Robotski crtač za ispis fotografija u vertikalnoj ravnini

Listeš, Lovre

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Lovre Listeš

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Lovre Listeš

Zagreb, 2022.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se svim profesorima i asistentima na prenesenom znanju koje sam mogao primijeniti prilikom izrade diplomskog rada. Posebno bih se zahvalio mentoru Dr. sc. Tomislavu Stipančiću, dipl. ing. na stručnim savjetima i sugestijama za izradu što boljeg diplomskog rada. Nakraju bih se zahvalio svojoj obitelji, djevojci i prijateljima koji su mi bili velika podrška tijekom cijelog studija i izrade diplomskog rada.

Lovre Listeš

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske radove studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala te mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa: 602-04/22-6/1	
Ur. broj: 15-1703-22-	

DIPLOMSKI ZADATAK

Student: **LOVRE LISTEŠ** Mat. br.: 0035210211

Naslov rada na hrvatskom jeziku: **Robotski crtač za ispis fotografija u vertikalnoj ravnini**

Naslov rada na engleskom jeziku: **Robotic plotter for printing photos in the vertical plane**

Opis zadatka:

U radu je potrebno osmisliti koncept te izraditi 2D crtač sastavljen od elektroničkih komponenti (mikroprocesor i električni motori) te jednostavne konstrukcije.

U sklopu rada potrebno je:

1. Osmisliti koncept robotskog crtača te definirati njegove komponente.
2. Osmisliti i implementirati matematički model koji definira položaj vrha alata za crtanje u vertikalnoj ravnini.
3. Točke gibanja alata definirati fotografijom koju crtač treba nacrtati. Koristeći razvijenu računalnu podršku, fotografije je najprije potrebno grafički obraditi (pojednostavniti) te prilagoditi za crtanje.
4. Softversku podršku temeljiti na odgovarajućem programskom okruženju te pripadajućim programskim bibliotekama.

Prilikom izrade nosača motora i alata za crtanje potrebno je koristiti FDM (engl. Fused Filament Fabrication) tehnologiju za 3D ispis.
Razvijeno cjelovito rješenje potrebno je eksperimentalno evaluirati. U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan: 29. rujna 2022.

Rok predaje rada: 1. prosinca 2022.

Predviđeni datum obrane: 12. prosinca do 16. prosinca 2022.

Zadatak zadao: doc. dr. sc. Tomislav Stipančić

Predsjednica Povjerenstva: prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS TEHNIČKE DOKUMENTACIJE

10 – 01	Nosač olovke
10 – 02	Nosač motora
10 – 03	Spojka
10 – 04	Podsklop 1
10	Robotski crtač

POPIS OZNAKA

Oznaka	Mjerna jedinica	Opis oznake
dX	–	pomak s obzirom na X os u koracima
dY	–	pomak s obzirom na Y os u koracima
I_{max}	A	maksimalna struja po fazi
L_1	–	udaljenost olovke od motora A u koracima
L_2	–	udaljenost olovke od motora B u koracima
n	–	broj segmenata linije
R_s	Ω	otpor za mjerenje struje
S	mm	udaljenost između koračnih motora u koracima
V_{ref}	V	referentni napon
X_0	–	početna koordinata apscise u koracima
X_1	–	ciljna koordinata apscise u koracima
Y_0	–	početna koordinata ordinate u koracima
Y_1	–	ciljna koordinata ordinate u koracima

POPIS KRATICA

SCARA	*Selective Compliance Articulated Robot Arm* – Selektivno usklađena robotska ruka
CNC	*Computer Numerical Control* – Računalno numeričko upravljanje
CAD	*Computer Aided Design* – Računalom potpomognuto oblikovanje
itd.	i tako dalje
FDM	*Fused Deposition Modeling* – Tehnologija taložnog očvršćivanja
AM	*Additive Manufacturing* – Aditivna proizvodnja
STL	*Stereolithography* – Stereolitografija
ABS	*acrylonitrile butadlene styrene* - akrilonitril butadien stiren
PLA	*polylactic acid* – polilaktična kiselina
AMF	*Additive Manufacturing File* – Format datoteke aditivne proizvodnje
PMW	*pulse-width modulation* – modulacija širine impulsa
ICSP	*In Circuit Serial Programming* – Serijsko programiranje u krugu
USB	*Universal Serial Bus* – Univerzalna serijska sabirnica
AC	*Alternating current* – Izmjenična struja
DC	*Direct current* – Istosmjerna struja
IDE	*Integrated Development Environment* – Integrirano razvojno okruženje
RAM	*Random Access Memory* – Memorija s nasumičnim pristupom
I2C	*Inter-Integrated Circuit* – Serijski komunikacijski protokol
AREF	*Analog Reference* – Analogna referenca
IOREF	*input/output reference* – Ulaz/izlaz referenca
I/O	*Input/Output* – Ulaz/izlaz
SRAM	*Static Random Access Memory* – Statička memorija s nasumičnim pristupom

EEPROM	*Electrically Erasable Programmable Read-Only Memory* – Električno izbrisiva programibilna ispisna memorija
GND	*Ground* - Uzemljenje
VIN	*Voltage Input* – Ulazni napon
RX	*Receive Data* – Primi podatke
TX	*Transmit Data* – Prenesi podatke
TTL	*Transistor-Transistor Logic* – Tranzistor-transistorski logički sklop
SPI	*Serial Peripheral Interface* – Serijsko periferno sučelje
LED	*Light Emitting Diode* – Svjetleća dioda
TWI	*Two-Wire Interface* – Dvožično sučelje
SDA	*Serial Data* – Serijski podaci
SCL	*Serial Clock* – Serijski sat
UART	*Universal Asynchronous Receiver / Transmitter* - Univerzalni asinkroni prijemnik-odašiljač
COM	*Communication Port* - komunikacijski priključak
NEMA	*National Electrical Manufacturers Association* – Nacionalna udruga proizvođača električne opreme
VDD	*Voltage Drain Drain* – Napajanje logičkog napona
VMOT	*Voltage Motor* – Napajanje koračnih motora
STEP	*Step* – Korak
DIR	*Direction* – Smjer
SLP	*Sleep* – Spavanje
RST	*Reset* – Resetiranje
EN	*Enable* - Omogući
MOSFET	*Metal – Oxide – Semiconductor Field – Effect Transistor* – tranzistor sa efektom polja na bazi spoja metal–oksid–poluvodič
BJT	*Bipolar Junction Transistor* – Bipolarni spojni tranzistor

PFC	*Power Factor Correction* – korekcija faktora snage
VAC	*Voltage Alternating Current* – Napon izmjenične struje
DIN	*Deutsches Institut für Normung* – Njemački institut za standardizaciju
VREF	*Reference Voltage* – Referentni napon
PDE	*Processing Development Environment* – Processing razvojno okruženje
GUI	*Graphical User Interface* – Grafičko razvojno okruženje

SAŽETAK

U radu je osmišljen te izrađen robotski crtač za ispis fotografija u vertikalnoj ravnini. Osmišljen je koncept koji je sastavljen od odgovarajućih elektroničkih komponenti i jednostavne konstrukcije. Nestandardni dijelovi poput nosača motora i alata za crtanje izrađeni su FDM (engl. Fused Deposition Modeling) tehnologijom. Robotskom crtaču za ispis fotografija u vertikalnoj ravnini bilo je potrebno razviti odgovarajuću softversku podršku. Koristeći računalnu podršku, fotografije su grafički obrađene te prilagođene za crtanje. U Arduino integriranom razvojnom okruženju je implementiran softver za komunikaciju sa mikroprocesorom i upravljačima motora. Razvijen je matematički model koji definira položaj vrha alata za crtanje u vertikalnoj ravnini te su implementirane različite naredbe koje obavljaju određene zadatke tijekom crtanja. Cjelovito rješenje je tada eksperimentalno evaluirano.

Ključne riječi: 2D robotski ploter, manipulator, koračni motor, Arduino, servo motor, obrada fotografije

SUMMARY

In this work, a robotic plotter for drawing photos in the vertical plane was designed and built. A concept has been devised that is composed of suitable electronic components and a simple construction. Non-standard parts such as engine mounts and drawing tool are made using FDM (Fused Deposition Modeling) technology. It was necessary to develop the appropriate software support for the robotic plotter for drawing photos in the vertical plane. Using computer support, the photos were graphically processed and adjusted for drawing. Software for communication with the microprocessor and motor controllers was implemented in the Arduino integrated development environment. A mathematical model has been developed that defines the position of the top of the drawing tool in the vertical plane, and various commands have been implemented that perform certain tasks during drawing. The complete solution was then experimentally evaluated.

Keywords: 2D robotic plotter, manipulator, stepper motor, Arduino, servo motor, photo processing

1. UVOD

Robotika je grana tehnologije koja se koristi za poboljšanje produktivnosti, sigurnosti te uštede vremena i novca. Roboti su napravljeni da daju rješenja za različite oblike zadataka i ispune nekoliko različitih svrha te stoga zahtijevaju niz specijaliziranih komponenti za obavljanje takvih zadataka[1]. Roboti čije je radno područje ravnina, većinom rade na principu g – koda . G – kod je programski jezik za CNC (Computer Numerical Control) strojeve. G – kod je kratica za „Geometric Code“. G – kod je jezik koji govori stroju što da učini ili kako da nešto učini. Naredbe G – koda daju stroju upute kamo da se kreće, kojom brzinom da se kreće i koju putanju da slijedi[2]. Takvi strojevi imaju široku paletu uporabe kao što je rezanje, tokarenje, glodanje, elektroerozijska obrada žicom, obrada vodenim mlazom i laserom, izrada elektroničkih dijelova pa čak i za dvodimenzionalne uporabe poput laserskog graviranja i crtanja. Za dobivanje 2D ispisa mogu se koristiti razne izvedbe robota. Primjerice, SCARA, cilindrični, kartezijski, sferni pa naposljetku i revolutni, mogu se iskoristiti u svrhe dvodimenzionalne uporabe. No, s obzirom da takvi roboti imaju više od dva stupnja slobode gibanja, a i time su mnogo skuplji, potrebno je konstruirati i izraditi robota s dvije osi. Robotski crtač sa dva stupnja slobode gibanja je način na koji se može realizirati ispis fotografija. Dvodimenzionalni robotski crtač implementiran je na principu računalnog numeričkog upravljanja (CNC). Iako robotski crtač ne koristi uobičajeni g – kod, radi na sličnom principu. Dvodimenzionalni robotski crtač je CNC kontroler koji je izrađen pomoću mikroupravljačke ploče, nekoliko motora koji služe za ostvarivanje gibanja i pomoćnih radnji te različitih standardiziranih i modeliranih dijelova za konstrukciju.

2. ROBOTSKI CRTAČ ZA ISPIS FOTOGRAFIJA

2.1. Osnove rada robotskog crtača

Robotski crtač za ispis fotografija je vrsta CNC uređaja koji crta slike pomicanjem olovke po okomitoj površini. On koristi platformu Arduino (mikroupravljačku ploču) za upravljanje dva koračna motora koja se koriste za promjenu duljine krakova na kojim se nalazi olovka za ispis. Kako bi se pogonili koračni motori nije dovoljan samo mikroupravljač već i dva upravljača motora koji omogućuju njihovo korištenje. Također, ugrađen je i jedan servo motor koji se koristi za podizanje i spuštanje olovke. Cijeli mehanizam je montiran na ploču na koju se postavlja papir. Površina za crtanje slike će ovisiti o dimenzijama robota.

2.2. Razlike između robota i CNC stroja

Ljudi često kupuju robota misleći da će biti poput CNC stroja. To nije točno, ali granice između njih danas se brišu. Postoje velike razlike između ove dvije tehnologije. Međutim, te su razlike u posljednjem desetljeću sve manje. Roboti sada mogu obavljati neke zadatke strojne obrade s izrazito usporedivim rezultatima[3].

Osnovna značajka strojeva s računalnim numeričkim upravljanjem (CNC) je njihova točnost. Mogu postići visoke točnosti za vrlo specifične operacije strojne obrade. Obično se mogu pomicati samo u osi X, Y i Z, iako je ponekad moguća kontrola orijentacije alata. Osi se precizno upravljaju pomoću računala, što omogućuje vrlo precizno skidanje materijala u usporedbi s ručnom obradom[3].

Općenito, pojedinačni CNC stroj je prikladan samo za jedan zadatak. Međutim, postoji niz različitih strojeva, od kojih je svaki konstruiran za određenu operaciju strojne obrade poput glodanja, tokarenja, bušenja, rezanja, itd.[3]

Slika 1. 5 – osna glodalica[4]

Osnovna značajka robota je njihova prilagodljivost. Oni mogu obaviti veliki broj različitih zadataka (uključujući i strojnu obradu). Štoviše, jedan robot se može koristiti za mnoge operacije. Robot se općenito sastoji od krutih mehaničkih veza koje pokreću motori kojima se može precizno upravljati robot[3]. Najčešći su roboti sa 6 stupnjeva slobode gibanja, ali postoje i drugi roboti (npr. kartezijski ili delta) koji koriste različite mehaničke konfiguracije.

Bilo bi nemoguće dati potpuni popis svih mogućih zadataka za koje se robot može koristiti, ali jedni od najčešćih su: Izuzimanje i postavljanje, slaganje, zavarivanje, strojna obrada itd[3].

Slika 2. Robotska ruka FANUC LR Mate 200iD[5]

Što se tiče zadataka koje dvije tehnologije mogu obaviti, možemo generalizirati razliku između robota i CNC strojeva ovako:

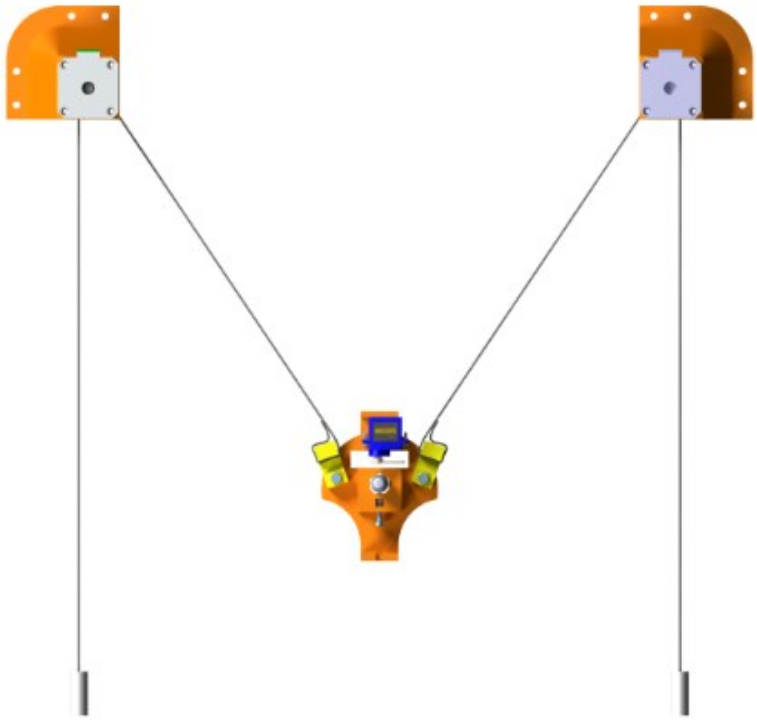
Jedan CNC stroj daje visoke performanse za određeni zadatak obrade.

Jedan robot može izvršiti mnogo zadataka s različitim performansama za svaki.

Robotski crtač za ispis fotografija u vertikalnoj ravnini povezuje oba svijeta.

3. MODELIRANJE ROBOTSKOG CRTAČA

Kako bi se krenulo u izradu robotskog crtača, najprije se mora osmisliti koncept i konstrukcija crtača. Robotski crtač za ispis fotografija je osmišljen na način da se crtanje obavlja na ploči koja stoji vertikalno u odnosu na pod.

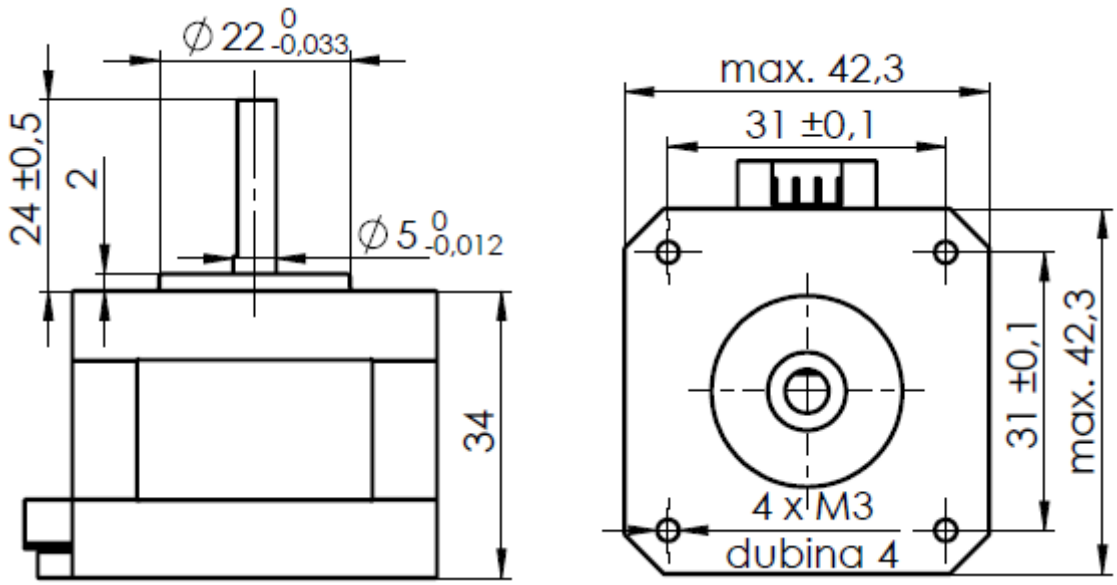


Slika 3. Koncept robotskog crtača za ispis fotografija vertikalnoj ravnini

Uz standardne dijelove koji su potrebni za montažu, potrebno je modelirati i konstruirati nosače za motore i olovku. Olovka će biti u konstantnom dodiru sa pločom pod utjecajem same težine utega koji se nalaze na nosaču. Prijenos snage i momenta omogućit će zupčaste remenice koje će povezivati motore i nosač olovke. Kao što je prikazano na slici potrebno je dodati i dva utega za zatezanje kako bi zubi mogli doći u zahvat između remenika i remenice te tako omogućili prijenos snage. Dijelovi su modelirani u CAD programu CATIA V5R21.

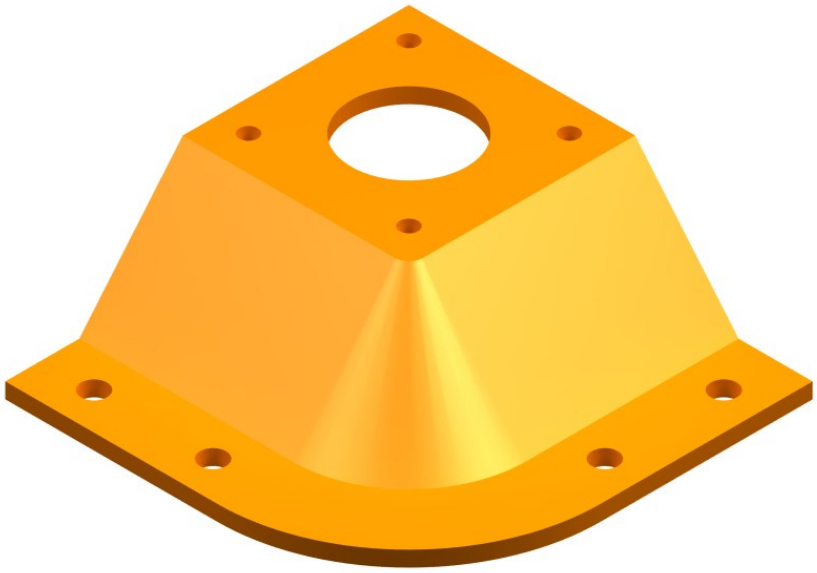
3.1. Modeliranje nosača motora

Prva funkcija nosača motora je pričvršćivanje motora za ploču. Uz pomoć vijaka, podložnih pločica te matica može se sigurno i jeftino omogućiti spajanje. Kako bi se napravio takav rastavljivi spoj potrebno je unutar modela napraviti četiri cilindrične rupe. Druga funkcija nosača je sam prihvat motora. Odabrani su koračni motori 17HS3401. Udaljenost između centrirajućih navojnih uvrta je 31 mm sa tolerancijom od ± 0,1 mm, osovina za pozicioniranje je promjera je ϕ22 mm sa tolerancijom – 0,033 mm.



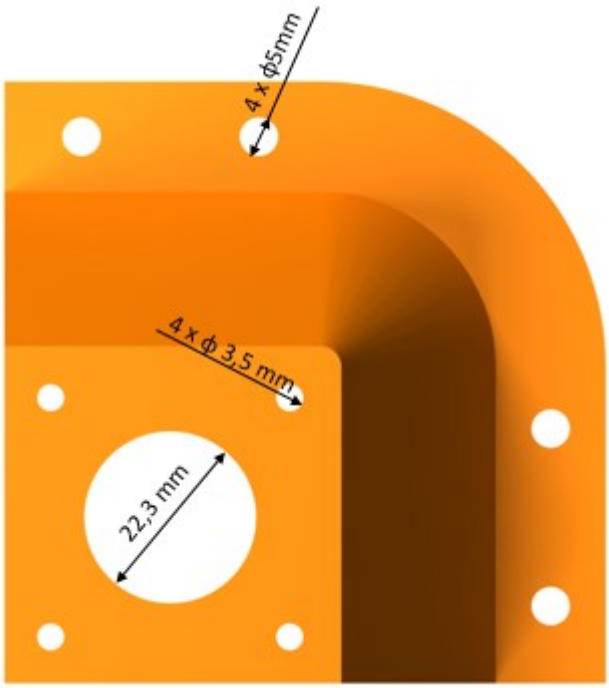
Slika 4. Specifikacije NEMA 17HS3041 koračnog motora

Uz prikazane podatke moguće je konstruirati nosač. Na slijedećoj slici je prikazana izometrija nosača motora.



Slika 5. Izometrija nosača motora

Kako bi nosač imao svoju funkcionalnost, potrebno je modelirati nekoliko značajki. Središnji utor za rukavac je ϕ22,2 mm. Zbog tehnologije FDM (Fused Filament Fabrication) čiji CNC pisač će izraditi ovaj komad, dimenzijska točnost komada je ± 0,02 mm. Ako bi utor bio ϕ22 mm, postoji mogućnost da bi stvarni promjer nakon izrade bio ϕ21,8 mm. Takav utor bi se morao na silu nabiti na rukavac motora, a to bi oštetilo dijelove. Nadalje, potrebno je izraditi četiri provrta ϕ3,5 mm koja služe za montažu koračnog motora na nosač. Naposljetku, nosači se moraju nekako pričvrstiti za ploču. Za to služe četiri provrta ϕ5 mm.

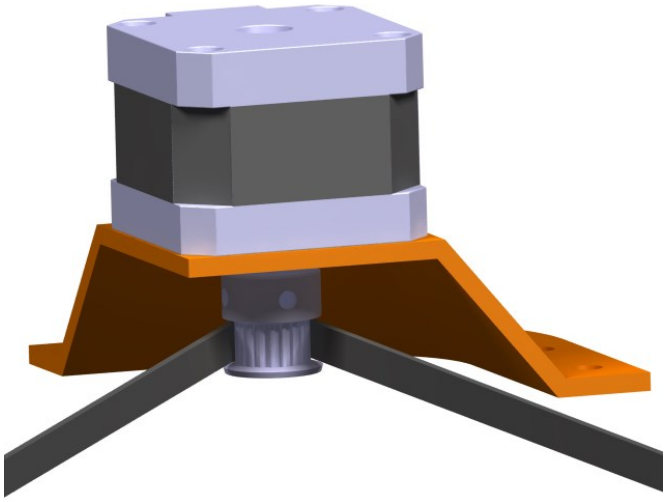


Slika 6. Tlocrt nosača motora

Udaljenost između čeone plohe vratila i čeone plohe rukavca je ϕ22 ± 0,5 mm. To znači da visinska dimenzija unutarnje i vanjske stranice nosača mora biti najmanje 22,5 mm. Uzeta je mjera 24 mm.

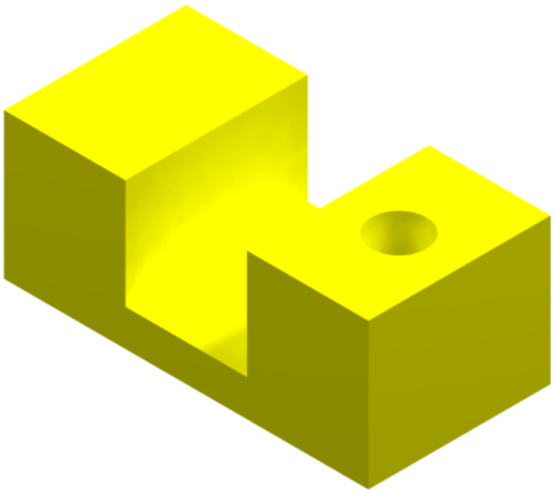
Slika 7. Nacrt nosača motora

Motori su montirani tako da vratila gledaju u ploču kako bi remeni bili što bliže ploči za crtanje da bi se izbjeglo ikakvo savijanje remenica.



Slika 8. Spoj koračnog motora s nosačem

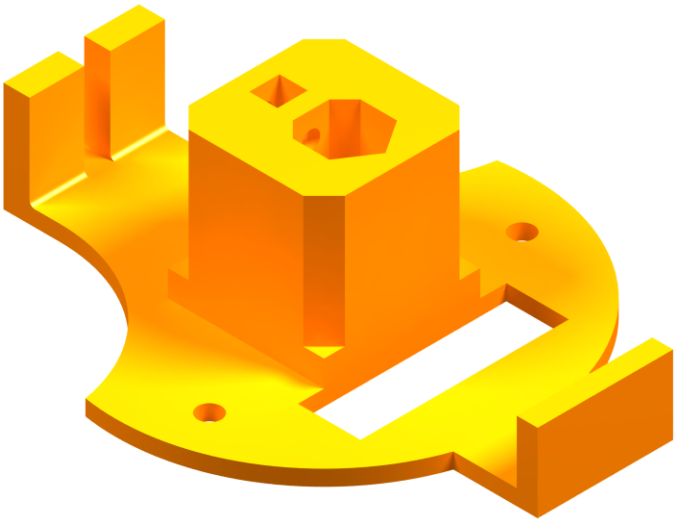
Slijedeći komad koji je potrebno konstruirati je spojka. Ona služi za povezivanje remenica i nosača olovke. Ovaj spoj je vijčano povezan s nosačem olovke.



Slika 9. Spojka

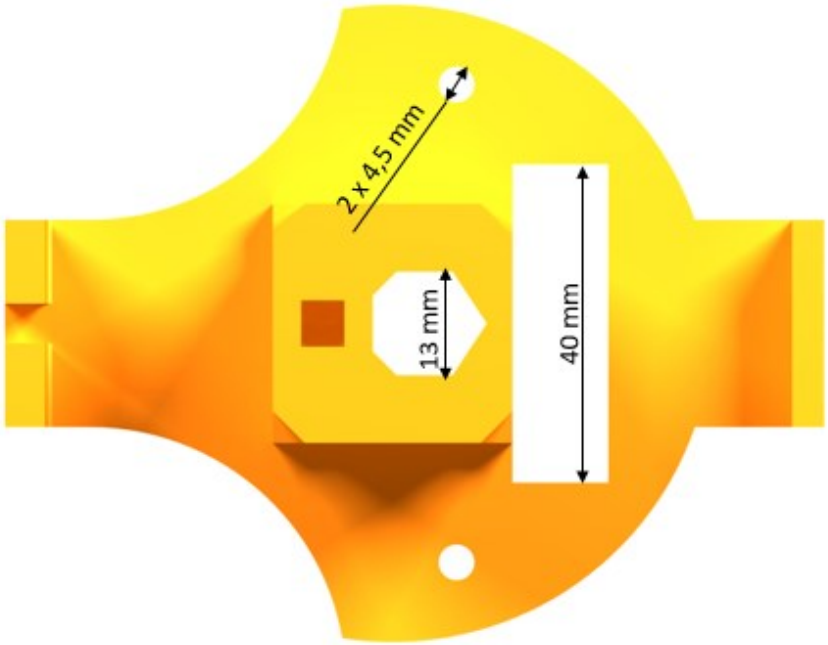
3.2. Modeliranje nosača olovke

Nosač olovke je dio sklopa robotskog crtača čija je osnovna funkcija osiguravanje kontakta olovke sa vertikalnom pločom. Olovka se montira na držač koji osigurava krutu vezu olovke i robota kako bi se eliminirale greške te dobila što točnija slika. Spajanje olovke se može osigurati na bezbroj načina. Ostale funkcije su veza sa servo motorom kako se osiguralo podizanje i spuštanje olovke, prihvat utega koji osiguravaju kontakt olovke sa pločom, te provrti za povezivanje s ostatkom robotskog crtača.



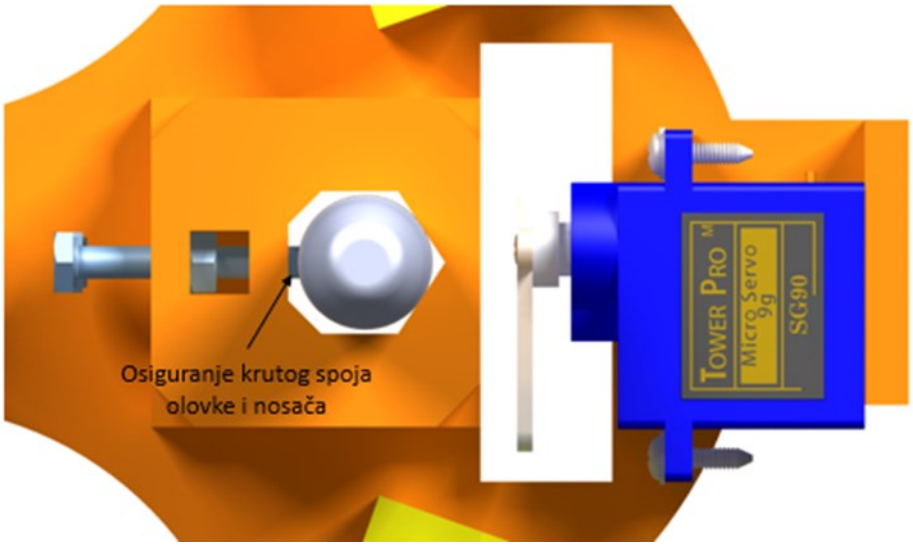
Slika 10. Izometrija nosača olovke

Na nosaču se nalaze više vrsta otvora koji imaju različite funkcionalnosti. Dva provrta ϕ4,5 mm služe za povezivanje nosača olovke i spojke, a preko nje i sa ostatkom robotskog crtača. Pravokutni otvor je oblikovan kako bi kroz njega mogla proći ručica servo motora koja služi za podizanje olovke od ploče za onemogućavanje kontakta s podlogom. Ako je olovka spuštena, cijeli mehanizam počiva na tri točke na ravnini crtanja tako što se olovka povlači okomito preko ploče. Taj mehanizam mora biti što lakši da sile koje djeluju na remen budu niske. Što je olovka bliža koračnim motorima, to su veće sile koje djeluju na remene i stoga su veća odstupanja uzrokovana elastičnošću remena. Na kraju je modeliran otvor heptagonskog oblika kroz koji prolazi olovka. Olovka može biti maksimalno promjera ϕ12,8 mm.

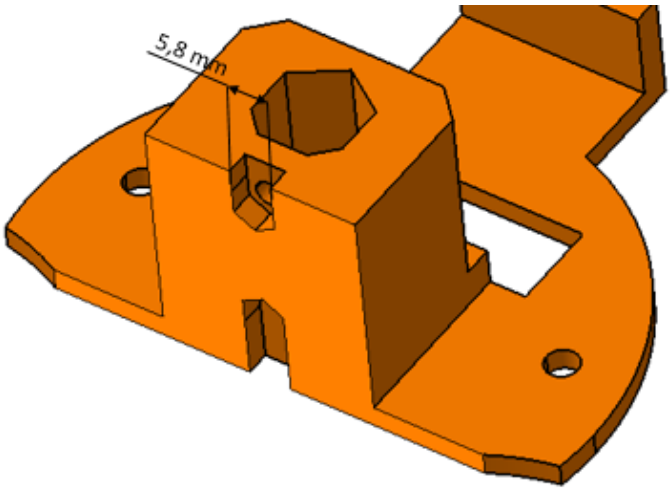


Slika 11. Tlocrt nosača olovke

Otvor je modeliran na 13 mm, no kako stvarna dimenzija može varirati ± 0,2 mm zbog FDM tehnologije, pretpostavlja se da maksimalan promjer olovke smije biti 12,8 mm. Zadnja značajka nosača je način na koji se osigurava krutu vezu. Postoje dva utora čiji je presjek nalik heksagonskoj matici. Odabirom odgovarajućih vijaka i matica, ti utori osiguravaju da se matica ne okreće pri zatezanju vijaka. Čim vijak dotakne olovku, pritezanjem vijaka se osigurava kruti spoj. Širina utora je 5,8 mm.



Slika 12. Osiguranje krutog spoja olovke i nosača



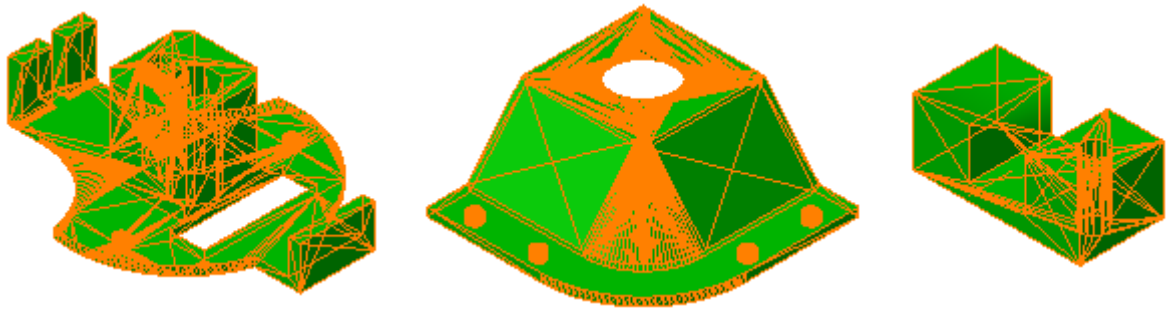
Slika 13. Presjek A nosača motora

4. IZRADA FDM DIJELOVA

Tehnologija taložnog očvršćivanja (FDM) je proces aditivne proizvodnje (AM) koji se često koristi za izradu prototipova i geometrijski složenih dijelova. Ovo je popularna tehnologija jer smanjuje vrijeme ciklusa za razvoj proizvoda bez potrebe za skupim alatima. Međutim komercijalizacija FDM tehnologije u različitim industrijskim primjenama trenutno je ograničena zbog nekoliko nedostataka, kao što su nedovoljna mehanička svojstva, loša kvaliteta površine i niska dimenzijska točnost. Na kvalitetu proizvoda proizvedenih FDM–om utječu različiti procesni parametri kao što su debljina sloja, orijentacija građe, širina rastera ili brzina ispisa[6]. Ipak, u slučajevima gdje nema velikih opterećenja i nije potrebna visoka dimenzijska točnost, komadi izrađeni tehnologijom taložnog očvršćivanja, mogu poslužiti i funkcionalno.

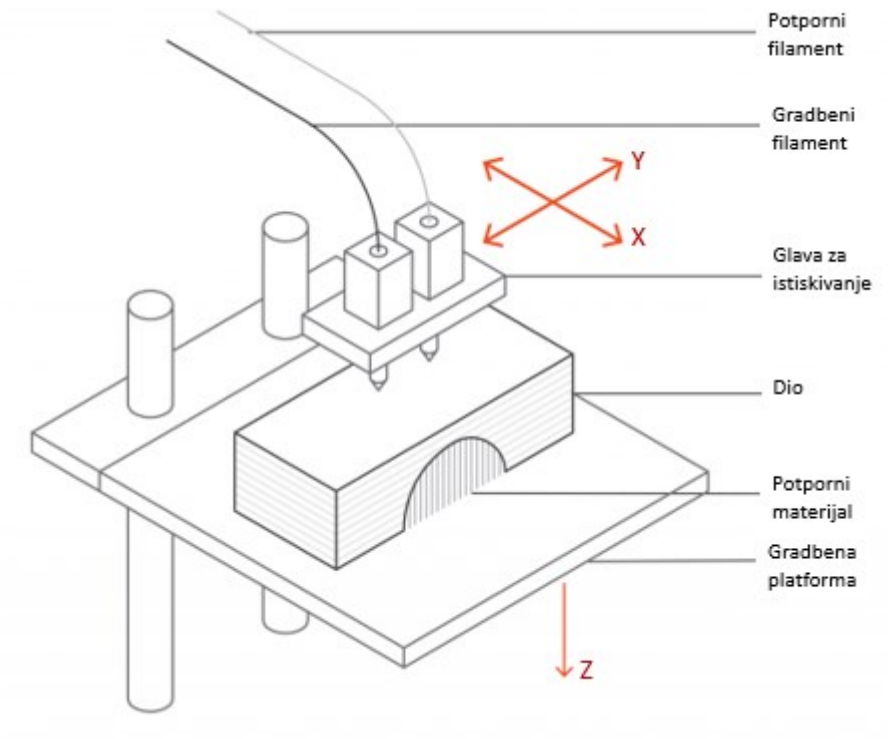
4.1. Proces izrade komada tehnologijom FDM

Prije nego što se bude raspravljano o svojstvima FDM dijela, u ovom poglavlju će se objasniti proces izrade FDM dijela. Prvi korak je stvaranje trodimenzionalnog čvrstog modela. To se može postići dostupnim CAD paketima. Model se zatim izvozi u softver putem stereolitografskog (STL) formata. Prednost STL formata je da ga većina CAD sustava podržava i pojednostavljuje geometriju dijela reducirajući ga na najosnovnije trokutaste komponente. Nedostatak je taj što dio gubi nešto razlučivosti jer je izgrađen od trokuta, a ne pravih lukova, krivulja, itd. Međutim, pogreške uvedene ovim aproksimacijama su prihvatljive sve dok su manje od netočnosti svojstvenih proizvodnom procesu[7].



Slika 14. STL formati dijelova

Nakon što je STL datoteka izvezena u softver koji se općenito zove Slicer, tada se vodoravno reže na mnogo tankih odjeljaka. Ovi dijelovi predstavljaju dvodimenzionalne konture koje će FDM proces generirati koji će, kada se naslažu jedan na drugi, poprimiti oblik izvornog trodimenzionalnog dijela. Što su dijelovi tanji, dio je točniji. Softver zatim koristi te informacije za generiranje plana procesa koji kontrolira hardver FDM stroja[7].



