Razvoj pretvornika STEP/DIR signala u analogni upravljački signal za servo motor

Petek, Nikola

Sveučilište
Sjever

Završni rad br. 492/EL/2021

Razvoj pretvornika STEP/DIR signala u analogni upravljački signal za servo motor

Nikola Petek, 36488338

Varaždin, srpanj 2021. godine

Sveučilište Sjever

Odjel za Elektrotehniku

Završni rad br. 492/EL/2021

Razvoj pretvornika STEP/DIR signala u analogni upravljački signal za servo motor

Student

Nikola Petek, 36488338

Mentor

Miroslav Horvatić, dipl. ing

Varaždin, srpanj 2021. godine

Sveučilište Sjever
Sveučilišni centar Varaždin
104. brigade 3, HR-42000 Varaždin

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL: Odjel za elektrotehniku

STUDIJ: preddiplomski stručni studij Elektrotehnika

PRISTUPNIK: Nikola Petek | JMBAG: 36488338

DATUM: 23.08.2021. | KOLEGIJ: Automatsko upravljanje

NASLOV RADA: Razvoj pretvornika STEP/DIR signala u analogni upravljački signal za servo motor

NASLOV RADA NA ENGL. JEZIKU: Development of STEP / DIR signal to analog control signal converter for servo motor

MENTOR: Miroslav Horvatić, dipl. ing. | ZVANJE: viši predavač

ČLANOVI POVJERENSTVA:
1. mr. sc. Ivan Šumiga, dipl. ing., viši predavač
2. mr. sc. Matija Mikac, dipl. ing., viši predavač
3. Miroslav Horvatić, dipl. ing., viši predavač
4. dr. sc. Ladislav Havaš, docent, rezervni član
5.

Zadatak završnog rada

BROJ: 492/EL/2021

OPIS:

Potrebno je projektirati i izraditi pretvornik signala koji STEP/DIR signale namijenjene upravljanju koračnim elektromotorom pretvara u analogni signal mjernog područja od -10V do 10V, a koristi se za upravljanje analognim servo motorima. Pretvornik realizirati korištenjem STM32F411 mikrokontrolera.

U radu je potrebno:
- opisati elektromotorni pogon u kojem će se koristiti pretvornik signala
- osmisliti sklopovski i programski dio pretvornika signala
- realizirati sklopovski i programski dio pretvornika signala
- ispitati rad pretvornika signala.

ZADATAK URUČEN: 31.8.2021. | POTPIS MENTORA

Predgovor

Zahvaljujem mentoru Miroslavu Horvatiću, dipl. ing. na pomoći, savjetima prilikom izrade završnog rada.

Zahvaljujem svim profesorima i asistentima Sveučilišta Sjever na prenesenom znanju tijekom studiranja.

Sažetak

Ideja ovog završnog rada je napraviti sklop koji pretvara STEP/DIR digitalni upravljački signal namijenjen upravljanju stepper motora u analogni signal od -10V do +10V namijenjen upravljanju analognih servo motora. Za pokretanje analognih servo motora Omron R88M-U40030VA-S1 preko drivera Omron R88D-UA12V koristi se sklop upravljan mikrokontrolerom STM32F411 koji na temelju povratne veze enkodera servomotora te STEP/DIR signala na ulazu regulira položaj osovine servo motora. Položaj osovine se regulira PID regulatorom koji na temelju razlike zadane i stvarne vrijednosti određuje vrijednost PWM izlaza. Za dobivanje analognog signala od -10V do +10V, potrebnog na izlazu sklopa, PWM signal se pretvara u analogni signal pomoću analognog filtra koji sadrži operacijska pojačala.

KLJUČNE RIJEČI: mikrokontroler, STM32, operacijsko pojačalo, servo motor, servo driver

Abstract

Idea of this paper is to make electrical circuit which converts STEP/DIR digital control signal used to control a stepper motor into an analog signal from -10V to + 10V intended to control analog servo motors. To run analog Omron servo motors R88M-U40030VA-S1 via Omron R88D-UA12V driver, a circuit controlled by STM32F411 microcontroller is used, this circuit regulates the position of the servo motor shaft based on the feedback signal of the servo motor encoder and STEP / DIR signal at the input. The position of the shaft is regulated by a PID controller which determines the value of the PWM output based on the difference between the setpoint and the process value. To get the analog signal from -10V to + 10V required at the output of the circuit, the PWM signal is converted to an analog signal using an analog filter which contains operational amplifiers.

KEYWORDS: microcontroller, STM32, operational amplifier, servo motor, servo drive

Popis korištenih kratica

PWM	Pulsno širinska modulacija
AC-DC	Izmjenična struja - istosmjerna struja

List of abbreviations

PWM	Pulse width modulation
AC-DC	Alternating current – Direct current

Sadržaj

Uvod

Zbog porasta zahtjeva za većom brzinom i preciznošću u industriji koriste se servo motori. Njihove karakteristike čine ih idealnim za korištenje u strojevima za obradu metala, drva, plastike, kao i kod automatizacije proizvodnih procesa koji zahtijevaju visoku preciznost. Zbog navedenih povoljnih svojstava i u ovom završnom radu će se razmotriti elektromotorni pogon koji sadrži servo elektromotore.

Cilj završnog rada je omogućiti upravljanje servo motora s analognim ulazom od -10V do +10V uz pomoć signala koji daje sklop za upravljanje CNC strojem namijenjen za stepper motore. Zbog visoke cijene sklopova koji omogućuju direktnu komunikaciju između servo motora i računala za upravljanje NC strojem odlučeno je da će se u sklopu ovog završnog rada izraditi pretvornik signala stepper motora za koje su sklopovi mnogo jeftiniji. Zadatak pretvornika je da na temelju izlaza iz računala namijenjenih stepper motorima (STEP/DIR) i položaja enkodera servo motora izračuna potreban napon na analognom ulazu servo elektromotornog pogona kako bi se servo motor pomaknuo na željenu poziciju.

1. Servo motor

1.1. Općenito

Slika 1. Servo motori s driverima [1]

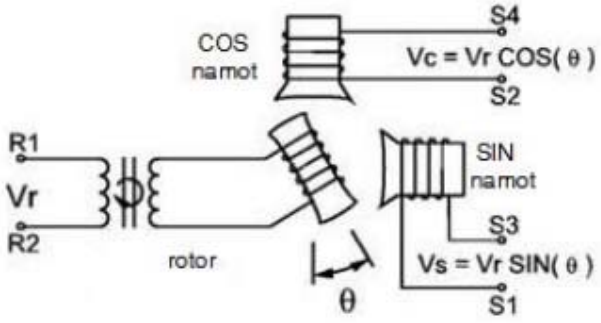
Servo motori su elektromehanički pretvornici koji se koriste za preciznu kontrolu položaja, brzine i akceleracije u proizvodnom procesu. Kako bi se postigla željena pozicija vratila elektromotora koristi se mjerni član položaja vratila i regulacijski sustav koji sadrži negativnu povratnu vezu. Na temelju povratne veze i ulaznog signala servo pogonskoi uređaj vrši proračun vrijednosti koju je potrebno dovesti na stezaljke motora. Servo motori i sustavi za određivanje položaja vratila elektromotora mogu se grupirati prema načinu izvedbe. Servo sustavi najčešće koriste istosmjerne bezkolektorske motore ili sinkrone motore s permanentnim magnetima, a za određivanje položaja vratila elektromotora najčešće se koriste inkrementalni i apsolutni enkoderi te rezolveri.

1.2. Određivanje položaja motora

1.2.1. Rezolveri

Slika 2. Rezolver [2]

Rezolver je induktivni mjerni pretvarač koji daje analogni signal. Radi na principu elektromagnetskog polja između dva bliska vodiča. Sastoji se od 3 namota. Jedan namot je primarni, a preostala dva namota su sekundarni namoti prostorno pomaknuti za 90 stupnjeva. Sekundarni namoti se nazivaju sinusni i kosinusni namot. Na primarni namot je priključen izmjenični napon visoke frekvencije te se na sekundarnim namotima inducira napon slično kao kod transformatora. Promjenom kuta rotora mijenja se kut kojim silnice prolaze kroz sekundarne namote što znači da dolazi do promjene amplitude induciranog napona. Na temelju mjerenih napona na stezaljkama sekundarnih namota može se izračunati kut rotora. [10]



Slika 3. Skica rezolvera [3]

Ako se na primarni namot spoji napon u_0:

$$u_0 = U_0 \sin \omega t$$

Inducirani naponi na sekundarnim namotima iznose:

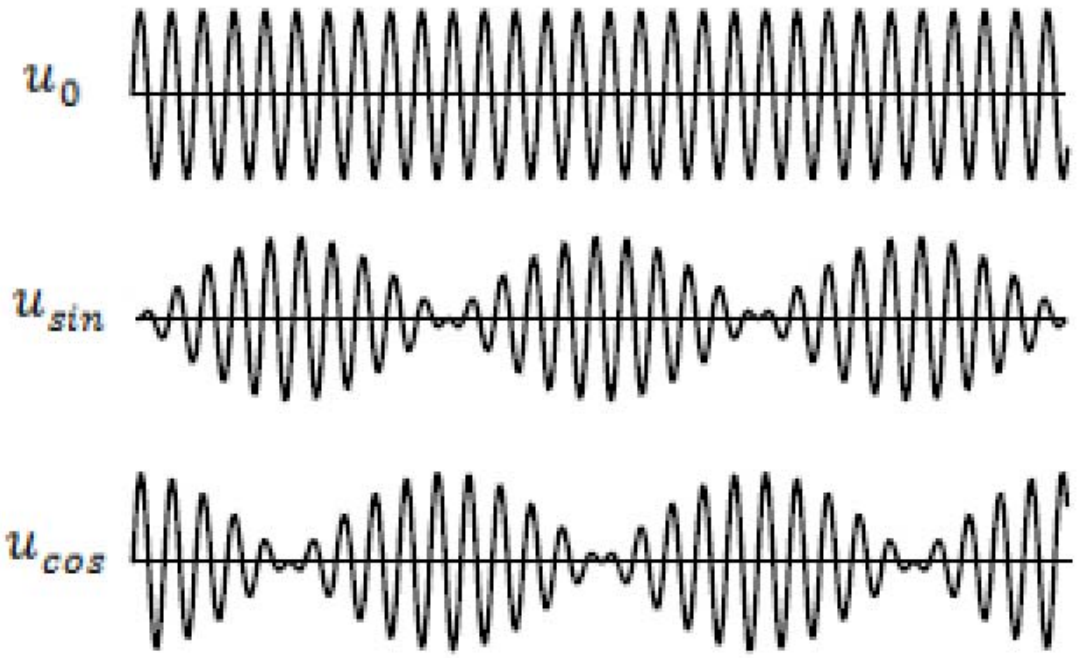
$$u_{sin} = k u_0 \sin(p\varepsilon) = k U_0 \sin(\omega t) \sin(p\varepsilon)$$

$$u_{cos} = k u_0 \cos(p\varepsilon) = k U_0 \sin(\omega t) \cos(p\varepsilon)$$

Gdje je:

p – broj pari polova

ε – kut zakreta rotora



Slika 4. Signal rezolvera [4]

1.2.2. Enkoderi

Slika 5. Enkoder [5]

Enkoderi su digitalnu mjerni pretvornici koji prema izvedbi mogu biti kontaktni i beskontaktni. Kontaktni enkoderi imaju pinove ili četkice koje prenose signal s rotacijskog djela, a beskontaktni enkoderi koriste optičke, magnetske ili kapacitivne senzore za određivanje položaja. Osim toga, enkoderi se mogu podijeliti na inkrementalne i apsolutne. Inkrementalni enkoderi na izlazu daju impulse koji označavaju promjenu zakreta te se brojenjem impulsa može odrediti relativna pozicija rotora dok kod apsolutnih enkodera postoji jedinstvena binarna riječ koja odgovara trenutnom kutu zakreta i predstavlja točan položaj rotora bez potrebe za prethodnim referenciranjem položaja.

- **Inkrementalni enkoder**

Inkrementalni enkoderi se koriste za određivanje brzine vrtnje ili relativnog pomaka u odnosu na nultu poziciju. To rade tako da na izlazu daju 2 pulsirajuća signala koji se broje kako bi se odredio pomak, a praćenjem redoslijeda promjene signala određuje se smjer vrtnje. Jednostavni su za korištenje i jeftini za proizvesti, a mana im je da u slučaju nestanka električne energije ne daju informaciju o apsolutnom položaju.

- **Apsolutni enkoder**

Za razliku od inkrementalnog enkodera koji mjere relativni pomak, apsolutni enkoderi mjere točan položaj osovine. Svaka pozicija apsolutnog enkodera ima poseban binarni kod pozicije. Obrada informacija apsolutnog enkodera je složenija od inkrementalnih, a zahtijevaju i veći broj ulaza. Osim toga skuplji su za proizvesti, ali nakon nestanka napajanja i ponovnog pojavljivanja električne energije apsolutni enkoderi daju točnu informaciju o poziciji u kojoj se nalaze.

