TEHNIŠKI ŠOLSKI CENTER MARIBOR
VIŠJA STROKOVNA ŠOLA
STROJNIŠTVO

Domen VESELKO

UČNE SITUACIJE ZA ŠOLSKO ROBOTSKO CELICO ABB NA REALNEM ROBOTU

DIPLOMSKO DELO

Višješolski strokovni študij

Maribor, 2026

TEHNIŠKI ŠOLSKI CENTER MARIBOR
VIŠJA STROKOVNA ŠOLA
STROJNIŠTVO

Domen VESELKO

UČNE SITUACIJE ZA ŠOLSKO ROBOTSKO CELICO ABB NA REALNEM ROBOTU

DIPLOMSKO DELO

Višješolski strokovni študij

LEARNING SCENARIOS FOR A SCHOOL ABB ROBOTIC CELL ON A REAL ROBOT

GRADUATION THESIS

Higher vocational studies

Maribor, 2026

ZAHVALA

Zahvaljujem se mentorju Iztoku Milošiču, univ. dipl. inž. el., ki mi je pomagal pri izdelavi tega diplomskega dela predvsem z njegovimi izkušnjami, nasveti in z vsem znanjem, ki mi ga je predal na predavanjih in tudi individualno. Za to se mu iskreno zahvaljujem.

Seveda pa se zahvaljujem tudi staršem, ki so me podpirali ne samo skozi študij, ampak skozi celotno življenje.

Za vso znanje, ki ga premorem, bi se rad zahvalil tudi vsem predavateljem, saj mi je njihovo znanje prišlo marsikje prav.

Hvala!

IZJAVA O AVTORSTVU

Podpisani Domen Veselko, rojen 9. 1. 2003 v Mariboru, študent Tehniškega šolskega centra Maribor, Višje strokovne šole, programa strojništvo izjavljam, da je diplomsko delo z naslovom *Učne situacije za šolsko robotsko celico ABB na realnem robotu* avtorsko delo.

V diplomskem delu so vsi uporabljeni viri in literatura konkretno navedeni; teksti niso prepisani brez navedbe avtorjev.

Diplomsko delo je lektorirala Anka Jemenšek prof, and, in slov, jezika, ključno dokumentacijsko informacijo je prevedel Domen Veselko.

Kraj in datum: _____

Lastnoročni podpis študenta/-ke: _____

MENTORSTVO

Diplomsko delo je zaključek Višješolskega strokovnega študija, smer strojništvo, opravljeno je bilo na Tehniškem šolskem centru Maribor, Višji strokovni šoli.

Študijska komisija Tehniškega šolskega centra Maribor, Višje strokovne šole je za mentorja diplomskega dela imenovala Iztoka MILOŠIČA, univ. dipl. inž. el.

Komisija za oceno in zagovor:

Predsednik: _____

Član/mentor: _____

Član: _____

Član/somentor: _____

Datum diplomskega izpita: _____

POVZETEK

V DD sem na šolski robotski celici moral pripraviti ABB vaje, ki bodo delovale kot učni proces. DD je moralo tudi zajemati varno delo z robotom, ročno vodenje robota in vaje za učenje osnovnih gibov robota. Zraven tega je bilo tudi potrebno pojasniti, kako se uporablja prijemalo, kako se nastavljajo koordinatni sistemi, uporabo vhodnih in izhodnih signalov ter logične funkcije in uporabo strojnega vida. Za izhodišče sem dobil že nekatere modele, ki sem jih bil primoran popraviti, nekatere pa narediti. Te komponente so se izdelale s 3D tiskom. Po tisku sem lahko začel z vajami in s programiranjem.

Pred začetkom programiranja sem dobil že 3D modele, ki sem jih mogel dopolnit po zahtevah, nekatere pa tudi na novo narisati in izdelati delavniško risbo.

Na robotu bom pripravil vse potrebne programe, ki bodo zagotavljali pravilno in zanesljivo delovanje robota. Programi bodo napisani in razloženi na način, ki bo razumljiv tudi osebam, ki z robotom nimajo veliko izkušenj, tako da bo lahko vsak, ki bo prebral navodila, razumel osnovno logiko delovanja in način uporabe. Poseben poudarek bo namenjen jasni in sistematični razlagi posameznih korakov pri programiranju.

V nalogi bo opisano, kako se pravilno zapisujejo programske vrstice na učni enoti robota ter kako poteka postopek ustvarjanja posameznih programov. Prav tako bodo predstavljeni primeri programov in razlaga njihovega delovanja, da bo bralec lažje razumel, kako robot izvaja določene naloge. Namen tega dela je prikazati celoten proces programiranja robota – od osnovnih nastavitev do končne izvedbe programa.

Poleg programiranja bo v nalogi opisan tudi pravilen način upravljanja robota. Predstavljena bodo osnovna pravila varne uporabe ter priporočila, ki pomagajo preprečiti morebitne napake ali zaplete pri delu z robotom. S tem bo zagotovljeno, da bo uporaba robota varna, učinkovita in razumljiva tudi za nove uporabnike.

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

ŠD Dd

DK 621.865.8:004.42

KG Učne situacije, ABB Robotics, robotska roka ABB, modeliranje, programiranje

AV VESELKO, Domen

SA MILOŠIČ, Iztok (mentor)

KZ SI-2000 Maribor, Zolajeva 12

ZA Tehniški šolski center Maribor, Višja strokovna šola

LI 2026

IN UČNE SITUACIJE ZA ŠOLSKO ROBOTSKO CELICO ABB NA REALNEM ROBOTU

TD Diplomsko delo (višješolski strokovni študij)

OP XIII, 107 str., 160 sl., 14 pril., 12 vir.

