SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Laboratorijska vježba: Tehnologije umrežavanja

CAN komunikacija

Juraj Peršić
Ivan Petrović

Zagreb, 2024

Sadržaj

Uvod

Cilj ove laboratorijske vježbe je stjecanje praktičnog razumijevanja protokola komunikacije Controller Area Network (CAN). Vježba je podijeljena u tri zadatka. U prvom zadatku proučit ćemo jedan stvarni primjer CAN komunikacije dizajniran za Delphi ESR radar, analizirajući eksperiment proveden u Zagrebu. Cilj prvog zadatka je razumjeti sintaksu datoteka definicije CAN poruka, tj. DBC datoteke, koje se koriste za kodiranje i dekodiranje CAN okvira (engl. *frame*). U drugom zadatku će biti potrebno implementirati virtualnu CAN mrežu koja se koristi u povratnoj vezi zatvorenog sustava adaptivnog tempomata (engl. *Adaptive Cruise Control* - ACC) između senzora automobila i elektroničke jedinice za upravljanje (engl. *Electronic Control Unit* - ECU). Cilj drugog zadatka je dizajnirati cjelokupni sustav koji se oslanja na CAN komunikaciju. U trećem zadatku se izvodi analiza pristiglih CAN poruka iz prvog zadatka. Cilj je razumjeti granice CAN komunikacije i procijeniti je li naša mreža uspješno radi. U nastavku, kratki uvod u CAN komunikaciju dan je u poglavlju , dok poglavlje navodi potrebne alate potrebne za izvođenje vježbe.

CAN komunikacija

CAN sabirnica je komunikacijski standard razvijen za sustave unutar automobila koji zahtijevaju visok stupanj robusnosti (pouzdanosti). Dizajnirana je kao multi-master serijska sabirnica gdje svi čvorovi u mreži imaju bitnu sinkronizaciju. Obično se koristi za povezivanje više senzora i ECU-a u vozilu.

Osnova CAN komunikacije je CAN okvir (engl. *frame*). Za aplikacijski sloj, dva najvažnija polja jednog okvira su podaci i identifikator (ID). Podaci okvira mogu sadržavati do 8 bajtova (sl. 5). Navedeni podaci pohranjuju vrijednosti temeljene na unaprijed definiranom protokolu poruke. S druge strane, osnovni ID format CAN okvira jest 11-bitno polje (CAN 2.0A). Osim korištenja ID-a za ispravno dekodiranje podataka, protokol se oslanja na ID kada dva čvora pokušavaju poslati okvire u isto vrijeme. U takvom scenariju, poruka s nižim ID-em, tj. višim prioritetom, se šalje prva, dok drugi čvor pokušava ponovno poslati poruku višeg prioriteta nakon toga. Više informacija o izgledu kompletnog CAN okvira možete pronaći online, npr. Wikipedia. Vrijedno je napomenuti da CAN protokol dopušta alternativne proširene ID CAN okvire koji su dugi 29 bita (CAN 2.0B). Nadalje, proširenje CAN protokola pod nazivom CAN FD (engl. *flexible data-rate*) dopušta do 64 bajta podataka. Međutim, unutar ove vježbe radit ćemo samo s osnovnim CAN formatom (CAN 2.0A). CAN sabirnica omogućuje brzine do 1 Mbit/s, ovisno o duljini mreže. Pri procjeni maksimalne brzine okvira koja se može poslati preko mreže, važno je razumjeti nisku razinu stvaranja okvira. Kao što je prethodno spomenuto, osnovni CAN okvir sastoji se od 11 bitova za ID i $8n$ bita za n bajtova podataka. Dodatnih 36 bita koristi se kao utore za potvrdu (engl. *acknowledge slots*), cikličku provjeru redundantnosti (engl. *cyclic redundancy check*), početak okvira (engl. *start of frame*), razmak između okvira (engl. *inter-frame spacing*) itd. To dovodi do ukupnog broja od $47+8n$, što je minimalni broj bitova potrebnih za slanje jednog okvira. Međutim, budući da CAN sabirnica koristi kodiranje bez povratka na nulu (engl. *non-return-to-zero*), iako zahtijeva sinkronizaciju između čvorova, koristi praksu koja se zove umetanje bita (engl. *bit stuffing*). Sinkronizacija se oslanja na padajuće i rastuće rubove signala, ali se u slučaju pet uzastopnih bitova s istim polaritetom ubacuje dodatan bit suprotnog polariteta. Dakle, točna duljina okvira ovisi o podacima koje šalje. U najgorem slučaju (11111000011110000...), okvir je dopunjen s dodatna 24 bita.

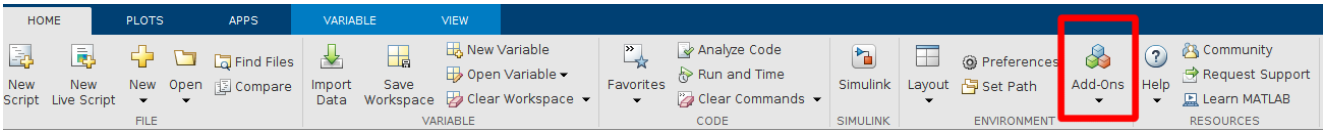
Potrebni alati

U ovoj laboratorijskoj vježbi će se koristiti Matlab i njegov alat Simulink za (i) analizu podataka snimljenih s fizičke CAN sabirnice i (ii) proučavanje simuliranog upravljačkog sustava koji koristi virtualnu CAN sabirnicu. Studenti FER-a putem svojih studentskih licenci imaju pristup kompletnoj funkcionalnosti Matlab okruženja. Upute za instalaciju Matlaba i dobivanje licence možete pronaći ovdje: https://www.fer.unizg.hr/matlab. Prilikom instalacije, uključite sljedeće dodatne pakete (engl. *toolboxes*) jer se koriste u vježbi:

- Simulink
- Simulink Control Design
- Vehicle Network Toolbox
- Model Predictive Control Toolbox

Ako već imate instaliran Matlab bez potrebnih dodatnih paketa, možete ih instalirati pomoću Add-on Explorera (Slika 1).

Dokumentacija CAN komunikacijskog modula unutar Vehicle Network Toolbox paketa se može pronaći ovdje.



Slika 1: Instalacija dodatnih paketa unutar Matlaba

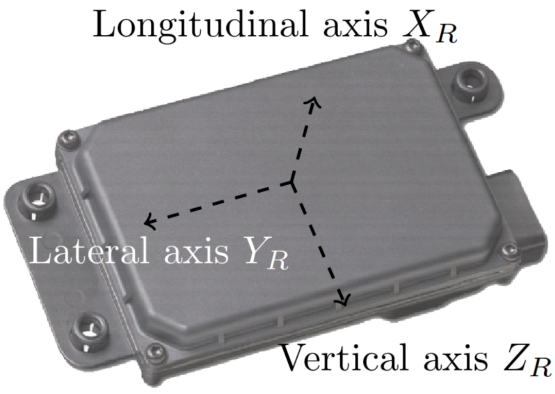
1 Zadatak 1: Konfiguracija CAN protokola

CAN komunikacija prioritizira robusnost, odnosno pouzdanost nad brzinom. Međutim, količina informacija koja se šalje kroz moderna vozila je značajna. Dakle, potrebno je učinkovito koristiti svaki bit u podatkovnom polju okvira.

