Edukacijski robot s pet stupnjeva slobode gibanja

Peter, Kristijan

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Kristijan Peter

Zagreb, 2019.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:
Prof. dr. sc. Mladen Crneković

Student:
Kristijan Peter

Zagreb, 2019.

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

Zahvaljujem se svima koji su mi pomogli pri izradi samog rada svojim znanjem i iskustvom, a posebno bih istaknuo svog mentora prof. dr. sc. Mladena Crnekovića, jer je prihvatio mentorstvo za ovaj rad.

Zahvalio bih se još i svojoj obitelji i prijateljima na pruženoj podršci tokom studija, a ponajviše bih se htio zahvaliti svojoj djevojci Maji za svu podršku koju mi je pružala.

Kristijan Peter

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:	
Ur. broj:	

DIPLOMSKI ZADATAK

Student: **KRISTIJAN PETER** Mat. br.: 0035198029

Naslov rada na hrvatskom jeziku: **Edukacijski robot s pet supnjeva slobode gibanja**

Naslov rada na engleskom jeziku: **Educational robot with five degrees of freedom**

Opis zadatka:

Za mnoge učenike edukacija iz robotike počinje željom za izradom svojeg robota. Ubrzo shvate da nemaju dovoljno znanja pa traže gotov projekt s uputama tijekom kojega će uvećati svoje znanje. U današnje vrijeme 3D printera, jeftine elektronike i ostalih elemenata od kojih je sastavljen robot, moguće je za prihvatljiv iznos sredstava izraditi vlastitog robota.
U zadatku treba istražiti tržište gotovih edukacijskih robota i napraviti njihovu usporedbu, a zatim predložiti svoje rješenje uz pretpostavku da korisnik ima 3D printer i može nabaviti hobističke dijelove koje sam ne izrađuje (motori, senzori, kontroler itd.).
U radu je potrebno:
1. Definirati mehaničku konstrukciju robota.
2. Odabrati motore s pripadnim kontrolerima.
3. Odabrati senzore i mikrokontroler.
4. Definirati sučelje robota.
5. Procijeniti ukupnu cijenu robota.

Potrebno navesti korištenu literaturu i ostale izvore informacija te eventualno dobivenu pomoć.

Zadatak zadan:
15. studenog 2018.

Rok predaje rada:
17. siječnja 2019.

Predviđeni datum obrane:
23. siječnja 2019.
24. siječnja 2019.
25. siječnja 2019.

Zadatak zadao:
prof. dr. sc. Mladen Crneković

Predsjednica Povjerenstva:
prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

SAŽETAK

Nakon uvodnog dijela u diplomskom radu opisan je postupak izrade edukacijske robotske ruke ERA.

Opisana je konstrukcija robotske ruke, od izrade modela do 3D pritanja realnog modela; razvijen je kinematički model robotske ruke ERA te su pomoću njega izvedeni direktni i inverzni kinematički problem koji, osim *joysticka,* služe za upravljanje robotskom rukom ERA.

Odabran je aktuatorski dio robotske ruke ERA, koji se sastoji od koračnih motora, te je razvijen električni sustav za upravljanje motorima. Električni sustav se sastoji od upravljačke jedinice *Arduino MEG2560, Driver*-a za koračne motore, napajanja i mikroprekidača.

Razvijeno je i korisničko sučelje robotske ruke ERA, u programskom paketu *Python,* i napisan programski kod za upravljanje robotsko rukom, u programskom paketu *Arduino,* kako bi bilo omogućeno davanje naredbi i upravljanje robotskom rukom po zglobovima.

Na samom kraju rada nalazi se tablica u kojoj su dane karakteristike robotske ruke ERA, uspoređene sa trenutno najboljom i sa najjeftinijom edukacijskom robotskom rukom na tržištu.

Ključne riječi: Edukacijski robot, edukacijska robotika, edukacijska robotska ruka, robotska ruka, inverzni i direktni kinematički problem

SUMMARY

This Master's Thesis follows the process of developing the Educational Robotic Arm, ERA. The construction of the robotic arm is described, from making the model, to 3D printing the finished product.

The kinematic model of ERA was developed and used to calculate the direct and inverse kinematic problems which are, apart from when using the joystick, used to control the robotic arm.

The actuator part of the robotic arm, comprised of stepper motors, was chosen, and an electrical system to control the motors was developed. The electrical system contains the *Arduino MEG2560* control unit, *Drivers* for stepper motors, power supply and limit switches.

A graphic interface for ERA was developed, written in *Python.* A code for controlling the robotic arm was written in *Arduino*, to enable operating each separate joint of the arm.

Enclosed at the end of this Thesis is a table, comparing various characteristics of ERA to the best and to the cheapest educational robot of this type, currently on the market.

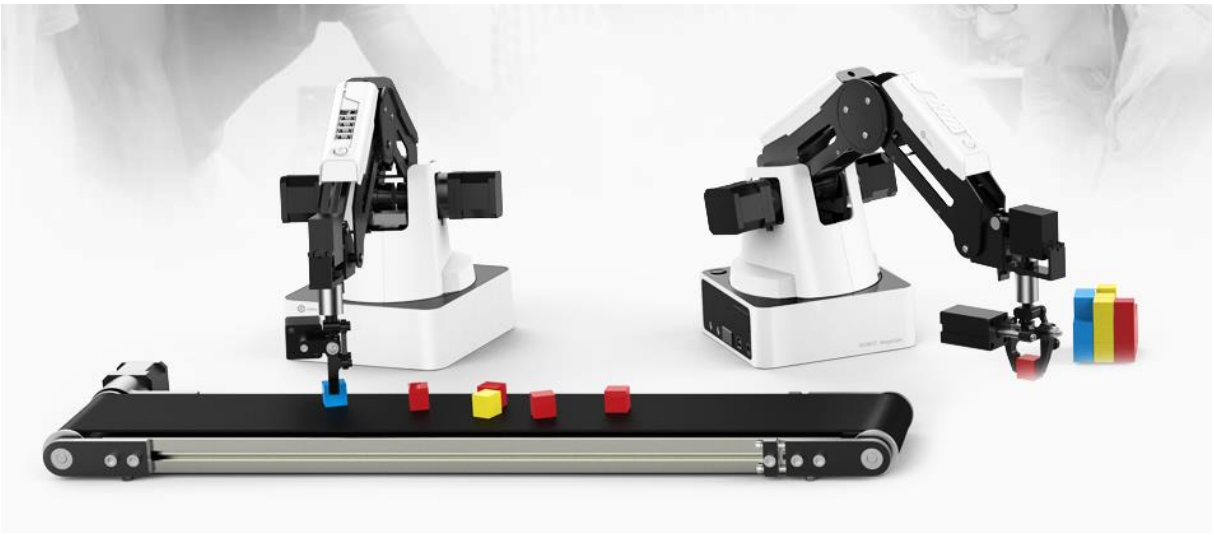
Key words: Educational robot, educational robotics, educational robotic arm, robotic arm, inverse and forward kinematics problem

1. UVOD

Robotika je povezana sa povijesnim razvojem tehnologije i znanosti, te pokriva područje mehanike, elektronike , informatike i automatike. Robotika je grana inženjerske znanosti koja se bavi proizvodnjom, dizajnom, teoretskim proučavanjem i uporabom robota. Robot je, sam po sebi, stroj koji može bez prestanka ponavljati isti posao, te na njegovu produktivnost ne utječe umor, kao kod čovjeka. Glavni razlog razvoja robota je da pomogne čovjeku u proizvodnji, te da ga zamjeni kod opasnih i monotonih poslova. U posljednje vrijeme robotika se često počela koristiti i u drugim područjima poput medicine i edukacije.

Slika 1. Industrijski robot [1]

Budući da tehnologija jako brzo napreduje, a sami industrijski roboti su skupi i veliki, na tržištu su se pojavili edukacijski roboti, koji su jeftiniji , manji i *open source* te iz tog razloga poželjni u školama. Edukacijska robotika uči kako dizajnirati robota, raditi analizu i upravljati njime. U edukacijsku robotiku ubrajaju se robotske ruke, mobilni roboti i autonomna vozila. Edukacijska robotika može se učiti u osnovnim i srednjim školama, pa čak i na fakultetima, kao dio diplomskih programa. Robotika se u školama može koristiti za motiviranje učenika i olakšavanje nastave, ali isto tako za programiranje, upoznavanje sa umjetnom inteligencijom i inženjerskim svijetom. Prednost edukacijske robotike, u odnosu na samu robotiku, je niža cijena robotskih sustava, jednostavnije korištenje sustava i mogućnost nadogradnje sustava bez dodatnog naplaćivanja.



Slika 2. Edukacijski robot [2]

U diplomskom radu biti će prikazan proces izrade edukacijske robotske ruke ERA sa pet stupnjeva slobode gibanja. Kroz diplomski rad bit će opisani procesi izrade konstrukcije, upravljanja robotom pomoću kinematike i joystick-a, mehanički i električni sustav robota, te programski dio robota. Uz sve navedeno bit će prikazana usporedba ERA sa trenutnim tržištem.

2. KONSTRUKCIJA ROBOTSKE RUKE ERA

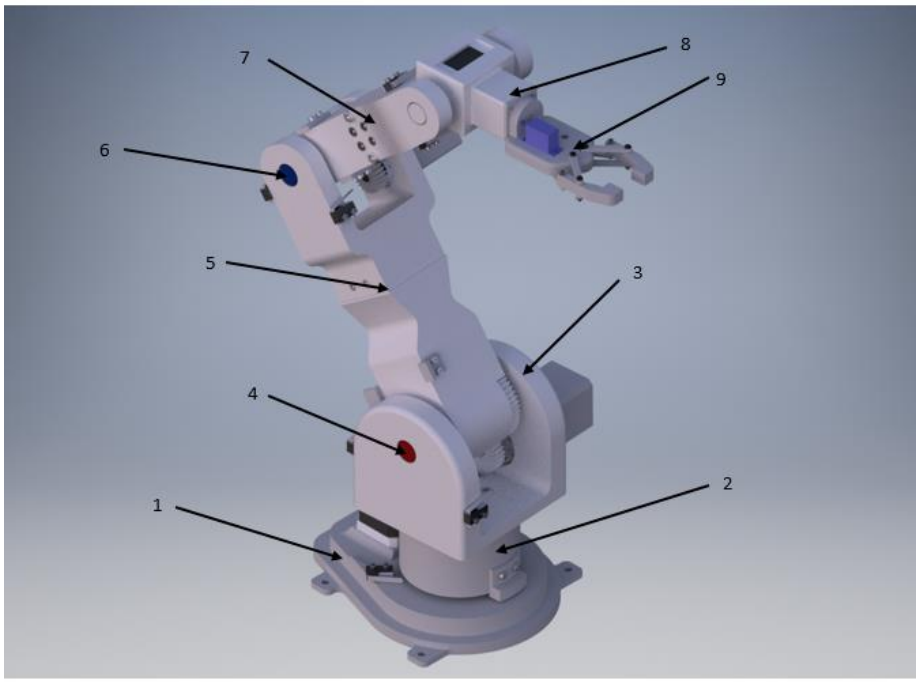
U poglavlju će biti opisan proces modeliranja i izrade robotske ruke ERA sa pet stupnjeva slobode gibanja. ERA je rotacijske strukture sa pet stupnjeva slobode gibanja, što znači da ima pet rotacijskih zglobova.

2.1. Model robotske ruke ERA

Prije same izrade modela treba odrediti zahtjeve robotske ruke i uzeti u obzir njezinu izradu. Budući da je u pitanju edukacijska robotska ruka, ti su zahtjevi: jednostavno upravljanje pomoću joystick-a te upravljanje preko sučelja kroz koje dobivamo informaciju pozicije vrha alata robotske ruke. Potrebno je maksimalno smanjiti cijenu te pojednostavniti izradu robotske ruke. Da bi se zadovoljio ovaj uvjet robotska ruka je 3D printana, stoga je poželjno da škola ili fakultet ima 3D printer ukoliko se odluči za izradu ove robotske ruke.

Model robotske ruke rađen je u programskom paketu *Inventor 2018*. Prilikom izrade modela, uzelo se u obzir, da svaki dio bude moguće 3D printati na 3D printeru *Ultimaker 3*, kojemu su dimenzije 3D printanja 215x215x200 mm.

Na slici 3. prikazan je model robotske ruke ERA koja se sastoji od sljedećih pozicija.



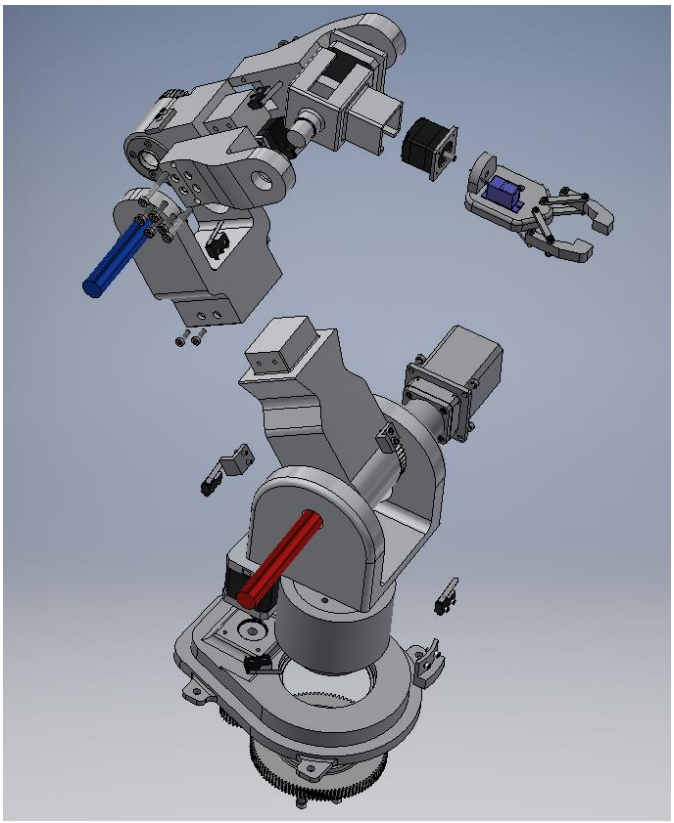
Slika 3. Model robotske ruke ERA

Pozicije sa slike 3. su :

1. **Baza prvog zgloba**
2. **Prvi zglob**
3. **Članak prvog i drugog zgloba**
4. **Osovina drugog zgloba**
5. **Članak drugog i trećeg zgloba**
6. **Osovina trećeg zgloba**
7. **Članak trećeg i četvrtog/petog zgloba**
8. **Četvrti/peti zglob**
9. **Hvataljka**

Glavna karakteristika i ono što razlikuje model robotske ruke ERA od ostalih modela je što zamjenjuje uobičajeni remenski prijenos zupčastim prijenosom, dakle pojednostavljuje samu montažu robotske ruke. Nedostatak prijenosa preko zupčanika jest vanjska pozicija motora, prikazana na slici 3.

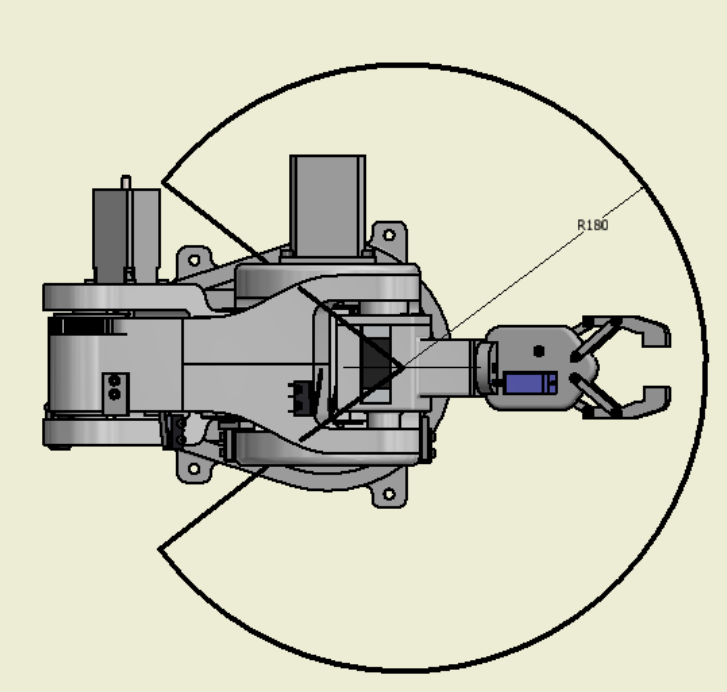
Na slici 4 prikazan je način sklapanja robotske ruke ERA.



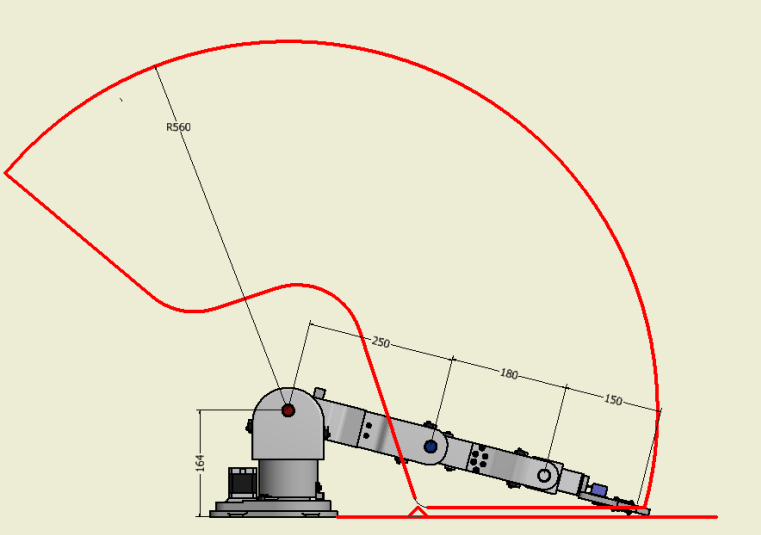
Slika 4. Sklapanje robotske ruke ERA

2.2.Radni prostor robotske ruke ERA

Radni prostor robotske ruke je skup točaka u prostoru koje se mogu dohvatiti ručnim zglobom manipulatora, na koji je pričvršćen završni mehanizam. Veličina radnog prostora robotske ruke ovisi o broju i tipu zglobova manipulatora, duljinama članaka te o postojećim fizičkim ograničenjima koja su neposredno povezana s konkretnom građom i izgledom manipulatora.[3]



Slika 5. Kut zakreta robotske ruke ERA



Slika 6. Okomiti pomak robotske ruke ERA

2.3. Realan model robotske ruke ERA

Model robotske ruke ERA 3D printan je na 3D printeru *Ultimaker 3* kojemu su dimenzija printanja 215 x 215 x 200 mm. Na slici 7. prikazan je *Ultimaker 3* 3D printer, a u tablici 1 prikazane su njegove karakteristike.

Slika 7. Ultimaker 3

Tablica 1. Karakteristike 3D printera Ultimaker 3

Dimenzije printera	342 x505 x588 mm
Masa printera	10,6 kg
Dimenzije printanja	215 x 215 x 200 mm
Promjer mlaznice	0.25 mm, 0.4 mm, 0.8 mm
Materijali za printanje	Nylon, PLA, ABS, CPE, PVA, PP, TPU, PC, 95A
Glava printera	Glava sa dva ekstrudera
Komunikacija	WiFi, LAN, USB
Brzina printanja	30 – 300 mm/s
Temperatura mlaznice	180°C – 280 °C
XYZ rezolucija	12.5, 12.5, 2,5 micron
Tehnologija printanja	Fused filament febrication (FFF)
Cijena	22 000 kn

Opcija 3D printanja modela robotske ruke ERA odabrana je zato što u današnje vrijeme većina škola posjeduje 3D printer, te je iz tog razloga to najjeftiniji proces izrade edukacijske robotske ruke. Nije potrebno da se za izradu robotske ruke ERA koristi *Ultimaker 3*, bilo koji printer sa istim ili većim dimenzijama printanja je upotrebljiv.

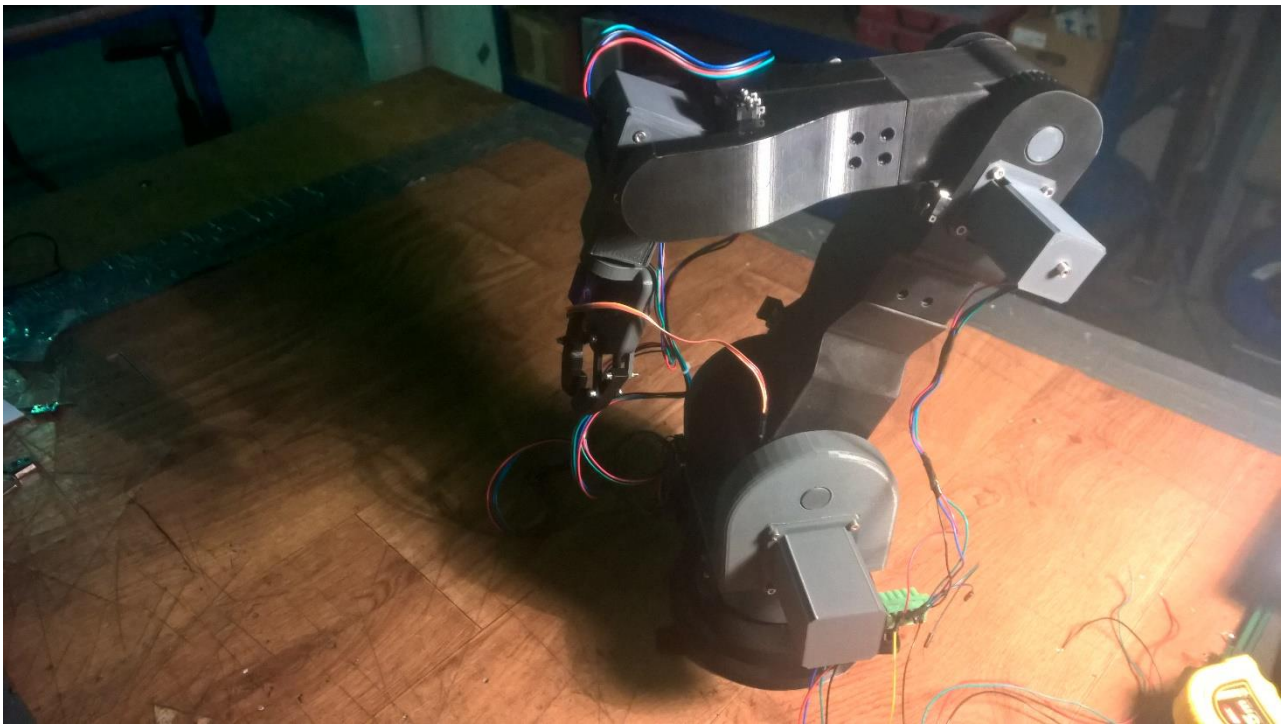
Materijal korišten za 3D printanje je polilaktid(PLA), biorazgradiv plastomer dobiven iz obnovljivih izvora, poput kukuruznog škroba, šećerne trske, korijena tapioke i slično. PLA kao materijal za 3D printanje je vrlo popularan, a razlog tome je njegova niska toksičnost i ekološka prihvatljivost u odnosu na ostale materijale. Njegova dostupnost je sve veća,

dostupan je u različitim bojama, vrlo je jednostavan za naknadnu obradu brušenjem i premazivanje premazima, no prognoze su da će uskoro postati periferni izbor.

Glavni nedostatak je, da ne može podnijeti visoke temperature, jer već pri temperaturi od 50°C omekša. Ovo je za neke primjene i prednost jer se predmet može lako ponovno zagrijati što omogućuje popravljanje eventualnih grešaka, savijanje ili spajanje dijelova [4].

PLA je tvrd materijal, pomalo krhak nakon što se ohladi. Potrebne su mu niže temperature za rad, obično oko 160°C – 220°C. Grijana podloga i nije toliko potrebna, ali može biti korisna za kvalitetu 3D printanja na temperaturi od oko 50 – 60°C. Jako se sporo hladi pa se preporuča ventilator za hlađenje, kako bi se ubrzao proces [4].

Osim navedenih karakteristika glavni razlog uporabe PLA-a je njegova niska cijena. Na slici 8 prikaz je 3D printan model robotske ruke ERA.



Slika 8. Realan model robotske ruke ERA

3. UPRAVLJANJE ROBOTSKOM RUKOM ERA

Karakteristika svakog robota je broj stupnjeva slobode gibanja, to jest broj osi za rotacijsko ili translacijsko gibanje. Postoji mnogo načina upravljanja robotskom rukom od:

- Upravljanja od točke do točke
- Kontinuiranog upravljanja po putanji
- Upravljanju po vektoru položaja
- Upravljanju po vektoru brzine
- Upravljanje po kriteriju minimalnog utroška energije

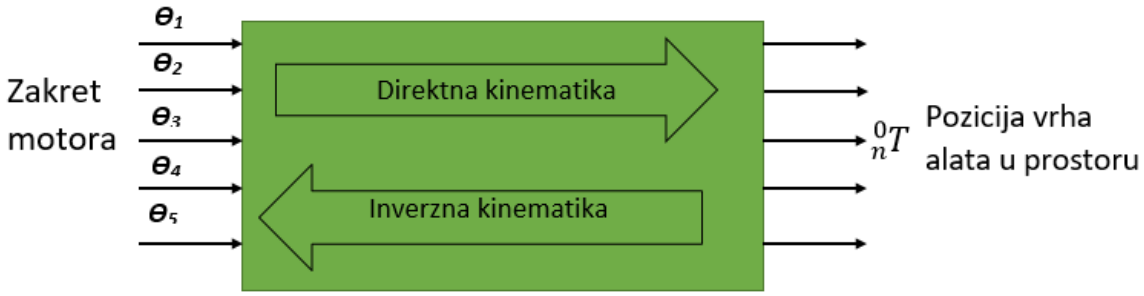
Budući da se radi o edukacijskoj robotskoj ruci, poželjno je da upravljanje robotske ruke bude što jednostavnije te je iz tog razloga za našu robotsku ruku upravljanje pomoću inverzne i direktne kinematike, te pomoću *joystic*-a. Odabran je ovakav način upravljanja iz razloga što je kasnije puno lakše nadograditi sustav jer je kinematički problem riješen.

3.1.Kinematički model robotske ruke ERA

Kinematika je dio fizike koji se bavi proučavanjem gibanja određenog tijela ne uzimajući u obzir sile ili momente koji utječu na gibanje. Pojam kinematika robota odnosi se na analitičko proučavanje gibanja robota. Da bismo mogli proučavati ponašanje robota potrebno je formulirati prikladan kinematički model.

Za modeliranje kinematike robota uglavnom se koristi kartezijski koordinatni sustav. Transformacija između dva kartezijska koordinatna sustava može se podijeliti na translaciju i rotaciju. Rotacija se može prikazati na razne načine: Eulerovim kutovima, Gibbs-ovim vektorom, Cayley-Klein parametrima, ortogonalnim matricama, itd. Od navedenih, u robotici se najčešće koristi homogena transformacija koordinatnih sustava temeljena na realnim matricama dimenzija 4x4. Prema Denavit i Hartenberg-u za opću transformaciju između dvaju zglobova potrebna su četiri parametra. Ti parametri, znani kao DH parametri postali su standard pri opisu kinematike robota.[5]

Kinematika robota dijeli se na direktnu i inverznu kinematiku. Rješavanje direktnog kinematičkog problema je jednostavno i ne zahtjeva izvođenje kompleksnih jednadžbi, dok je rješavanje inverznog kinematičkog problema puno kompliciraniji analitički problem. Na slici 9. prikazan je odnos između inverzne i direktne kinematike.



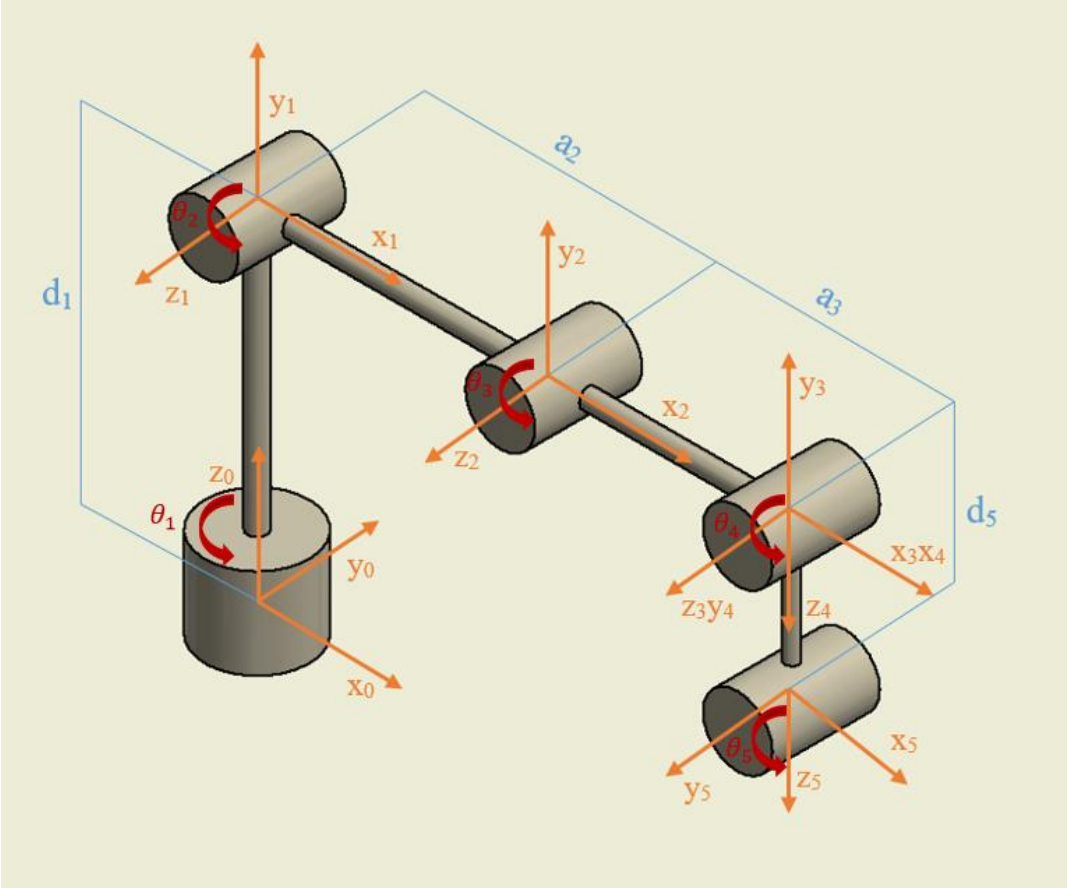
Slika 9. Odnos direktnog i inverznog kinematičkog problema

3.1.1. Direktni kinematički problem

Direktni kinematički problem možemo definirati na slijedeći način: pozicije zglobova robotske ruke definiraju poziciju i orijentaciju vrha alata robotske ruke.

Prilikom rješavanja direktne kinematike potrebno je ustanoviti simboličku shemu robotske ruke, u odnosu sa nepokretnim koordinatnim sustavom. Kod simboličke sheme robotske ruke u obzir treba uzeti realne dimenzije robota. Uz simboličku shemu potrebno je, pomoću Denavit- Hartenberg -ove metode, pridružiti koordinatni sustav pojedinom segmentu robotske ruke, a broj segmenata jednak je broju stupnjeva slobode gibanja robotske ruke.