IJ sl

JI sl/en

AI *Diplomsko delo obravnava pripravo učnih situacij za šolsko robotsko celico ABB. Učne situacije na robotski roki ABB bodo delovale kot učni proces. Potrebno je bilo da zajema kako je pravilno in varno delovanje s robotom, ročno vodenje robota, vaje za učenje osnovnih gibov robota, uporaba prijemala, nastavljanje koordinatnih sistemov, uporaba vhodnih in izhodnih signalov ter logične funkcije in uporaba strojnega vida. Potrebno je bilo tudi dodelati komponente, ki so bile pridobljene pred začetkom diplomske naloge, nekatere pa izdelati na novo. Ti modeli, so se izdelali s 3D tiskom. Po končanem tisku so se lahko začele izdelovati vaje.*

KEY WORDS DOCUMENTATION

DN Dd

DC 621.865.8:004.42

CX Training scenarios, ABB Robotics, ABB robotic arm, modeling, programming

AU VESELKO, Domen

AA MILOŠIČ Iztok (mentor)

PP SI-2000 Maribor, Zolajeva 12

PB Technical School Centre Maribor, Higher Vocational College

PY 2026

TI NASLOV DIPLOMSKEGA DELA v angleščini

DT Graduation Thesis (Higher vocational studies)

NO XIII, 107 p., 160 fig., 14 ann., 12 ref.

LA sl

AL sl/en

AB *The thesis deals with the preparation of learning scenarios for a school robotic cell using an ABB robot. The learning scenarios on the ABB robotic arm function as part of the teaching process. It was necessary to include proper and safe robot operation, manual robot control, exercises for learning basic robot movements, the use of a gripper, setting coordinate systems, the use of input and output signals, logical functions, and the application of machine vision. It was also necessary to refine components that had been acquired before the start of the thesis and to design and manufacture some new ones. These models were produced using 3D printing. After the printing process was completed, the development of the exercises could begin.*

KAZALO VSEBINE

KAZALO SLIK

1 UVOD

1.1 OPREDELITEV PROBLEMA

V moji diplomski nalogi bom na začetku zmodeliral vse potrebne pripomočke za pomoč pri izvedbi programov, se uspešno seznanil z delovanjem robota, tudi z lastnostmi in ročnim vodenjem. Na robotu bom napisal programe ki bodo delovali brezhibno. Za brezhibno delovanje programov bom pa tudi potrebne koorditnatne sisteme (tool, wobj, user frame …) ki jih bom potreboval pri določenih programih. Ker pa bo robot dostopen drugim študentom in dijakom, se bodo tudi oni lahko iz njegovega delovanja nekaj naučili.

1.2 NAMEN IN CILJI DIPLOMSKEGA DELA

Namen diplomskega dela je pripraviti učne situacije za ABB robotsko celico in hkrati pridobiti znanje iz programiranja industrijskega robota ter se seznaniti z drugim načinom programiranja sodelujočega robota.

Cilji so:

- Modeliranje vseh potrebnih pripomočkov, orodij in obdelovancev.
- Seznaniti se z delovanjem industrijskega robota GoFa.
- Naučiti se programirati robotsko roko.
- Podati poročilo o uspešno opravljenem delu.
- Uspešno programirati robota in ga spraviti v pravilno delovanje.

2 PREGLED STANJA

Namen moje diplomske naloge je bil, da se na robotu naredijo programi za robotsko roko. Potrebno je tudi nastaviti potrebne koorditnatne sisteme (tool, wobj, user frame …). Na robotu so v delujočem stanju vse osi, tudi prijemalo se zapira. Robota se da ročno voditi. Na robotu sem pripravil vaje:

Vaja 1: Prenos 1 valja preko ovire
Vaja 2: Prenos 3 valjev preko ovire
Vaja 3: Prenos 3 valjev preko ovire z računanjem offset
Vaja 4: Prenos 10 valjev preko ovire
Vaja 5: Prenos 10 valjev iz ene strani na drugo in nazaj preko ovire
Vaja 6: Prenos 10 valjev v poševnino
Vaja 7: Varjenje 2D konture
Vaja 8: Varjenje 2D konture z oblikami
Vaja 9: Varjenje 3D konture z oblikami

2.1 PODJETJE ABB

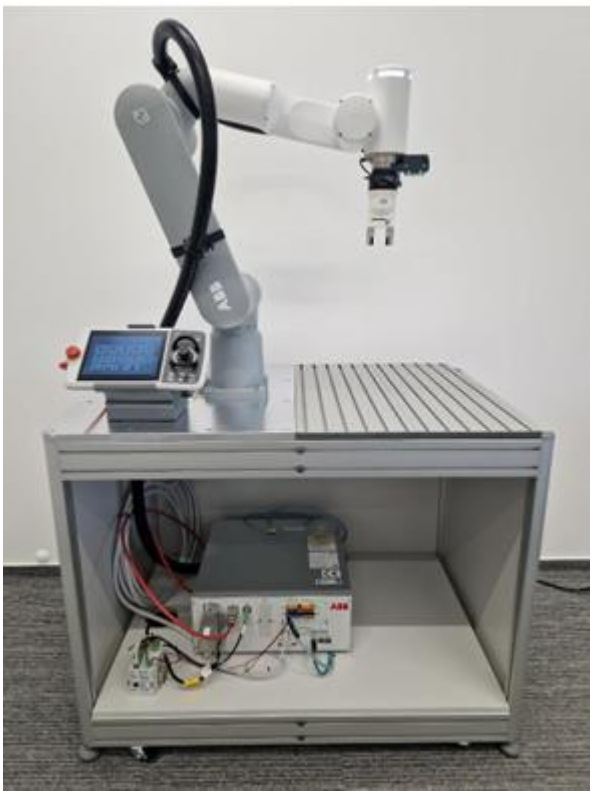
Podjetje ABB je vodilno podjetje in svetovni tehnološki proizvajalec na področju elektrifikacije in avtomatizacije. Njihov cilj je omogočiti bolj trajnostno in z viri učinkovito prihodnost. Z znanjem na področju inženiringa in digitalizacije pomagajo industrijam, da lahko delujejo z visoko zmogljivostjo, hkrati pa postajajo učinkovitejši in produktivnejši. Njihova zgodovina sega 140 let nazaj. Njihov logotip je obarvan z rdečimi velikimi črkami ABB, ki ga lahko vidimo na spodnji sliki.