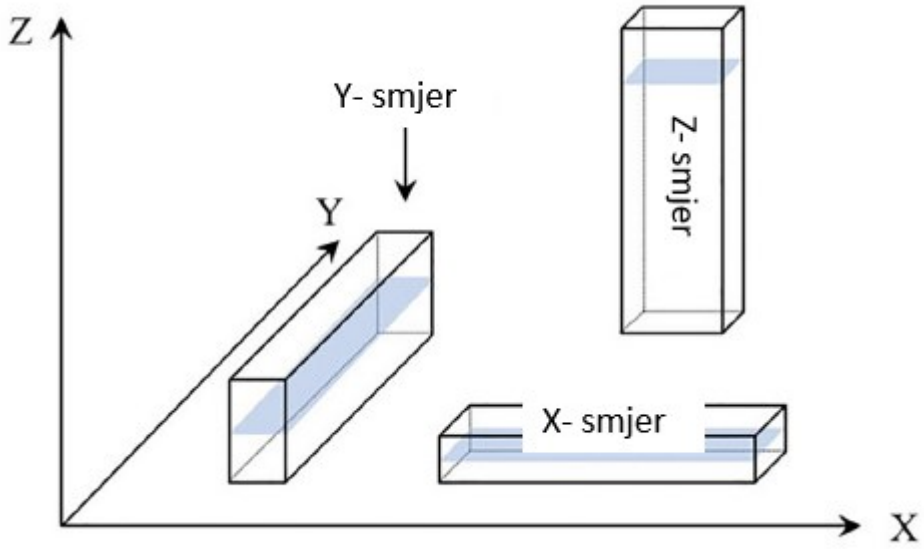
Slika 15. Sastavni dijelovi hardvera[8]

Koncept je da filament, u slučaju ovog rada ABS, prolazi kroz grijaći element, koji ga zagrijava do polu rastaljenog stanja. Filament se zatim dovodi kroz mlaznicu i nanosi na djelomično izgrađeni dio. Ovaj dio procesa se može usporediti sa istiskivanjem paste za zube iz tube. Budući da je materijal ekstrudiran u polu taljenom stanju, stapa se s materijalom oko sebe koji je već nataložen. Glava se zatim pomiče u ravnini X–Y i istiskuje materijal prema zahtjevima dijela iz STL datoteke. Glava se zatim pomiče okomito u Z ravnini kako bi se počeo polagati novi sloj kada je prethodni završen. Nakon određenog vremena, obično nekoliko sati, fizički prikaz izvorne CAD datoteke je izrađen[7].

4.2. Parametri FDM procesa

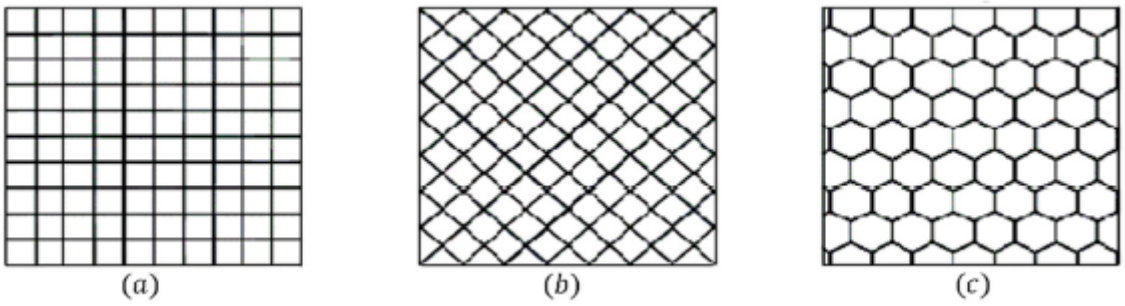
FDM proces ima nekoliko bitnih parametara, a oni imaju značajan utjecaj na proizvodnju dijelova. Neki od najčešćih parametara procesa su zračnost, orijentacija, temperatura ekstruzije, gustoća ispune, uzorak ispune, visina sloja, brzina ispisa, orijentacija rastera te širina. Glavni parametri procesa opisani su u nastavku[7].

1. Orijentacija komada: Opisuje način na koji je komad postavljen na platformu za izradu u odnosu na tri glavne osi X, Y i Z alatnog stroja[6].



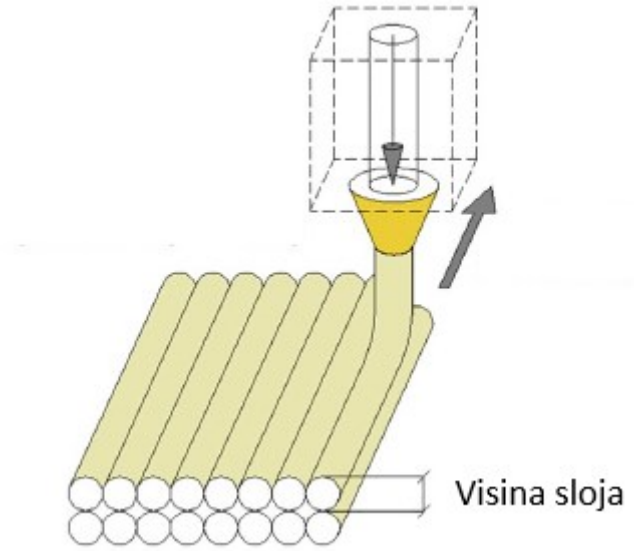
Slika 16. Smjer orijentacije komada u odnosu na glavne osi[7]

2. Zračnost: Razmak između dva susjedna rastera na nataloženom sloju. Zračnost je negativna vrijednost kada se dva susjedna sloja preklapaju[6].
3. Temperatura ekstruzije: Temperatura na kojoj se filament materijala zagrijava tijekom FDM procesa. Temperatura ekstruzije ovisi o različitim aspektima, na primjer o vrsti materijala ili brzine ispisa[6].
4. Gustoća ispune: Vanjski slojevi trodimenzionalnog (3D) objekta pisača su čvrsti. Međutim, unutarnja struktura, obično poznata kao ispuna, nevidljivi je unutarnji dio prekriven vanjski slojevima i ima različite oblike, veličine i uzorke. Gustoća ispune se iskazuje kao postotak od volumena ispune filamentnim materijalom. Snaga i masa dijelova FDM konstrukcije ovise o gustoći ispune[6].
5. Uzorak ispune: Različiti uzorci ispune koriste se za proizvodnju snažne i izdržljive unutrašnjosti struktura. Najčešće su korišteni šesterokutni, dijamantni i linearni uzorci ispune[6].



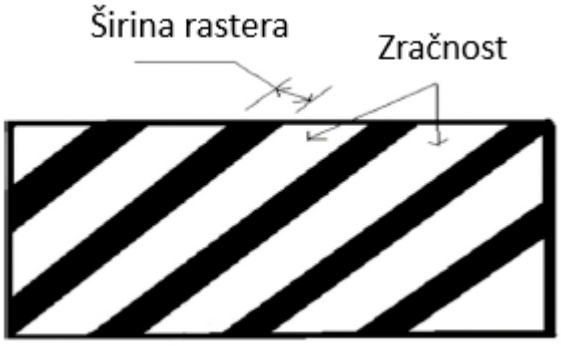
Slika 17. Uzorak ispune: (a) linearni, (b) dijamanti, (c) heksagonalni[6]

6. Visina sloja: Ovo je visina nanesenih slojeva duž Z–osi, što je općenito okomita os FDM stroja. Ovisi o promjeru mlaznice[6].



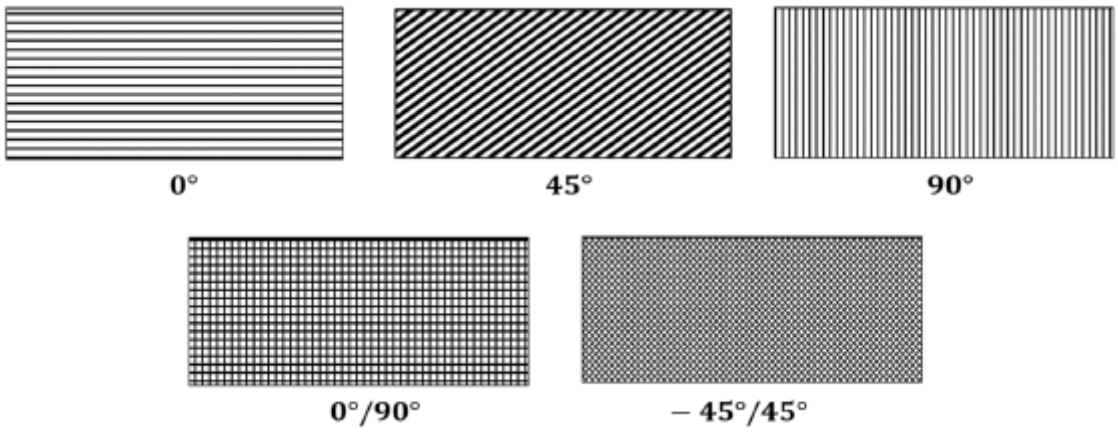
Slika 18. Visina sloja[7]

7. Brzina ispisa: Ovo je udaljenost koju prijeđe ekstruder duž ravnine XY u jedinici vremena. Vrijeme ispisa ovisi o brzini ispisa, a brzina ispisa se mjeri u mm/s[6].
8. Širina rastera: Širina rastera definirana je kao širina zrna koje se taloži. Ovisi o promjeru ekstruzijske mlaznice[6].



Slika 19. Širina rastera i zračnost[6]

9. Orijentacija rastera: Ovo je smjer zrna u odnosu na X–os izgradbene platforme FDM stroja[6].



Slika 20. Orijentacija rastera[7]

4.3. Materijal za tiskanje

U FDM postupku, komad se proizvodi filamentom koji prolazi kroz mlaznicu. Različiti plastomeri se koriste kao filamenti. Za tiskanje je korišten ABS. Akrilonitril butadien stiren jedan je od najčešće korištenih materijala za izradu 3D tiskanih dijelova putem FDM procesa. Otpornost na udarce i žilavost su dva važna mehanička svojstva ABS–a. ABS ima talište od 230°C, što je više od tališta polilaktične kiseline (PLA)[6]. ABS zahtijeva manje sile za istiskivanje od PLA jer ima niži koeficijent trenja. Loša strana ABS–a je ta što se mora ekstrudirati na višoj temperaturi. ABS je amorfan i stoga nema pravo talište, međutim 230°C je standard za tisakanje[9].

4.4. Ispis FDM dijelova

Nakon opisa parametara FDM procesa i uobičajenih materijala, u ovom poglavlju će se izvršiti njihov odabir kao i njihov ispis. Za materijal filamenta odabran je ABS. U slijedećoj tablicu su navedena najvažnija svostva ABS-a:

Tablica 1. Svojstva ABS–a[10][6]

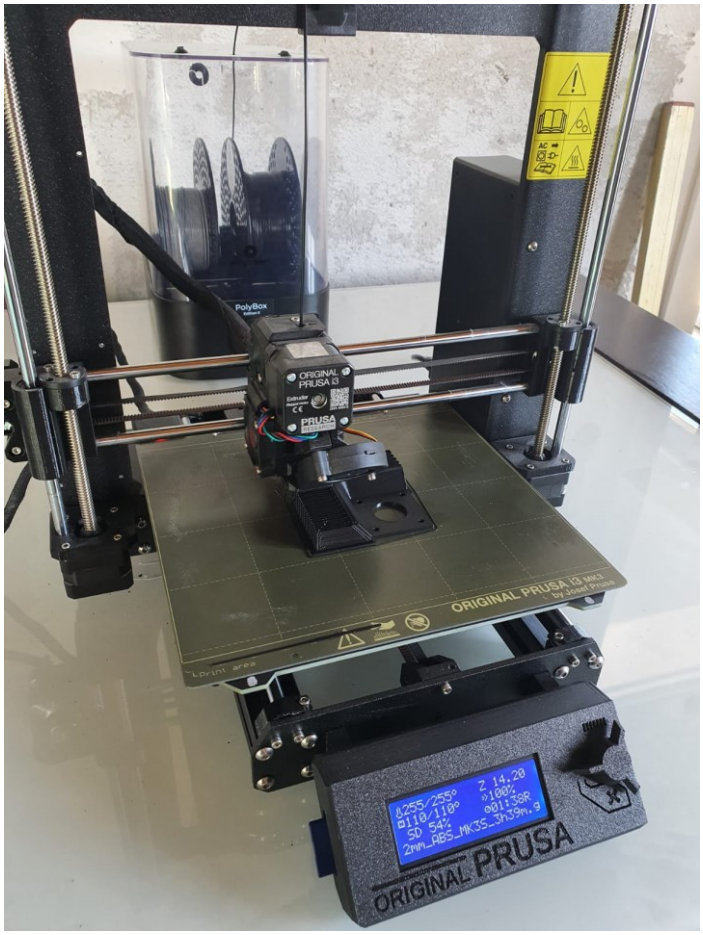
Svojstvo	**Iznos**	**Mjerna jedinica**
Gustoća	1010 – 1210	kg/m^3
Granica razvlačenja	$1,85 \cdot 10^7 – 5,1 \cdot 10^7$	Pa
Vlačna čvrstoća	$2,76 \cdot 10^7 – 5,52 \cdot 10^7$	Pa
Tvrdoća (Vikers)	$5,49 \cdot 10^7 – 1,5 \cdot 10^8$	Pa
Lomna žilavost	$1,19 \cdot 10^6 – 4,29 \cdot 10^6$	$Pa/m^{0,5}$
Youngov modul	$1,19 \cdot 10^9 – 2,9 \cdot 10^9$	Pa
Specifični toplinski kapacitet	1390 – 1920	J/kg°C
Temperatura tiskanja	210 – 250	°C
Temperatura gradbene platforme	80 – 110	°C

Tiskanje 3D dijelova obavit će se na Prusa i3 MK3+ pisaču. Prusa i3 MK3S+ je stolni 3D pisač koji proizvodi češka tvrtka Prusa Research. S cijenom nešto ispod 1000 dolara, MK3S+ nudi dobar omjer cijene i performansi. Ovaj 3D pisač je gotovo identičan popularnom Prusa i3 MK3. Glavna razlika između MK3S+ i MK3 je redizajnirani ekstruder. MK3S+ ekstruder ima hibridni sustav s optičkim senzorom niti i mehaničkom polugom. Volumen izrade ostaje sličan prethodnoj verziji i trebao bi biti više nego dovoljan za većinu ispisa. Ova verzija MK3S također se može pohvaliti manjim poboljšanjima u softveru i hardveru[12].

Tablica 2. Tehničke specifikacije Prusa i3 mk3+ pisača[11]

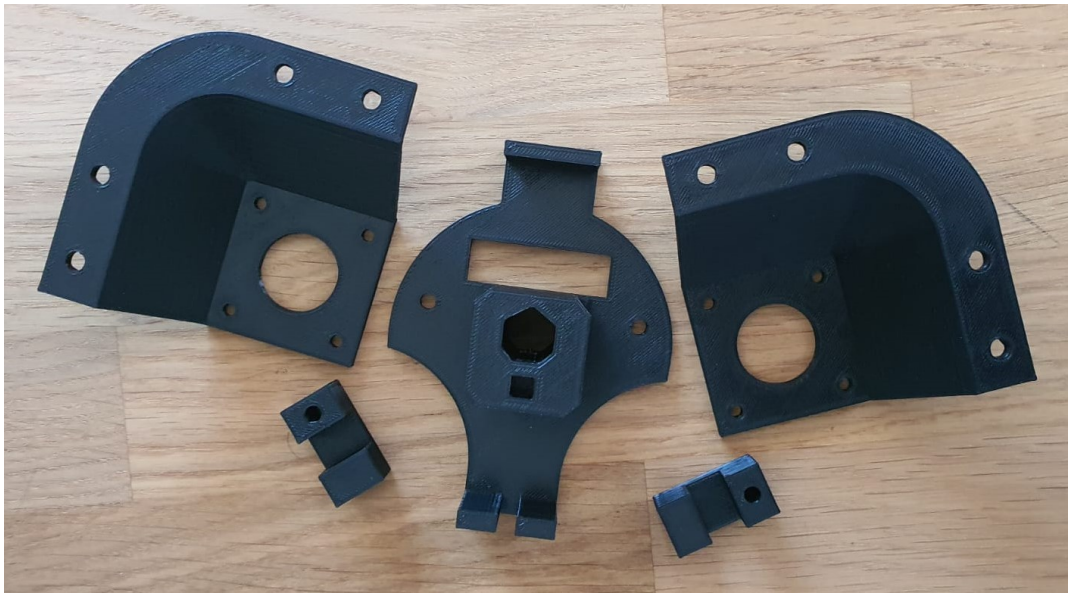
Tehničke specifikacije	
Podržani format datoteke	STL, AMF
Volumen tiska	250 mm x 210 mm x 210 mm
Minimalna visina sloja	0,05 mm
Maksimalna brzina ispisa	Ovisi o materijalu, stroj je sposoban se kretati brzinom od 200 mm/s
Promjer mlaznice	0,4 mm
X/Y rezolucija	0,01 mm/korak
Z rezolucija	0,05 mm/korak

STL datoteke se ubacuju u PrusaSlicer gdje se režu na slojeve. Odabire se orijentacija ispisa. Uzorak ispune je linearni, a gustoća ispune je 100%. Temperatura tiskanja je 230 °C. Visina sloja je 0,01 mm, dok je širina rastera odabrana prema promjeru mlaznice.



Slika 21. Ispisivanje nosača za motor

Nosači motora kao i nosač olovke trebaju potpornu strukturu kako bi dobili kvalitetan obradak. FDM pisači koriste vrlo mali vodoravni pomak (jedva primjetan) između uzastopnih slojeva. Dakle, sloj se ne slaže savršeno preko prethodnog sloja, već se slaže s malim pomakom. To pisaču omogućuje ispis prevjesa koji se ne naginju previše od okomice. Sve ispod 45 stupnjeva može biti podržano prethodnim slojevima, sve veće od te brojke ne. Sve više od 45 stupnjeva smatra da se komad neće moći izraditi. S obzirom na veličinu komada, ne možemo ih sve poslagati odjednom na platformu, već se moraju tiskati iz više puta.



Slika 22. Ispisani dijelovi

Nakon ispisa svih komponenata, pomičnim mjerilom su izmjerene sve najbitnije mjere definirane u prethodnom poglavlju. Sve izmjere su zadovoljavajuće te se može krenuti u slijedeću fazu izrade robotskog crtača.

5. STANDARDIZIRANE KOMPONENTE ROBOTA

Nakon modeliranja, konstruiranja te proizvodnje nestandardnih dijelova robota, potrebno je odabrati ostatak dijelova koji su potrebni za izradu robotskog crtača za ispis fotografija u vertikalnoj ravnini. Potrebno je izabrati pogonske komponente, gonjene komponente, prijenosne komponente, te vijke i matice kako bi osigurali demontažu za transport robotskog crtača. Uz sve to, robot mora imati upravljačke komponente poput mikroupravljača i upravljača koračnih motora. U ovom poglavlju će se promatrati izbor svih standardiziranih dijelova kako bi se krenulo u montažu robotskog crtača. U nastavku slijedi tablica svih komponenti.

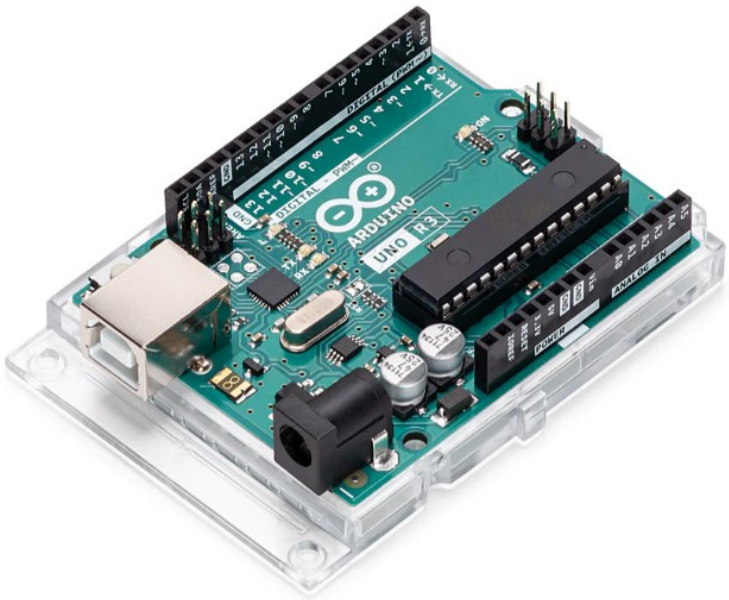
Tablica 3. Popis dijelova robota

Komponenta	**Količina**
Nosač olovke	1
Nosač motora	2
Spojka	2
Arduino UNO R3	1
CNC Shield V3	1
A4988 upravljač koračnih motora	2
Mikro servo SG90	1
GT2 remenica	2
GT2 remen	2
Ispravljač 12 V, 12,5 A	1
Kabel mrežni 3 x 1,5 mm	1
Priključak dupont žica žensko- ženski	1
Priključak dupont žica muško- ženski	8
Priključak dupont žica muško- muški	2
Uteg nosača olovke	1
Uteg	2
DIN 7986 Križni vijak M3 x 6 mm	8

DIN 913 utični vijak M3 x 4 mm	4
DIN 7985 M4 x 25 mm Križni vijak	8
DIN 934 M4 Šesterokutne matice	10
DIN 127 M4 Elastični prsten	8
DIN 9427 M3 x 20 mm vijak	2
DIN 934 M3 matica	2
DIN 84 A M4 x 20 mm vijak	2

5.1. Arduino UNO R3

Arduino Uno R3 je mikroupravljačka ploča temeljena na ATmega328P procesoru. Ima 14 digitalnih ulazno/izlaznih priključaka (od kojih se 6 može koristiti kao PWM izlazi), 6 analognih ulaza, keramički rezonator od 16 MHz, USB priključak, utičnicu za napajanje, ICSP zaglavlje i tipku za resetiranje . Sadrži sve što je potrebno za podršku mikroupravljača, jednostavno se spaja na računalo USB kabelom te se može napajati AC/DC adapterom ili baterijom[14].



Slika 23. Arduino UNO R3[14]

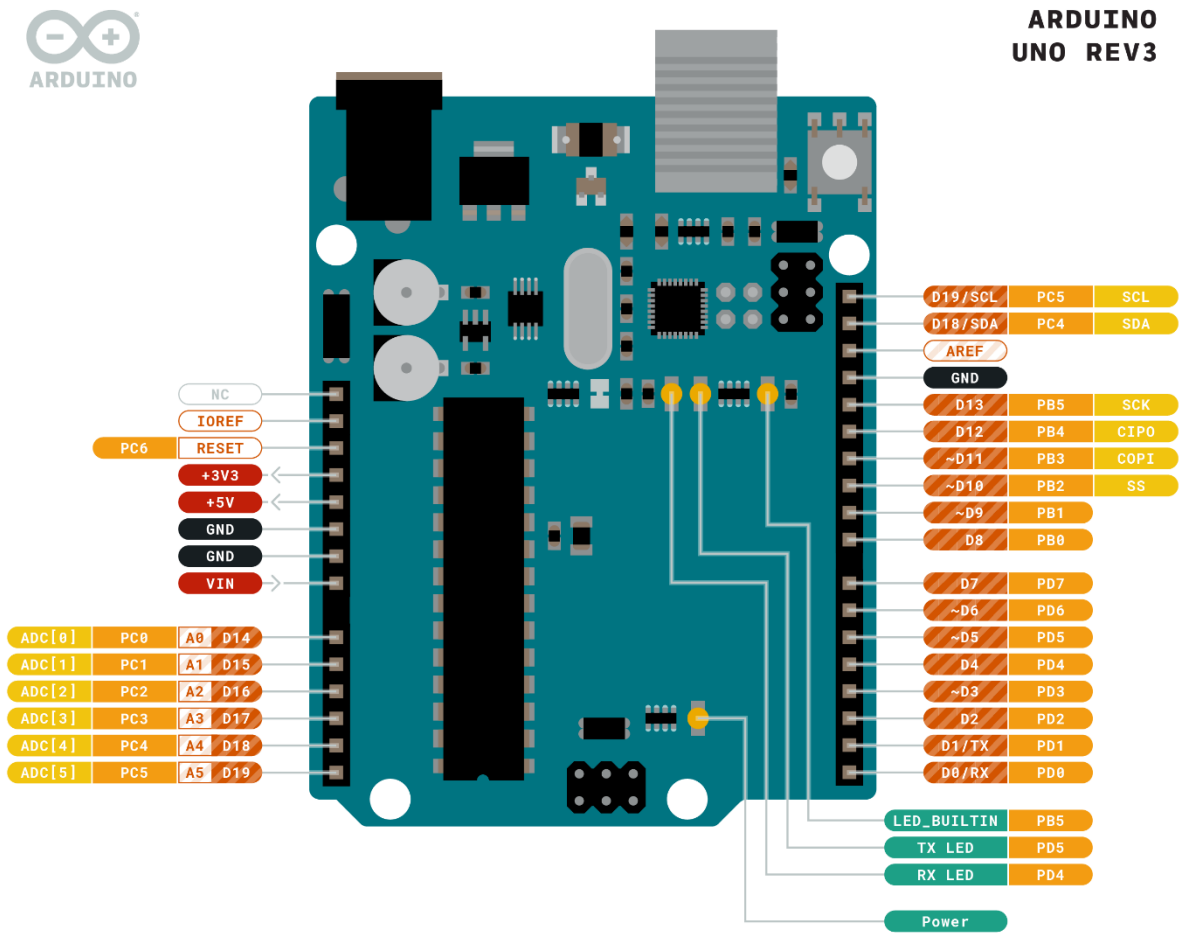
„Uno“ znači jedan na talijanskom i to ime odabrano je za obilježavanje izdanja Arduino softvera (IDE) 1.0. Uno ploča i verzija 1.0 Arduino softvera (IDE) bile su referentne verzije Arduina, koje su sada evoluirale u novija izdanja. Uno ploča je prva u nizu USB Arduino ploča

te je reprezentativni model za Arduino platformu[14]. Ovo je treća verzija Una (R3), koja ima nekoliko promjena u odnosu na prijašnje modele:

1. Čip USB kontrolera promijenio se iz ATmega16U2 (8K flash) u ATmega16U2 (16K flash). Ovo ne povećava dostupni flash ili RAM[15].
2. Dodana su tri nova priključka, a sve su duplikati prethodnih priključaka. I2C priključci (A4, A5) također su izvučeni sa strane ploče blizu AREF–a. Pored priključka za resetiranje nalazi se IOREF priključak, koji je duplikat 5V priključka[15].
3. Gumb za resetiranje sada se nalazi pored USB priključka, što ga čini dostupnijim kada se koristi CNC štit[15].

Tablica 4. Tehničke specifikacije Arduino Uno R3[14]

Mikrokontroler	ATmega328P
Radni napon	5 V
Ulazni napon (preporučeno)	7 – 12 V
Ulazni napon (granični)	6 – 20 V
Digitalni I/O priključci	14 (od kojih 6 daje PWM izlaz)
PMW digitalni I/O priključci	6
Analogni ulazni priključci	6
DC struja po I/O priključku	20 mA
DC struja za 3.3V priključak	50 mA
Flash memorija	32 KB (ATmega328P)
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Brzina sata	16 MHz
LED_BUILTIN	13
Duljina	68,6 mm
Širina	53,4 mm
Masa	25 g



Slika 24. Dijagram priključaka[14]

Narančasto obojani priključci su priključci na ATmega328P procesoru koji odgovaraju različitim Arduino funkcijama (digitalni priključci, analogni ulazi, napajanje, uzemljenje, itd.).

5.1.1. Napajanje

Arduino Uno ploča može se napajati putem USB veze ili vanjskim napajanjem. Vanjsko (ne–USB) napajanje može dolaziti iz AC/DC adaptera ili baterije. Adapter se može spojiti uključivanjem 2,1 mm središnjeg pozitivnog utikača u utičnicu za napajanje ploče. Izvodi iz baterije mogu se umetnuti u GND i VIN priključke. Ploča može raditi na vanjskom napajanju od 6 do 20 V. Međutim, ako se napaja s manje od 7 V, priključak od 5 V može dati manje od 5 V i ploča može postati nestabilna. Ako se koristi više od 12 V, regulator napona se može pregrijati i oštetiti ploču. Preporučeni raspon je od 7 do 12 V[14]. Priključci za napajanje su sljedeći:

1. VIN: Ulazni napon za Arduino ploču kada se koristi vanjski izvor napajanja (za razliku od 5 volti iz USB veze ili drugog reguliranog izvora napajanja) [14].

2. 5 V: Ovaj priključak emitira reguliranih 5 V iz regulatora na ploči. Ploča se može napajati iz DC priključka za napajanje (7 – 12 V), USB konektora (5 V) ili VIN priključka na ploči (7 – 12 V). Napajanje preko priključaka od 5 V ili 3,3 V zaobilazi regulator i može oštetiti vašu ploču što znači da se 5 V i 3,3 V ne koriste za napajanje Arduina, već za napanje drugih komponenti iz Arduina[14].
3. 3,3 V: Napajanje od 3,3 V koje stvara ugrađeni regulator. Maksimalna potrošnja struje je 50 mA[14].
4. GND: Uzemljenje[14].
5. IOREF. Ovaj priključak na Arduino ploči daje referentni napon s kojim mikrokontroler radi[14].

5.1.2. Ulazi i izlazi

Svaki od 14 digitalnih priključaka na Unu može se koristiti kao ulaz ili izlaz, koristeći funkcije pinMode() (funkcija koja daje mogućnost hoće li priključak biti ulaz ili izlaz), digitalWrite() (omogućuje postavljanje stanja na priključku NISKO ili VISOKO) i digitalRead() (omogućuje čitanje stanja na priključku NISKO ili VISOKO). Rade na 5 V. Svaki priključak može dati ili primiti 20 mA struje i ima unutarnji otpornik za podizanje struje (isključen prema zadanim postavkama) od 20 – 50 kΩ. 40 mA je maksimalna vrijednost koja se ne smije prekoračiti ni na jednom I/O priključku kako bi se izbjeglo trajno oštećenje mikroupravljača[14]. Osim toga, neki priključci imaju posebne funkcije:

1. Serijski: 0 (RX) i 1 (TX). Koristi se za primanje (RX) i prijenos (TX) TTL serijskih podataka. Ovi priključci su povezani s odgovarajućim priključcima na ATmega16U2 serijskom čipu[14].
2. Vanjski prekidači: 2 i 3. Ovi se priključci mogu konfigurirati za pokretanje prekida na niskoj vrijednosti, rastućem ili padajućem rubu ili na promjeni vrijednosti. Pogledati funkciju attachInterrupt() za detalje[14].
3. PWM: 3, 5, 6, 9, 10 i 11. Omogućava 8-bitni PWM izlaz s funkcijom analogWrite()[14].
4. SPI: 10, 11, 12, 13. Ovi priključci podržavaju SPI komunikaciju pomoću SPI knjižnice[14].
5. LED: 13. Ugrađena je LED dioda koju pokreće digitalni priključak 13. Kada je priključak VISOK, LED je uključen, a kada je NIZAK, isključen je[14].
6. TWI: A4 ili SDA priključak i A5 ili SCL priključak. Podržava TWI komunikaciju pomoću Wire knjižnice[14].

Uno ima 6 analognih ulaza, označenih od A0 do A5, od kojih svaki daje 10 bita rezolucije (tj. 1024 različite vrijednosti). Prema zadanim postavkama mjeri napon do 5 V, iako je moguće promijeniti gornju granicu njegovog raspona pomoću priključka AREF i funkcije analogReference()[14]. Postoji nekoliko drugih priključaka na ploči:

1. AREF: Referentni napon za analogne ulaze. Koristi se s analogReference() funkcijom[14].
2. Reset priključak: Koristi NISKO stanje za resetiranje mikrokontrolera. Obično se koristi za dodavanje gumba za resetiranje štitovima koji blokiraju onaj na ploči[14].

5.1.3. Komunikacija

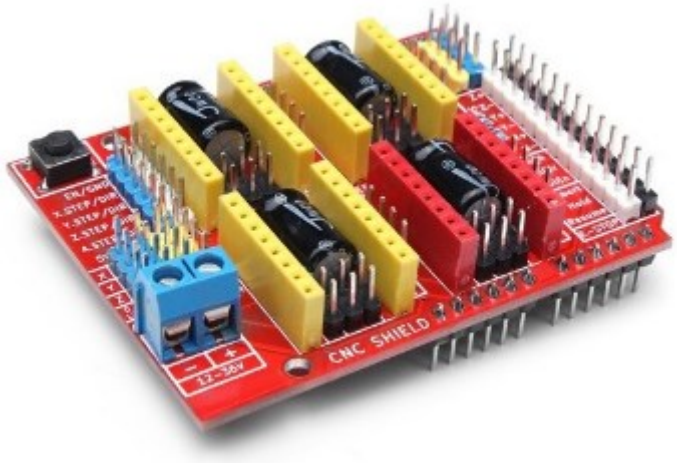
Arduino Uno ima niz mogućnosti za komunikaciju s računalom, drugom Arduino pločom ili drugim mikrokontrolerima. ATmega328P pruža UART TTL (5 V) serijsku komunikaciju, koja je dostupna na digitalnim priključcima 0 (RX) i 1 (TX). ATmega16U2 na ploči kanalizira ovu serijsku komunikaciju preko USB – a i pojavljuje se kao virtualni COM priključak za softver na računalu. Firmware 16U2 koristi standardne USB COM upravljačke programe i nije potreban vanjski upravljački program. Međutim, sustavu Windows potrebna je .inf datoteka. Softver Arduino (IDE) uključuje serijski monitor koji omogućuje slanje jednostavnih tekstualnih podataka na i s ploče. RX i TX LED na ploči će treperiti kada se podaci prenose preko USB – a na serijski čip i USB veze na računalo. Knjižnica SoftwareSerial omogućuje serijsku komunikaciju na bilo kojem od Uno digitalnih priključaka[14].

ATmega328P također podržava I2C (TWI) i SPI komunikaciju. Softver Arduino (IDE) uključuje knjižnicu Wire za pojednostavljenje korištenja I2C sabirnice. Za SPI komunikaciju mora se koristiti SPI knjižnica[14].

5.2. CNC V3 štit

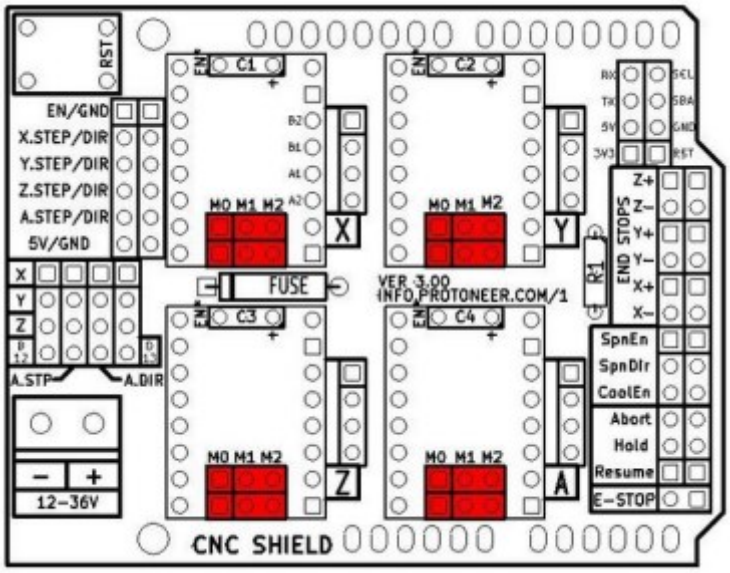
CNC V3 štit je dizajniran kako bi se zadovoljila potražnja za jeftinim upravljačkim rješenjem za male CNC strojeve. Dizajniran je tako da bude potpuno kompatibilan s Arduino Uno mirkoupravljačkom pločom. CNC štit se može koristiti za upravljanje raznim CNC strojevima, uključujući CNC strojeve za glodanje, strojeve za lasersko graviranje / rezanje, strojeve za crtanje, 3D pisače ili bilo koji projekt koji zahtijeva preciznu kontrolu koračnih motora. Pogodan je za korištenje s upravljačem koračnog motora A4988, ali i sa DRV8825. DRV885 omogućuje veće struje u usporedbi s A4988[16]. U nastavku slijedi nekoliko osnovnih funkcija:

1. Verzija 3.00
2. Prilagođen za A4988 ili DRV8825 upravljače koračnog motora
3. Podrška za 4 osi (X, Y, Z, A. A os može duplicirati X, Y, Z os ili se koristiti kao 4. os s prilagođenim firmwareom pomoću priključaka D12 i D13)
4. 2 x granične sklopke za svaku os (6 prekidača ukupno – svaki par prekidača na istoj osi dijeli isti IO priključak)
5. Omogućavanje i usmjeravanje funkcija za vreteno
6. Koristi GRBL kao upravljački softver
7. Napajanje: DC 12 – 36 V (samo upravljači DRV8825 mogu izdržati do 36 V, ne preporuča se prekoračiti 24 V kada se koristi A4988)
8. Koračni motori se mogu spojiti na 4 – pinske priključke
9. Pomoću kratkospojnika se može postaviti mikrokorak koračnog motora



Slika 25. CNC Shield V3[16]

Svaka os ima 3 kratkospojnika koji se mogu postaviti za konfiguriranje mikrokoraka za A4988 upravljače koračnih motora. U tablici ispod „Visoko“ označava da je kratkospojnik umetnut, a „Nisko“ označava da kratkospojnik nije umetnut[17].



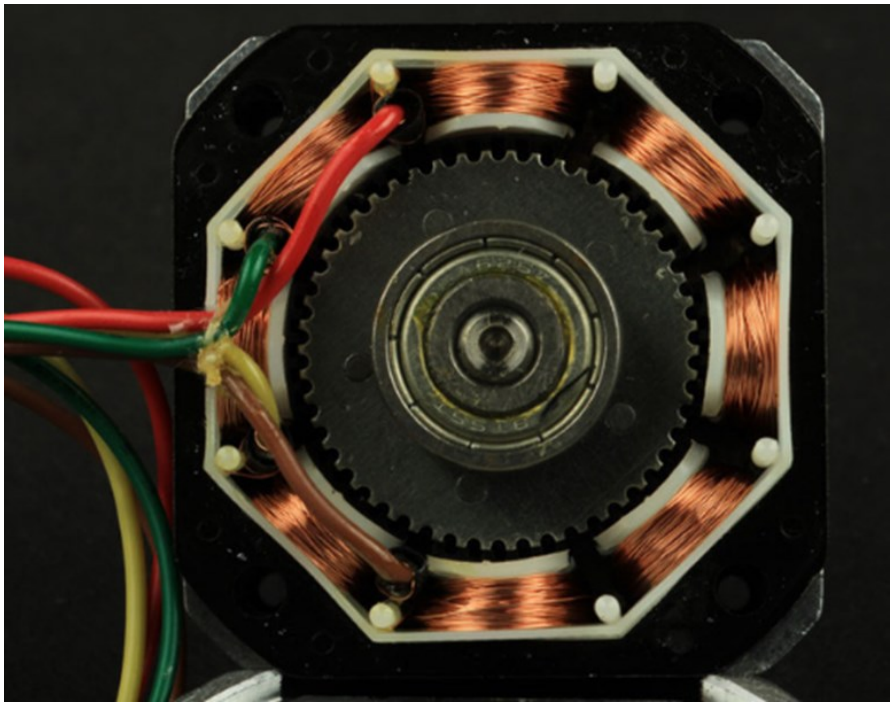
Slika 26. Lokacija kratkospojnika za mikrokorak[17]

Tablica 5. Mikrokorak[17]

MS0	MS1	MS2	Rezolucija mikrokoraka
Nisko	Nisko	Nisko	Puni korak
Visoko	Nisko	Nisko	1/2 koraka
Nisko	Visoko	Nisko	1/4 koraka
Visoko	Visoko	Nisko	1/8 koraka
Visoko	Visoko	Visoko	1/16 koraka

5.3. Koračni motor

Koračni motori su istosmjerni motori koji se kreću u diskretnim koracima. Imaju više zavojnica koje su organizirane u skupine koje se nazivaju faze. Uključujući svaku fazu u nizu naizmjenično, motor će se okretati, korak po korak. S računalno kontroliranim korakom može se postići vrlo precizno pozicioniranje i/ili kontrola brzine. Iz tog razloga, koračni motori služe za preciznu kontrolu pokreta[18].



