Filtriranje i klasifikacija prometa s pronalaženjem grešaka u automotiv podatkovnom prometu

Roguljić, Luka

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

FILTRIRANJE I KLASIFIKACIJA PROMETA S PRONALAŽENJEM GREŠAKA U AUTOMOTIV PODATKOVNOM PROMETU

Diplomski rad

Luka Roguljić

Osijek, 2019.

SADRŽAJ

1. UVOD

Automobilska industrija jedna je od najraširenijih i najbrže rastućih industrija na svijetu. Trenutni smjer u kojemu ide industrija orijentiran je na stvaranje potpuno autonomnog vozila na električni pogon. Potpuno autonomno vozilo podrazumijeva nepostojanje potrebe za intervencijom vozača pri vožnji. Očekuje se da uvođenje autonomne vožnje uveliko smanji broj nesreća na cestama, naročito onih sa smrtnim posljedicama, jer su većina nesreća na cestama posljedica čovjekove greške, a ne okoline. Ako je suditi po statistikama, broj nesreća sa smrtnim posljedicama u sljedećih 10 godina će se udvostručiti [1]. Da bi vozilo postiglo razinu potpune autonomnosti, mora obrađivati informacije iz svog okruženja koje inače obrađuje vozač. Percipiranje okoline od strane automobila zahtjeva veliki broj senzora u automobilu, čije je signale potrebno obrađivati pomoću različitih računalnih algoritama.

Da bi se omogućio prijenos signala sa senzora do mjesta obrade i da bi se nakon obrade informacije poslale poruke vozilu o tome koju radnju treba odraditi, potrebno je imati siguran, brz, pouzdan i jasno definiran način prijenosa podataka u raznim smjerovima. Kako bi se osiguran takav prijenos, postoji niz definiranih protokola za prijenos. Neki od protokola koji se koriste u automobilskoj industriji su CAN (engl. *Controller Area Network)*, FlexRay, LIN (engl. *Local Interconnect Network),* Ethernet, J1939, MOST (engl. *Media Oriented Systems Transport).*

U sklopu ovog diplomskog rada izrađena je Python skripta koja omogućuje raščlanjivanje snimljenog prometa s CAN sabirnice, pretraživanje poslanih i primljenih poruka, izradu statističke analize te detekciju pogrešaka ili izgubljenih poruka. Rezultati su prikazani u tabličnom zapisu i pomoću odgovarajućeg grafičkog prikaza radi lakše predodžbe rezultata.

Rad je strukturiran na sljedeći način. U drugom poglavlju opisan je CAN protokol i dva glavna zahtjeva koja se odnose na pravovremenost očitavanja poslane poruke i ispravnost podatkovnog bloka poslane poruke. Također dane su i sažete informacije o postojećim rješenjima koja se bave sličnom tematikom. Period očitavanja poslane poruke predstavlja vrijeme koje prođe od trenutka slanja poruke do trenutka njenog očitavanja. Ako je poslana poruka očitana unutar zadanog vremena (u ovom slučaju 7.5 milisekundi), radi se usporedba poslanog podatkovnog bloka i primljenog podatkovnog bloka. U trećem poglavlju opisan je vlastiti algoritam za filtriranje i klasifikaciju podatkovnog prometa te detekciju pogrešaka i izgubljenih poruka stvoren u sklopu ovog diplomskog rada. Svaki modul stvorenog vlastitog algoritma detaljno je opisan. U četvrtom poglavlju opisan je način verifikacije ispravnosti rada algoritma za filtriranje, klasifikaciju

podatkovnog prometa i detekciju pogrešaka i izgubljenih poruka. Verifikacija se radila tako da se provjeravala ispunjenost unaprijed postavljenih zahtjeva. Verifikacija je rađena u sklopu programskog rješenja s vlastitim funkcijama za provjeru ispravnosti određenog dijela algoritma. U petom poglavlju dan je zaključak rada.

2. KOMUNIKACIJSKI PROTOKOLI ZA KOMUNIKACIJU U AUTOMOBILIMA

Komunikacijski protokol je skup određenih pravila za razmjenu informacija između dvaju ili više entiteta u mreži. Entitet može biti kamera, različiti senzori na automobilu koji šalju informacije putem sabirnice, a entiteti koji obrađuju te informacije i na njih reagiraju su jedinice za kontrolu motora (*engl. Engine Control Unit),* aktuatori, procesori i sl. Svaki protokol mora proći postupak standardizacije koju obavlja Međunarodna organizacija za standardizaciju (*engl. International Organization for Standardization*) ili neka druga organizacija. Nakon provedene standardizacije protokol je spreman za široku upotrebu. U prošlosti su elektronički uređaji u automobilima bili povezani na *„point-to-point"* način. Razvojem automobilske industrije k autonomnim vozilima, povećava se broj elektroničkih komponenata u automobilima i samim time povećava se broj žica u automobilima, što dovodi do povećanja cijene samog vozila. Zbog povećanja broja elektroničkih uređaja u automobilima koji međusobno komuniciraju, potrebno je koristiti komunikacijske protokole radi utvrđivanja pravila komunikacije i prava pristupa sabirnici. Poznatiji protokoli koji se koriste u automobilskoj industriji su: CAN, FlexRay, LIN, Ethernet, J1939 i MOST [2].

FlexRay je serijski komunikacijski protokol koji je razvijen od strane BMW-a i DaimlerChysler u suradnji s tvrtkama za proizvodnju čipova Motorola i Philips. Dizajniran je s namjerom da bude brži od CAN protokola. FlexRay podržava brzinu prijenosa podataka do 10 Mb/s [3]. Sastoji se od FlexRay čvorova koji su povezani na sabirnicu. Podržava vremenski-okidivu komunikaciju, ali i komunikaciju upravljanu događajem. Mreža zasnovana na FlexRay protokolu ima dvije sabirnice, što doprinosi većoj toleranciji na greške i njegovom korištenju u sigurnosno-kritičnim sustavima. Ako se dogodi situacija da je jedna sabirnica izvan pogona, komunikacija na drugoj sabirnici se može nastaviti. FlexRay podatkovni okvir sastoji se od: zaglavlja (*engl. Header*), podatkovnog bloka i završnog dijela. Uz statički segment poruke koji služi za slanje statičkih poruka tijekom komunikacije, podatkovni okvir može sadržavati i dinamički segment. U dinamičkom segmentu mogu se slati podaci različitih duljina i on je upravljan događajem *(engl. Event driven)*. Mnogi se inženjeri diljem svijeta bave tematikom vezanom za FlexRay protokol i komunikaciju zasnovanu na njemu, a neki od rezultata njihova rada mogu se pronaći u [4],[5],[6].

LIN protokol je serijski komunikacijski protokol. Razvijen je za potrebe sustava u kojima vrijeme nije kritičan parametar. Brzina prijenosa podataka ide do 20 kb/s. Njegova primjena je ograničena

i on nikako ne smije biti korišten u sustavima gdje kašnjenje stvara problem. On se koristi u sustavima za kontrolu rada klima uređaja, unutarnjih svjetala u automobilu i sl [7].

MOST protokol je komunikacijski protokol u automobilima zadužen za multimediju i sustave za informiranje i zabavu (engl. *Infotaiment*). Omogućava prijenos audio i video podataka i kontrolnih informacija. Mrežna topologija MOST-a može biti zvijezda ili prsten. Brzina prijenosa podataka ide do 25 Mb/s [7]. Aplikacije koje koriste MOST su: GPS (engl. *Global Positioning System*), sustavi za zabavu i informiranje, prikaz videa, radio i sl.

Ethernet protokol je standardni protokol koji se koristi u lokalnim mrežama. U automobilima se počeo koristiti nedavno kada su uočene njegove prednosti. Prednosti Ethernet protokola su: brzina prijenosa podataka do 100 Mb/s, niska cijena implementacije, pruža visoki stupanj skalabilnosti (*engl. Scalability*) [8]. Implementacijom Ethernet-a u automobilu, moguće je smanjiti cijenu povezivanja do 80% i masu kablova (*engl. Cable Weight*) do 30% [9].

SAE (*engl. Society of Automotive Engineers)* J1939 je protokol u automobilima koji se koristi za komunikaciju i dijagnostiku između komponenata automobila. SAE J1939 koristi CAN kao fizički sloj. Podržava prijenos podataka do 250 kb/s. Komunikacija može biti svaki sa svakim (*engl. Peer-to-peer*) ili emitiranje jedan svima (*engl. Broadcast*) [10]. Daljnji fokus ovoga rada je na CAN komunikacijskom protokolu te će on u nastavku biti detaljnije opisan i analiziran.

2.1 Komunikacija zasnovana na CAN protokolu

Početkom 1980-tih godina Bosch je počeo razvijati serijski komunikacijski sustav zbog potrebe za standardnim protokolom za komunikaciju u automobilima. Godine 1986. Bosch je svijetu predstavio CAN protokol u Detroitu. CAN protokol je poznat po tome što ima vrlo veliku pouzdanost prijenosa podataka i, što je najvažnije, zadovoljava zahtjeve za prijenosom podataka u stvarnom vremenu u automobilskoj industriji [11].

CAN tehnologija je standardizirana od 1994. godine s četiri ISO dokumenata. ISO 11898-1 [12] opisuje sami CAN protokol i komunikaciju upravljanu događajem. ISO 11898-2 [13] i ISO 11898-3 [14] pokrivaju dva podsloja za podatkovnu komunikaciju: PMA (engl. *Physical Medium Attachment*) i PMS (engl. *Physical Medium Specification*). Oni opisuju dva različita CAN fizička sloja: „*high-speed*“ CAN i „*low-speed*“ CAN. Prethodno navedena dva CAN fizička sloja razlikuju se po naponu i brzini prijenosa podataka. Brzina prijenosa podataka koju opisuje ISO 11898-3 je 125 *kbit/s,* a brzina prijenosa podataka koju opisuje ISO 11898-2 je 1 *Mbit/s* i zbog te

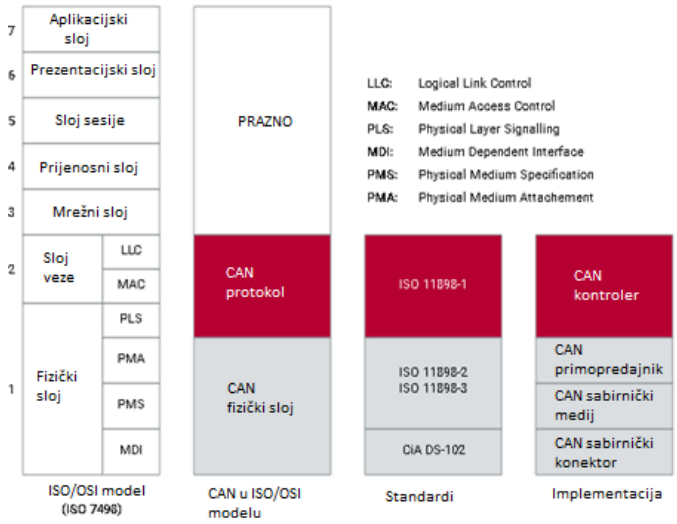
velike brzine, ISO 11898-2 se koristi u pogonskim sklopovima i podvozju. Konačno, ISO 11898-4 je produljenje DLL (*engl. Data Link Layer*) sloja koje dodaje mogućnost vremenski okidive komunikacije (vrsta komunikacije koja se događa periodično, okida se nakon definiranog perioda, a ne okida se određenim događajem).

