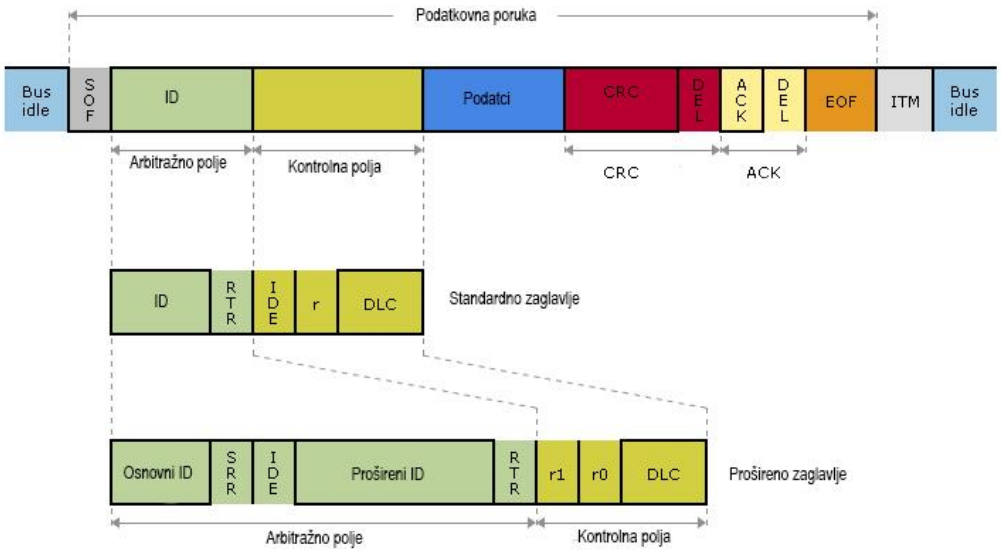
nalazi dominanti bit koji neće biti prepisan u slučaju da treći čvor mreže postavi 2,5 V na CANH i CANL vodiče (ostat će 3,5 V za CANH i 1,5 V za CANL). Primjerice; čvor *A* može na sabirnicu postaviti recesivni bit a čvor *B* u istom trenutku dominanti. Dominantni bit čvora *B* će prepisati recesivni bit i svi čvorovi u CAN mreži će vidjet dominantni bit čvora *B* – važno u trenutku odlučivanja koji će čvor slati poruku.



Slika 2.3: Naponske razine CAN sabirnica [2].

2.1.3. CAN protokolni stog

CAN poruke su standardizirane i podijeljene u 3 vrste; podatkovne poruke (*eng. Data frame*), kontrolne poruke (*eng. Remote frame*) i poruke pogreške (*eng. Error frame*). Kontrolne poruke zahtijevaju slanje određenog podatka (podatkovnom porukom) od drugih čvorova dok poruke pogreške obavještavaju čvorove u mreži o pogreškama komunikacije. Podatkovne poruke prenose korisne podatke, jedine su korištene u ovom radu pa će samo one i biti opisane.



Slika 2.4: Protokolni stog CAN poruke [2].

Svaka podatkovna poruka počinje s jednim dominantnim bitom (*eng. Start Of Frame* - SOF) koji označava početak poruke. Slijedi 11 bita polja identifikacije (*eng. Identifier* - ID) koje je ujedno i oznaka prioriteta. Svaki čvor u mreži osluškuje ID polje i ako na svom popisu poruka za primanje pronađe ID koji se trenutno šalje, čita ostatak poruke, u protivnom preskače trenutnu poruku. Za daljnju obradu bitna su jedno bitna polja IDE (*eng. Identifier Extension bit*) (označava vrstu zaglavlja; standardno ili prošireno) i RTR (*eng. Remote Transmission Request*) (podatkovna poruka ili kontrolna poruka). U slučaju proširenog zaglavlja identifikacijska oznaka naraste do 29 bita što omogućava preciznije usmjeravanje poruke. DLC (*eng. Data Length Code*) polje sa svoja 4 bita informira prijemnik o dužini podatka koji se prenosi CAN porukom nakon čega slijede konkretni podatci - njih maksimalno 64 bita. Konzistentnost podataka je osigurana cikličkom provjerom redundancije (*eng. Cyclic Redundancy Check* - CRC). Predajnik na sabirnicu zapisuje CRC kod izvorne poruke a prijemnik računa svoju verziju CRC-a primjene poruke. Ako se CRC kodovi predajnika i prijemnika ne podudaraju, došlo je do pogreške u slanju CAN poruke te prijemnik na sabirnicu zapisuje dominantni bit na mjesto ACK (*eng. Acknowledgement)* polja. Dominanti bit na mjestu ACK polja signalizira predajnom čvoru da je došlo do pogreške u komunikaciji. CAN poruka završava sa 7 recesivnih bitova kao oznake kraja poruke (*eng. End Of Frame* - EOF) [2].

U svrhu smanjenja elektromagnetskih smetnji CAN standard nalaže korištenje NRZ kodiranja (*eng. Non Return to Zero*). U slučaju dugog niza istih bita, predajnik i prijemnik mogu ispasti iz sinkronosti. Zbog toga CAN standard definira dodatno umetanje redundantnih bitova

(*eng. Bit stuffing*). Predajnik mora nakon svakih pet isti bitova umetnuti jedan dodatni bit suprotne logike kako bi se održala sinkronost predajne i prijemne strane [2].

2.1.4. Pristup sabirnici

Svaki CAN čvor kada dobije poruku za slanje, ima pravo spojiti se na sabirnicu i odaslati poruku (*eng. Event - driven*). Problem nastaje u slučaju kolizije kada više čvorova žele poslati svoje poruke u isto vrijeme. Prvi uvjet koji čvor mora zadovoljiti da bi postavljao svoje podatke na sabirnicu je prazna sabirnica. Tek kada je *Bus idle* (Slika 2.4) čvor može počet pisati podatke na sabirnicu. Ako više čvorova u isto vrijeme počne pisati po sabirnici, početni bit (SOF) će biti postavljen od svih čvorova jednako te se kolizija neće moći detektirati. Koliziju je moguće detektirati tek pri upisivanju identifikacijske oznake na sabirnicu. Kako identifikacijska oznaka ujedno predstavlja i oznaku prioriteta, od natjecanja za pisanje po sabirnici će odustati oni čvorovi koji šalju poruku s većim ID – om. Što je ID veći broj to on sadrži veći broj recesivnih bitova koji se mogu prepisati dominantnim bitovima. Na primjeru iz Tablica 2.2 čvor *A* će postaviti nulti bit ID – a na sabirnicu („1“). Drugi čvorovi će postaviti nulte bitove svojih ID-ova koji su dominantni bitovi („0“) te će prepisati recesivni bit čvora *A*. Čvor *A* istovremeno piše i čita vrijednosti na sabirnici pa će uočiti da njegov zapisani bit i bit na sabirnici nisu isti što čvor *A* izbacuje iz natjecanja za pisanje po sabirnici. Ista će se situacija dogoditi čvorovima *B* i *C* na drugom odnosno petom bitu – bitovi koji su oni zapisali neće biti jednaki bitu koji je na sabirnici i oni odustaju od pisanja po sabirnici po sljedećeg slobodnog trenutka. Čvor *D* šalje poruku s najviše dominantnih bitova (najvišim prioritetom) i zauzima sabirnicu za slanje svoje poruke. Nakon slanja poruke čvora *D*, sabirnica se oslobađa i prethodno opisani postupak se ponavlja. U sljedećim iteracijama slanja poslati će se poruke čvorova (redom): *C, B, A* [2].

Tablica 2.2: Primjer natjecanja za pisanje po sabirnici.

Čvor	**ID poruke za slanje**	**Status**
A	1 0 1 0 1 0 1 0 1 0 1	Odustaje od pisanja nakon nultog bita
B	0 0 1 0 1 1 1 1 0 0 1	Odustaje od pisanja nakon drugog bita
C	0 0 0 0 0 1 1 1 0 1 1	Odustaje od pisanja nakon petog bita
D	0 0 0 0 0 0 0 0 0 1 1	Kolizija nije detektirana
Sabirnica	0 0 0 0 0 0 0 0 0 1 1	Šalje se poruka čvora *D*

2.1.5. CAN FD

CAN FD je proširenje originalnog CAN standarda koje donosi veće brzine prijenosa podataka i veću robusnost. CAN FD je razvijen od strane istog tvorca kao i CAN (Bosch) 2011.godine.

Karakteristike CAN FD standarda:

- Povećanje korisnog dijela poruke (*eng. payload*) s 8 B na 64 B.
- Povećanje brzine prijenosa podataka s 1 Mb/s (CAN) na 2 Mb/s (standard CAN FD) pa čak do 5 Mb/s (*point to point CAN FD*).
- Upravljači CAN FD su unazad kompatibilni i podržavaju CAN standard (ali CAN upravljači ne podržavaju CAN FD).
- Smanjenje potrebnih sabirnica u vozilu.
- Rezervirani bit (r) iz CAN zaglavlja transformiran u FDF bit koji označava CAN FD poruku.
- Dodavanjem ESI polja u zaglavlje omogućeno označavanje čvorova *Error Active/Passive* – transparentniji način praćenja pogrešaka.
- Smanjenje problema opterećenja sabirnice [2].

2.2. FlexRay standard

FlexRay je komunikacijski standard namijenjen komunikaciji elektroničkih elemenata unutar modernog vozila. FlexRay u osnovi ima istu namjenu kao i CAN standard (komunikacija u autoindustriji) ali sam po sebi ima brojne prednosti.

U automobilskoj industriji sigurnost ima veliki utjecaj pri izradi svakog sklopa. Potreba za sigurnim elektroničkim sustavima je dodatno narasla u vozilima najnovije generacije gdje se već implementiraju pasivni ali i aktivni sustavi autonomne vožnje. Takvi sustavi obrađuju veliku količinu podataka u jako malom vremenu. Rezultat navedenog su povećani kriteriji u komunikaciji pojedinih sustava unutar vozila. Brzine prijenosa podataka koje pruža CAN standard postaju nedovoljne a i slabije upravljanje pogreškama u komunikaciji postaje izraženije. Dodatno, CAN se temelji na događajima – poruka se pokušava poslati onda kada je nastala. Taj koncept događaja nije dobar za uređaje koji moraju često komunicirati (npr. računalo za održavanje stabilnosti automobila može i do 1000 puta u jednoj sekundi očitati senzor broja okretaja svakog kotača posebno). Zbog navedenih problema pojavila se potreba za razvojem

deterministočkog komunikacijskog standarda. Kako razvoj modernog komunikacijskog standarda nije jednostavan i jeftin, BMW i DaimlerChrysler su se udružili u konzorcij 1999.godine. Godinu dana kasnije FlexRay konzorciju se pridružuje Motorola i Philips. Rezultat konzorcija je razvoj modernog deterministočkog komunikacijskog standarda s visokom otpornošću na pogreške – FlexRay. 2010. godine FlexRay konzorcij se raspušta jer FlexRay postaje međunarodnim standardom po oznakom ISO 17458 [2][3].