Robotska ruka ERA ima pet stupnjeva slobode gibanja. Zglob 1 je baza robotske ruke ERA i ima rotaciju $\theta_1$oko z_0 osi u x_0y_0 ravnini. Zglob 2 je rame robotske ruke ERA i okomit je na bazu robotske ruke ERA i vrši rotaciju θ_2 oko z_1 osi. Zglob 3 je podlaktica robotske ruke ERA i vrši rotaciju θ_3 oko osi z_2, dok je zglob 4 zglob robotske ruke ERA i vrši rotaciju θ_4 oko z_3 osi. Z osi podlaktice i zgloba su paralelne sa Z osi ramena. Zglob 5 je rotacija alata robotske ruke ERA i njegova z_4 os je vertikalna na z_3 os i vrši rotaciju θ_5 oko z_4 osi. Na slici 10. prikazana je simbolička shema robotske ruke ERA.



Slika 10. Simbolička shema robotske ruke ERA

Na slici 10 $\theta_1, \theta_2, \theta_3, \theta_4$ *i* θ_5 su kutevi zakreta robotske ruke ERA, a d_1, a_2, a_3 *i* d_5 su dimenzije članaka robotske ruke ERA i iznose:

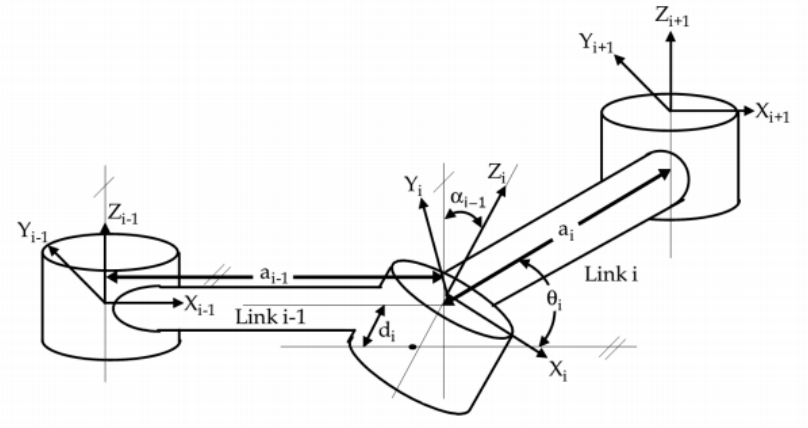
$$d_1 = 100\ mm$$

$$a_2 = 225\ mm$$

$$a_3 = 215\ mm$$

$$d_5 = 130\ mm$$

Sljedeći korak kod direktnog kinematičkog problema jest Denavit-Hartenberg-ovom (DH) metodom odrediti matricu direktnog kinematičkog problema robotske ruke. Denavit-Hartenberg-ova metoda koristi četiri parametra kako bi opisala kinematiku robotske ruke, a to su: kut zakreta zgloba oko *Z* osi (θ_i), odmak članka po *Z* osi (d_i), zakret članka oko *X* osi (α_i) i odmak članka po *X* osi (a_i). Zbog lakšeg određivanja DH parametra, na svaki zglob je postavljen koordinatni sustav. Na slici 11. prikazan je postupak određivanja DH parametara članka *i* te članka *i+1* koji su povezani sa zglobom *i*.



Slika 11. Denavit – Hartenberg parametri [5]

Članak je kruto tijelo koje određuje prostorni odnos između dva susjedna zgloba. Članak je definiran preko pomaka d_i i kuta zakreta zgloba θ_i, koji su promjenjivi, dok su zakret članak α_i i duljina članka a_i, konstantni.[5]

U tablici 2. su prikazani DH parametri robotske ruke ERA.

Tablica 2. DH parametri robotske ruke ERA

Zglob	Naziv zgloba	Kut zakreta zgloba oko Z osi ,θ_i	Odmak članka po Z osi, d_i	Odmak članka po X osi, a_i	Zakret članka oko X osi, α_i	Raspon kuta zakreta zgloba oko Z osi
1	Baza	θ_1	100	0	90	-120 do 120
2	Rame	θ_2	0	225	0	-140 do 10
3	Podlaktica	θ_3	0	215	0	-90 do 90
4	Nagib alata	θ_4	0	0	90	-180 do 10
5	Rotacija alata	θ_5	130	0	0	-180 do 180

Nakon određenih DH parametara, možemo pomoću homogenih transformacija odrediti bilo koji položaj vrha alata robotske ruke. Kako bismo si olakšali računanje matrice transformacija za robotsku ruku ERA, podijelili smo matematički problem na dva manja

matematička problema. Prvi problem je povezan sa pozicioniranjem, a drugi sa orijentacijom vrha alata u prostoru:

$$ {}^{0}_{5}T = {}^{0}_{3}T * {}^{3}_{5}T \tag{3.1} $$

Gdje je ${}^{0}_{3}T$ matrica transformacije prva tri zgloba koja određuje poziciju vrha alata u prostoru:

$$ {}^{0}_{3}T = {}^{0}_{1}T * {}^{1}_{2}T * {}^{2}_{3}T $$

$$ {}^{0}_{3}T = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c\_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_3 & -s\_3 & 0 & a_3c_3 \\ s_3 & c_3 & 0 & a_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2} $$

gdje je $c_1 = \cos(\theta_1)\,, s_1 = \sin(\theta_1)\,, c_2 = \cos(\theta_2),\ s_2 = \sin(\theta_2)$.

${}^{3}_{5}T$ je matrica transformacije četvrtog i petog zgloba koji određuju rotaciju vrha alata:

$$ {}^{3}_{5}T = {}^{3}_{4}T * {}^{4}_{5}T $$

$$ {}^{3}_{5}T = \begin{bmatrix} c_4 & 0 & s_4 & 0 \\ s_4 & 0 & -c_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3} $$

gdje je $c_4 = \cos(\theta_4)\,, s_4 = \sin(\theta_4)\,, c_5 = \cos(\theta_5)\,, s_5 = \sin(\theta_5)$.

Matrica transformacije može se dobiti još i množenjem matrica transformacija svakog zgloba te je matrica transformacije robotske ruke ERA jednaka matrici (3.4)

$$ {}^{0}_{5}T = {}^{0}_{1}T * {}^{1}_{2}T * {}^{2}_{3}T * {}^{3}_{4}T * {}^{4}_{5}T = \begin{vmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{3.4} $$

Matrica transformacija (3.4) je matrica pomoću koje određujemo orijentaciju i položaj vrha alata robotske ruke u prostoru. Kod matrice transformacije (3.4) prva ti stupca predstavljaju orijentaciju vrha alata robotske ruke , dok zadnji stupac predstavlja položaj vrha alata robotske ruke u prostoru. U ovom nas slučaju ne zanima orijentacija, nego samo pozicija vrha alata.

Kada izjednačimo četvrti stupac matrice transformacija robotske ruke ERA (3.1) i četvrti stupac matrice transformacija (3.4), dobijemo X, Y i Z koordinate položaja vrha alata u radnom prostoru robotske ruke ERA.

$$p_x = \big((c_1c_2s_3 + c_1c_3s_2)c_4 - (c_1s_2s_3 - c_1c_2c_3)s_4\big)d_5 + (c_1c_2c_3 - c_1s_2s_3)a_3 + c_1c_2a_2$$

$$p_y = (-(s_1s_2s_3 - c_2c_3s_1)s_4 + (c_2s_1s_2 + c_3s_1s_2)c_4)d_5 + (c_2c_3s_1 - s_1s_2s_3)a_3 + s_1c_2a_2$$

$$p_z = (-(c_2c_3 - s_2s_3)c_4 + (c_2s_3 + c_3s_2)s_4)d_5 + (c_2s_3 + c_3s_2)a_3 + s_2a_2 + d_1$$

3.1.2. Inverzni kinematički problem

Želimo li robotsku ruku postaviti u neku točku nama poznatih koordinata u radnom prostoru, moramo znati izračunati kako iz tih koordinata dobiti rotacije svakog zgloba robotske ruke, a za to nam služi inverzni kinematički problem. Inverzni kinematički problem rješava obrnuti postupak direktnog kinematičkog problema; za željenu točku vrha alata u radnom prostoru robota pomoću inverzne kinematike dobijemo rotacije, odnosno kut, svih zglobova robotske ruke, takve da vrh alata robotske ruke bude u željenoj točki.

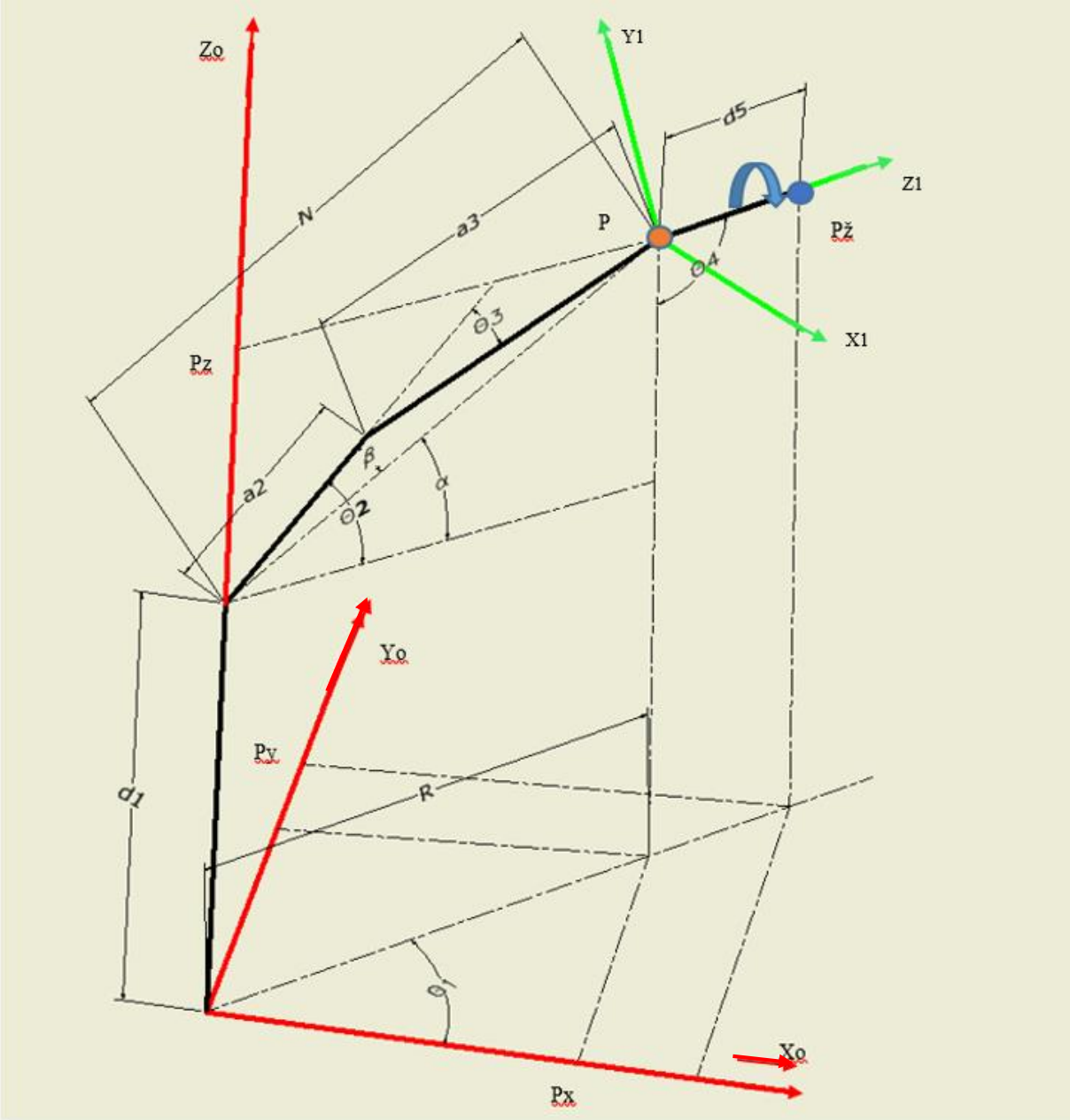
Inverzni kinematički problem je kompleksniji od direktnog kinematičkog problema,zato što, kod direktnog kinematičkog problema imamo za svaku kombinaciju kutova rotacije jednoznačno rješenje, dok kod inverznog kinematičkog problema imamo:

- Višestruka rješenja
- Kod redundantne robotske ruke beskonačan broj rješenja
- Moguća neprihvatljiva rješenja zbog strukture robotske ruke
- Ponekad izostanak konačnog rješenja zbog nelinearnih jednadžbi

Osim navedenih problema, nemamo gotove analitičke metode za rješavanje kinematičkog problema, već za svaku robotsku ruku treba odrediti metodu za rješavanje kinematičkog problema ovisno o strukturi i ograničenjima robotske ruke.

Postoji veliki broj pristupa rješavanja inverznog kinematičkog problema, ali u diplomskom radu je odlučeno ići geometrijskim pristupom, kod kojeg se struktura robotske ruke gleda u prostoru te se pomoću trigonometrije dođe do rješenja. Taj pristup je odabran zato što je ERA rotacijske strukture sa 5 stupnjeva slobode gibanja i takva struktura unosi veliki broj nelinearnih jednadžbi koje usporavaju procesor.

Kako bismo pojednostavnili inverzni kinematički problem, ponovno smo podijelili strukturu robotske ruke na dva dijela: prvi dio, sastavljen od prva tri rotacijska zgloba, koji određuju poziciju vrha alata robotske ruke u radnom prostoru, te drugi dio, sastavljen od zadnja dva zgloba, koji određuju orijentaciju vrha alata robotske ruke u radnom prostoru. Na slici 12 prikazan je geometrijski prikaz robotske ruke ERA.



Slika 12. Geometrijski prikaz robotske ruke ERA

Točka *P* je točka koja dijeli prvi dio robotske ruke, onaj koji određuje poziciju alata robotske ruke, od drugog dijela robotske ruke, onog koji određuje orijentaciju alata robotske ruke.

Rješavanje inverznog kinematičkog problema provedeno je tako, da su prvo određene jednadžbe kutova zakreta θ_1, θ_2 i θ_3 koji određuju poziciju alata robotske ruke u radnom prostoru. Nakon njih određene su jednadžbe kuteva zakreta θ_4 i θ_5 koji određuju orijentaciju alata robotske ruke u radnom prostoru. Za kut zakreta θ_5 ne postoji izraz za izračun iz razloga

što njegov kut ne mijenja X,Y i Z koordinate alata robotske ruke u prostoru, već samo njegovu rotaciju koju ne možemo prikazati sa X,Y i Z koordinatama.

Točka *Pž* sadrži koordinate pozicije alata robotske ruke u koje korisnik želi dovesti robotsku ruku.

Konstante d_1, a_2, a_3, d_5 su dimenzije robotske ruke ERA, a vrijednosti su im navedene u tablici 2., dok će nam varijable R i N pomoći kod proračuna, a njihove vrijednosti dobivamo iz sljedećih izraza :

$$R = \sqrt{P_x^2 + P_y^2} \tag{3.5}$$

$$N = \sqrt{(P_z - d_1)^2 + R^2} \tag{3.6}$$

Prije određivanja izraza za izračun kutova zakreta treba spomenuti, da će se za određivanje izraza za neke kutove zakreta koristiti funkcija $atan2()$, kako bi nam predznak bio određen i vrijednost kuta bila spremljena za sva četiri kvadranta. Prvo ćemo odrediti izraz za izračun kuta zakreta θ_1. Sa slike 12. vidimo da je θ_1 :

$$\theta_1 = atan2(P_y, P_x) \tag{3.7}$$

Da bismo odredili izraze za izračun kutova zakreta θ_2 i θ_3, prvo trebamo odrediti izraze za izračun kuteva α i β, koristeći kosinusov poučak:

$$a^2 = b^2 + c^2 - 2 * b * c * \cos(\theta) \tag{3.8}$$

U našem slučaju izraz (3.7) glasi:

$$a_3 = \sqrt{N^2 + a_2^2 - 2 * a_2 * N * \cos(\beta)} \tag{3.9}$$

Iz izraza (3.9) sada možemo dobit izraz za izračun kuta β koji glasi :

$$\beta = \cos^{-1}\left(\frac{N^2 + a_2^2 - a_3^2}{2 * a_2 * N}\right) \tag{3.10}$$

Izraz za izračun kuta α glasi:

$$\alpha = atan2\big((P_z - d_1), R\big) \tag{3.11}$$

Nakon što smo odredili izraze za α i β možemo odrediti izraz za izračun kuta zakreta θ_2 koji glasi:

$$\theta_2 = \alpha - \beta \tag{3.12}$$

Za određivanje izraza kuta zakreta θ_3 ponovno ćemo iskoristiti kosinusov poučak koji sada glasi:

$$N = \sqrt{a_2^2 + a_3^2 - 2 * a_2 * a_3 * \cos(\pi - \theta_3)} \qquad (3.13)$$

Iz izraza (3.13) možemo dobiti izraz za izračun kuta zakreta θ_3 koji glasi:

$$\theta_3 = \cos^{-1}\left(\frac{N^2 - a_2^2 - a_3^2}{2 * a_2 * a_3}\right) \qquad (3.14)$$

Sa slike 12. vidimo da kut zakreta θ_4 možemo izračunati prema sljedećem izrazu:

$$\theta_4 = 90 - \theta_2 - \theta_3 \qquad (3.15)$$

4. AKTUATORSKI SUSTAV ROBOTSKE RUKE ERA

Aktuatorski sustav služi za postavljanje robotske ruke u određenu poziciju ili kretanje, odnosno služi za gibanje segmenata robotske ruke. Kod edukacijskih robota najčešće se koriste električna vrsta pogona, dakle koračni motori ili servomotori. Za aktuatorski dio robotske ruke ERA izabrani su koračni motori.

4.1. Koračni motor

Koračni motori su elektromehanički pretvornici energije koji električne impulse pretvaraju u diskretne mehaničke pomake. Karakteristika koračnih motora jest, da su svi namotaji dio statora, dok je rotor permanentni magnet. Ostale karakteristike koračnih motora su:

- Kada je motor pod napajanjem u mirovanju drži maksimalni moment
- Kut rotacije proporcionalan je ulaznom impulsu
- Precizno pozicioniranje i ponovljivost
- Jednostavno upravljanje
- Veliki raspon brzina
- Brzi odziv na pokretanje, zaustavljanje i promjenu smjera vrtnje

Koračni motori nemaju povratnu vezu te su stoga jeftiniji i jednostavniji za upravljanje. Upravljački program šalje dvije vrste signala koje upravljački dio koračnih motora, *driver*, koristi za pogon motora. Ti impulsi definiraju brzinu vrtnje i smjer vrtnje motora.[6]

Razlog odabira koračnih motora kao aktuatorskog dijela sustava za robotsku ruku ERA je prije svega želja za eksperimentiranjem. Osim navedenog razloga, faktor je, također, niža cijena koračnih motora s obzirom na servomotore, uzimajući u obzir podjednaki performans te iste karakteristike motora, kao što su moment, ponovljivost i jednostavnost.

Nedostatak koračnih motora u odnosu na servomotore je nizak stupanj korisnosti, potreban enkoder za povratnu informaciju položaja i sporo ubrzanje kod većih opterećenja.

4.2. *Nema* koračni motori

Kako bismo u potpunosti mogli provesti proračun nosivosti, moramo izabrati elektromotore jer njihova masa najviše utječe na izbor elektromotora. Tako su za robotsku ruku ERA izabrani koračni motori *Nema.* Razlog izboru *Nema* koračnih motora je niska cijena, pojednostavljen način fiksiranja motora na konstrukciju, dostupne informacije o svim karakteristikama i dimenzijama motora te lako planiranje konstrukcije, pouzdanost i sigurnost, veliki izbor elektromotora sa i bez reduktora i dugi životni vijek.

Nema koračni motori za svaki zglob većinom su birani iskustveno i provedbom proračuna nosivosti tako dugo, dok se nisu dobili zadovoljavajući rezultati i karakteristike izabranih motora. Tako su za robotsku ruku ERA izabrani sljedeći koračni motori:

- Rotacija baze robotske ruke – *Nema 17*
- Rotacija ramena robotske ruke – *Nema 17* sa reduktorom omjera 5:1
- Rotacija podlaktice robotske ruke – *Nema 11* sa reduktorom omjera 5:1
- Nagib alata robotske ruke – *Nema 17*
- Rotacija alata robotske ruke – *Nema 11*

4.2.1. Karakteristike koračnog motora *Nema* 17

Na slici 13. prikazan je koračni motor *Nema* 17, a u tablici 3. prikazane su njegove karakteristike.

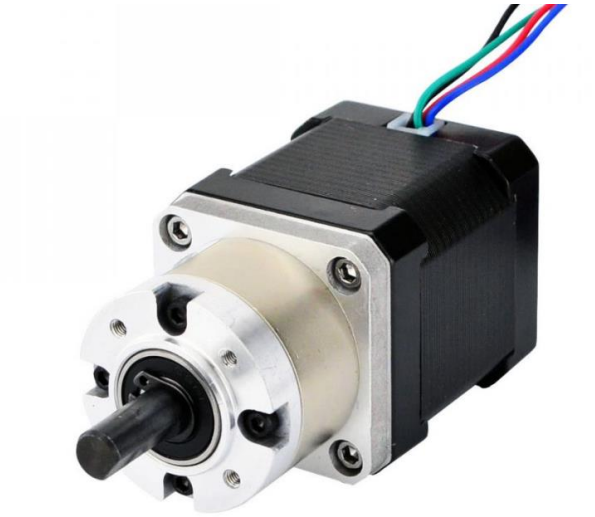
Slika 13. *Nema 17* [7]

Tablica 3. Karakteristike koračnog motora *Nema* 17 [7]

Kut koraka	1,8 °
Moment držanja	0,59 Nm
Dopuštena struja po fazi	2 A
Nazivni napon	2,8 V
Dimenzije motora	42x42x67 mm
Promjer osovine	5 mm
Masa	390 g

4.2.2. Karakteristike koračnog motora *Nema* 17 sa reduktorom 5:1

Na slici 14. prikazan je koračni motor *Nema* 17 sa reduktorom 5:1, a u tablici 4. prikazane su njegove karakteristike.



Slika 14. *Nema 17* sa reduktorom 5:1 [7]

Tablica 4. Karakteristike koračnog motora *Nema* 17 sa reduktorom 5:1 [7]

Kut koraka	0.35°
Moment držanja	2 Nm
Dopuštena struja po fazi	1,68 A
Nazivni napon	2,8 V
Dimenzije motora	42x42x95 mm
Promjer osovine	8 mm
Masa	520 g

4.2.3. Karakteristike koračnog motora *Nema* 11

Na slici 15. prikazan je koračni motor *Nema* 11, a u tablici 5. prikazane su njegove karakteristike.

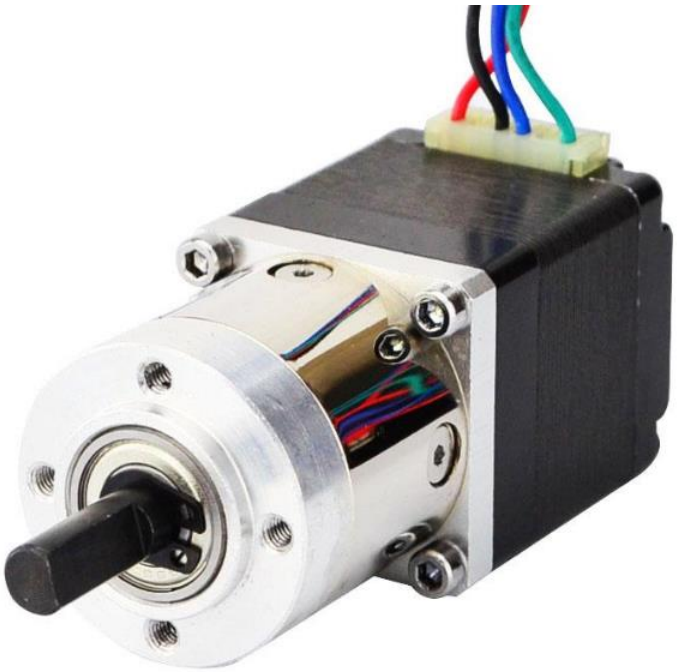
Slika 15. *Nema 11* [7]

Tablica 5. Karakteristike koračnog motora *Nema* 11 [7]

Kut koraka	1,8°
Moment držanja	0.07 Nm
Dopuštena struja po fazi	0,67 A
Nazivni napon	3,75 V
Dimenzije motora	28x28x51,5 mm
Promjer osovine	5 mm
Masa	110 g

4.2.4. Karakteristike koračnog motora *Nema* 11 sa reduktorom 5:1

Na slici 16. prikazan je koračni motor *Nema* 11 sa reduktorom 5:1, a u tablici 6. prikazane su njegove karakteristike.



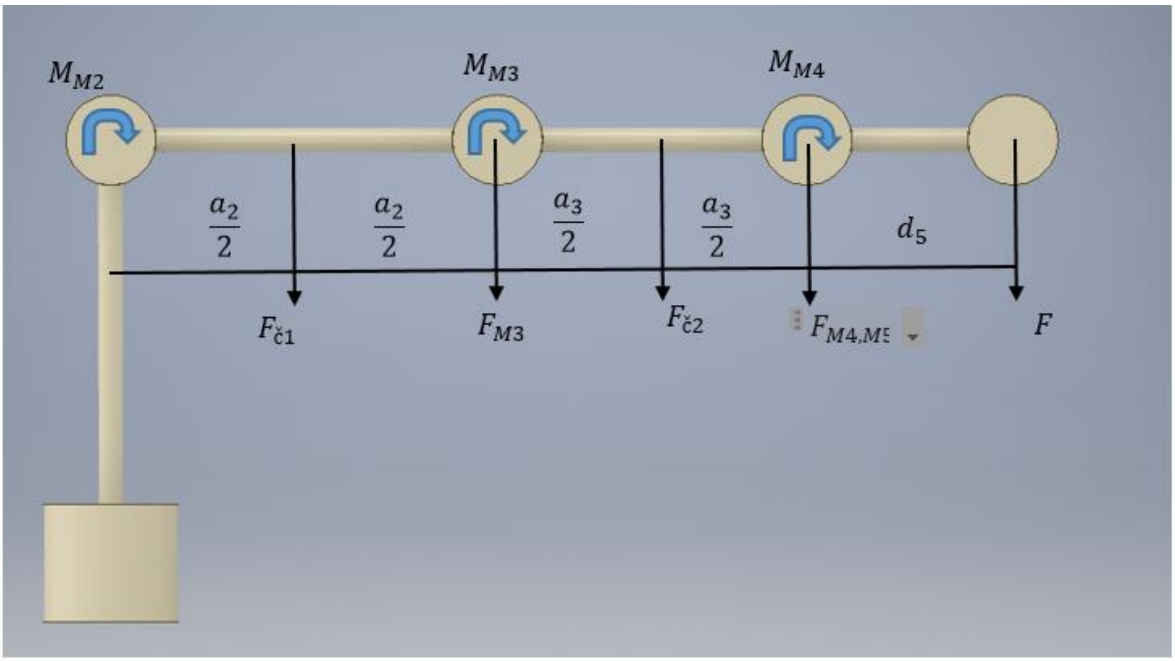
Slika 16. *Nema 11* sa reduktorom 5:1 [7]

Tablica 6. Karakteristike koračnog motora *Nema* 11 sa reduktorom 5:1 [7]

Kut koraka	0,35°
Moment držanja	2 Nm
Dopuštena struja po fazi	0,67 A
Nazivni napon	3,8 V
Dimenzije motora	30x30x84 mm
Promjer osovine	6 mm
Masa	200 g

4.3. Proračun nosivosti robotske ruke

Potrebno je napraviti proračun maksimalnog ukupnog momenta, prema kojem bismo mogli potvrditi da izabrani elektromotori zadovoljavaju uvjete. Na slici 17. prikazan je 2D prikaz robotske ruke ERA sa svim gravitacijskim silama koje djeluju na nju u najkritičnijem položaju.



Slika 17. Konstrukcija robotske ruke ERA sa gravitacijskim opterećenjem

Sile sa slike 13. imaju sljedeće vrijednosti:

$$F_{č1} = 0{,}9\ N \qquad F_{M3} = 2{,}1\ N$$

$$F_{č2} = 0{,}93\ N \qquad F_{M4,M5} = 3{,}88\ N$$

Vrijednosti dimenzija $a_2, a_3\ i\ d_5$ dane su u prijašnjem poglavlju.

Moment koračnog motora drugog zgloba treba iznositi:

$$M_{M2} > \frac{F_{č1} * \frac{a_2}{2} + F_{M3} * a_2 + F_{č2} * \left(a_2 + \frac{a_3}{2}\right) + F_{M4M5} * (a_2 + a_3)}{2} > 1{,}31\ Nm \qquad (4.1)$$

Moment M_{M2} je dvostruko manji zbog prijenosnog omjera na zupčanicima, a isto vrijedi i za moment koračnog motora M_{M3}.

Moment koračnog motora trećeg zgloba treba iznositi:

$$M_{M3} > \frac{F_{č2} * \frac{a_3}{2} + F_{M4M5} * a_3}{2} > 0{,}98\ Nm \tag{4.2}$$

Moment M_4 je moment koračnog motora koji je određen karakteristikama izabranog koračnog motora te iznosi $M_4 = 0{,}59\ Nm$

Kada smo odredili sve momente koji imaju utjecaj na nosivost robotske ruke, možemo zaključiti da je M_{M2} najkritičniji te da iz njega trebamo dobiti nosivost robotske ruke, koja iznosi:

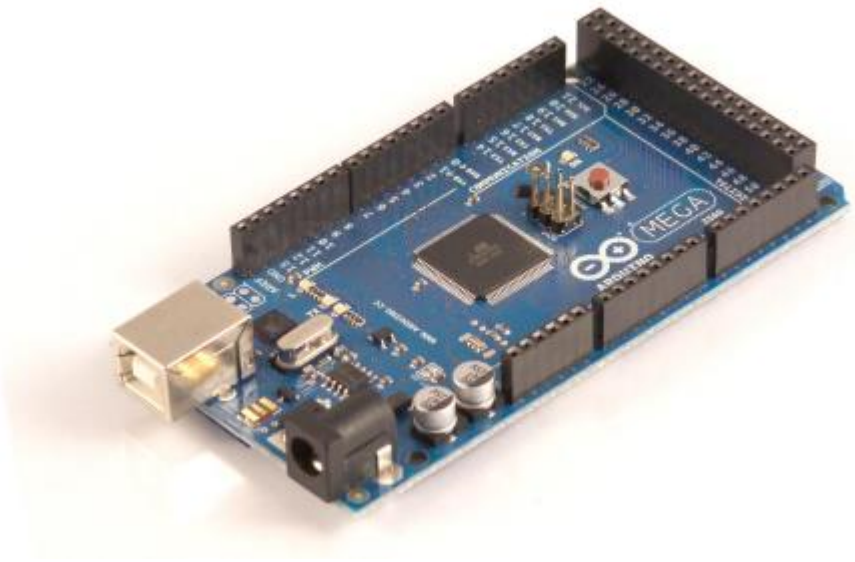
$$F = \frac{M_{M2}}{a_2 + a_3 + d_5} = 2{,}6\ N \tag{4.3}$$

Iz jednadžbe (4.3) vidimo da je nosivost robotske ruke ERA 0,26 kg.