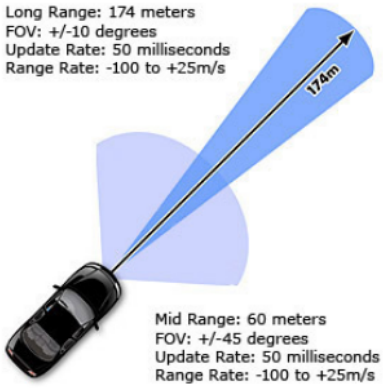
U ovom zadatku proučit ćemo često korištenu datoteku pod nazivom *DBC* koja pomaže u prevođenju CAN podataka u smislene informacije. Koristit ćemo podatke iz stvarnog eksperimenta provedenog u Zagrebu pomoću Delphi ESR radara koji svoja mjerenja prenosi CAN sabirnicom. Kratak uvod o radaru je predstavljen u poglavlju 1.1. Poglavlje 1.2 uvodi glavne koncepte *DBC* datoteke s primjerima iz Delphi ESR DBC datoteke. U poglavlju 1.3 će biti potrebno proširiti DBC datoteku u svrhu identifikacije traženog vozila. Na kraju, kroz poglavlje 1.4 će biti potrebno napraviti analizu podataka koja opisuju traženo vozilo.

1.1 Delphi ESR Radar

Moderni napredni sustavi za pomoć vozaču (engl. *Advanced Driver-Assistance System* - ADAS) često koriste radare za identifikaciju i lokalizaciju okolnih vozila. Radari pružaju pouzdan rad u lošim vremenskim uvjetima te rade na velikim udaljenostima, što je osobito važno u vožnji autocestom. Nadalje, osim određivanja lokacije okolnih vozila, radari također mogu mjeriti njihove



(a) Koordinatni sustav radara [1]



(b) Karakteristike vidnog polja radara (engl. *Field of view* - FOV), izvor : AutonomouStuff

Slika 2: Delphi ESR radar

brzine u radijalnom smjeru koristeći Dopplerov efekt, pružajući brze informacije za sustave kao što je Upozorenje na prednji sudar (engl. *Forward Collision Warning* - FCW).

Primjer takvog radara je Delphi ESR prikazan na sl. 2, čiji su podatci korišteni u ovom zadatku. To je automotive dual-mode rotirajući (engl. *scanning*) radar koji je dizajniran za ACC i FCW sustave. Takvi radari obavljaju nisku razinu obrade podataka radara što značajno smanjuje količinu podataka koji se moraju poslati preko ograničene CAN mreže. Neobrađena očitanja radara pretvaraju se u popis detekcija po svakom okviru ili u popis praćenih objekata na temelju vremenskog filtriranja detekcija. Baza primljenih podataka u ovoj vježbi koristi radar u režimu praćenja objekata (engl. *tracking mode*), koji u tom načinu rada daje 64 praćenih objekata po okviru. Svaka detekcija je opisana dometom (engl. *range*), kutom azimuta, brzinom, amplitudom, jedinstvenim ID-em itd. Više informacija o radaru možete pronaći u priloženoj dokumentaciji: *ESR Startup v2.1*. Međutim, nije potrebno čitati cijelu dokumentaciju, već samo određene dijelove koji će biti istaknuti u nastavku.

1.2 DBC: datoteka definicije CAN poruke

Kako bi se ograničena brzina prijenosa bitova učinkovito iskoristila, podatci koji se prenose putem CAN sabirnice često su tijesno pakirani unutar okvira. Budući da usko pakiranje bitova može postati komplicirano, dobavljač CAN opreme Vector Informatik GmbH dizajnirao je strukturu datoteke *DBC* 1990-ih kako bi pružio standard za opisivanje pravila pakiranja u CAN porukama. Najvažniji dijelovi datoteke *DBC* su definicije poruka, tj. okvira i signala, gdje je signal podskup poruke. Sintaksa predloška za definiranje poruke sa signalom dana je u nastavku.

```
BO_ can_id message_name: data_length sender
 SG_ signal_name : start|length@encoding (scale,offset) [min|max] \
 "unit" reciever
```

Definicija poruke počinje s nazivom *BO_* nakon čega slijede polja sljedećih varijabli :

- *can_id* je decimalni prikaz 11-bitnog IDa poruke
- *message_name* je jedinstveno ime poruke od 1 do 32 znaka

- *data_length* je broj bajtova podataka u okviru
- *sender* je naziv čvora pošiljatelja (Vector__XXX ako nije navedeno)

Definicija signala počinje razmakom i nazivom *SG_* nakon čega slijede polja sljedećim varijabli:

- *signal_name* je jedinstveno ime signala od 1 do 32 znaka
- *start* je početna pozicija bita u podatkovnom sadržaju [0,63]
- *length* je duljina signala u bitovima
- *encoding* (kodiranje) može biti
  - *0+* za *big-endian unsigned*
  - *0-* za *big-endian signed*
  - *1+* za *little-endian unsigned*
  - *1-* za *little-endian signed*
- *scale* i *offset* pretvaraju cjelobrojnu vrijednost u float zapis
- *min*, *max* i *unit* su neobavezne meta informacije signala
- *receiver* je naziv čvora primatelja (Vector__XXX ako nije navedeno)

Primjerice, sljedeći kod opisuje poruku koja sadrži informacije o praćenom objektu s jedinstvenim ID-em 1 (ID unutar radarovog, a ne ID CAN okvira), dok slika 3 pruža vizualizaciju rasporeda bitova.