1.3. Servo motor Omron R88M-U40030VA-S1

Servo motor Omron R88M-U40030VA-S1 je sinkroni motor s permanentnim magnetima koji koristi inkrementalni enkoder za mjerenje pomaka. Neke od prednosti sinkronih motora s permanentnim magnetima su:

- Nema gubitaka u rotoru
- Laka odvodnja gubitaka u željezu i bakru statora što znači da nema potrebe za dodatnim hlađenjem
- Visoka korisnost
- Mali moment inercije što im omogućuje brzu akceleraciju i deceleraciju

Slika 6. Omron R88M-U40030VA-S1

Karakteristike motora Omron R88M-U40030VA-S1:

- Nazivni napon: 200 V
- Nazivna struja: 2,6 A
- Nazivna snaga: 400 W
- Brzina: 3000 r/min
- Moment: 1,27 Nm
- Rezolucija enkodera: 2048 p/rev

[11]

Slika 7. Natpisna pločica motora

1.4. Pogonski uređaj servo motora Omron R88D-UA12V

Omron R88D-UA12V je pogonski uređaj (driver) servo motora sa sljedećim specifikacijama:

- Nazivni napon: 200 V
- Ulazna struja: 6 A
- Izlazna struja: 2,6 A
- Nazivna snaga: 400 W
- Način upravljanja: ±10 V upravljački signal za kontrolu brzine ili momenta [11]

Slika 8. Omron R88D-UA12V

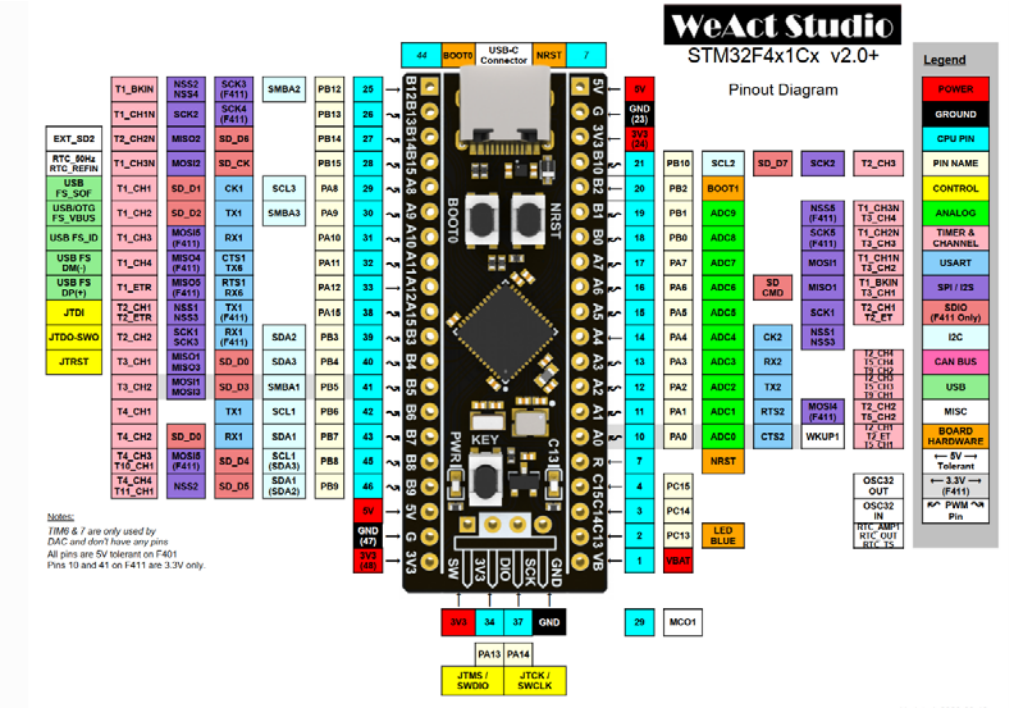
2. Sklopovlje pretvornika signala

U sljedećim poglavljima su navedene komponente korištene u pretvorniku signala. Za svaku komponentu su istaknute karakteristike, razlog odabira navedene komponente kao i uloga pojedine komponente u sklopu.

2.1. Mikrokontroler - STM32F411

STM32F411 je mikrokontroler proizvođača ST microelectronics koji ima visoke performanse i velik broj perifernih jedinica [12]:

- ARM procesor takta 100MHz koji sadrži aritmetičko-logičku jedinicu s pomičnim zarezom
- 3 USART-a do 12,5Mbps
- 5 SPI do 50Mbps
- 3 I^2C do 1Mbps
- SDIO do 48MHz
- USB OTG
- 2 full duplex I^2S
- 3 simplex I^2S
- 12 bitni ADC
- 11 timera na 100MHz



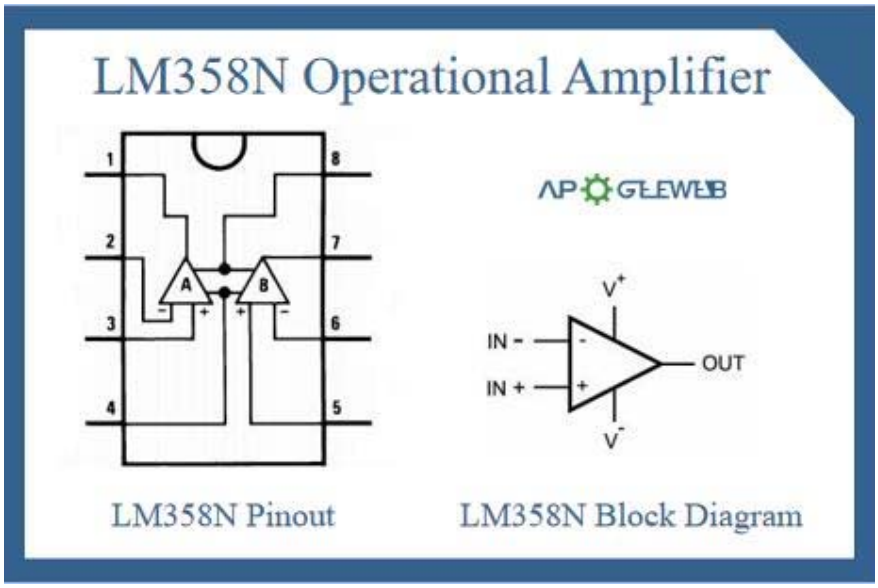
Slika 9. STM32F411 [6]

Iz navedenih specifikacija mikrokontrolera ističu se procesorska jezgra s pomičnim zarezom koja je vrlo učinkovita kod proračuna PID regulacije. Timeri ovog mikrokontrolera spojeni su na vanjske pinove što im omogućuje naprednije mogućnosti kao što su spajanje enkodera direktno na timer, upravljanje PWM modulacijom izlaza direktno iz timera, brojanje vanjskih signala, integrirani komparator. Korištenjem dodatnih mogućnosti timera kao što je ulaz za enkoder značajno smanjujemo opterećenje jezgre procesora koja se može koristiti za bitnije stvari kao što su PID regulacija. Mikrokontroler radi na 3.3VDC, ali većina pinova može primiti 5VDC signal, što je u slučaju ovog završnog rada smanjilo potrebu za dodatnim komponentama.

2.2. Operacijsko pojačalo - LM358N

Integrirani krug LM358N sadrži dva operacijska pojačala s navedenim osnovnim karakteristikama [13]:

- Napajanje 32V ili ±16V
- Diferencijalni ulazni napon ±32V
- Ulazni napon od -0.3V do 32V

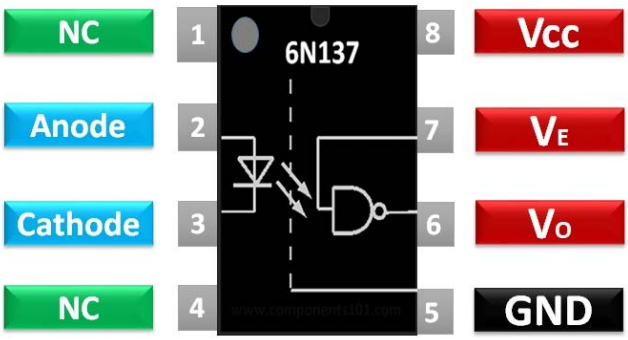


Slika 10. LM358N [7]

Niskopropusni filtar na ulazu pojačala i operacijsko pojačalo omogućuju pretvorbu digitalnog PWM signala iz mikrokontrolera u analogni signal ±10V na izlazu operacijskog pojačala.

2.3. Optoizolator - 6N137

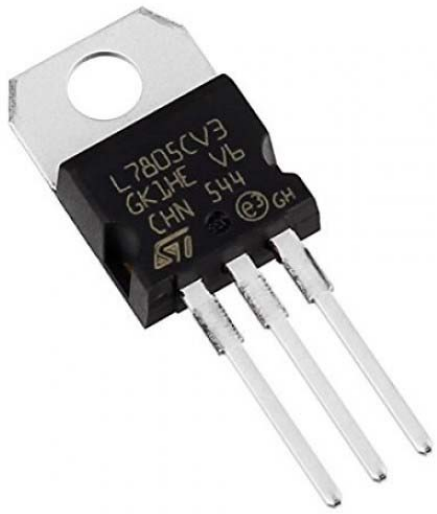
Optoizolator 6N137 je optoizolator brzine do 10 Mbps i radnog napona 5V [14]. Uloga optoizolatora je galvanska izolacija ulaznog kruga od izlaznog kruga. U slučaju pojave nedozvoljenog napona na ulaznom krugu sklopa, optoizolator štiti mikrokontroler i elektroniku koja se nalazi s izlazne strane. [14]



Slika 11. 6N137 [8]

2.4. Regulator napona – 7805

Integrirani krug LM7805 ima ulogu regulatora napona koji na izlazu daje stabilnih 5V. Za ispravan rad, na ulaz mu se može dovesti napon od 7V do 25V. Regulator sadrži toplinsku zaštitu i zaštitu od prevelike struje. [15]



Slika 12. 7805 [9]

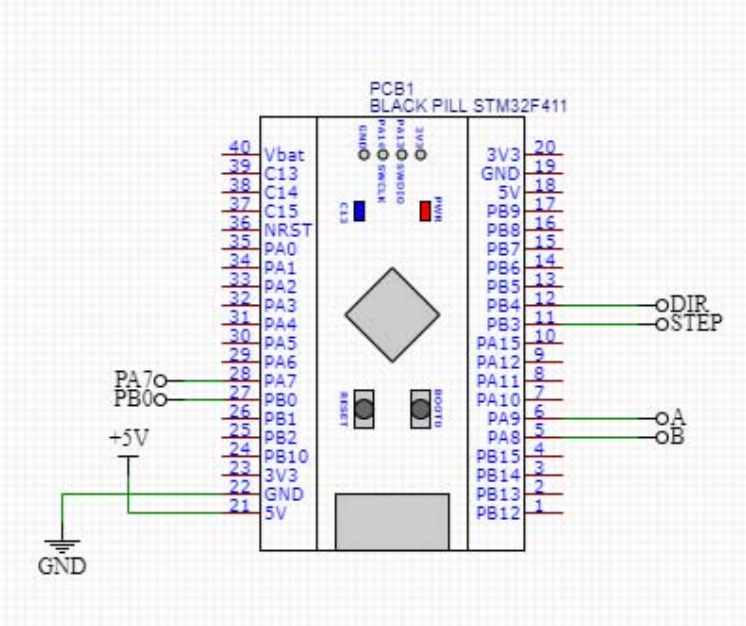
2.5. Napajanje - Meanwell MDR 20 – 12

Meanwell MDR 20-12 je industrijsko switch mode napajanje namijenjeno pričvršćivanju na DIN šinu. Napajanje daje izlazni napon 12V snage 20W. Sadrži zaštite od prenapona, kratkog spoja, preopterećenja i previsoke temperature. Da bi se dobilo bipolarno napajanje ± 12V, u sklopu se koriste dva serijski spojena napajanja.

Slika 13. Meanwell MDR 20 – 12

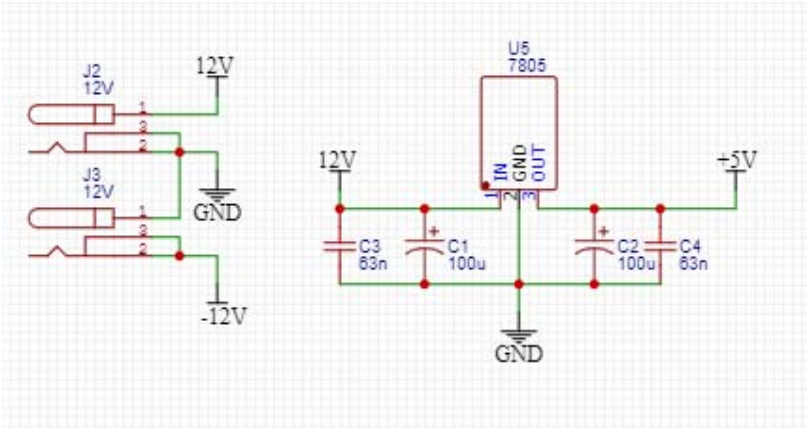
2.6. Shema sklopa

Glavni dio sklopa je naravno mikrokontroler odnosno razvojna pločica „black pill“ mikrokontrolera (Slika 9.) . Na mikrokontroler se spajaju ulazni signali DIR i STEP kao i signal inkrementalnog enkodera A i B. Od izlaza se koriste 2 PWM izlaza spojena na operacijsko pojačalo. Osim ulaza i izlaza svaki mikrokontroler treba i napajanje. Razvojna pločica zahtjeva napajanje 5V te na sebi sadrži regulator napona na 3.3V koji osigurava napajanje mikrokontrolera.



