Izrada mikroprocesorskih modula na primjeru upravljanja svjetlosnom signalizacijom automobila putem CAN BUS protokola

Ljuba, Milan

Undergraduate thesis / Završni rad

2015

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Polytechnic Pula - College of Applied Sciences / Politehnika Pula - Visoka tehničko-poslovna škola s pravom javnosti**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:212:586501

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**

Image not found or type unknown

Repository / Repozitorij:

Digital repository of Istrian University of applied sciences

zir.nsk.hr

Image not found or type unknown

1. UVOD

1.1. Općenito

U razvoju elektronike, posebice u svijetu integriranih poluvodičkih krugova, veliki dio zauzimaju mikroprocesori. Mikroprocesori su već razvijeni do visokog stupnja, a razvoj im niti u današnje vrijeme nije zaustavljen, te predstavljaju posebnu granu u digitalnoj elektronici. Na suvremenom tržištu sve je teže pronaći proizvod u kojem nema ugrađenog niti jednog, barem najosnovnijeg, pripadnika neke obitelji mikroprocesora. Ta tehnologija omogućila je veliki korak u napretku te ponudila mnogobrojna poboljšanja i olakšala životu ljudi u mnogim pogledima. Ono što je nekad bila ljudska mašta pretočena u znanstveno fantastičke filmove, danas je realnost.

1.2. Opis problema

Ovaj završni rad obrađuje problematiku upravljanja mikroprocesorskim modulima putem mrežnog protokola. Problematika je primijenjena u izradi modula za upravljanje automobilskom svjetlosnom signalizacijom.

1.3. Cilj i svrha

Cilj ovog završnog rada je osmisliti, razviti, realizirati, opisati te navesti neke od smjerova nadogradnji i novih primjena mikroprocesorskih modula koji komuniciraju putem mrežnog protokola.

Svrha ovog rada je stjecanje novih te nadopuna postojećeg znanja koje se koristi pri cjelokupnoj izradi rada te usavršavanje praktične vještine koja je primijenjena u izradi praktičnog dijela.

1.4. Polazna hipoteza

Polazna hipoteza je da je na jednostavnom primjeru moguće prikazati kompleksan sustav te izraditi pri tome jedinstvene, ali široko primjenjive mikroprocesorske module.

1.5. Metode istraživanja

Prilikom izrade pisanog dijela završnog rada, te prilikom izrade pripadajućeg praktičnog dijela korištene su sljedeće znanstveno – istraživačke metode:

- metoda analize
- metoda sinteze
- metoda deskripcije
- metoda modeliranja
- metoda promatranja
- metoda mjerenja.

U pisanom radu korištene su metode analize i sinteze u svrhu obrade teorije i podataka prikupljenih iz literature, dok je metodom deskripcije prikupljeno opisano kroz tekst. Metoda modeliranja poslužila je u praktičnom dijelu rada prilikom izrade električnih shema, tiskanih pločica i programskog rješenja. Metodama promatranja i mjerenja dobiveni su podaci kojima je potvrđen prvobitni model.

1.6. Struktura diplomskog rada

Diplomski rad se sastoji od sedam poglavlja, od čega je tema rada obuhvaćena u četiri poglavlja. Prvo poglavlje obuhvaća uvod u temu, iznosi se opis problema, cilj i svrha rada te hipoteza, zatim upotrijebljena metodologija te struktura rada. Drugo poglavlje odnosi se na CAN mrežni protokol gdje su prikazani povijesni pregled i tehnička obilježja. Treće poglavlje opisuje izvedbe modula, električne sheme, tiskane pločice i programska rješenja. U četvrtom poglavlju opisano je testiranje izrađenih modula. Peto poglavlje odnosi se na nadogradnje i daljnje primjene praktičnog dijela rada. U šestom se poglavlju donosi zaključak. Na samom kraju, u sedmom poglavlju rada, navedena je literatura, popis tablica, popis slika te su priloženi dodaci.

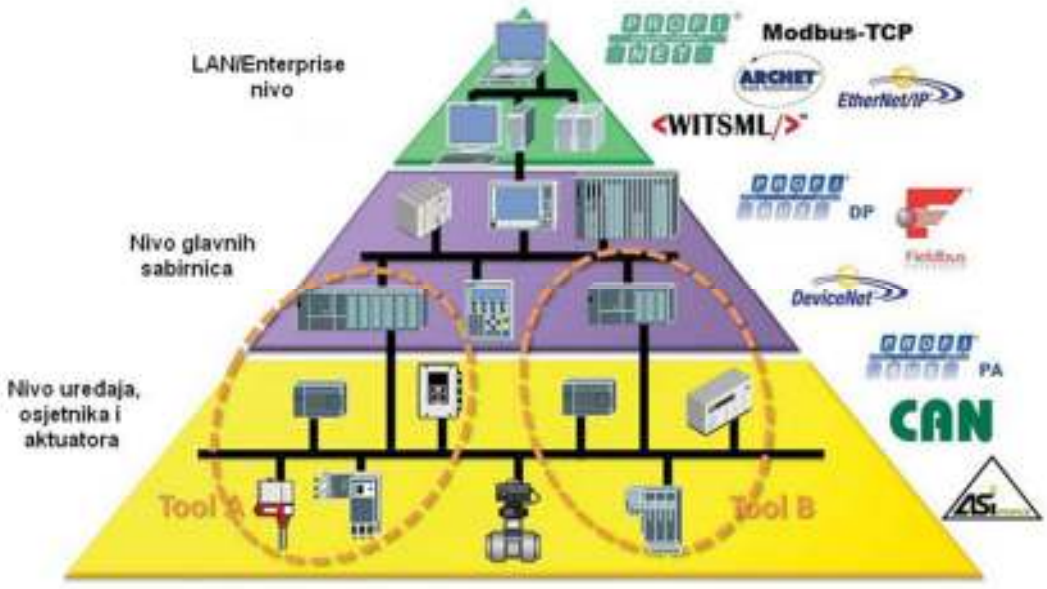
2. CAN MREŽNI PROTOKOL

2.1. Primjena u praksi

Moderni proizvodi današnjice sve su opremljeniji elektroničkim sklopovima i uređajima. U tim sklopovima sve češće se nalaze spojevi sa raznim obiteljima mikroprocesora, „malih“ računala, koja u jednom paketu sadrže gotovo sve elemente kao i osobna računala. Trend rasta takvih proizvoda na tržištu zahtijeva novi pristup, kako s konstrukcijskog pogleda, tako i s funkcionalnog.

Tržište se počelo popunjavati najnovijim generacijama proizvoda iz kategorije IoT, Internet Of Things, koji ne samo da nude korisniku viši stupanj interakcije s ugrađenom logikom, već uređaji sami osjećaju okolinu u kojoj se trenutno nalaze te stanja svojih senzora šalju putem digitalne mreže drugim uređajima ili korisnicima. Upravo ta zadaća s konstrukcijskog i funkcionalnog aspekta iziskuje više od jednog ugrađenog mikroprocesora, što stvara potrebu za komunikacijom među njima. Tipični primjer su automobili čiji sustavi u realnom vremenu prate stanje i prometne znakove te obavještavaju i upozoravaju vozača. Kao drugi primjer mogu se navesti uređaji za zabavu i dječje igračke koji prate vitalna stanja korisnika dok je u kontaktu s predmetom. Još jedan primjer, s više od jednog ugrađenog mikroprocesora, svakako su automatizirana postrojenja u tvornicama i same tvornice kao cjelina.

Kad govorimo o komunikacijskim protokolima u automatizaciji, postoje mnogi protokoli. Najpreglednije ih je kategorizirati po nivou ili sloju u kojem se koriste i za koji su namijenjeni te na taj način stvoriti piramidu kakva je prikazana na sljedećoj slici (Slika1.).



Slika 1. Piramida komunikacijskih protokola u automatizaciji

(izvor: http://www.drillingcontractor.org/wp-content/uploads/2011/09/RigComm-01.jpg, kolovoz, 2015.)

Sama piramida podijeljena je u tri sloja. Najniži, donji sloj rezerviran je za jednostavne uređaje, senzore i aktuatore. Uređaji u tom sloju predstavljaju po arhitekturi jednostavni mikroprocesori na čije su ulaze spojeni senzori koji pretvaraju fizikalne veličine u električne signale. Mikroprocesori digitaliziraju primljene signale, putem analogno digitalne konverzije, te im obradom kroz algoritme mijenjaju vrijednosti te pretvaraju u izlazne signale, putem digitalno analogne konverzije. Izlazi se upućuju prema aktuatorima koji ih pretvaraju u rad. Senzori su često izvedeni kao temperaturno promjenjivi otpornici, svjetlosno osjetljive diode ili tranzistori, mogu biti i klizne otporne staze pa sve do onih najjednostavnijih poput tipkala ili prekidača. Aktuatori, pak, mogu biti obični elektromotori istosmjerne struje, koračni elektromotori, pneumatski ventili, zasuni i pipci, ovisno o primjeni.

Pravilo je da svi elementi piramide ne djeluju „svojevoljno“ i bez komunikacije s ostalim susjednim slojevima. Zbog toga su razvijeni razni protokoli, a među najzastupljenijim je CAN mrežni protokol.

CAN protokol, dakle, spaja najniži sa srednjim slojem, u kojem se nalaze sve glavne sabirnice automatiziranog postrojenja. Također, i tu se nalaze po arhitekturi složeniji mikroprocesori koji imaju ugrađene algoritme kojima obrađuju pristigle podatke. Obrada tih podataka i svrha istih ovisi o stazi koju se automatizira i s kojom se želi upravljati. Dio tih podataka odlazi u više slojeve, kao informacija za prikaz na zaslonima, dok se ostali mijenjaju kroz algoritme. Izmjena se vrši prema ulaznim

podacima dospjelim iz višeg sloja i služi kao izlazni podatak prema nižem sloju. Sva komunikacija između dva niža sloja piramide vrši se preko CAN mreže u obliku standardnih i proširenih CAN digitalnih poruka. CAN protokol ima i svoju inačicu višeg standarda CANOpen i DeviceNet. Najviši sloj piramide sadrži protokole poput ProfiNet, ModBusTCP, ArcNET i već spomenutih CANOpen te DeviceNet. Ti protokoli spadaju u više slojeve jer su kompleksnijeg algoritma i već ih možemo smatrati aplikacijama. Aplikacije služe interakciji sa krajnjim korisnikom i preko njih korisnik očitava stanja senzora i položaje aktuatora na grafičkom sučelju radnog terminala ili računala postrojenja te može postrojenjem upravljati.

2.2. Povijesni razvoj

CAN protokol je prvi put predstavila tvrtka Robert Bosch GmbH 1986. godine na kongresu u gradu Detroitu u Sjedinjenim Američkim Državama. Sredinom 1987. godine veliki proizvođači integriranih krugova prepoznali su potencijalno širenje novo predstavljenog mrežnog protokola te je Intel predstavio tržištu CAN kontroler integrirani krug pod oznakom 82526. Nedugo zatim Philips predstavlja svoju inačicu pod oznakom 82C200.

Tvrtka Robert Bosch GmbH, danas poznatija pod nazivom BOSCH GmbH, nastavila je razvijati dobro prihvaćen CAN protokol. Naknadno je objavljeno još nekoliko verzija, a 1991. godine objavljena je verzija CAN 2.0. Ta verzija nosi dvije oznake. Prva, oznaka „A“ (standardna poruka), odnosi se na verziju CAN poruke koja sadrži jedanaest bitni identifikacijski kód, dok se druga, oznaka „B“ (proširena poruka), odnosi na CAN poruku koja sadrži dvadeset i devet bitni identifikacijski kód. Bit predstavlja jednu znamenku binarnog brojevnog sustava.

Godine 1993. ISO organizacija objavila je ISO 11898 standard za CAN protokol. Taj je standard restrukturiran u dva dijela te kao takav 2003. godine objavljen. Prvi dio, ISO 11898-1, pokriva podatkovni sloj dok drugi, ISO 11898-2, pokriva fizički sloj CAN protokola.

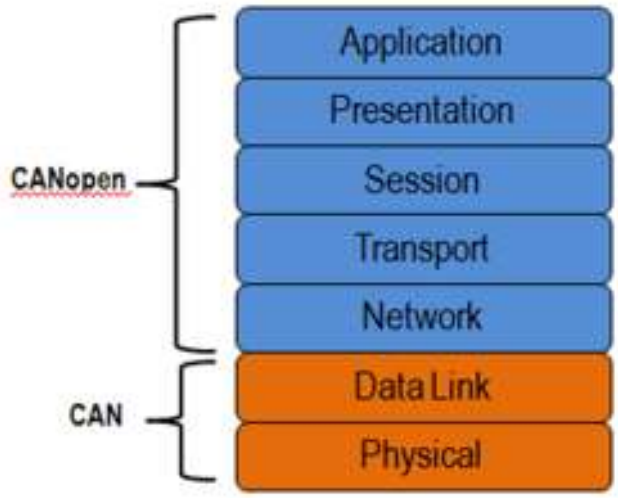
Dakako, BOSCH GmbH je i dalje aktivan u razvoju protokola. Godine 2012. tvrtka objavljuje CAN FD 1.0 (CAN Flexible Data-Rate). Ta specifikacija koristi fleksibilnu brzinu te još neke nadogradnje u komunikaciji. CAN FD je kompatibilan sa CAN 2.0 A i B uređajima te oni mogu bez problema koegzistirati na istoj mreži.

Ugrubo se procjenjuje da je do danas proizvedeno više od pet stotina milijuna uređaja opremljenih CAN tehnologijom.

2.3. Fizički i podatkovni sloj

2.3.1. Fizički sloj

Standardni model mrežnih komunikacija temeljen je na sedam ISO slojeva. Većina mikroprocesora i integriranih krugova ima ugrađen samo prvi, fizički sloj. Dakako, postoje izvedbe koje objedinjuju fizički, podatkovni pa čak i neke više slojeve u istom uređaju. Na slici 2. Prikazan je ISO referentni model slojeva koji koristi CAN protokol.

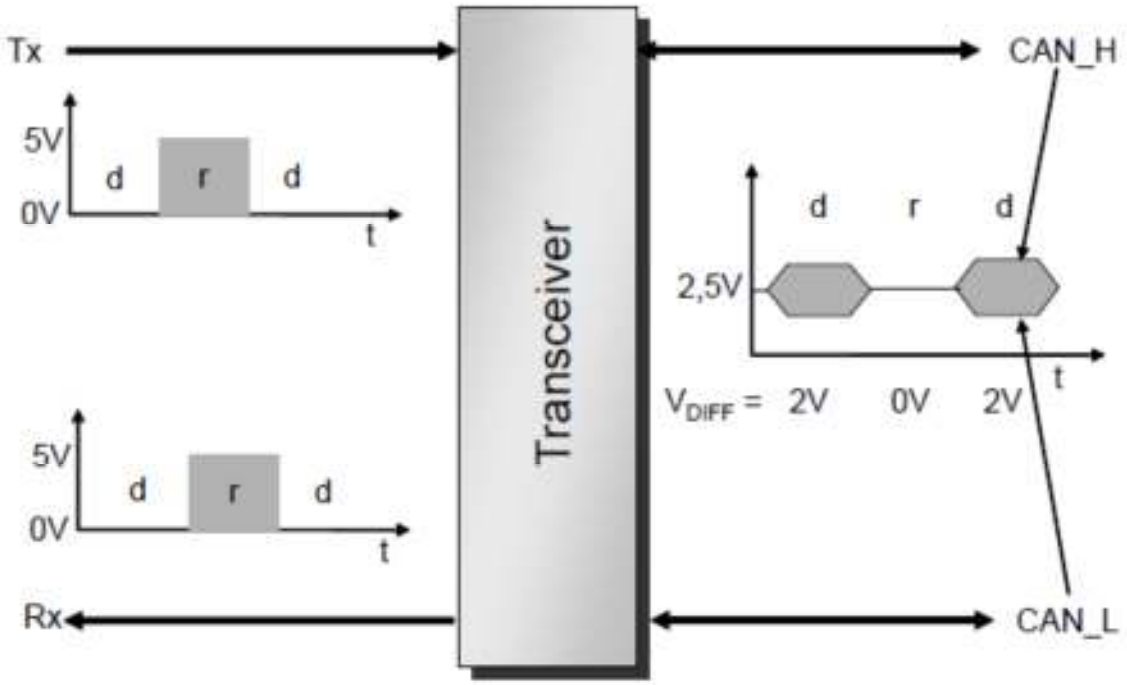


Slika 2. ISO referentni model prema CAN protokolu

(izvor: http://www.ni.com/white-paper/14162/en/, kolovoz, 2015.)

Slika 2. jasno prikazuje da su donja dva sloja rezervirani za CAN protokol koji se koristi u praktičnom dijelu ovog rada. Prije je spomenut (poglavlje 2.1) CANOpen viši protokol za koji se sa slike 2. razaznaje da pripada gornjim slojevima i doseže skroz do aplikativnog dijela, dakle najvišeg sloja, u poglavlju 2.1 objašnjene piramide.

Fizički sloj započinje sa sabirnicom, vodičima kroz koje teku struje i javljaju se naponska stanja. Vodiči su spojeni na primopredajnik koji je s druge strane spojen na TTL (tranzistor-transistorski logički sklop) vod koji povezuje primopredajnik i CAN kontroler. Primopredajnik, poput NXP-ovog modela TJA 1050 koji je korišten u praktičnom primjeru, standardiziran je prema uputama ISO – 11898 objašnjenim u poglavlju 2.2. Slika 3. prikazuje primjer spoja CAN primopredajnika.

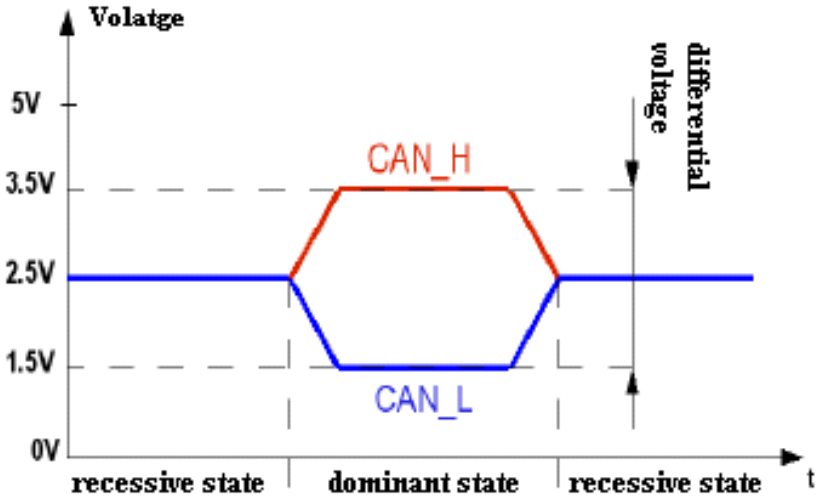


Slika 3. Primjer spoja CAN primopredajnika

(izvor: literatura [1], stranica 207, slika 5.1)

Primljeni TTL signal iz CAN kontrolera primopredajnik preoblikuje u diferencijalni CAN signal na vodičima sabirnice. Sabirnica se sastoji od dva vodiča. Prvi vodič je oznake CAN_H te označava visoko stanje signala (eng. high) dok drugi vodič ima oznaku CAN_L te označava nisko stanje signala (eng. low).

Oznake „d“ i „r“ na slici 3. označavaju dominantna odnosno recesivna stanja signala. U TTL-u recesivno stanje predstavlja logičku 1 (jedinica), tj. napon od 5V na liniji, dok napon od 0V predstavlja logičku 0 (nula) ili dominantno stanje. Slijedeća slika, slika 4. prikazuje naponske nivoe u vodičima CAN sabirnice.



Slika 4. Naponski nivoi CAN sabirnice

(izvor: http://canbus.pl/images/iso11898-levels_en.png, kolovoz, 2015.)

Za vrijeme mirovanja naponi na vodičima CAN_H i CAN_L iznose 2.5V, tj. diferencijalni napon je jednak 0V, suprotno TTL-u to označava recesivno stanje tj. logičku 1. Dominantno stanje, logička 0, CAN sabirnice je kad napon na CAN_L vodiču iznosi 1.5V, a napon na CAN_H vodiču 3.5V. Dakle dominantno stanje ima iznos diferencijalnog napona od 2V. Glavna prednost tako postavljenih naponskih nivoa za fizički sloj predstavlja otpornost CAN sabirnice na smetnje uzrokovane vanjskim utjecajima. Te smetnje, elektromagnetne interferencije (EMI) kad se pojave u okruženju CAN sabirnice, inducirati će na oba vodiča jednaku naponsku vrijednost što ne mijenja iznos diferencijalnog napona u oba slučaja, bilo recesivnog ili dominantnog stanja.

Izbor vodiča za sabirnicu su neoklopljeni upleteni kabeli, UTP (eng. Unshielded Twisted Pair) u područjima sa niskim EMI smetnjama, dok se suprotno koriste oklopljeni upleteni kabeli, STP (eng. Shielded Twisted Pair) čiji je oklop spojen na uzemljenje.

Dužina sabirničkog voda ovisi o brzini prijenosa podataka izraženoj u bitovima po sekundi (bit/s). U tablici 1. prikazane su najveće brzine prijenosa podataka u ovisnosti duljine CAN sabirnice. Vodiči na krajevima, neovisno o broju čvorova na sabirnici, moraju biti terminirani (spojeni) otpornikom vrijednosti 120Ω između CAN_L i CAN_H vodiča.

Tablica 1. Najveće brzine prijenosa podataka u ovisnosti duljine CAN sabirnice

(izvor: literatura [1], stranica 218, slika 5.10)

Max. brzina prijenosa podataka [kbit/s]	Duljina sabirnice L [m]
1000	40
500	100
250	250
125	500
40	1000

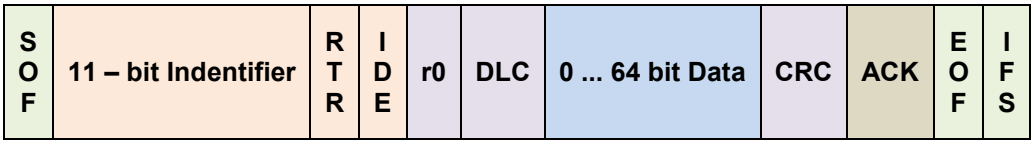
2.3.2. Podatkovni sloj

Podatkovni sloj CAN protokola odnosi se na poruku tj. podatke koje šaljemo između čvorišta na sabirnici. Poruke se šalju uobličene u podatkovne okvire. Postoje četiri vrste podatkovnih okvira koje možemo poslati i primiti. To su:

1. podatkovni okvir (eng. data frame)
2. okvir sa zahtjevom za podacima (eng. remote frame)
3. okvir pogreške (eng. error frame)
4. okvir odgode slanja (eng. overload frame).

1.Podatkovni okvir

Najčešće prisutni na CAN sabirnici je podatkovni okvir. Slika 5. grafički prikazuje podatkovni okvir.



Slika 5. Standardni podatkovni okvir CAN sabirnice

(izvor: autor, prema literaturi [3])

Prikazani okvir na slici 5. odnosi se na standardnu CAN poruku sa jedanaest bitnim identifikacijskim kodom. Dakle, to je poruka standarda CAN 2.0 A. Objašnjenja polja od kojih je sastavljen podatkovni okvir su sljedeća:

- **SOF** – (eng. start of frame) dominantni bit kojim se označava početak okvira. U mirovanju sabirnica je u recesivnom stanju, tako da svaki prijelaz iz recesivnog u dominantno stanje predstavlja početak okvira.
- **11 – bit Identifier** – dio okvira u kojem je upisan identifikacijski kód. U standardnoj poruci on je jedanaest bitni. Taj kód mora biti jedinstven za svaki čvor u sabirnici. Jedanaest bitni kód dozvoljava 2048 različitih čvorova (2^{11}). Bitno je naglasiti da čvorovi sa nižom binarnom težinom imaju prednost u prijenosu poruke nad čvorovima koji imaju viši težinski binarni broj. Primjer: Ako čvor 1 ima identifikacijski kód dekadske vrijednosti 5 (binarno 000 000 001 01), taj čvor će za vrijeme slanja poruke imati prednost nad svim čvorovima sa većom dekadskom vrijednosti (6,7,8...), ali ne nad onima s nižom (0,1,2,3 i 4).
- **RTR** – (eng. remote transmission request) nominalno dominantni bit koji se postavlja recesivno kad čvor koji ga šalje zahtijeva podatke od čvora kojem upućuje poruku. Na sabirnici svi će čvorovi primiti taj zahtjev, no identifikacijskim će se kodom odlučiti kome je namijenjen i od kojeg čvora se traže podaci.
- **IDE** – (eng. identifier extension) dominantni bit koji govori da je upravo poslani identifikacijski kód standardni, tj. jedanaest bitni.
- **r0** – rezervni bit, koristit će se za buduće nadogradnje CAN poruka.
- **DLC** – (eng. data length code) četiri bitno polje koje označava dužinu poruke koja slijedi.
- **0 ... 64 bit Data** – poruka koja se sastoji od osam polja po osam bitova, dakle ukupne duljine 64 bita.
- **CRC** – (eng. cyclic redundancy check) ovisno o duljini poruke pošiljatelj u ovom polju generira šesnaest bitni slog kojim primatelj može provjeriti integritet primljene poruke. Ciklička provjera redundancije koristi se za detekciju grešaka pri slanju.
- **ACK** – (eng. acknowledgement) svaki čvor koji primi ispravno poslanu poruku upiše u ovo polje dominantni bit. Taj bit prikazuje da je poruka poslana bez grešaka.

- **EOF** – (eng. end of frame) sedam bitno polje koje označava kraj poslane poruke.
- **IFS** – (eng. interframe space) sadrži sedam bitno polje u kojem je upisan vremenski interval potreban prijemnom kontroleru da poruku smjesti u memorijsku poziciju, te drugim kontrolerima da pokušaju započeti slanje nakon isteka ovog intervala.

Razlika između jedanaest bitne poruke (standard CAN 2.0A) i dvadeset i devet bitne poruke (standard CAN 2.0B) je u sljedećim poljima okvira:

- **SRR** – (engl. substitute remote request) zamjenjuje **RTR** bit iz standardne poruke i označava proširenu verziju poruke
- **IDE** – ovdje je to recesivni bit koji označava proširenu poruku standarda CAN 2.0B. Dvadeset i devet bitni identifikacijski kód, dozvoljava 2^{29}, što iznosi 536 870 912, različitih čvorova na istoj sabirnici.
- **r1** – nakon **RTR** i **r0** polja dodano je još jedno polje koje je rezervirano za buduće nadogradnje.

Ostatak poruke je jednak kao i kod standardne verzije. Slika 6. Grafički prikazuje prošireni CAN okvir.

SOF	11 – bit Indentifier	SRR	IDE	18 – bit Identifier	RTR	r1	r0	DLC	0 ... 64 bit Data	CRC	ACK	EOF	IFS

Slika 6. Prošireni podatkovni okvir CAN sabirnice

(izvor: autor, prema literaturi [3])

2. Okvir poruke sa zahtjevom za podacima

Svrha ovog okvira je zatražiti podatke od nekog čvora na sabirnici. Ovaj okvir je sličan podatkovnom, uz dvije razlike. RTR bit je u recesivnom stanju i eksplicitno označava zahtjev za podacima te je shodno tome podatkovno polje prazno.