5. ELEKTRIČNI SUSTAV ROBOTSKE RUKE ERA

5.1. Upravljačka dio

Kako bi bilo moguće upravljati robotskom rukom ERA, potreban je upravljački dio koji će upravljati radom sustava, odnosno upravljati motorima preko komunikacije sa računalom i *joystick-om.* Kriteriji za izbor upravljačke jedinice su: dovoljan broj analognih i digitalnih ulaza i izlaza, dovoljno velika memorija, posjedovanje mogućnosti komuniciranja sa računalom, sposobnost upravljanja koračnim motorima i dovoljno niska cijena. Upravljačka jedinica koja zadovoljava sve navedene uvjete je Arduino MEGA 2560.



Slika 18. Arduino MEGA 2560 [8]

Arduino MEGA 2560 je mikrokontrolerska razvojna pločica koja je moderno rješenje za obavljanje raznih zadataka širokog spektra tehničkog djelovanja. Glavna karakteristika Arduino MEGA 2560 jest njegova niska cijena i lakoća nabavljivosti s obzirom na mogućnosti korištenja.

Jezgra Arduino MEGA 2560 je mikrokontroler ATmega2560 koji može pohraniti veliku količinu memorije i s njom brzo rukovati. Arduino MEGA 2560 može se napajati preko USB-a ili preko eksternog napajanja od 7 – 12 V.

Arduino MEGA 2560 ima veliki raspon primjene od mjerenje temperature, komunikacije sa računalom i ostalim kontrolerima, do upravljanja motorima i rješavanja kompliciranih

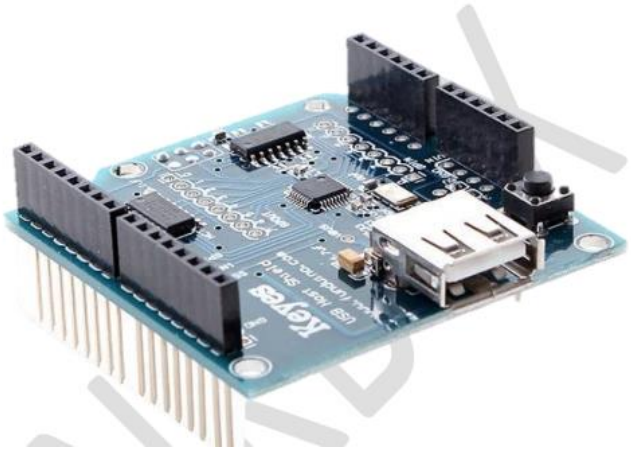
matematičkih operacija i problema. Ostale karakteristike Arduino MEGA 2560 navedene su u tablici 7.

Tablica 7. Karakteristike Arduino MEGA 2560 [8]

Mikrokontroler	ATmega2560
Radni napon	5 V
Ulazni napon	7 – 12 V
Digitalni ulazi i izlazi	54
Analogni ulazi	16
Struja ulaznih i izlaznih pinova	40 mA
Flash memorija	256 KB
SRAM memorija	8 KB
EEPROM memorija	4 KB
Radna frekvencija	16 MHz

Komunikacija Arduino MEGA 2560 sa računalom i samo programiranje kontrolera vrši se pomoću USB priključka B tipa. Arduino MEGA 2560 programira se u programskom sučelju Arduino.

Arduino MEGA 2560 ima mogućnost nadograđivanja sa gotovim elementima koji se nazivaju *Shield*-ovi. Postoji veliki broj *Shield*-ova za Arduino uređaje, ali za upravljanje robotskom rukom ERA korišten je samo jedan *Shield*, USB *Host Shield* prikazan na slici 19.



Slika 19. USB *Host Shield* [9]

USB *Host Shield* omogućuje komunikaciju uređaja koji imaju USB priključke sa Arduino-m. U našem slučaju USB *Host Shield* potreban nam je zbog komunikacije *joystick*-a sa Arduino-m, mogućeg upravljanja robotom u slučaju prezentacije ili manjka računala. USB *Host Shield* je baziran na MAX3421E kontroleru , USB *Host* kontroleru koji posjeduje digitalnu logiku i analogne krugove potrebne za implementaciju perifernog USB-a velike brzine. Prednost izabranog USB *Host Shield*-a je u tome, što digitalni i analogni ulazno/izlazni pinovi ostaju slobodni; dakle USB *Host Shield* koristi samo 6 pinova Arduino MEGA 2560 za ostvarivanja komunikacije. USB *Host Shield* prikladan je za Arduino MEGA 2560 jer u Arduino programskom sučelju već postoji *Library* koji pojednostavljuje programiranje izabranog *joystick*-a.

Za upravljanje robotskom rukom ERA preko USB *Host Shield* koristi se P3 *joystick.* Razlog odabira navedenog *joystick*-a je niska cijena i sposobnost ostvarivanja komunikacije sa Arduino MEGA 2560 preko USB *Host Shield*. Na slici 20. prikazan je *P3 joystick.*

Slika 20. P3 *joystick* [10]

5.2.Driver za koračni motor

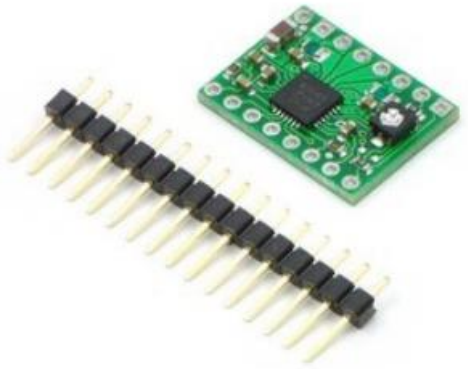
Da bismo pomoću Arduino pločice mogli upravljati sa koračnim motorima potreban nam je dodatni sklop, koji će biti posrednik između Arduino pločice, koja šalje informacije, i motora; takav sklop nazivamo *Driver*. *Nema* nudi svoje *Drivere* za Nema koračne motore proizvođača *Stepperonline*, međutim njihova cijena se kreće između 100 i 200 kuna, ovisno o željenim karakteristikama. Na slici 21. prikazan je *Stepperonline*-ov *Driver* za koračne motore Nema 11 i Nema 17.

Slika 21. *Stepperonline Driver* [7]

Svaki koračni motor mora imati svoj *Driver, Stepperonline Driver*-i bili bi preskupa opcija za edukacijsku robotsku ruku, te su odabrani jeftiniji *Driveri* A4988. *Driver* A4988 tiskana pločica koristi mikrokontroler A4988 za upravljanje bipolarnim koračnim motorima. Karakteristike koje opisuju *Driver* A4988 su mogućnost podešavanja maksimalne struje, zaštita od preopterećenja i pet različitih rezolucija upravljanja mikrokorakom. Radni napon *Driver*-a A4988 je od 8 V do 35 V i izlazna struja mu je do 2 A po jezgri. Osim navedenih karakteristika *Driver*-a A4988 treba spomenuti još slijedeće karakteristike[11]:

- Jednostavno upravljanje korakom i smjerom koračnog motora
- Pet različitih rezolucija koraka - puni korak, pola koraka, četvrtina koraka, osmina koraka i šesnaestina koraka
- Mogućnost podešavanja struje pomoću potenciometra
- Sustav za sprječavanja pregrijavanja, niskih napona i kratkih spojeva

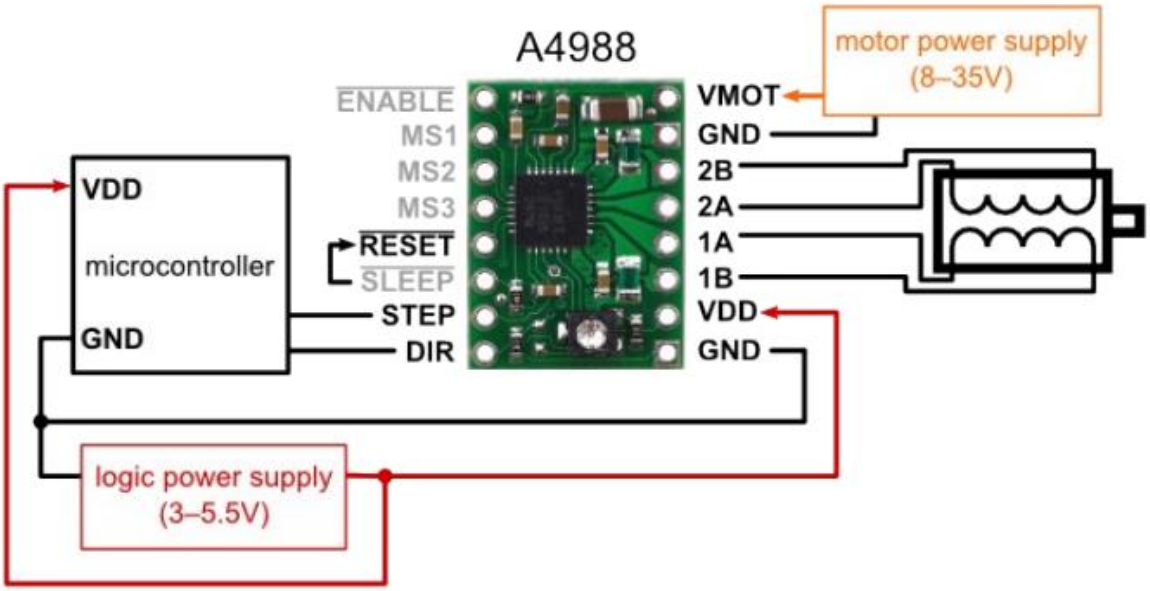
Na slici 22. prikazan je *Driver* A4988.



Slika 22. *Driver* A4988 [11]

Nedostatak *Drivera* A4988 jest pregrijavanje i potreba za izradom kućišta sa dobrim hlađenjem, no taj je nedostatak jednostavno riješiti sa aluminijskim hladnjacima ili ventilatorom, ovisno o uporabi. Glavna prednost *Drivera* A4988 jest niska cijena za dobivene karakteristike. Naime, cijena *Driver-a* A4988 je 7 kuna te se može nabaviti preko bilo kojeg većeg *online retailer*-a poput *eBay-a, Amazona*, itd.

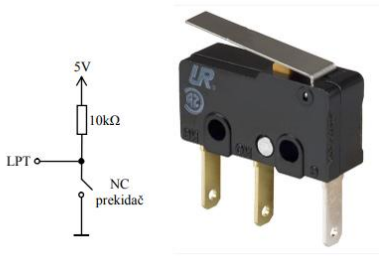
Na slici 23.prikazan je način spajanja *Driver-a* A4988 sa mikrokontrolerom i koračnim motorom u modu punog koraka.



Slika 23. Dijagram spajanja *Driver*-a A4988 sa mikrokontrolerom i koračnim motorom[11]

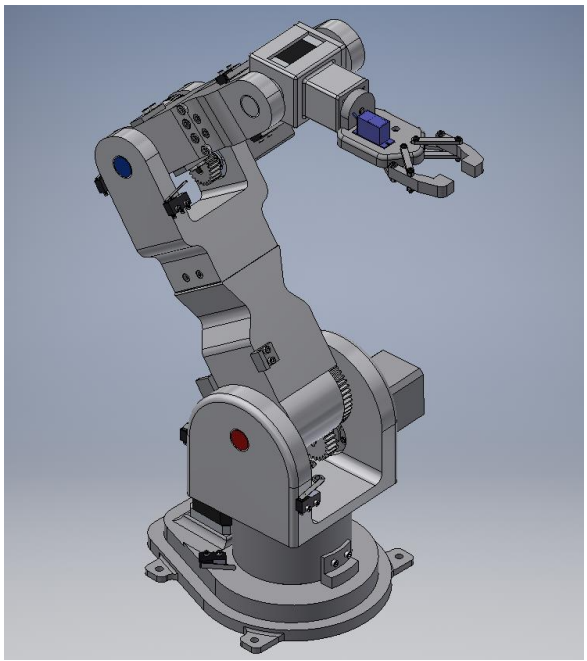
5.3. Mikroprekidači

Mikroprekidači su mehanički senzori koji se koriste za definiranja ishodište (*Home Position)* i kao graničnici *(Limit Switches)*. Mikroprekidači koriste 3 izlaza: NO (*Normally open)*, NC (*Normally Closed)* i uzemljenje (*Common)*. Na slici 24. prikazan je mikroprekidač i način spajanja mikroprekidača.



Slika 24. Prikaz spajanja mikroprekidača

Kod robotske ruke ERA koristi se 10 mikroprekidača, po dva za svaki zglob, koji služe kao graničnici i kao elementi za definiranje ishodišta robotske ruke ERA. Kalibracija ili postavljanje ishodišta robotske ruke ERA vrši se pomoću GUI sučelja robotske ruke ERA. Na slici 25. prikazani su smješteni mikroprekidači .



Slika 25. Prikaz pozicija mikroprekidača na robotskoj ruci ERA

5.4. Napajanje sustava

Električni sustav robotske ruke zahtjeva posebno napajanje, a kriterij postavljaju *Driver* A4988 i koračni motori. *Driver* A4988 radi na naponu od 8 V do 35 V, te je iz tog razloga potrebno napajanje izlaznog napona u tom rasponu. Koračni motori rade sa strujama između 1A i 2A. Iz razloga što su *Driver*-i A4988 spojeni paralelno potrebno je da napajanje može dati struju veću od 10 A. Napajanje koje zadovoljava navedene kriterije prikazano je na slici 26.

Slika 26. Napajanje sustava robotske ruke ERA [12]

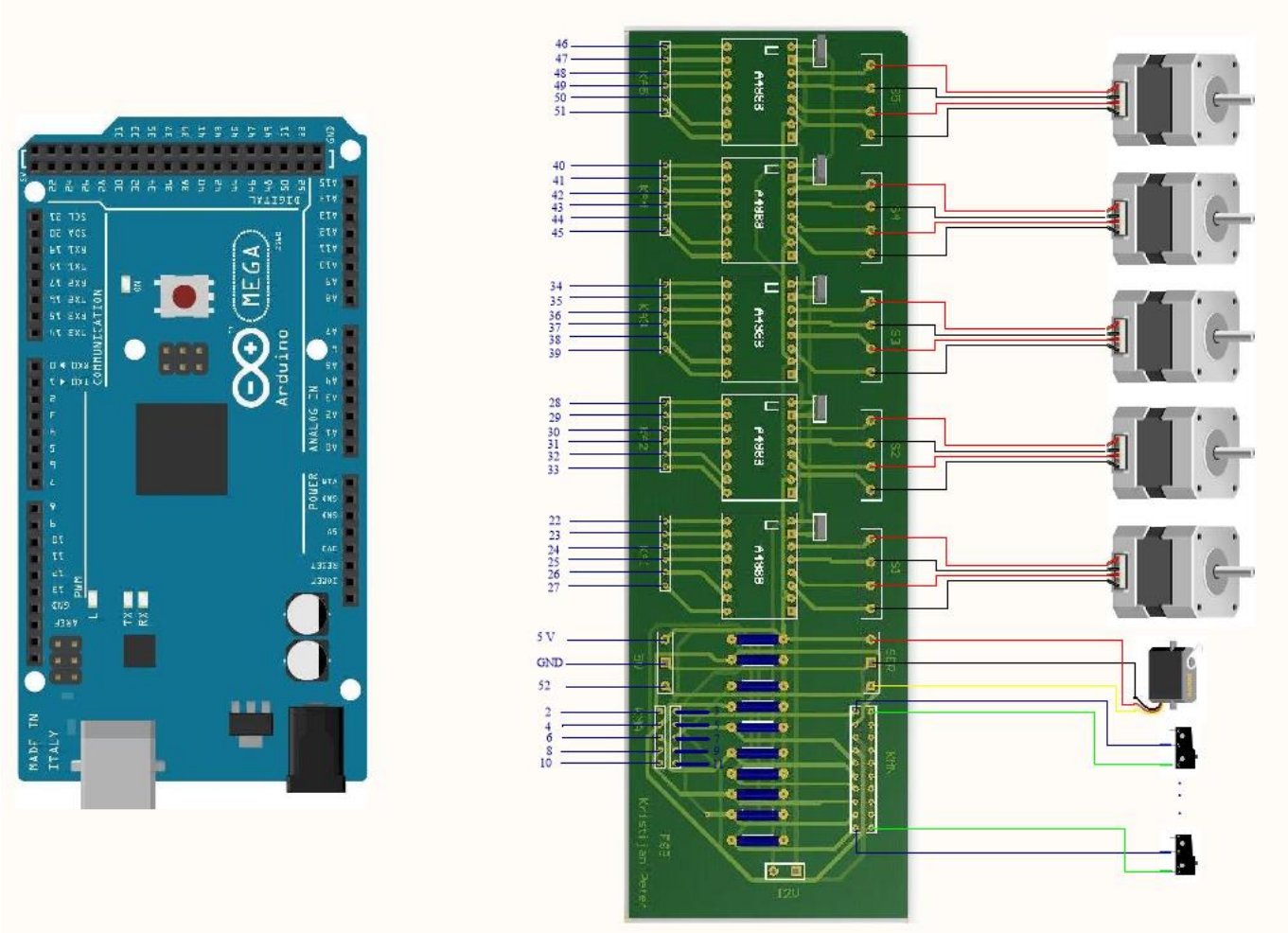
Karakteristike izabrano napajanja električnog sustava robotske ruke ERA dane su u tablici 8.

Tablica 8. Karakteristike napajanja sustava robotske ruke ERA [12]

Izlazni DC napon	12 V
Izlazna nazivna struja	30 A
Raspon izlazne struje	0 – 30 A
Nazivna snaga	360 W
Ulazni AC napon	170 – 264 V
Frekvencija	47 – 63 Hz
Korisnost	83 %
Dimenzija	215x115x50 mm

5.5. Električna shema sustava

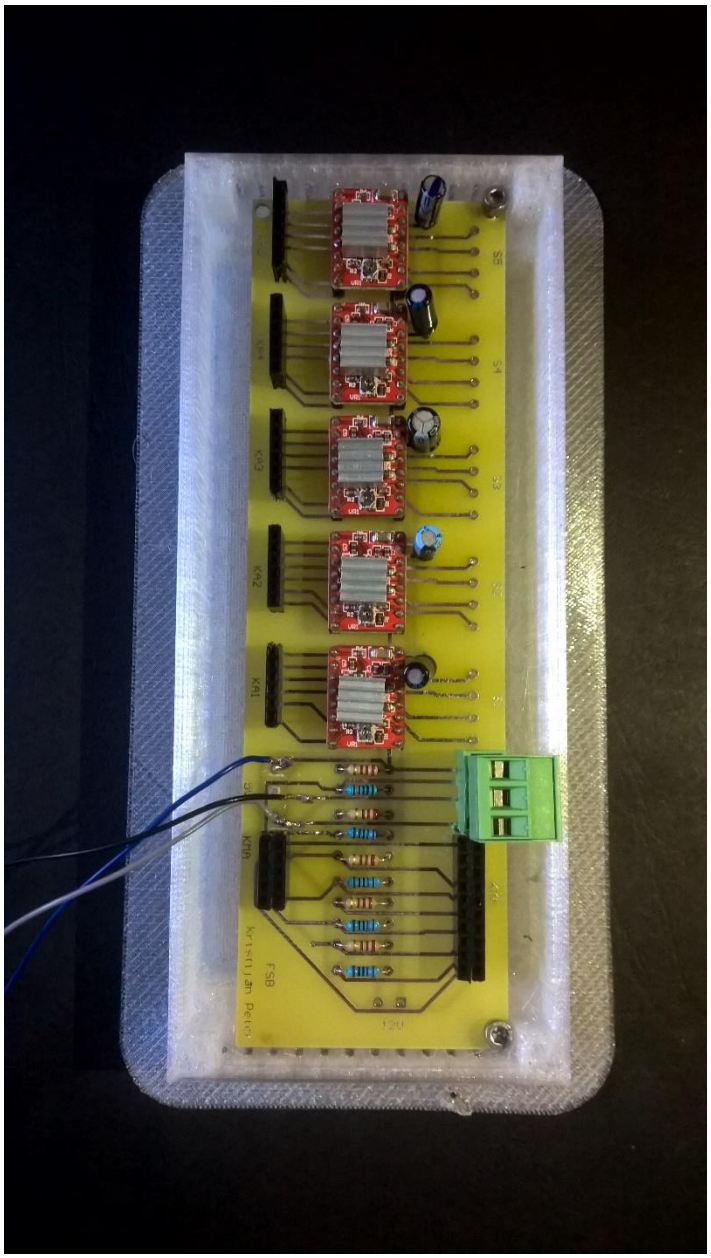
Električna shema sustava sastoji se od Arduino MEGA 2560 pločice, tiskane pločice sa *Driver*-ima A4988 i konektorima za komunikaciju sa Arduino MEGA 2560 i koračnim motorima. Na slici 27. prikazana je shema sustava za upravljanje robotskom rukom ERA.



Slika 27. Shema sustava za upravljanje

Tiskana pločica sa *Driver*-ima A4988 i konektorima za komunikaciju između Arduino MEGA 2560 i koračnih motora, rađena je u programskom paketu *Altium Designer*. Kompletna elektronička shema, nacrti tiskane pločice te ostala dokumentacija priložena je na kraju rada.

Tiskanu pločicu izradila je tvrtka *Tiplon* . Razlog ovom pristupu izrade tiskane pločice jest jednostavnost, značajna ušteda vremena, tehnološki točnija izvedba pločice, izbjegavanje kontakta sa štetnim kemikalijama, te ušteda kod nabave istih. Na slici 28. nalazi se izrađena tiskana pločica.



Slika 28. Tiskana pločica sa *Driver*-ima a4988

6. PROGRAMSKI SUSTAV ROBOTSKE RUKE ERA

U ovom poglavlju bit će opisan tok izrade grafičkog korisničkog sučelja(*GUI*) i upravljački program robotske ruke ERA. Originalna je ideja bila grafičko sučelje i upravljački dio pisati u programskom sučelju *Matlab*, međutim, budući da se radi o edukacijskom robotu, naglašena je potreba da korištenje, reprogramiranje i nadogradnje budu dostupne svima bez potrebe za kupovanjem skupih licenci. Stoga je odlučeno grafičko sučelje raditi u programskom paketu *Python*, a upravljački dio u programskom paketu *Arduino*.

Struktura sustava robotske ruke ERA sastoji se od:

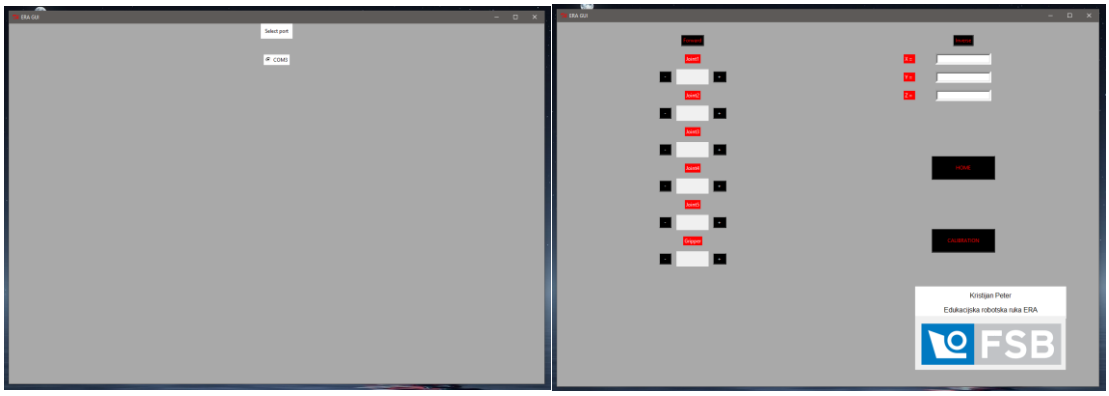
1. Računala kojem je zadatak prikazivanje podataka o sustavu i primanje naredbi od korisnika
2. Upravljačkog dijela kojem je zadatak prikupljanje podataka sa senzora i računala te obrada istih
3. Robotska ruka ERA

Slika 29. Struktura sustava robotske ruke ERA

6.1. Grafičko korisničko sučelje robotske ruke

Grafičko korisničko sučelje, skraćeno *GUI*, sastoji se od vizualnih elementa, poput ikona, gumbova i izbornika, koji predstavljaju različite naredbe. Prednost *GUI*-a je mogućnost prebacivanja s jednog zadatka na drugi, mogućost komuniciranja s više aplikacija i opcija nelinearnog izvršavanja naredbi. Glavna je prednost *GUI*-a kod edukacijskih robota njegova lakoća korištenja, te ga iz tog razloga mogu koristiti i korisnici bez iskustva i poznavanja sustava.

Kao što je spomenuto *GUI* robotske ruke ERA rađen je u programskom paketu *Python. Python* je programski jezik orijentiran objektnom programiranju, koji je snažan i istodobno jednostavan, jer spoj tradicionalnih skriptnih jezika i sistematskih jezika programeru omogućuje lakšu orijentiranost rješavanju problema. *Python* je besplatan i open-source s velikom potporom, literaturom i dokumentacijom te stoga i savršen izbor za edukacijski robot. Na slici 31. prikazano je grafičko sučelje robotske ruke ERA.



Slika 30. Grafičko sučelje robotske ruke ERA

Grafičko sučelje sastoji se od dva dijela, prvi dio(slika 31. lijevo) u kojem biramo *Port* na koji se želimo spojiti, odnosno na koji je spojen *Arduino*, i drugi dio(slika 31. desno) koji je glavno grafičko sučelje. Glavno grafičko sučelje sastoji se od 3 elemenata. Prvi element sastoji se od gumba, koji služe za upravljanje kutom zakreta koračnog motora te za otvaranje i zatvaranje hvataljke, gumba *Forward* ,koji služi sa računanje direktnog kinematičkog problema i ispisuje X, Y i Z koordinate robotske ruke u radnom prostoru , i *Label*-a koji služe za prikaz trenutnog kuta zakreta koračnog motora. Drugi element sastoji se od *Entry box*-eva

koji služe za unos željenih X,Y i Z koordinati vrha alata robotske ruke u radnom prostoru i gumba *Invers* koji služi za računanje inverznog kinematičkog problema i postavlja koračne motore u odgovarajući kut zakreta za željenu poziciju vrha alata u prostoru. Treći element sastoji se od dva gumba: *Home*, pomoću kojeg robota vraćamo u početni položaj, i *Calibration,* koji je potrebno pritisnuti kod svakog ponovnog korištenja robotske ruke, zbog kalibracije same robotske ruke.

Komunikacija *Pythona* i *Arduina* ostvarena je pomoću programskog koda *arduinoComms.py* koji iz grafičkog sučelja prima naredbe i šalje ih na *Arduino.*

Programski kod grafičkog sučelja i *arduinoComms.py* dani su u prilogu.

6.2. Upravljački program robotske ruke

Upravljački program služi prikupljanju informacija sa grafičkog sučelja robotske ruke ERA i obradi istih, te slanju informacija *Driver*-ima A4988 za upravljanje koračnim motorima.

Upravljački program napisan je u programskom paketu *Arduino*. *Arduino* programski paket služi za programiranje mikrokontrolera za sve *Arduino* pločice, na način da se *Arudino* spoji putem USB priključka na računalo. Programski jezik koji koristi programski paket *Arduino* sličan je programskim jezicima C i C++. Prednost programskog paketa *Arduino* jest posjedovanje gotovih *Library*-a za upravljanje servomotorima i koračnim motorima, open-source programsko sučelje. Također je i besplatan. Na slici 32. prikazano je programsko sučelje programskog paketa *Arduino.*

Slika 31. Sučelje programskog paketa *Arudino* [13]

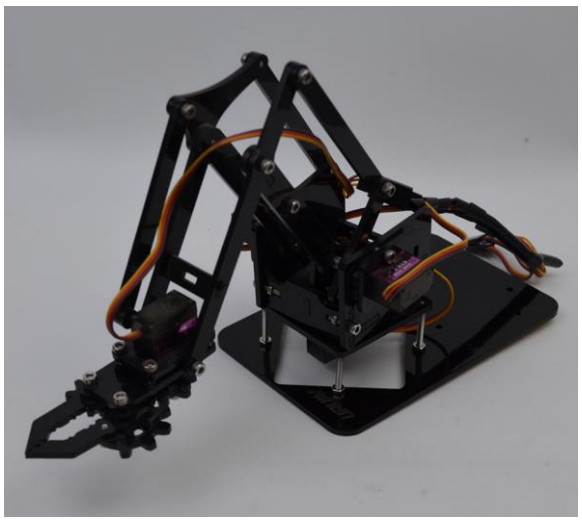
Struktura programa sastoji se od *Setup* funkcije i *Loop* funkcije. *Setup* funkcija sastoji se od definirani ulaza i izlaza na *Arduino* pločici i pokreće se samo jednom, prilikom pokretanja programa, dok se *Loop* funkcija sastoji od programa koji služi za upravljanje željenim izlazima za upravljanje motorima i senzorima te se ponavlja tako dugo dok *Arudino* pločica ima napajanje.

Programski kod upravljačkog programa dan je u prilogu

7. USPOREDBA S TRENUTNIM TRŽIŠTEM

Robotska ruka ERA uspoređena je sa trenutno najboljom edukacijskom robotskom rukom *Dobot Magician(Slika 34.)* i trenutno najjeftinijom robotskom rukom *Uruav(Slika 35.)* na tržištu. *Dobot* je vodeći proizvođač edukacijskih robotski ruka, a karakteriziraju ga odlični proizvodi i podrška, jednostavnost proizvoda i ideja za osposobljavanje studenata i učenika za Industiju 4.0. *Uruav* je *DIY* kit koji se može nabaviti preko bilo kojeg većeg *online retailer*-a poput *eBay-a, Amazona,* itd. *Uruav* karakterizira jednostavna konstrukcije i niska cijena. *Uruav* je više prilagođen za djecu.

Slika 32. Dobot Magician [2]



Slika 335. Uruav [15]

U tablici 9. dane su usporedbe svih triju robotskih ruka.

Tablica 9. Usporedba robotske ruke ERA sa tržištem

	ERA	DOBOT Magician	URUAV
Cijena	2000 kn	9750 kn	220 kn
Stupnjevi slobode gibanja	5	4	3
Vrsta	3D printan, DIY	Sastavljen	DIY
Masa	2 kg	8 kg	0,450 kg
Pogon	*Nema* koračni motori	Koračni motori	MG90S servomotori
Napajanje	220 V	100 V – 240 V	5 V
Minimalna dob korisnika	14+ godina	18 + godina	8+ godina
Upravljačka jedinica	Arduino MEGA 2560	Dobot Integrated Controller	Arduino UNO
Upravljanje	Software-sko, pomoću joysticka	Software-sko	Pomoću potenciometara
Nosivost	260 g	500 g	Nije navedeno
Mapiranje pozicija	Nema	Ima	Nema
Lokalizacija	Ima	Ima	Nema
Komunikacija	USB	USB/WiFi/Bluetooth	Nema
Software	ERA GUI	DobotStudio, Repetier Host, DobotBlockly	Nema
Alati	Hvataljka	Glava za 3D printanje, Držač olovke, Laser, Hvataljka, Vakumska sisaljka	Hvataljka
Mogućnost nadogradnje	Ima	Ima	Nema

8. ZAKLJUČAK

Industrijom 4.0 potražnja za znanjem robotike postaje sve veća te se javlja potreba za razvojem edukacijskih robota koji će mlade inženjere pripremiti za rad ovom području. Kako bi to bilo izvedivo, edukacijski roboti moraju biti jeftiniji od komercijalnih, jednostavni, nadogradivi i open-source. Ideja je da svaka srednja škola i fakultet imaju barem jedan edukacijski robot kako bi učenike i studente lakše upoznali sa robotikom.