Slika 27. Unutrašnjost dvofaznog koračnog motora[18]

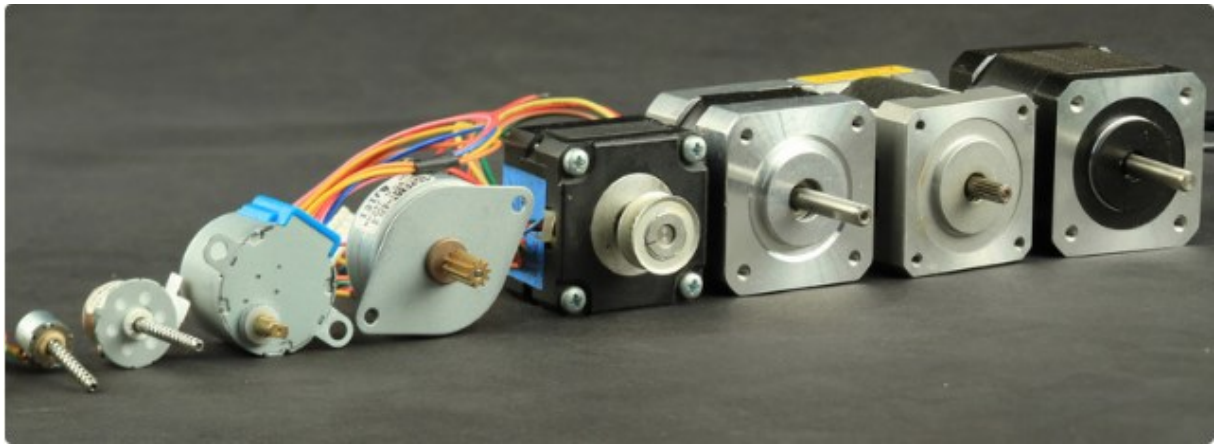
Jedna od prvih stvari koje treba uzeti u obzir je koliko težak posao motor mora obaviti. Kao što se može očekivati, veći motori mogu isporučiti više snage. Većina proizvođača u specifikacije motora stavlja maksimalni okretni moment. NEMA 17 je uobičajeni koračni motor koji se koristi u 3D pisačima i manjim CNC glodalicama. Manji motori nalaze primjenu u mnogim robotskim i mehatroničkim aplikacijama. Veći NEMA motori uobičajeni su u CNC strojevima i industrijskim aplikacijama[18].

NEMA brojevi definiraju standardne dimenzije prednje ploče za montažu motora. Oni ne definiraju nikakve druge karakteristike motora. Dva različita NEMA 17 motora mogu imati potpuno različite električne ili mehaničke specifikacije i nisu nužno zamjenjivi[18].

5.3.1. Broj koraka

Sljedeća stvar koja se treba uzeti u obzir je razlučivost pozicioniranja koja je potrebna. Broj koraka po okretu kreće se od četiri do četiristo. Uobičajeno dostupni broj koraka je 24, 46 i 200. Rezolucija se često izražava u stupnjevima po koraku. Motor od 1,8° je isto što i motor od 200 koraka[18].

Kompromis za visoku rezoluciju je manja brzina i zakretni moment. Motori sa većim brojem koraka imaju manji okretni moment od motora slične veličine s manjim brojem koraka pri sličnim brzinama[18].



Slika 28. Koračni motori[18]

5.3.2. Ožičenje

Postoje mnoge varijante ožičenja koračnog motora. Najčešći koračni motori su ožičeni kao dvofazni bipolarni ili četverofazni unipolarni koračni motori[18].

Koračni motor može imati bilo koji broj zavojnica. Ali oni su povezani u skupine koje se nazivaju faze. Unipolarni upravljači, uvijek napajaju faze na isti način. Unipolarni upravljači mogu se implementirati jednostavnim tranzistorskim sklopom. Nedostatak je manje dostupnog okretnog momenta jer se samo polovica zavojnica može napajati odjednom[18].

Bipolarni upravljački programi koriste sklop H – mosta da preokrenu protok struje kroz faze. Napajanjem faza s izmjeničnim polaritetom, sve zavojnice se mogu pokrenuti okretanjem motora[18].

5.3.3. 17HS3401 koračni motor

Za robotskog crtača izabran je 17HS3401 koračni motor. Broj 17 u imenu označava dimenzije prednje ploče za montažu prema NEMA standardu. 17 zapravo znači 1,7 inča što odgovara 42 mm dimenziji. Broj 34 u imenu označava duljinu motora u milimetrima bez osovine. Ostali tehnički podaci dani su u slijedećoj tablici:

Tablica 6. Tehničke specifikacije 17HS3401 koračnog motora[19]

Stupanj koraka	1,8°
Duljina motora	34 mm
Nazivna struja	1,3 A
Otpor faze	2,4 Ω
Fazni iduktivitet	2,8 mH
Zaostali moment (minimalni)	28 Ncm
Zakretni moment (maksimalni)	1,6 Ncm
Inercija rotora	34 gcm^2
Broj električnih vodiča	4
Masa motora	220 g

Otpor faze će ograničiti koliko struje možemo primijeniti na svaku fazu. Induktivitet zavojnica motora ograničit će koliko brzo možemo povećati struju zavojnice. Kao i u stvarnom životu, ništa se ne može promijeniti iz jednog stanja u drugo bez ikakve vremenske odgode. U slučaju koračnog motora, struji koja prolazi kroz svaku zavojnicu, potrebno je neko vrijeme da zavojnica bude pod naponom. To je zbog toga što zavojnica ima induktivitet (izražen u Henrijima, skraćeno slovom H) koji ima prirodnu tendenciju da se odupre protoku struje koja se brzo mijenja. Veći induktivitet zavojnice rezultira sporijom brzinom promjene struje, a time i sporijom brzinom širenja i skupljanja magnetskog polja[20].

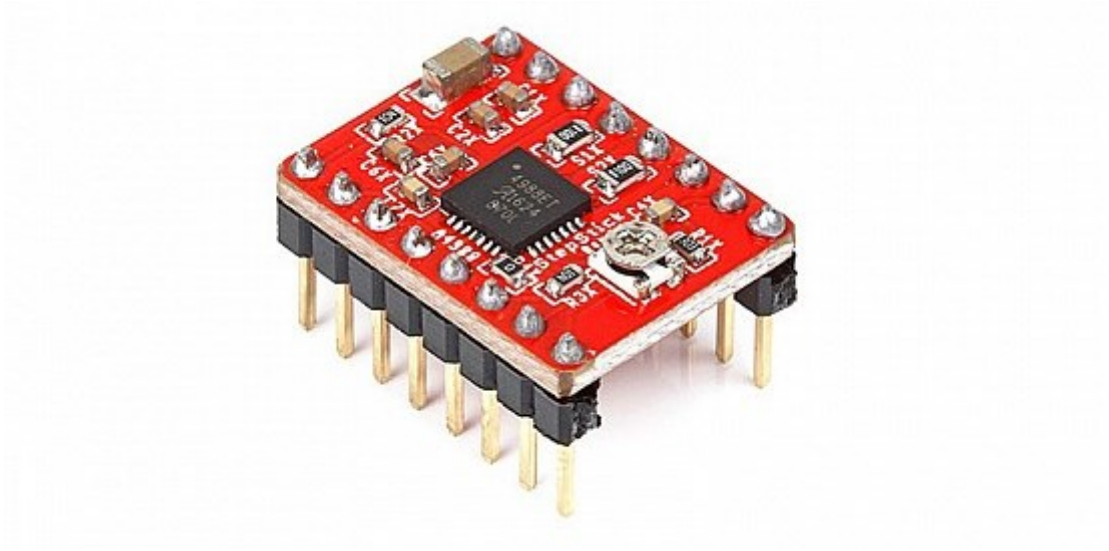
Maksimalni moment koji koračni motor može postići je kada motor miruje s jednim namotajem pod naponom. Kako se brzina kojom se svaka zavojnica napaja i isključuje povećava kako bi izazvala rotaciju osovine, vrijeme u kojem svaka zavojnica može izvršiti svoju punu magnetsku privlačnost na rotoru se smanjuje, čime se smanjuje ukupni zakretni moment. Ovaj odnos između brzine i momenta je uglavnom obrnuto proporcionalan[20]. To znači što je manji induktivitet, potrebno je manje vremena da se postigne puna struja na zavojnici, a time i nazivni zakretni moment.

Jedan od načina povećanja brzine motora uz održavanje momenta je povećanje brzine pri kojoj se magnetsko polje zavojnica motora može širiti i skupljati. Najlakši način da se to postigne je povećanje napona napajanja kako bi se struja u svakom namotu povećala i smanjila mnogo brže[20].

Koračni motor također može držati mali teret na mjestu kada na namotajima nema struje (na primjer, u stanju isključenosti). To se naziva zaostali moment. Drugim riječima, zaostali moment je količina momenta koji motor ima kada namoti nisu pod naponom. Učinak zaostalog momenta može se osjetiti kada se osovina motora pomiče rukom[21].

5.4. A4988 Upravljač koračnih motora

Upravljanje koračnim motorom je kompliciranije od upravljanja običnog istosmjernog motora. Koračni motori zahtijevaju koračni upravljač za napajanje faza u pravodobnom slijedu kako bi se motor okrenuo. A4988 je mikrokoračni upravljač za upravljanje bipolarnim koračnim motorima koji ima ugrađeni prevoditelj za jednostavno rukovanje. To znači da koračni motor se može upravljati sa samo dva Arduino priključka, jednim za kontrolu smjera vrtnje, a drugim za kontrolu koraka[22].

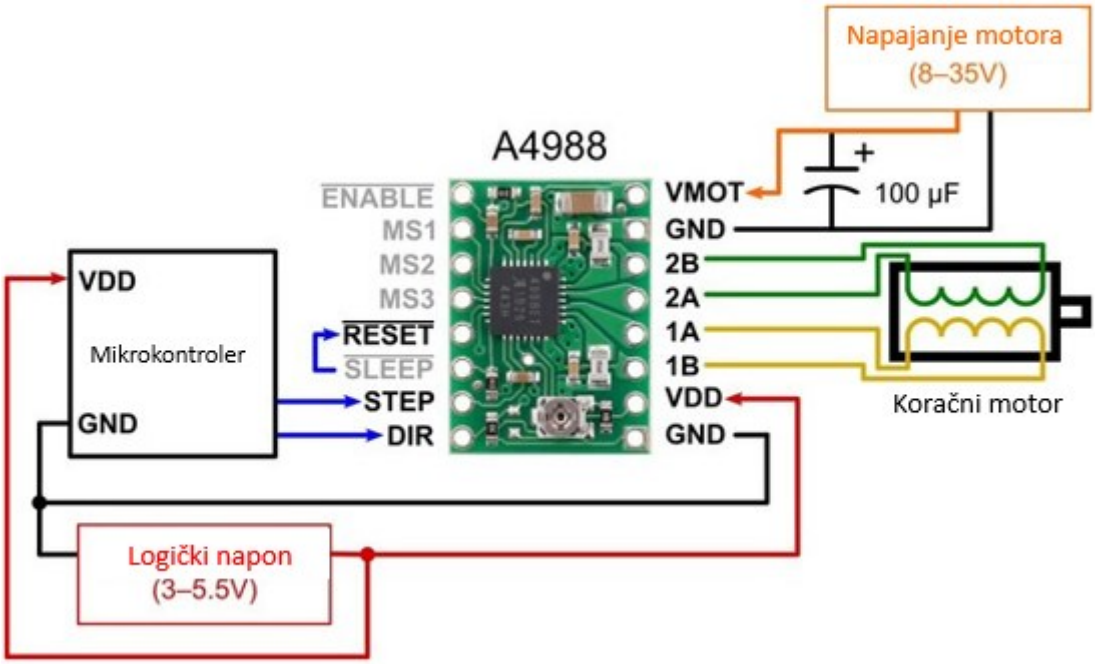


Slika 29. A4988 Upravljač[23]

A4988 nudi pet različitih razlučivosti koraka: cijeli korak, polu-korak, četvrtinu koraka, osminu koraka i šesnaestinu koraka. Također, ima potenciometar za podešavanje izlazne struje, i termičko isključivanje od previsoke temperature. Njegov logički napon je od 3 do 5,5 V, a maksimalna struja po fazi je 2 A ako je osigurano dobro dodatno hlađenje ili 1 A kontinuirane struje po fazi bez hladnjaka ili hlađenja[22].

Tablica 7. Specifikacije A4988 upravljača koračnih motora[22]

Minimalni logički napon	3 V
Maksimalni logički napon	5,5 V
Kontinuirana struja po fazi	1 A
Maksimalna struja po fazi	2 A
Minimalni radni napon	8 V
Maksimalni radni napon	35 V



Slika 30. Shema upravljanja koračnog motora pomoću A4988 upravljača[24]

VDD i GND koriste se za pokretanje unutarnjeg logičkog sklopa koji može biti u rasponu od 3 V do 5,5 V. Arduino Ploča će osigurati 5 V logičkog napona. Sljedeća četiri priključka služe za spajanje motora. Priključci 1A i 1B bit će spojeni na jednu zavojnicu motora, a kontakti 2A i 2B na drugu zavojnicu motora. Za napajanje motora se koriste sljedeća dva priključka, GND i VMOT koji su potrebni za spajanje na napajanje od 8 do 35 V[22].

Sljedeća dva priključka, STEP (korak) i DIR (smjer) su priključci koji se koriste za kontrolu pokreta motora. Priključak DIR kontrolira smjer rotacije motora i spaja se na jedan od digitalnih priključaka na Arduinu. Pomoću priključka STEP kontrolira se korak motora i sa svakim impulsom koji se pošalje na ovaj priključak, motor se pomakne za jedan korak. To znači da za upravljanje ne treba nikakvo složeno programiranje, tablice slijeda faza, linije za kontrolu frekvencije itd., jer ugrađeni prevoditelj A4988 upravljača se brine o svemu[22]. Svaki visoki

impuls poslan na ovaj priključak pokreće motor u skladu s brojem mikrokoraka određenih priključcima za odabir mikrokoraka. Što je puls brži, motor će se brže vrtjeti[25].

Sljedeći je SLP Priključak i logičko stanje nisko stavlja ploču u stanje mirovanja radi smanjenja potrošnje energije kada se motor ne koristi. Zatim, RST priključak postavlja prevoditelj u unaprijed definirano početno stanje. Dakle, ako je stanje ulaza na ovaj priključak logički nizak, svi STEP ulazi će biti zanemareni. RST priključak je plutajući priključak pa ako nema namjere se njime upravljati u programu, mora se spojiti na SLP priključak kako bi se postavio na logičko stanje visoko i omogućio ploču da upravlja koračnim motorima. Sljedeća tri priključka (MS1, MS2 i MS3) služe za odabir jedne od pet razlučivosti koraka. Zadnji priključak EN je aktivni niski ulaz. Kada je ovom priključku logičko stanje nisko, A4988 upravljač je omogućen. Prema zadanim postavkama ovaj priključak ima logičko stanje nisko tako da je upravljač uvijek omogućen osim ako se ne definira unutar programa visoko[22].

5.5. Ispravljač

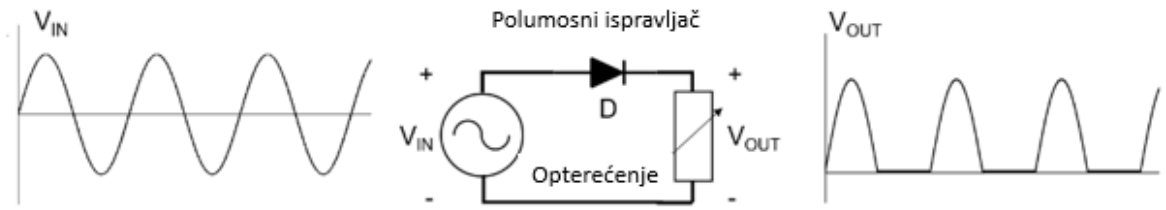
Dugi niz godina linearni AC/DC izvori napajanja pretvaraju izmjeničnu struju iz komunalne mreže u istosmjerni napon za rad kućanskih aparata i rasvjete. Linearno napajanje koristi transformator za smanjenje ulaznog napona. Zatim se napon ispravlja i pretvara u napon istosmjerne struje, koji se zatim filtrira kako bi se poboljšala kvaliteta valnog oblika. Linearni izvori napajanja koriste linearne regulatore za održavanje konstantnog napona na izlazu. No potreba za manjim izvorima energije za aplikacije velike snage znači da su linearni izvori napajanja potisnuti u specifične industrijske i medicinske svrhe, gdje su još uvijek potrebni zbog niske razine buke. Napajanja s prekidanjem struje se sve više koriste jer su manji, učinkovitiji i sposobni su podnijeti veliku snagu[26]. Slika ispod ilustrira opću transformaciju iz izmjenične struje (AC) u istosmjernu struju (DC) u napajanju s prekidanjem struje.

Slika 31. Shema ispravljača AC/DC[26]

5.5.1. Ulazni ispravljač

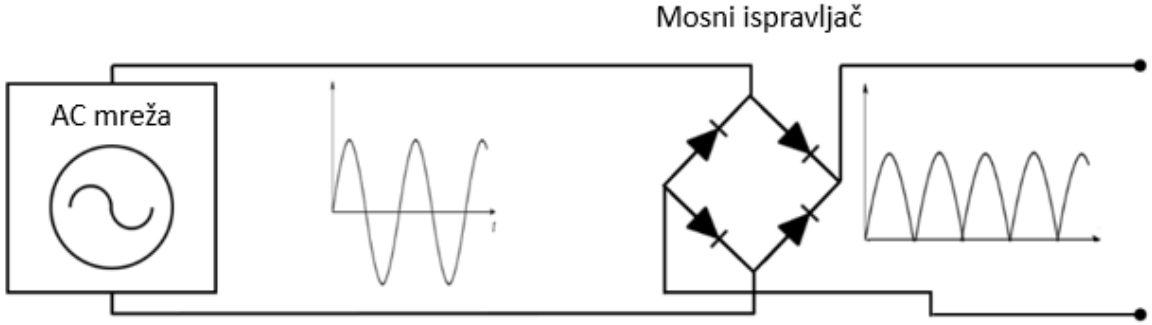
Ispravljanje je proces pretvaranja izmjeničnog napona u istosmjerni napon. Ispravljanje ulaznog signala je prvi korak u AC/DC napajanju s prekidanjem struje. Obično se misli da je istosmjerni napon ravna linija konstantnog napona, poput one koja izlazi iz baterije. Međutim, ono što definira istosmjernu struju (DC) je jednosmjerni tok električnog naboja. To znači da

napon teče u istom smjeru, ali nije nužno konstantan. Sinusni val je najtipičniji valni oblik izmjenične struje (AC) i pozitivan je u prvom poluciklusu, ali negativan u ostatku ciklusa. Ako se negativni poluciklus preokrene ili eliminira, tada struja prestaje biti izmjenična i postaje istosmjerna. To se može postići procesom koji se naziva ispravljanje. Ispravljanje se može postići korištenjem pasivnog polumosnog ispravljača za uklanjanje negativne polovice sinusnog vala pomoću diode. Dioda dopušta protok struje kroz nju tijekom pozitivne polovice vala, ali blokira struju kada teče u suprotnom smjeru[26].



Slika 32. Polumosni ispravljač[26]

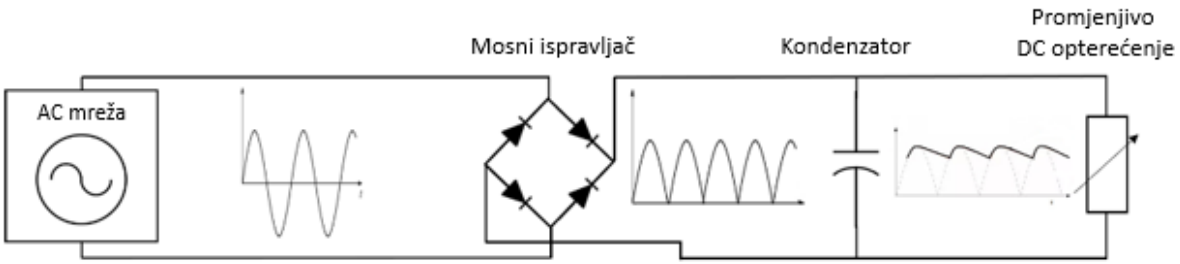
Nakon ispravljanja, rezultirajući val imat će nisku prosječnu snagu i neće moći učinkovito napajati uređaje. Mnogo učinkovitija metoda bila bi promijeniti polaritet negativnog poluvala i učiniti ga pozitivnim. Ova se metoda naziva punovalno ispravljanje i zahtijeva samo četiri diode u mosnoj konfiguraciji. Ovaj raspored održava stabilan smjer protoka struje, bez obzira na polaritet ulaznog napona[26].



Slika 33. Mosni ispravljač[26]

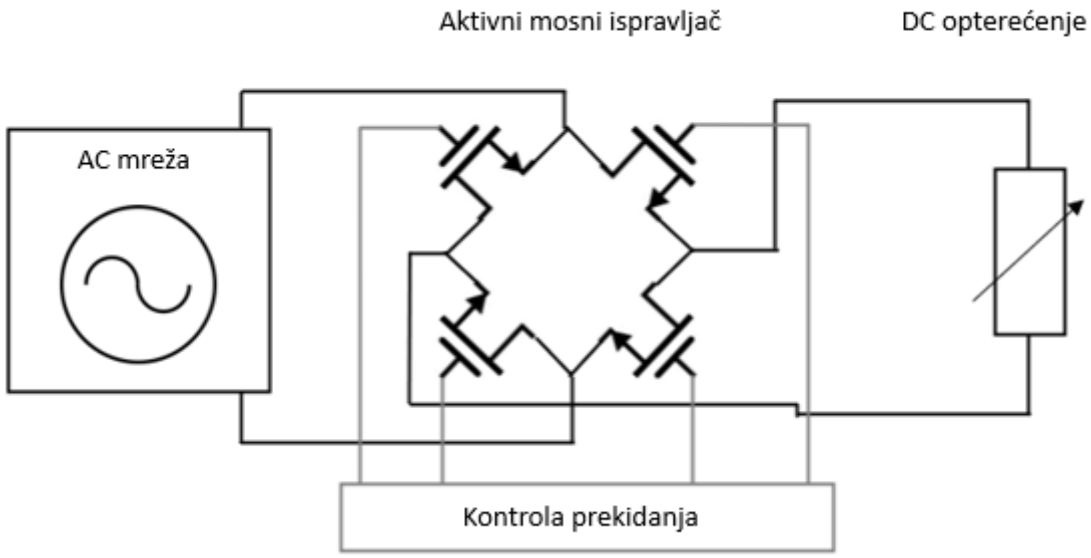
Potpuno ispravljeni val ima viši srednji izlazni napon od onog koji proizvodi polumosni ispravljač, ali je još uvijek vrlo daleko od konstantnog istosmjernog valnog oblika potrebnog za napajanje elektroničkih uređaja. Iako se radi o istosmjernom valu, njegovo korištenje za napajanje uređaja bilo bi neučinkovito zbog oblika naponskog vala koji vrlo brzo i vrlo često mijenja vrijednost. Ova periodična promjena istosmjernog napona naziva se valovitost. Smanjenje ili eliminacija valovitosti ključno je za učinkovito napajanje. Najjednostavnija i

najčešće korištena metoda za smanjenje valovitosti je uporaba velikog kondenzatora na izlazu ispravljača, koji se naziva rezervoarski kondenzator ili filter za izravnavanje. Kondenzator pohranjuje napon tijekom vrhunca vala, a zatim opskrbljuje opterećenje strujom sve dok njegov napon ne bude manji od sada rastućeg ispravljenog naponskog vala. Rezultirajući valni oblik mnogo je bliži željenom obliku i može se smatrati istosmjernim naponom bez izmjenične komponente. Ovaj konačni valni oblik napona sada se može koristiti za napajanje istosmjernih uređaja[26].



Slika 34. Mosni ispravljač s filtrom za izravnavanje[26]

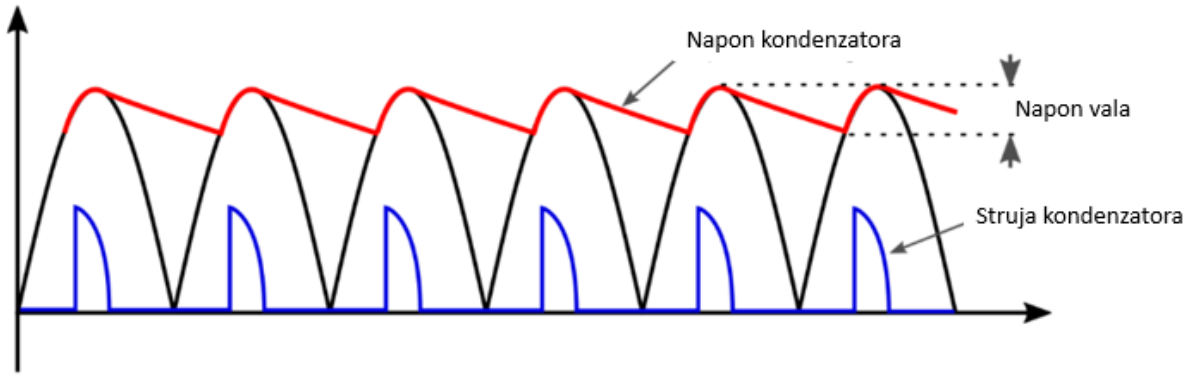
Pasivno ispravljanje koristi poluvodičke diode kao nekontrolirane sklopke i najjednostavnija je metoda za ispravljanje izmjeničnog vala, ali nije najučinkovitija. Diode su relativno učinkovite sklopke, mogu se brzo uključiti i isključiti uz minimalan gubitak snage. Jedini problem s poluvodičkim diodama je taj što imaju pad napona od 0,5 V do 1 V, što smanjuje učinkovitost. Aktivno ispravljanje zamjenjuje diode kontroliranim sklopkama, kao što su MOSFET ili BJT tranzistori. Prednosti su dvojake: Prvo, ispravljači temeljeni na tranzistorima eliminiraju fiksni pad napona od 0,5 V do 1 V povezan s poluvodičkim diodama, jer se njihovi otpori mogu učiniti proizvoljno malima, i zbog toga imaju mali pad napona. Drugo, tranzistori su kontrolirane sklopke, što znači da se frekvencija može kontrolirati i stoga optimizirati. Loša strana je što aktivni ispravljači zahtijevaju kompliciranije upravljačke sklopove, što zahtijeva dodatne komponente i čini ih skupljima[26].



Slika 35. Aktivno ispravljanje[26]

5.5.2. Korekcija faktora snage

Druga faza u dizajnu napajanju s prekidanjem struje je korekcija faktora snage (PFC). PFC sklopovi nemaju mnogo veze sa pretvorbom izmjenične struje u istosmjernu, ali su važna komponenta većine komercijalnih izvora napajanja[26].



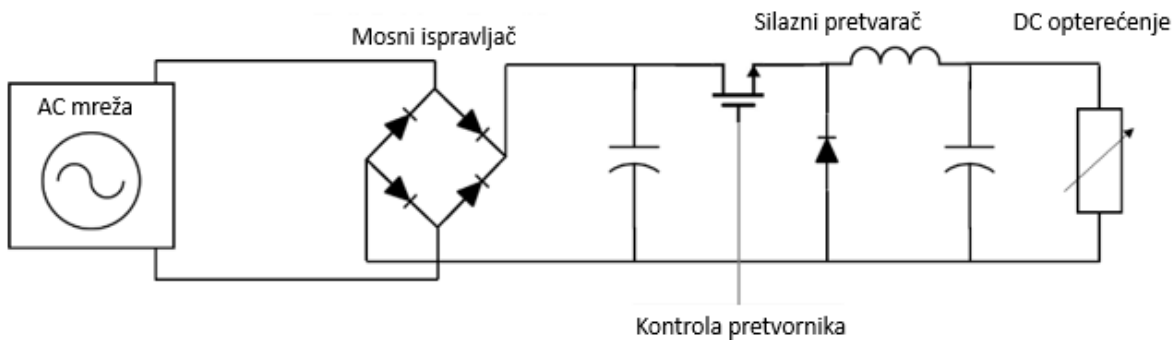
Slika 36. Valni oblici napona i struje na izlazu ispravljača[26]

Promatrajući valni oblik struje kondenzatora ispravljača, struja punjenja teče kroz kondenzator tijekom vrlo kratkog vremenskog raspona, posebno od točke gdje je napon na ulazu kondenzatora veći od naboja kondenzatora do vrha ispravljenog signala. Ovo generira niz kratkih strujnih skokova u kondenzatoru, stvarajući značajan problem ne samo za napajanje, već i za cijelu električnu mrežu zbog velike količine harmonijskih valova. Ti harmonijski valovi mogu generirati izobličenja koja mogu utjecati na druge izvore napajanja i uređaje spojene na mrežu. Cilj sklopovlja za korekciju faktora snage je minimizirati ove harmonijske valove njihovim filtriranjem.

5.5.3. Izolirani naspram neizoliranih napajanja s prekidanjem struje

Bez obzira postoji li PFC sklopovlje ili ne, posljednji korak za pretvorbu snage je smanjivanje ispravljenog istosmjernog napona. Budući da je ulazni AC valni oblik ispravljen na ulazu, izlazni istosmjerni napon će biti visok: ako nema PFC – a, izlazni istosmjerni napon iz ispravljača bit će oko 320 V. Ako postoji aktivan PFC, izlaz će bit stabilan istosmjerni napon od 400 V ili više. Oba su scenarija iznimno opasna i beskorisna za većinu aplikacija koje obično zahtijevaju znatno niže napone[26].

Uređaji koje korisnik ne mora izravno dodirivati, kao što su svjetla, senzori, i drugo, ne trebaju izolaciju koju osigurava transformator, jer se svaka manipulacija parametrima uređaja vrši s zasebnog uređaja, poput mobilnog telefona, tableta, ili računala. To nudi velike prednosti u smislu težine, veličine i izvedbe. Ovi pretvarači smanjuju razine izlaznog napona pomoću visokonaponskog pretvornika koji se također naziva i silazni pretvarač. U ovom slučaju, kada je tranzistorska sklopka zatvorena, struja koja teče kroz induktor stvara napon preko induktora koji se suprotstavlja naponu iz izvora napajanja, smanjujući napon na izlazu. Kada se sklopka otvori, induktor oslobađa struju koja teče kroz opterećenje, održavajući vrijednost napona na opterećenju dok je krug odsječen od izvora napajanja[26].



Slika 37. Neizolirano AC/DC napajanje s aktivnim PFC–om[26]

5.5.4. MEAN WELL LRS 150–15 ispravljač

Za potrebe robotskog crtača koristit će se MEAN WELL LRS–150–12 ispravljač serije LRS–150. Serija LRS–150 je izvor napajanja zatvorenog tipa s jednim izlazom od 150 W i dizajnom niskog profila od 30 mm. Uz ulazni napon od 115 VAC ili 230 VAC, cijela serija pruža izlazni napon od 12 V, 15 V, 24 V, 36 V i 48 V. Uz visoku učinkovitost do 90%, konstrukcija kućišta od perforiranog lima poboljšava rasipanje topline tako da cijela serija radi od -30°C do 70°C pod konvekcijom zraka bez ventilatora[27].

LRS–150–12 je 230 VAC napajanje sa kućištem od perforiranog lima, podesivog napona. Za unutarnju upotrebu, sa dobrim hlađenjem, malih dimenzija te vrlo izdržljiv na udarce. U nastavku slijede tehničke specifikacije:

Tablica 8. Tehničke specifikacije LRS–150–12 ispravljača[27]

DC izlazni napon	12 V
Izlazna nazivna struja	12,5 A
Nazivna snaga	150 W
Raspon podesivog izlaznog napona	10,2~13,8 V
Raspon ulaznog napona	85~132 VAC/170~264 VAC
Raspon frekvencije	47~63 Hz
Učinkovitost	87,5%
Dimenzije	159x97x30 mm

Slika 38. MEAN WELL LRS–150–12 ispravljač[28]

5.6. Servo motor SG90

Servo motori su samostalni električni uređaji koji rotiraju ili guraju dijelove stroja s velikom preciznošću. Servo motor se nalazi na mnogim mjestima: od igračaka preko kućne elektronike do automobila i aviona[29].

Jednostavnost servo uređaja jedna je od značajki koje ih čine tako pouzdanima. Srce servo uređaja je mali istosmjerni DC motor. Ovi motori rade na struju iz napajanja i vrte se pri visokom broju okretaja u minuti, ali daju vrlo nizak okretni moment. Raspored zupčanika preuzima veliku brzinu motora i usporava ga, dok u isto vrijeme povećava okretni moment. Dizajn zupčanika unutar servo kućišta pretvara izlaz u mnogo sporiju brzinu rotacije, ali s više okretnog momenta. Zupčanici u jeftinom servo motoru općenito su izrađeni od plastike kako bi bili lakši i jeftiniji[29]. Servo motori se sastoje od istosmjernog motora, zupčanika, potenciometra za određivanje položaja i male elektroničke upravljačke ploče[30].

Servo motor Micro SG90 se može koristiti u aplikacijama gdje je potrebna mala veličina motora i gdje nije potreban veliki okretni moment. Zupčanici su najlonski, što je slučaj s većinom jeftinijih servo motora[30].

Slika 39. Servo motor[31]

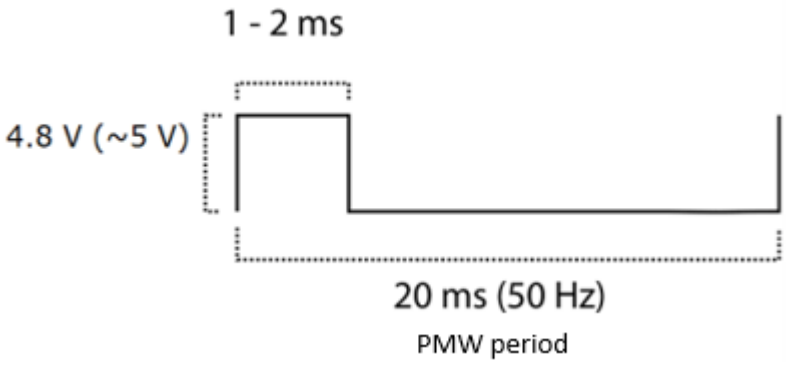
Standardni servo uređaji imaju određeni ograničeni raspon zakreta. To se obično navodi kao 180 stupnjeva. No često je stvarni raspon manji od punih 180 stupnjeva i ograničen je mehaničkim zupčanicima i potenciometrom koji se koristi za očitavanje položaja. Ako se motor pokrene sve do 0 ili 180, mogao bi ispuštati neugodne zvukove i početi vibrirati dok pokušava u doći položaj do kojeg ne može. To može uzrokovati oštećenje zupčanika i motora, tako da ga

je najbolje koristiti na smanjenom rasponu kao što je 20-160 stupnjeva ili malo eksperimentirati kako bi se odredio stvarni korisni raspon ako se želi maksimizirati raspon[30].

Iz ovog motora izlaze tri žice. Da bi se ovaj motor okretao, mora se napajati s +5 V pomoću crvene i smeđe žice (koja služi za uzemljenje) i poslati PWM signale na narančastu žicu. Arduino će generirati PWM signale kako bi ovaj motor radio[32].

Slika 40. Značenje svake žice[32]

Servo motori očekuju da vide puls na svom PWM priključku svakih 20 ms. Puls je aktivan na 5 V, a širina impulsa određuje položaj (kut) osovine servo motora. Puls može varirati između 1 ms i 2 ms. Impuls od 1 ms pozicionira osovinu na 0 stupnjeva. Impuls od 1,5 ms pozicionira osovinu na 90 stupnjeva. Impuls od 2 ms postavlja osovinu na 180 stupnjeva. Impulsi s vrijednostima između ovih mogu se koristiti za proizvoljno pozicioniranje osovine[30].



Slika 41. PMW puls servo motora[32]

Servo motor SG90 služi za primak i odmak olovke od papira. S obzirom na njegovu kompaktnost i dimenzije, on je najprikladniji izbor za izvod takve radnje s obzirom da se mora nalaziti u nosaču olovke. U nastavku slijede tehnički podaci motora:

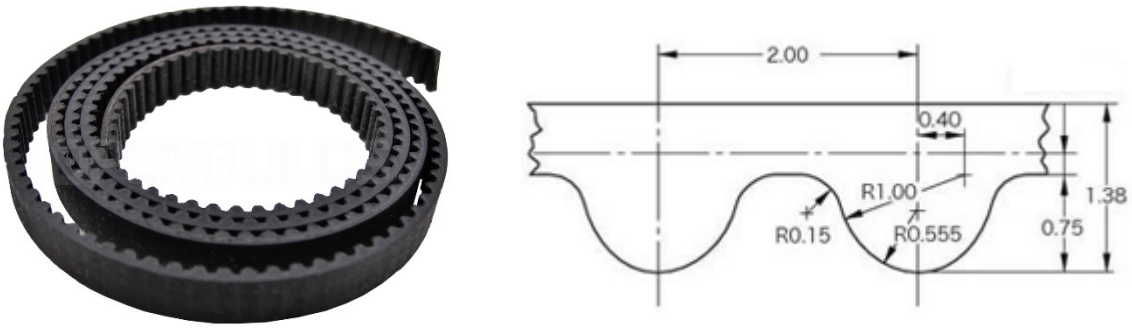
Tablica 9. Tehnički podaci servo motora SG90[30]

Radni napon	+5 V
Okretni moment	2,5 kg/cm
Brzina	0,1 s/60°
Raspon zakreta	0°-180°
Masa motora bez kabela	9 g
Struja (mirovanje)	10 mA
Struja (prilikom kretnje)	100–250 mA
Dimenzije kućišta motora	23x12x26 mm
Visina motora (sa osovinom)	32 mm

5.7. Ostale komponente

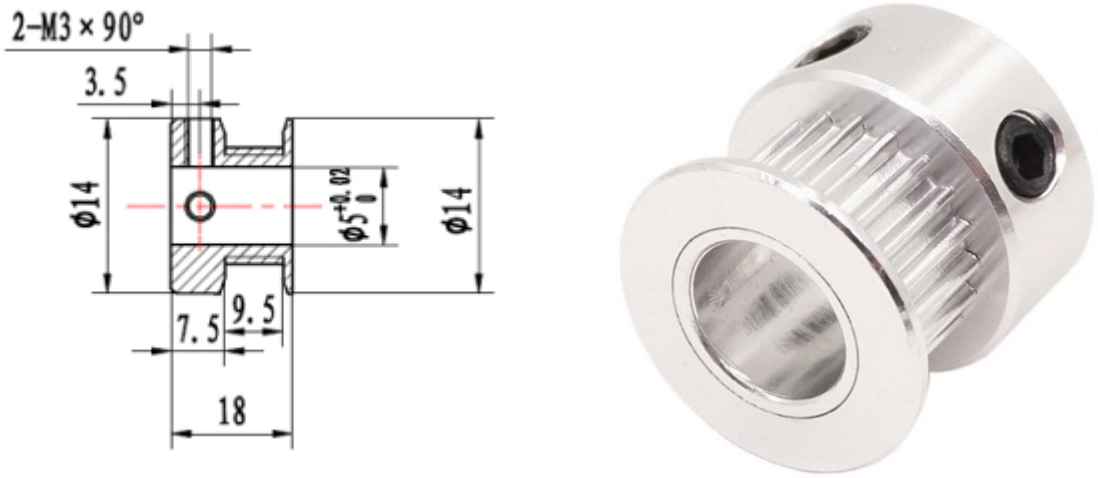
Ostale komponente robotskog crtača su protutezi, olovka, dvije remenice, dva remena, vijci i matice. Remenice služe za prijenos snage na remene, a remeni na nosač olovke. Protutezi služe za zatezanje remena jer bez zatezanja nije moguće prenijeti snagu.