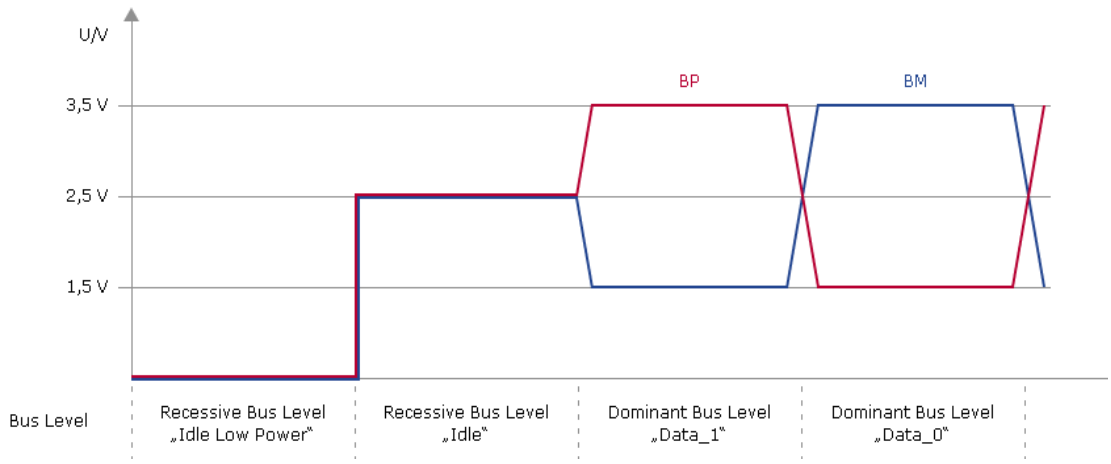
2.2.1. Topologija i sklopovlje FlexRaya

Povezivanje FlexRay čvorova je moguće obaviti na nekoliko različitih načina što povećava fleksibilnost dizajneru komunikacijskog sustava. Najčešće topologije povezivanja su:

- točka – točka (*eng. Point to point*),
- linijska topologija (*eng. Line topology*),
- pasivna zvijezda (*eng. Passive star topology*),
- aktivna zvijezda (*eng. Active star topology*),
- Hibridna topologija.

Topologije točka – točka i linijska topologija su otprije poznate. Kod topologije pasivne zvijezde svi se čvorovi spajaju u jednu fizički točku – logika komunikacije je slična linijskoj topologiji. Aktivna zvijezda posjeduje aktivni centralni čvor – usmjeritelj. Hibridna topologija je kombinacija prethodno nabrojanih topologija [3].

FlexRay standard definira maksimalnu duljinu vodova (sabirnice) između čvorova od 24 metra – za sve vrte topologija. Kao i CAN, FlexRay koristi UTP kabel s dva bakrena vodiča: BP (*eng. Bus Plus*) i BM (*eng. Bus Minus*). Unutar svakog FlexRay čvora, između BP i BM vodiča nalazi se završni otpornik (*eng. termination resistor*) iznosa 80 do 110 ohma. Kao i kod CAN-a, prijenos struje bitova izvršava se pomoću razlike napona između BP i BM vodiča. Kada nema podatka za slanje oba vodiča (BP i BM) mogu imati vrijednosti 0 V (ušteda energije) ili 2,5 V. U tom slučaju je razlika napona BP i BM 0 V što označava praznu sabirnicu. U slučaju prijenosa struje bita dva su moguća stanja; stanje „*Data_1*“ (logička jedinica) i stanje „*Data_0*“ (logička nula). Napon razlike vodiča je 2 V a ovisno o stanju mijenja se predznak napona (polaritet) (Slika 2.5).

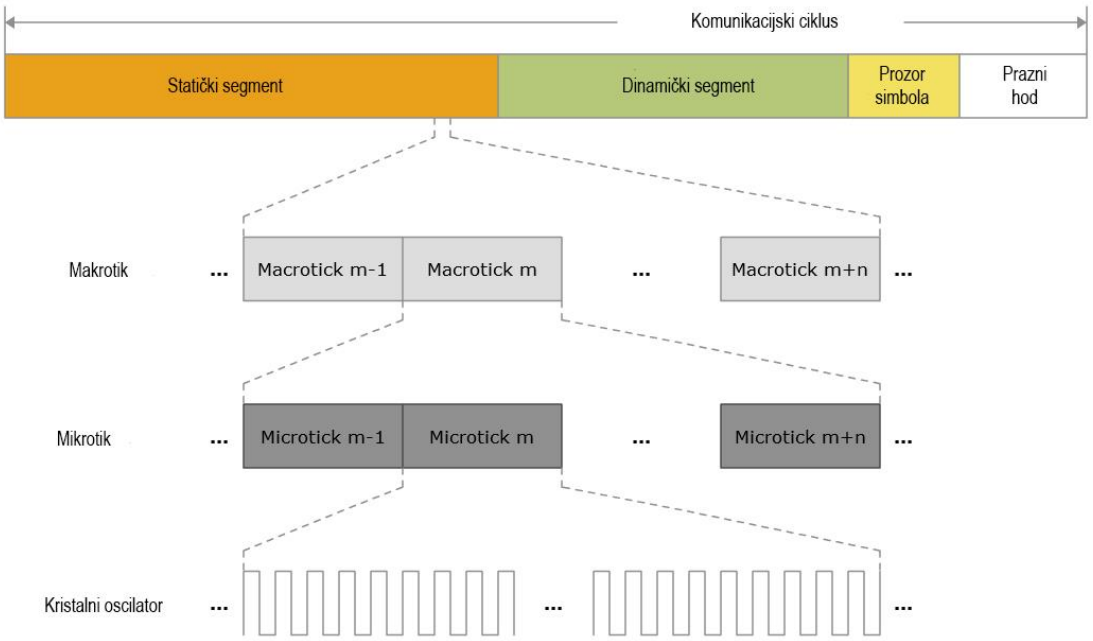


Slika 2.5: Naponske razine FlexRay standarda [2].

Spomenuta 2 vodiča nisu jedini gradivni objekt FlexRay sabirnica. Sabirnice mogu biti jednokanalne i dvokanalne. U slučaju dvokanalnih sabirnica, svaki kanal (*eng. Channel A and Channel B*) posjeduje svoje vlastite sabirne vodove s kojima ostvaruje brzine prijenosa podataka do 10 Mb/s za jedan kanal. Ako su potrebne veće brzine, čvorovi mogu kombinirati oba kanala i ostvariti brzinu prijenosa podataka do 20 Mb/s. U slučaju pogreške u komunikaciji na jednom kanalu, pogreška se ne prenosi na drugi što rezultira povećanjem pouzdanosti FlexRay standarda [2].

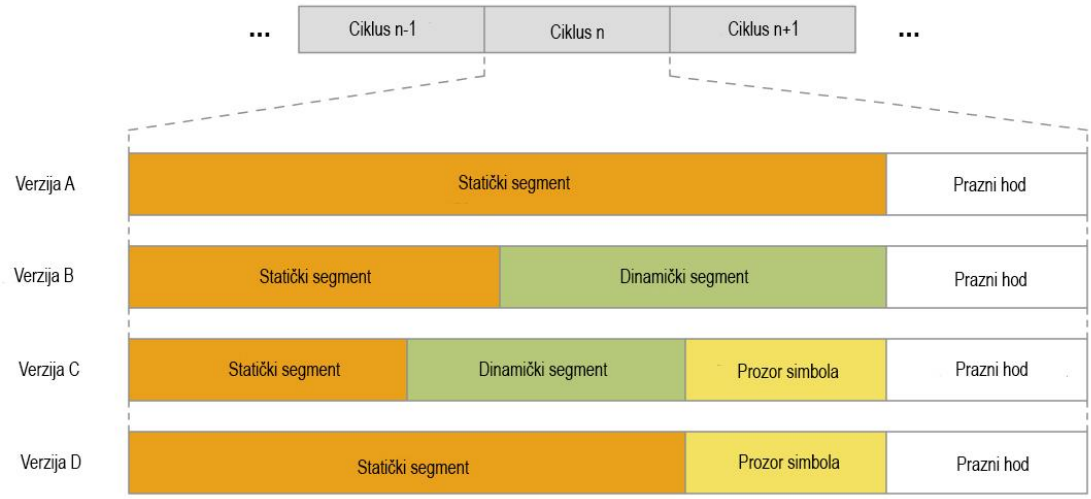
2.2.2. Pristup mreži

FlexRay je deterministički komunikacijski standard što znači da se poruke šalju u vremenski točno određenim intervalima (*eng. Time Division Multiple Acces* - TDMA). Najmanja vremenska jedinica je jedan takt kristalnog oscilatora. Prva veća vremenska jedinica sastavljena od niza taktova oscilatora je *mikrotik*, a slijedi ju *makrotik* i na kraju kao najveća vremenska jedinica jedan ciklus (Slika 2.6).



Slika 2.6: Hijerarhija vremena [2].

Svaki ciklus traje između jedne i pet milisekundi – ovisno kako dizajner mreže odluči. Postoje različite vrste ciklusa koje se razlikuju po vrsti poruka koje prenose (Slika 2.7).



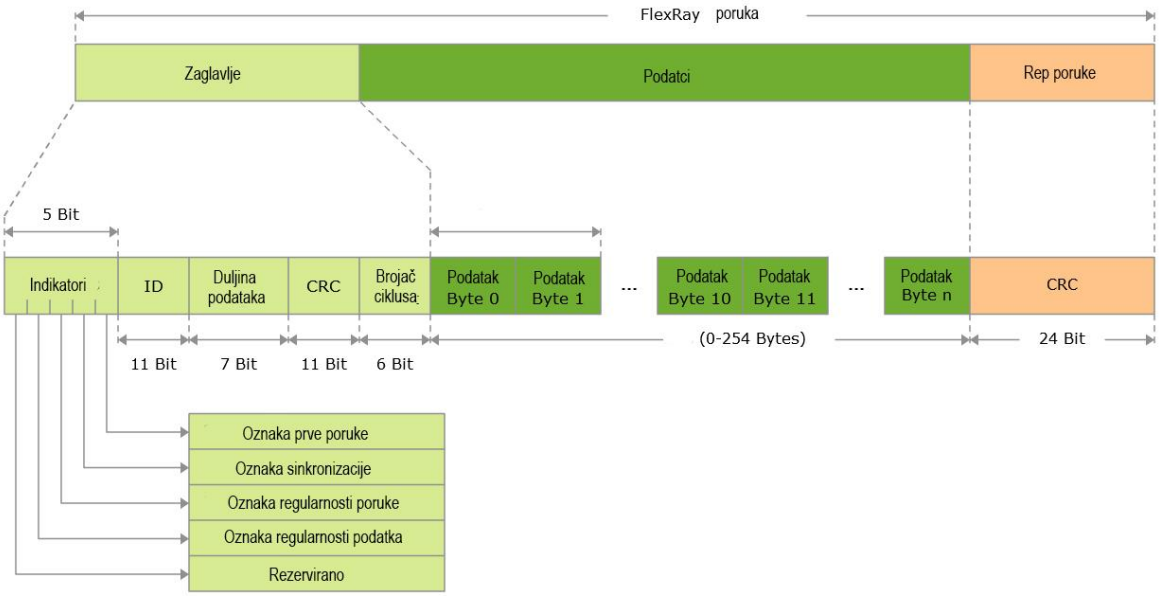
Slika 2.7: Vrste ciklusa [2].

