»Mladi za napredek Maribora 2016«

33. srečanje

Krmiljenje enosmernega motorčka

Raziskovalno področje: Elektrotehnika, elektronika

Raziskovalna naloga

Avtor: TILEN KURNIK, TADEJ GOLOB

Mentor: MILAN IVIČ

Šola: SREDNJA ELEKTRO-RAČUNALNIŠKA ŠOLA MARIBOR

Maribor, februar 2016

1. Kazalo

Kazalo slik

1. POVZETEK

Raziskovalna naloga predstavlja krmiljenje in regulacijo enosmernega motorčka s pomočjo vmesnika, izdelanega v programskem okolju LabVIEW in uporabo razvojne plošče Arduino Uno, ki preko gonilnika L293 krmili DC motorček.
Cilj raziskave je bila proučitev možnosti krmiljenja in regulacije DC motorčka z razvojno ploščo Arduino Uno in izdelavo programske kode v okolju LabVIEW, s pomočjo katerega bi lahko izdelali ustrezen vmesnik, ki bi nam služil za krmiljenje, regulacijo in s pomočjo Web kamere prikazovanje dogajanja na DC motorčku.
DC motorček smo najprej krmilili le z razvojno ploščo Arduino Uno. Izdelali smo programsko kodo, s pomočjo katere smo lahko krmilili smer vrtenja in hitrost vrtenja DC motorčka. Nato smo vključili programsko okolje LabVIEW. Z izdelano kodo smo lahko krmilili smer vrtenja in hitrost vrtenja DC motorčka preko vmesnika. Nazadnje smo vključili PID regulacijo, ki nam je omogočila čim manjšo napako dejanske hitrosti vrtenja DC motorčka pri spremembi želene hitrosti vrtenja DC motorčka.
Pri delu smo uporabili metodo praktične povezave ustreznih elementov z razvojno ploščo Arduino Uno, metodo programiranja oziroma izdelave programske kode za Arduino Uno in za LabVIEW ter metodo merjenja in praktičnega preizkušanja.

2. ZAHVALA

Za nasvete, pomoč, potrpljenje in za veliko pridobljenega znanja, se zahvaljujeva mentorju. Pridobljeno znanje bova lahko izkoristila pri najinem nadaljnjem delu.

3. Hipoteze

- Z razvojno ploščo Arduino Uno in ustrezno programsko kodo lahko spreminjamo smer in hitrost vrtenja DC motorčka.
- Vzpostavimo lahko komunikacijo med Arduino Uno in LabVIEW.
- S programsko kodo v okolju LabVIEW lahko preko izdelanega vmesnika krmilimo smer in hitrost vrtenja DC motorčka. Dogajanje lahko na vmesniku spremljamo s pomočjo Web kamere.

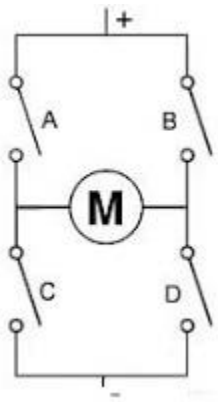
4. VSEBINSKI DEL

4.1 Krmiljenje enosmernega, DC motorčka

DC motorček je enosmerni motorček. Ko nanj priključimo enosmerno napetost, se zaradi menjavanja smeri magnetnega polja vrti. Zgrajen je iz mirujočega, statorskega železnega jedra, na katerem se nahaja vzbujalno navitje za ustvarjanje magnetnega polja. Med magnetnimi poli statorja se nahaja rotor z navitjem, povezanim preko komutatorja in ščetk na zunanji vir enosmerne napetosti. Če menjamo polariteto priključene enosmerne napetosti, se motorček vrti v drugo smer.
Za spreminjanje polaritete priključene napetosti in s tem spreminjanje smeri vrtenja DC motorčka uporabimo H-mostiček v integriranem vezju L293.

H-mostiček:

Princip delovanja vezja, po katerem ima H-mostiček ime vidimo na sliki 1.



Slika 1: Prikaz vezja za spreminjanje polaritete napetosti na priključkih DC motorčka, H-mostiček.

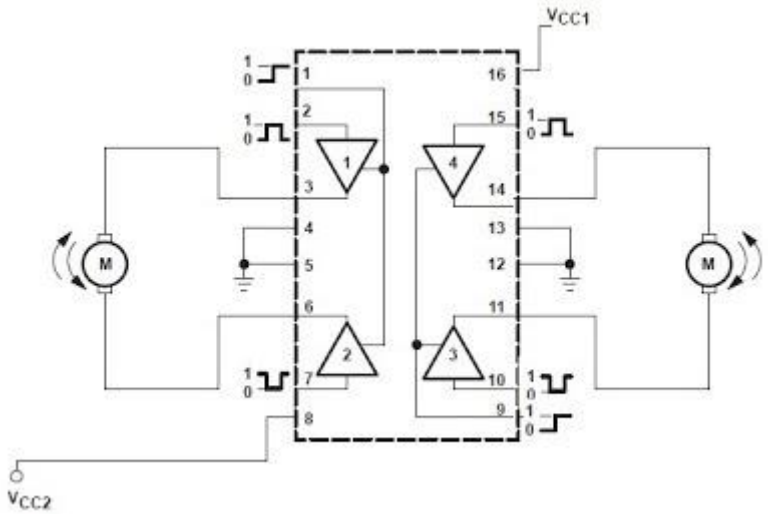
Vezje H-mostička vsebuje štiri stikala: A, B, C in D. S preklapljanjem teh stikal motorček krmilimo na različne načine:

- Če vklopimo stikalo A in stikalo D, se bo motorček vrtel v smeri urinega kazalca,
- Če vklopimo stikalo B in stikalo C, se bo motorček vrtel v nasprotni smeri urinega kazalca,
- Če vklopimo stikalo A in stikalo B, se bo vrtenje motorčka ustavilo,
- Če vsa stikala izklopimo, bo motorček brez napajalne napetosti,
- Če istočasno vklopimo stikalo A in stikalo C ali stikalo B in stikalo D, je vezje v kratkem stiku, zato tega ne počnemo.

H-mostiček lahko izdelamo z različnimi elementi kot so releji, MOSFET, FET ali bipolarnimi BJT tranzistorji. Če pa naše zahteve po tokovni porabi niso prevelike, če želimo krmiliti DC motorčke z majhno porabo, ki ne potrebujejo večjih tokov, lahko enostavno uporabimo integrirano vezje L293.

4.1.1 Integrirano vezje L293

Integrirano vezje L293 je namenjeno za krmiljenje dveh enosmernih motorčkov, da se lahko vrtita v obe smeri. Mi bomo krmilili en motorček. Nanj lahko priključimo motorček, ki za svoje delovanje ne potrebuje večjega toka od 1 A (L293D 600 mA) in za napetosti od 4,5 V do 36 V. Vsi vhodi so združljivi s TTL signali. Za zaščito ima L293 vgrajene hitre diode, ki ščitijo integrirano vezje pred napetostnimi konicami, ki nastanejo pri vklopu in izklopu motorčka (predvsem pri izklopu). Če se L293 segreje čez mejo 70°C, vgrajeni senzorji ustavijo delovanje motorčka.



Slika 2: Priključitev DC motorčkov na L293 (vir: Texas Instruments).

Motorček bomo priključili na pina 3 in 6, krmilili pa ga bomo s pini 1, 2 in 7. Na pin 8 (Vcc2) priključimo napetost, ki jo potrebuje DC motorček za delovanje, na pin 16 (Vcc1) pa napetost za gonilnik L293, ki znaša 5 V.

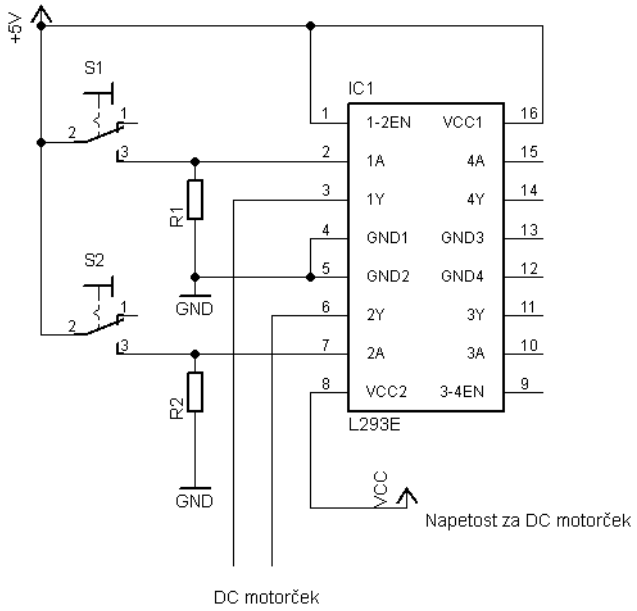
Tabela 1: Opis priključitvenih pinov L293.

Pin 1 (EN)	Pin 2	Pin 7	Funkcija
1	0	1	Vrtenje motorčka v smeri urinega kazalca.
1	1	0	Vrtenje proti smeri urinega kazalca.
1	0	0	Stop.
1	1	1	Stop.
0	neuporabno	neuporabno	Stop.

Pina 1 in 9 sta enable pina. Povezana morata biti na +5 V (logična 1), če želimo, da se bosta motorčka vrtela. Ker imamo mi priključen le en motorček, priključen na levo stran, poglejmo način krmiljenja motorčka (enako velja za drugi motorček):

- Pina 4 in 5 morata biti povezana na skupno maso (GND).
- Pina 2 in 7 sta vhodna pina gonilnika L293 in sta kontrolna pina, ki nadzorujeta delovanje motorčka.
- Pina 3 in 6 sta izhodna pina gonilnika L293, kamor priključimo DC motorček.
- Pin 16 je pin za napetostno napajanje gonilnika L293. Priključen mora biti na napetost +5 V
- Pin 8 je pin, kamor priključimo pozitivno napetost iz drugega napajalnika oziroma baterije za napetost DC motorčka (od 4,5 V do 36 V, odvisno od uporabljenega motorčka).

