UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA U NOVOM SADU

Stefan Velbl

Digitalno upravljanje DC motorom sa povratnom spregom po brzini i struji

Diplomski rad
-Osnovne akademske studije-

Novi Sad, 2021.

UNIVERZITET U NOVOM SADU ● **FAKULTET TEHNIČKIH NAUKA**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj, **RBR**:		
Identifikacioni broj, **IBR**:		
Tip dokumentacije, **TD**:		Monografska publikacija
Tip zapisa, **TZ**:		Tekstualni štampani primerak
Vrsta rada, **VR**:		Diplomski rad
Autor, **AU**:		Stefan Velbl
Mentor, **MN**:		dr Ivan Todorović
Naslov rada, **NR**:		Digitalno upravljanje DC motorom sa zatvorenom povratnom spregom po brzini i struji
Jezik publikacije, **JP**:		Srpski
Jezik izvoda, **JI**:		Srpski
Zemlja publikovanja, **ZP**:		Srbija
Uže geografsko područje, **UGP**:		AP Vojvodina
Godina, **GO**:		2021
Izdavač, **IZ**:		Autorski reprint
Mesto i adresa, **MA**:		Fakultet tehničkih nauka, 21000 Novi Sad, Trg Dositeja Obradovića 6
Fizički opis rada, **FO**: (poglavlja/strana/ citata/tablea/slika/grafika/priloga)		(7/52/0/0/51/8/2)
Naučna oblast, **NO**:		Elektrotehnika
Naučna disciplina, **ND**:		Energetska elektronika i električne mašine
Predmetna odrednica/Ključne reči, **PO**:		C programski jezik, mikrokontroler, jednosmerna mašina, energetski pretvarač, upravljanje, regulacija
UDK		
Čuva se, **ČU**:		Biblioteka Fakulteta tehničkih nauka, Trg Dositeja Obradovića 6
Važna napomena, **VN**:		
Izvod, **IZ**		U radu je opisan softverski kod za PWM upravljanje jednosmernom mašinom regulisane po brzini i struji
Datum prihvatanja teme, **DP**:		xx.xx.2020.
Datum odbrane, **DO**:		xx.01.2021.
Članovi komisije, **KO**:	Predsednik:	xx
	Član:	xx
	Član, mentor:	dr Ivan Todorović

Potpis mentora

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual Printed Material	
Contents code, **CC**:		Bachelor Thesis	
Author, **AU**:		Stefan Velbl	
Mentor, **MN**:		Ivan Todorović, Ph.D.	
Title, **TI**:		Digital DC motor control with speed and current closed loop	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		English	
Country of publication, **CP**:		Serbia	
Locality of publication, **LP**:		AP Vojvodina	
Publication year, **PY**:		2021	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Faculty of technical sciences, 21000 Novi Sad, Trg Dositeja Obradovića 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		(7/52/0/0/51/8/2)	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Power electronics and electrical machines	
Subject/Key words, **S/KW**:		C programming language, microcontroller, direct current machine, power converter, control, regulation	
UC			
Holding data, **HD**:		The library of Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad	
Note, **N**:			
Abstract, **AB**:		This paper describes the software code for PWM control of a DC machine regulated by speed and current	
Accepted by the Scientific Board on, **ASB**:		xx.xx.2020.	
Defended on, **DE**:		xx.01.2021.	
Defended board, **DB**:	President:	xx	
	Member:	xx	Menthor's sign
	Member, Mentor:	Ivan Todorović, Ph.D.	

	UNIVERZITET U NOVOM SADU ● **FAKULTET TEHNIČKIH NAUKA** 21000 NOVI SAD, Trg Dositeja Obradovića 6	Broj:
	ZADATAK ZA IZRADU DIPLOMSKOG (BACHELOR) RADA	Datum:

(Podatke unosi predmetni nastavnik - mentor)

Vrste studija:	Osnovne akademske studije
Studijski program:	Energetika, elektronika i telekomunikacije
Rukovodilac studijskog programa:	dr Milan Sečujski

Student:	Stefan Velbl	Broj indeksa:	EE 217/2016
Oblast:	Energetska elektronika i električne mašine		
Mentor:	dr Ivan Todorović		

NA OSNOVU PODNETE PRIJAVE, PRILOŽENE DOKUMENTACIJE I ODREDBI STATUTA FAKULTETA IZDAJE SE ZADATAK ZA ZAVRŠNI (Bachelor) RAD, SA SLEDEĆIM ELEMENTIMA:

- problem – tema rada;
- način rešavanja problema i način praktične provere rezultata rada, ako je takva provera neophodna;
- literatura

NASLOV DIPLOMSKOG (BACHELOR) RADA:

Digitalno upravljanje DC motorom sa povratnom spregom po brzini i struji

TEKST ZADATKA:

U ovom radu potrebno je:

- upoznati se sa dsPIC30F4011 kontrolerom, mogućnostima njegovog hardvera i odgovarajućim integrisanim razvojnim okruženjem za razvoj C koda;
- izvršiti proračun parametara regulatora;
- osmisliti arhitekturu softvera za upravljanje DC motorom;
- implementirati C kod za upravljanje DC motorom;

Rukovodilac studijskog programa:	Mentor rada:
	dr Ivan Todorović

Primerak za: ☐ - Studenta; ☐ - Mentora

Sadržaj

1. Uvod

Razvojem energetske elektronike i potrebama za pretvaranjem električne energije iz jedne vrste u drugu došlo je do prelaska sa čisto mehaničkih naprava na napredne sisteme u kojima se koristi veliki broj pretvarača energetske elektronike. Sam pretvarač energetske elektronike oblikuje i upravlja tokom energije na svojim pristupnim krajevima i za njegov pouzdan i efikasan rad zaslužan je upravljački sklop tj. kontroler/mikrokontroler. Mikrokontroleri izvršavaju definisan program, koji je često napisan korišćenjem C programskog jezika, a koji je u mikrokontroleru implementiran korišćenjem mašinskog jezika (asemblera). Dobar primer primene velikog broja upravljačkih sklopova jesu savremeni automobili koji za razliku od automobila proizvedenih sredinom i krajem 20. veka imaju neuporedivo veći broj elektronskih uređaja. Broj elektronskih uređaja unutar jednog vozila je neprestano rastao, od jednog centralnog računara do oko 150 elektronskih kontrolnih jedinica koliko poseduje jedan savremeni automobil. Iz tog razloga, došlo je do povećane potražnje za softverom višeg i nižeg nivoa kojim će se omogućiti digitalno upravljanje raznim elektronskim uređajima i sistemima elektronskih uređaja unutar vozila.

1.1. Cilj rada

Primarni cilj istraživanja u okviru diplomskog rada jeste realizacija softvera namenjenog za PWM (eng. *pulse-with modulation)* upravljanje mašinom jednosmerne struje sa povratnom spregom po brzini i struji.

Mašine jednosmerne struje u današnje vreme imaju izuzetno veliku primenu u automobilskoj industriji. Počevši od motora malih napona koji služe kao pokretački mehanizmi raznih delova u automobilu (podizači prozora, brisači, motori za pomeranje sedišta, ventilacija kabine) sve do motora srednjih i visokih napona koji predstavljaju pogon savremenog električnog/hibridnog vozila. Pored mašina jednosmerne struje kao pogon električnih/hibridnih vozila koriste se naravno i ostale električne mašine (asinhrone, sinhrone). Međutim, upravljanje mašinama jednosmerne struje je mnogo jednostavnije od ostalih električnih mašina. Uzimajući to u obzir ovaj rad će se bazirati upravo na takvoj mašini, jer je cilj ovog rada da bude koristan narednim generacijama studenata koji će se prvi put u laboratoriji susresti sa digitalnim upravljanjem električnim mašinama.

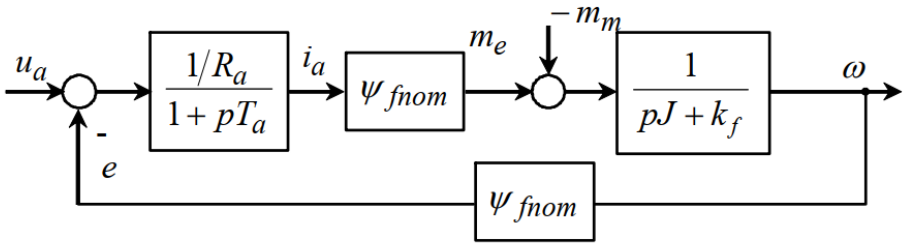
1.2. Organizacija rada

Rad je podeljen u sedam celina.

- U prvom delu, istaknut je značaj energetske elektronike i softvera u savremenim elektromotornim pogonima kao i cilj izrade rada. Data je organizacija rada po celinama i objašnjeno šta je u kojoj celini rađeno.
- U drugom delu dat je teorijski uvod u kojem je predstavljen matematički model jednosmerne mašine, princip rada četvorokvadrantnog čopera, zadaci i značaj primene mikrokontrolera u upravljanju pogonima jednosmerne struje kao i upravljačka struktura u pogonima jednosmerne struje.
- U trećem delu fokus je na samoj implementaciji - odabir hardverskog okruženja, proračun parametara korišćenih PI regulatora i arhitektura softvera. Na kraju detaljno je opisana implementacija C koda za ostvarivanje algoritma za upravljanje DC (eng. *direct current)* motorom sa regulacijom po brzini i struji, upotrebom četvorokvadrantnog čopera.
- U četvrtom delu opisana su sprovedena testiranja.
- U petom delu dati su zaključci izvedeni na osnovu rezultata testiranja kao i mogućnosti za dalji rad.
- U šestom delu navedena je korišćena literatura.
- U sedmom delu, kao prilog, opisan je sistem normalizovanih vrednosti i aritmetika sa nepokretnim zarezom.

2. Teorijski uvod

2.1. Matematička predstava jednosmerne mašine



Slika 2.1.1 Model motora jednosmerne struje sa konstantnim fluksom

U radu je kao izvršni organ, korišćen DC motor koji poseduje permamentne magnete i shodno tome relativno konstantan fluks rotora. U narednom tekstu navedene su jednačine DC motora sa konstantnim fluksom.

Jednačina naponske ravnoteže rotora ima oblik:

$$U_a = R_a \cdot i_a + L_a \cdot \frac{di_a}{dt} + \Psi_f \cdot \omega \quad \text{2.1.1}$$

gde je:

U_a - napon armature,

R_a - otpornost armature,

i_a- struja armature,

L_a- induktivnost armature,

Ψ_f- ukupan fluks,

ω - električna ugaona brzina.

Vremenska konstanta indukta (armaturnog kola) se računa sa:

$$T_a = \frac{L_a}{R_a}. \quad \text{2.1.2}$$

Ostvareni elektromagnetni momenat se računa sa:

$$m_e = \Psi_f \cdot i_a. \quad \text{2.1.3}$$

Jednačina mehaničke ravnoteže na vratilu ima oblik:

$$J\frac{d\omega}{dt} = m_e - m_m - k_f \cdot \omega \quad \text{2.1.4}$$

gde je:

J - moment inercije,

m_m - mehanički moment,

k_f - faktor ugaone brzine.

Nominalni podaci DC motora korišćenog u ovom radu su:

- $U_{an} = 230\,V$
- $I_{an} = 18\,A$
- $\omega_n = 353\ rad/s$
- $n_n = 3370\ obr/min$
- $L_a = 3.32\,mH$
- $R_a = 0.75\,\Omega$
- $T_a = \frac{L_a}{R_a} = 4.43\,ms$
- $\Psi_{fnom} = 0.7\,Wb$
- $J = 0.02\,kg\,m^2$

2.2. Princip rada čopera

Čoperi spadaju u grupu DC/DC energetskih pretvarača. Oni jednosmeran napon određene srednje vrednosti, koji se obično dobija iz neregulisanog izvora jednosmernog napona (akumulatorske baterije, neupravljivi ispravljači, itd.) pretvaraju u jednosmeran napon druge srednje vrednosti. Napon na izlazu čopera se može regulisati tako da se pomoću čopera može dobiti regulisani izvor jednosmernog napona. Osim čopera, za dobijanje regulisanog jednosmernog napona postoje razni linearni elektronski regulatori jednosmernog napona i poluupravljivi ili potpuno upravljivi ispravljači. Međutim, svi ovi izvori regulisanog jednosmernog napona imaju u odnosu na čopere nekoliko bitnih nedostataka.

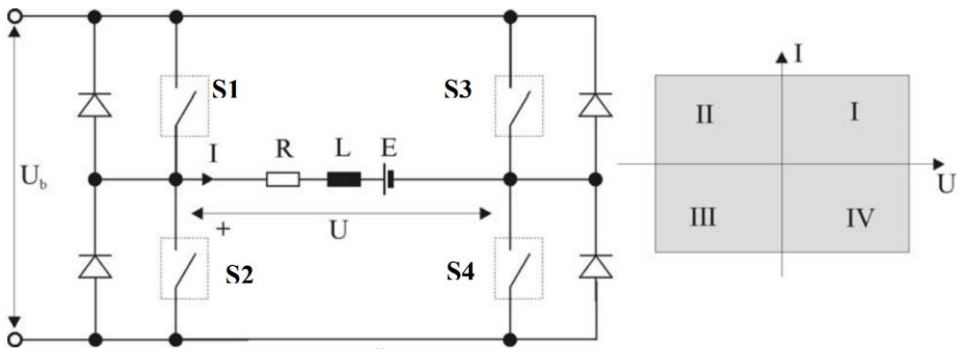
Linearni elektronski regulatori jednosmernog napona imaju zbog svog načina rada značajne gubitke energije, tako da se njihova primena isplati samo pri napajanju elektronskih uređaja vrlo malih snaga. Njihova prednost u odnosu na primenu čopera se ogleda u jednostavnosti i niskoj ceni.

Poluupravljivi i potpuno upravljivi ispravljači su, s druge strane isplativi i za napajanje potrošača velikih snaga, a dugo vremena su bili i jedina varijanta za dobijanje regulisanog jednosmernog napona iz mrežnog naizmeničnog napona. Otkako je postala aktuelna tema kvaliteta električne energije, došlo se do zaključka da svi ovi ispravljači unose dosta visok nivo harmonijskog izobličenja u elektroenergetsku mrežu. Njihov faktor snage vrlo varira u zavisnosti od režima rada, jer zavisi od ugla paljenja. Osim toga, ovi pretvarači omogućavaju nisku frekvenciju promene izlaznog napona, te za mašine jednosmerne struje, čiji armaturni namotaji generalno imaju malu induktivnost, oni predstavljaju loše rešenje zbog velike talasnosti izlazne struje.

Čoperi u odnosu na linearne regulatore jednosmernog napona imaju mnogo manje gubitke energije u toku rada. U odnosu na poluupravljive i potpuno upravljive ispravljače postižu mnogo veće prekidačke učestanosti, jer u svojoj topologiji sadrže prekidače tipa MOSFET-a ili IGBT-a, pa na taj način zahtevaju mnogo manje filtre. U kombinaciji sa neupravljivim ispravljačem za dobijanje jednosmernog napona iz mrežnog naizmeničnog napona, faktor snage je mnogo bolji nego kod poluupravljivih i potpuno upravljivih ispravljača koji su bazirani na tiristorima i GTO tiristorima.

Konačno, čoperi obezbeđuju mnogo manju talasnost struje armature, čime se ujedno umanjuje i talasnost ostvarenog momenta jednosmernog pogona. Primenom čopera je moguća i brza regulacija struje armature, čime se umnogome poboljšavaju dinamičke karakteristike pogona.

Četvorokvadrantni čoperi omogućavaju rad u bilo kom od četiri kvadranta na U-I karakteristici i upravo su oni korišćeni u ovom radu. Na slici 2.1.1 prikazani su topologija i kvadranti na U-I karakteristici.

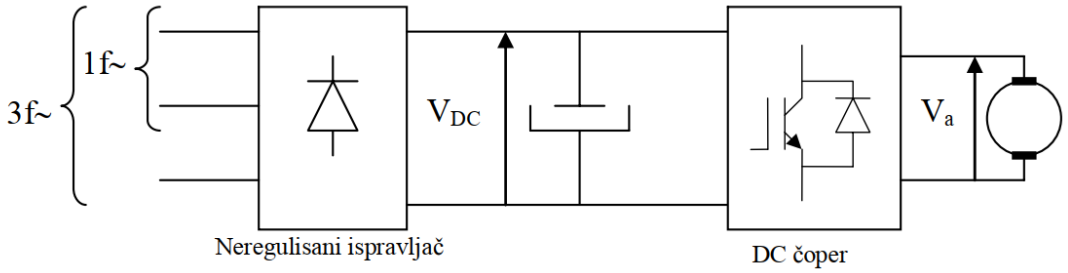


Slika 2.2.1 Četvorokvadrantni čoper (H-most)

Izvor: https://vdocuments.mx/primena-mikroprocesora-u-energetici-pogoni-jednosmerne-struje-.html

Na slici 2.2.1 potrošač je simbolično prikazan kao redna veza otpornika R, induktivnosti L i elektromotorne sile E koja simbolizuje, u slučaju mašine za jednosmernu struju, indukovanu ems – elektromotornu silu u namotaju. Tranzistori su simbolično prikazani kao idealni prekidači.

Četvorokvadrantni čoperi omogućavaju rad u sva četiri kvadranta na U-I karakteristici. Moguća su oba smera struje kroz potrošač i moguće je promeniti polaritet napona na potrošaču. Naročito dobra osobina je mogućnost promene polariteta napona na potrošaču bez upotrebe ikakvih preklopnih uređaja. Osnovna šema primene čopera data je na slici 2.2.2.



Slika 2.2.2 Osnovna energetska šema jednosmernog pogona sa čoperom

Izvor: https://vdocuments.mx/primena-mikroprocesora-u-energetici-pogoni-jednosmerne-struje-.html

Neregulisani ispravljač se priključuje na mrežu i služi za generisanje napona jednosmernog međukola. Za monofazni ispravljač napon jednosmernog DC kola koji napaja H-most iznosi:

$$V_{dc} = \sqrt{2} \cdot V_{nf} = \sqrt{2} \cdot 230 = 324[V], V_{nf} = 230[V] \quad \text{2.2.1}$$

gde je:

V_{dc}– napon DC kola ,

V_{nf}– nominalni fazni napon mreže.

Talasni oblik napona na izlazima četvorokvadrantnog čopera, i na potrošaču prikazan je na slici 2.2.3. Ovakav napon se dobija kada se naizmenično uključuju parovi prekidača S1, S4 i S2, S3 u H-mostu.