3. Okvir pogreške

Okvir pogreške predstavlja specijalnu vrstu poruke na CAN sabirnici. Poslan je kad se detektira pogreška u prijenosu. Tada su svi čvorovi, osim pošiljatelja u čijoj je poruci pogreška otkrivena, prisiljeni poslati takav okvir. Pošiljatelj, nakon završetka slanja, zamjećuje okvir pogreške na sabirnici te šalje ponovno izvornu poruku. Uz osnovni mehanizam detekcije pogreške postoji i dodatni. Taj mehanizam detektira nenormalno ponašanje nekog od čvorova koji uzastopce šalje okvir pogreške iako za njim nema potrebe tj. pogreška ne postoji.

4. Okvir odgode slanja

Odgoda slanja je okvir vrlo sličan okviru pogreške, ali sa izmijenjenim poljima. Šalje ga čvor koji je zbog prioritetnijih zadaća mikroprocesora u nemogućnosti primiti i obraditi poruku koja mu je namijenjena.

Prikazani okviri predstavljaju vrste poruka kakve možemo zateći na sabirnici. Međutim, u kontrolere čvorišta ugrađene su i logike koje prepoznaju određena stanja, uz redovne okvire i osiguravaju normalan rad i brzu reakciju da bi se slanje i primanje između čvorova odvijalo nesmetano. Primjer takvih logika je kolizija s arbitražom. Ta logika je jedna od ključnih u CAN komunikaciji. Ukoliko se dogodi da više čvorova započne slanje poruka u isto vrijeme, samo će poruka veće važnosti (važnost se definira identifikacijskim kodom) proći do ciljanog odredišta. U trenutku istodobnog započinjanja poruka nastaje kolizija koja se rješava arbitražom među njima. Arbitraža se odvija temeljem troje ključnih predispozicija CAN sabirnice, a to su:

- svaki čvor ima jedinstven identifikacijski kód
- prilikom upisa logičke 0 (dominantno stanje) nekog promatranog čvora upis biva prepisan u logičku 1 (recesivno stanje) od strane drugog čvora sa višim prioritetom
- svaki čvor koji upisuje bit istovremeno sluša da li je upis promijenjen prije slanja drugog bita

Primjer: Ako dva čvora započnu istovremeno slanje poruke kolizija će nastati u trenutku kad oba upisuju svoje identifikacijske kodove na sabirnicu. Budući da identifikacijski kód čvora s većim prioritetom ima manju binarnu težinski vrijednost, prepisati će binarne jedinice drugog čvora, s nižim prioritetom, u binarne nule. U tom trenutku drugi čvor osluškuje sabirnicu i primjećuje da mu je kód promijenjen, da je nastala kolizija te da je izgubio arbitražu. Taj čvor prestaje sa slanjem poruke i čeka dok se na sabirnici ne pojave zadnja polja okvira (EOF i IFS) čvora koji je pobijedio u arbitraži. Zatim pokušava ponovno slanje nakon isteka IFS postavljenog vremenskog intervala. Algoritam koji je upisan u svaki hardver kontrolera (integrirani krug) pojednostavljeno djeluje na ovakav način:

1. *Čekaj dok sabirnica nije slobodna (dok se ne završi trenutni prijenos poruke)*
2. *Šalji dominantni početni bit (SOF)*
3. *Šalji prvi (ili sljedeći) bit, 11 – bitnog identifikacijskog kóda*
4. *Pročitaj poslano na sabirnicu*
5. *Ako je pročitani bit (pod 4.) <u>različit</u> od poslanog (pod 3.) primi dolaznu poruku (očito je drugi čvor pobijedio nastalu arbitražu) i vrati se na korak 1.*
6. *Ako je pročitano (pod 4.) <u>isto</u> kao poslano (pod 3.), šalji sljedeći bit tj. vrati se na korak 3.*
7. *Arbitraža dobivena, šalji podatkovni okvir.*

Upravo opisana logika omogućuje prijenos podataka putem CAN sabirnice na vrlo pouzdan i brz način. Poznat je statistički podatak, objavljen od strane proizvođača elektroničkih modula za auto industriju, da ako postavimo CAN sabirnicu na brzinu prijenosa podataka od 250 kbit/s te je u pogonu 2000 sati neprekidno, pod prosječnim opterećenjem od 25%, vjerojatnost ne detektirane pogreške u poruci jednaka je jednoj pogrešci svakih tisuću godina.

3. IZVEDBA MODULA

3.1. Opis

Praktični dio zamišljen je kao dva odvojena modula koja će komunicirati putem CAN sabirnice. Moduli su nazvani imenima „Dashboard“ (eng. naziv kojim se naziva upravljačka ploča) i „BDM“ (eng. naziv, skraćenica od Body Control Module) koji označava modul za kontrolu rada perifernih jedinica, u ovom primjeru, svjetala.

„Dashboard“ modul služi kao upravljački modul na kojem se nalazi pet tipkala kojima se kontrolira stanje svjetala spojenih na „BDM“ modulu. Tipkala su spojena na digitalne ulaze mikroprocesora „Dashboard“ modula i služe za upravljanje slijedećim svjetlima: „Pozicija, Kratko svjetlo, Dugo svjetlo, Lijevi žmigavac, Desni žmigavac“. Uz tipkala, na „Dashboard“ modulu, nalazi se LCD (eng. Liquid Crystal Display) zaslon na kojem je prikazana CAN poruka u binarnom obliku, CAN identifikacijski kód „Dashboard“ modula, te vrijednosti parametara temperature i radnih okretaja motora. Mikroprocesor „Dashboard“ modula učitava trenutna stanja tipkala, te na temelju toga, programskim algoritmom sastavlja CAN poruku koja se prenosi putem CAN kontrolera i sabirničkog primopredajnika. Uz navedeno, isti mikroprocesor prima CAN poruku pristiglu iz „BDM“ modula sa vrijednostima parametara temperature i radnih okretaja motora te ju ispisuje putem LCD zaslona.

Na „BDM“ modulu nalazi se mikroprocesor koji putem CAN kontrolera i sabirničkog primopredajnika, prima CAN poruku pristiglu iz „Dashboard“ modula. Ovisno o vrijednostima iz *Data* polja, opisanog u poglavlju 2.3.2, primljene CAN poruke, mikroprocesor mijenja digitalno stanje na izlazima. Izlazi su spojeni na LED (eng. Light Emitting Diode) svjetleće diode. Na analogne ulaze mikroprocesora spojena su dva promjenjiva otpornika (trimera) kojima se mijenja vrijednost napona u rasponu od 0 – 5 V na nožici klizača. Ti trimeri služe kao simulacija temperature i radnih okretaja motora.

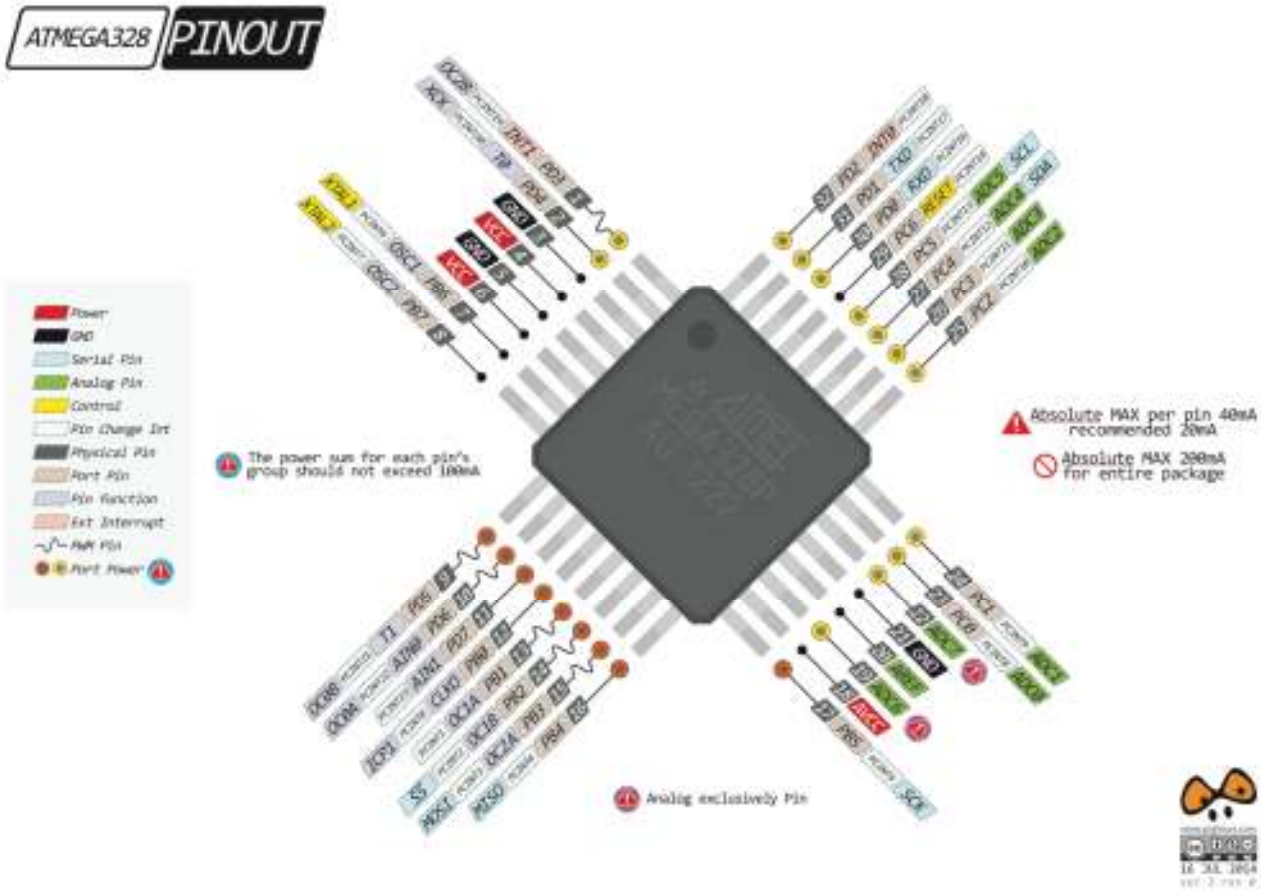
Za svaki modul razvijena je električna shema te na temelju sheme dizajnirana je tiskana pločica. Tehnologija koja je korištena za izradu modula je SMD (engl. Surface Mounted Devices), tehnologija površinske montaže elektroničkih elemenata. Tiskane pločice obložene su bakrenim slojem debljine 35 μm s obje strane. Osnovni materijal, na kojem je položen bakreni sloj, je laminat staklenih vlakana vezan epoksi fenol smolom, komercijalnog naziva *Vetronit FE2*. Razvijanje tiskanih pločica vršeno je foto postupkom, a izrada jetkanjem u klorovodičnoj kiselini.

Oba modula izgrađena su s Atmelovim 8 bitnim mikroprocesorom oznake ATmega328P u SMD kućištu oznake TQFP. Tablica 2. prikazuje osnovne te električne vrijednosti mikroprocesora, a slika 7. raspored nožica te pripadajući opis svake.

Tablica 2. Osnovne i električne karakteristike mikroprocesora ATmega328P

(izvor: autor, prema literaturi [4])

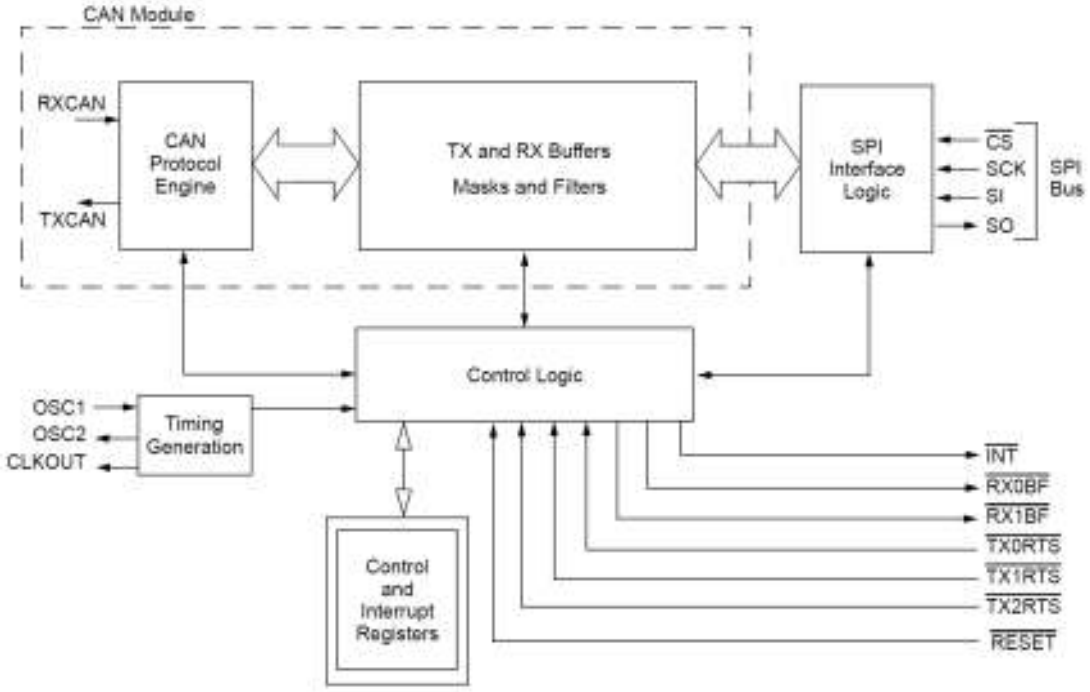
PARAMETAR	VRIJEDNOST
Radni napon	1.8 - 5.5V
Temperaturno područje rada	40°C do 85°C
Maksimalno strujno opterećenje jednog izlaza	40 mA
Maksimalno strujno opterećenje	200 mA
Programska memorija (FLASH)	32 Kbytes
Procesna memorija (RAM)	2 Kbytes
Broj nožica	28
Maksimalni radni takt	20 MHz
Tip procesora (CPU)	8-bit AVR
Ukupni broj nožica od toga slobodnih digitalnih ulaza / izlaza slobodnih analognih ulaza	26 - 14 8



Slika 7. Raspored nožica s opisom mikroprocesora ATmega328p

(izvor: http://www.pighixxx.com/test/wp-content/uploads/2014/11/atmega382mlf.png, kolovoz, 2015.)

CAN kontroleri, na oba modula, oznake su MCP2515, proizvođača Microchip. Također, za oba CAN kontrolera, odabran je brzi CAN primopredajnik oznake TJA1050, proizvođača NXP. Na slici 8. prikazan je osnovni blok dijagram za CAN kontroler MCP2515.



Slika 8. Blok diagram CAN kontrolera MCP2515

(izvor: literatura [5], strana 3, slika 1-1)

Kontroler je spojen putem SPI (eng. Serial Peripheral Interface) veze s mikroprocesorom. Tim putem predaje i prima poruku, te obavlja svu ostalu komunikaciju s mikroprocesorom. MCP2515 posjeduje, u svojoj unutarnjoj arhitekturi, dva prijemna buffera (sabirnika) te tri buffera za slanje što omogućuje nesmetanu komunikaciju na CAN sabirnici za vrijeme dok je mikroprocesor zauzet primopredajom ili obradom poruke. Jedna od posebnih mogućnosti ovog CAN kontrolera je prekidački (eng. interrupt) izvod, kojim se naređuje mikroprocesoru prekid tekuće rutine te posvećivanje CAN kontroleru. Interrupt kontroler oglašava logičkom nulom na INT nožici. Takav pristup korišten je u izradi ovih praktičnih primjera.

Primljenu CAN poruku kontroler prosljeđuje na sabirnicu putem TJA1050 primopredajnika. Primopredajnik je spojen s kontrolerom TTL vezom te primljene podatke konvertira u standardni CAN signal i odašilje na sabirnicu. I MCP2515 i TJA1050 dizajnirani su za CAN 2.0 A i B standarde, s maksimalnom brzinom prijenosa od 1 Mbit/s.

Uz CAN sekciju na oba modula ugrađen je pretvarač USB (eng. Universal Serial Bus) na UART (eng. universal asynchronous receiver/transmitter). Sklop služi za upisivanje programskog kóda u mikroprocesor te za interakciju mikroprocesora i računala. Sklop je građen s integriranim krugom CH340G, proizvođača WCH.

Za potrebe napajanja svih sklopova na oba modula, ugrađen je linearni stabilizator napona oznake AMS1117, proizvođača Advanced Monolithic Systems. Stabilizator pretvara viši ulazni napon na stabilnih 5V potrebnih za rad svih sklopova. Sklopovi se napajaju vanjskim 9V adapterom preko zajedničkog voda ili USB priključkom svaki zasebno. U tablici 3. prikazane su električne karakteristike modula.

Tablica 3. električne karakteristike modula

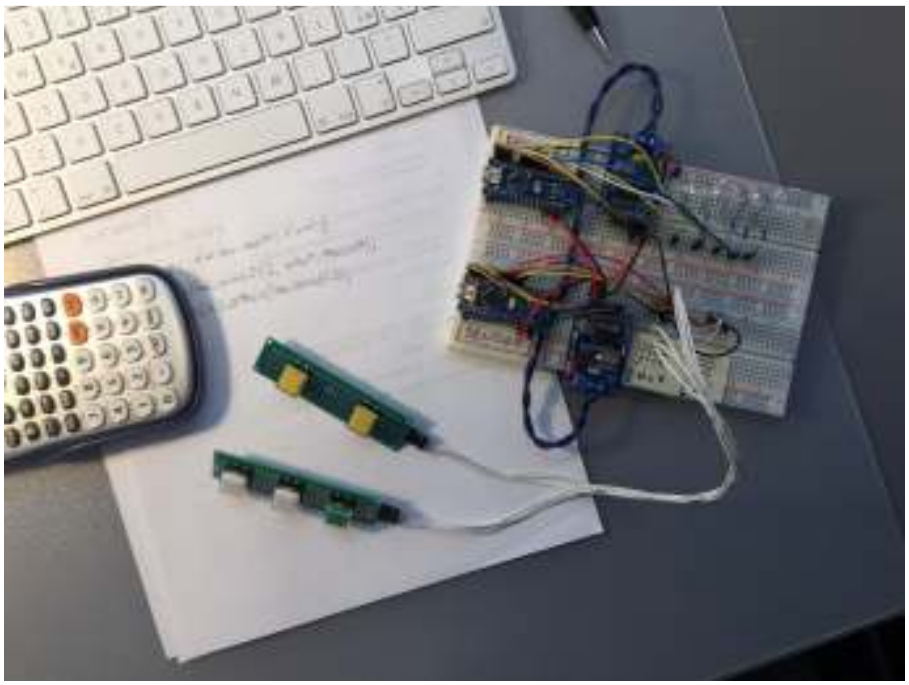
(izvor; autor, kolovoz, 2015)

PARAMETAR	„Dashboard“ modul	„BDM“ modul
Maksimalni ulazni napon	14V DC	14V DC
Struja u mirovanju	130mA	50mA
Struja u radu	130mA	135mA
Temperaturno područje rada	40°C do 85°C	40°C do 85°C

Modul „Dashboard“ s LCD zaslonom ugrađeni je na pokazni nosač izrađen od akrilne staklo-plastike. Nosač je prikazan na slici 9. Slika 10. prikazuje maketu sportskog automobila u koji je smješten „BDM“ modul s pripadajućim LED svjetlećim diodama, ugrađenim u stjenke makete.

3.2. Električne sheme

Izrada modula, a s time i električnih shema i tiskanih pločica, započinje nakon potvrde dizajna elektroničkog sklopovlja. Provjera dizajna vršena je na ubodnoj prototipskoj ploči. Slika 9. prikazuje izgled ploče sa sklopovima oba modula.

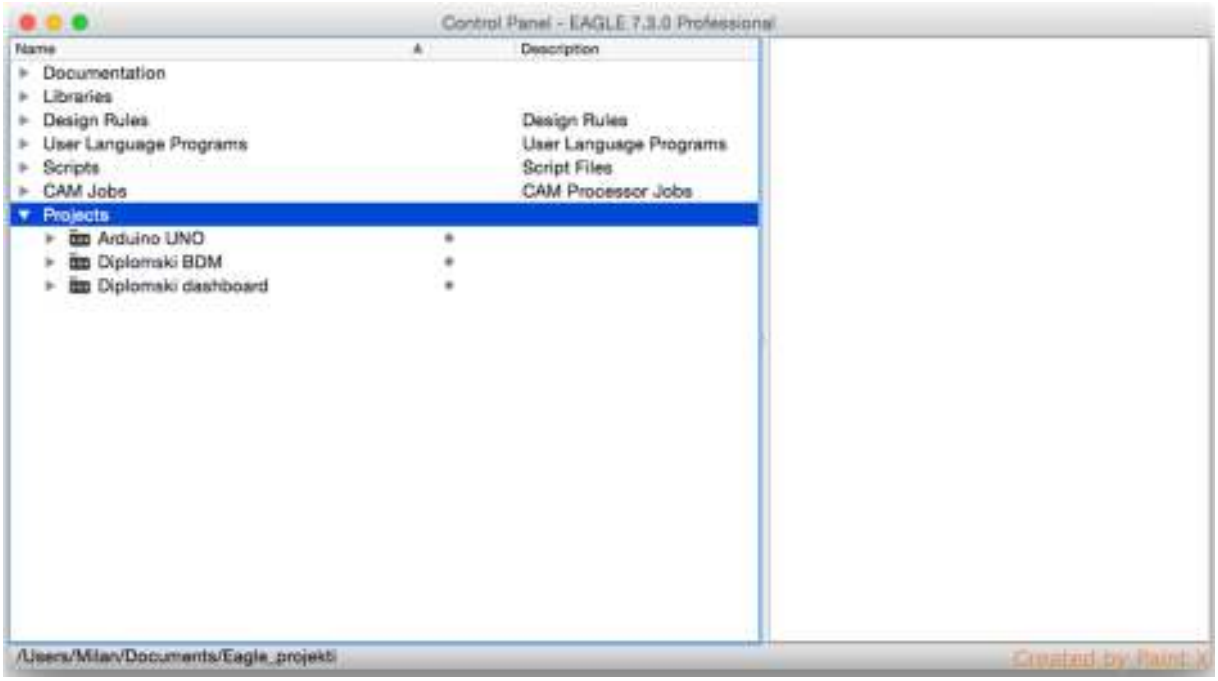


Slika 9. Izgled ploče sa sklopovima oba modula.

(izvor: autor, kolovoz, 2015.)

Električne sheme izrađene su u softveru EAGLE PCB Design, proizvođača CadSoft USA. EAGLE se sastoji od dvije glavne aplikacije; „Schematic“ u kojoj se izrađuju električne sheme i „Board editor“ u kojoj se dizajnira izgled tiskane pločice.

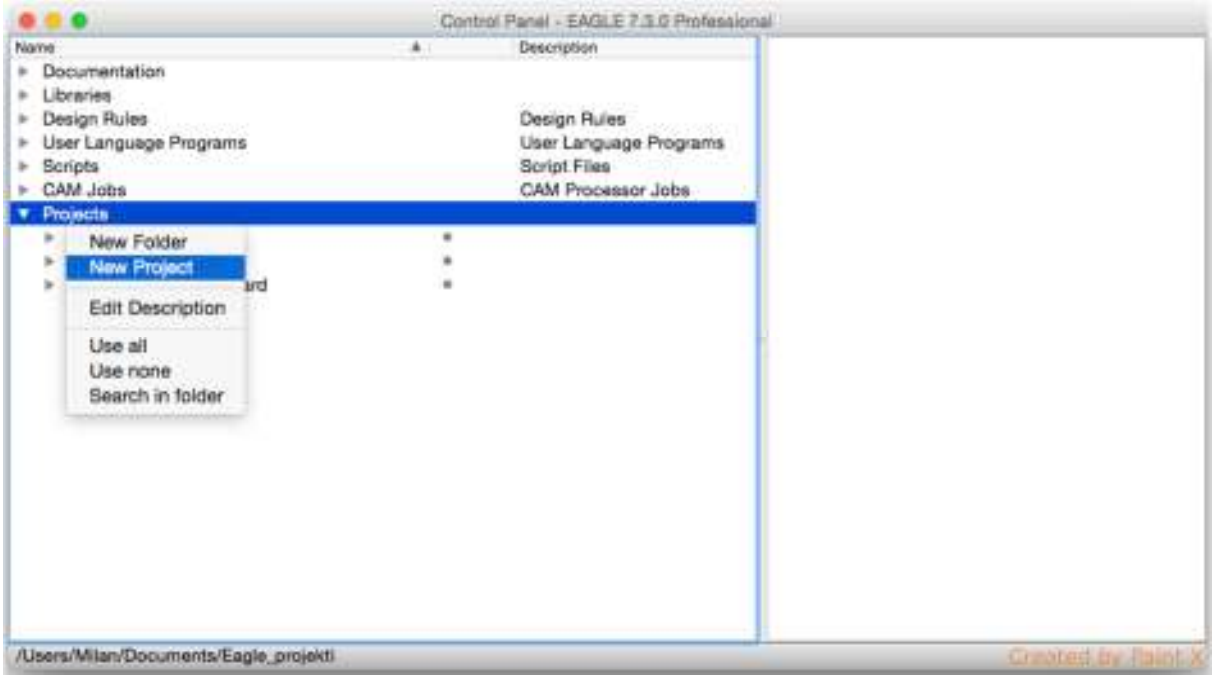
Pri pokretanju aplikacije EAGLE prvo se prikaže prozor upravljačkog panela (eng. control panel). Na slici 10. prikazan je izgled tog prozora.



Slika 10. Izgled upravljačkog panela EAGLE aplikacije

(izvor: autor, kolovoz 2015.)

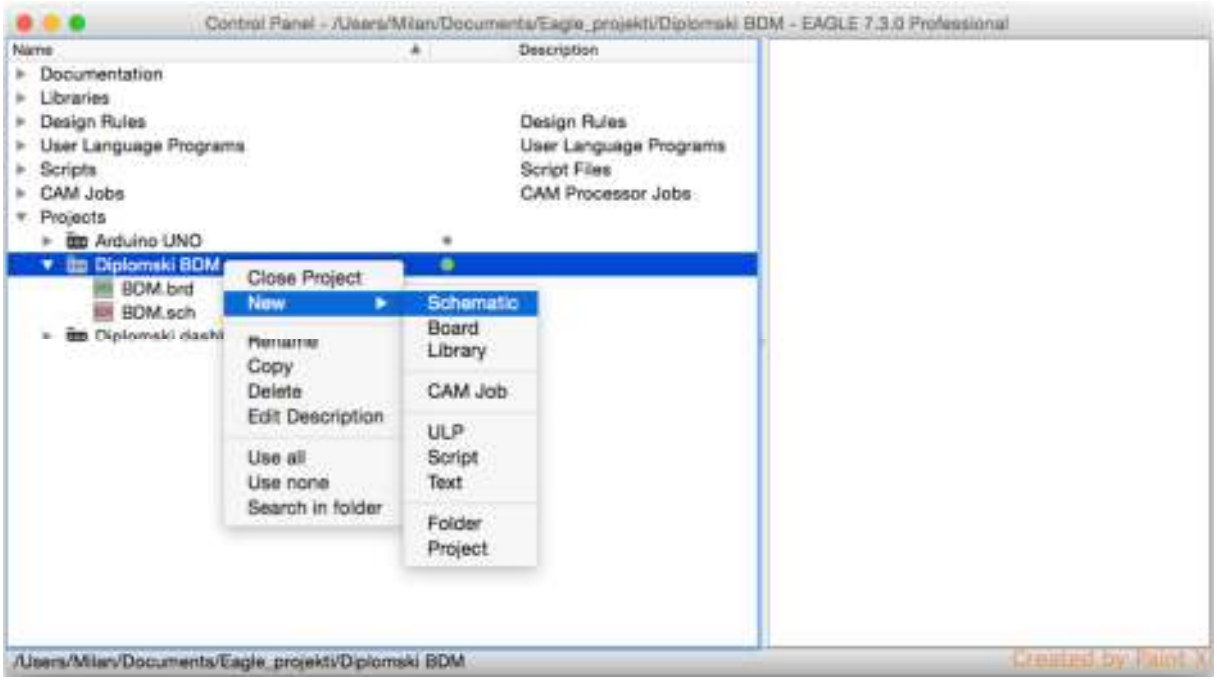
U control panelu pod menijem „Projects“ potrebno je desnim klikom miša otvoriti pod meni te lijevim klikom započeti novi projekt, pritiskom na stavku „New Project“ (slika 11.).