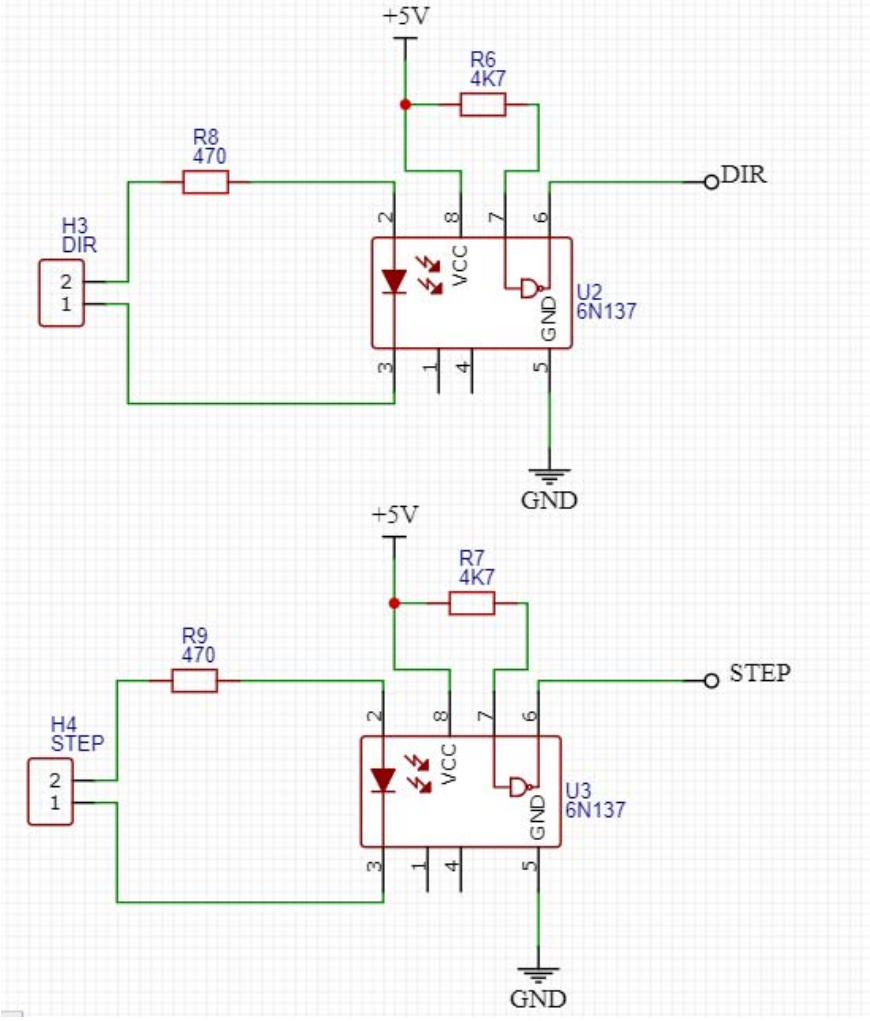
Slika 14. Shema spajanja signala na mikrokontrolersku pločicu

Zbog potrebe za naponom ± 12 V na operacijskom pojačalu, kako bi se na izlazu moglo dobiti ± 10V, sklop sadrži dva DC ulaza na koja se spajaju dva Meanwell napajanja nazivnog napona 12V. Za dobivanje napona 5V potrebnog za rad ostatka elektronike, koristi se regulator napona 7805 s pratećim kondenzatorima koji osiguravaju stabilnost napona na ulazu i izlazu regulatora.



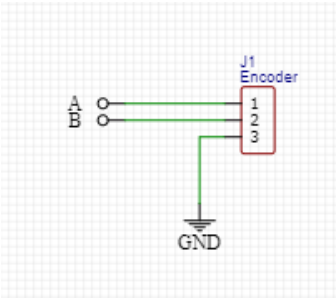
Slika 15. Shema sklopa kojim je realizirano napajanje 5V

Za zaštitu sklopa od nedozvoljenih napona na ulazima koristi se optoizolator 6N137 koji galvanski odvaja izlaze iz računala od ulaza mikrokontrolera i tako štiti mikrokontroler od previsokog napona na ulazu. Isto tako štiti i računalo u slučaju da dođe do kvara na sklopu.



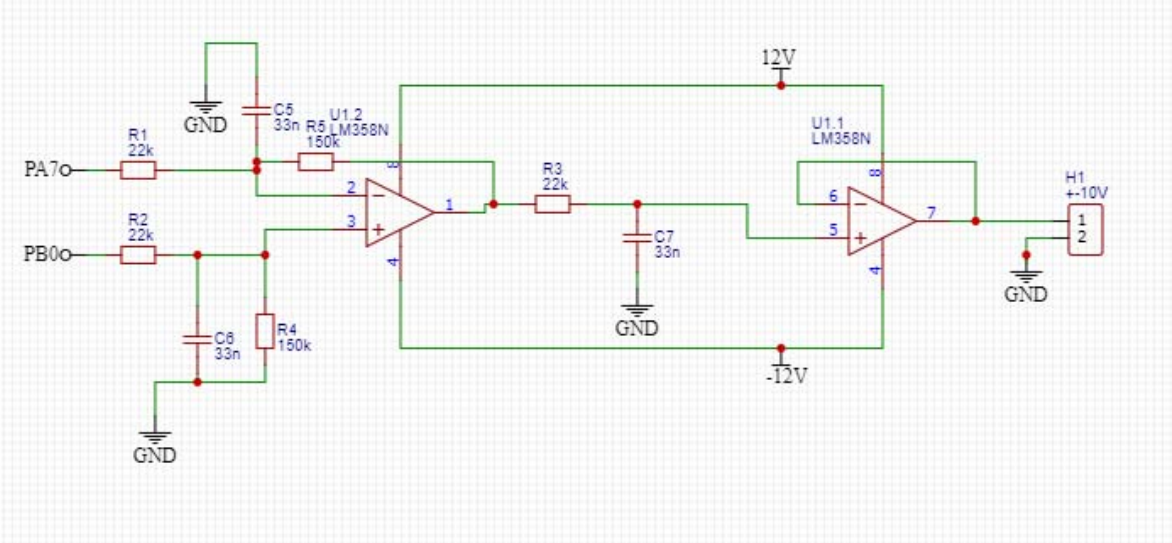
Slika 16. Shema sklopa za galvansko odvajanje signala

Izlazi enkodera spajaju se direktno na ulaze mikrokontrolera. Navedene ulaze mikrokontrolera koristi timer s mogućnošću direktnog čitanja enkodera koji zbog svoje sklopovske izvedbe ne koristi prekid i ne troši vrijeme mikroprocesora ugrađenog unutar mikrokontrolera.



Slika 17. Shema sklopa spajanja enkodera

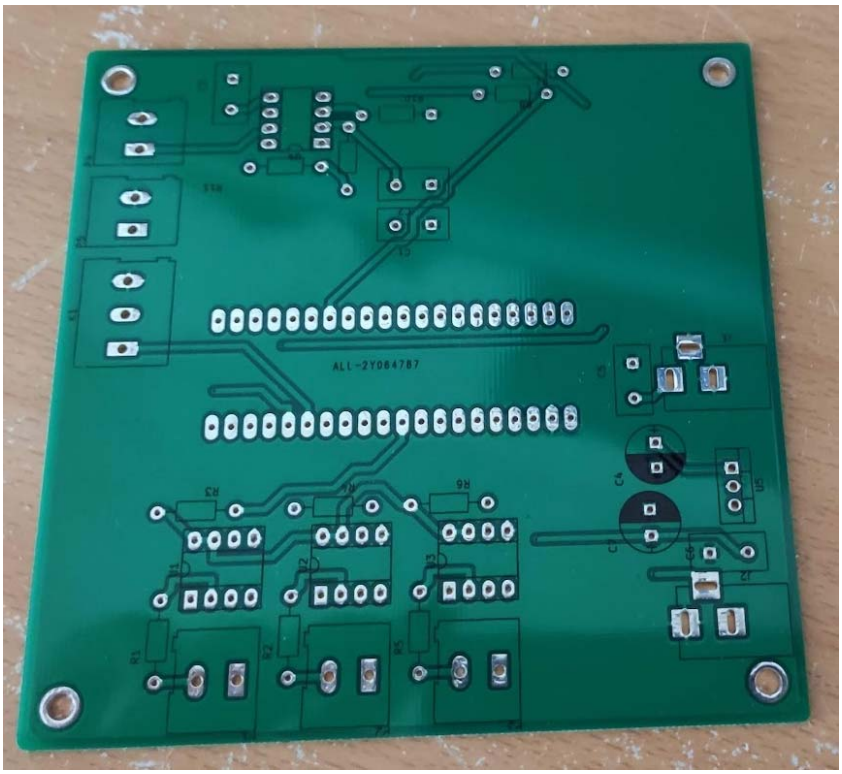
Mikrokontroler STM32F411 nema ugrađen digitalno analogni pretvornik, zbog čega nema mogućnost direktnog analognog izlaza ± 10 V. Za dobivanje željenog analognog izlaza koristi se operacijsko pojačalo LM358N s niskopropusnim filterima na svakom ulazu. Na izlazu PB0 se dovodi PWM signal faktora popunjenosti 50%, a PWM signal na izlazu PA7 se mijenja ovisno o proračunu PID regulatora u mikrokontroleru. Ovakvim sklopom se u slučaju da je faktor popunjenosti signala na PA7 0%, na izlazu sklopa dobiva -10V, a u slučaju faktora popunjenosti 100%, na izlazu sklopa dobiva se +10V.



Slika 18. Shema sklopa za dobivanje analognog izlaznog signala

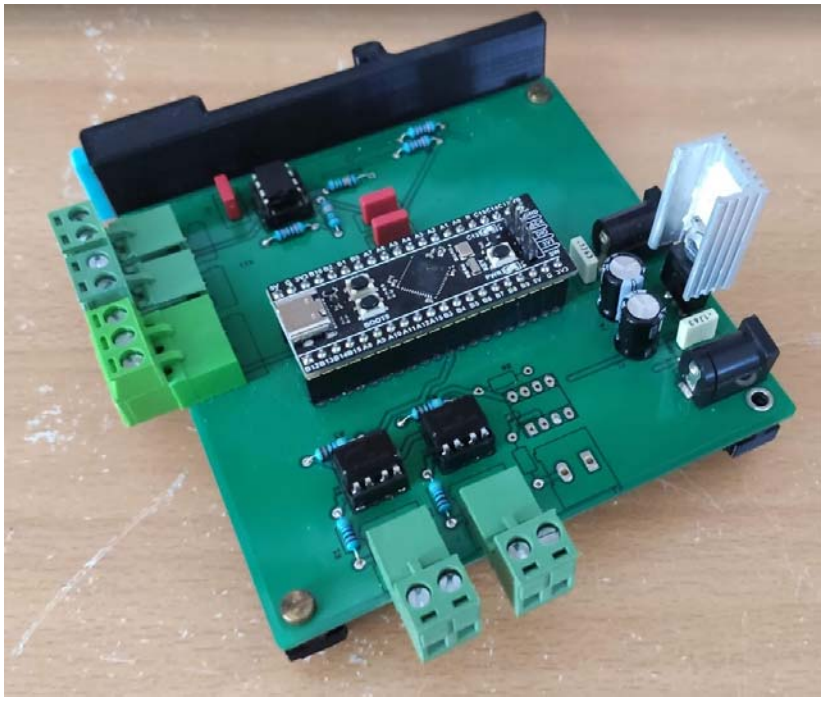
2.7. Tiskana pločica sklopa

Nakon uspješnog testiranja sklopa na univerzalnoj eksperimentalnoj tiskanoj pločici, u programskom paketu KiCad dizajnirana je tiskana pločica. Prototip tiskane pločice je naručen putem internet stranice allpcb.com.



Slika 19. Tiskana pločica sklopa

Nakon dolaska tiskanih pločica bilo je potrebno zalemiti komponente i testirati sklop. Sklop je radio kao i prethodno testirani sklop koji je projektiran na univerzalnoj eksperimentalnoj tiskanoj pločici. Za lakšu montažu pločice u elektroormar dizajniran je i isprintan na 3D printeru, nosač PCB-a za DIN šinu.



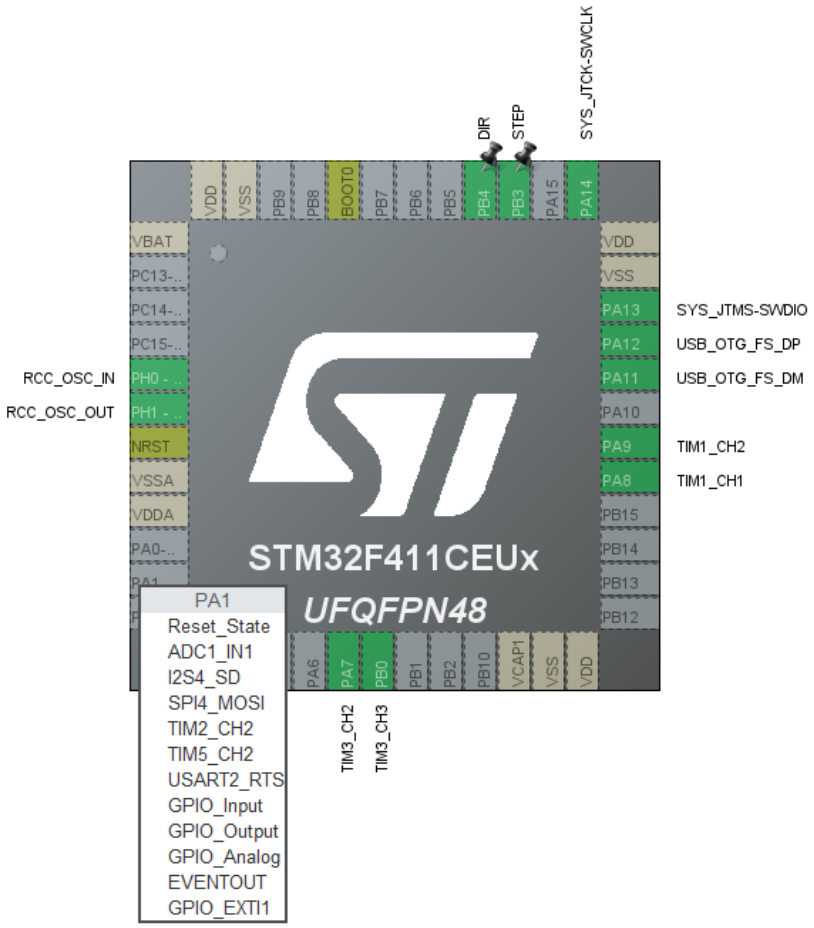
Slika 20. Gotova tiskana pločica pretvornika

3. Programska podrška pretvornika signala

Programiranje mikrokontrolera koji se koristi u sklopovlju pretvornika može se realizirati korištenjem Arduino i STM32 HAL platforme. Nakon istraživanja zaključeno je da STM32 HAL platforma podržava puno bolje iskorištavanje sklopova integriranih u mikrokontroler što omogućuje puno učinkovitije korištenje resursa mikrokontrolera.