Srednja vrednost napona na izlazima četvorokvadrantnog čopera je:

$$V_{sr} = \frac{1}{T} \cdot \int_0^t u(t) \cdot dt = \frac{1}{T} \cdot \int_0^{ton} V_{dc} \cdot dt + \frac{1}{T} \cdot \int_{ton}^T -V_{dc} \cdot dt \quad \text{2.2.2}$$

$$V_{sr} = \frac{1}{T} \cdot V_{nf} \cdot t_{on} - \frac{1}{T} \cdot V_{dc} \cdot (T - t_{on}) = V_{dc} \cdot (2 \cdot a - 1) \quad \text{2.2.3}$$

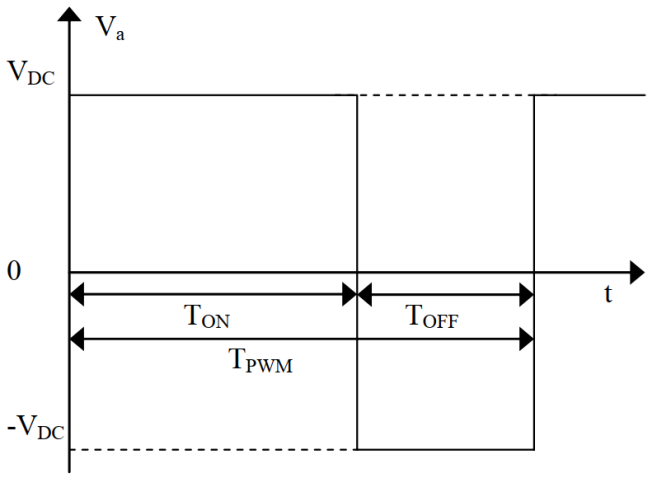
Faktor ispune se definiše kao:

$$a = \frac{t_{on}}{T} \quad \text{2.2.4}$$

gde je:

t_{on}– vreme propuštanja prekidača,

T – vremenska perioda PWM signala.



Slika 2.2.3 Talasni oblik napona na izlazu četvorokvadrantnog čopera

Izvor: https://vdocuments.mx/primena-mikroprocesora-u-energetici-pogoni-jednosmerne-struje-.html

Faktor ispune a = 0,5 kod četvorokvadratnog čopera dovodi do srednje vrednosti napona jednakoj nuli, odn. pretvarač radi kao monofazni invertor. Za faktore ispune preko 0.5, napon na armaturi motora je pozitivan, za faktore ispune manje of 0.5 , napon je negativan. Maksimalan pozitivan napon je +VDC, maksimum negativnog napona je -VDC. Ovo važi kada se noseći signal menja od nule do jedan.

Takođe, neophodne je da se ostvari dovoljno velika PWM učestanost

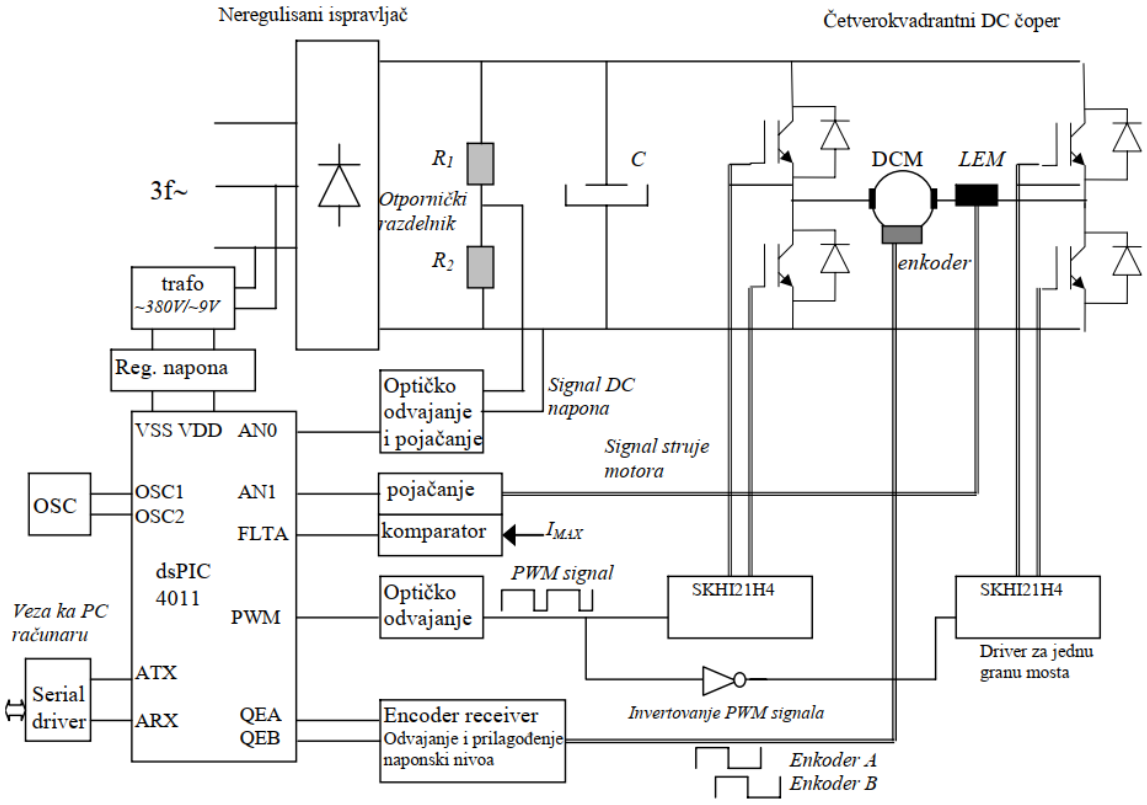
$$\frac{1}{f_{pwm}} = T_{pwm} \ll T_a = \frac{L_a}{R_a} \quad \text{2.2.5}$$

gde je:

f_{pwm}– frekvencija PWM signala.

U ovom slučaju se varijacija napona ±VDC ne prenosi značajno na struju armature. Nadalje, ni rotor mašine zbog inercije ne prati ove promene. Dakle, mašina vidi samo usrednjenu vrednost napona koju možemo menjati promenom faktora ispune. Time regulišemo struju i brzinu. Faktor ispune manji od 0,5 daje negativnu vrednost napona i menja smer obrtanja rotora bez dodavanja ikakvih preklopnih naprava.

2.3. Primena mikrokontrolera u pogonima jednosmerne struje sa čoperom



Slika 2.3.1 Blok dijagram regulisanog jednosmernog pogona sa četvorokvadrantnim čoperom

Izvor: https://vdocuments.mx/primena-mikroprocesora-u-energetici-pogoni-jednosmerne-struje-.html

Zadaci mikroprocesora:

- *Akvizicija podataka*. Merenje svih neophodnih veličina. Na primer, ukoliko želimo da ostvarimo tačan napon na motoru, moramo znati DC napon. Pritom, mereni signal je analogan i neophodno je izvršiti njegovo galvansko odvajanje, prilagođenje nivoa, i A/D konverziju.
- *Zaštita pogona.* Rad pogona je neophodno zaustaviti u slučaju prekomerne struje, brzine, u slučaju greške u naponu jednosmernog kola, prevelike temperature mosta ili motora.
- *Izvršenje digitalnog zakona upravljanja*. Zavisi od odabrane strukture regulacije. Može da bude a) prosta generacija PWM signala, b) regulacija brzine i c) regulacije struje i brzine.
- *Svetlosna signalizacija*. LED signalizacija rada, grešaka, itd...
- *Komunikacija sa spoljnim svetom*. Na slici 2.3.1 je prikazana serijska veza koja omogućuje prihvatanje komandnih signala, slanje izveštaja i prikupljenih podataka. Komandni signali se mogu zadavati i lokalno, primenom tastera i/ili potenciometra.

Neophodna merenja u digitalnom pogonu jednosmernog motora:

- Merenje napona jednosmernog međukola.
- Merenje struje motora. Osnovni razlozi merenja struje su zaštita i regulacija struje, tj. momenta. Da bi zaštita bila brza i efikasna, uobičajeno je da se realizuje hardverski, primenom komparatora. Ukoliko je struja veća od maksimalne signal na izlazu komparatora menja vrednost i *fault* ulaz trenutno zaustavlja generisanje PWM signala. Struja se može i konvertovati na AD ulazu, i digitalno regulisati u kontroleru. Ovim je obezbeđena bolja kontrola momenta.
- Merenje brzine. Neophodno da bi se zatvorila povratna sprega po brzini. Ukoliko se koristi DC tahogenerator, analogni signal je neophodno odvojiti i naponski nivo prilagoditi ulazu AD konvertora. Na slici 2.3.1 je prikazan primer u kome se brzina meri inkrementalnim enkoderom, koji generiše dva digitalna signala A i B. Za obradu ovih signala postoji QE modul unutar kontrolera, koji direktno izračunava brzinu. Ove signale je takođe neophodno odvojiti i prilagoditi im naponski nivo pre ulaska u kontroler.

Odabir prekidačke frekvencije rada čopera (PWM signala):

- Ispunjen uslov $f_s \gg \frac{1}{T_a} \Rightarrow$ mala talasnost struje, gubici u motoru i prekidni režim vođenja.
- Znatno veća od propusnog opsega brzinske petlje, bar 10 puta.
- Dovoljno visoka da je van slušnog opsega ($f_s > 16kHz$) a dovoljno niska da prekidački gubici ne budu previsoki. Izbor f_s je kompromis i zavisi od primene pogona kao i snage pogona.
- Znatno veća od rezonantnih frekvencija.
- Dovoljno niska da ne uđe u opseg transportnog kašnjenja i mrtvog vremena. Ovo je nepovoljno jer se pojavljuje nelinearnost u prenosnoj karakteristici i izobličenje struje.

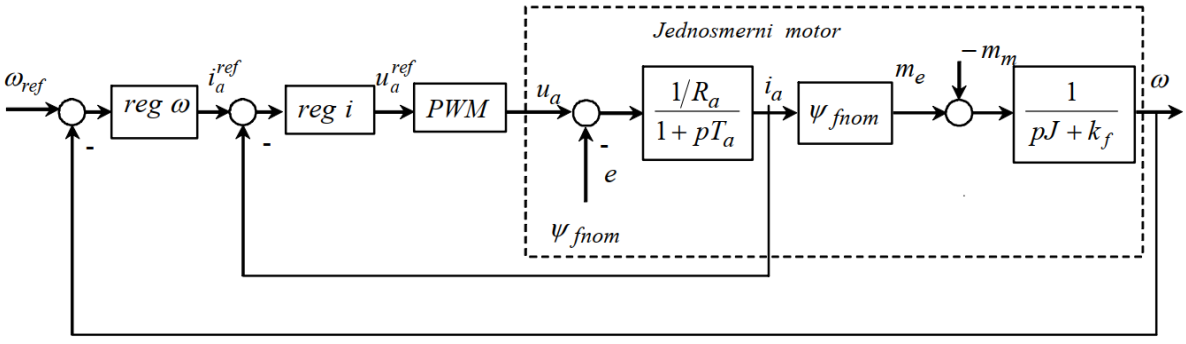
Izbor mikrokontrolera:

- Dovoljno brz (*sistemski clock*) da je moguće potrebne diskretne jednačine digitalnog zakona upravljanja izvršiti u realnom vremenu.
- Da ima dovoljno analognih ulaza. Potrebno je meriti napon DC kola i struju armature. Ukoliko je davač brzine DC tahogenerator, tada je neophodno rezervisati jedan analogni ulaz i za ovaj signal. Ukoliko se referentna brzina ne zadaje serijskom vezom, već lokalno preko potenciometra, još jedan dodatni analogni ulaz je neophodan.
- Ima PWM jedinicu sposobnu za generisanje adekvatnog PWM signala.
- Brojački ulaz. Ukoliko je na izlazu merača brzine digitalni signal, neophodno je da mikrokontroler poseduje brojački ulaz, koji može da radi u *counting* ili *capture* režimu. Prvi se koristi za merenje brzine na osnovu broja impulsa u toku konstantnog vremenskog intervala, a drugi se koristi za merenje vremenskog intervala između dva pridošla impulsa. Ukoliko je merenje brzine vršeno enkoderom (kao na slici), tada je najbolje da kontroler poseduje enkoderski kvadraturni ulaz (QE), koji direktno obrađuje standardne A i B signale sa enkodera.
- Dovoljno preostalih GPIO za ostale aktivnosti (LED, tasteri, upravljanje relejem, itd.).

2.4. Upravljačka struktura u pogonima jednosmerne struje

Regulacioni sistem za zadatak ima da obezbedi visoko kvalitetno upravljanje brzinom električne mašine u svim radnim režimima. Pod time se podrazumeva upravljanje koje obezbeđuje željeni karakter brzine u prelaznom režimu, veliku tačnost u ustaljenom stanju i malu osetljivost regulisane brzine na dejstvo poremećaja.

U ovom radu korišćen je regulacioni sistem sa zatvorenom povratnom spregom po brzini i struji, prikazan na slici 2.4.1. U ovakvom regulacionom sistemu regulator brzine eliminiše grešku između zadate i ostvarene brzine rotora zadajući odgovarajući momenat, tj. struju armature. Zatvaranje strujne povratne sprege umanjuje vreme uspostavljanja zadatog momenta, uvećava dinamički propusni opseg sistema i ostvaruje pogon sa izuzetno povoljnim statičkim i dinamičkim karakteristikama.



Slika 2.4.1 Regulacioni sistem sa zatvorenom povratnom spregom po brzini i po struji

Izvor: https://vdocuments.mx/primena-mikroprocesora-u-energetici-pogoni-jednosmerne-struje-.html

Šira klasa procesa u industriji se može kvalitetno aproksimirati kolom kašnjenja proizvoljnog reda čime se modeluje pojava konačne brzine prostiranja signala kroz sistem tj. inercija njegovih pojedinih podsistema. Za potrebe ove analize i iz razloga što najveći broj procesa u industriji iskazuje ponašanje koje se uz zadovoljavajući nivo preciznosti može opisati ponašanjem sistema prvog reda, dalja analiza će biti vršena posmatranjem procesa modelovanih tipom filtra prvog reda.

Veliki broj objekata se u praksi može opisati prenosnom funkcijom tipa filtra prvog reda. Sama priroda takvih objekata podazumeva da se oni u većini slučajeva pobuđuju signalom odskočne pobude. U elektromotornim pogonima kao primer se može navesti rotorsko kolo pogona mašine jednosmerne struje. Kolo RL tipa predstavlja filtar prvog reda kojim se modeluje elektromagnetna inercija konture namotaja prilikom uspostavljanja struje na zadatu vrednost napona u namotaju indukta. Od brzine i tačnosti uspostavljanja rotorske struje u kolu indukta direktno zavise dinamičke karakteristike promene brzine celokupnog pogona.

Uobičajeni blok dijagram digitalnog sklopa upravljanja sa jednim ulazom i jednim izlazom sastoji se od digitalnog regulatora koji upravlja kontinualnim objektom upravljanja. Između ovih delova sa jedne strane stoji tok odabiranja, a sa druge strane tok prevođenja odbiraka upravljačke promenljive u kontinualni signal.

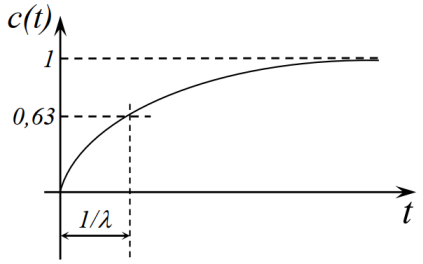
Upotrebom digitalnog PI regulatora sa pravilno podešenim pojačanjima mogu se obezbediti sve razmatrane performanse vezane za kvalitet željenog odziva procesne veličine od interesa:

- Stabilnost odziva.
- Eliminacija greške u stacionarnom stanju.
- Performanse tranzijentnog odziva spram tipa pobudnog signala.
- Osetljivost na promene parametara sistema.
- Sposobnost potiskivanja sistemskih poremećaja.
- Ponašanje sistema u graničnim režimima rada – limitima.

2.4.1. Sinteza PI regulatora struje

Postupak sinteze PI regulatora struje za upravljanu jednosmernu mašinu u ovom radu je sproveden primenom Dalinovog postupka.

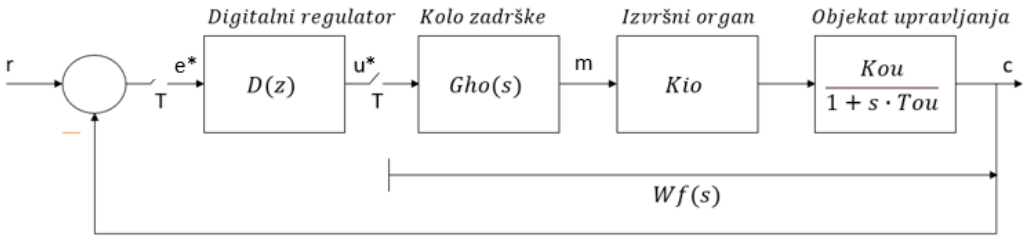
Dalinov postupak proračuna regulatora predstavlja metodu koja koristi opisni pristup u određivanju željenog odziva regulacione petlje, primenjuje se na sklopove u kojima se želi aperiodičan odziv. Naime, zadaje se željena brzina odziva petlje – ako se zadata vrednost skokovito promeni, upravljana promenljiva, $c(t)$, će imati aperiodičan odziv čija je brzina promene određena parametrom λ [Hz], prikazano na slici 2.4.1.1.



Slika 2.4.1.1 Zadati odskočni odziv upravljane promenljive

Izvor: http://www.ftn.uns.ac.rs/993541252/primena-dalinovog-proracuna

Na primer, ako je željena brzina odskočnog odziva takva da se 63% krajnje vrednosti dostigne za 1 ms, parametar λ ima vrednost 1000 Hz.



Slika 2.4.1.2 Izgled sklopa na koji se primenjuje Dalinov postupak

Elementi prikazani na slici 2.4.1.2 su:

r-referentni signal,

e-signal greške,

u-upravljački signal,

m-modulišući signal,

c-upravljani signal,

$D_{(z)}$-funkcija prenosa digitalnog regulatora,

$G_{ho(s)}$-funkcija prenosa kola zadrške nultog reda,

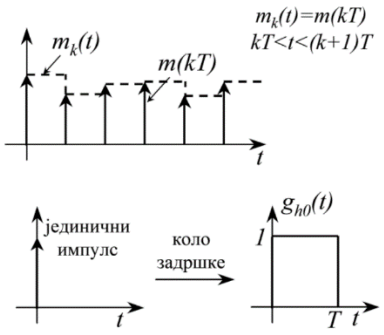
K_{io}– pojačanje izvršnog organa,

K_{ou}– pojačanje objekta upravljanja,

T_{ou}– vremenska konstanta objekta upravljanja i

$W_{f(s)}$ – funkcija prenosa kontinualnog dela.

Za prevođenje povorke odbiraka upravljačke promenljive na izlazu digitalnog regulatora u kontinualnu veličinu koja se dovodi na ulaz kontinualnog objekta upravljanja koristi se kolo zadrške nultog reda koje na svom izlazu drži vrednost prethodnog odbirka do dolaska novog. D/A pretvarač i impulsno širinski modulator su zapravo kola zadrške nultog reda. Rad kola zadrške prikazan je na slici 2.4.1.3



Slika 2.4.1.3 Način rada kola zadrške nultog reda

Izvor: http://www.ftn.uns.ac.rs/308004754/upravljanje-energetskim-pretvaracima---skripta

Oblik impulsnog odziva ovog kola $g_{ho}(t)$ dobija se kao:

$$g_{ho}(t) = h(t) - h(t-T) \quad \text{2.4.1.1}$$

gde h(t) predstavlja Hevisajdov signal.