Najznačajniji dio svakog ciklusa je statički segment unutar kojeg se šalju statičke poruke unaprijed definiranog trajanja i rasporeda kojim čvorovi šalju. Najčešće se čvorovi redaju slijedno npr. čvor *A* prvi šalje poruku, zatim čvor *B* i t.d. Unutar dinamičkog segmenta poruke nemaju predefiniran trajanje i duljinu pa taj segment služi za slanje poruka događaja (*eng. Event Driven Message*). Dinamičke poruke se postavljaju na sabirnicu onim redom kako su nastajale s

dodatnim uvjetom trajanja – poruke čijim bi se slanjem prešlo vrijeme dano dinamičkom segmentu se odbacuju. Prozor simbola služi za razmjenu podataka potrebnih za održavanje komunikacije a vrijeme praznog hoda je vrijeme za sinkronizaciju čvorova. Da bi se poštivao raspored korištenja sabirnice, svaki čvor posjeduje svoga „čuvara“ (*eng. Bus Guardian* - BG) koji čvoru omogućava slanje podataka na sabirnicu samo u trenutku koji je tome čvoru unaprijed propisan [2], [3].

2.2.3. Protokolni stog i sinkronizacija

Sve poruke u FlexRay standardu imaju isti protokolni stog (Slika 2.8). Prvih 5 bita poruke su zastavice (oznaka prve poruke u nizu, oznake regularnosti poruke i podataka i dr.). Zaglavlje se nastavlja sa 11 bita identifikacijske oznake poruke, oznake količine korisničkih podataka, CRC koda zaglavlja i brojača ciklusa. Polje korisničkih podataka može biti duljine između 0 i 256 bajta a slijedi ga CRC kod tih istih podataka [2].



Slika 2.8: Protokolni stog FlexRay poruke [2].

FlexRay predstavlja sinkroni komunikacijski protokol što znači da svi aktivni elementi komunikacije (čvorovi) moraju biti međusobno sinkronizirani. Tek će u sinkroniziranoj mrežni svi čvorovi u isto vrijeme početi čitati vremenske cikluse i pisati u njima zadanim vremenskim intervalima. Zbog starenja kristalnog oscilatora, s vremenom mu se mijenja frekvencija. Zbog te stohastičnosti vlastite frekvencije oscilatora u svakom čvoru, čvorovi se sinkroniziraju u radu.

Svaka mreža posjeduje jedan globalni sat s kojim se sinkroniziraju svi drugi čvorovi koji posjeduju svoje lokalne satove. Na početku komunikacije, u slučaju hladnog starta (*eng. Cold - start*) prva dva ciklusa će imati postavljena polja oznake prve poruke i oznake sinkronizacije. Na temelju tih polja obavit će se početna sinkronizacija frekvencije. Na kraju ciklusa čvorovi mjere trajanje praznog hoda. Ako je to trajanje različito od definiranoga, došlo je do pomaka faze. Čvorovi će potom obaviti sinkronizaciju faze pomicanjem za dobivenu razliku vremena [2], [3].

3. O POVEZNIKU

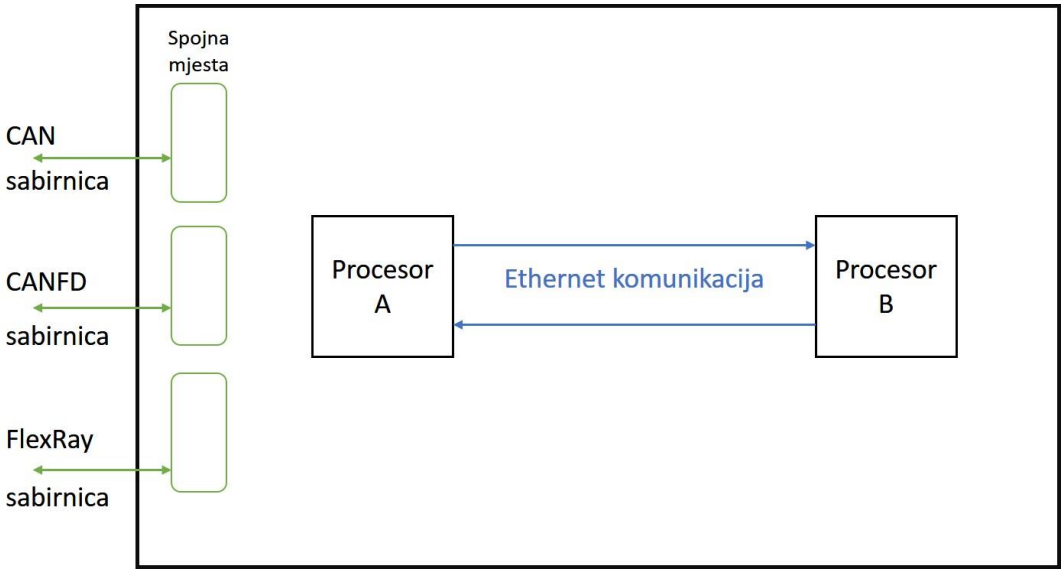
Poveznik slično kao i usmjeritelj (*eng. router*) povezuje dvije ili više mreža i usmjerava podatke između isti. Ono što razlikuje poveznika od usmjeritelja je viši nivo apstrakcije i složeniji algoritmi jer poveznik ima mogućnost rada s različitim protokolima i standardima dok usmjeritelju sve mreže moraju imati isti protokol (standard komunikacije).

Poveznici su računalni programi koji se mogu izvršavati paralelno s drugim programima na istom računalu ili se mogu samostalno izvršavati (iznad operativnog sustava) na posebnom sklopovlju (*eng. hardware*).

U nastavku će biti opisan korišteni poveznik i problemi koji su se morali riješiti prije nego bi program (poveznik) proradio. Rješavanje problema nije bilo u opisu rada ali je oduzelo puno vremena pa uključeno u ovaj rad.

3.1. Osnovno o povezniku

GWH (*eng. GateWay Host*) poveznik je vlasništvo instituta RT-RK Osijek d.o.o. Radi jednostavnosti, sloj primopredajnika je logički i fizički odvojen na drugom uređaju (Slika 3.1).



Slika 3.1: Blok prikaz ploče na kojoj se izvodi poveznik.

Takav pristup problemu odvaja problematiku fizičkog sloja od problema usmjeravanja podataka (*eng. routing*). Procesor *A* je posredstvom nekoliko primopredajnika spojen na različite sabirnice. Sve podatke koje primi sa sabirnica prosljeđuje u jednostavnijem obliku procesoru *B* koji ih usmjerava i vrača procesoru *A* s drugim zaglavljem. Procesori međusobno komuniciraju

Ethernet protokolom što znači da se pojednostavljene CAN i FlexRay poruke ovijaju (*eng. encapsulation*) unutar UDP (*eng. User Datagram Protocol*) i Ethernet protokola. Povezniku koji se izvršava na procesoru *B* nisu potrebni podatci o sinkronizaciji podatka, ciklusima i slično zbog čega koristi pojednostavljeni CAN i FlexRay oblik poruke (Tablica 3.1). Na Izvorni kod 3.1 i Izvorni kod 3.2 prikazan je konkretni primjer oblika CAN i FlexRay poruke koji se mora koristi u radu s poveznikom. Potrebno je uočiti razliku u definiranu zaglavlja (CAN za svako polje zaglavlja koristi jedan bajt a FlexRay jedan bit) i korištenje *big endian* metrike kod FlexRaya.

Tablica 3.1: Izgled poruka (lijevo CAN, desno FlexRay) za komunikaciju s poveznikom.

Broj bajta	**CAN poruka**
0	Oznaka CAN poruke (uvijek 0)
1	Identifikacijska oznaka
2	
3	
4	
5	Duljina podataka
6	Identifikacija čvora
7	Bit Rate preklopka
8	CAN/CAN FD (0 - CAN, 1 - CAN FD)
9	Oznaka zaglavlja
10	Korisnički podatci
11	
12	
13	
14	
15	
16	
17	

Broj bita	**FlexRay poruka**
0-7	Oznaka FlexRay poruke (uvijek 1)
8-18	Identifikacijska oznaka
19-25	Duljina podataka
26-31	Broj ciklusa
32	Kanal (0 – A, 1 - B)
33	PPI
34	NFI
35	SFI
36	FlexRay ID čvora
37 - 2084	Korisnički podatci