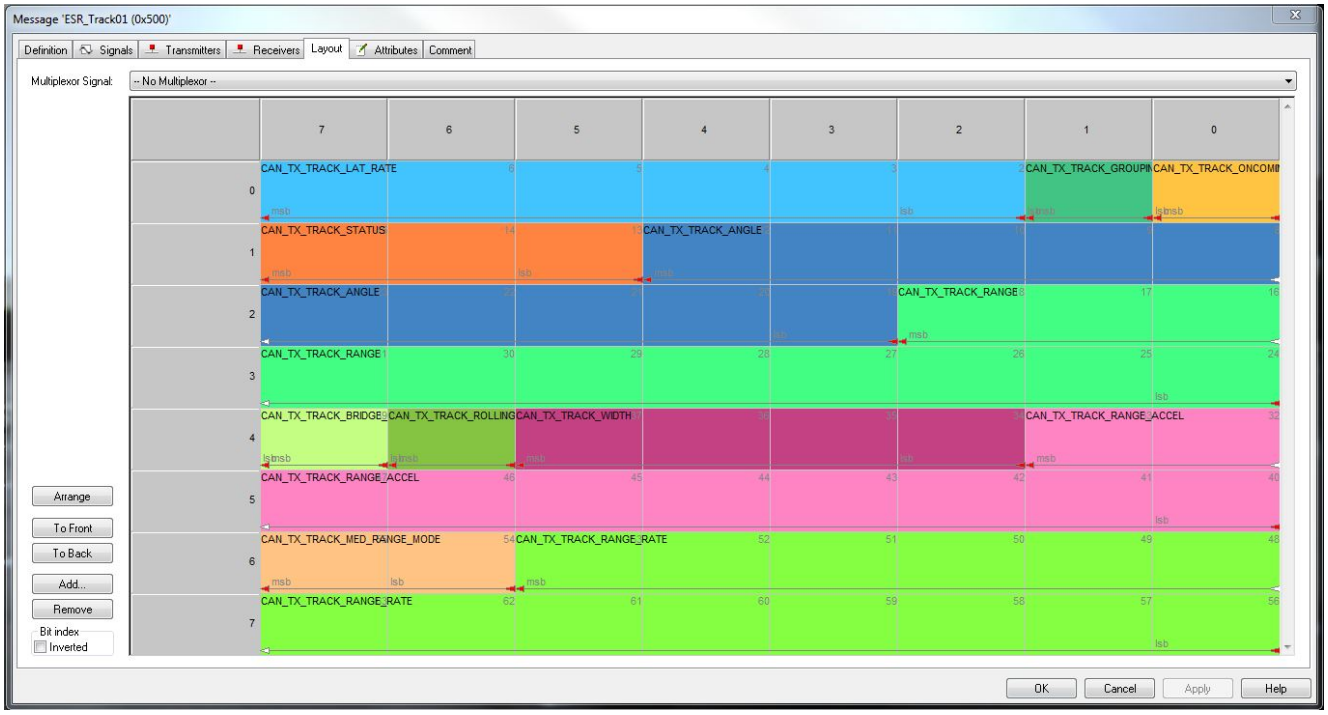
```
BO_ 1280 ESR_Track01: 8 ESR
 SG_ CAN_TX_TRACK_GROUPING_CHANGED : 1|1@0+ (1,0) [0|0] "" Vector__XXX
 SG_ CAN_TX_TRACK_ONCOMING : 0|1@0+ (1,0) [0|0] "" Vector__XXX
 SG_ CAN_TX_TRACK_LAT_RATE : 7|6@0- (0.25,0) [-8|7.75] "" Vector__XXX
 SG_ CAN_TX_TRACK_BRIDGE_OBJECT : 39|1@0+ (1,0) [0|0] "" Vector__XXX
 SG_ CAN_TX_TRACK_WIDTH : 37|4@0+ (0.5,0) [0|7.5] "m" Vector__XXX
 SG_ CAN_TX_TRACK_STATUS : 15|3@0+ (1,0) [0|7] "" Vector__XXX
 SG_ CAN_TX_TRACK_ROLLING_COUNT : 38|1@0+ (1,0) [0|1] "" Vector__XXX
 SG_ CAN_TX_TRACK_RANGE_RATE : 53|14@0- (0.01,0) [-81.92|81.91] "m/s" Vector__X
 SG_ CAN_TX_TRACK_RANGE_ACCEL : 33|10@0- (0.05,0) [-25.6|25.55] "m/s/s" Vector_
 SG_ CAN_TX_TRACK_RANGE : 18|11@0+ (0.1,0) [0|204.7] "m" Vector__XXX
 SG_ CAN_TX_TRACK_MED_RANGE_MODE : 55|2@0+ (1,0) [0|3] "" Vector__XXX
 SG_ CAN_TX_TRACK_ANGLE : 12|10@0- (0.1,0) [-51.2|51.1] "deg" Vector__XXX
```

Osim poruka i signala, *DBC* datoteka također može ispisati sve čvorove na mreži s:

```
BU_: node_name_0 node_name_1 ...
```

Nadalje, određeni signal može se dekodirati kao float ili double sa sljedećom dodatnom naredbom:

```
SIG_VALTYPE_ message_id double_signal_name : 2;
SIG_VALTYPE_ message_id float_single_name : 1;
```

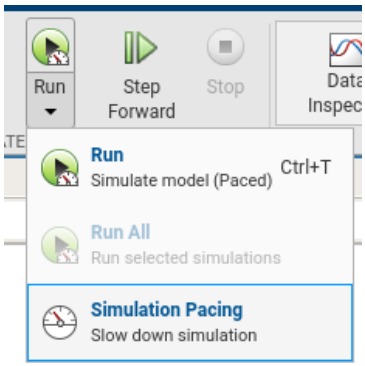


Slika 3: Rasporeda bitova u CAN Track01 okviru.

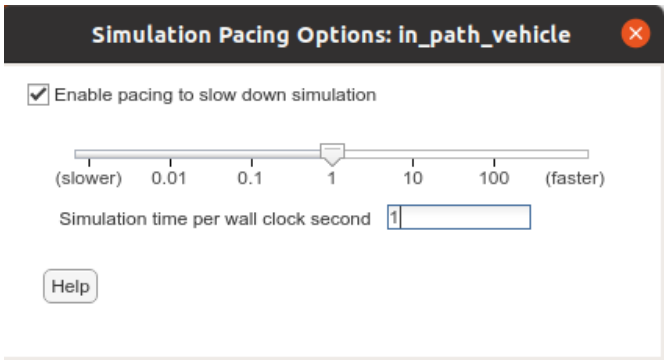
1.3 Identifikacija traženog vozila

Cilj ovog zadatka je odrediti ID detekcije radara koji dolazi od traženog vozila, tj. vozila ispred našeg vozila. Da biste razumjeli scenu, pogledajte video *task1/vukovarska.mp4* koji prikazuje odgovarajući prikaz scene s prednje kamere postavljene na naše vozilo. Otvorite Simulink model *task1/in_path_vehicle.slx*. Obavezno koristite *paced* način pokretanja simulacija (sl. 4). Ova postavka osigurava pokušavanje simuliranja u stvarnom vremenu. U suprotnom, simulacija bi radila prebrzo i ne bi ispravno obradila sve CAN poruke. Držite ovu postavku simulacije tijekom cijele vježbe. Pokušajte pokrenuti simulaciju.