Primenom Laplasove transformacije iz 2.4.1.1 sledi funkcija prenosa kola zadrške nultog reda Gho(s):

$$G_{h0}(s) = L[g_{h0}(t)] = \frac{1-e^{-sT}}{s} \quad \text{2.4.1.2}$$

Pretvarač energetske elektronike, koji vrši ulogu izvršnog organa, se može predstaviti proporcionalnim elementom, a strujni podsklop mašine, koji predstavlja objekat upravljanja, se modeluje elementom prvog reda (RLE kolo).

Primenom z transformacije na sklop sa slike 2.4.1.2 dobija se:

$$C(z) = Wf(z) \cdot D(z) \cdot [R(z) - C(z)] \quad \text{2.4.1.3}$$

Iz formule 2.4.1.3 može se izraziti prenosna funkcija digitalnog regulatora $D(z)$ kao:

$$D(z) = \frac{1}{Wf(z)} \cdot \frac{\frac{C(z)}{R(z)}}{1-\frac{C(z)}{R(z)}} \quad \text{2.4.1.4}$$

gde je:

$Wf(z)$ – diskretizovana funkcija prenosa kontinualnog dela,

$\frac{C(z)}{R(z)}$ – funkcija spregnutog diskretnog prenosa.

Proračun diskretizovane funkcije prenosa kontinualnog dela glasi:

$$Wf(z) = Z\left\{Gho(s) \cdot Kio \cdot \frac{Kou}{1+s \cdot Tou}\right\} = Z\left\{\frac{1-e^{-sT}}{s} \cdot Kio \cdot \frac{Kou}{1+s \cdot Tou}\right\} = \frac{K1}{z-e^{-\frac{T}{Tou}}} \quad \text{2.4.1.5}$$

$$K1 = Kio \cdot Kou \cdot \left(1 - e^{-\frac{T}{Tou}}\right) \quad \text{2.4.1.6}$$

Proračun funkcije spregnutog diskretnog prenosa se izvodi na osnovu željenog odskočnog odziva petlje. Aperiodičan odziv c(t) sa željenom brzinom promene izraženom pomoću ekvivalentne vremenske konstante $\frac{1}{\lambda}$ izgleda kao na slici 2.4.1.1.

Ako se uzme da $r(t)$ ima skok u nultom trenutku ($t = 0s$) sledi:

$$R(s) = \frac{1}{s} \quad \textit{2.4.1.7}$$

$$C(s) = \frac{1}{1+s\cdot\frac{1}{\lambda}} \cdot \frac{1}{s} \quad \textit{2.4.1.8}$$

Primenom z transformacije dobija se:

$$R(z) = \frac{z}{z-1} \quad \textit{2.4.1.9}$$

$$C(z) = \frac{z\cdot\left(1-e^{-\lambda T}\right)}{(z-1)(z-e^{-\lambda T})} \quad \textit{2.4.1.10}$$

Da bi se izačunala funkcija spregnutog prenosa potrebno je podeliti 2.4.1.8 sa 2.4.1.9 čime se dobija:

$$\frac{C(z)}{R(z)} = \frac{1-e^{-\lambda T}}{z-e^{-\lambda T}} \quad \textit{2.4.1.11}$$

Ubacivanjem izraza 2.4.1.5 i 2.4.1.11 u jednačinu 2.4.1.4, dolazi se do krajnjeg izraza za prenosnu funkciju digitalnog regulatora, koja glasi:

$$D(z) = \frac{1-e^{-\lambda T}}{K1}\left[e^{-\frac{T}{Tou}} + \left(1 - e^{-\frac{T}{Tou}}\right)\cdot\frac{1}{1-z^{-1}}\right] \quad \textit{2.4.1.12}$$

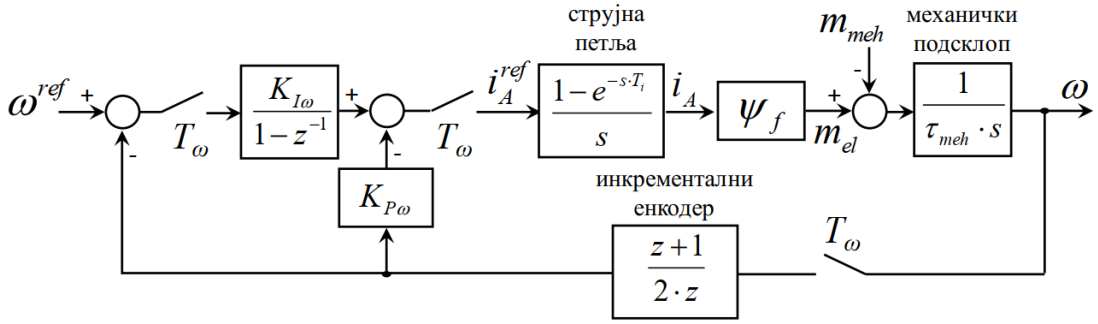
Jednačina 2.4.1.12 predstavlja funkciju PI regulatora sa proporcionalnim i integralnim dejstvom jednakim:

$$Kp = \frac{(1-e^{-\lambda T})}{K1}\cdot e^{-\frac{T}{Tou}} = \frac{1-e^{-\lambda T}}{Kio\cdot Kou\cdot(e^{\frac{T}{Tou}}-1)} \quad \textit{2.4.1.13}$$

$$Ki = \frac{1-e^{-\lambda T}}{K1}\cdot\left(1 - e^{-\frac{T}{Tou}}\right) = \frac{1-e^{-\lambda T}}{Kio\cdot Kou} \quad \textit{2.4.1.14}$$

2.4.2 Sinteza PI regulatora ugaone brzine

Sinteza PI regulatora ugaone brzine rotora električne mašine se zasniva na postupku pronalaženja optimalnih vrednosti parametara regulatora, $Kp\omega$ i $Ki\omega$, pri čemu je njegova postavka unapred određena kako je prikazano slikom 2.4.2.1.

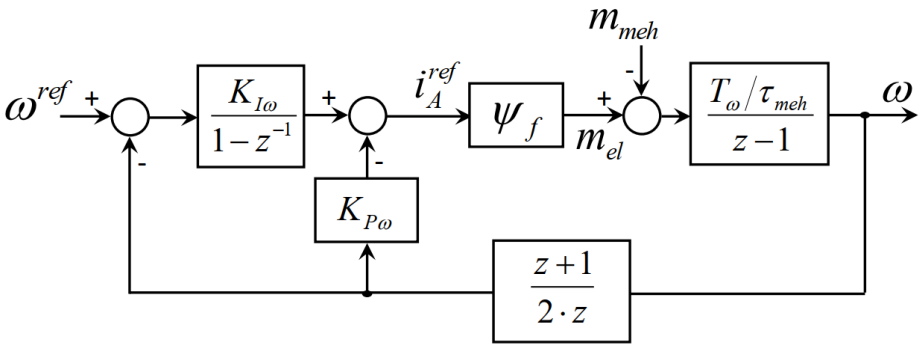


Slika 2.4.2.1 Regulaciona petlja ugaone brzine rotora

Izvor: http://www.ftn.uns.ac.rs/308004754/upravljanje-energetskim-pretvaracima---skripta

Pretpostavlja se da regulaciona petlja po brzini obuhvata prethodno podešenu regulacionu petlju po struji koja određuje elektromagnetni moment u pogonu mašine jednosmerne struje.

Podešena strujna petlja struje i_a predstavlja kolo zadrške nultog reda, pošto je njen ulaz, tj. izlaz regulatora brzine, zadata vrednost struje i_{aref}, a njen izlaz trenutna struja i_a.



Slika 2.4.2.2 Diskretizovana regulaciona petlja po brzini

Izvor: http://www.ftn.uns.ac.rs/308004754/upravljanje-energetskim-pretvaracima---skripta

Prenosna funkcija od ulaza ω do izlaza ω^{ref} regulacione petlje dobija se kao:

$$\frac{\omega(z)}{\omega^{ref}(z)} = \frac{\frac{Kiw}{1-z^{-1}} \cdot \Psi_f \frac{\frac{T_\omega}{\tau_{meh}}}{z-1}}{1+\Psi_f \frac{\frac{T_\omega}{\tau_{meh}}}{2} \cdot \frac{z+1}{z\cdot(z-1)} \cdot \left(\frac{Ki\omega \cdot z}{z-1} + Kp\omega\right)} \quad \text{.....2.4.2.1}$$

Imenioc prethodnog izraza predstavlja karakterističnu jednačinu sklopa $f(z)$:

$$f(z) = z^3 + (K^* \cdot Ki\omega + K^* \cdot Kp\omega - 2) \cdot z^2 + (1 + K^* \cdot Ki\omega) \cdot z - K^* \cdot Kp\omega \text{.....2.4.2.2}$$

Ako su z_1, z_2, z_3 polovi karakteristične jednačine ona se može prestaviti u obliku:

$$f(z) = (z - z_1) \cdot (z - z_2) \cdot (z - z_3) \quad 2.4.2.3$$

Neophodni izrazi za proračun parametara regulatora ugaone brzine proističu iz 2.4.2.1 uvođjenjem dodatnih konstanti i izjednačavanjem koeficijenata u polinomima 2.4.2.2 i 2.4.2.3 i oni glase:

$$K^* = \Psi_f \cdot (\frac{T_\omega}{\tau_{meh}})/2 \quad 2.4.2.4$$

$$K1 = K^* \cdot Kpw \quad 2.4.2.5$$

$$K2 = K^* \cdot Kiw \quad 2.4.2.6$$

$$z_1 \cdot z_2 \cdot z_3 = K1 \quad 2.4.2.7$$

$$z_1 \cdot z_2 + z_2 \cdot z_3 + z_1 \cdot z_3 = 1 \cdot K2 \quad 2.4.2.8$$

$$z_1 + z_2 + z_3 = 2 - K1 - K2 \quad 2.4.2.9$$

$$z_1 \cdot z_2 \cdot z_3 + z_1 \cdot z_2 + z_2 \cdot z_3 + z_1 \cdot z_3 + z_1 + z_2 + z_3 = 3 \quad 2.4.2.10$$

$$z_1 = z_2 = z_3 = z_p \quad 2.4.2.11$$

$$z_p{}^3 + 3 \cdot z_p{}^2 + z_p - 3 = 0 \quad 2.4.2.12$$

3. Implementacija

3.1. Odabir hardvera i razvojnog okruženja

Za izradu ovog diplomskog rada odabran je *dsPIC30F4011*, digitalni signalni kontroler (eng. *digital signal controller – DSC*), proizvođača Microchip Technology.

DSC kontroler predstavlja čip koji kombinuje kontrolne karakteristike mikrokontrolera (eng. *microcontroller – MCU*) sa računarskim sposobnostima digitalnog signalnog procesora (eng. *digital signal processor – DSP)* u samo jednom jezgru.

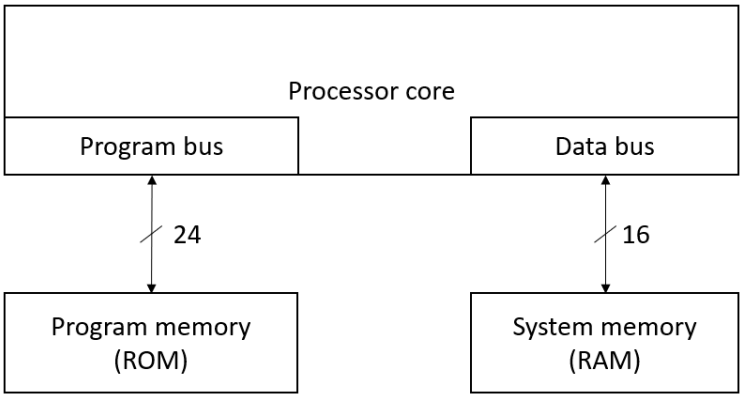
DSP su specijalni procesori čija je arhitektura optimizovana tako da može da podrži efikasnu obradu digitalnih signala dok su mikrokontroleri optimizovani u pravcu integracije većeg broja kola, upravljanja procesima u stvarnom vremenu, masovnu proizvodnju, nisku cenu i malu potrošnju struje.

Slika 3.1.1 Izgled dsPIC30F4011 digitalnog signalnog kontrolera

Izvor: https://www.microchip.com/wwwproducts/en/dsPIC30F4011

Korišćeni kontroler je 16-bitni kontroler čiji se radni takt kreće u opsegu od 30 do 120 MHz i poseduje 2 KB sistemske *RAM* (eng. *random access memory*) kao i 48 KB programske *ROM* (eng. *read only memory*) memorije. Unutar čipa integrisan je *DSP Engine* za matematički intenzivne operacije kao i *MAC* jedinica (eng. *Multiply-Accumulate*) za izvršavanje operacije množenja i sabiranja u jednom ciklusu. Ono što je zanimljivo jeste i mogućnost istovremenog preuzimanja više podataka ili instrukcija (eng. *dual data fetch*).

Zastupljena je Harvard arhitektura koja omogućuje 24-bitni instrukcijski opseg i magistralu podataka širine 16 bita koji zajedno sa *double fetch* mehanizmom omogućuju brže procesiranje instrukcija jer dsPIC procesor može preuzeti naredne željene podatke ili instrukcije za vreme izvršavanja trenutne instrukcije koja pristupa sistemskoj RAM memoriji.

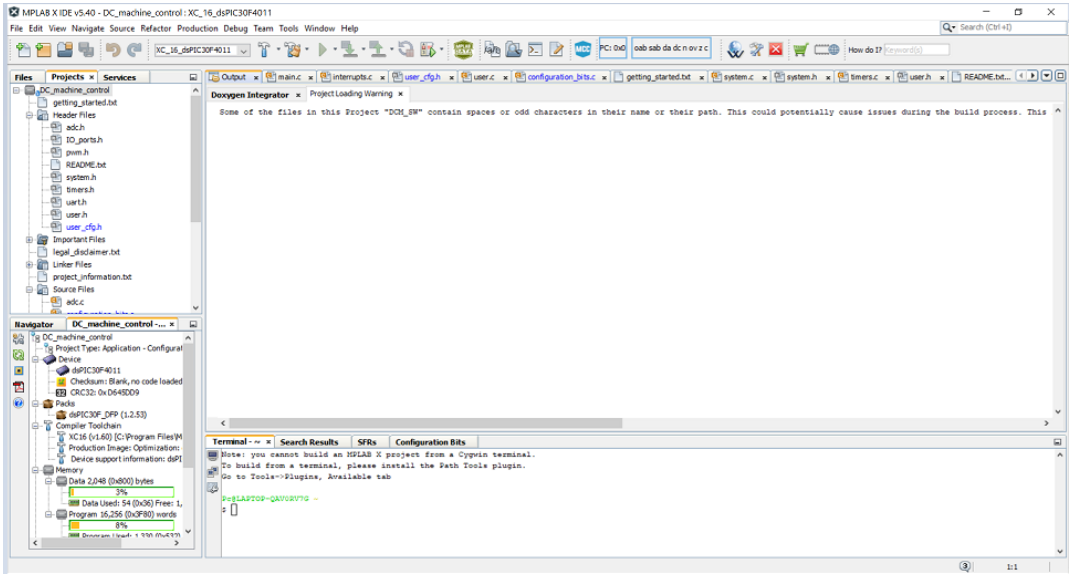


Slika 3.1.2 Hardvard arhitektura zastupljena u dsPIC30F4011 kontroleru

Korišteni kontroler poseduje:

- Pet 16-bitnih i dva 32-bitna tajmera.
- Tri SPI (eng. *Serial Peripheral Interface)* modula.
- Jedan I^2C (eng. *I-squared-C*) modula.
- Dva UART (eng. *Universal Asyncronous Receiver-Transmitter*) modula.
- Jedan CAN (eng. *Controller Area Network*) modul.
- Jedan Motor Control PWM modul sa 6 PWM kanala.
- Jedan QEI (eng. *Quadrature Encoder Interface)* modul.
- 10-bitni AD (eng. *Analog to Digital*) konverter.
- Četiri *Sample and Hold* kanala.

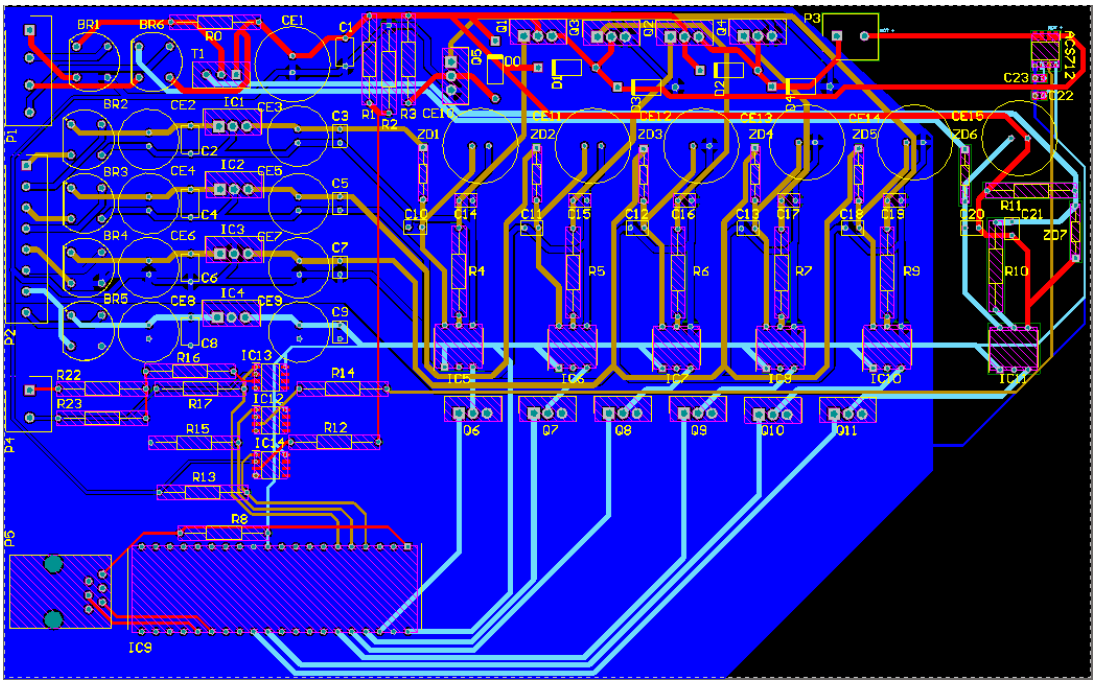
Za programiranje *dsPIC*30*F*4011 kontrolera korišćeno je integrisano razvojno okruženje “*MPLAB X IDE v5.40*”, kreirano od strane firme *Microchip Tehnology*. *MPLABX* predstavlja proširivi, vrlo konfigurabilni softverski program koji omogućuje razvoj softvera za većinu mikročipovih kontrolera i digitalnih signalnih kontrolera.