Slika 11. Kreiranje novog projekta

(izvor: autor, kolovoz 2015.)

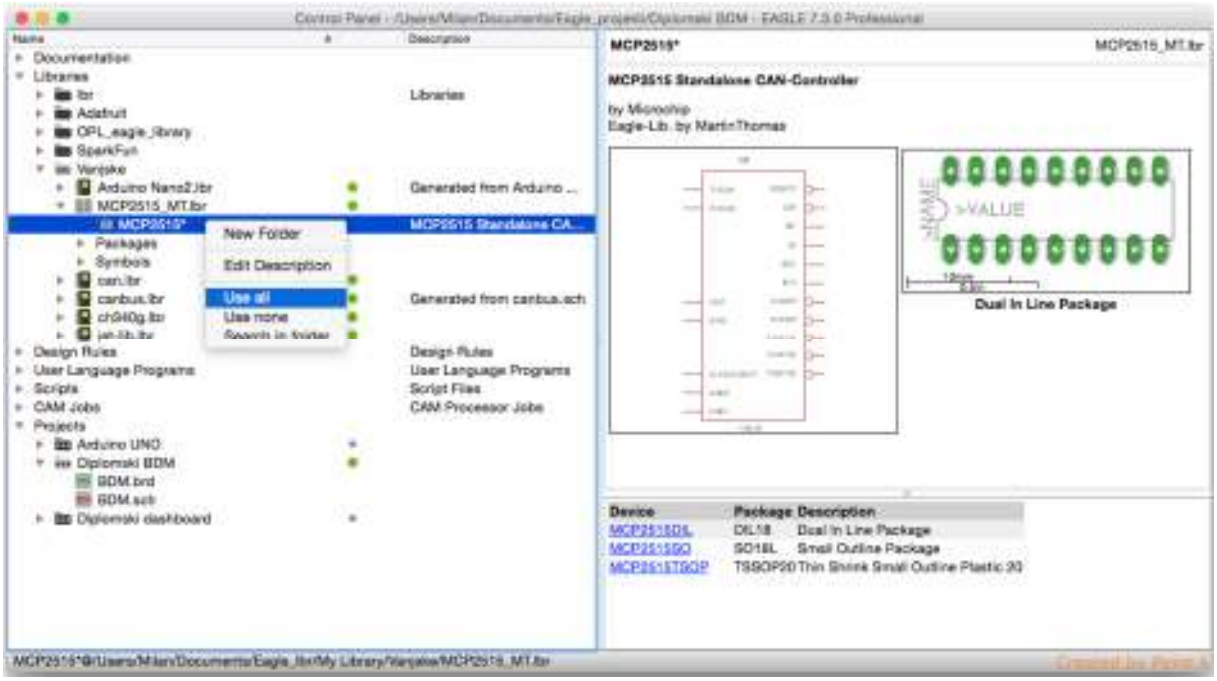
Po završetkom tog koraka, projektu je potrebno kreirati električnu shemu. Istim principom, kakvim je opisano otvaranje projekta (kombinacija desnih i lijevih klikova miša), stvara se električna shema. Slika 12. prikazuje izgled prozora prilikom kreiranja električne sheme.



Slika 12. Kreiranje električne sheme

(izvor: autor, kolovoz 2015.)

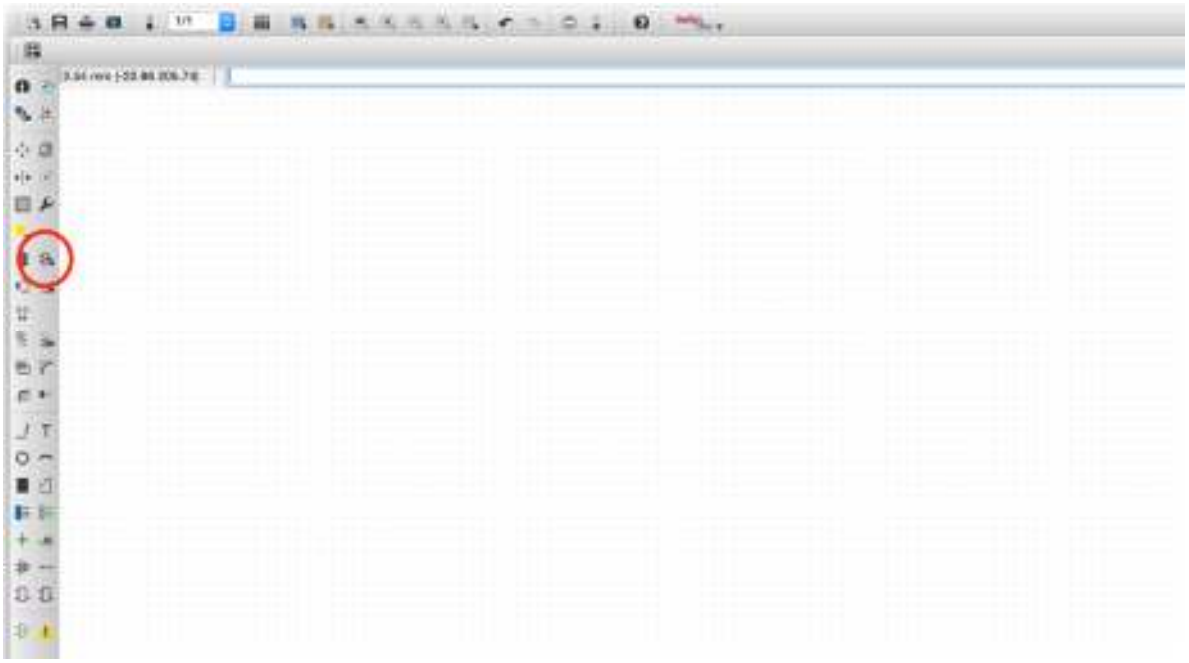
Po kreiranju nove električne sheme istoj je potrebno dodijeliti ime, a program sam dodaje ekstenziju imenu .sch. Nakon toga dodjeljuju se biblioteke elektroničkih elemenata koji će se koristiti u izradi. Biblioteke pronalazimo u meniju „Libraries“. Na slici 13. prikazan je pregled biblioteka s elektroničkim elementima. Na lijevoj strani prozora prikazan je naziv elementa, te naziv pripadajuće mu biblioteke. S desne strane prikazan je njegov izgled, kakav se koristi u izradi električne sheme, i izgled kućišta s pripadajućim otiskom nožica (eng. footprint). Nakon pronalaska zadovoljavajuće biblioteke za projekt ista se dodjeljuje projektu desnim klikom miša na ime biblioteke (ili elementa) te potvrdom lijevim klikom na stavku „Use all“.



Slika 13. Određivanje biblioteke projekta

(izvor: autor, kolovoz 2015.)

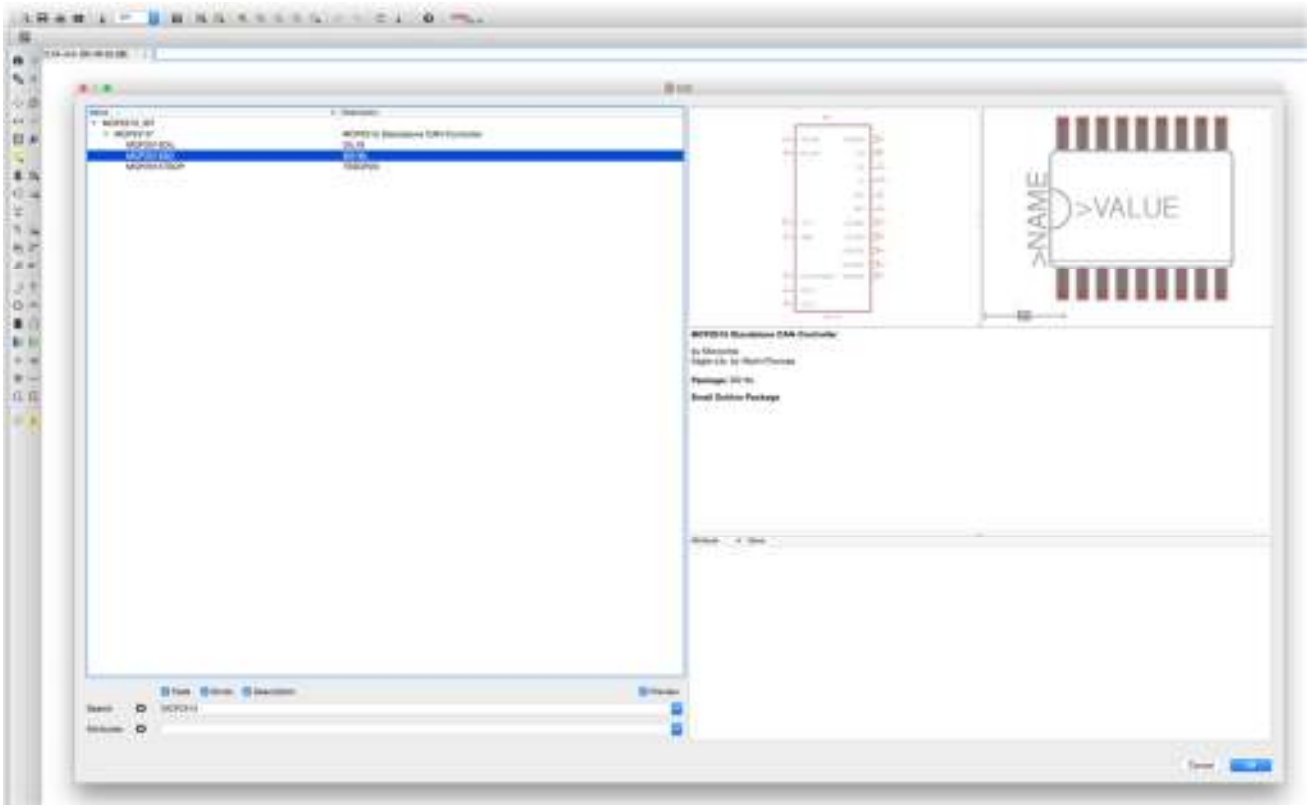
Nakon pridruživanja biblioteke pristupa se izradi električne sheme. Dvostrukim klikom miša na „*ime*.sch“ u meniju „Projects“ pokreće se aplikacija „Schematic“. Pri izradi električnih shema u početku je preporučljivo odabrati sve elemente koji se koriste u dizajnu te ih razmjestiti na radnu površinu aplikacije. Odabir se vrši na način da se iz alatnog izbornika, s lijeve strane prozora u aplikaciji, odabere alat „Add a part“. Na slici 14. prikazan je osnovni prozor „Schematic“ aplikacije na kojem je crvenim kružićem označen alat „Add a part“ u alatnoj traci.



Slika 14. Osnovni prozor „Schematic“ aplikacije

(izvor: autor, kolovoz 2015.)

Po pokretanju alata prikazuje se prozor izbornika iz kojeg, upisom u polju „Search“ smještenom u donjem lijevom dijelu prozora, tražimo element iz prije odabranih biblioteka. Slika 15. prikazuje izgled izbornika „Add a part“.

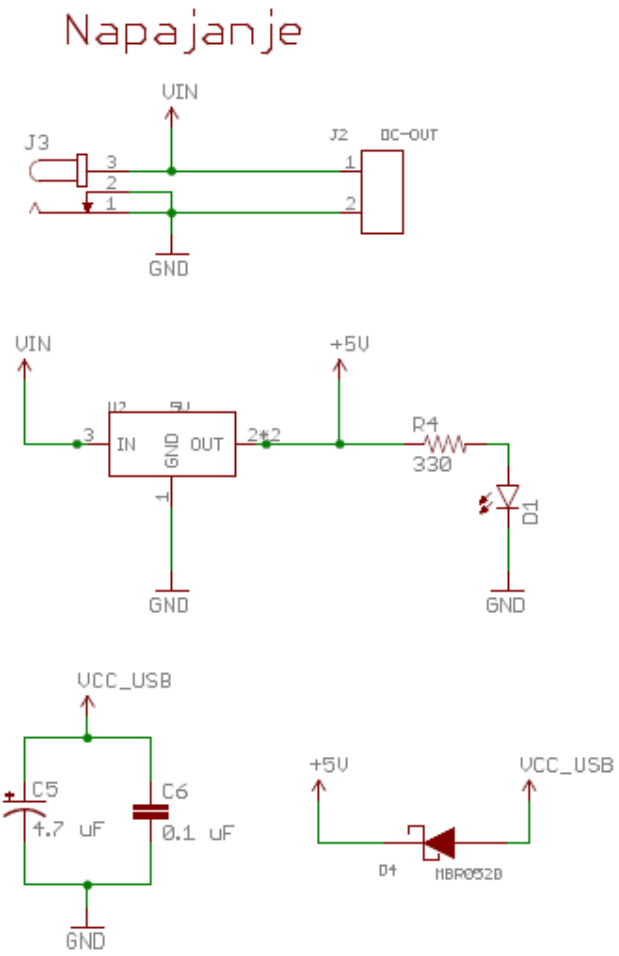


Slika 15. Izgled izbornika „Add a part“

(izvor: autor, kolovoz, 2015.)

Elementi se pretražuju po tvorničkoj oznaci ili oznaci proizvođača za skupni odabir. Na lijevoj strani izbornika (slika 15.) prikazuje se simbolički izgled elementa, kakav se koristi u „Schematic“ aplikaciji, te varijanta kućišta (paketa) s pripadajućim otiskom nožica (eng. footprint), kakvo se koristi u „Board editor“ aplikaciji. Točan odabir kućišta elementa je od velike važnosti iz razloga što se na temelju električne sheme, automatskim putem, generira tiskana pločica sa svim odabranim elementima u aplikaciji „Board editor“. Po završetku izbora započinje se s razmještanjem i povezivanjem izvoda elemenata.

Moduli „Dashboard“ i „BDM“ imaju jednak dizajn napajanja, u pogledu naponske regulacije i filtriranja. Razlika je jedino što na „Dashboard“ modulu postoji priključak za ulaz vanjskog izvora napajanja i priključna stezaljka koja odvodi taj izvor prema „BDM“ modulu. Vanjski izvor napajanja je u obliku naponsko-strujnog adaptera. Adapter pretvara 220V izmjenični izvor na 9V istosmjerni. Snaga adaptera iznosi 2.88VA. Shema napajanja prikazana je na slici 16.



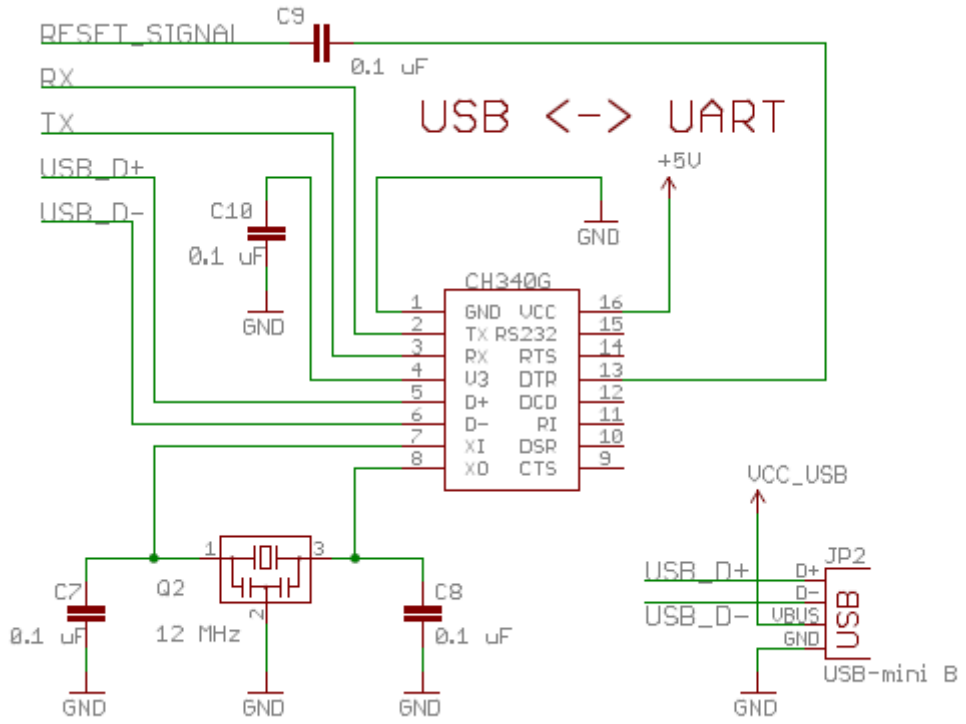
Slika 16. Električna shema napajanja

(izvor: autor, kolovoz, 2015.)

Priključak J3, sa slike 16., predstavlja ulaz vanjskog izvora napajanja, dok J2 predstavlja izlaz napajanja prema „BDM“ modulu. Paralelno na liniji J3 – J2 spojen je linearni naponski regulator oznake AMS1117 koji stabilizira i spušta ulazni napon na iznos od 5V. LED dioda D1 služi kao indikator prisutnosti vanjskog napona te ispravnog rada stabilizatora. Otpornik R4 ograničava struju LED diode. Budući da na USB priključku postoji 5V linija za napajanje, potrebno je istu izolirati od ostalog dijela napajanja. Potreba za tim javlja se u situaciji kad se koristi USB priključak, a moduli

se napajaju vanjskim izvorom. Kako ne bi došlo do strujnog preopterećenja jednog od napajanja, između USB 5V izvora i linearnog regulatora postavlja se brza Schottky dioda. Na taj način mogu se koristiti oba izvora, bez opasnosti od povratnih struja i preopterećenja. Kondenzator C5, koji je tantal elektrolitski, služi za stabilizaciju napona sa USB 5V linije. Na njega je paralelno spojen keramički blok kondenzator C6 koji propušta visokofrekventne smetnje iz linije prema negativnom polu napajanja – masi.

Na sljedećoj slici, slici 17., prikazan je USB na UART sklop. U oba modula ugrađen je identični sklop u svrhu upisivanja programskog koda te mogućnosti korištenja monitora serijske komunikacije na računalu.



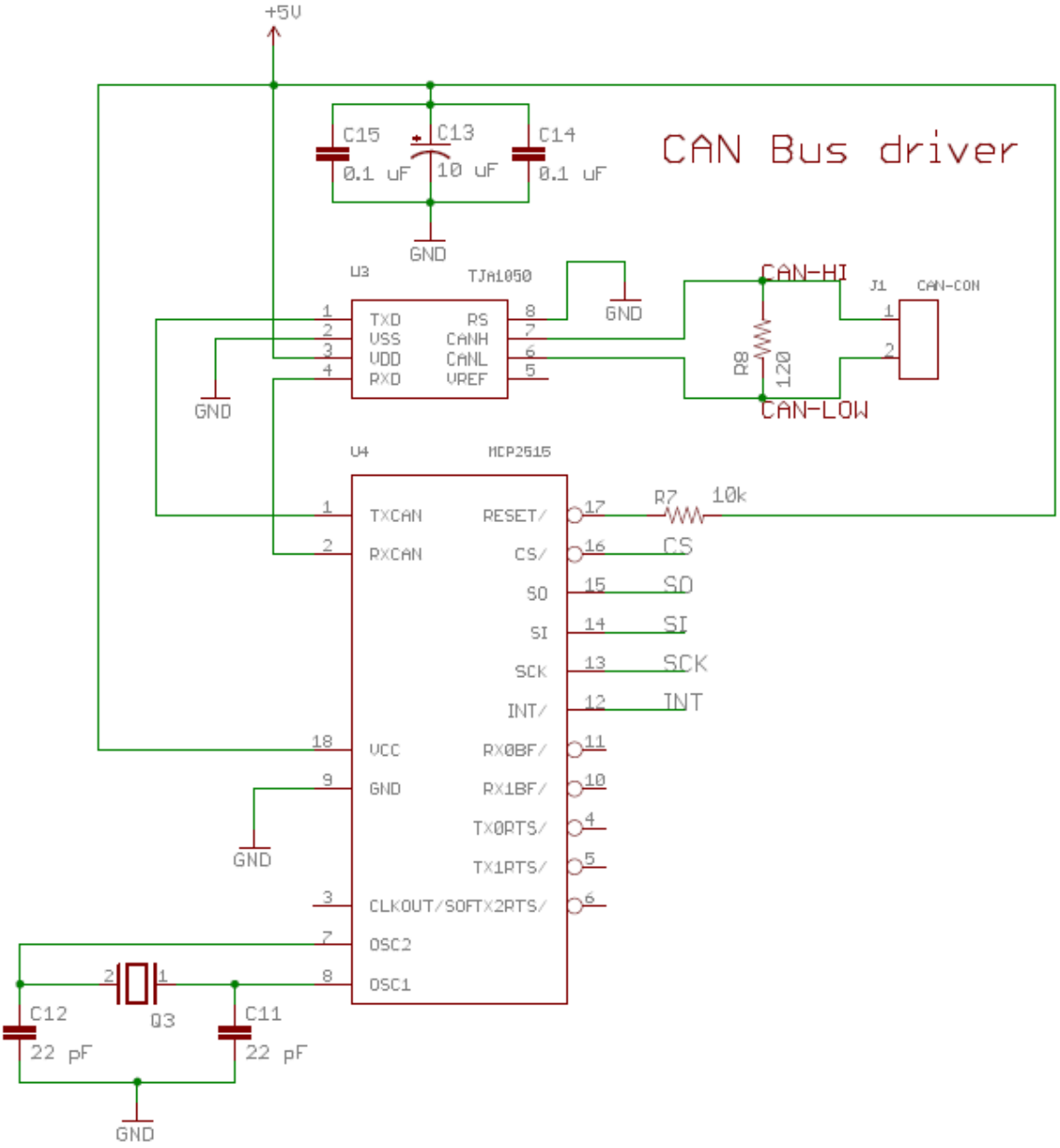
Slika 17. USB – UART sklop

(izvor: autor, kolovoz, 2015.)

Dizajn sklopa je preuzet iz literature [8], stranica 5. Kao glavni element sklopa korišten je integrirani krug CH340G USB na UART pretvarač. Uz osnovnu funkciju, pretvarač posjeduje ugrađen interni linearni stabilizator napona iznosa 3.3V (na nožici 4) pa je pogodan za korištenje kod obitelji mikroprocesora sa 3.3V napajanjem. U ovom primjeru se 3.3V ne koristi, pa je izvod spojen preko blok kondenzatora C10 na masu. Uz naponski izvor, pretvarač dakako posjeduje i nožicu za reset (ponovno pokretanje) mikroprocesora. Reset signal sa nožice broj 16 šalje se preko blok kondenzatora C9 na nožicu reset signala mikroprocesora. Reset signal je aktivan uvijek kada se pretvarač spoji na USB port računala te kada se s računala šalje programski kód prema mikroprocesoru. Uz reset signal prema mikroprocesoru se spajaju još dvije nožice sa pretvarača. Nožica broj 2 oznake Tx (eng. transmite) šalje podatkovne pakete prema mikroprocesoru, te je spojena na prijemnu nožicu mikroprocesora oznake Rx (eng. recive). Nožica broj 3 pretvarača, oznake Rx služi kao prijemna nožica za podatkovne pakete pristigle s mikroprocesora. Rx nožica pretvarača spojena je na Tx nožicu mikroprocesora.

Izvodi i pripadajuće nožice za komunikaciju prema računalu putem USB priključka su nožice broj 5 i 6. Nožica broj 5 je spojena na USB priključak oznake DATA +, a nožica broj 6 na USB priključak DATA –. Budući da je USB priključak četvero-polni, ostala dva priključka su 5V napajanje (spojeno prema sklopu napajanja) i masa. Pretvaraču je potreban vanjski kristal za rad unutarnjeg oscilatora. U izrađenom primjeru korišten je kristalni rezonator s već ugrađenim blok kondenzatorima, a kondenzatori C7 i C8 nalaze se u shemi zbog izrade tiskane pločice, odnosno druge varijante kućišta i kristala. Ukoliko se koristi kristalni oscilator bez ugrađenih blok kondenzatora, potrebno ih je ugraditi. Vrijednosti kristala i kondenzatora navedene su u literaturi [8], stranica 3, tablica 3.3, a iznose 22pF za kondenzator te 12 MHz za kristal.

Slika 18. prikazuje CAN sekciju koja se koristi na oba modula. Korišteni integrirani krugovi MCP2515 (CAN kontroler) i TJA1050 (brzi CAN primopredajnik) ne zahtijevaju puno vanjskih elemenata. Za CAN kontroler potrebno je odabrati vanjski kristalni oscilator. Odabir je vršen po proizvođačevim uputama iz literature [5], poglavlje 8.0 stranica 55 i 56. Izabran je kristal vrijednosti 8 MHz. Za tu vrijednost kristala, kapacitet kondenzatora (C11 i C12) iznosi 22pF po kondenzatoru. Osim kristalnog oscilatora potrebno je dodati blok kondenzatore (C14 i C15), te tantalni elektrolitski (C13) kondenzator kao filtre 5V napajanja.

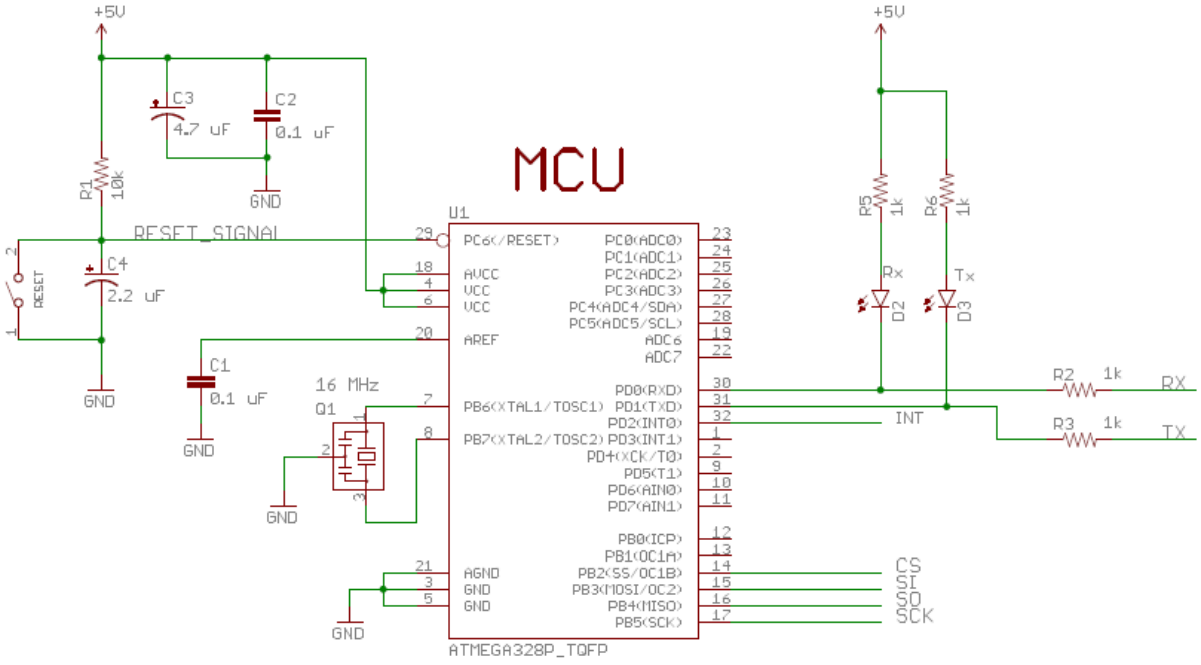


Slika 18. CAN sekcija

(izvor: autor, kolovoz, 2015.)