GT2 zupčasti remen služi za prijenos rotacijskog gibanja (s koračnog motora) u linearno gibanje. Imaju poseban profil sa zaobljenim zubima koji smanjuju zračnost. Često se koristi za precizne 3D pisače i CNC strojeve. Širok je 6 mm a korak je 2 mm. Duljina remena varira ovisno o primjeni, a kod robotskog crtača duljina pojedinačnog remena je 1500 mm[33].



Slika 42. GT2 remen[33][34]

GT2 (16 zubi) remenice su snažne, ali lagane, zahvaljujući aluminijskom materijalu. Ima unutarnji promjer od 5 mm i kompatibilan je sa GT2 zupčastim remenom širine 6 mm te se koristi za glatko vođenje zupčastog remena.



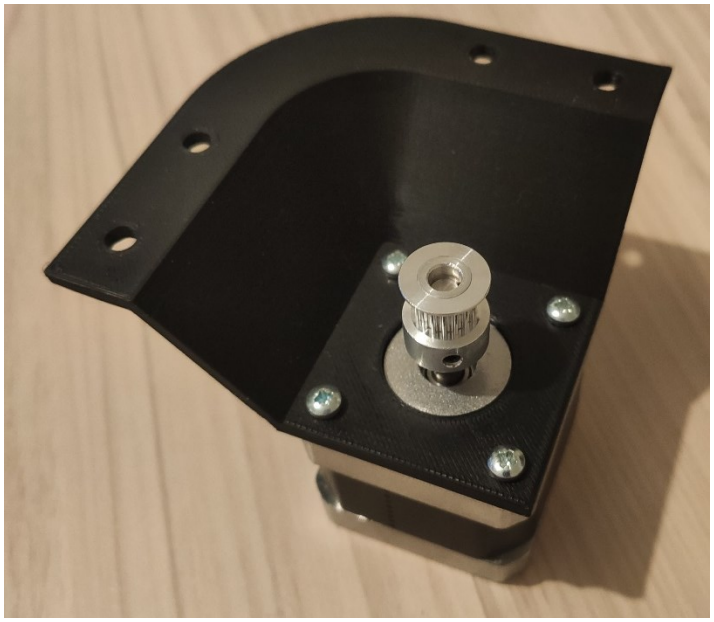
Slika 43. GT2 remenica sa 16 zubi[35][36]

6. MONTAŽA ROBOTSKOG CRTAČA ZA ISPIS FOTOGRAFIJA U VERTIKALNOJ RAVNINI

Nakon konstruiranja nosača motora, olovke i spojki te odabira raznovrsnih standardiziranih komponenti, potrebno je montirati robotski crtač za ispis fotografija. Cjelokupni sklop se može podijeliti na nekoliko podsklopova: Sklop motora i ploče, sklop nosača za olovku te sklop elektronskih komponenti.

6.1. Sklop koračnih motora i ploče

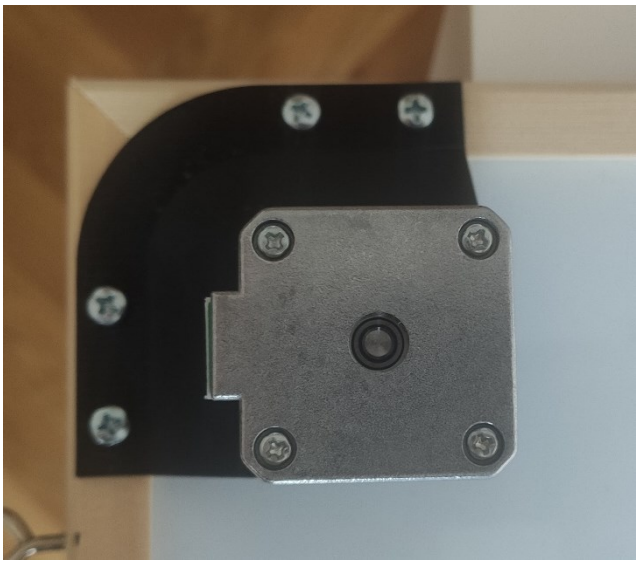
Prvo, potrebno je sklopiti remenicu sa motorom. GT2 remenica ima dvije navojne rupe M3 u razmaku od 90° koje služe za pritezanje remenice na izlazno tzv. „D“ vratilo koračnog motora. Za pritezanje se koriste utični vijci s upuštenom glavom M3 x 4 mm prema DIN 913 standardu. Kako je opisano u trećem poglavlju, nosači motora imaju nekoliko funkcionalnih provrta. Kako bi se koračni motor spojio sa nosačem, mora se ostvariti dosjed između središnje rupe nosača i osovine motora. Dosjed je zračni kako bi se osigurala ručna montaža i kako ne bi došlo do pucanja nosača pri sili uprešavanja. Nadalje, koračni motori 17HS3401 imaju četiri navojne rupe M3 dubine 4 mm. Kako bi se ostvario vijčani spoj, koriste se četiri križna vijka M3 duljine 6 mm (DIN 7986) za pozicioniranje i ostvarivanje spoja između nosača i motora.



Slika 44. Spoj nosača i koračnog motora

Ploča na koju se montira robotski crtač je 800 x 600 mm. S obzirom na drveni okvir, jednostavno je probušiti rupe za spajanje sa nosačima motora. Za montažu potrebno je: 8 vijaka, 8 matica te 8 podložnih pločica. U ploču za crtanje je probušeno 8 provrta ϕ4 mm. Debljina okvira je 15 mm i debljina stijenke nosača je 3 mm, stoga su uzeti vijci DIN 7985 M4 x 25 mm.

Odabrane matice su DIN 934 M4, dok su za podložne pločice odabrani elastični prsteni DIN 127 M4. Elastični prsteni uzrokuju uzdužno prednaprezanje silom te tako sprječavaju okretanje matice.

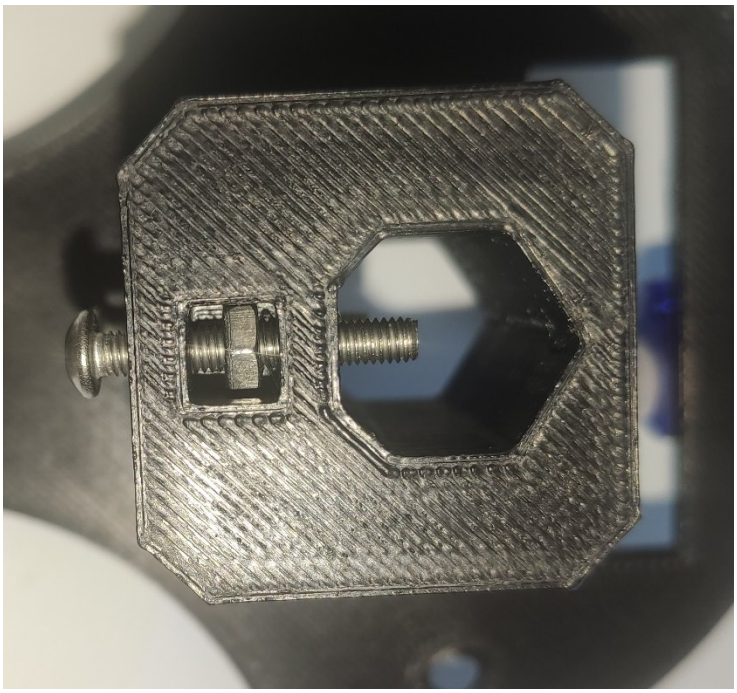


Slika 45. Spoj ploče i nosača motora

6.2. Sklop nosača za olovku

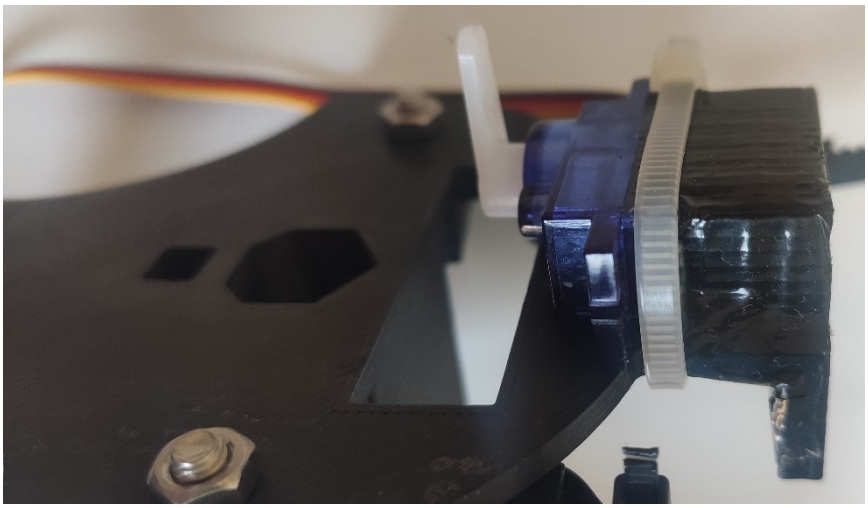
Nosač za olovku ima nekoliko funkcija koja treba ispuniti. Najprije, mora držati olovku. Drugo, mora se povezati sa ostatkom robotskog crtača za ispis fotografija. Treće, mora pridržavati SG90 servo motor koji služi za podizanje i spuštanje olovke. Nakraju, mora osigurati dovoljnu težinu kako bi se remeni zategli za ispravan rad robota.

Držanje olovke se osigurava pomoću dva vijka sa šesterokutnom upuštenom glavom DIN 9427 M3 x 20 mm i dvije matice DIN 934 M3. Matice se ubacuju u utore koji imaju oblik šesterokuta koji odgovara maticama. Pri okretanju vijka matice se počinju gibati prema zidovima utora. Kad se naslone na zid, šesterokutni oblik utora onemogućava gibanje matice. Tada počinje gibanje vijaka. Kada vijci dotaknu olovku, započinje pritezanje olovke.



Slika 46. Vijci za pritezanje olovke

SG90 servo motor se spojio pomoću samoljepljive trake i elektro vezica. Na izlazno zupčasto vratilo motora se postavlja ručica koja omogućuje odmak i primak olovke. Remeni se pomoću elektro vezica postavljaju na spojke, a one se pomoću vijaka DIN 84 A M4 x 20 mm sa cilindričnom glavom i urezom i matice DIN 934 M4 spajaju sa ostatkom nosača. Uteg je sastavljen od pločice 38 x 25 x 2 mm, šesterokutnog vijka DIN 933 M8 x 65 mm, sedam matica DIN 934 M8 i jednom DIN 934 M10 maticom. Uteg je sa DIN 912 vijkom sa šesterokutnom upuštenom glavom M4 x 15 mm i maticom DIN 934 M4 vijčano spojen sa nosačem olovke.



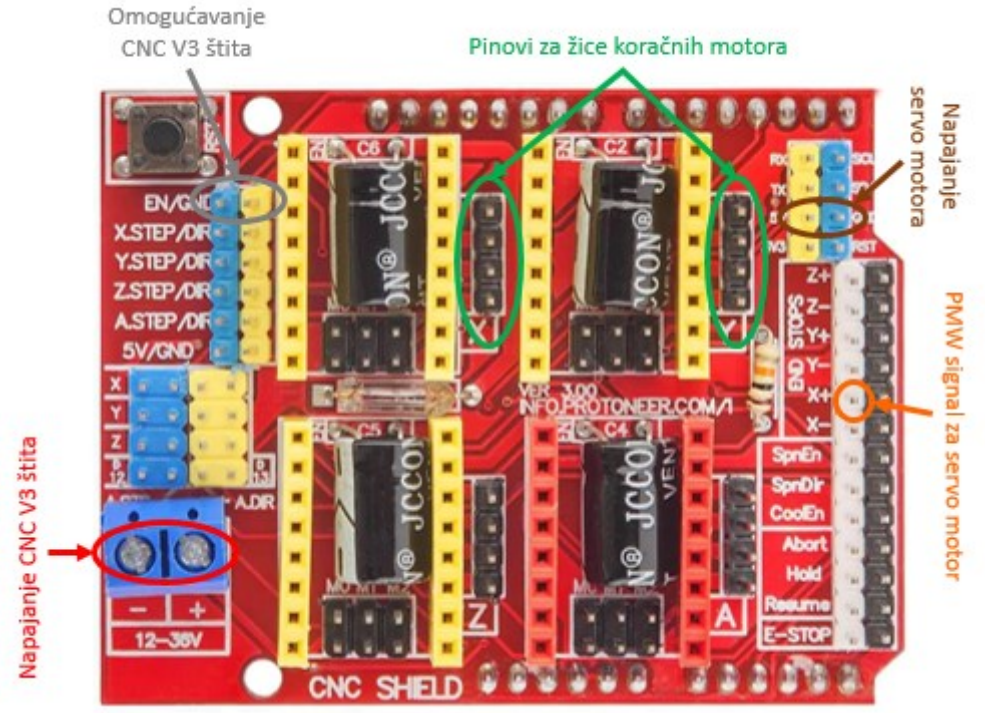
Slika 47. Spajanje SG90 servo motora sa nosačem olovke

Slika 48. Podsklop nosača olovke

6.3. Elektronske komponente

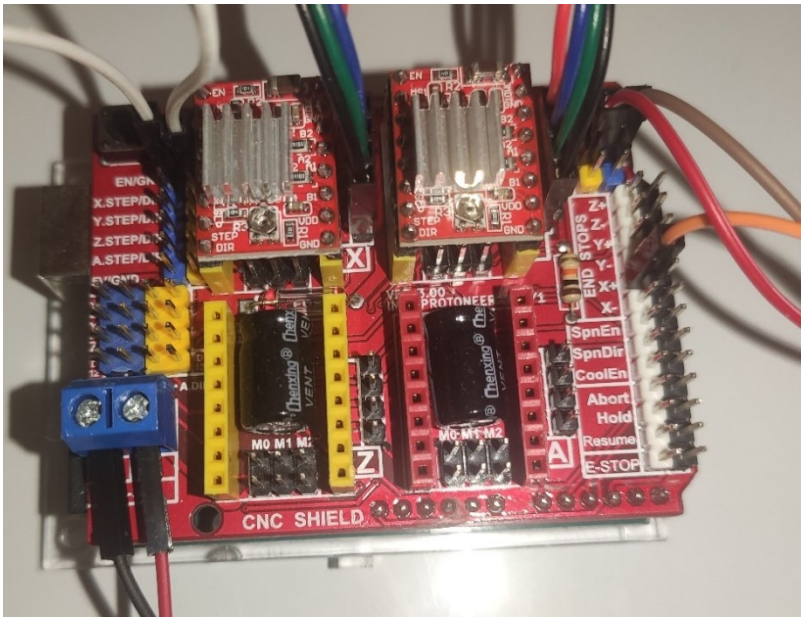
Elektronski dio robota crtača se sastoji od: Arduino Uno mikroupravljačke ploče, CNC V3 štita, dva A4988 upravljača koračnih motora, ispravljača MEAN WELL LRS150-12, SG90 servo motora te brojnih žica.

CNC V3 štit se postavlja na Arduino Uno. Poslije toga, A4988 upravljači se stavljaju u odgovarajuće utore u CNC V3 štit. Jedan upravljač se postavlja na X os, dok se drugi postavlja na Y os. Pri postavljanju mora se osigurati da EN priključak na upravljaču se postavi u EN utor na CNC V3 štitu. U suprotnom ako se orijentacija upravljača zamjeni, može doći do uništavanja upravljača. Sada je potrebno pripremiti žice koje se postavljaju na CNC V3 štit radi lakšeg postavljanja robota pri svakom novom korištenju.



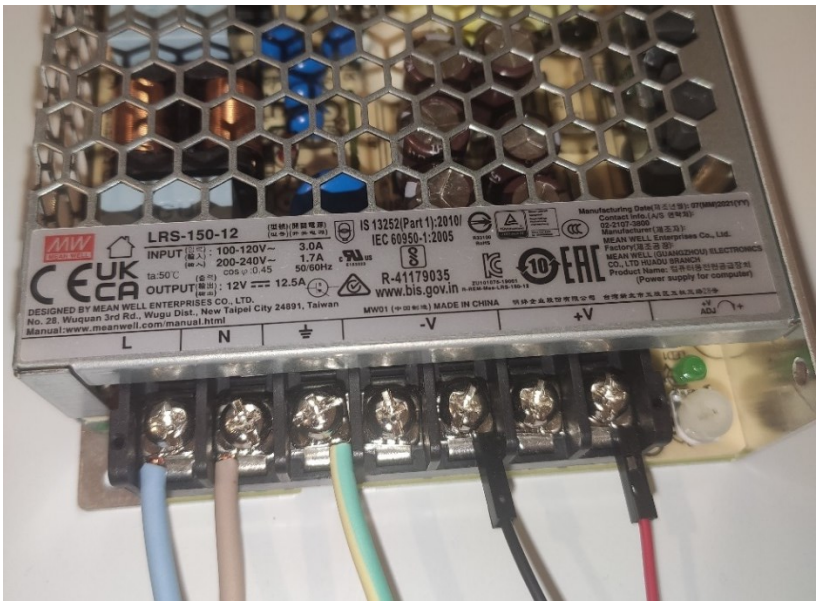
Slika 49. Shema žica[37]

Najprije potrebno je spojiti žice za napajanje CNC V3 štita. Preko ispravljača se šalje struja napona 12 V na štit koja služi za napajanje koračnih motora. Na priključke označene zelenom bojom se stavljaju žice koje se spajaju na koračne motore. Prilikom spajanja potrebno je pripaziti na orijentaciju žica. Kriva orijentacija neće napraviti kvar, ali će utjecati na smjer gibanja motora. Također potrebno je staviti kratkospojnik na priključke EN i GND kako bi se omogućio rad štita. Nakraju, spaja se servo motor na štit. Na priključke označene smeđom bojom se stavlja napajanje servo motora, a na priključak X+ se stavlja žica za PMW signal servo motora. Ovi definirani priključci odgovaraju priključcima na Arduinu, što je potrebno znati pri programiranju softvera. Pa tako priključci za korak i smjer na X osi odgovaraju priključcima 2 i 5, a priključci za korak i smjer na Y osi priključcima 3 i 6. Priključak za servo motor X+ odgovara priključku 9 na Arduinu.



Slika 50. Ožičenje CNC V3 štita

Ispravljač se mora spojiti preko mrežnog kabela na gradsku mrežu. Gradska mreža ima napon od 220 V i frekvenciju 50 Hz. Nul – vodič i fazni vodič mrežnog kabela se priključuje na L i N priključnice ispravljača. S obzirom da se ovdje rukuje s naponima većim od 220 V te je kućište ispravljača metalno, kao mjera zaštite od strujnog udara, u monofaznim mrežama postoji tzv. zaštitni vodič, koji je uvijek žuto – zelene boje. Taj vodič se spaja na priključak ispravljača koji ima ikonicu za uzemljenje. Nul – vodič i fazni vodič nije bitno kako se spajaju na L i N priključke. Nakon spajanja mrežnog kabela sa ispravljačem, potrebno je povezati ispravljač sa CNC V3 štitom kako bi mogao isporučiti istosmjernu struju 12 V do koračnih motora.



Slika 51. Ožičenje ispravljača

6.4. Podešavanje struje koračnih motora

A4988 upravljači reguliraju struju koja dolazi iz ispravljača 12,5 A te je spuštaju na nižu vrijednost. Upravljač A4988 može isporučiti maksimalno 2 A struje po fazi, iako je preporuka da se upotrebljava 1 A po fazi, ako nije omogućeno dodatno hlađenje pomoću ventilatora ili toplinskog odvoda. U tehničkim specifikacijama koračnog motora 17HS3401 najbolja učinkovitost motora (brzina i moment) se dobiva ako se motor pogoni sa njegovom nazivnom strujom od 1,3 A. No ako upravljač predaje više od 1,3 A motoru, to može oštetiti motor i upravljač. Zato mora postojati način na koji se regulira ulazna struja koračnog motora. A4988 upravljač ima potenciometar preko kojeg se može upravljati referentnim naponom. Referentni napon VREF odgovara maksimalnoj struji koja će teći do koračnog motora. A4988 omogućuje da se postavi ciljana struja bilo gdje između nekoliko mA do 2 A. Okretanjem potenciometra u smjeru kazaljke na satu, VREF napon će se povećavati, a smanjivati kada se okreće u smjeru suprotnom od kazaljke na satu. Stvarna vrijednost VREF može se izračunati pomoću formule:

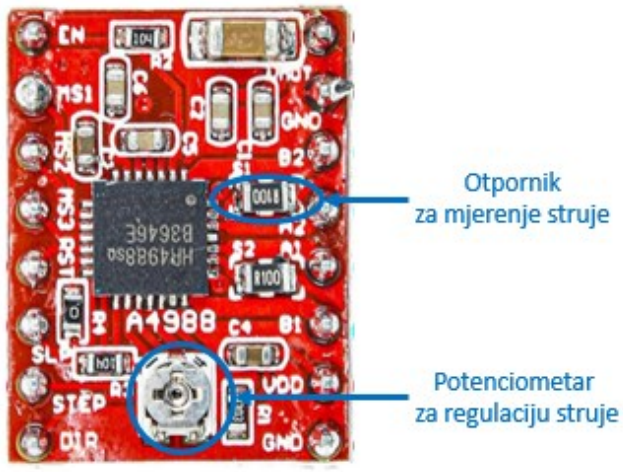
$$V_{\text{ref}} = I_{\text{max}} \cdot 8 \cdot R_{\text{s}} \tag{1}$$

Gdje je:

V_{ref}	V	referentni napon
I_{max}	A	maksimalna struja po fazi
R_{s}	Ω	otpor za mjerenje struje

R_{s} je otpor za mjerenje struje i ta vrijednost ovisi o proizvođaču ploče A4988. Vrijednost R_{s} se očitava na otporniku upravljača A1988. Otpornik je R100, što odgovara vrijednosti od 0,1 Ω. Nakon ubacivanja podataka u jednadžbu (1) dobiva se vrijednost referentnog napona od:

$$V_{\text{ref}} = 1{,}3\ \text{A} \cdot 8 \cdot 0{,}1\ \Omega = 1{,}04\ \text{V} \tag{2}$$

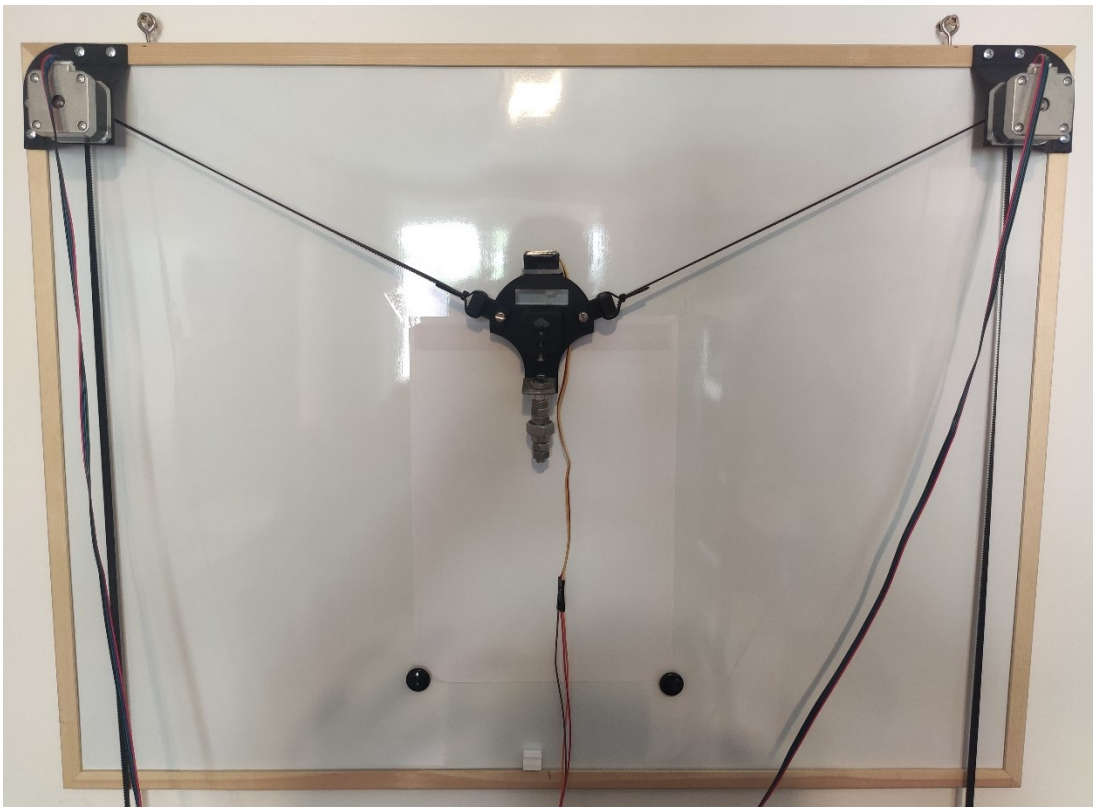


Slika 52. Potenciometar na A4988 upravljaču[38]

Za sigurnost uzima se 90% izračunate vrijednosti što sada iznosi 0,936 V. Pomoću multimetra se izmjeri referentni napon na vrhu glave potenciometra i bilo koje točke uzemljenja na ploči te se pomoću križnog odvijača podesi željena vrijednost.

6.5. Krajnja verzija robotskog crtača za ispis fotografija u vertikalnoj ravnini

Nakon montaže svih podsklopova robotskog crtača, vrijeme je za krajnju montažu robota. Ploča se objesi za zid, remeni se prebace preko remenica, žice za koračne motore se spoje na motore te se žice za servo motor spoje. Konačna verzija robotskog crtača za ispis fotografija u vertikalnoj razini izgleda ovako:



Slika 53. Robotski crtač za ispis fotografija u vertikalnoj ravnini

7. ARDUINO IDE

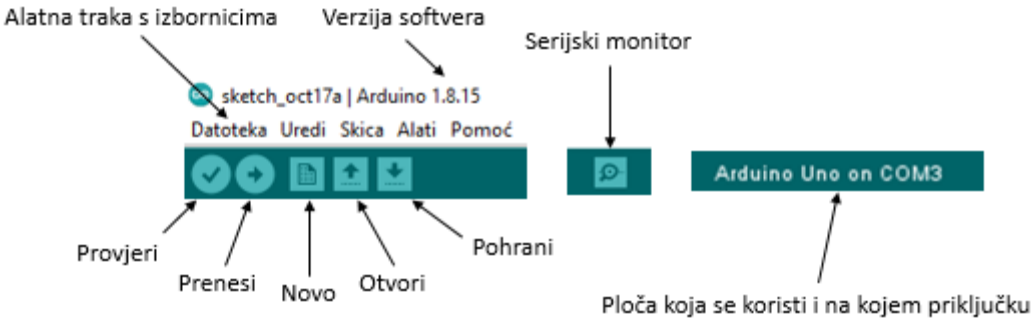
Arduino kod je napisan u C++ programskom jeziku s dodatkom posebnih metoda i funkcija. C++ je čovjeku čitljiv programski jezik. Kada se stvori skica (naziv dan Arduino programskim datotekama), ona se obrađuje i kompajlira u strojni jezik[39].

Arduino IDE je glavni program za uređivanje teksta koji se koristi za Arduino programiranje. To je mjesto gdje će se upisivati kod prije nego što se postavi na mikroupravljačku ploču[39].

Arduino integrirano razvojno okruženje ili Arduino softver (IDE) sadrži uređivač teksta za pisanje koda, područje za poruke, tekstualnu konzolu, alatnu traku s gumbima za uobičajene funkcije i niz izbornika. Arduino IDE se povezuje na Arduino hardver kako bi učitao program i komunicirao s njim[40].

7.1. Arduino IDE sučelje

Programi napisani pomoću Arduino softvera (IDE) nazivaju se skice. Ove skice su napisane u uređivaču teksta i spremaju se s ekstenzijom datoteke .ino. Uređivač ima značajke za rezanje/lijepljenje i za pretraživanje/zamjenu teksta. Područje za poruke daje povratne informacije tijekom spremanja i izvoza programa na hardver i prikazuje pogreške. Konzola prikazuje tekstualni izlaz Arduino softvera (IDE), uključujući poruke o pogreškama i drugim informacijama. Donji desni kut prozora prikazuje ploču koja se koristi i na kojem serijskom priključku. Gumbi na alatnoj traci omogućuju provjeru i učitavanje programa, stvaranje, otvaranje i spremanje skica te otvaranje serijskog monitora. Dodatne naredbe se nalaze unutar pet izbornika: Datoteka, Uređivanje, Skica, Alati i Pomoć[40].



Slika 54. Sučelje Arduino IDE – a

7.2. Struktura koda

7.2.1. Knjižnice

Knjižnice pružaju dodatne funkcionalnosti, npr. rad s hardverom (primjerice postoje knjižnice za olakšano upravljanje servo i koračnim motorima) ili manipuliranje podacima. Za korištenje knjižnica u skici, knjižnica se poziva iz izbornika Skica > Include Library. Ovo će umetnuti jednu ili više naredbi #include na vrh skice i kompajlirati knjižnicu s skicom. Budući da se knjižnice učitavaju na ploču sa skicom, one povećavaju količinu prostora koje zauzimaju. Ako skica više ne treba knjižnicu, jednostavno se izbrišu njezine naredbe #include s vrha koda[40].

U referencama se nalazi popis knjižnica. Neke su knižnice uključene u softver Arduino, dok se ostale mogu preuzeti sa interneta[40].

7.2.2. Definiranje priključaka

Za korištenje Arduino priključaka, potrebno je definirati koji se priključak koristi i njegovu funkcionalnost. Prikladan način za definiranje korištenih priključaka je korištenjem #define pinName pinNumber. Funkcionalnost je ulazna ili izlazna i definirana je pomoću metode pinMode () u odjeljku za postavljanje.

7.2.3. Deklaracija instanci i varijabli

Prilikom korištenja Arduina, potrebno je deklarirati globalne varijable koje će se koristiti kasnije. Ukratko, varijabla omogućuje imenovanje i pohranjivanje vrijednosti koja će se koristiti u budućnosti. Na primjer, pohranjivanje podataka prikupljenih sa senzora kako bi se kasnije mogli koristiti. Za deklariranje varijable jednostavno se definira njen tip, naziv i početna vrijednost.

U softverskom programiranju, klasa je zbirka funkcija i varijabli koje se drže zajedno na jednom mjestu. Svaka klasa ima posebnu funkciju poznatu kao konstruktor, koja se koristi za stvaranje instance klase. Kako bi se koristile funkcije klase, mora se deklarirati instanca za nju.

Svaka Arduino skica mora imati funkciju setup. Ova funkcija definira početno stanje Arduina prilikom pokretanja i pokreće se samo jednom. Ovdje se definira sljedeće:

1. funkcionalnost priključka pomoću funkcije pinMode
2. početno stanje priključaka
3. inicijalizacija klasa
4. inicijalizacija varijabli

Funkcija loop također je obavezna za svaku Arduino skicu i izvršava se kada se setup() funkcija završi. To je glavna funkcija i kao što joj ime kaže, radi u petlji uvijek iznova. Petlja opisuje glavnu logiku sklopa.

7.3. Učitavanje

Prije učitavanja skice, treba se odabrati ispravna ploča i priključak na koji se spaja na računalo iz izbornika Alati > Ploča i Alati > Port. U slučaju ovog rada, to će biti priključak COM3 u sustavu Windows. Nakon što se odabrao ispravan USB priključak i ploča, pritisne se gumb za prijenos na alatnoj traci. Trenutačna Arduino ploča automatski će se resetirati i započeti učitavanje. Na većini ploča LED diode RX i TX trepere dok se skica učitava. Arduino IDE će prikazati poruku kada je učitavanje završeno ili prikazati pogrešku[40].

Kada se skica učita, koristi se Arduino bootloader, mali program koji je učitan na mikrokontroler na ploči. On omogućuje učitavanje koda bez korištenja dodatnog hardvera. Bootloader je aktivan nekoliko sekundi kada se ploča resetira; zatim počinje ona skica koja je posljednja učitana u mikroupravljač. Bootloader će treperiti LED na ploči (priključak 13) kada se pokrene (tj. kada se ploča resetira)[40].

7.4. Processing

Processing je besplatna grafička knjižnica i integrirano razvojno okruženje (IDE) izgrađeno za elektroničku umjetnost, vizualni dizajn i grafičko korisničko sučelje. Processing Development Environment (PDE) sastoji se od jednostavnog uređivača teksta za pisanje koda, područja za poruke, tekstualne konzole, kartica za upravljanje datotekama, alatne trake s gumbima za uobičajene akcije i niza izbornika[41].

Programi napisani korištenjem Processing programskog jezika nazivaju se skice i pokreću pritiskom na gumb Run. Skice su napisane u uređivaču teksta. Područje s porukama daje povratne informacije tijekom spremanja i izvoza te također prikazuje pogreške. Konzola prikazuje izlaz teksta Processing skica uključujući poruke o pogreškama (Konzola radi dobro za povremene poruke, ali nije namijenjena za brzi izlaz u stvarnom vremenu.). Dodatne naredbe nalaze se unutar šest izbornika: File, Edit, Sketch, Debug, Tools i Help[41].

Kada se pogledaju sama sučelja Processing Development Environment–a i Arduino IDE–a, može se zaključiti da su jako slična. To je zato što je PDE zapravo bio preteča Arduino IDE–u kao i mnogim drugim razvojnim okruženjima.

Mogućnosti Processinga su proširene različitim knjižnicama i alatima. Knjižnice omogućuju skicama da rade stvari izvan osnovnog koda za obradu. Postoje stotine knjižnica

koje je doprinijela Processing zajednica i koje se mogu dodati skicama kako bi se omogućile nove stvari poput reprodukcije zvukova, računalnog vida i rada s naprednom 3D geometrijom. Alati proširuju PDE kako bi olakšali stvaranje skica pružajući sučelja za zadatke poput odabira boja[41].

Processing ima različite načine programiranja kako bi omogućila implementacija skica na različite platforme i programiranje na različite načine. Zadani programski jezik je Java. Ostali načini programiranja mogu se preuzeti odabirom „Add Mode...“ iz izbornika u gornjem desnom kutu PDE–a[41].

8. GUI ROBOTSKOG CRTAČA

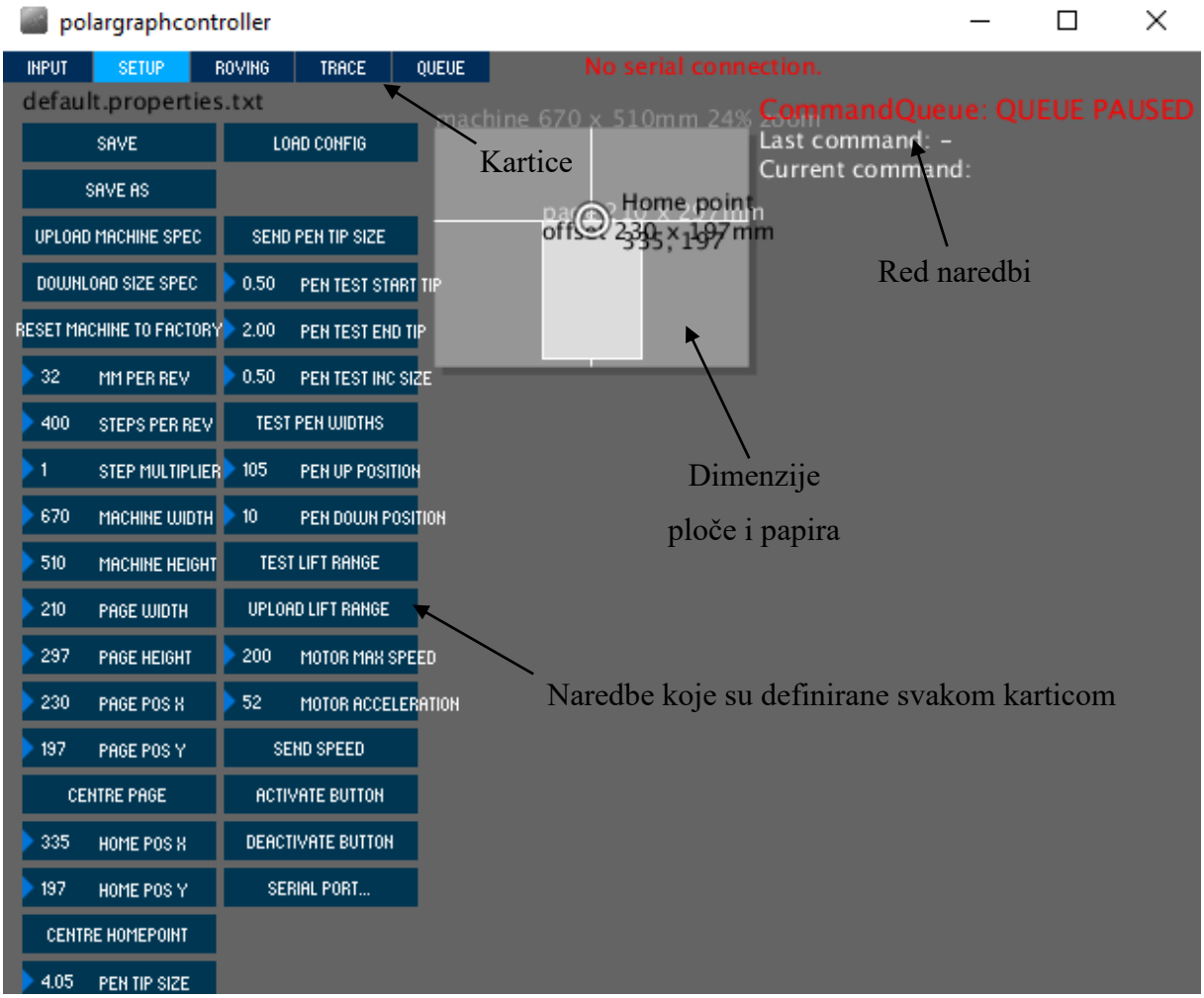
Grafičko korisničko sučelje (GUI) vrsta je korisničkog sučelja putem kojeg korisnici komuniciraju s elektroničkim uređajima putem vizualnih indikatora. Načela dizajna grafičkog korisničkog sučelja u skladu su sa softverskim uzorkom model–pogled–upravljač, koji odvaja interne prikaze informacija od načina na koji se informacije prezentiraju korisniku, što rezultira sučeljem koje pokazuje korisnicima koje su funkcije moguće, a ne zahtijeva unos kodova naredbi[42].

