SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1705

Sustav za neinvazivno praćenje i analizu prometa na CAN sabirnici

Jurica Šestok

ZAGREB, lipanj 2018.

Sadržaj:

Zahvaljujem se svojim roditeljima, bakama i djedovima, koji su bili uz mene i podržavali me tijekom svih godina studija, u najlakšim i najtežim trenucima.

Veliku zahvalnost izražavam i svom mentoru, prof.dr.sc. Davoru Petrinoviću, koji je uz savršen omjer savjeta, kritike i razumijevanja iz mene izvukao ono najbolje i naučio me kako pristupiti stvarnim inženjerskim problemima.

Također, zahvaljujem se mag.ing. Saši Tepiću na pomoći i savjetima pri izradi ovog rada.

Hvala i mom prvom mentoru, izv.prof.dr.sc Hrvoju Mlinariću, koji mi je pomogao pri izradi mog prvog projekta na fakultetu.

Posebno hvala mojim prijateljima koji su mi, osim timskog rada u zajedničkom studiranju, pružili dozu opuštanja, zabave i razumijevanja te mi bili podrška tijekom cjelokupnog studija.

Hvala Vam!

Diplomski rad je izrađen u okviru suradnje Sveučilišta u Zagrebu, Fakulteta elektrotehnike i računarstva i Ericsson Nikole Tesle d.d. na istraživačko-razvojnom projektu „Poboljšanje karakteristika rada LTE radio pristupnih uređaja“ (Improvements for LTE radio access equipment – ILTERA)

1. Uvod

Korištenje raznih komunikacijskih protokola danas uvelike olakšava komunikaciju i razmjenu podataka između ljudi, računala i ljudi i računala. Jedan od ključnih takvih protokola, pogotovo u automobilskoj industriji, je i CAN komunikacijski protokol. CAN se koristi u mnogim sustavima u kojima je ključno da podaci u sustavu dolaze na vrijeme i budu točni. Važno je napomenuti da je CAN komunikacija koja se odvija između računala koja, najčešće, nemaju sučelje za prikaz podataka.

Kada se takav sustav stavlja u pogon, ispravnost rada potrebno je testirati tako da se provjerava ispravnost podataka koji se nalaze na sabirnici. Štoviše, često se želi imati uvid i u performanse takvog sustava u realnom vremenu, u smislu očitanja statističkih podataka komunikacije poput popunjenosti sabirnice, broja poruka na sabirnici, broja poruka s pogreškom, itd.

Cilj ovog rada je razviti sustav za neinvazivno (bez remećenja toka podataka) praćenje, pohranu i prikaz podataka te analizu toka podataka CAN sabirnice u realnom vremenu. Sustav je potrebno implementirati na Linux operacijskom sustavu kako bi kao takav bio primjenjiv na relativno jeftinim računalima poput Raspberry Pija. Naposljetku, važno je opisati kako sustav radi te priložiti upute za instalaciju i korištenje.

2. Razrada

2.1. CAN – *Controller area network*

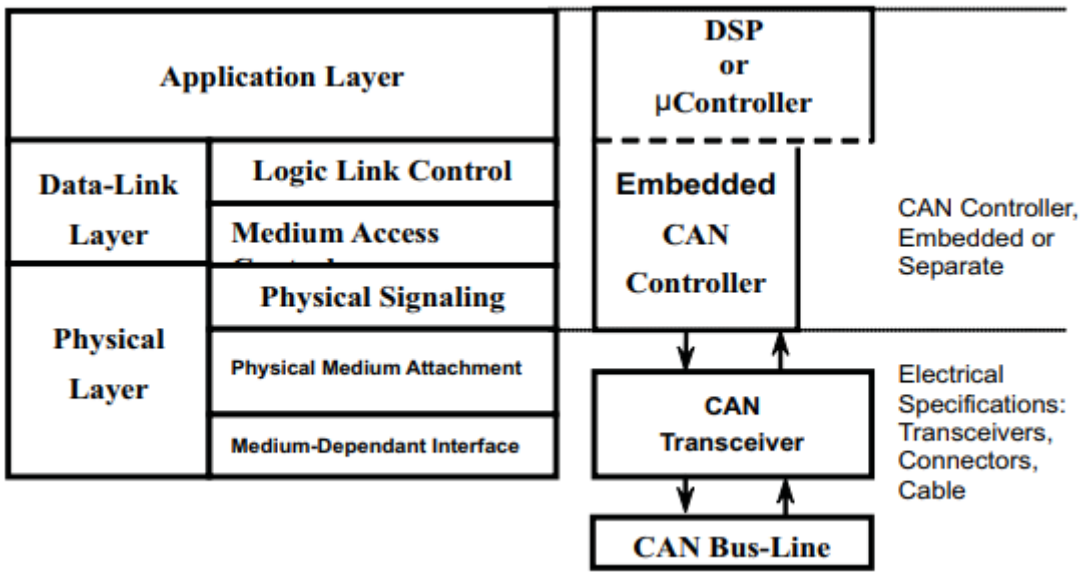
CAN [1] je serijska komunikacijska sabirnica koja je izvorno razvijena za uporabu u automobilskim aplikacijama početkom 1980-tih. CAN protokol je međunarodno standardiziran 1993. kao ISO 11898-1. Ovaj standard definira podatkovni sloj i dio fizičkog sloja referentnog ISO/OSI modela.

CAN protokol na podatkovnom sloju osigurava:

- *Multi-master* hijerarhiju – umjesto da sabirnicom upravlja strogo jedan čvor, upravljanje sabirnicom distribuirano je između više čvorova, što omogućuje kreiranje inteligentnih, redundantnih i distribuiranih sustava
- *Broadcast* komunikaciju – pošiljatelj informacije odašilje poruku svim čvorovima na sabirnici, a odluku o primitku i upotrebi informacije donosi svaki čvor primatelj pojedinačno. Rezultat je smanjenje potrebne širine

pojasa i povećani integritet podataka (jer svi čvorovi na sabirnici koriste iste informacije).

- Mehanizme za otkrivanje, signalizaciju i ponovno slanje oštećenih poruka, što povećava pouzdanost same komunikacije

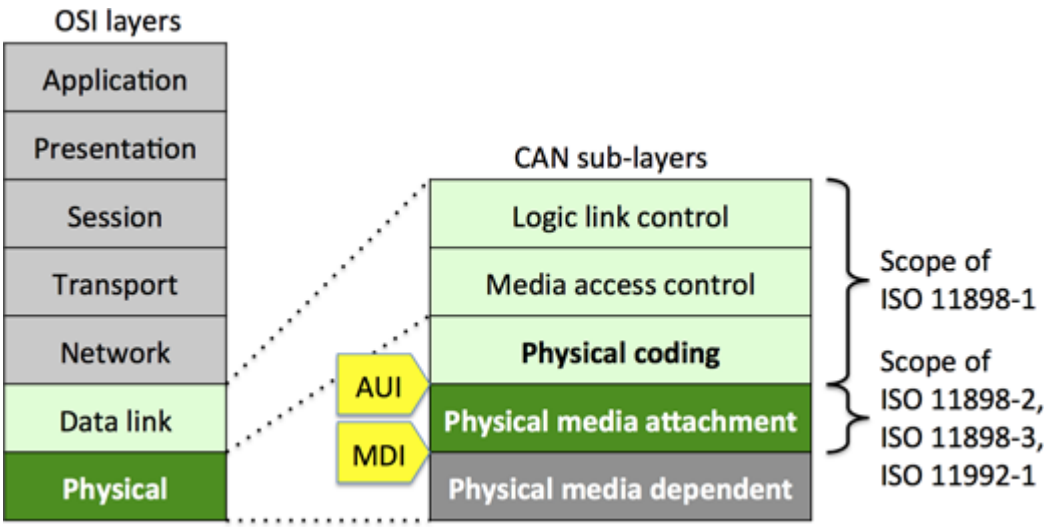


Slika 2.1.1 – Slojevita ISO 11898 standardna arhitektura [2]

2.2. Fizički sloj

CAN fizički sloj podijeljen je u tri dijela: dio fizičkog kodirajna (engl. *PCS – physical coding*) implementiran u upravljačkim čipovima, dodatak fizičkom mediju (engl. *PMA – physical media attachment*), koji definira primopredajne karakteristike te dio koji se odnosi na fizičke pod-slojeve ovisne o mediju (engl. *PMS – physical media-dependent sub-layers*). PMS ovisi o aplikaciji i nije standardiziran.

U PCS spadaju kodiranje i dekodiranje bitova, (re)sinkronizacija te vremenski odnosi unutar bitova. Ovaj pod-sloj pruža sučelje prema primopredajnim čipovima implementiranim pomoću TxD i RxD pinova. Postoje dvije verzije implementacija: 5-V verzija i 3,3-V verzija [3], [1].



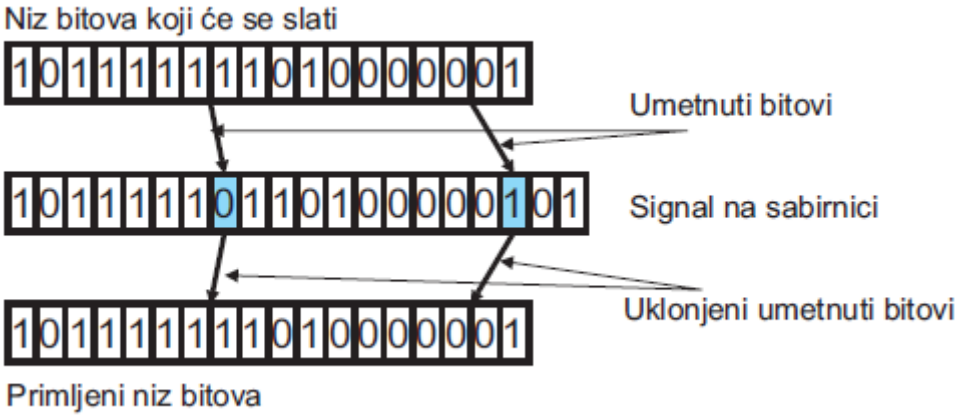
Slika 2.2.1 – Slojevi, sučelje priključne jedinice[1] (engl. *AUT – attachment unit interface*) i sučelje ovisno o mediju[2] (engl. *MDI – medium-dependent interface*) [3]

2.2.1. PCS - Kodiranje bitova

CAN koristi kodiranje bez povratka u nulu (engl. *NRZ – non-return-to-zero coding*). Kodiranje bez povratka u nulu znači da razina signala ostaje stalna tijekom cijelog vremena trajanja bita, tj. za reprezentaciju bita dovoljan je jedan vremenski prozor. Uzevši u obzir činjenicu da razina signala može ostati stalna tijekom duljeg vremenskog perioda, potrebno je osigurati da vrijeme između dva uzastopna brida signala ne bude predugo kako bi čvorovi na sabirnici ostali sinkronizirani.

Zbog toga CAN protokol koristi tzv. umetanje bitova (engl. *bit stuffing*). Nakon pet bitova iste vrijednosti CAN upravljač automatski umeće jedan bit suprotne vrijednosti. Analogno tome, CAN čvorovi primatelji nakon svakih pet bitova iste vrijednosti brišu jedan bit koji slijedi. Umetnuti bitovi su također vidljivi na sabirnici.

[1] AUT služi kao sučelje između CAN upravljača (engl. CAN *controller*) i CAN primopredajnog čipa
[2] MDI služi kao sučelje prema fizičkom dijelu sabirnice



Slika 2.2.1.1 – Primjer umetanja bitova u poruku - *bit stuffing* [1]

U teoriji, najveći mogući broj umetnutih bitova računa se sljedećom formulom:

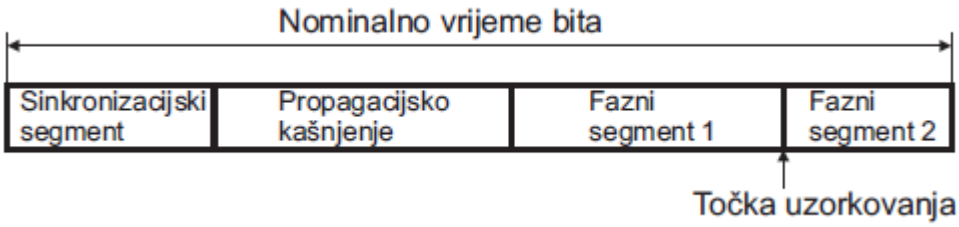
$$\frac{34 + 8 \times DLC - 1}{4}$$

Ako je sabirnica prazna (recesivno stanje u trajanju najmanje tri vremenska prozora), prvi padajući brid (recesivno --> dominantno stanje) se koristi za globalnu sinkronizaciju svih CAN upravljača. Svaki sljedeći padajući brid (recesivno --> dominantno stanje) koristi se za lokalnu sinkronizaciju čvorova. Resinkronizacija je limitirana podešenim vremenima bita (poglavlje 2.2.2).

2.2.2. Postavke vremena bita kod klasičnog CAN-a

Na razini bita, klasični CAN koristi sinkroni prijenos bitova. Na taj način se povećava kapacitet prijenosa uz potrebu za sofisticiranim metodama sinkronizacije bitova. U odnosu na asinkroni prijenos bitova, u kojem se sinkronizacija provodi nakon prijema prvog bita svakog znaka, protokol sinkronog prijenosa bitova sadrži samo jedan sinkronizacijski bit na početku poruke. Da bi prijemnik mogao ispravno čitati poruke, potrebna je stalna resinkronizacija. Zbog toga se prije i nakon nominalne točke uzorkovanja, unutar intervala bitova, umeću segmenti faznog međuspremnika.

CAN bit sadrži, ovisno o namještenoj „nedjeljivoj“ vremenskoj jedinici, vremenske kvante (tq). Trajanje vremenskog kvanta izvodi se iz brzine prijenosa bita preddijelila i frekvencije korištenog oscilatora.