Nožice broj 1 i 2 CAN kontrolera spajaju se redom na nožice broj 1 i 4 CAN primopredajnika. One predstavljaju TTL komunikaciju po kojoj se odvija prijem i slanje poruke između CAN sabirnice i kontrolera. Nožice od broja 13 do 16 CAN kontrolera su izvodi standardne SPI veze (CS, SO, SI, SCK) kojom se odvija komunikacija s mikroprocesorom. Uz SPI vezu, prema mikroprocesoru je upućen interrupt sa nožice broj 12 (INT) CAN kontrolera. Nožice broj 6 i 7 primopredajnika spojene su na CAN sabirnicu. Prije sabirničkog priključka (CAN – CON) ugrađen je terminirajući otpornik R8, iznosa 120 Ω.

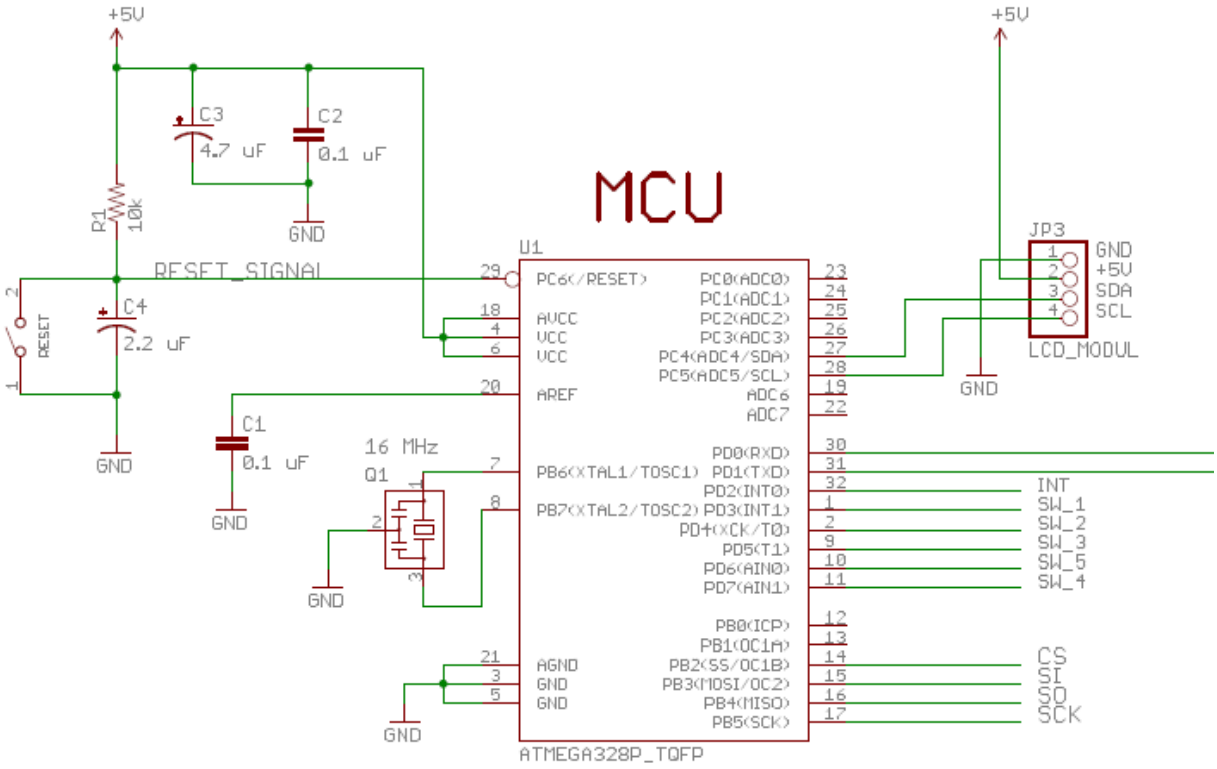
Osnovno okruženje, u kojem se nalazi mikroprocesor Atmega328P (slika 19.), za oba je modula isto. Osnovno okruženje odnosi se na ulazni naponski filtar (C2 i C3), reset tipkalo s RC spojem (R1 i C4), blok kondenzator referentnog ulaznog napona C1 te UART podatkovnu liniju, koja je priključena prema USB – UART modulu putem Tx i Rx linija. Na toj liniji ugrađene su signalne LED diode (D2, D3) i otpornici za ograničavanje struje dioda (R5 i R6). Također, mikroprocesori na oba modula imaju jednaki radni takt, definiran vanjskim kristalnim oscilatorom iznosa 16 MHz, te jednak spoj prema CAN sekciji. Odabrani iznos kristalnog oscilatora uvjetovan je pred-programskom podrškom (eng. bootloader) upisanom u mikroprocesor, a omogućuje programiranje istog u „Arduino IDE“ okruženju.



Slika 19. Osnovno okruženje mikroprocesora

(izvor: autor, kolovoz, 2015.)

U ostatku okruženja mikroprocesora razlike između „Dashboard“ i „BDM“ modula su u spojevima prema drugim elementima. Slika 20. prikazuje spoj mikroprocesora u „Dashboard“ modulu.



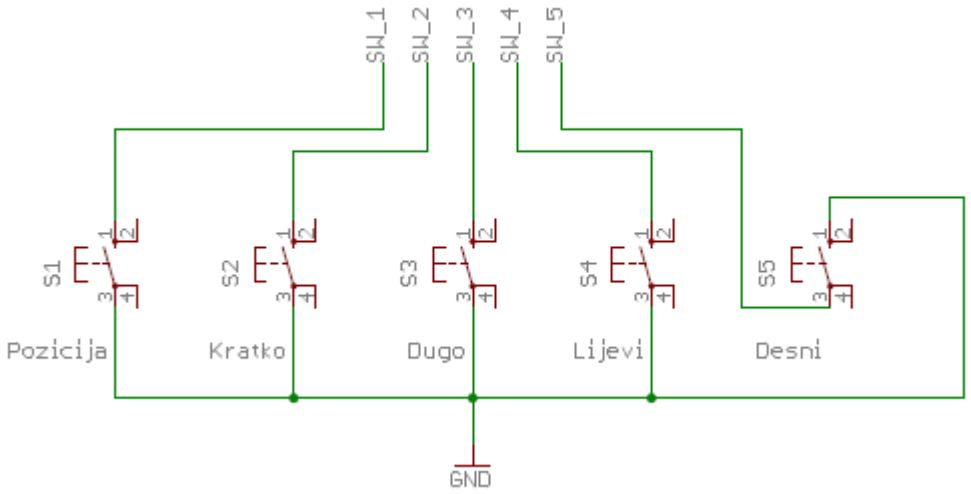
Slika 20. Spoj mikroprocesora u „Dasboard“ modulu

(izvor: autor, kolovoz, 2015.)

Sa slike 20. vidljivo je da su mikroprocesoru na „Dasboard“ modulu dodani spojevi prema tipkalima, kojima se upravlja svjetlima spojenim na „BDM“ modulu, te I^2C (eng. Inter-Integrated Circuit) spoj prema LCD zaslonu (JP3). I^2C veza prema LCD zaslonu koristi samo dva priključka mikroprocesora, stoga je kao povoljnija od paralelne veze (koja koristi šest ili više priključaka) prihvaćena u izradi modula. Naravno, to iziskuje da LCD zaslon posjeduje I^2C priključak.

Tipkala su spojena na port „D“ mikroprocesora (skup od osam digitalnih ulazno-izlaznih nožica mikroprocesora Atmega_328P) na način da tipkalo oznake SW_1 kontrolira stanje pozicionog svjetla i spojen je na PD3 dio porta, tipkalo oznake SW_2 kontrolira stanje kratkog svjetla i spojen je na PD4 dio porta, dok je tipkalo oznake SW_3 spojeno na PD5 dio porta i kontrolira stanje dugog svjetla. Za kontrolu stanja žmigavaca zadužena su tipkala SW_4 i SW_5 na način da je SW_4 spojeno

na PD7 dio porta, a SW_5 na PD6. SW_4 kontrolira stanje lijevog žmigavca, a SW_5 stanje desnog. Slika 21. prikazuje električni dio sheme koji se odnosi na tipkala.



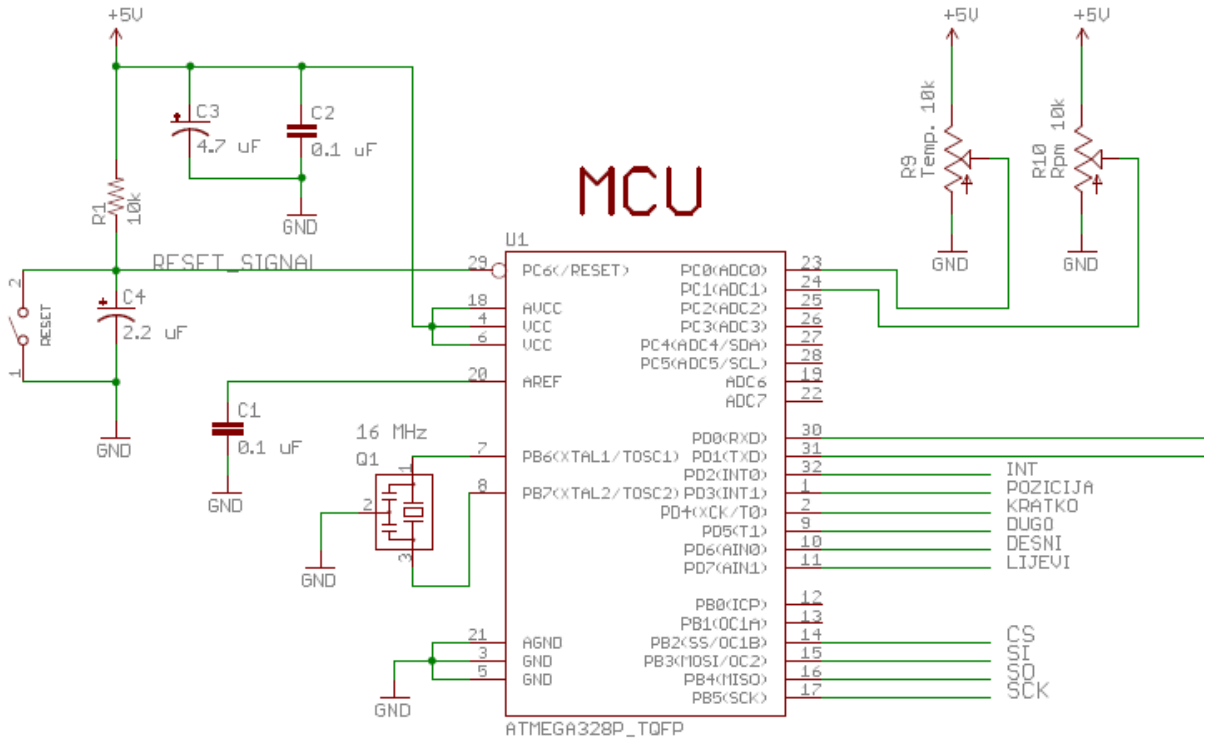
Slika 21. Tipkala „Dasboard“ modula

(izvor: autor, kolovoz, 2015.)

Sa slike 21. vidljivo je da pritiskom na tipkala ostvarujemo direktan spoj jedne od nožica porta „D“ mikroprocesora s masom. U ovakvom spoju ne može doći od strujnog preopterećenja tipkala niti mikroprocesora, iz razloga što je programskim putem, za vrijeme inicijalizacije mikroprocesora, port „D“ deklariran kao digitalni ulaz (visoka impedancija) te su postavljeni unutarnji pull-up otpornici za taj port.

Pull-up otpornicima digitalni portovi mikroprocesora spajaju se s pozitivnim dijelom napajanja (5V). Na mjestu spoja dijela digitalnog porta i pull-up otpornika (što predstavlja nožicu mikroprocesora) spojeno je tipkalo prema masi (0V). Za vrijeme kad tipkalo nije pritisnuto, mikroprocesor očitava logičku jedinicu na tom ulazu (5V preko pull-up otpornika). Suprotno tome, kad je tipkalo pritisnuto mikroprocesor očitava logičku nulu, jer je nožica preko tipkala spojena na masu, a pull-up otpornikom ograničava se struja. Vrijednost unutarnjih pull-up otpornika za mikroprocesor Atmega_328P iznosi 20 kΩ.

Slika 22. prikazuje spoj mikroprocesora „BDM“ modula. U ovom modulu, na iste brojeve nožica mikroprocesora na koje su u „Dashboard“ modulu spojena tipkala, u „BDM“ modulu su spojene signalne LED diode (pozicija, kratko i dugo svjetlo te žmigavci). Razlika je i u tome što na „BDM“ modulu ne postoji LCD zaslon, no dodana su dva naponska djelila koja simuliraju vrijednosti temperature i radnih okretaja motora. Naponsko djelilo izvedeno je promjenjivim otpornikom R9 za temperaturu, a R10 za radne okretaje. Ulazne i izlazne nožice otpornih staza promjenjivih otpornika spojene su između pozitivnog dijela napajanja (5V) i mase. Klizači promjenjivih otpornika spojeni su na analogne ulaze mikroprocesora (ADC0 za temperaturu i ADC1 za radne okretaje).

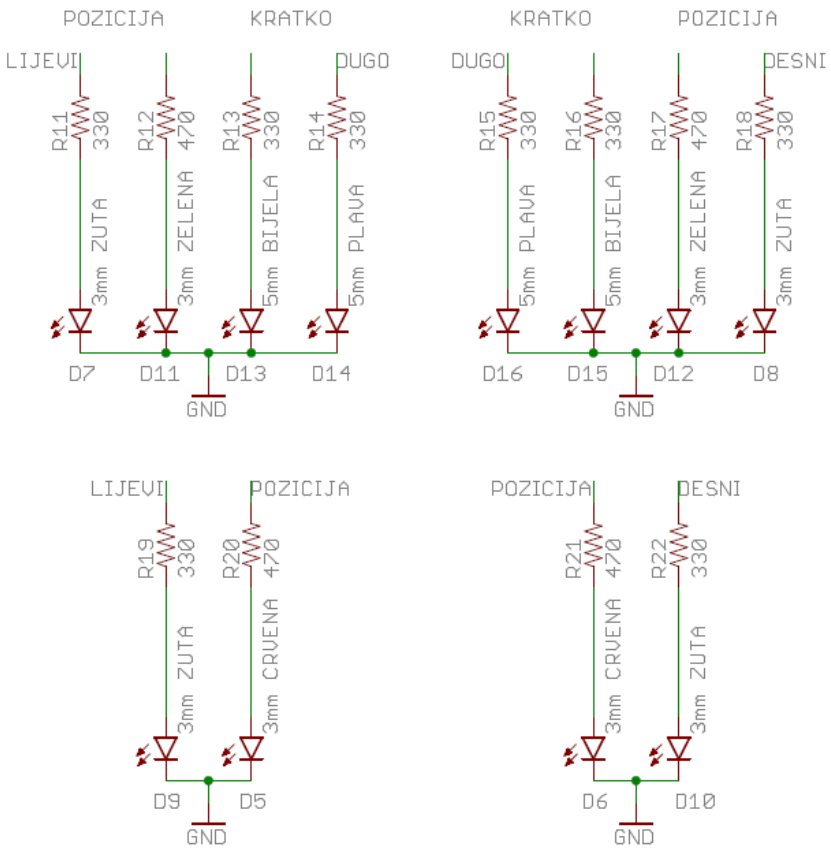


Slika 22. Spoj mikroprocesora u „BDM“ modulu

(izvor: autor, kolovoz, 2015.)

Budući da se LED diode ugrađuju u maketu sportskog automobila određen je potreban broj istih. Za poziciono svjetlo potrebne su četiri LED diode, od kojih će dvije biti ugrađene na maketu sprijeda, a dvije straga. Za kratko, odnosno dugo svjetlo, potrebne su četiri LED diode i sve se ugrađuju sprijeda na maketu. Za žmigavce su korištene po dvije LED diode za svaku stranu, te se ugrađuju bočno na maketu.

Programski je mikroprocesoru „BDM“ modula definirano da port „D“ predstavlja izlaze. Stanja svjetala određuju se postavljanjem logične jedinice (5V) za stanje uključeno, odnosno logičke nule (0V) za stanje isključeno. Budući da mikroprocesor za vrijeme uključenog stanja napaja LED diode, potrebno je na iste postaviti otpornike u svrhu ograničenja maksimalne struje. Slika 23. prikazuje električne spojeve LED dioda „BDM“ modula s otpornicima.

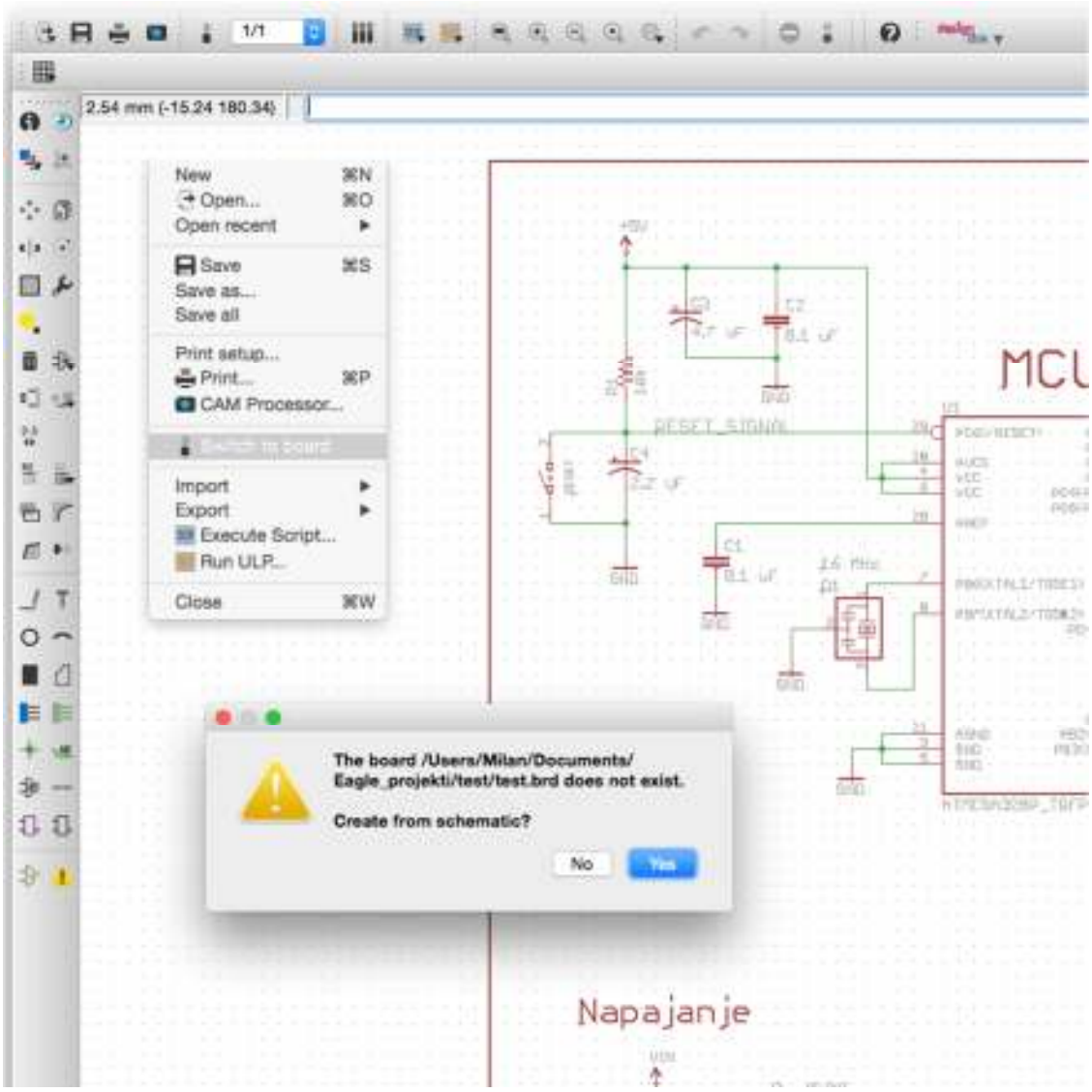


Slika 23. Električni spoj LED dioda i otpornika „BDM“ modula

(izvor: autor, kolovoz, 2015.)

3.3. Izrada tiskanih pločica

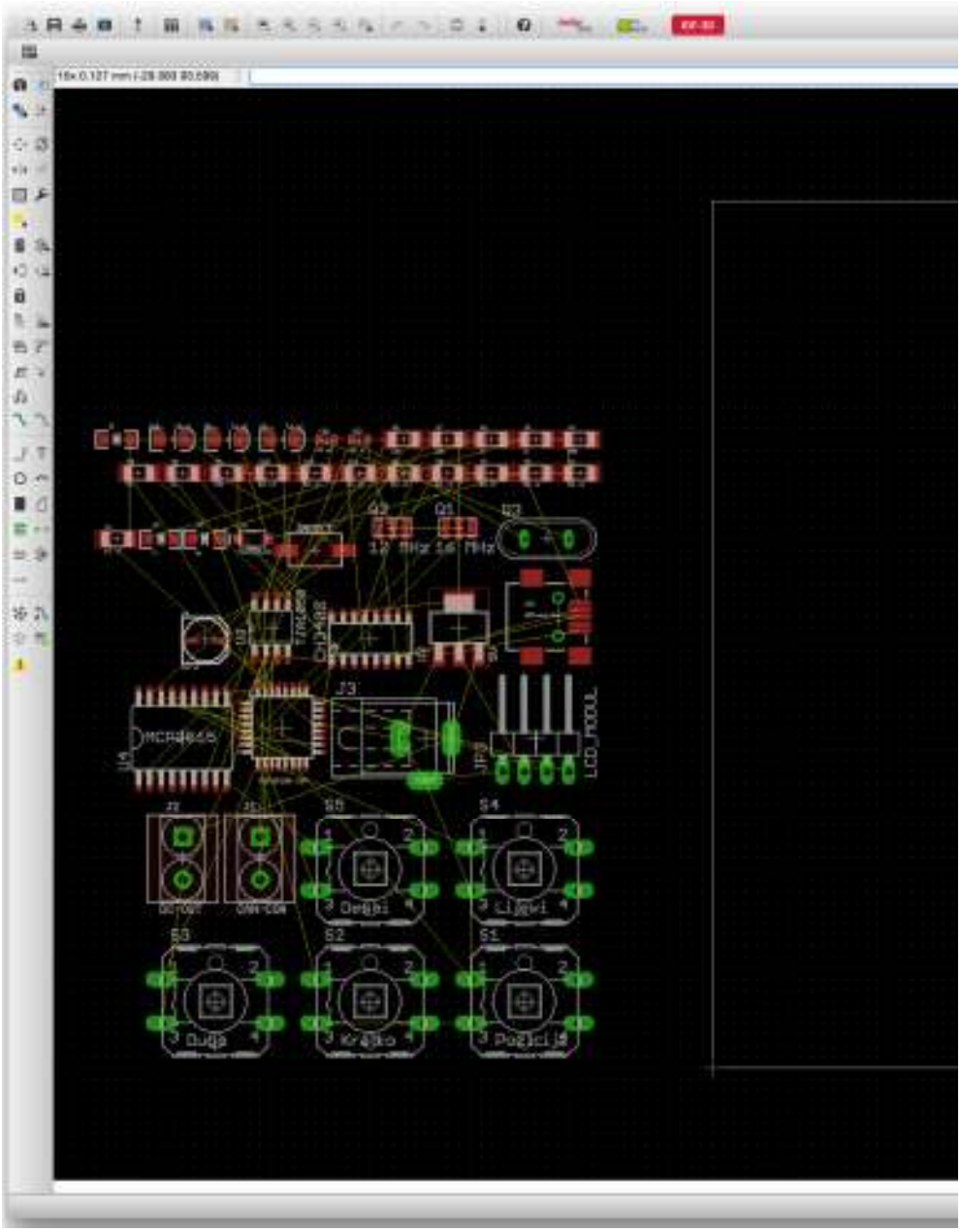
Po završetku stvaranja električne sheme, započinje stvaranje izgleda dvostrane tiskane pločice. Iz aplikacije „Schematic“ prelazimo u aplikaciju „Board editor“. Prijelaz se vrši odabirom stavke „Switch to board“ iz menija „File“ u „Schematic“ aplikaciji (slika 24. lijevo gore). Aplikacija prepoznaje da za navedenu električnu schemu ne postoji pridružena datoteka tiskane pločice te ponudi stvaranje iste. Ukoliko potvrdno odgovorimo (slika 24. pri dnu) automatski se poziva aplikacija „Board editor“, kreira datoteka te dodaju svi korišteni elementi iz električne sheme. Logika aplikacija uvelike olakšava rad, jer preskačemo korak koji smo u aplikaciji „Schematic“ već odradili – ručno dodavanje komponenti iz biblioteka.



Slika 24. Prijelaz iz „Schematic“ aplikacije u aplikaciju „Board editor“

(izvor: autor, kolovoz, 2015.)

Slika 25. prikazuje radno okruženje „Board editor“ aplikacije sa svim automatski dodanim elementima.

