KONSTRUKCIJA POGONSKO UPRAVLJAČKOG SKLOPA ELEKTRIČNOG ROMOBILA I EKSPERIMENTALNA PROVJERA FUNKCIONALNOSTI

Pošta, Dejan

VELEUČILIŠTE U KARLOVCU

STROJARSKI ODJEL

DIPLOMSKI STUDIJ STROJARSTVA

DEJAN POŠTA

KONSTRUKCIJA POGONSKO UPRAVLJAČKOG SKLOPA ELEKTRIČNOG ROMOBILA I EKSPEREMENTALNA PROVJERA FUNKCIONALNOSTI

ZAVRŠNI RAD

KARLOVAC, 2021.

VELEUČILIŠTE U KARLOVCU

Diplomski studij strojarstva

KONSTRUKCIJA POGONSKO UPRAVLJAČKOG SKLOPA ELEKTRIČNOG ROMOBILA I EKSPEREMENTALNA PROVJERA FUNKCIONALNOSTI

DESIGN OF THE POWERTRAIN AND THE CONTROL UNIT OF THE ELECTRIC SCOOTER AND EXPERIMENTAL VERIFICATION OF FUNCTIONALITY

Završni rad

Dejan Pošta

Karlovac, 2021.

VELEUČILIŠTE U KARLOVCU Karlovac University of Applied Sciences	Klasa: 602-11/__-01/____ Ur.broj: 2133-61-04-__-01
ZADATAK ZAVRŠNOG / DIPLOMSKOG RADA	Datum:

Ime i prezime	Dejan Pošta		
OIB / JMBG			
Adresa			
Tel. / Mob./e-mail			
Matični broj studenta			
JMBAG			
Studij (staviti znak **X** ispred odgovarajućeg studija)	preddiplomski	X	specijalistički diplomski
Naziv studija	Strojarske konstrukcije		
Godina upisa	2019		
Datum podnošenja molbe	14.07.2021		
Vlastoručni potpis studenta/studentice			

.

Naslov teme na hrvatskom: Konstrukcija pogonsko upravljačkog sklopa električnog romobila i eksperementalna provjera funkcionalnosti
Naslov teme na engleskom: Design of the powertrain and the control unit of the electric scooter and experimental verification of functionality
Opis zadatka: U prvom dijelu rada potrebno je opisati princip pogonskih sustava električnih romobila. Opisati tipove pogonskih sustava i baterijskog sklopa električnih romobila. Opisati metode upravljanja i punjenja električnih romobila. U drugom, praktičnom dijelu rada, treba konstruirati proračunati konkretan pogonski sustav električnog romobil. Odabrati motor i baterije iz kataloga proizvođača. Kodirati korisničko sučelje u java skripti. Izraditi romobil korištenjem crteža iz završnog rada Konstrukcija električnog romobila i provjera čvrstoće metodom konačnih elemenata; Josipa Obrovac. Eksperimentalno provjeriti funkcionalnost izrađenog električnog romobila, izraditi zapise o istome. Obrazložiti dobivene rezultate, predložiti poboljšanja.

Mentor:	Predsjednik Ispitnog povjerenstva:

IZJAVA:

Izjavljujem da sam ja – student Dejan Pošta, matični broj: 0123419013, upisan na stručni studij strojarstva akademske godine 2020./2021., radio ovaj rad samostalno, koristeći se znanjem stečenim tijekom obrazovanja, te uz stručnu pomoć i vođenje mentora Tihomira Mihalića, Doc. Dr. sc. i Filipa Žugčića, mag. ing. el., kojima se zahvaljujem na pruženoj pomoći.

U Karlovcu, 2021.

Dejan Pošta

_____

SAŽETAK

U ovom završnom radu će biti opisani pogoni električnog romobila. Napravit će se proračun upravljačke elektronike i baterije. Odabrat će se komponente po tom proračunu i razviti elektronika upravljanja koja će se programirati u C++ jeziku. Napravit će se web stranica električnog romobila za upravljanje.

SUMMARY

In this diploma thesis different electric drives will be described. Battery and electronic systems will be dimensioned. Components will be chosen according to the design which will be programmed in C++ language. Web page for electronic scooter control will be made.

KLJUČNE RIJEČI

Električni romobil, mikrotransport, upravljanje, eksperimentalna provjera funkcionalnosti

KEY WORDS

Electric scooter, microtransport, control, experimental verification of functionality

Sadržaj

Popis slika

Popis tablica

Popis oznaka

Napomena: Neke oznake imaju isti opis, a različite vrijednosti. One sadrže znak ljestvica („#“) koji zamjenjuje broj vrijednosti, npr. R_8 jest otpor koji je u ovoj tablici opisan kao $R_{\#}$.

Oznaka, mjerna jedinica	Opis
F_P, N	Pogonska sila.
F_V, N	Sila otpora zraka.
G, N	Težina romobila i korisnika.
F_N, N	Natražna sila koja vuče romobil niz padinu.
α, °	Nagib
v_V, km/h	Brzina romobila
C_d	Koeficijent trenja romobila i korisnika (aerodinamika)
A, m^2	Poprečni presjek romobila i korisnika
ρ, kg m^{-2}	Gustoća zraka
m, kg	Masa romobila i korisnika
P, W	Snaga motora
C, Wh	Kapacitet baterije
$R_{\#}$, Ω	Otpor
U_{RX}, V	Napon priključka 2 na Arduino Nano
U_{RXA}, V	Ulazni napon dolazne komunikacije invertera
$I_{CQ\#}$, mA	Struja kolektora.
$I_{CQ\#,MAX}$, mA	Maksimalna struja kolektora
V_{CC}, , V	Napon zajedničkog kolektora (5 V)
$V_{CEQ\#,SAT}$, V	Napon zasićenja između kolektora i emitera
$I_{BQ\#}$, mA	Struja baze
h_{fe}	Faktor pojačanja struje za hibridni model tranzistora (β)
$V_{BEQ\#}$, V	Napon između baze i emitera
U_{TXC}, V	Napon odlazne komunikacije prema ESP8266
$C_{\#}$, µF	Kapacitet
$h_{fe,Q\#}$	Hibridni parametar pojačanja struje
$P_{Q\#}$, W	Snaga na tranzistoru

V

T_j, °C	Temperatura poluvodičkog spoja
T_a, °C	Temperatura okoline
$R_{\theta ja}$, °CW^{-1}	Termički otpor
$I_{EQ\#}$, mA	Struja emitera
f_{lpf}, Hz	Frekvencija koljena niskopropusnog filtera
π	Omjer opsega i promjera kružnice (3,1415927…)

1.0 Uvod

Slika 1 - Električni romobil [1]

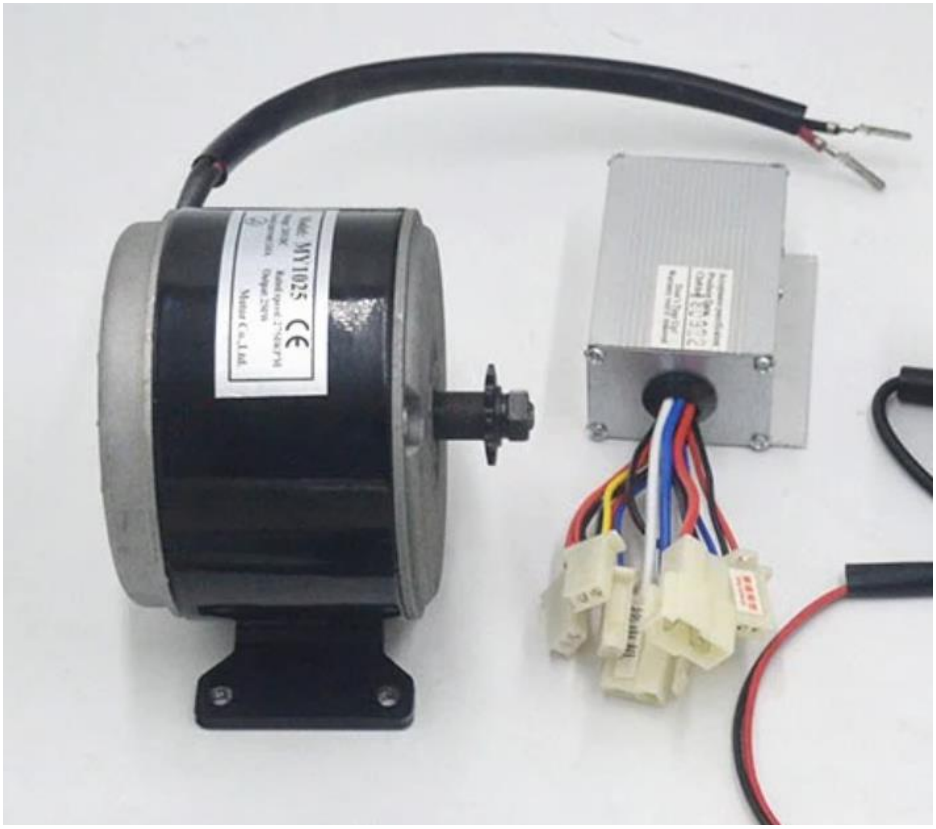
Električni romobil je sve češći oblik mikrotransporta koji se odlikuje malim dimenzijama, niskom cijenom, pouzdanošću i niskom potrošnjom energije. Popularnost električnog romobila je počela rasti adventom jeftinih baterija i sustava električnih pogona. Električni romobil ne treba imati veliku autonomiju jer ih većina korisnika koristi za putovanje do gradskog prijevoza (vlak, tramvaj, bus…). Električni romobil se puni 2 do 3 sata, i spretan je za izbjegavanje prometnih gužvi [2].

E-romobili nisu brza vozila, najčešće idu 25 km/h zbog ugrađene blokade brzine ili samog ograničenja pogona. Većina država dozvoljava upotrebu električnih romobila vršne brzine do 25 km/h bez registracije. Neke države ograničavaju snagu motora. Postoje i brži električni romobili ali za njih je većinom potrebna registracija.

Održavanje električnih romobila je minimalno zbog malo pokretnih dijelova. Baterije koje često mogu izdržati najviše 600 ciklusa su često prvi kvar. Kočione obloge treba provjeravati s vremena na vrijeme, a ležajne površine na upravljačkom stupu obično ne treba dirati za radni vijek romobila (slično kao na biciklu).

1.1 Pogoni električnih romobila

1.1.1 Istosmjerni motor



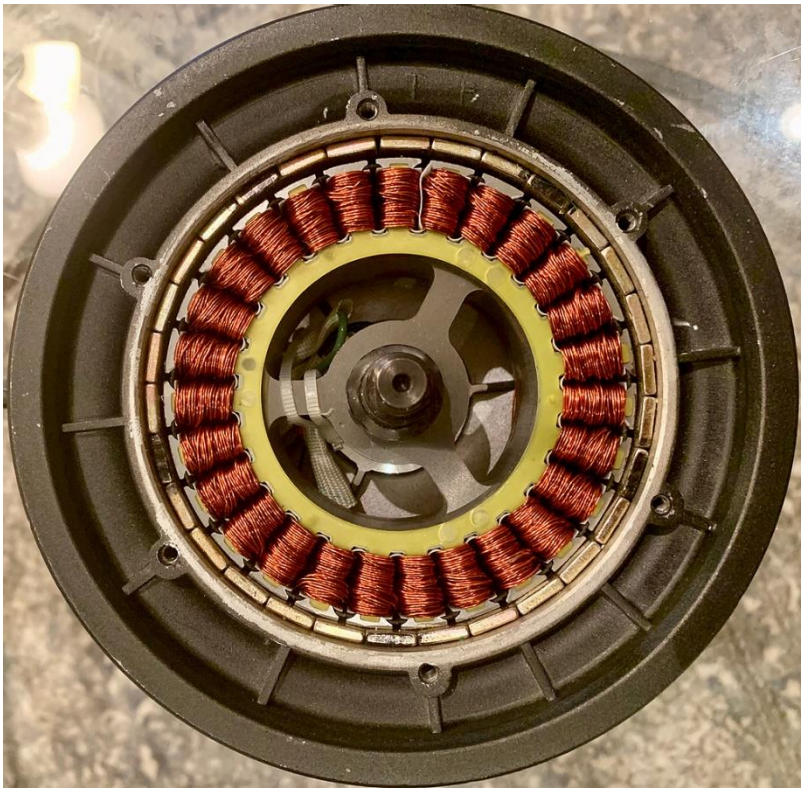
Slika 2 - Pogonski sustav električnog romobila sa klasičnim istosmjernim elektromotorom

Ovaj tip pogona je najjednostavniji i najjeftiniji. Pogonski sustav se sastoji od elektromotora sa četkicama i regulatorom napona. Motor sadrži četkice koje se moraju periodično mijenjati i generiraju električni šum i iskrenje. Uz redovnu zamjenu četkica krajnji kvar je erozija kolektora, nakon čega je motor trajno neupotrebljiv ukoliko se kolektor ne zamijeni. Zamjena kolektora je problematična jer njihova zamjena nije predviđena na tako malim motorima.

Upravljački sklop je jednostavan i sastoji se od više tranzistora snage koji pogone motor PWM (pulse-width modulation; pulsno-širinska modulacija) modulacijom. Širina pulsa je proporcionalna željenoj brzini tj. položaju ručke akceleratora. Upravljački modul tj. regulator napona sadrži kondenzatore koji sa motorom čine filtar koji PWM modulaciju pretvara u konstantan napon i smanjuje povrat smetnji u ostatak elektronike [3].

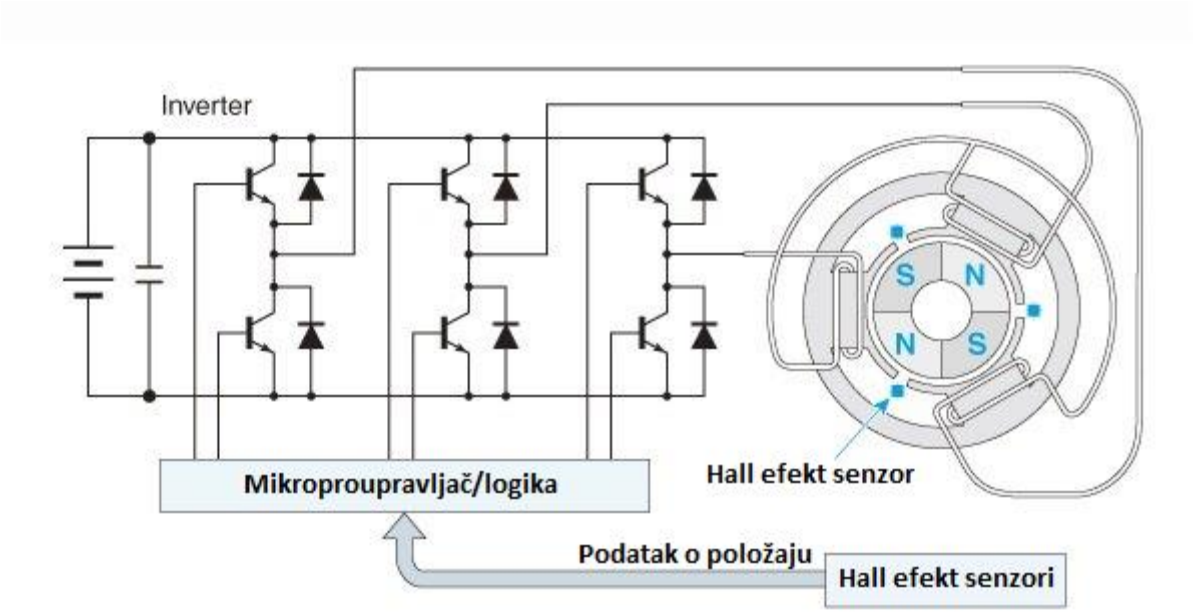
Slika 3 - Dječji električni romobil sa istosmjernim motorom.

1.1.2 Pogon sa bezčetkičnim motor



Slika 4 - Bezčetkični motor ugrađen u glavčinu kotača

Bezčetkični motori su trenutno najčešće korišteni pogon na električnim motorima. Oni su jednostavnije konstrukcije ali zahtijevaju komplicirani pogonski sustav. Ovaj motor je sličniji trofaznim motorima i motorima sa stalnom magnetskom uzbudom (PMSM – permanent magnet syncronous motor; sinkroni motor sa magnetnom uzbudom) nego istosmjernim motorima sa četkicama makar mu tehnički fali samo komutator i četkice. Komutacija se tako vrši elektronički. Upravljački sklop mora znati položaj rotora i po tome energizira određene namote. To daje veću kontrolu nad parametrima rada motora i omogućuje veću učinkovitost nego obični elektromotor sa komutacijom.

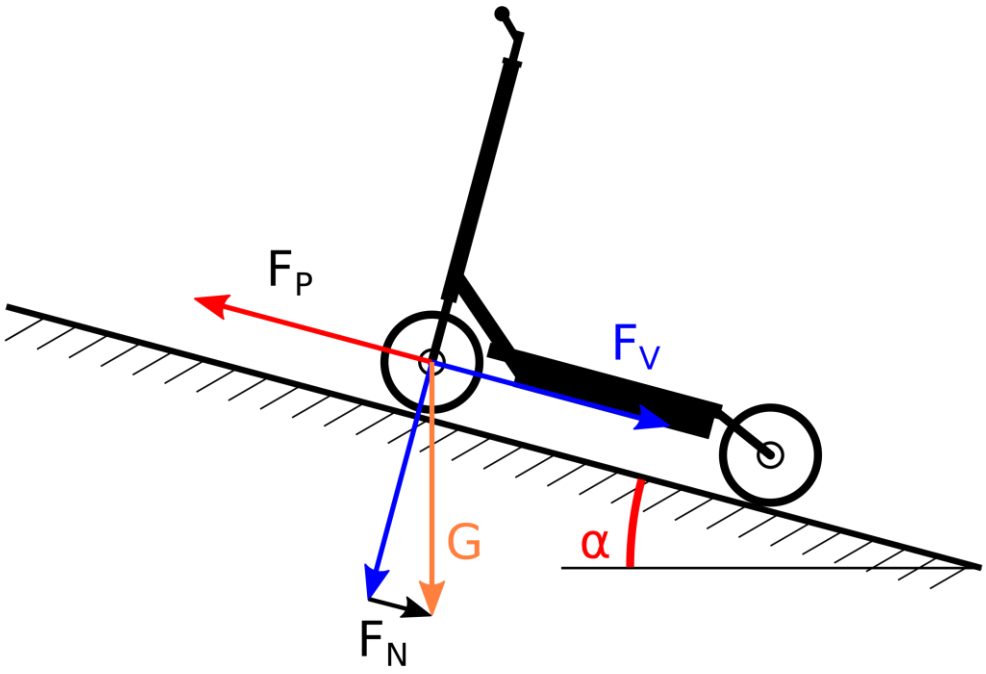


Slika 5 - Shema upravljanja bezčetkičnog motora [4]

Elektronika se sastoji od trofaznog tranzistorskog mosta upravljanog mikroupravljačem. Postoje varijante pogonskog sustava bez senzora, gdje upravljački sustav mjeri inducirani napon na neaktivnim fazama i tako određuje položaj rotora [5].