Od sedam slojeva ISO/OSI modela, CAN se prostire na fizički sloj i sloj veze. Dijelovi fizičkog sloja su:

- Sučelje ovisno o mediju (*engl. Medium Dependent Interface - MDI*)
- Fizička medijska specifikacija (*engl. Physical Medium Specification - PMS*)
- Prilog fizičkom mediju (*engl. Physical Medium Attachment - PMA*)
- Signalizacija fizičkog sloja (*engl. Physical Layer Signalling - PLS*)

Dijelovi sloja veze su:

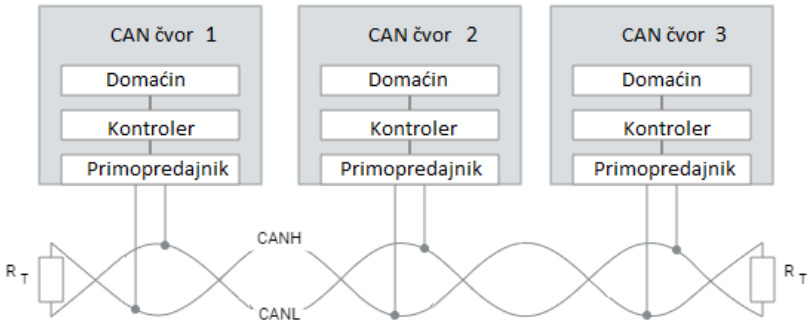
- Nadzor pristupa mediju (*engl. Medium Access Control*)
- Kontrola logičke veze (*engl. Logical Link Control*)



Sl. 2.1. *Standardizacija i implementacija CAN protokola* [15]

2.1.1 Struktura CAN mreže

CAN mreža se sastoji od određenog broja CAN čvorova i oni su povezani preko CAN sabirnice. Standardna topologija mreže koja se koristi je linijska topologija gdje je svaki ECU (engl. *Electronic Control Unit*) povezan s CAN sabirnicom. Topologija pasivne zvijezde se također može koristiti kao alternativna topologija. Koristi se neoklopljena upletena parica (engl. *Unshielded Twisted Pair*) za izvedbu mreže u fizičkom sloju, gdje linijski otpor ne bi trebao prelaziti 60 mΩ. Maksimalna duljina CAN sabirnice je 40 metara za brzinu prijenosa od 1 Mb/s, i na krajevima sabirnice su stavljeni otpornici koji služe za prevenciju refleksije. U ISO 11898 standardu maksimalno dozvoljeni broj čvorova na jednoj liniji je 32. Na slici 2.2. dan je prikaz spajanja triju CAN čvorova na sabirnicu. CANH predstavlja žicu sabirnice čiji je napon 3.75 V prilikom prijenosa podataka, a CANL predstavlja žicu sabirnice čiji je napon 1.25 V prilikom prijenosa podataka. Kada sabirnica ne prenosi podatke, napon na obje žice je 2.5 V. Na svakom kraju sabirnice nalazi se otpornik koji sprječava refleksiju signala.

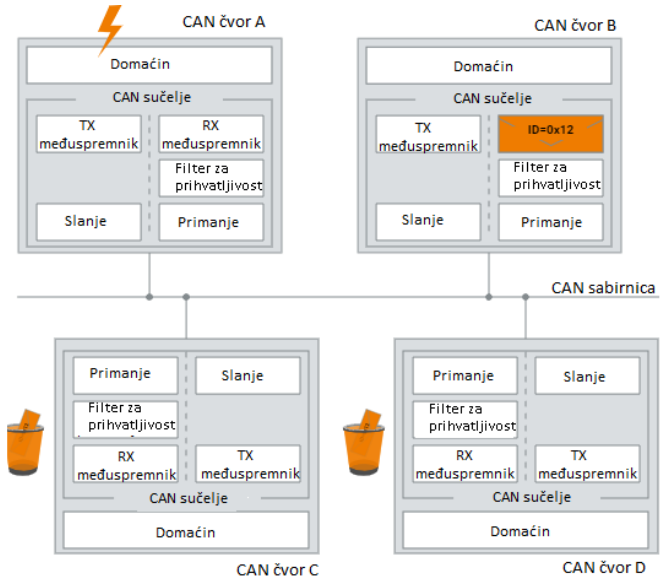


Sl. 2.2. *Prikaz CAN čvorova spojenih na uparenu CAN sabirnicu* [16]

2.1.2 Način komunikacije u CAN protokolu

Svi akteri koji sudjeluju u komunikaciji su ravnopravni i mogu proizvoljno komunicirati. U slučaju kolizije (događaj kada se na istoj sabirnici istovremeno žele poslati dvije ili više poruka) tijekom komunikacije, koristi se arbitraža da se odredi tko ima prioritet. Komunikacija je decentralizirana tako što svaki čvor ima pravo pristupa sabirnici. Kada to ne bi bilo tako, onda bi kvar čvora koji jedini ima pristup sabirnici značio kvar cijele komunikacije. Arhitektura više-nadređenih i linijska topologija omogućavaju svakom čvoru da postavi poruku u CAN mrežu. Komunikacijski kanal će biti zauzet samo onda kada postoji nova poruka koja treba biti poslana. Takav pristup omogućuje vrlo brzi pristup samoj sabirnici.

Svaka poruka koja je poslana može biti primljena od strane svakog čvora u mreži. Poruka je prepoznata preko svoga identifikatora koji je jedinstven za svaki tip poruke. Važna stavka u primanju poruke je filter za prihvaćanje, odnosno filter koji provjerava je li primljena poruka relevantna za čvor koji ju je primio ili ne. U slučaju da je filter prihvatio poruku, onda je ona primljena. Ako poruka nije prihvaćena ona se onda ne upisuje u RX međuspremnik i ne obrađuje se nego ide u „smeće“. RX međuspremnik predstavlja spremnik u CAN čvoru koji je zadužen za prihvaćanje poruke sa sabirnice. Takva komunikacija je prikazana na slici 2.3., gdje CAN čvor A šalje poruku s identifikatorom vrijednosti 0x12, CAN čvor B je primatelj te poruke, dok CAN čvorovi C i D to nisu. Poruka iz TX međuspremnika CAN čvora A stavlja se na sabirnicu te ju primaju svi čvorovi. Pošto je samo CAN čvor B primatelj, njegov filter za prihvaćanje će dati signal da se poruka može spremiti u RX međuspremnik, a CAN čvorovi C i D poruku neće proslijediti u RX međuspremnik nego će je baciti u „smeće“. Važno je napomenuti da ID manje vrijednosti ima prednost u zauzeću sabirnice. Taj postupak određivanja zove se postupak arbitraže. On dodjeljuje pristup sabirnici onoj poruci koja ima najmanju vrijednost identifikatora. Npr. u slučaju da poruka s identifikatorom vrijednost 0x12 želi pristupiti sabirnici u isto vrijeme kada i neka druga poruka s identifikatorom vrijednosti 0xFF, postupak arbitraže dodjeljuje poruci s identifikatorom 0x12 prednost pristupa jer je njezina vrijednost manja.



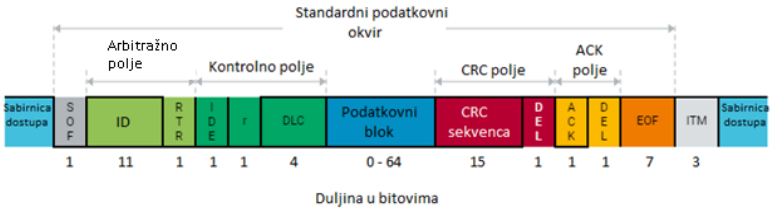
Sl. 2.3. *Primjer stanja poslane poruke u CAN mreži* [17]

2.1.3 Tipovi podatkovnih okvira u CAN protokolu

Za prijenos podataka ISO 11898-1 opisuje podatkovni okvir. Jedan podatkovni okvir prenosi maksimalni podatkovni blok od 8 bajtova. Oko podatkovnog bloka nalaze se ostale varijable poruke nužne za uspješnu komunikaciju, a to su: identifikator (ID), kod podatkovne duljine (engl. *Data Length Code, DLC*), kontrolna suma (*engl. Cyclic Redundancy Check - CRC*) i RX polje potvrde primitka poruke.

Dijelovi poruke su kako slijedi:

- SOF (engl. *Start Of Frame*) – početak okvira, koristi se za sinkronizaciju cijele mreže;
- ID i RTR – određuju prioritet poslanog podatkovnog okvira (ID), a RTR bit (engl. *Remote Transmission Request*) omogućava primatelju poruke da prepozna o kojem tipu okvira se radi.
- IDE (engl. *Identifier Extension Bit*) – služi za razlikovanje standardnog formata i produljenog formata poruke. Standardni format poruke ima 11 bitova, a produženi ima 29 bitova.
- DLC (engl. *Data Length Code*)– polje koje daje podatak koliko bajtova ima podatkovni blok. U tablici 2.1 predstavljen je broj u binarnom zapisu za svaku vrijednost DLC-a u odnosu na broj bajtova koji se prenose u podatkovnom bloku. Zelena polja u tablici označavaju koji bajtovi se prenose u podatkovnom bloku za svaku moguću vrijednost DLC-a.
- CRC i ACK polje (engl. *Acknowledge Field*) – CRC se koristi za provjeru uspješno prenesenog podatkovnog bloka, odnosno za otkrivanje greške u podatkovnom bloku. U slučaju da je došlo do pogreške u prijenosu podataka, vrijednost prvog bita ACK polja će ostati nula te će odbaciti poruku i zatražit će se ponovno slanje poruke. Drugi bit ACK polja DEL, odvaja ACK polje od polja koje slijedi.
- EOF (engl. *End Of Frame*) – polje koje određuje završetak slanja podatkovnog okvira .
- IFS (engl. *Interframe Space*) – njegova vrijednost predstavlja vrijeme koje je potrebno da se primljena poruka premjesti u podatkovni međuspremnik na odgovarajuću poziciju. IFS nije dio CAN poruke.



Sl. 2.4. *Prikaz poruke i duljina svakog dijela poruke u bitovima.*

Udaljeni okvir (engl. *Remote frame*) ima iste dijelove poruke kao podatkovni okvir, osim podatkovnog bloka. On ne postoji kao dio poruke u udaljenom okviru. Ovaj okvir se koristi za zahtijevanje podataka od CAN čvorova. Ovaj okvir se jako rijetko koristi u komunikaciji unutar automobila zbog toga što se prijenos podataka ne zasniva na zahtjevima.

