ŽIČNA CAN BUS KOMUNIKACIJA IZMEĐU SENZORSKIH ČVOROVA

Milutin, Filip

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Specijalistički diplomski stručni studij elektrotehnike

FILIP MILUTIN

ZAVRŠNI RAD

ŽIČNA CAN BUS KOMUNIKACIJA IZMEĐU SENZORSKIH ČVOROVA

Split, kolovoz 2021.

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Specijalistički diplomski stručni studij elektrotehnike

Predmet: Senzorske mreže

ZAVRŠNI RAD

Kandidat: Filip Milutin

Naslov rada: Žična CAN BUS komunikacija između senzorskih čvorova

Mentor: Marko Meštrović, struč. spec. ing. el., pred.

Split, kolovoz 2021.

Sadržaj

Sažetak

Žična CAN Bus komunikacija između senzorskih čvorova

Jedan od velikih uspjeha u autoindustriji je razvoj CAN komunikacijskog protokola. CAN je uvelike smanjio korištenje bakrene žice u automobilu, koje ionako ima otprilike oko 1,5 kilometar, dok bi bez CAN protokola ta brojka bila puno veća. Smanjili su se troškovi i unesena je jedna uredna struktura u automobilu gdje svaki modul može komunicirati sa drugim modulom. U ovom radu objasnit će se teorijski dio CAN komunikacije, od fizičkog dijela do dijela prijenosa poruke, te će se raditi na razvijanju prototipa CAN komunikacije između senzorskih čvorova na Buggy vozilu.

Ključne riječi: CAN Bus, Buggy vozilo

Summary

Wired CAN Bus communication between sensor nodes

One of the great successes in the automotive industry is the development of the CAN communication protocol. CAN has greatly reduced the use of copper wire in the car, the wire is about 1.5 kilometers anyway, while without the CAN protocol that number would be much higher. Costs have been reduced and one neat structure has been introduced in the car where each module can communicate with another module. This paper will explain the theoretical part of CAN communication, from the physical part to the part of message transmission, and will work on the development of a prototype of CAN communication between sensor nodes on a Buggy vehicle.

Key words: CAN Bus, Buggy vehicle

1. UVOD

Iznesena su mnoga predviđanja da će više od 50 milijardi "stvari" biti povezano s internetom, oblakom *(engl. Cloud)* i međusobno, a prosječni broj povezanih uređaja po kućanstvu biti će 50. Prema svemu sudeći, ovo se predviđanje obistinilo. Ovaj broj uključuje sve uređaje, strojeve i senzore koji postoje svugdje u našem modernom svijetu i on će samo rasti. No, postoji jedan aspekt sistemske komunikacije koji je jednako važan za budućnost ovih sve složenijih tehnologija - CAN Bus. Način na koji uređaji, senzori i sustavi komuniciraju lokalno jednako je važan za proizvođače, dobavljače i integratore hardvera kao i način obrade podataka u oblaku. Iako je na raspolaganju mnoštvo komunikacijskih protokola i metoda koji omogućavaju da jedan sustav izravno "razgovara" s drugim, sve je veći interes za korištenjem CAN Bus komunikacije za širok spektar industrijskih aplikacija.

Moderni automobili uključuju ogroman broj senzora i aktuatora, koji kontinuirano razmjenjuju podatke i upravljačke naredbe, s toga je Controller Area Network (CAN) u automobilskom sustavu najčešće korišteni protokol za komunikaciju različitih komponenata.

U ovo radu objasnit će se teorijski dio principa rada CAN Bus protokola, a glavni cilj ovog rada biti će izrada prototipa žične CAN Bus komunikacije između senzorskih čvorova za Buggy vozilo. Za izradu prototipa koristiti će se Arduino softver i nekoliko Arduino mikrokontrolera koji će preko CAN sabirnice upravljati senzorima i LED svjetlima.

2. POVIJEST RAZVOJA CAN BUS KOMUNIKACIJE

Početkom 1980-ih inženjeri iz Bosch-a procjenjivali su postojeće serijske sabirničke sustave s obzirom na njihovu moguću upotrebu u osobnim automobilima. Budući da niti jedan od dostupnih mrežnih protokola nije mogao ispuniti zahtjeve automobilskih inženjera, 1983. godine Uwe Kiencke započeo je razvoj novog sustava serijskih sabirnica. Novi sustav trebao je dodati nove funkcionalnosti poput smanjenja broja kabela. [1]

U veljači 1986. Robert Bosch predstavio je serijski sabirnički sustav Controller Area Network (CAN) na kongresu Društva Automobilskih Inženjera. Bilo je to stvaranje jednog od najuspješnijih mrežnih protokola ikad, a predstavljen je kao "Automotive Serial Controller Area Network". Bazirala se na mehanizmu koji omogućuje pristup sabirničkom okviru s najvišim prioritetom bez ikakvih kašnjenja. Bosch je također implementirao nekoliko mehanizama za otkrivanje pogrešaka koji bi svojom obradom automatski isključivala neispravni čvor kako bi se nastavila komunikacija između ostalih čvorova. [1]

Sredinom 1987. godine Intel je izradio prvi CAN čip, 82526 CAN kontroler. Bila je to prva hardverska implementacija CAN protokola. U samo četiri godine ideja je postala stvarnost. Ubrzo nakon toga, Philips je predstavio 82C200. Ova dva najranija predaka CAN kontrolera bila su prilično različita u pogledu filtriranja prihvaćanja i rukovanja okvirom. S jedne strane, FullCAN koncept koji je Intel favorizirao zahtijevao je manje opterećenje procesora od implementacije mikrokontrolera BasicCAN-a koju je odabrao Philips. S druge strane, FullCAN uređaj bio je ograničen s obzirom na broj okvira koji se mogu primiti. U današnjim CAN kontrolerima primijenjena je mješavina oba koncepta filtriranja prihvaćanja i rukovanja okvirom. [1]

Verzija 2.0 Bosch CAN predana je na međunarodnu standardizaciju početkom 1990-ih. Nakon nekoliko političkih sporova, posebno oko "Vehicle Area Network" (VAN), koju su razvili neki od najvećih francuskih proizvođača automobila, u studenom 1993. objavljen je standard ISO 11898. Uz CAN protokol, također je standardiziran fizički sloj za brzine do 1 Mbit/s, a 1995. godine norma ISO 11898 proširena je dodatkom koji opisuje format proširenog okvira pomoću 29-bitnog CAN identifikatora. Nažalost, sve objavljene specifikacije i standardizacije CAN-a sadržavale su pogreške ili su bile nepotpune. Kako bi

izbjegao nekompatibilne CAN implementacije, Bosch se pobrinuo da svi CAN čipovi budu u skladu s Bosch CAN referentnim modelom. [1]

Godine 1993. Bosch je na čelu europskog konzorcija u okviru projekta Espirt ASPIC razvijao prototip onoga što danas poznamo kao CANopen. Nakon završetka projekta, na daljnji razvoj i održavanje CANopen je predan CiA-i (*CAN-in-automation*). 1995. godine objavljen je potpuno prerađen komunikacijski profil CANopen koji je za samo pet godina postao najvažnija standardizirana ugrađena mreža u Europi. CANopen nudi vrlo visoku fleksibilnost i podesivost, a prve CANopen mreže su korištene za unutarnju strojnu komunikaciju, posebno za pogone. Protokol višeg sloja koji je 2003. godine međunarodno standardiziran koristio se u nekoliko različitim područjima primjene (industrijska automatizacija, pomorska elektronika, vojna vozila, itd.). CANopen se posebno počeo koristi u Europi. Strojevi za injekcijsko prešanje u Italiji, pile i strojevi za drvo u Njemačkoj, strojevi za cigarete u Velikoj Britaniji, dizalice u Francuskoj, strojevi za rukovanje u Austriji i strojevi za izradu sata u Švicarskoj samo su nekoliko primjera u industrijskoj automatizaciji i strojogradnji. U Sjedinjenim Državama CANopen se preporučuje za viličare i koristi se u strojevima za sortiranje slova. [1]

General Motors i Bosch početkom 2011. godine započeli su razvoj poboljšanja nekih CAN protokola u pogledu veće propusnosti. Automobilska industrija posebno je patila prilikom preuzimanja sve većih softverskih paketa u elektroničke upravljačke jedinice. Ovakav dugotrajan zadatak morao je biti skraćen s boljim komunikacijskim sustavom. Jedna od ideja da se poveća brzina prijenosa sa uvođenjem druge brzine prijenosa nije bila nova. Nekoliko studenata je objavilo pristupe od početka 2000. Problem je bio što nitko od njih nije bio dovoljno zreo da uvjeri proizvođače automobila. U suradnji s drugim stručnjacima za CAN, Bosch je unaprijed razvio specifikaciju CAN FD, službeno predstavljenu 2012. godine na 13. međunarodnoj konferenciji CAN u dvorcu Hambach u Njemačkoj. [1]

Uvođenjem CAN FD protokola životni vijek CAN tehnologije je produžen. Automobilska industrija već je počela usvajati CAN FD protokol za sljedeću generaciju mreža u vozilu. Može se očekivati da će sve buduće aplikacije koristiti CAN FD protokol. Nije važno trebaju li veću propusnost ili ne. I dalje možemo koristiti CAN FD s jednom postavkom vremenskog ograničenja bita. Duljina okvira je ionako moguće konfigurirati od 0 do 64 bajta. CiA je također razvila CANopen FD protokol koji se temelji na donjim slojevima

CAN FD. Pogotovo za industrijsku primjenu kontrole kretanja, veće brzine prijenosa i duži okviri (do 64 bajta) su vrlo dobrodošli. [1]

Na zahtjev Volkswagena krajem 2018. godine CiA je započela razvoj CAN XL treće generacije protokola podatkovnog sloja temeljenih na CAN-u. Maksimalno podatkovno polje bi iznosilo 2048 bajta. [1]

2.1. Prvi CAN protokol

Iako je CAN izvorno razvijen za uporabu u osobnim automobilima, prve su aplikacije došle iz različitih tržišnih segmenata. Pogotovo u sjevernoj Europi, CAN je već bio vrlo popularan u svojim ranim danima. U Finskoj, proizvođač dizala Kone je koristio CAN Bus. Inženjerski ured Kvaser u Švedskoj predložio je CAN kao komunikacijski protokol između strojeva proizvođačima i njihovim dobavljačima. U Nizozemskoj, Philips Medical Systems pridružio se industrijskim korisnicima CAN-a odlučivši koristiti CAN za interno umrežavanje njihovih rendgenskih uređaja. Još jedan akademski pristup slijedio je "Njemačka udruga poljoprivrednih vozila". Od kasnih 1980-ih razvijen je autobusni sustav zasnovan na CAN-u za poljoprivredna vozila. Također je razvijena standardizacija CAN-a za kamione. Umrežavanje kamiona i prikolice standardizirano je kao ISO 11992. [1]

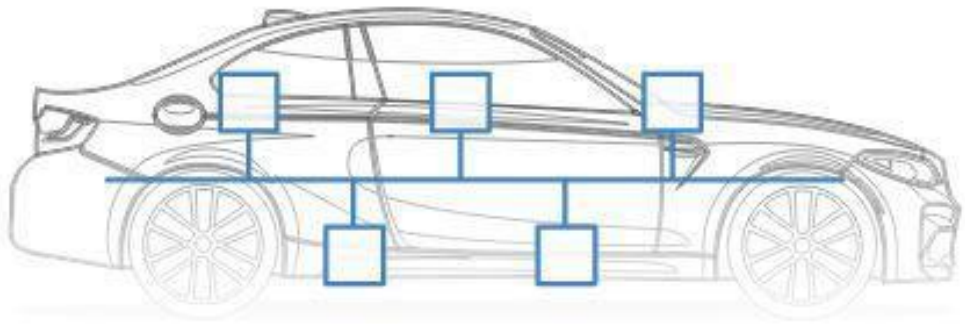
Od 1991. Mercedes-Benz koristi CAN u svojim putničkim automobilima više klase. Kao prvi korak, elektroničke upravljačke jedinice koje vode brigu o upravljanju motorom povezane su putem CAN-a. 1995. BMW je u svojoj seriji 7 automobila koristio CAN stablastu topologiju s pet ECU-a *(engl. Electornic Control Units).* U drugom koraku slijedile su upravljačke jedinice potrebne za karoseriju. Provedene su dvije fizički odvojene CAN mreže, često povezane putem poveznika *(engl. Gateway).* Ostali proizvođači automobila slijedili su primjer svojih vršnjaka iz Stuttgarta i obično ugrađivali dvije CAN mreže u svoje osobne automobile. U današnja vozila svi proizvođači ugrađuju višestruke CAN mreže. [1]

3. CAN BUS

Controller Area Network (CAN) je serijska komunikacijska sabirnica dizajnirana za robusne i fleksibilne performanse u zahtjevnim okruženjima, a posebno za industrijske i automobilske primjene. CAN koristi dvožični serijski komunikacijski protokol koji se sastoji od dvije električne žice CAN_High i CAN_Low. CAN Bus definira sloj podatkovne veze i fizički sloj iz OSI *(engl. Open Systems Interconnection)* referentnog modela. Konkretno, CAN je razvijen kako bi se smanjilo kabelsko ožičenje, tako da zasebne elektroničke upravljačke jedinice (ECU) unutar vozila mogu komunicirati samo s jednim parom žica. Danas je gotovo svaki novi osobni automobil opremljen barem jednom mrežom CAN protokola. Također se koristi u drugim vrstama vozila, od vlakova do brodova, kao i u sustavima upravljanja u industriji, jer zbog svoje ekonomičnosti, pouzdanosti prijenosa, mogućnost brze detekcije i otklanjanja greške našao je široku primjenu.