Slika 2.2.2.1 – Logički segmenti bita

CAN bit je logički podijeljen u četiri dijela:

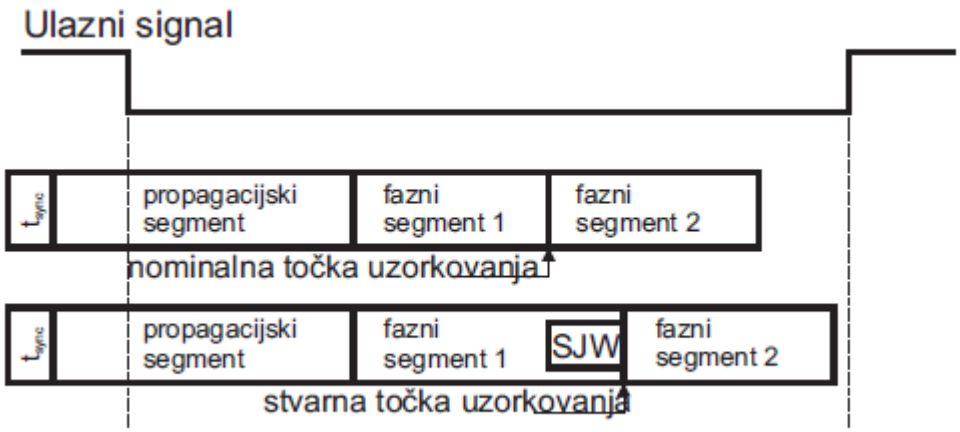
1. Sinkronizacijski segment – uvijek traje 1 tq i koristi se za sinkronizaciju različitih čvorova.
2. Propagacijski segment – može trajati 1-8 tq i koristi se za kompenzaciju propagacijskih kašnjenja signala na sabirnici. Segment uzima u obzir i kašnjenja koja unose odašiljući i primajući čvorovi.
3. Fazni segment 1 – može trajati 1-8 tq i koristi se za kompenzaciju faznih grešaka. Također, može biti produžen tijekom resinkronizacije.
4. Fazni segment 2 – koristi se za kompenzaciju faznih grešaka i može biti skraćen tijekom resinkronizacije, suprotno faznom segmentu 1. Fazni segment 2 mora biti dovoljno dugačak kako bi se primljena poruka stigla obraditi tijekom njegova trajanja. Vrijeme obrade poruke traje najviše 2 tq.

Kasno uzorkovanje omogućuje veću dužinu sabirnice, dok rano uzorkovanje dozvoljava sporije rastuće i padajuće bridove signala, tj. mogućnost resinkronizacije. Postoje dva tipa sinkronizacije: tvrda (engl. *Hard synchronization*) sinkronizacija i resinkronizacija (engl. *Bit resynchronization*).

Tvrda sinkronizacija događa se na početku poruke. Nakon nje je vrijeme bita resetirano na kraju sinkronizacijskog segmenta, neovisno o faznoj grešci brida. Brid koji je uzrokovao tvrdu sinkronizaciju nalazi se unutar sinkronizacijskog segmenta resetiranog vremena bita.

Resinkronizacija dovodi do skraćivanja ili produživanja vremena bita. Resinkronizacija uzrokuje da se točka uzorkovanja pomiče prema detektiranom bridu.

Kada je fazna pogreška brida koji je izazvao resinkronizaciju pozitivna, fazni segment 1 se produžuje. Ako je veličina fazne pogreške manja od širine sinkronizacijskog skoka (engl. *SJW – synchronization jump width*), fazni segment 1 se produljuje za veličinu fazne pogreške. Inače se produljuje za SJW.



Slika 2.2.2.2 – Primjer pozitivne fazne pogreške

Kada je fazna pogreška brida koji je uzrokovao resinkronizaciju negativna, fazni segment 2 se skraćuje. Ako je veličina fazne pogreške manja od širine SJW-a, fazni segment 2 skraćuje se za veličinu fazne pogreške. U suprotnom, skraćuje se za SJW-a [1], [4].

Slika 2.2.2.3 – Primjer resinkronizacije negativne fazne pogreške

2.3. Podatkovni sloj

CAN podatkovni sloj sadrži dva protokola: klasični CAN i CAN FD, koji je standardiziran 2015. normom ISO 11898-1 [1], [3]. U ovome radu neće se opisivati karakteristike CAN FD protokola[3], no važno je navesti njihove zajedničke značajke:

- Svaki čvor ima pravo tražiti pravo na odašiljanje u bilo kojem trenutku

[3] Korištenje fraza poput „CAN poruka“ ili „CAN sabirnica“ podrazumijeva korištenje klasičnog CAN protokola u ostatku ovog rada

- Arbitracija se izvodi na jednak način – poruka s najvišim dodijeljenim identifikatorom ima pravo prioriteta bez vremenske odgode
- Svi paketi (podatkovni, udaljeni, paket pogreške, paket zagušenja) šalju se metodom razašiljanja (engl. *broadcast*)
- Neka polja podatkovnog paketa su jednaka

2.3.1. CAN poruka

CAN protokol podržava dva formata poruka između kojih je najvažnija razlika duljina identifikatora u arbitražnom polju. Prvotni standard definirao je duljinu identifikatora poljem od 11 bitova što je kasnije prošireno na 29-bitovni identifikator. Za razlikovanje formata koristi se bit IDE (engl. *ID extension*) u upravljačkom polju poruke.

CAN poruka definirana je sljedećim standardima:

1.x Oznaka za izvornu specifikaciju

2.0A Podržava samo 11-bitovne identifikatore. 29-bitovni identifikator izazvat će pogrešku.

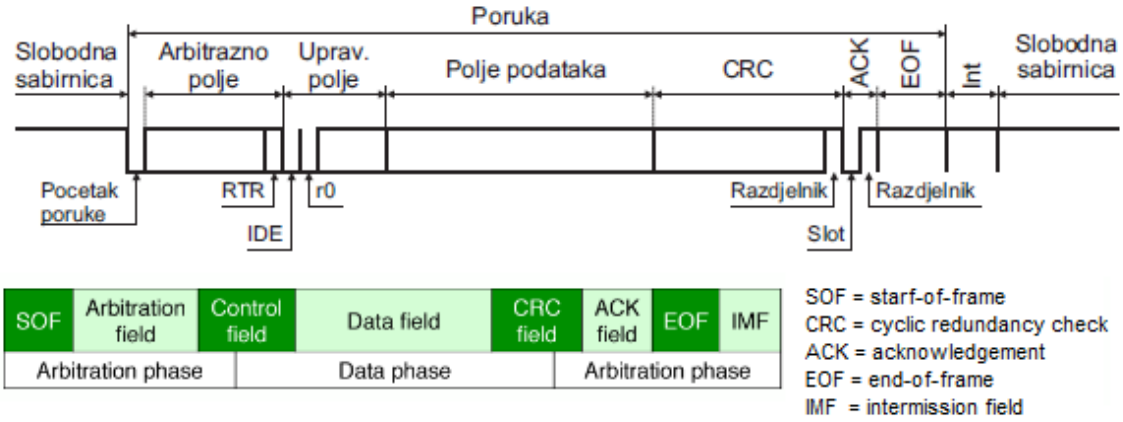
2.0B Proširena verzija s 29-bitovnim identifikatorom. Ova verzija podržava i 11-bitovni identifikator. Čvor 2.0B može poruke s proširenim identifikatorom tretirati na dva različita načina, ovisno o tome kako je podešen. Prvi način je da poruke s proširenim identifikatorom normalno prima i šalje. U drugom načinu čvor prima i šalje samo poruke s 11-bitovnim identifikatorom. Ako se u ovom načinu primi poruka s proširenim identifikatorom, ona ne izaziva pogrešku, ali čvor poruku, ako je ona ispravna, zanemaruje. Format 2.0B podržava istovremeno postojanje poruka oba formata na sabirnici.

Protokolom su definirana četiri tipa paketa (poruka):

1. Paket podataka (engl. *Data frame*) – osnovni i najčešći tip poruke koji može sadržavati i do 64 bita korisnih podataka.
2. Daljinski paket (engl. *Remote frame*) – poruka koja predstavlja zahtjev za određenim podacima (u njoj nema samih podataka).
3. Paket pogreške (engl. *Error frame*) – poruka koja signalizira pogrešku na sabirnici

4. Paket zagušenja (engl. *Overload frame*) – čvor koji odašilje ovaj paket nije spreman za prihvat sljedeće poruke. Većina CAN upravljača ga danas više ne koristi.

Čvorovi samostalno šalju podatke na CAN sabirnicu kada to žele ili kada to od njih zatraži drugi čvor pomoću daljinskog paketa (CiA danas ne savjetuje korištenje daljinskih paketa). U jednoj CAN poruci može se prenijeti i do 8 bajtova podataka.



Slika 2.3.1.1 – Dijelovi CAN poruke [1], [3]

Svaka CAN poruka počinje dominantnim **bitom početka poruke** (engl. *SOF – start of frame*) koji izaziva tvrdu sinkronizaciju svih čvorova spojenih na sabirnicu. Sinkronizacija se događa prilikom svake promjene bita iz recesivnog u dominantno stanje, kada je sabirnica slobodna, na kraju pauze između poruka i za vrijeme kada je čvor u stanju zaustavljene komunikacije (engl. *Suspended state*). Ovakvo ponašanje definirano je sinkronizacijskim pravilima CAN sabirnice.

Nadalje, slijedi **arbitražno polje** koje definira identifikator poruke – oznaku sadržaja i prioriteta poruke. Slijedi **bit zahtjeva za slanjem** (engl. *RTR - Remote transmission request*), koji definira je li poruka paket podataka ili daljinski paket. Taj je bit kod podatkovnog paketa u dominantnom stanju, dok je kod daljinskog paketa u recesivnom stanju. Zbog toga paket podataka ima viši prioritet nego daljinski paket. Sljedeće polje je **upravljačko polje** (engl. *Control field*). Upravljačko polje sadrži informaciju o broju prenesenih bajtova podataka u podatkovnom polju. Ono počinje **IDE bitom**, koji određuje radi li se o osnovnoj ili

proširenoj CAN poruci. Standardni identifikator određen je dominantnim stanjem IDE bita, čime je osnovnim CAN porukama dan viši prioritet od proširenih.

U upravljačkom polju, nakon IDE bita, slijede četiri bita naziva **kod duljine poruke** (engl. *Data length code*), koji određuju broj prenesenih bajtova u podatkovnom polju. DLC može biti broj između 1 i 8, a ako DLC ima vrijednost veću od 8 to se tumači kao 8 prenesenih bajtova. U proširenoj poruci nakon IDE bita slijedi 18 nižih bitova identifikatora, a tek nakon njih DLC polje. Nakon upravljačkog polja slijedi **polje podataka** (engl. *Data field*) u kojem se nalaze podaci koje prenosi poruka.

Nakon podatkovnog polja slijedi **polje cikličnog zaštitnog koda** (engl. *CRC - Cyclic redundancy check*) koje se koristi za detekciju mogućih grešaka u komunikaciji. Polje se sastoji od 15 bitovnog zaštitnog koda nakon kojeg se nalazi recesivni razdjelni bit. CRC je definiran sljedećim polinomom:

$$x^{15} + x^{14} + x^{10} + x^{8} + x^{7} + x^{4} + x^{3} + 1$$

Nakon CRC polja nalazi se **polje potvrde** (engl. *Acknowledgement field*), u kojem se nalaze dva bita postavljena u recesivnu vrijednost, od strane čvora koji šalje poruku – *Slot* bit i razdjelni bit. Čvorovi koji su primili poruku postavljaju *Slot* bit u dominantnu vrijednost, čime se signalizira da je poruka primljena sa sabirnice.

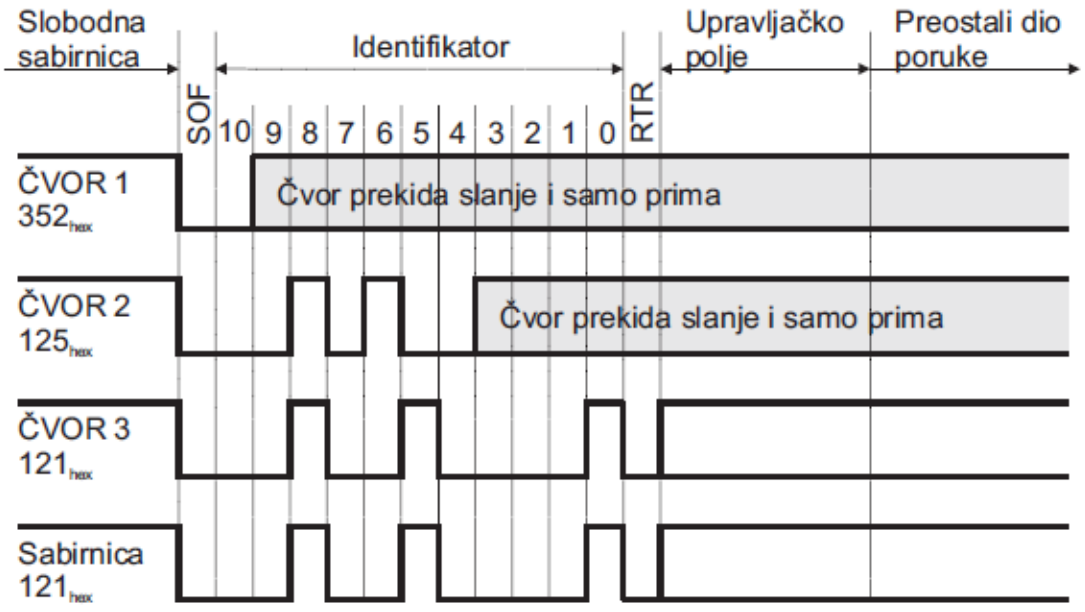
Naposljetku, šalje se sedam recesivnih bitova koji obilježavaju **kraj poruke** (engl. *End-of-frame*). Budući da sabirnica mora biti u recesivnom stanju u trajanju od barem još tri vremenska okvira, na samom kraju poruke nalazi se **polje pauze** (engl. *Intermission field*). Ova pauza osigurava čvorovima koji su primili poruku minimalno vrijeme za obradu poruke, kako bi bili spremni za prijem i obradu sljedeće poruke.

2.3.2. Metoda arbitraže sabirnice

CAN protokol podržava istovremen pristup sabirnici dvaju i više čvorova. U slučaju da više čvorova istovremeno želi pristupiti sabirnici, potrebno je odraditi proces arbitraže, tj. dodijeliti sabirnicu čvoru s najvećim prioritetom. Prioritet čvoru dodjeljuje dizajner sustava i čvorovi s nižim CAN-ID-jem imaju viši prioritet – ID „0“ ima najviši prioritet. CAN koristi metodu nedestruktivne arbitraže koja se temelji na detekciji kolizije i prioritetu poruke – CSMA/CD+AMP (engl. *Carrier sense multiple access with collision detection and arbitration on message priority*).

Preduvjet početku slanja poruke je neaktivno stanje sabirnice. Čvorovi započinju slanje poruke bitom početka te za vrijeme slanja čitaju stanje sabirnice i uspoređuju ga s vrijednošću koju su poslali na sabirnicu. Svi čvorovi spojeni su na sabirnicu u spoj „i“ („*AND*“). Svi čvorovi koji šalju poruke recesivne razine, u odnosu na dominantnu razinu kojoj je dodijeljena sabirnica, prelaze u stanje slušanja. Iznimka ovome događa se kada daljinski paket i paket podataka imaju jednak CAN-ID, no sabirnicu u ovom slučaju dobiva paket podataka, zbog RTR bita.

Na slici 2.3.2.1 prikazan je primjer arbitraže triju čvorova na sabirnici, od kojih jedan ima identifikator 0x352, drugi 0x125 i treći 0x121. Svi čvorovi započinju slanje bitom početka poruke i, odmah potom, identifikatora. Čvorovi identifikator šalju bit po bit, počevši s najznačajnijim bitom. Deseti bit sva tri identifikatora je dominantan pa će zbog toga sva tri čvora nastaviti slanje ostatka bitova.