2.0 Proračun pogona

Da bi se mogle uzeti komponente korektnih kvaliteta, potreban je proračun spomenutih komponenti. Napravljen proračun je aproksimacijske prirode i pruža neke smjernice odabira komponenti [6]. Za točan proračun su potrebni empirički podaci poput načina vožnje korisnika, navika korisnika pa i teže dostupnijih podataka poput izdržljivosti baterija/ćelija.



Slika 6 - Sile koje djeluju na romobil.

Na slici 6 su parametri:

- F_P – Pogonska sila
- F_V – Sila otpora zraka.
- G – Težina romobila i korisnika
- F_N – Natražna sila (sila koja vuče romobil niz padinu)

Za proračun će se uzeti sljedeće pretpostavke:

- α = 3,5 ° = 6% nagib
- v_V = 20 km/h - brzina romobila
- C_d = 1 – koeficijent trenja romobila i korisnika (aerodinamika)
- A = 0,7 m^2 – Presjek romobila i korisnika
- ρ = 1,2 kg/m^3
- m = 100 kg – masa romobila i korisnika

Sila trenja F_V glasi:

$$F_V = \frac{1}{2}\,\rho\, {v_V}^2\, C_d\, A = 12{,}96\ \text{N} \tag{1}$$

Natražna sila F_N glasi:

$$F_N = mg \sin\alpha = 59{,}88\ \text{N} F_V \tag{2}$$

Potrebna snaga motora je zbroj natražne sile i sile trenja:

$$P = v_V(F_V + F_N) = \ 404{,}67\ \text{W} \tag{3}$$

Ako uzmemo da je domet romobila 60 *km*, i tu razdaljinu postigne sa brzinom v_V u spomenutim uvjetima, potreban kapacitet baterije iznosi (ako uzmemo u obzir idealnu bateriju):

$$C = P\frac{d}{v_V} = 1202\ \text{Wh} \tag{4}$$

Izračunat kapacitet baterije je enorman, i on važi za spomenute uvjete. Ako korisnik putuje po ravnici 25 km/h moramo savladati snagu trenja od 140 W. Ako zanemarimo gubitke trenja kotača, tada dobivamo kapacitet baterije od 336 Wh, što je normalniji iznos. Naravno, izabrana baterija će imati veći kapacitet.

3.0 Komponente

3.1 Inverter

Slika 7 – Inverter

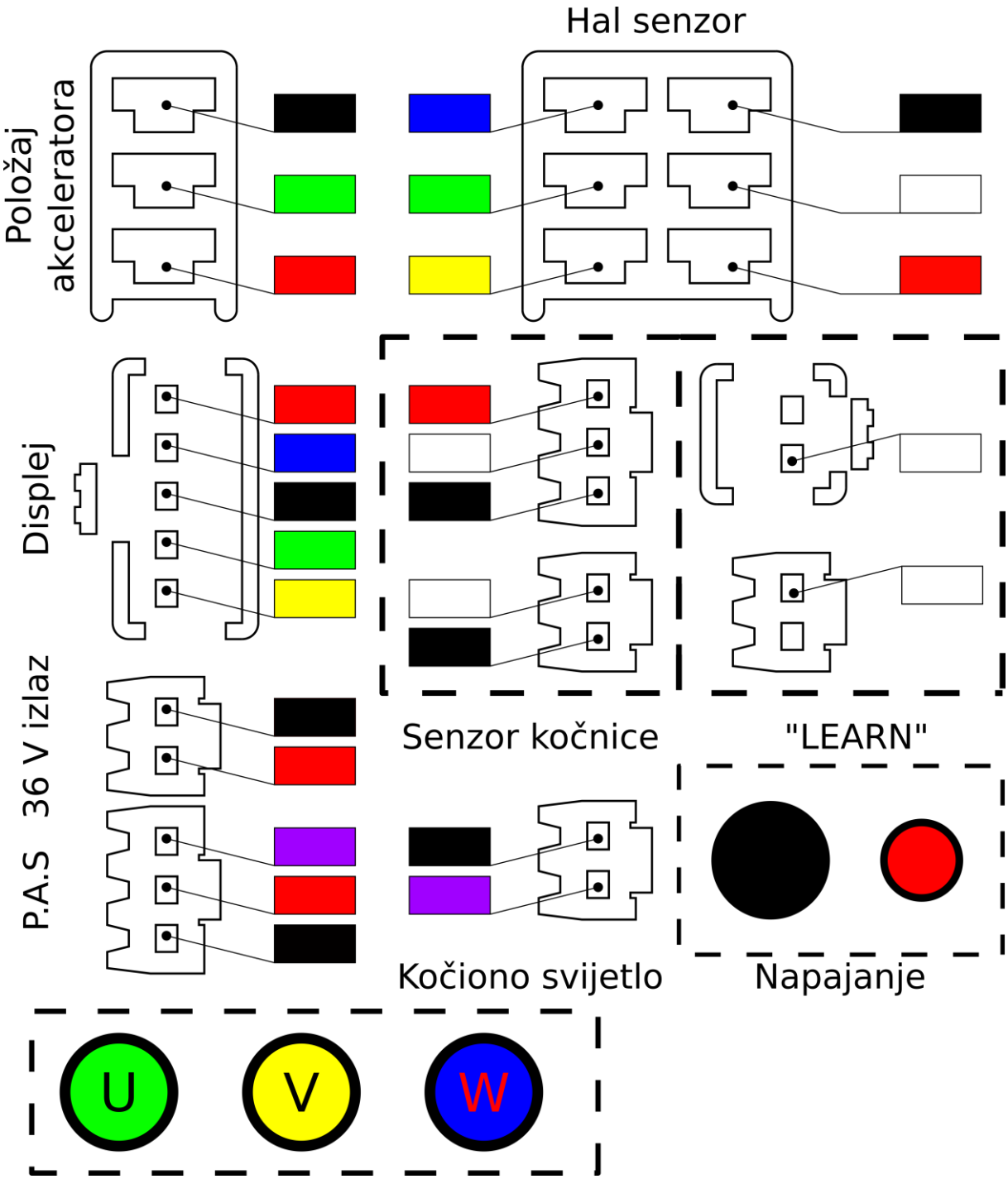
Korišteni motor je bezčetkične konstrukcije sa fiksnom armaturom. Za takav motor je potreban poseban inverter za generiranje rotacijskog magnetnog polja u namotu armature. Korišteni inverter zahtjeva motor sa senzorom

Specifikacije:

- Napon : 36 V
- Vršna struja: 25 A
- Vršna snaga: 500 W
- P.A.S. sustav (Sustav za potporu pedala)
- Senzorski sustav.
- E-kočnica

Bezčetkični motori mogu biti sa senzorima položaja i bez. Bez vanjskog senzora položaja inverter zna položaj rotora motora mjerenje natražnog EMF-a koji nastaje kad rotor (uzbuda) prolazi preko otvorene faze[7]. Pošto se natražno elektromagnetsko polje (EMF) ne može generirati u stacionarnom stanju, inverter mora početi sa arbitrarne faze. To može dovesti do

kratkotrajne rotacije u krivom smjeru, što je ovdje nepoželjno. Ovaj inverter je namijenjen za upotrebu na električnim biciklima, te zbog toga ima P.A.S. sustav.



Slika 8 - Boje vodiča invertera

Konektor	Boja	Opis	Konektor	Boja	Opis
Konektor položaja akceleratora		GND	Konektor za senzore Hallovog efekta		W
		Signal			U
		Vcc			V
Konektor zaslona		Vbatt			GND
		Ignition			???
		GND			Vcc
		RX	Priključak senzora položaja kočnice		Vcc
		TX			Sense
Napajanje		GND			GND
		Vbatt	Nepoznato		???
P.A.S. Senzor		Signal			GND
		Vcc	Priključak za "učenje" upravljačkog sklopa		Learn
		GND			
Konektor štop svijetla		GND	Baterija, kružna stopica		Vbatt
		Štop	Masa, kružna stopica		GND
Faza, kružna stopica		V			
Faza, kružna stopica		W			
Faza, kružna stopica		U			

Tablica 1 - Opis vodiča invertera

Zbog nepostojanja dokumentacije, ožičenje invertera je odgonetnuto eksperimentalno korištenjem multimetra i mjerenjem napona na vodičima. Većina priključaka je bila već označena, što je olakšalo utvrđivanje vodiča.

3.2 Prikaz informacija i informatički sustav

Informacijski sustav služi za informiranje vozača o stanju romobila. Informacijski sustav je najraznolikiji sustav kod svih vozila i uređaja, jer njegova implementacija ovisi o programeru. Prikaz informacija vozaču je važan za dugotrajnost vozila i njegovom efikasnom upotrebom.

Na ovom romobilu prikaz informacija se vrši internetom. Romobil odašilje Wi-Fi pristupnu točku na koju se vozač spaja i vidi informacije na svom mobitelu. Ovo je rijedak način prikaza informacija te bijaše korišten više kao alternativa nego primarni izvor informacija. Klasični električni romobili imaju jednostavan integriran displej na kojem se nalaze najbitnije

informacije i par kontrola. Sustav upravljanja i informacija je tako često integriran u jednu upravljačku ploču.

Slika 9 - Upravljačka ploča romobila Xiaomi M365 Pro

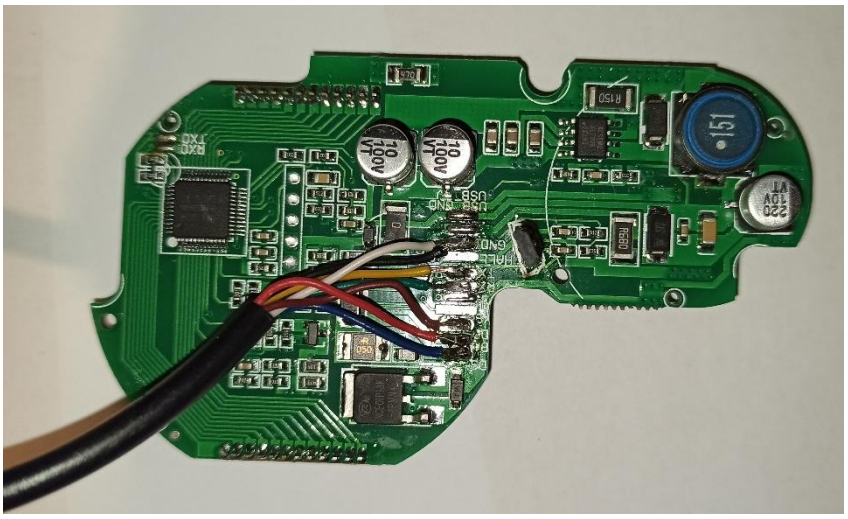
Wi-Fi i Bluetooth se tako koriste kao sekundarni način prikaza informacija o romobilu. To se radi primarno zbog pouzdanosti jer to su napredni i komplicirani protokoli koji zahtijevaju skuplji sustav i duži razvoj.

U ovom slučaju taj pristup je odabran zbog raširenosti upotrebe pametnih telefona. Svaka osoba, a osobito korisnik električnog vozila će imati pametni telefon koji ima jak procesor, velik ekran i ostale resurse koji se ne koriste tako opsežno tokom vožnje. Tako taj inače neiskorišten komad hardvera koji nosi velika većina populacije na sebi se ovdje iskorištava za prikaz informacija. Vozač ili korisnik učvrsti svoj pametni telefon na romobil, uključi Wi-Fi i spoji se na romobil, otvori internet pretraživač i informacije o romobilu su mu dostupne pod prstima. Kod većine pametnih telefona korisnik će dobiti notifikaciju o romobilu jer pristupna točka na romobilu sadrži kaptivni portal (Captive portal). Taj kaptivni portal na sve web upite odgovara sa web stranicom romobila.

3.2.1 LCD modul sa ručkom akceleratora

Slika 10 - LCD modul

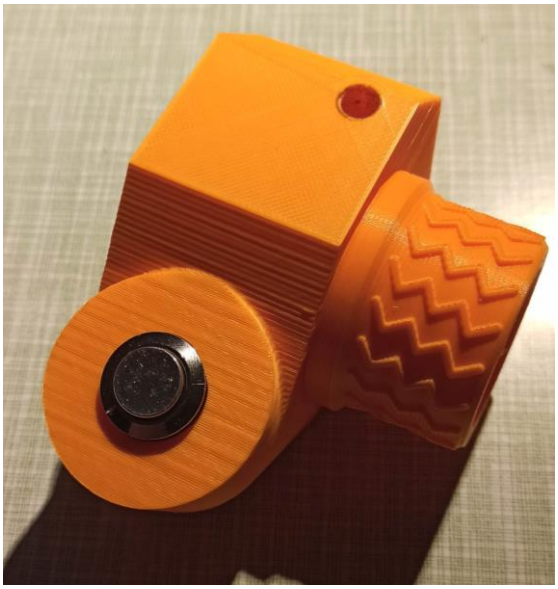
Sa inverterom dolazi poseban modul za upravljanje. Na modulu se nalazi ručka akceleratora, gumb za paljenje i tipka za postavke. Ovaj modul neće biti korišten u gotovom proizvodu nego u razvoju el. romobila. Ovaj modul je važan jer bez njega se ne može otkriti kako se vrši komunikacija sa inverterom.



Slika 11 - Rastavljen LCD modul otkriva jasno označene konekcije

LCD modul sadrži mikroupravljač (koji ovdje nije važan) koji je vidljiv na slici gore u gornjem lijevom kutu. Ispravljač i senzor Hallovog učinka za položaj akceleratora se nalaze u gornje desnom kutu.

3.2.2 Korišteni LCD modul



Slika 12 - Ručka akceleratora izrađena 3D ispisom

Slika 13 - Notifikacija kaptivnog portala koja otvara web stranicu brzinometra

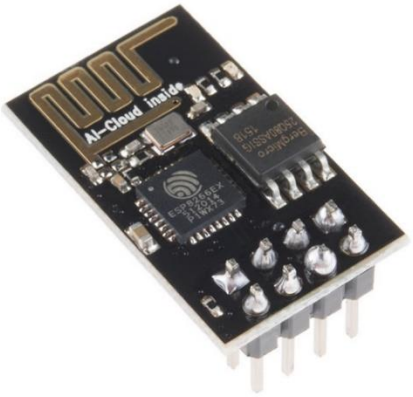
Korišteni LCD modul je minimalističke prirode i sadrži samo gumb za uključivanje električnog romobila. Unutar ručke akceleratora (Slika 12) se nalazi ESP8266 modul zajedno sa elektronikom. Na upravljaču električnog romobila se nalazi držač pametnog telefona. Nakon paljenja, ostatak upravljanja se vrši ručkom akceleratora i web aplikacijom.

3.3 ESP 8266

ESP8266 je Wi-Fi modul sa integriranim L106 32-bitni RISC mikroprocesorom. Originalno jeftin, funkcijski ograničen i bez dokumentacije na engleskom jeziku, postao je veoma popularan mikročip zbog svojih svojstava.

Jedan ESP-01 modul sa ESP8266 mikročipom sadrži [8]:

- Procesor: L106 32-bit RISC mikroprocesorska jezgra bazirana na Tensilica Xtensa Diamond Standard 106Micro i radi na frekvenciji od 80 MHz.
- Memorija
    - 32 KiB RAM za instrukcije
    - 32 KiB RAM međuspremnik za instrukcije
    - 80 KiB RAM za korisničke podatke
    - 16 KiB ETS RAM za podatke sustava.
- Eksterni QSPI flash: podržava do to 16 MiB (tipično 512 KiB do 4 MiB)
- IEEE 802.11 b/g/n Wi-Fi
- Integrirani TR usmjerivač, LNA, pojačalo snage i mreža za uparivanje impedancije
- WEP ili WPA/WPA2 autentifikacija. Podržava otvorene mreže.
- 17 GPIO pinova
- SPI
- I²S sklop sa DMA (dijeli priključak sa GPIO)
- UART na posebnim priključcima, uz poseban UART na GPIO2 bez predajnika
- 10-bit ADC (ADC sa sukcesivnom aproksimacijom)



Slika 14 - ESP-01 modul.

3.4 Pogon

Slika 15 - Korišteni bezčetkični motor

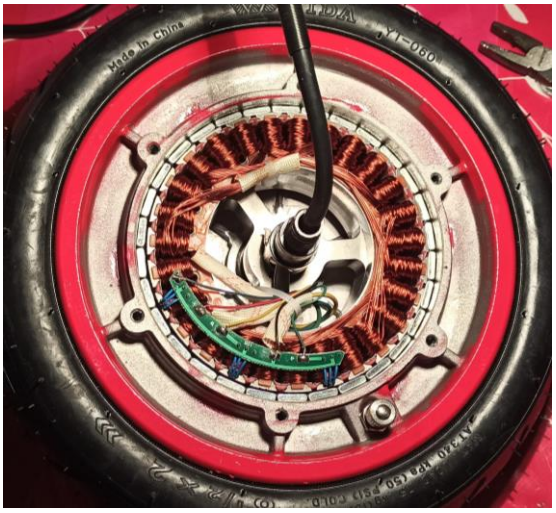
Pogon na ovom električnom romobilu je bezčetkični elektromotor integriran u glavčinu kotača. Motor nema mehaničku redukciju i rotor je ujedno i kotač. Kao i ostali motori, ovaj motor koristi magnete za generaciju polja.