Prednosti CAN protokola su:

- Jednostavnost i niska cijena
- Potpuno centralizirana
- Izuzetno robustan
- Učinkovitost



Slika 3.1. CAN mreža u automobilu sa elektroničkim upravljačkim jedinicama [3]

Na čvorovima CAN mreže obično su spojeni senzori i aktuatori, dok svaki čvor zahtjeva:

- Mikrokontroler *(engl. host processor)* - odlučuje što znače primljene poruke i koje poruke želi poslati i na njega su spojeni senzori, aktuatori
- CAN kontroler - često sastavni dio mikrokontrolera. Prima i pohranjuje serijske bitove sa sabirnice sve dok cijela poruka ne bude dostupna i tek tada ju mikroprocesor može pročitati. Prilikom slanja poruke mikroprocesor prvo šalje poruku CAN kontroleru, koji bitove serijski prenosi na sabirnicu tek onda kad je sabirnica slobodna.
- Primopredajnik *(engl. Transceiver)* - pretvara podatke s razine CAN Bus na razinu koju koristi CAN kontroler i obratno.

U modernim automobilima može se nalaziti oko 70-tak elektroničkih upravljačkih jedinica. Svaka jedinica kontrolira zaseban dio automobila, najveći procesor je upravljačka jedinica motora *(engl. ECM - Engine Control Module)*, dok se ostale koriste za zračne jastuke, audio sisteme, vrata, navigaciju, prozore, rasvjeta auta, ABS itd. To bih značilo da hrpa elektroničkih upravljačkih jedinica kojih međusobno povezuje CAN BUS komuniciraju na temelju emitiranja i svaka upravljačka jedinica presreće svako emitiranje, ali pojedinačno odlučuje hoće li na njega reagirati ili ne. [2] Ugrađena dijagnostika *(engl. On-Board diagnostics - OBD*) je izvještajni sustav vašeg vozila koji vama ili tehničaru omogućuje otklanjanje poteškoća. CAN sabirnica je jedan od protokola koji se koriste u ugrađenoj dijagnostici (OBD). OBD je danas obavezan u svim automobilima i lakim kamionima širom svijeta.

U usporedbi s drugim komunikacijskim protokolima poput UART-a, SPI-a i I2C-a, tu je CAN BUS protokol puno pouzdanija opcija jer su standardi u automobilskoj komunikaciji osjetljivi i koriste se za prijenos vitalnih podataka poput položaja papučice gasa. Ako se dogodi pogrešna komunikacija ili gubitak podataka, to bi moglo dovesti do kritičnih kvarova. [4]

3.1. CAN fizički sloj

CAN je jedna od najsnažnijih mrežnih tehnologija, posebno ako koristimo linijsku topologiju s vrlo kratkim spojevima na čvoru i upletenim paricama. Svi povezani čvorovi moraju podržavati iste brzine podataka i vremensko ograničenje bita, tada sa bakrenim kabelom s upletenim paricama i zajedničkim uzemljenjem obično ostvarujemo fizički prijenos. Fizički sloj komunikacijske veze pokriva aspekte fizičkog prijenosa podataka između čvorova mreže. Fizički sloj CAN BUS definira karakteristike poput vrste kabela, razine električnog signala, zahtjeva za čvorovima, impedancije kabela i slično.

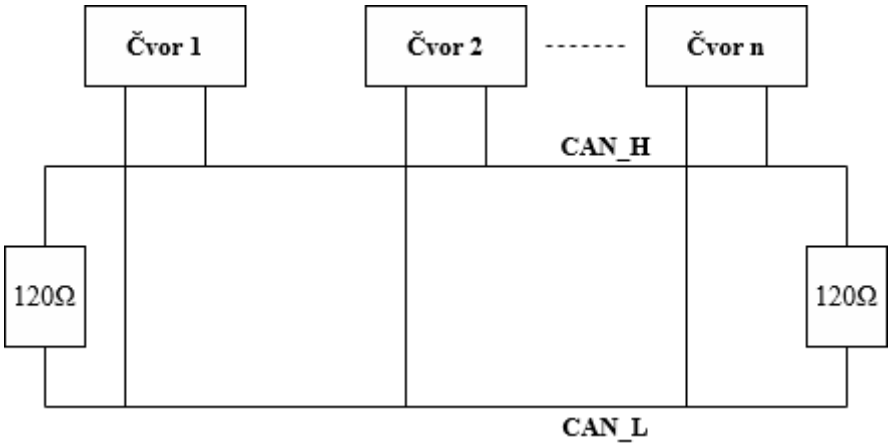
Fizički sloj CAN-a podijeljen na tri podsloja:

- PLS *(engl. Physical Signaling)*
- PMA *(engl. Physical Medium Attachment)*
- MDI *(engl. Medium Dependent Interface)*

PLS sloj je implementiran u čipove CAN kontrolera. Sloj PMA opisuje karakteristike primopredajnika. MDI sloj određuje karakteristike kabela i konektora. Slojevi PMA i MDI podliježu različitim međunarodnim, nacionalnim i industrijskim standardima, kao i vlasničkim specifikacijama. Najčešći je standard ISO 11898 koji specificira primopredajnik velike brzine za mreže temeljene na CAN-u. ISO 11898 je međunarodna norma za brze CAN komunikacije u cestovnim vozilima. ISO-11898-2 specificira PMA i MDI podslojeve fizičkog sloja.

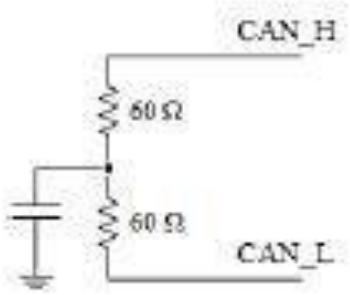
3.1.1. Topologija

Sustav CAN sabirnice najčešće se koristi u linijskoj topologiji, sastavljen pomoću dva međusobno upletena kabela. Sabirničke linije su na krajevima zaključene sa otpornicima vrijednosti 120 Ω kako bi se smanjili učinci refleksije. Dopušteni interval impedancije otpornika je od 108 Ω do 132 Ω. Smanjenjem vrijednosti otpornika smanjujemo broj mogućih priključenih čvorova u mreži. Zaključni otpornici uvijek trebaju biti spojeni na oba kraja sabirnice, nikada se ne smiju fizički priključiti na čvor jer u slučaju uklanjanja čvora sabirnica bi izgubila pravilno zaključenje.

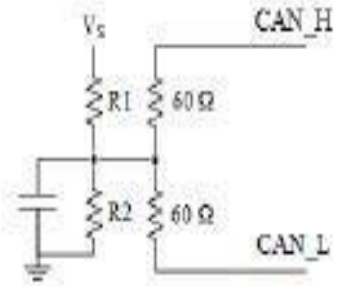


Slika 3.2. CAN topologija

Uz ovaj standard koristi se još dva načina, razdijeljeno zaključenje *(engl Split termination)* i prednaponsko razdijeljeno zaključenje *(engl. Bias split termination).* Smanjenje zračenja linije omogućava razdijeljeno zaključenje, dok prednaponsko razdijeljeno zaključenje smanjuje osjetljivost na utjecaje elektromagnetskih zračenja i nižu razinu emisije.



Slika 3.3. Razdijeljeno zaključenje



Slika 3.4. Prednaponsko razdijeljeno zaključenje

Maksimalna brzina signalizacije iznosi do 1 Mbit/s s duljinom sabirnice od 40 m i maksimalno 30 čvorova. Također se preporučuje maksimalna duljina priključnih ogranaka

za povezivanje čvorova na sabirnicu od 0,3 m. U osnovi, maksimalna duljina sabirnice određuje se, odnosno predstavlja kompromis s odabranom brzinom signalizacije. [6]

Tablica 3.1. Duljina CAN sabirnice i brzina prijenosa podataka [6]

Duljina sabirnice (m)	Brzina (Mbps)
40	1
100	0.5
200	0.25
500	0.1
1000	0.05

Brzina prijenosa signala smanjuje se kako se povećava udaljenost prijenosa. Iako ravnotežni gubici mogu postati čimbenik na najvećim udaljenostima prijenosa, glavni čimbenici koji ograničavaju brzinu signalizacije s povećanjem udaljenosti variraju u vremenu. Ograničenja propusnosti kabela koja pogoršavaju vrijeme prolaza signala i uvode inter-simbolne smetnje (ISI) *(engl. Inter-symbol interference)*, primarni su čimbenici koji smanjuju dostižnu brzinu prijenosa signala kada se poveća udaljenost prijenosa. [6]

Za CAN sabirnicu, brzina prijenosa signala također se određuje iz ukupnog kašnjenja sustava. Između dva najudaljenija čvora sustava gleda se potez od jednog čvora do drugog i natrag. Također, mora se uzeti u obzir gubitak amplitude signala zbog otpora kabela i ulaznog otpora primopredajnika. [6]

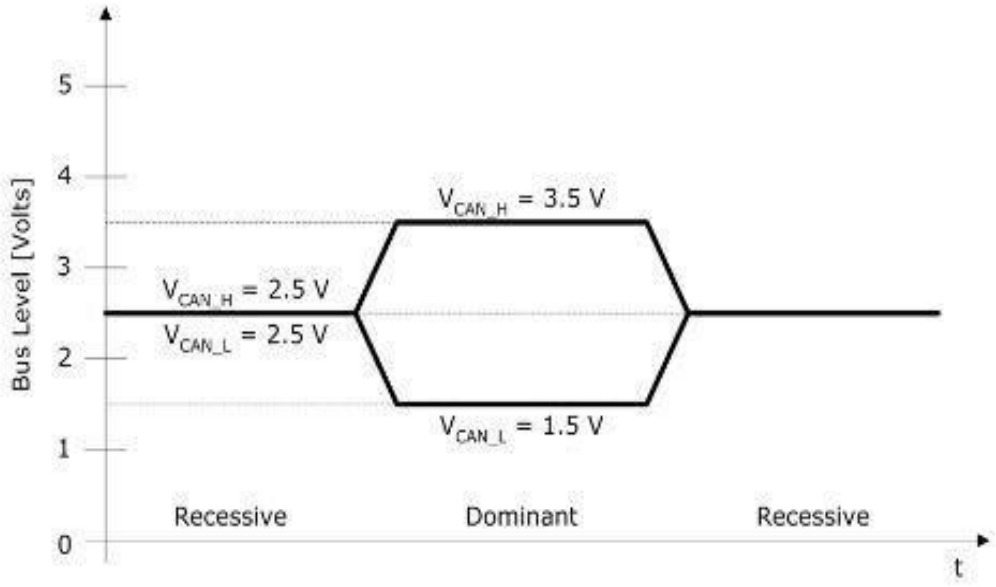
Konzervativno pravilo za duljinu sabirnice veće od 100 m izvedeno je iz umnoška brzine signalizacije u Mbps i duljine sabirnice u metrima, koja bi trebala biti manja ili jednaka 50.

$$\text{Brzina prijenosa signala (Mbps)} \times \text{duljina sabirnice (m)} \leq 50$$

Ako aplikacija zahtijeva sabirnicu od 1000 m, tada se prema ovoj aproksimaciji može sigurno koristiti brzina signalizacije od 50 kbps. [6]

3.1.2. CAN signal

Kao što je ranije navedeno, CAN je serijska, dvožilna, diferencijalna sabirnička tehnologija. To znači da se podaci šalju po jedan bit putem dva komplementarna signala CAN High (CANH) i CAN Low (CANL). Medij CAN sabirnice podržavaju dva logička stanja: recesivno i dominantno. Dominantno stanje nastaje kada se niska logička razina primijeni na prijenosni ulazni pin (obično se naziva TxD) primopredajnika. Recesivno stanje odgovara visokoj logičkoj razini na ulaznom pinu odašiljača primopredajnika.



Slika 3.5. Razina signala na CAN sabirnici [7]

Kao što vidimo, u recesivnom stanju i pinovi sabirnice CANH i CANL su postavljeni na istu razinu: ~ 2.5V. Tijekom dominantnog stanja, pin sabirnice CANH je postavljen na viši naponski potencijal (~ 3.5V), a pin sabirnice CANL je pomaknut na potencijal nižeg napona (~ 1.5V). Oduzimanjem naponskog potencijala dva pina sabirnice, možemo odrediti logičko stanje sabirnice pomoću jednadžbe.

Kad je vrijednost na sabirnici manja od 0,5 V, smatra se da je sabirnica u recesivnom stanju. Alternativno, vrijednosti veće od 0,9 V ukazuju na to da je sabirnica u

dominantnom stanju. Na kraju, za vrijednosti između 0,5 V i 0,9 V stanje sabirnice je nedefinirano. Budući da se razlika između dva signala koristi za definiranje stanja sabirnice, ovaj tip signalizacije poznat je kao diferencijalna signalizacija.