Slika 3.1.3 Integrisano razvojno okruženje “*MPLAB X IDE v5.40*”

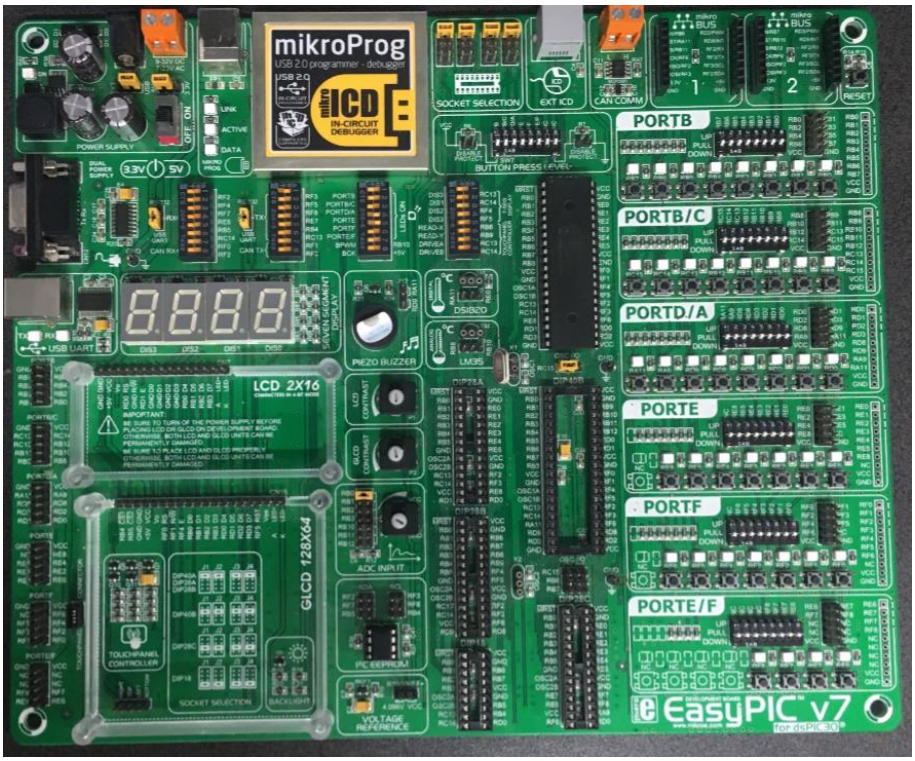
Programski jezik korišćen za kreiranje softvera u ovom radu je C programski jezik. Čitav kod sačuvan je na online repozitorijumu i može mu se pristupiti koristeći sledeći link: https://github.com/Velbl/DCM_SW. Motivacija za ovim jeste brža i efikasnija integracija studenata sa kodom i razvojnim okruženjem. Za kreiranje repozitorijuma korišćen je *GitHub*, servis koji omogućuje brže i efikasnije upravljanje razvojem softvera.

Kao hardversko okruženje dizajnirana je PCB (eng. *printed circuit board*) ploča namenjena upravo za svrhe ovog rada. Inicijalni dizajn PCB ploče prikazan je na slici 3.1.4. Dizajn PCB ploče u celokupnosti je izvršen koristeći *Altium Designer*. Dizajn same ploče i svi šematici mogu se takodje pronaći na *GitHub* repozitorijumu koristeći sledeći link: https://github.com/Velbl/DCM_HW.



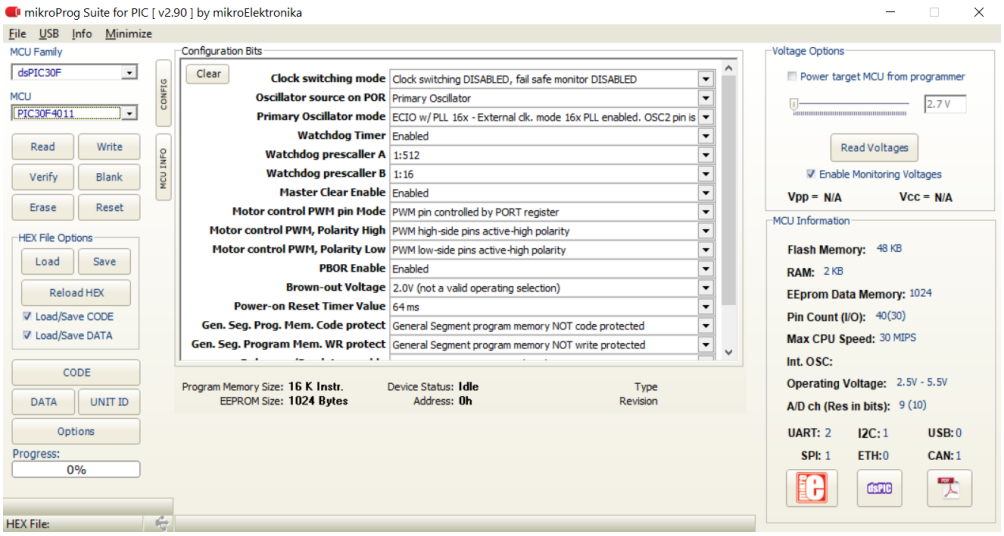
Slika 3.1.4 Dizajnirano hardversko okruženje za upravljanje DC motorom sa povratnom spregom po brzini i struji, upotrebom četvorokvadrantnog čopera

Ploča sa slike 3.1.4 u trenutku pisanja i odbrane rada nije bila izrađena. Usled čega se za svrhe razvoja softvera koristila razvojna ploča “*EasyPIC v7 for dsPIC30*“ prikazana na slici 3.1.5.



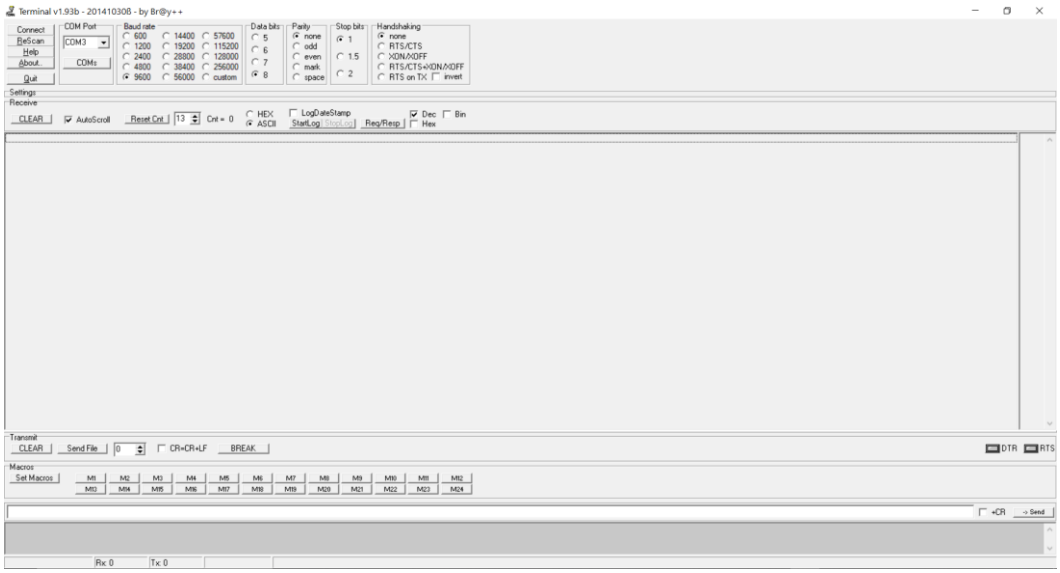
Slika 3.1.5 Razvojno okruženje “*EasyPIC v7 for dsPIC30*”

Za programiranje *dsPIC30F4011* kontrolera na razvojnom okruženju sa slike 3.1.5 korišćen je “*mikroProg Suite For PIC*”, program koji obezbeđuje jednostavno programiranje *Microchip* mikrokontrolera.



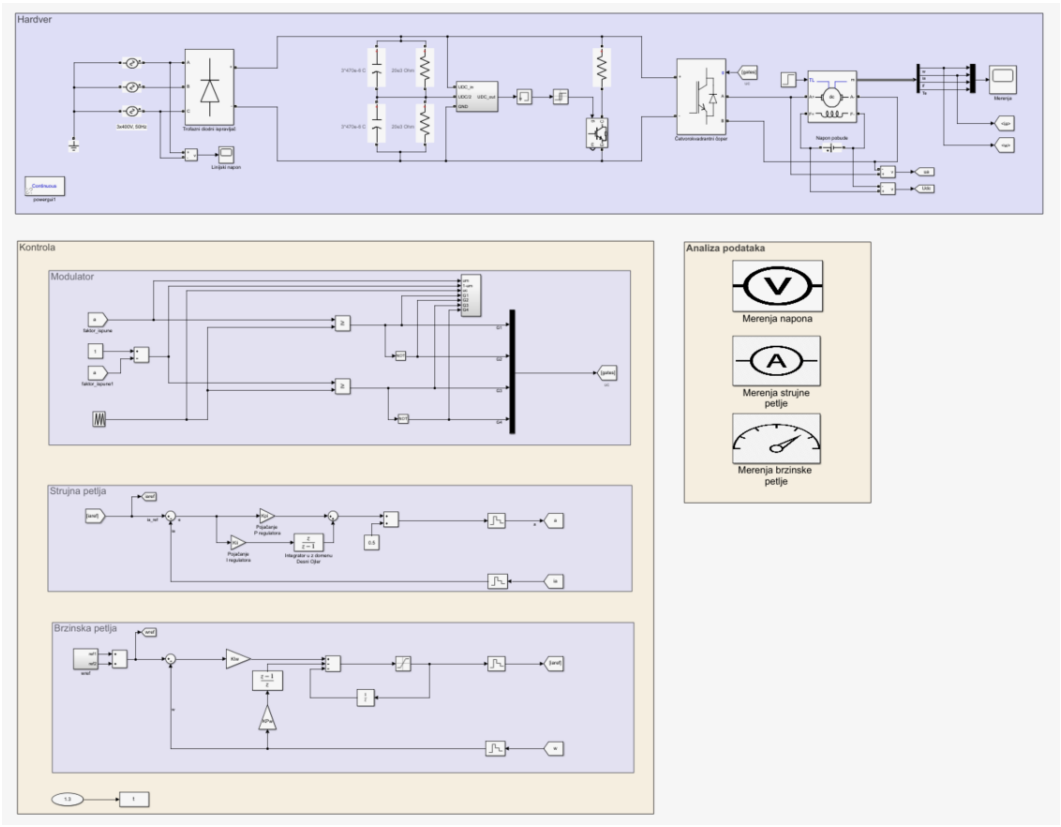
Slika 3.1.6 Aplikacija “*mikroProg Suite For PIC*”

Komunikacija između računara i razvojnog okruženja sa slike 3.1.5 ostvarena je preko UART komunikacionog protokola. Za prikaz željenih podataka na računaru, u svrhe testiranja, korišćena je aplikacija pod nazivom “*Terminal*”.



Slika 3.1.7 Aplikacija “*Terminal*”

Simulacioni model kreiran za testiranje proračunatih parametara PI regulatora kreiran je u Matlab Simulink-u - grafičko programsko okruženje zasnovano na MATLAB-u za modeliranje, simulaciju i analizu višedomenskih dinamičkih sistema. Njegov primarni interfejs je grafički alat za blok dijagrame i prilagodljivi skup biblioteka blokova prikazan na slici 3.1.8.



Slika 3.1.8 Simulacioni model kreiran za testiranje proračunatih parametara PI regulatora u grafičkom programskom okruženju “*Matlab/Simulink*”

3.2. Proračun parametara regulatora

3.2.1. Strujni regulator

Vrednosti parametara koji su neophodni za proračun proporcionalnog i integralnog dejstva regulatora struje armature:

- $\lambda = \frac{5}{T_a} = 1000\ Hz$
- $T_a = 4.43\ ms$
- $T_i = 0.5\ ms$
- $Kio = 2 \cdot U\_dc = 1200\ V$
- $Kou = \frac{1}{R_a} = 1.33\ s$
- $K = Kio \cdot Kou = 1600\ A$

gde je:

λ - ekvivalentna vremenska konstanta,

T_a- elektromagnetna vremenska konstanta indukta,

T_i– definisan period na kojem se izvršava strujna petlja (regulator armaturne struje),

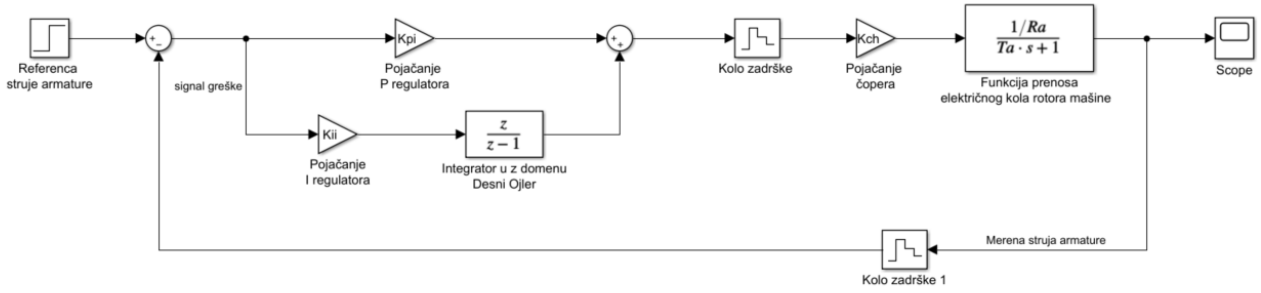
Kio – pojačanje izvršnog organa (četvorokvadrantni čoper),

Kou - pojačanje objekta upravljanja,

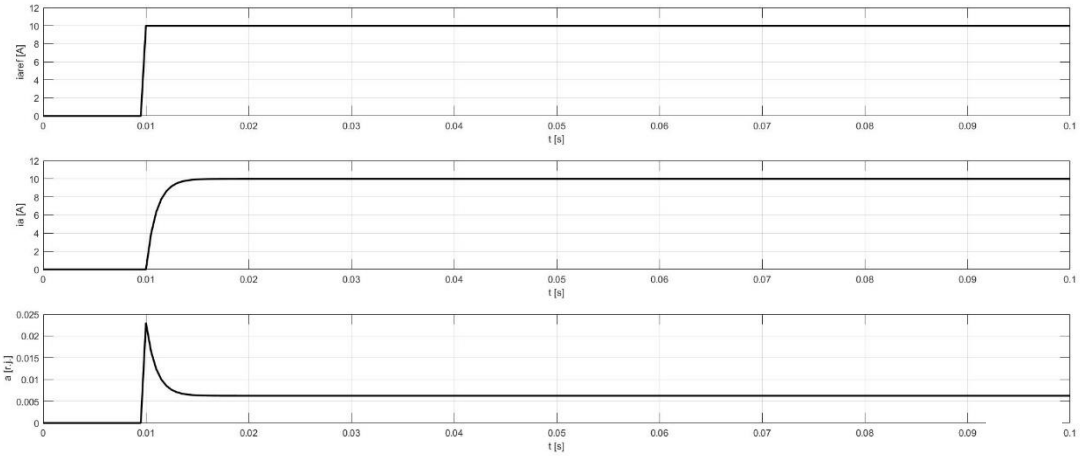
K – pojačanje u direktnoj grani.

Uvrštanjem iznad navedenih parametara u izraze 2.4.1.13 i 2.4.1.14 dobijamo vrednosti parametara regulatora struje rotora:

- $Kpi = 0.002344$
- $Kii = 0.0002465$



3.2.1.1 Uprošten model regulacione petlje struje armature jednosmerne mašine sa proporcionalnim dejstvom u direktnoj grani



3.2.1.2 Odzivi uproštenog modela regulacione petlje struje armature jednosmerne mašine

Na slici 3.2.1.2 može se primetiti da u trenutku zadavanja referentne vrednosti struje ($t = 0.01s$), greška između merene i zadate vrednosti struje naglo raste što dovodi do trenutnog povećanja faktora ispune. Povećanje faktora ispune dovodi do naglog povećanja trenutne vrednosti struje čime se brže dostiže referentna vrednost struje i samim tim brže eliminiše signal greške. Nakon dostizanja zadate vrednosti struje sistem odlazi u stacionarno stanje gde je greška stacionarnog stanja uspešno eliminisana upotrebom PI regulatora.