Slika 2.3.2.1 – Primjer arbitraže na sabirnici

No, već na devetom bitu dolazi do „ispadanja“ jednog od čvorova – čvora 1, koji, nakon što pročita stanje sabirnice i prepozna da se ona nalazi u dominantnom stanju, a ne u recesivnom (koje on pokušava postaviti). Čvor 1 zbog toga prekida slanje i počinje sa slušanjem sabirnice. Analogno tome, čvorovi 2 i 3 nastavljaju slanje identifikatora sve do trećeg bita, kada čvor 2 postavlja recesivan bit na sabirnicu i gubi arbitražu, jer je čvor 3 postavio dominantno stanje na sabirnicu. Čvor 2 tada također prestaje sa slanjem i počinje slušati sabirnicu. Čvor 1 pobjeđuje arbitražu i nastavlja sa slanjem poruke.

2.3.3. Metoda pronalaženja pogrešaka

Podatkovni slojevi CAN protokola su vrlo pouzdani zbog svoje mogućnosti da otkrivaju sve jedno-bitne pogreške, a s visokom vjerojatnošću otkrivaju i više-bitovne pogreške. Također, mehanizmi otkrivanja pogrešaka osiguravaju ispravan rad cjelokupnog sustava jer na odgovarajuće načine upravljaju „problematičnim“ čvorovima kako oni ne bi ometali promet na sabirnici.

Proces u slučaju pojavljivanja CAN pogreške vrlo je jednostavan i sastoji se od 5 koraka:

1) CAN upravljač uočava pogrešku
2) Odmah se šalje paket greške
3) Poruka se poništava na svim čvorovima
4) Osvježavaju se stanja svih CAN upravljača
5) Poruka se ponovno šalje. Ako više upravljača želi poslati poruke, koristi se uobičajena arbitraža.

CAN protokol sadrži 5 mehanizama za pronalaženje pogrešaka, od kojih su 3 na razini poruke i 2 na razini bita. Za razliku od većine drugih komunikacijskih sustava, oni ne koriste poruke potvrđivanja, nego signaliziraju bilo kakve pogreške koje su se pojavile.

Metode za pronalaženje pogreške na razini poruke su sljedeće:

1) **Pogreška CRC** – kod primitka poruke, čvor primatelj računa CRC iz dobivene poruke. Ako se izračunati bitovi ne podudaraju s bitovima u CRC polju poruke znači da se dogodila pogreška.
2) **Provjera okvira** – provjeravanjem strukture prenesenih podataka i daljinskog okvira (usporedbom bitovnih polja s fiksno određenim vrijednostima i provjerom veličina okvira) otkrivaju se pogreške naziva „pogreške formata“.
3) **Pogreška potvrde** – svi čvorovi potvrđuju da su primili poruku tako da na sabirnicu postave dominantni bit tijekom vremena predviđenog za *ACK* bit. Ako čvor pošiljatelj ne vidi dominantni bit tijekom vremena *ACK* bita, to može značiti ili da je došlo do pogreške prijenosa koju su prepoznali čvorovi prijemnici, da je polje ACK bita iskvareno ili da ni jedan čvor nije primio poruku.

Metode za pronalaženje pogreške na razini bita su sljedeće:

1) **Nadgledanje** – mogućnost pošiljatelja da pronađe pogrešku temelji se na promatranju signala na sabirnici. Svaki čvor koji šalje podatke ujedno i provjerava postoje li razlike između poslanih podataka i stanja na sabirnici te na taj način pouzdano otkriva globalne pogreške i pogreške koje su se dogodile na pošiljatelju.
2) **Umetanje bitova** – pogreška koja se u ovom slučaju može dogoditi je dolazak više od 5 uzastopnih bitova iste razine na primatelja, što primatelj vrlo rako raspoznaje kao pogrešku.

Pronađene pogreške razlučuju se postavljanjem zastavice pogreške i takav paket naziva se paket pogreške. Zastavica pogreške sastoji se od šest bitova koji imaju jednake razine i osam bitova recesivne vrijednosti, zvanih razdjelnik pogreške (engl. *Error delimiter*). Pronalaženjem pogreške i slanjem paketa pogreške, sprječava se prijem poruke s pogreškom u svim čvorovima, čime se osigurava konzistencija podataka na sabirnici. Čvor koji je poslao poruku s pogreškom ponovno pokušava poslati istu poruku. To se naziva automatska retransmisija. Ovakvo slanje poruke također podvrgava već utvrđenim pravilima arbitraže na sabirnici.

Moć CAN protokola da utvrdi i otkrije pogreške je iznimno velika zbog čega se javlja teoretska opasnost od prevelikih kašnjenja na sabirnici ili potpunog prestanka rada sustava. Ovaj je problem riješen uvođenjem mjera samodijagnostike čvorova. CAN protokol definira dva brojača pogrešaka: brojač pogrešaka u primljenim porukama (engl. *REC – received error messages counter*) i brojač pogrešaka u poslanim porukama (engl. *TEC – transmitted error messages counter*). Ovisno o vrijednostima ova dva brojača, CAN upravljač nalazi se u jednom od tri moguća stanja:

1) Aktivan na pogrešku (engl. *error active*) – stanje normalnog načina rada. Čvor može primati i slati poruke. U slučaju da čvor otkrije pogrešku, šalje aktivnu poruku o grešci, koja sadrži šest dominantnih bitova.
2) Pasivan na pogrešku (engl. *error passive*) – način rada u koji upravljač ulazi kada često ima problema s primanjem ili slanjem poruka. Čvor u ovom stanju i dalje može primati i slati poruke, a u slučaju pogreške šalje se

pasivna poruka o grešci, koja sadrži šest recesivnih bitova. Ovime se osigurava da čvor koji ima problema s prijemom ne blokira sabirnicu.

3) Odspojen sa sabirnice (engl. *buss off*) – način rada u kojem upravljač ne može niti slati niti primati poruke. Jednom kada upravljač uđe u ovaj način rada, mora biti resetiran od strane vanjskog sklopa, nadređenog mikrokontrolera ili procesora, kako bi nastavio s radom. Upravljač je dozvoljeno upaliti tek nakon 128 prijema 11 uzastopnih recesivnih bitova.

Tablica 2.3.3.1 – Načini rada upravljača i uvjeti ulaska u njih

Način rada	Uvjet stanja
Error active	TEC[4] <= 127 ***AND*** REC[5] <= 127
Error passive	(TEC > 127 ***OR*** REC > 127) ***AND*** TEC < 255
Buss off	TEC > 255

2.4. CANopen

CANopen je komunikacijski sustav temeljen na CAN-u. Sadrži više slojeve protokola i specifikacije profila. Sustav je razvijen kao standardizirana ugradbena mreža[6] s iznimno visokom fleksibilnošću konfiguracije. Danas se koristi u različitim područjima primjene kao što je medicinska oprema, automobilska industrija ili automatizacija zgrada.

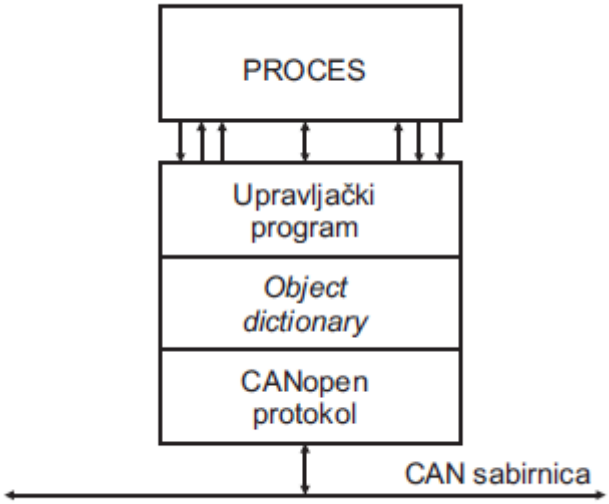
Glavni cilj CANopen-a je rasteretiti dizajnere i razvojne inženjere sustava od bavljenja detaljima koji su usko vezani uz detalje specifične za niže slojeve CAN protokola. CANopen pruža standardizirane komunikacijske objekte (engl. *COB – communication objects*) za vremenski kritične procese, konfiguraciju i upravljanje mrežnim podacima.

Temeljitijom provedbom standardizacije komunikacijskih profila povećava se jednostavnost integracije različitih čvorova na sabirnicu, a samim time i integracija cijelog sustava. Komunikacijski profili specificiraju koji komunikacijski i aplikacijski objekti moraju ili mogu biti podržani, specificiraju i predodređuju procesne podatke koji će biti mapirani u poruke te određuju reakcije pojedinih čvorova na primljene

[4] Oznaka se odnosi na broj pogrešaka u poslanim porukama. Analogija vrijedi za ostale oznake u tablici
[5] Oznaka se odnosi na broj pogrešaka u primljenim porukama. Analogija vrijedi za ostale oznake u tablici
[6] Mreža = sabirnica u kontekstu ovog diplomskog rada

poruke. Standardizacija profila također omogućuje ponovno korištenje dijelova aplikacijskog programa, međudjelovanje i izmjenjivost čvorova.

Svaki se CANopen čvor može podijeliti na tri logičke cjeline. **Stog** CANopen protokola upravlja komunikacijom pomoću CAN mreže. **Software aplikacije** pruža funkcionalnost unutarnje kontrole i služi kao sučelje prema procesu hardverskog sučelja. Treća cjelina je CANopen **rječnik objekata** (engl. *object dictionary*). On sadrži reference za sve korištene tipove podataka i sprema sve komunikacijske i aplikacijske parametre. [3], [1]



Slika 2.4.1 – CANopen logičke cjeline čvora

2.4.1. CANopen rječnik objekata

CANopen rječnik objekata je najvažniji za konfiguraciju uređaja i dijagnostiku. Kao unutarnja referenca uređaja koristi se 16-bitovni indeks zapisan kao 4 heksadekadske znamenke. To je skupina objekata dostupnih putem sabirnice.

Tablica 2.4.1.1 - Adresna područja CANopen rječnika objekata

Indeks	**Opis**
0	rezervirano
0x0001 – 0x025F	Tipovi podataka
0x0260 – 0x0FFF	Rezervirano
0x1000 – 0x1FFF	Komunikacijski objekti
0x2000 – 0x5FFF	Proizvođačka specifikacija
0x6000 – 0x9FFF	Specifikacija definirana uređajem
0xA000 – 0xBFFF	Specifikacija definirana sučeljem
0xC000 – 0xFFFF	Rezervirano

Područje rezervirano za tipove podataka definira, kao što sama fraza kaže, korištene tipove podataka. Nadalje, u području rezerviranom za komunikacijske objekte definira se ponašanje komunikacijskog sučelja čvora. 0x2000 – 0x5FFF definira se ponašanje aplikacije prema proizvođačkoj specifikaciji. Ovdje su smješteni parametri potrebni za rad specifičnih funkcija koje je definirao proizvođač te rezultati tih funkcija. Također, na ovim adresama mogu se prikazati i statusne informacije i podaci iz samog procesa.

U sljedećem adresnom području – području standardnog profila, definira se ponašanje aplikacije prema standardiziranim CANopen profilima. Ovo područje podijeljeno je u 8 sekcija i zbog toga je moguće u pojedinom čvoru implementirati do 8 različitih profila. Specifikacijom je definirano da su sve mrežne i sustavne varijable smještene na adresama u rasponu 0xA000 – 0xBFFF.

CANopen standard definira obavezne i izborne objekte, a svaki čvor mora implementirati barem obavezne objekte. Funkcionalnost samog uređaja opisuje se implementacijom pripadajućih objekata.

2.4.2. Standardne brzine komunikacije CANopen-a

Sljedeća tablica prikazuje standardne brzine komunikacije i pripadajuće maksimalne dužine sabirnice, maksimalne dužine nezaključenih ogranaka (engl. *stub*) i maksimalnu dozvoljenu dužinu ogranaka. Čvor mora podržavati barem jednu brzinu navedenu u tablici, a specifikacijom je određeno da trenutak uzorkovanja bude što je moguće bliže 87.5% nominalnog vremena bita.

Tablica 2.4.2.1 – Standardne brzine CANopen komunikacije

Brzina komunikacije	Dužina sabirnice	Max. Dužina ogranka	Ukupna dužina ogranaka
1 Mbit/s	25 m	1,5 m	7,4 m
800 Kbit/s	50 m	2,5 m	12,5 m
500 Kbit/s	100 m	5,5 m	27,5 m
250 Kbit/s	250 m	11 m	55 m
125 Kbit/s	500 m	22 m	110 m
50 Kbit/s	1000 m	55 m	275 m
20 Kbit/s	2500 m	137,5 m	687,5 m
10 Kbit/s	5000 m	275 m	1375 m

2.5. CANopen protokoli

Stog protokola CANopen-a implementira nekoliko komunikacijskih objekata koji su komunicirani jednom od brzina bitova. Ti objekti omogućavaju razvojnim inženjerima sustava da prenose kontrolne informacije, reagiraju na određene uvjete pogrešaka ili da utječu i upravljaju ponašanjem mreže. [1], [3]

2.5.1. SDO protokol

Service data objects ili servisni objekti omogućuju pristup svim objektima u rječniku objekata. SDO omogućuje uspostavu komunikacijskog kanala između 2 CANopen čvora. Jedan SDO sastoji se od dva CAN podatkovna paketa s različitim ID-jevima, budući da je SDO komunikacija s potvrdom. Čvor čijem se rječniku objekata pristupa naziva se SDO poslužitelj.

Klijentski čvor odabire kojim će podacima iz poslužiteljevog rječnika objekata pristupati, koji SDO tip se koristi i hoće li se podaci čitati ili upisivati. SDO se najčešće koristi za prijenos konfiguracijskih podataka i za prijenos većih podataka. Veliki podaci prenose se kao niz segmenata, a jedan se segment prenosi jednom CAN porukom. Također, prijenos svakog segmenta potvrđuje se jednom CAN porukom.

2.5.2. PDO protokol

Process data objects ili procesni podatkovni objekti se koriste u CANopen-u za razašiljanje visokoprioritetnih kontrolnih i statusnih informacija. PDO se sastoji od jednog CAN paketa i može poslati do 8 bajtova čistih aplikacijskih podataka. Dizajneri sustava moraju procijeniti količinu podataka koju će čvorovi slati i primati. Rezultat ove procjene je broj PDO poruka. Svaka PDO poruka odgovara nekom unosu u rječniku objekata i osigurava sučelje prema objektima u aplikaciji. Jedan PDO zahtjeva skup komunikacijskih parametara i skup parametara mapiranja.

Komunikacijski parametri definiraju identifikator poruke i događaj koji pokreće slanje poruke, a parametri mapiranja definiraju koja će se informacija lokalnog rječnika objekata slati i gdje će primljena informacija biti spremljena. Komunikacijski parametri su spremljeni na adresama 0x1400 – 0x15FF za primanje, a 0x1800 – 0x19FF za slanje. Parametri mapiranja spremljeni su na indeksima 0x1600 – 0x17FF (primanje) i 0x1A00 – 0x1BFF (slanje).