Dominantno stanje je označeno logičkom "0", dok je recesivno stanje označeno logičkom "1". U slučaju više spojenih uređaja na CAN sabirnicu napon linije je određen logičkim "I" *(tzv. Wired AND konfiguracija).*

Tablica 3.2. Wired AND konfiguracija

Čvorovi			
A	B	C	Sabirnica
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Razina sabirnice bit će na dominantnoj razini u slučaju da bilo koji broj tranzistora u mreži daje dominantnu (logička "0") razinu. Razina sabirnice bit će na recesivnoj razini samo kada svi tranzistori u mreži daju recesivnu (logičku "1") razinu. [7]

Prednost upotrebe diferencijalnog napona između CANH i CANL leži u otpornosti na elektromagnetske smetnje. Bilo koji elektromagnetski utjecaji utjecat će na obje žice na isti način, ali će diferencijalni napon ostati konstantan. [7]

3.1.3. Struktura bita

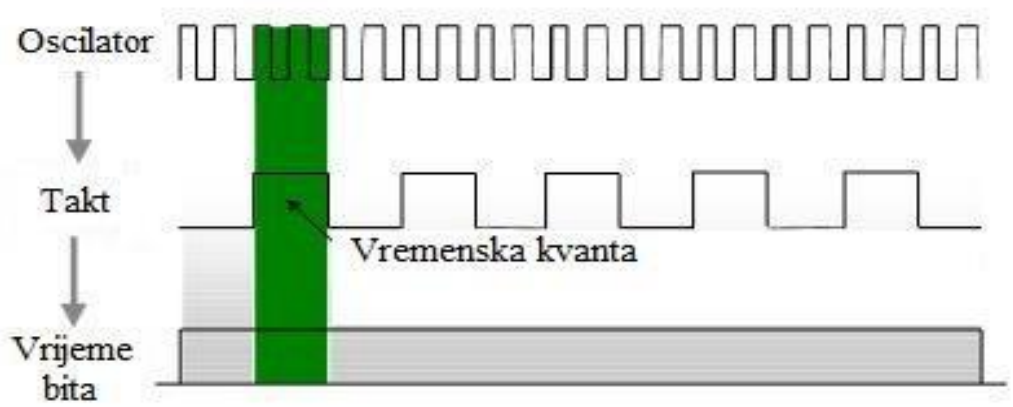
Svi čvorovi na CAN sabirnici moraju imati isti nominalnu brzinu prijenosa *(engl Nominal Bit Rate)*. CAN protokol koristi kodiranje Non-Return-to-Zero (NRZ), koji ne kodira takt unutar toka podataka. Stoga čvorovi koji primaju moraju oporaviti takt prijema i sinkronizirati ga sa taktom odašiljača. Budući da se oscilatori i vrijeme prijenosa mogu razlikovati od čvora do čvora, prijemnik mora imati neku vrstu "Phase-Locked Loop" (PLL) sinkroniziranu s rubovima prijenosa podataka kako bi se sinkroniziralo i održavalo takt prijemnika. Kada su se podaci NRZ kodirali, potrebno je uključiti umetanje bitova *(engl. Bit Stuffing)* kako bi se osiguralo da se takt pogodi.

Jedna od karakteristika koda Non-Return-to-Zero jest da signal ne pruža rubove koji se mogu koristiti za ponovnu sinkronizaciju ako se prenosi velik broj uzastopnih bitova s istim polaritetom. Stoga se umetanje bitova koristi za osiguravanje sinkronizacije svih čvorova sabirnice. To znači da tijekom prijenosa poruke maksimalno pet uzastopnih bitova može imati isti polaritet. Umetnuti bit u CAN okviru uključuje SOF, polje arbitraže, kontrolno polje, polje podataka i CRC polje. [8]

3.1.4. Vrijeme bita

Vrijeme bita sastoji se od segmenata koji se ne preklapaju. Svaki od ovih segmenata sastoji se od cjelobrojnih jedinica, nazvanih vremenska kvanta *(engl. Time Quanta)* (Tq). Vremenska kvanta je najmanja diskretna vremenska razlučivost koju koristi CAN čvor. Njegova duljina generira se programski podjelom frekvencije oscilatora CAN čvora. Postoji najmanje 8, a najviše 25 vremenskih kvanta po bitu. [8]

Nominalna brzina prijenosa *(engl. Nominal Bit Rate)* (NBR) definirana je u CAN specifikaciji kao broj bitova u sekundi, koje prenosi idealan odašiljač, bez ponovne sinkronizacije.



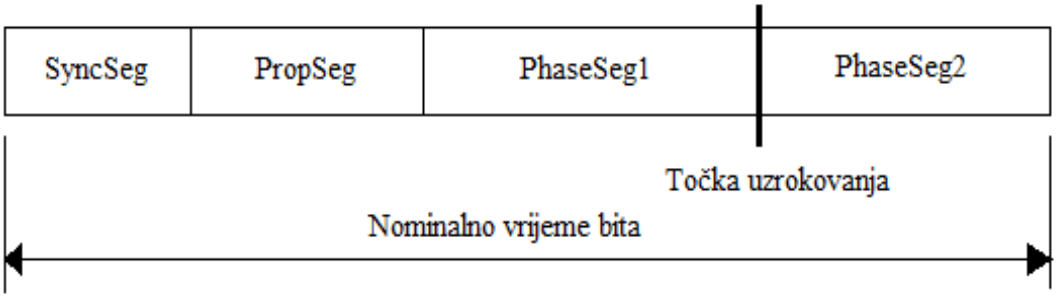
Slika 3.6. Vrijeme bita [8]

Nominalno vrijeme bita (NBT) ( ) sastoji se od 4 segmenata. Stoga je NBT zbroj sljedećih segmenata: [9]

- Segment Sinkronizacije *(engl. Synchronization Segment -)* - Segment sinkronizacije prvi je segment u NBT-u i koristi se za sinkronizaciju čvorova na sabirnici. Očekuje se da će se rubovi bitova pojaviti unutar SyncSega. Sinkronizacija će se ostvariti u odnosu na početni brid impulsa. Ovaj je segment fiksiran na 1 Tq

- Propagacijski Segment *(engl. Propagation Time segment -)* - Propagacijski segment postoji kako bi nadoknadio fizička kašnjenja između čvorova kao što je propagancijsko kašnjenje. PropSeg se može programirati od 1-8 TQs.

- Segment Faze 1 i Faze 2 *(engl. Phase Segment 1/2 -)* - Segment faze 1 i 2, koriste se za kompenzaciju pogrešaka rubne faze na sabirnici. PS1 se može produljiti (ili PS2 skratiti) resinkronizacijom. PS1 je moguće programirati od 1-8Tqs, a PS2 je moguće programirati od 2-8 Tqs.



Slika 3.7. CAN segmenti

- Točka uzrokovanja *(engl. Sample point)* - Točka uzorka je točka u vremenu bita u kojem se logička razina čita i tumači. Točka uzorka nalazi se na kraju PhaseSeg1. Izuzetak od ovog pravila je ako je način uzorkovanja konfiguriran za uzorkovanje tri puta po bitu. U ovom slučaju, dok je točka uzrokovanja na kraju PhaseSeg1, uzimaju se dva dodatna uzorka u polovici vremenskog kvanta prije kraja PhaseSeg1, pri čemu se vrijednost bita određuje većinskom odlukom. [9]

- Vrijeme procesiranja informacije *(engl. Information Processing Time)* - Vrijeme procesiranja informacija (IPT) je potrebno logičko vrijeme za određivanje razine bita od točke uzrokovanja. IPT započinje na točki uzorkovanja, mjeri se u Tq i fiksira se na 2 Tq. Budući da PhaseSeg2 također započinje na točki uzorkovanja i posljednji je segment u bitnom vremenu, potrebno je da minimum PhaseSeg2 ne bude manji od IPT-a. [9]

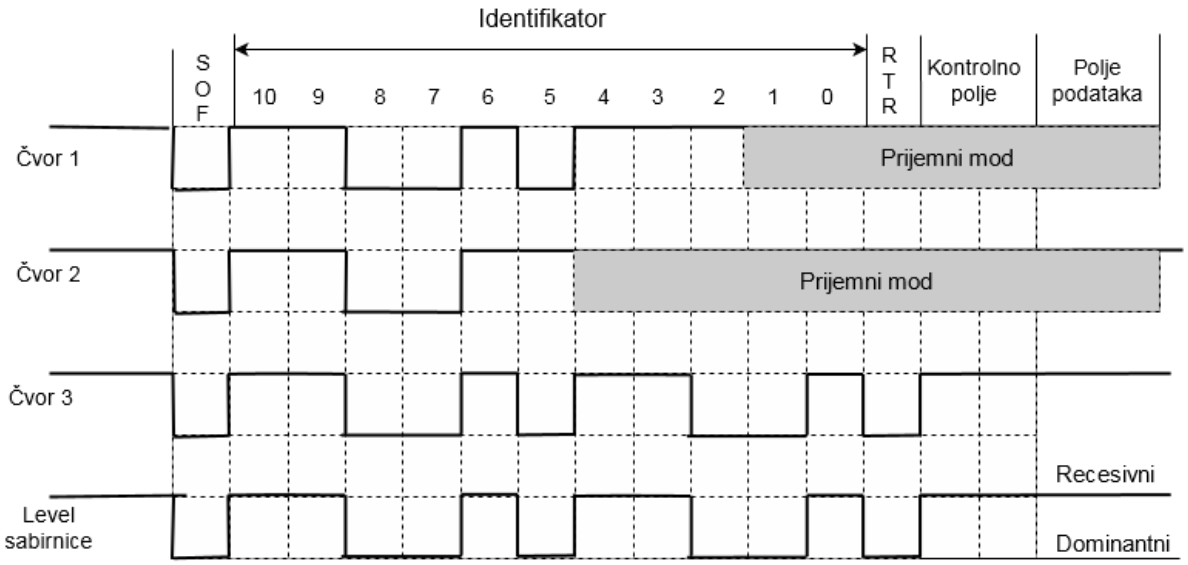
3.2. CAN podatkovni sloj

Sloj podatkovne veze standardiziran je u ISO 11898. Usluge sloja podatkovnih veza implementirane su u podslojevima Logical Link Control (LLC) i Medium Access Control (MAC) CAN kontrolera. LLC pruža filtriranje prihvaćanja, obavijesti o preopterećenju i upravljanje oporavkom. MAC je odgovoran za enkapsulaciju podataka (de-kapsulacija), kodiranje okvira (umetanje/uklanjanje), upravljanje pristupom mediju, otkrivanje pogrešaka, signalizacija pogreške i potvrđivanje. CAN Bus radi na konceptu *Broadcast* komunikacije što znači da svaka postaja mreže može slušati okvire odašiljačke stanice. Nakon primanja okvira, zadatak je svakog čvora da odluči treba li poruku prihvatiti ili ne. Dakle, filtriranje prihvaćanja mora biti implementirano u svaki CAN čvor. *Broadcast* komunikacija može se usporediti s radio stanicom koja emitira informacije o prometu za sve vozače. Svaki vozač mora odlučiti jesu li mu poruke važne ovisno o cesti koju želi koristiti. [9]

3.2.1. Arbitraža sabirnice

CAN protokol omogućuje istovremeni pristup sabirnici s različitih čvorova. Ako više čvorova pristupa sabirnici, potrebna je arbitraža. Metoda pristupa sabirnici koja se koristi u CAN-u je CSMA/CD+AMP *(engl. Carrier Sense Multiple Access with Collision Detection and Arbitration on Message Priority).* Prioritet poruke dekodira se u CAN identifikatoru. Kad je sabirnica u stanju mirovanja, nekoliko čvorova može započeti prijenos okvira. Svaki čvor čita sa sabirnice bit po bit, tijekom cijele poruke i uspoređuje prenesenu vrijednost bita s primljenom vrijednošću bita. Prema definiciji, primopredajnik mora osigurati da bitovi s dominantnom vrijednošću prepisuju one s recesivnom vrijednošću. [9]

Ako dva ili više čvorova sabirnice započnu svoj prijenos istodobno nakon što su utvrdili da je sabirnica u stanju mirovanja, sudar poruka izbjegnuto je implementiranom metodom pristupa sabirnici CMSA/CA+AMP. Svaki čvor šalje bitove svog identifikatora poruke i nadgleda razinu sabirnice. Sve dok su bitovi svih odašiljača identični, ništa se ne događa. [9]



Slika 3.8. CAN Bus arbitraža na sabirnici

Iz slike 14. vidimo da postupak arbitraže započinje istovremeno slanjem startnog bita okvira (SOF) na sabirnicu. Počevši od najznačajnijeg bita slijedi slanje identifikacijskih bita. Na bitu 5 čvorovi 1 i 3 šalju dominantni bit identifikatora. Čvor 2 šalje recesivni bit identifikatora, ali čita dominantni. Čvor 2 gubi arbitražu sabirnice i prebacuje se u prijemni mod rada koji prenosi recesivne bitove i samo može primiti poruku. Na bitu 2 čvor 1 gubi arbitražu protiv čvora 3. To znači da identifikator poruke čvora 3 ima nižu binarnu vrijednost i samim time veći prioritet od poruka čvorova 1 i 2. Na taj način čvor sabirnice s najvišom prioritetnom porukom pobjeđuje u arbitraži bez gubitka vremena i na taj način ne mora ponoviti poruku. Čvorovi 1 i 2 poslat će svoje poruke nakon što čvor 3 završi svoj prijenos i detektiraju neaktivnost linije. Neaktivnom linijom se smatra linija slobodna za slanje nakon što je čvor 3 poslao cijeli okvir i 3 bita razdvajanja okvira (IFS). [9]

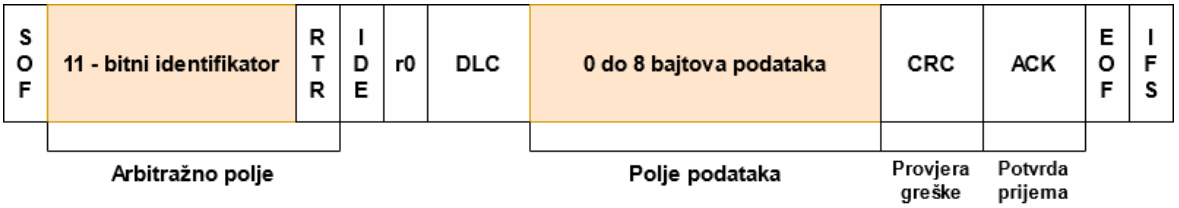
3.2.2. Struktura CAN poruke

CAN mreža može biti konfigurirana za slanje s dva formata poruke:

- standardni format okvira (CAN 2.0 A)
- prošireni format okvira (CAN 2.0 B)

CAN standardni okvir podržava duljinu od 11 bita za identifikator, a dok prošireni format podržava duljinu od 29 bita za identifikator koji je rastavljen na dva dijela: 11-bita osnovni identifikator i 18-bita prošireni identifikator. Razlika između formata osnovnog okvira i proširenog okvira postignuta je upotrebom IDE bita, koji se prenosi kao dominantan u slučaju 11-bitnog okvira, a prenosi se kao recesivni u slučaju 29-bitnog okvira.