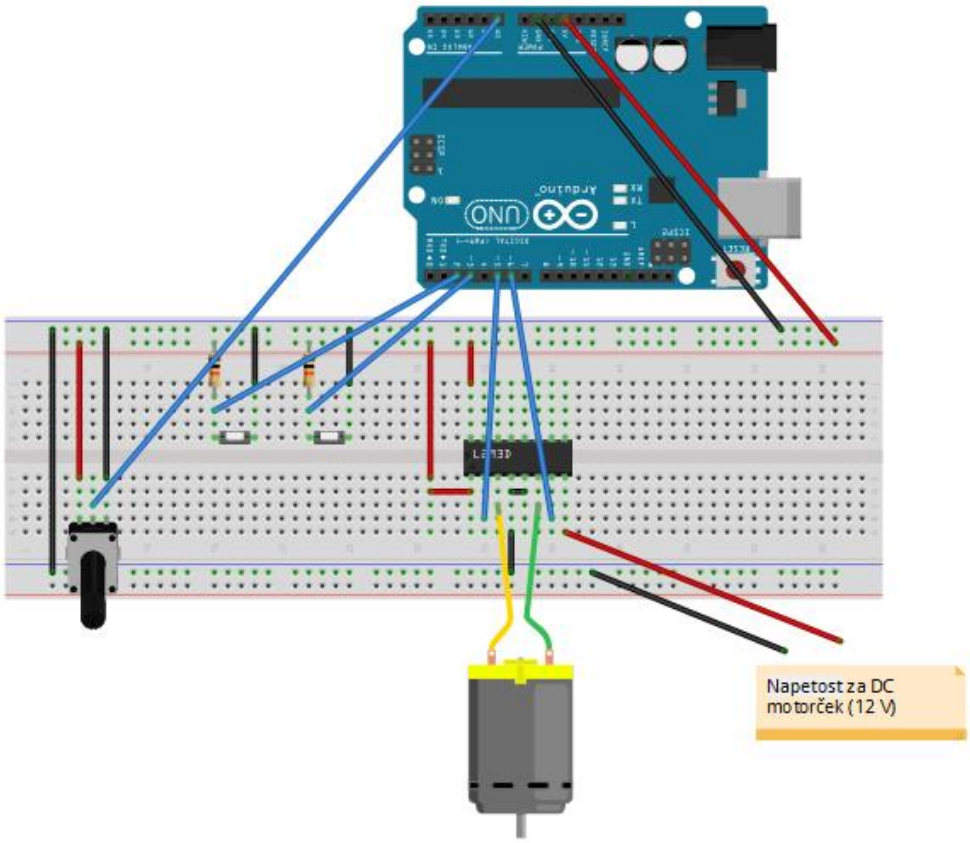
Spreminjanje smeri vrtenja DC motorčka z uporabo gonilnika L293 smo preizkusili s priključitvijo tipk. S prvo tipko smo spreminjali logično stanje na pinu 2, z drugo tipko pa smo spreminjali logično stanje na pinu 7. Načrt priključitve prikazuje slika 3. Upora R1 in R2 sta pull down upora vrednosti 10 kΩ in skrbita za to, da sta pina 2 in 7 v stanju logične 0, ko tipki nista sklenjeni. Če pritisnemo na katero od tipk, pošljemo na ustrezni pin napetost +5 V (logična 1).



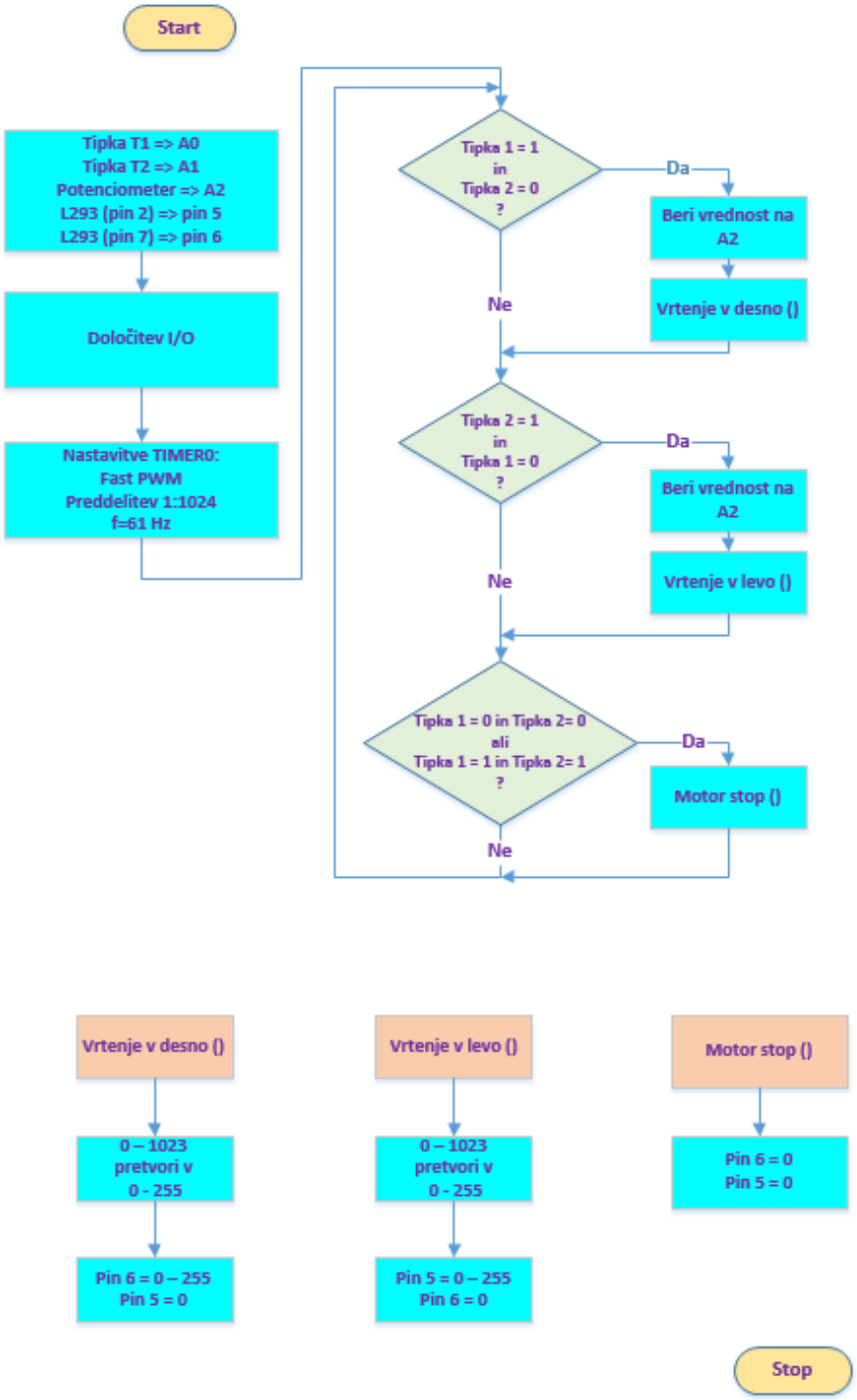
Slika 3: Krmiljenje smeri vrtenja DC motorčka s tipkama.

4.2 Krmiljenje DC motorčka z Arduino Uno

Smer vrtenja DC motorčka spreminjamo s pritiskom na eno oziroma drugo tipko, s potenciometrom pa spreminjamo hitrost vrtenja DC motorčka.



Slika 4: Priključitev elementov vezja na razvojno ploščo Arduino Uno.



Slika 5: Diagram poteka.

Program za krmiljenje DC motorčka:

```
int Tipka1 = 2;                 //Tipka za smer vrtenja levo (proti smeri urinega kazalca).
int Tipka2 = 3;                 //Tipka za smer vrtenja desno (v smeri urinega kazalca).
int Potenciometer = A0;         //Potenciometer za spreminjanje hitrosti vrtenja.
int Motor_Pin2 = 5;             //Krmilni pin 2 na gonilniku L293.
int Motor_Pin7 = 6;             //Krmilni pin 7 na gonilniku L293.
int Vrednost;                   //Spremenljivka, kamor se shranjuje vrednost na A2.

void setup()
{
 //Nastavitev vhodnih in izhodnih pinov:
 pinMode(Tipka1, INPUT);
 pinMode(Tipka2, INPUT);
 pinMode(Potenciometer, INPUT);
 pinMode(Motor_Pin2, OUTPUT);
 pinMode(Motor_Pin7, OUTPUT);

 /*Nastavimo Timer0 na frekvenco 61 Hz (Če naslednji dve vrstici izpustimo, imamo frekvenco 970
    Hz).
    Pri frekvenci 61 Hz lažje reguliramo nizke vrtljaje DC motorčka.
 */
 TCCR0A = _BV(COM0A1) | _BV(COM0B1) | _BV(WGM01) | _BV(WGM00);
 TCCR0B = _BV(CS00) | _BV(CS02);
}

void loop()
{
 //Preverjamo ali je Tipka1 sklenjena:
 if((digitalRead(Tipka1) == HIGH)  && (digitalRead(Tipka2) == LOW))
 {
  Vrednost = analogRead(Potenciometer);      //Preberemo vrednost potenciometra.
  Vrtenje_proti_smeri_urinega_kazalca();     //Klicanje funkcije.
 }

 //Preverjamo ali je Tipka2 sklenjena:
 if((digitalRead(Tipka1) == LOW)  && (digitalRead(Tipka2) == HIGH))
 {
  Vrednost = analogRead(Potenciometer);
  Vrtenje_v_smeri_urinega_kazalca();         //Klicanje funkcije.
 }

 //Ali ni nobena tipka sklenjena?
 if((digitalRead(Tipka1) == LOW) && (digitalRead(Tipka2) == LOW))
 {
  Motor_stop();                              //Klicanje funkcije.
 }

 //Ali sta obe tipki sklenjeni?
 if((digitalRead(Tipka1) == HIGH) && (digitalRead(Tipka2) == HIGH))
 {
  Motor_stop();                              //Klicanje funkcije.
 }
```

```
}

void Vrtenje_proti_smeri_urinega_kazalca()
{
  //Na pin 2 gonilnika L293 pripeljemo PWM signal, na pin 7 gonilnika L293 pa logično 0:
  analogWrite(Motor_Pin2, map(Vrednost, 0, 1023, 0, 255));
 //Z operatorjem map vrednost iz območja 0 - 1023 prestavimo v območje 0 - 255.
  analogWrite(Motor_Pin7, 0);              //Pin 7 gonilnika L293 postavimo na 0.
}

void Vrtenje_v_smeri_urinega_kazalca()
{
  //Na pin 7 gonilnika L293 pripeljemo PWM signal, na pin 2 gonilnika L293 pa logično 0:
  analogWrite(Motor_Pin7, map(Vrednost, 0, 1023, 0, 255));
  //Z operatorjem map vrednost iz območja 0 - 1023 prestavimo v območje 0 - 255.
  analogWrite(Motor_Pin2, 0);              //Pin 2 gonilnika L293 postavimo na 0.
}

void Motor_stop()
{
  //Oba krmilna pina na gonilniku L293 postavimo na 0 => vrtenje DC motorčka se ustavi:
  analogWrite(Motor_Pin7, 0);
  analogWrite(Motor_Pin2, 0);
}
```