Specifikacije:

- Tip senzora: Hallov učinak
- Nazivni napon: 36 V
- Nazivna snaga: 300 W (kratkotrajno do 600 W)
- Broj polova statora: 27

- Broj polova rotora: 30
- Tlak pneumatika: 200 kPa
- Dimenzije pneumatika: 8 ½ x 2

Korišteni motor je isti kakav koristi Xiaomi M365 Pro. To je popularan električni romobil i dobro je uzeti ako ne standardne barem lako nabavljive komponente. Ovaj elektromotor sadrži 3 senzora Hallovog učinka za očitanje pozicije rotora.



Slika 16- Unutrašnost BLDC motora sa vidljivim magnetima rotora i namotom armature

3.5 Baterija

Slika 17 – Baterija

Korištena baterija se bazira na Li-ion tehnologiji. Baterija je sastavljena u Češkoj i koristi LG ćelije. Baterija je 10s5p, što znači da ima 10 ćelija u serijskom spoju, te 5 u paralelnom. Tako se dobi veći napon (serija) i veći kapacitet (paralela). Ova baterija sadrži 50 ćelija i sadrži BMS (battery management system). Baterija ima integrirani osigurač od 30 A.

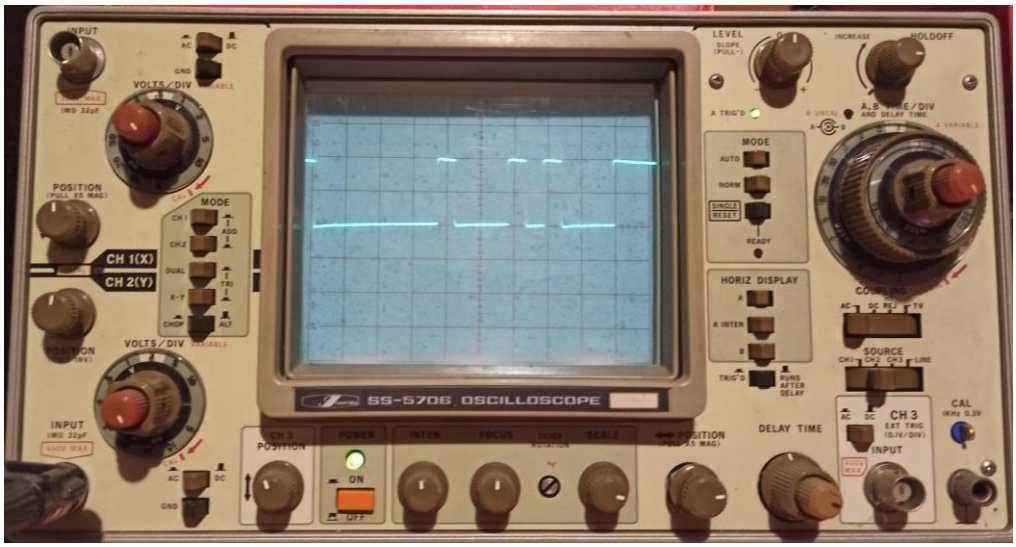
Specifikacije:

- Konfiguracija: 10s5p
- Nazivni napon : 36 V
- Kapacitet: 522 Wh (14,5 Ah)

4.0 Komunikacija između invertera i LCD modula

Da bi se ostvario željeni način prikaza informacija, potrebno je znati način komunikacije između LCD modula i invertera. Trenutni (tvornički) LCD modul ne podržava Wi-Fi protokol, pa ga je potrebno zamijeniti sa posebnim upravljačkim sklopom. Ta preinaka zahtjeva poznavanje komunikacije između ta dva modula.

Tvornički LCD modul je rastavljen (Slika 11 na stranici 11) jasno pokazuje da se koristi serijski prijenos podataka. To otkrivaju natpisi „RX“ za prijam i „TX“ za predaj podataka. Mjerenjem napona tijekom rada je otkriven napon na tim vodičima od 5 V referentno od „GND“ vodiča (masa). Odsutnost priključka za takt, oznake „CLK“, „CK“ ili slično govori da se radi o asinkronoj komunikaciji.



Slika 18 - Snimak jednog od dva komunikacijska signala na osciloskopu

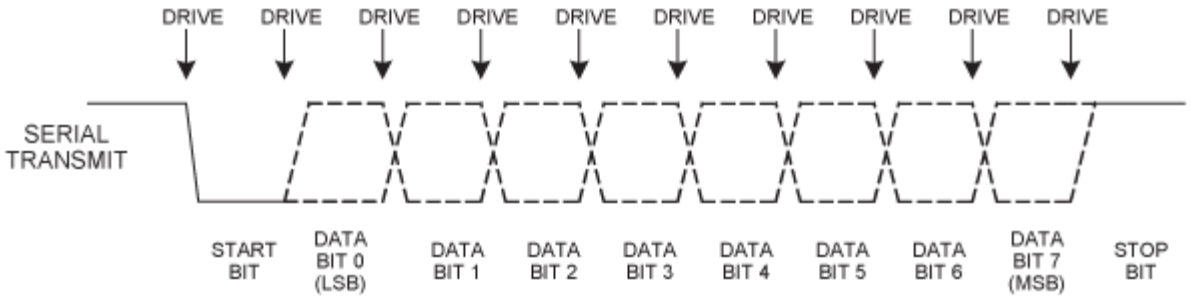
Snimanje vodiča tokom rada je pokazalo da se komunikacija vrši samo na „RX“ i „TX“ zbog vremenski promjenjivog signala na tim vodičima tijekom rada. Na slici 18 je vidljiv snimak početka komunikacije. Trenutno nije važno radi li se o odlaznom ili dolaznom signalu, jer se oni razlikuju samo za poruku koja će se analizirati kasnije. Snimak je na kanalu 2 ovog osciloskopa i snimljen je sa 2 V/div (jedan vertikalni podjeljak vrijedi 2 V) i sa 1 ms/div (jedan horizontalni podjeljak traje 1 milisekundu) sa povećanjem od 5 puta. Najkraća promjena traje samo 100 mikrosekundi (na osciloskopu je prikazano 500 µs zbog povećanja).

$$baud = \frac{1}{trajanje\ najkraće\ promjene} \cdot povečanje \tag{5}$$

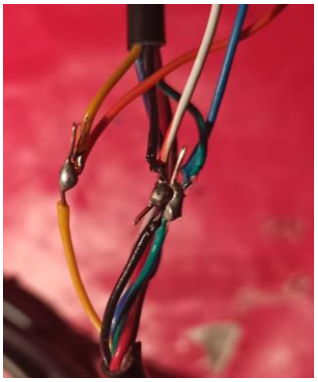
Asinkroni protokol koristi određenu brzinu prijenosa podataka, koja se zove „Baud rate“, sinonim za broj bitova u sekundi. To je jedan od podataka koji opisuje način asinkrone komunikacije. Taj parametar računamo po formuli gore i dobivamo:

$$baud = \frac{1}{0{,}5 \cdot 10^{-3}\ s} \cdot 5 = 10000\ s^{-1} \tag{6}$$

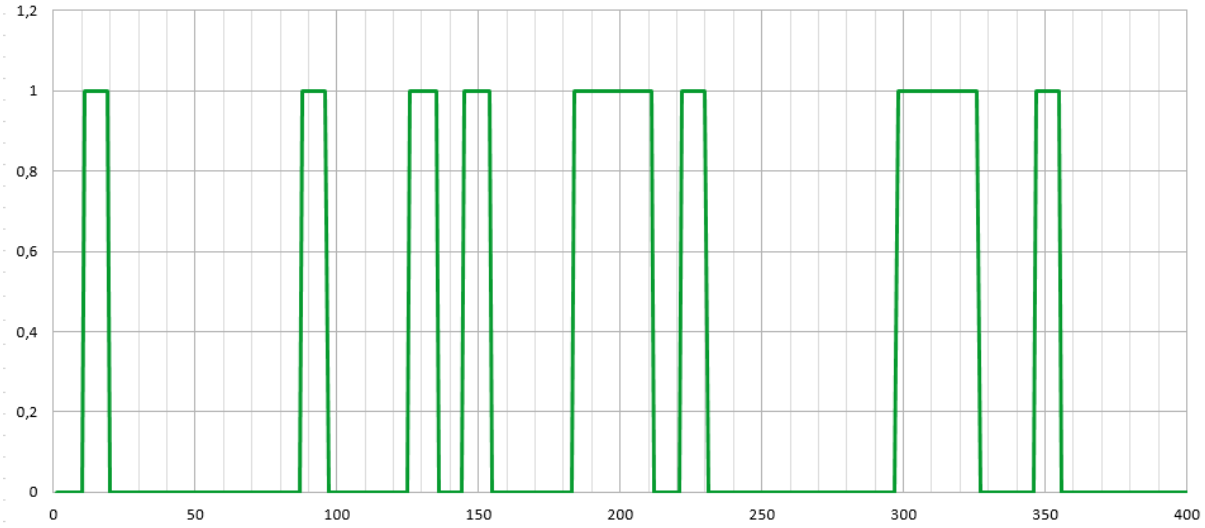
Najbliža standardna brzina je 9600 bitova u sekundi, što je vjerojatna brzina komunikacije. Sljedeći korak je snimanje signala sa logičkim analizatorom. Zbog nedostupnosti takve skupe opreme, signal je sniman koristeći mikroupravljač Atmega328 na Arduino razvojnoj pločici svakih 10 µs na prvi pad napona na komunikacijskoj liniji.



Slika 19 - Format UART komunikacije, prijam i predaj izgleda isto.

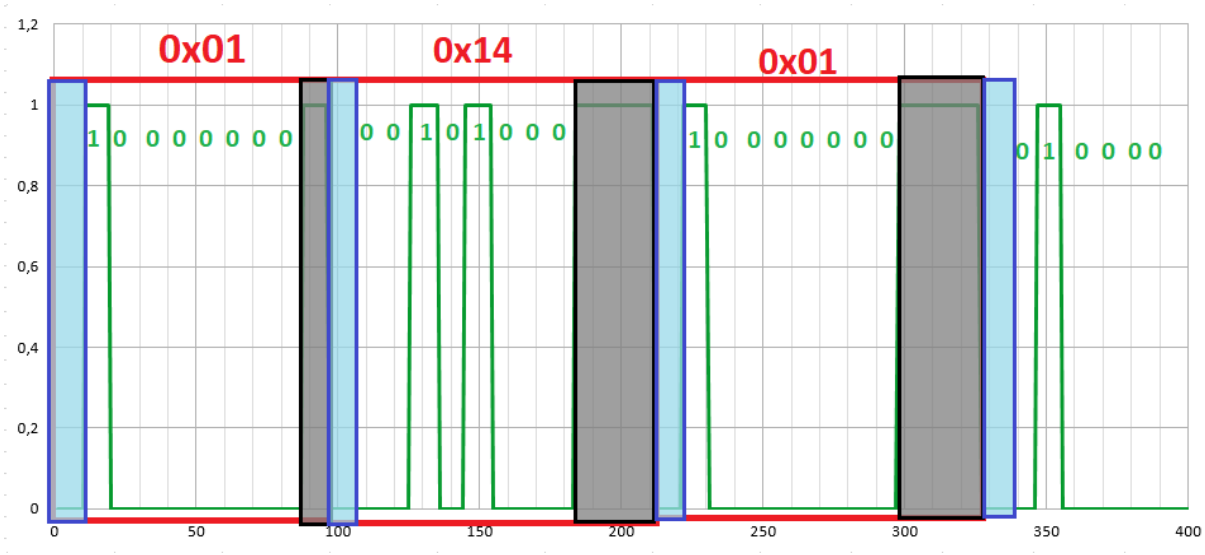


Slika 20 - Presretanje komunikacije tj. postavljanje sonde



Slika 21 – Snimljen signal za analizu. Ovo je signal predajnika.

Snimljen signal (Slika 21) izgleda kao tipičan signal asinkrone serijske komunikacije (UART). Pošto znamo da je duljina najkraćeg impulsa 104,16 µs, možemo izračunati da je naš grubo izrađeni logički analizator bilježio stanje vodiča svakih 11,57 µs. Format komunikacije nije poznat, pa se kreće od najčešćeg formata, 8N1. To znači 8 bitova podataka, N znači bez paritetnog bita i 1 znači koliko je zaustavnih bitova. Broj bitova može biti od 5 do 8, paritet može biti parni, neparni ili bez, a broj zaustavnih bitova je jedan ili dva.



Slika 22 - Protumačen signal.

Slika 22 prikazuje signal protumačen kao 9600 8N1. Oznake su:

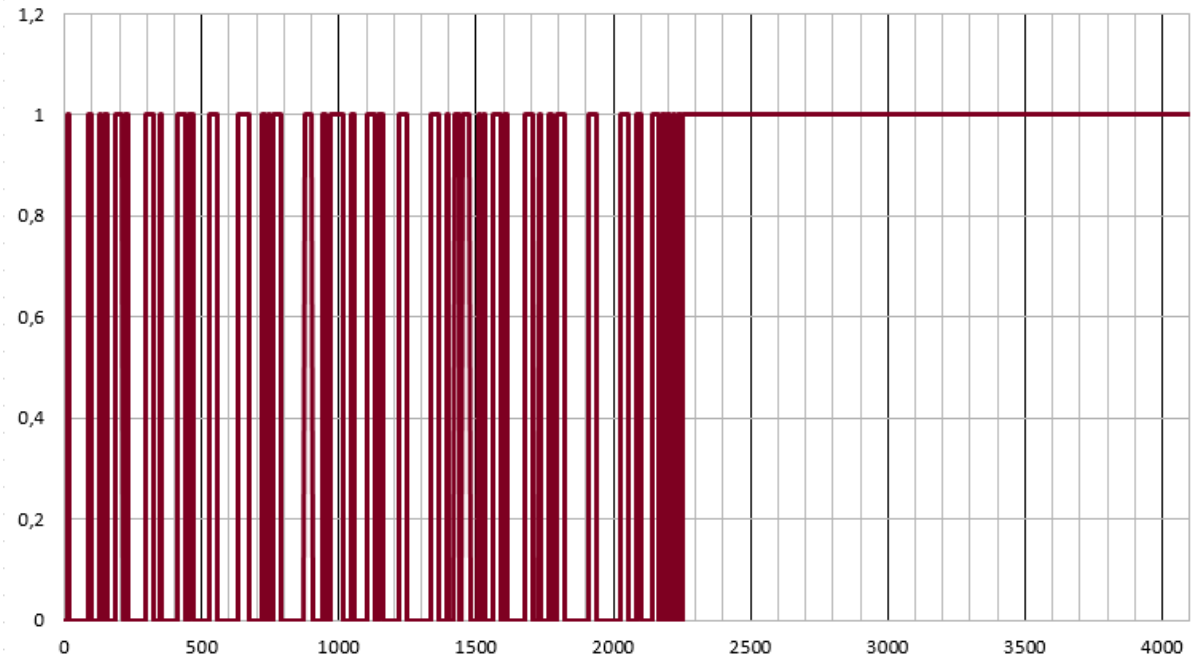
- Plavo – Početni tj. start bit
- Crno – Završni tj. stop bit
- Zeleno – Vrijednost bita i podatkovni bitovi
- Crveno – Okvir, vrijednost je upisana iznad u heksadekadskoj bazi

Početni bitovi, podatkovni bitovi i zaustavni bitovi čine okvir (eng. „Frame“). To čini jednu informaciju. Snimka od 4096 točki (ukupnog trajanja 47,4 ms, komunikacija traje 26 ms ili 2550 mjerenja) ima sličan oblik i ne krši oblik, te se možemo osigurati da se zaista radi o pretpostavljenom formatu.

Trajanje snimke od 26 ms govori da je poslano 25 bajtova. Ako uzmemo u obzir da samo prvi bajt ima jednostruki zaustavni bit, a svi ostali okviri dva, dobivamo 21 podatkovni bit (otprilike). Stvarni broj bajtova je 20, što je utvrđeno brojanjem. Na ovoj snimci izgleda da se koriste dva zaustavna bita (ostalih 20 bajtova ima dvostruki zaustavni bit). To ne treba značiti da je stvarni format 8N2 jer mikroupravljaču treba vrijeme da pripremi podatak za slanje ili čak da počne slati podatke, što uzrokuje uočeni dvostruki zaustavni bit. Zaustavni bit prvog bajta je jednostruki iz nepoznatog razloga, što isto pridonosi teoriji da je korišten 8N1 format.

Bez pravog logičkog analizatora koji često može odmah prepoznati vrstu komunikacije, pretpostavljeni format možemo jedino potvrditi čitanjem podataka sa pravim UART sučeljem

(koja su lako dostupna; svaka Arduino razvojna pločica ima barem jedan). Koristit ću Arduino kao UART sučelje, s kojim ću snimati prenesene podatke i neispravne okvire. Snimati neispravne okvire je veoma važno jer to može pokazati prisutnost paritetnog bita i krivo pretpostavljeni broj bitova podataka. Većina UART sučelja može detektirati neispravne okvire po tome što je zaustavni bit na krivom mjestu, ili što paritetni bit nije konzistentan sa podatkom.



Slika 23 - Snimka cjelokupnog "paketa" poslanog inverteru.