3.2.3. Struktura CAN standardne poruke



Slika 3.9. Struktura standardne CAN poruke

- Početak okvira - SOF *(engl. Start-of-Frame)* 1-bit
  Ovaj bit označava početak poruke. Sinkronizira čvorove nakon razdoblja mirovanja.

- Polje identifikatora - 11 bita
  Polje identifikatora postavlja prioritet poruke. Niže vrijednosti znače veće prioritete.

- RTR *(engl. Remote Transmission Request)* - 1 bit
  Ovaj je bit dominantan kada informacije traži drugi čvor. Svi čvorovi će primiti zahtjev, ali identifikator određuje željeni čvor.

- Identifikator formata okvira - IDE *(engl. IDentifier Extension)* - 1 bit
  Označava koji format poruke se prenosi. Za standardni format koristi se dominantni bit.

- r0 - 1 bit
  Rezerviran bit za buduće namjene.

- Duljina polja podataka - DLC *(engl. Data Lenght Code)* - 4 bita
  Sadrži broj bajta u prijenosu.

- Polje podataka - 0-8 bajta
  Stvarni podaci koji se prenose određene DLC duljinom.

- Provjera greške - CRC *(engl. Cyclic Redundancy Check)* - 16 bita
  Sastoji se od 15 bita CRC niza i 1 CRC bit razdvajanja. Sadrži kontrolnu sumu (broj prenesenih bitova) prethodnih podataka aplikacije (od SOF do polja podataka) za prijenos otkrivanja pogrešaka. CRC bit razdvajanja je uvijek recesivan i ima ulogu da omogući dovoljno vremena za proračun CRC niza.

- Potvrda prijema - ACK *(engl. ACKnowledge)* - 2 bita
  Sastoji se od ACK bita i ACK bita razdvajanja, gdje se ACK bit i ACK bit razdvajanja šalju kao recesivan bit. Kada čvor uspješno primi poruku, to znači da je CRC provjera prošla i ACK polje potvrđuje tako da se ACK bit promijeni u dominantni bit. S druge strane, ako čvor pronađe pogrešku u poruci, dopušta da ACK bit ostane recesivan i ignorira poruku. Bit razdvajanja označuje razliku između prijema i mogućeg generiranog okvira greške.

- Završetak okvira - EOF *(engl. End-Of-Frame)* - 7 bita
  Završetak okvira je 7-bitno polje koje označava kraj svakog CAN okvira.

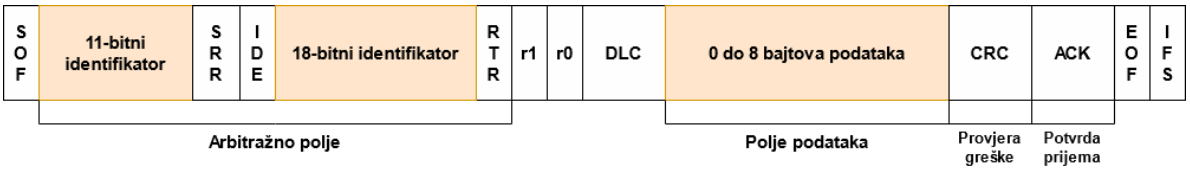
- Polje razdvajanja okvira - IFS *(engl. Intermission Frame Space)* - 3 bita
  IFS je vrijeme koje je potrebno za pomicanje okvira (poruke) u međuspremnik. IFS sadrži tri uzastopna recesivna (1) bita. Nakon što su prošla tri recesivna bita, kada se detektira dominantni bit, on postaje SOF bit sljedećeg okvira.

3.2.4. Struktura CAN proširene poruke

Jedina razlika između standardne i proširene CAN poruke je u 18-bitnom identifikatoru te sa dva dodatna polja:

- SRR *(engl. Substitute Remote Request)* - 1 bit
  Uvijek je u recesivnom stanju. Zajedno sa IDE bitom koji je u proširenom formatu okvira recesivan osigurava da je poruka koja se šalje standardnim okvirom uvijek većeg prioriteta.

- r1 - 1 bit
  Dodatni rezervni bit za buduće namjene.



Slika 3.10. Proširena CAN poruka

U proširenoj CAN poruci arbitražno polje je prošireno na 32 bita jer uključuje 29-bitni identifikator, te SRR, IDE i RTR bit.

Ako su im osnovni identifikatori jednaki prioritet će uvijek imati poruke sa standardnim formatom okvira.

CAN kontroleri koji podržavaju standardni format ne mogu primati poruke proširenog formata, dok CAN kontroleri koji podržavaju prošireni format mogu slati i primati poruke standardnog formata.

3.2.5. Vrste okvira

Postoje četiri tipa CAN okvira:

- Okvir podatka *(engl. Data Frame)*

Okvir podataka sastoji se od sedam polja: početak okvira (SOF), arbitraža, kontrola, podaci, provjera greške (CRC), potvrda prijema (ACK) i kraj okvira (EOF). Bitovi CAN poruke nazivaju se "dominantni" (0) ili "recesivni" (1). SOF polje sastoji se od jednog dominantnog bita. Svi mrežni čvorovi koji čekaju na prijenos sinkroniziraju se sa SOF-om i započinju s emitiranjem u isto vrijeme. Shema arbitraže određuje koji čvorovi će zapravo kontrolirati sabirnicu. [10]

Kontrolno polje okvira podataka sastoji se od 6 bitova (od kojih se koriste samo donja 4) koji ukazuju na količinu podataka u poruci. Budući da se u jednoj poruci može poslati do 8 bajtova podataka, kontrolno polje može imati vrijednosti u rasponu od 000000 do 000111. Podaci koji se prenose sadržani su u polju podataka. Najprije se šalje najznačajniji bit bajta podataka. [10]

CAN implementira pet razina otkrivanja pogrešaka. Na razini poruke izvodi cikličke provjere greške, provjere okvira i provjere potvrde. Provjere razine bitova sastoje se od nadzora bita *(engl. Bit Monitoring)* i greške umetanja bita *(engl. Bit Stuffing Error)*. Provjere greške otkrivaju se pomoću 15-bitnog CRC-a koji izračunava odašiljač iz sadržaja poruke. Svaki primatelj koji prihvaća poruku ponovno izračunava CRC i uspoređuje ga s prenesenom vrijednošću. Neusklađenost između dva izračuna uzrokuje postavljanje zastavice pogreške. Provjere okvira koje će označiti pogrešku su primanje neispravnog bita u CRC okviru, ACK okviru, EOF okviru. Na kraju, svaki prijemni čvor upisuje dominantan bit u ACK dio poruke koji čita odašiljač. Ako poruka nije potvrđena (možda zato što prijemnik nije uspio), označava se pogreška ACK. [10]

Na razini bita već smo primijetili da odašiljač "čita" svaki preneseni bit. Ako se nadgledana vrijednost razlikuje od vrijednosti koja se šalje, otkriva se pogreška bita. Uz to, pogreške u bitu otkrivaju se umetanjem. Nakon što je preneseno pet uzastopnih identičnih bitova, odašiljač će umetnuti bit suprotnog polariteta u tok bitova (bitovi se pune iz SOF-a kroz CRC polje). Prijemnici automatski "uklanjaju stvari" iz poruke. Ako bilo koji čvor otkrije šest uzastopnih bitova iste razine, označava se greška umetanja bita. [10]

- Okvir upita *(engl. Remote Frame)*

  Čvor koji zahtijeva podatke s drugog čvora na mreži može zatražiti prijenos slanjem okvira upita. Na primjer, mikroprocesor koji upravlja središnjim

zaključavanjem našeg automobila možda će trebati znati stanje brzine mjenjača s kontrolera pogonskog sklopa (je li automobil parkiran?). [10] Dvije su razlike između okvira podataka i okvira upita. Prvo, RTR-bit u okviru podatak se prenosi kao dominantni bit,a u okviru upita kao recesivni bit, a drugo u okviru upita nema podatkovnog polja. U slučaju da okvir podataka i okvir upita počnu istodobno prenositi, okvir podatka pobjeđuje u arbitraži zbog dominantnog RTR bita. Dakle, čvor koji je poslao okvir upita odmah prima željene podatke.

- Okvir greške *(engl. Error Frame)*

  Ako čvor odašiljanja ili prijema otkrije pogrešku, odmah će prekinuti prijenos i emitirati okvir pogreške koji se sastoji od polja zastavice greške *(engl. Error Flag)* koja se sastoji od šest dominantnih bitova i polje razdvajanja greške *(engl. Error Delimiter)* sastavljene od osam recesivnih bitova. Budući da ovaj bitni niz krši pravilo umetanja bitova, svi ostali čvorovi reagiraju i prenošenjem polja zastavice greške. Nakon što se otkrije dovoljan broj pogrešaka, čvor će se na kraju sam isključiti. Robusnost, posebno u proizvodnim i automobilskim okruženjima u kojima prevladava CAN, zahtijeva da mreža utvrdi jesu li pogreške prolazne (zbog skokova napona, buke ili nekih drugih privremenih stanja) privremeno stanje) ili trajni kvar čvora zbog neispravnog hardvera. Slijedom toga, čvorovi pohranjuju i prate broj otkrivenih pogrešaka. [10]

- Okvir preopterećenja *(engl Overload Frame)*

  Ako CAN čvor prima poruke brže nego što ih može obraditi, generirat će se okvir preopterećenja kako bi se osiguralo dodatno vrijeme između uzastopnih podataka ili udaljenih okvira. Slično okviru greške, okvir preopterećenja ima dva polja:

  - Polje zastavice preopterećenja *(engl. Overload Flag)* koja se sastoji od šest dominantnih bitova
  - Polje razdvajanja preopterećenja *(engl. Overload Delimiter)* koji se sastoji od osam recesivnih bitova.

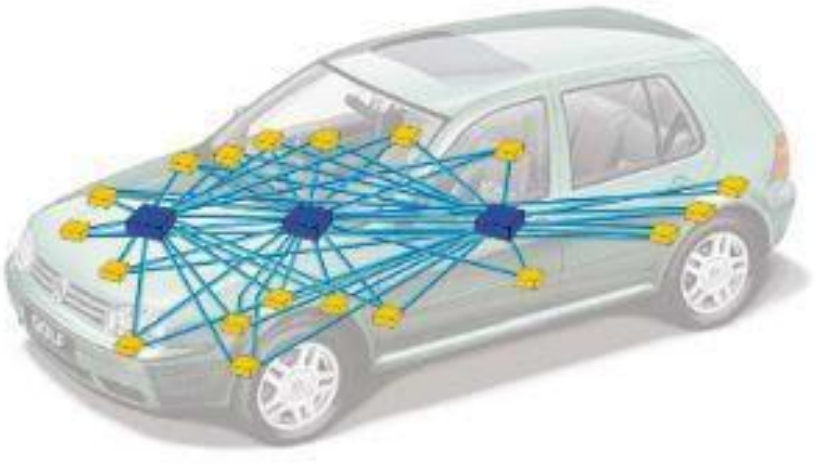
3.3. Upotreba CAN Bus u automobilu

Uporaba CAN Bus protokola u automobilu omogućuje spajanje elektroničkih modula poput elektroničkih upravljačkih jedinica ili inteligentnih senzora.

Neki od prednosti CAN BUS komunikacije u automobilu:

- Razmjena podataka između upravljačkih jedinica odvija se na jedinstvenoj platformi. CAN djeluje kao takozvana magistrala podataka.
- Sustavi koji uključuju nekoliko upravljačkih jedinica, npr. ESP[1] *(engl. Electronic Stability Programme)*, može se učinkovito implementirati.
- CAN je otvoreni sustav koji omogućuje prilagodbu različitim prijenosnim medijima poput bakrenih ili optičkih vlakana.
- Dijagnoza više sustava moguća je na nekoliko kontrolnih jedinica.