Tablica 2.1. *Vrijednosti DLC-a i broj pripadajućih bajtova u podatkovnom bloku*

DLC	BAJT 1	BAJT 2	BAJT 3	BAJT 4	BAJT 5	BAJT 6	BAJT 7	BAJT 8
0000								
0001								
0010								
0011								
0100								
0101								
0110								
0111								
1000								

2.2 Stvarno-vremenski zahtjevi i opterećenje sabirnice

Commented [M1]: Nisi boldao – pregledaj cijeli rad ovaj problem

Kako bi se osigurao rad u stvarnom vremenu uz potreban brzi i deterministički odziv, preporučeni maksimum vremenske zauzetosti sabirnice je 20% do 30% [18]. Postotak zauzetosti sabirnice je temeljni parametar svake komunikacije, a predstavlja omjer vremena u kojemu je sabirnica u stanju zauzetosti od strane jednog ili više čvorova i ukupnog promatranog vremena. Omjer zauzetosti sabirnice od strane jednog čvora predstavljen je matematičkim izrazom:

$$\eta_{jedan\ čvor} = \frac{\sum_{m=1}^{k} T_m}{T_{ukupno}} \tag{2-1}$$

gdje je:

- T_m - vrijeme zauzetosti sabirnice od strane jedne poruke
- k – broj poruka koje se šalju u promatranom vremenu
- T_{ukupno} – ukupno promatrano vrijeme komunikacije na sabirnici (zbroj vremena zauzetosti sabirnice i dostupnosti sabirnice)

Ako postoji više čvorova u mreži, onda se omjer zauzetosti sabirnice od strane svih čvorova računa kao:

$$\eta_{ukupno} = \sum_{m=1}^{n} \eta_{jedan\ čvor,m} \tag{2-2}$$

gdje je:

- n – broj čvorova u mreži

2.3 CAN komunikacija u automotiv podatkovnom prometu i postojeća rješenja na tržištu

U radu [2] autori opisuju potrebu implementacije različitih komunikacijskih protokola u automobilima. Predstavljeni su neki od najkorištenijih protokola kao CAN, LIN, FlexRay, D2B i MOST. Opisan je svaki navedeni protokol sa svojim prednostima i nedostatcima. Zaključak navedenog rada je da je CAN protokol i dan danas najpouzdaniji protokol, dok je njegova vremenska zauzetost sabirnice ispod 30%. Prema [2] FlexRay protokol je također protokol koji obećava, ali će trebati proći još vremena za njegovu standardizaciju i ozbiljniju komercijalnu

implementaciju. Učinkovito korištenje CAN protokola i njegovih prednosti jedino je rješenje dok se ne standardizira neki bolji, robusniji i sigurniji protokoli [2].

U radu [19] autori opisuju dizajn CAN protokola prema ISO/OSI modelu. Rad je napisan na osnovu testiranja mreže čija se komunikacija temelji na CAN komunikacijskom protokolu. Prikazani su primjeri arbitraže poruka na sabirnici. Eksperimentalno kreirana mreža sastoji se od osam CAN čvorova koji su spojeni na zajedničku sabirnicu i međusobno komuniciraju. Čvorovi si međusobno šalju poruke preko sabirnice i mjeri se vrijeme u kojemu je sabirnica u stanju zauzetosti. Nakon završetka eksperimenta zaključeno je da je omjer zauzetosti sabirnice 28%, što spada unutar intervala između 20% i 30% opisanog u prethodnom poglavlju. Dodatno je korišten CANoe programski alat za uspoređivanje pojavljivanja poruka na sabirnici. Zaključak je da je vrijeme kašnjenja signala vrlo malo što je potvrda da algoritam raspoređivanja radi dobro i da je sustav vrlo pouzdan[19]. Mana ovog rješenja je ta što kreirana mreža nije zaštičena od vanjskih faktora koji utječu na komunikaciju, nego je ona izložena okruženju.

U radu [20] autori opisuju način komunikacije u mreži zasnovan na CAN protokolu, s posebnim naglaskom na detekciju kolizije. Opisan je princip komunikacije u mreži zasnovanoj na CAN komunikacijskom protokolu. Zaključak je da CAN protokol pruža efektivni prijenos podataka između različitih čvorova. CAN je fleksibilan, pouzdan, robustan i standardiziran protokol koji može ispuniti stvarno-vremenske zahtjeve u komunikaciji u automobilima. Velika prednost CAN protokola je ta što kvar jednog CAN čvora ne može srušiti cijelu mrežu, nego se neispravan čvor isključuje iz mreže[20].

Najveći trenutni problem je sve veća potreba za što bržim odzivom na poslanu poruku zbog sve većeg broja poruka koje se šalju preko sabirnice, a vrijeme predstavlja kritičan parametar u većini situacija. Količina podataka koju treba prenijeti i obraditi povećava se iz dana u dan zbog sve većeg broja senzora i ostalih uređaja koji polako zamjenjuju vozačeve uloge u automobilima. Postavljaju se vremenski intervali u kojima poruka mora biti pročitana nakon što je poslana i to vrijeme odziva mjeri se u milisekundama.

Prvi zadatak ovog diplomskog rada bio je upoznati se s rješenjima koja postoje na tržištu. Nakon toga, od strane tvrtke Institut RT-RK Osijek, u suradnji s kojom je i izrađen oaj diplomski rad, dobiveno je postojeće rješenje za filtriranje i klasifikaciju poruka u mreži zasnovanoj na CAN protokolu. Ono predstavlja algoritam koji obrađuje poruke koje su poslane u CAN mreži sa četiri CAN kanala. Korištenjem četiri CAN kanala može se napraviti filtriranje i klasifikacija

podatkovnog prometa za 16 vrsta usmjeravanja, a trenutno rješenje tvrtke RT-RK u sebi sadrži samo četiri vrste usmjeravanja. Vrsta usmjeravanja opisuje s kojeg CAN kanala se šalje poruka kojem CAN kanalu. U tablici 2.2 zelenom bojom prikazane su četiri vrste usmjeravanja koje su podržane u rješenju tvrtke Institut RT-RK. Imena stupaca predstavljaju kanale koji šalju poruku, a imena redova u tablici predstavljaju kanale koji primaju poruku. Polja označena bijelom bojom predstavljaju vrste usmjeravanja koja nisu obuhvaćena rješenjem tvrtke RT-RK. Upravo ta bijela polja predstavljaju nedostatak rješenja tvrtke RT-RK.

Tablica 2.2. *Sve moguće vrste usmjeravanja u mreži sa 4 CAN kanala i vrste usmjeravanja podržane u postojećem rješenju tvrtke Institut RT-RK(označene zelenom bojom)*

CAN KANAL	CAN 1 RX	CAN 2 RX	CAN 3 RX	CAN 4 RX
CAN 1 TX				
CAN 2 TX				
CAN 3 TX				
CAN 4 TX				

U poglavlju 3. objašnjen je novo-kreirani algoritam za filtriranje, klasifikaciju podatkovnog prometa i detekciju pogrešaka i izgubljenih poruka u automobilima. Prilikom izrade programskog rješenja prethodno proučeni radovi i posotjeća rješenja na tržištu pomogli su u izradi samog rješenja i shvaćanju principa komunikacije u mrežama zasnovanim na CAN komunikacijskom protokolu. Također predstavljeni su zahtjevi koji moraju biti zadovoljeni kako bi kreirano programsko rješenje bilo funkcionalno. Kreirano rješenje daje uvid u komunikaciju u mreži zasnovanoj na CAN protokolu koji može služiti kao vodič za daljnje unaprjeđenje same mreže.

3. RJEŠENJE ZA FILTRIRANJE PODATKOVNOG PROMETA I DETEKCIJU POGREŠAKA I IZGUBLJENIH PORUKA PRI KOMUNIKACIJI ZASNOVANOJ NA CAN PROTOKOLU

Sukladno proučenim radovima i postojećim rješenjima na tržištu koja se bave CAN protokolom i podatkovnim prometom u mreži zasnovanoj na CAN komunikacijskom protokolu, u suradnji s tvrtkom prnerom u izradi diplomskog rada kreirani su zahtjevi koje mora ispuniti novo rješenje za filtriranje, klasifikaciju podatkovnog prometa i detekciju pogrešaka i izgubljenih poruka u automobilima. Svi zahtjevi moraju biti ispunjeni kako bi rješenje bilo funkcionalno i mora biti provedena verifikacija svakog zahtjeva. Zahtjevi su kako slijedi:

1. ispravno parsiranje podatkovnih okvira iz .asc datoteke
2. prepoznavanje jedinstvenih poruka koje se pojavljuju više puta na istom kanalu i kreiranje jedinstvenih objekata koji predstavljaju jednu jedinstvenu poruku
3. prepoznavanje ispravnosti podatkovnog bloka
4. prepoznavanje neusmjerenog ulaza
5. prepoznavanje poruke koja nije zadovoljila vrijeme odziva od 7.5 milisekundi
6. kreiranje izvještaja za sve jedinstvene poruke, neusmjerene poruke i grafički prikaz svih 16 vrsta usmjeravanja opisanih u tablici 2.2
7. brzina izvođenja programa je približno jednaka za log određene veličine
8. obuhvatiti svih 16 mogućih vrsta usmjeravanja

U izradi programskog rješenja korišten je Python kao skriptni jezik za izradu skripte koja obrađuje podatke iz .asc log-a. Uz Python korišten je *xlsxwriter* modul [21] u Python-u, koji omogućava manipulaciju Excel datotekom u kojoj je pohranjen izvještaj o rezultatima analize .asc datoteke. Korišten je također CANoe [22] za kreiranje logova, a TTX Connexion Configurator je korišten za kreiranje tablice usmjeravanja.

3.1 Praćenje podatkovnog prometa u komunikaciji unutar automobila

Prvi korak u izradi vlastitog rješenja bio je kreirati .asc log s porukama u CAN komunikaciji, kako bi se moglo vidjeti gdje se nalaze ključne vrijednosti u log-u [23]. Log se kreira tako da se u CANoe okruženju kreira mreža zasnovana na CAN protokolu i dodaje se pripadajuća baza podataka poruka koje će se koristiti u komunikaciji. Dijelovi podatkovne poruke u .asc datoteci su (Sl. 3.1):