Slika 1: Logotip podjetja ABB

(ABB, 2025)

2.2 UPORABLJENA ROBOTSKA CELICA

Slika 2: Robotska roka CRB 15000



(ABB, 2024)

Za delujočo robotsko celico je potrebno več delov, da lahko robotska roka deluje. Te komponente, ki sestavljajo robotsko celico, so:

- Robotski krmilnik OmniCore.
- Manipulator CRB15000 (GoFa).
- Krmilna konzola Omnicore Flexpendant.
- DSQC1030 vzhodno – izhodni modul s 16 digitalnimi vhodnimi signali in 16 digitalnimi izhodnimi signali.
- DSQC1032 vzhodno – izhodni modul s 4 analognimi vhodnimi signali in 4 analognimi izhodnimi signali.
- Sick PLOC2D-611-6RB ABB GoFa strojni vid.
- Zimmer LWR50L-03-00003-A el. dvoprstno prijemalo z Zimmer LWR50F-13-01-A hitromenjalno ploščo za orodja.
- ABB PMA paket za zaščito povezanih kablov na manipulatorju.

Voziček na kolesih z zavoro in s prilagodljivih fiksirnih nogah, s polico za krmilnik ter delovno površino iz 19 x 270 alu profili.

3 PRIPRAVA ROBOTSKE CELICE ZA UČNE SITUACIJE

Na začetku, je bilo potrebno načrtovati vse vaje, podrobno se seznaniti s tem, kaj točno se želi, da bo robot v programu naredil ter kakšen je želen končni rezultat.

Kasneje je bilo potrebni izdelati modele obdelovancev, orodij in drugih komponent v programu SolidWorks. Te modele je nekatere bilo potrebno narisati na novo, nekatere pa je bilo potrebno samo malo spremeniti in jih ustrezno dodelati tako, da so bili primerni za vaje na robotu.

Pred začetkom programiranja na učni enoti sem od mentorja tudi prejel literaturo, kjer so bila navodila za osnovni začetek.

Za vsak program, ki je napisan na robotu, je opisana vsaka vrstica, ki je pomembna za sestavo programa na realnem robotu. Za točno vsako vrstico je opisan vsak korak, kaj pomenijo določene črke in besede, ki so zapisane v vrstici, ki jih robot kasneje prebere.

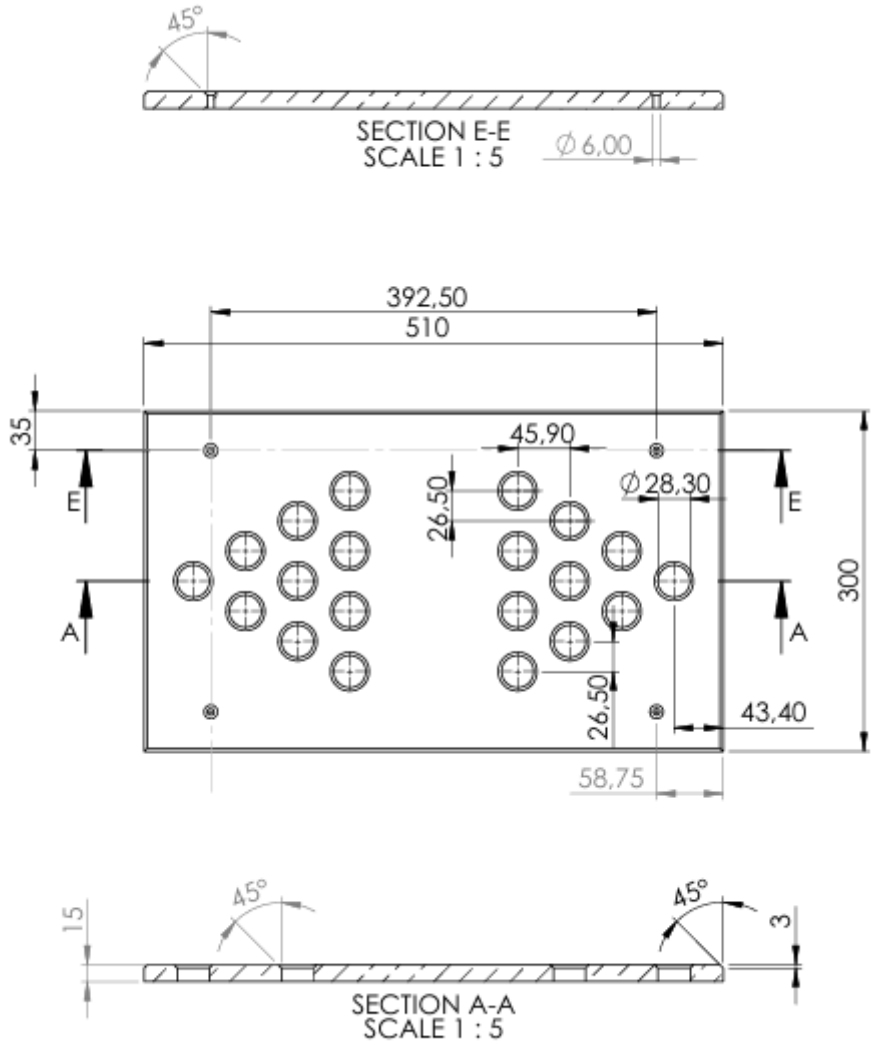
3.1 UČNE SITUACIJE NA ROBOTU

Na robotu je 7 učnih situacij. Imam pobiranje enega valja in prenos čez oviro, pobiranje 3 valjev in prenos čez oviro ter pobiranje 10 valjev in prenos čez oviro. Z valji je še ena vaja, in sicer pobiranje 10 valjev in prenos na poševnino, kjer robot tudi opravi različno gibanje pri tej vaji, saj more valje dati poševno v zastavljen cilj. Nato pa so vaje, kjer ima robot nalogo, da vari. Za prvo enostavno vajo varjenja je bila 2D kontura, malo zahtevnejša vaja je bila 2D kontura z oblikami ter najzahtevnejša vaja 3D kontura z oblikami. Robot mora imeti varilno pištolo vedno pod kotom 45°na linijo, kjer vari. Hitrost pri varjenju in prenosu valjev mora biti primerna zmožnostim robota ter kvalitetni, a hkrati hitri izvedbi.