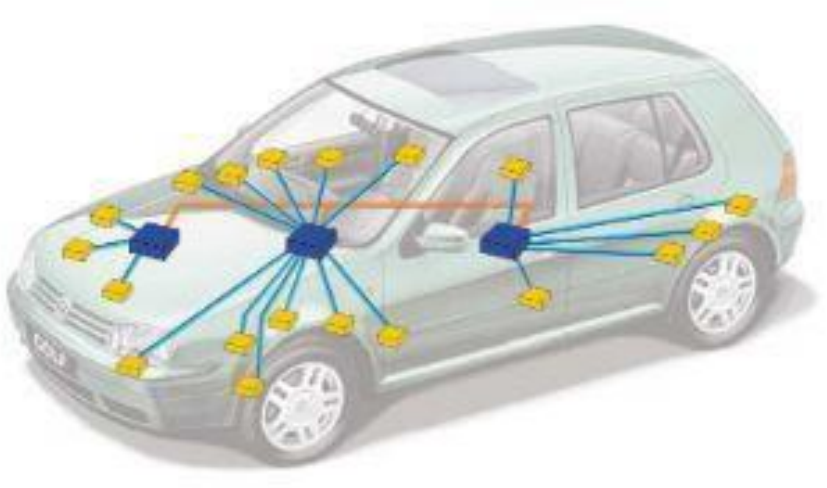
Elektronički moduli u vozilima bez CAN BUS protokola morali bi komunicirati koristeći izravne analogne signale vodova od točke do točke. Za svaki modul kojem je potrebna direktna povezanost za komunikaciju, ne samo da je dugotrajan, već će biti i neuredan sa svom prekomjernom količinom ožičenja. Također bi možda postojala nepouzdana komunikacija između uređaja. Prekomjerne žice mogu zahtijevati dodatnu opremu, što može stvarati probleme s troškovima. [4]



Slika 3.11. Auto sa 3 kontrolne jedinice [11]

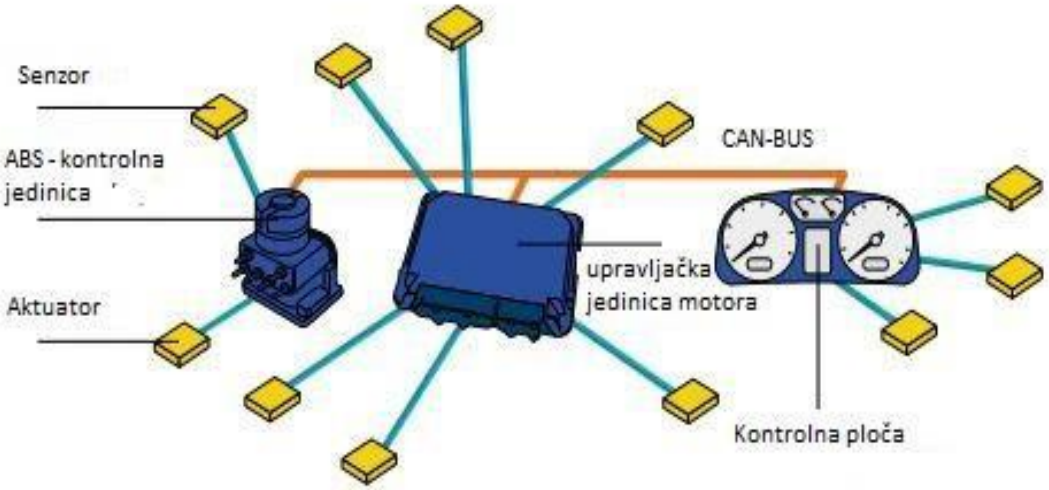
[1] ESP *(engl. Electronic Stability Programme*) je elektronički sustav za poboljšanje dinamičke stabilnosti vozila otkrivanjem i smanjenjem gubitka vuče. Pulsirajućim kočenjem sprječava nekontrolirano klizanje guma.

S druge strane, s uporabom CAN Bus protokola, on eliminira potrebu za svim tim ožičenjima i omogućava elektroničkim uređajima da međusobno komuniciraju jednom multipleksnom žicom koja povezuje svaki čvor u mreži s glavnom nadzornom pločom kao što se vidi na slici 3.112 [4]



Slika 3.12. Auto sa 3 kontrolne jedinice i CAN BUS protokolom [11]

Multipleksna žica i arhitektura omogućuju kombiniranje i prijenos signala po cijeloj mreži, dok istovremeno osiguravaju da svaki elektronički modul u vozilima prima podatke od senzora i aktuatora. To nam omogućava da možemo spojiti bilo koji broj elektroničkih upravljivih jedinica u vozilu putem dvožične sabirnice. Elektronička upravljiva jedinica može koristiti podatke s druge jedinice što eliminira potrebu za instaliranjem istih senzora u više uređaja. Također, putem softvera moguće je dodavanje novih značajki u sustav automobila. [4]



Slika 3.13. CAN mreža sklopa s 3 upravljačke jedinice [11]

4. ŽIČNA CAN BUS KOMUNIKACIJA IZMEĐU SENZORSKIH ČVOROVA

U ovom dijelu rada opisat će se izrada prototipa žične komunikaciju između senzorskog čvora i upravljačke ploče korištenjem CAN-BUS protokola, koja će služiti za potrebe razvoja Buggy vozila. Senzori će služiti za očitanje parametara kao što su temperatura motora, napon baterije, akceleracija i sl. Podaci su prikazani na grafičkom LCD-u na upravljačkoj ploči. Osim LCD-a, upravljačka ploča će biti opremljena i tipkama za upravljanje svjetlima i ostalim sustavima.

Za izradu ovog prototipa koristit ćemo sljedeće module:

- Arduino Nano 5V
- CAN BUS modul MCP2515 5V
- CJMCU-2812-8 8 WS2812 5050 RGB LED
- Ultrazvučni senzor HC-SR04
- Piezo Buzzer HP1470X
- DHT22 senzor temperature i vlažnosti zraka
- LCD displej

4.1. Arduino Nano

Arduino je elektronička platforma otvorenog koda koja se temelji na jednostavnom hardveru i softveru. Arduino ploče mogu čitati ulazne signale kao npr. svjetlo na senzoru, prst na gumbu ili čak Twitter poruku i zatim je pretvoriti u izlaz aktivirajući motor, paleći LED, objavljujući nešto na mreži. Sve to jednostavno možemo reći svojoj ploči što treba učiniti slanjem skupa uputa mikrokontroleru na ploči. Da bismo to učinili, koristimo programski jezik Arduino i Arduino softver.

U ovom radu Arduino će se koristit kao ECU na našem Buggy vozilu. Upravljat će se paljenje prednjih i zadnjih svjetla, žmigavci, također će se koristit za upravljanje senzorima (senzor parkinga, senzor temperature i vlažnosti)

Arduino je rođen u Institutu za dizajn interakcija u Ivrea kao jednostavan alat za brzo prototipiranje, namijenjen studentima bez iskustva u elektronici i programiranju.

Tijekom godina Arduino je bio glavni izvršitelj tisuća projekata, od svakodnevnih predmeta do složenih znanstvenih instrumenata. Studenti, umjetnici, programeri i profesionalci okupili su se oko ove platforme otvorenog koda, njihovi su doprinosi dodali nevjerojatnu količinu pristupačnog znanja koje može biti od velike pomoći kako novacima, tako i stručnjacima.

Početnicima Arduino softver je vrlo jednostavan za upotrebu, a opet dovoljno fleksibilan za napredne korisnike. Radi na Mac, Windows i Linux sustavima. Profesori i studenti koriste ga za izgradnju jeftinih znanstvenih instrumenata, za dokazivanje principa kemije i fizike ili u svrhu programiranja i robotike. [12]

Arduino nudi određenu prednost nastavnicima, studentima i zainteresiranim amaterima u odnosu na druge sustave:

- Niska cijena
- Radi na više platforma (Windows, Mac, Linux)
- Jednostavno, jasno programsko okruženje
- Softver otvorenog koda
- Hardver otvorenog koda

Arduino Nano je mali mikrokontroler, kompatibilna, fleksibilna i prilagodljiva ploči, koju je razvio Arduino.cc. Dolazi s radnim naponom od 5V, međutim, ulazni napon može varirati od 7 do 12V. Arduino Nano sadrži 14 digitalnih pinova, 8 analognih pinova, 2 resetirajuća i 6 naponskih pinova. Svaki od ovih digitalnih i analognih pinova ima više funkcija, ali njihova glavna funkcija mora biti konfigurirana kao ulaz ili izlaz. Djeluju kao ulazni pinovi kada su povezane sa senzorima, ali ako je spojeno neko opterećenje, koristimo ih kao izlaz. [13]

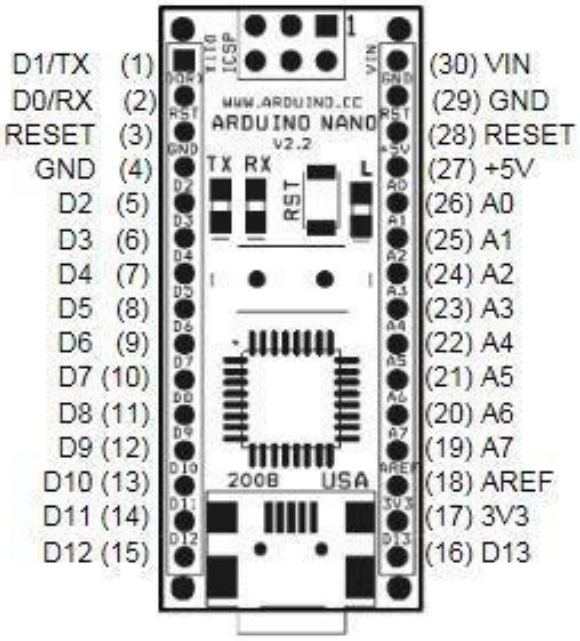
Postoji jedno ograničenje korištenja Arduino Nano. Ne dolazi sa DC utičnicom, što znači da vanjski izvor napajanja ne možete napajati putem baterije. Za napajanje se koristi Mini-B USB standard.

Male dimenzije i prirodno prilagođena pločama čine ovaj uređaj idealnim izborom za većinu aplikacija u kojima je veličina elektroničkih komponenata bitna.

Flash memorija je 16 KB ili 32 KB, što sve ovisi o ugrađenom ATmega mikrokontorleru, tj. Atmega168 dolazi s 16 KB flash memorije, dok Atmega328 dolazi s flash memorijom od 32 KB. Flash memorija koristi se za pohranu koda. Za rad uređaja koristi se 2 KB memorije od ukupne flash memorije. [13]

Tablica 4.1. Arduino Nano tehničke specifikacije

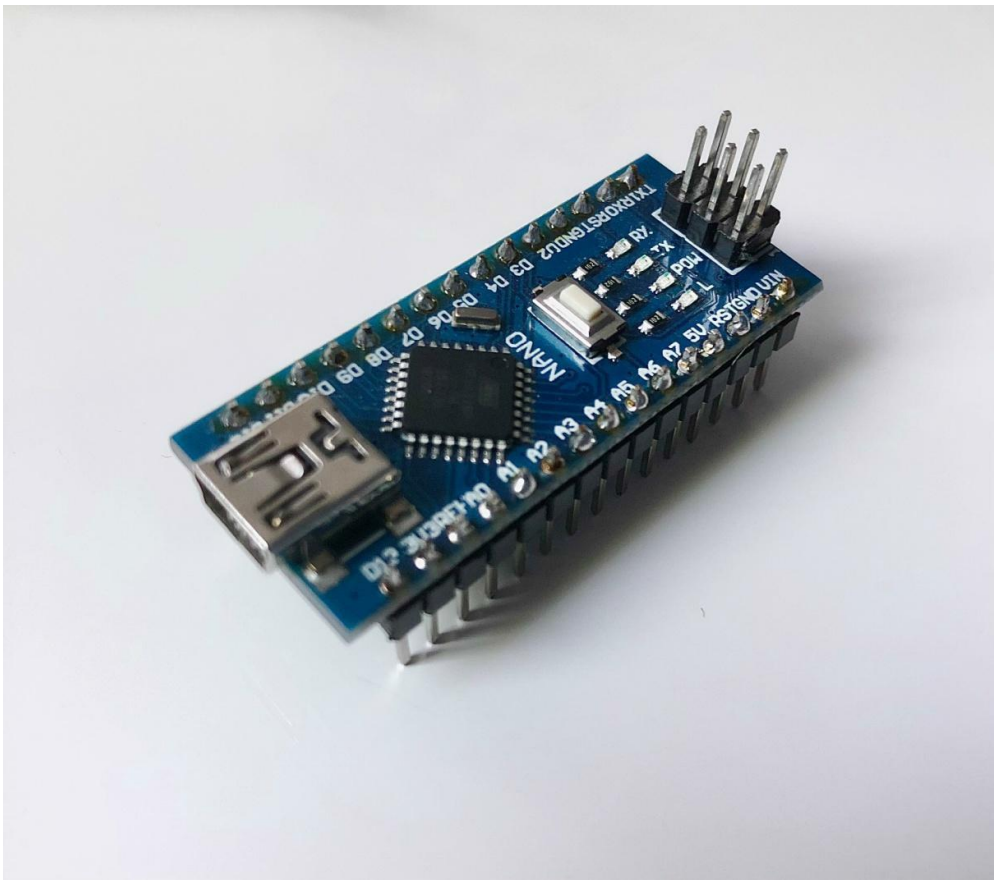
Mikrokontroler	ATmega328
Radni napon	5 V
Flash memorija	32 KB
SRAM	2 KB
Brzina sata (Clock Speed)	16 MHz
Analogni I/O pinovi	8
EEPROM	1 KB
DC struja po I/O pinovima	40 mA (I/O pinovi)
Ulazni napon	7-12 V
Digitalni I/O pinovi	22
PWM izlaz	6
Potrošnja energije	19 mA
PCB veličina	18 x 45 mm
Težina	7 g