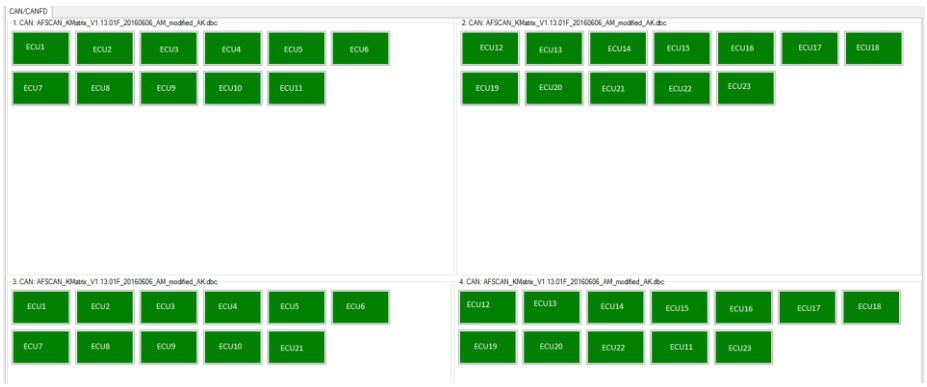
- 1. element: Vremenska oznaka kada je poslana/primljena oznaka (ovisno o vrijednosti 4. elementa) – npr. na slici 3.1 je ta vrijednost 0.100578
- 2. element: Kanal s kojega je poslana poruka (ovisno o vrijednosti 4. elementa) - npr. na slici 3.1 je ta vrijednost 3
- 3. element: Simbolično ime poslane poruke (npr. na slici 3.1 ta vrijednost je „Airbag_01“)
- 4. element: Informacija o tome da li je poruke poslana ili primljena (TX/RX). U ovom slučaju na slici 3.1 vrijednost 4. elementa je TX što znači da se radi o poslanoj poruci)
- 5. element: Oznaka okvira koji se šalje/prima (npr. na slici 3.1 oznaka je vrijednosti „d“ što znači da se prenosi podatkovni okvir)
- 6. element: Broj bajtova podatkovnog bloka. Ovisno o njegovoj vrijednosti, toliko sljedećih elemenata predstavljaju dijelove podatkovnog bloka (npr. na slici 3.1 vrijednost 6. elementa je 8. Svaki element podatkovnog bloka ima vrijednost 8 bajtova, što znači da će podatkovni blok imati 8 elemenata odnosno prenosit će 64 bajta)
- 7. element – 14. element: Predstavljaju podatkovni blok čija je veličina jednaka umnošku vrijednosti šestog elementa i broja 8 (8 bajtova) (npr. na slici 3.1 vrijednost 7.elementa do 14.elementa je 0 128 0 5 0 0 0 254)
- 15. element – Predstavlja najavu dolaska trajanja poruke u nanosekundama. Njegova vrijednost je uvijek „Length“
- 16. element – Znak jednakosti prije broja trajanja poruke u nanosekundama.
- 17. element: Predstavlja trajanje poruke u nanosekundama (npr. na slici 3.1 vrijednost 17. elementa je 242259)
- 18. element: Predstavlja najavu ukupnog broj bitova u poruci uključujući EOF i međuokvirni prostor. Vrijednost 18. elementa je uvijek „BitCount“
- 19. element: Znak jednakosti prije elementa koji pokazuje broj bitova u poruci
- 20. element: Predstavlja broj bitova koji se prenose u poruci uključujući EOF i međuokvirni prostor (npr. na slici 3.1 broj bitova koji se prenose je 124)
- 21. element: Predstavlja najavu numeričke oznake identifikatora poruke. Vrijednost 21. elementa je uvijek „ID“
- 22. element: Znak jednakosti prije elementa čija vrijednost određuje identifikator poruke
- 23. element: Numerička vrijednost identifikatora poruke (npr. na slici 3.1 njegova vrijednost je 64.)

```
0.100578 3  Airbag_01    Tx   d 8   0 128   0   5   0   0   0 254  Length = 242259 BitCount = 124 ID = 64
```

Slika 3.1. *Primjer podatkovne poruke u .asc datoteci*

3.2 Scenarij usmjeravanja i TTX Connexion Configurator

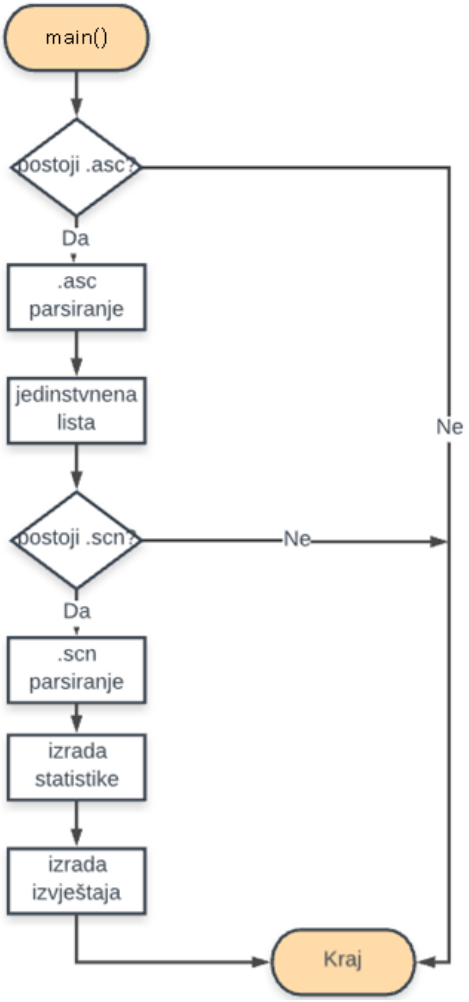
TTX Connexion Configurator koristi se za kreiranje tablice usmjeravanja koja opisuje sva pravila usmjeravanja poruka u mreži. Kreirana tablica usmjeravanja definira na kojem će kanalu izlaziti koji paket, odnosno s kojeg kanala će se poslati neki paket. Također, za kreiranje scenarija usmjeravanja potrebna je baza podataka poruka, a ta baza podataka mora biti ista kao baza podataka korištena u CANoe kada se kreira .asc log. Dodavanje ECU komponenti CAN kanalima vrši se tako što se iz baze podataka metodom „*drag-and-drop*“ postavljaju ECU-ovi na željene kanale (Sl. 3.2). Nakon toga se generira lista usmjerenih poruka te .scn datoteka koja se kasnije koristi u Python skripti, odnosno ona se parsira.



Sl. 3.2. *TTX Connexion Configurator sučelje za dodavanje ECU-ova*

3.3 Programsko rješenje algoritma za filtriranje, klasifikaciju podatkovnog prometa i detekiju pogrešaka i izgubljenih poruka

Programsko rješenje za filtriranje, klasifikaciju podatkovnog prometa i detekciju pogrešaka i izgubljenih poruka realizirano je u Python skriptnom jeziku, a korišten je PyCharm kao razvojno okruženje. Dijagram toka napravljenog programskog rješenja dan je na slici 3.3.



Slika. 3.3. *Dijagram toka kreiranog programskog rješenja za filtriranje, klasifikaciju podatkovnog prometa i detekciju pogrešaka i izgubljenih poruka*

Iz Python programskog jezika uvezena su četiri modula koja otklanjaju potrebu izrade vlastitih modula. Modul *xml.etree.ElementTree* se koristi za čitanje i parsiranje .scn datoteke koja predstavlja tablicu usmjeravanja. Modul *timeit* se koristi kao *timer* za računanje vremena izvršenja programa, dok se modul *datetime* koristi za dohvaćanje trenutnog datuma, sata, minute i sekunde

kada je kreirano izvješće. *Xlsxwriter* modul se koristi za upisivanje rezultata obrade .log datoteke u Excel datoteku. Implementacija modula prikazana je na slici 3.4.

```
54
55  import xml.etree.ElementTree as ET
56  from timeit import default_timer as timer
57  import xlsxwriter
58  import datetime
```

Slika 3.4. *Moduli implementirani u programsko rješenje*

3.3.1 Parsiranje .asc datoteke

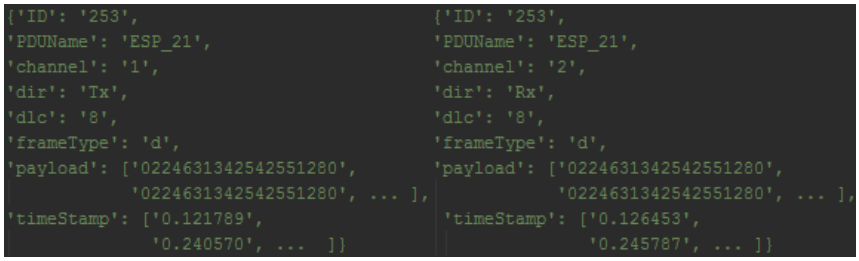
Parsiranje .asc datoteke obavlja tako da se čita cijeli .asc log red po red sve do kraja. Svaka poruka se učitava u niz tako da se varijable iz pročitanog reda spremaju u niz, a uvjet upisivanja je taj da vrijednost četvrtog elementa mora biti "d", što predstavlja podatkovni okvir. Nakon toga pročitana se podatkovna poruka sprema u listu s objektima svih podatkovnih poruka. U tom trenutku omogućeno je kreiranje jedinstvenih lista poslanih, odnosno primljenih poruka s pripadajućim vremenskim oznakama i podatkovnim okvirom.

3.3.2 Kreiranje jedinstvenih vrijednosti poslanih i primljenih poruka

Nakon isparsirane .asc datoteke i očitavanja samo podatkovnih okvira iz liste objekata s porukama, potrebno je kreirati jedinstvene objekte za poslane i primljene poruke. Jedan jedinstveni objekt u sebi sadrži:

- Identifikator poruke,
- Simbolično ime poruke,
- Kanal s kojeg je poruka poslana, ako se radi od TX usmjeravanju, odnosno kanal koji je primio poruku ako se radi o RX usmjeravanju,
- Podatak o broju bajtova koji se prenose,
- Tip okvira,
- Vrijednosti svih podatkovnih blokova koji su preneseni u cijelom .asc log-u s istim: ID, smjerom usmjeravanja i brojem kanala,
- Vrijednosti svih vremenskih oznaka koje su prenesene u cijelom .asc log-u s istim: ID, smjerom usmjeravanja i brojem kanala

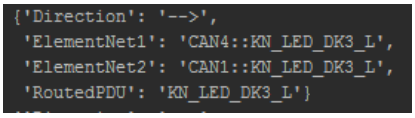
Na slici 3.5. prikazan je izgled jedinstvenog objekta s njegovim pripadajućim parom na strani kanala koji prima poruku. Svaki jedinstveni objekt ima svog para. Na slici 3.5. se prikazan je par jedinstvenog objekta. Par jedinstvenog objekta ima iste vrijednosti, osim oznake primatelja/pošiljatelja i vrijednosti vremenskih oznaka. U ovom slučaju radi se o usmjeravanju poruke s imenom „ESP_21“ s kanala 1 na kanal 2. Lijevi dio slike je jedinstveni objekt koji se šalje s kanala 1, a desni dio slike je jedinstveni objekt koji prima poslanu poruku na kanalu 2. Par jedinstvenih objekata služi za izradu jednog reda izvještaja o uspješnosti usmjeravanja.



Slika 3.5. *Primjer ispisa jednog para elementa jedinstvene liste*

3.3.3 Parsiranje tablice usmjeravanja

Tablica usmjeravanja opisuje usmjeravanje podatkovnog prometa u mreži, odnosno koji čvor šalje koju poruku te koji čvor prima koju poruku. Ona je kreirana u TTX Connexion Configurator-u. Ona ima zapis kao XML datoteka, i stoga se parsira pomoću modula *xml.etree.ElementTree*. Scn datoteka čita se red po red i traži se dijete u stablu s pripadajućom oznakom vrijednosti *<EqualFrames>potrebni podaci</EqualFrames>*. Kada je očitana odgovarajuća oznaka, u prethodno kreirani objekt spremaju se sljedeće vrijednosti (Sl. 3.6): smjer, prvi element u mreži, drugi element u mreži i ime poruke. Ako je vrijednost smjera '→' onda je '*ElementNet1*' pošiljatelj, a '*ElementNet2*' primatelj poruke. Ako je vrijednost smjera '←' onda je '*ElementNet2*' pošiljatelj poruke, a '*ElementNet1*' primatelj poruke. Na slici 3.6. '*ElementNet1*' je pošiljatelj poruke, a '*ElementNet2*' primatelj poruke. Tablica usmjeravanja u .scn datoteci sadrži također podatke o tome kojemu ECU je pridruženo koje ime. Kreirani objekti tablice usmjeravanja koriste se za najvažniji dio programa, a to je izrada statistike usmjeravanja.