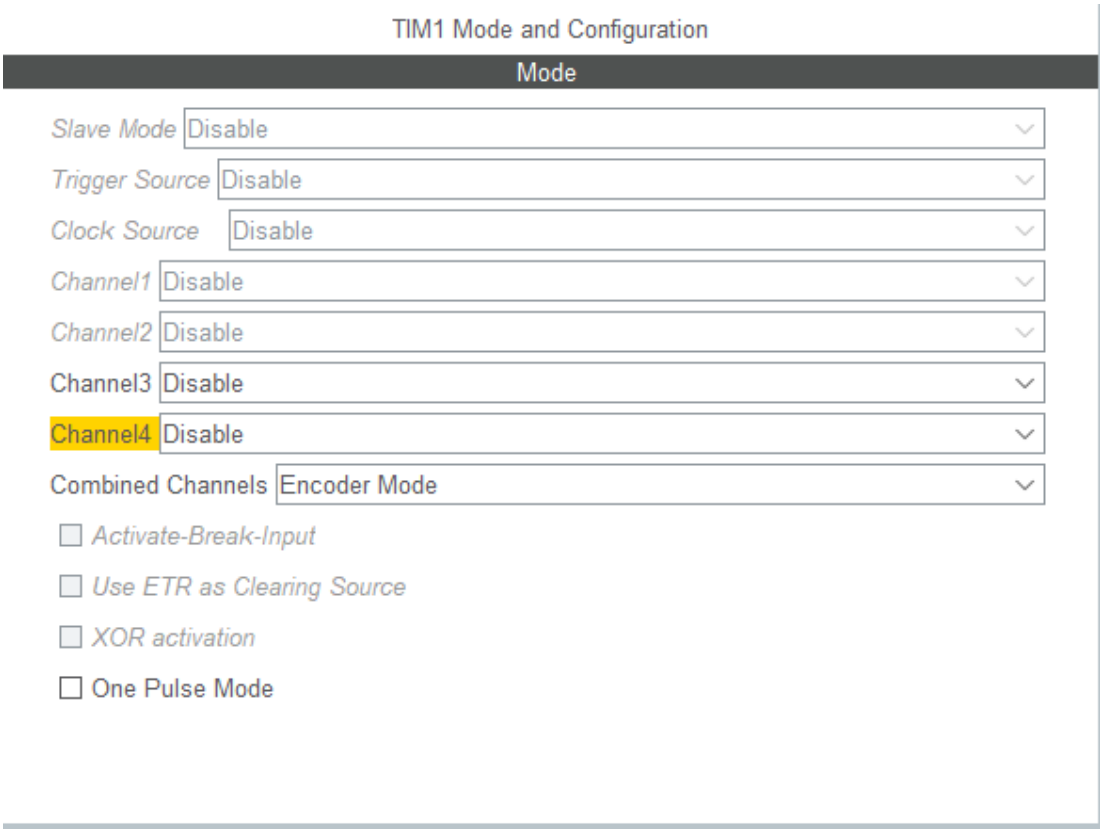
3.1. STM32CUBE IDE

Za programiranje na STM32 HAL platformi korišteno je razvojno okruženje STM32CUBE IDE koje se sastoji od STM32CUBEMX grafičkog sučelja koje generira osnovni kod i popularnog Eclipse IDE. Nakon kreiranja projekta i odabira željenog mikrokontrolera potrebno je u STM32CUBEMX odabrati parametre pojedinih integriranih sklopova. Klikom na pin mikrokontrolera otvaraju se mogućnosti koje taj pin nudi (Slika 21.) . Osim tog pristupa može se odabrati i periferna jedinica koja se želi kontrolirati, nakon čega se može konfigurirati baza za pisanje programa.



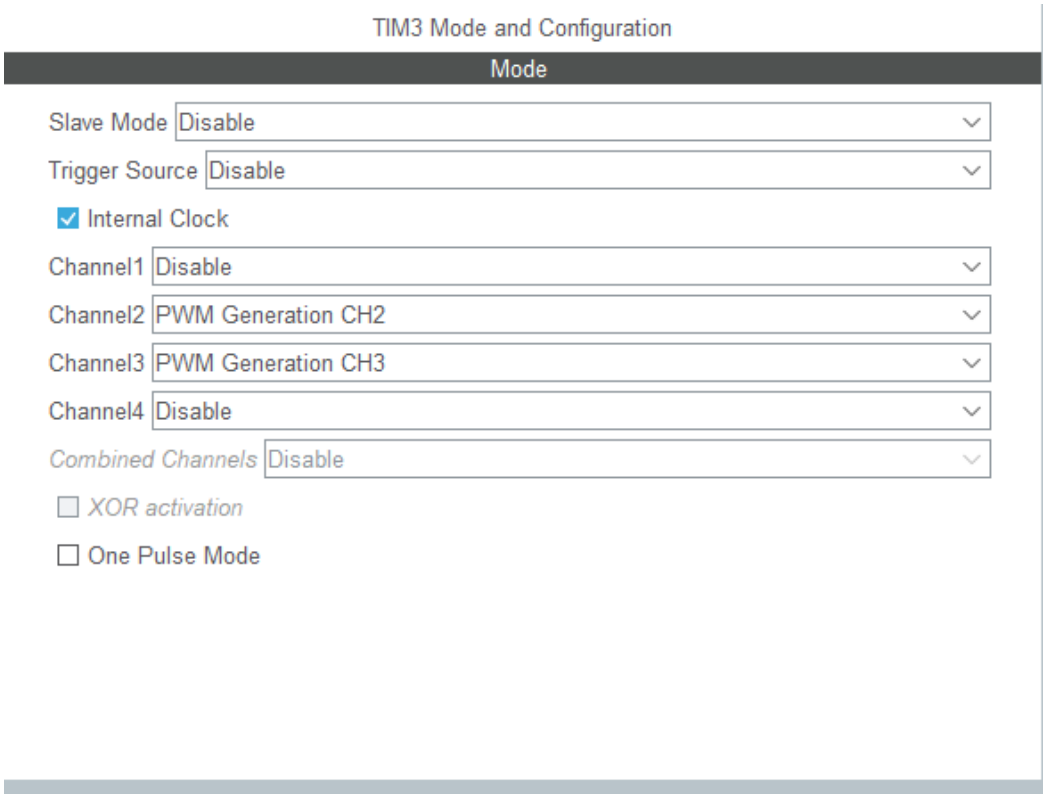
Slika 21. Odabir korištenih pinova unutar STM32CUBE IDE razvojnog okruženja

Kod većine mikrokontrolera pozicija enkodera se određuje prekidima i usporedbom vrijednosti na stezaljkama enkodera, što predstavlja veliko opterećenje za procesor. STM32F411 podržava brojanje impulsa enkodera korištenjem timera. Odabirom timera 1 otvaraju se sve mogućnosti (Slika 22.) . Taj timer se koristi kao brojač impulsa enkodera. Takvim pristupom centralna procesorska jedinica mikrokontrolera ne mora trošiti procesorsko vrijeme na obradu pozicije enkodera nego se sve odrađuje u sklopovski realiziranom timeru. Nakon konfiguracije timera 1 pozicija enkodera dostupna je u registru CNT timera.



Slika 22. Konfiguracija timera 1

Da bi se dodatno smanjilo opterećenje mikroprocesora, za generiranje PWM signala koristi se timer 3 koji ima mogućnost generiranja PWM signala na četiri pina istovremeno (Slika 23.) . Za potrebe generiranja signala ± 10V potrebna su 2 PWM signala na ulazima operacijskog pojačala. Navedene signale generira timer 3 na kanalima 2 i 3 što znači da će PWM signali biti generirani na pinovima PA7 i PB0.



Slika 23. Konfiguracija timera 3

Za čitanje kritičnog ulaza STEP koristi se prekid, a ulaz DIR definiran je kao običan ulaz.

3.2. Program

Nakon odabira svih parametara ulaza i izlaza STM32CUBEMX generira osnovni kod napisan u C programskom jeziku. Korisnički programski kod se upisuje između komentara koji sadrže „USER CODE BEGIN" i „USER CODE END".

```
/* USER CODE BEGIN Includes */
#include "string.h"
#include "usbd_cdc_if.h"
#include "stdbool.h"
/* USER CODE END Includes */
```

Na početku programa potrebno je dodati biblioteke koje se namjeravaju koristiti. U programu mikrokontrolera su potrebne biblioteke "string.h" za korištenje string varijabli, "stdbool.h" za korištenje boolean varijabli i "usb_cdc_if.h" za usb komunikaciju s računalom prilikom postavljanja parametara PID regulatora.

```
/* USER CODE BEGIN PTD */
float map(float val, float I_Min, float I_Max, float O_Min, float
O_Max)
    {
          return(((val-I_Min)*((O_Max-O_Min)/(I_Max-I_Min)))+O_Min);
   }
/* USER CODE END PTD */
```

Za postizanje željenih vrijednosti PWM izlaza mikrokontrolera koristi se funkcija map koju program koristi za skaliranje vrijednosti dobivenih na izlazu PID regulatora na vrijednost potrebnu na PWM izlazu.

```
/* Private variables ---------------------------------------------
---------*/
TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim3;
TIM_HandleTypeDef htim5;

/* Private function prototypes -----------------------------------
---------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM1_Init(void);
static void MX_TIM3_Init(void);
static void MX_TIM5_Init(void);
```

Privatne varijable htim1, htim3 i htim5 su automatski generirane varijable koje predstavljaju timere 1, 3 i 5. Prije korištenja navedenih timera potrebno im je dodjeliti njihovu funkciju. Timer 1 se koristi za čitanje encodera, timer 3 se koristi za generiranje PWM signala na izlazu mikrokontrolera, a timer 5 se koristi za mjerenje vremena proteklog od uključivanja mikrokontrolera. Funkciju svakom timeru dodjeljujemo u funkcijama MX_TIMx_Init (gdje x predstavlja redni broj timera) koje su automatski generirane funkcije u grafičkom sučelju. Osim timera potrebno je inicijalizirati ulaze i izlaze funkcijom MX_GPIO_Init te radni takt centralne procesorske jedinice i komunikacijskih sučelja što se radi u funkciji SystemClock_Config.

```
/* Private user code ---------------------------------------------
---------*/
/* USER CODE BEGIN 0 */
int Setpoint=0;
uint8_t receive[64];
/* USER CODE END 0 */
```

Varijabla Setpoint je varijabla koja određuje željenu poziciju servo motora, a receive[64] je buffer serijske komunikacije koji se koristi prilikom serijske komunikacije pretvornika sa programom za podešavanje parametara pretvarača.

```
int main(void)
 {
   /* USER CODE BEGIN 1 */
      int Input;
      int mul=0;
      int Kp=850000, Ki=3000, Kd=500;
      double kp=(double)Kp/10000000, ki=(double)Ki/10000000,
 kd=(double)Kd/10000000;
      int period = 5;
      float PID_p, PID_i, PID_d, PID_total;
      double error = 0, previous_error = 0;
      long time;
      int counter =0;
   /* USER CODE END 1 */
```

Varijabla Input predstavlja trenutnu poziciju vratila servo motora koju PID regulator koristi za određivanje odstupanja pozicije vratila servo motora u odnosu na željenu poziciju zadanu u varijabli Setpoint. Zbog limitirane veličine registra timera 1 korištenog za čitanje pozicije enkodera dolazi do prelijevanja (owerflow) u početnu vrijednost. Da bi se spriječio gubitak pozicije u navedenom slučaju, koristi se pomoćna varijabla mul u koju se sprema broj prelijeva do kojih je došlo od uključivanja mikrokontrolera. Svaki PID regulator sadrži proporcionalni, integracijski i derivacijski član, njihove vrijednosti spremaju se u varijable kp, ki i kd koje su varijable tipa double. Varijable Kp, Ki i Kd tipa integer se koriste zbog lakšeg prijenosa podataka prilikom serijske komunikacije. Vrijeme između dva proračuna parametara PID regulatora mora biti konstantno i to vrijeme je definirano variablom period, a izračunate vrijednosti pojedinih članova regulatora se spremaju u varijable PID_p, PID_i i PID_d koje se zbrajaju i njihov rezultat se sprema u varijablu PID_total. PID regulator se ne temelji samo na trenutnom odstupanju nego i na odstupanju u prošlom ciklusu regulacije. Zbog toga se koriste varijable error i previous_error gdje su spremljene vrijednosti trenutnog i prošlog odstupanja. U varijablu time se sprema vrijeme proteklo od uključivanja mikrokontrolera, a u varijablu counter broj uzoraka prikupljenih prilikom uzorkovanja vrijednosti korištenih kod podešavanja parametara pretvornika.

```
   HAL_Init();
   SystemClock_Config();
   MX_GPIO_Init();
   MX_USB_DEVICE_Init();
   MX_TIM1_Init();
   MX_TIM3_Init();
   MX_TIM5_Init();
```