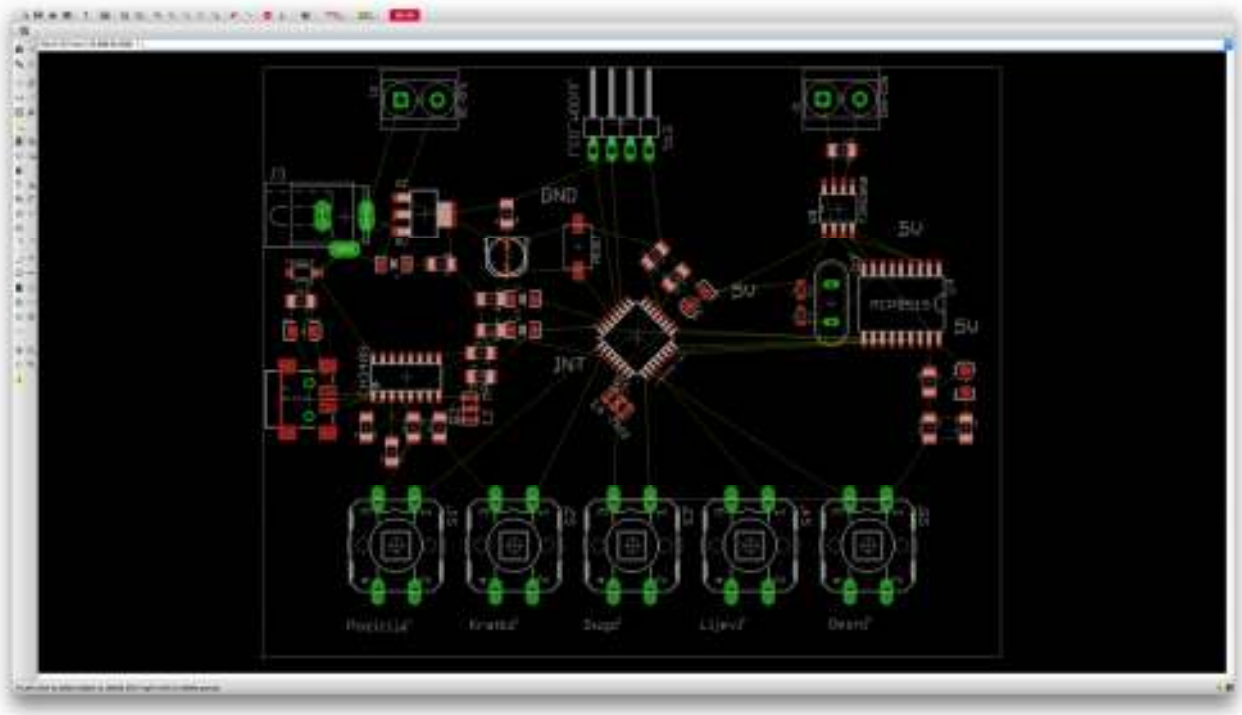


Slika 25. Radno okruženje aplikacije „Board editor“

(izvor: autor, kolovoz, 2015.)

Sljedeći korak je raspoređivanje elektroničkih elemenata po radnoj površini. Raspored se vrši s obzirom na međusobne spojeve elemenata. Potrebno je pozicionirati elemente na način da razmak između spojeva elemenata bude čim manji, tj. da elementi budu što je moguće bliže jedan drugome na tiskanoj pločici. Pri tome svakako treba uzeti u obzir ostale vrlo bitne varijable kao što su interferencija među elementima, radijacija topline i slično. Ukratko, elementi moraju biti

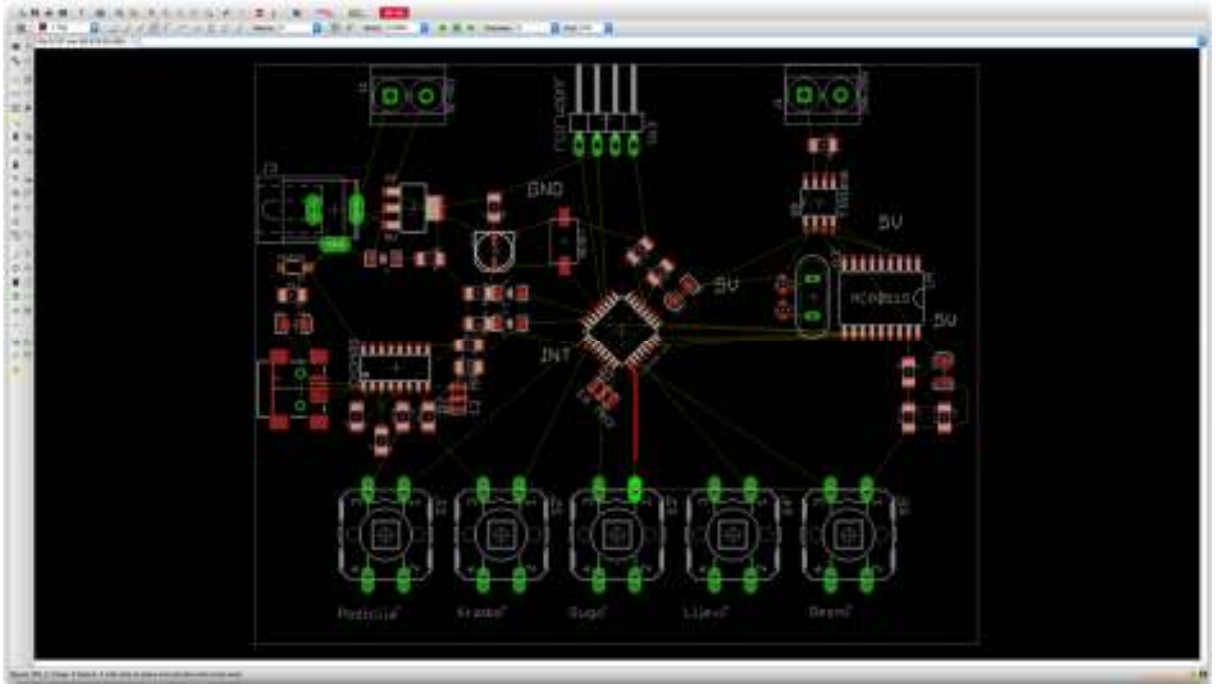
pozicionirani strateški, da ne smetaju jedni drugima u radu, a s druge strane, da tiskana pločica ne bude glomazna te da ima konstrukcijskog i dizajnerskog smisla. Slika 26. prikazuje izgled tiskane pločice s raspoređenim elementima.



Slika 26. Tiskana pločica s raspoređenim elementima

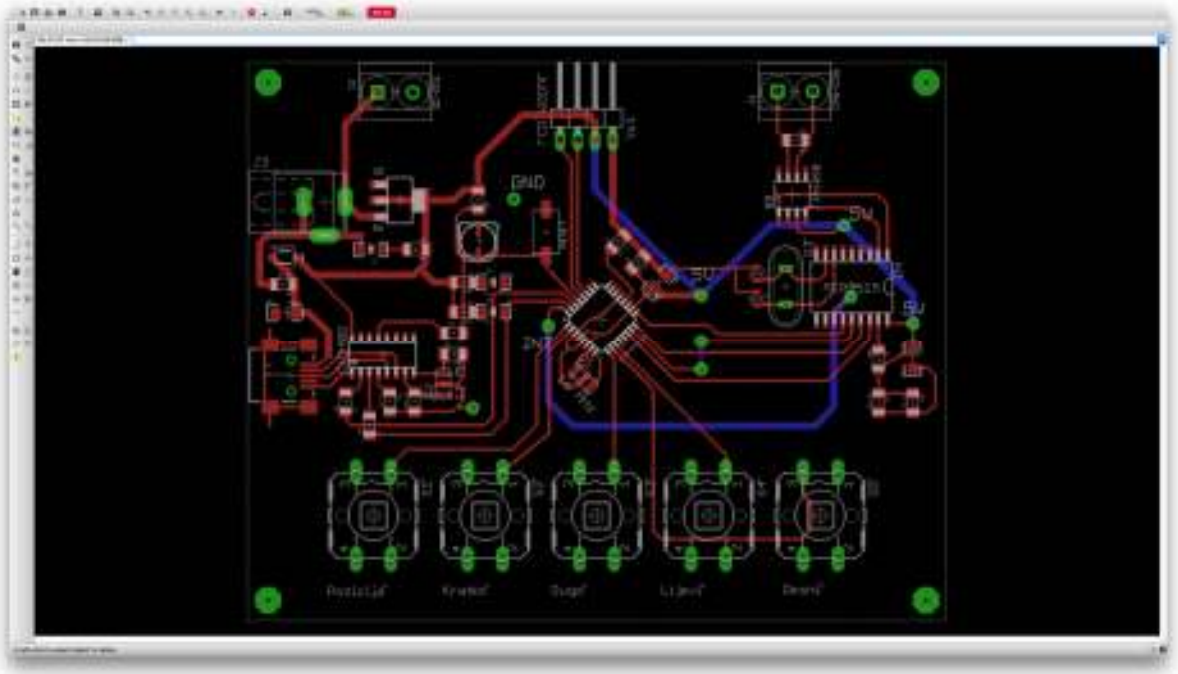
(izvor: autor, kolovoz, 2015.)

Po završetku raspoređivanja elemenata pristupa se rutiranju (izradi električnih vodova među elementima). Rutiranje započinjemo izborom alata „Route“ iz alatnog izbornika aplikacije. Nakon odabira alata, u gornjem lijevom kutu aplikacije prikazuje se meni s odabirom površine na kojoj će se obavljati rutiranje. Osnovne površine su gornja – naziva „Top“ i donja – naziva „Bottom“. „Top“ površina označena je crvenom bojom, a „Bottom“ plavom. Budući da su elementi modula u SMD izvedbi započinje se rutiranje na „Top“ površini. Slika 27. prikazuje rutiranje između mikroprocesora i tipkala, a slika 28. izgled rutirane (završene) pločice. Vodovi koji se ne mogu ostvariti na gornjoj površini, iz razloga što prelaze jedan preko drugoga, ostvaruju se na donjoj površini.



Slika 27. Rutiranje između mikroprocesora i tipkala

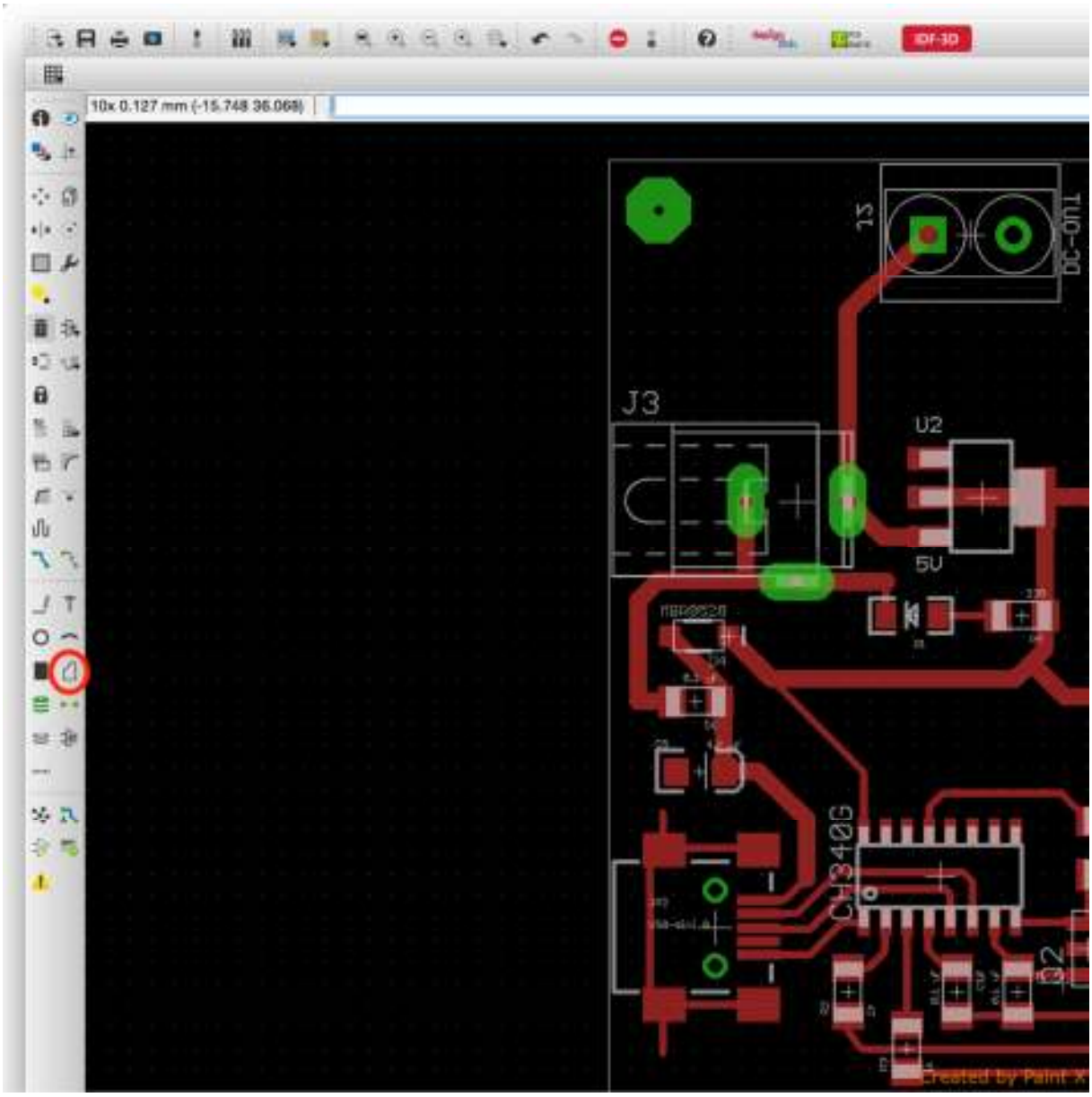
(izvor: autor, kolovoz, 2015.)



Slika 28. Izgled rutirane tiskane pločice

(izvor: autor, kolovoz, 2015.)

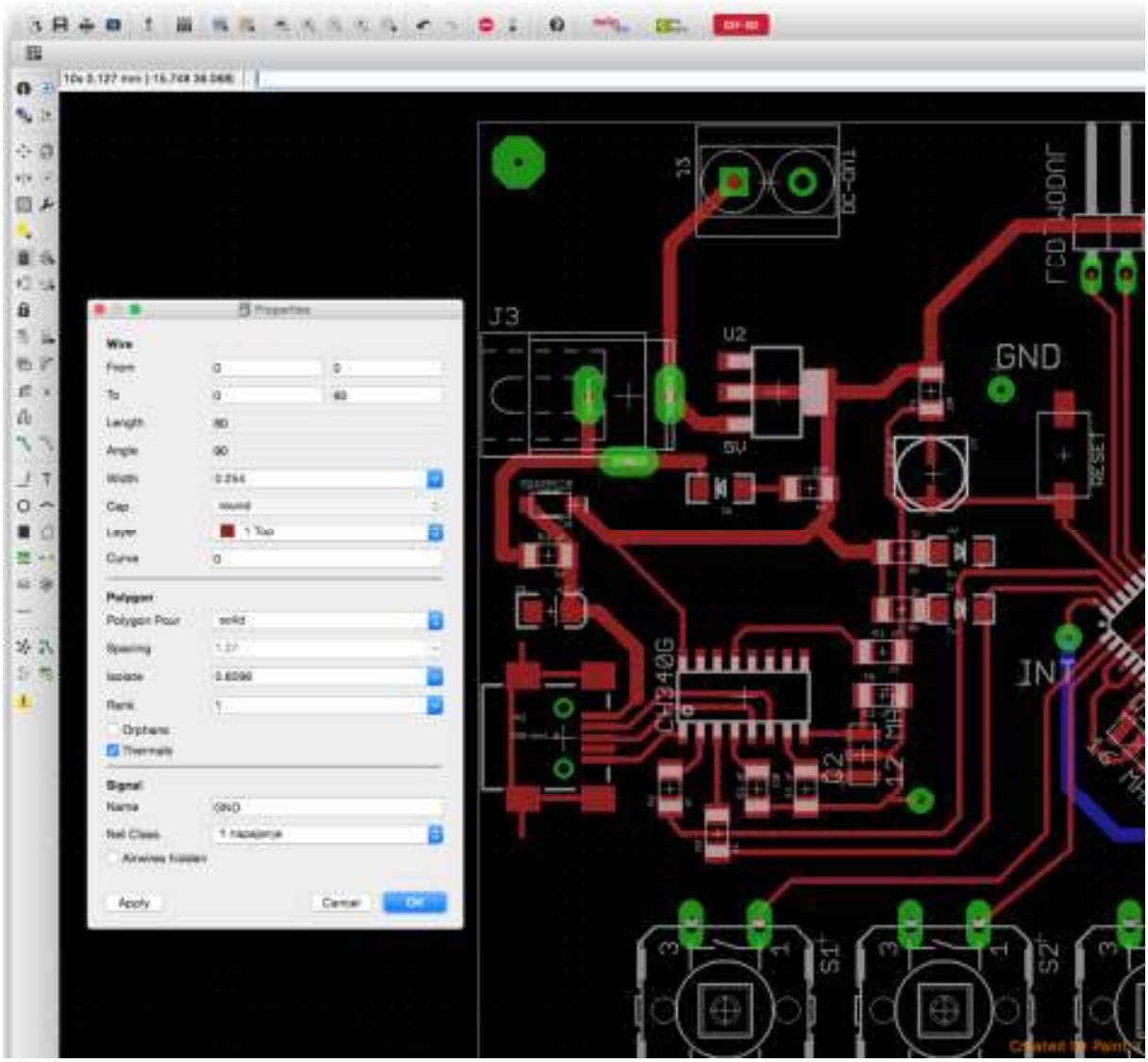
Nakon rutiranja tiskanoj pločici dodajemo masku („Copper pour“). Masku započinjemo odabirom alata „Poligon“ iz alatnog menija. Na slici 29. crvenim kružićem označen je alat „Poligon“.



Slika 29. Odabir alata „Poligon“

(izvor: autor, kolovoz, 2015.)

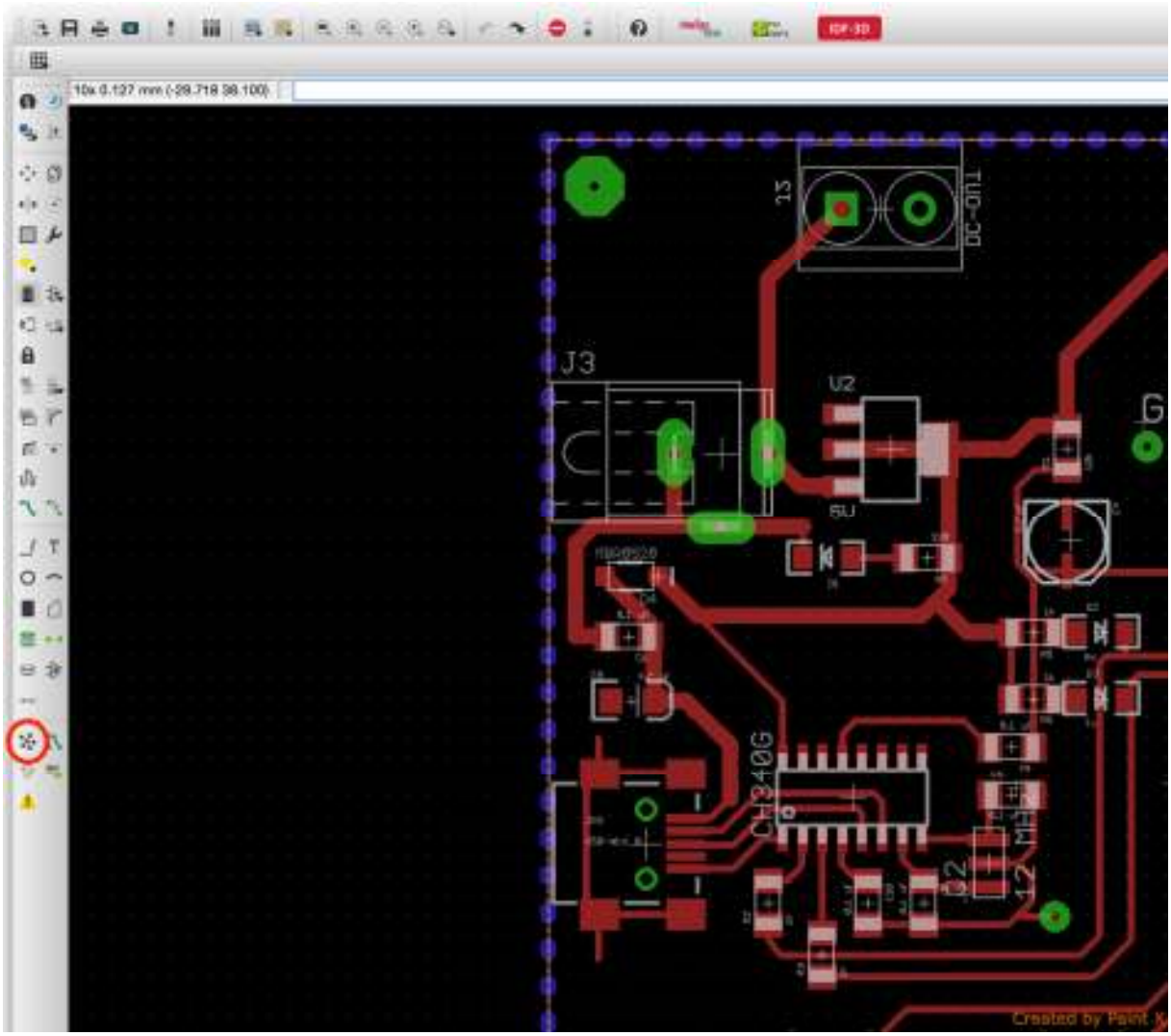
Tim alatom potrebno je nacrtati okvir oko cijele pločice. Desnim klikom miša na nacrtani okvir, odabiremo svojstva okvira te se u tu svrhu otvara izbornik prikazan na slici 30.



Slika 30. Izbornik sa svojstvima okvira

(izvor: autor, kolovoz, 2015.)

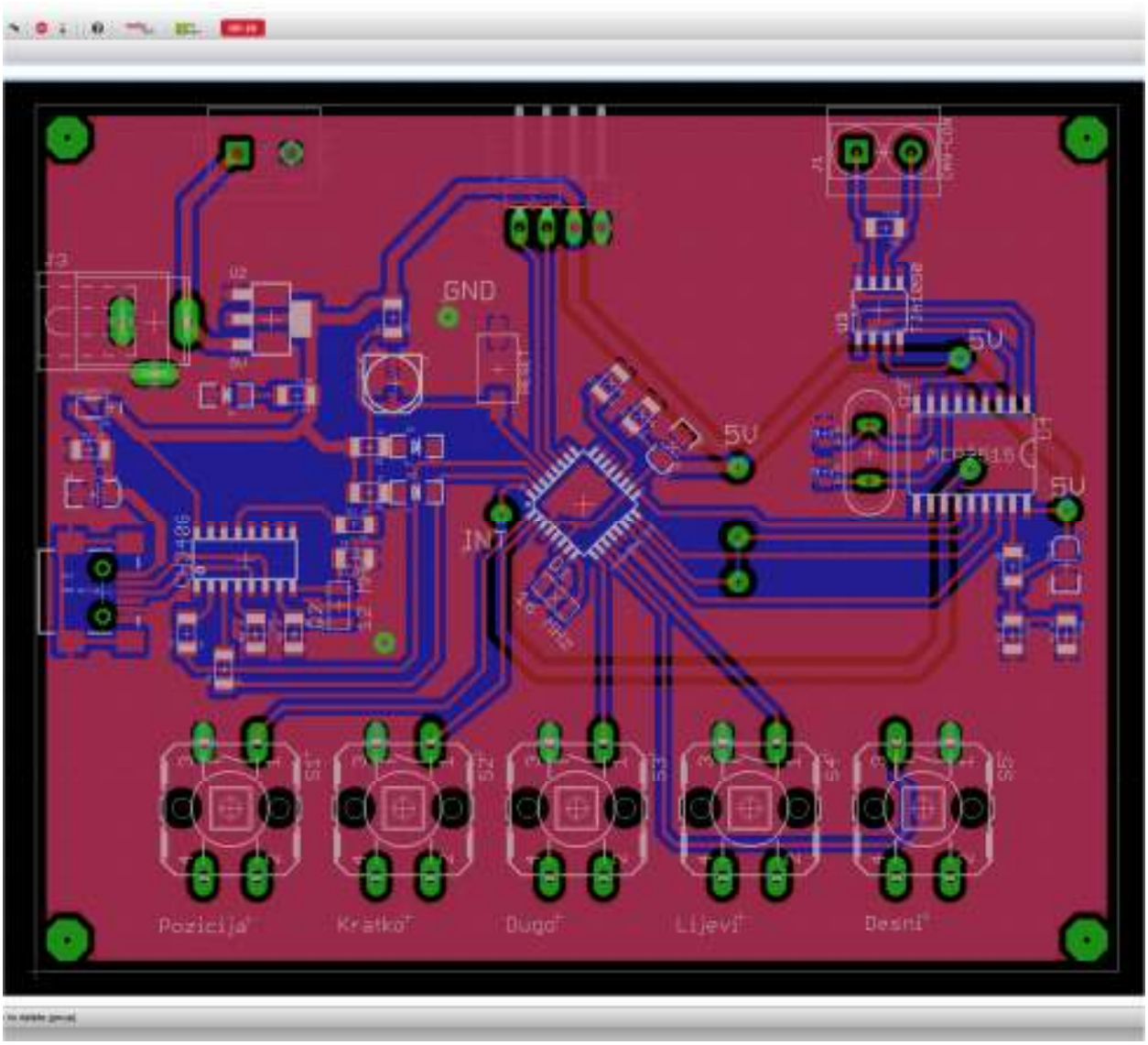
U izborniku, nacrtanom okviru dodjeljujemo površinu „Top“ te u odjeljku menija „Signal“ (pri dnu) dodjeljujemo signal „GND“ što predstavlja negativni pol (masu) izvora napajanja. Istu proceduru ponovimo, započevši od odabira alata „Poligon“, zatim crtanja okvira, osim što u svojstvima odabiremo površinu „Bottom“. Na taj način stvoriti će se poligoni za obje površine. Nakon završetka odabiremo alat „Ratsnest“ prikazan na slici 31. u crvenom kružiću.



Slika 31. Odabir alata „Ratsnest“

(izvor: autor, kolovoz, 2015.)