Slika 3.6. *Izgled isparsiranog elementa iz tablice usmjeravanja*

3.3.4 Izrada statistike i izvještaja usmjeravanja

Sve prethodno napravljeno u programskom rješenju, odnosno sve što je objašnjeno u dijelovima 3.3.1, 3.3.2 i 3.3.3 rađeno je s namjerom da omogući lakšu izradu statistike usmjeravanja. Statistika usmjeravanja obrađuje ulazne jedinstvene objekte, koji dolaze u paru (Sl. 3.5.) tako da prvo koristi objekte iz tablice usmjeravanja i ako postoji u jedinstvenim objektima podudarajuće usmjeravanje kao i u tablici usmjeravanja, kreće se s izradom statistike za taj objekt. Ako neki ulaz nije opisan u tablici usmjeravanja, on će se u izvještaju prikazati kao ulaz koji nije nigdje usmjeren, Ako je ulaz opisan, započinje se s uspoređivanjem vremenskih oznaka i podatkovnih blokova. Prethodno kreirani jedinstveni objekti za svaku poruku i za njoj pripadajući kanal ovisno o vrijednosti TX ili RX, u sebi sadrže sve vremenske oznake koje pokazuju kako se redom pojavljivala ta poruka i one su odgovarajuće sa svojim pozicijama u listama tako da se njihova vremena mogu oduzimati i mjeriti kašnjenje. Svaka vrijednost u RX listi vremenskih oznaka mora imati veću vrijednost nego vrijednost pripadajuće vremenske oznake u TX listi na istom indeksu (Sl. 3.7.). Ako bi element iz RX liste imao veću vrijednost nego element na istoj poziciji u TX listi to bi značilo da je došlo do greške u komunikaciji, odnosno da je jedna poruka izgubljena.

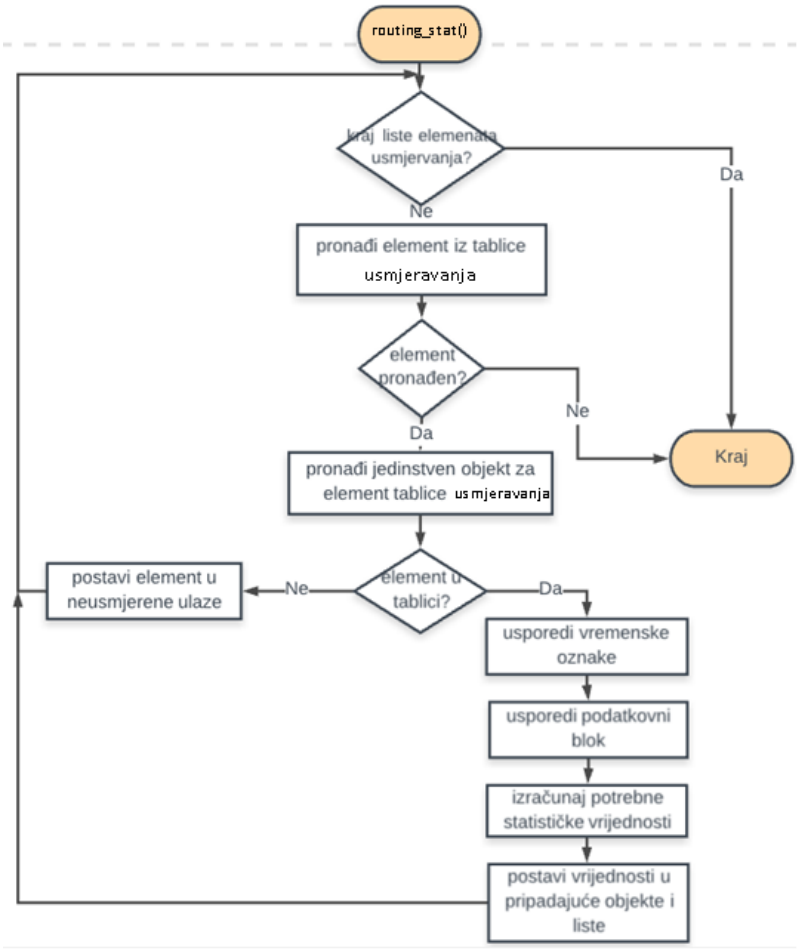
Tx lista:		Rx lista:
['0.567575',	→	['0.570672',
'0.600573',	→	'0.605978',
'1.134966',	→	'1.141279',
'1.201315',	→	'1.707754',
'1.701512',	→	'2.271276',
'1.801245',	→	'2.836230',
'2.268902',	→	'3.407961',
'2.401059',	→	'3.976524',
'2.835443']	→	'4.541703']

Slika 3.7. *Prikaz RX i TX vremenske liste za jedno odgovarajuće usmjeravanje*.

Ako je poruka očitana unutar 7.5 milisekundi, pristupa se provjeravanju ispravnosti podatkovnog bloka, na isti način kako se uspoređuju vremenske oznake. Nužno je da podatkovni blokovi s istim indeksima imaju istu vrijednost. Nakon usporedbe podatkovnih blokova i potvrde da je podatkovni blok ispravan, kreira se izračun za prikazivanje statistike. Vrijednosti koje se računaju u statističkoj obradi za jedno usmjeravanje poruka su:

- Broj detektiranih okvira na TX kanalu,
- Broj detektiranih okvira na RX kanalu,
- Razlika broja poslanih i broja primljenih okvira,
- Postotak uspješno prenesenih okvira s jednog kanala na drugi,
- Broj ispravno podatkovnih blokova,
- Broj neispravno primljenih podatkovnih blokova,
- Minimalno vrijeme kašnjenja,
- Srednje vrijeme kašnjenja,
- Maksimalno vrijeme kašnjenja.

Sve prethodno navedene vrijednosti za jednu vrstu usmjeravanja spremaju se u objekt i taj se objekt prosljeđuje modulu za realizaciju izvještaja. Na slici 3.8 je prikazan dijagram toka modula za izradu statistike i izvještaja usmjeravanja.



Slika 3.8. *Prikaz dijagrama tijeka modula za izradu statistike.*

3.3.4 Izvještaj o provedenoj komunikaciji

Izvještaj o provedenoj komunikaciji kreira se kao zadnji dio cijelog programskog rješenja. On se kreira pomoću *xlsxwriter* modula u Python-u. Modul za kreiranje izvještaja obrađuje dva ulazna skupa podataka. Jedan ulazni skup podataka su podaci o uspješno provedenom usmjeravanju, a drugi skup podataka je skup o neuspješno provedenom usmjeravanju. Izvještaj se sprema u .xlsx datoteku, a on se sastoji od tri dijela:

- Statistička tablica o uspješno provedenom usmjeravanju,
- Popis neusmjerenih poruka, odnosno ulaza koji nisu usmjereni,
- Grafički prikaz postotka uspješnosti usmjeravanja za svaku vrstu usmjeravanja.

Svakom izvještaju se automatski generira ime tako što se na riječ "Test" nadovezuju trenutni mjesec, dan, sat i minuta kada je program pokrenut (npr. Test_06-04 12-55-52). Statistička tablica sastoji se od 15 stupaca, a to su redom:

1. Ime poruke (engl. *PDU name*),
2. Identifikator,
3. Kanal s kojega se šalje poruka,
4. Broj detektiranih okvira na kanalu pošiljatelja (TX). Taj broj se dobije tako da se pomoću funkcije *list.length()* dohvati broj vremenskih oznaka za određenu poruku iz jedinstvenog objekta.
5. Smjer usmjeravanja,
6. Broj detektiranih okvira na kanalu primatelja (RX). Taj broj se dobije kao broj detektiranih poruka na kanalu pošiljatelja, samo što ovdje jedinstveni objekt ima vrijednost RX umjesto TX.
7. Kanal koji prima poruku,
8. Razlika između broja poslanih poruka i broja primljenih poruka. Ova vrijednost se dobije tako da se od broja detektiranih okvira na kanalu pošiljatelja oduzme broj detektiranih okvira na kanalu primatelja (Npr. s kanala 1 se slala poruka s imenom „Airbag_01“ deset puta, a na kanalu 2 se primala ta poruka i ona je detektirana 9 puta. U ovom slučaju vrijednost razlike će biti 1)
9. Postotak uspješnosti usmjeravanja. Postotak uspješnosti je predstavljen omjerom uspješno prenesenih poruka unutar 7.5 milisekundi i ukupno prenesenih poruka.
10. Broj ispravno primljenih podatkovnih blokova,
11. Broj neispravno primljenih podatkovnih blokova,
12. Minimalno kašnjenje [milisekunde]. Predstavlja najmanju vrijednost koja se dobije oduzimanjem elemenata iz liste vremenskih oznaka s istim indeksima za svaki odgovarajući par jedinstvenog objekta.
13. Maksimalno kašnjenje [milisekunde]. Predstavlja najveću vrijednost koja se dobije oduzimanjem elemenata iz liste vremenskih oznaka s istim indeksima za svaki odgovarajući par jedinstvenog objekta.

14. Prosječno kašnjenje [milisekunde]. Računa se tako da se sumiraju sve vrijednosti koje se dobiju oduzimanjem elemenata iz liste vremenskih oznaka s istim indeksima za svaki odgovarajući par jedinstvenog objekta. Nakon toga prosječno kašnjenje se dobije tako da se podijeli suma svih vremena s brojem vremenskih oznaka u jedinstvenom objektu.
15. Status o uspješnosti (USPJEŠNO / NEUSPJEŠNO)

Slika 3.9. predstavlja izvještaj koji je izgeneriran pomoću modula za računanje statistike usmjeravanja.