PDO mapiranje definira koji objekti aplikacije se šalju pomoću PDO-a. Mapiranje opisuje redoslijed i dužinu mapiranih objekata. CANopen razlikuje tri vrste PDO mapiranja:

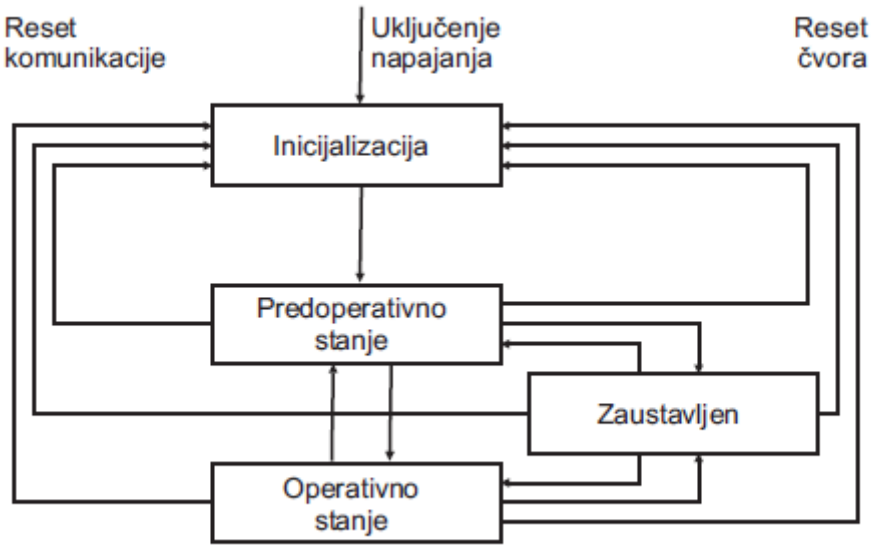
1) **Statičko mapiranje** – sadržaj PDO-a je strogo predodređen od proizvođača i kao takav ne može biti promijenjen korištenjem CANopen sučelja.
2) **Varijabilno mapiranje** – mapiranje PDO-a može biti promijenjeno tijekom NMT predoperativnog stanja (opisano u sljedećem poglavlju).
3) **Dinamičko mapiranje** – uređaj podržava dinamičko mapiranje ako PDO mapiranje može biti promijenjeno tijekom NMT operativnog stanja (opisano u sljedećem poglavlju).

Događaji koji okidaju slanje PDO poruke su sljedeći:

1) **Okidano događajem ili vremenskim sklopom** – događaj unutar uređaja okida slanje PDO poruke, npr. prekoračenje temperaturnog limita ili završetak brojanja brojača.
2) **Daljinski zahtjev** – drugi čvor, putem RTR-a, zahtjeva slanje PDO paketa.
3) **Sinkroni ciklički prijenos** – slanje PDO poruke započinje s primitkom SYNC poruke.
4) **Sinkroni aciklički prijenos** – događaj unutar sklopa je okidač slanja poruke, ali poruka se šalje tek po primitku SYNC poruke.

2.5.3. NTM – network management

Svi CANopen čvorovi moraju podržavati „*slave“* automat stanja. NMT automat stanja definira komunikacijsko ponašanje čvorova. Stanja automata su: **stanje inicijalizacije**, **predoperativno stanje**, **operativno stanje**, **zaustavljeno stanje**. Čvor ulazi u stanje inicijalizacije nakon uključivanja ili resetiranja. Jedan čvor na sabirnici mora ispunjavati funkciju nadređenog NMT čvora.



Slika 2.5.3.1 – NMT automat stanja

Nadalje, nakon što je inicijalizacija završena, čvor automatski prelazi u predoperativno stanje i pokazuje to slanjem *boot-up* poruke. Na taj način čvor ukazuje da je spreman za rad. Čvor u predoperativnom stanju može početi slati SYNC, *Heartbeat* ili *Time-Stamp* poruke, ako je bilo koji od navedenih servisa podržan i na ispravan način konfiguriran. Također, čvor može odašiljati i SDO poruke, no ne i PDO poruke. PDO komunikacija podržana je samo u operativnom načinu rada. U operativnom stanju čvor može koristiti sve podržane komunikacijske objekte i protokole. Kada čvor uđe u zaustavljeno stanje, on može jedino primati NMT naredbe. Također, u zaustavljenom stanju, čvor podržava protokol za nadzor pogrešaka.

Stanje inicijalizacije pokriva tri podstanja: **inicijalizacija**, **reset aplikacije** i **reset komunikacije**. Tijekom podstanja inicijalizacije, čvor se pokreće i inicijalizira interne parametre. Druga dva podstanja su stvorena kako bi omogućila naredbe djelomičnog reseta. Tijekom podstanja reseta aplikacije, svi parametre i rječniku objekata od adrese 0x2000 do 0x9FFF se postavljaju u inicijalne vrijednosti. Potom se ulazi u treće preostalo podstanje – reset komunikacije, u kojem se svi parametri komunikacijskog profila postavljaju u inicijalne vrijednosti – adrese 0x1xxx. Nakon izlaska iz ovog podstanja napušta se stanje inicijalizacije.

NMT protokol šalje aktivni NMT nadređeni čvor u CANopen mreži. Primanjem NMT protokola, *slave* čvor je prisiljen ući u naređeno stanje. NMT protokol je jedna

poruka dužine 2 bajta. Prvi bajt sadrži naredbu, a drugi ID čvora na kojeg se naredba odnosi. Ako je identifikatorski bajt postavljen u vrijednost '0', svi čvorovi moraju obaviti poslanu naredbu. Ova poruka također ima najviši mogući prioritet na CAN sabirnici. Postoji 5 NMT naredbi: pokreni udaljeni čvor, zaustavi udaljeni čvor, uđi u predoperativno stanje, resetiraj čvor i resetiraj komunikaciju.

2.5.4. Protokoli specijalnih funkcija

CANopen nudi tri protokola za stvaranje posebnog ponašanja na sabirnici:

- SYNC protokol – omogućuje sinkronizaciju.
- *Time-stamp* protokol – koristi se za prilagođavanje jedinstvenog vremena sabirnice.
- Hitan protokol (engl. *Emergency protocol*) – može se koristiti da bi se obavijestilo ostale čvorove na sabirnici o pogreškama unutar čvora.

SYNC protokol

SYNC signal periodički šalje jedan od čvorova na sabirnici, a vremenski period između dviju SYNC poruka predstavlja jedan komunikacijski ciklus. SYNC poruka mapirana je na jedan CAN paket s ID-jem 0x80.

***Time-stamp* protokol**

Ovaj protokol omogućava korisnicima CANopen sustava da podese jedinstveno vrijeme na sabirnici. *Time-stamp* je mapiran na jedinstveni CAN paket s podacima duljine 6 bajtova. U tih 6 bajtova sadržana je informacija o vremenu koje se računa kao broj milisekundi prošlih od ponoći i broja dana od 1. siječnja 1984. godine. ID ove poruke je 0x100.

***Emergency* protokol**

Hitne poruke okidaju se internim pogreškama CAN čvora. Hitna poruka veličine je jednog CAN paketa i sadrži 8 bajtova podataka. Prvi bajt označava registar pogreške (indeks 0x1001 lokalnog rječnika objekata), potom slijede 2 bajta koja označavaju kod hitne pogreške, a u ostatku bajtova sadržana je detaljnija informacija o pogrešci. ID hitne poruke je 0x80. Hitna poruka odašilje se samo jednom za svaku pojavu pogreške. Poruku ne mora nužno primiti ni jedan čvor, no može ju primiti i više čvorova, koji potom mogu poduzeti odgovarajuće protumjere.

Također, postoje protokoli za nadzor i upravljanje pogreškama. To su: *Heartbeat, Node-/Life Guarding* i *Boot-up* protokoli. Ovi protokoli služe nadzoru CANopen sabirnice i njenih čvorova. CANopen zahtjeva implementaciju barem jednog od prva dva navedena protokola, a preporuča se korištenje *Heartbeat* protokola.

***Heartbeat* protokol (*Node-/Life guarding protocol*)**

Heartbeat protokol je novija verzija zastarjelog *Node-/Life guarding* protokola. Ovaj protokol je zapravo periodičko slanje poruke koja obavještava sve slušatelje o dostupnosti onoga tko generira ovu poruku. Uz to, pošiljatelj javlja i svoje trenutno NMT stanje u kojem se nalazi. Period slanja poruke je podesiv u rječniku objekata na adresi 0x1017. poruka je dugačka jedan bajt, a ID joj je 0x700 + NODE-ID.

***Boot-up* protokol**

Boot-up poruka šalje se prilikom prelaska iz stanja inicijalizacije u predoperativno stanje. Primanje ove poruke ukazuje na to da se novi čvor registrirao na CAN sabirnicu. Nenamjerni prijem ove poruke može ukazivati na promjenu u postavkama sabirnice ili se smatra znakom za pogrešno stanje. Protokol koristi isti ID kao i *Heartbeat* protokol, a podatkovno polje duljine 1 bajt ima fiksnu vrijednost jednakoj '0'.

3. Sustav za praćenje i analizu prometa na CAN sabirnici

Radi olakšanja razvoja sustava koji koriste CAN protokol za komunikaciju između čvorova, te olakšavanja dijagnostike mogućih problema, potrebno je koristiti sustave i alate koji bi na jednostavan način mogli prikazivati razne podatke na sabirnici i to u stvarnom vremenu, a bez utjecaja na performanse sabirnice. Neki od tih podataka su:

- Prikaz cjelokupnog stanja komunikacije na sabirnici
- Prikaz poruka na sabirnici s pripadajućom vremenskom oznakom i pripadajućim identifikatorom
- Prikaz broja uočenih pogrešaka na sabirnici
- Prikaz popunjenosti sabirnice i učestalosti slanja poruka
- Pohrana prometa sa sabirnice ograničena količinom prostora u memoriji

Ovi će podaci omogućiti razvojnim inženjerima da na jednostavan način i sa što manje muke, uoče i na vrijeme isprave što više pogrešaka sustava prilikom njegova razvitka. Mnogi ovakvi alati su već sada dostupni za različite platforme, a u nastavku bit će opisan sustav za praćenje i analizu prometa implementiran na Raspberry Pi računalu. [1]

Sustav korišten u ovom radu sastoji se od:

- Raspberry Pi 3 Model B
- Pican2 CAN *Bus board*
- OBD2 to DB9 kabel
- SocketCAN - Aplikacija na Raspberry Piju

3.1. Raspberry Pi 3 Model B

Raspberry Pi je SBC[7] veličine kreditne kartice, razvijen u Ujedinjenom Kraljevstvu od strane Raspberry Pi zaklade, a s namjerom da promovira učenje osnova računalne znanosti u školama. Raspberry Pi je moćno računalo koje pruža mnogo sučelja i pinova za komunikaciju s „vanjskim svijetom“. Raspberry Pi 3 Model B ima sljedeće značajke:

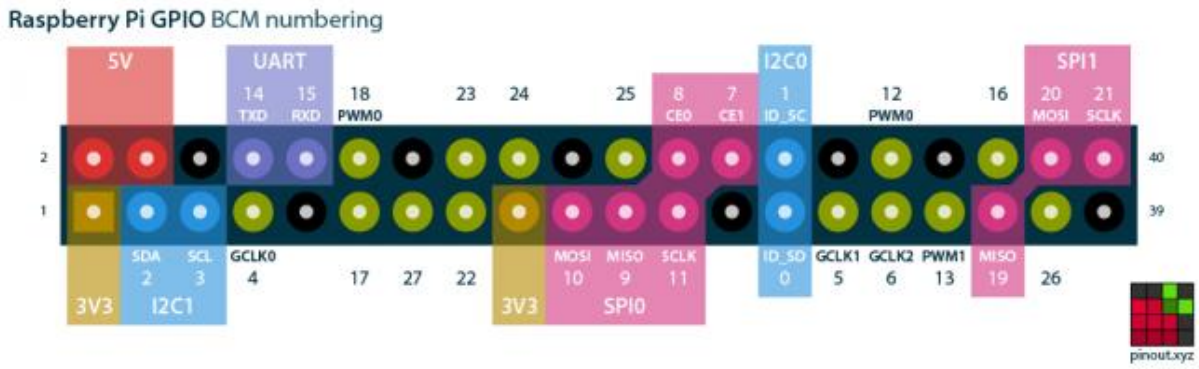
- 1.2GHz 64-bit četverojezgreni ARMv8 CPU
- 802.11n Wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy(BLE)
- 1GB RAM
- 4 USB porta
- 40 GPIO pinova
- Full HDMI port
- Ethernet port
- Kombinirani 3.5mm audio jack i miješani video
- Sučelje za kameru

Upravo zbog širokog spektra sučelja, svoje veličine, mobilnosti i visoke razine konfigurabilnosti, Raspberry Pi je prigodan za implementaciju upravljačke aplikacije ovakvog sustava. Temeljni i najčešći način upravljanja vanjskim

[7] *Single Board Computer* – računalo implementirano na jednoj tiskanoj pločici

uređajima su ulazno/izlazni pinovi za opću uporabu (engl. *GPIO* - general-purpose input/output). Raspberry Pi ima 40 ulazno-izlaznih pinova opće namjene na koje se mogu spojiti elektroničke komponente i pomoću kojih se može upravljati uređajima poput senzora ili svjetala. Oni podržavaju:

- 3V3 – stalni napon veličine 3.3 V
- 5V – stalni napon veličine 5 V
- GND – priključak mase napajanja, tj. 0 V
- GPIO – ulazno - izlazni pinovi opće namjene, tj. varijabilni 3V3 pinovi upravljani od strane korisnika
- SPI – Serial Peripheral Interface sabirnica
- I2C – Inter-integrated circuit
- UART – Univerzalni asinkroni prijemnik/transmitor

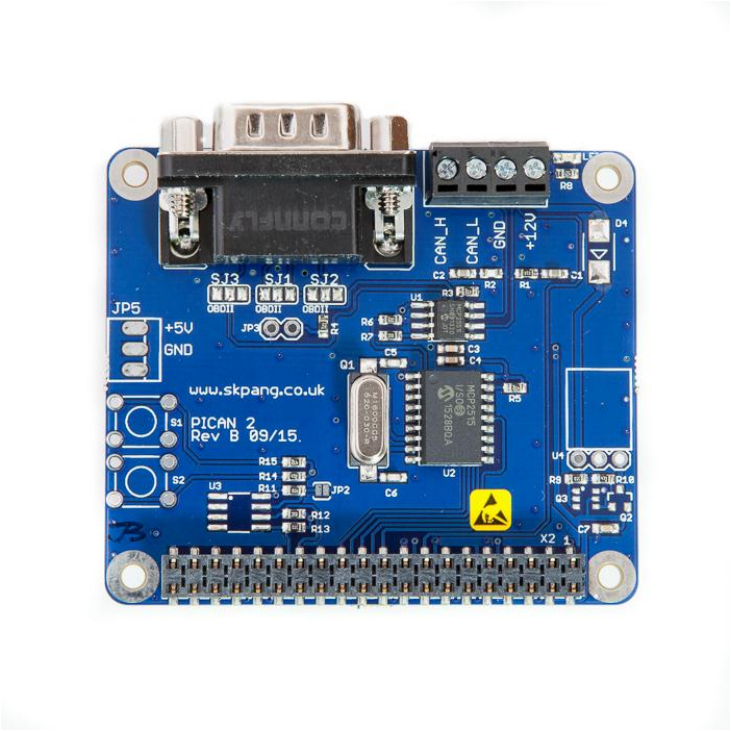


Slika 3.1.1 - Raspored GPIO pinova

Korištenje GPIO pinova na Raspberry Pi-ju nije pretpostavljeno, nego se za njihovo korištenje moraju instalirati GPIO biblioteke. Dvije su razvojne datoteke dostupne: Rpi.GPIO Python i WiringPi C. [5]

3.2. Pican2 – CAN-*Bus Board* za Raspberry Pi

Pican2 je pločica koja dodaje CAN-Bus sposobnost Raspberry Piju. Temeljena je na Microchip MCP2515 [7] CAN upravljaču s MCP2551 [8] CAN primopredajnikom. Spajanje na CAN mrežu može se vršiti preko DB9 konektora ili trodijelnog priključka na zavijanje. Spajanje Pican2 na Raspberry Pi izvodi se preko Raspberryjevog GPIO sučelja.