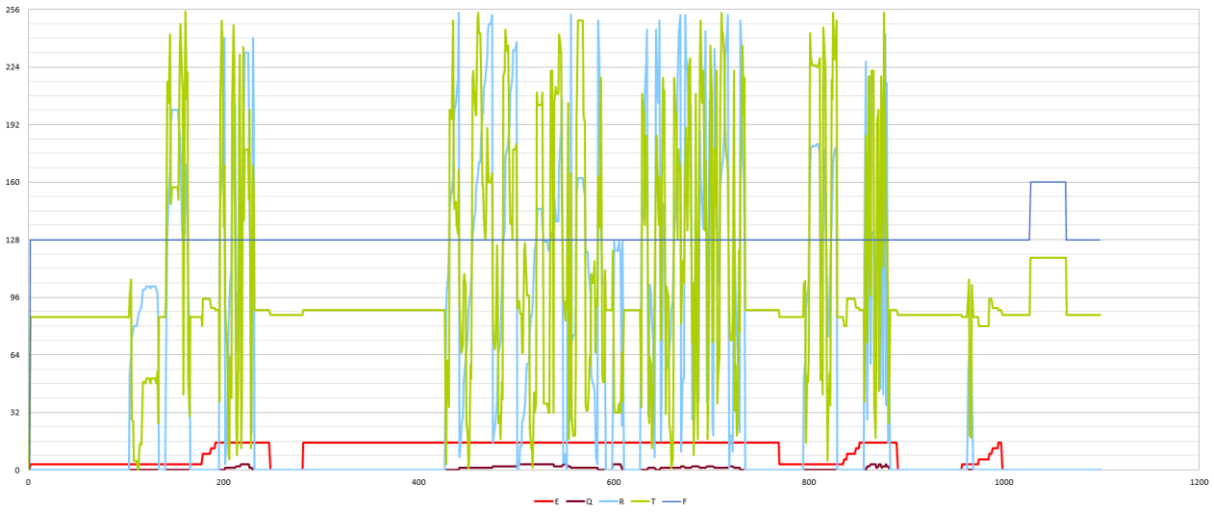
n	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	1	20	1	2	3	128	80	0	200	4	3	0	100	20	1	34	0	0	12	85
2	1	20	1	2	3	128	80	0	200	4	3	0	100	20	1	34	0	0	12	85
3	1	20	1	2	3	128	80	0	200	4	3	0	100	20	1	34	0	0	12	85
4	1	20	1	2	3	128	80	0	200	4	3	0	100	20	1	34	0	0	12	85
5	1	20	1	2	3	128	80	0	200	4	3	0	100	20	1	34	0	0	12	85
6	1	20	1	2	3	128	80	0	200	4	3	0	100	20	1	34	0	0	12	85
7	1	20	1	2	3	128	80	0	200	4	3	0	100	20	1	34	0	0	12	85
8	1	20	1	2	3	128	80	0	200	4	3	0	100	20	1	34	0	0	12	85
9	1	20	1	2	3	128	80	0	200	4	3	0	100	20	1	34	0	0	12	85
10	1	20	1	2	3	128	80	0	200	4	3	0	100	20	1	34	0	0	12	85
11	1	20	1	2	3	128	80	0	200	4	3	0	100	20	1	34	0	0	12	85
12	1	20	1	2	3	128	80	0	200	4	3	0	100	20	1	34	0	0	12	85
13	1	20	1	2	3	128	80	0	200	4	3	0	100	20	1	34	0	0	12	85
14	1	20	1	2	3	128	80	0	200	4	3	0	100	20	1	34	0	0	12	85
15	1	20	1	2	3	128	80	0	200	4	3	0	100	20	1	34	0	0	12	85
16	1	20	1	2	3	128	80	0	200	4	3	0	100	20	1	34	0	0	12	85
17	1	20	1	2	3	128	80	0	200	4	3	0	100	20	1	34	0	0	12	85

Slika 24 - Snimka vrijednosti poslanih bajtova inverteru. Sivi bajtovi su konstantni. Stupac n je redni broj paketa

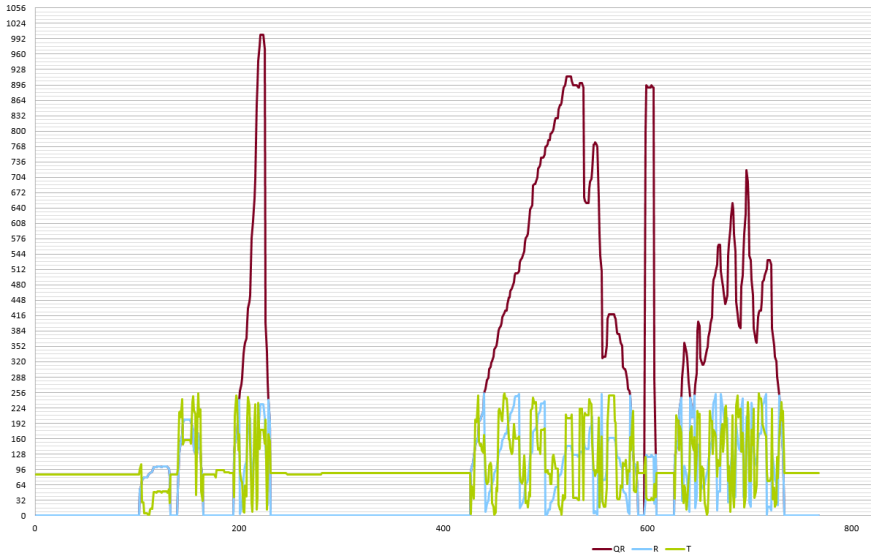
Na slici 24 je prikazana dekadska vrijednost bajtova. Kolumna „n“ označava broj paketa, tj. grupe od 20 bajtova koje šalje modul ka inverteru. Sivo su označeni bajtovi koji se nisu mijenjali tijekom mjerenja. Tijekom snimanja komunikacije je pomicana ručka akceleratora, paljeno svijetlo i pritiskana kočnica. Važno je napomenuti da su prikazani paketi izabrani, tj. paketi koji su bili drugačiji su izbačeni intuicijom mjeritelja.

	Minimalno	Maksimalno	Srednje	Područje	Tip podatka	Funkcija
A	1	1	1,0000	0		
B	20	20	20,0000	0		
C	1	1	1,0000	0		
D	2	2	2,0000	0		
E	0	15	9,1449	15	uint8_t	Stupanj prijenosa kao višekratnik broja 3
F	128	160	129,0793	32	uint8_t	Konfiguracija? 6ti bit je svijetlo
G	80	80	80,0000	0		
H	0	0	0,0000	0		
I	200	200	200,0000	0		
J	4	4	4,0000	0		
K	3	3	3,0000	0		
L	0	0	0,0000	0		
M	100	100	100,0000	0		
N	20	20	20,0000	0		
O	1	1	1,0000	0		
P	34	34	34,0000	0		
Q	0	3	0,4658	3	uint16_t	Akcelerator, 0 do 1000
R	0	254	50,2033	254		
S	12	12	12,0000	0		
T	0	255	103,0356	255	uint8_t	XOR prvih 19 bajtova

Slika 25 - Analiza bajtova zajedno sa odgonetnutom funkcijom i tipom podatka.



Slika 26 - Prikaz promjene bajtova tijekom korištenja.



Slika 27 - Prikaz podatka QR kao 16 bitna vrijednost (little endian format)

Iskustvenim promatranjem snimljenih podataka dolazimo do zaključka da bajtovi u kolumni Q i R predstavljaju željenu brzinu. Odgonetnut je format prijenosa „Little endian“, što znači da se bajtovi prenose od „najmanjeg“ kraja do „najvećeg“. Format je odgonetnut eksperimentalno promatrajući promjenu bajtova (bajt koji predstavlja veću vrijednost se sporije mijenja).

Stupanj prijenosa je parametar koji je teško procijeniti bez tereta. Inverter podržava 4 stupnja prijenosa. Ti stupnjevi prijenosa nemaju mehanički značaj, nego su čisto programerske prirode. Iz priručnika invertera se vidjelo da su to stupnjevi „pomoći“ kod vožnje bicikla (za što je odabrani inverter i namijenjen). U eksperimentalnom djelu će se utvrditi točan učinak tog parametra, koji u testnim uvjetima ograničava maksimalnu brzinu. Komunikacija koristi provjeru integriteta paketa metodom longitudinalne provjere pariteta (eng. XOR checksum; longitudinal parity check) [9]. Svi bajtovi se svedu u zadnji bajt operatorom „isključivo ili“. Provjera se vrši izvršavajući istu radnju na primatelju ali uključujući zadnji bajt. Paket podataka je ispravan ako je rezultat operacije 0.

n	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	2	14	1	0	128	0	0	0	23	112	0	0	255	21
2	2	14	1	0	128	0	0	0	23	112	0	0	255	21
3	2	14	1	0	128	0	0	0	23	112	0	0	255	21
4	2	14	1	0	128	0	0	0	23	112	0	0	255	21
5	2	14	1	0	128	0	0	0	23	112	0	0	255	21
6	2	14	1	0	128	0	0	0	23	112	0	0	255	21
7	2	14	1	0	128	0	0	0	23	112	0	0	255	21
8	2	14	1	0	128	0	0	0	23	112	0	0	255	21
9	2	14	1	0	128	0	0	0	23	112	0	0	255	21
10	2	14	1	0	128	0	0	0	23	112	0	0	255	21
11	2	14	1	0	128	0	0	0	23	112	0	0	255	21
12	2	14	1	0	128	0	0	0	23	112	0	0	255	21
13	2	14	1	0	128	0	0	0	23	112	0	0	255	21
14	2	14	1	0	128	0	0	0	23	112	0	0	255	21
15	2	14	1	0	128	0	0	0	23	112	0	0	255	21

Slika 28 - Prvih 15 paketa invertera prema LCD modulu

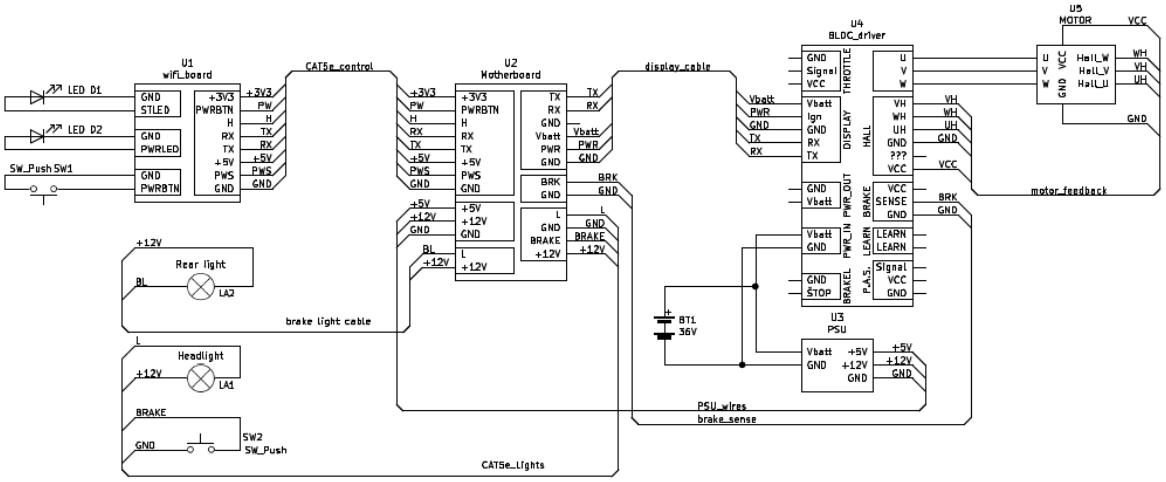
	Minimalno	Maksimalno	Srednje	Područje	Tip podatka	Funkcija
A	2	2	2,0000	0		
B	14	14	14,0000	0		
C	1	1	1,0000	0		
D	0	0	0,0000	0		
E	128	128	128,0000	0	uint8_t	Konfiguracija? Bit 5 je "štop" svijetlo
F	0	0	0,0000	0		
G	0	1	0,0089	1	uint16_t	Nepoznato
H	0	12	1,5751	12		
I	0	101	8,9156	101		Inverzno proporcijalan podatak o brzini
J	2	253	127,2139	251		
K	0	0	0,0000	0		
L	0	0	0,0000	0		
M	255	255	255,0000	0		
N	4	255	102,9674	251	uint8_t	XOR prvih 13 bajtova

Slika 29 - Analiza paketa što šalje inverter ka LCD modulu. Bajt „E“ je odgonetnut nakon snimanja.

Inverter također šalje podatke LCD modulu. Odgonetnuti podaci su bazirani na uočenim promjenama na LCD modulu, tako da nije bilo moguće odgonetnuti sve bajtove. Inverter isto koristi „little endian“ poredak bajtova, te šalje podatke o brzini i statusu. Detaljna analiza je potrebna da se svi bajtovi otkriju. Postojanje dokumentacije bi olakšalo odgonetavanje komunikacije, ali najbitniji podaci su odgonetnuti. Podaci su odgonetnuti na isti način kao i kod LCD modula.

Slika 30 - Vrijednosti bajtova od strane invertera.

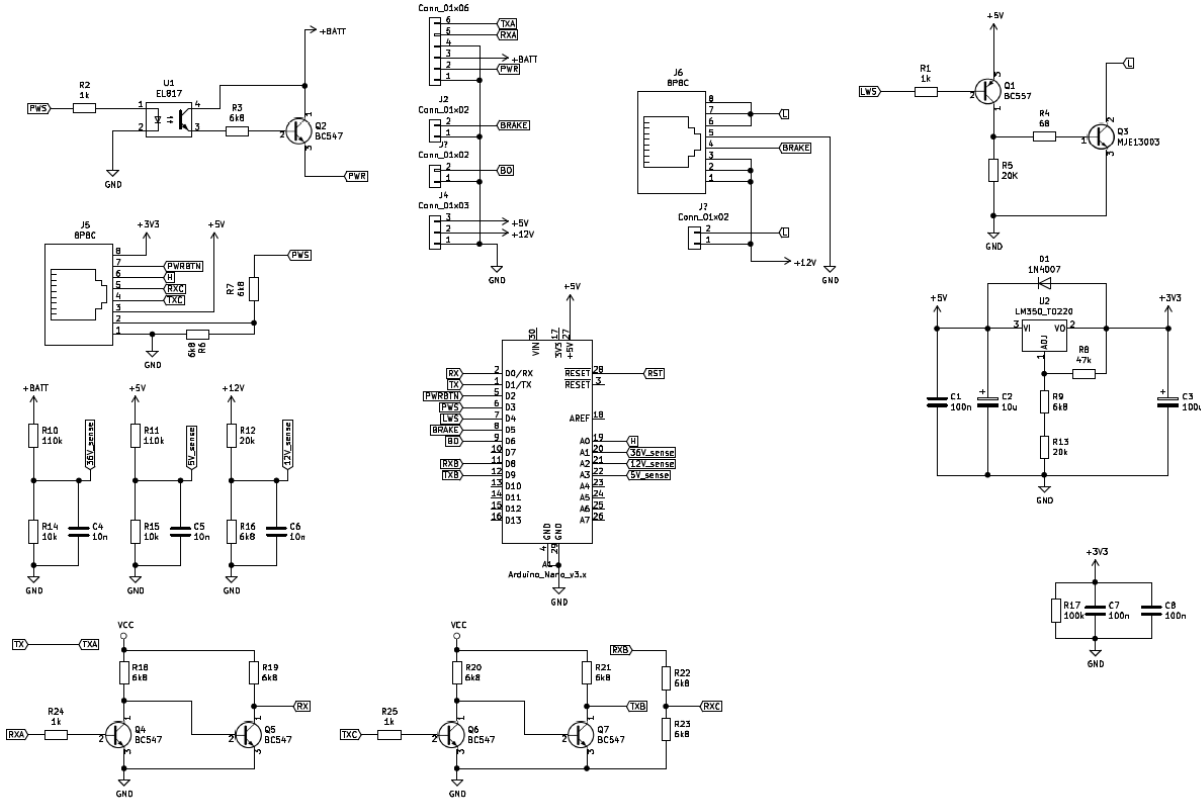
5.0 Elektronika



Slika 31 - Električni sustav romobila

U ovom poglavlju će biti opisan proračun elektronike, posebice matične ploče (motherboard, Slika 31). Na slici 31 je vidljiva električna shema romobila. Poznata je shema matične ploče i Wi-Fi ploče koja će biti opisana u ovom poglavlju. Ostali uređaji nemaju dostupnu shemu, koja nije prijeko potrebna za funkcionalnost.

5.1 Matična ploča



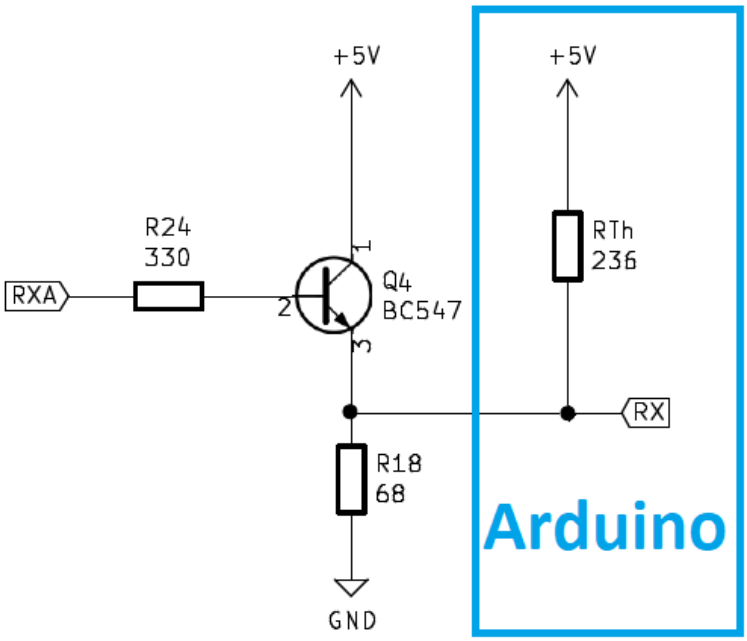
Slika 32 - Shema matične ploče

Matična ploča je centar upravljanja ovog električnog romobila. Na njoj se odvija komunikacija i preko nje se povezuju ostali moduli. Centar matične ploče tvori razvojna pločica Arduino sa Atmega328P mikroupravljačem [10]. Optimalno bi bilo koristiti jedan mikroupravljač za Wi-Fi komunikaciju i upravljanje ali trenutno dostupni ESP8266 podržava samo jedan analogni ulaz, pa je potrebno imati vanjski ADC pretvarač [8]. Taj pretvarač nije bio dostupan pa je odabran mikroupravljač. Uz to, sad se sustav dijeli na niži i viši nivo upravljanja (matična je niži), što povećava pouzdanost.