Redni broj stupca: 1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.
CAN1 --> CAN1														
AFS_Feh_LM_L	924	1	201	CAN1 --> CAN1	201	1	0	100	201	0	1,477	6,067	4,067	PASS
AFS_Feh_LM_R	926	1	201	CAN1 --> CAN1	201	1	0	100	201	0	1,145	6,485	4,038	PASS
AFS_Idt_LM_L	1172	1	310	CAN1 --> CAN1	310	1	0	100	310	0	1,221	6,887	4,243	PASS
AFS_Idt_LM_R	1174	1	321	CAN1 --> CAN1	321	1	0	100	321	0	0,496	7,157	4,17	PASS
AFS_Ken_LM_L	1176	1	201	CAN1 --> CAN1	201	1	0	100	201	0	1,141	7,379	4,53	PASS
AFS_Ken_LM_R	1178	1	201	CAN1 --> CAN1	199	1	2	99,005	199	0	1,398	7,146	4,623	FAIL
CAN1 --> CAN2														
AFS_Soll_Fkt	402	1	201	CAN1 --> CAN2	201	2	0	100	201	0	1,066	6,449	4,316	PASS
AFS_Sta_LM_L	410	1	201	CAN1 --> CAN2	200	2	1	99,5025	200	0	1,095	7,209	4,351	PASS
AFS_Sta_LM_R	412	1	201	CAN1 --> CAN2	200	2	1	99,5025	200	0	0,854	7,286	4,477	PASS
Airbag_01	64	1	201	CAN1 --> CAN2	201	2	0	100	201	0	1,235	6,301	4,315	PASS
BCM1_Soll_Fkt1	649497	1	201	CAN1 --> CAN2	195	2	6	97,0149	195	0	1,479	6,946	4,496	FAIL
ESP_21	253	1	201	CAN1 --> CAN2	201	2	0	100	201	0	1,106	6,46	4,257	PASS
GLW_Soll_Fkt	272	1	100	CAN1 --> CAN2	100	2	0	100	100	0	1,096	6,271	4,521	PASS
)_Funktionaler_Req_	1792	1	100	CAN1 --> CAN2	90	2	10	90	90	0	0,867	7,348	4,61	FAIL
CAN1 --> CAN3														
Kamera_FDD_Obj01	791	1	95	CAN1 --> CAN3	69	3	26	72,6316	69	0	0,896	6,378	2,981	FAIL
Kamera_FDD_Obj02	799	1	96	CAN1 --> CAN3	96	3	0	100	96	0	1,196	6,401	3,981	PASS
Kamera_FDD_Obj03	838	1	96	CAN1 --> CAN3	96	3	0	100	96	0	1,084	6,64	3,916	PASS
Kamera_FDD_Obj04	839	1	96	CAN1 --> CAN3	96	3	0	100	96	0	1,017	6,394	4,128	PASS
Kamera_FDD_Obj05	840	1	109	CAN1 --> CAN3	109	3	0	100	109	0	1,115	6,253	4,062	PASS
Kamera_FDD_Obj06	841	1	109	CAN1 --> CAN3	109	3	0	100	109	0	0,883	5,974	3,914	PASS
Kamera_FDD_Obj07	842	1	109	CAN1 --> CAN3	109	3	0	100	109	0	1,106	7,047	3,857	PASS
Kamera_FDD_Obj08	843	1	109	CAN1 --> CAN3	109	3	0	100	109	0	1,254	6,097	4,001	PASS

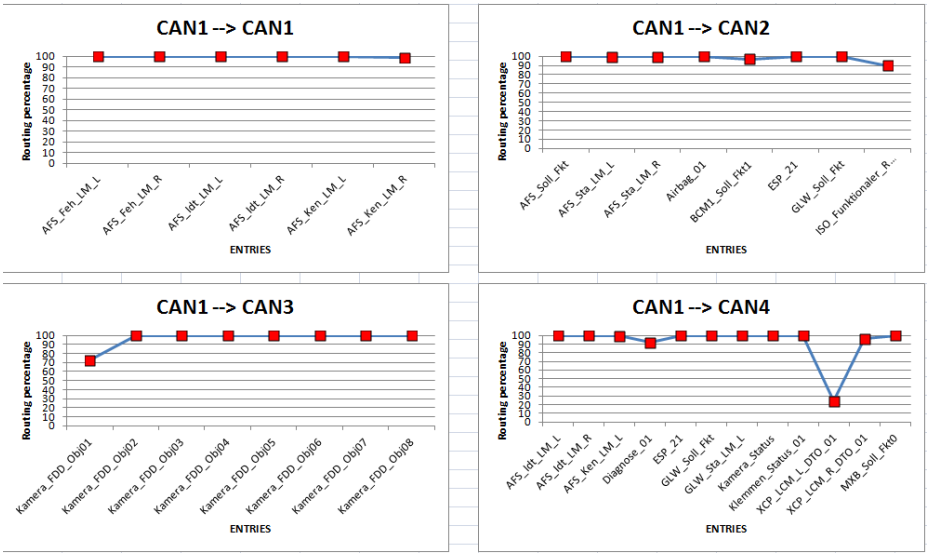
Slika 3.9. *Prikaz statističke tablice u izvještaju* .

Izvještaj o neusmjerenim porukama sastoji se od jednog stupca (Sl. 3.10). Neusmjerena je ona poruka čije usmjeravanje je opisano u tablici usmjeravanja, a to usmjeravanje se ne pojavljuje u asc log-u.

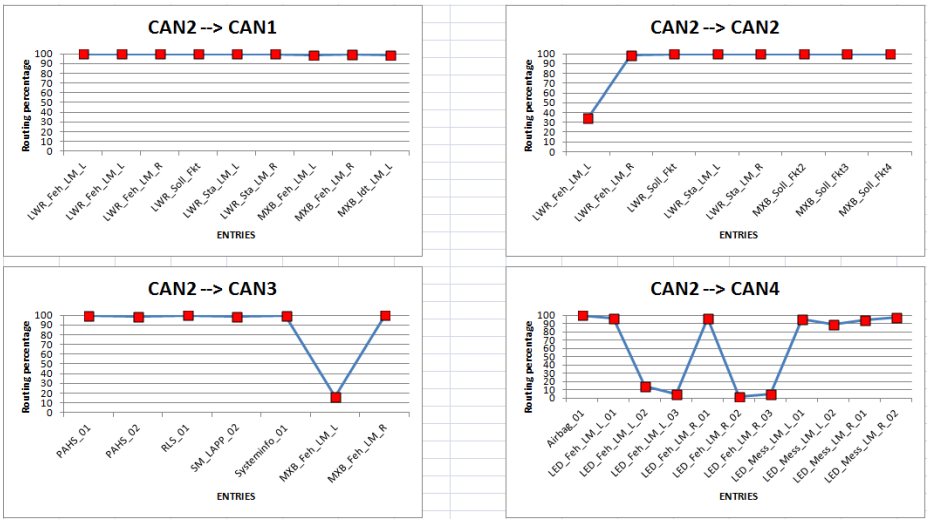
Routing entries:
CAN1::AFS_Feh_LM_R --> CAN1::AFS_Feh_LM_R
CAN1::AFS_Ken_LM_L --> CAN1::AFS_Ken_LM_L
CAN1::AFS_Sta_LM_L --> CAN2::AFS_Sta_LM_L
CAN1::ESP_21 --> CAN2::ESP_21
CAN2::MXB_Feh_LM_L --> CAN1::MXB_Feh_LM_L
CAN1::AFS_Idt_LM_L --> CAN4::AFS_Idt_LM_L
CAN2::SM_LAPP_02 --> CAN3::SM_LAPP_02
CAN2::Systeminfo_01 --> CAN3::Systeminfo_01
CAN2::MXB_Feh_LM_L --> CAN3::MXB_Feh_LM_L
CAN3::LED_Soll_Fkt1 --> CAN4::LED_Soll_Fkt1
CAN3::LED_Soll_Fkt2 --> CAN4::LED_Soll_Fkt2
CAN4::ESP_21 --> CAN1::ESP_21
CAN4::KN_LED_DK3_R --> CAN1::KN_LED_DK3_R

Slika 3.10. *Ispis ulaza koji su neusmjereni*

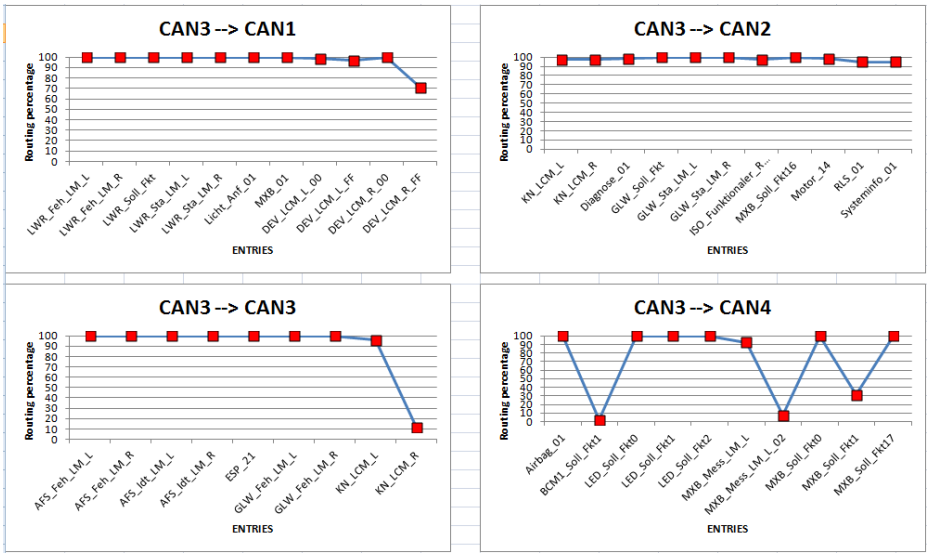
Grafički izvještaj sastoji se od 16 grafova koji prikazuju postotak uspješnosti usmjeravanja svake poruke (Sl. 3.11, Sl. 3.12., Sl. 3.13., Sl 3.14.). Ti grafovi su također napravljeni pomoću *xslxwriter-a.*



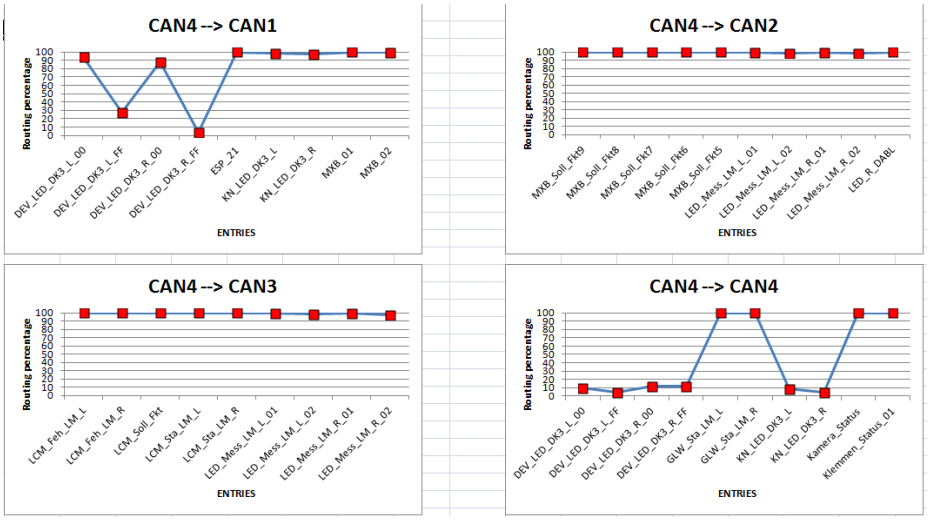
Slika 3.11. Primjer *grafičkog prikaza postotka uspješnosti usmjeravanja gdje se poruke šalju s kanala 1.*



Slika 3.12. Primjer *grafičkog prikaza postotka uspješnosti usmjeravanja gdje se poruke šalju s kanala 2.*



Slika 3.13. Primjer *grafičkog prikaza postotka uspješnosti usmjeravanja gdje se poruke šalju s kanala 3*



Slika 3.14. Primjer *grafičkog prikaza postotka uspješnosti usmjeravanja gdje se poruke šalju s kanala 4*

Nakon izrade predstavljenog programskog rješenja za filtriranje, klasifikaciju podatkovnog prometa i detekciju pogrešaka i izgubljenih poruka, pristupilo se provjeri ispravnosti njegova rada, što je detaljnije opisano u sljedećem poglavlju.