Iako simulacija radi, još ne prikazuje interpretabilne podatke. Vaš zadatak je konfigurirati blok DisplayInPathTrackId/FunctionCallSubsystem/CanUnpackEsrStatus4 tako da pravilno de-

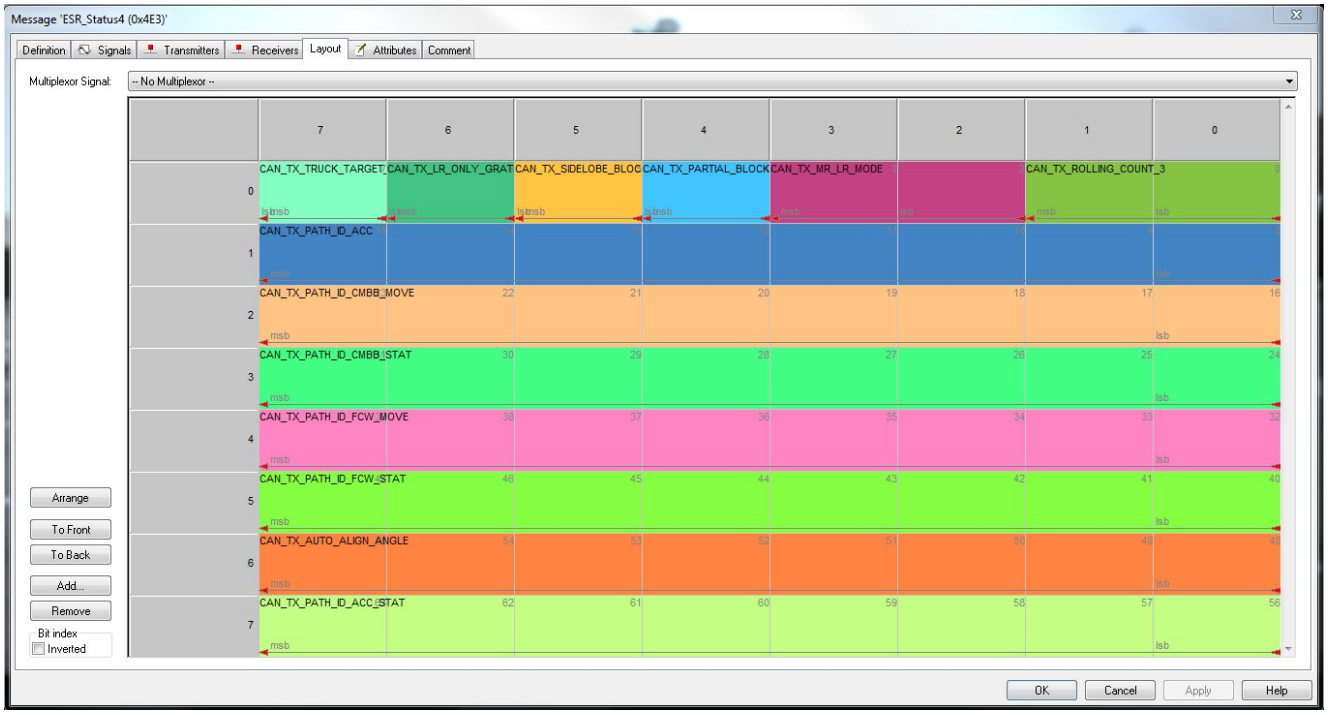


(a) Step 1



(b) Step 2

Slika 4: Omogućavanje *paced* načina pokretanja simulacije.



Slika 5: Rasporeda bitova u CAN ESR_status4 okviru.

kodira signal *CAN_TX_PATH_ID_ACC* iz poruke *ESR_Status4* koje se šalju s radara. Međutim, definicija te poruke nedostaje u datoteci *ESR_DV3_64Tgt_incomplete.dbc*. Kako biste proširili nepotpunu *DBC* datoteku, upotrijebite vizualizaciju rasporeda bitova prikazanu na sl. 5 i detalje dekodiranja na stranici 61 dokumentacije radara. Nakon uspješne konfiguracije raspakiranja poruke *ESR_Status4*, bit će moguće odrediti ID traženog vozila povezivanjem izlaza s nepovezanim *scope* blokom. Koji je ID traženog vozila?

1.4 Vizualizacija svojstva traženog vozila

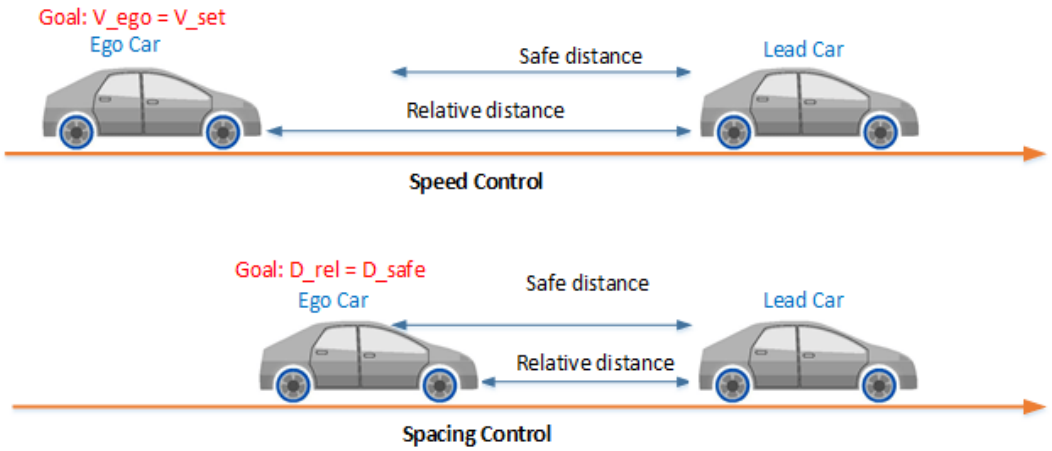
Nakon utvrđivanja ID-a traženog vozila, trebali biste moći jednostavno konfigurirati blok *DisplayTrackInformation/FunctionCallSubsystem1/CanUnpackChosenTrack* za prikaz podataka traženog vozila. Odnosno, unutar bloka *CanUnpackChosenTrack* odaberite poruku ESR_Track**X** gdje **X** predstavlja dobiveni ID iz prethodnog zadatka. *DBC* već sadrži informacije za raspakiranje signala, stoga je samo potrebno odabrati ispravnu poruku. Ponovno pokrenite simulaciju. Možete li prepoznati traženo vozilo na videu? O kojem se vozilu radi?

2 Zadatak 2: Upravljački sustav s CAN komunikacijom

U drugom zadatku ispitat ćemo tipičan primjer upravljačkog sustava koji koristi CAN sabirnicu kao povratnu petlju. Konfigurirat ćemo i analizirati performanse ACC sustava, ukratko opisanog u poglavlju 2.1 Sljedeća tri poglavlja (2.2, 2.3, 2.4) će vas voditi do potpunog konfiguriranja CAN komunikacije u Simulinku. Na kraju ćemo ispitati utjecaj mreže na kvalitetu performansi sustava upravljanja u poglavlju 2.5.

2.1 Arhitektura ACC sustava

Adaptivni tempomat (engl *Adaptive Cruise Control* - ACC) je uobičajena ADAS funkcija koja se najčešće koristi na autocestama. Cilj je održavanje zadane željene brzine uz održavanje sigurnosnog razmaka od vodećeg vozila kao što je prikazano na sl. 6. U ovom zadatku koristimo Simulinkov ACC modul koji se može pronaći ovdje.