Tipki sta priključeni na digitalna vhoda 2 in 3, potenciometer pa na analogni vhod A0 razvojne plošče Arduino Uno. Krmilna pina za gonilnik L293, ki določata smer vrtenja, pa sta priključena na pin 5 in na pin 6 razvojne plošče Arduino Uno. Ta dva pina sta izhodna PWM[1] pina časovnika Timer0. Z ustrezno nastavitvijo registrov TCCR0A in TCCR0B smo spremenili frekvenco PWM signala na teh dveh pinih na 61 Hz.
V odvisnosti od stanja tipk iz glavne zanke kličemo funkcije:

- za vrtenje DC motorčka v smeri urinega kazalca,
- za vrtenje DC motorčka v nasprotni smeri urinega kazalca,
- za zaustavitev vrtenja DC motorčka.

V vezju uporabimo pull-down upora vrednosti 10 kΩ in potenciometer vrednosti 10 kΩ.

4.2.1 Timer0 v Arduino Uno

Arduino Uno vsebuje mikrokontroler ATmega 328P. Vsebuje tri timerje, Timer0, Timer1 in Timer2. Timer0 in Timer2 sta 8-bitna, Timer1 pa je 16-bitni. Osnovna razlika med 8-bitnim in 16-bitnim je resolucija. 8-bitni ima resolucijo 255, 16-bitni pa 65535.
Vsi timerji so odvisni strojne ure. Arduino Uno poganja strojna ura (oscilator) frekvence 16 MHz.

Timer0 je v Arduino namenjen časovni funkciji kot so zakasnitve delay(), millis() in micros(). Zato moramo paziti pri uporabi tega timerja, saj s spremenjenimi nastavitvami spremenimo tudi časovne funkcije. Pri 16 MHz strojni uri se Timer0 register **TCNT0** poveča za 1 vsake 0,0625 µs.

[1] PWM (ang. **P**ulse **W**idth **M**odulation, pulzno širinska modulacija)

t = 1/16000000 Hz = 0,0625 µs

Timer0 lahko povečuje svojo vrednost s strojno uro, lahko pa uporabimo preddelilnik, ki deli prožilne impulze (impulze strojne ure). Preddelitev nastavljamo s prvimi tremi biti (CS00, CS01 in CS02) registra **T**imer/**C**ounter **C**ontrol **R**egister (**TCCR0B**).

Slika 6: Register TCCR0B (Timer/Counter Control register).

Tabela 2: Vloga bitov CS02, CS01 in CS00 v registru TCCR0B.

CS02	CS01	CS00	Opis:
0	0	0	Preddelitev ni izbrana, Timer0 stop
0	0	1	Preddelitev 1:1
0	1	0	Preddelitev 1:8
0	1	1	Preddelitev 1:64
1	0	0	Preddelitev 1:256
1	0	1	Preddelitev 1:1024
1	1	0	Zunanji oscilator, pin T0. (Impulzi iz 1 na 0)
1	1	1	Zunanji oscilator, pin T0. (Impulzi iz 0 na 1)

Način delovanja timerja0 nastavljamo tudi z registrom **TCCR0A** (**T**imer/**C**ounter **C**ontrol **R**egister):

Slika 7: Register TCCR0A (Timer/Counter Control register).

S setiranjem bitov COM0A1 in COM0A0 (bita 7 in 6) v registru TCCR0A določimo pina 5 in 6 razvojne plošče Arduino Uno kot izhodna PWM pina timerja0. Če pogledamo v tabelo, ki prikazuje različne načine delovanja timerja0 vidimo, da smo izbrali način Fast PWM (Mode 3). Bita WGM00 in WGM01 smo setirali, postavili na 1.

Tabela 3: Različni načini delovanja Timer0.

Mode	WGM02	WGM01	WGM00	Timer/Counter mode operation	Top	Update of OCRx	TOV Flag set ON Max = 0xFF Bottom = 0x00
0	0	0	0	Normal	0xFF	Takoj (Immediate)	Max
1	0	0	1	PWM, Phase Correct	0xFF	Top	Bottom
2	0	1	0	CTC	OCR0A	Takoj (Immediate)	Max
3	0	1	1	Fast PWM	0xFF	Bottom	Max
4	1	0	0	Rezervirano	-	-	-
5	1	0	1	PWM, Phase Correct	OCR0A	Top	Bottom
6	1	1	0	Rezervirano	-	-	-
7	1	1	1	Fast PWM	OCR0A	Bottom	Top

Z vrsticama:

```
TCCR0A = _BV(COM0A1) | _BV(COM0B1) | _BV(WGM01) | _BV(WGM00);
TCCR0B = _BV(CS00) | _BV(CS02);
```

smo določili preddelitev 1:1024 (CS00 in CS02 smo postavili na 1), način delovanja timerja0 pa na **Fast PWM** (WGM01 in WGM00 smo postavili na 1), Mode 3.

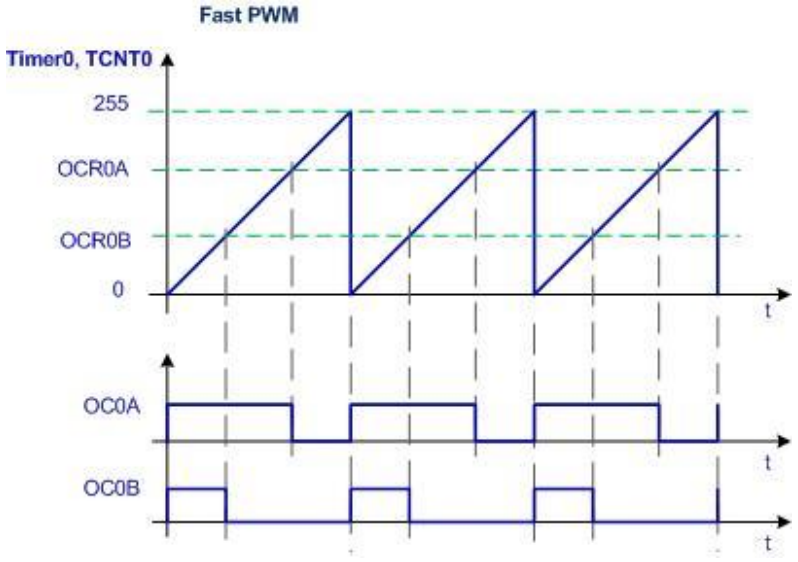
Fast PWM imenujemo, ker lahko generiramo pulzno širinske signale visokih frekvenc. Timer povečuje svojo vrednost od 0 do 255. V registra OCR0A in OCR0B lahko zapišemo vrednosti od 0 do 255.

Oba Output Compare Registra (OCR0A in OCR0B) ves čas primerjata vrednost timerjevega registra TCNT0. Razultat tega primerjanja lahko uporabimo za generiranje pulzno širinskih (PWM) impulzov na izhodnih pinih. Ta dva pina sta pri Arduino Uno pin 6 in pin 5. Vrednost OCR0A smo spreminjali s potenciometrom. Ker ima analogni vhod A2, kamor smo priključili potenciometer 10-bitni A/D pretvornik, smo z operatorjem *map* območje 0-1023 prestavili v območje 0-255. Primer delovanja prikazuje slika 8.

Pri teh nastavitvah lahko izračunamo frekvenco PWM signala.

$$f_{PWM} = \frac{f_{OSC}}{256 \cdot 1024} = \frac{16\ MHz}{256 \cdot 1024} = 61\ Hz$$

Mode3, Fast PWM:



Slika 8: Prikaz delovanja v načinu Fast PWM.