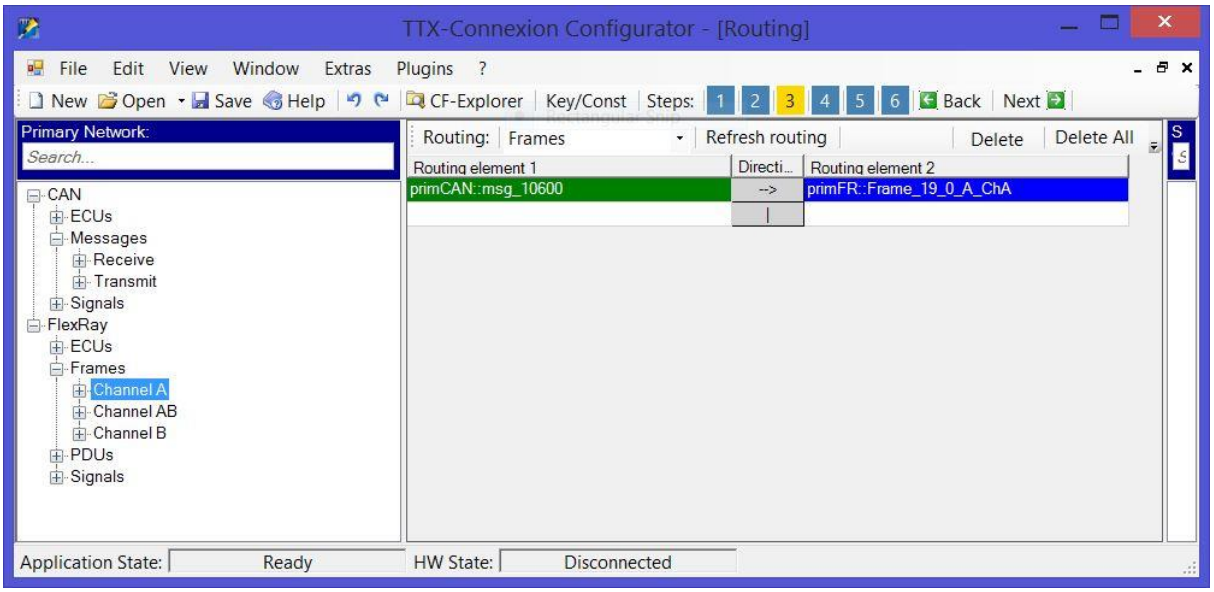
```
uint8_t msg_1027[] = {
        0x00,//Start bits - always 0 for CAN, 1 byte
        0x40,0x00,0x00,0x00,//ID array, 4 byte, bytes must be turned!
        0x08,//Data Length Code, 1 byte, max value 0x08
        0x00,//Node ID, 1byte, max value 0x04
        0x00,//Bit Rate Switch, 1 byte, 0 or 1
        0x00,//Message Type, 1 byte, 0 CAN, 1 CANFD
        0x00,//Frame ID type, 1 byte, 0 STD, 1 EXT
        0xAA,0xBB,0x00,0x00,0x00,0x40,0xBB,0xAA//Payload, 8 byte
};
```

Izvorni kod 3.1: Zapis jedne CAN poruke u programskom jeziku C.

```
uint8_t Frame_11_0_A_ChA[] = {
        0x01,       //Start bits - always 1 for FLX, 1 byte
        0b00001011, //first 8b of Frame ID. WARNING!!! low bits
        0b01001000, //first 5b of Payload Length, the rest 3b of
                    FrameID (most significant bits)
        0b00000000, //Cycle Code, the rest 2b (most significant) of
                    PayloadLength
        0b00000000, //3b pagging, FLXnode ID, SFI, NFI, PPI, Channel
        0x00,0x50,0x00,0x50,0x00,0x50,0x00,0x50,//first 8B of
                    payload
        0x00,0x50,0x00,0x50,0x00,0x50,0x00,0x50,
        0xaa,0xaa  //the rest of payload (2B)
};
```

Izvorni kod 3.2: Zapis jedne FlexRay poruke u programskom jeziku C.

Povezniku se prije početka usmjeravanja poruka moraju unijeti postavke. U postavkama su pravila usmjeravanja poruka koje povezniku zadaju upute koju ulaznu poruka će usmjeriti na koju izlaznu poruku. S tim informacija poveznik može npr. od ulazne CAN poruke načiniti izlaznu FlexRay poruku s drugačijim zaglavljem, ali istim korisničkim podatcima. Te postavke se ne definiraju ručno već pomoću programa TTX-Connexion Configurator (Slika 3.2). Navedeni je program vlasništvo TTTech – firme vodilje u razvoju komunikacijskih sustava u automobilskoj industriji.



Slika 3.2: Izgled TTX-Connexion Configurator programa za izradu i unos postavki u poveznik.

Poruke i postavke se šalju ovijene UDP - om na mrežnu adresu (*eng. Internet Protokol* - IP) uređaja s brojem vrata (*eng. port*) 40001. Poveznik osluškuje navedena vrata i sve primljene podatke pokušava raščlaniti (*eng. parse*) te u slučaju uspješnog raščlanjivanja, nad podacima obavlja definirane mu radnje.

Program je pisan za rad nad Linux platformom i prilagođen je izvršavanju na slabijem sklopovlju visoke razine pouzdanosti. Zbog primarne namjene izvršavanja na slabijem sklopovlju, posjeduje neke specifičnosti. Broj paralelnih niti (*eng. thread*) sveden je na broj 6 od čega su 4 pasivne (trajanje izvršavanja im je kraće i rade manje bitan posao) u usporedbi s ostale dvije. Glavnina programa se izvršava u jednoj niti dok druga nit prima i šalje podatke koje prva obrađuje. Razlog malog broja niti je sklopovlje kojem je namijenjeno – sporno sklopovlje sadrži mali broj fizički jezgri, ponekad samo jednu. Program se sastoji od 6 logičkih cjelina i ukupno 37 datoteka izvornog koda napisanog u *C* programskom jeziku. Ukupno 12 973 linije programskog koda unutar koji se pridržavalo MISRA C pravila programiranja.

3.2. Ispravljanje pogrešaka

U postojećem programu se naišlo na nekoliko problema koji su kočili rad programa. Problemi su otklonjenu prije mjerenja performansi.

Jedan od problema je bio prijenos adresa preko varijabli (Izvorni kod 3.3) zbog kojeg se program nije pravilno izvršavao. Problem je riješen prebacivanjem programa u 32 bitnu verziju.

```
uint32_t* pokazivacA;
uint32_t* pokazivacB;
uint32_t varijabla;
uint32_t varijablaAdrese;

pokazivacA = &varijabla;
varijablaAdrese = (uint32_t)pokazivacA;

/*Na drugom mjestu u kodu*/
pokazivacB = (uint32_t*)varijablaAdrese;
```

Izvorni kod 3.3: Prijenos adrese varijable pomoću cjelobrojne varijable – demonstracijski prikaz.

Još neki riješeni problemi:

- Poveznik nije usmjeravao CAN poruke ako bi se u postavkama usmjeravanja našla više od 4 CAN čvora. Problem je riješen uklanjanjem dijelova koda.
- Program je iz pogrešnog izvora vukao podatke o postavkama.
- Poveznik je imao problema s usmjeravanjem 1:n (npr. primjena jedna CAN poruka se usmjerava na dvije druge CAN poruke). Problem je bio u logici korištenja signalizacijskih zastavica koje označavaju imali podataka za obradu ili ne. Uveden je dodatni brojač koji signalizira koliko poruka se još mora obraditi.
- Povezniku nije radilo usmjeravanje na FlexRay poruke. Problem je riješen jednostavnom prepravkom koda.
- Riješen je problem pogrešnog označavanja kanala kod FlexRay poruka.
- Sve FlexRay poruke s ID - om većim od 64 program nije obrađivao. Napravljena je posebna funkcija koja u postavkama programa traži najveći ID i pridružuje ga potrebnoj varijabli.

3.3. Mjerenje performansi

Nakon ispravka problema koji su kočili normalni rad poveznika, izmjerene su performanse istoga. Da bi mjerenja bila objektivna, poveznik je testiran kao crna kutija – predamo mu jednu poruku i mjerimo vrijeme dok ne dobijemo odgovor. U tu svrhu napisan je novi C program koji šalje poruke povezniku i mjeri kružno vrijeme kašnjenja (*eng. Round Trip Time* - RTT) dok ne primi odgovor. Novi je program nazvan Client.

Client se sastoji od 3 niti; glavna upravljačka nit, nit za slanje podataka i nit za primanje odgovora i vođenje statistike. Komunikacija između Clienta i poveznika se odvija UDP protokolom. Nakon što Client UDP – om pošalje jednu poruku, zapiše vrijeme sustava. Kada primi odgovor, ponovno zapiše vrijeme sustava. Razlika 2 vremena je kružno kašnjenje. Proces se ponavlja 5000 puta nakon čega program daje statistiku svih mjerenja. Ta statistika uključuje minimalno vrijeme, maksimalno vrijeme i prosjek svih kružnih kašnjenja (Slika 3.3).

```
0x 1 2C 48 10 0 0 50 0 50 0 50 0 50 0 50 0 50 0 50 AA AA
Primljen odgovor nakon 0s, 7us:
0x 1 3C 48 C 0 0 50 0 50 0 50 0 50 0 50 0 50 0 50 AA AA
Primljen odgovor nakon 0s, 8us:
0x 0 91 3 0 0 8 1 0 0 0 AA BB 0 0 0 40 BB AA 1 54 48 8 0 AA BB 0 0 0 40 BB AA 0 0 0 0 0 0 0 0 0 0
Primljen odgovor nakon 0s, 3us:
0x 0 17 5 0 0 8 1 0 0 0 AA AA 0 60 0 60 AA AA
Primljen odgovor nakon 0s, 3us:
0x 0 84 0 0 0 7 1 0 0 0 AA AA 0 60 0 AA AA
Primljen odgovor nakon 0s, 4us:
0x 0 47 6 0 0 8 1 0 0 0 AA AA 0 80 0 60 AA AA
Primljen odgovor nakon 0s, 4us:
0x 1 3D 48 C 0 AB AB 0 60 0 10 AB AB 0 0 0 0 0 0 0 0 0 0
a

Statistika:
Broj poslanog: 5005
Broj Primjenog: 5005
min: 0s, 1923ns
max: 0s, 2006us
avr: 0s, 11us
```

Slika 3.3: Prikaz konzolnog ispisa programa za mjerenje statistike.

Vrijeme između slanja dvije poruke je bilo konstanto za jedan skup mjerenja (5000 mjerenja). Skupovi mjerenja su se razlikovali osim po drugim porukama i po vremenima između slanja pojedinih poruka (100 µs do 5 ms). Sve poruke kojima je testiran poveznik su zapisane u kodu Clienta.

Sva mjerenja su rađena na stolnom računalu. Stolno računalo raspolaže s jačim procesorom (Intel Core i7- 4790 @ 3,6 GHz) pa su i rezultati bolji nego što bi bili da se pretvornik izvodio na ploči za koju je pisan. Mjerenja nakon optimizacije su rađena na istom računalu što omogućava usporedbu performansi prije i poslije optimizacije. Računalo za mjerenja opremljeno je Linux Ubuntu 14,04 LTS operativnim sustavom.

Za potrebe mjerenja u programu TTX-Connexion Configurator napravljen je scenarij usmjeravanja poruka koji obuhvaća većinu mogući usmjeravanja: CAN – CAN, CAN – FlexRay, FlexRay – FlexRay, FlexRay – CAN, CAN – FlexRay i CAN (Tablica 3.2). Poruke čije ime počinje s „*Frame*“ označavaju FlexRay poruke i „*msg*“ je oznaka CAN poruka. FlexRay

poruke su sve veličine 184 bita od čega je 40 bita zaglavlje. CAN poruke su nešto manje ali imaju veće zaglavlje – 80 bita.