Slika 3.2.1 – Pican2 *Bus board*, gornja strana pločice

Na pločicu se mogu spojiti CAN kabel ili OBDII kabel (preko DB9 konektora), ili izravni signali pomoću žica, na priključak na zavijanje. Ovisno o kabelu spojenom na DB9 potrebno je uraditi određene korake opisane u Korisničkom priručniku pločice [6]. Također, pločica na sebi ima otpornik veličine 120Ω koji može služiti kao terminacijski otpornik. Na pločici postoji LED lampica spojena na GPIO22. Pican2 se s Raspberry Pijem spaja na GPIO priključke. Detaljnije značajke Pican2 pločice mogu se pronaći na poveznici reference [6].

3.3. „SocketCAN“ alat

SocketCAN paket je implementacija CAN protokola za Linux operacijski sustav u programskom jeziku „C“ [15]. SocketCAN koristi Berkeleyev priključnički[8] API, Linux mrežni stog i implementira CAN drajvere čvora kao mrežno sučelje. Ovaj API dizajniran je s ciljem što veće sličnosti TCP/IP protokolu kako bi se programerima koji su upoznatiji s mrežnim programiranjem, olakšalo korištenje CAN priključnica. [10]

[8] Hrv. Priključnica, Engl. *socket*

3.3.1. SocketCAN implementacija

Postoje mnoge implementacije CAN-a za Linux OS, no većina ih je zapravo drajver uređaja nekog hardvera i bazirane su na znakovnim implementacijama s ograničenom funkcionalnošću. Te implementacije sadrže samo sučelje za slanje i primanje CAN paketa dok se primjerice redovi poruka i viši transportni slojevi moraju implementirati u korisničkoj aplikaciji izvedenoj iz implementacije. Također, ovakve implementacije podržavaju postojanje samo jedne procesne dretve, poput serijskog sučelja.

SocketCAN je dizajniran da iskoristi podršku Linux mrežnog sloja kako bi nadišao ograničenja ranijih implementacija. Drajver uređaja za CAN upravljač registrira se u mrežnom sloju Linuxa kao mrežni uređaj što omogućava CAN paketima da se s upravljača prenesu u mrežni uređaj i modul CAN protokola, i obratno.

Budući da je ovo otvoreni projekt s platforme *GitHub* [9], to omogućuje razvojnim inženjerima da ga koriste kao temelj implementacije svog sustava za analizu CAN sabirnice i upravljanje CAN sabirnicom.

3.3.2. Koncept SocketCAN-a

Kao što je opisano u poglavlju 3.3.1, glavni cilj SocketCAN-a je, kao što samo ime kaže, pružiti priključničko sučelje korisničkoj aplikaciji koja se temelji na Linux mrežnom sloju. Za razliku od TCP/IP protokola, u CAN protokolu ne postoji adresiranje u smislu slanja poruke samo na određenu adresu, nego se adresa poistovjećuje s CAN ID-jem.

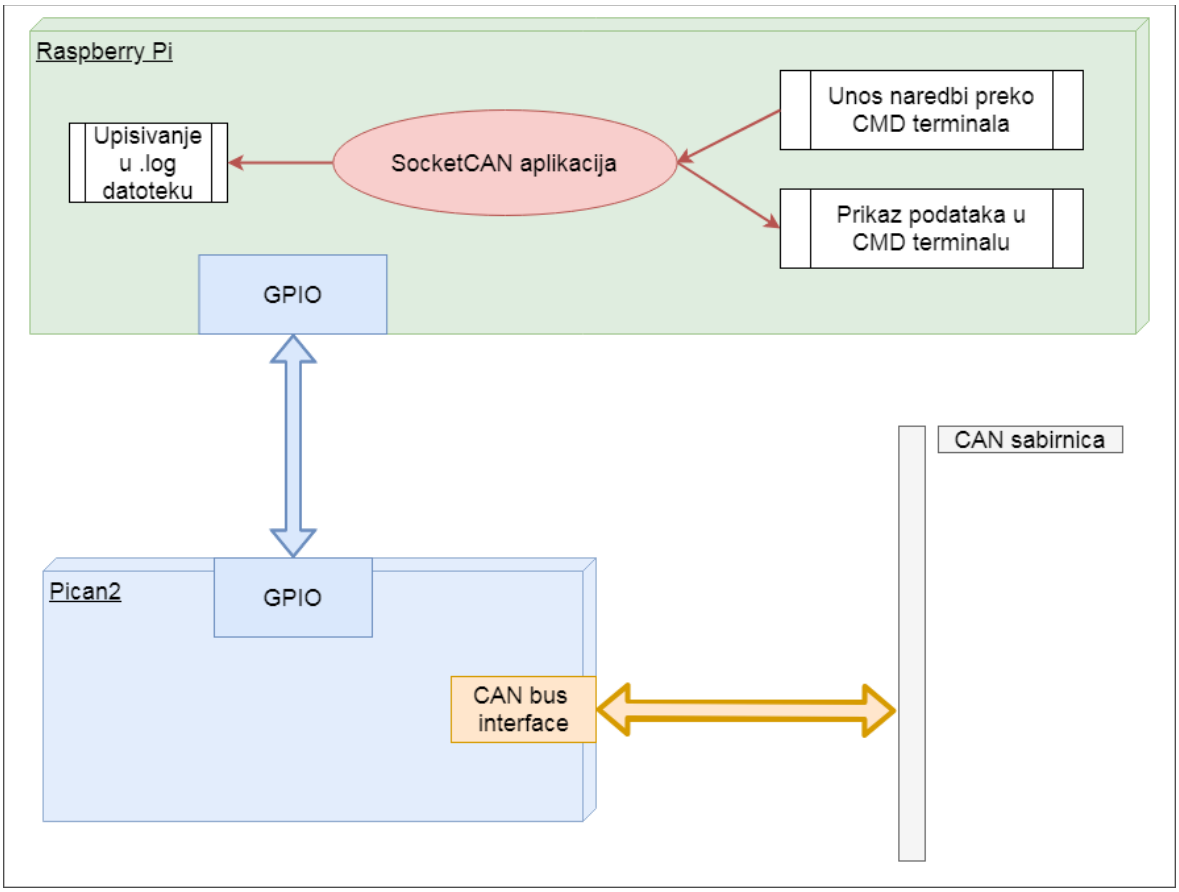
Koncept prijemnih lista rješava problem koji se pojavljuje kada više različitih aplikacija žele primiti poruku s istim CAN ID-jem preko istog CAN mrežnog sučelja. Na primjer, kada korisnička aplikacija otvori CAN *RAW socket*, *raw* protokol zatraži popis CAN ID-jeva koje traži korisnik, od SocketCAN jezgre.

Nadalje, SocketCAN implementira funkcionalnost povratne veze poslanih paketa. Zbog arbitracije na sabirnici, može se dogoditi da slanje manje prioritetne poruke može biti zakašnjeno prijemom prioritetnije poruke sa sabirnice. Potrebno je povratnom petljom poslati podatke odmah nakon uspješne transmisije kako bi se osigurao ispravan promet na čvoru. Ako CAN mrežno sučelje ne može slati podatke povratnom petljom, to za njega može učiniti SocketCAN kroz implementaciju *fallback* funkcije.

3.4. Implementacija rješenja

Na slici 3.4.1 može se vidjeti shema sustava. Podaci s CAN sabirnice primaju se pomoću pločice Pican2. Pican2 podatke dohvaća preko sučelja s CAN-om, koje može biti DB9 konektor ili trodijelni priključak na zavijanje. Pican2 je spojen s Raspberry Pijem preko GPIO pinova, preko kojih se vrši prijenos podataka i upravljanje.

Na Raspberry piju se izvodi aplikacija koja korisniku omogućava unos naredbi preko naredbenog ekrana Linux operacijskog sustava i prikaz podataka na istom naredbenom ekranu. Aplikacija također može upisivati podatke u .log datoteku, ako je korisnik tako odredio naredbom. [10]



Slika 3.4.1 – Shematski prikaz sustava

3.4.1. Korištenje SocketCAN alata

Slično kao TCP/IP protokol, na početku je potrebno otvoriti priključnicu za komunikaciju preko CAN sabirnice. Sustavskom pozivu funkcije koja instancira priključnicu potrebno je proslijediti određeni CAN protokol – *„raw socket“* ili *„BCM - broadcast manager“*. Nakon spajanja priključnice na CAN sučelje moguće je slati i primati CAN poruke.

Izgled strukture CAN paketa i definicija CAN_ID tipa podatka u kodu:

- CAN paket:

```
struct can_frame {
            canid_t can_id;  /* 32 bit CAN_ID + EFF/RTR/ERR flags */
            __u8    can_dlc; /* frame payload length in byte (0 .. 8) */
            __u8    __pad;   /* padding */
            __u8    __res0;  /* reserved / padding */
            __u8    __res1;  /* reserved / padding */
            __u8    data[8] __attribute__((aligned(8)));
    };
```

- Definicija CAN_ID-a:

```
/*
        * Controller Area Network Identifier structure
        *
        * bit 0-28 : CAN identifier (11/29 bit)
        * bit 29   : error message frame flag (0 = data frame, 1 = error
       message)
        * bit 30   : remote transmission request flag (1 = rtr frame)
        * bit 31   : frame format flag (0 = standard 11 bit, 1 = extended
       29 bit)
        */
       typedef __u32 canid_t;
```

Struktura *sockaddr_can* ima indeks sučelja poput PF_PACKET priključnice, koji se također veže na određeno sučelje. Kako bi se odredio indeks sučelja, treba iskoristiti prikladni poziv sistemske funkcije *ioctl()*, sa zahtjevom za `SIOCGIFINDEX`. Za povezivanje priključnice na sva CAN sučelja, indeks sučelja treba imati vrijednost '0'. U ovom slučaju priključnica prima CAN pakete od svih CAN sučelja. Također, korištenjem sistemske funkcije *ioctl()* može se dobiti i vremenska oznaka preciznosti jedne mikrosekunde, koristeći zahtjev za `SIOCGSTAMP`.

```
        sa_family_t can_family;
            int          can_ifindex;
            union {
                    /* transport protocol class address info (e.g. ISOTP)
*/
                    struct { canid_t rx_id, tx_id; } tp;

                    /*   reserved   for   future   CAN   protocols   address
information */
            } can_addr;
    };
```

3.4.2. Priključnice *RAW* protokola s raznim filtrima

Prilikom vezanja CAN_RAW priključnice postavljaju se neki zadani filtri priključnice, koji se mogu promijeniti prije ili nakon vezanja priključnice:

- Početni filtar je točno jedan filtar koji prihvaća sve poruke
- Priključnica prihvaća samo ispravne pakete (bez paketa s pogreškom)
- Povratna veza CAN paketa je uključena
- Priključnica ne prima svoje vlastite poslane pakete

RAW protokol - *CAN_RAW_FILTER*

Prihvat CAN poruka koristeći CAN_RAW priključnice može se upravljati određivanjem ni jednog ili više filtara s *CAN_RAW_FILTER* opcijom.

Struktura CAN filtra:

```
struct can_filter {
            canid_t can_id;
            canid_t can_mask;
    };
```

Filtar se poklapa kada:

```
<received can_id> & mask == can_id & mask
```

Analogno tome čvorovi CAN-a hardverski filtriraju poruke koje ih zanimaju ili ne zanimaju. Više filtara se pak postavlja definicijom polja *can_filter* strukture.

RAW protokol - *CAN_RAW_ERR_FILTER*

Poruke s pogreškom se također mogu proslijediti korisničkoj aplikaciji. Pojedini tipovi pogrešaka predefinirani su i mogu se filtrirati pojedinačno, koristeći pripadajuće konstante. Primjer propuštanja svih pogrešaka:

```
can_err_mask_t err_mask = CAN_ERR_MASK;
setsockopt(s, SOL_CAN_RAW, CAN_RAW_ERR_FILTER,
           &err_mask, sizeof(err_mask));
```

RAW protokol - *CAN_RAW_LOOPBACK*

U nekim slučajevima, implementacija ugradbenog sustava zahtjeva da samo jedna aplikacija koristi sabirnicu i tada povratna petlja nije potrebna. Ona se može ugasiti za svaku priključnicu pojedinačno. Također, prihvat vlastitih poruka čvora također može biti omogućen.

RAW protokol - *CAN_RAW_JOIN_FILTERS*

Ova mogućnost priključnice omogućava da se propuštaju samo one poruke koje „prolaze“ kroz sve filtre, analogno logičkom operatoru „I“ (engl. „*AND*“).

3.4.3. Priključnice *Broadcast manager*[9] protokola

Ovaj protokol pruža naredbeno sučelje za filtriranje i slanje CAN poruka u jezgrenom prostoru. Filtri primljenih poruka mogu se iskoristiti da bi se smanjila frekvencija ispisa poruka koje stižu vrlo često.

BCM protokol nije namijenjen slanju pojedinačnih CAN paketa korištenjem ranije definirane *can_frame* strukture. Za slanje se koristi posebna BCM konfiguracijska poruka.