U radu je opisan postupak izrade edukacijske robotske ruke ERA. Potrebno je bilo napraviti model robotske ruke u CAD programskom paketu, 3D printati pozicije konstrukcije i sklopiti robotsku ruku. Osim konstrukcije, potrebno je bilo odabrati aktuatorski i upravljački sustav, riješiti kinematički model i izraditi grafičko sučelje robotske ruke ERA. Aktuatorski sustav sastoji se od *Nema* koračnih motora za koje su odabrani *Driver*-i A4988, koji njima upravljaju prikupljanjem informacija sa upravljačke jedinice. Upravljačka jedinica sustava robotske ruke ERA je *Arduino MEGA 2560*. Robotskom rukom ERA može se upravljati preko grafičkog sučelja koje je izrađeno u programskom paketu *Python* ili sa *joystickom*. U radu je izveden i kinematički model robotske ruke pomoću kojeg su riješeni inverzni i direktni kinematički problemi koji služe za lokalizaciju robotske ruke ERA u njezinom radnom prostoru. Kinematički problemi bitan su dio edukacijskih robota jer je njihovim rješavanjem kasnije omogućeno spremanje točaka u prostoru i izvršavanje kompleksnijih zadataka.

Prilikom izrade robotske ruke ERA korišteno je znanje stečeno tokom studija iz područja mehanike, robotike, automatike i proučenih, već gotovih, edukacijskih robota. Zadovoljstvo mi je što sam to znanje uspio objediniti i iskoristit za izradu edukacije robotske ruke.

LITERATURA

[1] https://pcpress.rs/industrijski-roboti-produbljuju-nejednakost-plata/, 28.12.2018.

[2] https://www.dobot.cc/dobot-magician/product-overview.html, 28.12.2018.

[3] T. Šurina, M. Crneković, Industrijski roboti, školska knjiga, Zagreb, 1990., 28.12.2018.

[4] http://3dprintingforbeginners.com/filamentprimer/ , 28.12.2018.

[5] Kucuk, S., Bingul, Z.: Robot Kinematics: Forward and Inverse Kinematics, Industrial Robotics: Theory, Modelling and Control, pIV pro literatur Verlag Robert MayerScholz, Mammendorf, Germany, 2007 PWC

[6] Štefanić I.: Diplomski rad, FSB, Zagreb, 2014, 03.01.2019.

[7] https://www.omc-stepperonline.com , 03.01.2019

[8] http://www.mantech.co.za/datasheets/products/A000047.pdf , 03.01.2019.

[9] https://www.thaieasyelec.com/downloads/EFDV521/Datasheet_Keyes_USBHostShield.pdf, 03.01.2019.

[10] http://ecx.images-amazon.com/images/I/61HoEFF0GtL._SL1200_.jpg, 03.01.2019.

[11] https://www.robotshop.com/media/files/pdf/datasheet-1182.pdf, 04.01.2019.

[12] http://www.wlgled.com/estaticos/media/producto-pdf/8PS120360DCU.pdf, 04.01.2019.

[13] Macura.H.: Diplomski rad, FSB, Zagreb, 2015, 05.01.2019.

[14] Shabeeb.H.A.: Path planning of robot manipulator using Bezier technique, Iran, 2013., 05.01.2019.

[15] https://img.staticbg.com/images/oaupload/banggood/images/C3/0E/969c163e-4257-4a08-96ee-27903ec55c34.jpg, 05.01.2019.

PRILOG

1. CD ROM
2. Programski kod za upravljanje
3. Programski kod *arduinoComms.py*
4. Programski kod grafičkog sučelja
5. Tehnička dokumentacija
6. Električna shema tiskane pločice
7. PCB shema tiskane pločice

Prilog 2.

Programski kod za upravljanje

```
#include <Servo.h>
//Motor 1
#define ENB1 22
#define MS11 23
#define MS21 24
#define MS31 25
#define step_pin1 26
#define dir_pin1 27
//=============
//Motor 2
#define ENB2 28
#define MS12 29
#define MS22 30
#define MS32 31
#define step_pin2 32
#define dir_pin2 33
//=============
//Motor 3
#define ENB3 34
#define MS13 35
#define MS23 36
#define MS33 37
#define step_pin3 38
#define dir_pin3 39
//=============
//Motor 4
#define ENB4 40
#define MS14 41
#define MS24 42
#define MS34 43
#define step_pin4 44
#define dir_pin4 45
//=============
//Motor 5
#define ENB5 46
#define MS15 47
#define MS25 48
#define MS35 49
#define step_pin5 50
#define dir_pin5 51
//=============
//Servo
Servo gripper;
byte servo = 52;
//Limit switch
#define Limit01 2
#define Limit02 3
#define Limit03 4
#define Limit04 5
#define Limit05 6
#define Limit06 7
```

```
#define Limit07 8
#define Limit08 9
#define Limit09 10
#define Limit10 11
//=============
//Brzine motora
int step_speed = 1; //veci step_speed sporiji motor
//=============
//Pozicije
byte posJ1 = 0;
byte posJ2 = 0;
byte posJ3 = 0;
byte posJ4 = 0;
byte posJ5 = 0;
byte posGrip = 0;
byte newJ1pos;
byte newJ2pos;
byte newJ3pos;
byte newJ4pos;
byte newJ5pos;
byte newGrippos;
//=============
//Statusi
byte s1p = 0;
byte s1n = 0;
byte s2p = 0;
byte s2n = 0;
byte s3p = 0;
byte s3n = 0;
byte s4p = 0;
byte s4n = 0;
byte s5p = 0;
byte s5n = 0;
byte sGp = 0;
byte sGn = 0;
//=============
//Pomocne varijable
int razlika1;
int razlika2;
int razlika3;
int razlika4;
int razlika5;
int x1;
int x2;
int x3;
int x4;
int x5;
byte Calibration = 0;
byte INV = 0;
byte HOM = 0;
const byte buffSize = 40;
```

```
char inputBuffer[buffSize];
const char startMarker = '<';
const char endMarker = '>';
byte bytesRecvd = 0;
boolean readInProgress = false;
boolean newDataFromPC = false;
char messageFromPC[buffSize] = {0};
unsigned long curMillis;
unsigned long prevReplyToPCmillis = 0;
unsigned long replyToPCinterval = 1000;
//=============
void setup() {
  Serial.begin(9600);
  //Motor 1
  pinMode(MS11, OUTPUT);
  pinMode(MS21, OUTPUT);
  pinMode(MS31, OUTPUT);
  pinMode(dir_pin1, OUTPUT);
  pinMode(step_pin1, OUTPUT);
  pinMode(ENB1, OUTPUT);
  digitalWrite(ENB1, LOW);
  //=============
  //Motor 2
  pinMode(MS12, OUTPUT);
  pinMode(MS22, OUTPUT);
  pinMode(MS32, OUTPUT);
  pinMode(dir_pin2, OUTPUT);
  pinMode(step_pin2, OUTPUT);
  pinMode(ENB2, OUTPUT);
  digitalWrite(ENB2, LOW);
  //=============
  //Motor 3
  pinMode(MS13, OUTPUT);
  pinMode(MS23, OUTPUT);
  pinMode(MS33, OUTPUT);
  pinMode(dir_pin3, OUTPUT);
  pinMode(step_pin3, OUTPUT);
  pinMode(ENB3, OUTPUT);
  digitalWrite(ENB3, LOW);
  //=============
  //Motor 4
  pinMode(MS14, OUTPUT);
  pinMode(MS24, OUTPUT);
  pinMode(MS34, OUTPUT);
  pinMode(dir_pin4, OUTPUT);
  pinMode(step_pin4, OUTPUT);
  pinMode(ENB4, OUTPUT);
  digitalWrite(ENB4, LOW);
  //=============
  //Motor 5
  pinMode(MS15, OUTPUT);
```

```
  pinMode(MS25, OUTPUT);
  pinMode(MS35, OUTPUT);
  pinMode(dir_pin5, OUTPUT);
  pinMode(step_pin5, OUTPUT);
  pinMode(ENB5, OUTPUT);
  digitalWrite(ENB5, LOW);
  //=============
  //Limit switch
  pinMode(Limit01, INPUT);
  pinMode(Limit02, INPUT);
  pinMode(Limit03, INPUT);
  pinMode(Limit04, INPUT);
  pinMode(Limit05, INPUT);
  pinMode(Limit06, INPUT);
  pinMode(Limit07, INPUT);
  pinMode(Limit08, INPUT);
  pinMode(Limit09, INPUT);
  pinMode(Limit10, INPUT);
  //=============
  // servo
  gripper.attach(servo);
  //=============
  Serial.println("<Arduino is ready>");
}
//=============
void loop() {
  curMillis = millis();
  getDataFromPC();
  J1Move();
  J2Move();
  J3Move();
  J4Move();
  J5Move();
  Gripermove();
  Kalibracija();
  Inverse();
  Home();
}
//=============
void getDataFromPC() {
 // Prikupljanje podataka sa računala
  if(Serial.available() > 0) {
    char x = Serial.read();
    if (x == endMarker) {
      readInProgress = false;
      newDataFromPC = true;
      inputBuffer[bytesRecvd] = 0;
      parseData();
    }
    if(readInProgress) {
      inputBuffer[bytesRecvd] = x;
```

```
      bytesRecvd ++;
      if (bytesRecvd == buffSize) {
        bytesRecvd = buffSize - 1;
      }
    }
    if (x == startMarker) {
      bytesRecvd = 0;
      readInProgress = true;
    }
  }
}
//=============
void parseData() {
  // Odvajanje podataka
  char * strtokIndx;

  strtokIndx = strtok(inputBuffer,","); // Prvi podatak
  newJ1pos = atoi(strtokIndx); //  pretvaranje u integer

  strtokIndx = strtok(NULL, ",");
  newJ2pos = atoi(strtokIndx);

  strtokIndx = strtok(NULL, ",");
  newJ3pos = atoi(strtokIndx);

  strtokIndx = strtok(NULL, ",");
  newJ4pos = atoi(strtokIndx);

  strtokIndx = strtok(NULL, ",");
  newJ5pos = atoi(strtokIndx);

  strtokIndx = strtok(NULL, ",");
  Calibration = atoi(strtokIndx);

  strtokIndx = strtok(NULL, ",");
  newGrippos = atoi(strtokIndx);

  strtokIndx = strtok(NULL, ",");
  s1p = atoi(strtokIndx);

  strtokIndx = strtok(NULL, ",");
  s1n = atoi(strtokIndx);

  strtokIndx = strtok(NULL, ",");
  s2p = atoi(strtokIndx);

  strtokIndx = strtok(NULL, ",");
  s2n = atoi(strtokIndx);

  strtokIndx = strtok(NULL, ",");
  s3p = atoi(strtokIndx);
```

```
  strtokIndx = strtok(NULL, ",");
  s3n = atoi(strtokIndx);

  strtokIndx = strtok(NULL, ",");
  s4p = atoi(strtokIndx);

  strtokIndx = strtok(NULL, ",");
  s4n = atoi(strtokIndx);

  strtokIndx = strtok(NULL, ",");
  s5p = atoi(strtokIndx);

  strtokIndx = strtok(NULL, ",");
  s5n = atoi(strtokIndx);

  strtokIndx = strtok(NULL, ",");
  INV = atoi(strtokIndx);

  strtokIndx = strtok(NULL, ",");
  HOM = atoi(strtokIndx);
}
//=============
void replyToPC() {
  if (newDataFromPC) {
    newDataFromPC = false;
    Serial.print(" SrvPos ");
    Serial.print(newJ1pos);
    Serial.print(" Time ");
    Serial.print(curMillis >> 9);
    Serial.println(">");
  }
}
//=============
void J1Move() {
  if (s1p ==1){
        digitalWrite(dir_pin1,LOW);
        digitalWrite(step_pin1,HIGH);
        delay(step_speed);
        digitalWrite(step_pin1,LOW);
        delay(step_speed);
        digitalWrite(MS11,HIGH);
        digitalWrite(MS21,HIGH);
        digitalWrite(MS31,LOW);
  posJ1 = newJ1pos;
    }
    if (s1n ==1){
        digitalWrite(dir_pin1,HIGH);
        digitalWrite(step_pin1,HIGH);
        delay(step_speed);
        digitalWrite(step_pin1,LOW);
```

```
        delay(step_speed);
        digitalWrite(MS11,HIGH);
        digitalWrite(MS21,HIGH);
        digitalWrite(MS31,LOW);
  posJ1 = newJ1pos;
    }
}
//=============
void J2Move() {
  if (s2p ==1){
        digitalWrite(dir_pin2,LOW);
        digitalWrite(step_pin2,HIGH);
        delay(step_speed);
        digitalWrite(step_pin2,LOW);
        delay(step_speed);
        digitalWrite(MS12,HIGH);
        digitalWrite(MS22,HIGH);
        digitalWrite(MS32,LOW);
  posJ2 = newJ2pos;
    }
    if (s2n ==1){
        digitalWrite(dir_pin2,HIGH);
        digitalWrite(step_pin2,HIGH);
        delay(step_speed);
        digitalWrite(step_pin2,LOW);
        delay(step_speed);
        digitalWrite(MS12,HIGH);
        digitalWrite(MS22,HIGH);
        digitalWrite(MS32,LOW);
  posJ2 = newJ2pos;
    }
}

//=============
void J3Move() {
  if (s3p ==1){
        digitalWrite(dir_pin3,LOW);
        digitalWrite(step_pin3,HIGH);
        delay(step_speed);
        digitalWrite(step_pin3,LOW);
        delay(step_speed);
        digitalWrite(MS13,HIGH);
        digitalWrite(MS23,HIGH);
        digitalWrite(MS33,LOW);
  posJ3 = newJ3pos;
    }
    if (s3n ==1){
        digitalWrite(dir_pin3,HIGH);
        digitalWrite(step_pin3,HIGH);
        delay(step_speed);
        digitalWrite(step_pin3,LOW);
```

```
        delay(step_speed);
        digitalWrite(MS13,HIGH);
        digitalWrite(MS23,HIGH);
        digitalWrite(MS33,LOW);
  posJ3 = newJ3pos;
    }
}
//=============
void J4Move() {
  if (s4p ==1){
        digitalWrite(dir_pin4,LOW);
        digitalWrite(step_pin4,HIGH);
        delay(step_speed);
        digitalWrite(step_pin4,LOW);
        delay(step_speed);
        digitalWrite(MS14,HIGH);
        digitalWrite(MS24,HIGH);
        digitalWrite(MS34,LOW);
  posJ4 = newJ4pos;
    }
  if (s4p ==1){
        digitalWrite(dir_pin4,HIGH);
        digitalWrite(step_pin4,HIGH);
        delay(step_speed);
        digitalWrite(step_pin4,LOW);
        delay(step_speed);
        digitalWrite(MS14,HIGH);
        digitalWrite(MS24,HIGH);
        digitalWrite(MS34,LOW);
  posJ4 = newJ4pos;
    }
}
//=============

void J5Move() {
  if (s5p ==1){
        digitalWrite(dir_pin5,LOW);
        digitalWrite(step_pin5,HIGH);
        delay(step_speed);
        digitalWrite(step_pin5,LOW);
        delay(step_speed);

        digitalWrite(MS15,HIGH);
        digitalWrite(MS25,HIGH);
        digitalWrite(MS35,LOW);
  posJ5 = newJ5pos;
    }
  if (s4p ==1){
        digitalWrite(dir_pin5,HIGH);
        digitalWrite(step_pin5,HIGH);
        delay(step_speed);
```

```
        digitalWrite(step_pin5,LOW);
        delay(step_speed);

        digitalWrite(MS15,HIGH);
        digitalWrite(MS25,HIGH);
        digitalWrite(MS35,LOW);
  posJ5 = newJ5pos;
    }
}
//=============
void Gripermove() {
  if (posGrip != newGrippos) {
    posGrip = newGrippos;
    gripper.write(posGrip);
    delay(5);
  }
}
void Kalibracija(){
  if (Calibration ==1){
    if (!digitalRead(Limit01)){
      for(int x1 = 0; x1<201; x1++){
        digitalWrite(dir_pin1,HIGH);
        digitalWrite(step_pin1,HIGH);
        delay(step_speed);
        digitalWrite(step_pin1,LOW);
        delay(step_speed);

        digitalWrite(MS11,HIGH);
        digitalWrite(MS21,HIGH);
        digitalWrite(MS31,LOW);
        posJ1 = 0;
      }
    }
    else{
        digitalWrite(dir_pin1,LOW);
        digitalWrite(step_pin1,HIGH);
        delay(step_speed);
        digitalWrite(step_pin1,LOW);
        delay(step_speed);

        digitalWrite(MS11,HIGH);
        digitalWrite(MS21,HIGH);
        digitalWrite(MS31,LOW);
    }
    if (!digitalRead(Limit04)){
      for(int x2 = 0; x2<60; x2++){
        digitalWrite(dir_pin2,HIGH);
        digitalWrite(step_pin2,HIGH);
        delay(step_speed);
        digitalWrite(step_pin2,LOW);
        delay(step_speed);
```

```
      digitalWrite(MS12,HIGH);
      digitalWrite(MS22,HIGH);
      digitalWrite(MS32,LOW);
      posJ2 = 0;
    }
  }
  else{
      digitalWrite(dir_pin2,LOW);
      digitalWrite(step_pin2,HIGH);
      delay(step_speed);
      digitalWrite(step_pin2,LOW);
      delay(step_speed);

      digitalWrite(MS12,HIGH);
      digitalWrite(MS22,HIGH);
      digitalWrite(MS32,LOW);
  }
  if (!digitalRead(Limit05)){
    for(int x3 = 0; x3<500; x3++){
      digitalWrite(dir_pin3,LOW);
      digitalWrite(step_pin3,HIGH);
      delay(step_speed);
      digitalWrite(step_pin2,LOW);
      delay(step_speed);

      digitalWrite(MS13,HIGH);
      digitalWrite(MS23,HIGH);
      digitalWrite(MS33,LOW);
      posJ3 = 0;
    }
  }
  else{
      digitalWrite(dir_pin3,HIGH);
      digitalWrite(step_pin3,HIGH);
      delay(step_speed);
      digitalWrite(step_pin3,LOW);
      delay(step_speed);

      digitalWrite(MS13,HIGH);
      digitalWrite(MS23,HIGH);
      digitalWrite(MS32,LOW);
  }
  if (!digitalRead(Limit08)){
    for(int x4 = 0; x4<6; x4++){
      digitalWrite(dir_pin4,HIGH);
      digitalWrite(step_pin4,HIGH);
      delay(step_speed);
      digitalWrite(step_pin4,LOW);
      delay(step_speed);
```

```
        digitalWrite(MS14,HIGH);
        digitalWrite(MS24,HIGH);
        digitalWrite(MS34,LOW);
        posJ4 = 0;
      }
    }
    else{
        digitalWrite(dir_pin4,LOW);
        digitalWrite(step_pin4,HIGH);
        delay(step_speed);
        digitalWrite(step_pin4,LOW);
        delay(step_speed);

        digitalWrite(MS14,HIGH);
        digitalWrite(MS24,HIGH);
        digitalWrite(MS34,LOW);
    }
    Calibration = 0;
  }
}
void Inverse(){
  if(INV ==1){
    if (posJ1 < newJ1pos){
      razlika1 = newJ1pos - posJ1;
      razlika1 = razlika1/0.6;
      for (x1 = 0; x1 < razlika1;x1++){
        digitalWrite(dir_pin1,LOW);
        digitalWrite(step_pin1,HIGH);
        delayMicroseconds(step_speed);
        digitalWrite(step_pin1,LOW);
        delayMicroseconds(step_speed);
      }
      posJ1 = newJ1pos;
    }
    if (posJ1 > newJ1pos){
      razlika1 = posJ1 - newJ1pos;
      razlika1 = razlika1/0.6;
      for (x1 = 0; x1 < razlika1;x1++){
        digitalWrite(dir_pin1,LOW);
        digitalWrite(step_pin1,HIGH);
        delayMicroseconds(step_speed);
        digitalWrite(step_pin1,LOW);
        delayMicroseconds(step_speed);
      }
      posJ1 = newJ1pos;
    }
    if (posJ2 < newJ2pos){
      razlika2 = newJ2pos - posJ2;
      razlika2 = razlika2/0.18;
      for (x2 = 0; x2 < razlika2;x2++){
        digitalWrite(dir_pin2,LOW);
```

```
        digitalWrite(step_pin2,HIGH);
        delayMicroseconds(step_speed);
        digitalWrite(step_pin2,LOW);
        delayMicroseconds(step_speed);
      }
      posJ2 = newJ2pos;
    }
    if (posJ2 > newJ2pos){
      razlika2 = posJ2 - newJ2pos;
      razlika2 = razlika2/0.18;
      for (x2 = 0; x2 < razlika2;x2++){
        digitalWrite(dir_pin2,LOW);
        digitalWrite(step_pin2,HIGH);
        delayMicroseconds(step_speed);
        digitalWrite(step_pin2,LOW);
        delayMicroseconds(step_speed);
      }
      posJ2 = newJ2pos;
    }
    if (posJ3 < newJ3pos){
      razlika3 = newJ3pos - posJ3;
      razlika3 = razlika3/0.18;
      for (x3 = 0; x3 < razlika3;x3++){
        digitalWrite(dir_pin3,LOW);
        digitalWrite(step_pin3,HIGH);
        delayMicroseconds(step_speed);
        digitalWrite(step_pin3,LOW);
        delayMicroseconds(step_speed);
      }
      posJ3 = newJ3pos;
    }
    if (posJ3 > newJ3pos){
      razlika3 = posJ3 - newJ3pos;
      razlika3 = razlika3/0.18;
      for (x3 = 0; x3 < razlika3;x3++){
        digitalWrite(dir_pin3,LOW);
        digitalWrite(step_pin3,HIGH);
        delayMicroseconds(step_speed);
        digitalWrite(step_pin3,LOW);
        delayMicroseconds(step_speed);
      }
      posJ3 = newJ3pos;
    }
    if (posJ4 < newJ4pos){
      razlika4 = newJ4pos - posJ4;
      razlika4 = razlika4/1.8;
      for (x4 = 0; x4 < razlika4;x4++){
        digitalWrite(dir_pin4,LOW);
        digitalWrite(step_pin4,HIGH);
        delayMicroseconds(step_speed);
        digitalWrite(step_pin4,LOW);
```

```
        delayMicroseconds(step_speed);
      }
      posJ4 = newJ4pos;
    }
    if (posJ4 > newJ4pos){
      razlika4 = posJ4 - newJ4pos;
      razlika4 = razlika4/1.8;
      for (x4 = 0; x4 < razlika4;x4++){
        digitalWrite(dir_pin4,LOW);
        digitalWrite(step_pin4,HIGH);
        delayMicroseconds(step_speed);
        digitalWrite(step_pin4,LOW);
        delayMicroseconds(step_speed);
      }
      posJ4 = newJ4pos;
    }
  INV = 0;
  }
}
void Home(){
  if(HOM ==1){
    if (posJ1 < newJ1pos){
      razlika1 = newJ1pos - posJ1;
      razlika1 = razlika1/0.6;
      for (x1 = 0; x1 < razlika1;x1++){
        digitalWrite(dir_pin1,LOW);
        digitalWrite(step_pin1,HIGH);
        delayMicroseconds(step_speed);
        digitalWrite(step_pin1,LOW);
        delayMicroseconds(step_speed);
      }
      posJ1 = newJ1pos;
    }
    if (posJ1 > newJ1pos){
      razlika1 = posJ1 - newJ1pos;
      razlika1 = razlika1/0.6;
      for (x1 = 0; x1 < razlika1;x1++){
        digitalWrite(dir_pin1,LOW);
        digitalWrite(step_pin1,HIGH);
        delayMicroseconds(step_speed);
        digitalWrite(step_pin1,LOW);
        delayMicroseconds(step_speed);
      }
      posJ1 = newJ1pos;
    }
    if (posJ2 < newJ2pos){
      razlika2 = newJ2pos - posJ2;
      razlika2 = razlika2/0.18;
      for (x2 = 0; x2 < razlika2;x2++){
        digitalWrite(dir_pin2,LOW);
        digitalWrite(step_pin2,HIGH);
```

```
        delayMicroseconds(step_speed);
        digitalWrite(step_pin2,LOW);
        delayMicroseconds(step_speed);
      }
      posJ2 = newJ2pos;
    }
    if (posJ2 > newJ2pos){
      razlika2 = posJ2 - newJ2pos;
      razlika2 = razlika2/0.18;
      for (x2 = 0; x2 < razlika2;x2++){
        digitalWrite(dir_pin2,LOW);
        digitalWrite(step_pin2,HIGH);
        delayMicroseconds(step_speed);
        digitalWrite(step_pin2,LOW);
        delayMicroseconds(step_speed);
      }
      posJ2 = newJ2pos;
    }
    if (posJ3 < newJ3pos){
      razlika3 = newJ3pos - posJ3;
      razlika3 = razlika3/0.18;
      for (x3 = 0; x3 < razlika3;x3++){
        digitalWrite(dir_pin3,LOW);
        digitalWrite(step_pin3,HIGH);
        delayMicroseconds(step_speed);
        digitalWrite(step_pin3,LOW);
        delayMicroseconds(step_speed);
      }
      posJ3 = newJ3pos;
    }
    if (posJ3 > newJ3pos){
      razlika3 = posJ3 - newJ3pos;
      razlika3 = razlika3/0.18;
      for (x3 = 0; x3 < razlika3;x3++){
        digitalWrite(dir_pin3,LOW);
        digitalWrite(step_pin3,HIGH);
        delayMicroseconds(step_speed);
        digitalWrite(step_pin3,LOW);
        delayMicroseconds(step_speed);
      }
      posJ3 = newJ3pos;
    }
    if (posJ4 < newJ4pos){
      razlika4 = newJ4pos - posJ4;
      razlika4 = razlika4/1.8;
      for (x4 = 0; x4 < razlika4;x4++){
        digitalWrite(dir_pin4,LOW);
        digitalWrite(step_pin4,HIGH);
        delayMicroseconds(step_speed);
        digitalWrite(step_pin4,LOW);
        delayMicroseconds(step_speed);
```

```
      }
      posJ4 = newJ4pos;
    }
    if (posJ4 > newJ4pos){
      razlika4 = posJ4 - newJ4pos;
      razlika4 = razlika4/1.8;
      for (x4 = 0; x4 < razlika4;x4++){
        digitalWrite(dir_pin4,LOW);
        digitalWrite(step_pin4,HIGH);
        delayMicroseconds(step_speed);
        digitalWrite(step_pin4,LOW);
        delayMicroseconds(step_speed);
      }
      posJ4 = newJ4pos;
    }
    if (posJ5 < newJ5pos){
      razlika5 = newJ5pos - posJ5;
      razlika5 = razlika5/0.6;
      for (x5 = 0; x5 < razlika5;x5++){
        digitalWrite(dir_pin5,LOW);
        digitalWrite(step_pin5,HIGH);
        delayMicroseconds(step_speed);
        digitalWrite(step_pin5,LOW);
        delayMicroseconds(step_speed);
      }
      posJ5 = newJ5pos;
    }
    if (posJ5 > newJ1pos){
      razlika5 = posJ5 - newJ1pos;
      razlika5 = razlika5/0.6;
      for (x5 = 0; x5 < razlika5;x5++){
        digitalWrite(dir_pin5,LOW);
        digitalWrite(step_pin5,HIGH);
        delayMicroseconds(step_speed);
        digitalWrite(step_pin5,LOW);
        delayMicroseconds(step_speed);
      }
      posJ5 = newJ5pos;
    }
  HOM = 0;
  }
}
```

Prilog 3.

Programski kod *adruinoComms.py*

```
import serial
import time
import sys
import glob

# global variables for module
startMarker = 60
endMarker = 62


def valToArduino(Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,Kalibracija,
Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,statusN4,
statusP5,statusN5,Inverz,home):
    sendStr = "%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s" %(Joint1pos,
Joint2pos, Joint3pos, Joint4pos, Joint5pos,Kalibracija,Griperpos,statusP1,statusN1,
statusP2,statusN2,statusP3,statusN3,statusP4,statusN4,statusP5,statusN5,Inverz,home)
    print "SENDSTR %s" %(sendStr)

    sendToArduino(sendStr)

def listSerialPorts():
    # http://stackoverflow.com/questions/12090503/listing-available-com-ports-with-
python

    """Lists serial ports

    :raises EnvironmentError:
        On unsupported or unknown platforms
    :returns:
        A list of available serial ports
    """
    if sys.platform.startswith('win'):
        ports = ['COM' + str(i + 1) for i in range(256)]

    elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):
        # this is to exclude your current terminal "/dev/tty"
        ports = glob.glob('/dev/tty[A-Za-z]*')

    elif sys.platform.startswith('darwin'):
        ports = glob.glob('/dev/tty.*')

    else:
        raise EnvironmentError('Unsupported platform')

    result = []
    for port in ports:
        try:
            s = serial.Serial(port)
            s.close()
            result.append(port)
        except (OSError, serial.SerialException):
            pass
    return result

#=======================

def setupSerial(serPort):

    global  ser
```

```
    # NOTE the user must ensure that the serial port and baudrate are correct
    #~ serPort = "/dev/ttyS81"
    baudRate = 9600
    ser = serial.Serial(serPort, baudRate)
    print "Serial port " + serPort + " opened  Baudrate " + str(baudRate)

    waitForArduino()

#========================

def closeSerial():

    global ser
    if 'ser' in globals():
        ser.close()
        print "Serial Port Closed"
    else:
        print "Serial Port Not Opened"

#========================

def sendToArduino(sendStr):

    global startMarker, endMarker, ser

    ser.write(chr(startMarker))
    ser.write(sendStr)
    ser.write(chr(endMarker))


#============================

def recvFromArduino(timeOut): # timeout in seconds eg 1.5

    global startMarker, endMarker, ser

    dataBuf = ""
    x = "z" # any value that is not an end- or startMarker
    byteCount = -1 # to allow for the fact that the last increment will be one too many
    startTime = time.time()
    #~ print "Start %s" %(startTime)

    # wait for the start marker
    while  ord(x) != startMarker:
        if time.time() - startTime >= timeOut:
            return('<<')
        x = ser.read()


    # save data until the end marker is found
    while ord(x) != endMarker:
        if time.time() - startTime >= timeOut:
            return('>>')
        if ord(x) != startMarker:
            dataBuf = dataBuf + x
        x = ser.read()

    return(dataBuf)

#============================
```

```
def waitForArduino():

    # wait until the Arduino sends 'Arduino Ready' - allows time for Arduino reset
    # it also ensures that any bytes left over from a previous message are discarded

     print "Waiting for Arduino to reset"

     msg = ""
     while msg.find("Arduino is ready") == -1:

         msg = recvFromArduino(10)

         print msg
         print
```

Prilog 4.