Tablica 3.2: Popis i smjer usmjeravanja poruka scenarija za mjerenje.

r.b.	Poruka za slanje	Duljina poruke za slanje [bit]		Poruka odgovora	Duljina poruke odgovora [bit]
1.	Frame_11_0_A_ChA	184	>	Frame_3_0_A_ChA	184
2.	Freme_16_0_A_ChA	184	>	Frame_4_0_A_ChA	184
3.	Frame_18_0_B_ChB	184	>	msg_7563	144
4.	Frame_31_0_B_ChB	184	>	msg_7888	144
5.	Frame_63_0_A_ChA	184	>	Frame_44_0_A_ChA	184
6.	Frame_72_0_A_ChA	184	>	Frame_60_0_A_ChA	184
7.	msg_12679	144	>	msg_12388	144
8.	msg_10600	144	>	Frame_61_0_A_ChA	184
9.	msg_6518	144	>	msg_8699	144
10.	msg_5961	136	>	msg_3692	144
11.	msg_1027	144	>	Frame_84_0_A_ChA	184
12.				msg_10489	144

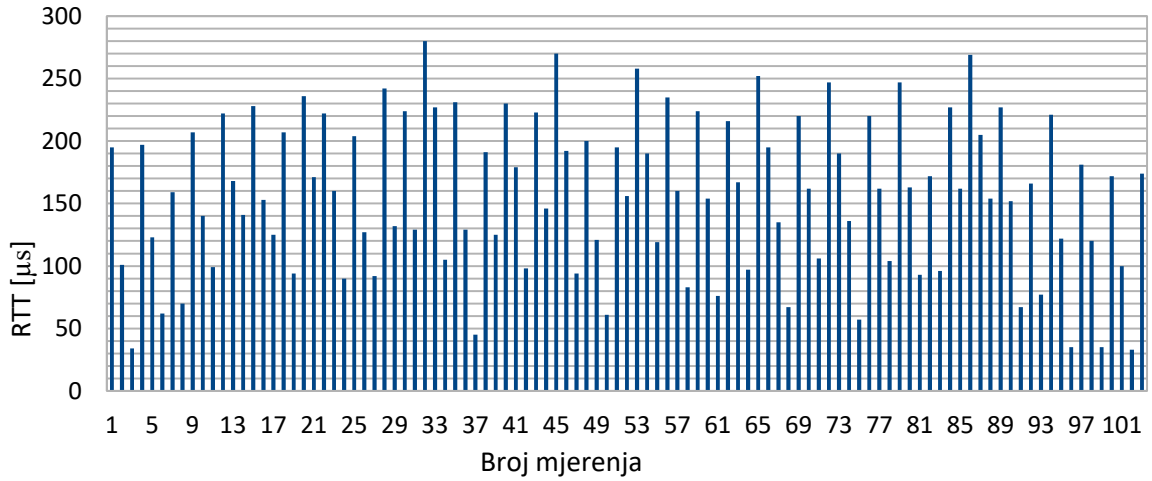
Svaka poruka je poslana 5000 puta i svaki RTT je ušao u statistiku. Statistika prikazuje minimalno vrijeme nakon kojeg je poveznik poslao odgovor, maksimalno vrijeme i najbitnije – prosječno vrijeme. Postotak odgovorenosti prikazuje omjer neodgovorenih poruka i poslanih poruka ka povezniku. Nakon što je obavljeno mjerenje 11 poruka posebno, izvršeno je mjerenje RTT-a svih poruka. U redu sažetka prikazano je najmanje vrijeme svih minimalnih vremena, najveće vrijeme svih maksimalnih vremena i prosjeci postotka odgovorenosti, broja odaslanih poruka i prosječnih RTT-ova. Brzina slanja podataka se računa kao omjer duljine poruke i vremenskog razmaka između slanja poruka a brzina obrade podataka koji ima poveznik je omjer duljine poruke i RTT – a (koristit će se srednja vrijednost).

Tablica 3.3: Skup mjerenja s vremenskim razmakom slanja poruka 5 ms.

	Kružno vrijeme kašnjenja – RTT [µs]				
Ime poruke	**Minimalno**	**Maksimalno**	**Prosječno**	**Broj mjerenja**	**Ispustito**
Frame_11_0_A_ChA	15	918	155	5000	0%
Frame_16_0_A_ChA	14	1129	152	5000	0%
Frame_18_0_B_ChB	13	352	153	5000	0%
Frame_31_0_B_ChB	13	504	151	5000	0%
Frame_63_0_A_ChA	14	912	152	5000	0%
Frame_72_0_A_ChA	13	1504	150	5000	0%
msg_1027	12	2548	151	5000	0%
msg_12679	12	3875	153	5000	0%
msg_5961	13	1587	150	5000	0%
msg_6518	15	488	152	5000	0%
msg_10600	13	1563	152	5000	0%
Sve poruke	12	880	149	5000	0%
Sažetak	**12**	**3875**	**152**	**5000**	**0%**
Sve poruke	12	1465	151	50539	0%

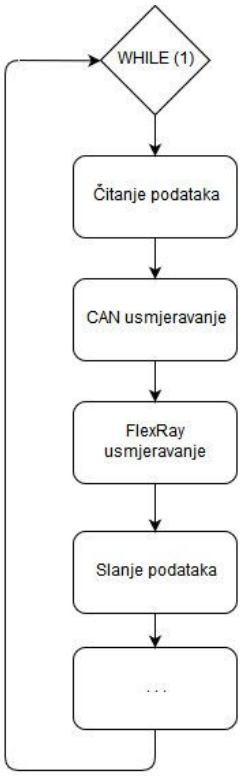
Prvi skup mjerenja performansi (Tablica 3.3) obavljen je s vremenskim razmakom slanja poruka od 5 ms. Toliki vremenski razmak neće dovesti do preopterećenja poveznika i odbacivanja podataka a dobar je za grafički prikaz rezultata. Također, kao što je vidljivo iz tablice, poveznik ima jednaku brzinu obrade svih poruka. Brzina slanja podataka je bila 36,8 Kb/s a uz prosječni RTT, brzina obrade 1,21 Mb/s.

Na Slika 3.4 prikazano je prvih 100 mjerenja RTT-a poruke msg_10600. Iz grafa je vidljiva stohastičnost RTT – ova. Nema pravila kojim se može predvidjeti koliki će biti pojedini RTT. Razlog takve stohastičnosti je mjesto u kodu na kojem se nalazi poveznik u trenutku dolaska poruke.

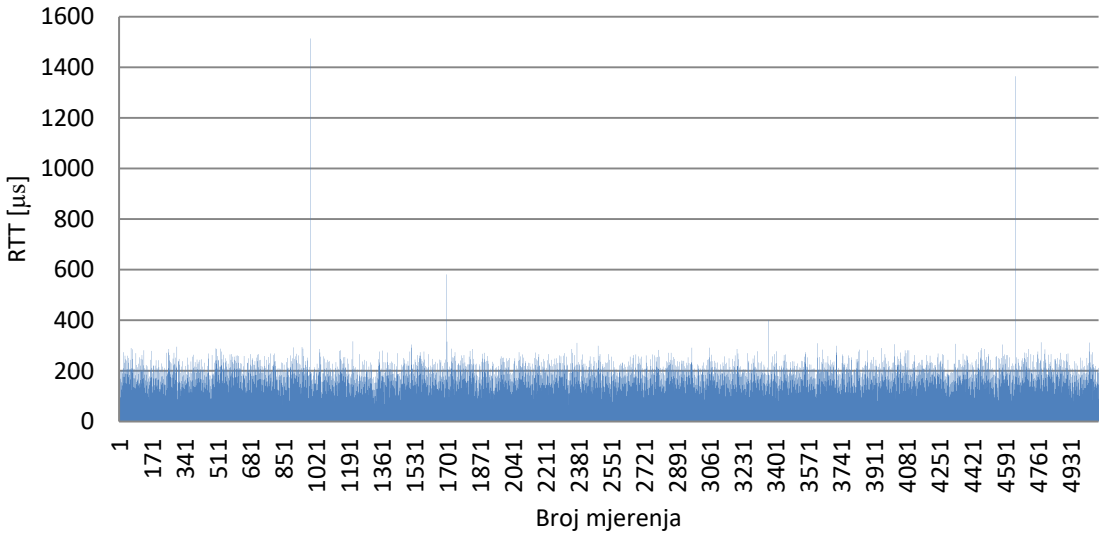


Slika 3.4: Grafički prikaz prvih 100 mjerenja za poruku msg_10600.

Poveznik u svojoj glavnoj niti koja radi usmjeravanje podataka ima blok shemu kao na Slika 3.5. Glavnina koda se izvršava u beskonačnoj *while* petlji. U svakoj iteraciji petlje prvo se pročitaju pristigli podatci, zatim se pristupa usmjeravanju poruka i potom se novonastale poruke šalju nazad pošiljatelju izvornih poruka. Ako je poruka pristigla neposredno prije izvršavanja bloka za čitanje poruka, odmah se obrađuje što će rezultirati malim RTT – om. Druga krajnost je kada poruka pristigne neposredno nakon izvršavanja bloka za čitanje podataka. U tom slučaju će poruka čekati u memorijskom među spremniku sljedeću iteraciju petlje kada će biti pročitana, usmjerena i odaslana. Posljedica toga će biti veliki RTT. Navedeni princip rada rezultira stohastičkim vremenima kružnog kašnjenja zbog čega se u razmatranje neće uzimati ekstremi (minimumi i maksimumi) već samo prosječne vrijednosti.



Slika 3.5: Blog dijagram glavne niti poveznika.



Slika 3.6: Kružno vrijeme kašnjenja s prikazom ekstrema maksimalne vrijednosti.

Neka mjerenja imaju jako veliku maksimalnu vrijednost (Slika 3.1). Na 5000 mjerenja pojavi se „nekoliko“ ekstrema koji kvare statistiku. Kako se ti ekstremi pojavljuju u nepravilnim razmacima (kao na Slika 3.1 tek na cca tisućitom mjerenju) nije bilo moguće pratiti što se

događa u kodu. Pretpostavka je da zbog složenosti koda, program uđe u neke petlje i provjerava uvjete koje inače ne provjerava.

Dodatno je u prvom skupu mjerenja obavljeno mjerenje svih poruka s 50539 odaslanih poruka u svrhu zaključivanja o promjeni statistike ovisno o broju poslanih poruka. Povećanjem broja mjerenja nije uočena promjena statističkih vrijednosti. Na drugu stranu, u slučaju manjeg broja mjerenja (do 500), događaju se velika odstupanja statističkih podataka (konkretno srednje vrijednosti) prilikom ponavljanja istih mjerenja. Zbog navedenog, brojka od 5000 mjerenja je empirijski odabrana te sva se daljnja statistika bazira na tih 5000 mjerenja.