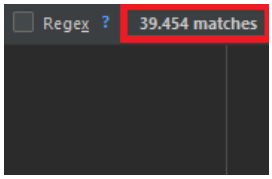
4. VERIFIKACIJA ISPRAVNOSTI RADA KREIRANOG ALGORITMA

Verifikacija ispravnosti rada je neophodni korak pri provjeri svakog novo-kreiranog programskog rješenja. Potrebno je ispitati jesu li zadovoljeni zahtjevi, regulacije, specifikacije i uvjeti koji su postavljeni prije početka izrade samog rješenja na početku poglavlja 3. Prvi korak verifikacije je iterativno pokretanje programskog rješenja 1000 puta, s kojim se željelo provjeriti da li postoje velike razlike u vremenima izvođenja za log iste veličine (npr. da se jednom izvodi 1600 ms, pa u drugoj iteraciji 10000 ms). Pri izvođenju svake iteracije mjeri se vrijeme izvođenja i konačno se računa srednje vrijeme izvođenja programa. Vrijeme izvođenja programa predstavlja vrijeme koje protekne od početka parsiranja .asc datoteke do završetka kreiranja izvještaja. Rezultat srednjeg vremena izvođenja programa nakon 1000 iteracija je 1639.99 milisekundi, a standardna devijacija izvođenja programa iznosi 39.29 milisekundi (Sl. 4.1). U prvoj verziji rješenja vrijeme izvođenja algoritma je bilo iznad 5000 milisekundi za .asc datoteku veličine 5 MB. Korak koji je bilo potrebno poduzeti za smanjenje vremena izvođenja algoritma je zamjena *for* petlji s ekspresijskim generatorima (*engl. Expression generators*) [24]. Provedenom verifikacijom potvrđeno je da algoritam zadovoljava zahtjev 7. koji je pred njega postavljen na početku 3. poglavlja.

```
1628.3187459999908 ms
1638.2640049999964 ms
1603.7733430000003 ms
1624.122595000017 ms
1629.7856019999983 ms
1636.9813260000114 ms
1646.281966999993 ms
1628.4903769999914 ms
1604.7620809999898 ms
1700.3158939999992 ms
1655.0240910000014 ms
Srednja vrijednost izvođenja programa je:  1639.99404419
Standardna devijacija vremena izvođenja programa je:  39.290798268338044
```

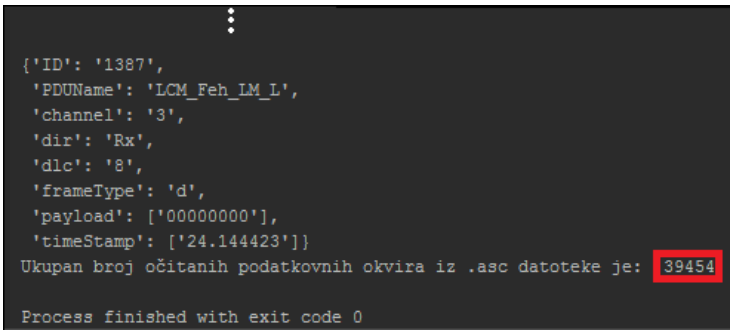
Slika 4.1. *Konačni ispis programa za izračun srednjeg vremena izvođenja programa i standardne devijacije izvođenja programa.*

Drugi korak u sklopu verifikacije bio je provjeravanje ispravnosti modula za parsiranje .asc log-a. Ovaj postupak je napravljen tako da se pomoću naredbe *„CTRL + F“* u PyCharm okruženju pretraživala sekvenca „ d“ (predstavlja podatkovni okvir – četiri razmaka i slovo „d“). Nakon toga PyCharm okruženje ispisuje broj redaka gdje se pojavljuje tražena sekvenca. Broj redaka podatkovnog okvira u .asc logu je 39454 (Sl. 4.2).



Slika 4.2. *Broj redaka podatkovnih okvira koji se pojavljuju u .asc logu*

Nakon dohvaćenog broja redaka podatkovnih okvira u .asc logu prebrojavaju se parsirani objekti iz .asc loga. Broj objekata isparsiranog loga mora biti jednak broju redova podatkovnih okvira prikazanom na slici 4.2. Broj isparsiranih podatkovnih okvira prikazan je na slici 4.3. Usporedbom dobivenih brojeva sa slike 4.2. i da slike 4.3. zaključuje se da su ti brojevi iste vrijednosti i da modul za parsiranje ispravno parsira .asc log. Time je zahtjev 1. koji je postavljen na početku poglavlja 3. zadovoljen.



Slika 4.3. *Ispis broja isparsiranih podatkovnih okvira iz .asc datoteke*

Treći korak u sklopu verifikacije obuhvaća provjeravanje ispravnosti modula za izradu statistike. U ovom koraku ručno se mijenja u .asc logu vrijednost podatkovnog bloka imena „AFS_Feh_LM_L“ kako bi se provjerio dio modula koji je zadužen za usporedbu ispravnosti podatkovnog bloka. Na slici 3.9 za usmjeravanje CAN1 → CAN1, vrijednost neispravno prenesenih podatkovnih blokova za poruku „AFS_Feh_LM_L“ je 0 što znači da su svi podatkovni blokovi ispravno preneseni. U .asc logu su promijenjene vrijednosti tri podatkovna bloka. Na slici 4.4. je prikazan ispis provjere pogrešno prenesenih podatkovnih blokova, a na slici 4.5. je prikazan broj neispravno prenesenih podatkovnih blokova koji je upisan u izvještaj. Vrijednost ispisa

neispravno prenesenih podatkovnih blokova sa slike 4.4. podudara se s brojem neispravno prenesenih blokova označenim plavim pravokutnikom na slici 4.5., što znači da je postavljeni zahtjev 3. na početku poglavlja 3. zadovoljen.

```
C:\Users\lroguljic\venv\Scripts\python.exe "C:/Users/lroguljic/Desktop/CAN DIPL/CAN.py"
CAN1 --> CAN1 usmjeravanje za poruku imena  AFS_Feh_LM_L
Broj neispravno prenesenih podatkovnih blokova nakon verifikacije je: 3
1738.3540979999998 ms

Process finished with exit code 0
```

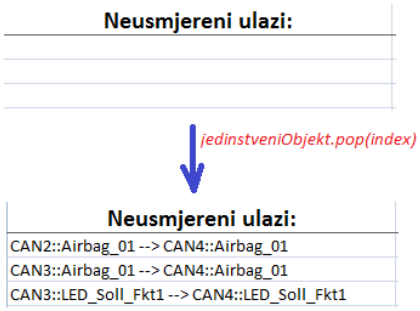
Slika 4.4. *Ispis broja pogrešno prenesenih podatkovnih blokova iz verifikacije za dio modula koji traži pogrešno prenesene podatkovne blokove*

CAN1 --> CAN1									
AFS_Feh_LM_L	1	0	100	198	3	1,477	6,067	4,067	PASS
AFS_Feh_LM_R	1	0	100	201	0	1,145	6,485	4,038	PASS
AFS_Idt_LM_L	1	0	100	310	0	1,221	6,887	4,243	PASS
AFS_Idt_LM_R	1	0	100	321	0	0,496	7,157	4,17	PASS
AFS_Ken_LM_L	1	0	100	201	0	1,141	7,379	4,53	PASS

Slika 4.5. *Izgled izvještaja nakon verifikacije usporedbe vrijednosti podatkovnih blokova.*

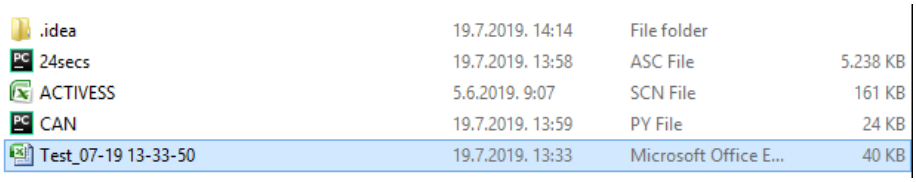
Za provjeru prepoznavanja poruke koja nije zadovoljila vrijeme odziva od 7.5 milisekundi nije potrebna dodatna verifikacija jer je ona implementirana u izradu izvještaja. Ako postoje dvije ili više poruka istog imena i smjera usmjeravanja koje nisu zadovoljile vrijeme odziva od 7.5 milisekundi, u zadnjem stupcu će ta poruka biti označena kao „FAIL“ (Sl. 3.9.). Zahtjev 5. je zadovoljen.

Četvrti korak verifikacije je ispravno prepoznavanje neusmjerenih ulaza. Ovaj postupak verifikacije se radi tako da se iz liste s jedinstvenim objektima pomoću naredbe *jedinstveniObjekt.pop(indeks)* obriše jedinstveni objekt na određenom mjestu u listi. Iz liste jedinstvenih objekata obrisani su objekti s indeksima 0 (Airbag_01), 1 (Airbag_01) i 2 (LED_Soll_Fkt1). Nakon operacije brisanja iz liste jedinstvenih objekata na slici 4.6 može se vidjeti da su prikazani neusmjereni ulazi odgovaraju onim ulazima koji su obrisani iz liste. Ovim korakom verifikacije je potvrđeno je da algoritam zadovoljava zahtjev 4.



Slika 4.6. *Prikaz izvještaja neusmjerenih ulaza prije i poslije verifikacije ispravnosti prepoznavanja neusmjerenih ulaza.*

Verifikacija zahtjeva 6 provodi se nakon pokretanja novo-kreiranog algoritma otvori datoteka u kojoj se nalazi kod algoritma i ako u toj datoteci postoji .xlsx datoteka znači da je zahtjev 6 zadovoljen i da je izvještaj o komunikaciji uspješno kreiran. Na slici 4.7 plavom bojom je označen kreirani izvještaj o komunikaciji.



Slika 4.7. *Prikaz postojanja izvještaja komunikacije nakon izvođenja algoritma*

Unutar kreiranog izvještaja verificira se zahtjev 8 postavljen na početku poglavlja 3. Unutar izvještaja kreiran je prozor za grafički opis komunikacije, ako je kreirano 16 grafova to znači da je zahtjev 8. zadovoljen (slike 3.10, 3.11, 3.12, 3.13). Unutar izvještaja o komunikaciji provjerava se zahtjev 2 postavljen na početku poglavlja 3. Verifikacija se provodi tako da se unutar svake vrste usmjeravanja uspoređuju poruke i ne smije se pojaviti ista poruka dva ili više puta. Na slici 4.8 prikazane su dvije vrste usmjeravanja i ne postoje iste poruke unutar svake vrste usmjeravanja što znači da je zahtjev 2. zadovoljen. Provedena je verifikacija za svaki postavljeni zahtjev i svi zahtjevi su zadovoljeni.