GUI robotskog crtača napisan je u Processing integriranom razvojnom okruženju. Njegova struktura kao i sam kod može se pronaći na GitHub–u. Originalni autor je Sandy Noble te se GUI može redistribuirati i/ili modificirati pod uvjetima GNU opće javne licence. U ovom radu će se opisati izgled i funkcije sučelja kao i načelo komuniciranja sa Arduino upravljačem, no neće se ulaziti u samu strukturu koda s obzirom da se GUI sastoji od nekoliko tisuća linija koda.

8.1. Izgled sučelja robotskog crtača

GUI se pokreće preko Processing programskog jezika. Nakon paljenja sučelja pojavit će se okno GUI–a. Okno upravljača se sastoji od četiri elementa:

1. Kartice koje služe za različite funkcije upravljača. Kartice su: INPUT, SETUP, ROVING, TRACE i QUEUE.
2. Svaka kartica ima svoje naredbe odnosno funkcije koje se nalaze s lijeve strane prozora GUI–a.
3. Sivi pravokutnik koji definira širinu i visinu samog robotskog crtača. Pomicanjem miša po njemu se pojavljuje se pokazivač koji predstavlja vrh alata.
4. Prikaz reda naredbi u gornjem desnom kutu prozora. Red naredbi je zaustavljen prilikom pokretanja upravljača. Zaglavlje je obojano crvenom bojom. Pritiskom miša na zaglavlje (postaje zelene boje) red naredbi se pokreće. Red naredbi mora biti pokrenut kako bi robotski crtač radio.



Slika 55. Sučelje robotskog crtača

Kartica INPUT ima mnoštvo različitih funkcija. Funkcije koje se koriste u ovom radu su: Pohranjivanje informacija o robotskom crtaču (SAVE i SAVE AS), učitavanje informacija o robotskom crtaču (LOAD CONFIG), brisanje naredbi iz reda naredbi (CLEAR QUEUE), postavljanje nulte pozicije olovke (SET HOME), povratak olovke u nultu poziciju (RETURN TO HOME), podizanje olovke (PEN LIFT), spuštanje olovke olovke (PEN DROP), postavljanje okvira u kojem se odvija crtanje (SELECT AREA i SET FRAME TO AREA) pomak olovke u točku (MOVE PEN TO POINT), učitavanje vektorske slike (LOAD VECTOR), pomicanje slike po papiru (MOVE VECTOR), skaliranje slike (RESIZE VECTOR), veličina vektora (POLYGONIZER LENGTH) te naredba za crtanje (DRAW VECTOR).

Kartica SETUP služi za unos informacija o robotskom crtaču. Uz pohranjivanje i učitavanje informacija o robotskom crtaču, nudi opcije upisivanja informacija koje se prenose dalje u Arduino kod koji služi za izvršavanje naredbi koje se prenose na robotskog crtača. U ovoj kartici se unose vrijednosti poput: Pomak u milimetrima po okretaju (MM PER REV), broj

koraka po okretaju (STEPS PER REV), faktor za izračun mikrokoraka (STEP MULTIPLIER), širina robotskog crtača (MACHINE WIDTH), visina robotskog crtača (MACHINE HEIGHT), širina papira (PAGE WIDTH), visina papira (PAGE HEIGHT), koordinate papira u odnosu na ploču (PAGE POS X, PAGE POS Y), funkcija za centriranje papira po X koordinati (CENTRE PAGE), koordinate nulte pozicije olovke (HOME POS X, HOME POS Y), centriranje nulte pozicije (CENTRE HOMEPOINT), debljina vrha olovke (PEN TIP SIZE), stupanj zakreta vratila servo motora za podizanje i spuštanje olovke (PEN UP POSITION, PEN DOWN POSITION), maksimalna brzina koračnih motora (MOTOR MAX SPEED) i ubrzanje koračnih motora (MOTOR ACCELERATION). U ovoj kartici se nalaze još neke funkcije poput: Testiranja podizanja i spuštanja olovke (TEST LIFT RANGE), odabira serijskog priključka kojim je računalo spojeno sa Arduinom (SERIAL PORT…) te učitavanje svih upisanih vrijednosti u Arduino (UPLOAD MACHINE SPEC).

Kartica TRACE služi za direktno učitavanje slika različitih formata u vektorsku grafiku i njihovo manipuliranje. Naredba BLUR služi za izglađivanje obrisa. Naredba SIMPLIFY za pojednostavljivanje slike. Naredba POSTERISE kontrolira na koliko je različitih boja slika smanjena. Veći broj znači više slojeva. Nakon usklađivanja ovih naredbi za crtanje se koriste naredbe CAPTURE i DRAW CAPTURE.

Kartice ROVING I QUEUE se ne upotrebljavaju u ovom radu. KARTICA ROVING služi za ispisivanje teksta , dok kartica QUEUE uvoz i izvoz reda naredbi u tekstualnu datoteku.

8.2. Komunikacija s Arduino upravljačem

Naredbe se šalju serijski, kao ASCII kod. ASCII (Američki standardni kod za razmjenu informacija) je najčešći format kodiranja znakova za tekstualne podatke u računalima i na internetu. U standardnim ASCII kodiranim podacima postoje jedinstvene vrijednosti za 128 abecednih, numeričkih ili posebnih dodatnih znakova i kontrolnih kodova[43]. Naredbe imaju oblik: <naredba>,<parametar1>,<parametar2>,<parametar3>,<parametar4>,END, gdje se stavke u uglatim zagradama mogu, ali i ne moraju pojaviti. Svaka naredba završava znakom ASCII 10 (novi red, \n ili 0x0A). Klijent bi trebao pričekati dok ne primi poruku READY prije slanja naredbe. Udaljenosti su navedene u koracima osim ako nije drugačije navedeno[44]. Naredbe postavljanja robotskog crtača se pohranjuju u EEPROM Arduina. Naredbe koje će se koristiti u ovom radu su:

1. Promjena duljine: C01,<lijeva ciljna duljina>,<desna ciljna duljina>,END pomiče motore maksimalnom brzinom dok remeni robotskog crtača ne budu na ciljnoj udaljenosti od motora, mjereno u koracima motora[44].

2. Promjena širine vrha olovke: C02,<veličina vrha olovke>,END postavlja novu širinu olovke. Mjerna jedinica je ovdje milimetar. Interno se čita kao float[44].
3. Postavljanje položaja olovke: C09,<lijeva udaljenost>,<desna udaljenost>,END govori uređaju gdje se olovka već nalazi. Ovo se prvenstveno koristi tijekom navođenja. Olovka se postavlja na određeni, unaprijed izmjereni položaj, a zatim funkcija SET HOME šalje naredbu C09 u Arduino[44].
4. Olovka dolje: C13,<pozicija stupnja zakreta vratila servo motora u stupnjevima>,END pomiče servo motor u položaj za crtanje, s vrhom olovke koji dodiruje papir. Stroj ima postavljenu poziciju spremljenu u EEPROM-u, ali ako mu se preda vrijednost postavljena u SETUP kartici, pomiče servo na tu poziciju i koristit će tu poziciju do sljedećeg resetiranja robotskog crtača. C45 naredba služi za spremanje određenih položaja servo motora u EEPROM[44].
5. Olovka gore: C14,< pozicija stupnja zakreta vratila servo motora u stupnjevima>,END pomiče servo motor za podizanje olovke u gornji položaj. Parametar se ponaša na isti način kao C13[44].
6. Izravno pomicanje olovke: C17,<lijeva ciljna duljina>,<desna ciljna duljina>,<duljina segmenta linije>,END pomiče olovku do cilja, crtajući ravnu liniju u kartezijskom prostoru. Crta može biti podijeljena na više segmenata. Parametar duljine segmenta linije kontrolira koliko su dugi segmenti linije. Zadana vrijednost je dva i ona će se koristiti pri crtanju[44].
7. Postavljanje veličine robotskog crtača: C24,<širina>,<visina>,END postavlja dimenzije robotskog crtača u milimetrima[44].
8. Resetiranje EEPROM memorije: C27,END Resetiranje na tvorničke postavke (kartica SETUP, funkcija RESET MACHINE TO FACTORY)[44].
9. Postavljanje pomaka remena po okretaju koračnog motora: C29,<mm po okretaju>,END tip podatka integer. Postavlja koliko se remen pomakne s jednim okretajem koračnog motora[44]. Ovo je jedno od najvažnijih podataka. Ako se ova vrijednost unese neispravno, slike će biti izobličene.
10. Postavljanje koraka motora po okretaju: C30,<koraci po okretaju>,END tip podatka integer. Različiti koračni motori imaju različit broj koraka. Ako ova vrijednost nije ispravna, slike mogu biti veće ili manje u odnosu na zadane dimenzije[44].
11. Postavljanje maksimalne brzine motora: C31,<maksimalna brzina motora>,END Mjereno u koracima po sekundi[44].

12. Postavljanje ubrzanja: C32,<ubrzanje>,END Mjereno u koracima po sekundi na kvadrat[44].

8.3. Vektorska grafika

Vektorska grafika su računalne slike stvorene korištenjem niza naredbi ili matematičkih funkcija koje postavljaju linije i oblike u dvodimenzionalni ili trodimenzionalni prostor. U vektorskoj grafici, rad ili datoteka umjetnika stvara se i sprema kao niz vektorskih izjava. Vektorska grafička datoteka opisuje niz točaka koje treba povezati[45]. Prednosti vektorske grafike su:

1. Skalabilnost: Ovo je glavna prednost vektorske grafike. Budući da je vektorska grafika izvedena iz matematičkih vektorskih odnosa ili odnosa između točaka koje stvaraju linije i krivulje, slika izgleda čista i točna u bilo kojoj veličini[45].
2. Mala veličina datoteke: Vektorska grafika općenito ima malu veličinu datoteke jer pohranjuje samo mali broj točaka i matematičkih odnosa između njih. Ti su odnosi izraženi u kodu, koji je manje memorijski intenzivan u usporedbi s pohranjivanjem piksela[45].
3. Jednostavna za uređivanje: Vektorske datoteke lako je uređivati jer korisnici mogu brzo mijenjati odnose vektora kako bi, na primjer, zamijenili boje ili promijenili oblike linija. Ovo je korisno u iterativnom procesu, poput grafičkog dizajna, koji zahtijeva mnogo uređivanja[45].
4. Lako se učitava: Budući da su veličine datoteka manje, lako je prenijeti i učitati vektorske datoteke na različite uređaje i programe[45].
5. Preciznost: Mogućnost skaliranja vektorske grafike omogućuje precizan izgled i dojam[45].

Razlika između vektorske i rasterske grafike je u tome što rasterska grafička slika preslikava bitove izravno u prostor prikaza, što se također naziva bitmapa. Rasterska grafika sastoji se od fiksnog broja piksela, što je čini manje skalabilnom od vektorske grafike. U određenoj točki, kada se rasterska slika dovoljno poveća, rubovi postaju neravni i ona izgleda pikselizirana tj. kada pikseli postanu vidljivi. Rasterska grafika ne može se povećati bez žrtvovanja kvalitete slike. Vektorske i rasterske slike mogu se pretvoriti jedna u drugu pomoću odgovarajućeg softvera[45].

Jedan od najčešćih načina pohranjivanja vektorske grafike je SVG format. Ako se takva slika otvori pomoću uređivača teksta, mogu se vidjeti razne točke, linije, krivulje i geometrijski oblici izraženi preko X i Y koordinata pomoću različitih parametara.

Najznačajniji element SVG formata je path. Element path stvara složene oblike kombiniranjem više ravnih linija ili zakrivljenih linija. Oblik elementa <path> definiran je jednim parametrom: d. Atribut d sadrži niz naredbi i parametara koje te naredbe koriste[46].

Svaka od naredbi poziva se i stvara određenim slovom. Na primjer, prelazak na x i y koordinate (10, 10). Ta naredba poziva se slovom M. Kada parser naiđe na ovo slovo, zna da se treba pomaknuti na točku. Dakle, za prelazak na (10, 10) naredba za korištenje bila bi M 10 10. Nakon toga, parser počinje čitati sljedeću naredbu. Sve naredbe također dolaze u dvije varijante. Veliko slovo navodi apsolutne koordinate na stranici, a malo slovo navodi relativne koordinate (npr. pomak od posljednje navedene točke)[46].

Postoji nekoliko naredbi koje se koriste za opisivanje linija i krivulja, no najčešće su dvije: ravna linija i kubična Bezierova krivulja. Postoje tri naredbe koje crtaju linije. Najopćenitija je naredba „Linija do“, koja se poziva s L. L uzima dva parametra, koordinate x i y, i povlači liniju od trenutnog položaja do novog položaja[46].

Postoje tri različite naredbe koje se mogu koristiti za stvaranje glatkih krivulja. Dvije od tih krivulja su Bézierove krivulje, a treća je luk ili dio kruga. Kubična Bézierova krivulja uzima dvije kontrolne točke za svaku točku. Stoga, da bi se stvorio kubični Bézier, potrebno je odrediti tri skupa koordinata. Posljednji skup koordinata (x, y) određuje gdje linija treba završiti. Druge dvije su kontrolne točke. Kontrolne točke (x1, y1) definiraju početak krivulje, a (x2, y2) kontrolna točka definira kraj krivulje. Kontrolne točke opisuju nagib linije u svakoj točki. Bézierova funkcija tada stvara glatku krivulju koja prelazi s nagiba utvrđenog na početku linije na nagib na drugom kraju[46].

9. ACCELSTEPPER KNJIŽNICA

AccelStepper je knjižnica za Arduino napisana u C++. Za korištenje, konstruiraju se objekti tipa AccelStepper. Ti softverski objekti obično imaju motor ili stepper kao dio svojih imena i izravno su povezani s fizičkim koračnim motorima i njihovim upravljačima. Podržano je nekoliko različitih sučelja. Odgovarajuća sučelja za pojedinu primjenu moraju se odrediti prilikom izgradnje objekta. Nakon što je objekt konstruiran, različite funkcije koje se nalaze u knjižnici mogu se koristiti za kontrolu objekata. Prva skupina funkcija su funkcije postavljanja koje određuju fizičke mogućnosti (kao što su brzina i ubrzanje) motora i fizičke vrijednosti (kao što je položaj na koji se treba pomaknuti). Sljedeća skupina su funkcije koje zapravo pokreću motor šaljući mu signale koji će uzrokovati korake. Ove funkcije koriste vrijednosti iz funkcija postavljanja za kontrolu kretanja. Sljedeće su funkcije koje daju informacije o statusu gibanja. Nakraju postoje funkcije upravljanja priključcima koje konfiguriraju i kontroliraju priključke koji su u interakciji s upravljačkim programom[47].

Koračni motor se pomiče kada primi električni signal koji uzrokuje da motor napravi korak. Knjižnica AccelStepper ima samo jednu funkciju koja se može pozvati i koja uzrokuje kretanje: runSpeed(). Druge funkcije koje rezultiraju kretanjem pozivaju runSpeed() da proizvede korake. Svaka takva funkcija u svom nazivu ima riječ run. Postoje dvije vrste funkcije kretanja: konstantna brzina (ograničena trenutnom vrijednošću brzine) i promjenjiva brzina (ograničena postavkama ubrzanja i maksimalne brzine, te položajem u odnosu na cilj)[47].

Funkcija runSpeed() određuje kada treba poduzeti korak. runSpeed() oduzima vrijeme zadnjeg koraka od trenutnog vremena. Ako je rezultat veći ili jednak vrijednosti stepIntervala, runSpeed će povećati (ili smanjiti, prema potrebi) currentPosition (trenutačna pozicija koračnog motora), pozvati step() funkciju i ažurirati vrijeme zadnjeg poduzetog koraka (step() je interna funkcija - ne može se izravno pozvati te uzrokuje slanje ispravnog električnog signala sučelju kako bi se koračni motor kretao.) Svaki poziv runSpeed() ponavlja ovaj proces. Obično to znači pozivanje funkcije runSpeed() u Arduino funkciji loop()[47].

Kontrolna varijabla za runSpeed() je stepInterval pa je potrebno razumjeti kako je AccelStepper izračunava. Internom varijablom stepInterval korisnik ne upravlja izravno. U najjednostavnijem slučaju, pozivanje setSpeed() uzrokovat će izračun nove vrijednosti brzine - i stepIntervala. Vrijednost brzine bit će ograničena na maxSpeed. Ta se vrijednost može promijeniti pomoću setMaxSpeed(). Brzina i smjer prate se odvojeno tako da je stepInterval uvijek pozitivan; smjer ovisi o predznaku brzine (u setSpeed()), ili o smjeru currentPosition u odnosu na targetPosition[47].

Funkcija runSpeed() služi za konstantnu brzinu. No što kada je potrebno ubrzanje ili kretanje motora do željene lokacije? Za oboje se koristi funkcija run(). Funkcija run() prvo poziva runSpeed() da napravi korak pri trenutnoj brzini i smjeru. Zatim poziva computeNewSpeed(). Funkcija computeNewSpeed() izračunava novu brzinu i postavlja je kao trenutnu brzinu[47].

Funkcija run() povećat će ili smanjiti brzinu na temelju položaja (udaljenost do cilja određuje je li vrijeme za usporavanje ili ubrzanje), ubrzanja (više ili niže) i maksimalne brzine (je li dostignuta?). Ova funkcija briše bilo koje vrijednosti brzine koje postavlja setSpeed(). Nema potrebe izravno pozivati ovu funkciju i to se ne bi trebalo činiti[47].

Druge funkcije koje će uzrokovati kretanje uključuju runSpeedToPosition(), koja provjerava je li dosegnuta ciljna pozicija i poziva runSpeed() ako nije. Druga je runToPosition(), koja jednostavno poziva run() dok se ne postigne ciljna pozicija to jest, blokira se dok se pozicija ne dosegne. Konačno, runToNewPosition() dopušta određivanje nove pozicije, zatim poziva moveTo() da to postavi kao cilj i poziva runToPosition() da izvrši premještanje[47].

9.1. Konstruiranje AccelStepper objekta

Za konstruiranje AccelStepper objekta, potrebno je znati vrstu upravljača koračnog motora koji se koristi i priključke koji se koriste za njegovu kontrolu[47].

Ključna stavka je vrsta sučelja (upravljača) koji se koristi. Najčešći upravljači su upravljački za korak i smjer, kao što je upravljač A4988 koji se koristi za robotskog crtača. A4988 upravljač se pomoću AccelStepper definira kao: AccelStepper::DRIVER (Upravlja se samo sa korakom i smjerom te su potrebna samo dva priključka. Bilo kakva upotreba mikrokoraka ovdje nije važna. Upravljaču su važni samo koraci[47]. Nakon tipa sučelja, sljedeći argumenti su Arduino priključci:

1. Pin1: Arduino broj digitalnog priključka. Ovaj priključak služi za korak (prijelaz s niskog na visoko stanje znači korak)[47].
2. Pin2: Arduino broj digitalnog priključka. Ovaj priključak služi za definiranje smjera. Visoko znači jedan smjer, nisko znači drugi[47].

Arduino priključci koje sučelje koristi bit će inicijalizirani u Output modu tijekom konstruktora pozivom enableOutputs(), tako da nema potrebe za njihovim inicijaliziranjem u setup() dijelu Arduino programa[47].

9.2. Funkcije postavljanja

Ove funkcije postavljaju vrijednosti za kasniju upotrebu. Prvo ih treba pozvati da postave željene uvjete, a potom ih se može ponovno pozvati da im se promijene vrijednosti. Pozitivne i negativne vrijednosti za smjer proizvoljne su ovisno o tome kako je motor ožičen. Ključna stvar je da pokreću motor u dva suprotna smjera. Pozitivne i negativne vrijednosti za brzinu nisu bitne, brzina se pohranjuje kao apsolutna vrijednost[47].

9.2.1. Funkcija moveTo((long) apsolutna_pozicija)

Argument funkcije je apsolutna pozicija u koracima. Željena apsolutna pozicija je tipa podatka long. Može biti pozitivan ili negativan[47]. Funkcija postavlja ciljanu poziciju. Funkcija run() će pomaknuti motor s trenutne pozicije na poziciju postavljenu pozivom ove funkcije. Funkcija moveTo() također ponovno izračunava brzinu za sljedeći korak. Za korištenje konstantne brzine, poziva se setSpeed() nakon pozivanja moveTo(). Ako se moveTo() pozove dok se motor kreće, ciljani položaj se odmah mijenja i algoritam ubrzanja koristi se za izračunavanje nove brzine[47].

9.2.2. Funkcija move((long) relativno_kretanje)

Argument funkcije je relativno kretanje u koracima. Željeno kretanje u odnosu na trenutnu poziciju. Tip podatka argumenta je long i može biti pozitivan ili negativan[47]. Funkcija postavlja ciljani položaj u odnosu na trenutni položaj. Funkcija move() također ponovno izračunava brzinu za sljedeći korak. Za korištenje konstantne brzine, poziva se setSpeed() nakon pozivanja move(). Ako se move() pozove dok se motor kreće, rezultat je isti kao moveTo(). Jedina je razlika način na koji se izračunava nova ciljna pozicija[47].

9.2.3. Funkcija setMaxSpeed((float) brzina)

Argument funkcije je brzina u koracima u sekundi. Najveća željena brzina kao tip podatka float. Negativna vrijednost se može proslijediti, ali će biti pohranjena kao apsolutna vrijednost. Ova funkcija postavlja najveću dopuštenu brzinu. Obično će ova funkcija biti prva pozvana kada se koristi objekt AccelStepper (to jest, koračni motor). Ako se koristi funkcija run(), tada će motor ubrzati do ove brzine. Svaka vrijednost brzine koju postavlja setSpeed() bit će zanemarena. Ako se koristi runSpeed(), onda se setSpeed() mora pozvati nakon setMaxSpeed()[47].

9.2.4. Funkcija setAcceleration((float) ubrzanje)

Argument funkcije je ubrzanje u koracima po sekundi na kvadrat kao tip podatka float. Može se navesti kao negativan, ali se pohranjuje samo apsolutna vrijednost. Postavlja stopu

ubrzanja ili usporenja. Ubrzanje koristi run() za povećanje ili smanjenje brzine kojom se motor okreće[47].

9.2.5. Funkcija setSpeed((float) brzina)

Argument funkcije je brzina u koracima po sekundi kao tip podatka float. Može biti pozitivan ili negativan. Ova funkcija postavlja željenu konstantnu brzinu za korištenje s runSpeed(). Brzina će biti ograničena trenutnom vrijednošću setMaxSpeed(). Ova će se brzina koristiti sve dok se poziva runSpeed(). Ako se pozove run(), ova vrijednost će biti zanemarena i prebrisana[47].

9.2.6. Funkcija setCurrentPosition((long) pozicija)

Argument funkcije je željena vrijednost položaja u koracima gdje god se motor trenutno nalazi. Može biti pozitivna ili negativna i tipa je long. Ova funkcija će trenutni položaj motora i ciljani položaj učiniti jednakima navedenoj vrijednosti. Na primjer, ako se motor pomaknuo na položaj (primjerice 213) i pozove se setCurrentPosition(100), trenutni položaj i ciljni položaj bit će postavljeni na 100. Bit će potrebno 100 pozitivnih koraka da se postigne položaj 200 (ako se poziva moveTo(200)) umjesto 13 koraka u negativnom smjeru. Ova funkcija također će vratiti vrijednost brzine na 0,0. Ova je funkcija najkorisnija za postavljanje nulte pozicije na koračnom motoru. Ova funkcija se ne smije pozvati dok se motori kreću. Vrijednosti trenutne pozicije i ciljne pozicije bit će odmah promijenjene, a brzina će odmah biti postavljena na 0. To će prisiliti motor da se pokuša trenutno zaustaviti i najvjerojatnije će rezultirati propuštenim koracima i mogućim oštećenjem sustava. Prije pozivanja setCurrentPostion() potrebno je provjeriti je li se steper zaustavio[47].

9.3. Funkcije kretanja

Funkcije runSpeed() i run() su objašnjene na početku poglavlja. Ostale funkcije kretanja su objašnjene u narednim poglavljima.

9.3.1. Funkcija runSpeedToPosition()

Ova funkcija izvršava runSpeed() osim ako se ne dosegne targetPosition. Ovu funkciju treba često pozivati baš kao i runSpeed() ili run(). Pokreće motor trenutno odabranom brzinom osim ako se ne postigne ciljni položaj. Ne implementira ubrzanja[47].

Sada se mogu definirati funkcije blokiranja kretanja. Blokiranje znači da se nijedan drugi programski kod ne pokreće dok se ove funkcije ne dovrše.

9.3.2. Funkcija runToPosition()

Ova funkcija će neprestano pozivati run(), blokirajući druge naredbe, dok ne vrati false, što znači da je željena pozicija dosegnuta. Pomiče motor (s ubrzavanjem ili usporavanjem) u ciljni položaj i blokira ostali programski kod dok ne dođe u položaj[47].

9.3.3. Funkcija runToNewPosition((long) apsolutna pozicija)

Ova funkcija izvršava moveTo(željena pozicija), zatim runToPosition(). Pomiče motor s ubrzavanjem ili usporavanjem na novi ciljni položaj i blokira ostatak koda dok ne dođe na tu poziciju. Argument ove funkcije je apsolutna pozicija u koracima. Željena apsolutna pozicija tipa podatka long. Može biti pozitivan ili negativan[47].

9.4. Informacijske funkcije

Oni se mogu pozvati za očitavanje vrijednosti određenih varijabli ili provjeru statusa kretanja. Brzina je uvijek u koracima po sekundi, a položaj u koracima od pozicije 0.

1. maxSpeed(): Maksimalna brzina koju postavlja setMaxSpeed(). Vraća vrijednost tipa float[47].
2. speed(): Vraća najnoviju brzinu kao vrijednost tipa float u koracima po sekundi. To može biti brzina koju postavlja setSpeed() ako je pozvan samo runSpeed() ili brzina izračunata ako se pozove run()[47].
3. targetPosition(): Ciljani položaj postavljen pomoću move() ili moveTo(). Vraća vrijednost tipa long[47].
4. currentPosition(): Gdje je trenutno motor. Vraća vrijednost tipa long[47].
5. distanceToGo(): Jednako je razlici targetPosition i currentPosition kao pozitivan cijeli broj. Vraća vrijednost tipa long. Udaljenost od trenutne pozicije do ciljne pozicije u koracima[47].
6. isRunning(): Vraća true ako brzina nije nula i distanceToGo nije 0[47].

9.5. Funkcije upravljanja priključcima

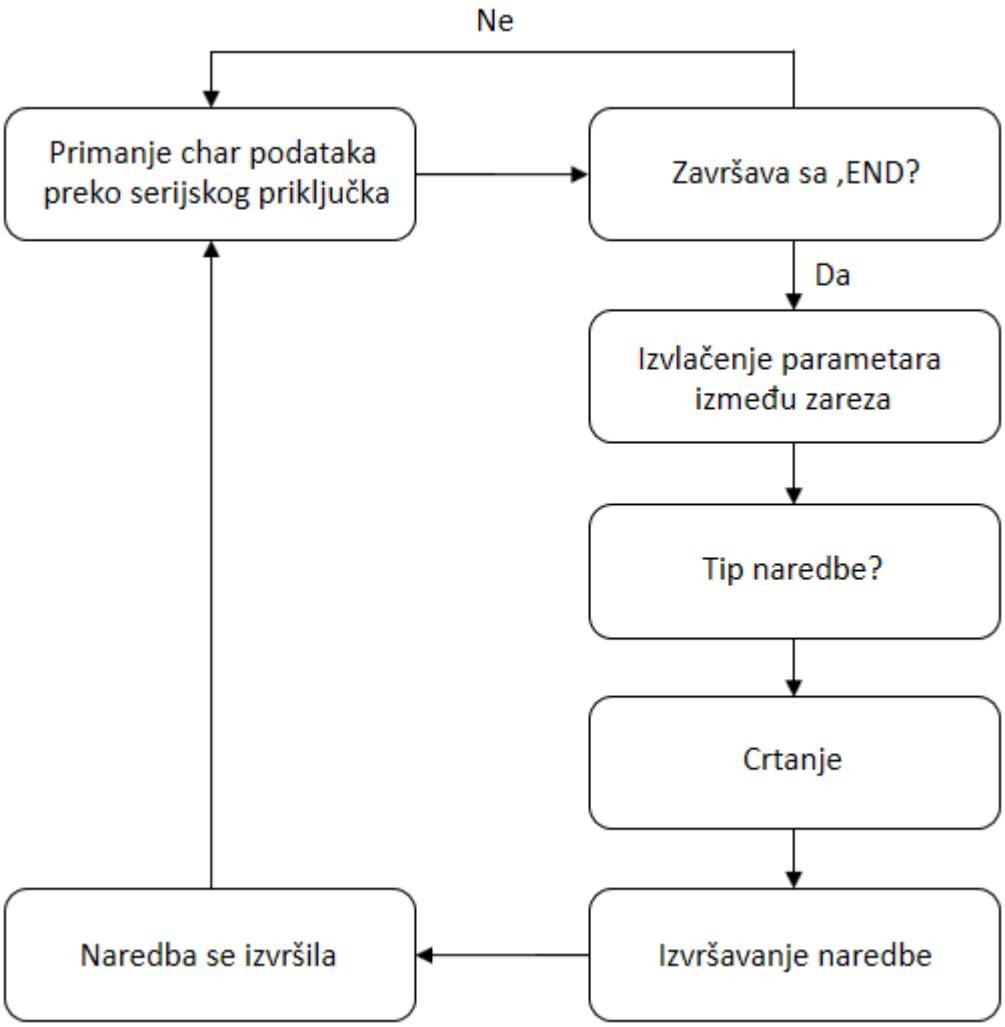
Priključci sučelja šalju signale da izazovu korak. Ove se funkcije mogu koristiti za njihovu daljnju konfiguraciju i kontrolu.

1. enableOutputs(): Omogućava da EN priključak se postavi kao izlaz. Automatski ga poziva ga konstruktor prilikom inicijalizacije. Treba ga pozvati izravno samo ako je pozvan disableOutputs(). Ako je omogućeni priključak definiran, ova funkcija će njime također upravljati[47].

2. disableOutputs(): Postavlja sve izlaze EN priključaka na LOW i onemogućuje EN priključke ako su postavljeni. Ovisno o dizajnu elektronskog sučelja, ovo može isključiti napajanje zavojnica motora, štedeći energiju. Preporučava se onemogućavanje EN priključaka tijekom mirovanja i zatim ponovno omogućavanje pomoću enableOutputs() prije kretanja[47].
3. setPinsInverted(): Ova funkcija može invertirati smisao bilo kojeg priključka sučelja. Postavljanje vrijednosti za priključak true ga invertira, dok ga postavljanje na false ostavlja istim. Ova se funkcija najčešće koristi za invertiranje signala za uključivanje. Obrazac setPinsInverted (directionPin, stepPin, enablePin) služi za upravljače koraka/smjera poput A4988 upravljača. Za postavljanje samo obrnutog signala za uključivanje, koristi se setPinInverted(false, false, true). Ako se ne koristi priključak za uključivanje, nije potrebno navesti nikakvu vrijednost[47].
4. setEnablePin(enablePin): Određuje broj priključka koji ćete koristi za uključivanje. Priključak će biti konfiguriran kao izlaz, a ispravna vrijednost će biti postavljena kada se ova funkcija pozove. Iz tog razloga, ako signal za uključivanje treba biti invertiran, setPinsInverted() treba pozvati prije setEnablePin(). Kao što je gore spomenuto, enableOutputs() i disableOutputs() će ispravno upravljati omogućivanjem priključka, zajedno sa svim inverzijama[47].

10. STRUKTURA KODA ROBOTSKOG CRTAČA

Nakon što je SVG slika učitana u GUI robotskog crtača, aplikacija tu sliku pretvara u set naredbi koje postavljaju logiku crtanja, kako bi robotski crtač ispravno funkcionirao. Naredbe započinju s tipom naredbe (primjerice C01), koje definiraju koja će se funkcija izvršiti. Odabirom bilo koje opciju unutar GUI–a, bilo to postavljanje dimenzija i parametara robota ili samo crtanje, naredbe se počinju slati preko serijskog priključka na Arduino. Arduino prima podatke tipa char preko serijskog priključka. Sve dok naredba ne završava s ‚END, Arduino će primati podatke. Nakon primanja cjelokupne naredbe, počinje izvlačenje parametara iz cjelokupne naredbe. Parametri su odvojeni zarezom, te pomoću njih se prepoznaje gdje započinje, a gdje završava pojedini parametar. Tada se provjerava koji je tip naredbe predan Arduinu i na temelju toga se odlučuje koja funkcija će se izvršiti unutar datoteke exec.ino. Svaka naredba ima svoju funkciju koja odrađuje pojedini zadatak. Arduino tada izvršava naredbu koja izravno utječe na proces crtanja.

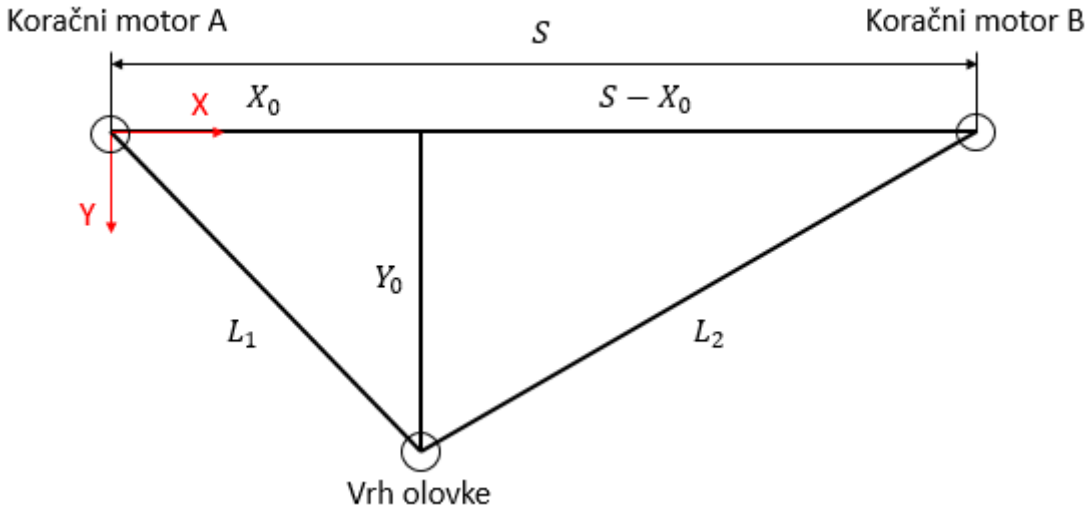


Slika 56. Tok programa

10.1. Matematički model robotskog crtača

Naredba C17,<lijeva ciljna duljina>,<desna ciljna duljina>,<duljina segmenta linije>,END postiže kretanje motora iz početne točke u ciljnu točku. Zbog složene arhitekture robotskog crtača problem je povezati koordinate olovke u ovisnosti o udaljenosti od koračnih motorima sa kartezijskim koordinatama koje definiraju same vektore slike po kojima olovka crta. Naredba C17 šalje koordinate udaljenosti od motora A i motora B. Potrebno je pronaći funkciju koja prebacuje te koordinate u kartezijski prostor te funkciju koja ciljnu točku prebacuje iz kartezijskih koordinata u koordinate udaljenosti od koračnih motora.

Započinje se pojednostavljenom geometrijom robotskog crtača. Poznati su podaci o udaljenosti olovke od motora A (L_1) i motora B (L_2) i udaljenost između osi motora A i B (S).



Slika 57. Početni položaj vrha olovke

Pitagorinim poučkom se mogu povezati koordinate X_0 i Y_0 sa udaljenostima L_1 i L_2:

$$Y_0^2 = L_1^2 - X_0^2 \tag{3}$$

$$L_2^2 = (S - X_0)^2 + Y_0^2 \tag{4}$$

Gdje je:

L_1	–	udaljenost olovke od motora A u koracima
L_2	–	udaljenost olovke od motora A u koracima
S	–	udaljenost između koračnih motora u koracima
X_0	–	početna koordinata apscise u koracima
Y_0	–	početna koordinata ordinate u koracima

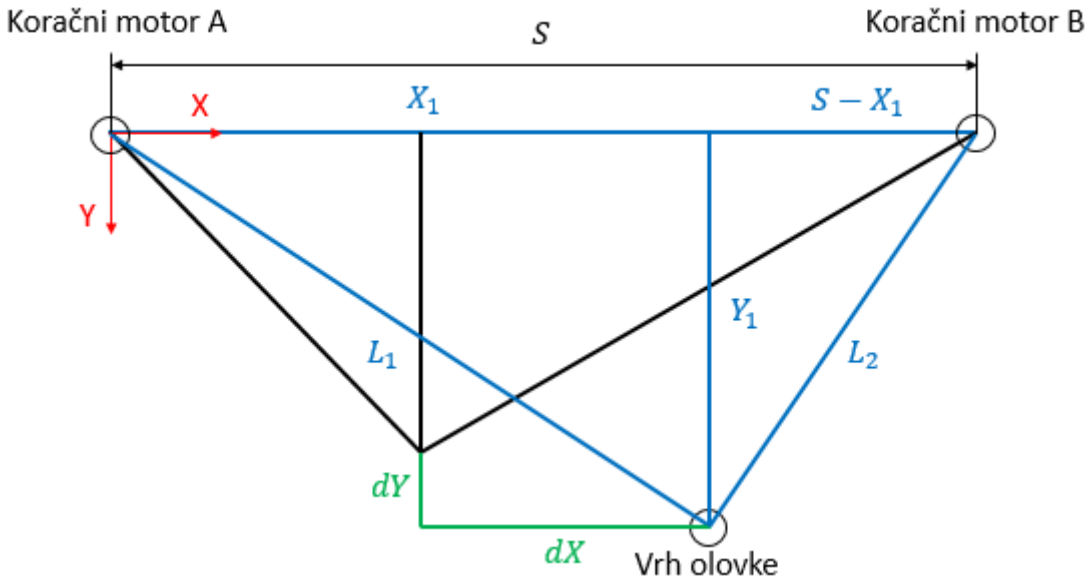
Uvrštavanjem jednadžbe (3) u (4) te izlučivanjem varijable X_0 dobiva se:

$$X_0 = \frac{S^2 + L_1^2 + L_2^2}{2S} \tag{5}$$

Napokon uvrštavanjem jednadžbe (5) u (3), dobiva se i Y_0 koordinata:

$$Y_0 = \sqrt{L_2^2 - (S - X_0)^2} \tag{6}$$

No za potrebe rada robotskog crtača bitno je i ciljnu točku koja je sada izražena u kartezijskim koordinatama vratiti u koordinate ovisno o udaljenosti koračnih motora.