Tablica 3.4: Skup mjerenja s vremenskim razmakom slanja poruka 500 µs.

	Kružno vrijeme kašnjenja – RTT [µs]				
Ime poruke	**Minimalno**	**Maksimalno**	**Prosječno**	**Broj mjerenja**	**Ispustito**
Frame_11_0_A_ChA	1	570	133	5000	0,18%
Frame_16_0_A_ChA	11	551	135	5000	0,04%
Frame_18_0_B_ChB	9	530	130	5000	0,02%
Frame_31_0_B_ChB	9	376	128	5000	0,00%
Frame_63_0_A_ChA	4	512	133	5000	0,14%
Frame_72_0_A_ChA	5	552	130	5000	0,16%
msg_1027	9	562	135	5000	0,04%
msg_12679	1	1676	133	5000	0,10%
msg_5961	10	510	136	5000	0,16%
msg_6518	10	574	132	5000	0,16%
msg_10600	7	523	134	5000	0,18%
Sve poruke	9	1070	135	5005	0,04%
Sažetak	**1**	**1676**	**132,63**	**-**	**0,1%**

Smanjenjem vremena između slanja poruka 10 puta (brzina slanja podataka 368 Kb/s) poveznik je počeo odbacivati neke pakete. Odbacivanje paketa je još uvijek na minimalnoj razini (0,1%) ali uzrokuje smanjenje RTT – ova. Poveznik je odbacivao pakete koji bi imali jako veliki RTT i zato drugi skup mjerenja (Tablica 3.4) ima značajno manje maksimalne RTT -ove od prvog skupa mjerenja. Posljedica manjih maksimalnih vrijednosti je i manja srednja vrijednost.

Povezniku se, ako nema podataka na ulazu, *while* petlja vrti u „praznom hodu“ (Slika 3.5). Zbog takvog režima rada operativni sustav predaje procesorske resurse drugim nitima i procesima koji se izvršavaju na računalu. Ako podatci brže dolaze, manji je broj iteracija *while* petlje u praznom hodu – operativni sustav povećava prioritet povezniku. S većim prioritetom su veći procesorski resursi i poveznik radi brže. Rezultat toga je smanjenje svih RTT – ova i povećanje brzine obrade na 1,39 Mb/s.

Tablica 3.5: Skup mjerenja s vremenskim razmakom slanja poruka 150 µs.

	Kružno vrijeme kašnjenja – RTT [µs]				
Ime poruke	**Minimalno**	**Maksimalno**	**Prosječno**	**Broj mjerenja**	**Ispustito**
Frame_11_0_A_ChA	0,021	1612	103	5000	0,72%
Frame_16_0_A_ChA	0,059	1327	107	5000	0,38%
Frame_18_0_B_ChB	0,006	3424	105	5000	0,68%
Frame_31_0_B_ChB	0,029	234	104	5000	0,70%
Frame_63_0_A_ChA	0,069	704	106	5000	0,96%
Frame_72_0_A_ChA	0,005	237	105	5000	0,82%
msg_1027	0,05	1016	104	5000	0,86%
msg_12679	0,122	3084	105	5000	0,66%
msg_5961	0,089	1191	105	5000	0,72%
msg_6518	0,018	1595	105	5000	1,52%
msg_10600	0,085	322	105	5000	0,96%
Sve poruke	0,181	3236	107	5005	0,82%
Sažetak	**0,005**	**3424**	**105**	**-**	**0,82%**

Kako se brzina slanja podataka povećava (1,23 Mb/s za Tablica 3.5), raste broj neodgovorenih poruka i prioritet koji poveznik dobiva od operativnog sustava. Minimalni i prosječni RTT – ovi padaju a odstupanje maksimalnih vrijednosti RTT – ova se pripisuje već opisanoj logici rada poveznika.

Tablica 3.6: Skup mjerenja s vremenskim razmakom slanja poruka 140 µs.

	Kružno vrijeme kašnjenja – RTT [µs]				
Ime poruke	**Minimalno**	**Maksimalno**	**Prosječno**	**Broj mjerenja**	**Ispustito**
Frame_11_0_A_ChA	0,002	485	106	5000	2,26%
Frame_16_0_A_ChA	0,058	225	104	5000	2,00%
Frame_18_0_B_ChB	0,013	343	106	5000	1,98%
Frame_31_0_B_ChB	0,061	1177	106	5000	1,84%
Frame_63_0_A_ChA	0,041	2237	109	5000	2,10%
Frame_72_0_A_ChA	0,124	322	106	5000	2,04%
msg_1027	0,08	1392	106	5000	2,02%
msg_12679	0,031	248	106	5000	2,14%
msg_5961	0,082	652	101	5000	2,22%
msg_6518	0,018	1237	114	5000	2,36%
msg_10600	0,005	1067	104	5000	2,46%
Sve poruke	0,14	224	109	5005	2,32%
Sažetak	**0,002**	**2237**	**106,4**	**-**	**2,14%**

Uz brzinu od 1,31 Mb/s (Tablica 3.6) ispuštanje paketa postaje značajnije. Ovisno je mjestu ugradnje, upitno je korištenje poveznika u automobilskoj industriji s ovim brzinama.

Tablica 3.7: Skup mjerenja s vremenskim razmakom slanja poruka 120 µs.

	Kružno vrijeme kašnjenja – RTT [µs]				
Ime poruke	**Minimalno**	**Maksimalno**	**Prosječno**	**Broj mjerenja**	**Ispustito**
Frame_11_0_A_ChA	0,077	195	86	5000	12,74%
Frame_16_0_A_ChA	0,007	735	88	5000	12,98%
Frame_18_0_B_ChB	0,023	216	88	5000	12,82%
Frame_31_0_B_ChB	0,013	630	87	5000	12,40%
Frame_63_0_A_ChA	0,061	1687	88	5000	12,00%
Frame_72_0_A_ChA	0,032	215	87	5000	12,30%
msg_1027	0,005	191	87	5000	12,48%
msg_12679	0,033	3034	94	5000	9,16%
msg_5961	0,029	200	87	5000	12,10%
msg_6518	0,021	1952	87	5000	12,34%
msg_10600	0,01	193	87	5000	12,58%
Sve poruke	0,039	2483	87	5005	11,97%
Sažetak	**0,005**	**3034**	**87,75**	**-**	**12,16%**

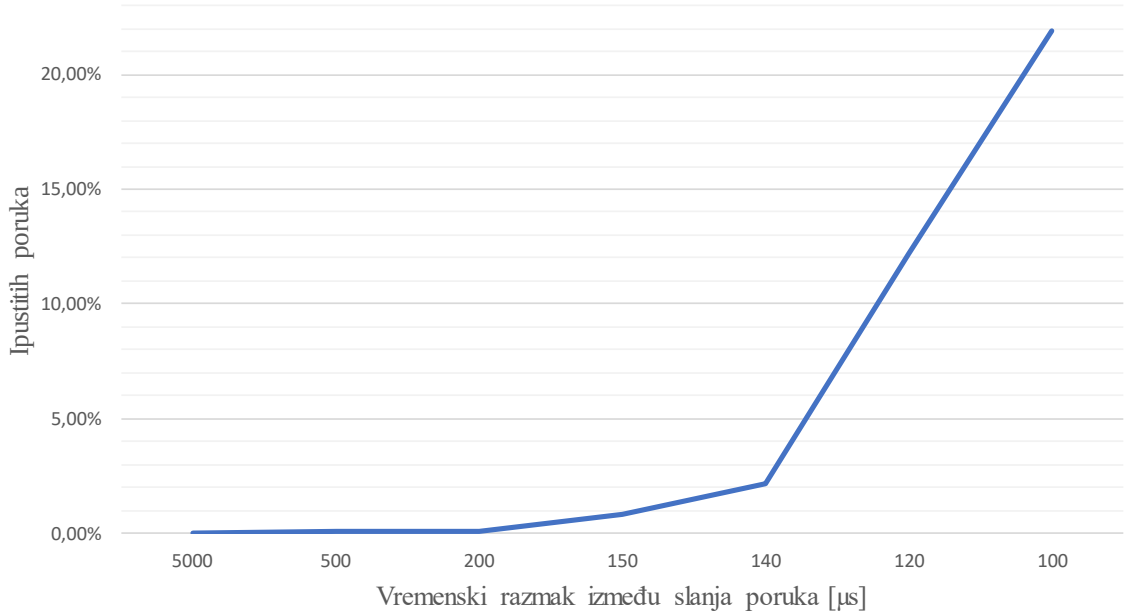
Kada se podatci šalju svakih 120 µs (brzina 1,53 Mb/s) broj neodgovorenih poruka je narastao na razinu koja je opasna u automobilskoj industriji. Brzina obrade podataka od 2,12 Mb/s nije realna brojka jer bi svaka neobrađena poruka trebala smanjivati konačnu brzinu.

Tablica 3.8: Skup mjerenja s vremenskim razmakom slanja poruka 100 µs.

	Kružno vrijeme kašnjenja – RTT [µs]				
Ime poruke	**Minimalno**	**Maksimalno**	**Prosječno**	**Broj mjerenja**	**Ispustito**
Frame_11_0_A_ChA	0,002	182	78	5000	22,02%
Frame_16_0_A_ChA	0,001	198	78	5000	21,80%
Frame_18_0_B_ChB	0,006	1418	78	5000	21,88%
Frame_31_0_B_ChB	0,058	1540	78	5000	21,78%
Frame_63_0_A_ChA	0,011	343	78	5000	22,10%
Frame_72_0_A_ChA	0,029	1523	79	5000	22,04%
msg_1027	0,115	1694	78	5000	21,80%
msg_12679	0,101	2837	79	5000	21,50%
msg_5961	0,023	613	78	5000	22,22%
msg_6518	0,035	248	78	5000	21,80%
msg_10600	0,032	375	78	5000	22,02%
Sve poruke	0,115	227	78	5005	21,92%
Sažetak	**0,001**	**2837**	**78,16**	**-**	**21,91%**

Posljednji skup mjerenja (Tablica 3.8) obavljen je isključivo zbog usporedbe performansi prije i poslije optimizacije. Poveznik na ovim brzinama predstavlja nesigurni uređaj i ne smije se koristiti.

Povezniku vrsta i veličina poruke ne utječe na performanse već samo brzina slanja podataka. Povećanjem brzine slanja podataka povećava se i odbacivanje istih (Slika 3.7) pa bi upravo na temelju dozvoljenog postotka odbačenih poruka dizajner automobilske mreže trebao odlučiti o implementaciji poveznika.



Slika 3.7: Grafički prikaz ovisnosti broja ispuštiti podataka o brzini slanja.