Namesto s potenciometrom bi lahko programsko nastavili vrednost OCR0A in OCR0B:

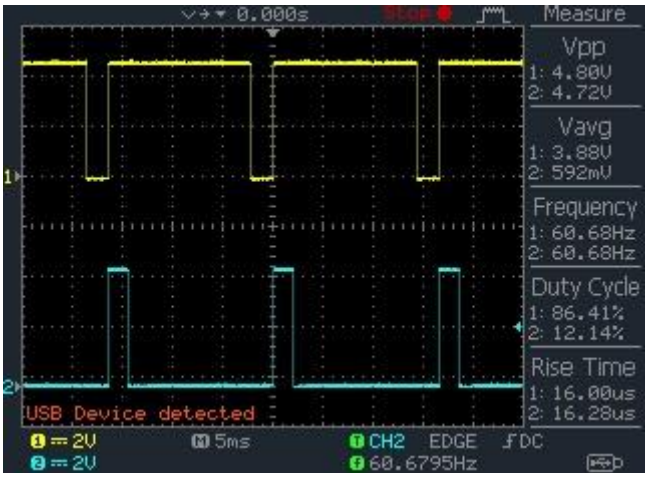
```
void setup()
{
 pinMode(5, OUTPUT);          //Timer0 (OCR0B)
 pinMode(6, OUTPUT);          //Timer0 (OCR0A)
}

 void loop()
{
  TCCR0A = _BV(COM0A1) | _BV(COM0B1) | _BV(WGM01) | _BV(WGM00);
  TCCR0B = _BV(CS00) | _BV(CS02);
  OCR0A = 220;                 //pin 6   Duty Cycle = (220 + 1)/(256) = 86,3 %
  OCR0B = 30;                  //pin 5   Duty Cycle = (30 + 1)/(256) = 12,1 %
}
```

Pri preizkusu delovanja smo OCR0A nastavili na vrednost 220, OCB0B pa na vrednost 30. Če izračunamo vrednost Duty Cycle, dobimo:

$$Duty\ Cycle_{pin\,6} = \frac{(220+1)}{256} \cdot 100\ \% = \ 86{,}3\ \%$$

$$Duty\ Cycle_{pin\,5} = \frac{(30+1)}{256} \cdot 100\ \% = \ 12{,}1\ \%$$

Pri načinu Fast PWM potrebuje timer0, da doseže zgornjo mejo, 1 strojni cikel več od primerjalne vrednosti.
Pri teh nastavitvah smo z digitalnim osciloskopom preverili dogajanje na teh dveh pinih in dobili enake rezultate, kot smo jih izračunali, kar prikazuje slika 9.



Slika 9: Prikaz PWM signalov (pin 6 zgoraj, pin 5 spodaj).

4.3 Programsko okolje LabVIEW

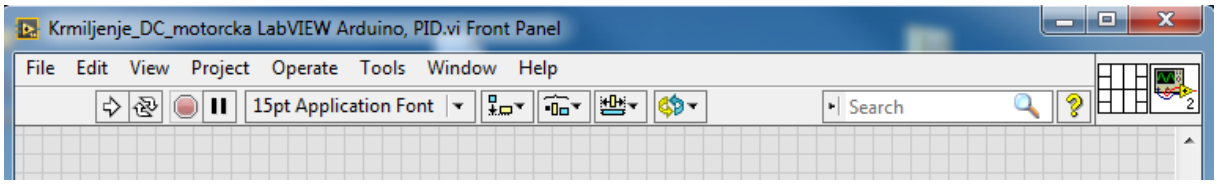
Programsko okolje LabVIEW (Laboratory Virtual Instrument Engineering Workbench) proizvajalca National Instruments (NI) je v celoti grafično zasnovano ter omogoča razvoj merilnih in testnih aplikacij. Programi napisani z orodjem LabVIEW se imenujejo virtualni instrumenti (VI). Vsak virtualni instrument je sestavljen iz čelne plošče in blok diagrama.

LabVIEW je blokvno orodje, kar pomeni, da se algoritem v tem programskem jeziku kodira blokovno oziroma z uporabo funkcij v obliki blokov, ki so med seboj povezani s povezavami. Vsebuje tudi Application Builder, ki omogoča kreiranje izvršne (exe) verzije, ki se lahko izvaja na računalniku, kjer razvojnega sistema LabVIEW ni nameščenega.

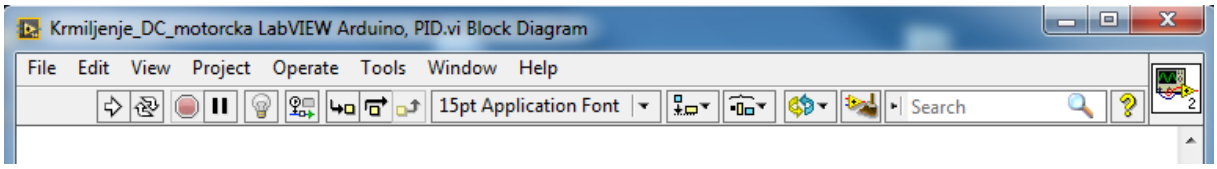
Ob zagonu programa LabVIEW se pojavi okno, kjer lahko izbiramo med kreiranjem novega projekta ali odpiranjem obstoječih, shranjenih projektov. Novi virtualni instrument (VI) kreiramo z izbiro: *File > New VI*. Ob tem se pojavita dve okni:

Čelna plošča (Front Panel)
Blok diagram (Block Diagram)

Čelna plošča ima privzeto sivo ozadje in je namenjena kreiranju uporabniškega vmesnika, Blok diagram pa kreiranju algoritma virtualnega instrumenta.

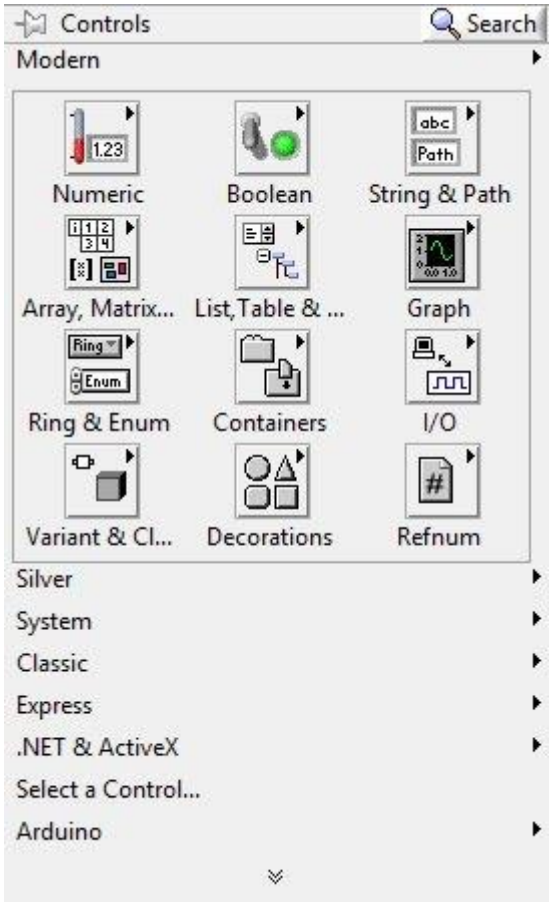


Slika 10: Čelna plošča (Front Panel).



Slika 11: Blok diagram (Block Diagram).

Čelna plošča (Front Panel) predstavlja uporabniški vmesnik virtualnega instrumenta (VI). Zgradimo ga iz objektov, ki se nahajajo na paleti kontrol in indikatorjev. Do te palete lahko dostopamo na dva ničina: Preko izbire menija *View > Controls Palette* ali pa kjerkoli v praznem prostoru Čelne plošče kliknemo desno tipko miške.



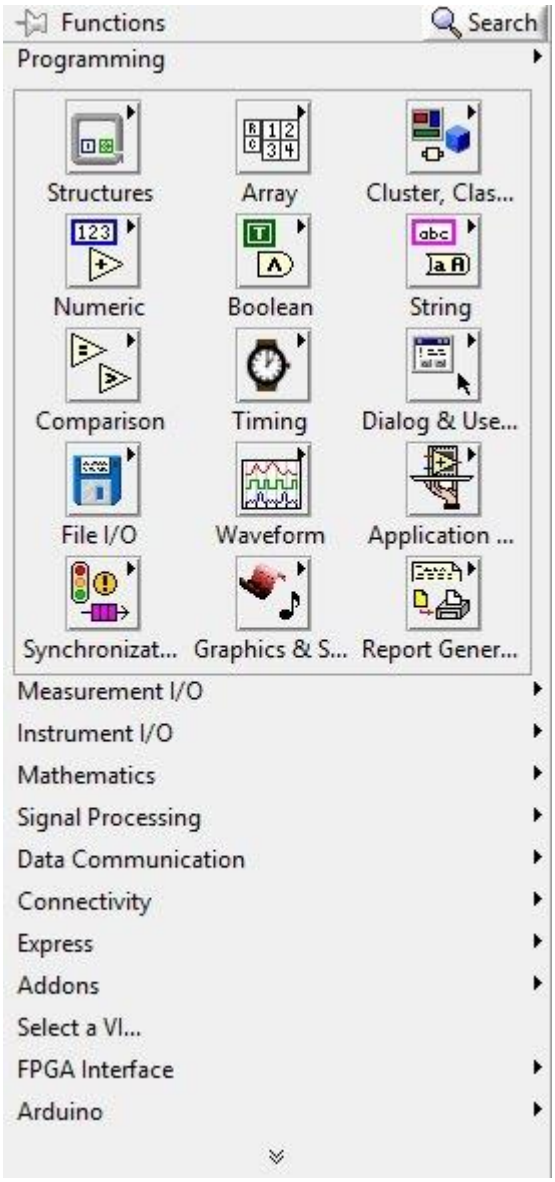
Slika 12: Paleta kontrol in indikatorjev.

Paleta kontrol in indikatorjev je sestavljena iz več podpalet. Objekt (kontrolo ali indikator) namestimo na čelno ploščo tako, da ga izberemo in ga povlečemo na Čelno ploščo. Za vsak objekt, ki ga vstavimo na Čelno ploščo, se v Blok diagramu avtomatsko kreira priključek (Terminal). Priključki služijo za prenos podatkov med Čelno ploščo in Blok diagramom. Ime priključka v Blok diagramu je enako, kot je ime pripadajočega objekta na Čelni plošči. Iz oblike priključka lahko razberemo, ali pripada kontroli ali indikatorju.