5.1.1 Ulazna komunikacija

Ulazni sklop sadrži dva digitalna invertera izvedena tranzistorima BC547. Ovdje se za razliku od izlazne komunikacije to ne radi zbog promjene logičke razine, nego zato što inverter ne može povući dovoljno struje (current sink), te tako ne može dovoljno pustiti napon da ga mikroupravljač registrira. Slijedilo napona bi bio bolji odabir ali LED lampica na Arduino pločici vuče previše struje, pa je potrebno imati veliku struju kolektora da se promjeni logička

razina [11]. Dobro je napomenuti da ovo pojačalo radi u zasićenju ili zapiranju (kao sklopka) pa ga zovemo inverterom (moguće je konstruirat i ostala logička vrata sa tranzistorom).



Slika 33 - Ulazno pojačalo u zajedničkom kolektoru sa prikazanim nadomjesnim otporom Arduina.

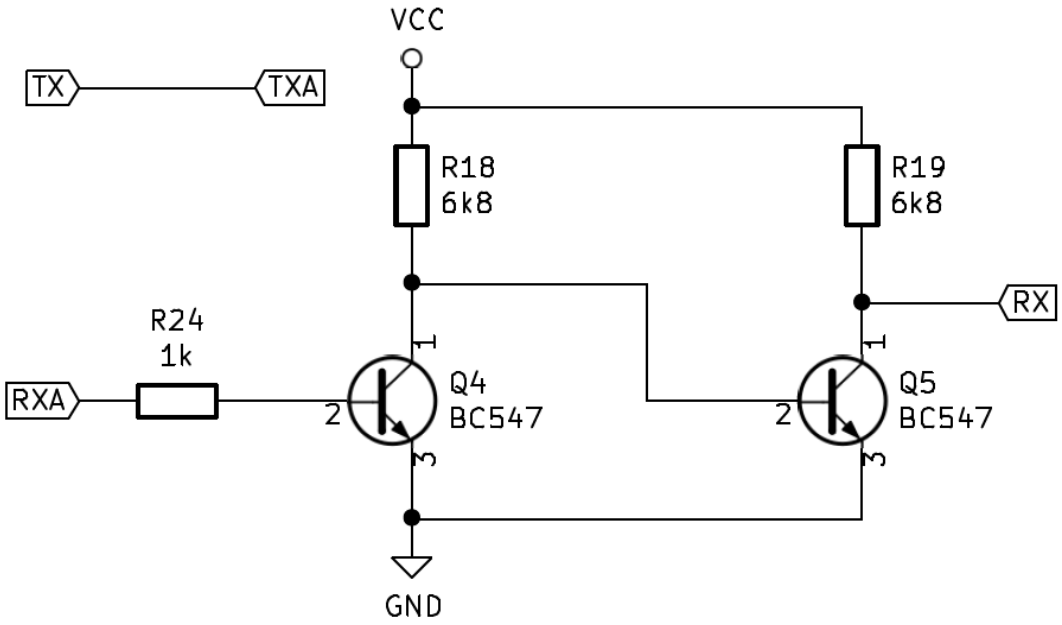
Napon U_{RX} kad je U_{RXA} = 0 V (tranzistor je u zapiranju), iznosi:

$$U_{RX} = 5 \cdot \frac{R_{18}}{R_{th} + R_{18}} = 1{,}1184 \text{ V} \tag{7}$$

Što je dovoljno nizak napon za logičku razinu „0“. Problem je struja kolektora u zasićenju:

$$I_{CQ4} = \frac{+5V - V_{CEQ4,SAT}}{R_{18}} = 68 \text{ mA} \tag{8}$$

Što je ispod maksimalne dopuštene struje kolektora za BC547 [12], ali moguće je smanjiti potrošnju energije sa 2 tranzistora (Slika 34). Osim toga, 2,5 V je preblizu napona U_{RX} da ova sklopka radi pouzdano u okruženju punom šuma (električni romobil; automotivna primjena).



Slika 34 - Obrada ulaznog signala sa dva tranzistora.

Q4 će raditi u zasićenju kad je U_{RXA} u višoj logičkoj razini tj. 5 V. Možemo izračunati struju kolektora u tom slučaju:

$$I_{CQ4,MAX} = \frac{V_{CC} - V_{CEQ4,SAT}}{R_{18}} = \frac{5\ \text{V} - \ 0{,}5\ \text{V}}{6{,}8\ \text{k}\Omega} = 0{,}6618\ \text{mA} \tag{9}$$

Kirhofov zakon o naponu na ulazu glasi:

$$U_{RXA} - I_{BQ4}R_{24} - V_{BEQ4} = 0 \tag{10}$$

Parametar h_{fe} za BC547 u tom režimu je minimalno 100 [12]:

$$I_{CQ4} = h_{fe}\ I_{BQ4} \tag{11}$$

Iz čega možemo dobiti maksimalni iznos otpornika R_{24}:

$$R_{24} = \frac{U_{RXA} - V_{BEQ4}}{I_{BQ4}} = \frac{U_{RXA} - V_{BEQ4}}{I_{CQ4}\ 10^{-2}} = \frac{5\ \text{V} - 0{,}7\ \text{V}}{0{,}0066\ \text{mA}} = 649{,}7431\ \text{k}\Omega \tag{12}$$

Zbog bolje pouzdanosti (rejekcije smetnje) se uzima R_{24} = 1 kΩ. Struja na bazi u tom slučaju iznosi:

$$I_{BQ4} = \frac{U_{RXA} - V_{BEQ4}}{R_{24}} = \frac{5\ \text{V} - 0.7\ \text{V}}{1\ \text{k}\Omega} = 4{,}3\ \text{mA} \tag{13}$$

Što je dovoljno da tranzistor bude u zasićenju. Struja baze je premala da bi oštetila tranzistor.

Tranzistor Q_5 je u zasićenju kad je I_{CQ4} = 0 mA, tj. kad je Q4 u zapiranju. Struja baze na tom tranzistoru je (Kirhofov zakon za napon):

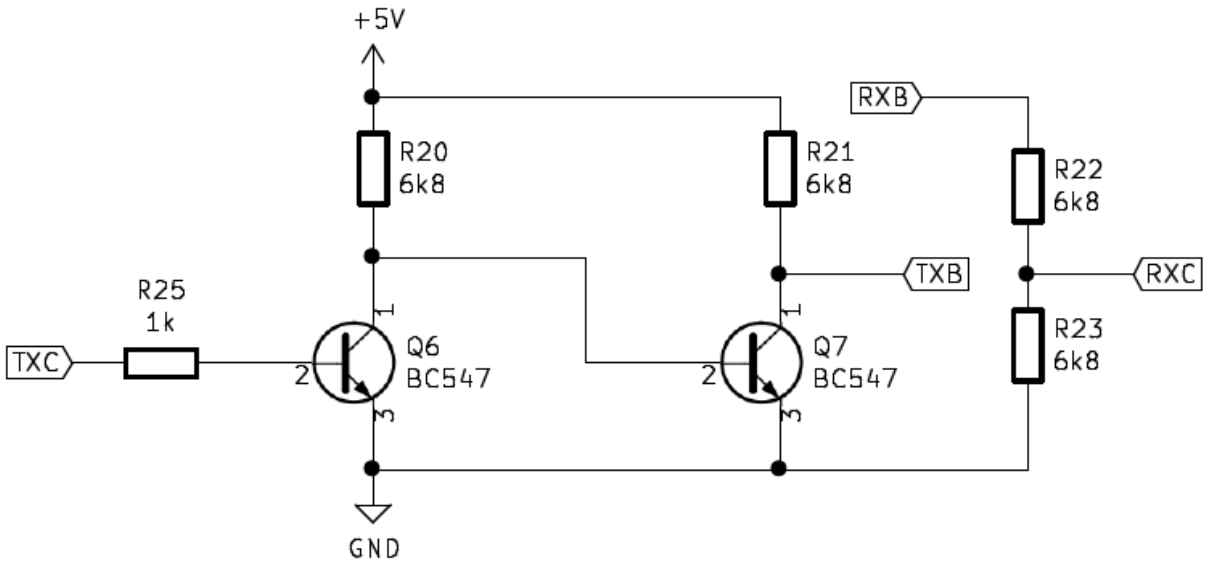
$$V_{CC} - I_{BQ5}R_{18} - V_{BEQ5} = 0 \tag{14}$$

Struja baze I_{BQ5} iznosi:

$$I_{BQ5} = \frac{V_{CC} - V_{BEQ5}}{R_{18}} = \frac{5\ \text{V} - 0{,}7\ \text{V}}{6{,}8\ \text{k}\Omega} = 0{,}6324\ \text{mA} \tag{15}$$

Što je dovoljno da tranzistor Q5 ode u zasićenje. Struja kolektora je ista kao i kod Q4. Izlazni napon se kreće od 0 V do $V_{CC} - V_{CEQ4,SAT}$ što iznosi 4,5 V. To je dovoljno za predviđen rad. Izlaz ostaje netaknut, makar bi bilo dobro dodati otpor za ograničenje struje (da mala odstupanja napona na inverteru i matičnoj ne stvore prevelike struje), to nije potrebno jer je ekvivalentan Theveninov otpor na inverteru dovoljno velik da to ograniči.

5.1.2 Izlazna komunikacija

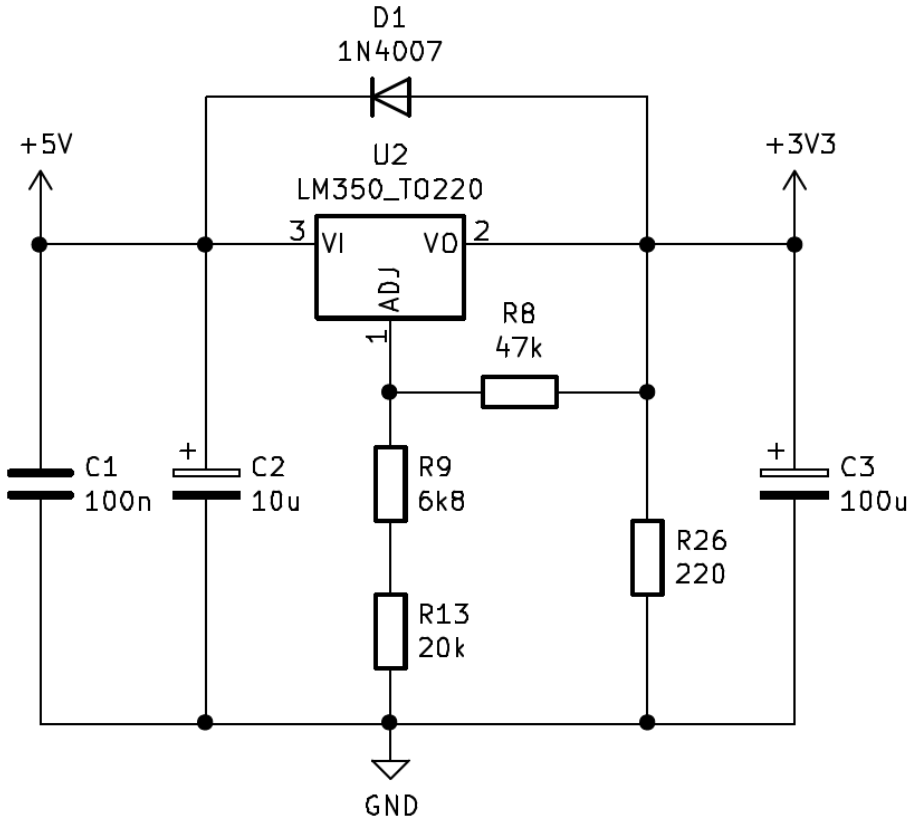


Slika 35 - Sklop promjene logičke razine izlaza

ESP8266 radi na naponu od 3,3 V i ne tolerira 5 V signale. Da bi se spriječilo uništenje ESP-a, potreban je sklop za promjenu logičke razine. Prvi dio se sastoji od 2 tranzistora i slijedi isti proračun kao i komunikacija sa inverterom (Slika 34), samo što U_{TXC} poprima napon od 3,3 V u logičkoj jedinici. Ovaj puta tranzistori Q6 i Q7 pojačavaju napon U_{TXC} na 5 V, tako da ga Atmega328P može pouzdano očitati. Drugi dio sheme je izlazni. Napon U_{RXB} se dijeli naponskim dijelilom na pola i osigurava dovoljnu impedanciju da ne vuče previše struje iz ESP-a (Jer izlaz djelila bude 2,5 V kod logičke razine „1“, dok ESP8266 radi na 3,3 V).

5.1.3 Izvor napajanja za ESP8266

ESP8266 zahtjeva napon od 3,3 V za normalan rad. To se ostvaruje linearnim regulatorom napona LM350. Proračun je napravljen po podatkovnom listu Texas Instruments-a [13].



Slika 36 - Regulator napona za 3,3V

Izlazni napon V_O glasi:

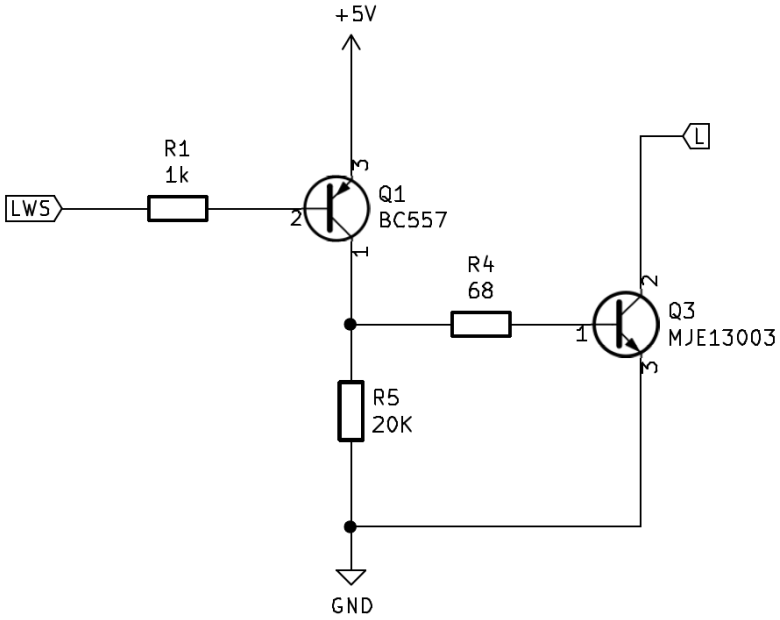
$$V_O = 1{,}25\left(1 + \frac{R_9 + R_{13}}{R_8}\right) + I_{ADJ} \cdot (R_9 + R_{13}) \tag{16}$$

$$V_O = 1{,}25\left(1 + \frac{26{,}8}{47}\right) + 50\mu\text{A} \cdot 26{,}8\,\text{k}\Omega = 3{,}3028\,\text{V} \tag{17}$$

Po preporuci proizvođača dodani su kondenzatori C2 i C3, dok je C1 dodan radi viših frekvencija. Dioda D1 je dodana kao zaštita od kratkog spoja na izlazu regulatora. Odabrana dioda je 1N4007 također prema preporuci proizvođača. Otpornik R_{26} je dodan kako bi se zadovoljila minimalna izlazna struja. Ako je izlazna struja premala, napon izlaza V_O raste, što je neprihvatljivo.

5.1.4 Sklop paljena rasvjete

Da bi se povećala pouzdanost, uštedjela energija i prostor, sklop za paljenje rasvjete je izveden sa tranzistorima. Ovaj sklop je moguće izvesti sa jednim darlington tranzistorom ili IGBT-om, ali je izabrana metoda sa 2 tranzistora.



Slika 37 - Sklop za paljenje rasvjete

Izlaz sklopa je otvoreni kolektor. To znači da rasvjetno tijelo (LED lampa) treba biti spojena na 12V i „L“ priključak. Tranzistor Q1 osigurava dovoljno veliku struju baze za zasićene tranzistora Q3. Ovaj sklop pali rasvjetu kada je U_{LWS} u logičkoj nuli. To je zbog tranzistora Q1 u spoju zajedničkog emitera (pojačanje je negativno).

Kirhofov zakon za napone kroz bazu Q1 (Ako je $U_{LWS} = 0$):

$$5 - V_{BEQ1} - I_{BQ1}R_1 = 0 \tag{18}$$

Iz toga slijedi:

$$I_{BQ1} = \frac{5 - V_{BEQ1}}{R_1} = 4{,}3\ \text{mA} \tag{19}$$

Maksimalna struja kroz R_5 iznosi:

$$I_{R5} = \frac{5 - V_{CEQ1,SAT}}{20\ \text{k}\Omega} = 0{,}225\ \text{mA} \tag{20}$$

Što znači da je tranzistor u zasićenju [14].

Kirhofov zakon za napone kroz bazu Q3:

$$5 - V_{CEQ1,SAT} - I_{BQ3} R_4 - V_{BEQ3} = 0 \tag{21}$$

Iz toga dobivamo struju baze:

$$I_{BQ3} = \frac{5 - V_{CEQ1,SAT} - V_{BEQ3}}{R_4} = \frac{5\,V - 0{,}5\ \text{V} - 0{,}7\ \text{V}}{68\ \Omega} = 55{,}8824\ \text{mA} \tag{22}$$

U idealnim uvjetima struja kolektora tranzistora Q3 iznosi:

$$I_{CQ3} = h_{fe,Q3} \cdot I_{BQ3} = 40 \cdot 55{,}8824\ \text{mA} = 2{,}235\ \text{A} \tag{23}$$

Ako uzmemo temperaturu okoliša od 35 °C i maksimalnu temperaturu čipa od 100 °C [15], možemo dobiti stvarnu maksimalnu snagu i struju kolektora:

$$P_{Q3} = \frac{T_j + T_a}{R_{\theta ja}} = \frac{100\ °\text{C} - 35\ °\text{C}}{89\ °\text{C/W}} = 0{,}7303\ \text{W} \tag{24}$$

$$I_{CQ3,MAX} = \frac{P_{Q3}}{V_{CEQ3,SAT}} - I_{BQ3} = \frac{0{,}7073\ \text{W}}{1\ \text{V}} - 55{,}8824\ \text{mA} = 674{,}4547\ \text{mA} \tag{25}$$

Što je skoro dovoljno za rasvjetu (600 mA). Potreban je hladnjak čija će veličina biti određena eksperimentalno.