3.2 MODELIRANJE KOMPONENT

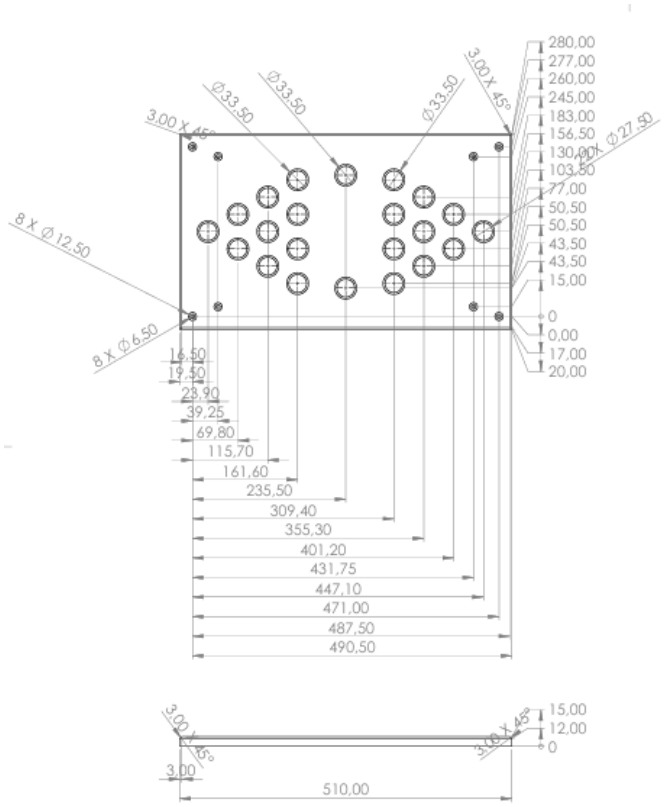
Preden sem lahko začel izdelovati programe na robotu, sem moral nekatere komponente, ki sem jih predhodno dobil, popraviti, nekatere sem moral tudi povsem na novo izdelati. Po izdelanih načrtih sem predal te načrte mentorju, da se je lahko naredila izdelava teh komponent v 3D printerju. Te komponente sem kasneje uporabil za mojo diplomsko delo.

Slika 3: Dobljena plošča, ki sem jo kotiral



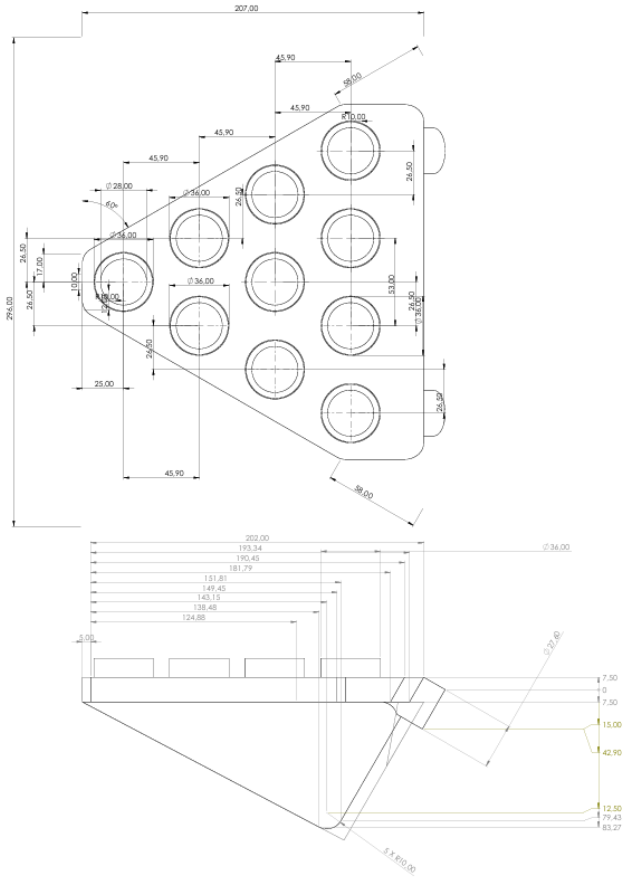
Na tej sliki imamo prikaz plošče, ki ima luknje za valje, kjer jih robot odlaga. Tukaj sem naredil dva prereza, skotiral sem širino in dolžino, velikost lukenj ter razmak med njimi.

Slika 4: Kotirana osnovna plošča, ki sem jo popravil



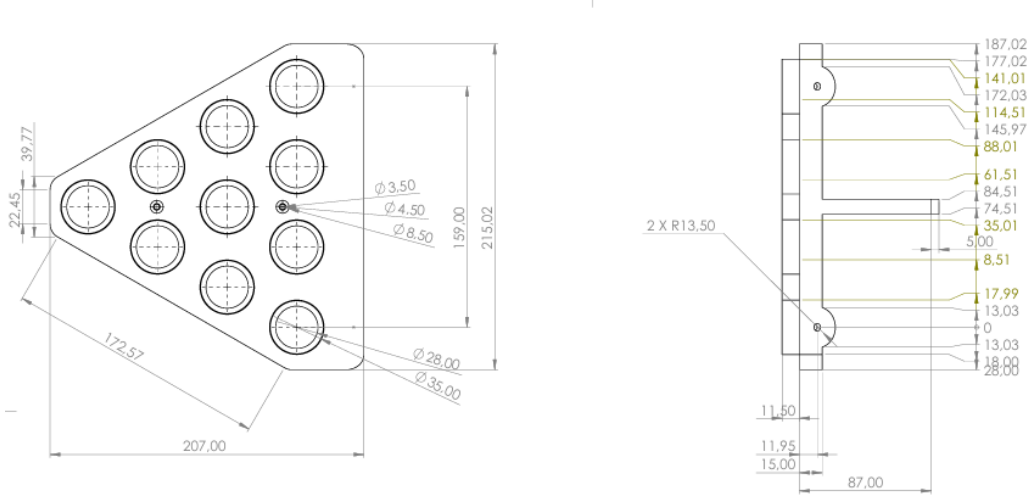
Na tej plošči sem naredil več sprememb, in sicer luknje po sredini, s katerim se bo ovira pritrdila na ploščo in se posledično ne bo mogla premakniti in bo vedno na pravem mestu. Tudi na koncih plošče so dodatne luknje s katerimi, se bo lahko plošča dodatno pritrdila na tudi več različnih mest.