Programski kod grafičkog sučelja

```
from Tkinter import *
from arduinoComms import *
import atexit
from numpy import *

def exit_handler():
    print 'My application is ending!'
    closeSerial()
    atexit.register(exit_handler)


#pozicije
Joint1pos = 0
Joint2pos = 0
Joint3pos = 0
Joint4pos = 0
Joint5pos = 0
Griperpos = 0

# status
Kalibracija = 0
Inverz = 0
home = 0
statusP1 = 0
statusN1 = 0
statusP2 = 0
statusN2 = 0
statusP3 = 0
statusN3 = 0
statusP4 = 0
statusN4 = 0
statusP5 = 0
statusN5 = 0









tkArd = Tk()
tkArd.minsize(width=1280, height=860)
tkArd.config(bg = 'white')
tkArd.title("ERA GUI")
var = DoubleVar()
def setupView():
    global masterframe
    masterframe = Frame(width=860, height=860,bg = "dark grey")
    masterframe.pack(fill=BOTH,expand=1)

    selectPort()
def runProgram():
    tkArd.mainloop()



def selectPort():
```

```
    global masterframe, radioVar
    for child in masterframe.winfo_children():
        child.destroy()
    radioVar = StringVar()

    lst = listSerialPorts()

    l1= Label(masterframe, text="Select port",width = 10, height = 2, bg = "white")
    l1.pack()
##        r1 = Radiobutton(masterframe,  variable=radioVar,  bg = "white")
##        r1.config(command = radioPress)
##        r1.pack(anchor=W)

    if len(lst) > 0:
        for n in lst:
            r1 = Radiobutton(masterframe, text=n, variable=radioVar, value=n, bg =
"white")
            r1.config(command = radioPress)
            r1.place(relx=.5, rely=.1, anchor="c")
    else:
            l2 = Label(masterframe, text = "No Serial Port Found")
            l2.pack()


def mainScreen():
    global masterframe

    for child in masterframe.winfo_children():
        child.destroy()
        def Cali():
            global Joint1pos,Joint2pos, Joint3pos, Joint4pos, Joint5pos, Kalibracija,
Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,statusN4,
statusP5,statusN5,Inverz,home


            Kalibracija = 1

            Joint1pos = 0
            Joint2pos = 0
            Joint3pos = 0
            Joint4pos = 0
            Joint5pos = 0


            Pozicija1.config(text=Joint1pos)
            Pozicija2.config(text=Joint2pos)
            Pozicija3.config(text=Joint3pos)
            Pozicija4.config(text=Joint4pos)
            Pozicija4.config(text=Joint5pos)


            valToArduino( Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,
Kalibracija,Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,
statusN4,statusP5,statusN5,Inverz,home)
            Kalibracija = 0

        def stop(event):
            global Joint1pos,Joint2pos, Joint3pos, Joint4pos, Joint5pos, Kalibracija,
Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,statusN4,
statusP5,statusN5,Inverz,home
```

```
            statusP1 = 0
            statusN1 = 0
            statusP2 = 0
            statusN2 = 0
            statusP3 = 0
            statusN3 = 0
            statusP4 = 0
            statusN4 = 0
            statusP5 = 0
            statusN5 = 0
            statusP6 = 0
            statusN6 = 0


            valToArduino( Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,
Kalibracija,Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,
statusN4,statusP5,statusN5,Inverz,home)



        def J1P(event):
            global Joint1pos,Joint2pos, Joint3pos, Joint4pos, Joint5pos, Kalibracija,
Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,statusN4,
statusP5,statusN5,Inverz,home
            statusP1 = 1
            Joint1pos +=1
            Pozicija1.config(text=str(Joint1pos))

            valToArduino( Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,
Kalibracija,Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,
statusN4,statusP5,statusN5,Inverz,home)

        def J1N(Event):
            global Joint1pos,Joint2pos, Joint3pos, Joint4pos, Joint5pos, Kalibracija,
Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,statusN4,
statusP5,statusN5,Inverz,home
            statusN1 = 1
            Joint1pos -=1
            Pozicija1.config(text=str(Joint1pos))


            valToArduino( Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,
Kalibracija,Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,
statusN4,statusP5,statusN5,Inverz,home)
        def J2P(event):
            global  Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos, Kalibracija,
Griperpos
                statusP2 = 1
                Joint2pos += 1
                Pozicija2.config(text=Joint2pos)
            valToArduino( Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,
Kalibracija,Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,
statusN4,statusP5,statusN5,Inverz,home)


        def J2N(event):
            global  Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos, Kalibracija,
Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,statusN4,
statusP5,statusN5,Inverz,home
```

```
                statusN2 = 1
                Joint2pos -= 1
                Pozicija2.config(text=Joint2pos)
            valToArduino( Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,
Kalibracija,Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,
statusN4,statusP5,statusN5,Inverz,home)

        def J3P(event):
            global  Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos, Kalibracija,
Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,statusN4,
statusP5,statusN5,Inverz,home
                statusP3 = 1
                Joint3pos += 1
                Pozicija3.config(text=Joint3pos)
            valToArduino( Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,
Kalibracija,Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,
statusN4,statusP5,statusN5,Inverz,home)


        def J3N(event):
            global  Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos, Kalibracija,
Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,statusN4,
statusP5,statusN5,Inverz,home
                statusN3 = 1
                Joint3pos -= 1
                Pozicija3.config(text=Joint3pos)
            valToArduino( Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,
Kalibracija,Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,
statusN4,statusP5,statusN5,Inverz,home)

        def J4P(event):
            global  Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos, Kalibracija,
Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,statusN4,
statusP5,statusN5,Inverz,home
                statusP4 = 1
                Joint4pos += 1
                Pozicija4.config(text=Joint4pos)
            valToArduino( Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,
Kalibracija,Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,
statusN4,statusP5,statusN5,Inverz,home)

        def J4N(event):
            global  Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos, Kalibracija,
Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,statusN4,
statusP5,statusN5,Inverz,home
                statusN4 = 1
                Joint4pos -= 1
                Pozicija4.config(text=Joint4pos)
            valToArduino( Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,
Kalibracija,Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,
statusN4,statusP5,statusN5,Inverz,home)

        def J5P(event):
            global  Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos, Kalibracija,
Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,statusN4,
statusP5,statusN5,Inverz,home
                statusP5 = 1
                Joint5pos += 1
                Pozicija5.config(text=Joint5pos)
            valToArduino( Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,
Kalibracija,Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,
statusN4,statusP5,statusN5,Inverz,home)
```

```
        def J5N(event):
            global  Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos, Kalibracija,
Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,statusN4,
statusP5,statusN5,Inverz,home
                statusN5 = 1
                Joint5pos -= 1
                Pozicija5.config(text=Joint5pos)
            valToArduino( Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,
Kalibracija,Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,
statusN4,statusP5,statusN5,Inverz,home)
        def GRP(event):
            global  Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos, Kalibracija,
Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,statusN4,
statusP5,statusN5,Inverz,home

                Griperpos += 1
                PozicijaGrip.config(text=Griperpos)
            valToArduino( Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,
Kalibracija,Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,
statusN4,statusP5,statusN5,Inverz,home)

        def GRN(event):
            global  Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos, Kalibracija,
Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,statusN4,
statusP5,statusN5,Inverz,home

                Griperpos -= 1
                PozicijaGrip.config(text=Griperpos)
            valToArduino( Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,
Kalibracija,Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,
statusN4,statusP5,statusN5,Inverz,home)


        def FOW():
            global masterframe,Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,
Inverz
            x = StringVar()
            y= StringVar()
            z = StringVar()

            a1=100
            a2=225
            a3=215
            a4=50
            a5=80+a4
            pi = 3.14159265359

            Theta1 = Joint1pos
            Theta2= Joint2pos
            Theta3= Joint3pos
            Theta4= Joint4pos
            Theta5= Joint5pos

            T1 = float(Theta1)
            T2 = float(Theta2)
            T3 = float(Theta3)
            T4 = float(Theta4)
            T5 = float(Theta5)
```

```
T1 = (T1/180)*pi

T2 = (T2/180)*pi
T3 = (T3/180)*pi
T4 = (T4/180)*pi
T5 = (T5/180)*pi


R0_1= [[cos(T1),0,sin(T1)],[sin(T1),0,-cos(T1)],[0,1,0]]
R1_2= [[cos(T2),-sin(T2),0],[sin(T2),cos(T2),0],[0,0,1]]
R2_3= [[cos(T3),-sin(T3),0],[sin(T3),cos(T3),0],[0,0,1]]
R3_4= [[cos(T4),0,sin(T4)],[sin(T4),0,-cos(T4)],[0,1,0]]
R4_5= [[cos(T5),-sin(T5),0],[sin(T5),cos(T5),0],[0,0,1]]



d0_1 = [[0],[0],[a1]]
d1_2 = [[a2*cos(T2)],[a2*sin(T2)],[0]]
d2_3 = [[a3*cos(T3)],[a3*sin(T3)],[0]]
d3_4 = [[0],[0],[0]]
d4_5 = [[0],[0],[a5]]

c1 = cos(T1)
c2 = cos(T2)
c3 = cos(T3)
c4 = cos(T4)
c5 = cos(T5)
s1 = sin(T1)
s2 = sin(T2)
s3 = sin(T3)
s4 = sin(T4)
s5 = sin(T5)
s234 = sin(T2+T3+T4)
c234 = cos(T2+T3+T4)



H0_1 = concatenate((R0_1, d0_1),1)
H0_1 = concatenate((H0_1,[[0,0,0,1]]),0)


H1_2 = concatenate((R1_2, d1_2),1)
H1_2 = concatenate((H1_2, [[0,0,0,1]]),0)

H2_3 = concatenate((R2_3, d2_3),1)
H2_3 = concatenate((H2_3, [[0,0,0,1]]),0)

H3_4 = concatenate((R3_4, d3_4),1)
H3_4 = concatenate((H3_4,[[0,0,0,1]]),0)

H4_5 = concatenate((R4_5, d4_5),1)
H4_5 = concatenate((H4_5, [[0,0,0,1]]),0)

H0_2 = dot(H0_1,H1_2)
H2_4 = dot(H2_3,H3_4)
H0_4 = dot(H0_2,H2_4)
H = dot(H0_4,H4_5)
```

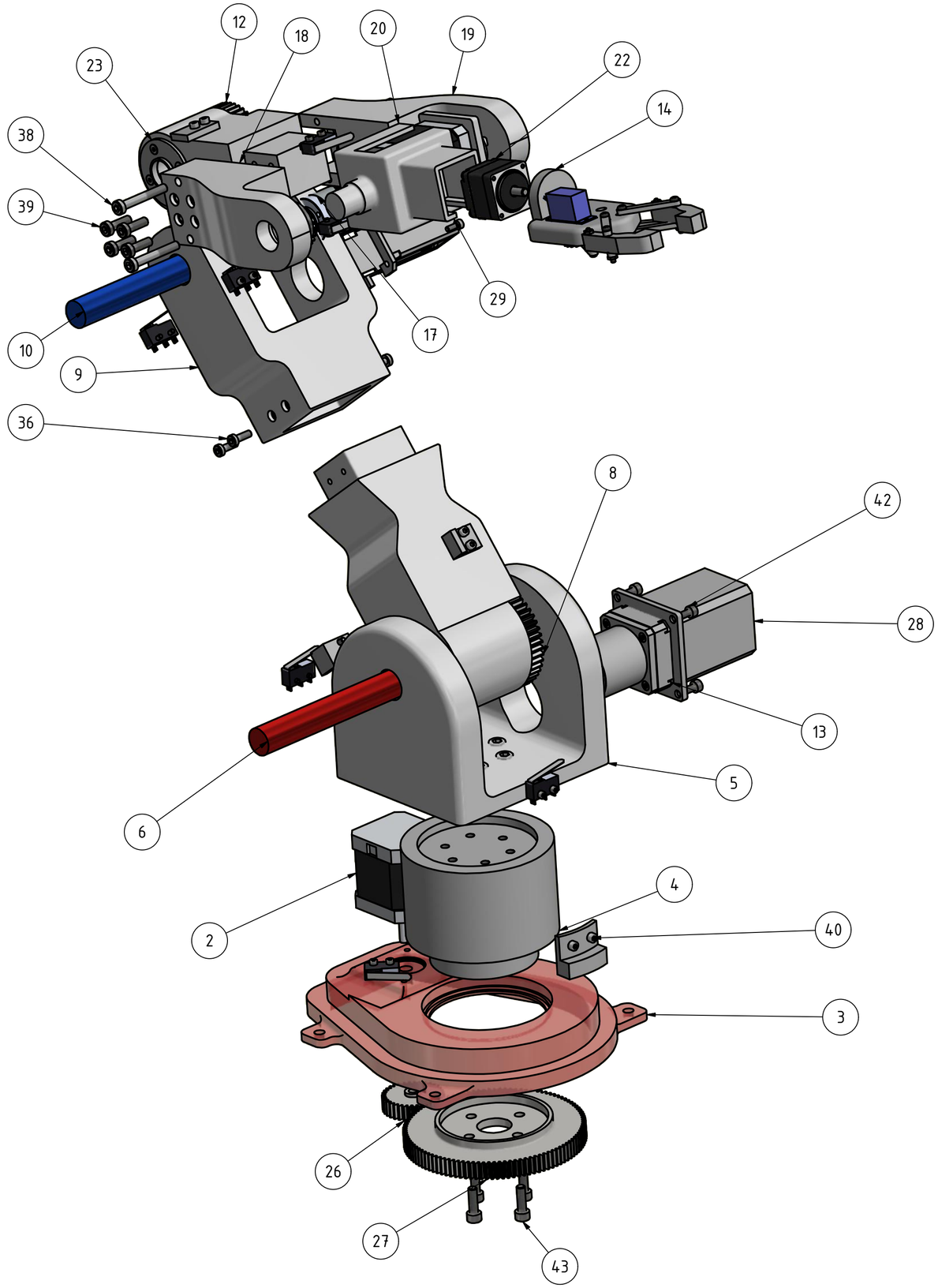
```
            PosX= int(H[0][3])
            PosY = int(H[1][3])
            PosZ =  int(H[2][3])

            x.set(str(PosX))
            y.set(str(PosY))
            z.set(str(PosZ))

            labelX = Label(masterframe, textvariable=x,bg="white", fg="black",width =
10, height = 2)
            labelY = Label(masterframe, textvariable=y,bg="white", fg="black",width =
10, height = 2)
            labelZ = Label(masterframe, textvariable=z,bg="white", fg="black",width =
10, height = 2)

            labelX.place(relx=.85, rely=.10, anchor="c")
            labelY.place(relx=.85, rely=.15, anchor="c")
            labelZ.place(relx=.85, rely=.2, anchor="c")


        def INV():
            global  Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos, Kalibracija,
Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,statusN4,
statusP5,statusN5,Inverz,home
            a1=100
            a2=225
            a3=215
            a4=50
            a5=80+a4
            pi = 3.14159265359

            Xinv = X.get()
            Yinv = Y.get()
            Zinv = Z.get()

            Xinv = int(Xinv)
            Yinv = int(Yinv)
            Zinv = int(Zinv)


            Ph = 0
            T234 = 0.5*pi+Ph


            R = sqrt(power(Xinv,2)+power(Yinv,2))-a5*cos(Ph)
            N = sqrt(power((Zinv-a1),2)+power(R,2))


            alfa = arctan2((Zinv-a1),R)
            beta = arccos((power(N,2)+power(a2,2)-power(a3,2))/(2*a2*N))



            T1 =arctan2(Yinv,Xinv)
            T2 = alfa - beta

            T3 = arccos((power(N,2)-power(a2,2)-power(a3,2))/(2*a2*a3))
```

```
            T4 = T234-T2-T3



            T1 = T1*180/pi
            T2 = T2*(180/pi)
            T3 = T3*(180/pi)
            T4 = T4*(180/pi)



            Joint1pos = int(T1)
            Joint2pos = int(T2)
            Joint3pos = int(T3)
            Joint4pos = int(T4)

            Pozicija1.config(text=Joint1pos)
            Pozicija2.config(text=Joint2pos)
            Pozicija3.config(text=Joint3pos)
            Pozicija4.config(text=Joint4pos)

            Inverz = 1

            valToArduino( Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,
Kalibracija,Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,
statusN4,statusP5,statusN5,Inverz,home)


        def Home():
            global  Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos, Kalibracija,
Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,statusN4,
statusP5,statusN5,Inverz,home

            Joint1pos = 0
            Joint2pos = 0
            Joint3pos = 0
            Joint4pos = 0
            Joint5pos = 0


            Pozicija1.config(text=Joint1pos)
            Pozicija2.config(text=Joint2pos)
            Pozicija3.config(text=Joint3pos)
            Pozicija4.config(text=Joint4pos)
            Pozicija4.config(text=Joint5pos)

            home = 1

            valToArduino( Joint1pos, Joint2pos, Joint3pos, Joint4pos, Joint5pos,
Kalibracija,Griperpos,statusP1,statusN1,statusP2,statusN2,statusP3,statusN3,statusP4,
statusN4,statusP5,statusN5,Inverz,home)




        labelJ1 = Label(masterframe, text="Joint1",bg="red", fg="white")
        labelJ2 = Label(masterframe, text="Joint2",bg="red", fg="white")
```

```
        labelJ3 = Label(masterframe, text="Joint3",bg="red", fg="white")
        labelJ4 = Label(masterframe, text="Joint4",bg="red", fg="white")
        labelJ5 = Label(masterframe, text="Joint5",bg="red", fg="white")
        labelGriper = Label(masterframe, text="Gripper",bg="red", fg="white")

        labelxje = Label(masterframe, text="X = ",bg="red", fg="white")
        labelyje = Label(masterframe, text="Y = ",bg="red", fg="white")
        labelzje = Label(masterframe, text="Z = ",bg="red", fg="white")

        Pozicija1 = Label(masterframe, text="  0  ",width = 10, height = 2)
        Pozicija2 = Label(masterframe, text="  0  ",width = 10, height = 2)
        Pozicija3 = Label(masterframe, text="  0  ",width = 10, height = 2)
        Pozicija4 = Label(masterframe, text="  0  ",width = 10, height = 2)
        Pozicija5 = Label(masterframe, text="  0  ",width = 10, height = 2)
        PozicijaGrip = Label(masterframe, text="  0  ",width = 10, height = 2)


        labelPotpis = Label(masterframe, text="Kristijan Peter",bg="white", fg="black",
width = 39, height = 2,font = 10)
        labelNaziv = Label(masterframe, text="Edukacijska robotska ruka ERA",
bg="white", fg="black",width = 39, height = 2,font =10)
        photo = PhotoImage(file="FSB.gif")
        slika = Label(masterframe, image=photo,width = 352)
        slika.photo = photo

        X = Entry(masterframe,bd = 5)
        Y = Entry(masterframe,bd = 5)
        Z = Entry(masterframe,bd = 5)



        FowKin = Button(masterframe, text="Forward", fg="red", bg="black")
        FowKin.config(command = FOW)

        InvKin = Button(masterframe, text="Inverse", fg="red", bg="black")
        InvKin.config(command = INV)




        HomeBut = Button(masterframe, text="HOME", fg="red", bg="black",height = 3,
width = 20)
        HomeBut.config(command = Home)

        CaliBut = Button(masterframe, text="CALIBRATION", fg="red", bg="black",height
= 3, width = 20)
        CaliBut.config(command = Cali)


        Joint1Pos = Button(masterframe, text="  +  ", fg="white", bg="black")
        Joint1Pos.bind('<ButtonPress-1>',J1P)
        Joint1Pos.bind('<ButtonRelease-1>',stop)




        Joint1Neg = Button(masterframe, text="  -  ", fg="white", bg="black")
        Joint1Neg.bind('<ButtonPress-1>',J1N)
        Joint1Neg.bind('<ButtonRelease-1>',stop)
```

```
Joint2Pos = Button(masterframe, text="  +  ", fg="white", bg="black")
Joint2Pos.bind('<ButtonPress-1>',J2P)
Joint2Pos.bind('<ButtonRelease-1>',stop)

Joint2Neg = Button(masterframe, text="  -  ", fg="white", bg="black")
Joint2Neg.bind('<ButtonPress-1>',J2N)
Joint2Neg.bind('<ButtonRelease-1>',stop)


Joint3Pos = Button(masterframe, text="  +  ", fg="white", bg="black")
Joint3Pos.bind('<ButtonPress-1>',J3P)
Joint3Pos.bind('<ButtonRelease-1>',stop)

Joint3Neg = Button(masterframe, text="  -  ", fg="white", bg="black")
Joint3Neg.bind('<ButtonPress-1>',J3N)
Joint3Neg.bind('<ButtonRelease-1>',stop)


Joint4Pos = Button(masterframe, text="  +  ", fg="white", bg="black")
Joint4Pos.bind('<ButtonPress-1>',J4P)
Joint4Pos.bind('<ButtonRelease-1>',stop)

Joint4Neg = Button(masterframe, text="  -  ", fg="white", bg="black")
Joint4Neg.bind('<ButtonPress-1>',J4N)
Joint4Neg.bind('<ButtonRelease-1>',stop)



Joint5Pos = Button(masterframe, text="  +  ", fg="white", bg="black")
Joint5Pos.bind('<ButtonPress-1>',J5P)
Joint5Pos.bind('<ButtonRelease-1>',stop)

Joint5Neg = Button(masterframe, text="  -  ", fg="white", bg="black")
Joint5Neg.bind('<ButtonPress-1>',J5N)
Joint5Neg.bind('<ButtonRelease-1>',stop)


GriperPos = Button(masterframe, text="  +  ", fg="white", bg="black")
GriperPos.bind('<ButtonPress-1>',GRP)
GriperPos.bind('<ButtonRelease-1>',stop)

GriperNeg = Button(masterframe, text="  -  ", fg="white", bg="black")
GriperNeg.bind('<ButtonPress-1>',GRN)
GriperNeg.bind('<ButtonRelease-1>',stop)



FowKin.place(relx=.25, rely=.05, anchor="c")

InvKin.place(relx=.75, rely=.05, anchor="c")

labelJ1.place(relx=.25, rely=.10, anchor="c")
Joint1Pos.place(relx=.3, rely=.15, anchor="c")
Joint1Neg.place(relx=.2, rely=.15, anchor="c")
```

```
            Pozicija1.place(relx=.25, rely=.15, anchor="c")


            labelJ2.place(relx=.25, rely=.20, anchor="c")
            Joint2Pos.place(relx=.3, rely=.25, anchor="c")
            Joint2Neg.place(relx=.2, rely=.25, anchor="c")
            Pozicija2.place(relx=.25, rely=.25, anchor="c")

            labelJ3.place(relx=.25, rely=.30, anchor="c")
            Joint3Pos.place(relx=.3, rely=.35, anchor="c")
            Joint3Neg.place(relx=.2, rely=.35, anchor="c")
            Pozicija3.place(relx=.25, rely=.35, anchor="c")

            labelJ4.place(relx=.25, rely=.40, anchor="c")
            Joint4Pos.place(relx=.3, rely=.45, anchor="c")
            Joint4Neg.place(relx=.2, rely=.45, anchor="c")
            Pozicija4.place(relx=.25, rely=.45, anchor="c")

            labelJ5.place(relx=.25, rely=.50, anchor="c")
            Joint5Pos.place(relx=.3, rely=.55, anchor="c")
            Joint5Neg.place(relx=.2, rely=.55, anchor="c")
            Pozicija5.place(relx=.25, rely=.55, anchor="c")

            labelGriper.place(relx=.25, rely=.60, anchor="c")
            GriperPos.place(relx=.3, rely=.65, anchor="c")
            GriperNeg.place(relx=.2, rely=.65, anchor="c")
            PozicijaGrip.place(relx=.25, rely=.65, anchor="c")



            HomeBut.place(relx=.75, rely=.4, anchor="c")

            CaliBut.place(relx=.75, rely=.6, anchor="c")

            X.place(relx=.75, rely=.1, anchor="c")
            Y.place(relx=.75, rely=.15, anchor="c")
            Z.place(relx=.75, rely=.2, anchor="c")

            labelxje.place(relx=.65, rely=.1, anchor="c")
            labelyje.place(relx=.65, rely=.15, anchor="c")
            labelzje.place(relx=.65, rely=.2, anchor="c")


            labelPotpis.place(relx=.8, rely=.75, anchor="c")
            labelNaziv.place(relx=.8, rely=.79, anchor="c")
            slika.place(relx=.8, rely=.88, anchor="c")

    def radioPress():
        global radioVar
            setupSerial(radioVar.get())
        mainScreen()


    setupView()
    tkArd.mainloop()
```



PARTS LIST			
ITEM	QTY	PART NUMBER	DESCRIPTION
2	2	Nema 17	STEP AP214
3	1	Baza zgloba 1	
4	1	Zglob 1	
5	1	Baza zgloba 2	
6	1	Osovina 1	
7	1	Manji zupčanik zgloba 2	
8	1	Veliki zupčanik zgloba 2	
9	1	Baza zgloba 3	
10	1	Osovina 2	
11	1	Manji zupčanik zgloba 3	
12	1	Veliki zupčanik zgloba 3	
13	1	NEMA 17 5:1	STEP AP203
14	1	Hvataljka	
16	5	DIN 625 SKF - SKF 61803	Single row ball bearings SKF
17	1	NEMA 11 5:1	
18	1	Prvi dio baze zgloba 4 i 5	
19	1	Drugidio baze zgloba 4 i 5	
20	1	Zglob 4 i 5	
22	1	NEMA 11	
23	2	Distantni_prsten 2	
25	2	Distantni_prsten 1	
26	1	Manji zupčanik zgloba 1	
27	1	Veliki zupčanik zgloba 1	
28	1	Kućište NEMA 17 5:1	
29	1	Kućište NEMA 11 5:1	
31	8	Limit switch	
36	4	DIN 6912 - M4 x 12	Cylinder Head Cap Screw
38	2	DIN 6912 - M5 x 35	Cylinder Head Cap Screw
39	8	DIN 6912 - M5 x 16	Cylinder Head Cap Screw
40	18	DIN 912 - M3 x 8	Cylinder Head Cap Screw
41	6	DIN 912 - M6 x 20	Cylinder Head Cap Screw
42	4	DIN 912 - M4 x 12	Cylinder Head Cap Screw
43	4	DIN 912 - M5 x 16	Cylinder Head Cap Screw
44	4	DIN 912 - M3 x 10	Cylinder Head Cap Screw
45	4	DIN 912 - M2.5 x 10	Cylinder Head Cap Screw
46	12	DIN 912 - M2.5 x 8	Cylinder Head Cap Screw

		Datum	Ime i prezime	Potpis
Broj naziva - code	Projektirao		Kristijan Peter	
	Razradio		Kristijan Peter	
	Crtao		Kristijan Peter	
	Pregledao			

FSB Zagreb

ISO - TOL

Objekt broj

R.N. broj

Napomena

Kopija

Materijal

Masa

Naziv Robotska ruka ERA

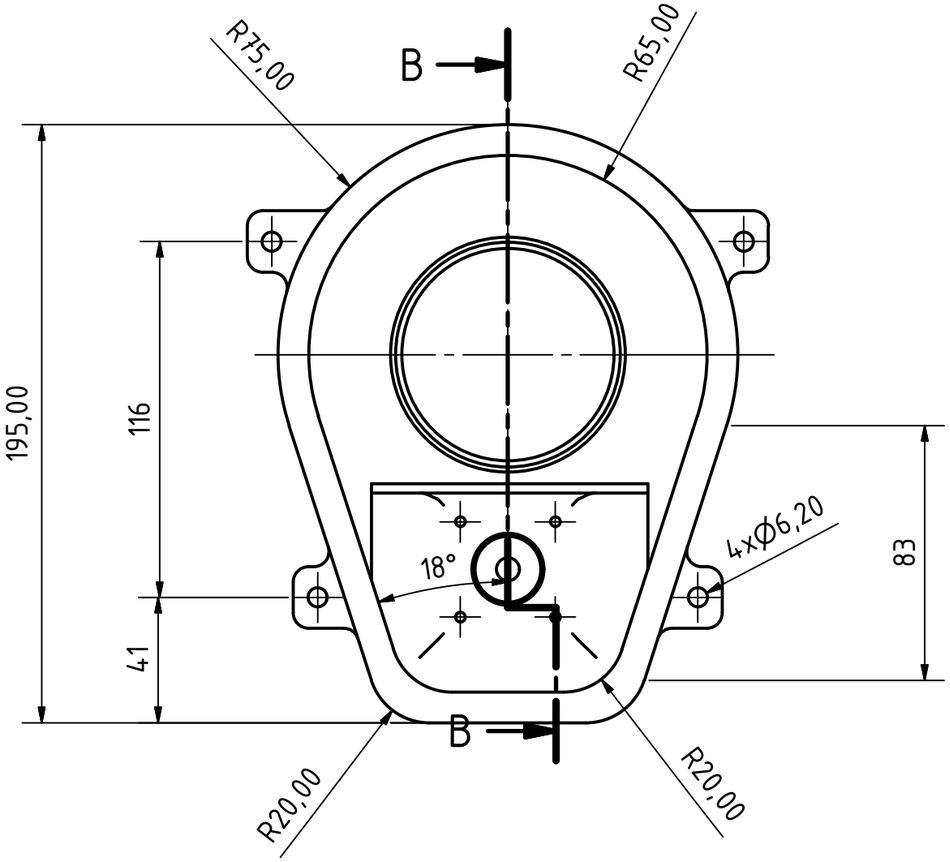
Pozicija A2

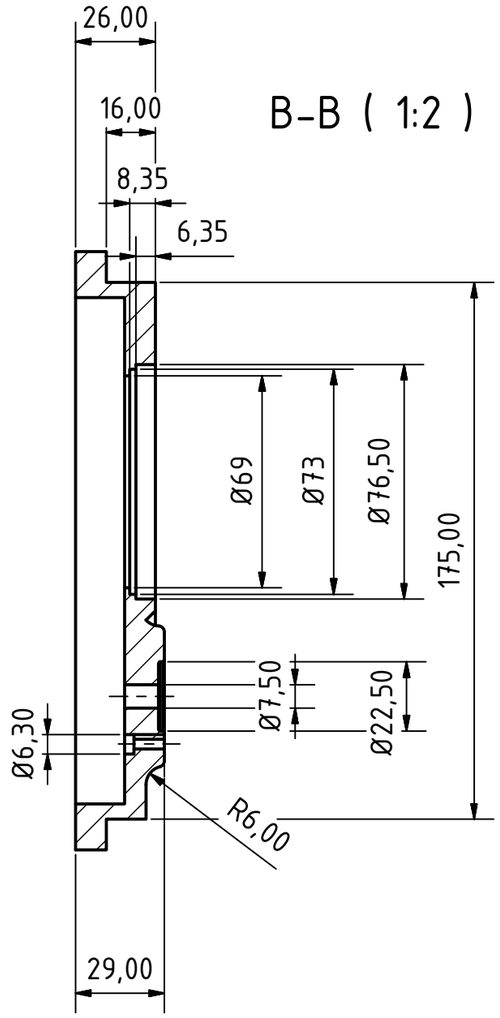
Mj. orginala 1:2

Listova

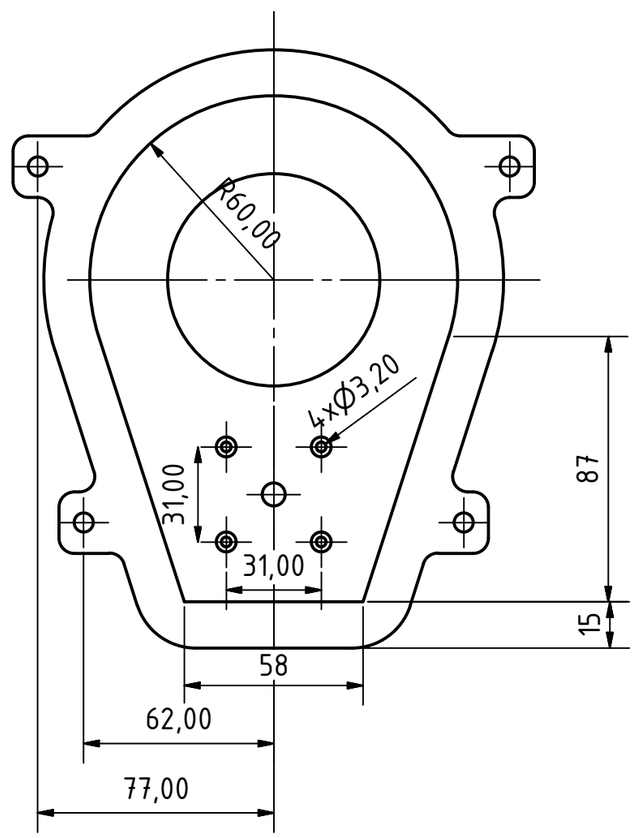
Crtež broj ERA 01-00-00

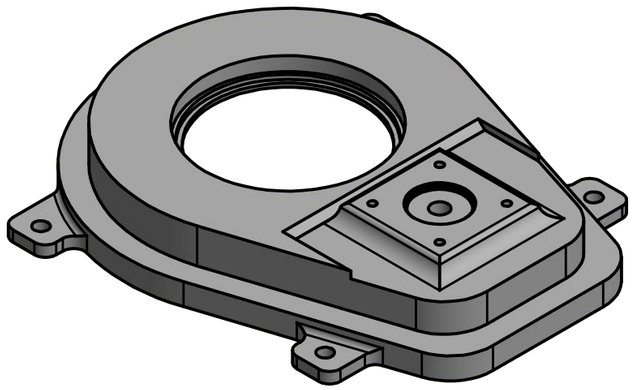
List



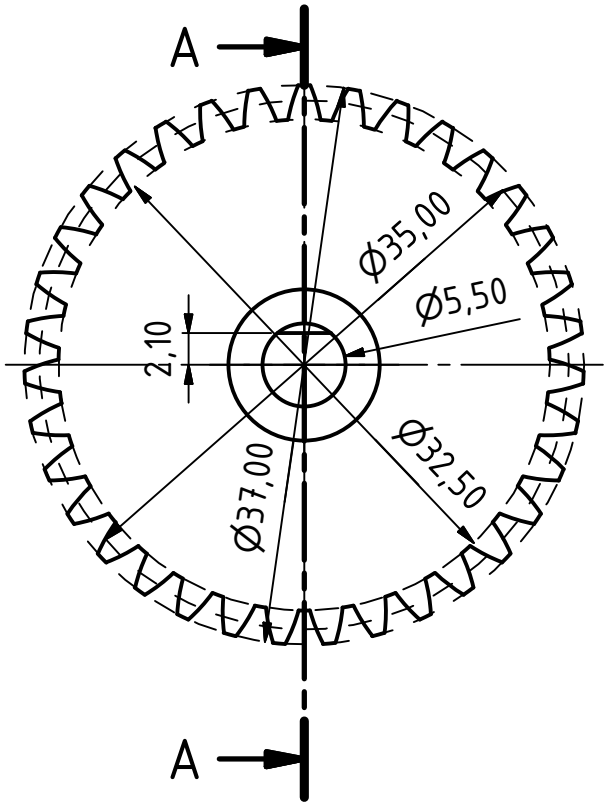


B-B (1:2)