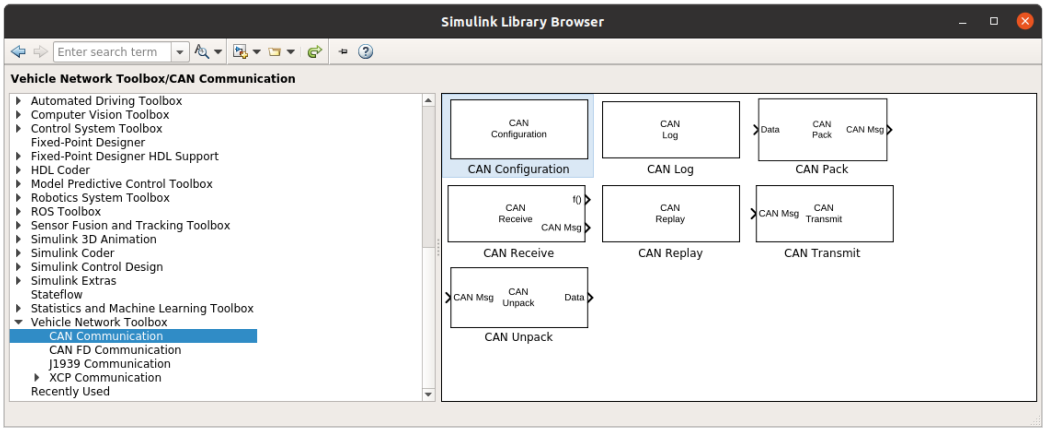
Slika 6: ACC dual-mode kontrola, izvor: Mathworks

Otvorite Simulink model *task_2/acc_system_incomplete.slx*. Unutar modela možete vidjeti četiri različita podsustava: *EgoCar*, *LeadCar*, *EgoCarSensors* i *EgoCarEcu*. Podsustavi *EgoCar* i *LeadCar* simuliraju 1-D trajektoriju oba vozila. Vodeće vozilo se kreće konstantnom brzinom sve dok ne počne kočiti i potpuno se zaustavi. S druge strane, stražnje vozilo slijedi naredbe za ubrzanje iz ACC modula opisanog u nastavku. Modelirali smo komunikaciju između ECU-a i motora kao trenutnu radi jednostavnosti, tj. ne koristeći CAN sabirnicu. Podsustav *EgoCarSensors* modelira mjerenja (i) enkodera kotača koji daju brzinu stražnjeg vozila (engl. *ego-velocity*) i (ii) radara koji daje relativnu udaljenost i brzinu između vodećeg i stražnjeg vozila. Blokovi zadržavanja nultog reda (engl. *zero-order hold*) modeliraju uzorkovanje svakog senzora. Vaš će zadatak biti slati informacije senzora preko CAN sabirnice, kao što je opisano u 2.3. Naposljetku, podsustav *EgoCarEcu* sluša CAN sabirnicu, kao što je opisano u 2.4, kako bi ACC modulu dostavio potrebna mjerenja.

Prije pokretanja bilo kakvih simulacija, izvršite *task_2/parameters.m* skriptu za učitavanje potrebnih parametara u radni prostor.

2.2 Konfiguracija CAN komunikacije

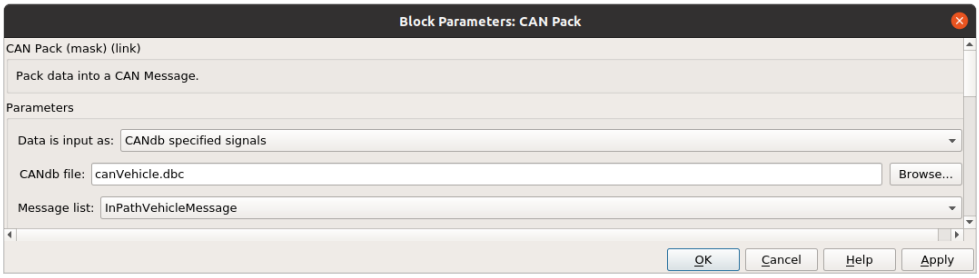
Prvi korak u dizajniranju Simulink modela koji koristi CAN komunikaciju je umetanje bloka konfiguracije sabirnice. Pronađite blok *CAN Configuration* u tražilici biblioteke (engl. *Library Browser*) pod Vehicle Network Toolbox/CAN Communication kartici kao što je prikazano na sl. 7. Povucite i ispustite dva bloka *CAN Configuration* u dijelu najviše, odnosno generalne razine modela *acc_system_incomplete.slx*. Konfigurirajte oba modula tako da koriste brzinu sabirnice od 500000 bita/s i način potvrde (engl. *Acknowledge mode*) *normal*. Na kraju, postavite jedan modul da koristi Kanal 1, a drugi da koristi Kanal 2 Mathworks Virtual 1 uređaja.



Slika 7: CAN Configuration blok

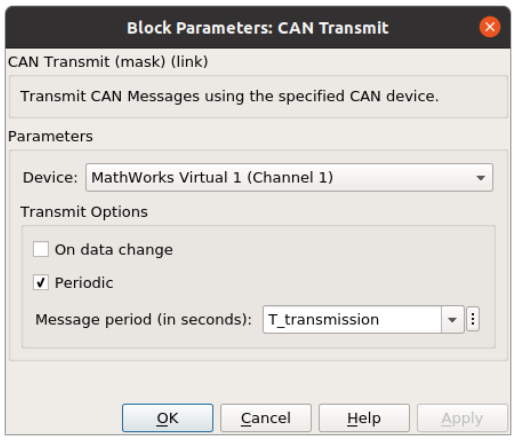
2.3 Slanje mjerenja sa senzora

Otvorite podsustav *EgoCarSensors*. Naš cilj je poslati mjerenja senzora preko CAN mreže. Prvo je potrebno upakirati signale u CAN okvire. Dakle, dodajte dva *CAN Pack* bloka. Konfigurirajte oba za korištenje dostupne *canVehicle.dbc* datoteke. Blok koji pakira podatke radara treba koristiti *InPathVehicleMessage* kao što je prikazano na sl. 8, dok enkoderi kotača trebaju koristiti *SpeedMessage*.



Slika 8: Svojstva CAN Pack bloka koji pretvara mjerenja radara u odgovarajuće CAN okvire.

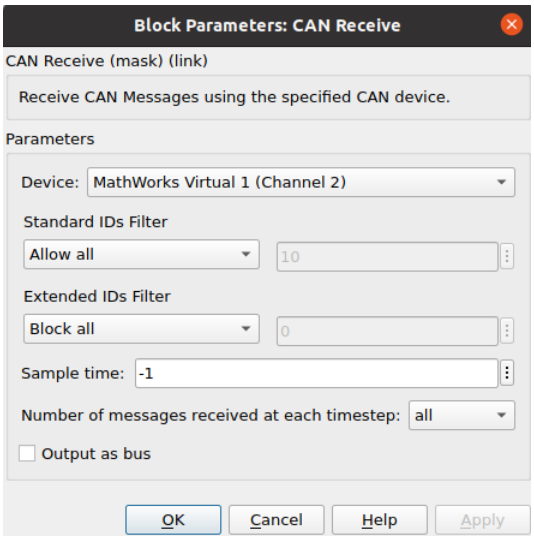
Nakon što poruke upakirane, šaljemo ih na CAN mrežu koristeći dva *CAN Transmit* bloka. Konfigurirajte svaki blok kao što je prikazano na sl. 9 i povežite ih s *CAN Pack* blokovima. Ulaz u *CAN Transmit* blok se može smatrati kontinuiranim, dok će poruke biti poslane s fiksnim *T_transmission* periodom. Imajte na umu da mogli smo koristiti *Mux* blok i samo jedan *CAN Transmit* blok jer simuliramo senzore s istom brzinom prijenosa okvira (engl. *frame rate*) .