Slika 5: Dobljena poševnina za odlaganje 8 valjev



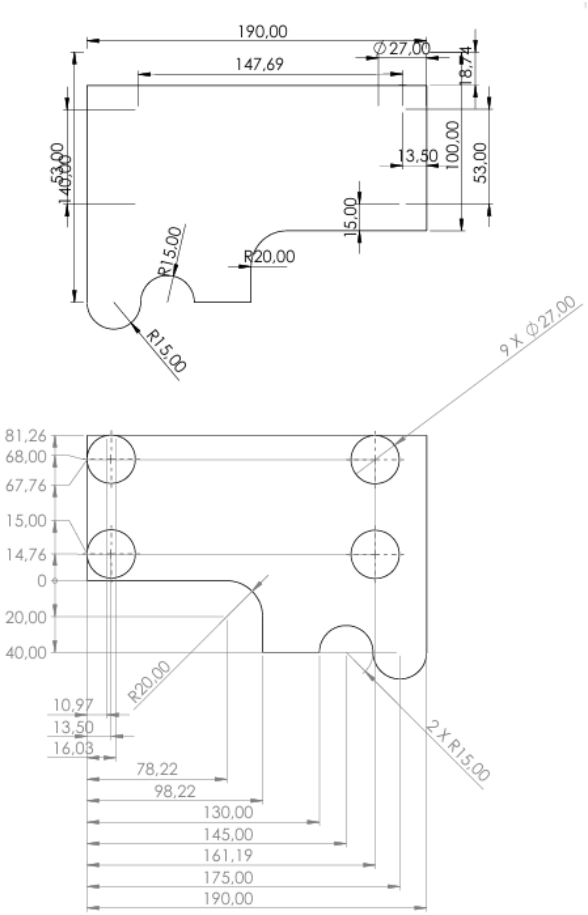
Na tej sliki imamo prikazano poševen del, kjer imamo luknje za odlaganje valjev. Tukaj sem se odločil za kotiranje lukenj, razmaka med njimi, skotiral sem tudi spodnji luknji, s katerimi je ta predmet nameščen v ploščo. Odzadaj je tudi del poševnine, ki drži ploščo na mestu, da ne pade.

Slika 6: Popravljena poševnina



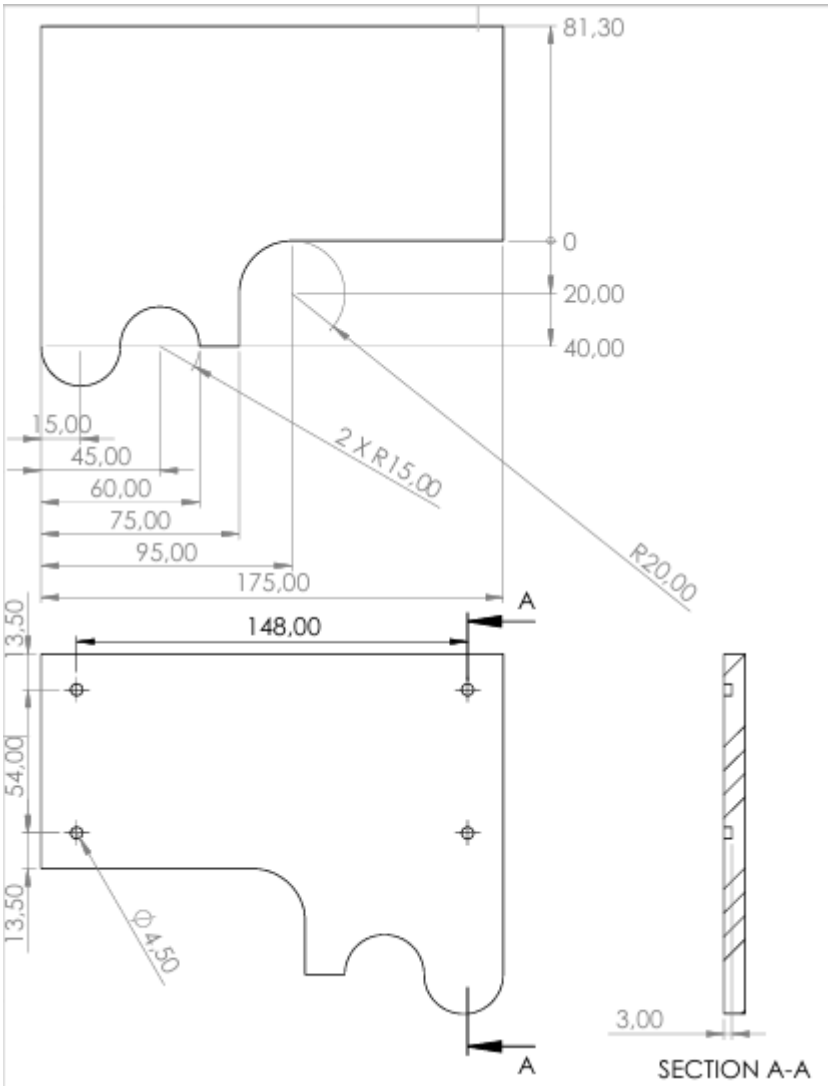
Na tej plošči so se naredile sredinske luknje, saj se bo zadnji del kasneje pritrdil na del, kamor se zlagajo valji. S tem se bo doseglo, da se bo porabilo manj materiala.