Vhodni objekti, ki jim pravimo kontrole (Controls), predstavljajo vhodne podatke virtualnega instrumenta. Te objekte lahko med izvajanjem virtualnega instrumenta spreminjamo in s tem vplivamo na njegovo delovanje. Izhodni objekti, ki jim pravimo indikatorji (Indicators), pa služijo za

prikaz podatkov. Med izvajanjem virtualnega instrumenta se izračunani podatki posredujejo indikatorjem, ki jih nato prikažejo v tekstovni ali grafični obliki.

Priključki (Terminal) kontrol imajo poudarjeno obrobo. Priključki vsebujejo majhen črni trikotnik, ki je na priključku kontrole usmerjen iz priključka, pri indikatorjih pa v priključek. Pri kontrolah se podatek pridobi iz objekta Čelne plošče in se preko priključka posreduje v Blok diagram. Pri indikatorjih se podatek pridobi iz Blok diagrama in se preko priključka prenese v pripadajoči objekt na Čelni plošči, kjer se prikaže.

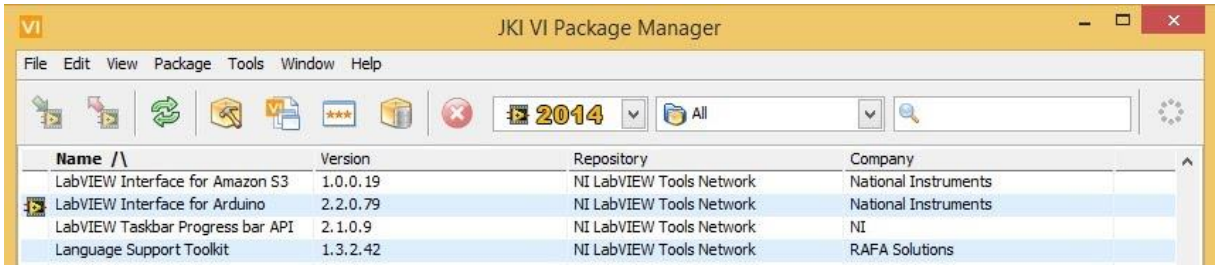


Slika 13: Paleta funkcij.

Objektom čelne plošče in blok diagrama je možno preko menija objekta spreminjati njegove lastnosti. Do priročnega menija objekta dostopamo tako, da se s kurzorjem miške pomaknemo na objekt in pritisnemo na desno tipko miške.

Blok diagram (Block Diagram) je namenjen kodiranju algoritma virtualnega instrumenta (VI). Algoritem zgradimo iz objektov, ki se nahajajo na paleti funkcij, te objekte pa med seboj povežemo s povezavami. Do palete funkcij dostopamo na dva načina: Preko izbire menija *View > Function Palette* ali pa kjerkoli v praznem prostoru kliknemo desno tipko miške.

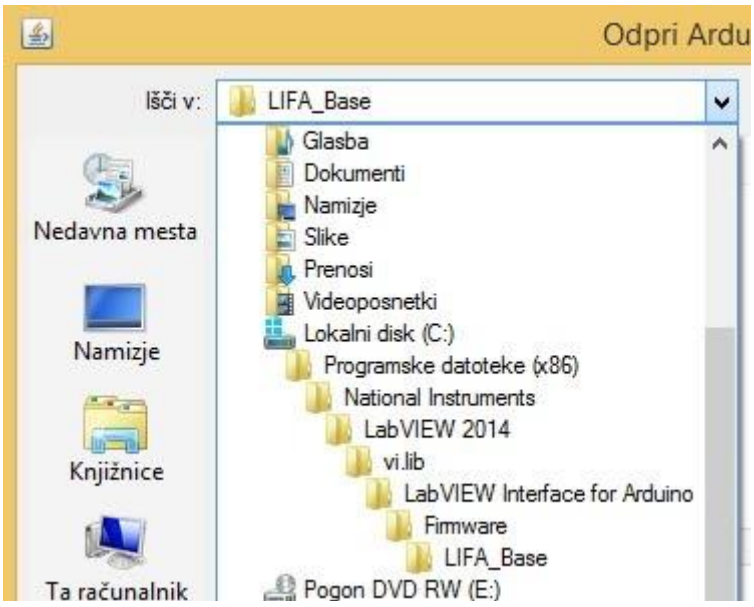
Funkcijo namestimo v Blok diagram tako, da jo na paleti funkcij izberemo in povlečemo v Blok diagram.
V Blok diagramu virtualnega instrumenta se lahko nahajajo naslednji objekti:
Priključki (Terminals), to so priključni elementi objektov Čelne plošče,
Podprogrami (SubVIs), to so dejansko virtualni instrumenti, ki prav tako vsebujejo Čelno ploščo in Blok diagram,
Funkcije (Functions),
Konstante (Constants), to so objekti v Blok diagramu, katerim se vrednost med izvajanjem virtualnega instrumenta ne spreminja,
Programske strukture (Structures), to so objekti, kot so zanka While, zanka For, pogojni stavki (Case) zaporedje (Flat Sequence) in drugi objekti,
Povezave (Wires), ki povezujejo posamezne objekte Blok diagrama in služijo prenosu podatkov.
Zaradi obširnosti bomo posamezne operacije, programiranje in oblikovanje vmesnika opisali sproti, odvisno od tega, kaj smo pri našem programu potrebovali.
Z VI Package Manager-jem naložimo (instaliramo) LIFA (LabVIEW Interface for Arduino).



Slika 14: LIFA (LabVIEW Interface for Arduino).

Ko naložimo LIFA (LabVIEW Interface for Arduino), imamo na razpolago paleto kontrol, indikatorjev in funkcij, ki jih potrebujemo za izdelavo in razvoj programov v okolju LabVIEW, ki bodo prek serijske komunikacije nadzorovali (krmilili vhode in izhode) razvojno ploščo Arduino Uno.
Za komunikacijo LabVIEW z Arduinom prek serijskih vrat moramo imeti instalirane gonilnike NI-VISA (Virtual Instrument Software Architecture).

Odpremo okolje Arduino in v menijski vrstici izberemo *Datoteka > Odpri... > Lokalni disk (C:) > Programske datoteke (x86) > National Instruments > LabVIEW (2014) > vi.lib > LabVIEW Interface for Arduino > Firmware > LIFA_Base*.



Slika 15: V računalniku poiščemo in odpremo Arduino LIFA_Base.

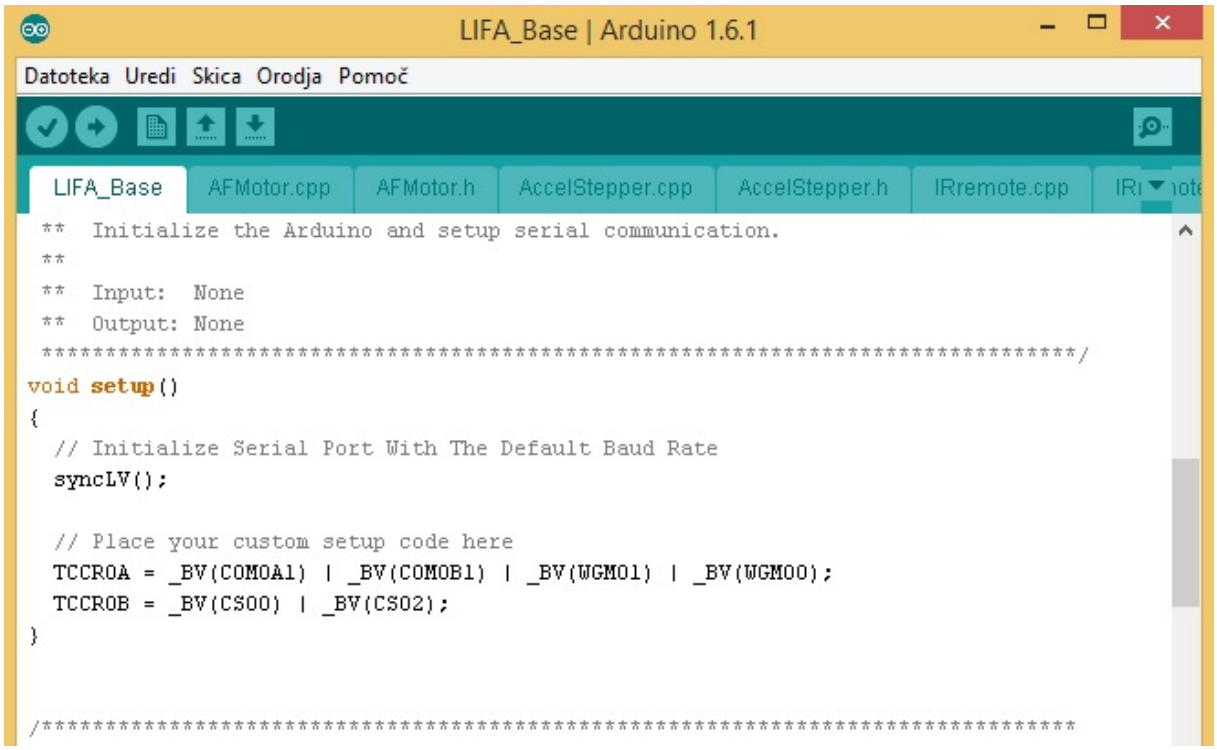
Odpre se Arduino skica LIFA_Base, kot je prikazano na sliki 16.

Slika 16: Arduino skica LIFA_Base.

Uporabimo lahko razvojno ploščo Arduino Uno ali Arduino Mega. Mi smo uporabili Arduino Uno, vrata pa so bila v našem primeru COM 3. Program LIFA_Base zapišemo v razvojno ploščo Arduino Uno. Ko je program zapisan, lahko okolje Arduino zapremo.