CAN1 --> CAN1														
AFS_Feh_LM_L	924	1	201	CAN1 --> CAN1	201	1	0	100	201	0	1,477	6,07	4,07	PASS
AFS_Feh_LM_R	926	1	201	CAN1 --> CAN1	201	1	0	100	201	0	1,145	6,48	4,04	PASS
AFS_Idt_LM_L	1172	1	310	CAN1 --> CAN1	310	1	0	100	310	0	1,221	6,89	4,24	PASS
AFS_Idt_LM_R	1174	1	321	CAN1 --> CAN1	321	1	0	100	321	0	0,496	7,16	4,17	PASS
AFS_Ken_LM_L	1176	1	201	CAN1 --> CAN1	201	1	0	100	201	0	1,141	7,38	4,53	PASS
AFS_Ken_LM_R	1178	1	201	CAN1 --> CAN1	199	1	2	99	199	0	1,398	7,15	4,62	FAIL
CAN1 --> CAN2														
AFS_Soll_Fkt	402	1	201	CAN1 --> CAN2	201	2	0	100	201	0	1,066	6,45	4,32	PASS
AFS_Sta_LM_L	410	1	201	CAN1 --> CAN2	200	2	1	100	200	0	1,095	7,21	4,35	PASS
AFS_Sta_LM_R	412	1	201	CAN1 --> CAN2	200	2	1	100	200	0	0,854	7,29	4,48	PASS
Airbag_01	64	1	201	CAN1 --> CAN2	201	2	0	100	201	0	1,235	6,3	4,32	PASS
BCM1_Soll_Fkt1	1649497	1	201	CAN1 --> CAN2	195	2	6	97	195	0	1,479	6,95	4,5	FAIL
ESP_21	253	1	201	CAN1 --> CAN2	201	2	0	100	201	0	1,106	6,46	4,26	PASS
GLW_Soll_Fkt	272	1	100	CAN1 --> CAN2	100	2	0	100	100	0	1,096	6,27	4,52	PASS
ISO_Funktionaler_Req_All	1792	1	100	CAN1 --> CAN2	90	2	10	90	90	0	0,867	7,35	4,61	FAIL

Slika 4.8. *Prikaz izvještaja komunikacije za verifikaciju zahtjeva 2 (zahtjev za kreiranje jedinstvenih poruka)*

5. ZAKLJUČAK

U sklopu ovog diplomskog rada bilo je potrebno razviti programsko rješenje koje služi za filtriranje, klasifikaciju podatkovnog prometa i detekciju pogrešaka i izgubljenih poruka u automobilima. Rješenje ispravno parsira podatkovne okrive iz .asc datoteke i kreira jedinstvene objekte, što je i potvrđeno verifikacijom. Ispravno kreira izvještaj o komunikaciji u mreži zasnovanoj na CAN protokolu (ispravno prepoznaje podatkovne blokove, ispravno prepoznaje neusmjerene ulaze, prepoznaje poruke koje nisu zadovoljile vrijeme odziva od 7.5 milisekundi, a vremena izvođenja programa za log iste veličine su približno slična). Samo rješenje kreirano je u Python skriptnom jeziku pomoću PyCharm razvojnog okruženja i omogućava korisniku da sa svojim .asc logom i tablicom usmjeravanja detektira greške u komunikaciji u mreži zasnovanoj na CAN komunikacijskom protokolu. Nakon što algoritam za filtriranje, klasifikaciju podatkovnog prometa i detekciju pogrešaka i izgubljenih poruka obradi podatke iz .asc i .scn datoteka, kreira se izvještaj o uspješnosti usmjeravanja poruka iz .asc loga. Verifikacijom programskog rješenja pokazano je da su zadovoljeni svi zahtjevi postavljeni prije same izrade rješenja, čime je zadatak diplomskog rada u potpunosti ispunjen.

LITERATURA

[1] M. V Rajasekhar, A. K. Jaswal: „*Autonomous vehicles: The future of automobiles“,* IEEE International Transportation Electrification Conference (ITEC), Chennai, India, 2015, str. 1-6

[2] V. Kumar, J. Ramesh: „*Automotive In Vehicle Network Protocols“,* IEEE International Conference on Computer Communication and Informatics, Coimbatore, India, 2014, str. 1-5

[3] X. Yi-Nan, I. G. Jang, Y. E. Kim, J. G. Chung, S.C. Lee: *„Implementation of FlexRay Protocol with An Automotive Application“,* IEEE International SoC Design Conference, Busan, Južna Koreja, 2008, vol. 2, str 25-28

[4] S. Sedhumandhavan, M. Anitha, B. Renkadevi, B. Vinothini: *„Extensible Flexray Communication Methodology for the Application of Advanced Automobile Applications“,* IEEE Second Internationar Conference on Inventive Communication and Computational Technologies, Coimbatore, Indija, 2018, str. 1160 - 1165

[5] Y. Zhou, Y. Zhang, M. Liu, H. Li, F. Gao: *„Design of Vehicle Remote Monitoring System Based on 4G and FlexRay“,* IEEE 18th International Conference on Communication Technology, Chongqing, Kina, 2018, str. 478-482

[6] J. Turjak, R. Grbić, D. Spasojević, M. Kovačević: *„Recovery from Error States During Communication based on CAN and FlexRay“,* Zooming Innovation in Consumer Technologies Conference, Novi Sad, 2019

[7] J.S. Artal, J. Caraballo, R. Dufo: „*CAN/LIN-Bus protocol. Implementation of a low-cost serial communication network“,* IEEE conference on Technologies Applied to Electronics Teaching, Bilbao, Španjolska, 2014, str. 1-8

[8] J. Huang, M. Zhao, Y. Zhou, and C-C. Xing: „*In-Vehicle Networking: Protocols, Challenges, and Solutions“,* IEEE Network, vol. 33, no. 2, 2019., str. 92-98

[9] Broadcom: https://www.broadcom.com/blog/the-case-for-ethernet-in-cars, pristup ostvaren: 16-07.2019.

[10] Vector SAE J1939 documentation: https://assets.vector.com/cms/content/know-how/_application-notes/AN-ION-1-3100_Introduction_to_J1939.pdf, pristup ostvaren: 17.07.2019

[11] Vector E-Learning (CAN) https://elearning.vector.com/mod/page/view.php?id=334, pristup ostvaren: 17.07.2019

[12] ISO11898-1 http://read.pudn.com/downloads209/ebook/986064/ISO%2011898/ISO%2011898-1.pdf, pristup ostvaren: 13.07.2019

[13] ISO 11898-2: https://www.iso.org/standard/67244.html , pristup ostvaren: 13.07.2019.

[14] ISO 11898-3: https://www.sis.se/api/document/preview/907445/ , pristup ostvaren: 13.07.2019.

[15] Standradization of CAN protocol: https://elearning.vector.com/mod/page/view.php?id=335 , pristup ostvaren: 13.07.2019.

[16] CAN Controller Vector E_Learning: https://elearning.vector.com/mod/page/view.php?id=338 , pristup ostvaren: 13.07.2019.

[17] Communication Principle Vector E_Learning: https://elearning.vector.com/mod/page/view.php?id=343 , pristup ostvaren: 18.07.2019

[18] Y. Zhang, L. Hong, W. Jian, D. Jun, L. Duan: *„A CAN-Based Protocol for Reliability Testing of Embedded Software“,* IEEE 2nd International Conference on Future Computer and Communication, Wuhan, Kina, 2010, vol. 2, str. 384-389

[19] Z. Hongwei, J. Hongguang, Z. Yue: *„Design of CAN Bus Application Layer Protocol for Aerocraft Control System“,* IEEE 2nd International Conference on Industrial Mechatronics and Automation, Wuhan, Kina, 2010,vol. 1, str. 164-167

[20] C. Hanxing, T. Jun: *„Research on the Controller Area Network“,* IEEE International Conference on Networking and Digital Society, Guiyang, Guizhou, Kina, 2009, vol. 2, str. 251-254

[21] Xlsxwriter modul (PyPi): https://pypi.org/project/XlsxWriter/ , pristup ostvaren: 20.12.2018.

[22] CANoe Vector Product Information Documentation: https://assets.vector.com/cms/content/products/canoe/canoe/docs/Product%20Information s/CANoe_ProductInformation_EN.pdf , pristup ostvaren: 15.12.2018.

[23] Specification CAN, Log & Trigger ASC Logging Format (documentation), version 1.4.3 dated 2015-09-01.

[24] Generator Expression Documentation: https://www.python.org/dev/peps/pep-0289/ , pristupljeno: 16.01.2019.

SAŽETAK

Ovaj diplomski rad bavi se tematikom pronalaženja grešaka i izguljenih poruka u mreži zasnovanoj na CAN komunikacijskom protokolu. U sklopu zadatka rada kreiran je novi algoritam za filtriranje, klasifikaciju podatkovnog prometa i detekciju pogrešaka i izgubljenih poruka koji obrađuje .asc log i scenarij usmjeravanja poruka u mreži zasnovanoj na CAN protokolu. Kreirano vlastito rješenje u sebi analizira 16 vrsta usmjeravanja u mreži sa 4 čvora. Algoritam služi za otkrivanje izgubljenih poruka i poruka s neispravnim podatkovnim blokom. Kao rezultat algoritma dobije se izvještaj u obliku .xlsx datoteke kako bi se moglo proučiti na kojim kanalima se stvaraju najveću problemi i koje poruke se najviše gube. Na temelju tog izvještaja moguće je analizirati komunikaciju i poduzeti korake prema poboljšanju komunikacije. Algoritam je kreiran u Python skriptom jeziku u PyCharm razvojnom okruženju. Kreirani algoritam ima značajke modularnosti i nadogradivosti jer je kreiran u dijelovima koji mogu biti izmijenjeni neovisno o drugim modulima. Verifikacija je provedena unutar programskog rješenja tako da su se kreirali slučajevi unošenjem anomalija u .asc (ručno promijenjeni podaci) datoteku. Rezultati verifikacije su pozitivni i udovoljavaju zahtjevima postavljenim prije izrade samog rješenja.

Ključne riječi: CAN protokol, otkrivanje grešaka, CAN sabirnica, usmjeravanje, komunikacije

TRAFFIC FILTERING AND CLASSIFYING WITH FINDING OF ERRORS IN AUTOMOTIVE DATA TRAFFIC

ABSTRACT

This thesis deals with the topic of fault finding and lost messages in a network based on CAN communication protocol. As part of the assignment, a new algorithm for filtering, classifying data traffic and detecting errors and lost messages has been created, that processes the .asc log and the message routing scenario in a network based on the CAN protocol. Created own solution analyzes 16 types of routing in a 4 node network. The algorithm serves to detect lost messages and messages with faulty data block. As a result of the algorithm, a report is given in form of .xlsx file so we could find out which channels are biggest problem and which messages are most losing ones. Based on this report, it is possible to analyze communication and take steps to improve communication. The algorithm is created in the Python script language in PyCharm development environment. The created algorithm has features of modularity and upgradability because it is created in parts that can be modified independently of other modules. Verification was carried out within the software solution so that cases were created by entering anomalies into the .asc (manually modified data) file. The results of the verification are positive and correspond to the set requirements before making the solution itself.

Key words: CAN protocol, error detection, CAN bus, routing, communications

ŽIVOTOPIS

Luka Roguljić rođen je 20. prosinca 1994. godine. Osnovnu školu završava u Ernestinovu i 2009. godine upisuje Elektrotehničku i prometnu školu Osijek, smjer tehničar za računalstvo. Završetkom srednjoškolskog obrazovanja 2013. godine upisuje stručni studij informatike na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija. Završetkom stručnog studija 2016. godine stekao je zvanje: prvostupnik (baccalareus) inženjer elektrotehnike (bacc. ing. el.) i iste godine upisuje Razlikovne obveze. 2017. godine upisuje diplomski studij računarstva, smjer programsko inženjerstvo. Trenutno je stipendist tvrtke Institut RT-RK Osijek.