Slika 4.1. Arduino Nano pinovi [14]

Tablica 4.2. Arduino Nano opis pinova

Pin	***Ime***	***Opis***
1-2, 5-16	*D0 - D13*	*Digitalni I/O*
3, 28	*Reset*	*Reset*
4, 29	*GND*	*Uzemljenje*
17	*3V3*	*+3,3V izlaz*
18	*AREF*	*ADC referenca*
19-26	*A7 - A0*	*Analogni ulazi*
27	*+5V*	*+ 5V izlaz (iz ugrađenog regulatora) ili + 5V (ulaz iz vanjskog napajanja)*
30	*VIN*	*Napon napajanja*



Slika 4.2. Arduino Nano

4.2. CAN BUS modul MCP2515

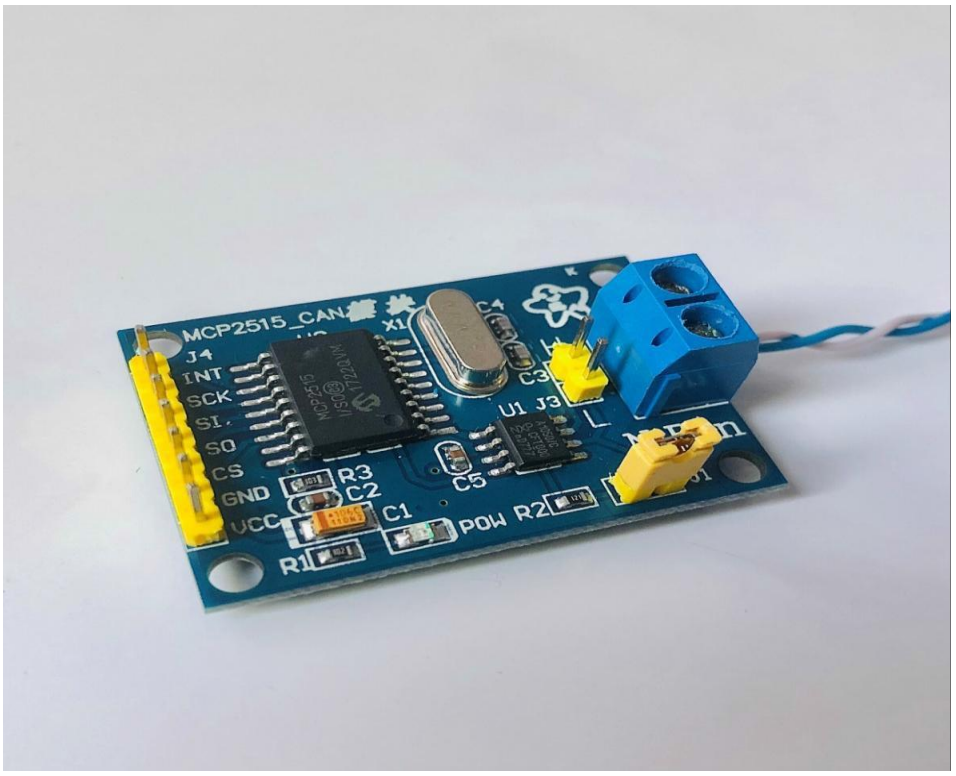
MCP2515 modul tvrtke Microchip Technology samostalni je kontroler koji implementira CAN specifikaciju, verzija 2.0B za brzine od 1 Mb/s. Sposoban je za prijenos i primanje standardnih i proširenih podataka i okvira upita. MCP2515 ima dvije maske za prihvaćanje i šest filtara za prihvaćanje koji se koriste za filtriranje neželjenih poruka, čime se smanjuje opterećenje gčavnpog mikrokontrolera. MCP2515 se povezuje s mikrokontrolerima putem industrijskog standardnog serijskog perifernog sučelja (SPI) *(engl. Serial Peripheral Interface)*. Modul je također opremljen i CAN primopredajnikom TJA1050, čija uloga je prilagodba pretvoriti digitalne podatke u diferencijalnu signalizaciju koja se koristi na žicama CAN sabirnice. Sabirnica se sastoji od dva krajnja čvora koja se nalaze na krajevima sabirnice. Kabel upletene parice koji je obično zaštićen spaja dva čvora. Za kućnu upotrebu, bilo koje dvije žice će koristiti za povezivanje čvorova. Na krajevima sabirnice čvorovi imaju otpor od 120 Ω. Oni se jednostavno dodaju u krug pomoću

kratkospojnika koji dolazi sa modulom. MCP2515 obrađuje osnovno rukovanje porukama s nekoliko međuspremnika za prijenos i primanje, rukovanje sabirnicom, otkrivanje sudara/kolizije podataka i otkrivanje CRC pogrešaka. Ova hardverska razina također ima mogućnost filtriranja dolaznih poruka na temelju ID-a poruke i prosljeđivanje poruka od interesa samo pridruženom mikrokontroleru. Na odlaznim porukama može prioritet dati porukama i osigurati da se poruke najvišeg prioriteta stave na sabirnicu ispred poruka niskog prioriteta. [15]

Ključne značajke MCP2515 modela:

- Brzina do 1 Mb/s
- Duljina sabirnice do 1000 m
- Standardni okvir, produženi okvir i okvir upita
- Izlaz prekida *(engl. Interrupt output)*
- SPI sučelje
- Radni napon 5 V

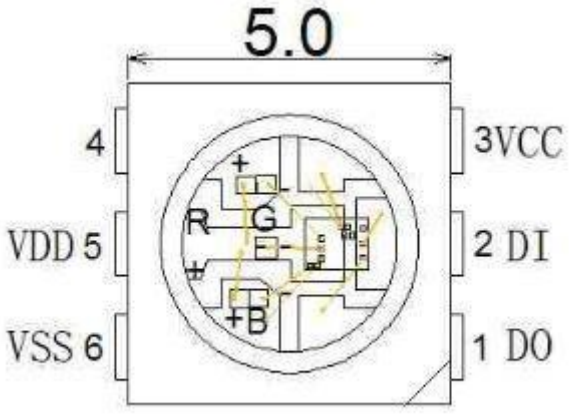
Povezanost sa mikrokontrolerom ostvarena je putem 7-pinskog muškog zaglavlja, koja predstavlja SPI komunikaciju između mikrokontrolera i MCP2515. CAN sabirnica može se povezati pomoću vijčane stezaljke ili sa dva muška pina.



Slika 4.3. CAN Bus modul MCP2515

4.3. CJMCU-2812-8 8 WS2812 5050 RGB LED

WS2812 je inteligentni upravljački LED izvor svjetlosti u koji su integrirani upravljački krug i RGB čip paket čije je kućište 5050 (5.0 x 5.0 mm). Uključuje inteligentni digitalni priključak za prikupljanje podataka i pogonski krug za pojačavanje signala i preoblikovanje signala. Također uključuje precizni unutarnji oscilator i 12-voltni programski dio za kontrolu konstantne struje, koji učinkovito osigurava jačinu i kvalitetu svjetlosti jednog piksela. Unutar kućišta nalaze se 3 LED diode (crvena, zelena i plava). WS2812 komunicira jedno-žični način preko NZR komunikacijskog protokola. LED trake moguće je kaskadno spojiti pomoću pinova Din (Data IN) i Dout (Dana OUT) tako da Dout pin prethodne LED trake spojimo na Din sljedeće. Nakon resetiranja i uključivanja piksela, Din priključak neprestano prima podatke od mikrokontrolera koji šalje niz logičke nule i logičke jedinice. Prvi piksel prikuplja početna 24 bita podatka, odnosno 3 bajta (po bajt za svaku boju G, R, B), a zatim se šalje u interni spremnik podataka i ostali podaci u unutarnjem krugu za pojačanje i preoblikovanja signala šalju se na sljedeću kaskadu piksela kroz Dout port, sve redom do zadnjeg piksela. [16]



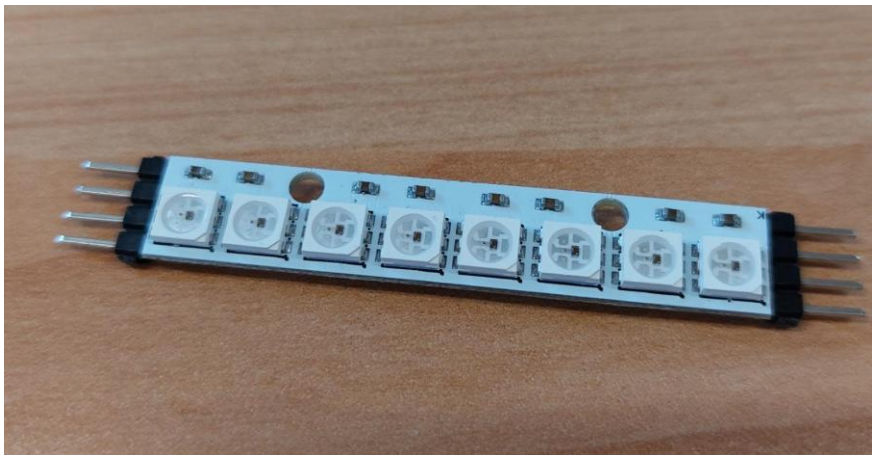
Slika 4.4. RGB kućište i pinovi [16]

Tablica 4.3. Opis pinova

Pin	Opis pina
DOUT	Upravljanje izlaznog signala
DIN	Upravljanje ulaznog signala
VCC	Napajanje kruga
VDD	Napajanje LED
VSS	Uzemljenje

WS2812 zahtjeva napajanje oko 5 V. WS2812 bi svakako trebao raditi negdje između 4 V i 7 V. Tijekom uporabe svih LED lampica na najjačoj svjetlini, svaka ploča može povući oko 60 mA, što bi značilo 20 mA po kanalu u boji. Svaki piksel od tri osnovne boje može postići 256 razina svjetline, što znači 16 777 216 različitih boja.

Ovakve opisane LED trake koristit će se u svrhu izrađivanja prototipa svijetla na Buggy vozilu. S njima će se upravljati žmigavci, svijetla i stop svijetla.



Slika 4.5. LED traka

4.4. Ultrazvučni senzor HC-SR04

Ultrazvučni senzor HC - SR04 za mjerenje udaljenosti koristi ultrazvučne valove kako bi odredio udaljenost od premeta [17]. Pruža mogućnost mjerenja udaljenosti od 2 do 400 cm, a točnost dosega može biti i do 3 mm. Modul uključuje ultrazvučne odašiljače, prijamnik i upravljački krug. Osnovno načelo rada: [18]

- Koristi IO prekidač za signale visoke razine od najmanje 10 µs.
- Modul automatski šalje osam impulsa od 40 kHz i otkriva postoji li povratni signal impulsa
- Testna udaljenost = (vrijeme visoke razine x brzina zvuka (340 m/s))/2

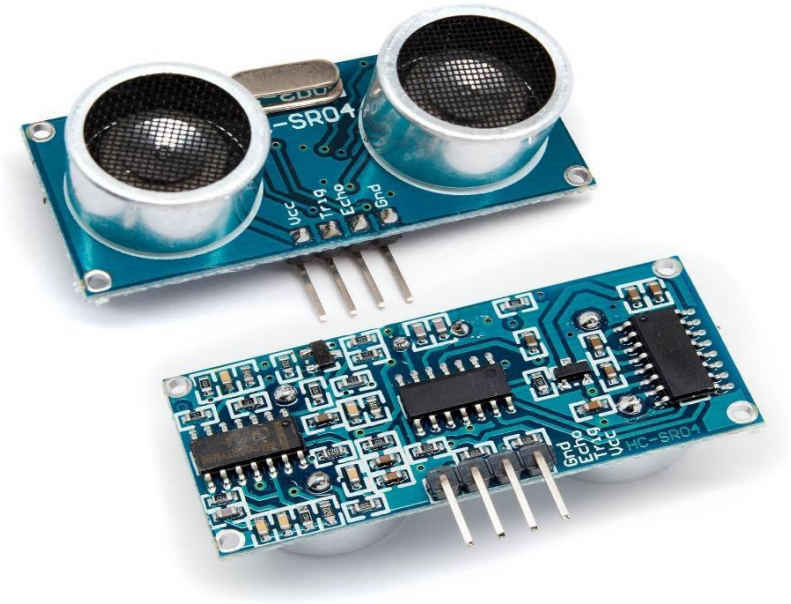
Žično povezivanje HC-SR04:

- VCC - 5 V napajanje
- Trig - Prekidač ulaznog impulsa
- Echo - Refleksija izlaznog impulsa
- GND - Uzemljenje

Električni parametri:

Tablica 4.4. Električni parametri senzora

Radni napon	DC 5 V
Radna struja	15 mA
Ultrazvučna frekvencija	40 kHz
Maksimalni domet	4 m
Minimalni domet	2 cm
Efektivni kut mjerenja	
Dimenzije	45*20*15 mm
Preciznost	3 mm

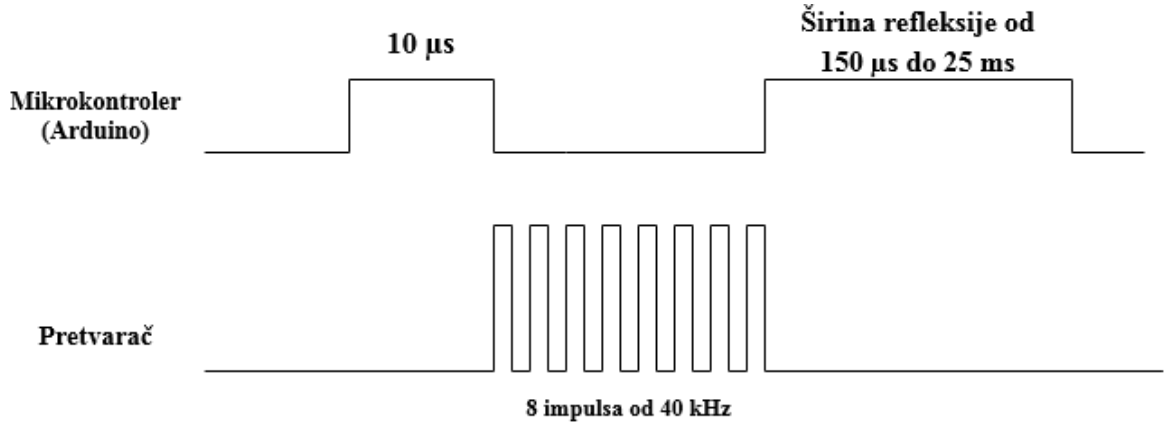


Slika 4.6. Ultrazvučni senzor

4.4.1. Princip rada modula

Princip rada vrši se preko dva osnovna pina, *Trig* i *Echo*. Mikrokontrolerom (u našem slučaju Arduino modul) na ulaz pina *Trig* šaljemo 5 V koji će unijeti samo kratki impuls od 10 µs. Na taj način uključujemo ultrazvučni pretvarač koji će zatim poslati 8 impulsa ultrazvuka od 40 kHz i čeka njihovu refleksiju. Preko *Echo* pina reflektirani impuls šalje se

nazad mikrokontroleru tek kad ga senzor registrira. Ovako navedeni podaci su zapravo vrijeme trajanja reflektiranog impulsa koji može trajati od 150 µs do 25 ms. S druge strane ako refleksija traje duže od 35 ms, senzor će od reagirati kao da predmet nije u dometu. [18]



Slika 4.7. Vremenski dijagram senzora

4.5. Piezo Buzzer HP1470X

Piezo Buzzer je vodootporna zujalica malih dimenzija koja će se koristit kao zvučna signalizacija u Buggy vozilu.

Slika 4.8. Piezzo Buzzer

Neke od električnih karakteristika navedeni su u tablici:

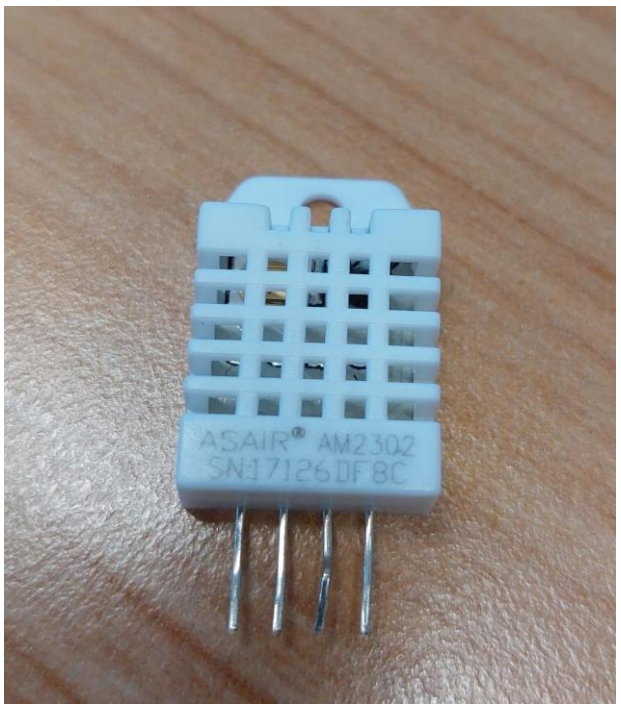
Tablica 4.5. Specifikacije

Min. izlazni zvuk na 10 cm	80 dB
Napon	5 V
Radni napon	3~9
Rezonantna fekvencija	4000±500 Hz
Težina	1 g
Dimenzija	14x7,5 mm

4.6. DHT22 senzor temperature i vlažnosti zraka

DHT22 je senzor temperature i vlažnosti. Senzor dolazi s namjenskim NTC termistorom za mjerenje temperature i 8-bitnim mikrokontrolerom koji služi za očitanje izlazne vrijednosti temperature i vlažnosti. Senzor je također tvornički kalibriran i stoga je jednostavan za povezivanje s drugim mikrokontrolerima. Senzor može mjeriti temperaturu od -40 °C do 80 °C i vlažnost od 0% do 100% s točnošću od ± 1 °C i ± 1%. [20]

Mala veličina, mala potrošnja i mogućnost prijenosa podataka na velike udaljenosti (20 m) omogućuju da DHT22 bude prikladan za sve vrste primjena, čak i u industriji.



Slika 4.9. DHT22

Tablica 4.6. Specifikacije

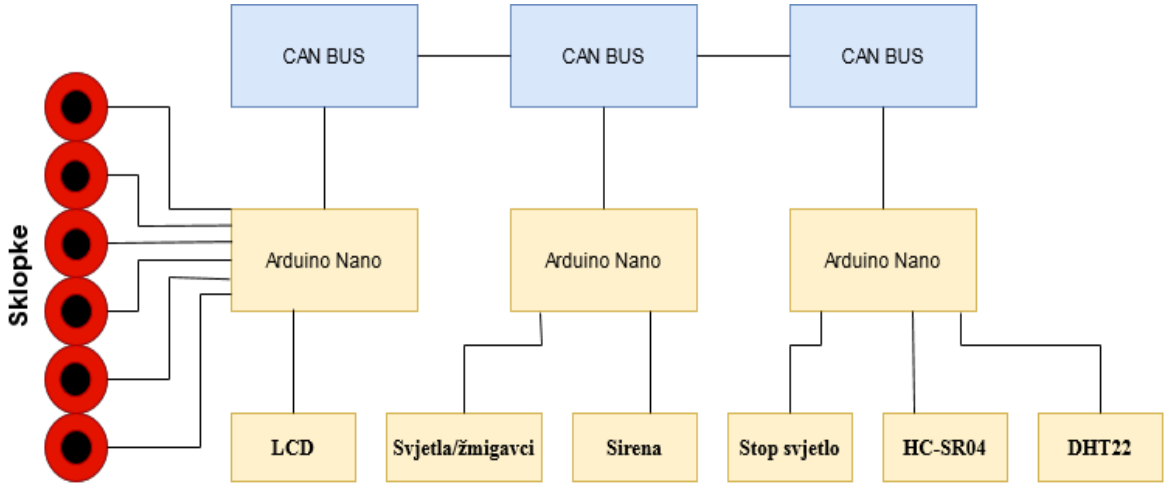
Radni napon	3.5 V do 5.5 V
Radna struja	0.3 mA (dok mjeri) 60 µA (u mirovanju)
Temperaturni domet	-40°C do 80°C
Domet vlažnosti	0% do 100%
Točnost	±0,5°C i ±1%
Dimenzije	14*18*5.5 mm

Tablica 4.7. Opis pinova

Pin	Opis
Vcc	Napajanje 3.5 V do 5.5 V
Data	Serijski izlaz podataka temperature i vlažnosti
NC	Nema konekcije i stoga se ne koristi
GND	Uzemljenje

4.7. Opis praktičnog rada

U ovom poglavlju opisat će se razrada protokola CAN Bus komunikacije koja će služit za daljnji razvoj Buggy vozila.



Slika 4.10. Shema spoja

Na slici 4.7. prikazana je arhitektura sustava. Sustav se sastoji od tri zasebna modula, koji su povezani putem CAN sabirnice. Preko Arduino Nano mikrokontrolera upravljat će se sa ulaznim i izlaznim funkcijama koje će preko CAN sabirnice komunicirati. Šest tipki koristit će se kao ulazne varijable preko kojih ćemo određivati koji senzor ili svijetlo želimo aktivirati. Svaka tipka upravljat će jednim senzorom ili svjetlom. Prilikom pritiska na jednu od tipki aktiviramo jedan od senzora, povratnu informaciju da je senzor aktivan dobit ćemo prikazom određenog simbola na displeju. Također, pritiskom na tipke koje uključuju senzor udaljenosti i senzor temperature, njihove vrijednosti biti će prikazane da displeju.

U nastavku će biti prikazan kod za prvi Arduino na kojeg su spojene tipke i LCD.

```
#include <CAN.h>      //Biblioteka za CAN modul

#include <LiquidCrystal_I2C.h>        //Biblioteka      za      I2C
komunikaciju sa LCD-om i Arduinom



LiquidCrystal_I2C lcd(0x27, 16, 2);         //Postavljanje    adrese
LCD-a



// Inicijalizacija varijabli za smještaj podataka o udaljenosti,
vlaznosti i temperaturi



int distance;

int humidity;

int temperature;

//Pomoću bitova definirati će se izgled svakog simbola

byte TurnLights[] = {

  B00000,

  B00000,

  B00100,

  B01111,

  B11111,

  B01111,

  B00100,
```

```
  B00000
};
byte Stop[] = {
  B00000,
  B01110,
  B11111,
  B11111,
  B11111,
  B01110,
  B00000,
  B00000
};
byte Lights[] = {
  B00000,
  B01110,
  B11111,
  B11111,
  B01110,
  B00100,
  B01110,
  B01110
};
byte Siren[] = {
  B00100,
  B10110,
  B01111,
  B00111,
  B00111,
  B01111,
  B10110,
```

```
  B00100
};
#define CAN_BUTTONS 0x12        //Definirana adresa za sklopke
// definiranje funkcionalnosti pojedinih pinova
// umjesto broja pojedinog pina koristimo ime varijable
#define TURN_LIGHTS 3
#define LIGHTS 4
#define SIREN 5
#define STOP_LIGHTS 6
#define TEMPERATURE 7
#define PARKING 8

void setup() {
//Postavljanje sklopki kao ulaz
  pinMode(TURN_LIGHTS, INPUT);
  pinMode(LIGHTS, INPUT);
  pinMode(SIREN, INPUT);
  pinMode(STOP_LIGHTS, INPUT);
  pinMode(TEMPERATURE, INPUT);
  pinMode(PARKING, INPUT);



  lcd.init();         //Inicijalizacija LCD-a
  lcd.backlight();    //Upaljeno svjetlo na displeju LCD-a
  lcd.createChar(0, TurnLights);      //Kreiranje simbola
  lcd.createChar(1, Lights);
  lcd.createChar(2, Siren);
  lcd.createChar(3, Stop);
  lcd.home();
```

```
  // Postavljanje komunikacije (set up serial)
  Serial.begin(9600);
  while (!Serial);

  // Pokreni CAN bus na brzini od 500 kpbs
  if (!CAN.begin(500E3)) {
    while (1);
  }
}
void loop() {
//Definira se koji simbol će se prikazati za određenu tipku
  if(digitalRead(TURN_LIGHTS) == 1)
  {
    lcd.home();
    lcd.write(0);
  }
  else
  {
  lcd.clear();
  }
  if(digitalRead(LIGHTS) == 1)
  {
    lcd.home();
    lcd.write(1);
  }
  else
  {
    lcd.clear();
  }
  if(digitalRead(STOP_LIGHTS) == 1)
```

```
  {
    lcd.home();
    lcd.write(2);
  }
  else
  {
  lcd.clear();
  }
  if(digitalRead(SIREN) == 1)
  {
    lcd.home();
    lcd.write(3);
  }
    else
    {
     lcd.clear();
    }

   if(digitalRead(TEMPERATURE) == 1)
   {
    int packetSize = CAN.parsePacket();
  Serial.println(packetSize);
  for (int i = 0; i<packetSize;i++) {
    Serial.println(CAN.read());
  }
   }
//   // humidity=CAN.read();
//    temperature=CAN.read();
//
//       lcd.clear();
```

```
//        lcd.setCursor(0,0);
//        lcd.print(humidity);
//        lcd.setCursor(2,1);
//        lcd.print(temperature);
//    }
//    }
//    else
//    {
//    lcd.clear();
//    }
//Za parking senzor nije se koristio simbol nego će se njegova vrijednost prikazivati na LCD-u. Prvo je trebalo odrediti da CAN pročita paket "distance" sa drugog Arduina i pošalje na Arduino gdje je spojen LCD, te se na LCD-u prikaže udaljenost koju daje senzor s dodatkom teksta "cm DETECTED OBJ".
  if(digitalRead(PARKING) == 1)
  {
   int packetSize = CAN.parsePacket();


  if (packetSize) {
    distance=CAN.read();
       lcd.clear();
       lcd.setCursor(0,0);
       lcd.print(distance);
       lcd.setCursor(2,1);
       lcd.print(" cm  DETECTED OBJ");
  }
  }
  else
  {
    lcd.clear();
  }
```

```
  //Kreirati ćemo CAN pakete do 8 bajtova koji sadrži olitanja na ulaznim pinovima definirani na vrhu
  CAN.beginPacket(CAN_BUTTONS);
  CAN.write(digitalRead(TURN_LIGHTS));
  CAN.write(digitalRead(LIGHTS));
  CAN.write(digitalRead(STOP_LIGHTS));
  CAN.write(digitalRead(SIREN));
  CAN.write(digitalRead(TEMPERATURE));
  CAN.write(digitalRead(PARKING));
  CAN.endPacket();
}
```