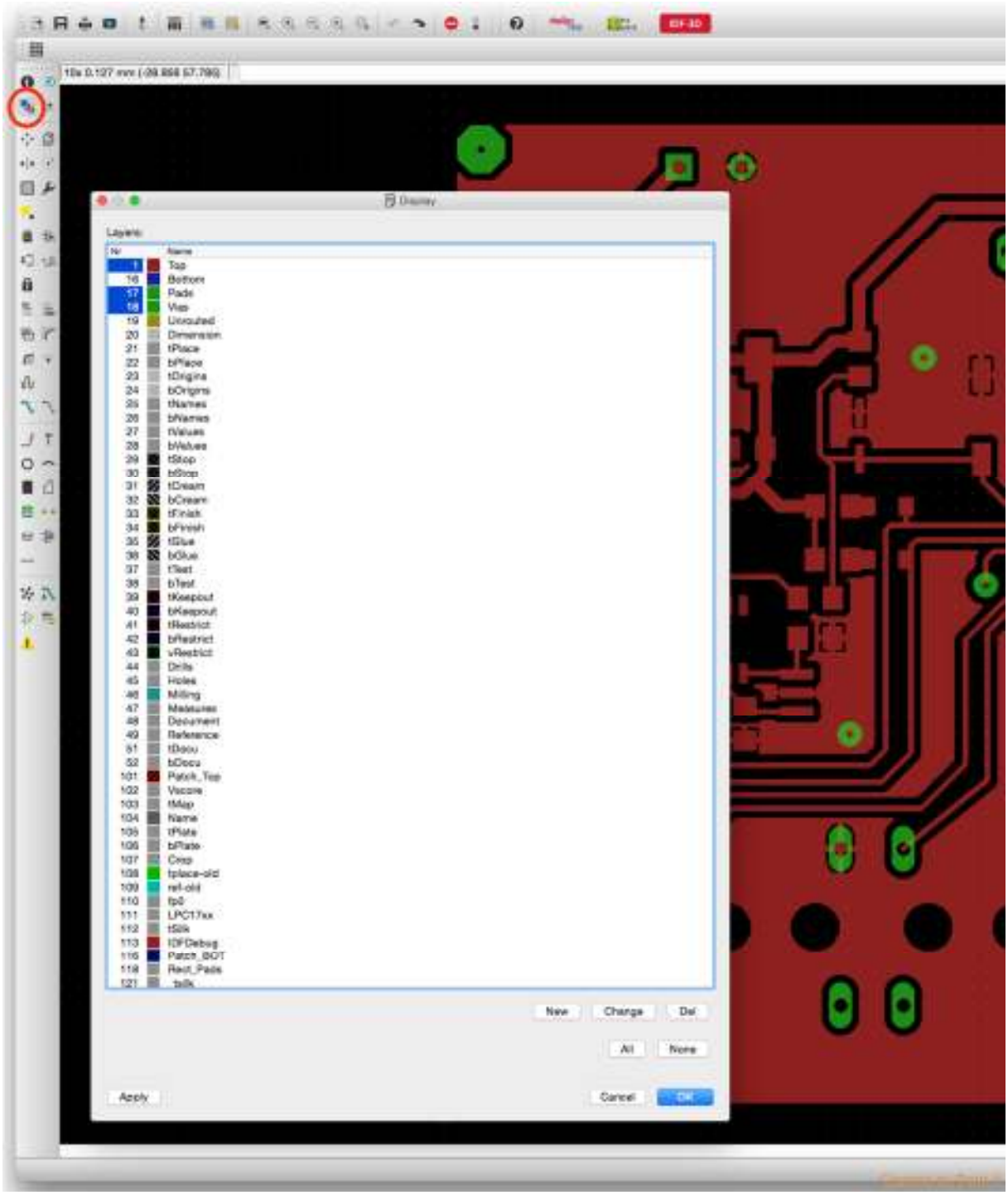
Odabirom alata „Ratsnest“ aplikacija automatski generira, na temelju nacrtanog poligona i pridruženih svojstava, punu površinu. Ta će površina, nakon razvijanja tiskanih pločica, ostati kao oblik bakrenog sloja kojim se popunjava prazan prostor među elementima. Budući da smo u svojstvima dodijelili signal GND, oblik će se automatski spojiti sa svim točkama koje su na GND signalu. Na taj način stvaramo štit interferencijama među elementima te unaprjeđujemo izgled gotove pločice. Slika 32. prikazuje izgled tiskane pločice nakon upotrebe alata „Ratsnest“.



Slika 32. Izgled tiskane pločice nakon upotrebe alata „Ratsnest“

(izvor: autor, kolovoz, 2015.)

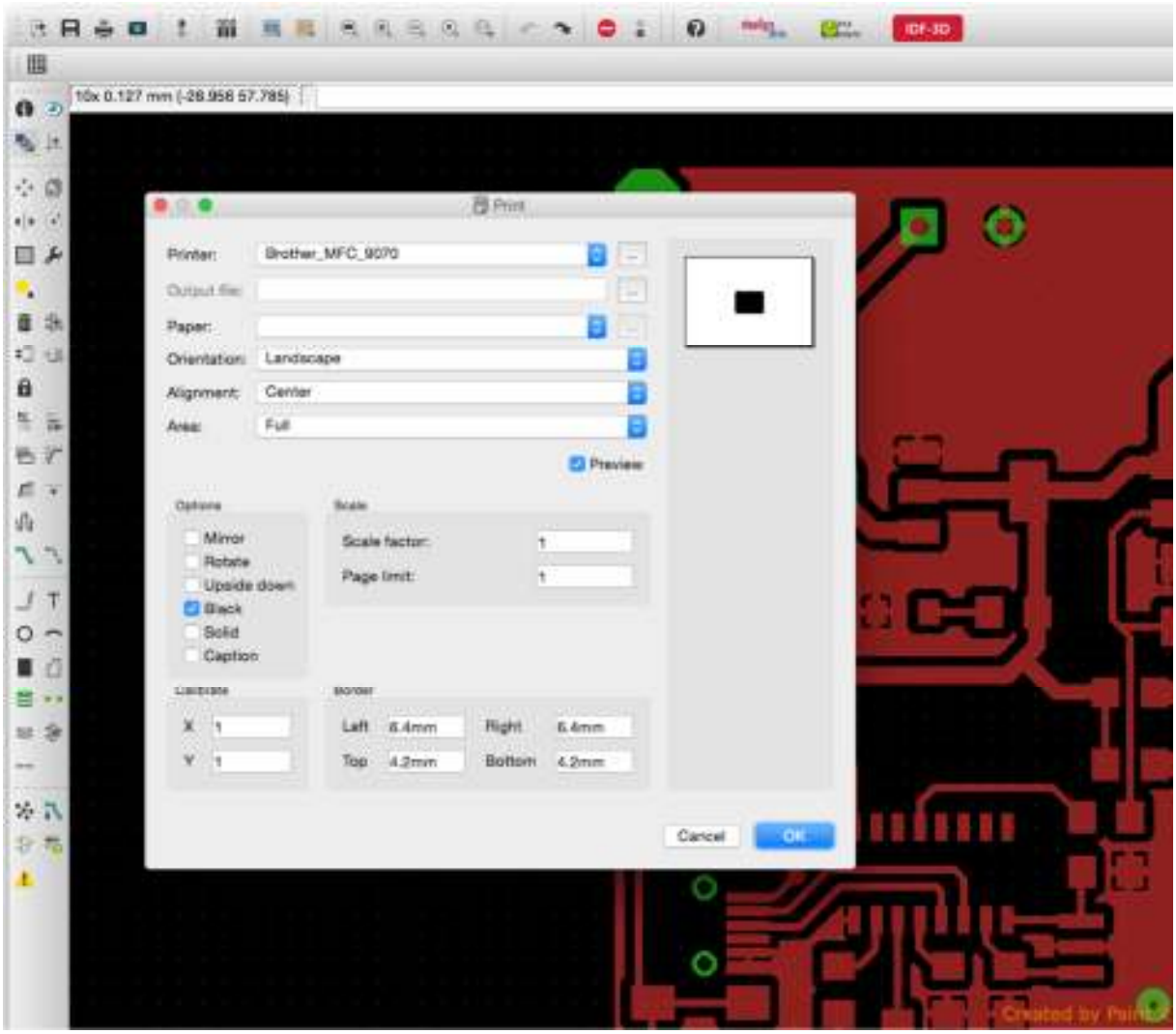
Dovršeni dizajn tiskane pločice ispisujemo na transparentnu (grafo) foliju koja služi kao film pri razvijanju iste. Ispis se vrši na način da se prvo prikažu samo slojevi „Top“ ili „Bottom“ na radnoj površini aplikacije. Za odabir prikaza koristi se alat „Layer“ iz alatne trake. Slika 33. prikazuje, u crvenom kružiću, izgled alata „Layer“.



Slika 33. Izgled alata „Layer“

(izvor: autor, kolovoz, 2015.)

Pritiskom lijeve tipke miša na alat „Layer“ otvara se izbornik (na slici 33. u sredini) iz kojeg odabiremo slojeve za ispis. Potrebno je izabrati slojeve: „Top“, „Pads“ i „Vias“ te odabir potvrditi pritiskom na dugme „OK“. Nakon odabira iz menija „File“ aplikacije „Board editor“ odabiremo stavku „Print“ koja pokreće izbornik iste. Slika 34. prikazuje izbornik stavke „Print“.

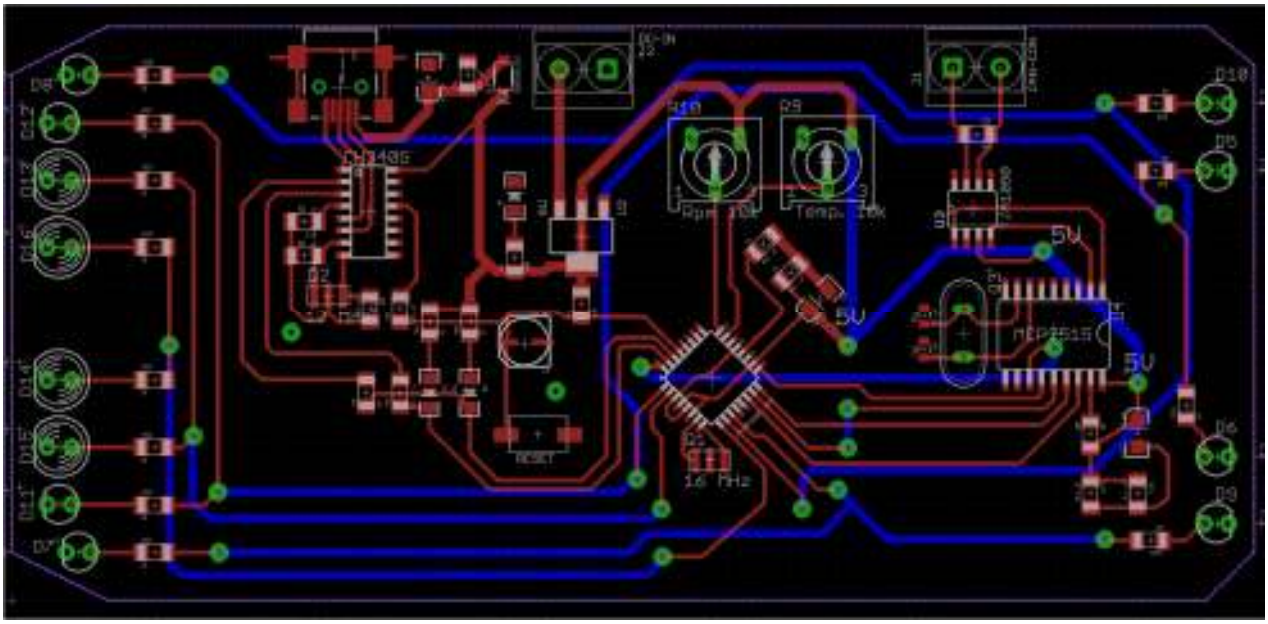


Slika 34. Izgled izbornika stavke „Print“

(izvor: autor, kolovoz, 2015.)

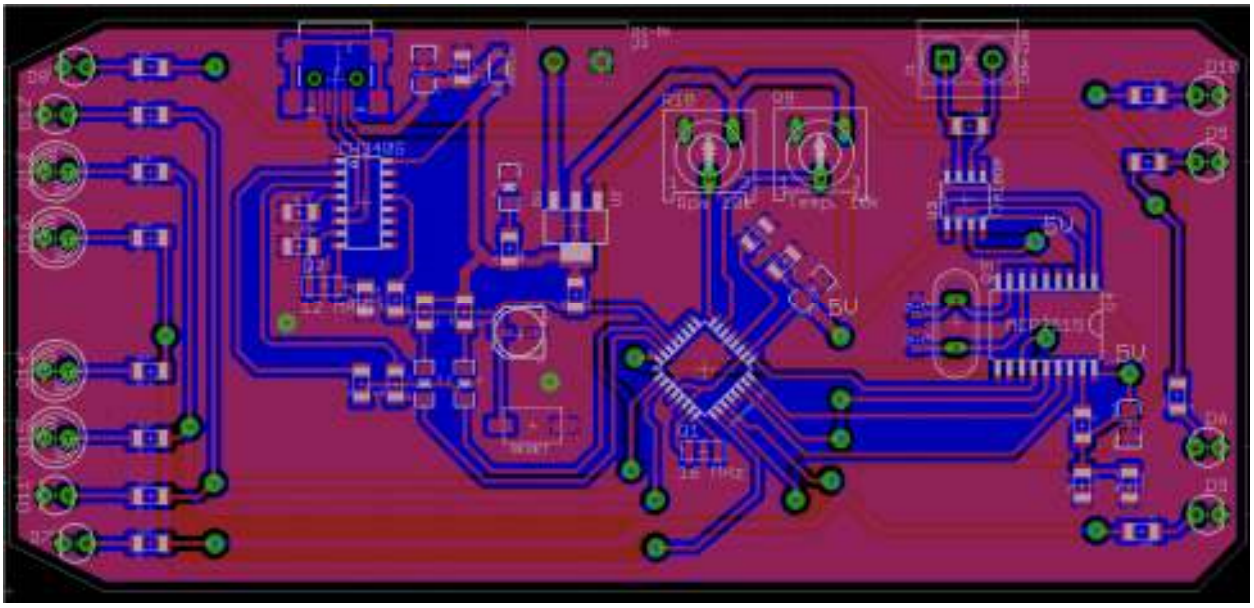
U navedenom izborniku izabiremo pisač te stavljamo kvačicu (oznaku) u polje „Black“. Polje se nalazi na lijevoj strani izbornika, pri dnu. Stavljanjem kvačice u to polje osiguravamo da ispis bude u crnoj boji, što je važno kod razvijanja tiskane pločice.

Budući da je u opisanom primjeru prikazana izrada tiskane pločice „Dasboard“ modula, potrebno je sve navedene postupke (počevši od prijelaza iz aplikacije „Schematic“ u aplikaciju „Board editor“) ponoviti pri izradi druge tiskane pločice modula „BDM“. Na slici 35. prikazan je kraj koraka rutanja izrade tiskane pločice „BDM“ modula, a slika 36. prikazuje istu s maskom.



Slika 35. Izgled tiskane pločice „BDM“ nakon rutanja

(izvor: autor, kolovoz, 2015.)



Slika 36. Izgled maske tiskane pločice „BDM“

(izvor: autor, kolovoz, 2015.)

Slika 37. prikazuje ispisane filmove na transparentnoj (grafo) foliji te materijal za izradu tiskanih pločica, *Vetronit FE2*. Foto osjetljivi lak tvornički je nanesen na bakreni sloj materijala, s toga je materijal zaštićen od svjetlosti neprozirnom folijom.



Slika 37. Film za razvijanje tiskanih pločica i materijal za njihovu izradu

(izvor: autor, kolovoz, 2015.)

Materijal je potrebno odrezati u oblik kakav će imati gotove tiskane pločice. Za tu svrhu korištena je ručna pila. Prikaz obrađenog materijala je na slici 38.



Slika 38. Obrađeni materijal za izradu tiskanih pločica

(izvor: autor, kolovoz, 2015.)

Nakon obrade materijala pristupa se osvjetljavanju foto osjetljivog laka na materijalu. Foto lak se osvjetljuje UV (ultraljubičastim) svjetlom. Slika 39. prikazuje postavljanje materijala i filma u uređaj za osvjetljavanje.



Slika 39. Postavljanje materijala i filma u uređaj za osvjetljavanje

(izvor: autor, kolovoz, 2015.)

Na slici 39. vidljiv je izvor UV svijetla iz uređaja. Film i materijal, kojem je prethodno uklonjena zaštitna folija, postavljaju se na staklo uređaja, iznad izvora svjetla. Nakon poravnavanja materijala s filmom, zatvara se poklopac uređaja te pritiskom na tipkala (na slici 39. desno, pri dnu) pokreće proces osvjetljavanja. Uređaj će pomoću ugrađene vakumske pumpe izvući zrak iz prostora između poklopca i stakla. Vakumiranje prostora je potrebno da bi film što bolje prianjao na materijal, te da bi se uklonile neželjene sjene. Materijal je potrebno osvjetljavati točno petnaest minuta. Ukoliko se prekorači vremenski rok oštetit će se lak na materijalu, dok suprotno, ako je vremenski rok kraći, lak se neće dobro osvijetliti, a s time kasnije niti pravilno razviti. Za razvijanje foto laka koristi se lužina, otopina natrij – hidroksida (kemijska oznaka NaOH) u vodi. Omjer natrij – hidroksida i vode iznosi 7 do 9 grama

na 1 litru vode. Po završetku osvjetljavanja materijala isti je potrebno započeti razvijati u otopini čim prije. Nepoželjno je držati materijal blizu bilo kakvog izvora svjetlosti budući da je foto lak još uvijek aktivan.

Po završetku razvijanja materijal se ispere vodom te posuši. U procesu razvijanja, UV svjetlost promijenila je kemijsku strukturu laka te je osvijetljeni lak odstranjen lužinom. Neosvijetljeni lak, zaštićen filmom, otporan je na lužinu što za efekt ima foto preslikavanje izgleda filma na materijal. Opisanu proceduru osvjetljavanja i razvijanja potrebno je ponoviti za obje strane tiskanih pločica („Top“ i „Bottom“) te obje tiskane pločice (modul „Dashboard“ i modul „BDM“).

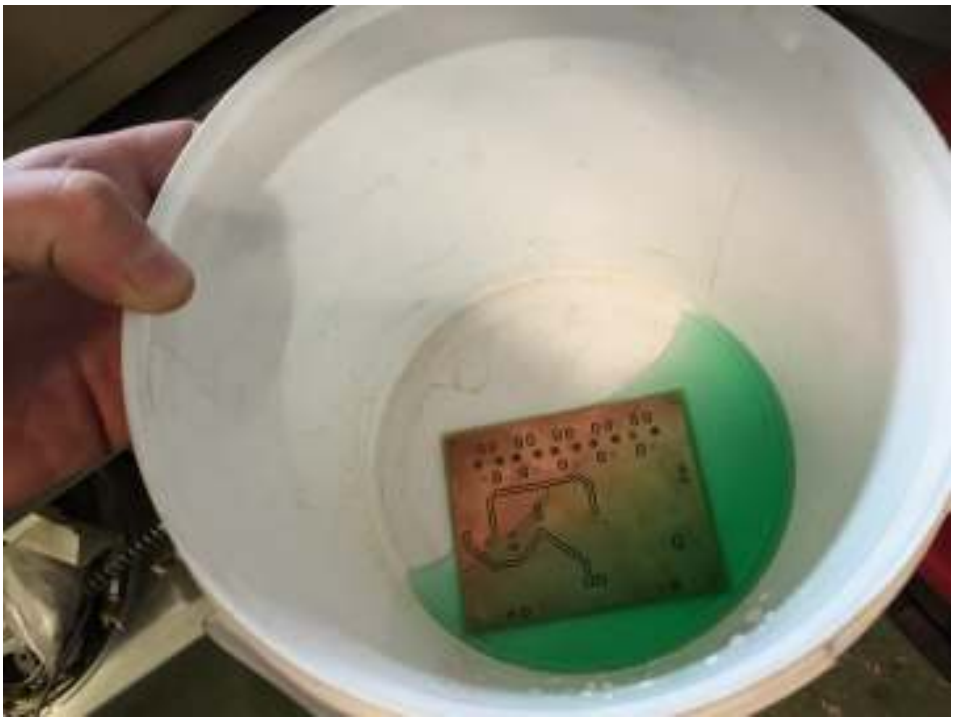
U staklenoj ili plastičnoj posudi, dovoljno velikoj da materijal stane u nju, priprema se mješavina za jetkanje. Jetkanje je kemijsko razlaganje bakrenog sloja s materijala. Mješavina je sastavljena od 19 % klorovodične kiseline (kemijska oznaka HCl) i 30% vodikovog peroksida (kemijska oznaka H_2O_2) u omjeru 100 na prema 1. Svugdje gdje je foto lak uklonjen lužinom iz procesa razvijanja, kiselina će nagristi bakreni sloj te ga u potpunosti ukloniti s materijala, dok će onaj lak koji je ostao na materijalu štititi bakreni sloj od nagrizanja, što za efekt ima stvaranje bakrenog sloja prema izgledu laka. Na slici 40. prikazane su kemikalije korištene u oba procesa (razvijanje i jetkanje).



Slika 40. Kemikalije korištene u procesima razvijanja i jetkanja materijala

(izvor: autor, kolovoz, 2015.)

Po završetku jetkanja, kad se vizualnim pregledom ustanovi da je sav nepoželjan bakreni sloj uklonjen, materijal se obilno ispere vodom (zaustavljanje kemijske reakcije) te posuši. Slike 41. i 42. prikazuju izgled materijala za izradu modula pri kraju procesa jetkanja.



Slika 41. Materijal „Dashboard“ modula pri kraju procesa jetkanja

(izvor: autor, kolovoz, 2015.)



Slika 42. Materijal „BDM“ modula pri kraju procesa jetkanja

(izvor: autor, kolovoz, 2015.)

Na slici 43. prikazan je krajnji rezultat, odnosno, izrađene tiskane pločice oba modula.



Slika 43. Izrađene tiskane pločice oba modula

(izvor: autor, kolovoz, 2015.)

Iako su tiskane pločice izrađene za SMD tehnologiju elektroničkih elemenata, postoji nekoliko komponenti na oba modula (tipkala na „Dashboard“ modulu, priključnice i kristalni oscilator CAN kontrolera oba modula te LED diode „BDM“ modula) za koje treba izbušiti rupe kroz tiskane pločice. Bušenje je izvršeno stolnom bušilicom proizvođača PROXXON, model IBS/E. Rupe koje se buše promjera su 1 mm za tipkala i LED diode, 1.2 mm za priključnice te 0.8 mm za kristalni oscilator. Slika 44. prikazuje bušenje rupa.



Slika 44. Bušenje rupa kroz tiskane pločice modula

(izvor: autor, kolovoz, 2015.)

Nakon završetka bušenja rupa pristupa se lemljenju elektroničkih elemenata. Lemljenje je izvršeno pomoću električne lemilice proizvođača ERSA model MS 8000. Za potrebe lemljenja SMD izvedbe elektroničkih elemenata korišten je najfiniji vrh lemilice (debljina 1 mm). Materijal za lemljenje je u obliku žice, kemijskog sastava Sn 60% Pb 40%. Slika 45. prikazuje proces lemljenja elemenata.



Slika 45. Proces lemljenja elektroničkih elemenata

(izvor: autor, kolovoz, 2015.)

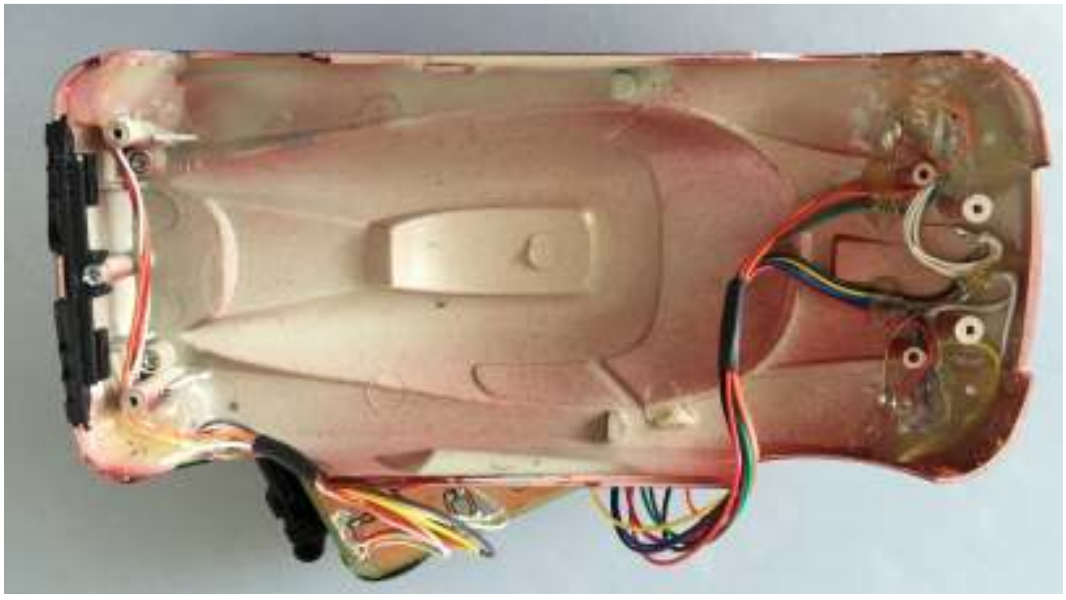
Završetkom procesa lemljenja oba modula, pristupa se ugradnji LED dioda na maketu sportskog automobila (slika 46.).



Slika 46. Maketa sportskog automobila

(izvor: autor, kolovoz, 2015.)

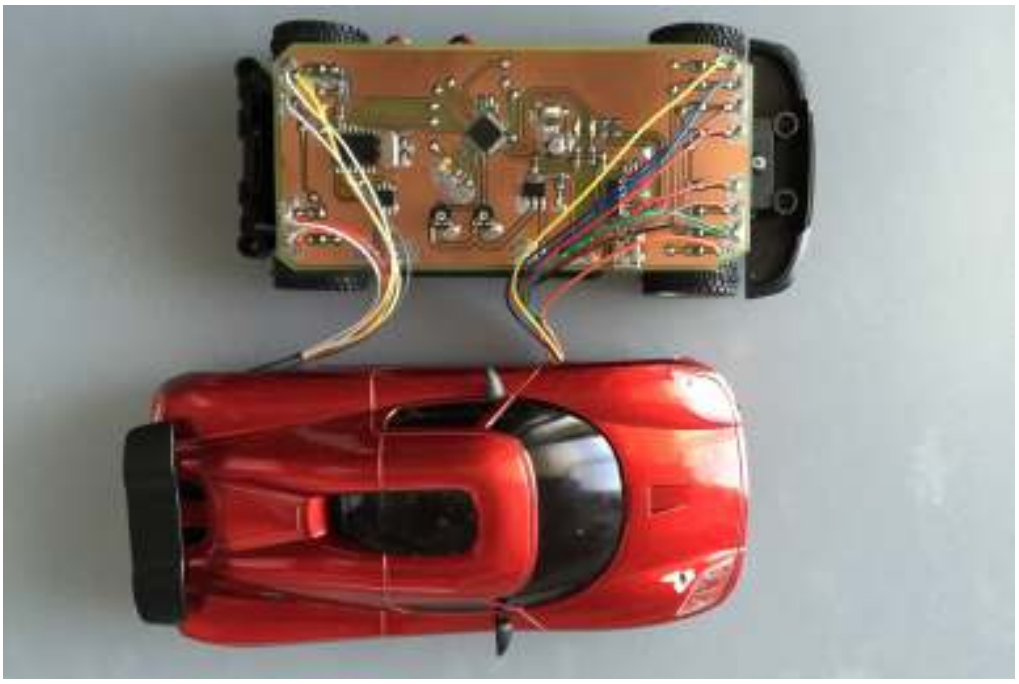
Stolnom bušilicom napravljene su rupe za svjetla kroz karoseriju makete. Rupe su promjera 1.5 mm. U rupe su postavljene LED diode na koje je prethodno zalemljen dvožilni kabel (u svrhu produžnog kabla). LED diode učvršćene su ljepilom s unutarnje strane karoserije makete. Slika 47. prikazuje unutarnju stranu karoserije makete s učvršćenim LED diodama na kojima su produžni kabeli.