Slika 58. Ciljani položaj vrha olovke

Koordinate Y_1 i X_1 se mogu izraziti kao koordinate Y_0 i X_0 zbrojene sa pomacima dX i dY:

$$X_1 = dX + X_0 \tag{7}$$

$$Y_1 = dY + Y_0 \tag{8}$$

Gdje je

dX	–	pomak s obzirom na X os u koracima
dY	–	pomak s obzirom na Y os u koracima
X_1	–	ciljna koordinata apscise u koracima
Y_1	–	ciljna koordinata ordinate u koracima

Pitagorinim poučkom se mogu povezati udaljenosti L_1 i L_2 sa koordinatama X_1 i Y_1:

$$L_1^2 = X_1^2 - Y_1^2 \tag{9}$$

$$L_2^2 = (S - X_1)^2 + Y_1^2 \tag{10}$$

Krajnjim uvrštavanjem jednadžbi (7) i (8) u jednadžbe (9) i (10), dobiva se transformacija iz točke u kartezijskim koordinatama u točku ovisnu o udaljenosti motora A i B:

$$L_1 = \sqrt{(dX + X_0)^2 + (dY + Y_0)^2} \tag{11}$$

$$L_2 = \sqrt{\left(S - (dX + X_0)\right)^2 + (dY + Y_0)^2} \tag{12}$$

Potrebno je napomenuti da ove vrijednosti nisu izražene u milimetrima već u koracima koji su potrebni kako bi se koristile funkcije AccelStepper knjižnice.

10.2. Datoteka configuration.ino

Ova datoteka služi za postavljanje odgovarajućeg sučelja. Datoteka configuration.ino konstruira Accel Stepper objekte (oni predstavljaju realne koračne motore u kodu) te služi za postavljanje bilo kakvih mikrokoračnih kombinacija. Robotski crtač koristi upravljače koračnih motora A4988 te se pomoću njih upravlja korakom i smjerom. Kako bi kod funkcionirao, potrebno je definirati priključke za korak, smjer i priključak koji omogućuju kretanje koračnih motora. Prvi koračni motor ima priključke 2 za korak i 5 za smjer, dok drugi 3 za korak i 6 za smjer. Pošto se oba upravljača A4988 nalaze na CNC V3 štitu koji ima jedan priključak za omogućivanje, koračni motori će imati isti priključak pod brojem 8.

Direktivom #define u datoteci main se definirala konstanta SERIAL_ STEPPER_DRIVER. Direktiva pretprocesora #ifdef očitava da postoji konstanta SERIAL_STEPPER_DRIVER te da se dio koda između #ifdef i #endif kompajlira. Unutar tog bloka se definiraju imena konstanti priključaka i njihove vrijednosti. Također se konstruiraju dva AccelStepper objekta koji predstavljaju koračne motore. Tim objektima se pridružuju odgovarajući priključci.

```
#ifdef SERIAL_STEPPER_DRIVERS
#define MOTOR_A_ENABLE_PIN 8
#define MOTOR_A_STEP_PIN 2
#define MOTOR_A_DIR_PIN 5

#define MOTOR_B_ENABLE_PIN 8
#define MOTOR_B_STEP_PIN 3
#define MOTOR_B_DIR_PIN 6
AccelStepper motorA(1,MOTOR_A_STEP_PIN, MOTOR_A_DIR_PIN);
AccelStepper motorB(1,MOTOR_B_STEP_PIN, MOTOR_B_DIR_PIN);
#endif
```

U funkciji configuration_motorSetup() ako je definirana konstanta SERIAL_STEPPER_DRIVER (u slučaju robotskog crtača je), priključak 8 se postavlja kao izlaz pomoću funkcije pinMode() sa vrijednošću HIGH pomoću funkcije digitalWrite(). Pomoću funkcija setEnablePin() i setPinsInverted() se pridružuje priključak za uključivanje koračnim motorima te se postavlja trenutno na stanje LOW. Funkcija setPinsInverted() također jednom od motora mijenja smjer.

```
void configuration_motorSetup()
{
#ifdef SERIAL_STEPPER_DRIVERS
  pinMode(MOTOR_A_ENABLE_PIN, OUTPUT);
  digitalWrite(MOTOR_A_ENABLE_PIN, HIGH);
  pinMode(MOTOR_B_ENABLE_PIN, OUTPUT);
  digitalWrite(MOTOR_B_ENABLE_PIN, HIGH);
  motorA.setEnablePin(MOTOR_A_ENABLE_PIN);
  motorA.setPinsInverted(false, false, true);
  motorB.setEnablePin(MOTOR_B_ENABLE_PIN);
  motorB.setPinsInverted(true, false, true); // this one
#endif
```

10.3. Datoteka pen_utilization.ino

Datoteka pen_utilization.ino služi za spuštanje i podizanje vrha olovke od ploče. Kada se preko GUI–a odaberu vrijednosti kuta dodira olovke i kuta odmaka, te vrijednosti se predaju ovoj datoteci za korištenje. Također, pen_utilization.ino provjerava je li olovka spuštena ili podignuta u slučaju da se ovoj datoteci preda naredba za podizanje, a olovka je već podignuta i obratno.

Ako se primi jednostavna naredba olovka gore ili podizanje olovke C14,END, stroj će neće podići olovku ako misli da je već podignuta. Kako bi odlučio o tome provjerava vrijednost globalne boolean varijable isPenUp. Ako se primi naredba olovka gore, koja uključuje položaj olovke (primjerice C14,150,END), tada se globalna varijabla položaja gore ažurira, a servo se pomiče na taj položaj.

Kod započinje sa funkcijom koja pomiče ručicu servo motora u željeni položaj. Toj funkciji se predaju argumenti start, end i delay_ms. Argumenti start i end ovisno o tome je li olovka podignuta ili spuštena, poprimat će vrijednosti koje će diktirati proces spuštanja ili podizanja. Ako je start vrijednost manja od end vrijednosti, olovka će se podizati. Proces podizanja se omogućava for petljom gdje se start vrijednost povećava za jedan stupanj svakom iteracijom petlje dok ne postane jednaka vrijednosti end. Funkcije attach() i write() iz knjižnice Servo omogućuju rad servo motora. Funkcija attach() služi sa pridruživanje servo varijable na određeni priključak. Funkcija write() zapisuje vrijednost u servo, kontrolirajući osovinu u

skladu s tim. Na standardnom servu, ovo će postaviti kut osovine u stupnjevima. Funkcijom detach() se odvaja servo varijabla od priključka.

```
#ifdef PENLIFT
void penlift_movePen(int start, int end, int delay_ms)
{
  penHeight.attach(PEN_HEIGHT_SERVO_PIN);
  if(start < end)
  {
    for (int i=start; i<=end; i++)
    {
      penHeight.write(i);
      delay(delay_ms);
#ifdef DEBUG_PENLIFT
      Serial.println(i);
#endif
    }
  }
  else
  {
    for (int i=start; i>=end; i--)
    {
      penHeight.write(i);
      delay(delay_ms);
#ifdef DEBUG_PENLIFT
      Serial.println(i);
#endif
    }
  }
  penHeight.detach();
}
```

Funkcija penlift_penUp() služi za podizanje olovke od radne površine. Ona predaje argumente funkciji penlift_movePen kako bi servo motor podigao olovku. Ako je ulazni broj parametara više od jedan, onda pomoću null uvjetnog operatora se provjerava boolean vrijednost isPenUp varijable. Ako je vrijednost true, onda će varijabla positionToMoveFrom poprimiti vrijednost varijable upPosition. Ako je vrijednost varijable isPenUp false, onda će varijabla positionToMoveFrom poprimiti vrijednost varijable downPosition. Tada će vrijednost upPosition poprimiti vrijednost koju korisnik predaje preko GUI robotskog crtača pomoću funkcije atoi() koja pretvara tip podatka string u integer. Ovaj dio koda služi ako je predan novi položaj uz naredbu C14. Tada se globalna varijabla položaja gore ažurira, a servo se pomiče na taj položaj čak i ako je već položaj olovke bio definiran kao gore.

U slučaju da je ulazni broj parametara u dolaznoj naredbi manji od jedan, tada funkcija penlift_penUp() podiže olovku prema zadanim vrijednostima upPosition i downPosition. U slučaju da je boolean vrijednost varijable isPenUp false, izvršava se funkcija sa zadanim

argumentima penlift_movePen(downPosition, upPosition, penLiftSpeed). Ako je isPenUp true, olovka je već podignuta i nema potrebe za izvršavanjem funkcije penlift_movePen().

```
void penlift_penUp()
{
  if (inNoOfParams > 1)
  {
    int positionToMoveFrom = isPenUp ? upPosition : downPosition;
    upPosition = atoi(inParam1);
    penlift_movePen(positionToMoveFrom, upPosition, penLiftSpeed);
  }
  else
  {
    if (isPenUp == false)
    {
      penlift_movePen(downPosition, upPosition, penLiftSpeed);
    }
  }
  isPenUp = true;
}
```

Funkcija penlift_penDown() funkcionira potpuno isto, samo sada se koristi naredba C13. Ako je uz naredbu C13 definiran i položaj, tada se globalna varijabla položaja dolje ažurira, a servo se pomiče na taj položaj čak i ako je već položaj olovke bio definiran kao dolje. U slučaju da je samo definirana naredba C13, tada ovaj dio koda provjerava je li olovka već spuštena te ako nije, spušta se.

10.4. Datoteka eeprom.ino

Datoteka eeprom.ino se koristi za čitanje i pisanje podataka u EEPROM memoriju. Ima nekoliko malih pomoćnih metoda za čitanje i pisanje u EEPROM pomoću EEPROM knjižnice. Ali uglavnom sadrži metodu loadMachineSpecFromEeprom, koja se koristi se za dohvaćanje spremljenih vrijednosti robotskog crtača kada se ponovno pokrene, ili kad se vrijednosti promijene. Također koristi knjižnicu EEPROMAnything.h koja služi za čitanje i pohranjivanje vrijednosti spremljenih u EEPROM memoriju.

Funkcija eeprom_resetEeprom() služi za brisanje svih vrijednosti iz adresa koje su definirane u main datoteci te postavljanje novih vrijednosti koje su definirane funkcijom eeprom_loadMachineSpecFromEeprom().

Nadalje slijede funkcije koje služe za čitanje vrijednosti iz EEPROM memorije. Svaka od tih funkcija ima sličnu strukturu: pomoću funkcije EEPROM_readAnything čitaju se vrijednosti koje se nalaze u EEPROM memoriji. Ako se slučajno nađe neka vrijednost koja je manja ili jednaka nuli, zamjenjuje se zadanom vrijednošću (primjerice, broj koraka po okretaju ne može biti nula niti negativna vrijednost). Funkcije koje će imati ovu strukturu su:

eeprom_loadMachineSize (služi za čitanje širine i visine stroja), eeprom_loadMachineSize (čitanje pomaka remena i broj koraka po okretaju), eeprom_loadPenLiftRange (vrijednost spuštene i podignute olovke u stupnjevima), eeprom_loadStepMultiplier (čitanje mikrokoraka), eeprom_loadSpeed (očitavanje brzine i ubrzanja).

Funkcija eeprom_loadMachineSpecFromEeprom() grupira sve ove funkcije, te kada se u setup dijelu programa ova funkcija pozove, ona će pozvati sve ostale funkcije koje su ovdje nabrojane.

10.5. Datoteka transformation.ino

Datoteka transformation.ino sadrži metode koje transformiraju koordinatne sustave kao i rutinu koja pogoni koračne motore do željenih točaka.

Funkcije getCartesianXFP i getCartesianYFP pretvaraju koordinate ovisne o udaljenosti od koračnih motora u kartezijske koordinate:

```
float getCartesianXFP(float aPos, float bPos)
{
  float calcX = (sq((float)pageWidth) - sq((float)bPos) + sq((float)aPos))
  / ((float)pageWidth*2.0);
  return calcX;
}
float getCartesianYFP(float cX, float aPos)
{
  float calcY = sqrt(sq(aPos)-sq(cX));
  return calcY;
}
```

Funkcije getMachineA i getMachineB služe za pretvaranje točke u koju se olovka želi pomaknuti iz kartezijskih koordinata u koordinate ovisne o udaljenosti od koračnih motora:

```
float getMachineA(float cX, float cY)
{
  float a = sqrt(sq(cX)+sq(cY));
  return a;
}
float getMachineB(float cX, float cY)
{
  float b = sqrt(sq((pageWidth)-cX)+sq(cY));
  return b;
}
```

Ove funkcije odgovaraju matematičkim operacijama pretvorbe koordinatnih sustava koje su definirane na početku ovog poglavlja.

Još jedna bitna funkcija datoteke transformation.ino je changeLenght. Ova funkcija se poziva u naredbama C01 i C17 za iskorištavanje AccelStepper knjižnice koja služi za

pokretanje samih motora. Pozivom funkcije joj se dodaju argumenti tA1 (udaljenost olovke od motora A) i tB1 (udaljenost olovke od motora B). U varijable currSpeedA i currSpeedB se postavlja trenutna brzina motora. Nakon toga se postavlja nova brzina na 0 i te pomoću funkcije moveTo() izračunava udaljenost između trenutne i željene koordinate u koracima. Kako funkcija moveTo() izračunava novu brzinu, za korištenje konstantne brzine potrebno je pozvati funkciju setSpeed() u koju se postavljaju varijable currSpeedA i currSpeedB. U while petlji se izvršava kretanje motora sve dok funkcije distanceToGo() od motora A i B nisu jednake nuli. Ako se koristi ubrzanje za kretanje od jedne do druge točke, onda se pozivaju funkcije run(), a ako se ne koristi ubrzanje, onda se poziva funkcija runSpeedToPosition() koja koristi konstantnu brzinu dok se ne dosegne ciljana točka.

```
while (motorA.distanceToGo() != 0 || motorB.distanceToGo() != 0)
{
  impl_runBackgroundProcesses();
  if (currentlyRunning)
  {
    if (usingAcceleration)
    {
      motorA.run();
      motorB.run();
    }
    else
    {
      motorA.runSpeedToPosition();
      motorB.runSpeedToPosition();
    }
  }
```

10.6. Datoteka execute_command.ino

Datoteka execute_command.ino sadrži osnovno stablo odlučivanja koje se grana na temelju dolazne naredbe iz GUI–a. Datoteka započinje s funkcijom exec_execute BasicCommand koja prima kao argument adresu varijable com tipa string (Varijabla com sadrži tip naredbe te se pomoću datoteke communication.ino obrađuje iz serijskog priključka kako bi bila pogodna za korištenje pomoću Arduina). Ovisno o tipu naredbe koji sadrži varijabla com, pomoću funkcije startWith() koja služi za odlučivanje je li neki string sadrži iste znakove kao i drugi, izvršit će se određena naredba. Primjerice, ako varijabla com je jednaka C14, u stablu odlučivanja izvršit će se funkcija penlift_penUp() jer varijabla com sadrži iste znakove kao i varijabla CMD_PENUP koja je definirana u glavnoj datoteci main. U stablu odlučivanja se nalaze sve funkcije koje su potrebne za ispravan rad robota.

Nakon odabira tipa naredbe, u datoteci se definiraju funkcije koje se pozivaju nakon što je odlučeno koja naredba se mora izvršiti iz stabla odlučivanja. Funkcija

exec_setMachineSizeFromCommand() služi za preuzimanje širine i visine robotskog crtača iz naredbe C24,<širina>,<visina>,END. Pomoću funkcije atoi() dobivamo vrijednosti širine i visine koje su sada tipa podatka int. Također ako već ista vrijednost nije zapisana u EEPROM memoriju, pomoću funkcije EEPROM_writeAnything će se pohraniti tamo. Funkcijom eeprom_loadMachineSize() se pozivaju parametri širine i visine.

Funkcijama exec_setMachineMmPerRevFromCommand(), exec_setMachineSteps PerRevFromCommand() i exec_setMachineStepMultiplierFromCommand() se parametri C29,<mm po okretaju>,END, C30,<koraci po okretaju>, END i C37,<množitelj>,END pretvaraju u tip podatka s kojim je moguće računati , te se pohranjuju u EEPROM memoriju. Nakon toga se ponovno učitavaju iz EEPROM memorije funkcijama eeprom_loadMachineSpecFromEeprom(), eeprom_loadMachineSpecFromEeprom(), i eeprom_loadMachineSpecFromEeprom().

Funkcija exec_setPenLiftRange() služi za postavljanje raspona stupnjeva osovine servo motora za podizanje i spuštanje olovke. Parametri se prebacuju u tip podatka integer. Ako je broj parametara tri, onda se samo upisuju podaci u EEPROM memoriju. Ako je broj parametara dva, onda se izvršava test podizanja i spuštanja olovke.

Funkcijama exec_setMotorSpeed() i exec_setMotorAcceleration() se postavlja maksimalna brzina i ubrzanje. Funkcija exec_setPosition() služi za postavljanje početne točke olovke robota iz koje će započeti proces crtanja. Funkcija exec_changeLength() služi za izravno upravljanje koračnim motorima, primjerice kada se želi testirati ispravnost kretanja motora. Koristi istu metodu changeLenght() kao i naredba C17 koja služi za crtanje SVG slika.

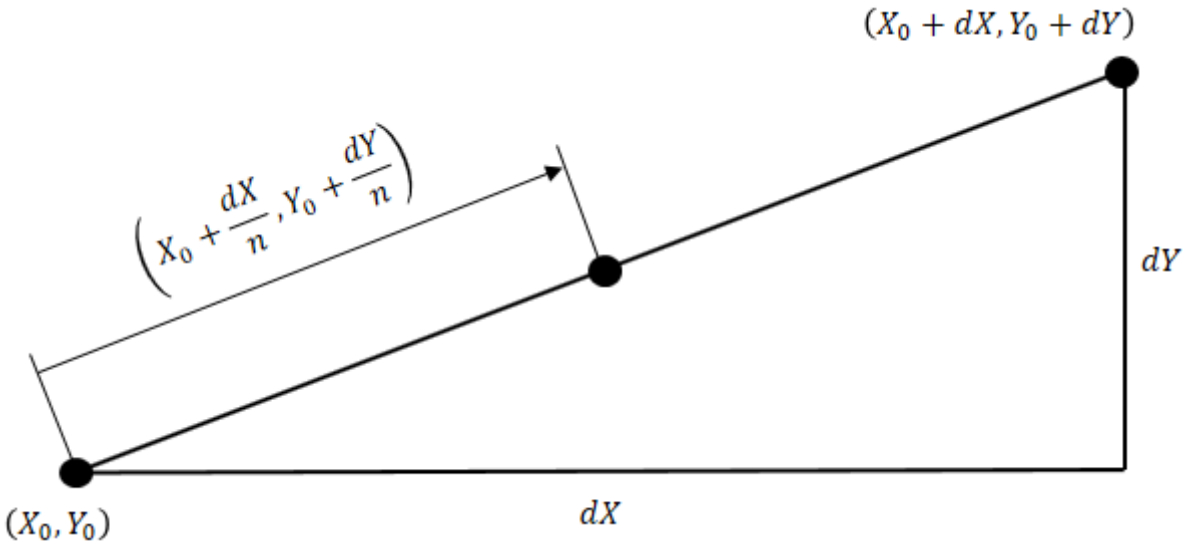
No možda najbitnija funkcija datoteke execute_command.ino je exec_drawBetweenPoints(). Ova funkcija služi za pomak olovke od početne točke do krajnje točke vektora slike. Vektori mogu imati dvije funkcije: za crtanje te za podizanje olovke i pomak na novu lokaciju za crtanje. Funkcija exec_drawBetweenPoints() je podfunkcija funkcije exec_changeLengthDirect(). Ona se poziva naredbom C17. Preuzimaju se tri parametra: ciljna udaljenost točke od motora A i od motora B te parametar duljine segmenta linije. Preuzimaju se početne koordinate olovke funkcijom knjižnice AccelStepper currentPosition(). Nakon što su preuzete sve vrijednosti koje su potrebne za izračun, poziva se funkcija exec_drawBetweenPoints() kojoj se predaje pet argumenata: početne i ciljne koordinate te duljina segmenta linije. Te točke se prebacuju u kartezijski prostor funkcijama getCartesianXFP i getCartesianYFP. Početna i ciljna točka su sada izražene preko X i Y

koordinata. Također ako se bilo koja točka nalazi izvan dimenzija robotskog crtača ona se preskače te se ne crta. Izračunava se razlika između početne i ciljne točke.

Linija koja se crta se dijeli na segmente. Svaki segment koji se crta ima svoju karakterističnu brzinu koja će ovisiti o prijeđenom putu. Brzina će se povećavati sve do sredine puta, a tada će početi usporavanje. Broj segmenata će biti jednak apsolutnoj vrijednosti količnika veće vrijednosti između *dX* i *dY* te parametra duljine segmenta linije. *dX* i *dY* se dijele sa brojem segmenata. U while petlji će se te nove vrijednosti zbrojiti s početnom točkom kako bi se dobila nova vrijednost početne točke koje se prebacuje u koordinate ovisne o udaljenosti motora A i B funkcijama getMachineA i getMachineB. Izračunat će se željena brzina koja postaje nova brzina motora funkcijom setSpeed() i nakraju će se pozvati funkcija changeLenght() iz datoteke transformation.ino koja izvršava pomicanje olovke. Petlja while će se izvršavati sve dok broj segmenata linije ne postane nula.

```
while (linesegs > 0)
    {
      clx = clx + deltaX;
      cly = cly + deltaY;

      float pA = getMachineA(clx, cly);
      float pB = getMachineB(clx, cly);

      runSpeed = desiredSpeed(linesegs, runSpeed, currentAcceleration*4);

      motorA.setSpeed(runSpeed);
      motorB.setSpeed(runSpeed);
      changeLength(pA, pB);

      linesegs--;
    }
```

Slika u nastavku geometrijski opisuje što se dešava kroz svaku iteraciju while petlje, gdje je n početni broj segmenata linije prije ulaska u while petlju:



Slika 59. Pomak olovke po svakoj iteraciji while petlje

Gdje je:

n – broj segmenata linije

10.7. Datoteka main

Ovo je dio programa koji sadrži funkcije setup() i loop(). Program započinje definiranjem ploče koja se koristi. Direktiva #define MICROCONTROLLER MC_UNO služi za definiranje ploče koja se koristi, a to je Arduino Uno. Također je potrebno definirati elektronsko sklopovlje za upravljanje koračnim motorima pomoću #define SERIAL_STEPPER_DRIVERS. S obzirom da se definirala ova riječ, u datoteci configuration.ino sve pod direktivom pretprocesora #ifdef SERIAL_STEPPER_DRIVERS će se izvršiti. To je potrebno kako bi definirali priključke za korak i smjer kao i kreirali objekt AccelStepper knjižnice (tj. objekte koji služe za upravljanje koračnim motorima.). Također definiraju se #define PENLIFT i #define VECTOR_LINES. Nadalje, uvode se knjižnice koje će poslužiti i olakšati izgradnju koda poput: #include <AccelStepper.h>, #include <Servo.h>, #include <EEPROM.h> i #include "EEPROMAnything.h".

U slijedećem dijelu programa varijablama se dodjeljuju EEPROM adrese u kojima će se pohranjivati različite vrijednosti kao što su: širina robotskog crtača, visina robotskog crtača, pomak u milimetrima po jednom okretaju koračnog motora, broj koraka po okretaju, brzina, akceleracija, faktor mikrokoraka i vrijednosti stupnjeva zakreta servo motora za podizanje i spuštanje olovke. Mikroupravljač Arduino Uno ima tzv. EEPROM: memoriju čije se vrijednosti čuvaju kada je ploča isključena (poput malenog tvrdog diska)[48]. Tip podatka je const byte što

znači da se te vrijednosti nakon postavljanja ne mogu mijenjati, već samo čitati. Tip podatka byte pohranjuje 8-bitni broj bez predznaka, od 0 do 255.

Ovisno o tipu podatka koji će se pohranjivati, potrebno je ostaviti dovoljno prostora za pohranu. Arduino Uno ima 512 bajta EEPROM memorije. Svaka EEPROM adresa može pohraniti bajt memorije. Većina vrijednosti su tipa podatka float te sadrže 4 bajta, stoga potrebno je ostaviti 4 adrese prostora između svakog od njih. Tip podatka int pohranjuje vrijednost 2 bajta. To je primjerice stupanj zakreta servo motora (EEPROM_PENLIFT_DOWN, EEPROM_PENLIFT_UP).

Nakon postavljanja EEPROM adresa, potrebno je definirati varijable i njihove vrijednosti kako bi robotski crtač ispravno funkcionirao. Varijable koje će biti definirane u narednom bloku su: upPosition (stupanj zakreta servo motora za podignutu olovku) downPosition (stupanj zakreta servo motora za spuštenu olovku), penLiftSpeed (broj mikro sekundi između promjene za jedan stupanj), motorStepsPerRev (broj koraka po jednom okretaju koračnog motora), mmPerRev (pomak po jednom okretaju koračnog motora) stepMultiplier (varijabla za izračun mikrokoraka), machineWidth (udaljenost između koračnih motora), machineHeight (visina početne pozicije olovke) currentMaxSpeed (maksimalna brzina), currentAcceleration (trenutačno ubrzanje) i usingAcceleration (korištenje ubrzanja). Servo penHeight kreira objekt servo knjižnice. Taj objekt služi za upravljanje servo motora prilikom rada robotskog crtača.

Kreiraju se dodatne varijable koje se koriste za robotskog crtača koje su tip podatka char. Char može biti samo jedan znak ili ako se želi više njih postaviti u ovaj tip podatka, potrebno ga je definirati kao polje. Kao što je prethodno pokazano, naredba se sastoji od tipa naredbe i maksimalno 4 parametra. Definiraju se četiri char varijable za pohranjivanje parametara. Također, definira se dodatna byte varijabla koja služi za brojanje parametara. Kada se koristi funkcija ili globalna varijabla koja je definirana u drugoj datoteci, potrebno je definirati ključnom riječju extren. Ona govori kompajleru da će povezivač pronaći adresu funkcije ili varijable među različitim datotekama uključenim u vezu. U ovom slučaju potrebno je to napraviti sa objektima knjižnice AccelStepper.

Svaka Arduino skica mora imati funkciju postavljanja. Ova funkcija definira početno stanje Arduina prilikom pokretanja i pokreće se samo jednom. Tada će pokrenuti funkcije postavljanja elektronskog sučelja i priključaka, te funkciju očitavanja podataka iz EEPROM memorije. Tada će objektima AccelStepper knjižnice (u ovom slučaju koračni motori A i B koje robotski crtač koristi) postaviti maksimalnu brzinu i ubrzanje. U varijablu startLenght se

pohranjuje udaljenost olovke od koračnih motora u koracima. Ta vrijednost služi kao nulta pozicija olovke i služi za daljnje izračune pri kretanju olovke. Poziva se funkcija comms_ready(). Poziva se funkcija penlift_penUp() kako bi se olovka podigla od ploče za crtanje. Postavljanje robotskog crtača je završeno.

```
void setup()
{
  configuration_motorSetup();
  eeprom_loadMachineSpecFromEeprom();
  configuration_setup();

  motorA.setMaxSpeed(currentMaxSpeed);
  motorA.setAcceleration(currentAcceleration);
  motorB.setMaxSpeed(currentMaxSpeed);
  motorB.setAcceleration(currentAcceleration);

  float startLength = ((float)startLengthMM/(float)mmPerRev)*(float)motorStepsPerRev;
  motorA.setCurrentPosition(startLength);
  motorB.setCurrentPosition(startLength);
  for (int i = 0; i<INLENGTH; i++) {
    lastCommand[i] = 0;
  }
  comms_ready();
#ifdef PENLIFT
  penlift_penUp();
#endif
  delay(500);
}
```

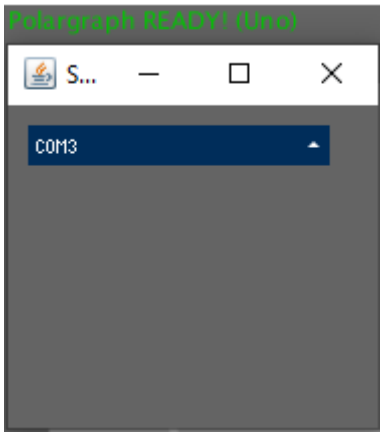
U funkciji loop() se izvršava logika programa. Kada funkcija comms_waitForNextCommand() vrati boolean vrijednost true, izvršava se funkcija comms_parseAndExecuteCommand(). Funkcija comms_waitForNextCommand() služi kako se ne bi krenulo u izvršavanje nove naredbe, bez izvršavanja prethodne. Tek kada se sve naredbe odrade, funkcija comms_parseAndExecuteCommand() može krenuti u novo izvršavanje naredbi. Funkcija comms_waitForNextCommand() i comms_parseAndExecuteCommand() su dio datoteke communication.ino koja služi za serijsku komunikaciju između računala i Arduina. Funkcija comms_parseAndExecuteCommand() prihvaća naredbe te izvlači pojedinačne parametre koje sprema u zasebne varijable. Kada završi pohranjivanje parametara, govori datoteci implement_uno.ino da je naredba spremna da se izvrši. Datoteka implement_uno.ino tada pohranjuje određeni tip naredbe u varijablu com. Napokon, execute_command.ino može krenuti u izvršavanje naredbi. Funkcija loop() se izvršava ciklički sve dok postoje naredbe koje je potrebno izvršiti.

11. CRTANJE

Nakon opisa načina i strukture koda robotskog crtača za ispis fotografija u vertikalnoj ravnini, jedino što je ostalo je ispisati nekoliko fotografija. Treba poduzeti nekoliko koraka prije nego što se može ispisati slika. Prvo, potrebno je povezati robotski crtač pomoću USB kabela s računalom. Tada je potrebno namjestiti sve informacije i dimenzije robotskog crtača te ih učitati na Arduino upravljač. Kada je ta radnja završena, učitava se slika za obradu u GUI te se odabiru različite opcije kako bi se slika prilagodila robotskom crtaču. Nakon svih obavljenih radnji, crtanje može započeti.

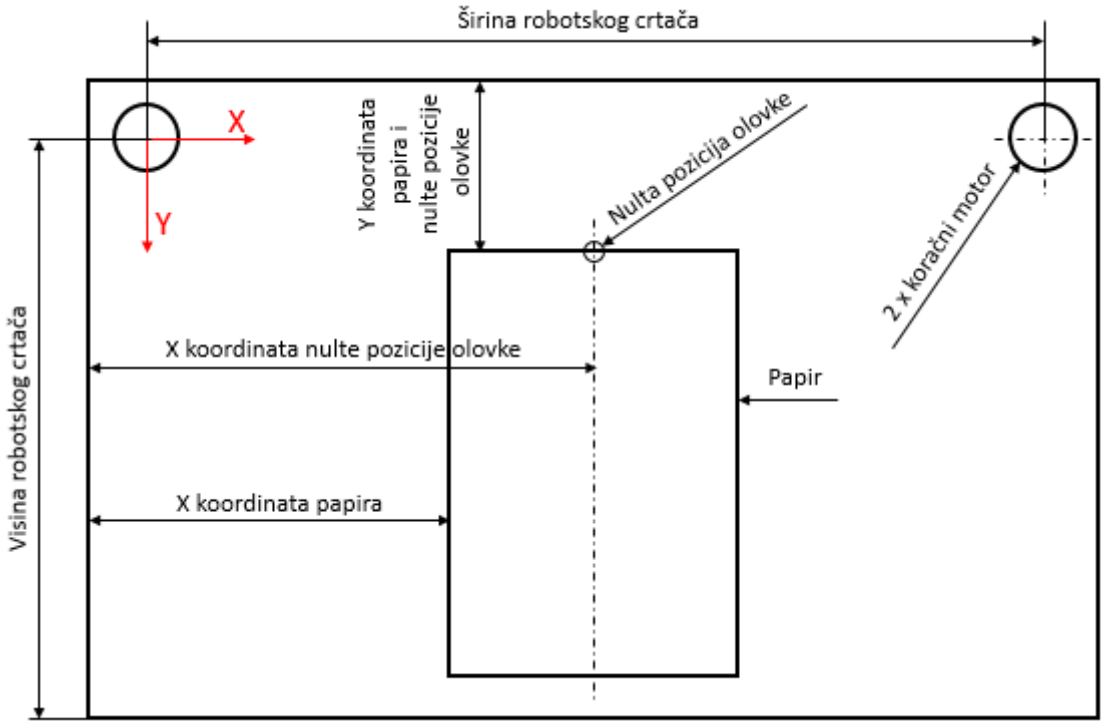
11.1. Priključivanje i postavljanje robotskog crtača

Prva radnja je pokretanje reda naredbi klikom miša na zaglavlje. Arduino se priključuje na USB COM3 priključak na računalu te se datoteka main pokreče preko ikone Prenesi u Arduino IDE–u. Preko GUI–a robotskog crtača pomoću funkcije SERIAL PORT… u SETUP kartici odabire se serijski priključak na koji je spojen Arduino. Ako je odabran ispravan priključak, GUI će signalizirati da je Arduino spreman za komunikaciju.



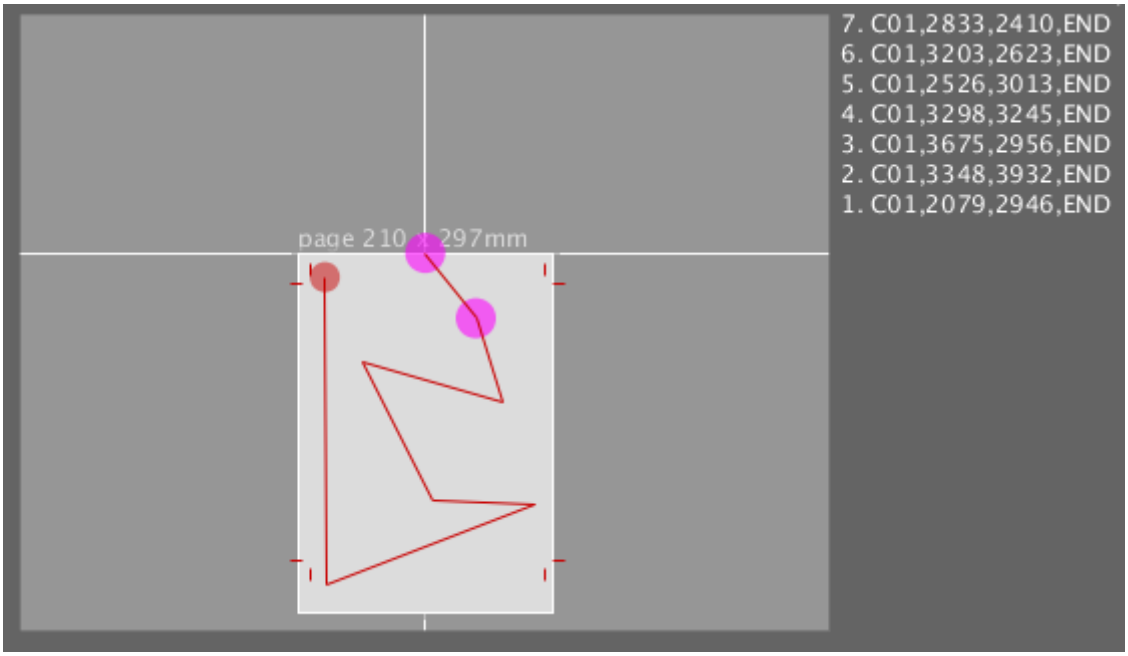
Slika 60. Odabir serijskog priključka

U kartici SETUP se namještaju potrebne dimenzije i informacije o robotskog crtaču. Širina robotskog crtača je 670 mm, dok je visina 510 mm. Papir na kojem će se crtati je A4. Stoga, širina je 210 mm, dok je visina 297 mm. X i Y koordinate papira su 230 mm i 197 mm. Nulta pozicija olovke se nalazi na koordinatama (335 mm, 197 mm). Ishodište se nalazi u gornjem lijevom kutu robotskog crtača. Stupanj servo motora za spuštenu olovku je 10°, dok je za podignutu 105°. Širina olovke je 0,3 mm. Maksimalna brzina motora je 160 koraka po sekundi, dok je ubrzanje 300 koraka po sekundi na kvadrat. Olovka robotskog crtača se ručno dovodi do X i Y koordinate nulte pozicije robota, kako bi se osigurala ispravnost crtanja.



Slika 61. Dimenzije robotskog crtača

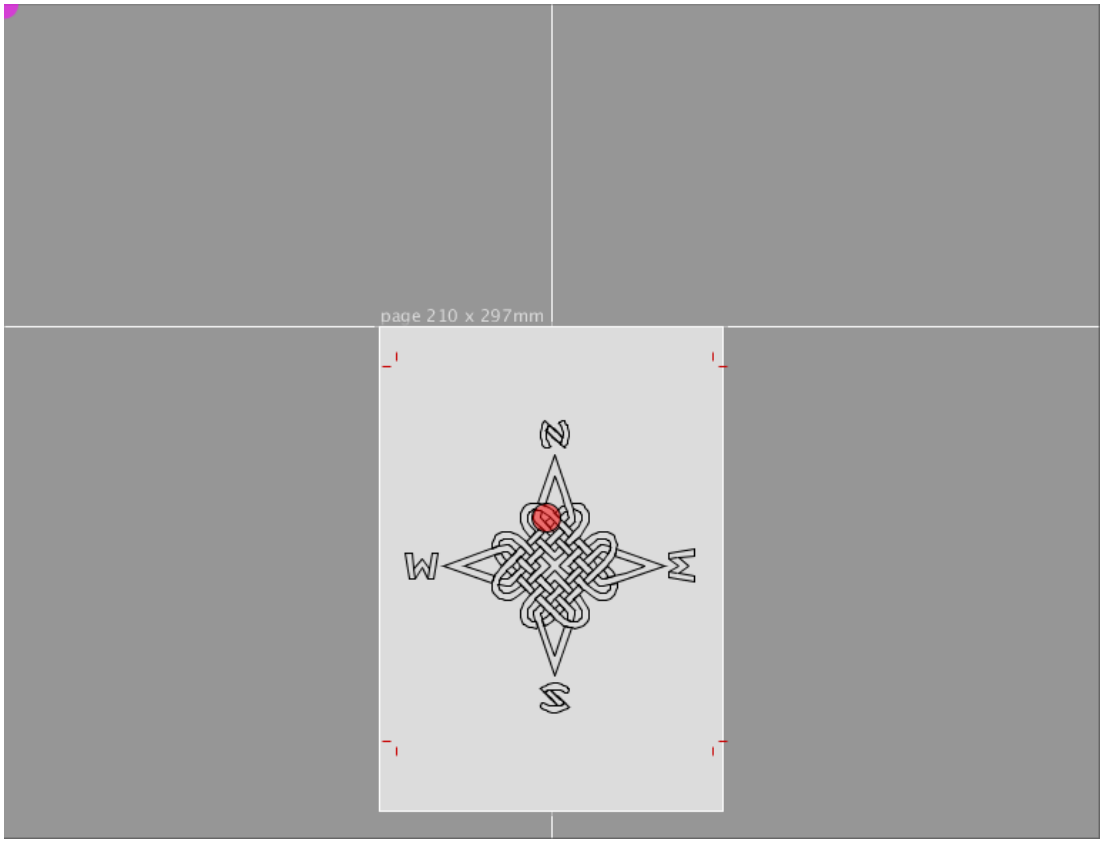
Funkcijom UPLOAD MACHINE SPEC se učitavaju sve navedene vrijednosti za korištenje u Arduino. Koriste se naredbe: C32,300,1,END, C31,160,1,END, C45,10,105,1,END, C37,1,END, C30,200,END C29,32,END i C24,670,510,END. Kada se olovka ručno postavi na nultu poziciju, funkcijom SET HOME postavlja se u softveru nulta pozicija. Šalje se naredba C09,2429,2429,END, što znači da je nulta pozicija udaljena od lijevog i desnog koračnog motora 2429 koraka. Prije crtanja može se provjeriti rad koračnih motora. Funkcijom MOVE PEN TO POINT mogu se odabrati točke na papiru kojima vrh alata robotskog crtača mora pristupiti. Naredbom C01 se postiže ta funkcionalnost. Na slici ispod se može vidjeti proizvoljna putanja te udaljenosti od motora u koracima svake točke.