Tablica 3.9: Prikaz vremena pokretana poveznika.

Redni broj	**Vrijeme pokretanja [µs]**
1.	7445
2.	6430
3.	6096
4.	6919
5.	7044
6.	6558
7.	6274
8.	6213
9.	7600
10.	6503
Prosjek	**6708,2**

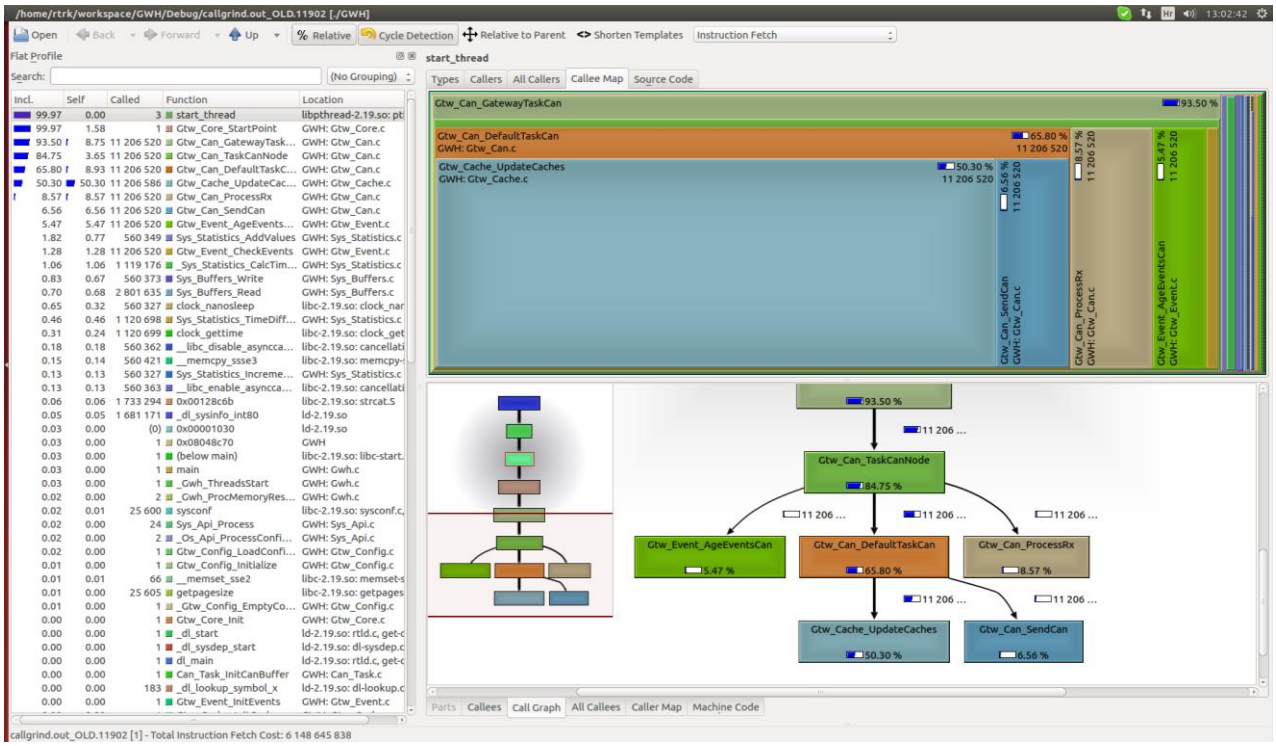
Na kraju mjerenja performansi izmjereno je vrijeme pokretanja poveznika (Tablica 3.9). To je vrijeme koje je potrebno proteći od pokretanja poveznika do trenutka kada poveznik bude spreman obrađivati podatke. Vrijeme pokretanja od 6,7 ms ne predstavlja problem u automobilskoj industriji.

4. OPTIMIZACIJA POVEZNIKA

Poveznik predstavlja veliku količinu programskog koda koji se izvršava na slabom sklopovlju. Rezultat toga su ograničenja u kapacitetu obrade podataka. Taj se kapacitet povećava optimizacijom koda koja je opisana u nastavku.

4.1. Izmjene koda

Na početku optimizacije je bilo potrebno odrediti koji dio koda optimizirati. Uvid u izvođenje programa i ispis statističkih podataka o izvođenju istog prikazuju programerski analitički alati. Jedan takav alat je Valgrind – analitički program otvorenog koda. Nakon jednostavne instalacije putem terminala, bilo je potrebno se navigirati u mapu s izvršnom datotekom poveznika, naredbom pokrenuti Valgrind i predati mu potrebne parametre poput imena izvrše datoteke. Poveznik se pokrenuo pod okriljem Valgrinda, poslano mu je nekoliko poruka na usmjeravanje i potom je prekinuto izvršavanje poveznika. Valgrind je kreirao *.out* datoteku. Novonastala datoteka je otvorena KCachegrind programom za prikaz informacija izvođenja programa (poveznika).



Slika 4.1: Prikaz informacija izvođenja programa u KCachegrindu.

Dobiveni prikaz (Slika 4.1) je sadržavao brojne informacije za svaku funkciju programa; apsolutna i relativna vremena trajanja funkcija, broj pozivanja funkcija, dijagram tijeka i dr. Lako uočljivi problem je bilo preveliko relativno trajanje jedne funkcije – *Gtw_Cache_UpdateCache*. Sporna je funkcija sama trošila 50,3 % ukupnog vremena izvršavanja poveznika pa je optimizacija krenula upravo od te funkcije.

Analizom izvornog koda ustanovljeno je da funkcija ne posjeduje kod koji bi mogao uzrokovati dugotrajno izvršavanje aplikacije. Problem je bio što se sporna funkcija često poziva. Navedeni je zaključak rezultirao izmjenom strukture koda poveznika čime se smanjio broj poziva *Gtw_Cache_UpdateCache* funkcije.

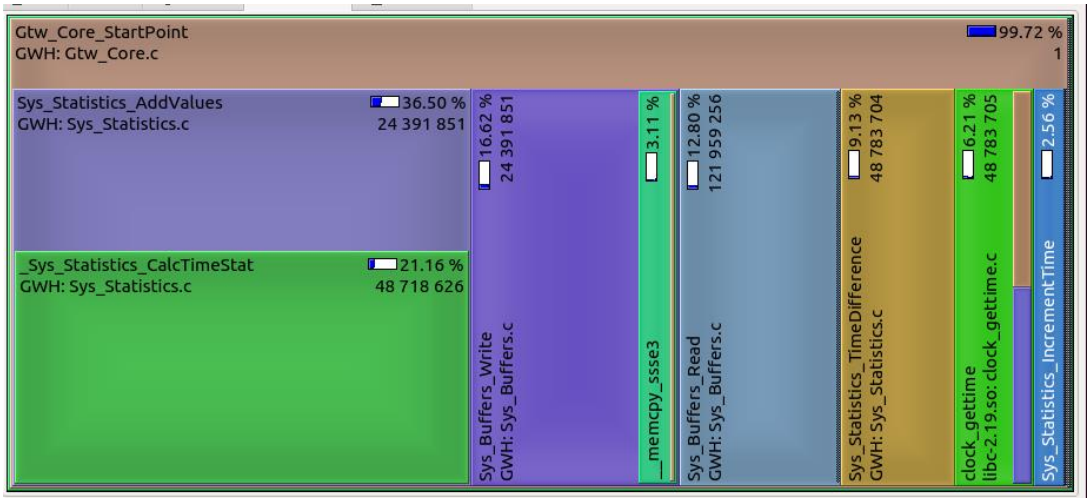
Analizom izvornog koda uočene su neke nepravilnosti u strukturi koda. U bloku čitanja podataka (Slika 3.5) pristigla poruka se analizira. Ako se pristigla poruka treba usmjeriti na FlexRay poruku, postavlja se zastavica za pokretanje bloka „FlexRay usmjeravanje“. Ista logika nije korištena za CAN poruke pa se blok „CAN usmjeravanje“ svaki puta izvršavao – upisano u kod 20 puta. Ovisno o pristigloj poruci, blok za CAN usmjeravanje se često ne bi trebao izvršavati. Upravo je sporni blok koda imao višestruka pozivanja *Gtw_Cache_UpdateCache* funkcije.

Optimizacija je obavljena na način da bi se u bloku čitanja podataka sve pristigle poruke analizirale; ako je potrebno usmjeravanje na FlexRay, postavlja se zastavica za pokretanje FlexRay bloka, ako je potrebno usmjeravanje na CAN, postavlja se zastavica za pokretanje CAN bloka. Bez postavljenih zastavica se blokovi ne izvršavaju. Blok za CAN usmjeravanje je sadržavao brojne mehanizme koji su bili neophodni i za FlexRay usmjeravanje (zato se početno uvijek izvršavao). Dio tih mehanizama je prebačen u blok čitanja podataka. Zbog velike količine koda i složene strukture, nakon restrukturiranja koda pojavili su se neki novi problemi:

- Mehanizmi za usmjeravanje FlexRay poruka su čitali podatke iz pogrešnih spremnika,
- Mehanizmi upravljanja zastavicama su morali biti izmijenjeni,
- Dodani su dodatni brojači koji uz zastavice signaliziraju broj preostalih poruka za obradu,
- Zastavice postojanosti CAN poruka su bile ograničene na usmjeravanje samo jedne CAN poruke po jednoj iteraciji bloka za CAN usmjeravanje.

Posljednje navedeni problem bi oduzeo previše vremena pa je poveznik ograničen na obradu samo jedne pristigle poruke po svakoj iteraciji *while* petlje. Restrukturiranje koda, bez obzira koliko jednostavno zvučalo, vremenski je potrajalo zbog dodatnih problema koji su se pojavili.

Pred kraj optimizacije uočeni su dijelovi koda čije se postojanje nije moglo samostalno objasniti a usporavali su poveznik. Brisanjem toga koda poveznik je postao brži u svakoj iteraciji *while* petlje. Nakon optimizacije koda, osim urednije strukture koda reduciran je broj poziva *Gtw_Cache_UpdateCache* funkcije što je rezultiralo povećanjem performansi. Slika 4.2. prikazuje raspodjelu vremena po funkcijama nakon optimizacije. Vidljiv je nestanak funkcija koje su trošile većinu vremena izvođenja aplikacije.



Slika 4.2: Prikaz raspodjele vremena po funkcijama nakon optimizacije.

4.2. Rezultati optimizacije

Nakon izmjena koda opisanih u prethodnom poglavlju, povezniku su ponovno izmjerene performanse. Performanse su uspoređene s performansama prije optimizacije. Kako su performanse poslije optimizacije značajno poboljšane, neće biti prikazane sve tablice kao u poglavlju 3.3 već samo one najznačajnije. Usporedba performansi će biti grafički prikazana.