Slika 7: Dobljena kontura 2D



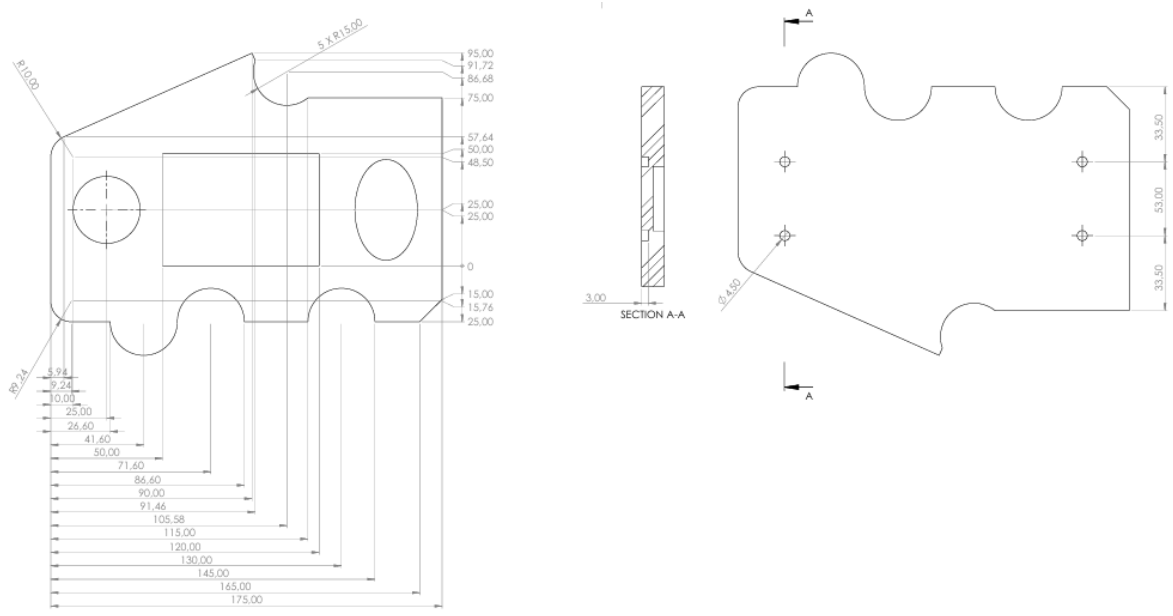
Na tej sliki imamo del, kjer bo robot po robu tega predmeta sledil robu in delal zvare. Tukaj sem kotiral vse neravne dele predmeta in vse štiri nogice. Prikazal sem tudi, koliko so nogice po širini visoke. Njihov fi je primeren za namestitev v začetno ploščo, kjer bo skozi na mestu in se med delom ne bo premikalo.

Slika 8: Popravil sem 2D konturo



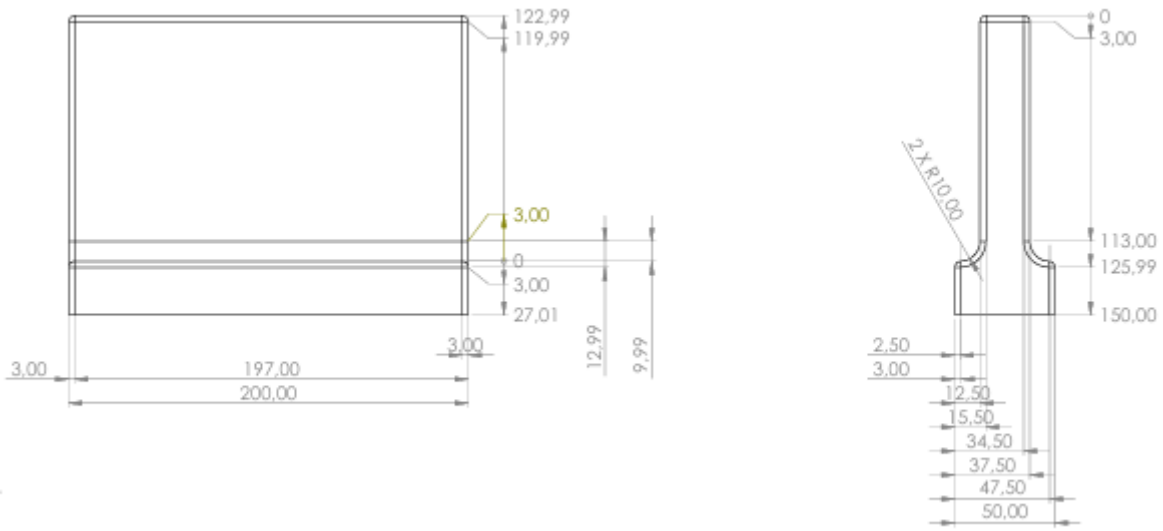
Pri tem liku so bile spremembe samo pri dolžini lika. Lik sem skrajšal na mero tako, da se zaključi vzporedno z nogicami, ki pa so odstranjene. Spodnji del konture je ostal enak.

Slika 9: Dobljena 2D kontura Adam z oblikami



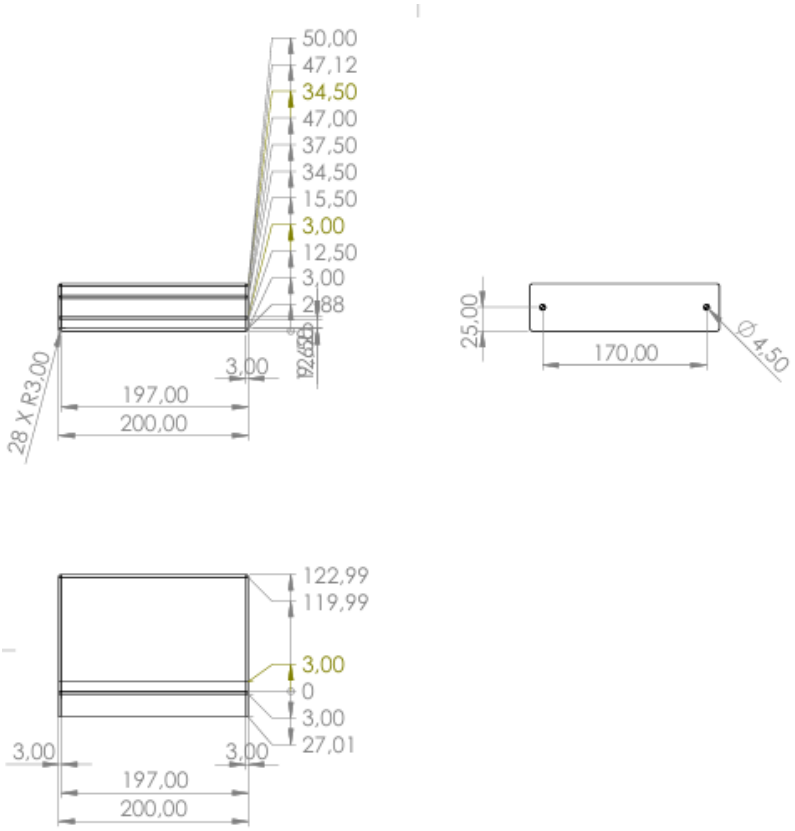
Na tej sliki vidimo zelo podoben model kot je bilo na prejšnji, vendar je opazna tudi bistvena razlika, saj ima ta model v telesu izdelane tri oblike; to so krog, kvadrat in valj.