Q3 je u zasićenju. Maksimalna struja kolektora tranzistora Q3 iznosi 1,5 A [15].

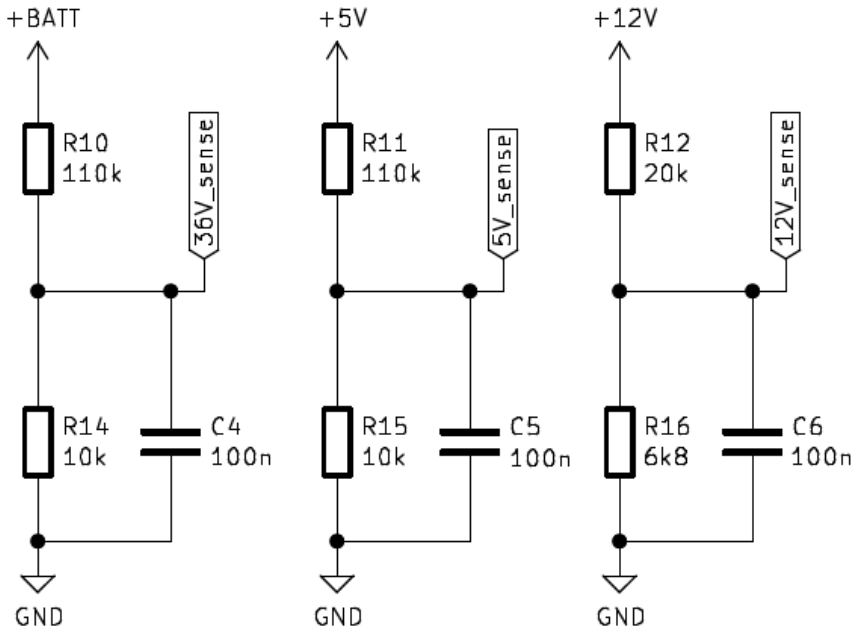
Struja emitera i snaga na tranzistoru Q1 iznosi:

$$I_{EQ1} = I_{BQ3} + I_{R5} = 56{,}1074\ \text{mA} \tag{26}$$

$$P_{Q1} = V_{CEQ1,SAT} \cdot I_{EQ1} = 0{,}5 \cdot 56{,}1074\ \text{mA} = 28{,}0537\ \text{mW} \tag{27}$$

5.1.5 Mjerenje napona

Mjerenje napona se vrši analogno digitalnim pretvaračem (ADC) integriranim u Atmega328P mikroupravljač. Da se ne bi prekoračio maksimalni ulazni napon, koriste se naponska dijelila.



Slika 38 - Naponska dijelila

Prvo dijelilo je za mjerenje napona baterije, dok je drugo za mjerenje radnog napona analogno digitalnog pretvarača. Makar je napon mjerenja drugačiji, 5 V dijelilo mora pustit napon ispod 1,1 V. To je zbog korištenja integrirane reference koja omogućava mjerenje radnog napona [10]. Treće dijelilo napona se koristi za dijagnostiku rasvjete i ispravnost 12 V ispravljača. Svakom dijelilu je dodan kondenzator koji s otpornicima tvori niskopropusni filter frekvencije koljena od:

$$f_{lpf} = \frac{1}{2\,\pi\,R\,C} = \frac{R_{10} + R_{14}}{2\,\pi\,R_{10}\,R_{14}\,C_4} = 173{,}6236\ \text{Hz} \tag{28}$$

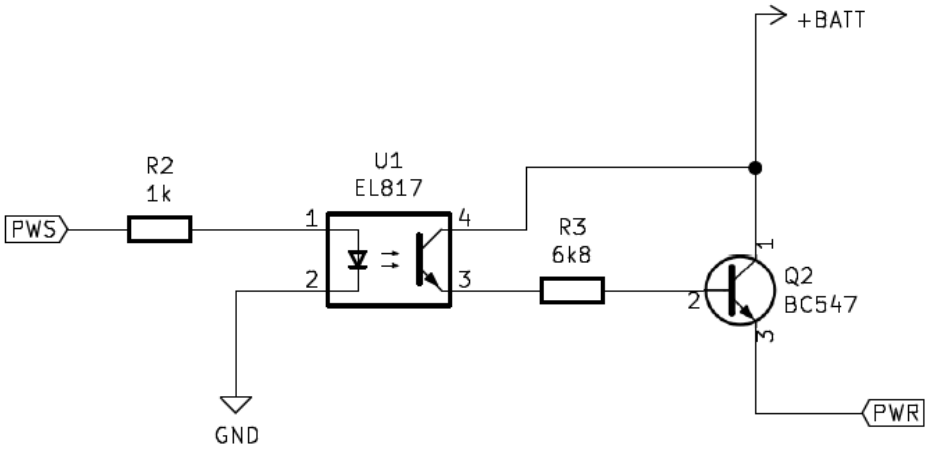
Gdje je R nadomjesni otpor R_{10} i R_{14} ili R_{11} i R_{15} koje kondenzator vidi kao paralelan spoj. Frekvencija koljena 12 V para:

$$f_{lpf} = \frac{R_{12} + R_{16}}{2\,\pi\,R_{12}\,R_{16}\,C_6} = 313{,}6289\ \text{Hz} \tag{29}$$

Filtriranje je dovoljno za eliminaciju šuma komunikacije i invertera, ali neće eliminirati niže frekvencije bez dovoljno velikog kondenzatora, koji će biti problem smjestiti na bakrenu pločicu. Umjesto većih kondenzatora, dodatno filtriranje će biti ostvareno softverski.

Nakon instalacije, korištenjem mjernog instrumenta će se izmjeriti stvarni naponi i kompenzacijski faktor će biti dodan u kod. To eliminira dodatne trimer potenciometre za fino ugađanje napona.

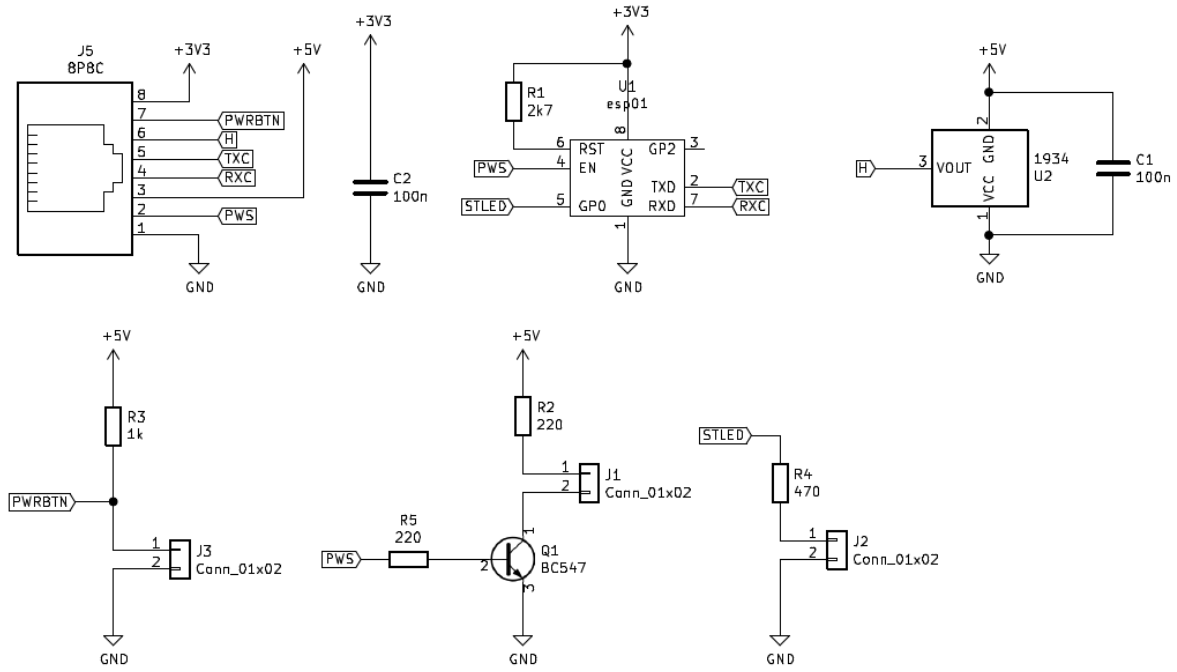
5.1.6 Sklop za paljenje invertera



Slika 39 - sklop paljenja invertera.

Inverter se uključuje u pogon tako da se na priključak PWR dovede napon baterije koji može dosegnuti 42 V. Theveninov otpor priključka PWR iznosi 2 kΩ. To zahtjeva velik napon na bazi da bi se stvorila dovoljna struja da dovede tranzistor u zasićenje. Najjednostavnije rješenje je dodati opto-izolator koji će dovesti napon baterije na bazu Q2 i tako ga staviti u područje zasićenja [16]. Otpor R_3 je dodan da ograniči struju baze i struju kolektora opto-izolatora.

5.1.7 Sklop LCD modula



Slika 40 - Shema LCD modula.

LCD modul koji će biti korišten na električnom romobilu sadrži ESP8266 na ESP01s modulu. LCD modul se spaja na matičnu ploču pomoću CAT5e kabela sa 8P8C RJ45 priključcima. Zbog nedostupnosti analogno-digitalnog pretvarača na ESP01s modulu pomak ručice akceleratora se šalje naponskim signalom na matičnu ploču. To može izazvati smetnje jer analogni signal putuje dugačkim vodičem do matične.

LCD modul sadrži gumb za paljenje romobila, lampicu indikatora stanja rada (kontakt) i statusnu lampicu (za greške ukoliko Wi-Fi zakaže). Na konektor J1 se spaja LED dioda, kao i na konektor J2. Na konektor J3 se spaja normalno otvoreni jedno položajni (SPST – single pole, single throw) gumb .

C1 i C2 su dodani za filtriranje visokofrekventnih šumova.

6.0 Programiranje

6.1 Matična ploča, Arduino (C++ jezik)

6.1.1 Inicijalizacija

```
void setup() {
  Serial.begin(9600, SERIAL_8N1);
  toEsp.begin(38400);
  packet_outbound[4] = 15; //gear
  DDRB |= (1<<5);

  DDRD = (1 << INVERTER_EN) | (1 << ESP_EN) | (1 << 1);
  DDRD &= ~((1 << POWERON_PIN) | (1 << 0));
  PORTD |= (1 << POWERON_PIN) | (1<<0); //pullup
}
```

Kod inicijalizacije mikroupravljača podešavaju se UART jedinice te smjer priključaka (ulaz ili izlaz). Otvara se hardverski asinkroni port za komunikaciju sa inverterom, te se stvara softverski asinkroni port za komunikaciju sa LCD modulom. Odabire se veća brzina komunikacije sa LCD modulom da ne traje predugo, jer zauzima vrijeme procesora (hardverska zahtjeva samo učitavanje sljedećeg bajta) koji mora emulirati UART [17]. Bit 5 na portu B je za indikatorsku LED lampicu. Postavljaju se smjerovi pinova na portu D za paljenje invertera, LCD modula. Osigurava se da je priključak POWERON_PIN ulaz.

6.1.2 Slanje podataka inverteru

```
void gt100_send(uint16_t spd, uint8_t gear) {
  packet_outbound[4] = 0;
  packet_outbound[16] = 0;
  packet_outbound[17] = 0;
  packet_outbound[19] = 0;
  packet_outbound[4] = gear;
  packet_outbound[17] |= spd & 0xFF; //donjih 8 bitova
  packet_outbound[16] |= spd >> 8; //gornjih 8 bitova

  packet_outbound[19] = chksum_xor(packet_outbound, 19); //provjera integr.

  for (uint8_t i = 0; i < 20; i++) {
    Serial.print((char)packet_outbound[i]); //slanje bajt po bajt
  }
}
```

Slanje podataka se vrši funkcijom gt100_send, tipa podatka void. Funkcija prima željenu brzinu kao broj od 0 do 1000 te stupanj prijenosa „gear“ od 0 do 15. Podatak „gear“ ne mijenja ponašanje invertera (barem ne u laboratorijskim testiranjima bez tereta), te je u glavnom kodu jednak 3.

Polje „packet_outbound“ je definirano na početku koda, a u ovoj funkciji se mijenjaju samo bitni bajtovi, podatak brzine i stupnja prijenosa. Pošto je format podatka „little endian“, logičkim operacijama se 16-bitna vrijednost brzine razdvaja na gornjih 8 bitova i donjih 8 bitova.

Nakon dodjeljivanja vrijednosti bajtova, izračunava se zadnji bajt kao „isključivo ili“ (bitwise XOR) svih 19 bajtova i sprema se kao 20. bajt. Bajtovi se šalju jedan po jedan metodom „print“ objekta „Serial“ [9].

6.1.3 Provjera integriteta paketa i računanje paritetnog bajta

```
uint8_t chksum_xor(uint8_t *message, uint8_t len) {
  uint8_t chksum = 0;
  for (uint8_t i = 0; i < len; i++) {
    chksum ^= message[i]; //xor
  }
  return  chksum;
}
```

Funkcija „chksum_xor“ služi za generiranje i provjeru integriteta paketa „*message“. Generacija pariteta se vrši operatorom „isključivo ili“ (bitwise XOR). Za neki paket od 20 bajtova paritetni bajt se računa tako da se pozove funkcija „chksum_xor“ sa adresom paketa kao argument i duljinom za jedan manjom. Funkcija tada vraća vrijednost zadnjeg bajta.

Za provjeru integriteta se poziva ista funkcija, ali se veličina polja ne umanjuje za jedan. Ako je rezultat funkcije nula, znači da je podatak došao ispravan (bez greške).

6.1.4 Slanje podataka LCD modulu

Podaci se šalju LCD modulu u istom formatu kao i inverteru, ali duljina paketa je 16 bajtova. Podaci koji se šalju LCD modulu:

- Period rotacije kotača, period (uint16_t)
- Prijeđeni put, trip (uint16_t)
- Sveukupni prijeđeni put, total (uint32_t)
- Napon baterije, vbat (uint16_t)
- Napon 5V, v5_vin (uint16_t)

Napon 5V je referentni napon za mjerenje napona baterije. Taj podatak je važan za točan izračun napona baterije, te se šalje zajedno sa naponom baterije. Naponi su izraženi kao vrijednosti od 0 do 1023 te se računaju na ESP8266 procesoru. To je napravljeno iz razloga što je EPS8266 brži procesor, i ne radi vremenski kritične zadatke kao Atmega328.

```
void toEsp_send(uint16_t period,
                uint16_t trip,
                uint32_t total,
                uint16_t vbat,
                uint16_t v5_vin) {
  packet_toEsp[3] = period & 0xFF;
  packet_toEsp[4] = period >> 8;

  packet_toEsp[5] = trip & 0xFF;
  packet_toEsp[6] = trip >> 8;

  packet_toEsp[7] = total & 0xFF;
  packet_toEsp[8] = total >> 8;
  packet_toEsp[9] = total >> 16;
  packet_toEsp[10] = total >> 24;

  packet_toEsp[11] = vbat & 0xFF;
  packet_toEsp[12] = vbat >> 8;

  packet_toEsp[13] = v5_vin & 0xFF;
  packet_toEsp[14] = v5_vin >> 8;
  packet_toEsp[15] = chksum_xor(packet_toEsp, 15);

  for (uint8_t i = 0; i < 16; i++) {
    toEsp.print((char)packet_toEsp[i]);
  }
}
```

6.1.5 Glavna funkcija (loop)

```
void loop() {
  if (!(PIND & (1 << POWERON_PIN))) {
    delay(1000);
    if (power) {
      power = 0;
      PORTB &= ~(1<<5);
      PORTD &= ~(1 << INVERTER_EN);
      PORTD &= ~(1 << ESP_EN);
    } else {
      power = 1;
      PORTB |= (1<<5);
      PORTD |= (1 << INVERTER_EN);
      PORTD |= (1 << ESP_EN);
    }
  }
  if (power) { //if turned on
    if(doOnceInAWhile == 0){
      analogReference(INTERNAL);
      v5Avg = 0;
      for(uint8_t vr = 0; vr<14; vr++){
        v5Avg += (float)analogRead(V5_IN)/14.0;
      }
      v5_vin = v5Avg;
      analogReference(DEFAULT);
    }

    delayMicroseconds(100);

    vbat = analogRead(VOLTAGE_IN);
    v12_vin = analogRead(V12_IN);
    throttle = analogRead(THROTTLE_IN);
    if (throttle < 50)
      throttle = 0;

    gt100_send(throttle, 9); //send data to inverter
    delay(10);

    if(Serial.available()){ //get data from inverter
      Serial.readBytes(packet_fromCnt, 14);
      uint8_t cnt = 0;

      if (!chksum_xor(packet_fromCnt, 14)) {
        period = (uint16_t)packet_fromCnt[9] | (uint16_t)packet_fromCnt[8] << 8;
      }

      while(Serial.available() > 0)
        char trash = Serial.read();
```

```
    }


  toEsp_send(period, trip, total, vbat, v5_vin); //send data to ESP

  if(toEsp.available()) { //get data from inverter
    toEsp.readBytes(packet_fromEsp, 2);
    if (!chksum_xor(packet_fromEsp, 2)) {
        if(packet_fromEsp[0] & (1<<0))    //svijetla
          PORTD |= (1<<LIGHTS);
        else
          PORTD &= ~(1<<LIGHTS);
    }
    while(toEsp.available() > 0)
        char trash = toEsp.read();
  }
  delay(100);
  doOnceInAWhile++;
  if(doOnceInAWhile > 1023)
      doOnceInAWhile = 0;
 }
}
```

U prvom djelu funkcije „loop“ se provjerava stanje priključka za paljenje električnog romobila. To se vrši bistabilnim „ili“ grananjem sa dodanim prisilnim čekanjem od 1000 milisekundi.