Prethodnim naredbama napravljena je inicijalizacija perifernih jedinica mikrokontrolera (timeri, GPIO, DMA) i definiranje radnog takta mikrokontrolera i perifernih jedinica.

```
/* USER CODE BEGIN 2 */
HAL_TIM_Encoder_Start(&htim1, TIM_CHANNEL_ALL);
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2);
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_3);
HAL_TIM_Base_Start(&htim5);
```

Naredbama koje počinju sa HAL pokreću se timeri u prethodno navedenim načinima rada.

```
TIM1->CNT=0;
TIM3->CCR2=500;
TIM3->CCR3=500;
time=TIM5->CNT;
```

Prije početka regulacije potrebno je postaviti početne vrijednosti u timer enkodera, postaviti faktora popunjenosti PWM signala na 50% za svaki kanal i upisati početno vrijeme u varijablu time. Navedeno se izvršava prethodno navedenim naredbama.

```
TIM1->CR1|= 0b0000000000000100;
TIM1->SR &= 0b1111111111110;
```

Za omogućavanje overflow zastavice u CR1 registru timera 1 potrebno je postaviti bit 2 u logičku jedinicu logičkom funkcijom ILI, a sama overflow zastavica nalazi se u bitu 0 SR registra timera 1 koji se prije pokretanja programa briše logičkom funkcijom I.

```
memset(receive, '\0',64);
char stringNum[50];
int collected = 0;
int sent=0;
int data[1000];
int maxSpeed=0;
int number=9999;
/* USER CODE END 2 */
```

Za lakše podešavanje parametara korisi se serijska komunikacija koja u mnogim slučajevima usporava izvršavanje primarnog programa (u ovom slučaju proračun PID regulatora). Zbog toga program koristi varijable u koje se spremaju uzorci što troši zanemarivu količinu resursa centralne procesorske jedinice mikrokontrolera. Nakon što program prikupi tisuću uzoraka spremljenih u polje integera data[1000], kreće prijenos uzoraka serijskom komunikacijom. Na taj način se u programu za podešavanje parametara dobivaju isti podaci koji će se dobiti bez utjecaja serijske komunikacije na centralnu procesorsku jedinicu. Prilikom zahtjeva za promjenom parametara ili zahtjeva uzoraka od strane računala, ključna riječ i vrijednost parametra se spremaju u varijablu polja receive. U polju stringNum[50] se generira odgovor koji mikrokontroler šalje računalu. Varijable number i maxSpeed su korištene kako bi se odredio najmanji mogući period između dva proračuna PID regulatora.

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
       if(strlen(receive)!=NULL){
            if(strstr(receive, "MOV=")!=NULL){
                 Setpoint=strtol(receive+4, NULL, 10);
                 collected = 0;
                 counter = 0;
                 sent=0;
            }
            else if(strstr(receive, "PER=")!=NULL){
                 period=strtol(receive+4, NULL, 10);
            }
            else if(strstr(receive, "KP=")!=NULL){
                 Kp=strtod(receive+3, NULL);
                 kp=(double)Kp/10000000;
            }
            else if(strstr(receive, "KI=")!=NULL){
                 Ki=strtod(receive+3, NULL);
                 ki=(double)Ki/10000000;
            }
            else if(strstr(receive, "KD=")!=NULL){
                 Kd=strtod(receive+3, NULL);
                 kd=(double)Kd/10000000;
            }
            else if(strstr(receive, "CONN")!=NULL){
                 sprintf(stringNum,"KP=%d;KI=%d;KD=%d\n",Kp,Ki,Kd);
                 CDC_Transmit_FS((uint8_t *) stringNum,
strlen(stringNum));
            }
            else if(strstr(receive, "DATA")!=NULL && sent<200){
                 sprintf(stringNum,"%d;%d;%d;%d;%d;\n",

data[sent*5],data[sent*5+1],data[sent*5+2],data[sent*5+3],data[sent*5
+4]);
                 CDC_Transmit_FS((uint8_t *) stringNum,
strlen(stringNum));
                 sent++;
            }
            else if(strstr(receive, "SPED")!=NULL){
                 sprintf(stringNum,"Speed=%d\n",maxSpeed);
                 CDC_Transmit_FS((uint8_t *) stringNum,
strlen(stringNum));
            }
            memset(receive, '\0',64);
       }
```

Prikazan je dio programa koji se koristi za komunikaciju s računalom. Navedeni dio programa se izvršava samo u slučaju kad je računalo poslalo informacije kroz serijsku komunikaciju. Ako postoje informacije u bufferu serijske veze, provjerava se da li poslana informacija odgovara bilo kojoj ključnoj riječi. U slučaju podudaranja ključne riječi s primljenim informacijama kreće obrada poslanih informacija.

Ključne riječi:

“MOV=” - Ključna riječ koja pomiče motor na željenu poziciju i započinje process uzorkovanja pozicija servo motora kroz vrijeme

“PER=” – Ključna riječ koja se koristi za promjenu varijable period koja služi za promjenu vremena između dva proračuna PID regulatora.

“KP=” – Ključna riječ koja služi za promjenu P parametra PID regulatora

“KI=” – Ključna riječ koja služi za promjenu I parametra PID regulatora

“KD=” – Ključna riječ koja služi za promjenu D parametra PID regulatora

“CONN” – Ključna riječ koju računalo šalje prilikom spajanja na mikrokontroler i traži informacije o trenutnim vrijednostima P, I i D parametara.

“DATA” – Ključna riječ kojom računalo traži od mikrokontrolera podatke prikupljene prilikom uzorkovanja

“SPED” – Ključna riječ koju sam koristio kako bi dobio najmanji mogući period koji se može koristiti između dva PID proračuna

```
if(TIM1->SR%2==1){
      if(TIM1->CNT<32000){
          mul++;
        }else{
          mul--;
        }
        TIM1->SR &= 0b1111111111110;
      }
    Input=mul*65535+TIM1->CNT;
```

Ako je ostatak djeljenja SR registra timera 1 jednak jedinici znači da je došlo do prelijeva registra CNT timera što se mora zabilježiti u varijabli mul. Do prelijeva registra može doći prilikom vrtnje enkodera u smjeru kazaljke na satu i u tom slučaju se registar prelijeva iz maksimalne vrijednosti u nulu i varijablu mul je potrebno povećati za jedan, dok se u slučaju vrtnje enkodera obrnuto od kazaljke na satu registar prelijeva iz nule u maksimalnu vrijednost i varijablu mul je potrebno smanjiti za jedan.

```
if(TIM5->CNT<time){
      time=0;
}
```

U slučaju da je vrijednost varijable time manje od vrijednosti CNT registra timera 5, došlo je do prelijeva u registru CNT timera i potrebno je postaviti varijablu time na vrijednost nula.

```
            if (TIM5->CNT > time+period*2)
                {
                number=0;
                  time = TIM5->CNT;
                  error = Setpoint - Input;
                  PID_p = kp * error;
                  PID_d = kd*((error -
previous_error)/period*100000);
                  if(-100 < error && error < 100)
                  {
                    PID_i = PID_i + (ki * error);
                  }
                  else
                  {
                    PID_i = 0;
                  }
                  PID_total = PID_p + PID_i + PID_d;
                  PID_total = map(PID_total, -150, 150, 60, 940);
                  if(PID_total < 60){PID_total = 60;}
                  if(PID_total > 940) {PID_total = 940; }
               TIM3->CCR2=PID_total;
                  previous_error = error;
                  if(counter > 1 && collected<1000){
                    data[collected]=Input;
                    collected++;
                    counter = 0;
                  }else{
                    counter++;
                  }

                 }else {
                      number++;
                      if(number>maxSpeed) maxSpeed=number;
                }

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

U navedenom djelu programa radi se proračun parametara PID regulatora i izračunata vrijednost se šalje na PWM izlaz (u registar timera 3). Funkcijom map se skalira PID vrijednost koja je u granicama od -150 do 150 na vrijednost od 60 do 940. Navedeno skaliranje na vrijednost od 60 do 940 je potrebno jer se za vrijednosti manje od 60 na izlazu operacijskog pojačala dobiva napon manji od -10V, a za vrijednost veću od 940 dobiva se napon veći od 10V. Nakon proračuna vrijednosti PID regulatora nalazi se programski kod korišten za uzorkovanje pozicije servo motora koji se kasnije koristi za grafički prikaz na računalu.

```
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 25;
  RCC_OscInitStruct.PLL.PLLN = 192;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK

|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_3) !=
HAL_OK)
  {
    Error_Handler();
  }
}
```

Funkcija SystemClock_Config je funkcija koja je automatski generirana u grafičkom sučelju STM32CUBEMX i zadužena je za postavljanje radnog takta centralne procesorske jedinice mikrokontrolera, te radnih taktova perifernih jedinica i komunikacijskih sučelja i sabirnica mikrokontrolera.

```
static void MX_TIM1_Init(void)
{

  /* USER CODE BEGIN TIM1_Init 0 */

  /* USER CODE END TIM1_Init 0 */

  TIM_Encoder_InitTypeDef sConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM1_Init 1 */

  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 0;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 65535;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  sConfig.EncoderMode = TIM_ENCODERMODE_TI12;
  sConfig.IC1Polarity = TIM_ICPOLARITY_RISING;
  sConfig.IC1Selection = TIM_ICSELECTION_DIRECTTI;
  sConfig.IC1Prescaler = TIM_ICPSC_DIV1;
  sConfig.IC1Filter = 10;
  sConfig.IC2Polarity = TIM_ICPOLARITY_RISING;
  sConfig.IC2Selection = TIM_ICSELECTION_DIRECTTI;
  sConfig.IC2Prescaler = TIM_ICPSC_DIV1;
  sConfig.IC2Filter = 10;
  if (HAL_TIM_Encoder_Init(&htim1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig)
!= HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM1_Init 2 */

  /* USER CODE END TIM1_Init 2 */

}
```

Funkcija MX_TIM1_Init je funkcija koja je automatski generirana u grafičkom sučelju STM32CUBEMX i njena uloga je podešavanje načina rada timera 1 u enkoderski način rada i definiranje ulaza mikrokontrolera koje koristi timer 1.

```c
static void MX_TIM3_Init(void)
{

  /* USER CODE BEGIN TIM3_Init 0 */

  /* USER CODE END TIM3_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM3_Init 1 */

  /* USER CODE END TIM3_Init 1 */
  htim3.Instance = TIM3;
  htim3.Init.Prescaler = 3;
  htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim3.Init.Period = 1000;
  htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim3, &sClockSourceConfig) !=
HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim3) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig)
!= HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_2) !=
HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_3) !=
HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM3_Init 2 */

  /* USER CODE END TIM3_Init 2 */
```

```
  HAL_TIM_MspPostInit(&htim3);

}
```

Funkcija MX_TIM3_Init je automatski generirana funkcija koja je automatski generirana u grafičkom sučelju STM32CUBEMX i služi za konfiguraciju načina rada timera 1 u PWM način rada na kanalima 2 i 3 timera 3 koji odgovaraju fizičkim pinovima PA7 i PB8 na mikrokontroleru.

```
static void MX_TIM5_Init(void)
{

  /* USER CODE BEGIN TIM5_Init 0 */

  /* USER CODE END TIM5_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM5_Init 1 */

  /* USER CODE END TIM5_Init 1 */
  htim5.Instance = TIM5;
  htim5.Init.Prescaler = 0;
  htim5.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim5.Init.Period = 4294967295;
  htim5.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim5.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim5) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim5, &sClockSourceConfig) !=
HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim5, &sMasterConfig)
!= HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM5_Init 2 */

  /* USER CODE END TIM5_Init 2 */

}
```

Funkcija MX_TIM5_Init je funkcija čija je uloga postaljanje timera 5 u klasični način rada koji broji vrijeme od uključivanja mikrokontrolera. Navedena funkcija je automatski generirana u grafičkom sučelju STM32CUBEMX.

```
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin : STEP_Pin */
  GPIO_InitStruct.Pin = STEP_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(STEP_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : DIR_Pin */
  GPIO_InitStruct.Pin = DIR_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(DIR_GPIO_Port, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI3_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI3_IRQn);

}
```