Dalje slijedi kod za drugi Arduino gdje je spojena LED traka koja će imitirati žmigavac i prednje svijetlo. Uz LED traku spojen je i piezzo buzzer koji imitira sirenu.

```
#include <CAN.h>      //Uključujemo biblioteku za CAN
#include "FastLED.h"       //Biblioteka za LED svjetla
#define NUMPIXELS 8        //Definirali smo koliko LED dioda ima na WS2812 modulu
#define DATA_PIN 5         //Definirali smo pin na koji je priključena LED traka
# define CAN BUTTONS 0x12       //Adresa sklopki

// Define the array of leds
CRGB leds[NUMPIXELS];      //Definiranje niza LED dioda

int packetID;
const int buzzer = 7;      //Definiranje pina za buzzer

// veličina LIGHTS niza
// Naziv LIGHTS je naziv za svaki senzor biti će svaki senzor koji ćemo koristit
```

```
#define SIZE 6


// definiramo array za smještaj varijabli "lights"
int lights[SIZE];
// Funkcija LightsOn
void LightsOn()
{
 // Uključujemo LED diode za prednja svjetla
 for(int i = 0; i < NUMPIXELS; i++)
 leds[i] = CRGB(255,255,255);
 FastLED.show();
}


// Funkcija LightsOff
void LightsOff()
  {
   // Funkcija za isključivanje LED dioda
 for(int i = 0; i < NUMPIXELS; i++)
 leds[i] = CRGB::Black;
 FastLED.show();
}
//Funkcija za žmigavac
void SignalLightsOn() {
// Uključivanje žmigavca i definiran raspon boja da dobijemo
narančastu boju
 for(int i = 0; i < NUMPIXELS; i++) {
 leds[i] = CRGB(60,255,0);
 FastLED.show();
    delay(25);
}
```

```
  delay(250);
  FastLED.clear();
  FastLED.show();
  delay(500);
}
//isključi žmigavce
void SignalLightsOff() {
 // Now turn the LED off
 for(int i = 0; i < NUMPIXELS; i++)
 leds[i] = CRGB::Black;
 FastLED.show();
}
//Funkcija za uključivanje sirene
void TurnOnSiren()
{
tone(buzzer, 1000); // Pošalji 1KHz zvučnog signala
}

//Funkcija za isključivanje sirene
void TurnOffSiren()
{
  noTone(buzzer);
}

void setup() {
//Inicijalizacija WS2812
  FastLED.addLeds<WS2812, DATA_PIN, RGB>(leds, NUMPIXELS);
 pinMode(buzzer, OUTPUT); // Postavi buzzer kao izlaz
//Inicijalizacija serijske komunikacije
Serial.begin(9600);
```

```
  while (!Serial);



  Serial.println("CAN Receiver");

  // Pokreni CAN na brzini od 500 kbps
  if (!CAN.begin(500E3)) {
    while (1);
  }
}




void loop() {

  // try to parse packet
  int packetSize = CAN.parsePacket();

  if (packetSize) {
    // ocitavamo CAN packet ID
    packetID = CAN.packetId();


    // ako je Packet ID == 0x12 (CAN_BUTTONS)
    if(packetID == CAN_BUTTONS)
    {
      // popuni niz „lights“ sa vrijednostima očitanima na CAN
sabirnici
      for(int i=0; i<SIZE; i++)
      {
        lights[i] = CAN.read();
      }
```

```
 }

 if(lights[0] == 1)
{
   // upali zmigavce
   SignalLightsOn();

}
 else {
   // ugasi zmigavce
   SignalLightsOff();
}
 if(lights[1] == 1)
 {
   // upali svjetla
   LightsOn();
 }
   else
 {
   // ugasi svjetla
   LightsOff();
 }
 if(lights[2] == 1)
 {
   //upali sirenu
   TurnOnSiren();
 }
 else
 {
  TurnOffSiren();
```

```
    }

  }

}
```

Slijedi kod za treći Arduino na kojeg su spojeni senzori DHT22, HC-SR04 i LED traka.

```
#include <CAN.h>
#include "FastLED.h"
#include "DHT.h"
DHT dht;
int packetID;

const int trigPin = 4;
const int echoPin = 6;
// defines variables
long duration;
int distance = 0;

#define NUMPIXELS 8

#define DATA_PIN 3
#define CLOCK_PIN 13

// Definiranje niza LED dioda
CRGB leds[NUMPIXELS];
# define CAN_BUTTONS 0x12

// velicina LIGHTS arraya
#define SIZE 6

// definiramo array za smjestaj varijabli "lights"
int lights[SIZE];

void setup() {

Serial.begin(9600);
  while (!Serial);
    FastLED.addLeds<WS2812, DATA_PIN, RGB>(leds, NUMPIXELS);
    pinMode(trigPin, OUTPUT); // Postavi trigPin kao izlaz
    pinMode(echoPin, INPUT); // Postavi echoPin kao ulaz
    dht.setup(5); // data pin

  Serial.println("CAN Receiver");

  // start the CAN bus at 500 kbps
  if (!CAN.begin(500E3)) {
    Serial.println("Starting CAN failed!");
    while (1);
  }
}
```

```
void loop() {

  // try to parse packet
  int packetSize = CAN.parsePacket();

  if (packetSize) {

    // ocitavamo CAN packet ID
    packetID = CAN.packetId();

    // ako je Packet ID == 0x12 (CAN_BUTTONS)
    if(packetID == CAN_BUTTONS)
    {
      // popuni array lights sa vrijednostima
      for(int i=0; i<SIZE; i++)
      {
        lights[i] = CAN.read();
      }
    }

    if(lights[3] == 1)
    {
      //upali stop svjetla
      StopLightsOn();

    }
    else
    {
      //ugasi stop svjetla
      StopLightsOff();

    }

   if(lights[4] == 1)
    {
      //ukljuci senzor temperature
      TurnOnTemp();
    }
    else
    {
      //iskljuci temp
      TurnOffTemp();
      }

    if(lights[5] == 1)
    {
      //ukljuci senzor parkinga
      TurnOnParking();
    }
    else
    {
      //iskljuci parking
      TurnOffParking();
    }
```

```
  }
}

void StopLightsOn()
{
 // Uključi LED
 for(int i = 0; i < NUMPIXELS; i++)
 leds[i] = CRGB(0,255,0);
 FastLED.show();
}

void StopLightsOff()
{
   // Isključi LED
 for(int i = 0; i < NUMPIXELS; i++)
 leds[i] = CRGB::Black;
 FastLED.show();
}

void TurnOnParking()
{
  delayMicroseconds(10);
// Postavi trigPin na low
digitalWrite(trigPin, LOW);

delayMicroseconds(2);
// Postavi trigPin na HIGH vrijednost 10 mikrosekundi
digitalWrite(trigPin, HIGH);

delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Pročitaj echoPin, vrati zvučni val u milisekundama

duration = pulseIn(echoPin, HIGH);

// Izračun za udaljenost
distance= duration*0.034/2;

delay(50);

CAN.beginPacket(CAN_BUTTONS);
  CAN.write(distance);
  CAN.endPacket();
}

void TurnOffParking()
{
  // Clears the trigPin
digitalWrite(trigPin, LOW);
}


// funkcija za slanje temperature sa DHT22 na CANBUS sabirnicu
```

```
void TurnOnTemp()
{
delay(dht.getMinimumSamplingPeriod());

  float humidity = dht.getHumidity();
  float temperature = dht.getTemperature();
  delay(50);

  CAN.beginPacket(CAN_BUTTONS);
  CAN.write(humidity);
  CAN.write(temperature);
  CAN.endPacket();
  }
void TurnOffTemp()
{
}
```

5. ZAKLJUČAK

Razne automobilske komunikacijske mreže su kroz godine razvijene u pokušaju da zamjene ili konkuriraju CAN-u. Međutim CAN je ostao industrijski standard prvenstveno iz razloga što je prva komunikacijska mreža razvijena za automobile. Njegova pouzdanost i efikasnost su i dalje pogodni za izazove modernog tržišta. Praćenje temperature guma, senzori protoka goriva, senzori visine vozila i senzori položaja udara samo su neki od mnogih CAN pametnih senzora koje bismo mogli dodati u našu postavku. Ako je znanje moć, CAN omogućuje svim prijemnicima mogućnost prikupljanja više podataka nego ikad prije. Sve dok je prijemnik dovoljno pametan da razumije podatke, sve više automobila trebalo bi raditi ono što se nekada smatralo nemogućim.

CAN je robustan serijski komunikacijski sabirnički sustav koji se uglavnom nalazi u automobilskim i industrijskim okruženjima. CAN koristi diferencijalni signal, što ga čini otpornijim na buku, zajedno s arbitražnom shemom za ne razorni prijenos poruka. Uz sve to, odličan je i za ugrađene aplikacije koje se nalaze u opasnim okruženjima ili područjima s puno elektromagnetskih smetnji.

Izradom prototipa za Buggy vozilo pokušali smo sebi približiti koliko je CAN bus jednostavan i učinkovit, te da korištenjem njega kao protokola uvelike smanjujemo upotrebu žica kroz automobil.

LITERATURA

[1] History of CAN technology,

https://www.can-cia.org/can-knowledge/can/can-history/, [pristupljeno 6.4. 2021]

[2] Silvano Jenčić (2015), Industrijske računalne mreže

[3] CAN Bus Explained - A Simple Intro (2021), https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en, [pristupljeno 7.4.2021]

[4] Introduction to CAN BUS and How to use it with Arduino, https://www.seeedstudio.com/blog/2019/11/27/introduction-to-can-bus-and-how-to-use-it-with-arduino/, [pristupljeno 7.6.2021]

[5] The OSI Model - Features, Principles and Layers, https://www.studytonight.com/computer-networks/complete-osi-model, [pristupljeno 8.4.2021]

[6] Controller Area Network Physical Layer Requirements, https://www.ti.com/lit/an/slla270/slla270.pdf?ts=1619435404297&ref_url=https%253A%252F%252Fwww.google.com%252F, [pristupljeno 10.4.2021]

[7] CAN Bus And SAE J1939 Bus Voltage, https://copperhilltech.com/blog/can-bus-and-sae-j1939-bus-voltage/, [5.5.2021]

[8] Physical layer, https://inp.nsk.su/~kozak/canbus/canphy.pdf, [pristupljeno 8.5.2021]

[9] Data Link Layer https://webcache.googleusercontent.com/search?q=cache:74P8-rzoMpoJ:https://weble.upc.edu/asig/ea/practica4/CANdll.pdf+&cd=15&hl=hr&ct=clnk&gl=hr&client=firefox-b-d [pristupljeno 11.5.2021]

[10] Controller Area Network (CAN), https://www.eecs.umich.edu/courses/eecs461/doc/CAN_notes.pdf, [pristupljeno 12.5.2021]

[11] Data Exchange On The CAN Bus, http://www.volkspage.net/technik/ssp/ssp/SSP_238.pdf, [pristupljeno 12.5.2021]

[12] What is Arduino?, https://www.arduino.cc/en/Guide/Introduction, [pristupljeno 20.5.2021]

[13] Introduction to Arudino Nano, https://www.theengineeringprojects.com/2018/06/introduction-to-arduino-nano.html, [pristupljeno 21.5.2021]

[14] Arduino Nano, https://www.arduino.cc/en/uploads/Main/ArduinoNanoManual23.pdf, [pristupljeno 22.5.2021]

[15] MCP2515 CAN Bus Interface Module, https://protosupplies.com/product/mcp2515-can-bus-interface-module/, [pristupljeno 25.5.2021]

[16] WS2812 Datasheet (PDF),

https://pdf1.alldatasheet.com/datasheet-pdf/view/553088/ETC2/WS2812.html, [pristupljeno 2.6.2021]

[17] KKM: Ultrazvučni modul HC-SR04, https://e-radionica.com/hr/blog/2015/08/19/kkm-ultrazvucni-modul-hc-sr04/, [pristupljeno 16.6.2021]

[18] Ultrasonic Ranging Module HC-SR04, https://www.electroschematics.com/wp-content/uploads/2013/07/HCSR04-datasheet-version-1.pdf, [pristupljeno 18.6.2021]

[19] Piezzo Buzzer Datasheet, https://www.kaili-buzzer.com/Buzzer-Datasheet/Piezo-Buzzer/HP1470X.pdf, [pristupljeno 18.6.2021]

[20] DHT22 - Temperature and Humidity Senzor, https://components101.com/sensors/dht22-pinout-specs-datasheet, [pristupljeno 19.6.2021]

POPIS SLIKA

POPIS TABLICA