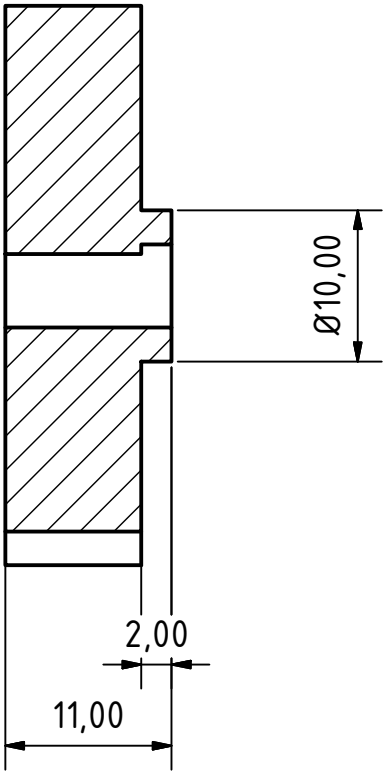




Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis
	Projektirao		Kristijan Peter	
	Razradio		Kristijan Peter	
	Crtao		Kristijan Peter	
	Pregledao			
				FSB Zagreb
ISO - TOL			Objekt broj	
			R.N. broj	
	Napomena			Kopija
	Materijal PLA	Masa		
			Pozicija	A3
	Mj. orginala 1:1	Baza Zgloba 1		Listova
		Crtež broj: ERA 01-00-01		List

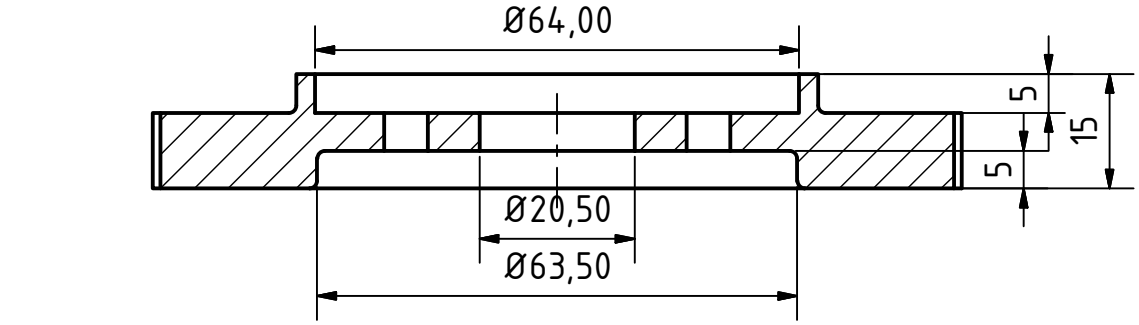


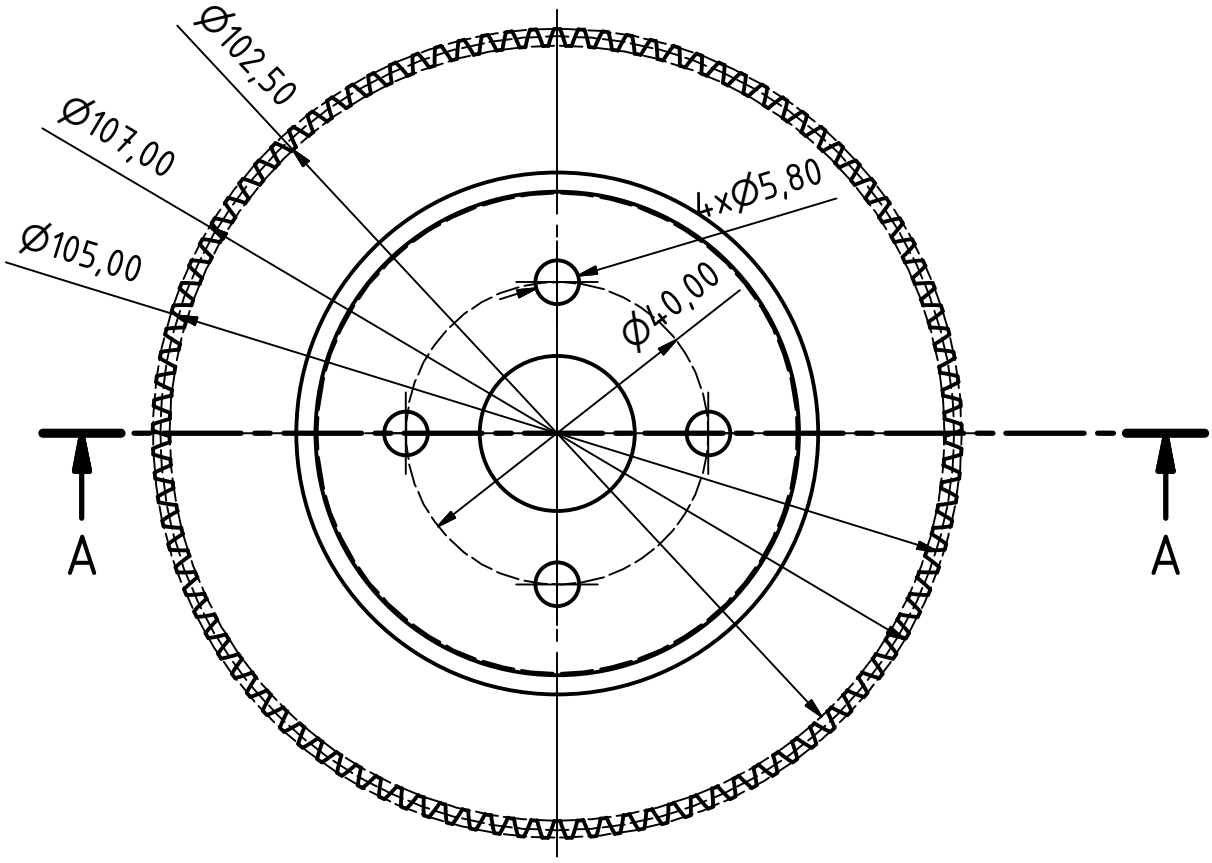
A-A (2)



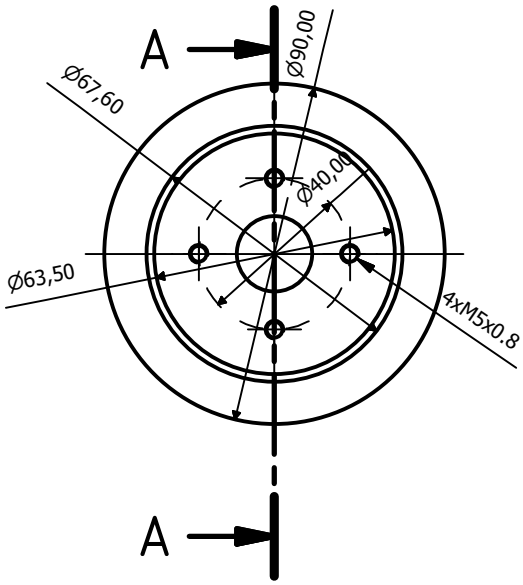
Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis	FSB Zagreb
	Projektirao		Kristijan Peter		
	Razradio		Kristijan Peter		
	Crtao		Kristijan Peter		
	Pregledao				
ISO - TOL				Objekt broj	
				R.N. broj	
	Napomena				Kopija
	Materijal PLA		Masa		
		Naziv Manji zupčanik zgloba 1		Pozicija	A4
	Mj. orginala 2:1				Listova
		Crtež broj: ERA 01-00-02			List

A-A (1)

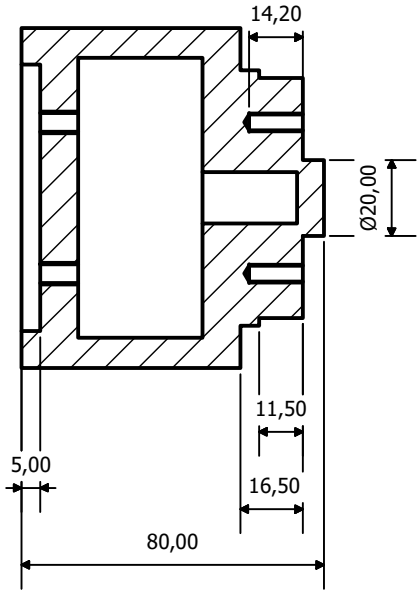


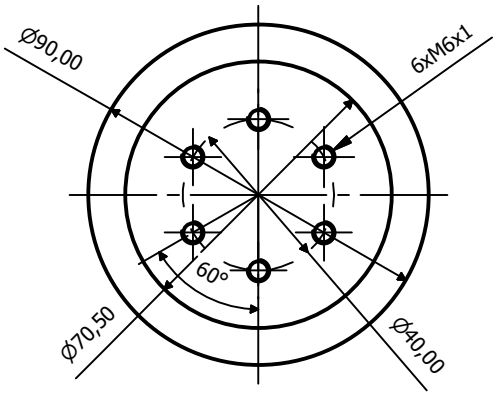


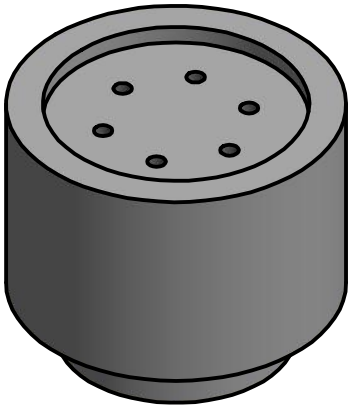
Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis	FSB Zagreb
	Projektirao		Kristijan Peter		
	Razradio		Kristijan Peter		
	Crtao		Kristijan Peter		
	Pregledao				
ISO - TOL				Objekt broj	
				R.N. broj	
	Napomena				Kopija
	Materijal PLA		Masa		
		Naziv Veliki zupčanik zgloba 1		Pozicija	A4
	Mj. orginala 1:1				Listova
		Crtež broj: ERA 01-00-03			List



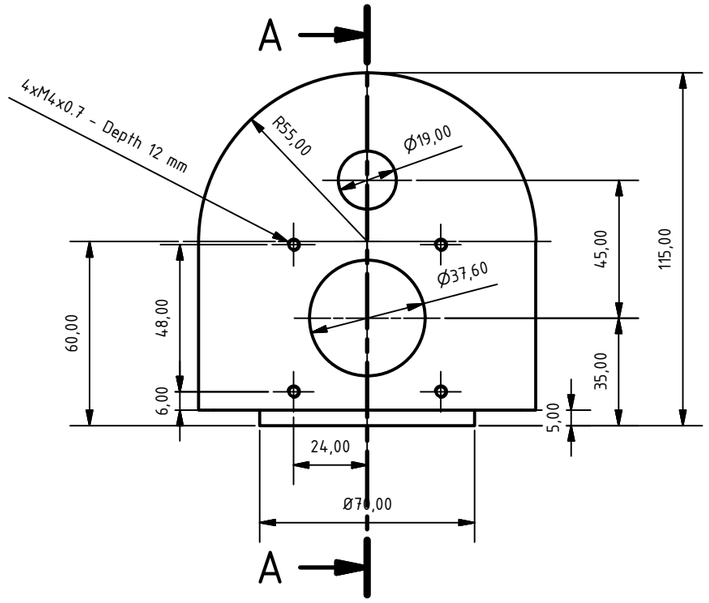
A-A (1 : 2)

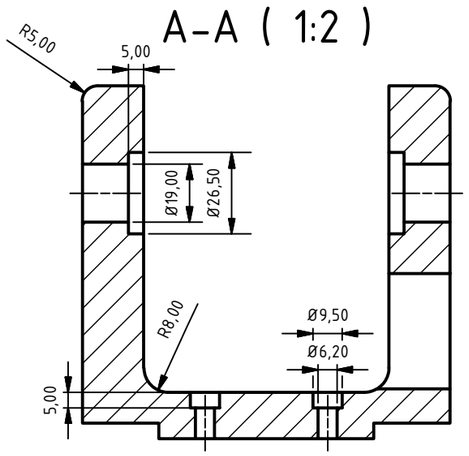


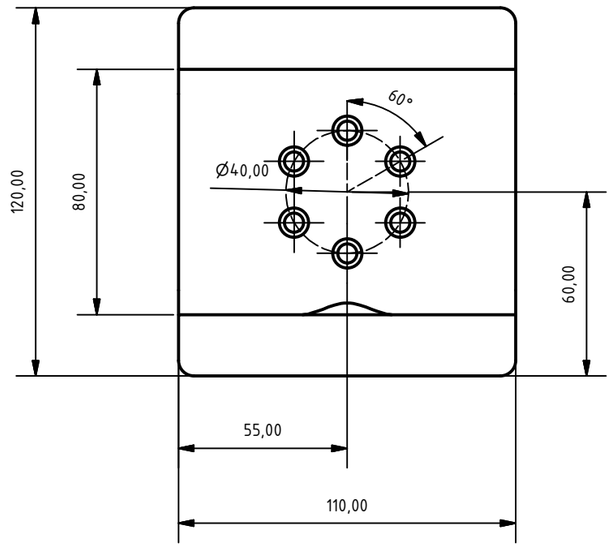




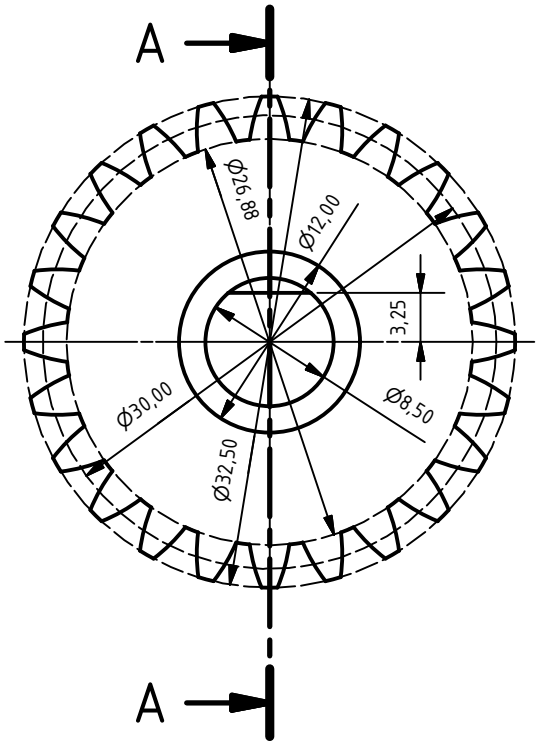
Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis	FSB Zagreb
	Projektirao		Kristijan Peter		
	Razradio		Kristijan Peter		
	Crtao		Kristijan Peter		
	Pregledao				
ISO - TOL				Objekt broj	
				R.N. broj	
	Napomena				Kopija
	Materijal PLA		Masa		
		Naziv	Zglob 1	Pozicija	A4
	Mj. orginala 1:2				Listova
		Crtež broj:	ERA 01-00-04		List

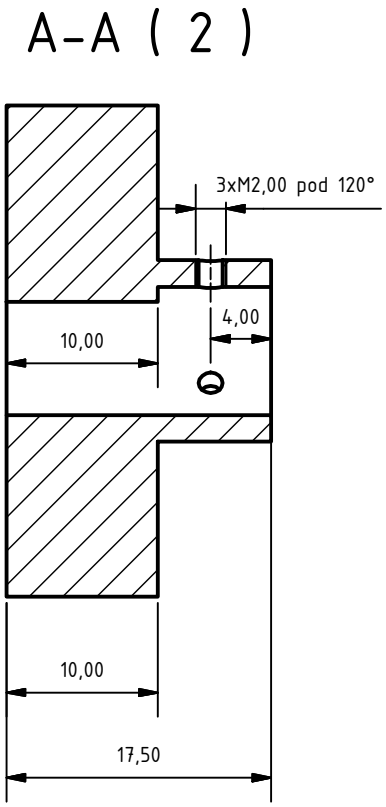




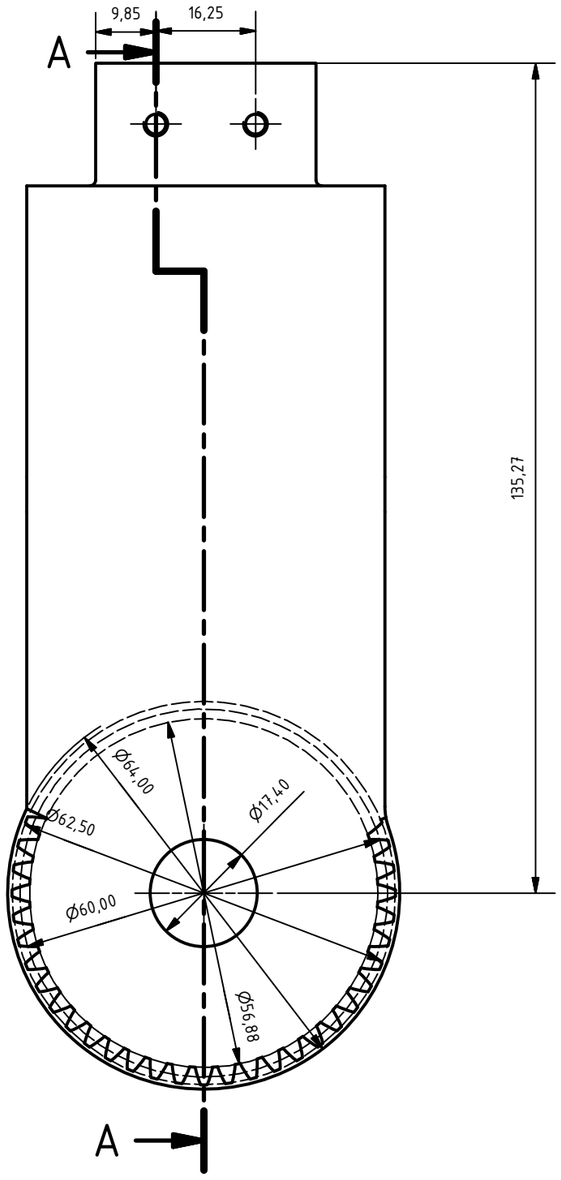


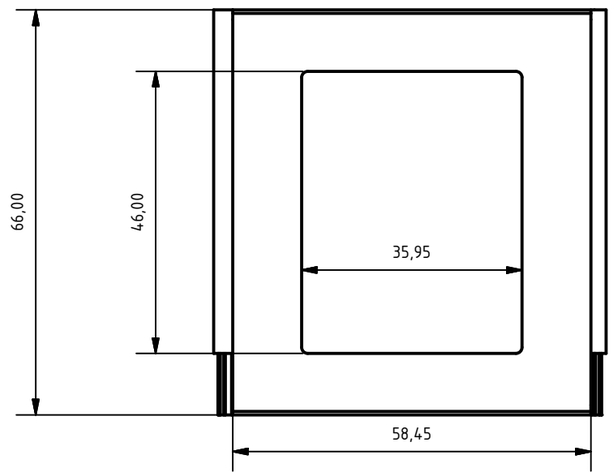
Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis	FSB Zagreb
	Projektirao		Kristijan Peter		
	Razradio		Kristijan Peter		
	Crtao		Kristijan Peter		
	Pregledao				
ISO - TOL				Objekt broj	
				R.N. broj	
	Napomena				Kopija
	Materijal PLA		Masa		
		Baza zgloba 2		Pozicija	A3
	Mj. orginala 1:1				Listova
		Crtež broj ERA 01-00-05			List

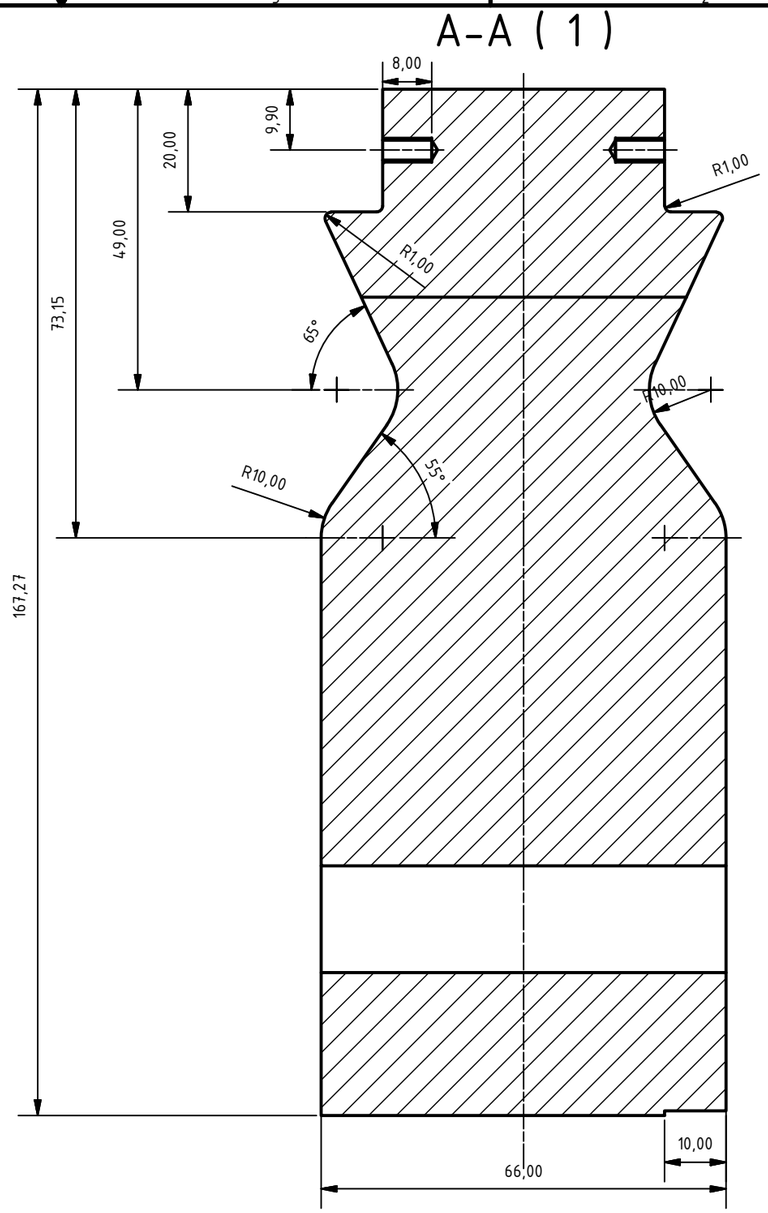




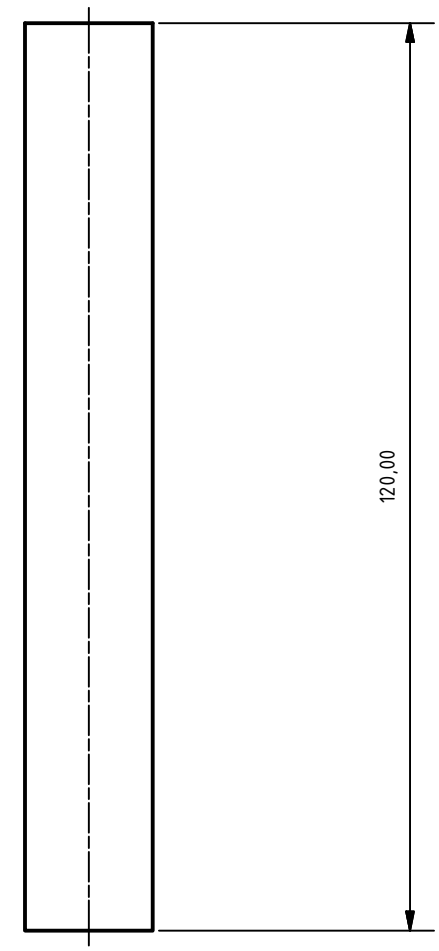
Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis	FSB Zagreb
	Projektirao		Kristijan Peter		
	Razradio		Kristijan Peter		
	Crtao		Kristijan Peter		
	Pregledao				
ISO - TOL				Objekt broj	
				R.N. broj	
	Napomena				Kopija
	Materijal PLA		Masa		
		Naziv Manji zubčanik zgloba 2		Pozicija	A4
	Mj. orginala 2:1				Listova
		Crtež broj: ERA 01-00-09			List

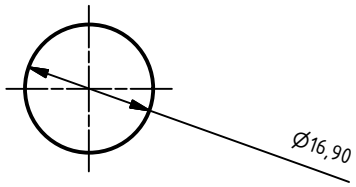






Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis	FSB Zagreb
	Projektirao		Kristijan Peter		
	Razradio		Kristijan Peter		
	Crtao		Kristijan Peter		
	Pregledao				
ISO - TOL				Objekt broj	
				R.N. broj	
	Napomena				Kopija
	Materijal PLA	Masa			
	Mj. orginala 1:1	Veliki zupčanik zgloba 2		Pozicija	A3
		Crtež broj ERA 01-00-10			Listova
					List

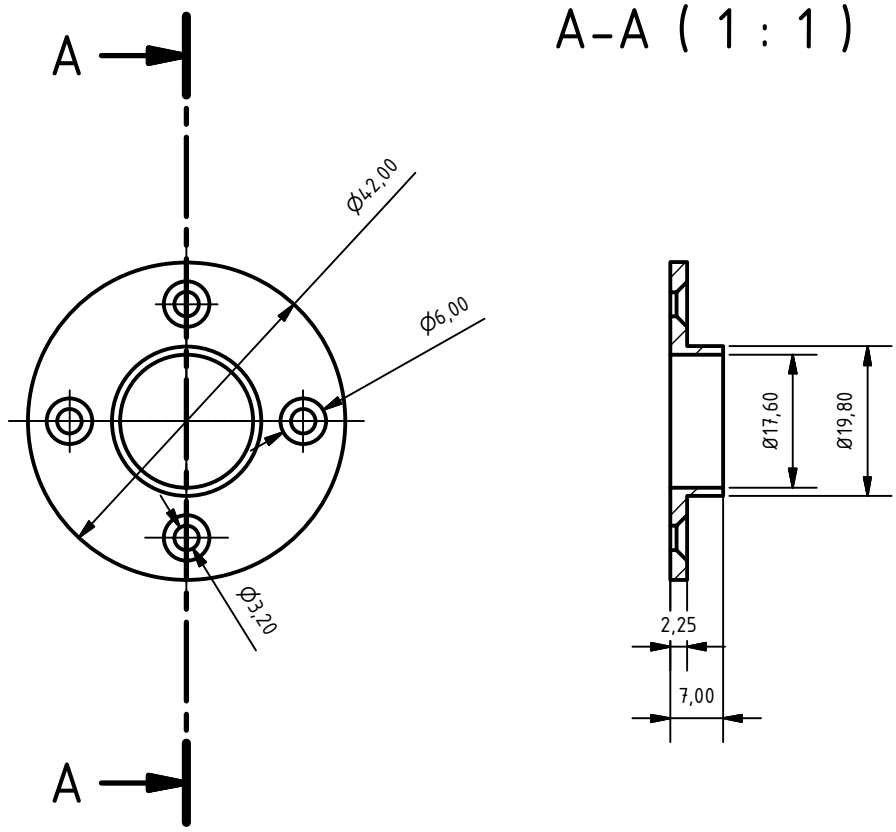




Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis
	Projektirao		Kristijan Peter	
	Razradio		Kristijan Peter	
	Crtao		Kristijan Peter	
	Pregledao			