Slika 9: Svojstva CAN Transmit bloka.

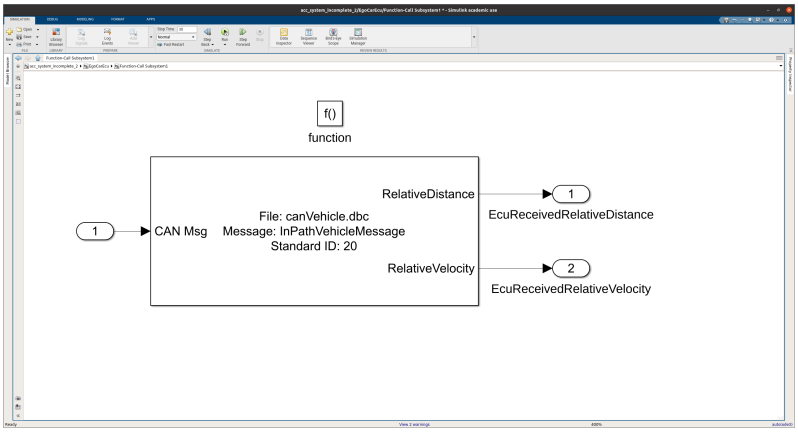
2.4 Primanje poruka u elektroničkoj jedinici za upravljanje

Na kraju, ECU mora primiti CAN poruke da bi se podatci upotrijebili za ACC modul. Da biste to učinili, dodajte dva *CAN Receive* bloka unutar podsustava *EgoCarEcu*. Konfigurirajte oba bloka za rad s *Mathworks Virtual 1 (Channel 2)* uređajem, koristite naslijeđeno (*inherited*) vrijeme uzorkovanja i blokirajte proširene CAN poruke. Slika 10 prikazuje spomenute postavke gdje filtriramo sve osim standardnih CAN 2.0A poruka. Konfigurirajte drugi blok za primanje *InPathVehicleMessage* poruka.



Slika 10: Postavke CAN Receive bloka za stražnje/vlastito vozilo (*EgoCar*).

Sljedeći korak je dodavanje dva *Function-Call Subsystem* bloka u podsustav *EgoCarEcu*. Možete ih pronaći dvoklikom na prazan prostor u Simulink modelu i upisivanjem naziva bloka. Ti blokovi nisu potrebni, ali osiguravaju da primimo sve poruke ako više od jedne poruke stigne unutar jednog vremenskog koraka simulacije. Unutrašnjost jednog takvog podsustava za radar ilustrirana je na sl. 11. Na kraju, konfigurirajte blokove *CAN Unpack* u oba *Function-Call Subsystem* bloka, povežite odgovarajuće signale i spremni ste za simulaciju cijelog sustava.



Slika 11: Podsustav za dekodiranje podataka radara.

2.5 Analiza i podešavanje rada ACC modula

S danim postavkama u datoteci *parameter.m*, trebali biste moći dobiti interpretabilan odziv kontrolnog sustava. Kako biste bolje razumjeli sustav pokušajte odgovoriti na sljedeća pitanja:

- Koliko je vremensko kašnjenje između stvarnih vrijednosti i onih koje prima ECU?

- Postoje li čudne pojave u komunikacijskom kanalu i kako oni utječu na sustav upravljanja?

- Možete li pronaći realnu situaciju u kojoj ACC konfiguracija i putanja vodećeg automobila dovode do sudara?

2.5.1 Smanjenje brzine prijenosa okvira

Pokušajte smanjiti brzinu slanja podataka (engl. *frame rate*) senzora i perodičnog prijenosa (engl. *periodic transmission*) povećanjem vremena uzorkovanja T_sensor i $T\_transmission$ na $1.0\,s$. Ponovo pokušajte odgovoriti na sljedeća pitanja:

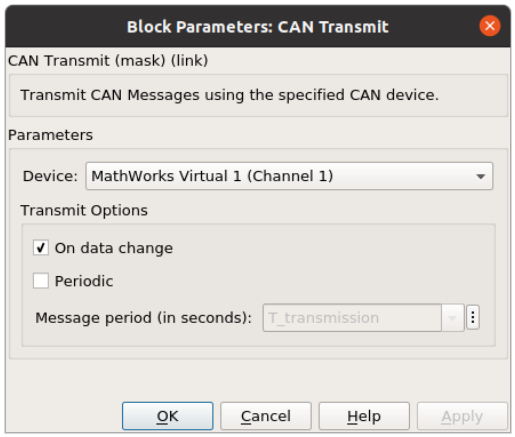
- Jesu li se vozila sudarila u ovom scenariju?

- Koliko je vremensko kašnjenje između stvarnih vrijednosti i onih koje prima ECU?

- Poštuje li brzina stražnjeg vozila postavljena pravila?

2.5.2 Komunikacija temeljena na događajima

Na kraju, pokrenite komunikaciju temeljenu na događajima (engl. *event based communication*) umjesto periodične komunikacije s postavkama iz prethodnog zadatka, tj. vremenima uzorkovanja od 1.0 s. Otvorite oba bloka *CAN Transmit*, onemogućite opciju periodičnog prijenosa i omogućite *On data change* postavku kao što je ilustrirano na sl. 12. Simulirajte model.



Slika 12: Komunikacija temeljena na događajima

Ponovo pokušajte odgovoriti na sljedeća pitanja:

- Jesu li se vozila sudarila u ovom scenariju?

- Koliko je vremensko kašnjenje između stvarnih vrijednosti i onih koje prima ECU?

- Poštuje li brzina stražnjeg vozila postavljena pravila?

Kao što je i vidljivo, smanjenje brzine slanja poruka degradira performanse sustava, što može dovesti do opasnih situacija. Korištenje komunikacije temeljene na događajima može pomoći u situaciji s malom brzinom slanja poruka. Međutim, komunikacija temeljena na događajima može preopteretiti mrežu ako šalje previše poruka. Stoga je važno osigurati da sustav ne koristi veću brzinu prijenosa od dostupne. Ako je to neizbježno, trebamo osigurati da su svi ID-evi poruka kritičnih za sigurnost postavljeni na najveći prioritet, tj. na najniži ID.