3.2.2. Brzinski regulator

Vrednosti parametara koji su neophodni za proračun proporcionalnog i integralnog dejstva regulatora ugaone brzine:

$\Psi_f = 0.7\ Wb$

$T_\omega = \frac{T_i}{10} = 5\ ms$

$\tau_{meh} = J = 0.02 kgm^2$

$K^* = 0.0875$

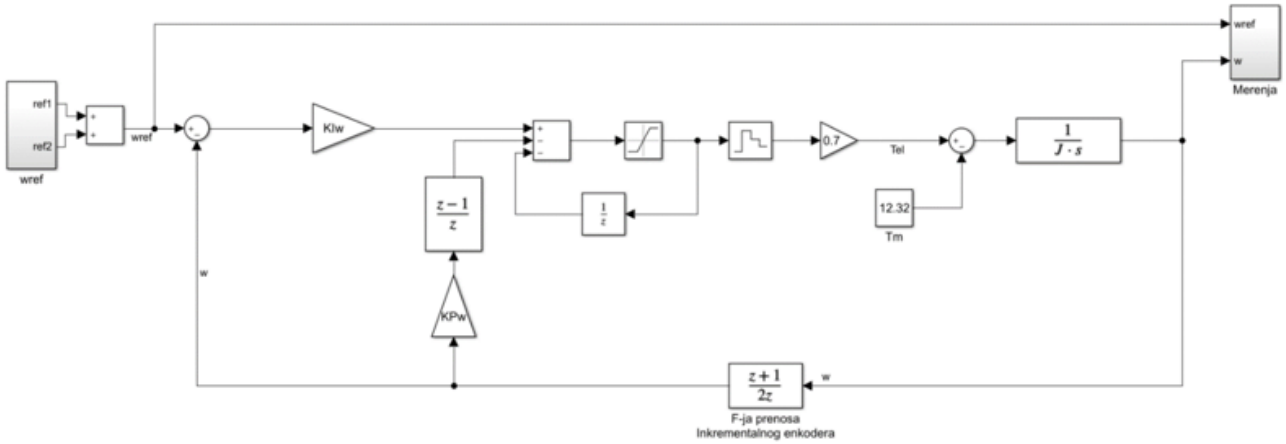
$K1 = 0.203$

$K2 = 0.035$

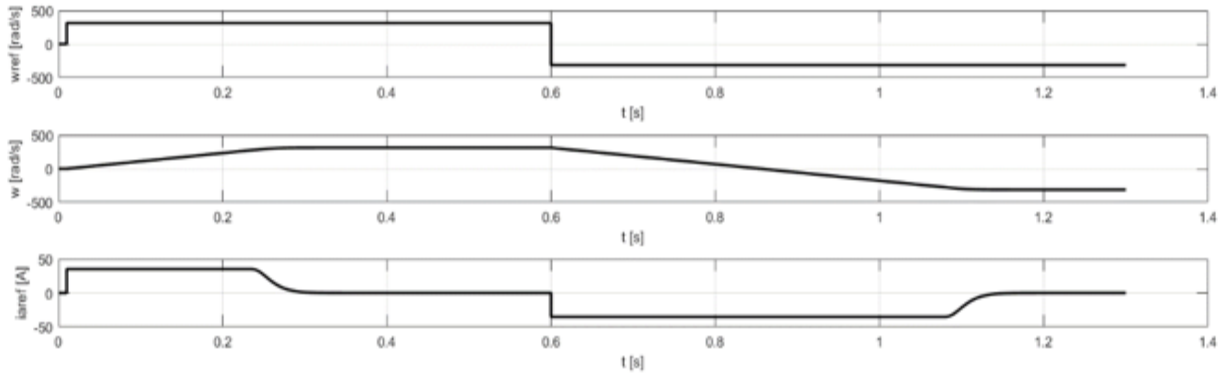
Uvrštanjem navedenih parametara $K1, K2, K^*$ u jednačine 2.4.2.1 i 2.4.2.2 dobijaju se parametri regulatora ugaone brzine:

$$Kpw = \frac{K1}{K^*} = 2.32$$

$$Kiw = \frac{K2}{K^*} = 0.4$$



3.2.2.3 Uprošten model regulacione petlje brzine rotora jednosmerne mašine sa onemogućenim namotavanjem

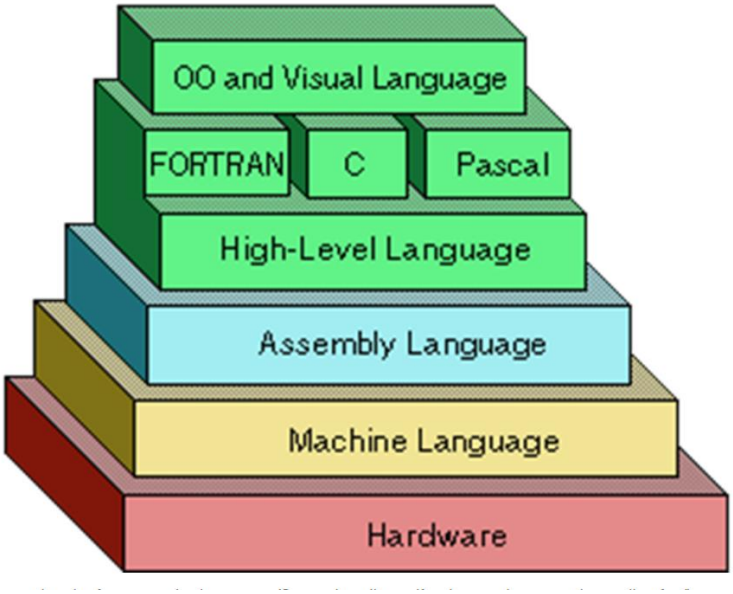


3.2.2.4 Odzivi uproštenog modela regulacione petlje brzine rotora jednosmerne mašine

Na slici 3.2.2.4 može se primetiti da u trenutku zadavanje referentne vrednosti brzine ($t = 0.01s$) referentna vrednost struje odlazi u njeno ograničenje. Prilikom dostizanja zadate vrednosti brzine, referentna struja se spušta na vrednost neophodnu za održavanje željene brzine, na ovoj slici to je nominalna vrednost pri nominalnoj brzini od $353\,rad/s$. Nakon dostizanja zadate vrednosti brzine, zadaje se negativna vrednost brzine koja doprinosi porastu struje u suprotnom smeru i sličnom ponašanju. Greška stacionarnog stanja eliminisana je korišćenjem PI regulatora, kao i u slučaju sa strujnom petljom.

Sa slika 3.2.1.2 i 3.2.2.4 može se primetiti da su pravilnom sintezom digitalnih PI regulatora uspešno obezbeđene sve razmatrane performanse vezane za kvalitet željenog odziva procesne veličine od interesa, spomenute u poglavlju 2.4.

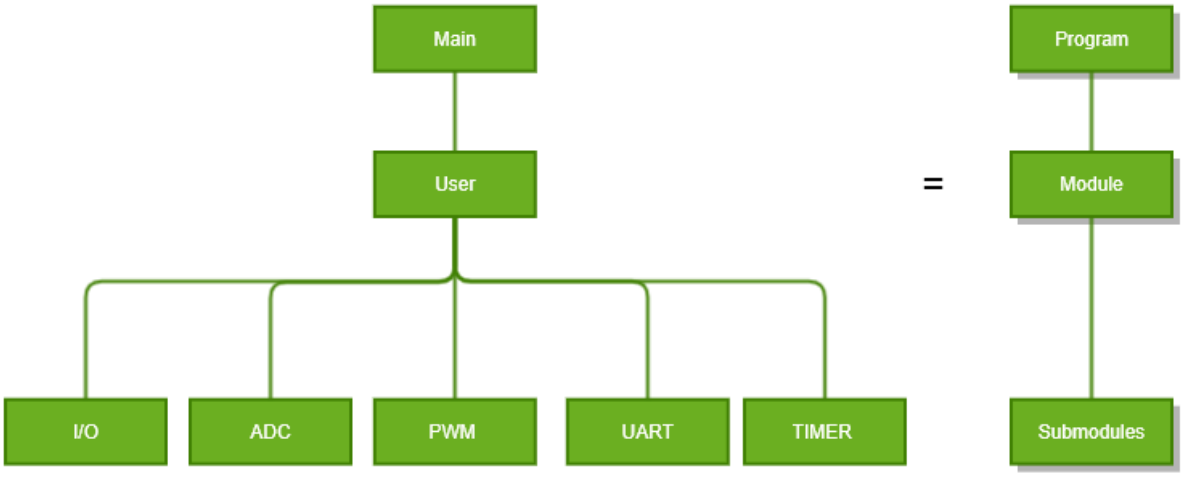
3.3. Arhitektura softvera



Slika 3.3.1 Osnovni nivoi programskih jezika

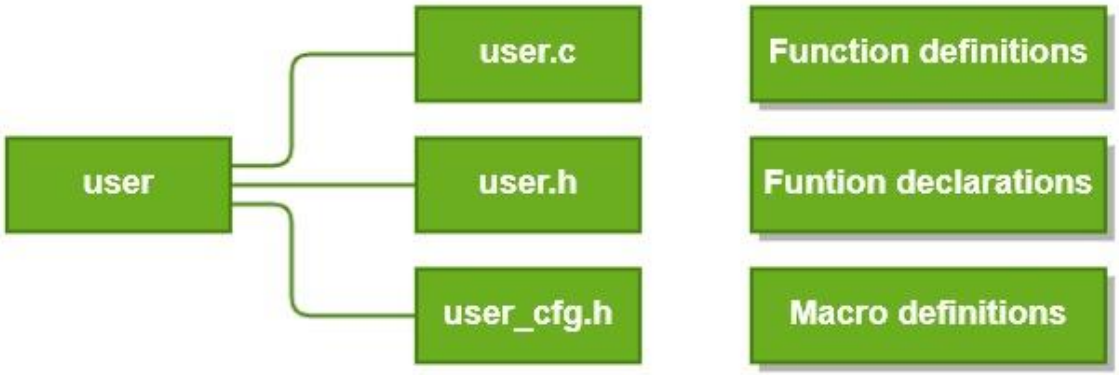
Kao što se vidi sa slike 3.3.1, za razvoj programa neophodno je pogodno hardversko okruženje na kojem će se izvršavati glavni program (eng. *main program*). Hardver i razvojno okruženje su opisani u poglavlju 3.1 dok će u ovom poglavlju akcenat biti na arhitekturi softvera.

Po svojoj arhitekturi program je podeljen na korisnički i platformski deo, prikazano na slici 3.3.2.



Slika 3.3.2 Arhitektura programa

Korisnički deo programa (eng. *user module*) je deo u kome se čuvaju sve funkcije i podaci koji su korišćeni direktno prilikom izvršavanja glavnog programa (eng. *main program*). Organizovan je tako da poseduje dva h fajla (eng. *header file)* i jedan c fajl (eng. *source file*). Jedan h fajl je namenjen za konfiguraciju u kojoj se čuvaju sve enumeracije i makroi kako bi se izbeglo direktno unošenje podataka u program prilikom njegovog izvršavanja, poznatije kao *hard-coding*. U drugom h fajlu nalaze se deklaracije[i] globalnih funkcija i deklarisane strukture podataka. U c fajlu nalaze se definicije[ii] svih funkcija za potrebe izvršavanja glavnog programa.



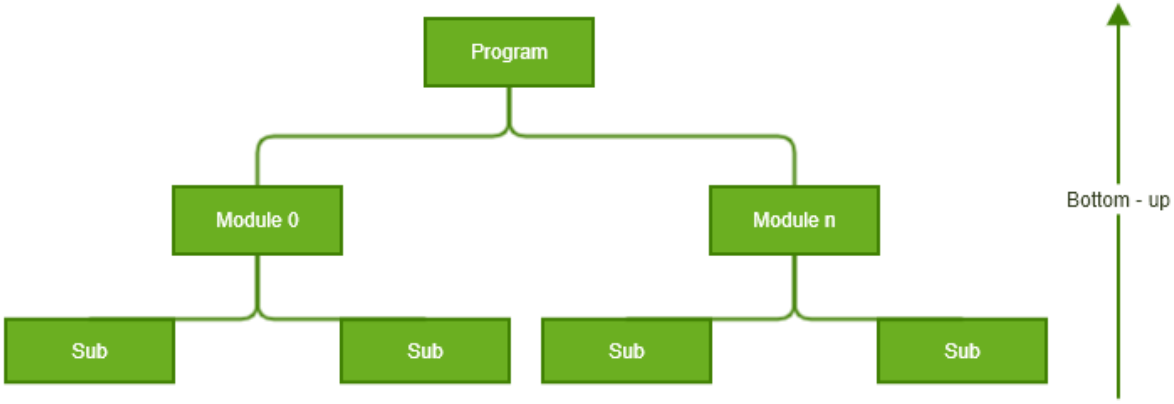
Slika 3.3.3 Arhitektura modula unutar korisničkog dela programa

Platformski deo programa kreiran je od skupa podmodula. Svaki podmodul čini jednu periferiju sa sopstvenom konfiguracijom. Podmodul je organizovan tako da poseduje jedan h i jedan c fajl. Kao i u prethodnom slučaju, h fajl je namenjen za deklaraciju globalnih funkcija, poznatije kao *interfejsi,* koji omogućuju vezu sa korisničkim delom programa i ostalim periferijama dok se u c fajlu nalazi željena konfiguracija određene periferije kao i definicija svih funkcija.

Slika 3.3.4 Arhitektura podmodula unutar platformskog dela programa

3.4. Dizajn softvera

C programski jezik predstavlja proceduralni jezik u kojem je najčešće primenjivan “*Top – down*” pristup dizajniranju softvera. Uprkos tome, u ovom radu je primenjen “*Bottom – up*” češće korišćen u programskim jezicima koji su po stilu programiranja objektno – orijentisani (npr. C++, Java, Python). “*Bottom – up*” pristup je korišćen zato što je autoru ovog rada to predstavljalo logičniji i intuitivniji način dizajniranja softvera.



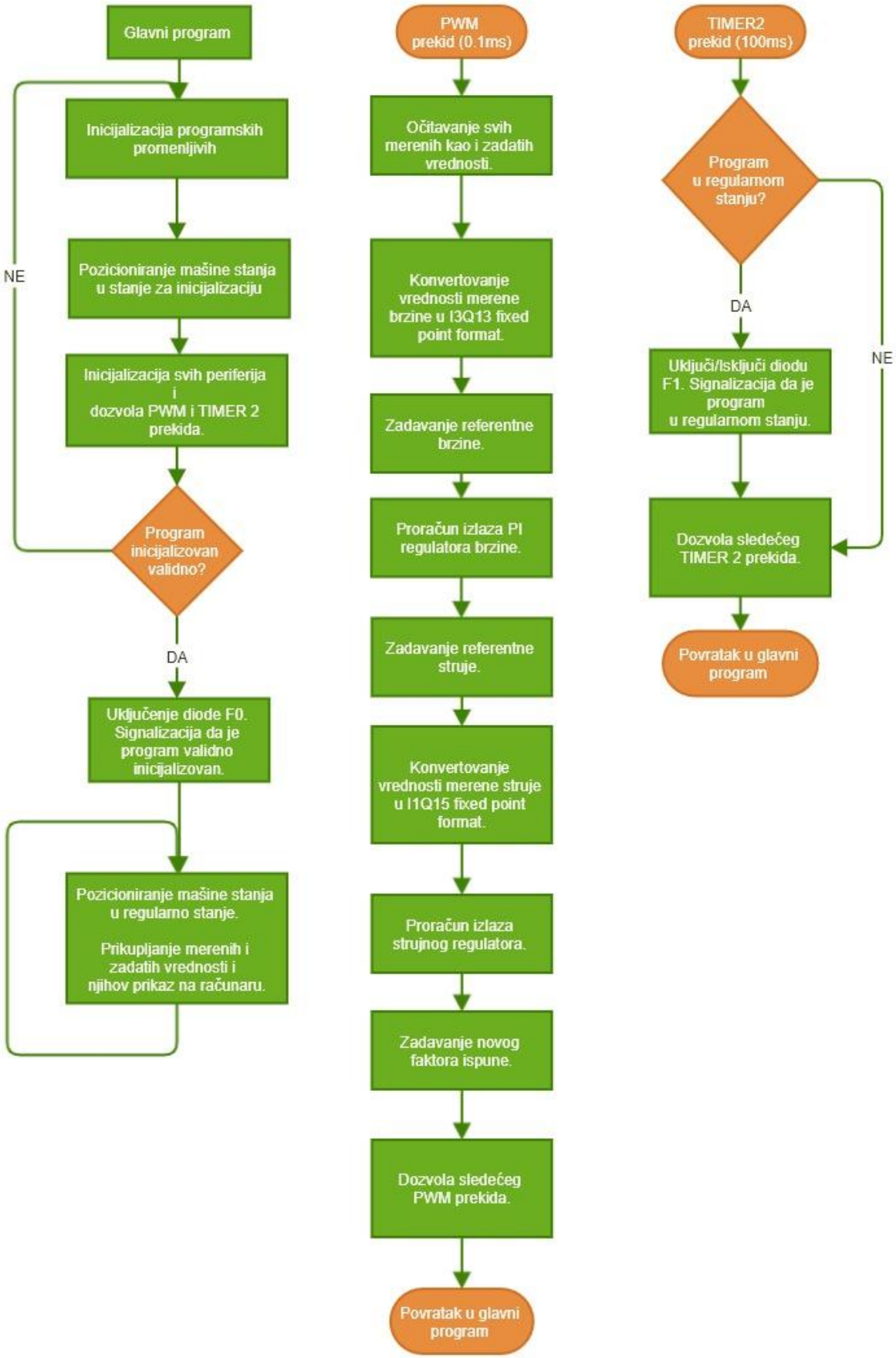
Slika 3.4.1 Bottom – up pristup dizajniranja softvera

Na slici 3.4.2 dat je pojednostavljeni blok dijagram algoritma za upravljanje brzinom jednosmernog motora.

Blok dijagram algoritma se sastoji iz četiri osnovna dela:

- Inicijalizacija nakon DSP reseta - Inicijalizacija Motor Control Modul-a koji omogućuje PWM izlaze, inicijalizacija UART, ADC i TIMER periferije, kao i svih korišćenih portova. Takođe, u ovom delu izvršena je i signalizacija da je sistem pravilno inicijalizovan uključenjem diode na portu Fo.
- Glavni program u kome se izvršavaju programski zadaci za koje nije kritičan početak, kao ni vreme izvršenja.
- PWM prekid koji se dešava svakih 0.1 *ms* i u kojem se čitaju sve veličine, rešava digitalni zakon upravljanja i definiše novi odgovarajući faktor ispune.
- TIMER2 prekid na svakih 100 *ms* koji se ponaša kao prekidač podešen da naizmenično uključuje/isključuje diodu na portu F1.

U nastavku će biti detaljnije opisan svaki od navedenih delova algoritma.

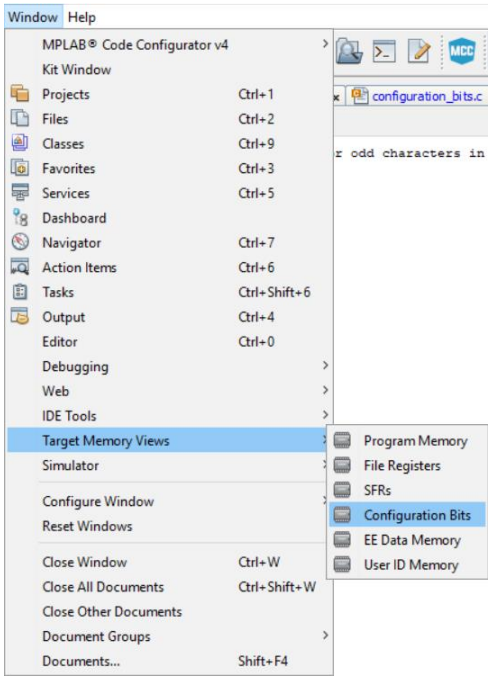


Slika 3.4.2 Blok dijagram algoritma upravljanja DC motorom sa povratnom spregom po brzini i struji

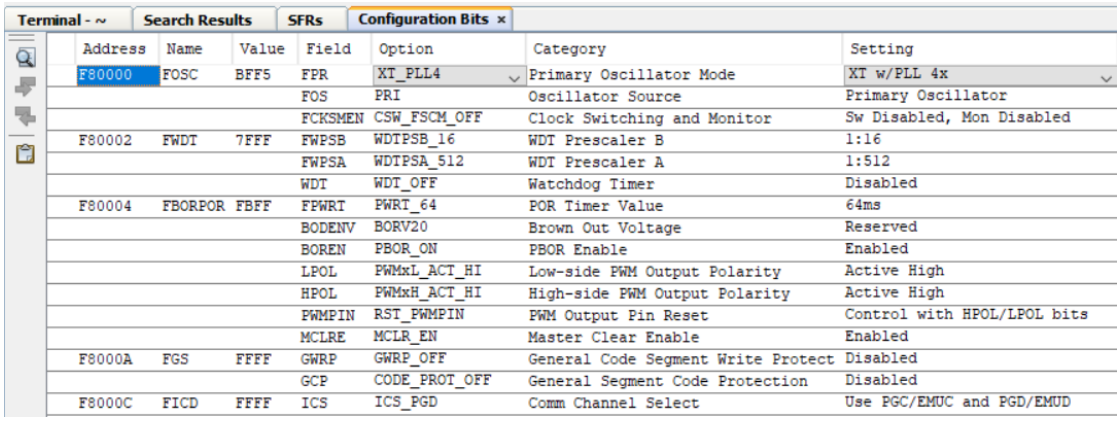
3.4.1. Kod za inicijalizaciju mikrokontrolera

Po resetu mikrokontrolera se izvršava upravo ovaj deo koda. U njemu je neophodno podesiti rad mikrokontrolera, pravilnim podešavanjem konfiguracionih bitova.