[9] Skraćeno BCM

Takva se poruka sastoji od zaglavlja s naredbom i CAN paketa. Jednako izgleda i primljena struktura podataka:

```
struct bcm_msg_head {
            __u32 opcode;                   /* command */
            __u32 flags;                    /* special flags */
            __u32 count;                       /* run 'count' times with
ival1 */
            struct timeval ival1, ival2;       /* count and subsequent
interval */
            canid_t can_id;                 /* unique can_id for task */
            __u32 nframes;                        /* number of can_frames
following */
            struct can_frame frames[0];
    };
```

BCM operacije

'opcode' jedinstveno određuje operaciju koju će BCM izvršiti ili detalje BCM-ova odziva na određene događaje, uključujući korisničke događaje.

Operacije slanja (korisnički prostor → BCM):

`TX_SETUP` – stvori (kružni) zadatak slanja.

`TX_DELETE` – obriše (kružni) zadatak slanja, traži samo *can_id.*

`TX_READ` – pročita svojstva (kružnog) zadatka slanja za *can_id.*

`TX_SEND` – pošalje 1 CAN paket.

Odgovori na slanje (BCM → korisnički prostor):

`TX_STATUS` – odgovor na `TX_READ` zahtjev.

`TX_EXPIRED` – obavijest kada brojač završi sa slanjem u početnom intervalu 'ival1'. Zahtjeva da `TX_COUNTEVT` zastavica bude postavljena tijekom `TX_SETUP`.

Operacije primanja (korisnički prostor → BCM):

`RX_SETUP` – stvori pretplatu na filtar sadržaja prilikom primanja.

`RX_DELETE` – obriše pretplatu na filtar sadržaja prilikom primanja, traži samo *can_id.*

`RX_READ` – pročita svojstva filtra sadržaja za *can_id.*

Odgovori na primanje (BCM → korisnički prostor):

`RX_STATUS` – odgovor na `RX_READ` zahtjev.

RX_TIMEOUT – otkriveno je da kružna poruka nedostaje (brojač *ival1* istekao).

RX_CHANGED – BCM poruka s osvježenim CAN paketom. Šalje se kada je primljena prva poruka ili kada je primljena revidirana poruka.

BCM - zastavice poruka

Prilikom slanja poruke BCM-u, u32 objekt *flags* može sadržavati sljedeće definicije zastavica:

SETTIMER – postavi vrijednost *ival1* , *ival2* i *count*.

STARTTIMER – pokreni brojanje sa stvarnim vrijednostima *ival1*, *ival2* i *count*. Započinjanje brojanja istovremeno dovodi do odašiljanja CAN paketa.

TX_COUNTEVT – stvori poruku TX_EXPIRED kad brojač izbroji do kraja.

TX_ANNOUNCE – promjena podataka procesa emitira se odmah.

TX_CP_CAN_ID – kopira *can_id* iz zaglavlja poruke u svaki sljedeći paket strukture *frames*.

RX_FILTER_ID – filtar prema *can_id*-ju, *nframes=0* (paketi nisu potrebni).

RX_CKECK_DLC – promjena DLC-a uzrokuje RX_CHANGED.

RX_NO_AUTOTIMER – spriječi automatsko pokretanje monitora za *timeout.*

RX_ANNOUNCE_RESUME – ako se proslijedi tijekom RX_SETUP-a i ako se dogodio *timeout*, RX_CHANGED poruka će se generirati prilikom ponovnog započinjanja (kružnog) prijema.

RX_RESET_MULTI_IDX – resetira indeks za slanje višestrukih paketa.

RX_RTR_FRAME – pošalje odgovor na RTR zahtjev

BCM – brojači intervala periodičkog slanja

Konfiguracije periodičkog slanja mogu koristiti do dva brojača intervala. U tom slučaju, BCM šalje poruke u intervalu *ival1*, a nakon određenog broja poruka mijenja interval u *ival2*. broj poruka koje se šalju u intervalu *ival1* određen je brojem *count*. U slučaju samo jednog brojača *count* je postavljen na '0' i koristi se samo *ival2* brojač. Brojači broje kada su postavljene zastavice SET_TIMER i START_TIMER. Vrijednosti brojača mogu se mijenjati tijekom izvršavanja ako je postavljena **samo** zastavica SET_TIMER.

U slučaju kružne TX konfiguracije, čak do 256 poruka može biti poslano u jednom slijedu. Broj poruka zadak je poljem *nframes* u zaglavlju BCM poruke.

3.4.4. Virtualni CAN drajver (*vcan*)

'vcan' pruža lokalno virtualno CAN sučelje. Budući da se puna adresa uređaja sastoji od CAN ID-ja i same sabirnice na koju se poruka šalje (npr. can0), obično je potrebno više od jednog virtualnog CAN sučelja.

Virtualno CAN sučelje dozvoljava slanje i primanje CAN paketa bez stvarnog CAN upravljačkog sklopovlja. Od verzije Linux jezgre 2.6.24 [16], moguće je kreirati *vcan* mrežne uređaje koristeći *ip* alat.

Primjer kreiranja virtualnog CAN mrežnog sučelja:

ip link add type vcan

Primjer kreiranja i brisanja virtualnog CAN mrežnog sučelja sa specifičnim imenom '*vcan42*':

ip link add dev vcan42 type vcan

ip link del vcan42

3.4.5. Sučelje CAN mrežnog drajvera uređaja

Ovo sučelje pruža generično sučelje za postavljanje, konfiguraciju i nadgledanje CAN mrežnih uređaja. Korisnik može konfigurirati CAN uređaj, na primjer postaviti vremenske parametre bitova, preko netlink sučelja, koristeći *ip* uslužni paket.

Ispis pomoći pri korištenju *ip* uslužnog paketa:

```
$ ip link set can0 type can help
    Usage: ip link set DEVICE type can
        [ bitrate BITRATE [ sample-point SAMPLE-POINT] ] |
        [ tq TQ prop-seg PROP_SEG phase-seg1 PHASE-SEG1
          phase-seg2 PHASE-SEG2 [ sjw SJW ] ]

        [ dbitrate BITRATE [ dsample-point SAMPLE-POINT] ] |
        [ dtq TQ dprop-seg PROP_SEG dphase-seg1 PHASE-SEG1
          dphase-seg2 PHASE-SEG2 [ dsjw SJW ] ]

        [ loopback { on | off } ]
        [ listen-only { on | off } ]
        [ triple-sampling { on | off } ]
        [ one-shot { on | off } ]
        [ berr-reporting { on | off } ]
```

```
[ fd { on | off } ]
[ fd-non-iso { on | off } ]
[ presume-ack { on | off } ]

[ restart-ms TIME-MS ]
[ restart ]

Where: BITRATE       := { 1..1000000 }
       SAMPLE-POINT  := { 0.000..0.999 }
       TQ            := { NUMBER }
       PROP-SEG      := { 1..8 }
       PHASE-SEG1    := { 1..8 }
       PHASE-SEG2    := { 1..8 }
       SJW           := { 1..4 }
       RESTART-MS    := { 0 | NUMBER }
```

Koristeći ovaj uslužni paket mogu se vidjeti detalji i statistike CAN uređaja, postaviti njegove vremenske parametre bitova te pokrenuti i zaustaviti CAN mrežni uređaj.

3.4.6. Funkcije za analizu toka podataka te praćenje, prikaz i pohranu podataka

Sustav za analizu, praćenje, pohranu i prikaz podataka implementiran je kroz 3 funkcije od kojih svaka pruža dio funkcionalnosti opisane zadatkom ovog rada. Funkcije kroz koje je implementiran sustav su: *canbusload, cansniffer i candump*.

Candump

Candump funkcija obuhvaća dio funkcionalnosti koji ispisuje na ekran i/ili sprema CAN poruke u .log datoteku, u stvarnom vremenu. Format spremljenih podataka sadrži sučelje s kojeg je primljena poruka, vremensku oznaku primitka, ID poruke i podatkovni dio poruke.

Cansniffer

Cansniffer funkcija obuhvaća dio funkcionalnosti koja ispisuje na ekran poruke poslane na sučelje od strane BCM-a. Format ispisa sadrži vremensku razliku posljednjeg primitka poruke istog ID-ja, ID poruke i podatkovni dio. Također, u ispisu je moguće vidjeti broj primljenih paketa s pogreškom i ukupni broj pristiglih poruka.

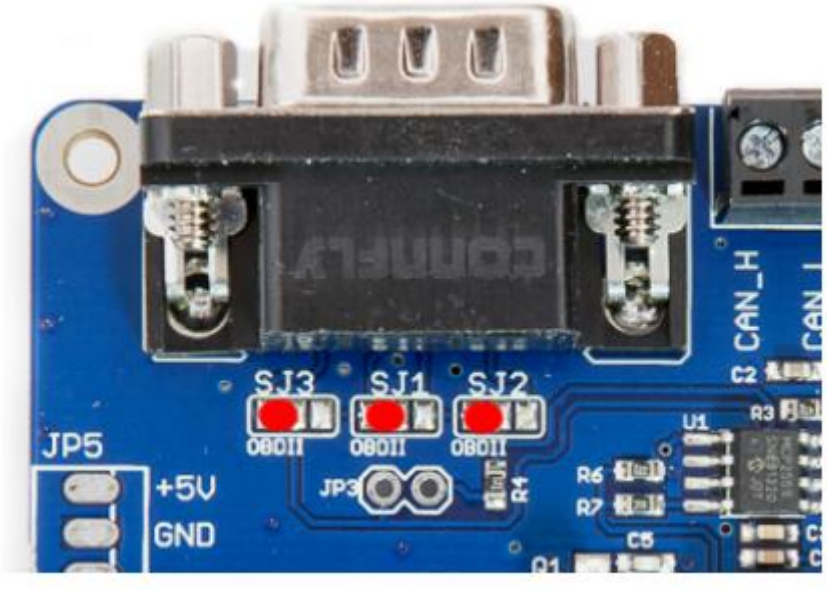
Canbusload

Canbusload funkcija obuhvaća dio funkcionalnosti koji korisniku prikazuje analizu toka podataka s periodičkim osvježenjem podataka. Ova funkcija prikazuje broj primljenih poruka u vremenu, ukupnu količinu bitova u tim porukama, količinu korisnih bitova u tim porukama, postotak zauzeća sabirnice i broj primljenih pogrešaka.

Sve tri funkcije se mogu pokrenuti u isto vrijeme, no ne u istom naredbenom prozoru Linux sustava. Funkcije imaju mogućnost konfiguracije ispisa i obrade podataka s CAN sabirnice, pa tako omogućuju npr. filtriranje poruka na razini pojavljivanja pogreške ili filtriranje na razini ID-ja i različite formate ispisa podataka na ekranu.

3.5. Upute za instalaciju

Za početak, potrebno je sastaviti sustav u smislu pripreme i spajanja Pican2 pločice na Raspberry Pi. Korisnički priručnik Pican2 pločice nalaže lemljenje, ovisno o korištenom kabelu, određenih lemnih mostova. Budući da se u radu koristi OBD2 kabel, potrebno je zalemiti mostove kao na slici 3.5.1.



Slika 3.5.1 – Lemni mostovi koje je potrebno zalemiti

Nadalje, na pločici se nalazi otpornik veličine 120Ω koji se koristi kao terminacijski otpornik, no kako bi ga se koristilo potrebno je kratko spojiti točke (JP3[10]) prikazane na slici 3.5.2.

Slika 3.5.2 – JP3 na Pican2 pločici

Terminacija se koristi iz dva razloga:

1) Poboljšanje kvalitete signala – terminacija sprječava refleksiju signala koja može biti izrazito snažna kada signal dolazi do kraja kabla.
2) Osiguravanje recesivne razine signala – određuje logičku '1' u komunikaciji

Kako bi se definirala razina signala tijekom slanja recesivnog bita, potrebno je koristiti pasivnu komponentu – najčešće otpornik. Otpornik može „pritegnuti“ napon na određenu razinu. Vrijednost otpornika treba odgovarati impedanciji sabirnice. Korišteni otpornik je dovoljno dobar za komunikaciju s brzinama promjene 50 nanosekundi i 1 Mbit/s brzine prijenosa podataka. [11]

Nakon što je Pican2 pločica spremna za uporabu, potrebno ju je pažljivo spojiti s Raspberry Pijem. Prije spajanja oba uređaja trebaju biti isključena i konektori trebaju biti pažljivo poravnati kako se neki od pinova ne bi slomio. Poželjno je osigurati pločice koristeći prikladne odstojnike.

[10] *Jumper port* 3

Slika 3.5.3 – Raspberry Pi spojen s Pican2 pločicom

Na Raspberry Pi potrebno je instalirati operacijski sustav Raspbian [12]. Raspbian je službeni operacijski sustav zaklade koja je razvila Raspberry Pi.

Nakon instalacije i prvog podizanja sustava, potrebno je instalirati alate koji će omogućiti dohvat izvornog koda SocketCAN-a s GitHub-a. potrebno je pokrenuti sljedeće naredbe:

sudo apt-get update
sudo apt-get upgrade
sudo reboot

Nakon ponovnog podizanja sustava potrebno je instalirati Git [13], sustav za verzioniranje koristeći naredbe:

sudo apt-get install git-core
sudo apt-get install git

Za korištenje Pican2 pločice, na kraj datoteke **./boot/config.txt** potrebno je dodati sljedeće 3 linije:
dtparam=spi=on
dtoverlay=mcp2515-can0,oscillator=16000000,interrupt=25
dtoverlay=spi-bcm2835-overlay

- Također, ako je potrebno, iz ***raspi-blacklist.conf*** datoteke potrebno je izbaciti mikročip korišten u projektu.

Nadalje, u željenu datoteku treba klonirati projekt s GitHub-a koristeći sljedeću naredbu:
git clone https://github.com/Balrog6236/SocketCAN-FER

Kako bi SocketCAN bio spreman za upotrebu potrebno je još instalirati „autoconf“ i „libtool“, koji su nužni za automatsku instalaciju alata korištenjem ugrađene skripte. Sustav i navedene biblioteke instaliravaju se pokretanjem sljedećih naredbi (potrebno se pozicionirati u **~./can-utils**):

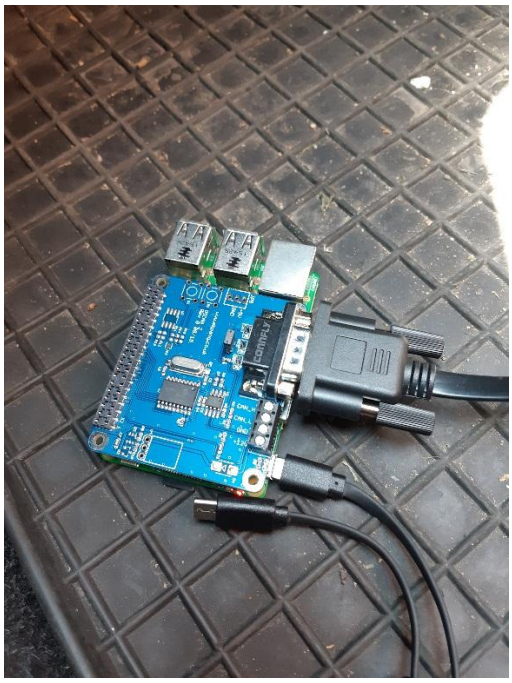
```
sudo apt-get install autoconf
sudo apt-get install libtool
./autogen.sh
./configure
make
sudo make install
```

Za ostvarenje konekcije i upravljanje Raspberry Pijem, korišten je alat zvan „VNC Viewer“ [14]. On omogućuje korisniku npr. stolnog računala, da se spoji (na primjer preko WiFi-ja) i upravlja Raspberry Pijem kroz upravljački ekran. [6], [9]

3.6. Upute za korištenje

Prije korištenja aplikacije sustava, potrebno je ispravno spojiti OBDII-DB9 kabel sa sustavom kojeg se želi promatrati (npr. automobil). Zatim je potrebno dovesti napajanje Raspberry Piju, nakon čega se pločica uključi.