Tablica 4.1: Skup mjerenja s vremenskim razmakom slanja poruka 5 ms.

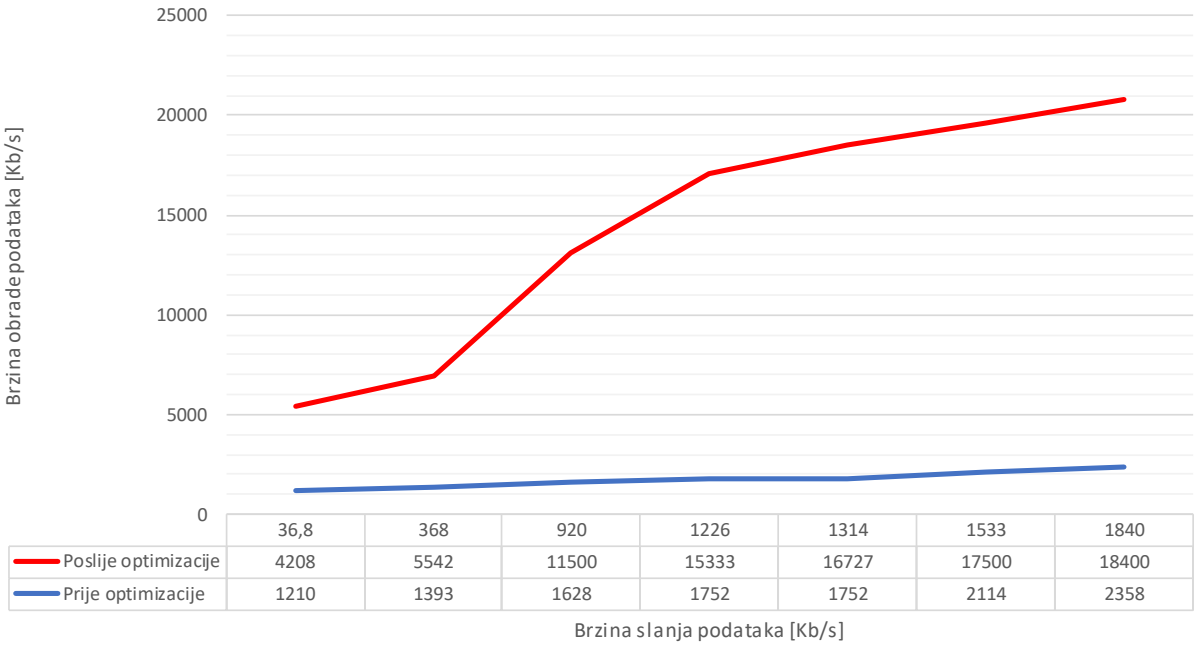
	Kružno vrijeme kašnjenja – RTT [µs]				
Ime poruke	**Minimalno**	**Maksimalno**	**Prosječno**	**Broj mjerenja**	**Ispustito**
Frame_11_0_A_ChA	5,34	5095	52	5000	0%
Frame_16_0_A_ChA	5,185	237	47	5000	0%
Frame_18_0_B_ChB	2,67	161	37	5000	0%
Frame_31_0_B_ChB	5,31	7476	52	5000	0%
Frame_63_0_A_ChA	3,74	106	39	5000	0%
Frame_72_0_A_ChA	4,87	322	43	5000	0%
msg_1027	4,545	157	43	5000	0%
msg_12679	6,139	3685	45	5000	0%
msg_5961	4,225	1135	45	5000	0%
msg_6518	0,9	478	37	5000	0%
msg_10600	6,82	140	41	5000	0%
Sve poruke	6,987	1617	42	5005	0%
Sažetak	**0,9**	**7476**	**43,72**	**-**	**0%**

U Tablica 4.1 vidljivo je značajno smanjenje kružnog kašnjenja s 151 µs na 43 µs. To trostruko smanjenje potrebnog vremena za obradu poruke je rezultat restrukturiranja izvornog koda poveznika. I dalje je uočljiva stohastičnost podataka koja se događa iz istih razloga kao i prije optimizacije. Također ostaje pojava povećavanja prioriteta programa s povećanjem brzine slanja podataka.

Tablica 4.2: Skup mjerenja s vremenskim razmakom slanja poruka 100 µs.

	Kružno vrijeme kašnjenja – RTT [µs]				
Ime poruke	**Minimalno**	**Maksimalno**	**Prosječno**	**Broj mjerenja**	**Ispustito**
Frame_11_0_A_ChA	1,311	284	10	5000	0%
Frame_16_0_A_ChA	0,628	1038	11	5000	0%
Frame_18_0_B_ChB	0,515	2047	11	5000	0%
Frame_31_0_B_ChB	2,999	171	9	5000	0%
Frame_63_0_A_ChA	2,678	394	9	5000	0%
Frame_72_0_A_ChA	1,784	178	10	5000	0%
msg_1027	0,4	4564	11	5000	0%
msg_12679	1,819	2426	10	5000	0%
msg_5961	853	1820	9	5000	0%
msg_6518	1,256	1687	11	5000	0%
msg_10600	0,1	160	9	5000	0%
Sve poruke	1,321	251	11	5005	0%
Sažetak	**0,1**	**4564**	**10**	**-**	**0%**

Najveći pomak vidljiv je tek u Tablica 4.2 gdje ispred brzine obrade dolaze podatci o odbačenim porukama. Pri ovoj brzini slanja podataka (1,84 Mb/s) poveznik je prije optimizacije odbacivao 21 % svih poruka što ga je činilo nesigurnim uređajem za automobilsku industriju. Nakon optimizacije nema odbacivanja podataka što govori da je optimizacija uspješno obavljena. U sjeni povećanja pouzdanosti ne smije neopaženo proći ni ubrzanje obrade poruka s 2,35 Mb/s na 18,4 Mb/s (RTT pao s 78 µs na 10 µs). Usporedba brzine prikazana je grafički na Slika 4.3 gdje je moguće vidjeti povećanje brzine obrade podataka preko 7 puta.



	36,8	368	920	1226	1314	1533	1840
Poslije optimizacije	4208	5542	11500	15333	16727	17500	18400
Prije optimizacije	1210	1393	1628	1752	1752	2114	2358

Slika 4.3: Usporedba performansi prije i poslije optimizacije.

5. ZAKLJUČAK

Komunikacija *točka – točka* u automobilskoj industriji je zahtijevala previše vodova i složenu komunikacijsku strukturu. Rješenje problema je bilo u korištenju sabirnih vodova. Predstavnici takvih sabirnih vodova su CAN i FlexRay standard komunikacije. Nešto stariji – CAN je događajno orijentiran standard komunikacije koji je u velikoj mjeri zaživio u automobilskoj industriji. Mlađi i brži FlexRay je deterministički orijentiran protokol koji sa svojim prednostima *krade* zastupljenost CAN - u.

Kako CAN i FlexRay standard nisu međusobno kompatibilni a zahtjevi industrije su da moraju imati mogućnost izmjenjivanja podataka, potrebni su mrežni poveznici. Jedan takav poveznik je razvilo poduzeće RT-RK Osijek d.o.o.

Poveznik je bilo potrebno *debugirati* kako bi ga se moglo testirati. U konačnici je poveznik proradio i izmjerene su mu performanse te su mu određene granice rada o ovisnosti o brzini i broju odbačenih paketa. Algoritam nije pouzdan pri brzinama slanja podataka većim od 1,2 Mb/s.

Optimizacijom je poveznik dobio nešto uredniju strukturu koda i poprilično poboljšanje performansi. To poboljšanje performansi je rezultiralo povećanjem brzine obrade podataka i do 7 puta ali ono još bitnije – povećana mu je pouzdanost što je rezultirao povećanjem mogućeg opterećenja (brzine slanja podataka) do 1,84 Mb/s odnosno 53 %.

LITERATURA

[1] Wikipedija, en.wikipedia.org/wiki/CAN_bus, (11.kolovoz 2017.).

[2] Vector, elearning.vector.com, (26. lipanj 2017.).

[3] National Instruments, www.ni.com/white-paper/3352/en/, (15.kolovoz 2017.)

SAŽETAK

Autoindustrija generira visoke zahtjeve za tehnologijom. Među ostalima tu se nalaze i komunikacijske tehnologije. Primjeri komunikacijskih tehnologija u autoindustriji su CAN i FlexRay standardi koji su pobliže opisani u radu. Rad opisuje jedan poveznik u svojim ranim fazama razvoja sa svojim manama (problemima). Opisane su metode primijenjene u svrhu otklanjanja problema i ispravnog izvršavanja programa. Program je testiran u izvršavanju svoje osnovne funkcije i prikazane su statistike performansi programa. Dodatno, program je optimiziran u skladu s mogućnostima. Uspoređene su performanse prije i poslije optimizacije.

Ključne riječi: CAN, FlexRay, poveznik, optimizacija, mjerenje performansi, ispravljanje problema

ABSTRACT

Automotive industry has high technological requirements and one of them are communication technologies. Some of those technologies in automotive industry are CAN and FlexRay standards, which will be described in this thesis. This final paper describes an early version of a gateway and its problems, along with methods used to fix those problems and make the gateway work properly. The program is tested and it has been monitored along with performance statistics. Additionally, the program has been optimized according to its possibilities and the performances of non-optimized and optimized versions were compared.

Keywords: CAN, FlexRay, gateway, optimization, performance measurement, debuging

ŽIVOTOPIS

Zlatko Janjić rođen je 3.srpnja 1993. u Vinkovcima. Do 2000. godine stanuje u Privlaci gdje završava osnovnu školu Stjepana Antolovića. Srednju Tehničku školu Ruđera Boškovića završava u Vinkovcima i stječe zvanje elektrotehničara. Za vrijeme srednje škole sudjeluje na brojnim natjecanjima od koji su izdvojena državna natjecanja iz Osnova elektrotehnike i mjerenja u elektrotehnici te fizika.

2012.godine upisuje preddiplomski studij elektrotehnike na Elektrotehničkom fakultetu u Osijeku, smjer Komunikacije i informatika. Godine 2015. dobiva Priznanje za postignut uspjeh u studiranju.

2015.godine upisuje diplomski studij elektrotehnike na Fakultetu elektrotehnike, računalstva i informacijskih tehnologija Osijek, smjer Komunikacije i informatika – Mrežne tehnologije. Godine 2017. dobiva drugo Priznanje za postignut uspjeh u studiranju.

Pasivno koristi engleski jezik, informatički je pismen i služi se računalnim tehnologijama poput programa za dizajn elektroničkih uređaja, multimedijskih alata, programskih jezika C, C++, C#, Python, Java, Assembler te alata za sistemsku administraciju.