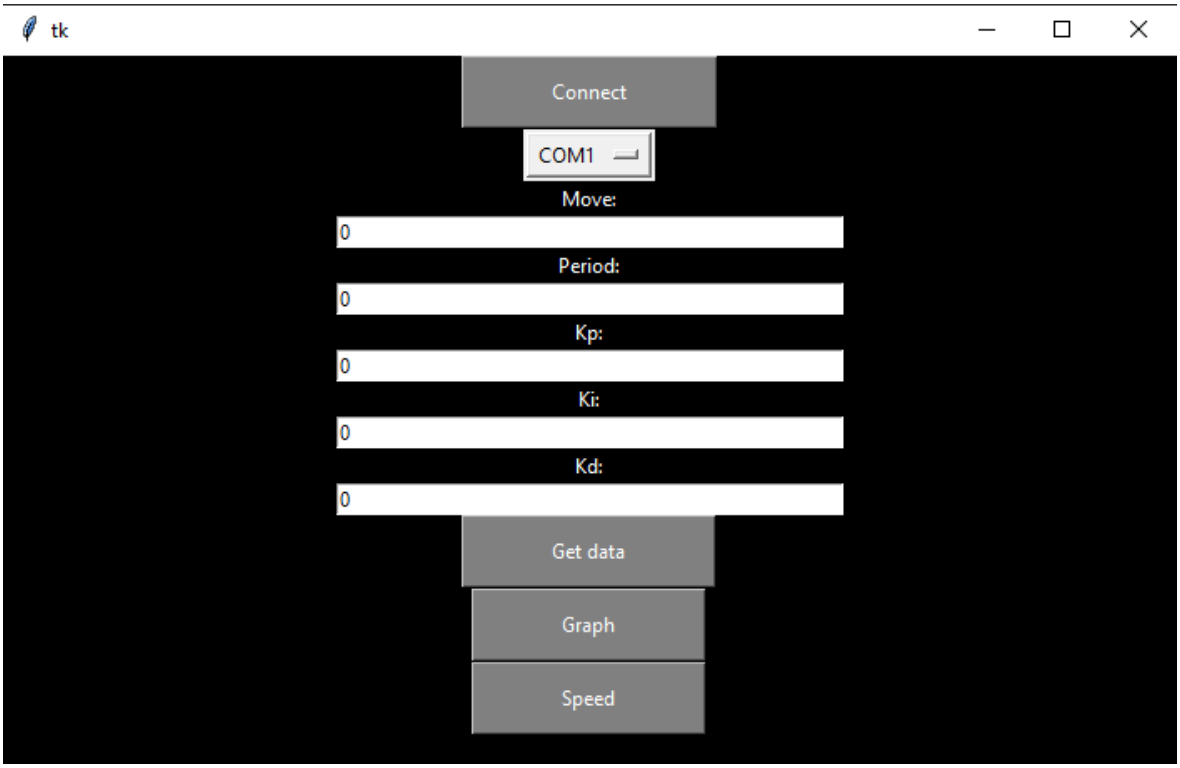
Funkcija MX_GPIO_Init je automatski generirani kod koji definira način rada ulazno izlaznih pinova mikrokontrolera. Za svaki korišteni pin određuje da li se radi o ulazu ili izlazu i u slučaju da se radi o ulazu može mu pridjeliti unutarnji pull up ili pull down otpornik čime se smanjuje potreba za dodatnim vanjskim komponentama. U navedenoj funkciji se također definira kojim ulazima se dodjeljuju prekidi te njihov prioritet.

```
/* USER CODE BEGIN 4 */
     void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin){
          if(GPIO_Pin == STEP_Pin){
               if(HAL_GPIO_ReadPin(DIR_GPIO_Port,DIR_Pin)){
                    Setpoint++;
               }else{
                    Setpoint--;
               }
          }
     }
/* USER CODE END 4 */
```

HAL_GPIO_EXTI_Callback je dio programa u kojem se obrađuje prekid na STEP ulazu mikrokontrolera. Prilikom prekida na STEP ulazu provjerava se stanje DIR ulaza. U slučaju da je stanje DIR ulaza TRUE povećava se ulazna vrijednost PID regulatora, a ako je FALSE onda se smanjuje ulaz PID regulatora.

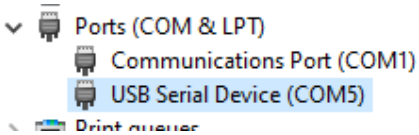
3.3. Podešavanje parametara

Za podešavanje parametara PID regulatora koristi se python program. Za izradu python programa korišten je PyCharm IDE u kojem se koristio modul serial, namijenjen za serijsku komunikaciju s mikrokontrolerom, tkinter za grafičko sučelje i matplotlib za crtanje grafa.



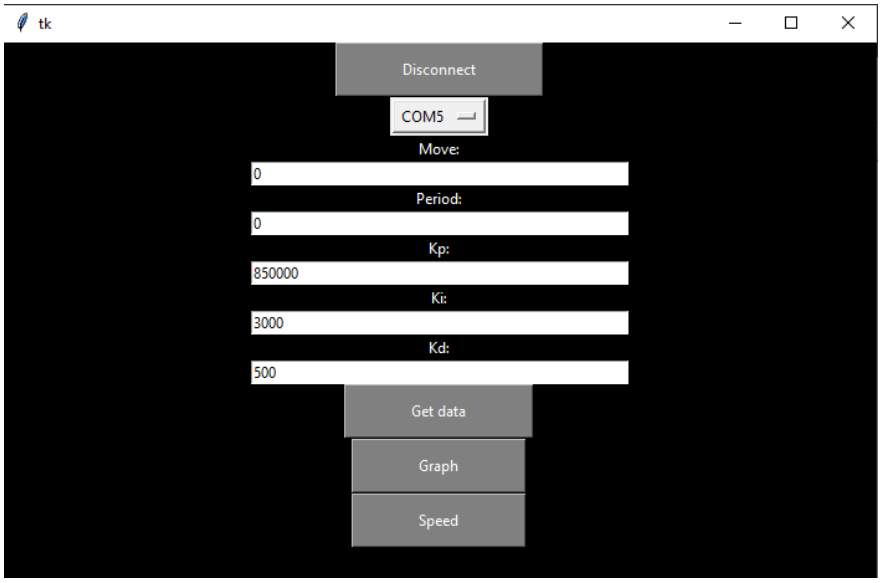
Slika 24. Program za podešavanje parametara

Nakon pokretanja programa odabire se COM port na kojem se nalazi mikrokontroler. Naziv porta na kojem je spojen mikrokontroler može se naći u Device manageru. Nakon odabira porta, pritiskom na tipku Connect program se spaja na mikrokontroler te traži informacije o P, I i D parametrima PID regulatora.

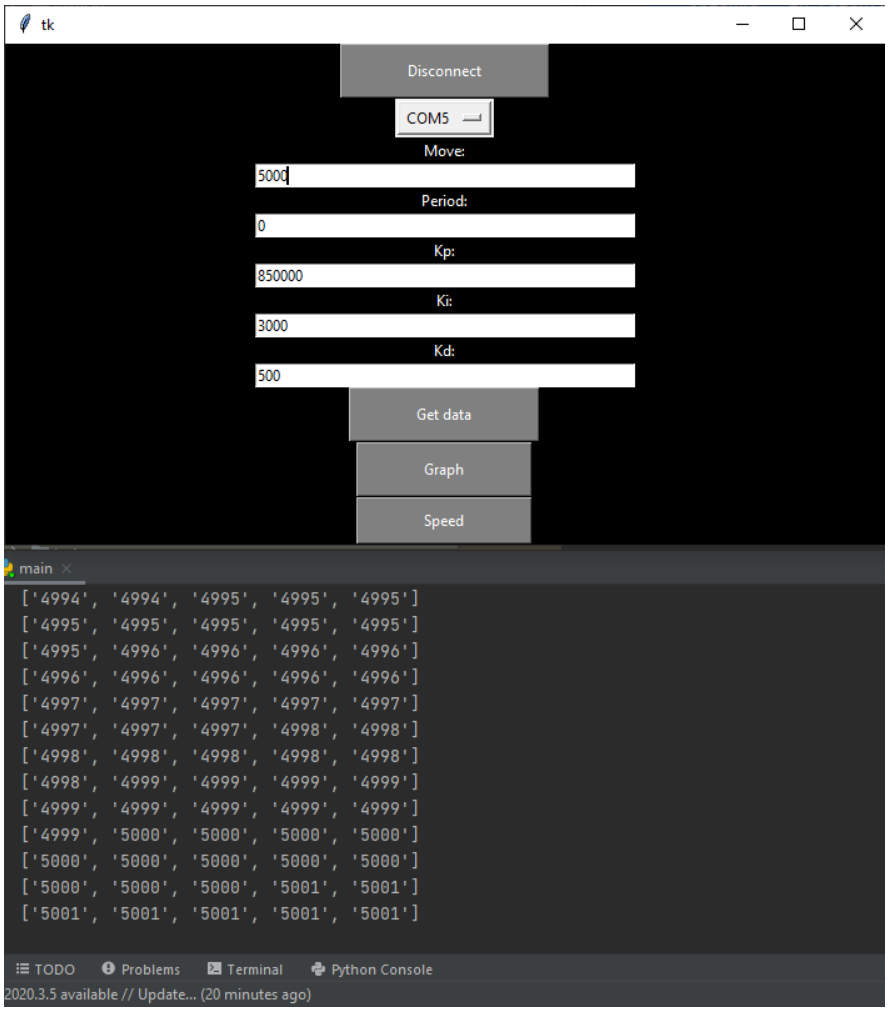


Slika 25. Device manager

Kako bi se moglo promijenili parametre u mikrokontroleru, u polje s vrijednošću (Slika 24.) potrebno je upisati novu željenu vrijednost, a nakon pritiska tipke enter nova vrijednost se šalje u mikrokontroler. Nakon odabira P, I i D parametara promjenom vrijednosti Move pomiče se servo motor na željeni položaj i započinje uzorkovanje.

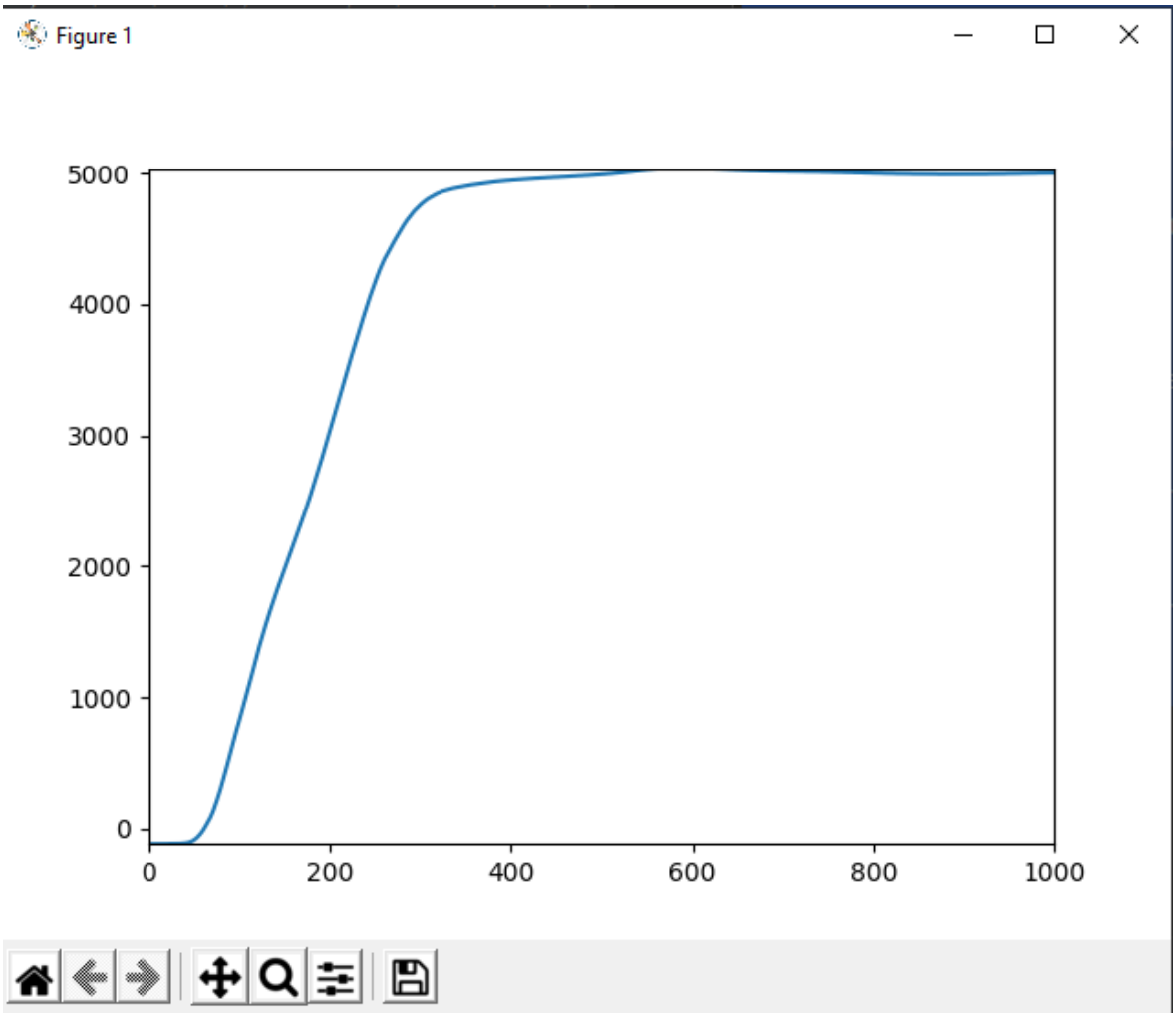


Slika 26. Mikrokontroler je povezan



Slika 27. Prijenos podataka

Nakon uzorkovanja pritiskom na tipku Get data program traži od mikrokontrolera uzorkovane podatke koje sprema u polje. Nakon što prijenos podataka završi pritiskom na tipku Graph crta se graf iz prenesenih podataka. Pomoću grafičkog prikaza promjene položaja vratila elektromotora kroz vrijeme lakše je podesiti parametre PID regulatora. Nakon promjene P, I i D parametara ponovno se nacrta graf i usporedbom grafova moguće je daljnje poboljšanje preciznosti servo motora.