Slika 47. Unutarnja strana karoserije makete s ugrađenim LED diodama

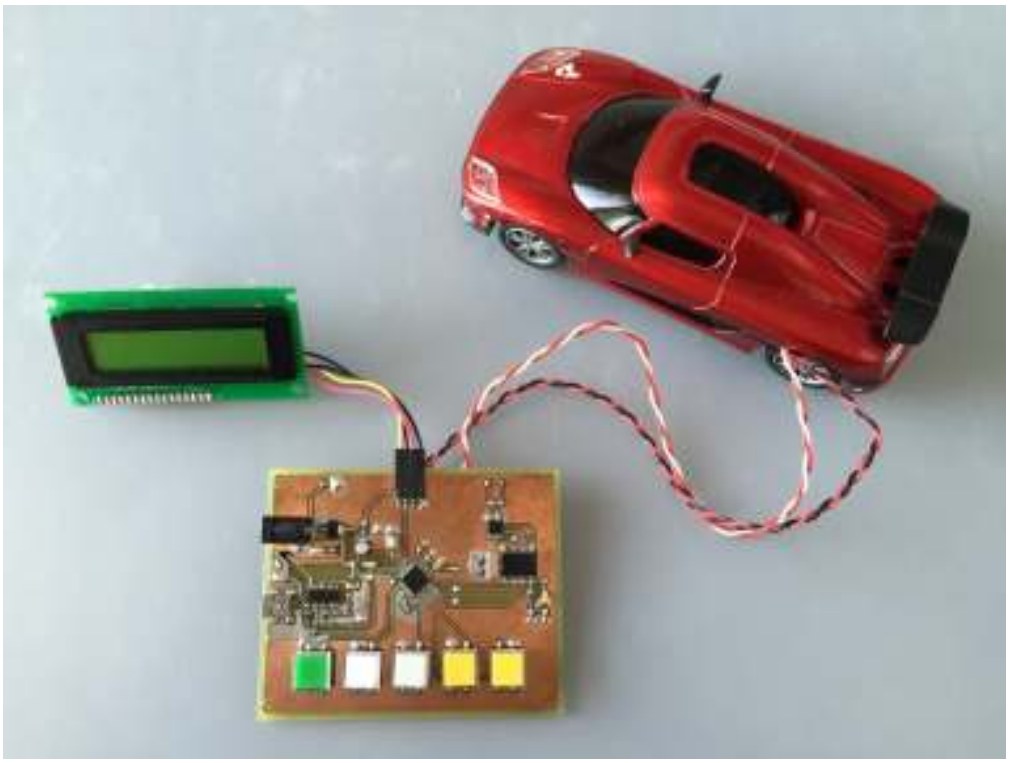
(izvor: autor, kolovoz, 2015.)

U maketu je potrebno još ugraditi „BDM“ modul. Tiskana pločica modula ugrađena je na donji dio karoserije makete. Nakon ugrađivanja, dvije polovice makete su povezane električnim kabelima. Slika 48. prikazuje izgled izrađene makete, dok slika 49. prikazuje kompletno izrađen praktični dio rada.



Slika 48. Izgled izrađene makete

(izvor: autor, kolovoz, 2015.)



Slika 49. Izgled kompletno izrađenog praktičnog dijela rada

(izvor: autor, kolovoz, 2015.)

3.4. Programsko rješenje

Programski kód za mikroprocesore modula pisan je u višem programskom jeziku C++, Arduino IDE (Arduino Integrated Development Environment) softvera. Prije početka pisanja koda potrebno je odrediti oznaku mikroprocesora na koji će se programski kód odnositi. Pokretanjem Arduino IDE softvera prikazuje se osnovni prozor (slika 50.), tj. radno okruženje.



Slika 50. Arduino IDE radno okruženje

(izvor: autor, kolovoz, 2015.)

Pritiskom lijeve tipke miša na izbornik „Tool“ otvara se meni prikazan na slici 51. Pod opcijom „Processor“ odabire se „ATmega328“ pritiskom lijeve tipke miša na naziv.



Slika 51. Izgled izbornika „Tool“ Arduino IDE softvera

(izvor: autor, kolovoz, 2015.)

Po završetku odabira, pristupa se pisanju programskog koda. Napisani programski kód u mikroprocesoru izvršavati će se redom, kako je i pisan u softveru, počevši od vrha liste prema dnu. Konstrukcija programskog koda u Arduino okruženju sastavljena je od tri polja. Polje iznad oznake „void setup()“ (slika 50.) služi za deklariranje svih globalnih varijabli (termin globalno označava mogućnost pristupa istom kroz cijeli programski kód) koje će se koristiti u kodu. Polje „void setup()“ služi pri prvom pokretanju mikroprocesora, bilo prvom uključivanju napajanja ili ponovnog pokretanja (reset). U tom se polju nalazi set instrukcija kojima definiramo postavke, bilo mikroprocesora ili globalne postavke. Posljednje polje „void loop()“ označava programsku petlju koju će mikroprocesor izvoditi neprekidno dok mu se ne isključi napajanje, pritisne tipka za ponovno pokretanje ili dogodi prekid (interrupt).

Cjelovita kopija programskog koda nalazi se u prilozima ove radnje. U nastavku teksta biti će objašnjena kroz dijelove programskog koda za svaki modul posebno.

3.4.1. Programski kód modula „Dashboard“

Programski kód započet je s nekoliko redaka, linija, komentara. Komentari služe samo za potrebe programera kao podsjetnik ili slično te nisu prilikom prebacivanja gotovog programa, snimljeni u mikroprocesor. Dio programa „Dasboard“ modula s komentarima je:

```
/*****************************************************************
* Plotehnika Pula - VTPŠ
* Akademska godina 2014 - 2015
* Završni rad - CAN Bus sustav, MCU Atmel AtMega 328
* DASHBOARD - verzija softvera 1.0
* Kolegij: Elektronika 2
* Mentor:  Sanja Grbac Babić mag.računarstva
* Student: Milan Ljuba
* Pula, kolovoz 2015.
*****************************************************************/
```

Po završetku dijela s komentarima, u programu su definirane (inicijalizirane) biblioteke. Biblioteke sadrže set instrukcija kojima se uvelike olakšava izrada programskog koda. Većina biblioteka, kao što je primjer i u ovoj radnji, su javno dostupne i besplatne. U ovom primjeru biblioteke su korištene za:

```
// inicijalizacija potrebnih biblioteka
#include <mcp_can.h>    // set instrukcija CAN protokola za MCP2515
#include <SPI.h>        // pokretanje SPI priključka (SPI Bus)
#include <Bounce2.h>    // debouncing tipkala
#include <Wire.h>       // I2C priključak LCD-a
#include <LiquidCrystal_I2C.h>  // upravljanje LCD-a
```

Većina navedenih biblioteka koristi se za povezivanje te upravljanje periferijom mikroprocesora, kao na primjer CAN kontrolera (MCP2515 putem SPI priključka), I^2C priključka za LCD te set instrukcija LCD-a. Biblioteka za „debouncing“ (poskakivanje) koristi se za tipkala. Tipkala prilikom pritiska proizvedu nekoliko digitalnih impulsa zbog mehaničke konstrukcije istih, odnosno poskakuju im kontakti. Mikroprocesor, zbog brzog rada (16 MHz) učita sva poskakivanja pa je nemoguće programski odrediti točan broj istih. „Debouncing“ biblioteka zadužena je za provjeru poskakivanja tipkala te filtriranje i prosljeđivanje točnog podatka o tome je li tipkalo pritisnuto ili nije, u program.

Nakon inicijalizacije biblioteka postavljaju se globalne konstantne varijable:

```
const int SPI_CS_PIN = 10;          // postavljanje CS pin-a za MCP2515
const int sw_ligh_count = 5;        // broj tipkala
const long can_interval = 100;      // interval slanja CAN poruke (100ms)
```

te zatim globalne varijable, čiju vrijednost program može mijenjati u radu:

```
int temp = 0;
int rpm = 0;
int CAN_temp = 0;
int CAN_rpm = 0;
int pozicija = LOW;
int kratko = LOW;
int dugo = LOW;
int desni = LOW;
int lijevi = LOW;
int sw_pins[sw_ligh_count] = { 3, 4, 5, 7, 6
int last_sw[sw_ligh_count] = { HIGH, HIGH, HIGH, HIGH, HIGH };
unsigned long previousMillis = 0;
unsigned char flagRecv = 0;
unsigned char len = 0;
unsigned char len2 = 8;
unsigned char primi[8];
unsigned char canId_Rx;
unsigned char canId_Tx = 0x00;
```

Postavljanjem varijable rezervira se dio memorije (za svaku varijablu posebno) u mikroprocesoru. Pri postavljanju varijabli može se dodijeliti početna vrijednost, ali to nije uvjet. Kod izvršavanja programa mikroprocesor pristupa memorijskom dijelu te očitava vrijednost varijable za potrebe obavljanja dijela koda koji se trenutno izvršava. Ukoliko se kodom promjeni prvobitno očitana vrijednost varijable, ta vrijednost zamijeniti će onu iz memorije.

Po postavljanju varijabli prelazi se na „void setup()“ polje. U tom polju pokrenuti će se serijska komunikacija između računala i mikroprocesora, LCD zaslon, postaviti će se interni pull-up otpornici, dodijeliti „debounce“ programski objekt, inicijalizirati CAN kontroler putem SPI priključka (te putem serijskog porta poslati potvrda o uspješnosti inicijalizacije) i na kraju dodijeliti prekidačka (interrupt) rutina.

```
void setup()
{
  Serial.begin(115200);                        // pokreće serijsku
komunikaciju MCU - PC
  lcd.begin();                                 // pokreće LCD
  lcd.backlight();                             // uključuje pozadinsko
svjetlo LCD-a

  for (int i = 0; i < sw_ligh_count; i++) {  // for loop iteracijska
petlja

    pinMode (sw_pins[i], INPUT_PULLUP);      // postavlja interne pull-up
otpornike na portove za tipkala

    debouncer[i].attach (sw_pins[i]);        // dodjeljuje debounce objekt
za svaki pin / tipkalo

    debouncer[i].interval(5);                // vrijeme potrebno da se
tipkalo smiri (5 = 5 ms)
  }

START_INIT:

  if (CAN_OK == CAN.begin(CAN_500KBPS)) {    // inicijalizacija can bus-a
sa brzinom (baudrate) = 500kBps

    start_lcd();                             // pokreće vizualnu
inicijalizaciju LCD-a

    Serial.println(F("CAN Bus OK!"));
    lcd.print(F("CAN Bus OK!"));
    lcd.setCursor(4, 1);
    lcd.print(F("Start prog.."));
    delay(2000);
    dsp();
  }

  else {
    Serial.println(F("CAN Bus pogreska!"));
    Serial.println(F("Provjeri modul i pokusaj ponovno"));
    lcd.print(F("CAN Bus pogreska!"));
    lcd.setCursor(0, 1);
    lcd.print(F("Provjeri modul!"));
    delay(500);
    goto START_INIT;
  }

  attachInterrupt(0, MCP2515_ISR, FALLING); // pokretanje interrupt-a na
pinu D2 - PIN2
}
```

Nastavak programa započinje ulaskom u polje „void loop()“. Polje je podijeljeno u tri dijela. Prvi dio zadužen je za učitavanje stanja tipkala u varijable istih.

Ovisno o učitanom, kroz if – else logiku odlučivanja, mijenja se vrijednost varijabli koje kontroliraju stanje svjetala (uključeno, isključeno) na „BDM“ modulu. If – else logika odlučuje na način da ispituje varijable (upisane u zagradama poslije if) po postavljenim kriterijima i ako je tvrdnja točna izvodi se dio programskog koda u sljedećoj liniji programa. Ukoliko tvrdnja nije točna izvodi se dio programskog koda u liniji ispod else.

```
void loop()
{
  // dio petlje  za tipkala
  for (int i = 0; i < sw_ligh_count; i++) {
    debouncer[i].update();
  }

  int sw1 = debouncer[0].read(); // tipkalo za poziciju (de bounced!)
  int sw2 = debouncer[1].read(); // tipkalo za kratko svjetlo (de bounced!)
  int sw3 = debouncer[2].read(); // tipkalo za dugo svjetlo (de bounced!)
  int sw4 = debouncer[3].read(); // tipkalo za desni žmigavac (de bounced!)
  int sw5 = debouncer[4].read(); // tipkalo za lijevi žmigavac (de bounced!)

  // pozicija on - off
  if (sw1 == LOW && last_sw[1] == HIGH) {
    pozicija = !pozicija;
    last_sw[1] = LOW;
  }
  else {
    last_sw[1] = sw1;
  }

  // kratko svjetlo on - off
  if (sw2 == LOW && last_sw[2] == HIGH) {
    kratko = !kratko;
    last_sw[2] = LOW;
  }
  else {
    last_sw[2] = sw2;
  }

  // dugo svjetlo on - off
  if (sw3 == LOW && last_sw[3] == HIGH) {
    dugo = !dugo;
    last_sw[3] = LOW;
  }
  else {
    last_sw[3] = sw3;
  }

  // lijevi žmigavac on - off
  if (sw4 == LOW && last_sw[4] == HIGH) {
    lijevi = !lijevi;
    last_sw[4] = LOW;
  }
  else {
    last_sw[4] = sw4;
  }

  // desni žmigavac on - off
  if (sw5 == LOW && last_sw[5] == HIGH) {
    desni = !desni;
    last_sw[5] = LOW;
  }
  else {
    last_sw[5] = sw5;
  }

}
```

Drugi dio polja zadužen je za prijem i slanje poruka putem CAN kontrolera.

```
  if (flagRecv) {                   // provjeri ima li poruka
    flagRecv = 0;                   // ako ima, spusti zastavicu
    while (CAN_MSGAVAIL == CAN.checkReceive()) {  // pročitaj poruku i CAN Id
      CAN.readMsgBuf(&len, primi);
      canId_Rx = CAN.getCanId();
      CAN_temp = primi[0];
      CAN_rpm = primi[1];
    }
  }

  unsigned long currentMillis = millis();
  unsigned char salji[8] = {pozicija, kratko, dugo, lijevi, desni, 0, 0, 0};

  if (currentMillis - previousMillis >= can_interval) {
    previousMillis = currentMillis;
    CAN.sendMsgBuf(canId_Tx, 0, len2, salji);       // šalji poruku i svoj CAN Id

    Serial.println(F("-----------------------------")); // ispiši sve poruke na
serial monitor

    Serial.print(F("Poslao sa ID-em: "));
    Serial.println(canId_Tx);
    for (int i = 0; i < len2; i++) {
      Serial.print(salji[i]);
      Serial.print(F("\t"));
      lcd.setCursor(i + 4, 1);
      lcd.print(salji[i]);
    }
    Serial.println();

    Serial.println(F("-----------------------------"));
    Serial.print(F("Primio od ID-a: "));
    Serial.println(canId_Rx);
    for (int i = 0; i < len2; i++)
    {
      Serial.print(primi[i]);
      Serial.print(F("\t"));
    }
    Serial.println();
  }
```

Ukoliko postoji dolazna poruka u CAN kontroleru, koju mikroprocesor još nije učitao, CAN kontroler to javlja mikroprocesoru prekidačkim signalom. Koji god dio programa se trenutno izvršavao, pojavom signala bezuvjetno se preskače na dio programa za prijem CAN poruke. Iz primljene poruke (u ovom praktičnom primjeru poruka sadrži vrijednosti temperature i radnih okretaja motora, a pristiže s „BDM“ modula) učitavaju se vrijednosti iz podatkovnog polja. Učitane se vrijednosti upisuju u varijable za temperaturu i radne okretaje motora.

Sljedeći korak je slanje poruke prema „BDM“ modulu . Poruka se sastavlja na način da se iz varijabli za kontrolu svjetala sastavi cjeloviti skup za slanje. Skup se sastoji od osam 255 bitnih mjesta (podatkovno polje CAN poruke). U sljedećoj liniji

skup se šalje prema CAN kontroleru, a poslana poruka ispisuje putem LCD zaslona, u donjem redu.

U trećem dijelu polja „void loop()“ ispisujemo vrijednosti varijabli temperature i radnih okretaja na gornji red LCD zaslona te potvrđujemo CAN kontroleru prijem interrupta.

```
  lcd.setCursor(4, 0);
  lcd.print(CAN_temp);
  lcd.setCursor(13, 0);
  lcd.print(CAN_rpm);
}
```

Posebni dio programskog koda predstavljaju još tri polja. To su polja „void MCP2515()“, „void start_lcd()“ i „void dsp()“.

```
// void MCP2515_ISR() interrupt objekt
void MCP2515_ISR()
{
  flagRecv = 1;
}

// void za vizualnu incijalizaciju LCD-a
void start_lcd()
{
  lcd.clear();
  lcd.print(F("Politehnika-Pula"));
  lcd.setCursor(0, 1);
  lcd.print(F("CANBus - M.Ljuba"));
  delay(2000);
  lcd.clear();
  lcd.noBacklight();
  delay(800);
  lcd.backlight();
}

void dsp()
{
  lcd.clear();
  lcd.print(F("Tmp:"));
  lcd.setCursor(6, 0);
  lcd.write(0xDF);
  lcd.print(F("C Rpm:"));
  lcd.setCursor(0, 1);
  lcd.print(F("CAN:"));
  lcd.setCursor(12, 1);
  lcd.print(F("ID:"));
  lcd.print(canId_Tx);
}
```

U Arduino IDE okruženju postoji mogućnost dodavanja dodatnih polja, uz osnovna „void setup()“ i „void loop()“. Program u tim poljima izvršiti će se samo ako ga pozovemo iz osnovnih. Polje „void MCP2515()“ sadrži program koji je zadužen za odgovor CAN kontroleru o primljenom prekidačkom signalu. „void start_lcd()“ ispisuje pozdravnu poruku na LCD zaslon pri prvom pokretanju modula, dok „void dsp()“ briše pozdravnu poruku i ispisuje masku na LCD. Maska se koristi kad na LCD zaslonu postoje karakteri koji se nikada u programu ne mijenjaju. Primjerice, na mjestu gdje ispisujemo vrijednost varijable za temperaturu motora, samo vrijednost (brojevi) će se mijenjati, ali ne i znak za mjernu jedinicu. Ukoliko ne koristimo masku, već sve karaktere pozivamo iz polja „void loop()“ postoji velika vjerojatnost da će LCD zaslon titrati, što je neugodno za korisnika.

Kraj programa se završava, nije uvjet već običaj u programiranju, komentarom koji označava isto.

```
/*********************************************************************
  Kraj programa
*********************************************************************/
```

3.4.2. Programski kód modula „BDM“

Programski kód modula „BDM“ također je započet komentarima, inicijalizacijom biblioteka te postavljanjem globalnih varijabli (konstantnih i programski promjenjivih). U inicijalizaciji biblioteka izostavljene su biblioteke za tipkala, LCD zaslon te pripadajuću I^2C biblioteku zaslona, jer se u ovom module ne koriste. Unutar polja „void setup()“ nožice koje su u prethodnom modulu služile kao ulaz tipkala, sada su postavljene kao izlazi za LED diode. Polje „void loop()“ se sastoji od četiri dijela.

Prvi dio odnosi se na učitavanje stanja senzora (promjenjivih otpornika) unutar pripadajućih varijabli, te konverziju 1024 bitnih vrijednosti (Atmega_328P posjeduje 10 bitne analogno digitalne pretvarače) u 255 bitnu vrijednost (funkcija map()) koja se kasnije šalje CAN porukom (četvrti dio polja).

```
void loop()
{
  // dio petlje za učitavanje stanja senzora
  temp = analogRead(temp_pin);
  rpm = analogRead(rpm_pin);
  CAN_temp = map(temp, 0, 1024, 20, 99);
  CAN_rpm = map(rpm, 0, 1024, 100, 250);
  salji[0] = CAN_temp;
  salji[1] = CAN_rpm;
```

Drugi dio odnosi se na prijem CAN poruke pristigle od strane „Dashboard“ modula (programska logika je ista za oba modula) čije podatkovno polje sadrži stanja svjetala.

Treći dio petlje odnosi se na učitavanje podatkovnog polja pristigle poruke u varijable za upravljanje stanjem svjetala (uključeno, isključeno), odnosno LED dioda. LED diode pozicionog, kratkog i dugog svjetla upravljaju se postavljanjem digitalne vrijednosti, upisane u pripadajuću varijablu svjetla, na izlaznu nožicu mikroprocesora na koju je svjetlo spojeno.

```
  // dio petlje za upravljanje svjetlima
  int pozicija = primi[0];
  int kratko = primi[1];
  int dugo = primi[2];
  int lijevi = primi[3];
  int desni = primi[4];
  digitalWrite (LED_pin[0], pozicija); //pozicija on - off
  digitalWrite (LED_pin[1], kratko);   //kratko svjetlo on - off
  digitalWrite (LED_pin[2], dugo);     //dugo svjetlo on - off
  unsigned long currentMillis_desni = millis();
  if (desni == HIGH && currentMillis_desni - previousMillis_desni >= zmg_interval) {
    previousMillis_desni = currentMillis_desni;       // spremi zadnje vrijeme
    if (s_desni == LOW)                               // žmigaj desni
      s_desni = HIGH;
    else
      s_desni = LOW;
  }

  if (desni == LOW) {    // vraća s_desni na LOW ukoliko je žmigavac ugašen u trenutku s_desni
== HIGH
    s_desni = LOW;
  }

  unsigned long currentMillis_lijevi = millis();
  if (lijevi == HIGH && currentMillis_lijevi - previousMillis_lijevi >= zmg_interval) {
    previousMillis_lijevi = currentMillis_lijevi;     // spremi zadnje vrijeme
    if (s_lijevi == LOW)                              // žmigaj lijevi
      s_lijevi = HIGH;
    else
      s_lijevi = LOW;
  }

  if (lijevi == LOW) {   // vraća s_lijevi na LOW ukoliko je žmigavac ugašen u trenutku
s_lijevi == HIGH
    s_lijevi = LOW;
  }

  if (desni == HIGH && lijevi == HIGH) {   //indikator da su oba žmigavca ostala uključena
    s_desni = HIGH;
    s_lijevi = HIGH;

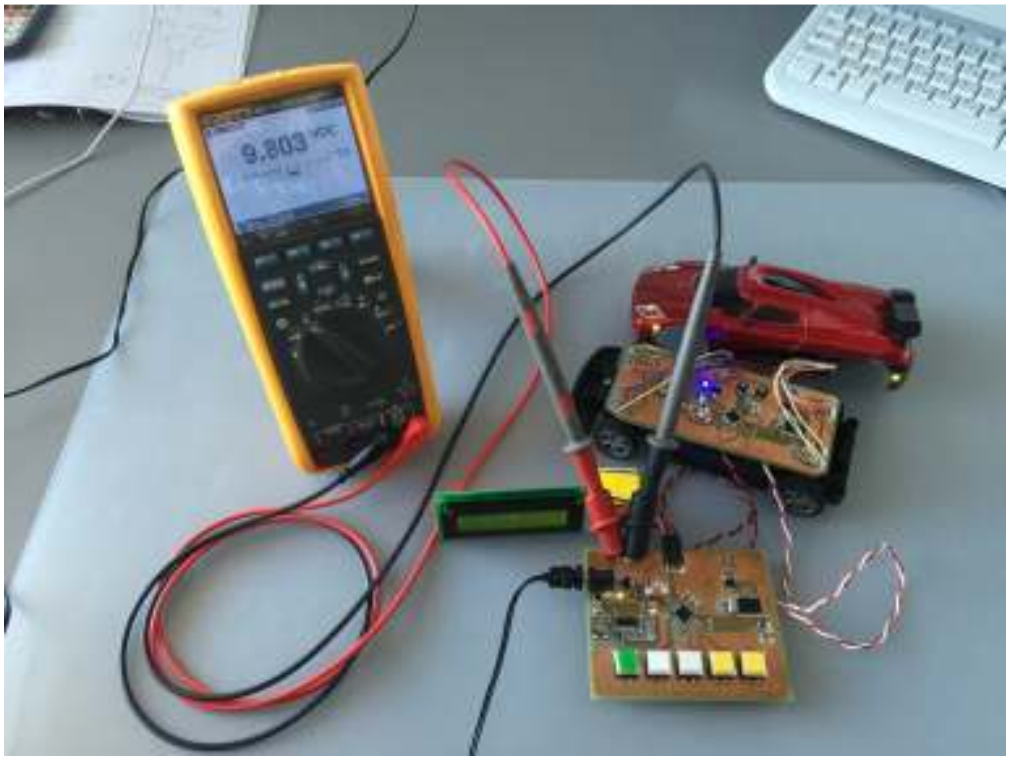
  }
  digitalWrite (LED_pin[3], s_desni);  //desni žmigavac on - off
  digitalWrite (LED_pin[4], s_lijevi); //lijevi žmigavac on - off
```

Za upravljanje lijevim i desnim žmigavcem koristi se if – else logika odlučivanja. Vremenski interval žmiganja iznosi 500ms. If – else logika provjerava da li je vremenski interval žmiganja veći od vremenskog intervala slobodnog vremenskog brojača mikroprocesora (funkcija millis()) te da li je žmigavac upaljen (stanje varijable LOW ili HIGH) te na temelju tih uvjeta kontrolira stanje LED dioda žmigavca (isključeno ili žmigaj po istoimenom vremenskom intervalu). Funkciju millis() potrebno je koristiti jer funkcija delay() zaustavlja program što nikako nije povoljno u ovom primjeru.

Četvrti dio odnosi se na slanje CAN poruke prema „Dashboard“ modulu (programska logika je ista za oba modula) koja sadrži 255 bitnu vrijednosti očitanih senzora temperature i radni okretaja motora. U tom dijelu poslana poruka ispisuje se putem serijske komunikacije. Program je završen komentarom koji označuje kraj.

4. TESTIRANJE MODULA

Cilj testiranja modula je provjera ispravnosti njihovog rada. Provjera se sastoji od ispitivanja ispravnosti elektroničkih komponenti te upisanih programa u mikroprocesore. Ispravnost elektroničkih komponenti izvodi se univerzalnim AVO (struja, napon, otpor) instrumentom. Korišten je instrument proizvođača Fluke, model 287 True RMS. Slika 52. prikazuje proces ispitivanja.



Slika 52. Proces ispitivanja modula univerzalnim instrumentom

(izvor: autor, kolovoz, 2015.)

Nakon ispitivanja elektroničkih komponenti upisan je programski kód u oba modula. Upis programskog koda izvršen je putem USB sučelja „avr-dude“ programom, koji je sastavni dio Arduino IDE softvera. Po završetku upisa ispituje se programski kód na način da se pritiskom tipkala „Dasboard“ modula zamjećuje ispravan rad „BDM“ modula tj. da se LED diode postavljene na maketi uključuju i isključuju. Uz navedeno, vizualno se prati ispis na LCD zaslonu. Oba modula zadovoljila su na testu.

5. NADOGRADNJA I DALJNJE PRIMJENE MODULA

Budući da u ovom primjeru nisu iskorišteni svi portovi (nožice) mikroprocesora Atmega328P, na modul „Dashboard“ moguće je dodati još tipkala kojima bi se kontrolirala stanja određenih dijelova sklopova „BDM“ modula. Također postoji određen broj slobodnih portova i na „BDM“ modulu. Primjerice, sklop za upravljanje brisača vjetrobranskog stakla automobila, brisači stražnjeg stakla, uređaj za ispiranje stakala ili trube automobila. Uz navedene, „BDM“ modulu mogu se dodati senzori koji bi pratili električne veličine uređaja. Primjer: ukoliko nastane pregaranje žarulje određenog svjetla, uz pomoć odgovarajućeg programskog koda, ispisati će se poruka upozorenja na LCD zaslonu, koja navodi mjesto kvara. Uz indikaciju pregaranja, istom se logikom može nadodati indikacija kratkog spoja na instalacijama ili pogrešno izabrane snage zamijenjene žarulje. Sva navedena stanja lako je uočiti praćenjem samo jednog električnog parametara kruga, električne struje. U tu svrhu razvijeni su razni integrirani krugovi, a krug oznake ACS712, proizvođača Allegro bio bi dobar kandidat za početak. ACS712 je integrirani krug koji se sastoji od strujnog senzor koji radi na principu Hallovog efekta (senzor je metalna pločica postavljena u blizini vodiča istosmjerne struje u čijem se magnetskome polju pojavljuju karakteristični napon i struje na pločici). Izlazna veličina integriranog kruga je napon u rasponu od 0V do 5V. Odnos mjerene struje i napona je linearan. Izlazna vrijednost napona se lako može mikroprocesorom digitalizirati.