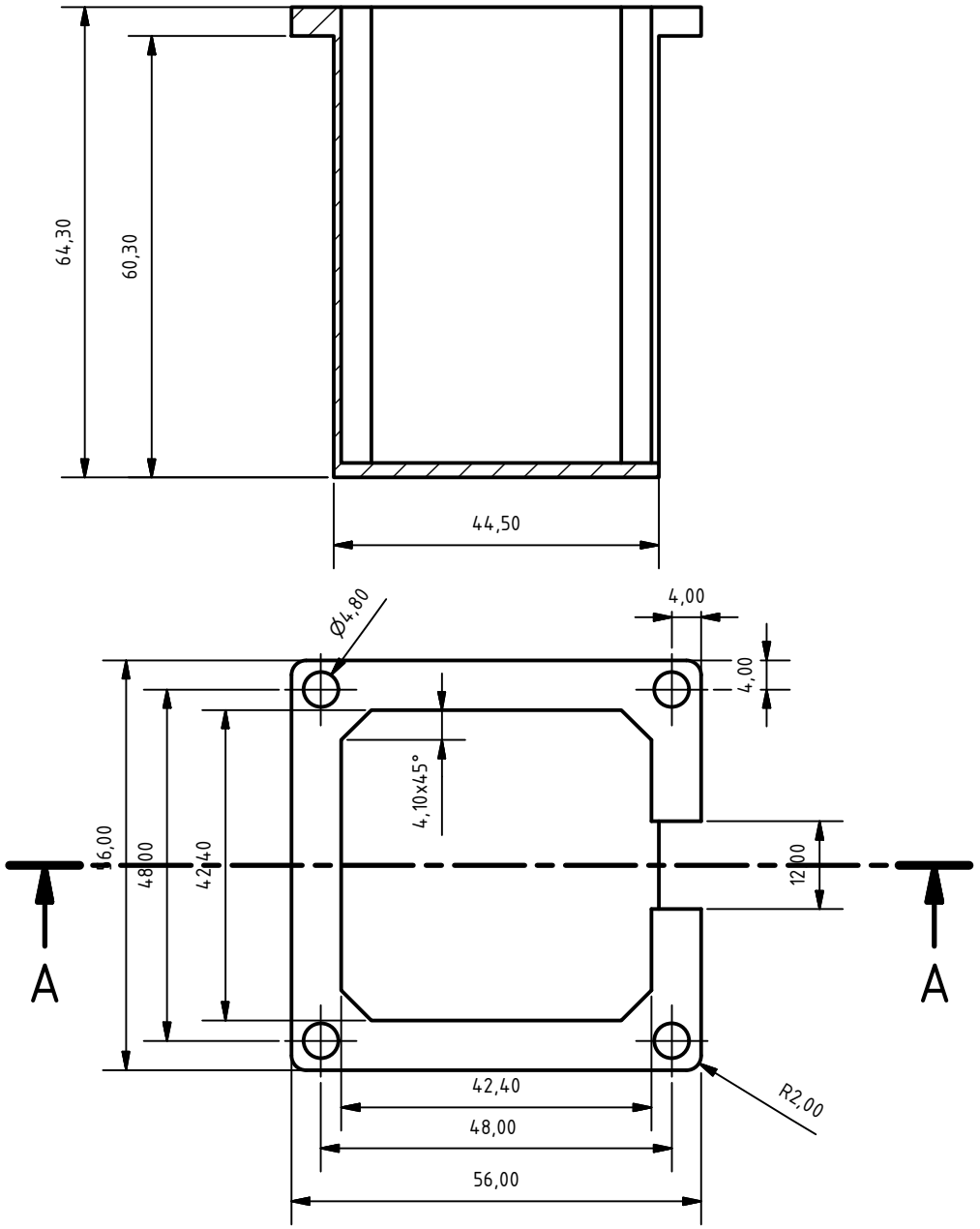
FSB Zagreb

ISO - TOL		Objekt broj	
		R.N. broj	
	Napomena		Kopija
	Materijal PLA	Masa	
	Naziv Osovina 1	Pozicija	A4
	Mj. orginala 1:1		Listova
	Crtež broj: ERA 01-00-07		List



Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis	FSB Zagreb
	Projektirao		Kristijan Peter		
	Razradio		Kristijan Peter		
	Crtao		Kristijan Peter		
	Pregledao				
ISO - TOL				Objekt broj	
				R.N. broj	
	Napomena				Kopija
	Materijal PLA		Masa		
		Naziv Distantni prsten 1		Pozicija	A4
	Mj. orginala 1:1				Listova
		Crtež broj: ERA 01-00-08			List

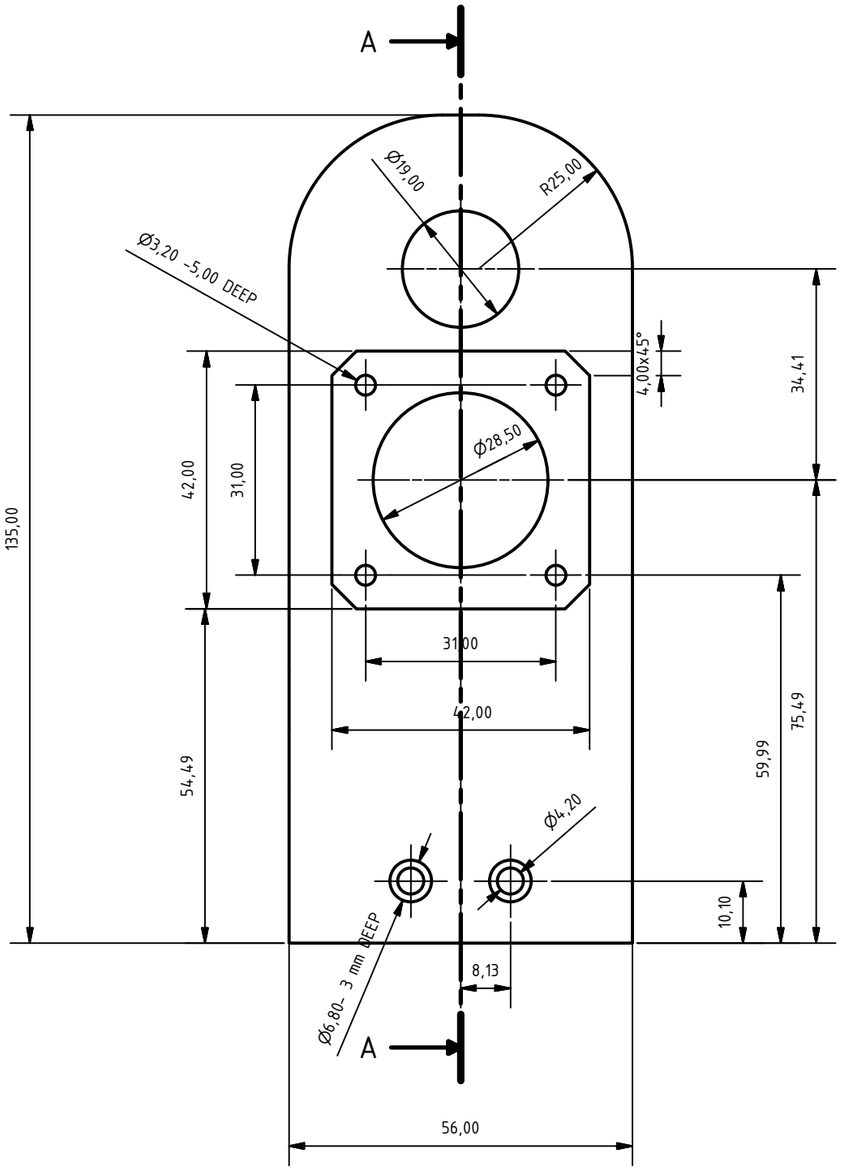
A-A (1 : 1)

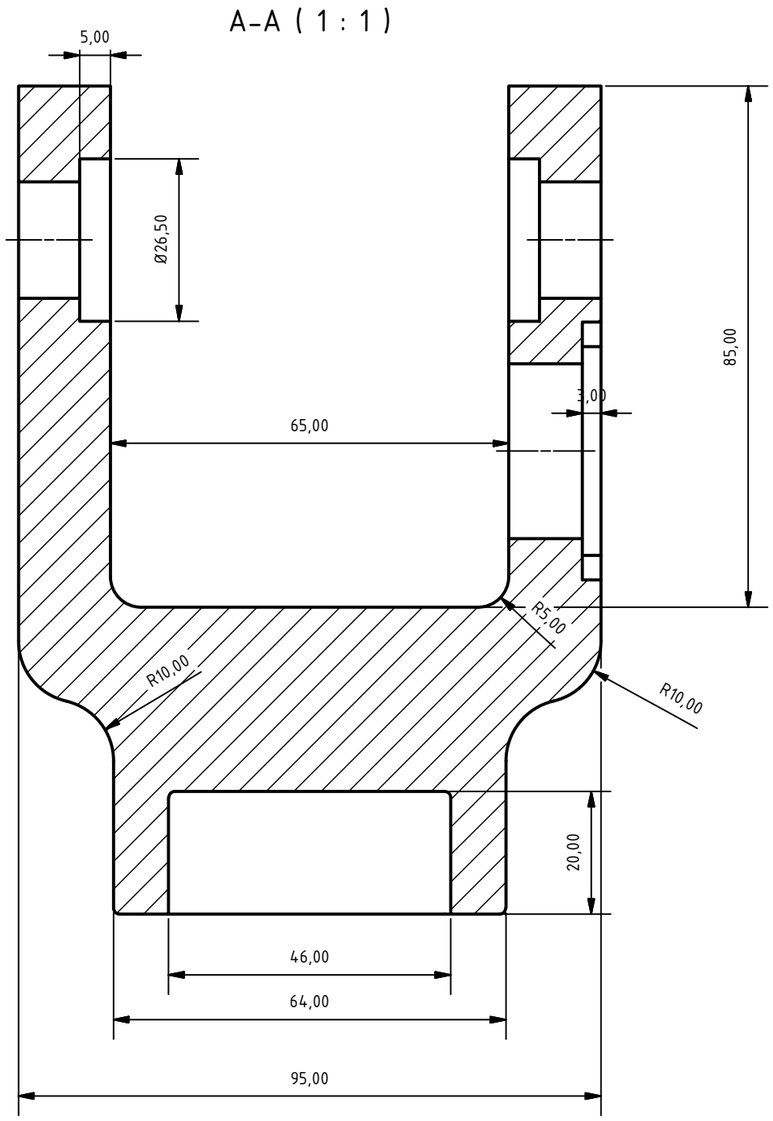


Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis
	Projektirao		Kristijan Peter	
	Razradio		Kristijan Peter	
	Crtao		Kristijan Peter	
	Pregledao			

FSB Zagreb

ISO - TOL		Objekt broj	
		R.N. broj	
	Napomena		Kopija
	Materijal PLA	Masa	
	Naziv: Kučište motora Nema 17 5:1	Pozicija	A4
	Mj. orginala 1:1		Listova
	Crtež broj: ERA 01-00-06		List



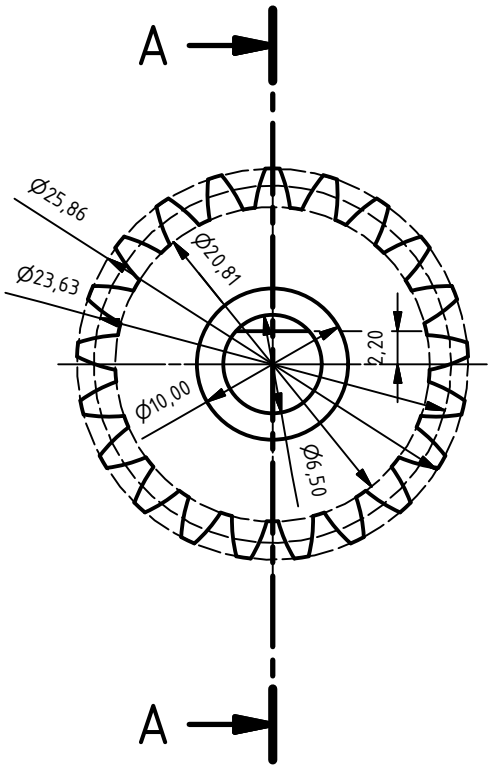


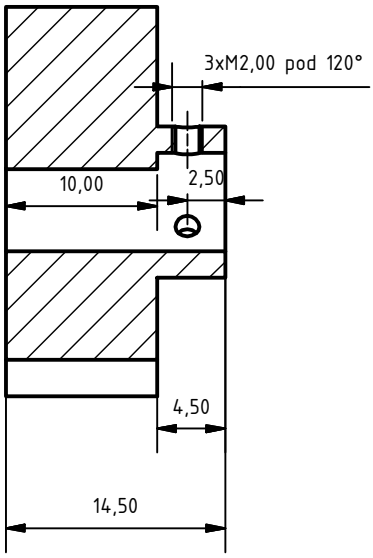
Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis
	Projektirao		Kristijan Peter	
	Razradio		Kristijan Peter	
	Crtao		Kristijan Peter	
	Pregledao			

FSB Zagreb

ISO - TOL		Objekt broj	
		R.N. broj	
	Napomena		Kopija
	Materijal PLA	Masa	
	Baza zgloba 3	Pozicija	A3
	Mj. orginala 1:1		Listova
	Crtež broj ERA 01-00-11		List

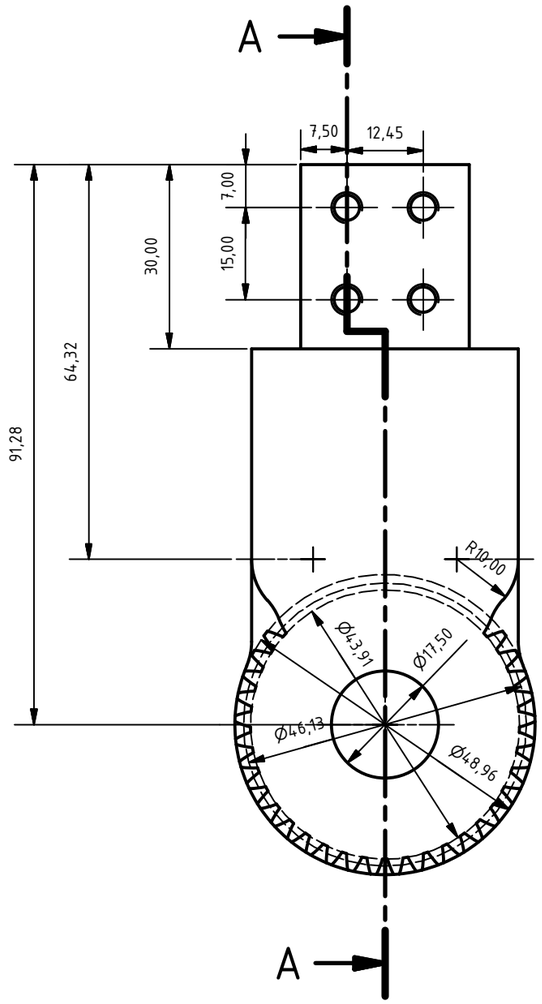
A–A (2:1)

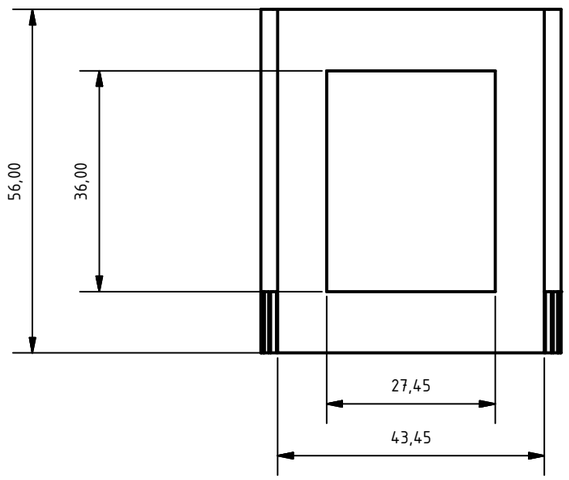




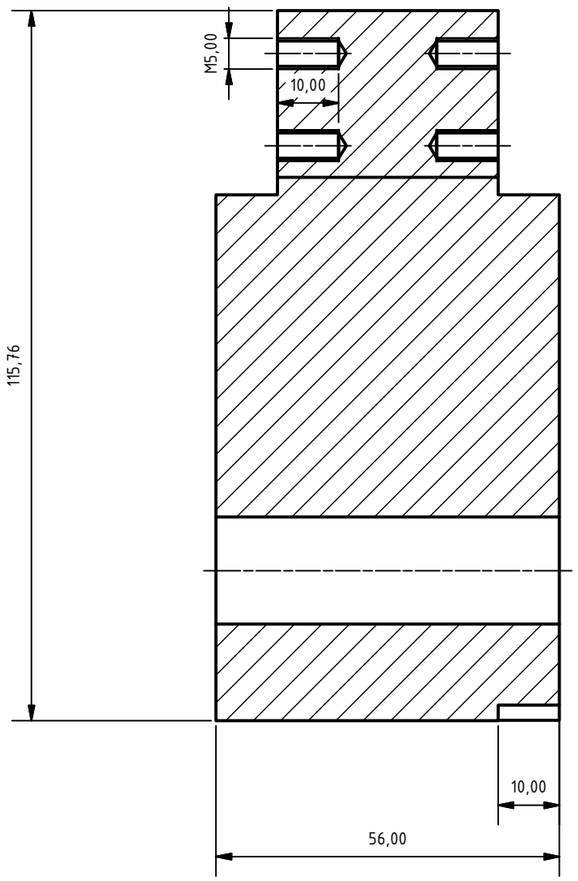
Broj naziva – code 0035198029		Datum	Ime i prezime	Potpis
	Projektirao		Kristijan Peter	
	Razradio		Kristijan Peter	
	Crtao		Kristijan Peter	
	Pregledao			
ISO – TOL			Objekt broj	
			R.N. broj	
	Napomena			Kopija
	Materijal PLA	Masa		
	Naziv: Manji zupčanik zgloba 3		Pozicija	A4
	Mj. orginala 2:1			Listova
	Crtež broj: ERA 01–00–14			List

FSB Zagreb

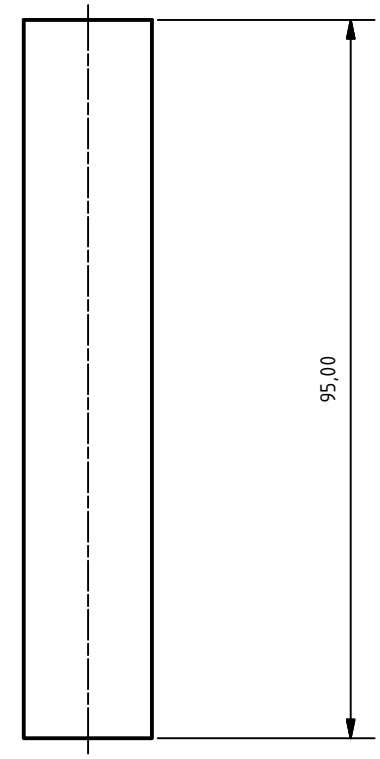


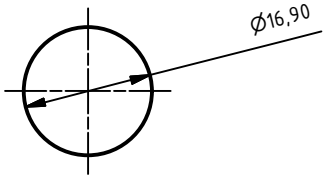


A-A (1 : 1)



Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis
	Projektirao		Kristijan Peter	
	Razradio		Kristijan Peter	
	Crtao		Kristijan Peter	
	Pregledao			
FSB Zagreb				
ISO - TOL			Objekt broj	
			R.N. broj	
	Napomena			Kopija
	Materijal PLA	Masa		
	Veliki zupčanik zgloba 3		Pozicija	A3
	Mj. orginala 1:1			Listova
	Crtež broj ERA 01-00-15			List



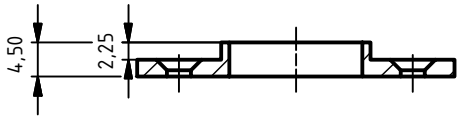


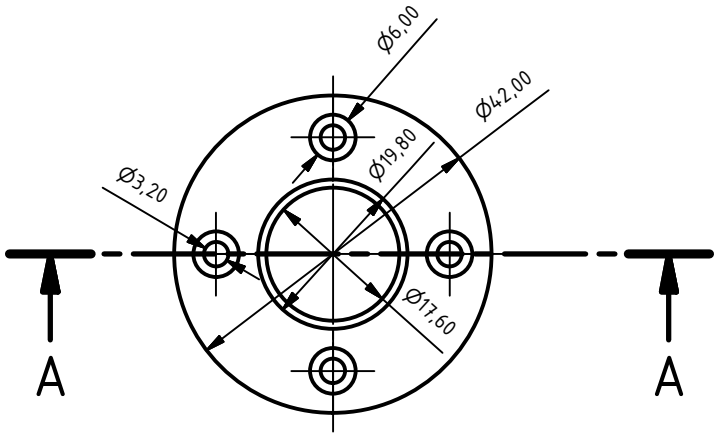
Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis
	Projektirao		Kristijan Peter	
	Razradio		Kristijan Peter	
	Crtao		Kristijan Peter	
	Pregledao			

FSB Zagreb

ISO - TOL		Objekt broj	
		R.N. broj	
	Napomena		Kopija
	Materijal PLA	Masa	
	Naziv Osovina 2	Pozicija	A4
	Mj. orginala 1:1		Listova
	Crtež broj: ERA 01-00-13		List

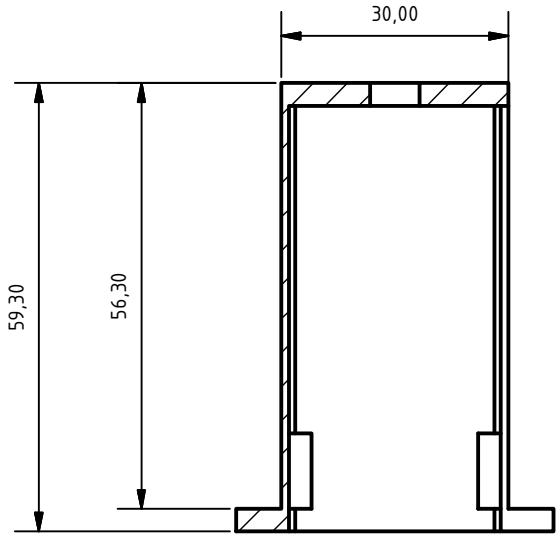
A-A (1 : 1)

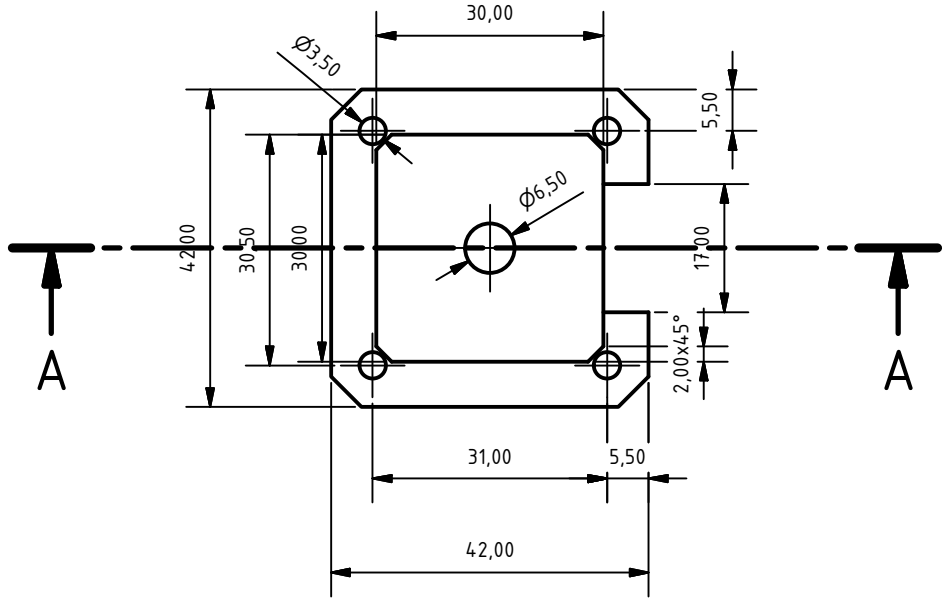




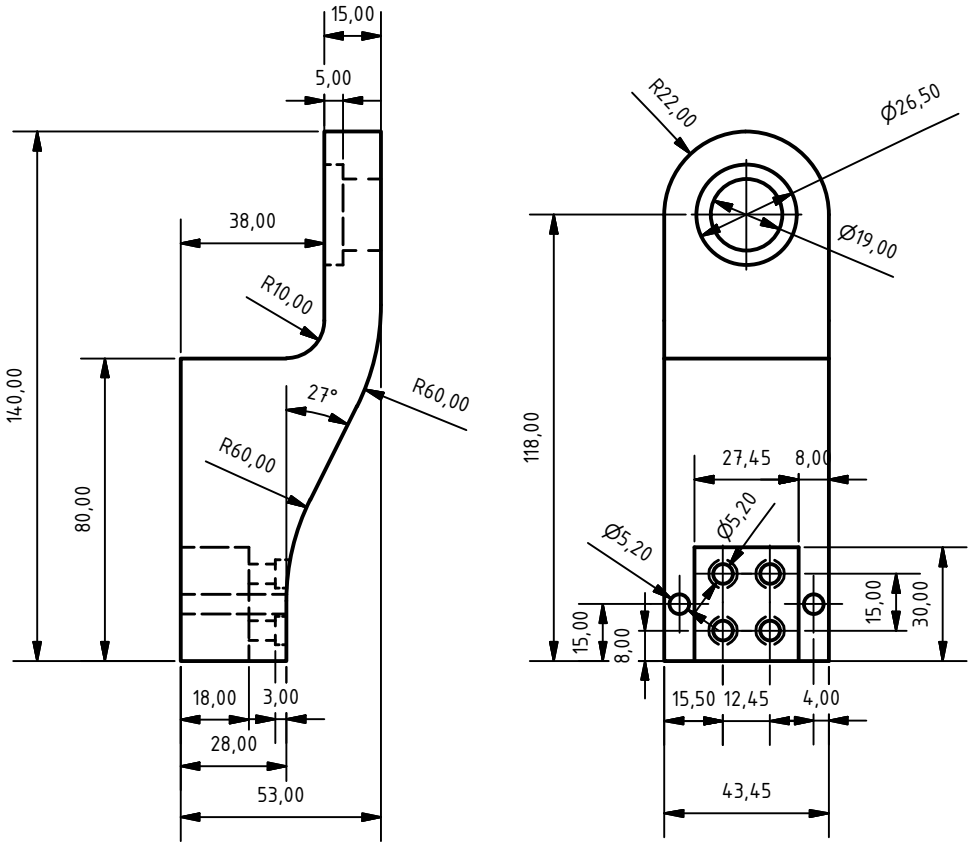
Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis	FSB Zagreb
	Projektirao		Kristijan Peter		
	Razradio		Kristijan Peter		
	Crtao		Kristijan Peter		
	Pregledao				
ISO - TOL				Objekt broj	
				R.N. broj	
	Napomena				Kopija
	Materijal PLA		Masa		
		Naziv Distantni prsten 2		Pozicija	A4
	Mj. orginala 1:1				Listova
		Crtež broj: ERA 01-00-16			List

A-A (1 : 1)

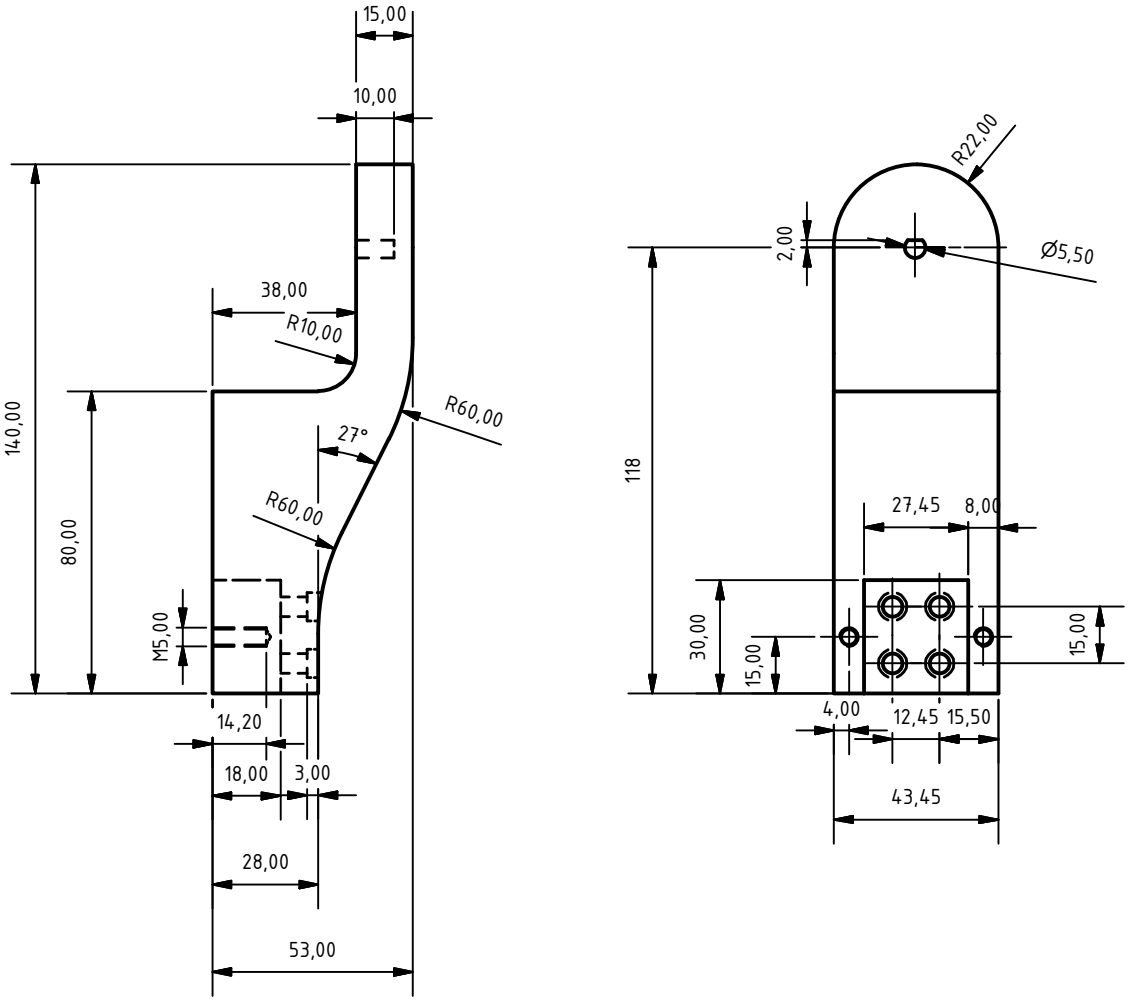




Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis	FSB Zagreb
	Projektirao		Kristijan Peter		
	Razradio		Kristijan Peter		
	Crtao		Kristijan Peter		
	Pregledao				
ISO - TOL				Objekt broj	
				R.N. broj	
	Napomena				Kopija
	Materijal PLA		Masa		
		Naziv Kučište motora Nema 11		Pozicija	A4
	Mj. orginala 1:1	5:1			Listova
		Crtež broj: ERA 01-00-12			List

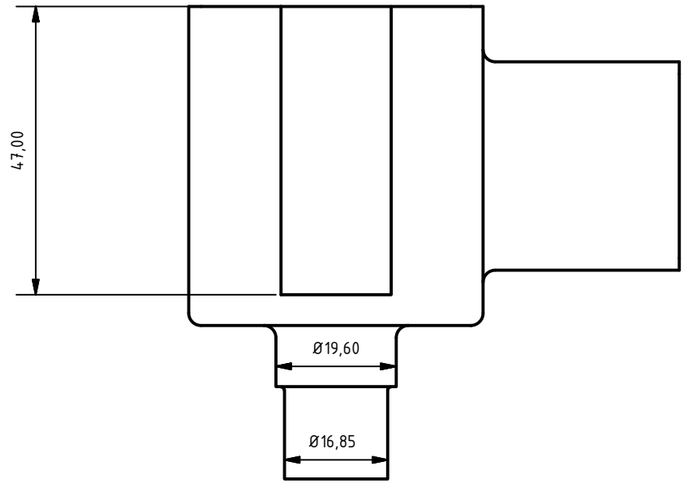


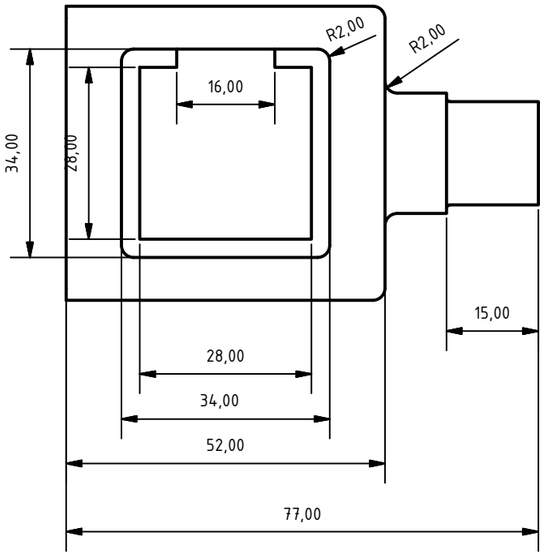
Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis	FSB Zagreb
	Projektirao		Kristijan Peter		
	Razradio		Kristijan Peter		
	Crtao		Kristijan Peter		
	Pregledao				
ISO - TOL				Objekt broj	
				R.N. broj	
	Napomena				Kopija
	Materijal PLA		Masa		
		Naziv Prvi dio baze zgloba 4 i 5		Pozicija	A4
	Mj. orginala 1:2				Listova
		Crtež broj: ERA 01-00-17			List