3 Zadatak 3: Analiza raspodjele širine pojasa

U posljednjem zadatku napravit ćemo analizu visoke razine primljenih poruka na CAN sabirnici, koristeći podatke iz Zadatka 1. Analiza visoke razine primljenih ID-ova okvira može nam pomoći da shvatimo kako se mreža koristi i radi li ispravno odgovarajući na neka od sljedećih pitanja:

- Možemo li zabilježiti sve podatke ili imamo ispadanje/gubljenje okvira?

- Je li mreža preopterećena?

- Možemo li raspakirati sve poruke koje smo primili?

Poglavlje 3.1 vodit će vas kroz analizu primljenih podataka. Nakon toga, kroz poglavlje. 3.2 ćemo analizirati redoslijed primljenih poruka. Na kraju, procijenit ćemo koliko je širine pojasa mreže alocirano u poglavlju 3.3.

3.1 Analiza primljenih ID-eva okvira

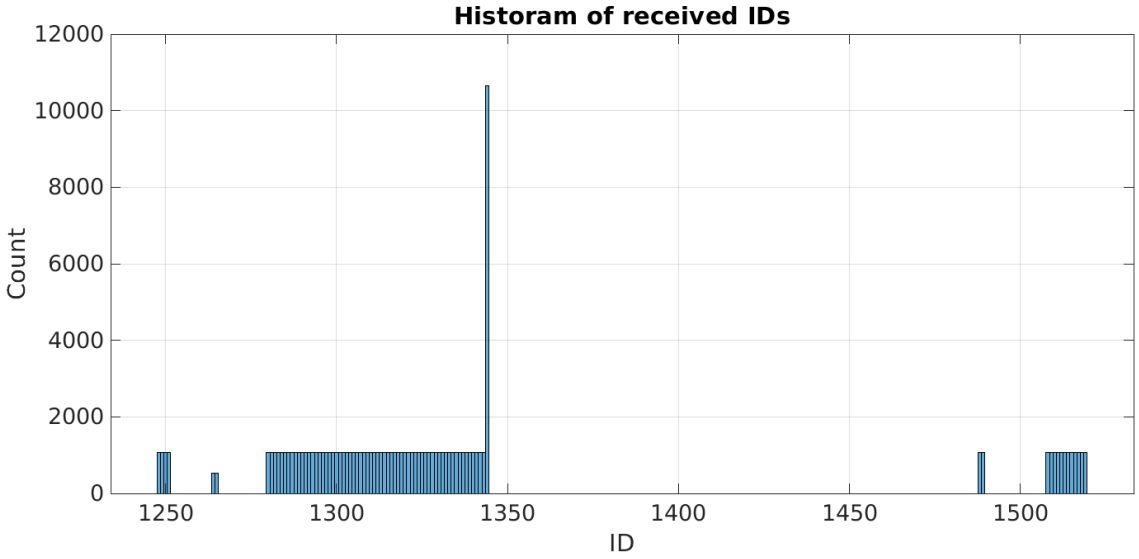
U ovom zadatku ispitat ćemo koliko je poruka primljeno kroz cijelu sekvencu po svakom jedinstvenom ID-u. Takva je analiza korisna za utvrđivanje niza stvari:

- Jesu li svi okviri zabilježeni ili je došlo do odbacivanja okvira?

- Je li učestalost svakog ID-a okvira očekivana?

- Možemo li prepoznati i raspakirati sve primljene poruke?

Za početak pokrenite skriptu *task_3/received_frame_id_analysis.m*. Skripta će učitati sve potrebne podatke i iscrtati histogram primljenih poruka prikazan na sl. 13.



Slika 13: Histogram primljenih Id-eva poruka.

Odmah možemo primijetiti da se jedna poruka šalje s frekvencijom 200Hz, dvije s 10Hz, a ostale s nominalnom frekvencijom radara od 20Hz. Iako se ovo ponašanje čini čudnim, ne mora nužno biti pogrešno. Međutim, trebali bismo provjeriti možemo li razumjeti koje informacije poruke prenose pomoću dostupne dokumentacije. Kao pomoć, dostupni ID-evi iz *ESR_DV3_64Tgt_incomplete.dbc* datoteke i Delphi ESR dokumentacije izvučeni su i spremljeni u datoteku *known_messages.mat*.

Vaš zadatak je odgovoriti na sljedeća pitanja:

- Možete li pronaći informacije o svim primljenim porukama u DBC datoteci odnosno dokumentaciji radara? Ako ne, koliko njih nema definicije?

- Možete li objasniti zašto se poruka 1344 mora slati 10 puta češće od ostalih?

- Koji uređaj šalje poruke (1264,1265) koje imaju upola manji broj brzine slanja okvira?

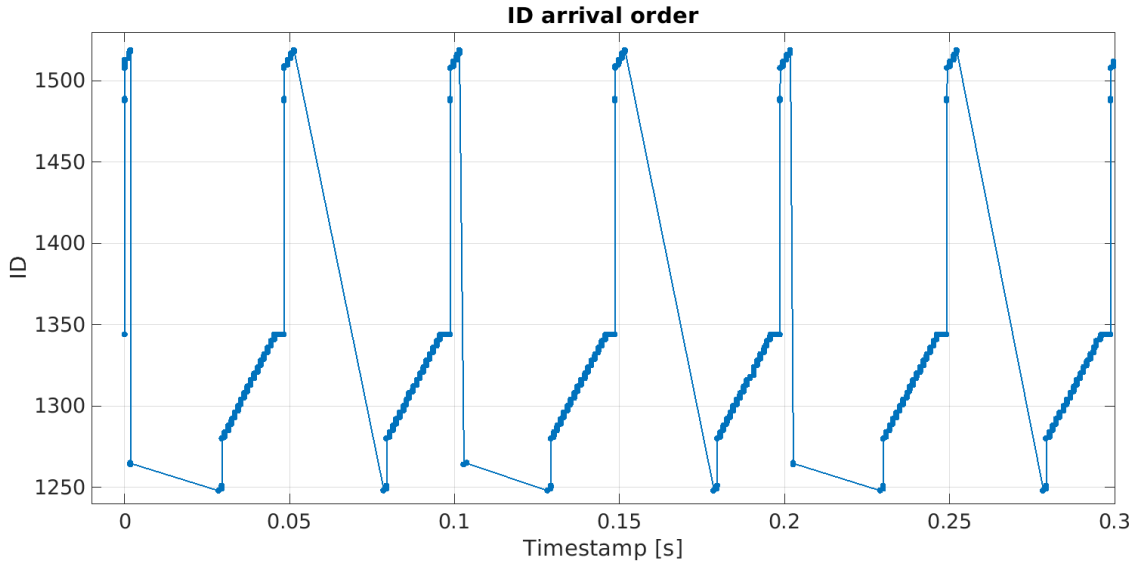
Nakon što odgovorite na sva pitanja, popunite tablicu 1 koja će se koristiti u nastavku. Trebali biste izbrojati koliko je različitih poruka poslano na svakoj frekvenciji. Stupac ID-evi nije obavezan i služi samo kao pomoć pri brojanju.