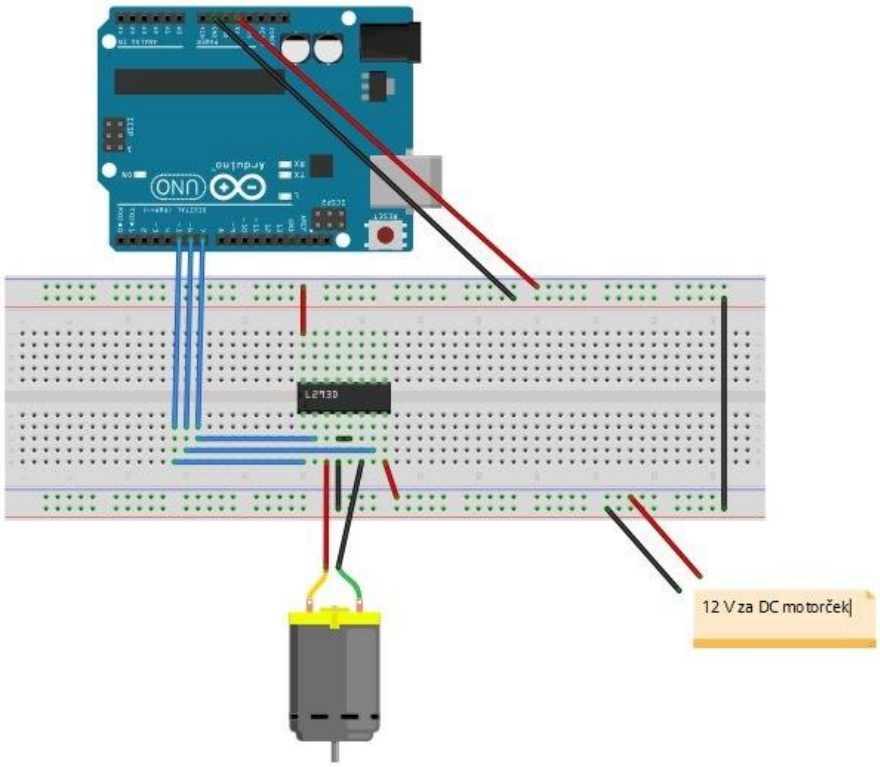
4.4 Krmiljenje DC motorčka z LabVIEW in Arduino Uno:

V razvojno ploščo Arduino Uno naložimo LIFA_Base, kjer pod nastavitveno funkcijo void setup() dodamo spremenjene lastnosti časovnika Timer0, da dosežemo frekvenco PWM signala na pinu 5 v vrednosti 61 Hz. Na pin 5 razvojne plošče Arduino Uno namreč priključimo Enable pin gonilnika L293.



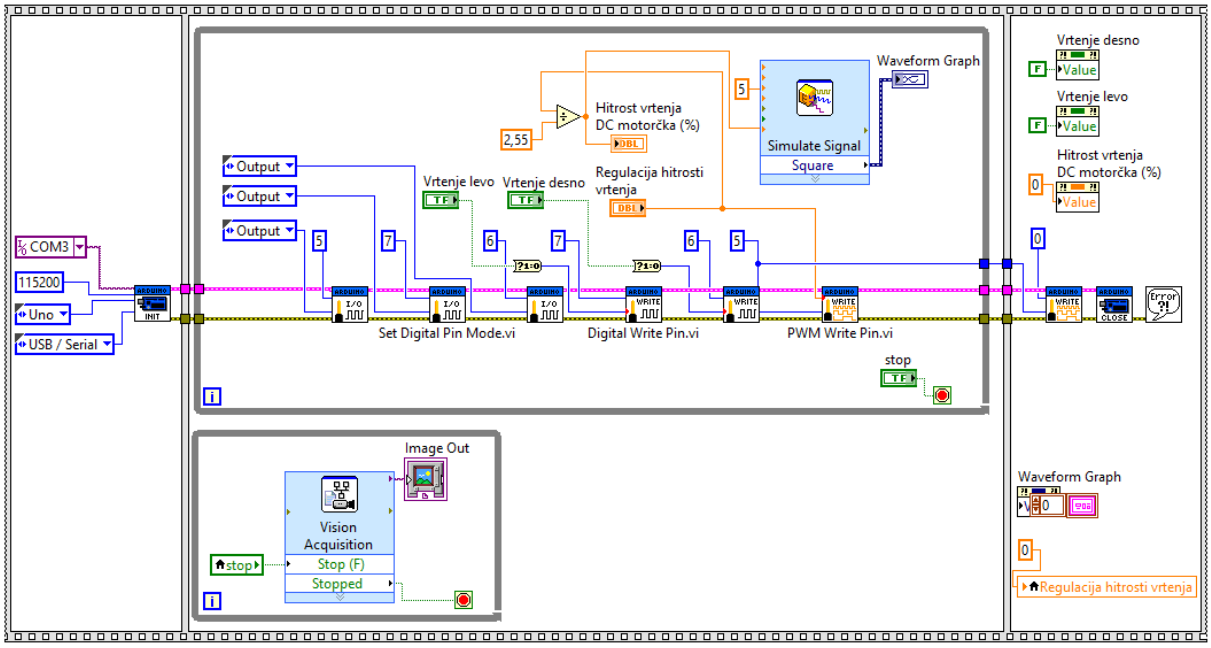
Slika 17: Nastavitev registrov časovnika Timer0 za frekvenco PWM 61 Hz.

V odvisnosti od stanja pina Enable in kontrolnih pinov 2 in 7 gonilnika L293, krmilimo DC motorček. Elemente smo priključili na razvojno ploščo Arduino Uno kot prikazuje slika 18.

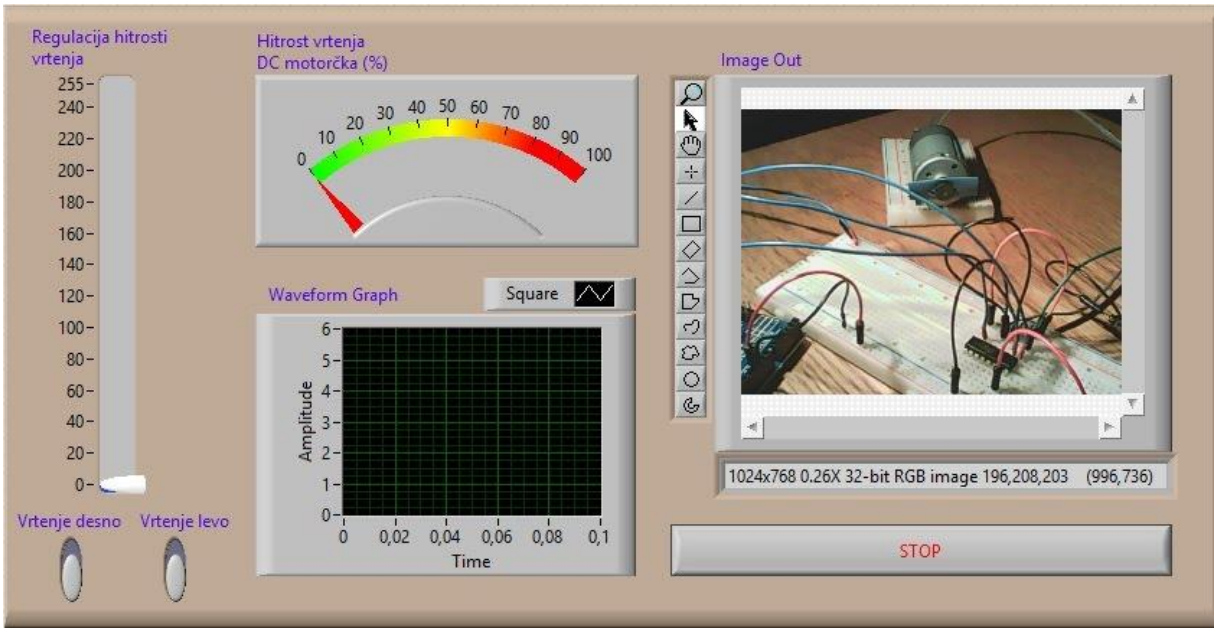


Slika 18: Priključitev elementov vezja na razvojno ploščo Arduino Uno.

Poglejmo še Blok diagram in Čelno ploščo programa za krmiljenje enega DC motorčka, izdelanega v okolju LabVIEW. Krmilimo lahko smer vrtenja ter hitrost vrtenja DC motorčka.



Slika 19: Blok diagram krmiljenja DC motorčka.



Slika 20: Čelna plošča krmiljenja DC motorčka.

Levi okvir Flat sekvence: Nastavimo začetne nastavitve, COM vrata, hitrost prenosa in tip razvojne plošče Arduino.

Sredinski okvirju Flat sekvence: V tem okvirju imamo dve zanki While, eno za algoritem programa krmiljenja DC motorčka in drugo za Web kamero. Obe zanki While se izvajata neodvisno, za zaustavitev izvajanja obeh hkrati pa uporabimo le en gumb Stop. V ta namen uporabimo lokalno spremenljivko gumba Stop. Z miško se postavimo nad gumb Stop in v njegovem priročnem meniju

izberemo *Create > Local Variable*. Lokalni spremenljivki v priročnem meniju izberemo *Change To Read* in jo nadomestimo z gumbom Stop v drugi zanki While. Če želimo z gumbom Stop ustaviti izvajanje obeh zank While, moramo gumbu Stop v priročnem meniju spremeniti privzeto nastavitev *Operation* na *Switch until released*.

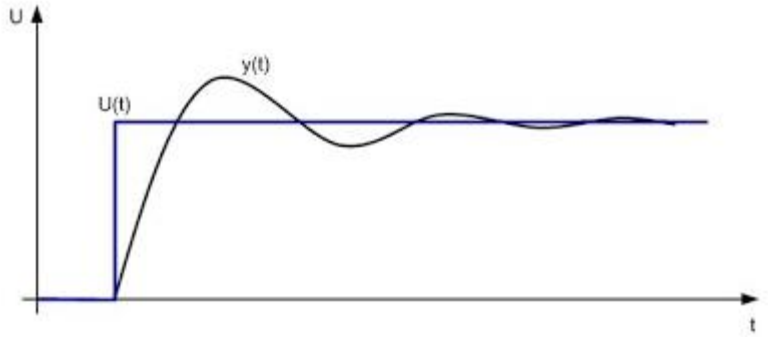
Druga zanka While se nam je avtomatsko kreirala, ko smo na paleti Functions izbrali Vision Acquisition (*Functions > Vision and Motion > Vision Express > Vision Acquisition*). V kolikor palete Vision and Motion ni na paleti Functions, jo prenesemo iz spleta, razširimo in instaliramo.