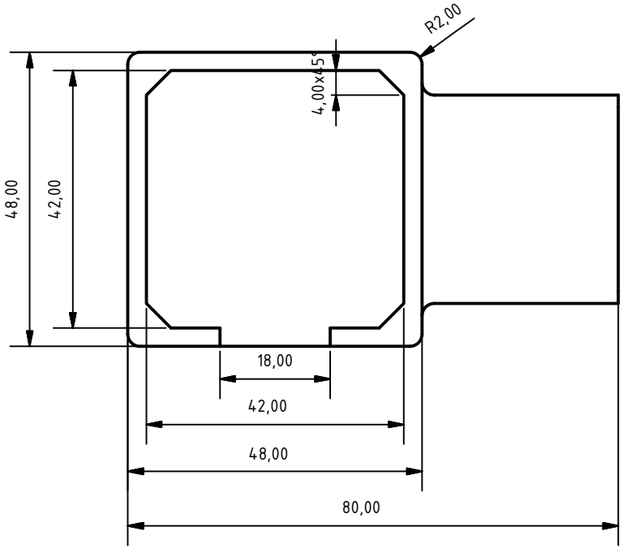


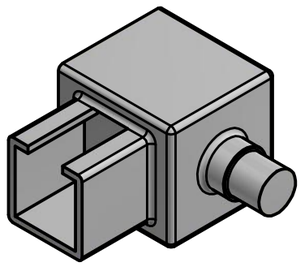
Broj naziva – code 0035198029		Datum	Ime i prezime	Potpis
	Projektirao		Kristijan Peter	
	Razradio		Kristijan Peter	
	Crtao		Kristijan Peter	
	Pregledao			
ISO – TOL			Objekt broj	
			R.N. broj	
	Napomena			Kopija
	Materijal PLA	Masa		
	Mj. orginala 1:2	Naziv: Drugi dio baze zgloba 4 i 5	Pozicija	A4
		Crtež broj: ERA 01-00-18		Listova / List

FSB Zagreb

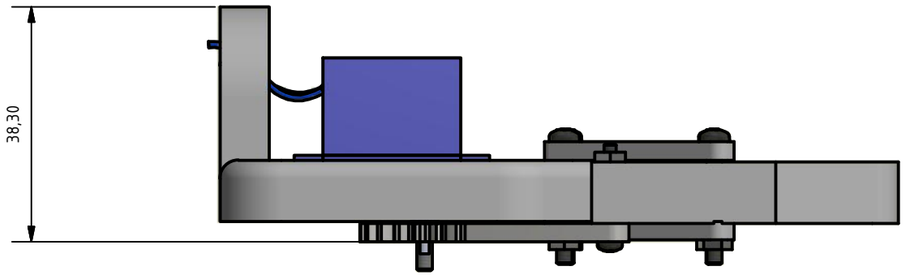


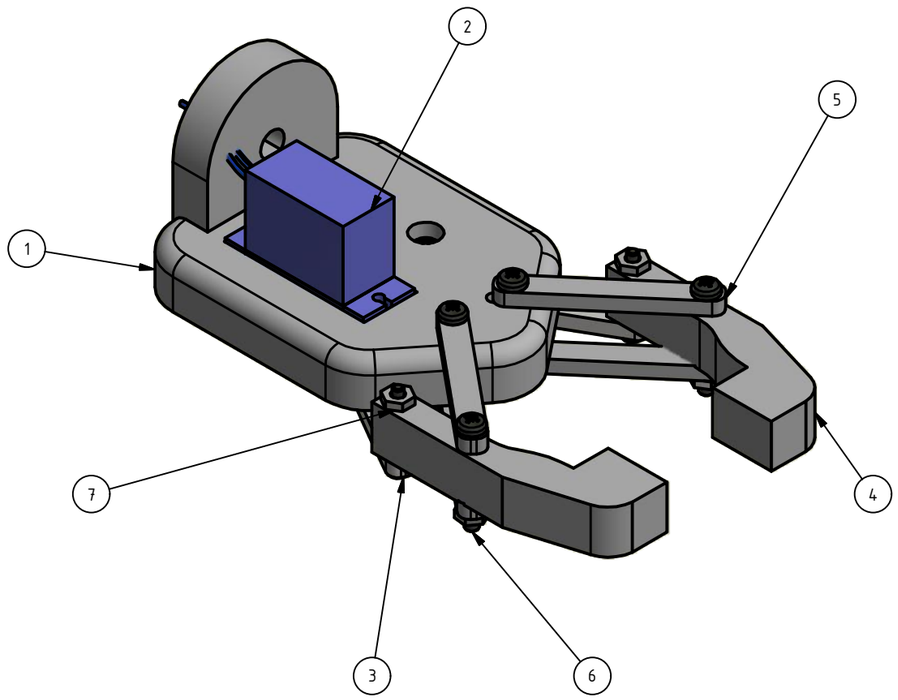


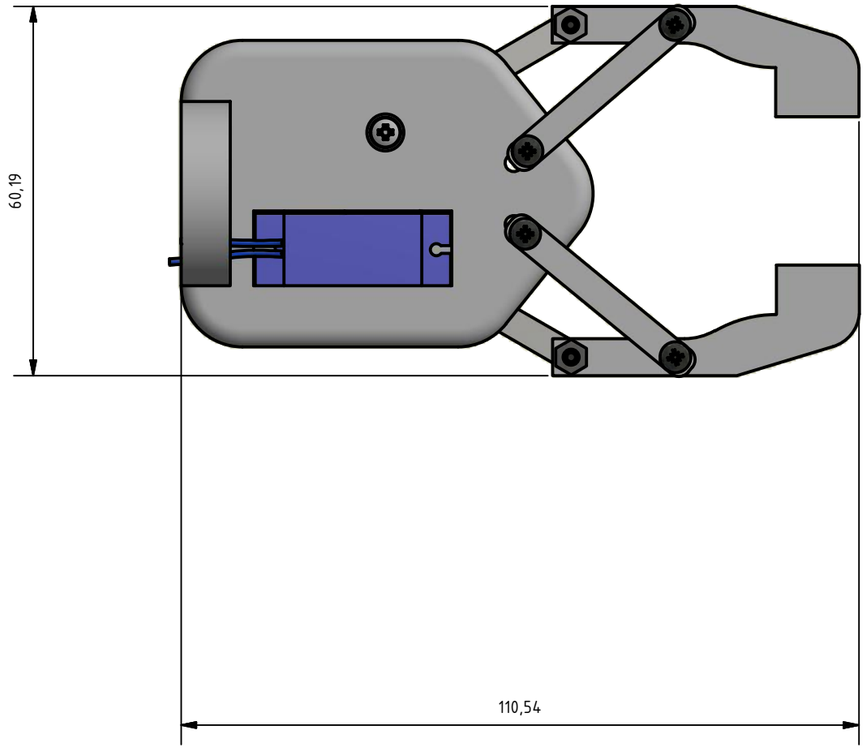




Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis
	Projektirao		Kristijan Peter	
	Razradio		Kristijan Peter	
	Crtao		Kristijan Peter	
	Pregledao			
FSB Zagreb				
ISO - TOL			Objekt broj	
			R.N. broj	
	Napomena			Kopija
	Materijal PLA	Masa		
	Zglob 4 i 5		Pozicija	A3
	Mj. orginala 1:1			Listova
	Crtež broj ERA 01-00-19			List







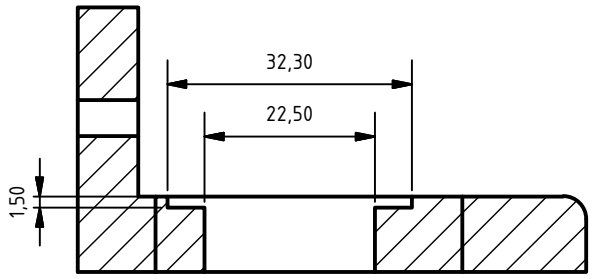
PARTS LIST			
ITEM	QTY	PART NUMBER	DESCRIPTION
1	1	Baza hvataljke	
2	1	Servo motor	
3	2	Zupčanik hvataljke	
4	2	Prst hvataljke	
5	4	Članak hvataljke	
6	7	ISO 7045 - M2.5 x 20 - 4.8 - Z	
7	6	DIN EN 24 036 - M2.5	

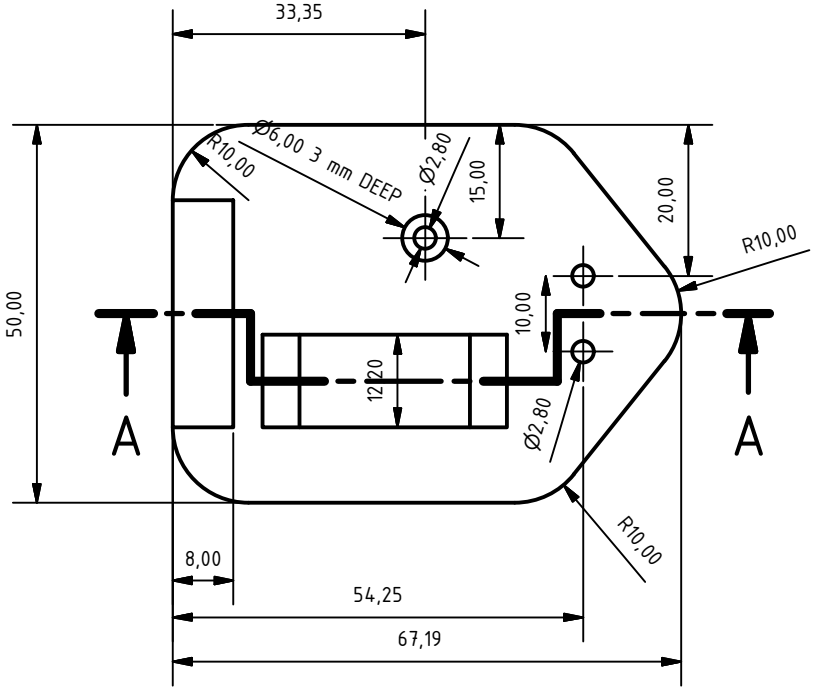
		Datum	Ime i prezime	Potpis
Broj naziva - code 0035198029	Projektirao		Kristijan Peter	
	Razradio		Kristijan Peter	
	Crtao		Kristijan Peter	
	Pregledao			

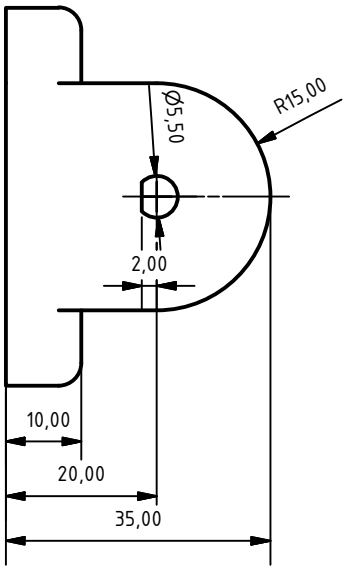
FSB Zagreb

ISO - TOL		Objekt broj	
		R.N. broj	
	Napomena		Kopija
	Materijal	Masa	
	Hvataljka	Pozicija	A3
	Mj. orginala 1:1		Listova
	Crtež broj ERA 02-00-00		List

A-A (1 : 1)



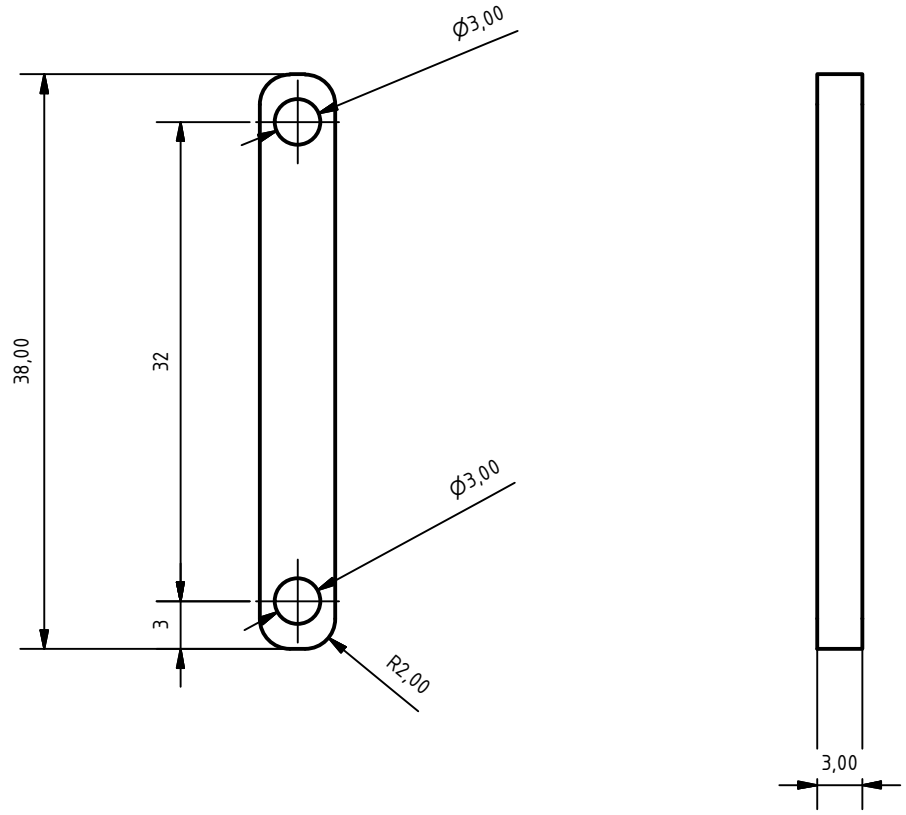




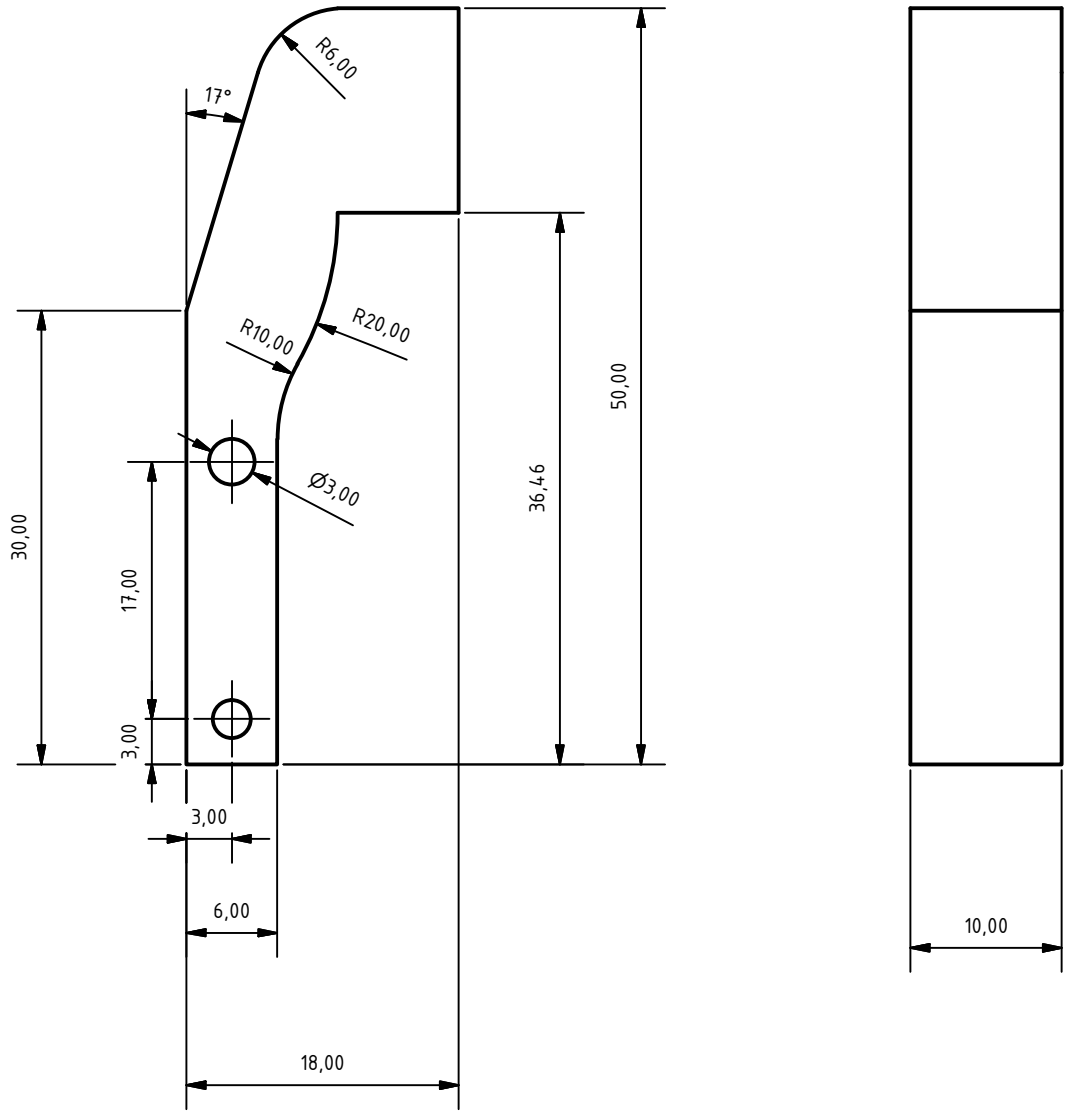
Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis
	Projektirao		Kristijan Peter	
	Razradio		Kristijan Peter	
	Crtao		Kristijan Peter	
	Pregledao			

FSB Zagreb

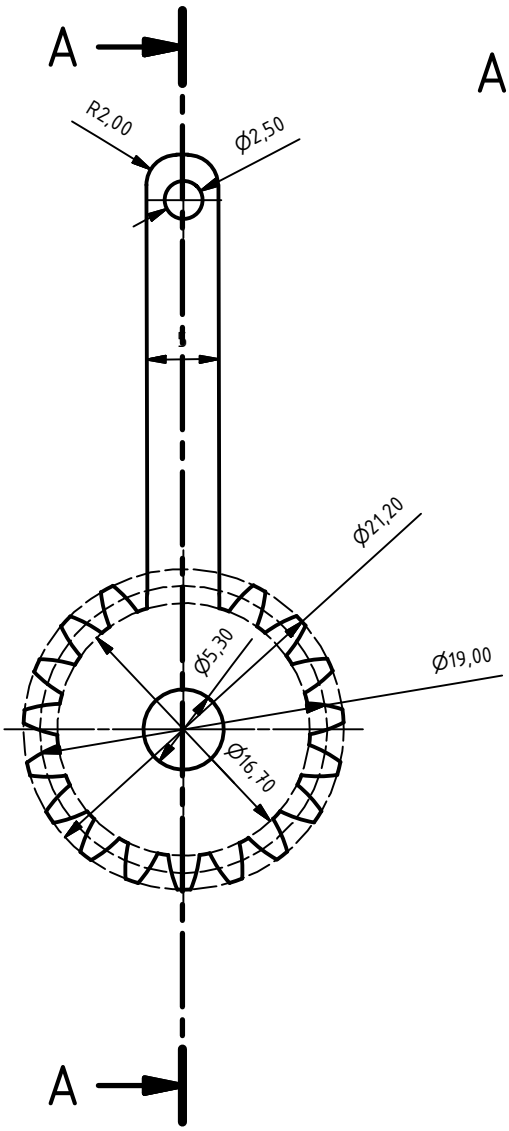
ISO - TOL		Objekt broj	
		R.N. broj	
	Napomena		Kopija
	Materijal PLA	Masa	
	Naziv: Baza hvataljke	Pozicija	A4
	Mj. orginala 1:1		Listova
	Crtež broj: ERA 02-00-01		List

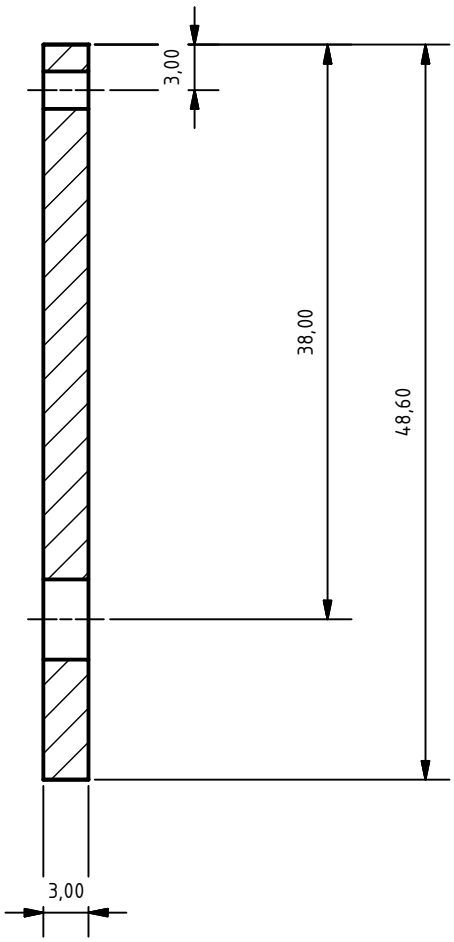


Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis	FSB Zagreb
	Projektirao		Kristijan Peter		
	Razradio		Kristijan Peter		
	Crtao		Kristijan Peter		
	Pregledao				
ISO - TOL				Objekt broj	
				R.N. broj	
	Napomena				Kopija
	Materijal PLA		Masa		
		Naziv Članak hvataljke		Pozicija	A4
	Mj. orginala 2:1				Listova
		Crtež broj: ERA 02-00-03			List

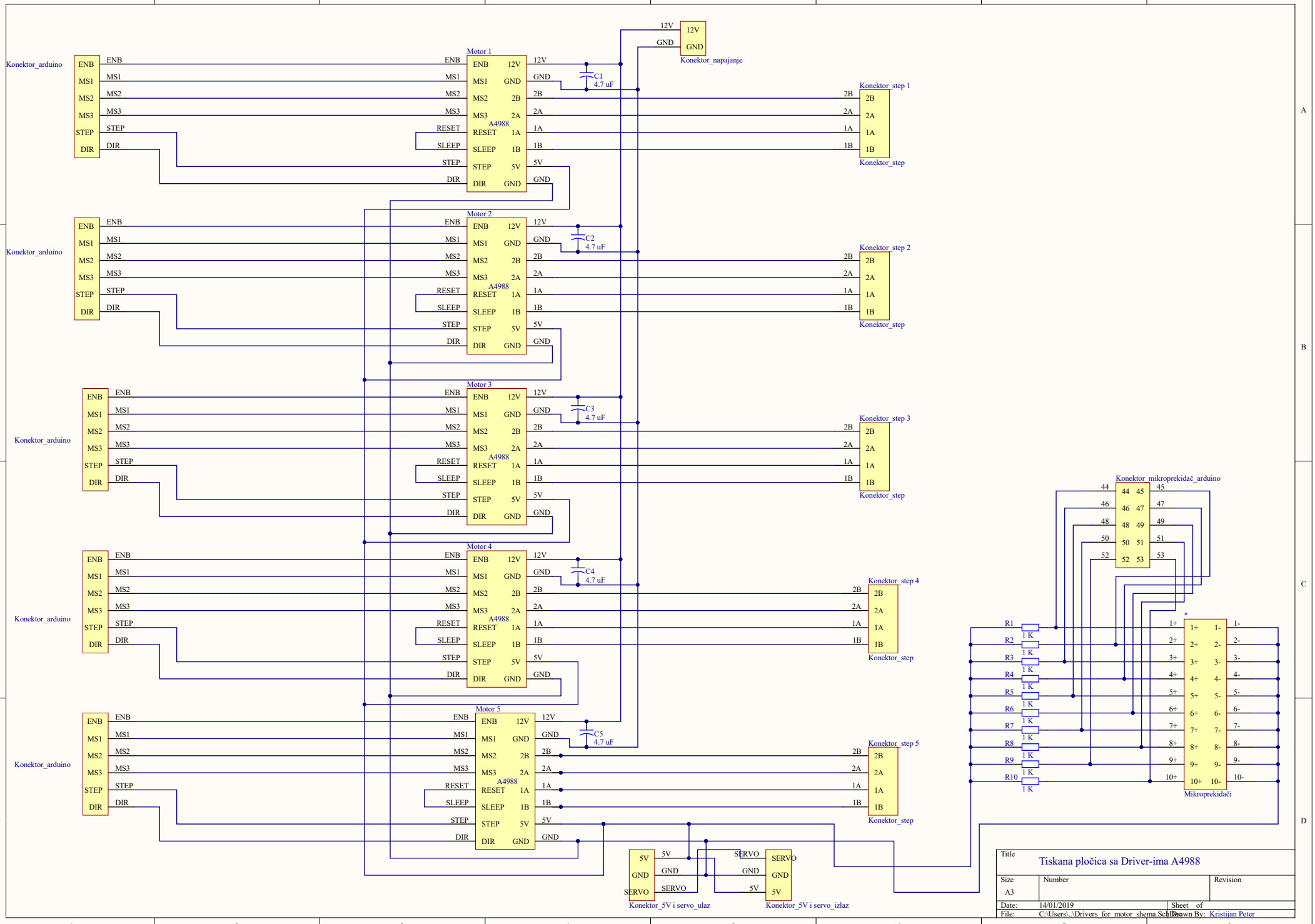


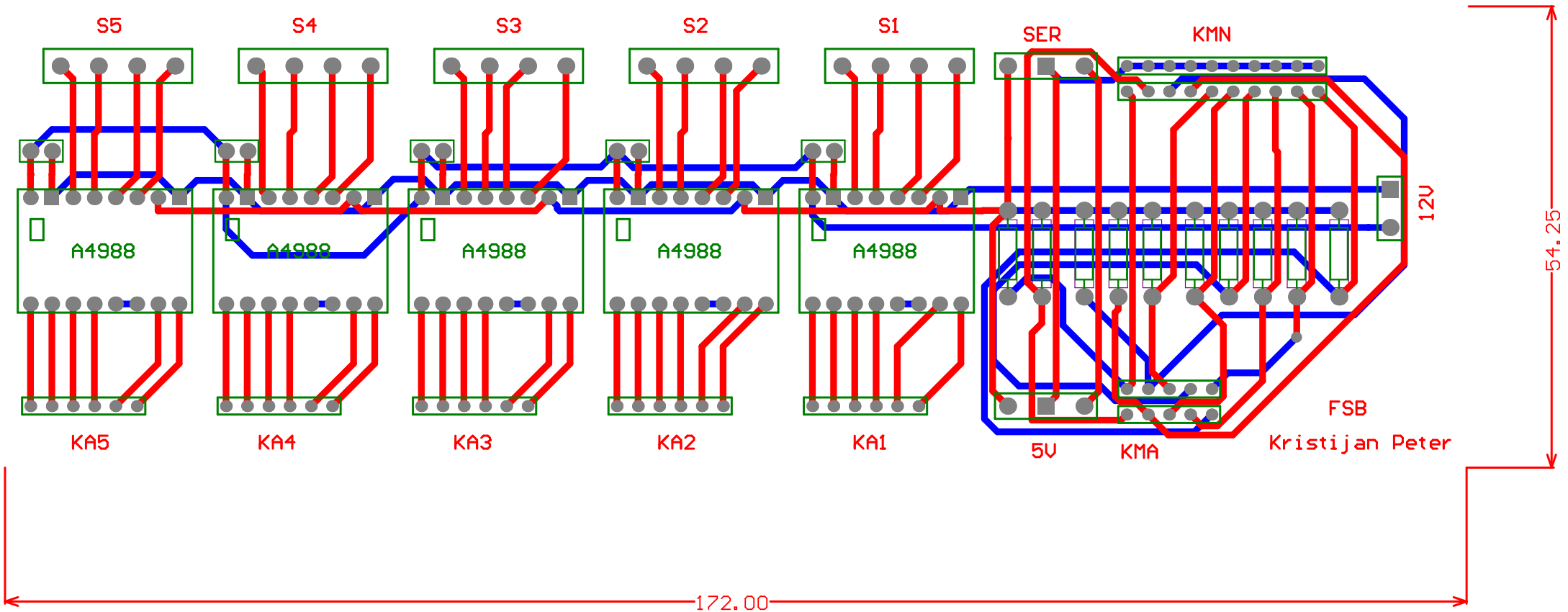
Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis	FSB Zagreb
	Projektirao		Kristijan Peter		
	Razradio		Kristijan Peter		
	Crtao		Kristijan Peter		
	Pregledao				
ISO - TOL				Objekt broj	
				R.N. broj	
	Napomena				Kopija
	Materijal PLA		Masa		
		Naziv Prst hvataljke		Pozicija	A4
	Mj. orginala 2:1				Listova
		Crtež broj: ERA 02-00-04			List





Broj naziva - code 0035198029		Datum	Ime i prezime	Potpis	FSB Zagreb
	Projektirao		Kristijan Peter		
	Razradio		Kristijan Peter		
	Crtao		Kristijan Peter		
	Pregledao				
ISO - TOL				Objekt broj	
				R.N. broj	
	Napomena				Kopija
	Materijal PLA		Masa		
		Naziv Zupčanik hvataljke		Pozicija	A4
	Mj. orginala 2:1				Listova
		Crtež broj: ERA 02-00-02			List


Konektor_arduino
Motor 1
Motor 2
Motor 3
Motor 4
Motor 5
A4988
ENB
MS1
MS2
MS3
STEP
DIR
RESET
SLEEP
12V
GND
5V
1A
1B
2A
2B
C1 4.7 uF
C2 4.7 uF
C3 4.7 uF
C4 4.7 uF
C5 4.7 uF
Konektor_napajanje
Konektor_step 1
Konektor_step 2
Konektor_step 3
Konektor_step 4
Konektor_step 5
Konektor_step
Konektor_mikroprekidač_arduino
Mikroprekidači
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
1 K
Konektor_5V i servo_ulaz
Konektor_5V i servo_izlaz
SERVO
Title
Tiskana pločica sa Driver-ima A4988
Size
A3
Number
Revision
Date: 14/01/2019
Sheet of
File: C:\Users\...\Drivers_for_motor_shema.SchDoc
Drawn By: Kristijan Peter


S5
S4
S3
S2
S1
SER
KMN
A4988
A4988
A4988
A4988
A4988
12V
KA5
KA4
KA3
KA2
KA1
5V
KMA
FSB
Kristijan Peter
54.25
172.00