Slika 10: Dobljena ovira za robota



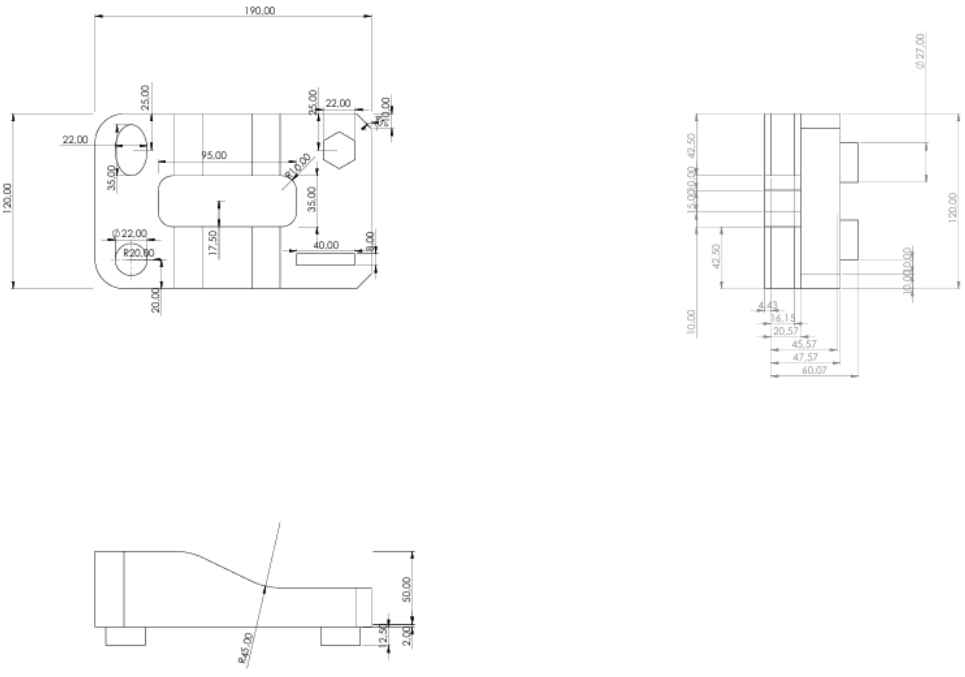
Na tej sliki je videna ovira. Kotiral sem jo po širini in višini, navedel sem tudi, pod katerim polmerom je narejen zaviti del te ovire. To oviro bom uporabil na plošči in bo ovira za robota. Preko nje bo robot prenašal valje in se preko nje vračal na začetno pozicijo. Ker v plošči ni lukenj za oviro, je na ploščo postavljena na približno pozicijo in je kasneje ne premikam.

Slika 11: Ovira z luknjama na spodnji strani, ki je popravljena



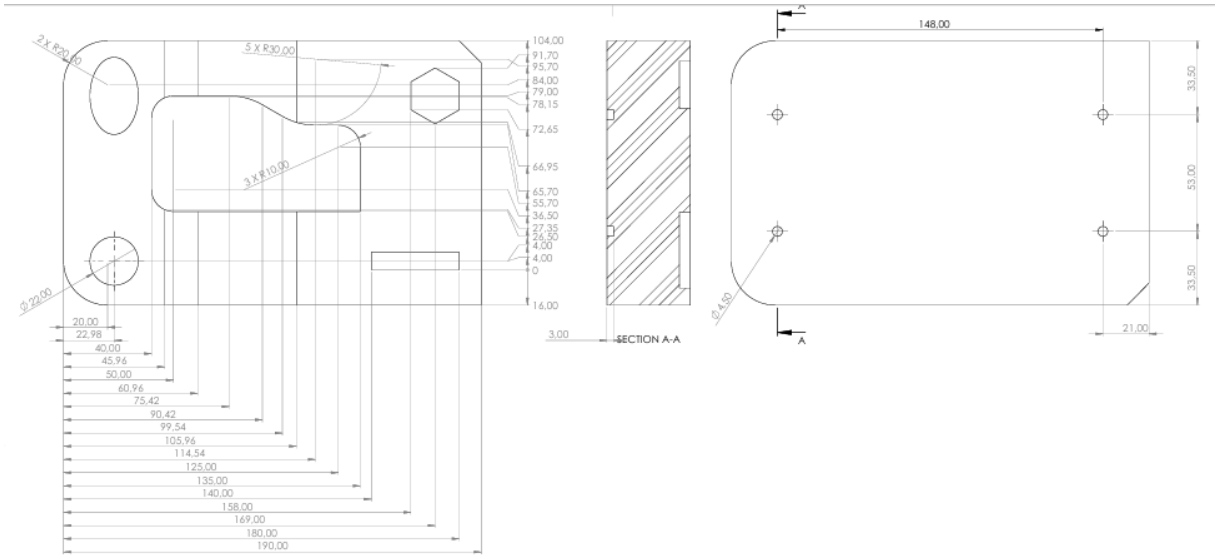
Na oviri sem spremenil to, da sem dodal dve luknji, kamor bo prišel vijak takrat, ko bodo nogice privijačene na oviro.