Podešavanje konfiguracionih bitova izvršeno je u dva koraka, prikazanim na slikama 3.4.1.1 i 3.4.1.2.



Slika 3.4.1.1 Putanja do opcije za podešavanje konfiguracionih bitova



Address	Name	Value	Field	Option	Category	Setting
F80000	FOSC	BFF5	FPR	XT_PLL4	Primary Oscillator Mode	XT w/PLL 4x
			FOS	PRI	Oscillator Source	Primary Oscillator
			FCKSMEN	CSW_FSCM_OFF	Clock Switching and Monitor	Sw Disabled, Mon Disabled
F80002	FWDT	7FFF	FWPSB	WDTPSB_16	WDT Prescaler B	1:16
			FWPSA	WDTPSA_512	WDT Prescaler A	1:512
			WDT	WDT_OFF	Watchdog Timer	Disabled
F80004	FBORPOR	FBFF	FPWRT	PWRT_64	POR Timer Value	64ms
			BODENV	BORV20	Brown Out Voltage	Reserved
			BOREN	PBOR_ON	PBOR Enable	Enabled
			LPOL	PWMxL_ACT_HI	Low-side PWM Output Polarity	Active High
			HPOL	PWMxH_ACT_HI	High-side PWM Output Polarity	Active High
			PWMPIN	RST_PWMPIN	PWM Output Pin Reset	Control with HPOL/LPOL bits
			MCLRE	MCLR_EN	Master Clear Enable	Enabled
F8000A	FGS	FFFF	GWRP	GWRP_OFF	General Code Segment Write Protect	Disabled
			GCP	CODE_PROT_OFF	General Segment Code Protection	Disabled
F8000C	FICD	FFFF	ICS	ICS_PGD	Comm Channel Select	Use PGC/EMUC and PGD/EMUD

Slika 3.4.1.2 Prozor za podešavanje konfiguracionih bitova

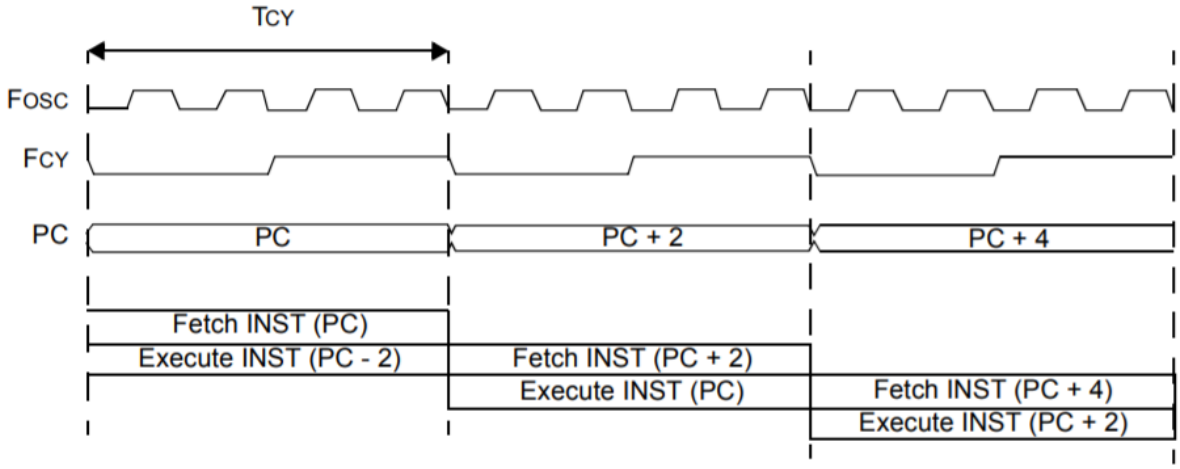
Nakon podešavanja željenih konfiguracionih bitova generisan je novi c fajl pod nazivom *configuration_bits.c*.

Konfiguracioni bitovi od interesa za potrebe ovog rada su:

1. *FOS* i *FPR* bitovi koji služe za odabir izvora kao i za odabir režima rada primarnog oscilatora. Njegovim podešavanjem omogućen je odabir željene PLL (eng. *Phase-Locked Loop*) komponente čija je uloga da generiše tačan izlazni signal frekvencije jednak ili višestruki od frekvencije ulaznog signala.
   Izvor oscilatora je podešen da bude spoljašnji kristal od 8 MHz sa uvećanjem ulazne frekvencije za četiri puta čime su izjednačene ulazna frekvencija oscilatora (*FOSC)* sa instrukcijskom frekvencijom (*FCY*). Ovim postupkom omogućeno je izvršenje 8 miliona instrukcija u sekundi.
   Odnos ulazne frekvencije i frekvencije jedne instrukcije kod dsPIC mikrokontrolera dat je u formuli 3.4.1.1 kao i na slici 3.4.1.3.

$$FCY = \frac{(FOSC \cdot PLL)}{4} \quad \text{.....3.4.1.1}$$

U formuli 3.4.1.1 broj 4 (delilac) predstavlja broj ciklusa potreban za izvršenje jedne asemblerske instrukcije.



Slika 3.4.1.3 Odnos ulaznog takta i instukcijskog takta kod dsPIC mikrokontrolera

2. *LPOL* i *HPOL* bitovi koji služe za podešavanje polarnosti izlaznih signala gornjih i donjih polovina korišćenih grana. U ovom radu, izlazni signali aktivnih grana su podešeni na jedinicu (eng. *High Polarity*).

3.4.2. Kod za konfiguraciju pojedinih periferija

Usled težnje ka modularnosti programa, svaka periferija je konfigurisana u zasebnom fajlu. U poglavlju 3.3 na slici 3.3.2 mogu se uočiti kreirani fajlovi za svaku periferiju kao i njihove veze za aplikativnim delom programa dok je na slici 3.3.4 prikazana unutrašnja organizacija fajlova vezanih za određenu periferiju.

U aplikativnom delu, unutar *user.c* fajla, kreirana je funkcija *dsPIC30F4011_v_Config* koja je prosleđena glavnom programu prilikom inicijalizacije i sadrži konfiguracije svih potrebnih periferija, prikazana na slici 3.4.2.1.

```
void dsPIC30F4011_v_Config()
{
  Pins_v_Config();
  Timers_v_Config();
  PWM_v_Config();
  ADC_v_Config();
  UART_v_Config();
}
```

Slika 3.4.2.1 Funkcija za konfiguraciju svih korišćenih periferija

Ovakvim pristupom izuzetno je jednostavno dodati konfiguraciju nove periferije ili izmeniti već korišćenu. U narednom tekstu će biti opisane konfiguracije svih korišćenih periferija potrebnih za upravljanja DC motorom u ovom radu.

Pins_v_Config() funkcija unutar koje su pozvane konfiguracije svih korišćenih ulaznih/izlaznih pinova.

```
void Pins_v_Config()
{
  v_GPIOPins();
  v_ControlPins();
  v_SensorPins();
}
```

Slika 3.4.2.2 Podela konfiguracije pinova

v_GPIOPins() funkcija unutar koje su konfigurisani svi GPIO (eng. *General Purpose Input Output Pins*), u ovom radu to su pinovi za signalizaciju pravilne inicijalizacije i rada u normalnom režimu rada, prikazano na slici 3.4.2.3.

```
static void v_GPIOPins()
{
/***************************TEST PINS*****************************************/
  TRISFbits.TRISF0 = 0u;  // Set RF0 pin as output.
  TRISFbits.TRISF1 = 0u;  // Set RF1 pin as output.
  TRISFbits.TRISF2 = 0u;  // Set RF2 pin as output.
  TRISFbits.TRISF3 = 0u;  // Set RF3 pin as output.
  TRISFbits.TRISF4 = 0u;  // Set RF4 pin as output.
  TRISFbits.TRISF5 = 0u;  // Set RF5 pin as output.
/*****************************************************************************/
}
```

Slika 3.4.2.3 Konfiguracija GPIO pinova

v_ControlPins() funkcija unutar koje su konfigurisani PWM pinovi i pinovi planirani za kontrolu nad mosfetom za disipaciju struje vraćene od strane motora ka čoperu (rekuperaciju) kao i nad tiristorom za regulisanje polazne struje.

```
static void v_ControlPins()
{
/*****************************PWM PINS*****************************************/
  TRISEbits.TRISE0 = 0u;  // Set RE0 pin as output.(PWM1L)
  TRISEbits.TRISE1 = 0u;  // Set RE1 pin as output.(PWM1H)
  TRISEbits.TRISE2 = 0u;  // Set RE2 pin as output.(PWM2L)
  TRISEbits.TRISE3 = 0u;  // Set RE3 pin as output.(PWM2H)
/******************************************************************************/

  TRISEbits.TRISE4 = 0u;  // Set RE4 pin as output.Pin for MOSFET control.
  TRISEbits.TRISE5 = 0u;  // Set RE5 pin as output.Pin for Thyristor control.
}
```

Slika 3.4.2.4 Konfiguracija kontrolnih pinova

v_SensorPins() funkcija za konfiguraciju svih analognih pinova namenjenih za merenja i za zadavanje referentne brzine.

```
static void v_SensorPins()
{
/*****************************SENSOR PINS**************************************/
  TRISBbits.TRISB0 = 1u;  // Set RB0 pin as input.(current)
  TRISBbits.TRISB1 = 1u;  // Set RB1 pin as input.(voltage)
  TRISBbits.TRISB2 = 1u;  // Set RB2 pin as input.(speed)
  TRISBbits.TRISB3 = 1u;  // Set RB3 pin as input.(ref current/speed)
/******************************************************************************/
}
```

Slika 3.4.2.5 Konfiguracija ulaznih pinova za merenje signala sa senora

Timers_v_Config() funkcija u kojoj je podešen tajmer 2 tako da generiše prekid na svakih 100 *ms* kako bi bila ostvarena signalizacija prilikom pravilnog rada glavnog programa. Konfigurisan je i tajmer 1, međutim bez dozvole prekida, kao opcija za njegovu kasniju lakšu integraciju, ako za to bude potrebe.

```
void Timers_v_Config()
{
  v_ConfigTimer1();
  v_ConfigTimer2();
}
```

```
static void v_ConfigTimer2()
{
  T2CONbits.TCS   = 0;      // Internal clock (FOSC/4)
  T2CONbits.TCKPS = 3;      // Prescaler 1:256
  // PR = (Wanted Time) / (FOSC / TCS); TCS = FOSC/4
  PR2 = 3076;               // 100ms
  _T2IP = 2;                // Priority level = 3
  _T2IF = 0;                // Flag status cleared.
  __write_to_IEC(IEC0bits.T2IE = 1); // Enable IP.
  T2CONbits.TON   = 1;      // Turn on Timer 2.
}
```

Slika 3.4.2.6 Konfiguracija tajmera

UART_v_Config() funkcija za konfiguraciju UART periferije namenjene za komunikaciju sa računarom. Brzina slanja podataka je podešena na 9600 bps, dužina podatka za slanje je podešena na 8 bita i prilikom slanja podataka se ne proverava bit parnosti. UART prekid nije dozvoljen. Pored konfiguracije, za ovu periferiju kreirane su funkcionalnosti koje se koriste direktno u glavnom programu kao što su *UART_v_Write(), UART_v_NewLine(), UART_v_Print()* i *UART_v_IsBusy(),* korišćene za pojednostavljeno slanje podataka ka računaru i njihov uređen ispis na terminalu.

```
void UART_v_Config()
{
  // U1BRG = ( ((FCY/Desired baud rate)/16) - 1 )= (((8000000/9600)/16) - 1)
  U1BRG = 52;                 // Baud Rate set to 9600bps.

  U1MODEbits.UARTEN = 0u;     // Disable UART.

/*******************************U1MODE register*******************************/
  U1MODEbits.USIDL  = 1u;     // Discontinue operation when device enters Idle mode.
  U1MODEbits.ALTIO  = 0u;     // UART communicates using UxTX and UxRX I/O pins.
  U1MODEbits.ABAUD  = 0u;     // Auto-Baud disabled
  U1MODEbits.PDSEL  = 0x00u;  // No Parity, 8-Data bits.
  U1MODEbits.STSEL  = 0u;     // 1-Stop bit.
/*****************************************************************************/

/*******************************UART INTERRUPT CONFIG*************************/
  U1STAbits.UTXISEL = 0u;     // Interrupt after one TX character is transmitted from
U1TXB to U1TSR.Transmit buffer has at least one empty word.
  IFS0bits.U1TXIF   = 0u;     // Clear interrupt flag.
  IEC0bits.U1TXIE   = 0u;     // Disable UART TX interrupt
  IPC2bits.U1TXIP   = 1u;     // UART interrupt priority.
/*****************************************************************************/

  U1MODEbits.UARTEN = 1;      // Enable UART

  U1STAbits.UTXEN   = 1u;     // Enable UART TX.
}
```

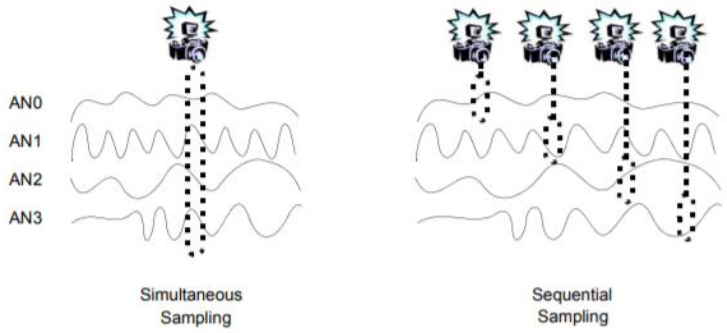
Slika 3.4.2.7 Konfiguracija UART periferije

```
static void v_Config()
{
  ADCON1bits.ADON   = 0u;    // Turn of A/D convertor.

/*******************************ADCON1 register*******************************/
  ADCON1bits.ADSIDL = 1u;    // Discontinue module operation when device enters Idle mode.
  ADCON1bits.FORM   = 0x00u; // Integer (DOUT = 0000 00dd dddd dddd)
  ADCON1bits.SSRC   = 0x03u; // Motor Control PWM interval ends sampling and starts
conversion.
  ADCON1bits.SIMSAM = 1u;    // Samples CH0, CH1, CH2, CH3 simultaneously (when CHPS = 1x).
  ADCON1bits.ASAM   = 1u;    // Sampling begins immediately after last conversion
completes. SAMP bit is auto set.
/*****************************************************************************/

/*******************************ADCON2 register*******************************/
  ADCON2bits.VCFG  = 0x00u;  // Set voltage references to AVdd and AVss.
  ADCON2bits.CHPS  = 0x03u;  // Converts CH0, CH1, CH2 and CH3.
  ADCON2bits.BUFM  = 1u;     // Buffer configured as two 8-word buffers ADCBUF(15...8),
ADCBUF(7...0).
/*****************************************************************************/

/*******************************ADCON3 register*******************************/
  ADCON3bits.SAMC  = 0x01u;  // Auto-Sample time = 1Tad.
  ADCON3bits.ADRC  = 0u;     // Clock derived from system clock (8MHz).
  //ADCS = (2*TADmin/Tcy) - 1    = (2*(154ns/125ns)) - 1  = 2
  //TAD = ( Tcy*(ADCS + 1) / 2 ) = ( 125ns*(2+1) / 2 )    = 187ns -> (TAD > TADmin)
  ADCON3bits.ADCS  = 0x2u;
/*****************************************************************************/

/*******************************ADCHS register********************************/
  ADCHSbits.CH123NA = 0x00u; // CH1, CH2, CH3 negative input is VREF.
  ADCHSbits.CH123SA = 0u;    // CH1 positive input is AN0, CH2 positive input is AN1, CH3
positive input is AN2.
  ADCHSbits.CH0NA   = 0u;    // CH0 negative input is VREF.
  ADCHSbits.CH0SA   = 0x0Du; // CH0 positive input is AN3.
/*****************************************************************************/

/*******************************ADPCFD register*******************************/
  ADPCFGbits.PCFG0 = 0u;     // Set AN0 (RB0) as analog input pin.
  ADPCFGbits.PCFG1 = 0u;     // Set AN1 (RB1) as analog input pin.
  ADPCFGbits.PCFG2 = 0u;     // Set AN2 (RB2) as analog input pin.
  ADPCFGbits.PCFG3 = 0u;     // Set AN3 (RB3) as analog input pin.
/*****************************************************************************/

  IEC0bits.ADIE    = 0u;     // A/D interrupt disabled.

  ADCON1bits.ADON  = 1u;     // Turn on A/D converter.
}
```

Slika 3.4.2.8 Konfiguracija ADC periferije

Sa slike 3.4.2.8 može se primetiti sledeće:

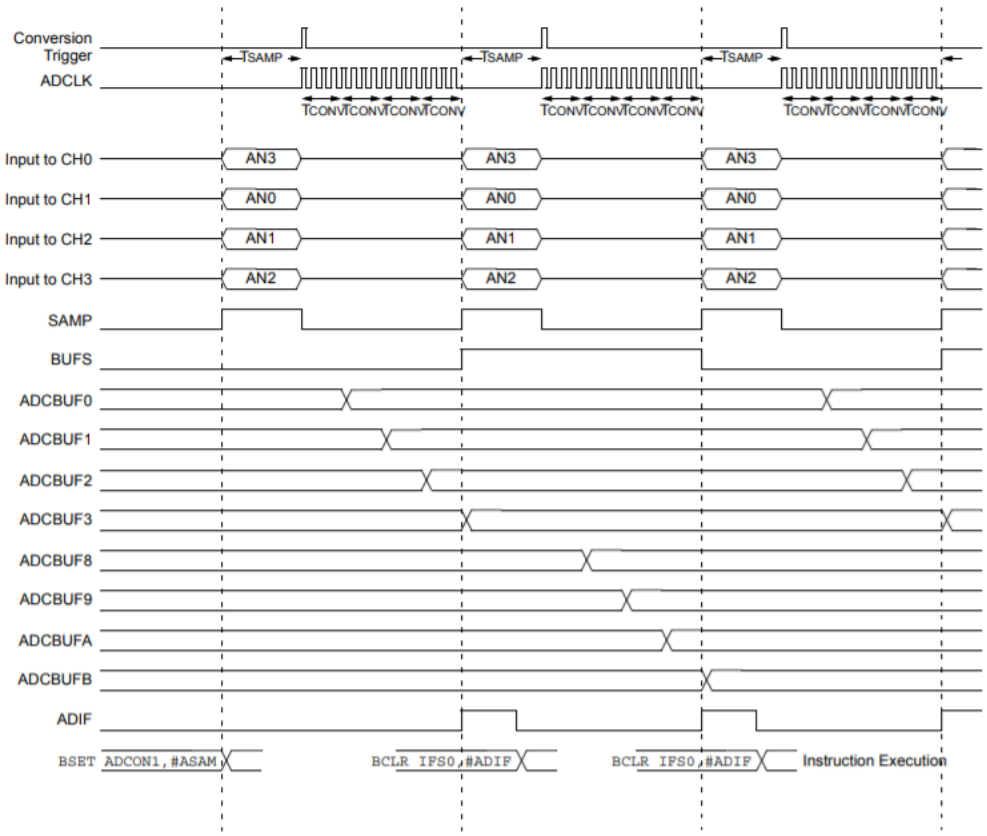
- Upotreba četiri S/H (eng. *Sample and Hold*) kanala. Kanali CH0, CH1, CH2 i CH3.
- *Motor Control PWM* je zadužen za prekidanje odabiranja (eng. *sampling*) i započinjanje naredne konverzije podataka.
- Svi S/H kanali podešeni tako da rade istovremeno (simultano) - mogućnost rada sa podacima odabranim u isto vreme, prikazano na slici 3.4.2.9.