U glavnoj petlji se prvo izmjeri napon napajanja 5 V. Napon napajanja nije potrebno stalno mjeriti te se mjeri otprilike svakih 107 sekundi. Nakon toga se mjeri napon baterije, reference (5 V) i 12 V izvora. Mjeri se i napon halovog senzora koji služi kao senzor položaja akceleratora.

Nakon toga podaci se šalju inverteru funkcijom „gt100_send“. Odmah nakon slanja podataka inverter šalje podatke o brzini, koji se interpretiraju i pripreme za slanje LCD modulu. Ostatak bajtova se čisti sa ulaznog stoga kako bi se sljedeći put podaci mogli čitati od nulte pozicije.

Onda se podaci šalju LCD modulu. To se ostvaruje funkcijom „toEsp_send“. Nakon slanja se očekuje odgovor modula i trenutno jedini podatak koji šalje nazad je podatak o stanju rasvjete.

Na kraju petlje mikroupravljač čeka 100 mikrosekundi, te poveća varijablu „doOnceInAWhile“ za jedan. To omogućava mjerenje napona svakih 102 sekunde.

6.2 ESP8266 Kod (C++)

6.2.1 Inicijalizacija, void setup()

```
void setup() {
  delay(1000);
  Serial.begin(38400);

  WiFi.softAP(ssid, password); //Otvori AP
  WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));


  if (!SPIFFS.begin()) {;
    return; //stani ako SPIFFS nije pronađen ili ne radi
  }

  server.on("/", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send(SPIFFS, "/index.html"); //Ako klijent zatraži index.html, daj mu //ga
  });
  server.on("/spm_n.png", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send(SPIFFS, "/spm_n.png");
  });
  server.on("/spm_f.png", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send(SPIFFS, "/spm_f.png");
  });
  server.on("/info.html", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send(SPIFFS, "/info.html");
  });
  server.on("/getVals", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send(200, "text/plane", vals);
  });

  dnsServer.start(DNS_PORT, "*", apIP); //otvori DNS server
  server.addHandler(new CaptiveRequestHandler()).setFilter(ON_AP_FILTER);
//only when requested from AP

  server.begin();
}
```

Kod inicijalizacije se prvo otvara serijski port (UART) na istoj brzini kao softverski UART na matičnoj. Metodom softAP se inicijalizira Wi-Fi hotspot i sa metodom softAPConfig mu se dodjeljuje IP adresa (192.168.10.10) [18]. Trenutna implementacija ne podržava izmjenu lozinke ili imena Wi-Fi hotspota. Inicijalizira se SPIFFS (SPI flash file system) gdje se nalaze resursi za web stranicu. Nakon toga se instanci „server“ daju instrukcije što raditi ako klijent zatraži određeni web resurs. Te instrukcije šalju web stranicu, ali i podatke (getVals). Na posljetku se inicijalizira DNS server na portu 53 i sve upite preusmjerava na IP adresu

192.168.10.10 [19]. To stvara kaptivni portal (eng. Captive Portal) na kojih većina operativnih sustava daje upozorenje. To omogućava da korisnik brzo i jednostavno pristupi sustavu romobila bez upisivanja web adrese u pretraživač.

Metoda „begin“ diže WEB server.

6.2.2 Slanje podataka matičnoj ploči

```
void send_to_avr(uint8_t conf) {
  outbound[0] = conf;
  outbound[1] = conf;

  for (uint8_t i = 0; i < 2; i++) {
    Serial.print((char)outbound[i]);
  }
}
```

Ova jednostavna funkcija šalje 2 bajta matičnoj ploči. Jedan bajt su podaci o konfiguraciji a drugi je kontrolna suma. Pošto se radi o samo jednom bajtu, rješenje XOR pariteta je trivijalan, i zamijenjen je istim podatkom jer vrijedi:

$$A \oplus A = 0 \tag{30}$$

Gdje je „ $\oplus$“ operator logičke funkcije „isključivo ili“. Podaci se naravno uspoređuju bit po bit.

6.2.3 Glavna petlja

```
void loop() {
  dnsServer.processNextRequest();
  if(Serial.available()) { //get data from atmega
      Serial.readBytes(data, 16);
      if (!chksum_xor(data, 16)) {
        period = (uint16_t)data[3] | (uint16_t)data[4] << 8;
        trip = (uint16_t)data[5] | (uint16_t)data[6] << 8;
        vbat = (uint16_t)data[11] | (uint16_t)data[12] << 8;
        vcc = (uint16_t)data[13] | (uint16_t)data[14] << 8;

        total  = (uint32_t)data[7] | (uint32_t)data[8] << 8 | (uint32_t)data[9] << 16 |
(uint32_t)data[10] << 24;

        sc_speed = (6602.9143/(double)period);
        sc_trip = (double)trip/10;
        sc_total = (double)total/10;
        sc_vcc = (double)vcc/77.57575757;
        sc_vbat = ((double)vbat*sc_vcc)/85.3333333;

        sprintf(vals, "%.2f %.2f %.2f %.2f", sc_speed, sc_trip, sc_total, sc_vbat);
      }
      while(Serial.available() > 0)
         char trash = Serial.read();
  }
  delay(150);
}
```

U glavnoj petlji se otprilike svakih 150 milisekundi obrađuju DNS upiti (koji se naspram web servera ne izvršavaju asinkrono). Nakon toga se čitaju podaci koje je poslala matična ploča. Prvo se provjerava integritet paketa funkcijom „chksum_xor" koja je identična verziji na matičnoj ploči. Paket se interpretira i izračunavaju se točni podaci pomoću konstanti. Sve vrijednosti se pretvore u tekst (za slanje klijentu, radi lakše interpretacije u Javascript-u). Višak podataka se briše (ili cijeli paket ako je neispravan).

6.2.4 Klasa CaptiveRequestHandler

```
class CaptiveRequestHandler : public AsyncWebHandler {
public:
  CaptiveRequestHandler() {}
  virtual ~CaptiveRequestHandler() {}

  bool canHandle(AsyncWebServerRequest *request){
    //request->addInterestingHeader("ANY");
    return true;
  }

  void handleRequest(AsyncWebServerRequest *request) {
    request->send(SPIFFS, "/index.html");
  }
};
```

Ova klasa postoji da proslijedi adresu web resursa DNS serveru kod preusmjeravanja prometa. To je napravljeno zato jer metoda „addHandler "instance „server" (ESPAsyncWebServer.h) zahtjeva instancu koja rukuje sa zahtjevima [18]. Zato je „CaptiveRequestHandler" namijenjen samo da pošalje resurs „indeks.html" klijentu na zahtjev.

6.3 Web stranica (HTML, Javascript, CSS)

Web odredište koje je namijenjeno prikazu informacija korisniku, je napisana u 3 jezika: HTML-u, Javascript-u i CSS-u. Zbog uštede prostora na ESP-u, sva tri se koriste odjednom (ne razdvojeno kao u praksi) u istoj datoteci.

Javascript služi za logiku. U JS-u su kodirani filteri i dinamičko uređivanje web odredišta, te slanje zahtjeva ESP-u za vrijednosti na romobilu. JS upravlja elementima web odredišta kao npr. analogne kazaljke brzine [20].

HTML služi za određivanje položaja elemenata na web stranici. HTML također određuje klase i imena elemenata, tako da se oni mogu uređivati preko CSS-a i JS-a [21].

CSS (eng. Cascading Style Sheets) služi za opisivanje izgleda elemenata na web odredištu. Bez CSS-a, web stranica ne bi mogla imati elemente lijepe oku. CSS također upravlja položajem nekih elemenata na web odredištu romobila (poput kazaljke za prikaz brzine) i bez CSS-a se takav kompliciran element ne može prikazati (CSS omogućuje postavljanje elemenata jedan iznad drugog) [22].

HTML i CSS neće biti pojašnjen u ovom poglavlju jer HTML samo određuje položaj, dok su u CSS-u jednokratno odrede atributi elemenata.

6.3.1 JS varijable

```
var speedAvg, vbat_pc, vbat_avg;
var raw_speed, raw_vbat, raw_km, raw_trip;
var inbound = [], processed = [];
var t1 = setInterval(updateVals, 200);
```

U prvom redu su deklarirane varijable za srednju brzinu, postotak baterije i srednju vrijednost napona na bateriji. U drugom redu su sirovi podaci koje web odredište primi sa ESP-a. U trećem redu se inicijaliziraju dva polja za primljene i obrađene podatke.

Varijabla t1 se ne koristi, ali u nju se pohranjuje ključ pomoću kojeg se kasnije može zaustaviti automatsko pozivanje funkcije „updateVals“ svakih 200 ms.

6.3.2 Funkcija refreshVals

```
function refreshVals(){
   setBatteryPc(vbat_pc);
   updateTrip(raw_trip);
   needle_setPos(raw_speed);
}
```

Ova funkcija poziva funkcije za modificiranje elemenata web odredišta kako bi se izmijenili elementi za prikaz informacija.

6.3.3 Funkcija updateVals()

```
function updateVals() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState === 4 && this.status === 200) {
      inbound = this.responseText;
      processed = inbound.split(" ");

      raw_speed = processed[0];
      speedAvg -= speedAvg/6;
      speedAvg += raw_speed/6;

      raw_trip = processed[1];
      raw_km = processed[2];

      raw_vbat = processed[3];
      vbat_avg -= vbat_avg/16;
      vbat_avg += raw_vbat/16;

      vbat_pc = 100*(raw_vbat - 30)/(42-30);
      refreshVals();
    }
  };
  xhttp.open("GET", "getVals", true);
  xhttp.send();
}
```

Ova funkcija se izvršava svakih 200 ms. Funkcija prvo šalje HTML zahtjev tipa „GET“ ESP-u. Primljene podatke tada sprema u varijablu „inbound“ i razdvaja ju po praznim mjestima (razmaci u poruci označavaju granice podataka). Razdvojeni podaci se filtriraju eksponencijalnim „moving average“ filtrom [23]. Filter prihvaća ovaj oblik:

$$x_n = x_n - \frac{x_{n-1}}{N} \tag{31}$$

$$x_n = x_n + \frac{U}{N} \tag{32}$$

Gdje je U nova vrijednost, dok je x srednja vrijednost. N je stupanje filtracije. Ovo se ponaša kao niskopropusni filtar, i implementiran je na web odredištu da se smanji opterećenje na sustavima romobila.

Web odredište također računa postotak baterije i poziva funkciju „refreshVals“.

6.3.4 Funkcija updateTrip

```
function updateTrip(tripValue){
   document.getElementById("trp").innerHTML = tripValue + ' km';
}
```

Ova funkcija traži element ključa „trp“ i postavi mu vrijednost trenutno prijeđenog puta i doda tekst „km“ kao mjernu jedinicu.

6.3.5 Funkcija needle_setPos

```
function needle_setPos(value) {
      if(value > 40 || value < 0)
         return;

      var current_rotation = 6.75 * value;
      document.getElementById("needle").style.transform = 'rotate(' + current_rotation
+ 'deg)';
}
```

Ova funkcija mijenja kut pokazivača „analognog“ pokazivača brzine. To ostvaruje mijenjanjem CSS atributa rotacije „rotation“.

6.3.6 Funkcija setBatteryPc

```
function setBatteryPc(perc) {
    if(perc > 100)
        perc = 100;
    if(perc < 0)
        perc = 0;

    document.getElementById("batteryIndicatorBar").style.width = perc + "%";

    if(perc < 21)
        document.getElementById("batteryIndicatorBar").style.backgroundColor =
"#f2300d";
    else
        document.getElementById("batteryIndicatorBar").style.backgroundColor =
"#4CAF50";
}
```

Ova funkcija radi na istom principu kao „needle_setPos“. Na početku funkcija provjerava dali je postotak u granicama od 0 do 100, i ako nije stavi ga u te granice. To je da element ne izađe iz viewport-a (van prikaza web odredišta; van ekrana). Ova funkcija direktno mijenja CSS atribut širine elementa zelene kućice proporcionalno postotku baterije. Ako je postotak baterije ispod 20 %, tada je kučica crvena.

7.0 Zaključak

Električni romobil je oblik mikrotransporta ili osobnog transporta praktičnog za kretanje po gradu i izbjegavanje gužvi. Oslobođenje od potrebne homologacije romobila do 25 km/h ide u prilog njegovom popularnošću [2]. Cijene baterija zbog električnih vozila će biti jeftinije, a i pogonski sustavi zbog sve više proizvođača pogonskih sustava. Dostupnost upravljačke elektronike će biti sve veća jer se bilježi rast industrije elektronike.

E-romobili ne zahtijevaju veliku pogonsku snagu i mogu bez problema ići 20 km/h i s tom brzinom ostvariti veću udaljenost. Tom prosječnom brzinom se kreću i biciklisti, te sa većom baterijom e-romobil može postići slične udaljenosti kao i prosječni biciklist. Električni romobil nije namijenjen za velike udaljenosti, ali u velikom gradu prosječna prijeđena dnevna udaljenost može biti nekih dvadesetak kilometara, ovisno o slučaju.

Neki električni romobili imaju odabir načina vožnje. Moguće je odabrati ekonomičnu vožnju (eng. Economy), normalnu vožnju dok neki električni romobili nude i „sport“ opciju. Ekonomičan način vožnje smanjuje krajnju snagu motora, produžuje se period ubrzavanja. To smanjuje prosječnu struju koja baterija treba dati, te tako produljuje autonomiju. Važno je napomenuti da je efikasnost baterije i dostupna energija veća kod manje struje. „Sport“ opcija se može naći na nekim romobilima i ona povećava snagu ali smanjuje efikasnost. To je zbog povećanja momenta motora jer pogonski sustav drži namote duže energizirane nego inače, što smanjuje efikasnost komutacije. Zbog povećanja srednje struje baterije smanjuje se kapacitet i tako smanjuje autonomija.

8.0 Kod

8.1 gt100_proto.ino (matična ploča)

```
#include<stdint.h>
#include <SoftwareSerial.h>

#define TOESP_TX_PIN 9
#define TOESP_RX_PIN 8

#define ESP_EN 3
#define INVERTER_EN 3
#define LIGHTS 4

#define POWERON_PIN 2

/*  ANALOG  */
#define VOLTAGE_IN A1
#define THROTTLE_IN A0

#define V5_IN A3
#define V12_IN A2

uint16_t v5_vin, v12_vin, vbat;

uint8_t packet_fromCnt[14];

uint8_t packet_outbound[20] = {
  1  ,
  20  ,
  1 ,
  2 ,
  0 , //gear
  128 ,
  80  ,
  0 ,
  200 ,
  4 ,
  3 ,
  0 ,
  100 ,
  20  ,
  1 ,
  34  ,
  0 , //ThrottleL
  0 , //ThrottleH
  12  ,
  0   //XOR
```

```
};
uint8_t packet_toEsp[16] = {
  1  ,
  20  ,
  4 ,
  0 , //periodL
  0 , //periodH
  0 , //tripL
  0 , //TripH
  0 , //totalL
  0 , //...
  0 , //...
  0 , //totalH
  0 , //BatL
  0 , //BatH
  0 , //VCCL
  0 , //VCCH
  0   //XOR
};

uint8_t packet_fromEsp[2];

uint16_t period, trip;
uint32_t total;


void gt100_send(uint16_t spd, uint8_t gear);
void toEsp_send(uint16_t period, uint16_t trip, uint32_t total, uint16_t vbat, uint16_t
v5_vin);
uint8_t chksum_xor(uint8_t *message, uint8_t len);

uint16_t throttle;
uint8_t power = 0;
SoftwareSerial toEsp(TOESP_RX_PIN, TOESP_TX_PIN); // RX, TX

void setup() {
  Serial.begin(9600, SERIAL_8N1);
  toEsp.begin(38400);
  packet_outbound[4] = 15; //gear
  DDRB |= (1<<5);

  DDRD = (1 << INVERTER_EN) | (1 << ESP_EN) | (1 << 1);
  DDRD &= ~((1 << POWERON_PIN) | (1 << 0));
  PORTD |= (1 << POWERON_PIN) | (1<<0); //pullup
}
uint16_t doOnceInAWhile = 0;
float v5Avg;
```

```
void loop() {
  if (!(PIND & (1 << POWERON_PIN))) {
    delay(1000);
    if (power) {
      power = 0;
      //PORTB &= ~(1<<5);
      PORTD &= ~(1 << INVERTER_EN);
      PORTD &= ~(1 << ESP_EN);
    } else {
      power = 1;
      //PORTB |= (1<<5);
      PORTD |= (1 << INVERTER_EN);
      PORTD |= (1 << ESP_EN);
    }
  }
  if (power) { //if turned on
    if(doOnceInAWhile == 0){
      analogReference(INTERNAL);
      v5Avg = 0;
      for(uint8_t vr = 0; vr<14; vr++){
        v5Avg += (float)analogRead(V5_IN)/14.0;
      }
      v5_vin = v5Avg;
      analogReference(DEFAULT);
    }

    delayMicroseconds(100);

    vbat = analogRead(VOLTAGE_IN);
    v12_vin = analogRead(V12_IN);
    throttle = analogRead(THROTTLE_IN);
    if (throttle < 50)
      throttle = 0;

    gt100_send(throttle, 9); //send data to inverter
    delay(10);

    if(Serial.available()){ //get data from inverter
      Serial.readBytes(packet_fromCnt, 14);
      uint8_t cnt = 0;

      if (!chksum_xor(packet_fromCnt, 14)) {
        period = (uint16_t)packet_fromCnt[9] | (uint16_t)packet_fromCnt[8] << 8;
      }

      while(Serial.available() > 0)
        char trash = Serial.read();
      }
```

```
    toEsp_send(period, trip, total, vbat, v5_vin); //send data to ESP

    if(toEsp.available()) { //get data from inverter
      toEsp.readBytes(packet_fromEsp, 2);
      if (!chksum_xor(packet_fromEsp, 2)) {
          if(packet_fromEsp[0] & (1<<0))    //svijetla
            PORTD |= (1<<LIGHTS);
          else
            PORTD &= ~(1<<LIGHTS);
      }
      while(toEsp.available() > 0)
          char trash = toEsp.read();
    }
    delay(100);
    doOnceInAWhile++;
    if(doOnceInAWhile > 1023)
       doOnceInAWhile = 0;
  }
}

void gt100_send(uint16_t spd, uint8_t gear) {
  packet_outbound[4] = 0;
  packet_outbound[16] = 0;
  packet_outbound[17] = 0;
  packet_outbound[19] = 0;
  packet_outbound[4] = gear;
  packet_outbound[17] |= spd & 0xFF;
  packet_outbound[16] |= spd >> 8;

  packet_outbound[19] = chksum_xor(packet_outbound, 19);

  for (uint8_t i = 0; i < 20; i++) {
    Serial.print((char)packet_outbound[i]);
  }
}

uint8_t chksum_xor(uint8_t *message, uint8_t len) {
  uint8_t chksum = 0;
  for (uint8_t i = 0; i < len; i++) {
    chksum ^= message[i];
  }
  return  chksum;
}
void toEsp_send(uint16_t period,
                uint16_t trip,
```

```
            uint32_t total,
            uint16_t vbat,
            uint16_t v5_vin) {

  packet_toEsp[3] = period & 0xFF;
  packet_toEsp[4] = period >> 8;

  packet_toEsp[5] = trip & 0xFF;
  packet_toEsp[6] = trip >> 8;

  packet_toEsp[7] = total & 0xFF;
  packet_toEsp[8] = total >> 8;
  packet_toEsp[9] = total >> 16;
  packet_toEsp[10] = total >> 24;

  packet_toEsp[11] = vbat & 0xFF;
  packet_toEsp[12] = vbat >> 8;

  packet_toEsp[13] = v5_vin & 0xFF;
  packet_toEsp[14] = v5_vin >> 8;

  packet_toEsp[15] = chksum_xor(packet_toEsp, 15);


  for (uint8_t i = 0; i < 16; i++) {
    toEsp.print((char)packet_toEsp[i]);
  }
}
```