Brzina prijenosa [Hz]	Broj jedinstvenih ID-eva	ID-evi (opcionalno)
10		
20		
200		

Tablica 1: Statistika ID-eva okvira po frekvenciji.

3.2 Analiza redoslijeda primanja ID-eva okvira

Razumijevanje redoslijeda pristizanja poruka može pomoći u upravljanju i obradu istih. Pokrenite skriptu *task_3/frame_id_arrival_order.m* za generiranje dijagrama prikazanog na sl. 14. Podatci na slici 14 predstavljaju ID-eve zabilježenih poruka u trenutcima njihova dolaska, dok linije povezuju uzastopne poruke.



Slika 14: Red primanja ID-eva poruka.

Odgovorite na sljedeća pitanja:

- Možete li primijetiti ponovljiv uzorak u primljenim porukama?

- Jesu li poruke koje šalje vozilo (1264,1265) sinkronizirane s radarom ili su potpuno asinkrone?

- Možete li primijetiti periode kada je CAN sabirnica aktivna i kada je neaktivna?

- Ako pretpostavimo da sve poruke radara po jednom uzorku (20Hz) moraju biti primljene prije njihove upotrebe u downstream algoritmima, koliko je vremensko kašnjenje uzrokovano CAN komunikacijom?

3.3 Procjena brzine prijenosa podataka

Kao što ste naučili u Zadatku 2 ove vježbe, sustavi upravljanja uvelike se oslanjaju na pravovremeni prijem potrebnih mjerenja. Stoga je važno biti siguran da naša CAN mreža nije preopterećena. Ovaj zadatak će vam pomoći da procijenite alociranu brzinu prijenosa pomoću dva pristupa: (i) teoretski koristeći znanje o niskorazinskoj implementaciji CAN-a i (ii) kvantitativnom analizom vremena mirovanja sabirnice.

3.3.1 Teorijska procjena

Poglavlje objašnjava strukturu i formiranje kompletnog CAN okvira koji se prenosi preko mreže. U slučaju Delphi ESR-a, CAN sabirnica koristi brzinu prijenosa od 500000 bit/s, dok sve poruke imaju 8 bajtova podataka. Dakle, svaki okvir treba najmanje 111 bita za slanje, dok je u najgorem slučaju umetanja bitovima (engl. *bit stuffing*) potrebno 135 bita. Vaš zadatak je izračunati koliko se bit/s koristi za slanje svih poruka navedenih u tablici 1. Odgovorite na sljedeća pitanja:

- Koliko se širine pojasa [%] koristi u najboljem slučaju umetanja bitova?

- Koliko se širine pojasa [%] koristi u najgorem slučaju umetanja bitova?

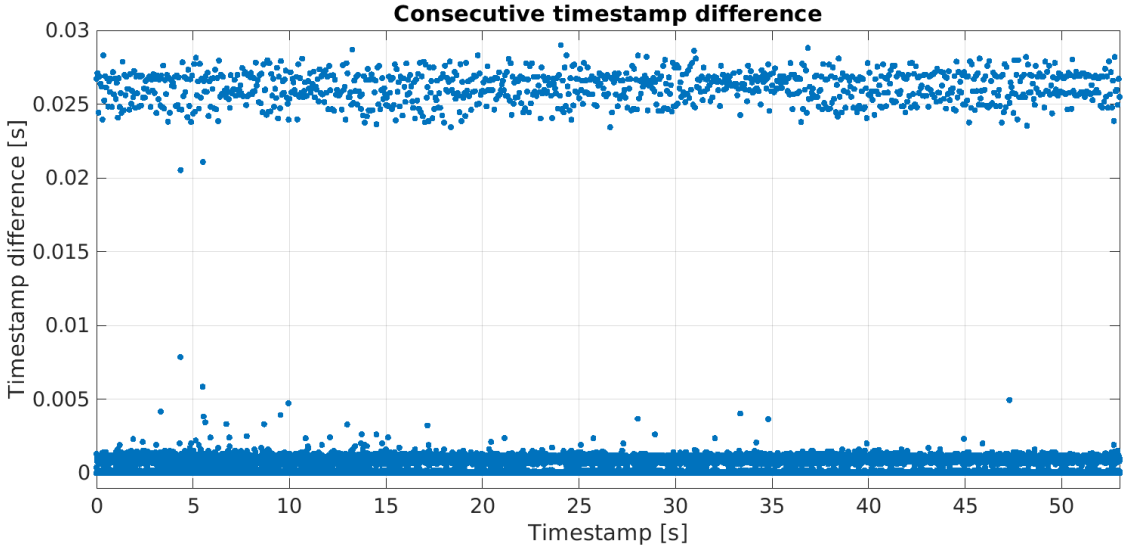
3.3.2 Kvantitativna procjena

Kako bismo potvrdili naše rezultate iz teorijske procjene, pokušat ćemo procijeniti vrijeme opterećenja i mirovanja CAN sabirnice. Treba napomenuti da ova analiza nije potpuno točna, jer na nju utječu i drugi utjecaji osim CAN sabirnice, npr. kašnjenje ili vremenska nestabilnost koje

uvodi sustav spremanja podataka. Međutim, to je prikladan alat za grubu procjenu postotka iskorištenosti CAN sabirnice.

Slika 14 jasno pokazuje kako postoje razdoblja gustog prometa (npr. $[0.03, 0.05]\,s$) kada pretpostavimo da je sabirnica potpuno iskorišten i razdoblja bez ikakvog prometa (npr. $[0.00, 0, 03]\,s$).

Cilj nam je procijeniti omjer zbrojeva tih razdoblja u određenom trajanju. Da bismo to učinili, prvo pogledajmo dijagram uzastopnih razlika vremenskih oznaka. Graf se može generirati pomoću skripte *task_3/bitrate_assessment.m*, a prikazan je i na sl. 15. Gornje točke grafikona (više od $0,02\,s$) predstavljaju dvije uzastopne poruke koje pripadaju dvama različitim porukama radara. S druge strane, niže točke koje su gušće predstavljaju uzastopne CAN poruke koje pripadaju istim porukama radara.



Slika 15: Vremenska razlika uzastopnih vremenskih oznaka (engl. *timestamps*).

Iako postoji minimalna vremenska nestabilnost u vremenskim oznakama, možemo jasno primijetiti prag od $10\,ms$. Vremenske razlike veće od praga pripadaju vremenu mirovanja sabirnice, dok one niže od praga pripadaju vremenu zauzetosti odnosno opterećenja. Vaš zadatak je jednostavno zbrojiti vremena zauzetosti i podijeliti ih s ukupnim trajanjem zapisa. Odgovara li ovaj kvantitativni rezultat izračunatom omjeru iz teorijske procjene?

Literatura

[1] L. Stanislas and T. Peynot, "Characterisation of the Delphi Electronically Scanning Radar for Robotics Applications," in *Australasian Conference on Robotics and Automation (ARAA)*, 2015.