Slika 28. Graf okretanja vratila elektromotora

U nastavku će biti naveden i ukratko objašnjen programski kod kojim je realizirano grafičko sučelje za zadavanje i prikaz parametara PID regulatora te crtanje promjene položaja vratila elektromotora.

```
import queue
import time
import serial
import tkinter as tk
import matplotlib.pyplot as plt
import threading
```

Naredbom import se u programskom jeziku Python dodaju moduli korišteni u programu. Modul queue je modul koji zluži za generiranje polja korištenog za spremanje uzoraka primljenih od mikrokontrolera. Iz modula time se koristi funkcija sleep koja se koristi za pauziranje programa na željeni vremenski period. Serial je modul korišten za serijsku komunikaciju između računala i mikrokontrolera. Za realizaciju grafičkog sučelja programa koristi se modul tkinter, a za crtanje

grafa se koristi pyplotlib dio modula matplotlib. Da bi program mogao normalno raditi prilikom prikupljanja uzoraka od mikrokontrolera potrebno je korištenje više dretvi što omogućuje modul threading.

```
s=serial.Serial()
data = []
q = queue.Queue()
```

Varijablom s je definirano serijsko sučelje računala, a u polje data se spremaju uzorci koji se koriste za crtanje grafa. Queue je kolekcija koja radi na FIFO (First In First Out) principu koji omogućuje spremanje podataka primljenih preko serijske komunikacije u istom redosljedu u kojem su primljeni.

```
def serial_ports():
    ports = ['COM%s' % (i + 1) for i in range(256)]
    result = []
    for port in ports:
        try:
            s = serial.Serial(port)
            s.close()
            result.append(port)
        except (OSError, serial.SerialException):
            pass
    return result
```

Navedeni kod se koristi za prikupljanje podataka o korištenim portovima računala koji se ispisuju u grafičkom sučelju gdje je potrebno odabrati port koji koristi pretvarač.

```
def connect():
    s.port=clicked.get()
    if(s.isOpen()):
        s.close()
    else:
        s.open()
    if(s.isOpen()):
        connected.set('Disconnect')
        s.write(str.encode("CONN\n"))
        read = s.readline().decode()
        print(read)
        parameters = read.split(';', 3)
        for parameter in parameters:
            if parameter.find("KP=") > -1:
                valueKp.set(parameter.split("=")[1])
            elif parameter.find("KD=") > -1:
                valueKd.set(parameter.split("=")[1])
            elif parameter.find("KI=") > -1:
                valueKi.set(parameter.split("=")[1])
    else:
        connected.set('Connect')
```

Funkcija connect služi za spajanje računala na mikrokontroler nakon odabira porta na koji je mikrokontroler spojen i pritiska tipke Connect.

```
def move():
    if(s.isOpen()):
        s.write(str.encode("MOV="+position.get()+"\n"))
```

Funkcija koja se pokreće nakon pritiska tipke enter kad je u grafičkom sučelju odabrano polje Move. Navedena funkcija šalje ključnu riječ MOV= i vrijednost upisanu u polje Move.

```
def per():
    if(s.isOpen()):
        s.write(str.encode("PER="+period.get()+"\n"))
```

Funkcija per se pokreće pritiskom tipke enter kad je u grafičkom sučelju odabrano polje Period. Navedena funkcija šalje ključnu riječ PER= i vrijednost upisanu u polje Period.

```
def updateKp():
    if(s.isOpen()):
        s.write(str.encode("KP="+valueKp.get()+"\n"))
```

Definiranje funkcije koja se pokreće pritiskom entera kod označenog polja Kp. Funkcija šalje ključnu riječ KP= i upisanu vrijednost za postavljanje proporcionalnog člana PID regulatora.

```
def updateKi():
    if(s.isOpen()):
        s.write(str.encode("KI="+valueKi.get()+"\n"))
```

Funkcija koja šalje ključnu riječ KI= i vrijednost upisanu u polje Ki kad se pritisne tipka enter.

```
def updateKd():
    if(s.isOpen()):
        s.write(str.encode("KD="+valueKd.get()+"\n"))
```

Funkcija updateKd šalje ključnu riječ KD= i vrijednost upisanu u polje Kd pritiskom na tipku enter kako bi se promjenio diferencijalni član PID regulatora u pretvorniku na željenu vrijednost.

```
def getData(q):
    q.queue.clear()
    if(s.isOpen()):
        while q.qsize()<200:
            s.write(str.encode("DATA\n"))
            read = s.readline().decode()
            readList = read.split(";")
            readList.pop()
            print(readList)
            q.put(readList)
            time.sleep(0.001);
```

Navedena funkcija pokreće prijenos uzorkovanih podataka s mikrokontrolera u Python program i sprema uzorke u kolekciju q.

```
def getSpeed():
    if(s.isOpen()):
        s.write(str.encode("SPED\n"))
        read = s.readline().decode()
        print(read)
```

Definiranje funkcije tipke Speed čijim pritiskom se u python konzoli ispisuje vrijednost varijable maxSpeed mikrokontrolera. Ako je vrijednost varijable veća od 1 potrebno je povećati vrijednost varijable period jer mikrokontroler nema dovoljno procesorske snage da u trenutno odabranom vremenu izvrši proračun parametara PID regulatora.

```
def threadGetData():
    th = threading.Thread(target=getData, args=(q,)).start()
```

Funkcija koja se pokreće pritiskom na tipku Get Data i poziva funkciju getData u novoj dretvi. Navedena funkcija se pokreće u novoj dretvi zato što njezino izvršavanje dugo traje pa bi izvršavanje funkcije getData u trenutnoj dretvi rezultiralo privremenim zamrzavanjem grafičkog sučelja programa.

```
def graph():
    data.clear()
    while(q.qsize()>0):
        data.extend(q.get())
    print(data)
    y = [float(i) for i in data]
    plt.plot(y)
    plt.axis([0,1000,min(y), max(y)])
    plt.show()
```

Pritiskom tipke Graph pokreće se funkcija graph koja otvara novi prozor i crta grafički prikaz vremenske promjene položaja vratila elektromotora.

```
if __name__ == '__main__':
    root = tk.Tk()
    canvas = tk.Canvas(root, height=700, width=700, bg='black')
    frame = tk.Frame(root, bg='black').place(relwidth=1,relheight=1)
```

Navedene funkcije definiraju prozor grafičkog sučelja, njegove osnovne parametre kao što su boja pozadine i dimenzije.

```
'''CONNECT/DISCONNECT'''
connected = tk.StringVar()
connected.set('Connect')
buttonConnect = tk.Button(frame, textvariable=connected,
padx=50,pady=10,fg='white',bg='gray', command=connect).pack();

'''DROPDOWN PORT MENU'''
clicked = tk.StringVar()
clicked.set(serial_ports()[0])
dropdownPort = tk.OptionMenu(frame, clicked, *serial_ports()).pack()

'''POSITION TEXTBOX'''
position = tk.StringVar()
position.set('0')
tk.Label(frame, text="Move:", bg="black", fg="white").pack()
entryPosition = tk.Entry(frame, textvariable=position, width=50)
entryPosition.bind("<Return>", (lambda event: move()))
entryPosition.pack()

'''PERIOD TEXTBOX'''
period = tk.StringVar()
period.set('0')
tk.Label(frame, text="Period:", bg="black", fg="white").pack()
entryPeriod = tk.Entry(frame, textvariable=period, width=50)
entryPeriod.bind("<Return>", (lambda event: per()))
entryPeriod.pack()

'''KP TEXTBOX'''
valueKp = tk.StringVar()
valueKp.set('0')
tk.Label(frame, text="Kp:", bg="black", fg="white").pack()
entryKp = tk.Entry(frame, textvariable=valueKp, width=50)
entryKp.bind("<Return>", (lambda event: updateKp()))
entryKp.pack()

'''KI TEXTBOX'''
valueKi = tk.StringVar()
valueKi.set('0')
tk.Label(frame, text="Ki:", bg="black", fg="white").pack()
entryKi = tk.Entry(frame, textvariable=valueKi, width=50)
entryKi.bind("<Return>", (lambda event: updateKi()))
entryKi.pack()

'''KD TEXTBOX'''
```

```
valueKd = tk.StringVar()
valueKd.set('0')
tk.Label(frame, text="Kd:", bg="black", fg="white").pack()
entryKd = tk.Entry(frame, textvariable=valueKd, width=50)
entryKd.bind("<Return>", (lambda event: updateKd()))
entryKd.pack()

'''GET DATA'''
buttonData = tk.Button(frame, text="Get data",
padx=50,pady=10,fg='white',bg='gray', command=threadGetData).pack()

'''GRAF'''
buttonGraph = tk.Button(frame, text="Graph",
padx=50,pady=10,fg='white',bg='gray', command=graph).pack()

'''SPEED'''
buttonSpeed = tk.Button(frame, text="Speed",
padx=50,pady=10,fg='white',bg='gray', command=getSpeed).pack()
```

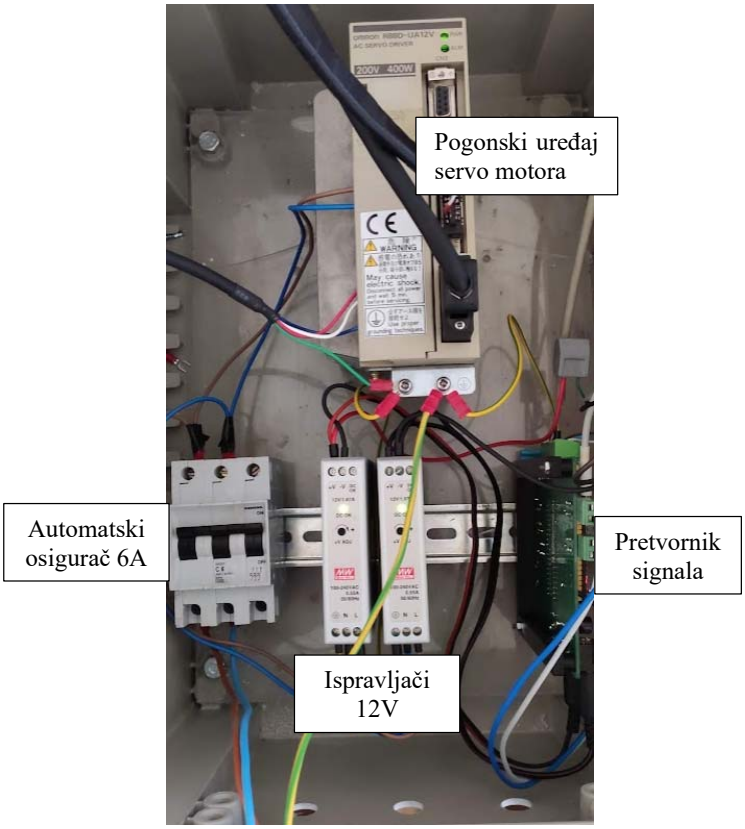
Navedeni kod generira elemente grafičkog sučelja koji su navedeni u komentaru iznad koda. Osim generiranja samog elementa, svakom element dodjeljuje funkciju koja se pokreće pritiskom na tipku enter u slučaju tekstualnog polja ili pritiskom na gumb.

```
canvas.pack()
root.mainloop()
```

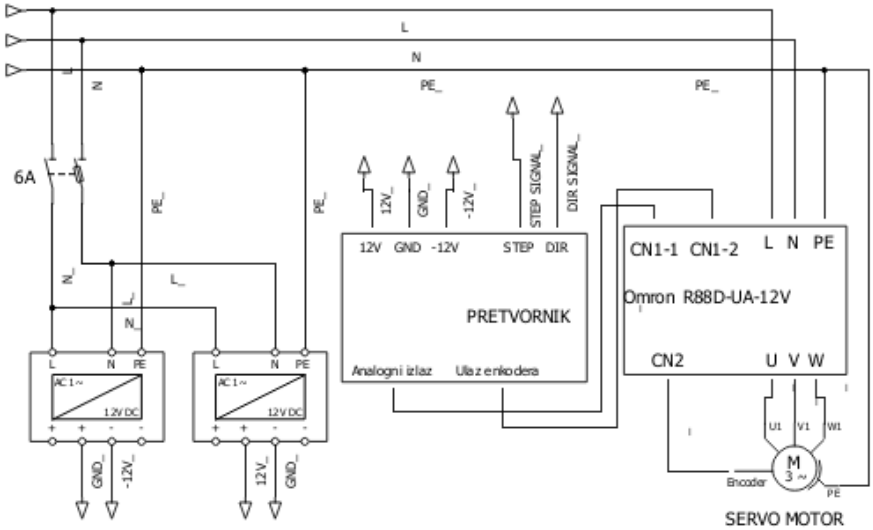
Za pokretanje prethodno definiranog grafičkog sučelja koriste se naredbe canvas.pack() i root.mainloop().

4. Ispitivanje rada pretvornika signala

Nakon spajanja pretvarača signala sa servo motorom i napajanjima te nakon podešavanja parametara PID regulatora provedeno je mjerenje kako bi se odredilo vrijeme reakcije sklopa i preciznost pozicioniranja servo motora. Na slici 29 nalazi se pretvornik signala montiran u elektroormaru, a na slici 30 shema spajanja elektroormara.