8.2 esp8266_sdcard.ino (ESP8266 – LCD modul)

```
#include <FS.h>
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <Hash.h>
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <SPI.h>
#include <DNSServer.h>
//#include <SD.h>
DNSServer dnsServer;

const char* ssid     = "speedometer";
const char* password = "lkj987asd123";

#define EMA_speed_N 4.0f //exp moving average filter
#define CS_PIN 15
```

```
const byte DNS_PORT = 53;
IPAddress apIP(192, 168, 10, 10);
AsyncWebServer server(80);

uint8_t chksum_xor(uint8_t *message, uint8_t len);
void send_to_avr(uint8_t conf);

uint16_t period, trip, vbat, vcc;
uint8_t lights;
uint32_t total;
char vals[128];

double sc_speed, sc_trip, sc_vbat, sc_batpc, sc_total, sc_vcc;


class CaptiveRequestHandler : public AsyncWebHandler {
public:
  CaptiveRequestHandler() {}
  virtual ~CaptiveRequestHandler() {}

  bool canHandle(AsyncWebServerRequest *request){
    //request->addInterestingHeader("ANY");
    return true;
  }

  void handleRequest(AsyncWebServerRequest *request) {
    request->send(SPIFFS, "/index.html");
  }
};


void setup() {
  // put your setup code here, to run once:
  delay(1000);
  Serial.begin(38400);

  WiFi.softAP(ssid, password);
  WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));
  /*IPAddress IP = WiFi.softAPIP();*/
  //Serial.print("AP IP address: ");
  //Serial.println(apIP);
  //dataFile.print("AP IP address: ");
  //dataFile.println(IP);

  // Print ESP8266 Local IP Address
  Serial.println(WiFi.localIP());
  //dataFile.println(WiFi.localIP());
```

```
  if (!SPIFFS.begin()) {
    //Serial.println("Error mounting the file system");
    return;
  }
  // Route for root / web page
  //File file = SPIFFS.open("/file.txt", "w"); for spiffs

  server.on("/", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send(SPIFFS, "/index.html");
  });
  server.on("/spm_n.png", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send(SPIFFS, "/spm_n.png");
  });
  server.on("/spm_f.png", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send(SPIFFS, "/spm_f.png");
  });
  server.on("/info.html", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send(SPIFFS, "/info.html");
  });
  server.on("/getVals", HTTP_GET, [](AsyncWebServerRequest * request) {
    //String sensor_value = String(sc_speed);
    request->send(200, "text/plane", vals);
  });


  dnsServer.start(DNS_PORT, "*", apIP);
  server.addHandler(new CaptiveRequestHandler()).setFilter(ON_AP_FILTER);
//only when requested from AP

  /*
  double sc_speed, sc_trip, sc_vbat, sc_batpc, sc_total;
  */
  // Start server
  server.begin();


  //dataFile.close();
}

uint8_t data[16];


uint8_t conf;
uint8_t outbound[2];

//uint32_t mAvg[MA_FILTER_SIZE];
```

```
void loop() {
  dnsServer.processNextRequest();
  if(Serial.available()) { //get data from atmega
      Serial.readBytes(data, 16);
      if (!chksum_xor(data, 16)) {
        period = (uint16_t)data[3] | (uint16_t)data[4] << 8;
        trip = (uint16_t)data[5] | (uint16_t)data[6] << 8;
        vbat = (uint16_t)data[11] | (uint16_t)data[12] << 8;
        vcc = (uint16_t)data[13] | (uint16_t)data[14] << 8;

        total  = (uint32_t)data[7] | (uint32_t)data[8] << 8 | (uint32_t)data[9] << 16 |
(uint32_t)data[10] << 24;

        sc_speed = (6602.9143/(double)period);
        //sc_speed = sc_speed * (EMA_speed_N-1)/EMA_speed_N +
(6602.9143/(double)period)/EMA_speed_N;

        //New average = old average * (n-1)/n + new value /n
        //sc_speed = (double)period/240;
        sc_trip = (double)trip/10;
        sc_total = (double)total/10;
        sc_vcc = (double)vcc/77.57575757;
        sc_vbat = ((double)vbat*sc_vcc)/85.3333333;

        sprintf(vals, "%.2f %.2f %.2f %.2f", sc_speed, sc_trip, sc_total, sc_vbat);


        /*
        double sc_speed, sc_trip, sc_vbat, sc_batpc, sc_total;
        */
      }
      while(Serial.available() > 0)
         char trash = Serial.read();
  }
  delay(150);
}

uint8_t chksum_xor(uint8_t *message, uint8_t len) {
  uint8_t chksum = 0;
  for (uint8_t i = 0; i < len; i++) {
    chksum ^= message[i];
  }
  return  chksum;
}

void send_to_avr(uint8_t conf) {
  outbound[0] = conf;
  outbound[1] = conf;
```

```
  for (uint8_t i = 0; i < 2; i++) {
    Serial.print((char)outbound[i]);
  }
}
```

8.3 indeks.html (Web stranica)

```
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<!DOCTYPE html>
<html>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>

.gauge1 {
  display: block;
  margin-left: auto;
  margin-right: auto;
  width: 90vw;
}

#needle {
    top: -90vw;
    left: 0%;
    position: relative;
    transition:all 0.2s;
}
.trip {
    position: relative;
    top: -104vw;
    text-align: center;
}
h1 {
    text-align: center;
}
footer {
    position: relative;
    top: -90vw;
    text-align: center;

}
```

```
.controleg1 {
  background-color: #4CAF50; /* Green */
  border: 1px solid black;
  color: white;
  padding: 15px 20px;
  width: 50%;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  cursor: pointer;
  float: left;
}

.control1 {background-color: #4CAF50;} /* Green */
.control2 {background-color: #008CBA;} /* Blue */

#batteryBar {
  .gauge1;
  background-color: #ddd;
  text-align: center;
}
#batteryText {
    position: relative;
    top: -40px;
    text-align: center;
}
#batteryIndicatorBar {
  .gauge1;
  width: 25%;
  height: 30px;
  background-color: #4CAF50;
}

</style>
<title>
Speedometer
</title>
<body onload="onLoad()">
<h1>Speedometer</h1>

<div id="batteryBar">
   <div id="batteryIndicatorBar"></div>
</div>
<p id="batteryText">Battery</p>
<div>
    <img alt="" class="gauge1" id="face" src="spm_f.png">
```

```
    <img alt="" class="gauge1" id="needle" src="spm_n.png">
</div>

<div id="trp" class="trip gauge1">
    0 km
</div>
<footer>
    <!-- <button type="button" class="controleg1 control1"
onclick="document.location='settings.html'">Settings</button> -->
    <button type="button" class="controleg1 control2"
onclick="document.location='info.html'">Info</button>
</footer>

<script>


//var t2 = setInterval(updateTrip(getVar("getTrip")), 1500);
var speedAvg, vbat_pc, vbat_avg;
var raw_speed, raw_vbat, raw_km, raw_trip;
var inbound = [], processed = [];
var t1 = setInterval(updateVals, 200);

function refreshVals(){
    setBatteryPc(vbat_pc);
    updateTrip(raw_trip);
    needle_setPos(raw_speed);
}

function updateVals() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState === 4 && this.status === 200) {
      inbound = this.responseText;
      processed = inbound.split(" ");

      raw_speed = processed[0];
      speedAvg -= speedAvg/6;
      speedAvg += raw_speed/6;

      raw_trip = processed[1];
      raw_km = processed[2];

      raw_vbat = processed[3];
      vbat_avg -= vbat_avg/16;
      vbat_avg += raw_vbat/16;

      vbat_pc = 100*(raw_vbat - 30)/(42-30);
      refreshVals();
```

```
    }
  };
  xhttp.open("GET", "getVals", true);
  xhttp.send();
}

function updateTrip(tripValue){
    document.getElementById("trp").innerHTML = tripValue + ' km';
}

function needle_setPos(value) {
        if(value > 40 || value < 0)
            return;

        var current_rotation = 6.75 * value;
        document.getElementById("needle").style.transform = 'rotate(' + current_rotation
+ 'deg)';
}
function setBatteryPc(perc) {
    if(perc > 100)
        perc = 100;
    if(perc < 0)
        perc = 0;

    document.getElementById("batteryIndicatorBar").style.width = perc + "%";

    if(perc < 21)
        document.getElementById("batteryIndicatorBar").style.backgroundColor =
"#f2300d";
    else
        document.getElementById("batteryIndicatorBar").style.backgroundColor =
"#4CAF50";
}
</script>
</body>
</html>
```

8.4 settings.html (Web stranica)

```
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
    <style>
        .backButton {
            background-color: tomato; /* Green */
```

```
            border: 1px solid black;
            color: white;
            padding: 15px 20px;
            width: 100%;
            text-align: center;
            text-decoration: none;
            display: inline-block;
            font-size: 16px;
            cursor: pointer;
            float: left;
        }
        h1 {
            text-align: center;
        }
        table{
            width: 100%;
            text-align: left;
        }
    </style>
    <head>
        <title>info</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h1>Info</h1>
        <button type="button" class="backButton"
onclick="document.location='index.html'">Return</button>
        <div>
            <table>
            <tr>
              <th style="width:70%" >Parameter</th>
              <th style="width:20%" >Unit </th>
              <th style="width:30%" >Value</th>
            </tr>
            <tr>
              <td>Total distance elapsed</td>
              <td>km</td>
              <td id="total" >1</td>
            </tr>
            <tr>
              <td>Average speed</td>
              <td>km/h</td>
              <td id="avgSpeed" >12.5</td>
            </tr>
            <tr>
              <td>Battery voltage</td>
              <td>V</td>
```

```
            <td id="batV" >36</td>
          </tr>
          <tr>
            <td>Battery percentage</td>
            <td> </td>
            <td id="batperc" >50</td>
          </tr>
          <tr>
            <td>Number of errors</td>
            <td> </td>
            <td id="nErrs" >0</td>
          </tr>
          <tr>
            <td>Locked</td>
            <td>bool</td>
            <td id="locked" >0</td>
          </tr>
        </table>
      </div>
    </body>
    <script>

    var speedAvg, vbat_pc;
    var raw_speed, raw_vbat, raw_km, raw_trip;
    var inbound = [], processed = [];
    var t1 = setInterval(getInfo, 2000);

    function getInfo() {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function() {
          if (this.readyState === 4 && this.status === 200) {
            inbound = this.responseText;
            processed = inbound.split(" ");

            raw_speed = processed[0];
            raw_trip = processed[1];
            raw_km = processed[2];
            raw_vbat = processed[3];
            vbat_pc = 100*(raw_vbat - 30)/(42-30);

            document.getElementById("avgSpeed").innerHTML = 666;
            document.getElementById("batV").innerHTML = Math.round(raw_vbat * 10) /
10;
            document.getElementById("batperc").innerHTML = Math.round(vbat_pc * 10)
/ 10;
            document.getElementById("total").innerHTML = raw_trip;

        }
```

```
        };
        xhttp.open("GET", "getVals", true);
        xhttp.send();
    }


    </script>
</html>
```

9.0 Izvori

Tehnička dokumentacija (koja postoji) komponenti korištenih u pisanju ovog završnog rada će biti priloženi na disku.

[1]. Službena stranica Xiaomi Inc.

I https://www.mi.com/global/mi-electric-scooter/

[2]. Završni rad: Dominik Sremić: Konstrukcija i upravljanje električnim romobilom

[3]. Brushed DC Motor Fundamentals

I http://ww1.microchip.com/downloads/en/appnotes/00905a.pdf (Pristupljeno 12.09.2021)

[4]. www.allaboutcircuits.com

I https://www.allaboutcircuits.com/technical-articles/sensorless-brushless-dc-bldc-motor-control/ (Pristupljeno 12.09.2021)

[5]. Brushless DC (BLDC) Motor Fundamentals

I http://ww1.microchip.com/downloads/en/AppNotes/00885a.pdf (Pristupljeno 12.09.2021)

[6]. Srivatsa Raghunath: Hardware Design Considerations for an Electric Bicycle Using a BLDC Motor

I https://www.ti.com/lit/an/slva642b/slva642b.pdf?ts=1631700530439 (Pristupljeno 12.09.2021)

[7]. Bilal Akin and Manish Bhardwaj: Sensorless Trapezoidal Control of BLDC Motors

I https://www.ti.com/lit/an/sprabq7a/sprabq7a.pdf?ts=1631459469685&ref_url=https%253A%252F%252Fwww.ti.com%252Fsolution%252Fdrone-propeller-esc (Pristupljeno 12.09.2021)

[8]. Dokumentacija za ESP8266

I https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf (Pristupljeno 12.09.2021)

II https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266__Datasheet__EN_v4.3.pdf (Pristupljeno 12.09.2021)

[9]. The Effectiveness of Checksums for Embedded Networks
- https://users.ece.cmu.edu/~koopman/thesis/maxino_ms.pdf (Pristupljeno 12.09.2021)

[10]. Dokumentacija za Atmega328P
- https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf (Pristupljeno 12.09.2021)

[11]. Dokumentacija za Arduino nano
- https://www.arduino.cc/en/uploads/Main/ArduinoNanoManual23.pdf (Pristupljeno 12.09.2021)

[12]. Dokumentacija za BC547
- https://www.sparkfun.com/datasheets/Components/BC546.pdf (Pristupljeno 12.09.2021)

[13]. Dokumentacija za LM350
- https://www.ti.com/lit/ds/snvs772b/snvs772b.pdf?ts=1631397648275&ref_url=https%253A%252F%252Fwww.google.com%252F (Pristupljeno 12.09.2021)

[14]. Dokumentacija za BC557
- https://www.onsemi.com/pdf/datasheet/bc556b-d.pdf (Pristupljeno 12.09.2021)

[15]. Dokumentacija za MJE13003
- https://www.onsemi.com/pdf/datasheet/mje13003-d.pdf (Pristupljeno 12.09.2021)

[16]. Dokumentacija za EL817
- https://www.tme.eu/Document/371e110151e1c8afec5198356913d01a/EL817.pdf (Pristupljeno 12.09.2021)

[17]. Arduino SoftwareSerial izvorni kod
- https://github.com/arduino/ArduinoCore-avr/tree/master/libraries/SoftwareSerial (Pristupljeno 12.09.2021)

[18]. ESPAsyncWebServer
- https://github.com/me-no-dev/ESPAsyncWebServer (Pristupljeno 12.09.2021)

[19]. DNSserver (ESP8266)

- https://github.com/esp8266/Arduino/tree/master/libraries/DNSServer (Pristupljeno 12.09.2021)

[20]. Javascript priručnik za programere

- https://developer.mozilla.org/en-US/docs/Web/JavaScript (Pristupljeno 12.09.2021)

[21]. HTML priručnik za programere

- https://developer.mozilla.org/en-US/docs/Web/HTML (Pristupljeno 12.09.2021)

[22]. CSS priručnik za programere

- https://developer.mozilla.org/en-US/docs/Web/CSS (Pristupljeno 12.09.2021)

[23]. Dealing with measurement noise (A gentle introduction to noise filtering)

- https://web.archive.org/web/20100329135531/http://lorien.ncl.ac.uk/ming/filter/filewma.htm (Pristupljeno 12.09.2021)

[24]. Dokumentacija za 1N4007

- https://www.vishay.com/docs/88503/1n4001.pdf (Pristupljeno 12.09.2021)

[25]. Dokumentacija za Atmega2560

- https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf (Pristupljeno 12.09.2021)

[26]. Završni rad: Josipa Obrovac: Konstrukcija električnog romobila i provjera čvrstoće metodom konačnih elemenata

[27]. RCA Designer handbook – Solid-State power Circuits