Slika 3.6.1



Slika 3.6.2

Slike 3.6.1 i 3.6.2 – OBDII priključak automobila, s kabelom i Pican2 pločica spojena na OBDII

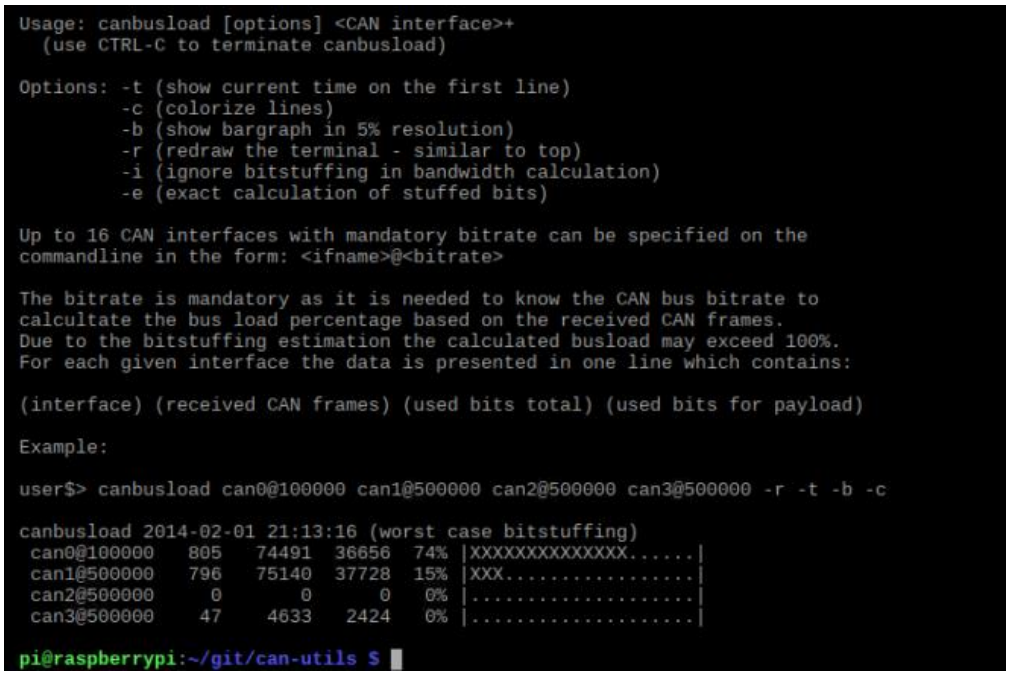
Ostvarivši konekciju s Raspberry Pijem, pomoću VNC viewer alata, korisnik sada može upravljati Raspberry Pi pločicom. Za podizanje CAN sučelja preko stvarnog hardvera potrebno je pokrenuti sljedeću naredbu:

sudo ip link set can0 up type can bitrate 500000

U naredbi je potrebno navesti brzinu sabirnice koja se koristi u sustavu (u slučaju CAN-a u automobilu to je najčešće 500 Kbit/s).

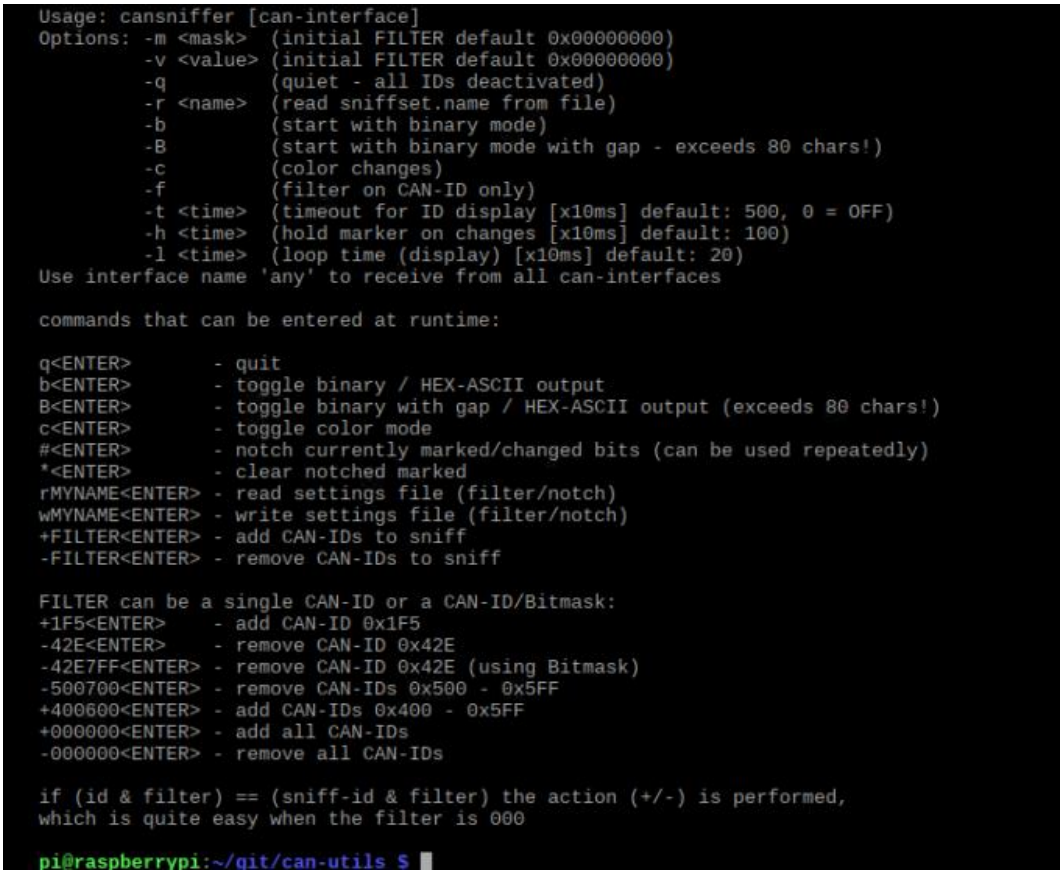
Za implementaciju sustava korištene su ranije navedene i objašnjene funkcije. Način korištenja može se dobiti pozivanjem pojedine funkcije bez ikakvih parametara, na primjer `./canbusload`. Analogija vrijedi i za ostale dvije korištene funkcije.

Upute za korištenje *canbusload* funkcije vide se na slici 3.6.3.



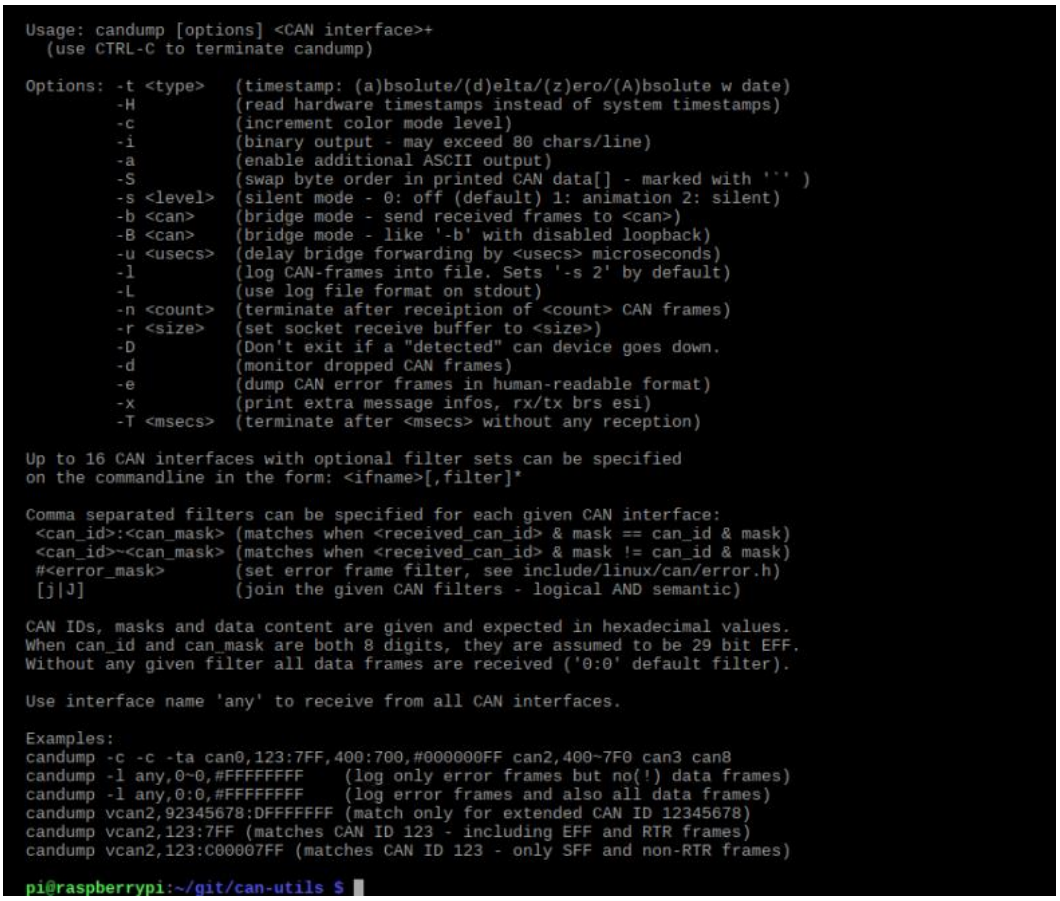
Slika 3.6.3 – Korištenje funkcije *canbusload*

Upute za korištenje *cansniffer* funkcije vide se na slici 3.6.4.



Slika 3.6.4 – Korištenje *cansniffer* funkcije

Upute za korištenje *candump* funkcije vide se na slici 3.6.5.



Slika 3.6.5 -Korištenje *candump* funkcije

Za poziv pojedine funkcije potrebno je prvo se pozicionirati u mapu u kojoj je instaliran alat. Nakon ispravnog pozicioniranja, uz korake opisane u poglavlju 3.5 – „Upute za instalaciju“, funkcija se poziva koristeći Linux poziv iz naredbenog sučelja (ekrana). Na primjer, za poziv funkcija s određenim sučeljem, a bez ostalih zastavica, mogu se iskoristiti sljedeće naredbe:

./canbusload can0@500000

./candump can0

./cansniffer can0

- Ove naredbe dat će osnovan, zadani izlaz definiran u implementacijama funkcija, za CAN sučelje *can0*.

Funkcije imaju i mogućnost dodatne prilagodne. Za korištenje naprednih mogućnosti potrebno je definirati dodatne zastavice prilikom poziva funkcije. Neki od primjera korištenja funkcija s dodatnim zastavicama:

./canbusload -t -c -b -r can0@500000

- Ova naredba ispisivat će trenutno vrijeme u prvoj liniji (argument *-t*), obojat će ispisane podatke u zadanu boju (argument *-c*), slikovito će prikazati postotak zauzetosti sabirnice (argument *-b*) i svaki novi ispis prebrisat će stari (argument *-r*). Sve ovo ispisivat će se za sučelje *can0* s brzinom prijenosa 500 KBit/s.

./candump -l can0,0:0,#FFFFFFFF

- Ova naredba spremat će sve primljene poruke s can0 sučelja u datoteku (argument *-l*). Filtar (0:0,#FFFFFFFF) definirani kao argument funkcije proslijedit će sve poruke koje stignu preko sučelja.

./candump -l any,0~0,#FFFFFFFF

- Ova naredba spremat će sve primljene poruke, s bilo kojeg sučelja (argument *any*), u datoteku (argument *-l*). Filtar (0~0,#FFFFFFFF) definiran kao argument funkcije proslijedit će sve poruke s pogreškom.

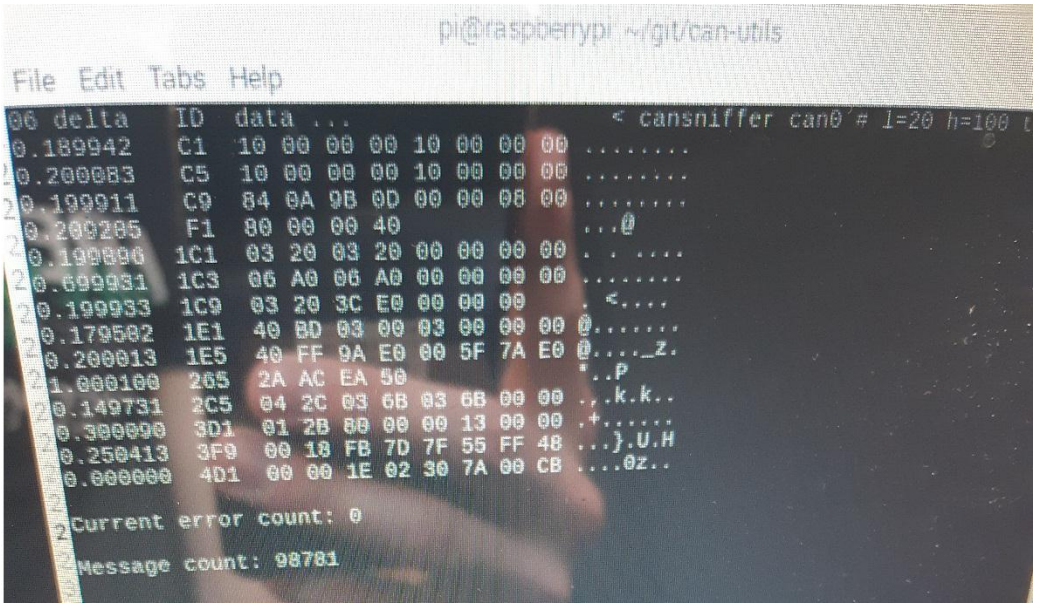
./candump -t A -c any

- Ova naredba prikazivat će podatke u pozvanom naredbenom prozoru s apsolutnom vremenskom oznakom koja uključuje datum i vrijeme (argument „*-t A*“). Također, svako sučelje bit će obojano drugačijom bojom, radi lakšeg raspoznavanja poruka (argument *-c*).