Daljnje primjene modula su raznovrsne. Kao jedan od primjera, bez ikakvih nadogradnji koje iziskuju promjenu električnih shema, ponovnu izradu tiskanih pločica te ostali dodatni rad, moduli se mogu ugraditi, u svrhu automatizacije, u kuće, stanove, hale tvornica. U toj primjeni mogu poslužiti kao centralno mjesto za upravljanje svjetlima objekta. Ugradnja bi bila vrlo jednostavna, uz male dodatne zahvate na postojećoj instalaciji.

6. ZAKLJUČAK

Ovim radom prikazan je široko primjenjiv CAN protokol mrežnih komunikacija. Objašnjeni su i prikazani softverski alati uz pomoć kojih je ideju jednostavnije realizirati u opipljivu stvarnost stoga je polazna hipoteza potvrđena.

Kroz rad je osmišljen, razvijen i realiziran mikroprocesorski sklop modula što zadovoljava cilj i svrhu istog.

Radom na praktičnom primjeru produbljene su tehnike te osnažene vještine koje će predstavljati veliku prednost i konkurentnost u praksi. Postavljeni su i čvrsti temelji za nadogradnje i daljnje primjene ograničene samo inovativnošću ideja onoga tko ih razmatra.

7. LITERATURA

[1] Olaf Pfeiffer, Andrew Ayre, Christian Keydel: Embedded Networking with CAN and CANopen, Copperhill Technologies Corporation, Greenfield USA, 2003. g.

[2] Robert Bosch GmbH: CAN Specification Version 2.0, Robert Bosch GmbH, Stuttgart Njemačka, 1991. g.

[3] Texas Instruments: Introduction to the Controller Area Network, Dallas USA, 2002. - 2008. g.

[4] Atmel Corporation: ATMEL 8-BIT microcontroller datasheet summary, San Hose CA USA, 2014. g.

[5] Microchip Technology Inc.: Stand-Alone CAN Controller with SPI Interface, Chandler USA, 2012. g.

[6] Microchip Technology Inc.: An In-depth Look at the MCP2510, Chandler USA, 2001. g.

[7] Philips Semiconductors: TJA1050 High speed CAN transceiver, California USA, 2003. g.

[8] WCH - DreamCity Innovations: CH340G USB to UART Interface, Kina, 2008. g.

INDEKSI

INDEKS SLIKA

Slika 1. Piramida komunikacijskih protokola u automatizaciji 4
Slika 2. ISO referentni model prema CAN protokolu 6
Slika 3. Primjer spoja CAN primopredajnika 7
Slika 4. Naponski nivoi CAN sabirnice 8
Slika 5. Standardni podatkovni okvir CAN sabirnice 9
Slika 6. Prošireni podatkovni okvir CAN sabirnice 11
Slika 7. Raspored nožica s opisom mikroprocesora ATmega328p 15
Slika 8. Blok diagram CAN kontrolera MCP2515 16
Slika 9. Izgled ploče sa sklopovima oba modula. 18
Slika 10. Izgled upravljačkog panela EAGLE aplikacije 19
Slika 11. Kreiranje novog projekta 19
Slika 12. Kreiranje električne sheme 20
Slika 13. Određivanje biblioteke projekta 21
Slika 14. Osnovni prozor „Schematic“ aplikacije 21
Slika 15. Izgled izbornika „Add a part“ 22
Slika 16. Električna shema napajanja 23
Slika 17. USB – UART sklop 24
Slika 18. CAN sekcija 26
Slika 19. Osnovno okruženje mikroprocesora 27
Slika 20. Spoj mikroprocesora u „Dasboard“ modulu 28
Slika 21. Tipkala „Dasboard“ modula 29
Slika 22. Spoj mikroprocesora u „BDM“ modulu 30
Slika 23. Električni spoj LED dioda i otpornika „BDM“ modula 31
Slika 24. Prijelaz iz „Schematic“ aplikacije u aplikaciju „Board editor“ 32
Slika 25. Radno okruženje aplikacije „Board editor“ 33

Slika 26. Tiskana pločica s raspoređenim elementima 34
Slika 27. Rutiranje između mikroprocesora i tipkala 35
Slika 28. Izgled rutirane tiskane pločice 35
Slika 29. Odabir alata „Poligon“ 36
Slika 30. Izbornik sa svojstvima okvira 37
Slika 31. Odabir alata „Ratsnest“ 38
Slika 32. Izgled tiskane pločice nakon upotrebe alata „Ratsnest“ 39
Slika 33. Izgled alata „Layer“ 40
Slika 34. Izgled izbornika stavke „Print“ 41
Slika 35. Izgled tiskane pločice „BDM“ nakon rutanja 42
Slika 36. Izgled maske tiskane pločice „BDM“ 42
Slika 37. Film za razvijanje tiskanih pločica i materijal za njihovu izradu 43
Slika 38. Obrađeni materijal za izradu tiskanih pločica 43
Slika 39. Postavljanje materijala i filma u uređaj za osvjetljavanje 44
Slika 40. Kemikalije korištene u procesima razvijanja i jetkanja materijala 45
Slika 41. Materijal „Dashboard“ modula pri kraju procesa jetkanja 46
Slika 42. Materijal „BDM“ modula pri kraju procesa jetkanja 46
Slika 43. Izrađene tiskane pločice oba modula 47
Slika 44. Bušenje rupa kroz tiskane pločice modula 48
Slika 45. Proces lemljenja elektroničkih elemenata 48
Slika 46. Maketa sportskog automobila 49
Slika 47. Unutarnja strana karoserije makete s ugrađenim LED diodama 49
Slika 48. Izgled izrađene makete 50
Slika 49. Izgled kompletno izrađenog praktičnog dijela rada 50
Slika 50. Arduino IDE radno okruženje 51
Slika 51. Izgled izbornika „Tool“ Arduino IDE softvera 52
Slika 52. Proces ispitivanja modula univerzalnim instrumentom 62

INDEKS TABLICA

Tablica 1. Najveće brzine prijenosa podataka u ovisnosti duljine CAN sabirnice 9

Tablica 2. Osnovne i električne karakteristike mikroprocesora ATmega328P 15

Tablica 3. električne karakteristike modula 17

INDEKS PRILOGA

1. Programski kód modula „Dashboard“ 69

3. Električna shema modula „Dashboard 76

4. Električna shema modula „BDM“ 77

5. Medij za pohranu podataka 78

PRILOZI

1. Programski kód modula „Dashboard“

```
/*********************************************************************
 * Plotehnika Pula - VTPŠ
 * Akademska godina 2014 - 2015
 * Završni rad - CAN Bus sustav, MCU Atmel AtMega 328
 * DASHBOARD - verzija softvera 1.0
 * Kolegij: Elektronika 2
 * Mentor:  Sanja Grbac Babić, mag.računarstva
 * Student: Milan Ljuba
 * Pula, kolovoz 2015.
*********************************************************************/

// inicijalizacija potrebnih biblioteka
#include <mcp_can.h>
#include <SPI.h>
#include <Bounce2.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// const int varijable čije se vrijednosti ne mogu mijenjati kroz petlju
const int SPI_CS_PIN = 10;
const int sw_ligh_count = 5;
const long can_interval = 100;

// int varijable čije se vrijednosti mogu mijenjati kroz petlju
int temp = 0;
int rpm = 0;
int CAN_temp = 0;
int CAN_rpm = 0;
int pozicija = LOW;
int kratko = LOW;
int dugo = LOW;
int desni = LOW;
int lijevi = LOW;
int sw_pins[sw_ligh_count] = { 3, 4, 5, 7, 6 };
int last_sw[sw_ligh_count] = { HIGH, HIGH, HIGH, HIGH, HIGH };
unsigned long previousMillis = 0;
unsigned char flagRecv = 0;
unsigned char len = 0;
unsigned char len2 = 8;
unsigned char primi[8];
unsigned char canId_Rx;
unsigned char canId_Tx = 0x00;

Bounce * debouncer = new Bounce[sw_ligh_count];
MCP_CAN CAN(SPI_CS_PIN);
LiquidCrystal_I2C lcd(0x27, 16, 2);

// void setup - izvršava se samo jednom prilikom prvog pokretanja MCU-a
void setup()
{
  Serial.begin(115200);
  lcd.begin();
  lcd.backlight();
  for (int i = 0; i < sw_ligh_count; i++) {
    pinMode (sw_pins[i], INPUT_PULLUP);
    debouncer[i].attach (sw_pins[i]);
```

```
    debouncer[i].interval(5);
  }

START_INIT:

  if (CAN_OK == CAN.begin(CAN_500KBPS)) {
    start_lcd();
    Serial.println(F("CAN Bus OK!"));
    lcd.print(F("CAN Bus OK!"));
    lcd.setCursor(4, 1);
    lcd.print(F("Start prog.."));
    delay(2000);
    dsp();
  }

  else {
    Serial.println(F("CAN Bus pogreska!"));
    Serial.println(F("Provjeri modul i pokusaj ponovno"));
    lcd.print(F("CAN Bus pogreska!"));
    lcd.setCursor(0, 1);
    lcd.print(F("Provjeri modul!"));
    delay(500);
    goto START_INIT;
  }

  attachInterrupt(0, MCP2515_ISR, FALLING);
}

// void loop - programska petlja
void loop()
{
  // dio petlje  za tipkala
  for (int i = 0; i < sw_ligh_count; i++) {
    debouncer[i].update();
  }

  int sw1 = debouncer[0].read();
  int sw2 = debouncer[1].read();
  int sw3 = debouncer[2].read();
  int sw4 = debouncer[3].read();
  int sw5 = debouncer[4].read();

  // pozicija on - off
  if (sw1 == LOW && last_sw[1] == HIGH) {
    pozicija = !pozicija;
    last_sw[1] = LOW;
  }
  else {
    last_sw[1] = sw1;
  }

  // kratko svjetlo on - off
  if (sw2 == LOW && last_sw[2] == HIGH) {
    kratko = !kratko;
    last_sw[2] = LOW;
  }
  else {
    last_sw[2] = sw2;
  }

  // dugo svjetlo on - off
  if (sw3 == LOW && last_sw[3] == HIGH) {
    dugo = !dugo;
    last_sw[3] = LOW;
```

```
  }
  else {
    last_sw[3] = sw3;
  }

  // lijevi žmigavac on - off
  if (sw4 == LOW && last_sw[4] == HIGH) {
    lijevi = !lijevi;
    last_sw[4] = LOW;
  }
  else {
    last_sw[4] = sw4;
  }

  // desni žmigavac on - off
  if (sw5 == LOW && last_sw[5] == HIGH) {
    desni = !desni;
    last_sw[5] = LOW;
  }
  else {
    last_sw[5] = sw5;
  }

// dio petlje za prijem CAN Bus poruke
  if (flagRecv) {
    flagRecv = 0;
    while (CAN_MSGAVAIL == CAN.checkReceive()) {
      CAN.readMsgBuf(&len, primi);
      canId_Rx = CAN.getCanId();
      CAN_temp = primi[0];
      CAN_rpm = primi[1];
    }
  }

// dio petlje za slanje CAN Bus poruke
 unsigned long currentMillis = millis();
 unsigned char salji[8] = {pozicija, kratko, dugo, lijevi, desni, 0, 0, 0};

  if (currentMillis - previousMillis >= can_interval) {
    previousMillis = currentMillis;
    CAN.sendMsgBuf(canId_Tx, 0, len2, salji);
    Serial.println(F("-----------------------------"));
    Serial.print(F("Poslao sa ID-em: "));
    Serial.println(canId_Tx);
    for (int i = 0; i < len2; i++) {
      Serial.print(salji[i]);
      Serial.print(F("\t"));
      lcd.setCursor(i + 4, 1);
      lcd.print(salji[i]);
    }
    Serial.println();

    Serial.println(F("-----------------------------"));
    Serial.print(F("Primio od ID-a: "));
    Serial.println(canId_Rx);
    for (int i = 0; i < len2; i++)
    {
      Serial.print(primi[i]);
      Serial.print(F("\t"));
    }
    Serial.println();
  }
```

```
// dio petlje za prikaz primljene CAN poruke na LCD
  lcd.setCursor(4, 0);
  lcd.print(CAN_temp);
  lcd.setCursor(13, 0);
  lcd.print(CAN_rpm);
}

// void MCP2515_ISR interrupt objekt
void MCP2515_ISR()
{
  flagRecv = 1;
}

// void za vizualnu inicijalizaciju LCD-a
void start_lcd()
{
  lcd.clear();
  lcd.print(F("Politehnika-Pula"));
  lcd.setCursor(0, 1);
  lcd.print(F("CANBus - M.Ljuba"));
  delay(2000);
  lcd.clear();
  lcd.noBacklight();
  delay(800);
  lcd.backlight();
}

void dsp()
{
  lcd.clear();
  lcd.print(F("Tmp:"));
  lcd.setCursor(6, 0);
  lcd.write(0xDF);
  lcd.print(F("C Rpm:"));
  lcd.setCursor(0, 1);
  lcd.print(F("CAN:"));
  lcd.setCursor(12, 1);
  lcd.print(F("ID:"));
  lcd.print(canId_Tx);
}
/*********************************************************************
  Kraj programa
*********************************************************************/
```

2. Programski kód modula „BDM“

```
/*********************************************************************
 * Plotehnika Pula - VTPŠ
 * Akademska godina 2014 - 2015
 * Završni rad - CAN Bus sustav, MCU Atmel AtMega 328
 * BDM - verzija softvera 1.0
 * Kolegij: Elektronika 2
 * Mentor:  Sanja Grbac Babić, mag.računarstva
 * Student: Milan Ljuba
 * Pula, kolovoz 2015.
*********************************************************************/

// inicijalizacija potrebnih biblioteka
#include <SPI.h>
#include "mcp_can.h"

// const int varijable čije se vrijednosti ne mogu mijenjati kroz
petlju
const int pin_count = 5;
const int SPI_CS_PIN = 10;
const long zmg_interval = 500;
const long can_interval = 100;
const int temp_pin = A0;
const int rpm_pin = A1;

// int varijable čije se vrijednosti mogu mijenjati kroz petlju
int LED_pin[pin_count] = { 3, 4, 5, 6, 7 };
int s_desni = LOW;
int s_lijevi = LOW;
int temp = 0;
int rpm = 0;
int CAN_temp = 0;
int CAN_rpm = 0;
unsigned long previousMillis = 0;
unsigned char flagRecv = 0;
unsigned char len = 0;
unsigned char len2 = 8;
unsigned char primi[8];
unsigned long previousMillis_desni = 0;
unsigned long previousMillis_lijevi = 0;
unsigned char canId_Rx;
unsigned char canId_Tx = 0x7B;
unsigned char salji[8];
MCP_CAN CAN(SPI_CS_PIN);

// void setup - izvršava se samo jednom prilikom prvog pokretanja MCU-a
void setup()
{
  Serial.begin(115200);

  for ( int i = 0; i < pin_count; i ++) {
    pinMode (LED_pin[i], OUTPUT);
    digitalWrite (LED_pin[i], LOW);
  }

START_INIT:

  if (CAN_OK == CAN.begin(CAN_500KBPS))
  {
```

```
    Serial.println(F("CAN BUS ok!"));
  }
  else
  {
    Serial.println(F("CAN BUS  pogreska!"));
    Serial.println(F("Provjeri modul i pokusaj ponovno"));
    delay(100);
    goto START_INIT;
  }

  attachInterrupt(0, MCP2515_ISR, FALLING);
}



void loop()
{
// dio petlje za čitanje stanja senzora
  temp = analogRead(temp_pin);
  rpm = analogRead(rpm_pin);
  CAN_temp = map(temp, 0, 1024, 20, 99);
  CAN_rpm = map(rpm, 0, 1024, 100, 250);
  salji[0] = CAN_temp;
  salji[1] = CAN_rpm;

// dio petlje za prijem CAN Bus poruke
  if (flagRecv) {
    flagRecv = 0;
    while (CAN_MSGAVAIL == CAN.checkReceive()) {
      CAN.readMsgBuf(&len, primi);
      canId_Rx = CAN.getCanId();
    }
  }

// dio petlje za upravljanje svjetlima
  int pozicija = primi[0];
  int kratko = primi[1];
  int dugo = primi[2];
  int lijevi = primi[3];
  int desni = primi[4];
  digitalWrite (LED_pin[0], pozicija);
  digitalWrite (LED_pin[1], kratko);
  digitalWrite (LED_pin[2], dugo);
  unsigned long currentMillis_desni = millis();
  if (desni == HIGH && currentMillis_desni - previousMillis_desni >=
zmg_interval) {
    previousMillis_desni = currentMillis_desni;
    if (s_desni == LOW)
      s_desni = HIGH;
    else
      s_desni = LOW;
  }

  if (desni == LOW) {
    s_desni = LOW;
  }

unsigned long currentMillis_lijevi = millis();
if (lijevi == HIGH && currentMillis_lijevi - previousMillis_lijevi >=
zmg_interval) {
    previousMillis_lijevi = currentMillis_lijevi;
```

```
    if (s_lijevi == LOW)
      s_lijevi = HIGH;
    else
      s_lijevi = LOW;
  }

  if (lijevi == LOW) {
    s_lijevi = LOW;
  }

  if (desni == HIGH && lijevi == HIGH) {
    s_desni = HIGH;
    s_lijevi = HIGH;

  }
  digitalWrite (LED_pin[3], s_desni);
  digitalWrite (LED_pin[4], s_lijevi);

// dio petlje za slanje CAN Bus poruke
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= can_interval) {
    previousMillis = currentMillis;
    CAN.sendMsgBuf(canId_Tx, 0, len2, salji);

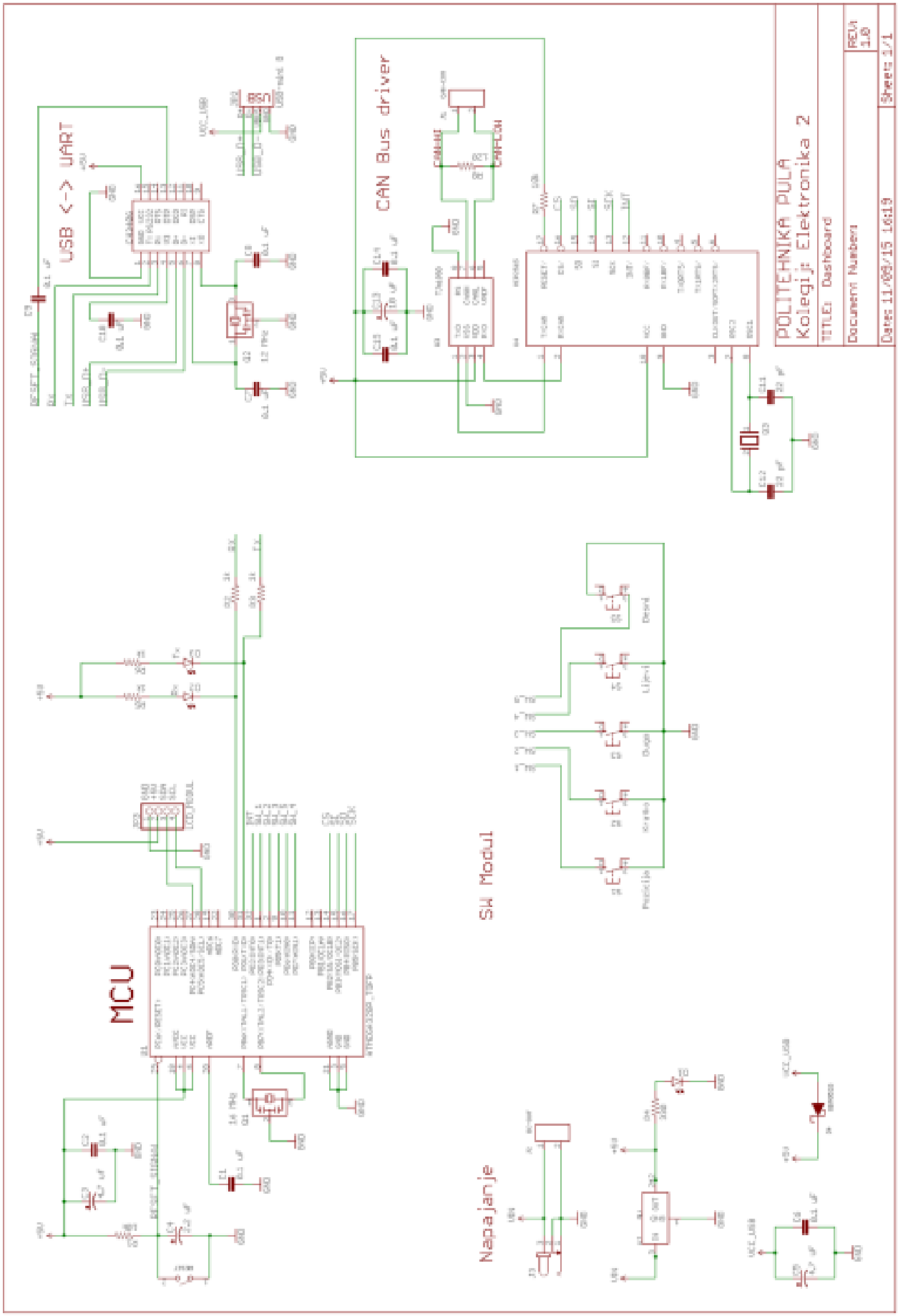

    Serial.println(F("-----------------------------"));
    Serial.print(F("Poslao sa ID-em: "));
    Serial.println(canId_Tx);
    for (int i = 0; i < len2; i++) {
      Serial.print(salji[i]);
      Serial.print(F("\t"));
    }
    Serial.println();

    Serial.println(F("-----------------------------"));
    Serial.print(F("Primio od ID-a: "));
    Serial.println(canId_Rx);
    for (int i = 0; i < len2; i++)
    {
      Serial.print(primi[i]);
      Serial.print(F("\t"));
    }
    Serial.println();
  }
}

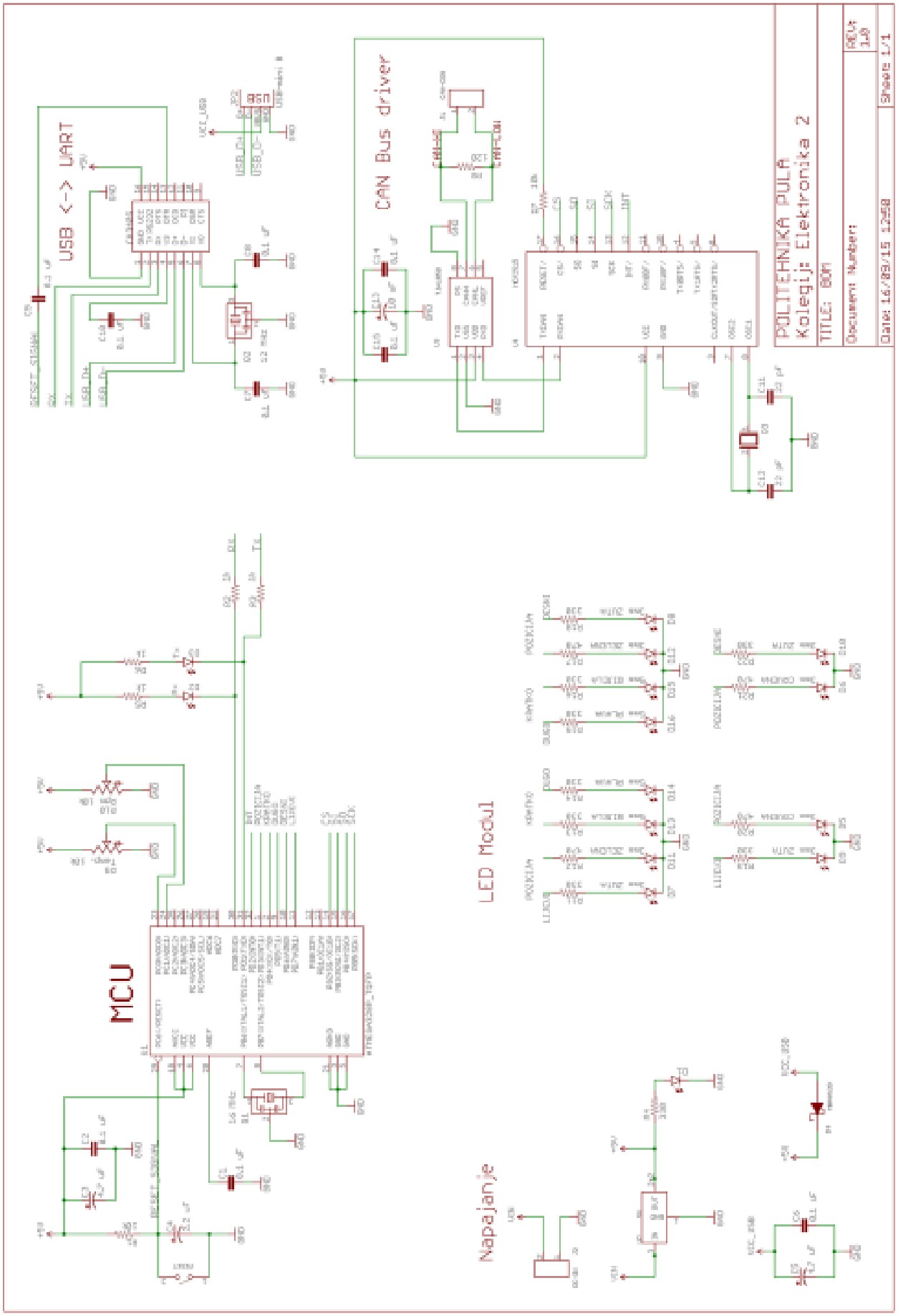
//void MCP2515_ISR() interrupt objekt
void MCP2515_ISR()
{
  flagRecv = 1;
}

/*********************************************************************
Kraj programa
*********************************************************************/
```

3. Električna shema modula „Dashboard



4. Električna shema modula „BDM“



5. Medij za pohranu podataka

Na mediju nalaze se sljedeće datoteke:

- električne sheme modula
- izgled tiskanih pločica modula
- programsko rješenje modula
- nacrt stalka „Dashboard“ modula
- USB na UART upravljački programi (Linux, MacOS, Windows)



IZJAVA O SAMOSTALNOSTI IZRADE ZAVRŠNOG RADA

Izjavljujem da sam diplomski rad na temu IZRADA MIKROPROCESORSKIH MODULA NA PRIMJERU UPRAVLJANJA SVJETLOSNOM SIGNALIZACIJOM AUTOMOBILA PUTEM CAN BUS PROTOKOLA izrado potpuno samostalno, koristeći se literaturom i vlastitim znanjem te znanjem stečenim tijekom studija na Politehnici Pula - Visokoj tehničko-poslovnoj školi – Politehničkom studiju u Puli pod voditeljstvom mentora Sanje Grbac Babić mag.računarstva. Rad je pisan primjenjujući metodologiju znanstveno – istraživačkog rada u duhu hrvatskog jezika.

U Puli, ……..

Student: Milan Ljuba

……..