Slika 3.4.2.9 Simultano odabiranje analognih signala

Izvor: https://ww1.microchip.com/downloads/en/DeviceDoc/70064E.pdf

- Odabiranje novih podataka počinje odmah po završetku poslednje konverzije.
- A/D takt (Tad) iznosi 187 ns. A/D konverzija zahteva 12 A/D taktova.
- *BUFM* bit je podešen na „1“. Ako procesor ne može da oslobodi bafer unutar vremena potrebnog za jednu A/D konverziju, BUFM bit bi trebao biti „1“ jer će se tada osam konverzija učitati u prvih osam bafera, nakon čega dolazi do prekida. Sledećih osam konverzija će se učitati u ostalih osam bafera. Procesor će imati sve vreme između prekida za pomeranje osam konverzija. Prikazano na slici 3.4.2.11
- RB0, RB1, RB2 i RB3, prethodno definisani kao ulazni pinovi (slika 3.4.2.5) su ovde definisani kao analogni pinovi na koje je planirano da se povežu željeni senzori. Veze između analognih pinova i S/H kanala su sledeće:
  - RB0 – CH1
  - RB1 – CH2
  - RB2 – CH3
  - RB3 – CH0



Slika 3.4.2.10 Prikaz simultane A/D konverzije sa konfigurisana četiri S/H kanala

Izvor: https://ww1.microchip.com/downloads/en/DeviceDoc/70064E.pdf

Buffer Address	Buffer @ 1st Interrupt	Buffer @ 2nd Interrupt
ADCBUF0	AN3 sample 1	
ADCBUF1	AN0 sample 1	
ADCBUF2	AN1 sample 1	
ADCBUF3	AN2 sample 1	
ADCBUF4		
ADCBUF5		
ADCBUF6		
ADCBUF7		
ADCBUF8		AN3 sample 2
ADCBUF9		AN0 sample 2
ADCBUFA		AN1 sample 2
ADCBUFB		AN2 sample 2
ADCBUFC		
ADCBUFD		
ADCBUFE		
ADCBUFF		

Slika 3.4.2.11 Prikaz popunjavanja ADC bafera sa podešenim *BUFM* bitom na jedinicu

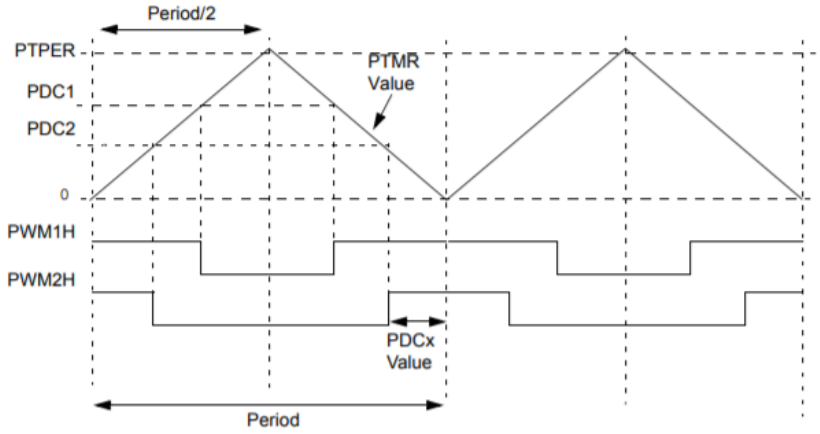
Izvor: https://ww1.microchip.com/downloads/en/DeviceDoc/70064E.pdf

```
static void v_Config(uint16_t u_Period)
{
  PTCONbits.PTEN     = 0u;    // PWM time base is OFF.
  PTPER = u_Period;           // Set PWM frequency to 10KHz.

/*****************************PTCON register*******************************/
  PTCONbits.PTSIDL   = 1u;    // PWM time base halts in CPU Idle mode.
  PTCONbits.PTOPS    = 0x00u; // 1:16 Post scale.
  PTCONbits.PTCKPS   = 0x00u; // PWM time base input clock period is Tcy (1:1 pre scale).
  PTCONbits.PTMOD    = 0x02u; // PWM time base operates in a continuous up/down.
/**************************************************************************/

/*****************************PWMCON1 register*****************************/
  PWMCON1bits.PMOD1  = 0u;    // PWM I/O pin pair is in the complementary output mode.
  PWMCON1bits.PMOD2  = 0u;    // PWM I/O pin pair is in the complementary output mode
  PWMCON1bits.PEN1H  = 1u;    // PWM1H pin is enabled for PWM output.
  PWMCON1bits.PEN1L  = 1u;    // PWM1L pin is enabled for PWM output.
  PWMCON1bits.PEN2H  = 1u;    // PWM2H pin is enabled for PWM output.
  PWMCON1bits.PEN2L  = 1u;    // PWM2L pin is enabled for PWM output.
/**************************************************************************/

/*****************************PWMCON2 register*****************************/
  PWMCON2bits.SEVOPS = 0x01u; // PWM Special Event Trigger Output Post scale bit
 is set to 1.
  PWMCON2bits.OSYNC  = 1u;    // Output overrides via the OVDCON register are
synchronized to the PWM time base.
  PWMCON2bits.UDIS   = 0u;    // Updates from duty cycle and period buffer registers
are enabled.
/**************************************************************************/

  DTCON1bits.DTA    = 0x00u;  // Clock period for Dead Time Unit A is TCY.

/*****************************OVDCON register******************************/
  OVDCONbits.POVD1H = 1u;     // Output on PWM1H I/O pin is controlled by the PWM
generator.
  OVDCONbits.POVD1L = 1u;     // Output on PWM1L I/O pin is controlled by the PWM
generator.
  OVDCONbits.POVD2H = 1u;     // Output on PWM2H I/O pin is controlled by the PWM
generator.
  OVDCONbits.POVD2L = 1u;     // Output on PWM2l I/O pin is controlled by the PWM
generator.
/**************************************************************************/

/*****************************PWM INTERRUPT CONFIG*************************/
  IFS2bits.PWMIF      = 0u;   // Clear the interrupt flags.
  SEVTCMPbits.SEVTDIR = 1u;   // A special event trigger will occur when the PWM
time base is counting downwards.
  IPC9bits.PWMIP      = 0x05u;// We MUST trigger ADC conversion in PWM interrupt.
  IEC2bits.PWMIE      = 1u;   // Enable PWM interrupt.
/**************************************************************************/

  PTCONbits.PTEN    = 1u;     // PWM time base is ON.
}
```

Slika 3.4.2.12 Konfiguracija PWM periferije

Sa slike 3.4.2.11 može se primetiti sledeće:

- PWM frekvencija je podešena na 10 kHz.
- Na svaku opadajuću ivicu PWM signala generiše se PWM prekid što znači da se prekidna rutina izvršava svakih 0.1 ms.
- PWM je podešen da radi u *Countinous up/down* režimu gde je noseći signal jednakokraki trougao.
- Podešene su dve PWM grane i obe su u komplementarnom režimu.
- I/O pinovi korišćeni za generisanje PWM signala su kontrolisani od strane PWM modula.
- ADC konverzija se započinje u PWM prekidnoj rutini.



Slika 3.4.2.13 Prikaz PWM signala sa prethodno konfigurisanim parametrima

Izvor: https://ww1.microchip.com/downloads/en/DeviceDoc/70062E.pdf

Nakon što su opisane sve funkcije za konfiguraciju potrebnih periferija, na slici 3.4.2.14 prikazan je poziv funkije *dsPIC30F4011_v_Config,* u kojoj su, od strane glavnog programa, pozvane sve opisane funkcije.

```
int main(void)
{
  v_InitState();

  // Initialization goes well.
  if ( State == IDLE_STATE )
  {
    // Signal that system is initialized.
    LATFbits.LATF0  = 1;

    v_Delay(ONE_SECOND);

    while(1)
    {
      v_IdleState();
    }
  }
  // Initialization is not done properly.
  else
  {
    //Do nothing, program will initialize
again.
  }
}
```

```
void v_InitState()
{
  // Initialize all the peripherals.
  dsPIC30F4011_v_Config();

  // Transit to IDLE state.
  SetState(IDLE_STATE);
```

Slika 3.4.2.14 Poziv konfiguracije svih potrebnih periferija od strane glavnog programa u toku procesa inicijalizacije

3.4.3. Osnovna petlja

Počinje da se izvršava nakon izvršenja koda za inicijalizaciju pozivom funkcije *v_IdleState()* koja je prikazana na slici 3.4.3.1.

Glavni zadatak osnovne petlje u ovom radu jeste rad sa potenciometrom za unos referentne brzine i rad sa UART protokolom koji omogućuje komunikaciju sa računarom pri čemu je najbitniji prikaz merenih i zadatih vrednosti brzine i struje, kako bi se imao uticaj na to da li postoji greška stacionarnog stanja kao i uvid o brzinu reakcije na zadatu referentnu vrednost brzine.

```
void v_IdleState()
{
/*******************CURRENT PI REGULATOR INFORMATIONS***********************/
#ifdef CURRENT_REG

  // Print the magic key which marks the beginning of printing current data.
  UART_v_Print( 0x01 );
  // Insert new line.
  UART_v_NewLine();

  // Print referent current.
  UART_v_Print( PIReg.s_CurrentReg.ReferentCurrent);
  // Insert new line.
  UART_v_NewLine();

  // Print measured current.
  UART_v_Print( PIReg.s_CurrentReg.MeasuredCurrent);
  // Insert new line.
  UART_v_NewLine();

  // Print output current from current PI regulator.
  UART_v_Print( PIReg.s_CurrentReg.Output);
  //Insert new line.
  UART_v_NewLine();

#endif // CURRENT_REG
/*****************************************************************************/

/*******************SPEED PI REGULATOR INFORMATIONS*************************/
#ifdef SPEED_REG

  // Print the magic key which marks the beginning of printing speed data.
  UART_v_Print( 0x02 );
  //Insert new line.
  UART_v_NewLine();

  // Print referent speed.
  UART_v_Print( PIReg.s_SpeedReg.ReferentSpeed);
  // Insert new line.
  UART_v_NewLine();

  // Print measured speed.
  UART_v_Print( PIReg.s_SpeedReg.MeasuredSpeed);
  // Insert new line.
  UART_v_NewLine();

  // Print output speed from speed PI regulator.
  UART_v_Print( PIReg.s_SpeedReg.Output);
  //Insert new line.
  UART_v_NewLine();

#endif // SPEED_REG
/*****************************************************************************/

  // Delay writing on terminal for one second.
  v_Delay(ONE_SECOND);
}
```

Slika 3.4.3.1 Prikaz osnovne petlje

3.4.4. Prekidna rutina

U ovom poglavlju detaljno će biti objašnjena PWM prekidna rutina sa slike 3.4.2.

Prvi deo unutar PWM prekidne rutine, prikazan na slici 3.4.4.1, jeste prikupljanje podataka o merenoj brzini i struji kao i o zadatoj brzini koja se zadaje preko potenciometra čime se diktira brzina DC motora.

```
void __attribute__((interrupt,no_auto_psv)) _PWMInterrupt(void)
{
  long Temp;

  // Increase system time in milli seconds, used inside DELAY function.
  SystemTimeMs++;

  v_CatchData();
.
.
.

// Save values from analog input pins inside the structure for storing all of the
measured/referent values.
static void v_CatchData(void)
{
  // A/D is currently filling buffer 0x8-0xF, user should access data in 0x0-0x7 ADC
buffers.
  if ( ADCON2bits.BUFS == 1u )
  {
    Measured.Current = ADC_v_Read(LB_MEASURED_CURRENT); // Read ADC value from ADCBUF1.

    Measured.Speed = ADC_v_Read(LB_MEASURED_SPEED);     // Read ADC value from ADCBUF2.

    // Catch ref speed set via potentiometer.
    ref_speed = ADC_v_Read(LB_REFERENT_SPEED);          // Read ADC value from ADCBUF3.
  }
  // A/D is currently filling buffer 0x0-0x7, user should access data in 0x8-0xF ADC
buffers.
  else
  {
    Measured.Current = ADC_v_Read(HB_MEASURED_CURRENT); // Read ADC value from ADCBUF9.

    Measured.Speed = ADC_v_Read(HB_MEASURED_SPEED);     // Read ADC value from ADCBUF10.

    ref_speed = ADC_v_Read(HB_REFERENT_SPEED);          // Read ADC value from ADCBUF11.
  }
}
```

Slika 3.4.4.1 Prikupljanje podataka neophodnih za proračun brzinskog i strujnog PI regulatora

Nakon prikupljanja svih podataka ulazi se u brzinsku regulacionu petlju, prikazanu na slici 3.4.4.2.

```
/***********************************SPEED   LOOP*******************************/

  // Convert caught measured speed to a fixed point, 3.13 format.
  // Update measured speed value inside speed PI regulator structure.
  PIReg.s_SpeedReg.MeasuredSpeed = i_ConvertToFixedPoint(Measured.Speed,FORMAT_3_13);

  v_SetReferentSpeed(ref_speed);

  v_CalculatePIRegOutput(SPEED_REGULATOR);

/******************************************************************************/
```

Slika 3.4.4.2 Brzinska regulaciona petlja

Sa slike 3.4.4.2 se vidi da se najpre merena vrednost brzine konvertuje u *fixed-point I3Q13* format, ova funkcionalnost prikazana je na slici 3.4.4.1. Više informacija u vezi *fixed-point* aritmetike dato je u prilogu, poglavlje 7.2.

```
int i_ConvertToFixedPoint(int MeasuredValue, e_FormatTypes FormatType)
{
  int ToReturn = 0;

  // The user has selected conversion to 1.15 format.
  if ( FormatType == FORMAT_1_15 )
  {
    // 0 - 1023 -> 0 - 32768
    ToReturn = (PII_REG_MAX_OUTPUT/1023)*MeasuredValue;
  }
  // The user has selected conversion to 3.13 format.
  if ( FormatType == FORMAT_3_13 )
  {
    // 0 - 1023 -> 0 - 8192
    ToReturn = (PIW_REG_MAX_OUTPUT/1023)*MeasuredValue;
  }

  return ToReturn;
}
```

Slika 3.4.4.3 Funkcionalnost za konverziju merenih vrednosti u željenu *fixed-point* aritmetiku

Funkcija *i_ConvertToFixedPoint()* kao parametre prima merenu vrednost očitanu iz određenog ADC bafera i *fixed-point* format u koji ta vrednost treba da se konvertuje. Mogući *fixed-point* formati su I1Q15 i I3Q13.

Nakon konverzije merene vrednosti, podešava se referentna vrednost brzine, prikazano na slici 3.4.4.4.

```
void v_SetReferentSpeed(int ReferentSpeed)
{

  // Entered value is in the range.
  if ( (ReferentSpeed >= MINIMAL_SPEED ) && (ReferentSpeed <= MAXIMAL_SPEED) )
  {
    // Scale entered referent value.
    // 0     ->   0
    // -1023 ->  -8192
    // 1023  ->   8192
    PIReg.s_SpeedReg.ReferentSpeed = (int)( PIW_REG_MAX_OUTPUT * ((float)ReferentSpeed /
(float)MAXIMAL_SPEED));
  }
  // Entered referent value is over the maximal limit.
  else if ( ReferentSpeed >= MAXIMAL_SPEED )
  {
    //Set referent speed to maximal speed value.
    PIReg.s_SpeedReg.ReferentSpeed = PIW_REG_MAX_OUTPUT;
  }
  // Entered referent value is over the negative limit.
  else
  {
    //Set referent speed to minimal speed value.
    PIReg.s_SpeedReg.ReferentSpeed = PIW_REG_MIN_OUTPUT;
  }
};
```

Slika 3.4.4.4 Funkcija za podešavanje referentne brzine

U funkciji *v_SetReferentSpeed()* kao parametar prosleđuje se vrednost brzine zadate preko potenciometra, koja se zatim skalira u opsegu između minimalne i maksimalne referentne brzine u I3Q13 *fixed-point* formatu. U slučaju pogrešnog očitavanja zadate vrednosti brzine, referentna brzina se ograničava na njene limite.

Nakon podešene merene i referentne vrednosti brzine poziva se funkcija *v_CalculatePIRegOutput()* koja predstavlja glavni algoritam za proračun izlaza željenog PI regulatora, prikazana na slici 3.4.4.5. Kao parametar, u ovu funkciju prosleđuje se tip regulatora za koji se vrši proračun.

```
// The user has selected calculation of speed PI regulator  output.
  if ( RegulatorType == SPEED_REGULATOR )
  {
    // Catch last error value.
    LastError = PIReg.s_SpeedReg.Error;

    // Calculate current error value.
   PIReg.s_SpeedReg.Error = PIReg.s_SpeedReg.ReferentSpeed -PIReg.s_SpeedReg.MeasuredSpeed;

    // Calculate deference between last and current error.
    ErrorDiference = LastError - PIReg.s_SpeedReg.Error;

    // Calculate increment in 3.13 format.
    Increment = (     (int)((long)PIReg.s_SpeedReg.Kpw*(long)ErrorDiference         >>
SHIFT_AMOUNT_3_13 ) + (int)((long)PIReg.s_SpeedReg.Kiw*(long)PIReg.s_SpeedReg.Error >>
SHIFT_AMOUNT_3_13 )
                );

    // Set regulator output.
    PIReg.s_SpeedReg.Output += Increment;

.

.

.
```

Slika 3.4.4.5 Algoritam za proračun izlaza PI regulatora brzine

Prvo što se radi unutar algoritma sa slike 3.4.4.5 jeste očitavanje vrednosti prethodne greške nakon čega se vrši proračun nove greške koja se dobija kao razlika prethodno proračunate zadate i merene vrednosti brzine.

U trenutku kada se poznaju prethodna i trenutna vrednost greške, računa se njihova razlika.