Slika 62. Testiranje kretanja koračnih motora

11.2. Crtanje učitanih slika

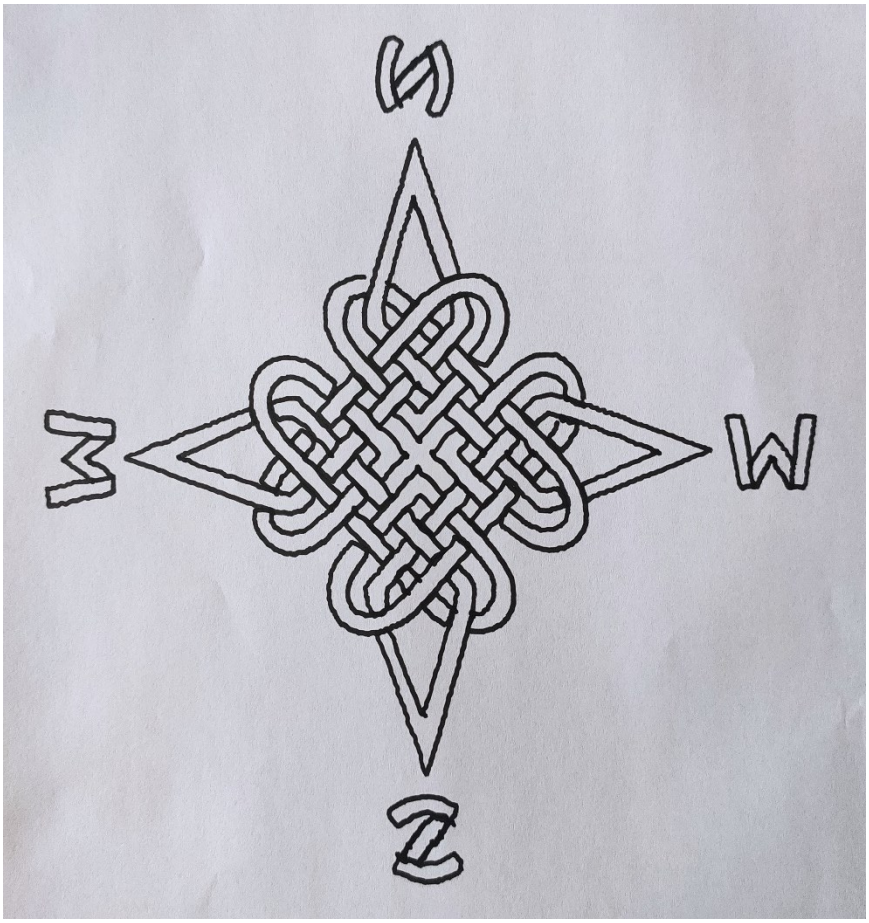
Nakon postavljanja ključnih vrijednosti za ispravan rad robotskog crtača, moguće je postaviti sliku kako bi se obradila pomoću robota. Potrebno je nekoliko koraka za učitavanje i oblikovanje ispisa. Pod karticom INPUT, funkcijom LOAD VECTOR se učitava slika SVG formata. Učitana slika se postavlja i skalira na papir pomoću funkcija MOVE VECTOR i RESIZE VECTOR. Također može se odrediti i sama rezolucija vektora. Funkcija POLYGONIZER LENGTH omogućava postavljanje minimalne duljine linije koja je moguća. Što je manja duljina linije, to se bolje aproksimiraju zakrivljene linije same vektorske slike. No to znači da će trebati duže vremena da se pojedina slika obradi.



Slika 63. Pripremljena slika za crtanje

Jedino što je preostalo je pokrenuti funkciju DRAW VECTOR koja započinje crtanje. Crtanje je trajalo 5 minuta i 37 sekundi. Rezultat crtanja je:

Slika 64. Izvorna SVG slika



Slika 65. Rezultat crtanja robotskog crtača

Za usporedbu će se prikazati još jedna fotografija. Sljedeća slika se ubacila u GUI i preko kartice TRACE se prilagodila za crtanje. Kartica TRACE omogućuje ubacivanje slike koja nije SVG formata u GUI. Potrebno je napomenuti kako je dobiveni rezultat zrcaljen. Ovo se dešava ako priključci desnog i lijevog motora su okrenuti i zamijenjeni na CNC V3 štitu. To ni u kojem slučaju ne izobličava sliku.

Slika 66. Prikaz rezultata crtanja slike ruže

Može se zamijetiti kako robotski crtač ne može obojati pojedine površine. SVG format naredbom fill može obojati zatvorene konture. Kako robotski crtač nema funkciju koja ispunjava površine, njegov algoritam detektira samo rubove na slici SVG formata.

11.3. Utjecaj broja koraka na crteže

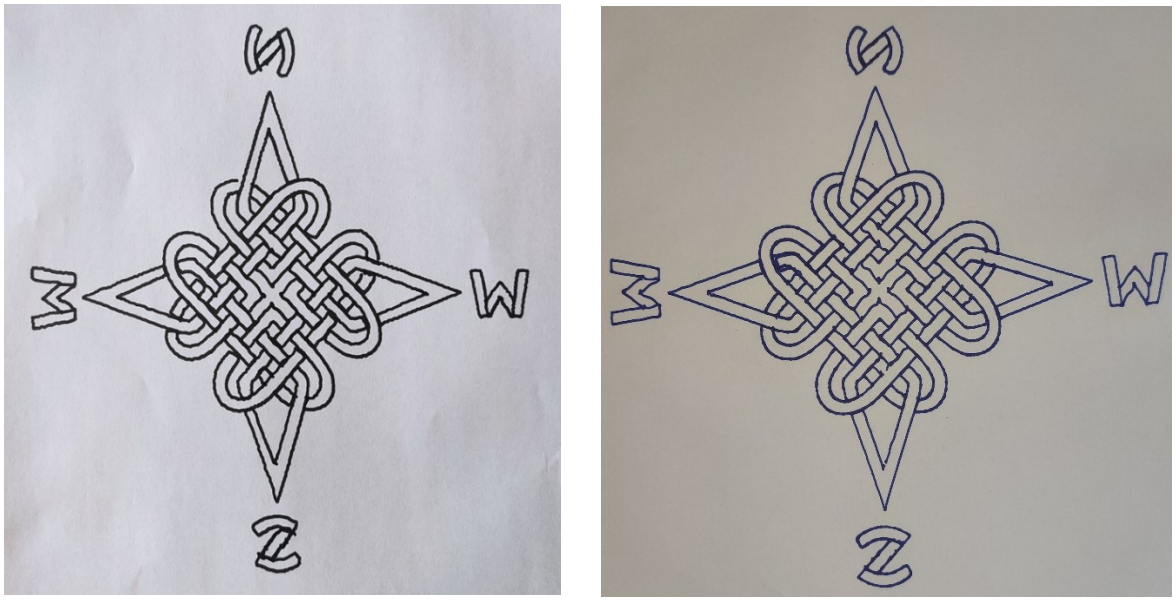
Koračni motori imaju neželjeni učinak koji se naziva rezonancija. Rezonancija može dovesti do preskakanja koraka i gubitka sinkronizacije. Ova pojava se događa ako frekvencija koraka se podudara s prirodnom rezonantnom frekvencijom rotora motora. Kada motor napravi korak, rotor ne dolazi odmah u novi položaj, već pravi prigušene oscilacije. Frekvencija osciliranja ovisi o momentu tromosti rotora plus opterećenju i magnetskom polju. Zbog složene konfiguracije magnetskog polja, rezonantna frekvencija rotora ovisi o tim amplitudama oscilacija[50].

U praksi, pojava rezonancije dovodi do poteškoća radeći na frekvencijama bliskoj rezonantnoj frekvenciji. Rezonancija može značajno narušiti točnost pogona. Postoji rizik od gubitka koraka u sustavima s niskim prigušenjem ili povećane buke kada motor radi blizu

rezonantne frekvencije. Uz vibracije, učinak rezonancije dovodi do gubitka momenta na osovini koračnog motora u određenom rasponu brzina[50].

Jedna od najjednostavnijih metoda rješavanja vibracija motora je korištenje mikro-koračnog načina rada. Dvije faze istovremeno uključene, ali njihove struje nisu jednake, te ravnotežni položaj rotora neće biti u sredini koraka nego na drugom mjestu, određenom omjerom faznih struja. Promjenom ovog omjera može se osigurati određeni broj mikrokoraka unutar jednog koraka. Korištenje mikrokoračnog načina omogućuje izglađivanje vibracija[50].

Broj koraka uvelike utječe na vibracije koračnih motora. Što je broj koraka veći , prijelaz s koraka na korak će biti glađi, a time i manje vibracije. Postavljanjem kratkospojnika na MS1 priključak na CNC V3 štitu, broj koraka po okretaju se povećava četiri puta. Broj koraka je sada 800. Na slijedećoj slici je prikazana usporedba crteža crtanog sa 200 koraka po okretaju naspram 800 koraka po okretaju.



Slika 67. Usporedba crteža sa različitim brojem koraka po okretaju

12. ZAKLJUČAK

U ovom diplomskom radu izrađen je robotski crtač u vertikalnoj ravnini za ispis fotografija. Glavna prednost ovakvog crtača je njegova jednostavnost i jeftinija izrada s obzirom na crtače koji izvode rad u horizontalnoj ravnini. Izrada robotskog crtača za ispis fotografija u vertikalnoj ravnini zahtijevala je znanja iz različitih dijelova znanosti kako bi se rad mogao objediniti. Crtač je morao biti osmišljen, konstruiran, izrađen te montiran. Montaža je uključivala spajanje nestandardnih komponenti koje su izrađene FDM postupkom sa standardiziranim elektronskim i mehaničkim dijelovima. Iako je Arduino platforma koja je izabrana za računalnu podršku jednostavna za korištenje i nije potrebno duboko znanje o elektronici i programiranju, jer postoje mnoge bibliografske reference, trebao se pronaći način kako ostvariti komunikaciju između knjižnica za upravljanje koračnim motorima i elektroničkog sklopovlja. GUI robotskog crtača je pomogao u postavljanju dimenzija i ostalih informacija za ispravan rad crtača. Slike su ubačene u GUI te su rezultati evaluirani nakon što je završeno crtanje. Robotski crtač za ispis fotografija u vertikalnoj ravnini je spoj mehaničke, elektronske, matematičke i računalne znanosti, te kao takav predstavlja multidisciplinarni rad koji se može nositi sa kompliciranim zadatkom kao što je crtanje.

LITERATURA

[1] Definicija robotike: https://builtin.com/robotics, Pristupljeno: 1. listopada 2022.

[2] Definicija g -koda: https://howtomechatronics.com/tutorials/g-code-explained-list-of-most-important-g-code-commands/, Pristupljeno: 1. listopada 2022.

[3] Razlika između CNC stroja i robota :https://robodk.com/blog/difference-robots-cnc-machines/, Pristupljeno: 2. listopada 2022.

[4] 5- osna glodalica: https://www.directindustry.com/prod/comi-spa/product-50365-1833052.html, Pristupljeno: 2. listopada 2022.

[5] FANUC LR Mate 200iD : https://www.fanuc.eu/hu/en/robots/robot-filter-page/lrmate-series/lrmate-200-id, Pristupljeno: 2. listopada 2022.

[6] Dey A, Yodo N. A systematic survey of FDM process parameter optimization and their influence on part characteristics. Journal of Manufacturing and Materials Processing. 2019;3(3):64. doi:10.3390/jmmp3030064

[7] Montero M., Roundy S., Odell D., Ahn S. H., Wright P. K. Material characterization of fused deposition modeling (FDM) ABS by designed experiments. Society of Manufacturing Engineers, 2001, 10(13552540210441166), 1-21.

[8] Shema FDM procesa: https://engineeringproductdesign.com/knowledge-base/fused-deposition-modeling/, Pristupljeno: 4. listopada 2022.

[9] ABS: https://reprap.org/wiki/ABS, Pristupljeno: 4. listopada 2022.

[10] Svojstva ABS-a: https://dielectricmfg.com/knowledge-base/abs/, Pristupljeno: 4. listopada 2022.

[11] Tehničke specifikacije Prusa i3 MK3+ pisača: https://libraries.uta.edu/services/technology/prusa-i3-mk3, Pristupljeno: 4. listopada 2022.

[12] Prusa i3 MK3+: https://www.aniwaa.com/product/3d-printers/prusa-research-original-prusa-i3-mk3s/, Pristupljeno: 4. listopada 2022.

[13] Potporna struktura: https://all3dp.com/1/3d-printing-support-structures/, Pristupljeno: 4. listopada 2022.

[14] Arduino UNO R3: https://store-usa.arduino.cc/products/arduino-uno-rev3, Pristupljeno: 5. listopada 2022.

[15] Arduino UNO R3 https://www.pololu.com/product/2191, Pristupljeno: 5. listopada 2022.

[16] CNC V3 štit: https://www.nkxmotor.si/hr/shop/krmilnik/breakout-board/cnc-shield-v3-arduino-uno/, Pristupljeno: 6. listopada 2022.

[17] Tehnički podaci CNC V3 štita: https://www.handsontec.com/dataspecs/cnc-3axis-shield.pdf, Pristupljeno: 6. listopada 2022.

[18] Koračni motori: https://learn.adafruit.com/all-about-stepper-motors/what-is-a-stepper-motor, Pristupljeno: 6. listopada 2022.

[19] Tehničke specifikacije NEMA17 HS3401 koračnog motora: https://www.datasheet4u.com/datasheet-pdf/MotionKing/17HS3401/pdf.php?id=928656, Pristupljeno: 8. listopada 2022.

[20] Induktivitet faze. https://linuxcnc.org/docs/html/integrator/steppers.html, Pristupljeno: 8. listopada 2022.

[21] Zakretni moment: https://www.motioncontroltips.com/faq-whats-the-difference-between-detent-torque-and-holding-torque/, Pristupljeno: 8. listopada 2022.

[22] A4988 upravljač: https://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/, Pristupljeno: 8. listopada 2022.

[23] A4988 upravljač: https://www.flyrobo.in/a4988-driver-stepper-motor-driver, Pristupljeno: 8. listopada 2022.

[24] Shema A4988 upravljača: https://www.pololu.com/product/1182, Pristupljeno: 15. studenog 2022.

[25] A4988 upravljač: https://lastminuteengineers.com/a4988-stepper-motor-driver-arduino-tutorial/, Pristupljeno: 8. listopada 2022.

[26] Napajanje s prekidanjem struje: https://www.monolithicpower.com/en/switching-power-supply, Pristupljeno: 9. listopada 2022.

[27] MEAN WELL LRS150-12 Ispravljač: https://www.meanwell-web.com/content/files/pdfs/productPdfs/MW/LRS-150/LRS-150-spec.pdf, Pristupljeno: 10. listopada 2022.

[28] MEAN WELL LRS150-12 Ispravljač: https://www.ronis.hr/ispravljac-meanwell-lrs-150-12/204978/product/, Pristupljeno: 10. listopada 2022.

[29] Servo motor SG90: https://phi-education.com/store/micro-servo-motor-SG90, Pristupljeno: 10. listopada 2022.

[30] Servo motor SG90: https://protosupplies.com/product/servo-motor-micro-sg90/, Pristupljeno: 10. listopada 2022.

[31] Servo motor SG90: https://www.google.com/url?sa=i&url=https%3A%2F%2Fardubotics.eu%2Fen%2Fmotors-actuators%2F1789-towerpro-micro-servo-motor-360-degree-sg90.html&psig=AOvVaw1Ai2pES6u8NuPfVL1XA0NW&ust=1650459511073000&s

ource=images&cd=vfe&ved=0CAwQjRxqFwoTCPjj7fqWoPcCFQAAAAAdAAAAABAD, Pristupljeno: 10. listopada 2022.

[32] Servo motor SG90: https://components101.com/motors/servo-motor-basics-pinout-datasheet, Pristupljeno: 10. listopada 2022.

[33] GT2 remen: https://www.adafruit.com/product/1184, Pristupljeno: 10. listopada 2022.

[34] GT2 remen: https://openbuildspartstore.com/gt2-2m-timing-belt-by-the-foot/, Pristupljeno: 10. listopada 2022.

[35] GT2 remenica: https://www.robotdigg.com/product/59/Rostock-16-Teeth-5mm-Bore-GT2-Pulley, Pristupljeno: 10. listopada 2022.

[36] GT2 remenica: https://hallroad.org/gt2-pulley-16-teeth-bore-5mm-timing-gear-alumium-for-gt2-belt-width-13mm-3d-printer-accessories-in-pakistan.html, Pristupljeno: 10. listopada 2022.

[37] CNC V3 štit: https://www.az-delivery.de/en/products/az-delivery-cnc-shield-v3, Pristupljeno: 14. listopada 2022.

[38] Potencimetar na A4988 upravljaču: https://electropeak.com/learn/interfacing-a4988-stepper-motor-driver-with-arduino/, Pristupljeno: 14. listopada 2022.

[39] Arduino IDE https://www.circuito.io/blog/arduino-code/, Pristupljeno: 17. listopada 2022.

[40] Arduino IDE: https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics#serial-monitor, Pristupljeno: 17. listopada 2022.

[41] Processing: https://processing.org/environment/, Pristupljeno: 17. listopada 2022.

[42] GUI: https://www.heavy.ai/technical-glossary/graphical-user-interface, Pristupljeno: 24. listopada 2022.

[43] ASCII: https://www.techtarget.com/whatis/definition/ASCII-American-Standard-Code-for-Information-Interchange, Pristupljeno: 24. listopada 2022.

[44] Naredbe robotskog crtača: https://github.com/euphy/polargraph/wiki/Polargraph-machine-commands-and-responses, Pristupljeno: 24. listopada 2022.

[45] Vektorska grafika: https://www.techtarget.com/whatis/definition/vector-graphics, Pristupljeno: 3. studenog 2022.

[46] SVG format: https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial/Paths, Pristupljeno: 3. studenog 2022.

[47] AccelStepper knjižnica: https://hackaday.io/project/183279-accelstepper-the-missing-manual/details#header, Pristupljeno: 18. listopada 2022.

[48] EEPROM memorija: https://docs.arduino.cc/learn/built-in-libraries/eeprom, Pristupljeno: 20. listopada 2022.

[49] static ključna riječ: https://www.arduino.cc/reference/en/language/variables/variable-scope-qualifiers/static/, Pristupljeno: 20. listopada 2022.

[50] Vernezi MA, Nazarenko DV, Abderrazzak EH. Vibration suppression of Stepper Motors by the Electric Method. 2021 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM). 2021; doi:10.1109/icieam51226.2021.9446

PRILOG

I. Programski kod

II. Tehnička dokumentacija

configuration.ino

```
#ifdef SERIAL_STEPPER_DRIVERS
#define MOTOR_A_ENABLE_PIN 8
#define MOTOR_A_STEP_PIN 2
#define MOTOR_A_DIR_PIN 5

#define MOTOR_B_ENABLE_PIN 8
#define MOTOR_B_STEP_PIN 3
#define MOTOR_B_DIR_PIN 6
AccelStepper motorA(1,MOTOR_A_STEP_PIN, MOTOR_A_DIR_PIN);
AccelStepper motorB(1,MOTOR_B_STEP_PIN, MOTOR_B_DIR_PIN);
#endif

void configuration_motorSetup()
{
#ifdef SERIAL_STEPPER_DRIVERS
  pinMode(MOTOR_A_ENABLE_PIN, OUTPUT);
  digitalWrite(MOTOR_A_ENABLE_PIN, HIGH);
  pinMode(MOTOR_B_ENABLE_PIN, OUTPUT);
  digitalWrite(MOTOR_B_ENABLE_PIN, HIGH);
  motorA.setEnablePin(MOTOR_A_ENABLE_PIN);
  motorA.setPinsInverted(false, false, true);
  motorB.setEnablePin(MOTOR_B_ENABLE_PIN);
  motorB.setPinsInverted(true, false, true);
#endif
}

void configuration_setup()
{
  mmPerStep = mmPerRev / multiplier(motorStepsPerRev);
  stepsPerMM = multiplier(motorStepsPerRev) / mmPerRev;
}
```

implementation_uno.ino

```
void impl_processCommand(String com)
{
#if MICROCONTROLLER == MC_UNO
  impl_executeCommand(com);
#endif
}

void impl_executeCommand(String &com)
{
  if (exec_executeBasicCommand(com))
  {

  }
  else
  {
    comms_unrecognisedCommand(com);
    comms_ready();
  }
}

void impl_runBackgroundProcesses()
{
  long motorCutoffTime = millis() - lastOperationTime;
  if ((automaticPowerDown) && (powerIsOn) && (motorCutoffTime >
motorIdleTimeBeforePowerDown))
  {

    impl_releaseMotors();
  }
}

void impl_loadMachineSpecFromEeprom()
{}

void impl_engageMotors()
{
  motorA.enableOutputs();
  motorB.enableOutputs();
  powerIsOn = true;
  motorA.runToNewPosition(motorA.currentPosition()+4);
  motorB.runToNewPosition(motorB.currentPosition()+4);
  motorA.runToNewPosition(motorA.currentPosition()-4);
  motorB.runToNewPosition(motorB.currentPosition()-4);
}

void impl_releaseMotors()
{
  motorA.disableOutputs();
  motorB.disableOutputs();

#ifdef PENLIFT
  penlift_penUp();
#endif
  powerIsOn = false;
}
```

pen_utilization.ino

```
#ifdef PENLIFT
void penlift_movePen(int start, int end, int delay_ms)
{
  penHeight.attach(PEN_HEIGHT_SERVO_PIN);
  if(start < end)
  {
    for (int i=start; i<=end; i++)
    {
      penHeight.write(i);
      delay(delay_ms);
    }
  }
  else
  {
    for (int i=start; i>=end; i--)
    {
      penHeight.write(i);
      delay(delay_ms);
    }
  }
  penHeight.detach();
}

void penlift_penUp()
{
  if (inNoOfParams > 1)
  {
    int positionToMoveFrom = isPenUp ? upPosition : downPosition;
    upPosition = atoi(inParam1);
    penlift_movePen(positionToMoveFrom, upPosition, penLiftSpeed);
  }
  else
  {
    if (isPenUp == false)
    {
      penlift_movePen(downPosition, upPosition, penLiftSpeed);
    }
  }
  isPenUp = true;
}

void penlift_penDown()
{
    if (inNoOfParams > 1)
  {
    int positionToMoveFrom = isPenUp ? upPosition : downPosition;
    downPosition = atoi(inParam1);
    penlift_movePen(positionToMoveFrom, downPosition, penLiftSpeed);
  }
  else
  {
    if (isPenUp == true)
    {
      penlift_movePen(upPosition, downPosition, penLiftSpeed);
    }
  }
  isPenUp = false;
}
#endif
```

transformation.ino

```
long multiplier(int in)
{
  return multiplier((long) in);
}
long multiplier(long in)
{
  return in * stepMultiplier;
}
float multiplier(float in)
{
  return in * stepMultiplier;
}
long divider(long in)
{
  return in / stepMultiplier;
}

void changeLength(long tAl, long tBl)
{
  float tA = float(tAl);
  float tB = float(tBl);
  lastOperationTime = millis();
  float currSpeedA = motorA.speed();
  float currSpeedB = motorB.speed();

  motorA.setSpeed(0.0);
  motorB.setSpeed(0.0);
  motorA.moveTo(tA);
  motorB.moveTo(tB);

  if (!usingAcceleration)
  {
    if (motorA.speed() < 0)
      currSpeedA = -currSpeedA;
    if (motorB.speed() < 0)
      currSpeedB = -currSpeedB;

    motorA.setSpeed(currSpeedA);
    motorB.setSpeed(currSpeedB);
  }

  while (motorA.distanceToGo() != 0 || motorB.distanceToGo() != 0)
  {
    impl_runBackgroundProcesses();
    if (currentlyRunning)
    {
      if (usingAcceleration)
      {
        motorA.run();
        motorB.run();
      }
      else
      {
        motorA.runSpeedToPosition();
        motorB.runSpeedToPosition();
      }
    }
  }
```

```
  reportPosition();
}

void changeLengthRelative(float tA, float tB)
{
  changeLengthRelative((long) tA, (long)tB);
}
void changeLengthRelative(long tA, long tB)
{
  lastOperationTime = millis();
  motorA.move(tA);
  motorB.move(tB);

  while (motorA.distanceToGo() != 0 || motorB.distanceToGo() != 0)
  {

    if (currentlyRunning)
    {
      if (usingAcceleration)
      {
        motorA.run();
        motorB.run();
      }
      else
      {
        motorA.runSpeedToPosition();
        motorB.runSpeedToPosition();
      }
    }
  }

  reportPosition();
}

long getMaxLength()
{
  if (maxLength == 0)
  {

    maxLength = long(getMachineA(pageWidth, pageHeight)+0.5);
    Serial.print(F("maxLength: "));
    Serial.println(maxLength);
  }
  return maxLength;
}


float getMachineA(float cX, float cY)
{
  float a = sqrt(sq(cX)+sq(cY));
  return a;
}
float getMachineB(float cX, float cY)
{
  float b = sqrt(sq((pageWidth)-cX)+sq(cY));
  return b;
}

void moveAxis(AccelStepper &m, int dist)
{
  m.move(dist);
```

```
  while (m.distanceToGo() != 0)
  {
    impl_runBackgroundProcesses();
    if (currentlyRunning)
      m.run();
  }
  lastOperationTime = millis();
}

void reportPosition()
{
  if (reportingPosition)
  {
    Serial.print(OUT_CMD_SYNC_STR);
    Serial.print(divider(motorA.currentPosition()));
    Serial.print(COMMA);
    Serial.print(divider(motorB.currentPosition()));
    Serial.println(CMD_END);
  }
}

float getCartesianXFP(float aPos, float bPos)
{
  float calcX = (sq((float)pageWidth) - sq((float)bPos) + sq((float)aPos))
/ ((float)pageWidth*2.0);
  return calcX;
}
float getCartesianYFP(float cX, float aPos)
{
  float calcY = sqrt(sq(aPos)-sq(cX));
  return calcY;
}

long getCartesianX(float aPos, float bPos)
{
  long calcX = long((pow(pageWidth, 2) - pow(bPos, 2) + pow(aPos, 2)) /
(pageWidth*2));
  return calcX;
}

long getCartesianX() {
  long calcX = getCartesianX(motorA.currentPosition(),
motorB.currentPosition());
  return calcX;
}

long getCartesianY() {
  return getCartesianY(getCartesianX(), motorA.currentPosition());
}
long getCartesianY(long cX, float aPos) {
  long calcY = long(sqrt(pow(aPos,2)-pow(cX,2)));
  return calcY;
}
```

EEPROMAnything.h

```
template <class T> int EEPROM_writeAnything(int ee, const T& value)
{
    const byte* p = (const byte*)(const void*)&value;
    unsigned int i;
    for (i = 0; i < sizeof(value); i++)
          EEPROM.write(ee++, *p++);
    return i;
}

template <class T> int EEPROM_readAnything(int ee, T& value)
{
    byte* p = (byte*)(void*)&value;
    unsigned int i;
    for (i = 0; i < sizeof(value); i++)
          *p++ = EEPROM.read(ee++);
    return i;
}
```

eeprom.ino

```
void eeprom_resetEeprom()
{
  for (int i = 0; i <50; i++)
  {
    EEPROM.write(i, 0);
  }
  eeprom_loadMachineSpecFromEeprom();
}
void eeprom_dumpEeprom()
{
  for (int i = 0; i <40; i++)
  {
    Serial.print(i);
    Serial.print(". ");
    Serial.println(EEPROM.read(i));
  }
}

void eeprom_loadMachineSize()
{
  EEPROM_readAnything(EEPROM_MACHINE_WIDTH, machineWidth);
  if (machineWidth < 1)
  {
    machineWidth = DEFAULT_MACHINE_WIDTH;
  }
  EEPROM_readAnything(EEPROM_MACHINE_HEIGHT, machineHeight);
  if (machineHeight < 1)
  {
    machineHeight = DEFAULT_MACHINE_HEIGHT;
  }
}

void eeprom_loadSpoolSpec()
{
  EEPROM_readAnything(EEPROM_MACHINE_MM_PER_REV, mmPerRev);
  if (mmPerRev < 1)
  {
    mmPerRev = DEFAULT_MM_PER_REV;
  }
  EEPROM_readAnything(EEPROM_MACHINE_STEPS_PER_REV, motorStepsPerRev);
  if (motorStepsPerRev < 1)
  {
    motorStepsPerRev = DEFAULT_STEPS_PER_REV;
  }
}

void eeprom_loadPenLiftRange()
{
  EEPROM_readAnything(EEPROM_PENLIFT_DOWN, downPosition);
  if ((downPosition < 0) || (downPosition > 360))
  {
    downPosition = DEFAULT_DOWN_POSITION;
  }
  EEPROM_readAnything(EEPROM_PENLIFT_UP, upPosition);
  if ((upPosition < 0) || (upPosition > 360))
  {
    upPosition = DEFAULT_UP_POSITION;
  }
}
```

```
void eeprom_loadStepMultiplier()
{
  EEPROM_readAnything(EEPROM_MACHINE_STEP_MULTIPLIER, stepMultiplier);
  if (stepMultiplier < 1)
  {
    stepMultiplier = DEFAULT_STEP_MULTIPLIER;
  }
}

void eeprom_loadSpeed()
{
  EEPROM_readAnything(EEPROM_MACHINE_MOTOR_SPEED, currentMaxSpeed);
  if (int(currentMaxSpeed) < 1) {
    currentMaxSpeed = 800.0;
  }

  EEPROM_readAnything(EEPROM_MACHINE_MOTOR_ACCEL, currentAcceleration);
  if (int(currentAcceleration) < 1) {
    currentAcceleration = 800.0;
  }
}

void eeprom_loadMachineSpecFromEeprom()
{
  impl_loadMachineSpecFromEeprom();

  eeprom_loadMachineSize();
  eeprom_loadSpoolSpec();
  eeprom_loadStepMultiplier();
  eeprom_loadPenLiftRange();
  eeprom_loadSpeed();
  EEPROM_readAnything(EEPROM_MACHINE_PEN_WIDTH, penWidth);
  if (penWidth < 0.0001){
    penWidth = 0.8;
  }
  mmPerStep = mmPerRev / multiplier(motorStepsPerRev);
  stepsPerMM = multiplier(motorStepsPerRev) / mmPerRev;
  pageWidth = machineWidth * stepsPerMM;
  pageHeight = machineHeight * stepsPerMM;
  maxLength = 0;
}
```

communication.ino

```
boolean comms_waitForNextCommand(char *buf)
{
  long idleTime = millis();
  int bufPos = 0;
  for (int i = 0; i<INLENGTH; i++) {
    buf[i] = 0;
  }
  long lastRxTime = 0L;
  boolean terminated = false;
  while (!terminated)
  {
    long timeSince = millis() - lastRxTime;
    if (bufPos != 0 && timeSince > 100)
    {
      for (int i = 0; i<INLENGTH; i++) {
        buf[i] = 0;
      }
      bufPos = 0;
    }
    impl_runBackgroundProcesses();
    timeSince = millis() - idleTime;
    if (timeSince > rebroadcastReadyInterval)
    {
      comms_ready();
      idleTime = millis();
    }
    if (Serial.available() > 0)
    {
      char ch = Serial.read();
      if (ch == INTERMINATOR || ch == SEMICOLON) {
        buf[bufPos] = 0;
        terminated = true;
        for (int i = bufPos; i<INLENGTH-1; i++) {
          buf[i] = 0;
        }

      } else {
        buf[bufPos] = ch;
        bufPos++;
      }
      lastRxTime = millis();
    }
  }
  idleTime = millis();
  lastOperationTime = millis();
  lastInteractionTime = lastOperationTime;
  return true;
}

void comms_parseAndExecuteCommand(char *inS)
{
  boolean commandParsed = comms_parseCommand(inS);
  if (commandParsed)
  {
    impl_processCommand(lastCommand);
    for (int i = 0; i<INLENGTH; i++) { inS[i] = 0; }
    commandConfirmed = false;
    comms_ready();
  }
```

```
  else
  {
    Serial.print(MSG_E_STR);
    Serial.print(F("Comm ("));
    Serial.print(inS);
    Serial.println(F(") not parsed."));
  }
  inNoOfParams = 0;
}

boolean comms_parseCommand(char *inS)
{
  char* sub = strstr(inS, CMD_END);
  sub[strlen(CMD_END)] = 0;
  if (strcmp(sub, CMD_END) == 0)
  {
    comms_extractParams(inS);
    return true;
  }
  else
    return false;
}

void comms_extractParams(char* inS)
{
  char in[strlen(inS)];
  strcpy(in, inS);
  char * param;
  byte paramNumber = 0;
  param = strtok(in, COMMA);

  inParam1[0] = 0;
  inParam2[0] = 0;
  inParam3[0] = 0;
  inParam4[0] = 0;

  for (byte i=0; i<6; i++) {
      if (i == 0) {
        strcpy(inCmd, param);
      }
      else {
        param = strtok(NULL, COMMA);
        if (param != NULL) {
          if (strstr(CMD_END, param) == NULL) {
            paramNumber++;
          }
        }
        switch(i)
        {
          case 1:
            if (param != NULL) strcpy(inParam1, param);
            break;
          case 2:
            if (param != NULL) strcpy(inParam2, param);
            break;
          case 3:
            if (param != NULL) strcpy(inParam3, param);
            break;
          case 4:
            if (param != NULL) strcpy(inParam4, param);
            break;
```

```
          default:
            break;
        }
      }
  }
  inNoOfParams = paramNumber;
}
void comms_ready()
{
  Serial.println(F(READY_STR));
}
void comms_drawing()
{
  Serial.println(F(DRAWING_STR));
}
void comms_requestResend()
{
  Serial.println(F(RESEND_STR));
}
void comms_unrecognisedCommand(String &com)
{
  Serial.print(MSG_E_STR);
  Serial.print(com);
  Serial.println(F(" not recognised."));
}
```

execute_command.ino

```
boolean exec_executeBasicCommand(String &com)
{
  boolean executed = true;
  if (com.startsWith(CMD_CHANGELENGTH))
    exec_changeLength();
#ifdef VECTOR_LINES
  else if (com.startsWith(CMD_CHANGELENGTHDIRECT))
    exec_changeLengthDirect();
#endif
  else if (com.startsWith(CMD_CHANGEPENWIDTH))
    exec_changePenWidth();
  else if (com.startsWith(CMD_SETMOTORSPEED))
    exec_setMotorSpeed();
  else if (com.startsWith(CMD_SETMOTORACCEL))
    exec_setMotorAcceleration();
  else if (com.startsWith(CMD_SETPOSITION))
    exec_setPosition();
#ifdef PENLIFT
  else if (com.startsWith(CMD_PENDOWN))
    penlift_penDown();
  else if (com.startsWith(CMD_PENUP))
    penlift_penUp();
  else if (com.startsWith(CMD_SETPENLIFTRANGE))
    exec_setPenLiftRange();
#endif
  else if (com.startsWith(CMD_SETMACHINESIZE))
    exec_setMachineSizeFromCommand();
  else if (com.startsWith(CMD_SETMACHINEMMPERREV))
    exec_setMachineMmPerRevFromCommand();
  else if (com.startsWith(CMD_SETMACHINESTEPSPERREV))
    exec_setMachineStepsPerRevFromCommand();
  else if (com.startsWith(CMD_SETMACHINESTEPMULTIPLIER))
    exec_setMachineStepMultiplierFromCommand();
  else if (com.startsWith(CMD_GETMACHINEDETAILS))
    exec_reportMachineSpec();
  else if (com.startsWith(CMD_RESETEEPROM))
    eeprom_resetEeprom();
  else
    executed = false;

  return executed;
}
void exec_reportMachineSpec()
{
  eeprom_dumpEeprom();
}

void exec_setMachineSizeFromCommand()
{
  int width = atoi(inParam1);
  int height = atoi(inParam2);

  int currentValue = width;
  EEPROM_readAnything(EEPROM_MACHINE_WIDTH, currentValue);
  if (currentValue != width)
    if (width > 10)
    {
      EEPROM_writeAnything(EEPROM_MACHINE_WIDTH, width);
    }
```

```
  EEPROM_readAnything(EEPROM_MACHINE_HEIGHT, currentValue);
  if (currentValue != height)
    if (height > 10)
    {
      EEPROM_writeAnything(EEPROM_MACHINE_HEIGHT, height);
    }
  eeprom_loadMachineSize();
}

void exec_setMachineMmPerRevFromCommand()
{
  EEPROM_writeAnything(EEPROM_MACHINE_MM_PER_REV, (float)atof(inParam1));
  eeprom_loadMachineSpecFromEeprom();
}
void exec_setMachineStepsPerRevFromCommand()
{
  EEPROM_writeAnything(EEPROM_MACHINE_STEPS_PER_REV, atoi(inParam1));
  eeprom_loadMachineSpecFromEeprom();
}
void exec_setMachineStepMultiplierFromCommand()
{
  EEPROM_writeAnything(EEPROM_MACHINE_STEP_MULTIPLIER, atoi(inParam1));
  eeprom_loadMachineSpecFromEeprom();
}

void exec_setPenLiftRange()
{
  int down = atoi(inParam1);
  int up = atoi(inParam2);

  if (inNoOfParams == 3)
  {
    EEPROM_writeAnything(EEPROM_PENLIFT_DOWN, down);
    EEPROM_writeAnything(EEPROM_PENLIFT_UP, up);
    eeprom_loadPenLiftRange();
  }
  else if (inNoOfParams == 2)
  {
    penlift_movePen(up, down, penLiftSpeed);
    delay(200);
    penlift_movePen(down, up, penLiftSpeed);
    delay(200);
    penlift_movePen(up, down, penLiftSpeed);
    delay(200);
    penlift_movePen(down, up, penLiftSpeed);
    delay(200);
  }
}

void exec_setMotorSpeed()
{
  exec_setMotorSpeed((float)atof(inParam1));
  if (inNoOfParams == 2 && atoi(inParam2) == 1)
    EEPROM_writeAnything(EEPROM_MACHINE_MOTOR_SPEED, currentMaxSpeed);
}

void exec_setMotorSpeed(float speed)
{
  currentMaxSpeed = speed;
  motorA.setMaxSpeed(currentMaxSpeed);
```

```
  motorB.setMaxSpeed(currentMaxSpeed);
}

void exec_setMotorAcceleration()
{
  exec_setMotorAcceleration((float)atof(inParam1));
  if (inNoOfParams == 2 && atoi(inParam2) == 1)
    EEPROM_writeAnything(EEPROM_MACHINE_MOTOR_ACCEL, currentAcceleration);
}
void exec_setMotorAcceleration(float accel)
{
  currentAcceleration = accel;
  motorA.setAcceleration(currentAcceleration);
  motorB.setAcceleration(currentAcceleration);
}

void exec_changePenWidth()
{
  penWidth = atof(inParam1);
}

void exec_setPosition()
{
  long targetA = multiplier(atol(inParam1));
  long targetB = multiplier(atol(inParam2));
  motorA.setCurrentPosition(targetA);
  motorB.setCurrentPosition(targetB);

  impl_engageMotors();

  reportPosition();
}

void exec_changeLengthRelative()
{
  long lenA = multiplier(atol(inParam1));
  long lenB = multiplier(atol(inParam2));

  changeLengthRelative(lenA, lenB);
}

void exec_changeLength()
{
  float lenA = multiplier((float)atof(inParam1));
  float lenB = multiplier((float)atof(inParam2));

  changeLength(lenA, lenB);
}

#ifdef VECTOR_LINES
void exec_changeLengthDirect()
{
  float endA = multiplier((float)atof(inParam1));
  float endB = multiplier((float)atof(inParam2));
  int maxSegmentLength = atoi(inParam3);

  float startA = motorA.currentPosition();
  float startB = motorB.currentPosition();

  if (endA < 20 || endB < 20 || endA > getMaxLength() || endB >
getMaxLength())
```

```
  {
  }
  else
  {
    exec_drawBetweenPoints(startA, startB, endA, endB, maxSegmentLength);
  }
}

void exec_drawBetweenPoints(float p1a, float p1b, float p2a, float p2b, int
maxSegmentLength)
{

  float c1x = getCartesianXFP(p1a, p1b);
  float c1y = getCartesianYFP(c1x, p1a);

  float c2x = getCartesianXFP(p2a, p2b);
  float c2y = getCartesianYFP(c2x, p2a);

  if (c2x > 20
    && c2x<pageWidth-20
    && c2y > 20
    && c2y <pageHeight-20
    && c1x > 20
    && c1x<pageWidth-20
    && c1y > 20
    && c1y <pageHeight-20
    )
    {
    reportingPosition = false;
    float deltaX = c2x-c1x;
    float deltaY = c2y-c1y;

    int linesegs = 1;
    if (abs(deltaX) > abs(deltaY))
    {
      while ((abs(deltaX)/linesegs) > maxSegmentLength)
      {
        linesegs++;
      }
    }
    else
    {
      while ((abs(deltaY)/linesegs) > maxSegmentLength)
      {
        linesegs++;
      }
    }

    deltaX = deltaX/linesegs;
    deltaY = deltaY/linesegs;
    long runSpeed = 0;

    usingAcceleration = false;
    while (linesegs > 0)
    {
      c1x = c1x + deltaX;
      c1y = c1y + deltaY;

      float pA = getMachineA(c1x, c1y);
      float pB = getMachineB(c1x, c1y);
      runSpeed = desiredSpeed(linesegs, runSpeed, currentAcceleration*4);
```