Krmilna pina 2 in 7 gonilnika L293 sta priključena na pina 7 in 6 razvojne plošče Arduino Uno, Enable pin 1 gonilnika L293 pa je priključen na pin 5 razvojne plošče Arduino Uno. Zato te pine (5, 6 in 7) z bloki Set Digital Pin Mode.vi določimo kot izhodne. Z gumboma Vrtenje desno in Vrtenje levo spreminjamo digitalno stanje na pinu 6 in pinu 7 Arduino ploščice, zato uporabimo dva bloka Digital Write Pin.vi. Ta dva pina sta povezana z gonilnikom L293, pin 6 Arduino s pinom 7 gonilnika in pin 7 Arduino s pinom 2 gonilnika L293. V odvisnosti od logičnega stanja teh dveh pinov krmilimo smer vrtenja DC motorčka (glej tabelo 1). Na pinu 5 Arduino, ki je povezan z Enable pinom 1 gonilnika L293, generiramo PWM signal (PWM Write Pin.vi), pin 5 Arduino je izhodni PWM pin 8-bitnega časovnika Timer0 Arduino Uno. V odvisnosti od Duty Cycle tega PWM signala se spreminja hitrost vrtenja DC motorčka. Da lahko spremljamo PWM signala na grafu (Waveform Graph) uporabimo virtualni instrument Simulate Signal, ki ga najdemo na paleti *Functions > Express > Input > Simulate Signal*. V njegovem priročnem meniju izberemo pravokotne signale (Square) in amplitudo 5.

Desni okvir Flat sekvence: Izvede se po pritisku na gumb Stop. Postavimo Duty Cycle PWM signala na 0, ustavimo komunikacijo z Arduino Uno (Close) ter inicializiramo gumba za izbiro smeri vrtenja DC motorčka, prikaz hitrosti vrtenja DC motorčka in grafični prikaz PWM signala. V priročnih menijih obeh gumbov in indikatorju hitrosti vrtenja izberemo *Create > Property Node > Value*, jih odložimo v desni okvir Flat sekvence in jih v priročnem meniju z izbiro parametra Change All To Read spremenimo v Change All To Write. Gumboma na terminalu Value dodelimo vrednost False, indikatorju hitrosti vrtenja pa 0. Tudi grafu Waveform Graph v priročnem meniju izberemo Property Node in mu na vhodnem terminalu Value dodelimo vrednost konstante (Constant). Tej v celoti ali v notranjem večjem okvirju (naš primer) spremenimo pogled v View Cluster As Icon. Prikaz PWM signala na grafu se bo v tem primeru po izklopu virtualnega instrumenta s pritiskom na gumb Stop izbrisal. Če želimo, da se po pritisku na gumb Stop postavi drsnik kontrole Regulacija hitrosti vrtenja na nič, kontroli izberemo lokalno spremenljivko in ji v desnem okvirju Flat sekvence dodelimo vrednost 0.

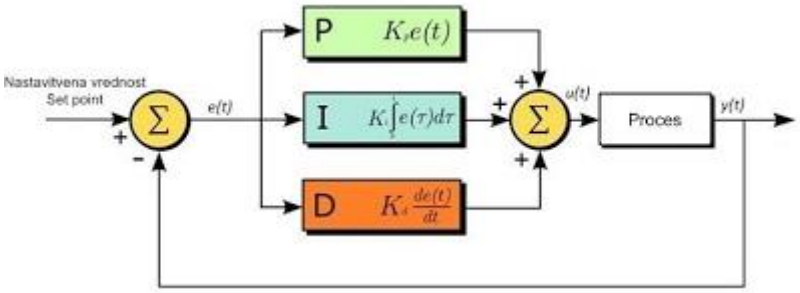
4.5 PID regulacija hitrosti vrtenje DC motorčka

Regulacija je zaprtozančno vodenje procesa, ki na osnovi primerjave dejanske in želene regulirane veličine na proces vpliva tako, da je napaka čim manjša. Poznamo dva načina delovanja regulacij: sledilno in regulacijsko. Pri sledilnem mora regulacijski sistem zagotoviti, da regulirana veličina dobro sledi referenci, spremembi nastavitvene točke (npr. primerno odzivanje DC motorčka ob spremembah nastavitvene točke). Pri regulacijskem delovanju pa je pomembno predvsem to, da regulator učinkovito odpravlja motnje.



Slika 21: Potek referenčne U(t) in regulirane y(t) veličine pri sledilni regulaciji.

Proporcionalno-integralno-diferencialna regulacija ali regulacija PID je najbolj razširjena metoda v procesni industriji. Regulacija je izvedena kot vsota proporcionalnega (P), integralnega (I) in diferencialnega (D) člena. Člen P je neposredno odvisen od trenutne velikosti napake, člen I je odvisen od integrala preteklih napak, člen D pa od hitrosti spreminjanja napake, s katerim lahko deloma predvidimo prihodnje delovanje sistema.



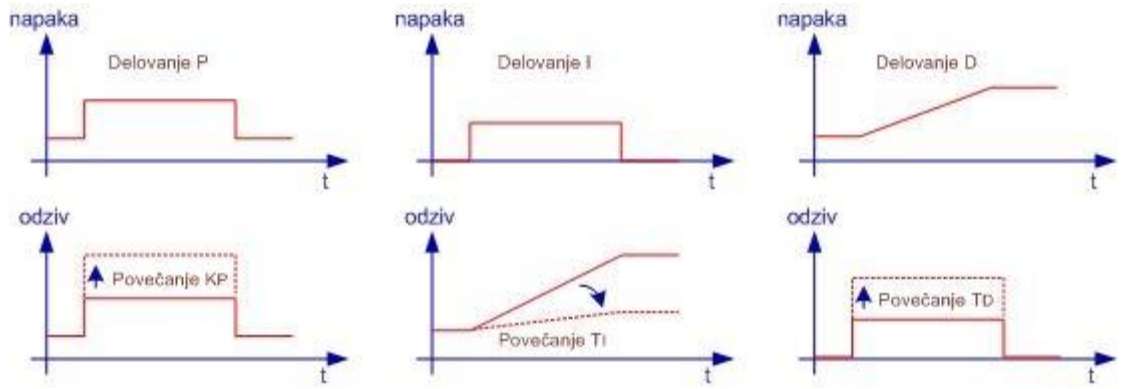
Slika 22: Blok shema regulacije PID.

KP = proporcionalni prirastek
KI = integralni prirastek
KD = diferencialni prirastek
e(t) = napaka, ki predstavlja razliko med nastavitveno vrednostjo in vrednostjo procesa, y(t)

Spodnja enačba opisuje delovanje regulacije PID:

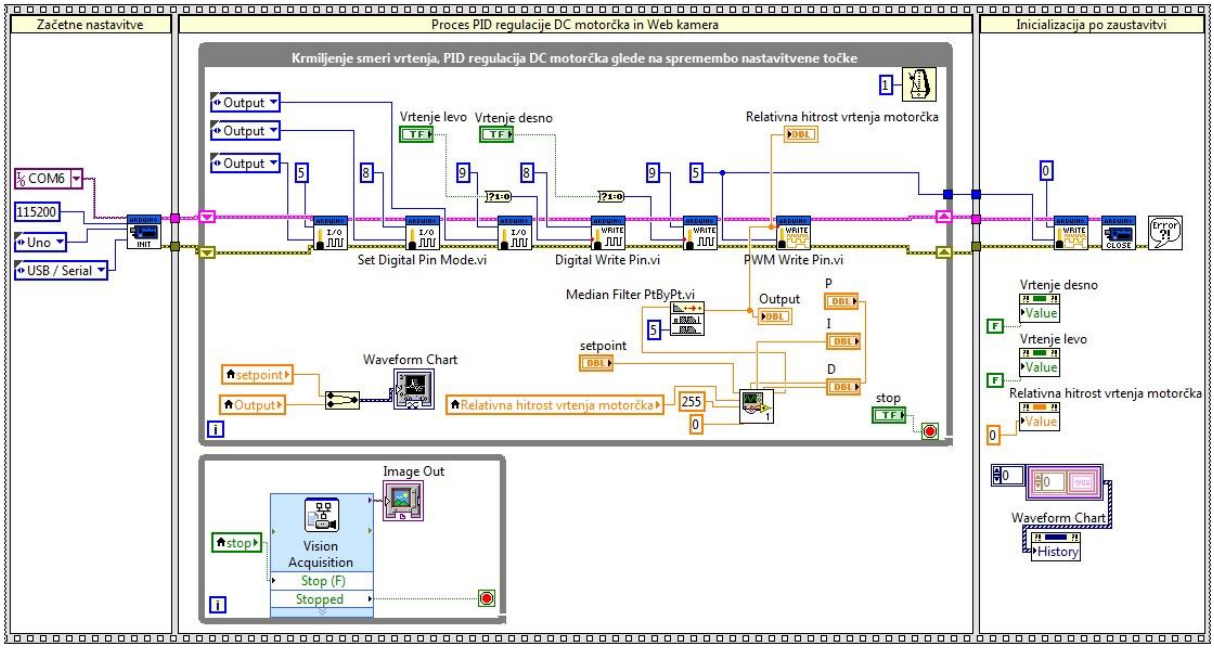
$$u(t) = K_P \cdot e(t) + K_I \cdot \int_0^t e(t) + K_D \cdot \frac{de(t)}{dt} = K_P(e(t) + \frac{1}{T_I} \cdot \int_0^t e(t) \cdot dt + T_D \cdot \frac{de(t)}{dt})$$

Regulator PID ima tri nastavljive parametre: ojačanje KP, integralno časovno konstanto TI in konstanto diferenciranja TD. Referenčno veličino predstavlja nastavitvena vrednost (Set point), to je vrednost, ki jo želimo v našem procesu.