3.7. Testiranje sustava i mogućnosti poboljšanja

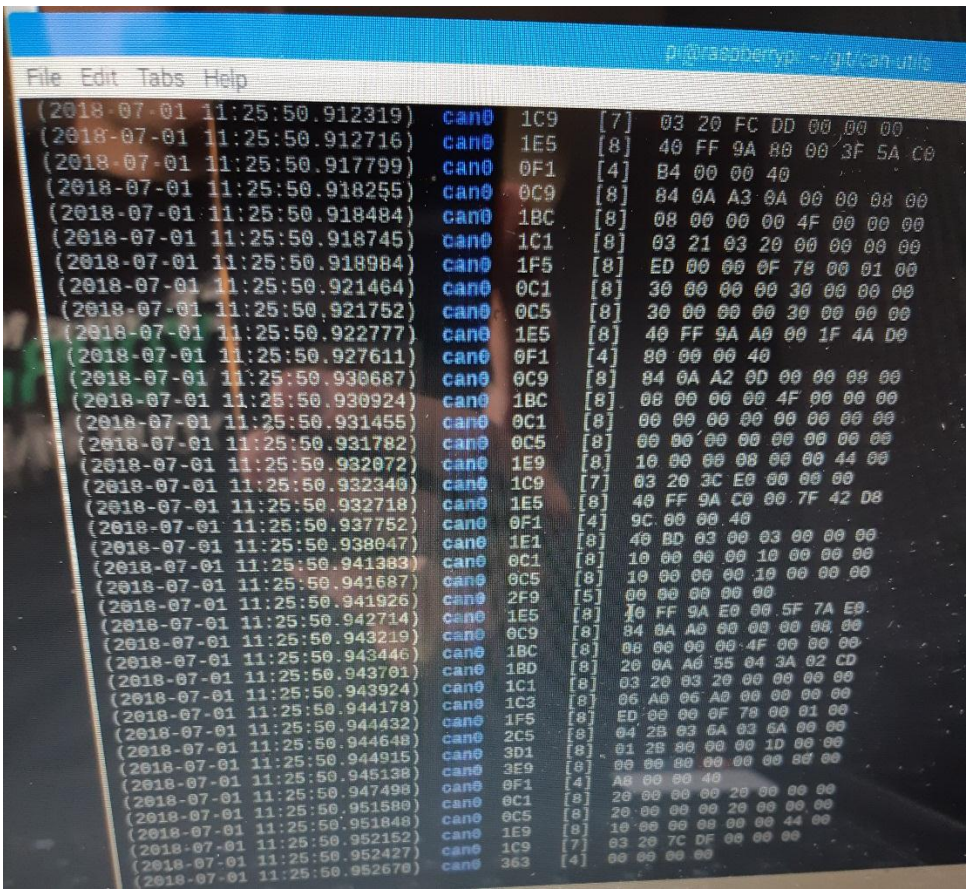
U radu je za testiranje rada sustava korišten OBDII konektor automobila Opel Corsa, godišta između 2006.-2014. Sustav je na ispravan način instaliran i postavljen, te su pozvane sljedeće funkcije:

a) Poziv: ***./cansniffer can0***



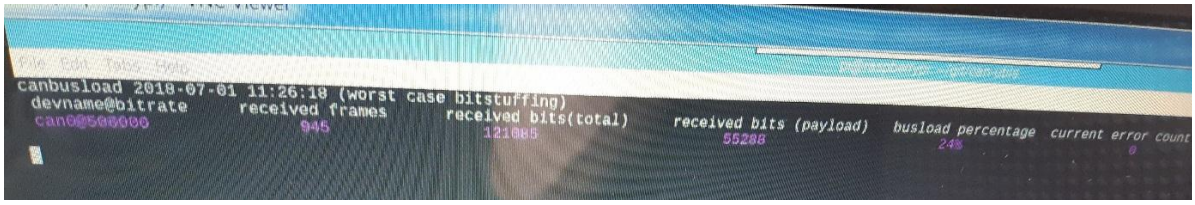
Slika 3.7.1 – Ispis pozvane *cansniffer* funkcije

b) Poziv: ***./candump can0 -c -t A***



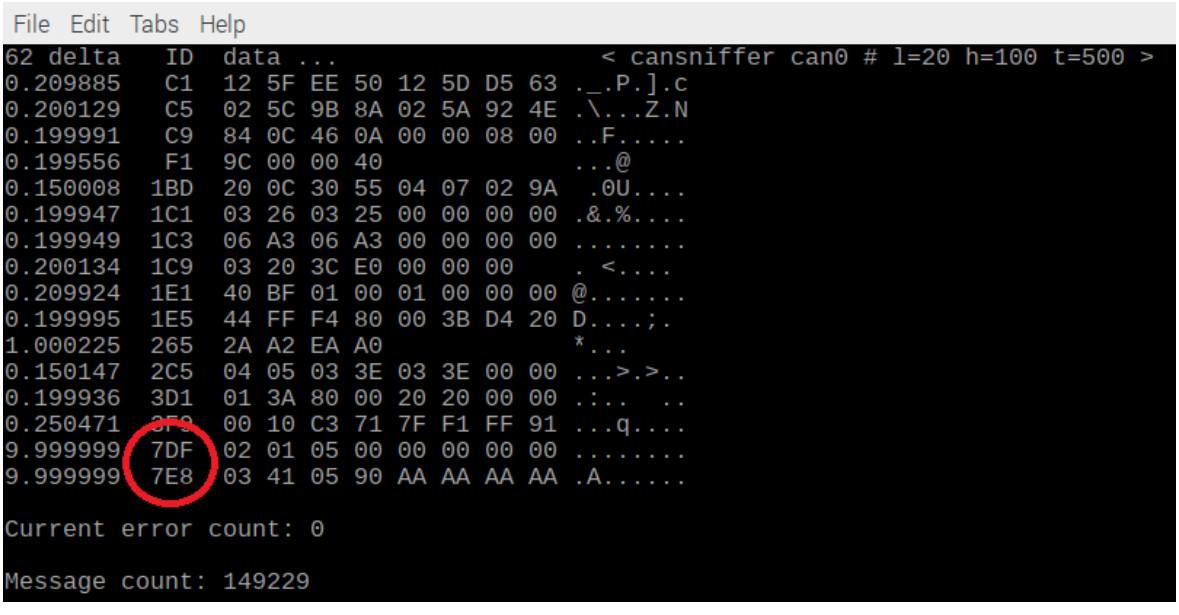
Slika 3.7.2 – Ispis pozvane *candump* funkcije

c) Poziv: ***./canbusload -t -c -b -r can0@500000***



Slika 3.7.2 – Ispis pozvane *canbusload* funkcije

Pozivom i proučavanjem ispisa funkcija dolazi se do zaključka da sustav za analizu, prikaz i spremanje podataka s CAN sabirnice radi ispravno. U Svrhu dodatnog testiranja, koristeći alat za slanje CAN poruka *cansend*, tijekom rada *cansniffer* funkcije, poslana je poruka „7DF#0201050000000000“, koja predstavlja zahtjev za očitanjem temperaturnog senzora automobila. Poruka i njen odgovor (paket s ID-jem „7E8“) vide se u ispisu *cansniffer* funkcije, što dodatno potvrđuje ispravnost rada sustava.



Slika 3.7.3 – Poruka s ID-jem 0x7DF i odgovor na nju (ID 0x7E8)

3.7.1. Dodatno unaprjeđenje sustava

Iako implementirani sustav implementira sve potrebne funkcionalnosti, postoje očiti načini dodatnog poboljšanja. Sustav trenutno ispisuje samo „sirove“ podatke na odabrane izlaze, što korisniku ne daje do znanja koje podatke čita, niti koji je iznos tih podataka. Sustav bi se dao nadograditi na način da se dodaju funkcije za prepoznavanje tipa podatka iz ID-ja primljenog paketa te da se dodaju funkcije koje će iz podatkovnog polja „dekodirati“ stvarnu vrijednost podatka iz primljenih

bitova. Također, *canbusload* funkcija bi se dala unaprijediti na način da nije potrebno eksplicitno kao argument navesti CAN sučelje i brzinu prijenosa bitova na njemu, već da sustav to odradi sam.

4. Zaključak

U ovom radu opisane su značajke i način rada CAN protokola. CAN protokol na podatkovnom sloju osigurava:

- *Multi-master* hijerarhiju
- *Broadcast* komunikaciju
- Mehanizme za otkrivanje, signalizaciju i ponovno slanje oštećenih poruka

Opisan je način kodiranja bitova u CAN protokolu te postavke vremena bitova, koji tvore nužni preduvjet postojanja ispravne komunikacije. Opisani su i formati poruka koje CAN protokol podržava, a koji se definiraju tijekom postavljanja sustava. To su: CAN s 11-bitovnim ID-jem i CAN s 29-bitovnim ID-jem. Nadalje, opisana su sva polja jednog paketa CAN poruke te je opisana funkcionalnost svakog od njih.

S obzirom da se radi o *multi-master* arhitektuti, potrebno je bilo i opisati sustav arbitraže sabirnice. On se temelji na prioritetima ID-jeva koji pristupaju sabirnici – najniži ID ima najviši prioritet. Naposljetku, opisani su načini pronalaženja pogrešaka, kao i stanja u kojima se pojedini čvorovi, s obzirom na broj pogrešaka, mogu nalaziti.

Nadalje, opisani su principi CANopen komunikacijskog sustava te je odabran jedan takav sustav, implementiran za rad na Linux operacijskom sustavu, kao temelj implementacije sustava za praćenje prometa na CAN sabirnici. Objašnjena je potreba za implementacijom i korištenjem takvog sustava za praćenje, prikaz i analizu poruka i toka podataka na CAN sabirnici.

Sustav je implementiran na Raspberry Pi računalu uz dodatnu pločicu – Pican2, koja pruža podršku Raspberry Piju da se ponaša kao čvor u CAN mreži. Pri implementaciji je korišten SocketCAN alat pomoću kojeg su podaci sa sabirnice pročitani, prikazani, spremljeni i analizirani.

Sama implementacija izvedena je kroz 3 funkcije: *canbusload, candump* i *cansniffer*. One pružaju podršku za sve funkcionalne zahtjeve opisane u zadatku rada, a pozivaju se kroz upravljački prozor Linux operacijskog sustava.

Naposljetku, detaljno je opisano kako se sustav za analizu CAN sabirnice instalira, postavlja i koristi, uz nekoliko primjera korištenja funkcija i ispisa rezultata na ekranu korisnika.

Sustav za analizu, spremanje, ispis i prikaz podataka na CAN sabirnici iznimno je koristan, čak i nužan, u razvoju sustava koji komuniciraju pomoću CAN sabirnice, a nemaju mogućnost prikaza komunikacije u obliku čitljivom razvojnom inženjeru. Korisnik u implementiranom sustavu ima relativno visoku mogućnost personalizacije prikaza i rada funkcija koja mu omogućava da jednostavnije i čitljivije testira sustav koji razvija ili analizira.

Sustav ima i prostora za poboljšanje, u što primarno ulazi mogućnost nadogradnje SocketCAN-a da prepoznaje i prikazuje tip podataka pomoću primljenih ID-jeva poruka, a uz to i da dekodira podatkovni dio paketa i prikazuje stvarnu reprezentaciju podatka u domeni čitljivoj korisniku.

Sustav je implementiran uspješno, a rezultati koje on daje su zadovoljavajući. Iako postoji mogućnost poboljšanja, u trenutnom stanju implementacije sustav je iskoristiv i uspješno obavlja zadaće za koje je namijenjen.

5. Literatura

[1] Ivanuš K. Sustav za praćenje i analizu prijenosa podataka putem CAN komunikacijskih sabirnica tračničkih vozila, magistarski rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb, siječanj 2009.

[2] Introduction to the Controller Area Network (CAN) (Rev. B), http://www.ti.com/lit/an/sloa101b/sloa101b.pdf

[3] CAN in Automation (CiA) Official page, https://www.can-cia.org/can-knowledge/can/can-data-link-layers/

[4] Bosch CAN User's Guide for Cygnal Devices, http://www.keil.com/dd/docs/datashts/silabs/boschcan_ug.pdf

[5] Šestok J. Izvršavanje glasovnih naredbi pomoću Raspberry Pi računala, diplomski seminar, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb, 2016./2017.

[6] Pican2 CAN-Bus pločica - http://skpang.co.uk/catalog/pican2-canbus-board-for-raspberry-pi-23-p-1475.html

[7] MCP2515 datasheet - http://ww1.microchip.com/downloads/en/DeviceDoc/20001801H.pdf

[8] MCP2551 datasheet - https://www.sparkfun.com/datasheets/DevTools/Arduino/MCP2551.pdf

[9] GitHub razvojna platforma - https://github.com/

[10] SocketCAN dokumentacija - https://www.kernel.org/doc/Documentation/networking/can.txt

[11] *Using termination to ensure recessive bit transmission*, Kvaser - https://www.kvaser.com/using-termination-ensure-recessive-bit-transmission/

[12] Raspbian OS za RPi - https://www.raspberrypi.org/downloads/raspbian/

[13] git-scm službena stranica - https://git-scm.com/

[14] RealVNC homepage, download link for VNC viewer - https://www.realvnc.com/en/connect/download/viewer/

[15] Službena stranica razvojne skupine programskog jezika C - http://www.open-std.org/jtc1/sc22/wg14/

[16] Linux ketnel 2.6.24 release - https://mirrors.edge.kernel.org/pub/linux/kernel/v2.6/

6. Sažetak

Korištenje raznih komunikacijskih protokola uvelike olakšava komunikaciju i razmjenu podataka. Jedan od komunikacijskih protokola je i CAN protokol koji je iznimno popularan u autoindustriji i šire. Kod stavljanja u pogon sustava koji koriste CAN komunikaciju potrebno je na neki način analizirati podatke i tok podataka. U ovom radu potrebno je istražiti i opisati specifikacije CAN protokola. Cilj ovog rada je razviti sustav za neinvazivnu analizu i praćenje, pohranu i prikaz podataka na CAN sabirnici. Sustav je potrebno implementirati u operacijskom sustavu Linux, na Raspberry Pi računalu. Potrebno je opisati kako taj sustav instalirati i koristiti.

Ključne riječi: komunikacijski protokol, CAN protokol, CANopen, sustav za analizu, Raspberry Pi, Linux

7. Abstract

The use of various communication protocols greatly facilitates communication and data exchange. One of the communication protocols is CAN protocol, which is extremely popular in automotive industry and beyond. When starting systems that use CAN communication it is necessary to analyze the data and the data stream in some way. In this paper, it is necessary to explore and describe the CAN protocol specifications. The aim of this pape ris to develop a system for non-invasive analysis and monitoring, storage and display of data on the CAN bus. The system needs to be implemented in Linux operating systems, on the Raspberry Pi computer. Iti s necessary to describe how to install and use the system.

Keywords: communication protocol, CAN protocol, CANopen, analysis system, Raspberry Pi, Linux