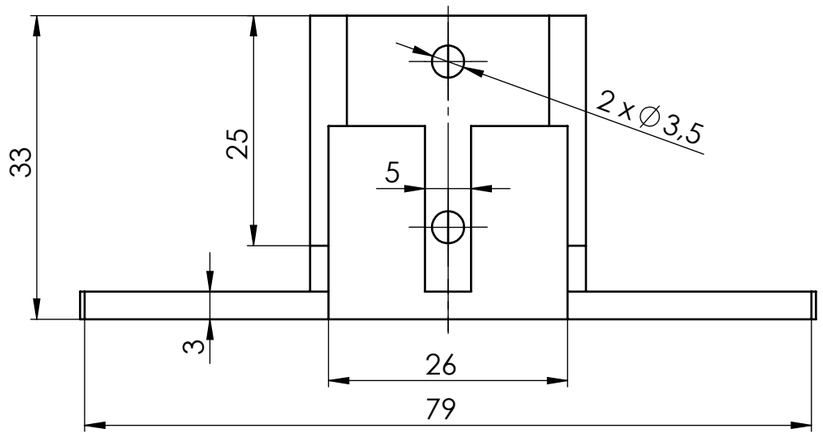
```
      motorA.setSpeed(runSpeed);
      motorB.setSpeed(runSpeed);
      changeLength(pA, pB);

      linesegs--;
    }
    reportingPosition = true;
    usingAcceleration = true;
    reportPosition();
  }
  else
  {

  }
}

float desiredSpeed(long distanceTo, float currentSpeed, float acceleration)
{
    float requiredSpeed;

    if (distanceTo == 0)
    return 0.0f;

    float sqrSpeed = sq(currentSpeed);
    if (currentSpeed < 0.0)
      sqrSpeed = -sqrSpeed;

    float twoa = 2.0f * acceleration;

    if ((sqrSpeed / twoa) < distanceTo)
    {

    if (currentSpeed == 0.0f)
        requiredSpeed = sqrt(twoa);
    else
        requiredSpeed = currentSpeed + fabs(acceleration / currentSpeed);

    if (requiredSpeed > currentMaxSpeed)
        requiredSpeed = currentMaxSpeed;
    }
    else
    {
    if (currentSpeed == 0.0f)
        requiredSpeed = -sqrt(twoa);
    else
        requiredSpeed = currentSpeed - fabs(acceleration / currentSpeed);
    if (requiredSpeed < -currentMaxSpeed)
        requiredSpeed = -currentMaxSpeed;
    }

    return requiredSpeed;
}
#endif
```

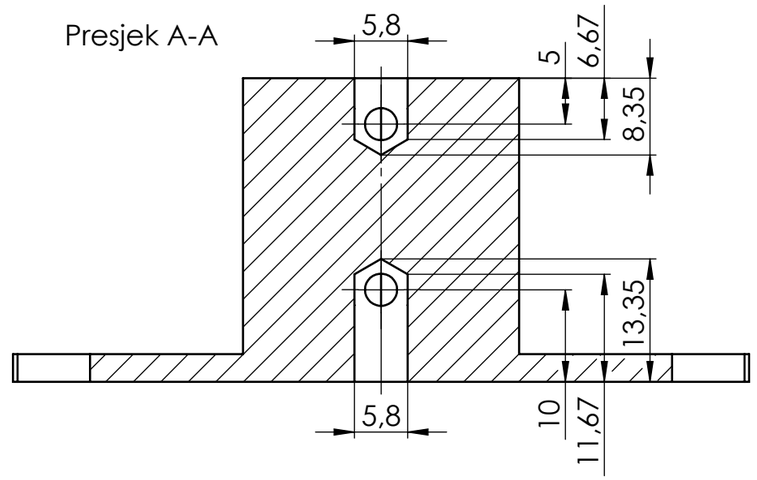
main

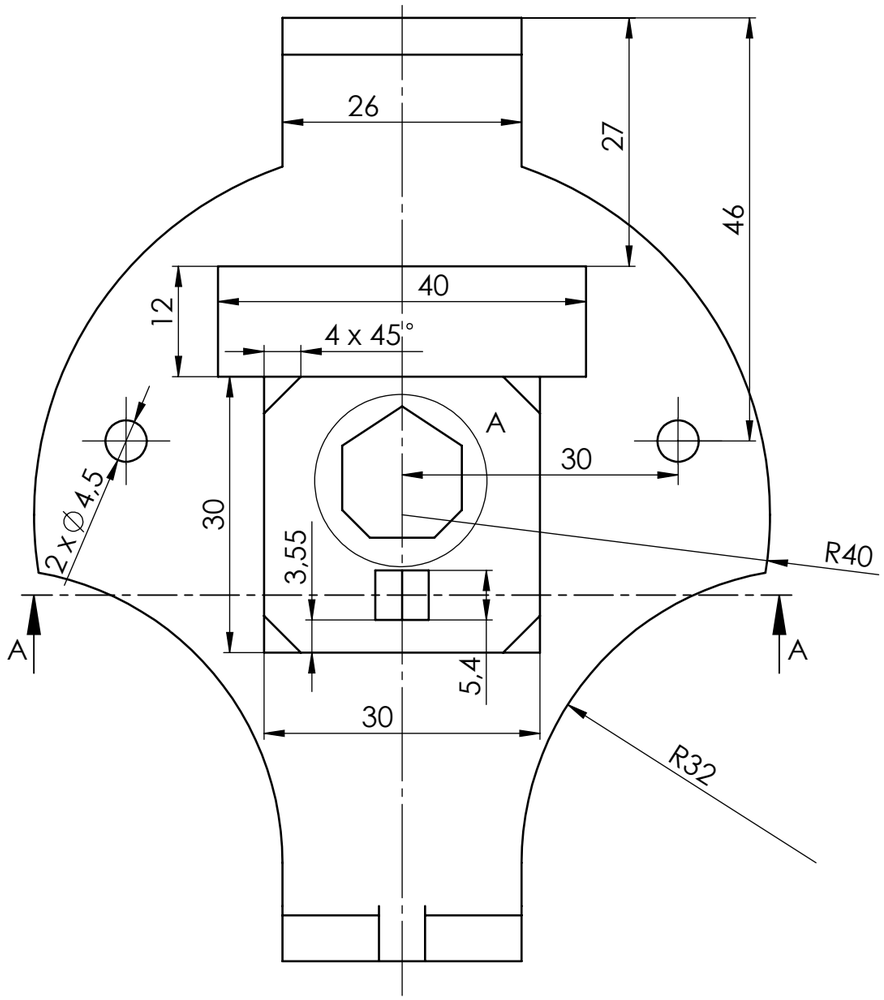
```
/*
Polargraph Server
Written by Sandy Noble
Program was modified
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.
*/

#ifndef MICROCONTROLLER
#define MICROCONTROLLER MC_UNO
#endif
#define SERIAL_STEPPER_DRIVERS
#define PENLIFT
#define VECTOR_LINES

#define MC_UNO 1
#define MC_MEGA 2

#include <AccelStepper.h>
#include <Servo.h>
#include <EEPROM.h>
#include "EEPROMAnything.h"


const byte EEPROM_MACHINE_WIDTH = 0;
const byte EEPROM_MACHINE_HEIGHT = 2;
const byte EEPROM_MACHINE_MM_PER_REV = 14;
const byte EEPROM_MACHINE_STEPS_PER_REV = 18;
const byte EEPROM_MACHINE_STEP_MULTIPLIER = 20;

const byte EEPROM_MACHINE_MOTOR_SPEED = 22;
const byte EEPROM_MACHINE_MOTOR_ACCEL = 26;
const byte EEPROM_MACHINE_PEN_WIDTH = 30;

const byte EEPROM_MACHINE_HOME_A = 34;
const byte EEPROM_MACHINE_HOME_B = 38;

const byte EEPROM_PENLIFT_DOWN = 42;
const byte EEPROM_PENLIFT_UP = 44;

Servo penHeight;
const int DEFAULT_DOWN_POSITION = 90;
const int DEFAULT_UP_POSITION = 180;
static int upPosition = DEFAULT_UP_POSITION;
static int downPosition = DEFAULT_DOWN_POSITION;
static int penLiftSpeed = 3;
const byte PEN_HEIGHT_SERVO_PIN = 9;
boolean isPenUp = false;

const int DEFAULT_MACHINE_WIDTH = 650;
const int DEFAULT_MACHINE_HEIGHT = 650;
const int DEFAULT_MM_PER_REV = 95;
const int DEFAULT_STEPS_PER_REV = 400;
const int DEFAULT_STEP_MULTIPLIER = 1;

static int motorStepsPerRev = DEFAULT_STEPS_PER_REV;
static float mmPerRev = DEFAULT_MM_PER_REV;
```

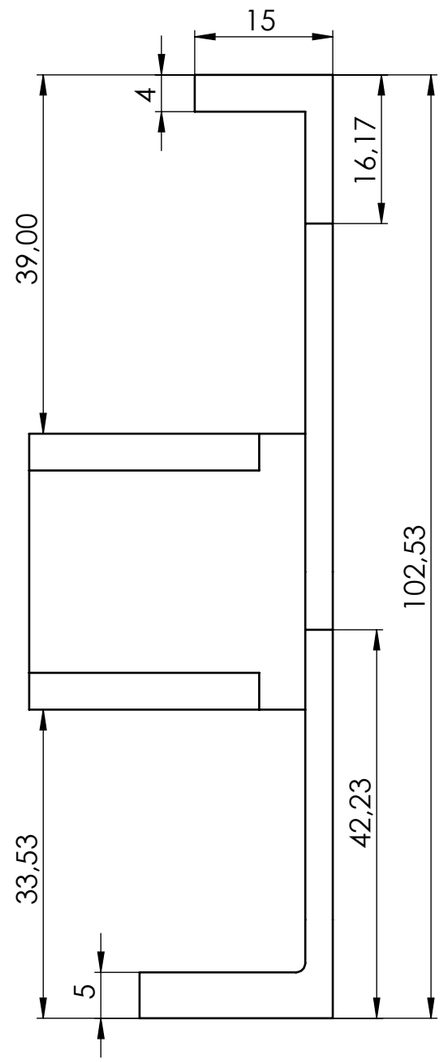
```
static byte stepMultiplier = DEFAULT_STEP_MULTIPLIER;
static int machineWidth = DEFAULT_MACHINE_WIDTH;
static int machineHeight = DEFAULT_MACHINE_HEIGHT;

static float currentMaxSpeed = 800.0;
static float currentAcceleration = 400.0;
static boolean usingAcceleration = true;

int startLengthMM = 800;

float mmPerStep = 0.0F;
float stepsPerMM = 0.0F;

long pageWidth = machineWidth * stepsPerMM;
long pageHeight = machineHeight * stepsPerMM;
long maxLength = 0;

const int INLENGTH = 50;
const char INTERMINATOR = 10;
const char SEMICOLON = ';';
float penWidth = 0.8F;
boolean reportingPosition = true;
boolean acceleration = true;

extern AccelStepper motorA;
extern AccelStepper motorB;

boolean currentlyRunning = true;

static char inCmd[10];
static char inParam1[14];
static char inParam2[14];
static char inParam3[14];
static char inParam4[14];

static byte inNoOfParams;

char lastCommand[INLENGTH+1];
boolean commandConfirmed = false;

int rebroadcastReadyInterval = 5000;
long lastOperationTime = 0L;
long motorIdleTimeBeforePowerDown = 600000L;
boolean automaticPowerDown = true;
boolean powerIsOn = false;

long lastInteractionTime = 0L;

#define READY_STR "READY"
#define RESEND_STR "RESEND"
#define DRAWING_STR "DRAWING"
#define OUT_CMD_SYNC_STR "SYNC,"

char MSG_E_STR[] = "MSG,E,";
char MSG_I_STR[] = "MSG,I,";
char MSG_D_STR[] = "MSG,D,";

const static char COMMA[] = ",";
const static char CMD_END[] = ",END";
const static String CMD_CHANGELENGTH = "C01";
const static String CMD_CHANGEPENWIDTH = "C02";
```

```
const static String CMD_SETPOSITION = "C09";
#ifdef PENLIFT
const static String CMD_PENDOWN = "C13";
const static String CMD_PENUP = "C14";
const static String CMD_SETPENLIFTRANGE = "C45";
#endif
#ifdef VECTOR_LINES
const static String CMD_CHANGELENGTHDIRECT = "C17";
#endif
const static String CMD_SETMACHINESIZE = "C24";
const static String CMD_GETMACHINEDETAILS = "C26";
const static String CMD_RESETEEPROM = "C27";
const static String CMD_SETMACHINEMMPERREV = "C29";
const static String CMD_SETMACHINESTEPSPERREV = "C30";
const static String CMD_SETMOTORSPEED = "C31";
const static String CMD_SETMOTORACCEL = "C32";
const static String CMD_SETMACHINESTEPMULTIPLIER = "C37";

void setup()
{
  Serial.begin(57600);
  configuration_motorSetup();
  eeprom_loadMachineSpecFromEeprom();
  configuration_setup();

  motorA.setMaxSpeed(currentMaxSpeed);
  motorA.setAcceleration(currentAcceleration);
  motorB.setMaxSpeed(currentMaxSpeed);
  motorB.setAcceleration(currentAcceleration);

  float startLength = ((float) startLengthMM / (float) mmPerRev) * (float)
motorStepsPerRev;
  motorA.setCurrentPosition(startLength);
  motorB.setCurrentPosition(startLength);
  for (int i = 0; i<INLENGTH; i++) {
    lastCommand[i] = 0;
  }
  comms_ready();

#ifdef PENLIFT
  penlift_penUp();
#endif
  delay(500);

}

void loop()
{
  if (comms_waitForNextCommand(lastCommand))
  {
    comms_parseAndExecuteCommand(lastCommand);
  }
}
```

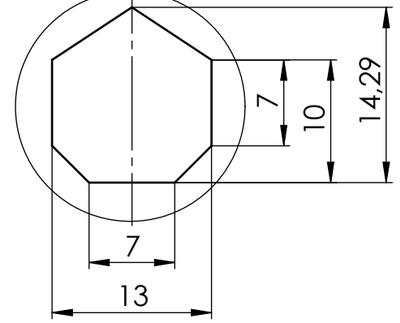








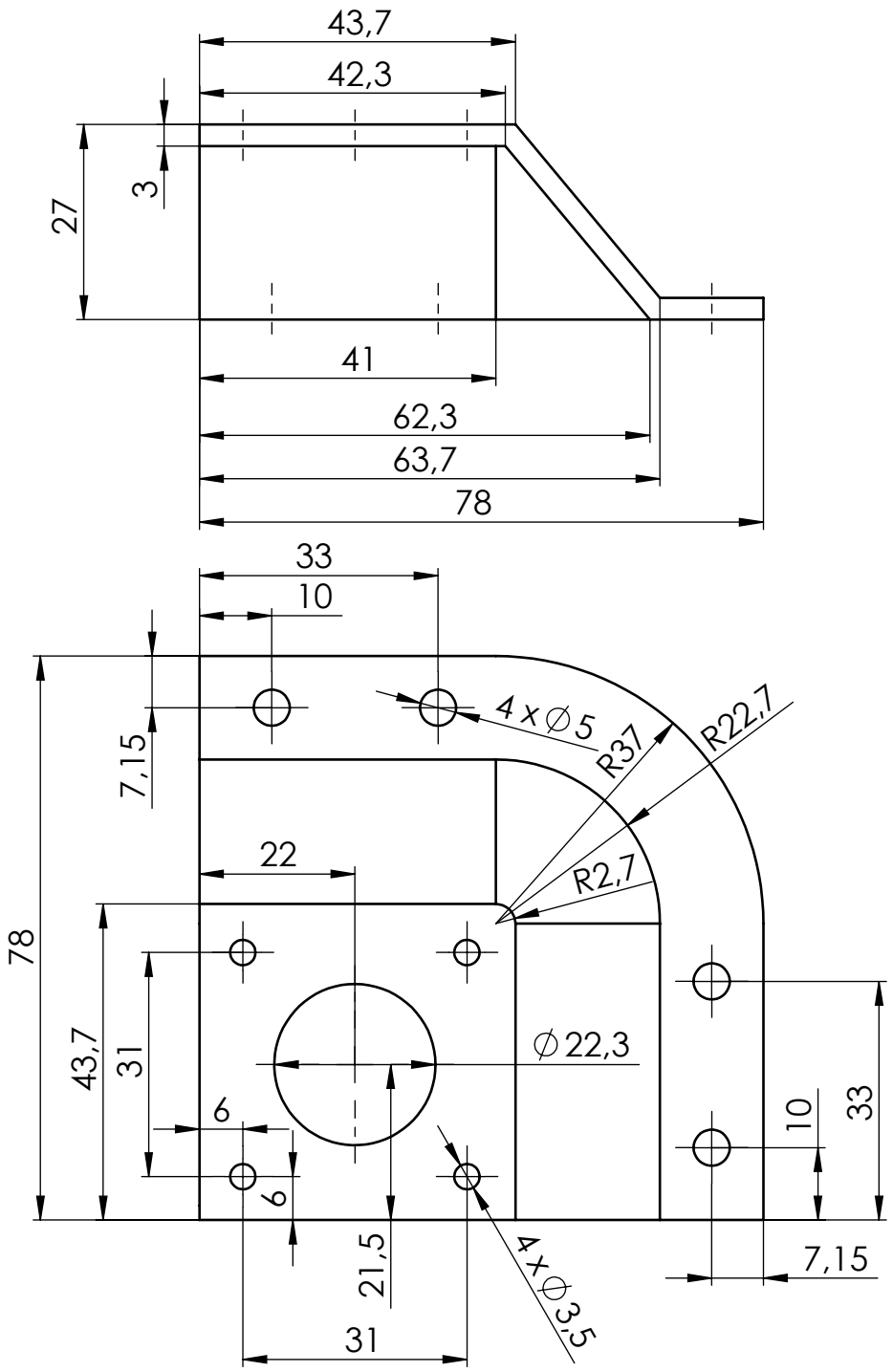
Detalj B M (2 : 1)



	Datum	Ime i prezime	Potpis
Projektirao		Lovre Listeš	
Razradio		Lovre Listeš	
Crtao		Lovre Listeš	
Pregledao		Dr. sc. Tomislav Stipančić	
Mentor			

FSB Zagreb

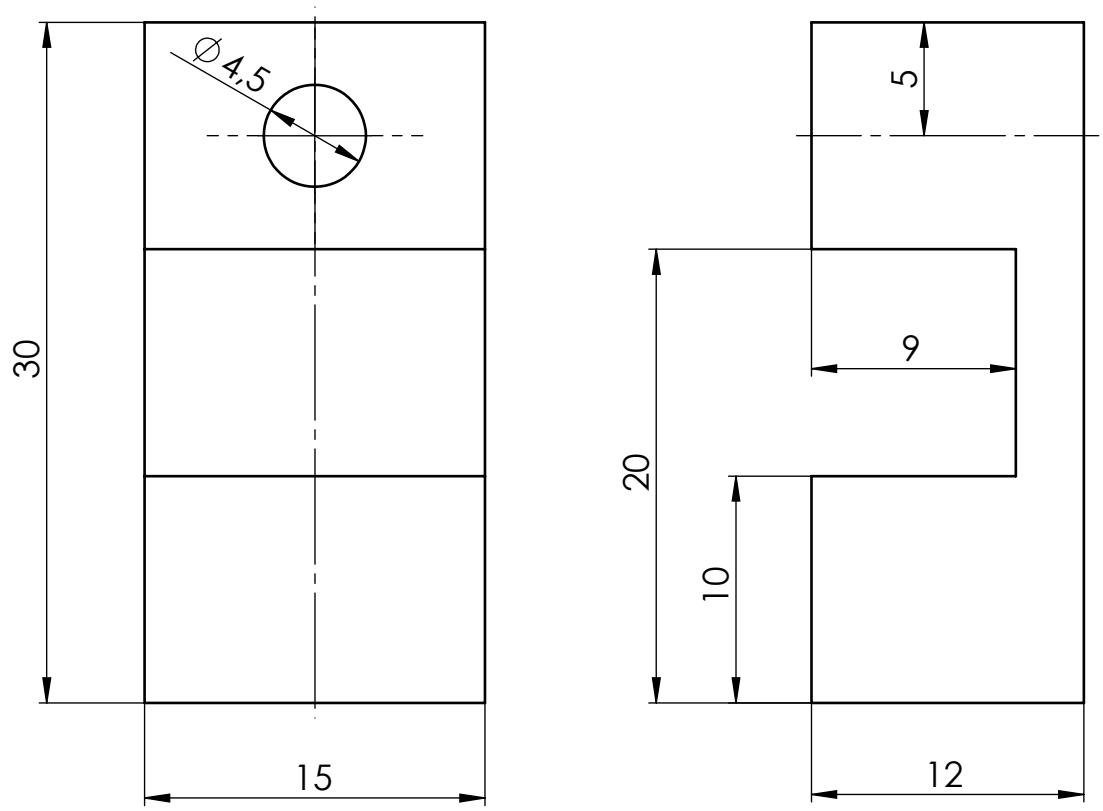
Objekt:	Objekt broj:
	R. N. broj:
Napomena:	Kopija
Materijal: ABS	Masa: 37,6 g
Mjerilo originala: M 1,5 : 1	Naziv: NOSAČ OLOVKE
Pozicija: 1	Format: A3
Listova: 1	List: 1
Crtež broj: 10 - 01	



	Datum	Ime i prezime	Potpis
Projektirao		Lovre Listeš	
Razradio		Lovre Listeš	
Crtao		Lovre Listeš	
Pregledao		Dr. sc. Tomislav Stipančić	

FSB Zagreb

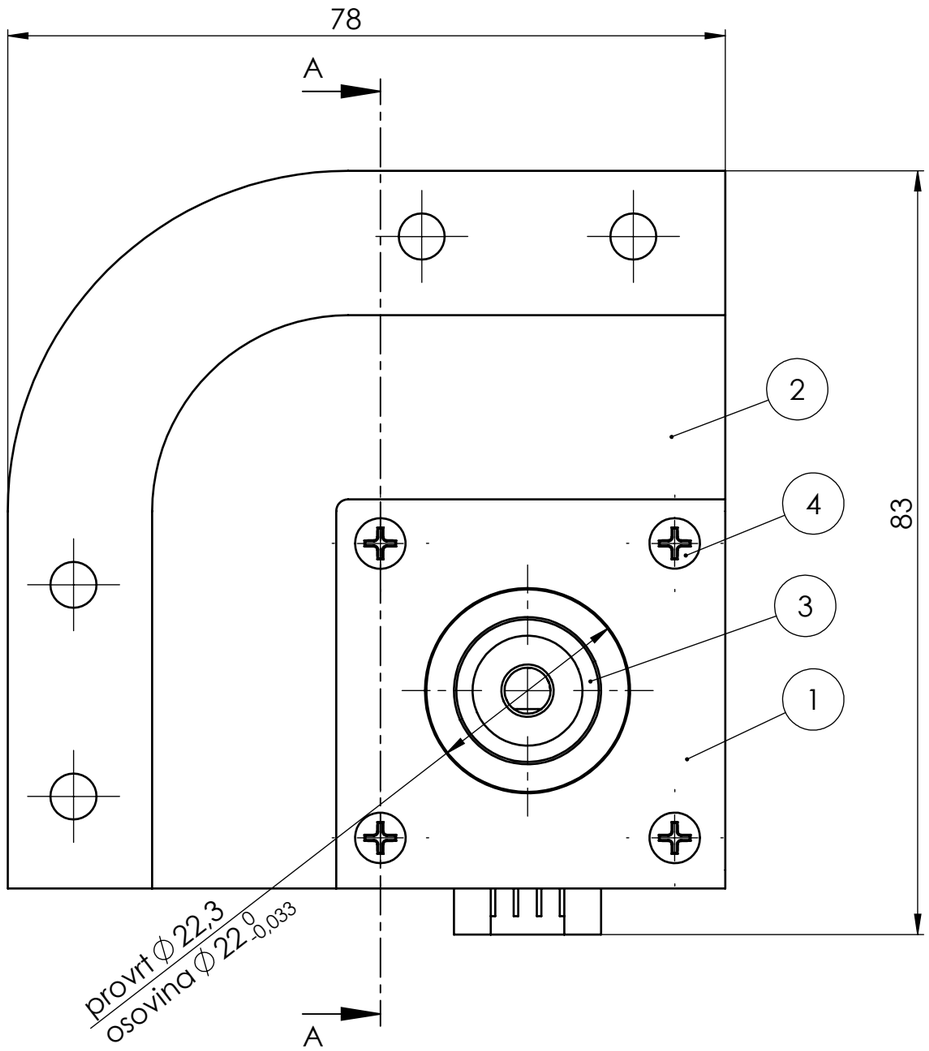
Objekt:	Objekt broj:
	R. N. broj:
Napomena:	Kopija
Materijal: ABS	Masa: 19,63 g
Mjerilo originala M 1:1	Naziv: NOSAČ MOTORA
Pozicija: 2	Format: A4
Listova: 1	List: 1
Crtež broj: 10 - 02	



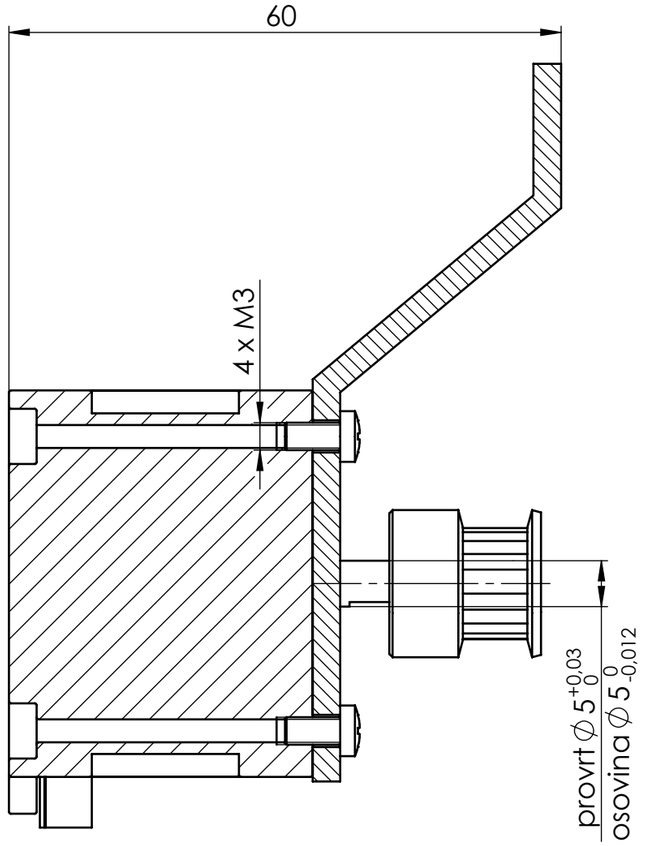
	Datum	Ime i prezime	Potpis
Projektirao		Lovre Listeš	
Razradio		Lovre Listeš	
Crtao		Lovre Listeš	
Pregledao		Dr. sc. Tomislav Stipančić	

FSB Zagreb

Objekt:	Objekt broj:
	R. N. broj:
Napomena:	Kopija
Materijal: ABS	Masa: 3,94 g
Mjerilo originala M 3 : 1	Naziv: SPOJKA
	Pozicija: 3
Crtež broj: 10 - 03	Format: A4
	Listova: 1
	List: 1



Presjek A - A
M 1,5:1



Poz.	Naziv dijela	Kom.	Crtež broj Norma	Materijal	Sirove dimenzije Proizvođač	Masa
4	Križni vijak	4	DIN 7986	SS316	M3x6	0,65
3	GT2 remenica 16 zubi	1	-	Aluminij	⌀14x18	5,65
2	Nosač motora	1	10 - 02	ABS	78x78x27	19,63
1	Koračni motor 17HS3041	1	-	-	42x42x58	213,1

Broj naziva - code		Datum	Ime i prezime	Potpis
	Projektirao		Lovre Listeš	
	Razradio		Lovre Listeš	
	Crtao		Lovre Listeš	
	Pregledao		Dr. sc. Tomislav Stipančić	
	Mentor		Dr. sc. Tomislav Stipančić	

FSB Zagreb

ISO - tolerancije

Objekt:

Objekt broj:

R. N. broj:

Napomena: Remenica je pritegnuta s dva DIN 913 M3x4 vijka

Kopija

Materijal:

Masa: 241 g

Mjerilo originala
M 1,5:1

Naziv: PODSKLOP 1

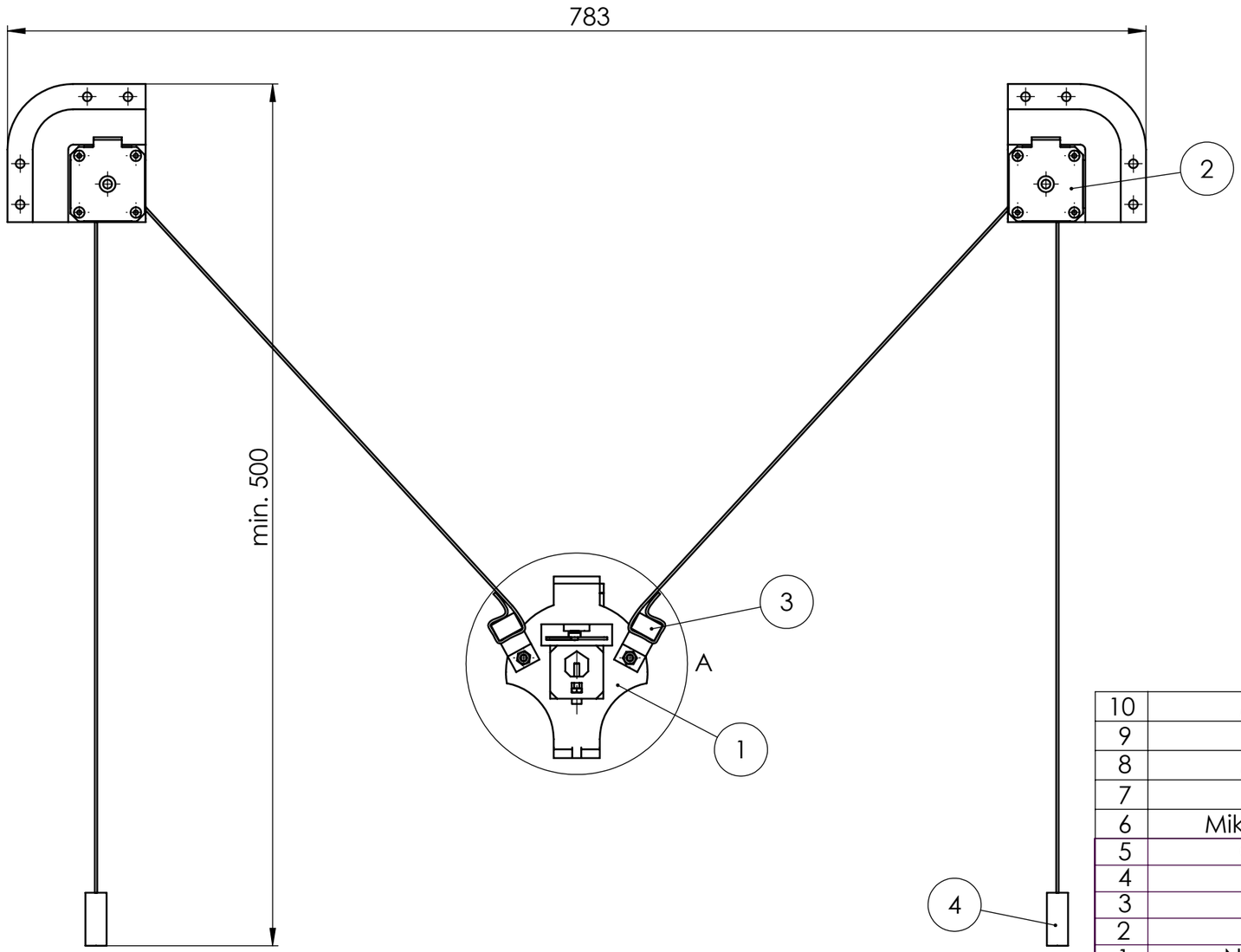
Pozicija: 2

Format: A3

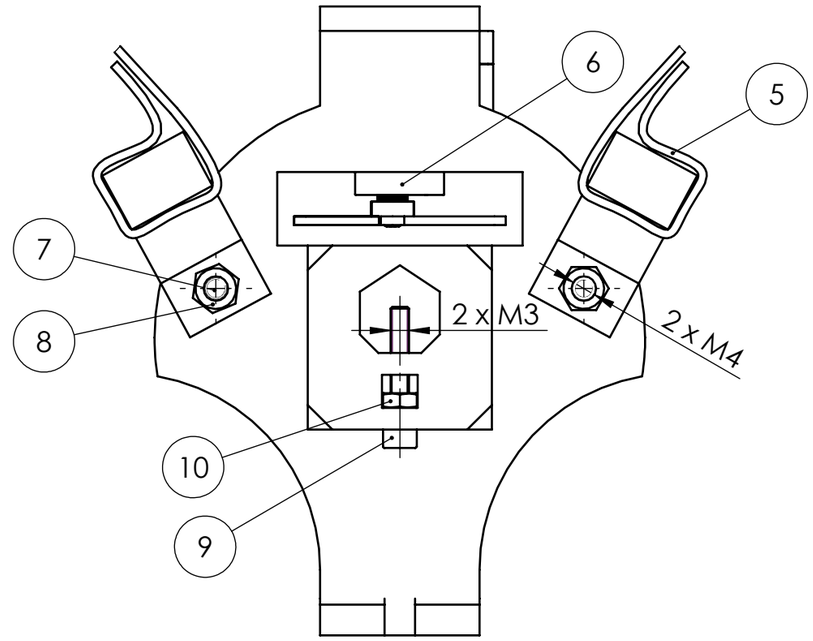
Listova: 1

Crtež broj: 10 - 04

List: 1



Detalj A
M 1:1



Poz.	Naziv dijela	Kom.	Crtež broj Norma	Materijal	Sirove dimenzije Proizvođač	Masa
10	Matica M3	2	DIN 934	8	M3	0,39
9	Vijak M3	2	DIN 9427	8.8	M3x20	1,6
8	Matica M4	2	DIN 934	8	M4	0,84
7	Vijak M4	2	DIN 84 A	8.8	M4x20	3
6	Mikro servo SG90	1	-	-	Tower Pro	9
5	GT2 remen	2	-	Guma	SDP/SI	16
4	Uteg	2	-	Čelik	19x13x58	90
3	Spojka	2	10 - 03	ABS	30x15x12	3,94
2	Podsklop 1	2	10 - 04	-	83x78x60	241
1	Nosač olovke	1	10 - 01	ABS	102,53x79x33	37,6

Broj naziva - code		Datum	Ime i prezime	Potpis	
	Projektirao		Lovre Listeš		FSB Zagreb
	Razradio		Lovre Listeš		
	Crtao		Lovre Listeš		
	Pregledao		Dr. sc Tomislav Stipančić		
	Mentor		Dr. sc. Tomislav Stipančić		

ISO - tolerancije	Objekt:	Objekt broj:
		R. N. broj:
	Napomena:	Kopija
	Materijal:	Masa: 519 g
	Mjerilo originala M 1:3	Naziv: ROBOTSKI CRTAČ
		Pozicija:
		Format: A3
		Listova: 1
	Crtež broj: 10	List: 1

Design by CADLab

0 10 20 30 40 50 60 70 80 90 100