Slika 23: Proporcionalno, integralno in diferencialno delovanje.

Pri regulatorju PID delujejo vsi trije členi istočasno, zato je odziv regulacije na spremembo nastavitvene točke rezultanta vseh treh členov.



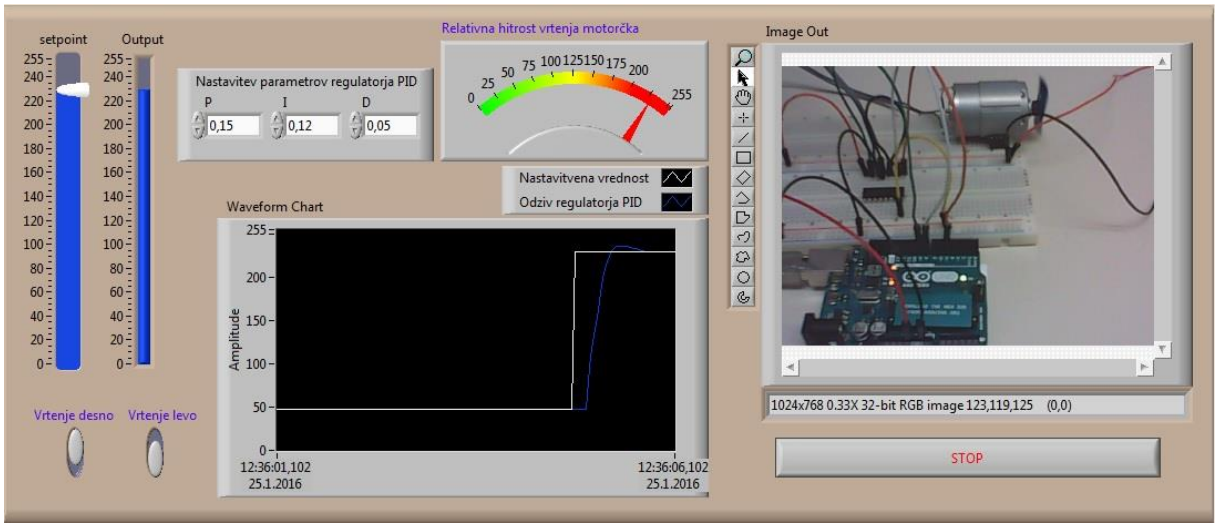
Slika 24: Blok diagram PID regulacije hitrosti vrtenja DC motorčka.

Levi okvir Flat sekvence: Nastavimo začetne nastavitve, COM vrata, hitrost prenosa in tip razvojne plošče Arduino.

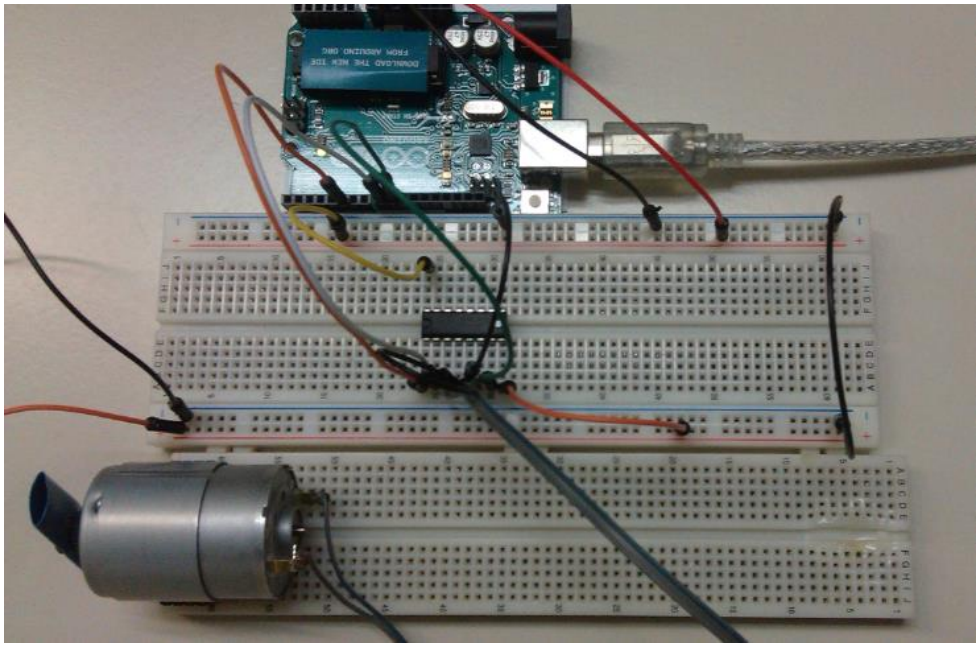
Sredinski okvirju Flat sekvence: V tem okvirju imamo dve zanki While, eno za algoritem programa PID regulacije DC motorčka in drugo za Web kamero.

S tremi funkcijskimi bloki Set Digital Pin Mode.vi smo pine 5, 8 in 9 določili kot izhodne pine. Nato smo dvema funkcijskima blokoma Digital Write Pin.vi povezali gumba za izbiro smeri vrtenja DC motorčka. Funkcijskemu bloku PWM Write Pin.vi smo povezali indikator za prikaz hitrosti vrtenja DC motorčka, indikator Output in izhod Median Filter.vi, ki je povezan z izhodom virtualnega instrumenta za PID regulacijo. Indikator Output nam prikazuje odzivanje regulacije v odvisnosti od spremembe nastavitvene točke (setpoint). Virtualnemu instrumentu za PID regulacijo smo povezali kontrole za spreminjanje parametrov P, I in D. Na grafu lahko spremljamo nastavitveno vrednost in odziv regulatorja PID.

Desni okvir Flat sekvence: Izvede se po pritisku na gumb Stop. Pin 5 na razvojni plošči Arduino Uno postavimo na 0 in s tem zaustavimo vrtenje DC motorčka. Inicializiramo gumba za vrtenje v levo oziroma za vrtenje v desno, indikator za prikaz hitrosti vrtenja DC motorčka in grafični prikaz nastavitvene vrednosti in odziva regulacije PID.



Slika 25: Čelna plošča PID regulacije hitrosti vrtenja DC motorčka.



Slika 26: Elementi priključeni na Arduino Uno.

5. REZULTAT

Krmiljenje enosmernega motorčka z razvojno ploščo Arduino Uno in uporabljenim gonilnikom L293 je potekalo brez večjih težav. Programska koda je dokaj enostavna. Zapletlo se je pri spoznavanju delovanja časovnika Timer0 in spreminjanju njegovih registrov ter registrov, ki vplivajo na njegovo delovanje. To je vzelo veliko časa, saj je bil potreben poglobljen vpogled v delovanje časovnikov mikrokontrolerja ATmega328p, ki skrbi za delovanje razvojne plošče Arduino Uno.
Da smo spremenili frekvenco PWM signala, je krivo to, ker se je motorček pri nižji srednji vrednosti napetosti (nižji duty cycle) zelo težko začel vrteti oziroma se je začel vrteti šele pri dosti višji napetosti. Ko pa smo znižali frekvenco PWM signala na 61 Hz, se je motorček odzival neprimerno bolje. Lahko bi spreminjali frekvenco še na druge vrednosti in opazovali razliko, vendar tega nismo storili.
Z vključitvijo okolja LabVIEW in vzpostavitvijo komunikacije z Arduino Uno, smo izdelali prijazen vmesnik, s katerim uspešno krmilimo DC motorček. Instalirati smo morali dodatek LIFA, da smo dobili na razpolago ustrezno paleto blokov, potrebnih za izdelavo programske kode.
Preden smo se lotili PID regulacije DC motorčka, smo morali obnoviti znanje o zaprtozančnih regulacijah. Ker s funkcijskim blokom PID v okolju LabVIEW nismo dobili dobrih rezultatov, smo iskali rešitve na spletu. Tam smo naleteli na virtualni instrument simplepid-1.vi in ga vključili v naš projekt. Rezultati so bili občutno boljši. S spreminjanjem nastavitvene točke je bila odzivnost hitrosti vrtenja DC motorčka zelo dobra.
Hipoteze, ki smo si jih zastavili na začetku, so se potrdile. Dejstvo pa je, da smo za to porabili veliko časa. Kljub temu, pa nam je porabljeni čas prinesel nova znanja, katera lahko v bodoče izkoristimo.

6. DRUŽBENA ODGOVORNOST

Preciznost in natančnost pri današnjih tehnologijah, v proizvodnih procesih, oddelkih kontrole in razvojnih oddelkih je zelo pomembna. Izdelke, ki jih strokovnjaki razvijajo zahtevajo visoko kvaliteto, saj so le tako lahko tržno zanimivi. Hkrati pa strokovnjaki stremijo k temu, da pri njihovi izdelavi nastaja čim manj izmeta. Zato morajo biti tudi tehnološki postopki v proizvodnji, posamezne operacije, polavtomati in avtomati natančni. Z ustreznimi aplikacijami in z uporabo ustreznih programskih okoljih lahko to dosežemo. Tudi PID regulacija DC motorčka, ki poganja določen proizvodni postopek, lahko pripomore k temu.

7. VIRI

http://rn-wissen.de/wiki/index.php/Timer/Counter_%28Avr%29 (13. 10. 2015)
http://maxembedded.com/2011/06/avr-timers-timer0/ (16. 10. 2015)
https://sites.google.com/site/qeewiki/books/avr-guide/timers-on-the-atmega328
(23. 10. 2015)
https://decibel.ni.com/content/groups/labview-interface-for-arduino (7. 11. 2015)
http://www.electricaltechnology.org/2015/10/what-is-pid-controller-how-it-works.html
(7. 11. 2015)
https://sites.google.com/site/solaelektronikesers/home (12. 11. 2015)