Kada su poznate sve greške kao i parametri regulatora ugaone brzine, proračunati u poglavlju 3.2, računa se inkrement preko inkrementalne forme digitalnog PI regulatora:

$$Inc(kT) = Kp[err(kT) - err(kT - T)] + Ki \cdot err(kT) \quad \text{.....3.4.4.1}$$

$$out(kT) = out(kT) + Inc(kT) \quad \text{.....3.4.4.2}$$

Implementirano je i ograničenje regulatora koje u slučaju da proračunat izlaz PI regulatora brzine, referenca struje, izlazi van okvira graničnih vrednosti. U tom slučaju, referentna struja se podešava na njenu maksimalnu ili minimalnu vrednost.

```
// Calculated output value is over the maximal limit.
if (PIReg.s_SpeedReg.Output > PIW_REG_MAX_OUTPUT)
{
  // Set output to maximal value.
  PIReg.s_SpeedReg.Output = PIW_REG_MAX_OUTPUT;
}
// Calculated output value is over the minimal limit.
if (PIReg.s_SpeedReg.Output < PIW_REG_MIN_OUTPUT)
{
  // Set output to minimal value.
  PIReg.s_SpeedReg.Output = PIW_REG_MIN_OUTPUT;
}
```

Slika 3.4.4.6 Ograničenje PI regulatora brzine

Nakon proračuna izlaza PI regulatora brzine, tj. referentne vrednosti struje armature, program ulazi u strujnu petlju, prikazanu na slici 3.4.4.7.

```
**********************************CURRENT LOOP*********************************/

// Output of the speed regulator is reference for the current regulator.
// 0 - 8192    -> 0 - 32768
// 0 - (-8192) -> 0 - (-32768)
int SpeedOutput = (int)(((float)MAXIMAL_CURRENT / (float)PIW_REG_MAX_OUTPUT) *
Reg.s_SpeedReg.Output);

v_SetReferentCurrent(SpeedOutput);

// Convert saved measured current to fixed point, 1.15 format.
// Update measured current value inside current PI regulator structure.
PIReg.s_CurrentReg.MeasuredCurrent = i_ConvertToFixedPoint(Measured.Current, FORMAT_1_15);

v_CalculatePIRegOutput(CURRENT_REGULATOR);

/*****************************************************************************/
```

Slika 3.4.4.7 Strujna regulaciona petlja

Referentna vrednost struje se konvertuje iz *fixed-point I3Q13* formata u *fixed-point I1Q15* format nakon čega se, preko funkcije *v_SetReferentCurrent(),* prosleđuje strukturi koja čuva sve neophodne vrednsti za proračun PI regulatora struje. Funkcija *v_SetReferentCurrent()* prikazana je na slici 3.4.4.8.

```
void v_SetReferentCurrent(int SpeedOutput)
{
  int ReferentCurrent = SpeedOutput;

  // Entered value is in the range.
  if ( ( ReferentCurrent >= MINIMAL_CURRENT ) && ( ReferentCurrent <= MAXIMAL_CURRENT ) )
  {
    // Scale entered referent value.
    // 0    ->   0
    // -18A -> - 32768
    // 18A  ->   32768
    PIReg.s_CurrentReg.ReferentCurrent = (int)(((float)PII_REG_MAX_OUTPUT /
(float)MAXIMAL_CURRENT ) * ReferentCurrent);
  }
  // Entered referent value is over the maximal limit.
  else if ( ReferentCurrent >= MAXIMAL_CURRENT )
  {
    // Set referent current to maximal current value.
    PIReg.s_CurrentReg.ReferentCurrent = PII_REG_MAX_OUTPUT;
  }
  // Entered referent value is over the negative limit.
  else
  {
    //Set referent current to minimal current value.
    PIReg.s_CurrentReg.ReferentCurrent = PII_REG_MIN_OUTPUT;
  }
};
```

Slika 3.4.4.8 Funkcija za podešavanje referentne struje

U funkciji *v_SetReferentCurrent()* kao parametar prosleđuje se vrednost proračunate referentne struje, koja se zatim skalira u opsegu između minimalne i maksimalne referentne struje u *fixed-point I1Q15* formatu. U slučaju zadavanja struje veće/manje od maksimalne/minimalne vrednosti, referentna struje se ograničava na njene limite.

Analogno brzinskoj petlji, merena vrednost struje se prevodi u *fixed-point I1Q15* format preko funkcije *i_ConvertToFixedPoint()* prikazane na slici 3.4.4.3, sa jedinom razlikom u prosleđenom parametru.

Nakon podešavanja referentne i merene vrednosti struje vrši se proračun izlaza PI regulatora struje - elektromagnetnog momenta DC motora.

Proračun momenta se vrši kao i proračun referentne struje, u funkciji *v_CalculatePIRegOutput()* sa tim što se kao parametar prosleđuje drugi tip PI regulatora.

```
void v_CalculatePIRegOutput(e_RegulatorTypes RegulatorType)
{
  int   LastError      = 0;
  int   ErrorDiference = 0;
  int   Increment      = 0;

  // The user has selected calculation of current PI regulator  output.
  if ( RegulatorType == CURRENT_REGULATOR )
  {
    // Catch last error value.
    LastError = PIReg.s_CurrentReg.Error;

    // Calculate current error value.
    PIReg.s_CurrentReg.Error = PIReg.s_CurrentReg.ReferentCurrent -
PIReg.s_CurrentReg.MeasuredCurrent;

    // Calculate deference between last and current error.
    ErrorDiference = PIReg.s_CurrentReg.Error - LastError;

    // Calculate increment in 1.15 format.
    Increment = ( (int)((long)PIReg.s_CurrentReg.Kpi*(long)ErrorDiference           >>
SHIFT_AMOUNT_1_15 ) +
                  (int)((long)PIReg.s_CurrentReg.Kii*(long)PIReg.s_CurrentReg.Error >>
SHIFT_AMOUNT_1_15 )
                );

    // Set regulator output.
    PIReg.s_CurrentReg.Output += Increment;
.
.
.
```

Slika 3.4.4.8 Algoritam za proračun izlaza PI regulatora struje

Nakon proračunatog izlaza PI regulatora struje, definisan je momenat koji treba zadati DC motoru kako bi se eliminisala greška između zadate i referentne vrednosti brzine.

Takođe, implementirano je i ograničenje PI regulatora struje, prikazano na slici 3.4.4.9.

```
// Calculated output value is over the maximal limit.
if (PIReg.s_CurrentReg.Output > PII_REG_MAX_OUTPUT)
{
  // Set output to maximal value.
  PIReg.s_CurrentReg.Output = PII_REG_MAX_OUTPUT;
}
// Calculated output value is over the minimal limit.
if (PIReg.s_CurrentReg.Output < PII_REG_MIN_OUTPUT)
{
  // Set output to minimal value.
  PIReg.s_CurrentReg.Output = PII_REG_MIN_OUTPUT;
}
```

Slika 3.4.4.9 Ograničenje PI regulatora struje

Ostalo je još samo da se proračunat izlaz PI regulatora struje skalira na vrednost poželjnu za zadavanje novog faktora ispune za obe grane PWM modula i da se dozvoli sledeći PWM prekid, kao što je prikazano na slici 3.4.4.10.

```
.
.
.
// Scale output value to PERIOD register value.
Temp = (long)PWM_PERIOD*(long)PIReg.s_CurrentReg.Output;

// Update duty cycle value.
// PDC1 = ((PWM/2) * Temp)*2
PDC1 = ( (int)(PWM_PERIOD >> 1) + (int)(Temp >> 16) ) << 1;

// Set PDC2 register value following complementary logic.
PDC2 = ( 1 - PDC1 );

// Allow next PWM interrupt.
IFS2bits.PWMIF = 0;
}
```

Slika 3.4.4.10 Zadavanje novog faktora ispune

4. Testiranje

Najpre je testiran simulacioni model u Matlab/Simulink okruženju, pomoću kojeg su proračunati parametri brzinskog i strujnog PI regulatora (odzivi sa slika 3.2.1.2 i 3.2.2.4). Nakon simulacije u Matlab/Simulink-u, testiran je konfigurisan sistemski takt tako što je definisan tajmerski prekid, čiji period zavisi od sistemskog takta. Podešeno je da dioda na portu Fo treperi na svakih 500ms što je uspešno ostvareno. Time je indirektno potvrđen sistemski takt od 8 miliona instrukcija po sekundi (end. *milion instructions per second - MIPS*).

Nakon testiranja sistemskog takta testirana je konfiguracija PWM periferije tako što je povezan osciloskop na gornje i donje nožice prve, pa zatim i druge grane. Time je utvrđen komplementaran režim rada, frekvencija PWM signala od 10 kHz kao i očekivana reakcija na promenu ručno unesenog faktora ispune.

Nakon testiranja PWM periferije, testirana je konfiguracija Timer periferije povezivanjem osciloskopa sa pinovima Fo i F1 koji služe za signalizaciju uspešne inicijalizacije i rada u normalnom režimu rada. Dioda na portu F1 je kao i po konfiguraciji generisala impulse na svakih 200 ms što je i očekivano jer se pali u svakom drugom ulazu u Timer2 prekid, koji se izvršava svakih 100 ms, dok je dioda na portu Fo upaljena nakon kratkog perioda potrebnog za inicijalizaciju.

Zatim, testirane su zajedno ADC i UART periferije tako što je vrednost sa potenciometra prosleđena na terminal čime je ustanovljeno da se na terminalu ispisuju očekivane vrednosti u opsegu od 0 do 1023, pomeranjem potenciometra od početne do krajnje tačke.

Za potrebe algoritma korišćen je potencimetar koji je integrisan u razvojnom okruženju sa slike 4.4. Testirana je samo strujna petlja tako što je softverski zadata referentna vrednost struje, dok je potenciometrom simulirana merena vrednost struje koju vidi PI regulator struje. Referentna i merena vrednost struje, kao i trenutna greška, ispisane su na terminalu. Na taj način vizuelno je potvrđeno da se strujna petlja ponaša kako je predviđeno. Takođe, korišćen je i osciloskop sa kojim je prikazana promena faktora ispune koji se menja sa odstupanjem merene vrednost od referentne vrednosti.

Nažalost brzinska petlja nije testirana usled nedostatka merne opreme kao i dodatnih potenciometara kojima bi se mogle simulirati merena brzina i struja u isto vreme.

5. Zaključak

Za potrebe ovog diplomskog rada prvenstveno je osmišljen hardver na kojem je planirano detaljno testiranje softvera kao i njegova krajnja upotreba za upravljanje mašinom jednosmerne struje. Zatim je kreiran model u Matlab/Simulink-u u kojem je simulirano upravljanje DC motorom upotrebom četvorokvadrantnog čopera i testiranje odziva proračunatih parametara PI regulatora brzine i struje. Po završetku testiranja u Matlab/Simulink-u osmišljena je arhitektura softvera, a zatim i sam softver. Na kraju su izvršena testiranja pomenuta u četvrtom poglavlju. Nažalost, hardver nije dovršen u toku procesa razvoja softvera. Stoga je kao zadatak ostalo da se testira obrada podataka sa mernih senzora koji su planirani da se koriste kao i da se testira kompletna regulaciona struktura sa brzinskom i strujnom petljom. Time bi vizuelno bilo demonstrirano ponašanje upravljanog motora.

Tokom kreiranja softvera velika pažnja je posvećena modularnosti (poglavle 3.3) kao i čitljivost koda kako bi bilo jednostavno promeniti parametre upravljane mašine, konfigurisati nove periferije kao i izmeniti konfiguracije već postojećih periferija.

Svrha ovog rada je da krajni korisnik (student) poseti GitHub, skine kod i time poseduje softver na svom računaru koji može da modifikuje. Takođe, ukoliko izrazi želju, može da testira dati kod na kreiranom razvojnom okruženju, tj. da samostalno utvrdi sve prikazano na vežbama.

Rad na projektu doprineo je produbljivanju autorovog znanja u oblasti mikroračunarskih *embedded* sistema kao i razvoju razumevanja C programskog jezika. Glavni izazov u izradi ovog rada predstavljao je adekvatan odabir hardvera i temeljno razumevanje njegove tehničke dokumentacija kao i strukturiranje softvera za ostvarivanje konfigurabilnosti.

Dalji tok razvoja softvera mogao bi se usmeriti ka kreiranju mašine stanja unutar glavnog programa koja bi izvršavala zadatke pritiskom na određeno dugme. Primera radi, pritiskom na dugme, faktor ispune se podesi na maksimalnu vrednost i zatim na minimalnu. Ta radnja se ponavlja ciklično, a za to vreme asistent može da se usresredi na objašnjavanje promene karakterističnih veličina. Po završetku objašnjavanja, asistent bi pritisnuo dugme namenjeno za rad u normalnom režimu rada, pri čemu bi studenti bili u mogućnosti da menjaju faktor ispune preko potenciometra.

6. Literatura

[1] Darko P. Marčetić, “Mikroprocesorsko upravljanje energetskim pretvaračima”, FTN Izdavaštvo, Novi Sad, 2014,

[2] Darko P. Marčetić, Vlado Porobić, “Primena miktroprocesora u elektroenergetici, praktikum laboratorijskih vežbi”, FTN Izdavaštvo, Novi Sad, 2014,

[3] Darko P. Marčetić, “Programabilni logički kontroleri i komunikacioni protokoli u elektroenergetici”, FTN Izdavaštvo, Novi Sad, 2013.

[4] Stevan Grabić, “Upravljanje energetskim pretvaračima”, FTN Izdavaštvo, Novi Sad, 2016.

[5] “Tehnička dokumentacija dsPIC30F4011 mikrokontrolera”, posećeno 03.03.2021. Dostupno na: //ww1.microchip.com/downloads/en/DeviceDoc/70135G.pdf

[6] “Referentni priručnik dsPIC30F4011 mikrokontrolera”, posećeno 03.03.2021. Dostupno na: http://ww1.microchip.com/downloads/en/devicedoc/70046d.pdf

[7] “Easy Pic v7 for dsPIC30F priručnik”, posećeno 03.03.2021. Dostupno na: https://download.mikroe.com/documents/full-featured-boards/easy/easypic-v7-dspic30/easypic-v7-dspic30-manual-v101a.pdf

[8] “UML dizajn za C programere”, posećeno 03.03.2021. Dostupno na: https://www.drdobbs.com/cpp/uml-for-c-programmers/184401948

7. Prilog

7.1. Sistem normalizovanih vrednosti

Da bi se dati model preveo u predstavu sa relativnim jedinicama potrebno je odrediti bazne vrednosti.

Osnovne bazne vrednosti:

$$U_{ab} = U_{anom}; \quad I_{ab} = I_{anom}; \quad \omega_b = \omega_{nom}$$

gde je:

U_{ab} - nominalni napon mašine,

I_{ab}- nominalna struja,

ω_b- nominalna električna ugaona brzina.

Na osnovu osnovnih baznih vrednosti dobijaju se izvedene bazne vrednosti:

$$R_{ab} = \frac{U_{ab}}{I_{ab}}; \qquad \Psi_b = \frac{U_{ab}}{\omega_b}; \qquad \Psi_b = c \cdot \varphi b; \qquad m_b = c \cdot \varphi b \cdot Iab;$$

Korišćenjem baznih i nominalnih vrednosti moguće je izračunati normalizovane vrednosti u relativnim jedinicama:

$$Xrel = \frac{Xn}{Xb}$$

gde je:

$Xrel$ - normalizovana vrednost parametra u relativnim jedinicama,

Xn – nominalna vrednost parametra u određenoj mernoj jedinici,

Xb – bazna vrednost parametra u određenoj mernoj jedinici.

7.2. Aritmetika sa nepokretnim zarezom (*fixed-point*)

Model upravljačke strukture objekta kao što je elektromotorni pogon, koju je neophodno izvršavati u realnom vremenu realizuje se primenom sistema normalizovanih vrednosti, kao što je to prikazano u poglavlju 6.1. Na ovaj način se svi signali koje je neophodno procesirati postavljaju u određeni fiksni opseg (recimo, od -1 do 1) koji je moguće predstaviti brojevima sa nepokretnim zarezom.

Brojevi sa nepokretnim zarezom zapravo predstavljaju obične celobrojne vrednosti (*integer* tip podatka) o kojima korisnik brine kako će ih interpretirati. Broj u nepokretnom zarezu čini celobrojni i decimalni deo i u zavisnosti od mesta decimalne tačke moguće je predstaviti različite opsege realnih brojeva. Kako je binarna prezentacija broja zasnovana na stepenu dvojke, sledi da će bitovi nakon zamišljene decimalne tačke označavati “polovine“ prethodne pozicije (1/2,1/4,1/8 itd.).

Na taj način ako su podaci 16-bitni, a decimalni zarez “postavimo“ odmah nakon MSB bita (bit najveće važnosti) celobrojna vrednost $2^{16-1} - 1 = 32767$ označavaće realan broj ~1.0 dok će ce broj $-2^{16} = -32768$ označavati broj -1.0. Ovakva prezentacija broja označava se obično kao I1Q15 format. Za razliku od I1Q15 formata u I3Q13 formatu se decimalni zarez postavlja dva mesta nakon MSB bita čime će broj $2^{14-1} - 1 = 8191$ označavati realan broj ~3.0 dok će broj $-2^{14} = -8192$ označavati realan broj -3,0.

Na slici 6.1 prikazani su parametri PI regulatora brzine i struje, proračunati u poglavlju 3.2 , predstavljeni u *fixed-point* aritmetici.

```
// Maximal and minimal base value for 1.15 fixed point format.
#define PII_REG_MAX_OUTPUT        (32768)
#define PII_REG_MIN_OUTPUT        (-32768)

// Maximal and minimal base value for 3.13 fixed point format.
#define PIW_REG_MAX_OUTPUT        (8192)
#define PIW_REG_MIN_OUTPUT        (-8192)

//RefCurrent
#define CURRENT_PROPORTIONAL_GAIN   (67)    //1.15 format       (0.0021f)           //Kpi
#define CUREENT_INTEGRAL_GAIN       (8)     //1.15 format       (0.00024592f)       //Kii

//RefSpeed
#define SPEED_PROPORTIONAL_GAIN     (10813) //3.13 format       (2.32f)             //Kpw
#define SPEED_INTEGRAL_GAIN         (3277)  //3.13 format       (0.4f)              //Kiw
```

Slika 7.2 Parametri PI regulatora brzine i struje kao i maksimalne vrednosti brzine i struje u fixed point aritmetici

[i] Deklaracija funkcije ili promenljive služi za informisanje kompajlera o sledećim informacijama: ime funkcije/promenljive, tip vrednosti koju promenljiva drži/koju funkcija vraća, i broj kao i tip argumenata funkcije, tj. deklaracija daje detalje o svojstvima promenljive.

[ii] Definicija promenljive/funkcije kaže gde se promenljiva/funkcija čuva. tj. memorija za promenljivu/funkciju se dodeljuje tokom definisanja promenljive/funkcije.