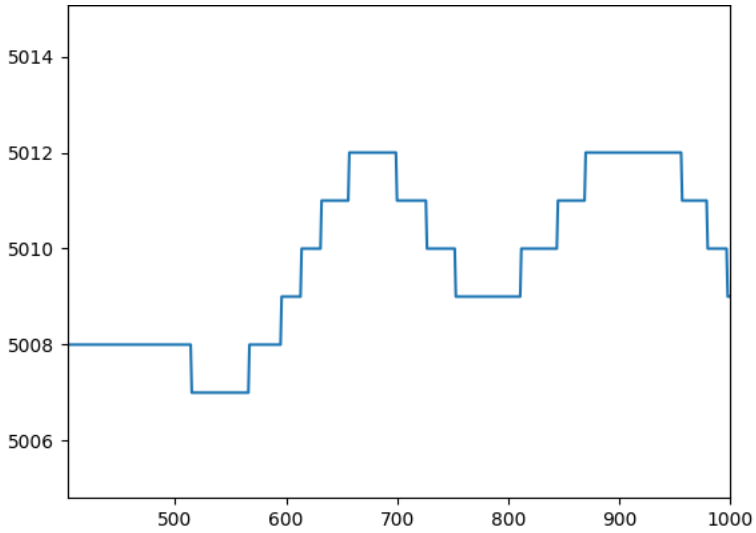


Slika 29. Pretvornik montiran u elektroormaru



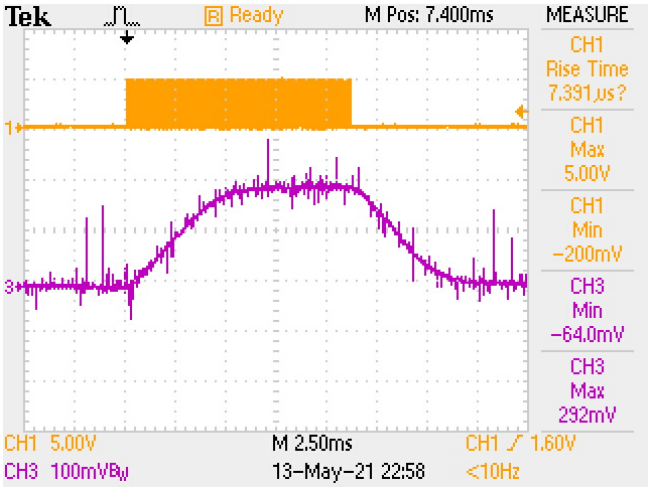
Slika 30. Shema spajanja elektroormara

Ispitivanje rada sklopa za pretvorbu signala provedeno je korištenjem programa za podešavanje parametara opisanog u poglavlju 3.3. Sklopu za pretvorbu poslana je naredba da se vratilo elektromotora okrene u poziciju koja odgovara broju 5010. Iz podataka snimljenih tijekom pomaka vratila nacrtana je slika 31. Iz grafa je vidljivo da kut rotacije vratila servo motora odstupa ±3 pulsa. Ako se uzme u obzir da je enkoder servo motora 2048 p/rev i da se svaki puls još dijeli na četiri djela dobiva se preciznost od (2,44 ± 7,32) x 10^{-3}mm u slučaju da je servo motor spojen direktno na kugličnu navojnu šipku hoda 20mm/rev.



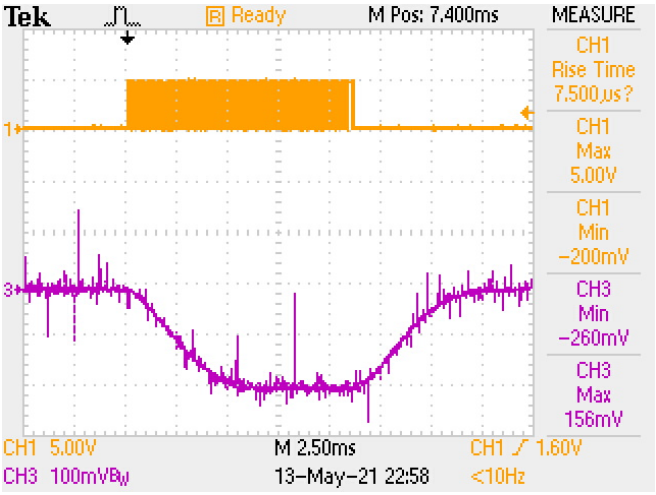
Slika 31. Odstupanje od zadane vrijednosti

Mjerenjem STEP ulaza u pretvarač i izlaza operacijskog pojačala osciloskopom (Slika 32.) te analizom podataka dobivenih ispisom mjerenja osicloskopom (Prilog 1), vidljivo je da u vremenu -0.0000000060000 μs osciloskop učitava step signal (CH1), a u vremenu 0.00055000060000 μs dolazi do značajnije promjene izlaznog signala (CH3). Iz navedenog možemo zaključiti da je vrijeme reakcije sklopa oko 550 ns.



Slika 32. Mjerenje osciloskopom 1

Isti rezultat se dobije mjerenjem u slučaju promjene smjera vrtnje pomoću signala na ulazu DIR.



Slika 33. Mjerenje osciloskopom 2

5. Zaključak

Zbog velike preciznosti i otpornosti na djelovanje poremećaja, u industiji se koriste sustavi s povratnom vezom. Upravljanje takvih sustava uglavnom zahtjeva skupu popratnu elektroniku. Cilj ovog završnog rada bio je realizacija pretvornika signala koji će omogućiti upravljanje analognim servo motorima pomoću digitalnog računala. Nakon izrade završnog rada i provedbe mjerenja osciloskopom te snimanja signala o pomaku vratila elektromotora pomoću povratnih informacija iz mikrokontrolera može se zaključiti da je moguća kontrola analognih servo motora uz pomoć jednostavne elektronike i digitalnog signala iz računala. Budući da su u sklopu pretvornika signala korištene periferne jedinice mikrokontrolera, npr. timer s vanjskim ulazom za enkoder, timer s vanjskim izlazom za PWM signal, dodatno je smanjeno opterećenje centralne procesorske jedinice mikrokontrolera. Spomenutim korištenjem perifernih jedinica te čitanjem kritičnih signala na ulazima uz pomoć prekida i PID regulacijom izlaza, moguće je ostvariti vrlo preciznu kontrolu pozicije analognog servo motora. Osim navedenog, prilikom ispitivanja sklopa može se primijetiti da je analogna elektronika vrlo osjetljiva na smetnje i da je potrebno obratiti pažnju na zaštitu signalnih vodiča od vanjskih utjecaja korištenjem oklopljenih kabela.

6. Literatura

[1] https://5.imimg.com/data5/LJ/UJ/ZV/SELLER-8855979/yaskawa-servo-pack-sigma-5-sigma-7-sgmgv-sgdv-500x500.jpg, dostupno 28.7.2021.

[2] https://i.ytimg.com/vi/yy3_A4ZV9f8/maxresdefault.jpg, dostupno 28.7.2021.

[3] https://www.motioncontroltips.com/faq-why-are-so-many-designers-replacing-resolvers-with-encoders/, dostupno 28.7.2021.

[4] https://www.roboteq.com/images/article-images/frontpage/resolver_blog/resolver_signals.png, dostupno 28.7.2021.

[5] https://res.cloudinary.com/rsc/image/upload/b_rgb:FFFFFF,c_pad,dpr_1.0,f_auto,q_auto,w_700/c_pad,w_700/F7450089-01.jpg, dostupno 28.7.2021.

[6] https://docs.zephyrproject.org/2.5.0/boards/arm/blackpill_f411ce/doc/index.html, dostupno 28.7.2021.

[7] https://www.apogeeweb.net/circuitry/lm358n-op-amp.html, dostupno 28.7.2021.

[8] https://microcontrollerslab.com/6n137-pinout-examples-applications-datasheet-features/, dostupno 28.7.2021.

[9] https://www.amazon.in/Versatile-7805-5v-voltage-regulator/dp/B00KHK1C6Y, dostupno 28.7.2021.

[10] https://repozitorij.etfos.hr/islandora/object/etfos%3A2261/datastream/PDF/view, dostupno 28.7.2021.

[11] https://www.edata.omron.com.au/eData/Servos/I501-E1-7.pdf, dostupno 28.7.2021.

[12] https://www.st.com/en/microcontrollers-microprocessors/stm32f411.html, dostupno 28.7.2021.

[13] https://www.ti.com/lit/ds/symlink/lm158-n.pdf, dostupno 28.7.2021.

[14] https://www.vishay.com/docs/84732/6n137.pdf, dostupno 28.7.2021.

[15] https://www.sparkfun.com/datasheets/Components/LM7805.pdf, dostupno 28.7.2021.

Popis slika

Prilozi

t	CH1	CH2
-00.00000000.60000	460.000	0.00000
00.00001000060000	460.000	0.00400
00.00002000080000	480.000	-0.00400
00.00003000020000	0.20000	0.00000
00.00004000000000	0.00000	0.00000
00.00005000020000	0.20000	0.00000
00.00006000000000	0.00000	-0.00400
00.00007000060000	460.000	0.00000
00.00008000040000	440.000	0.00000
00.00009000080000	480.000	0.04400
00.00010000060000	460.000	0.00000
00.00011000080000	480.000	0.00000
00.00012000060000	460.000	-0.01200
00.00013000000000	0.00000	-0.03600
00.00014000000000	0.00000	0.00400
00.00015000000000	0.00000	0.00000
00.00016000000000	0.00000	0.00000
00.00017000080000	480.000	0.00000
00.00018000060000	460.000	0.00400
00.00019000080000	480.000	0.00400
00.00020000060000	460.000	0.00400
00.00021000060000	460.000	0.00400
00.00022000080000	480.000	0.00800
00.00023000000000	0.00000	0.00400
00.00024000000000	0.00000	0.01200
00.00025000020000	-0.20000	0.00800
00.00026000000000	0.00000	-0.06400
00.00027000080000	480.000	0.01200
00.00028000080000	480.000	0.00400
00.00029000060000	460.000	0.00800
00.00030000080000	480.000	-0.02400
00.00031000060000	460.000	0.00400
00.00032000060000	460.000	0.01200
00.00033000000000	0.00000	0.00800
00.00034000000000	0.00000	0.00800
00.00035000000000	0.00000	0.00000
00.00036000000000	0.00000	0.00800
00.00037000080000	480.000	0.00800
00.00038000080000	480.000	0.00400
00.00039000080000	480.000	0.00000
00.00040000060000	460.000	0.01200
00.00041000080000	480.000	0.00800
00.00042000080000	480.000	0.01200

00.00043000020000	-0.20000	0.01200
00.00044000000000	0.00000	0.00800
00.00045000060000	460.000	0.01200
00.00046000080000	480.000	0.00800
00.00047000080000	480.000	0.03200
00.00048000000000	0.00000	0.00800
00.00049000000000	0.00000	0.01200
00.00050000000000	0.00000	0.01200
00.00051000060000	460.000	0.01200
00.00052000060000	460.000	0.00000
00.00053000060000	460.000	0.01600
00.00054000060000	460.000	0.01200
00.00055000060000	460.000	0.01200
00.00056000080000	480.000	0.02000
00.00057000080000	480.000	0.01200
00.00058000000000	0.00000	0.01600
00.00059000000000	0.00000	0.01600
00.00060000000000	0.00000	0.01600
00.00061000060000	460.000	0.01200
00.00062000080000	480.000	0.02000
00.00063000080000	480.000	0.01600
00.00064000060000	460.000	0.00800
00.00065000060000	460.000	0.02000
00.00066000080000	480.000	0.01600
00.00067000060000	460.000	0.03200
00.00068000000000	0.00000	0.02000
00.00069000000000	0.00000	0.01200
00.00070000000000	0.00000	0.02000
00.00071000060000	460.000	0.02000
00.00072000060000	460.000	0.02400
00.00073000060000	460.000	0.03200
00.00074000060000	460.000	0.02400
00.00075000060000	460.000	0.02400
00.00076000080000	480.000	0.02400
00.00077000080000	480.000	0.02400
00.00078000020000	-0.20000	0.02400
00.00079000020000	0.20000	0.02400
00.00080000020000	0.20000	0.02400
00.00081000080000	480.000	0.02400
00.00082000060000	460.000	0.03200
00.00083000060000	460.000	0.02000
00.00084000080000	480.000	0.02800
00.00085000060000	460.000	0.02800
00.00086000080000	480.000	0.03600
00.00087000080000	480.000	0.02400
00.00088000000000	0.00000	0.02800

Sveučilište
Sjever

IZJAVA O AUTORSTVU
I
SUGLASNOST ZA JAVNU OBJAVU

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, NIKOLA PETEK *(ime i prezime)* pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/diplomskog (*obrisati nepotrebno*) rada pod naslovom RAZVOJ PRETVORNIKA STEZNIK SIGNALA V *(upisati naslov)* te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

RAZVOJ PRETVORNIKA STEZNIK SIGNALA V UPRAVLJAČKI SIGNAL ZA SERVO MOTOR

Student/ica:
(upisati ime i prezime)
NIKOLA PETEK
(vlastoručni potpis)

Sukladno Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Ja, NIKOLA PETEK *(ime i prezime)* neopozivo izjavljujem da sam suglasan/na s javnom objavom završnog/diplomskog (*obrisati nepotrebno*) rada pod naslovom RAZVOJ PRETVORNIKA STEZNIK SIGNALA V UPRAVLJAČKI SIGNAL ZA SERVO MOTOR *(upisati naslov)* čiji sam autor/ica.

Student/ica:
(upisati ime i prezime)
NIKOLA PETEK
(vlastoručni potpis)