Slika 12: Dobljena 3D kontura



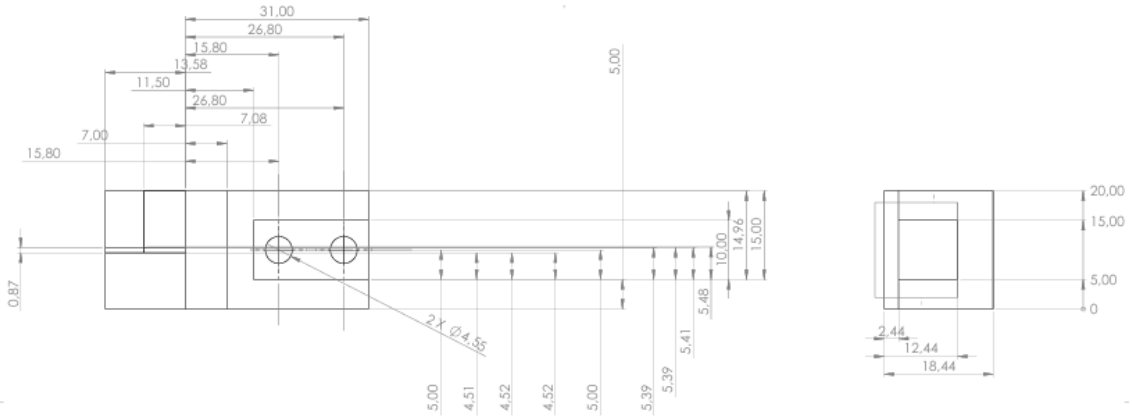
Pri tem predmetu sem dimenzioniral vse dele, ki so na vrhu tega modela. Na tem modelu sem posebej dimenzioniral krog, valj, kvadrat in šest kotnik. Vmes pa je tudi odprtina, ki je pod kotom, po kateri bo robot tudi sledil in delal zvare. Razvidno je, da sem tudi na tem modelu izdelal in dimenzioniral spodnje nogice, ki se nato v ploščo vstavi in ko robot dela na njem, se ne more premakniti.

Slika 13: 3D kontura, ki je popravljena



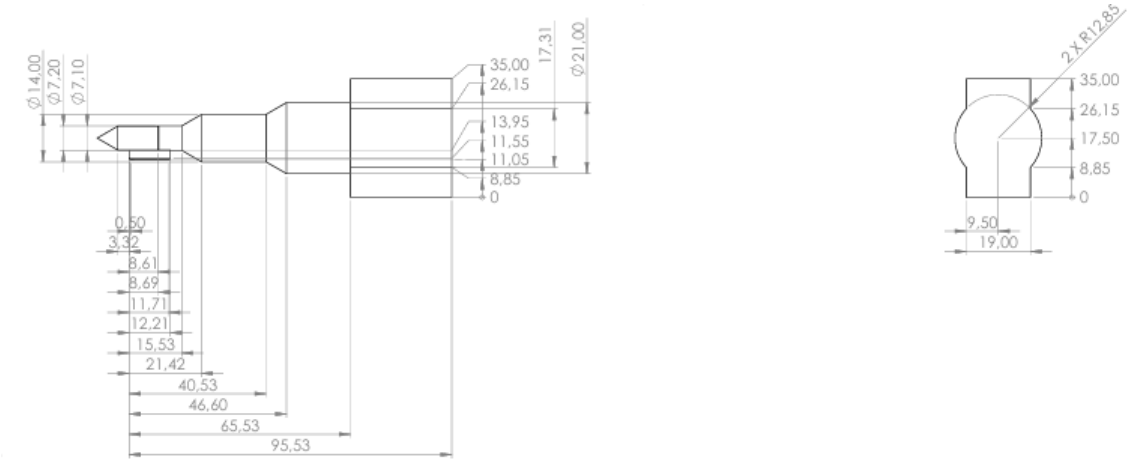
Pri tej konturi je sprememb malo več. Predvsem se pozna, da je ta kontura se zmanjšala v debelini. Osrednji del je dobil malo drugačno obliko, kjer bo robot varil. Kontura je tudi ostala brez nogic, saj se bo tako porabilo manj materiala. Naknadno se bodo nogice pritrdile na konturo.

Slika 14: Dobljen prsti krenker



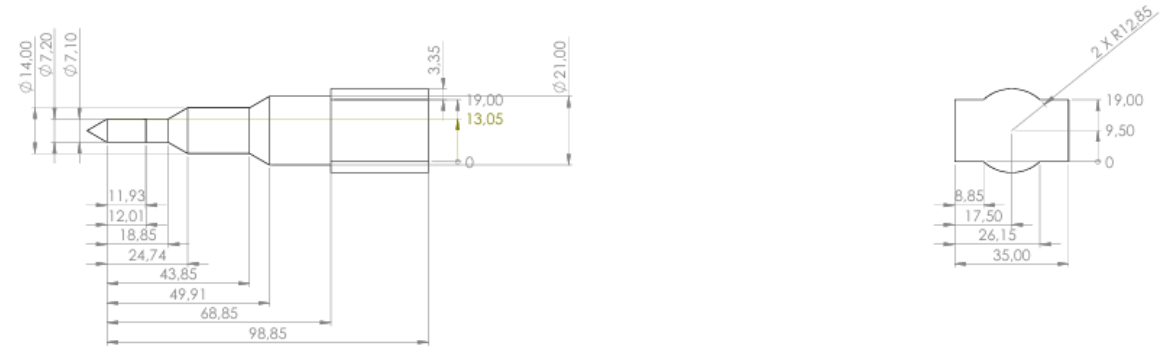
Pri tem predmetu sem dimenzioniral vse potrebne dele. Ta del se uporabi za prijemalo na orodju. Potrebno je bilo dimenzionirati vse odprtine, da se bo prijemalo pritrdilo na robota.

Slika 15: Dobljena konica za ležaj z zastavico



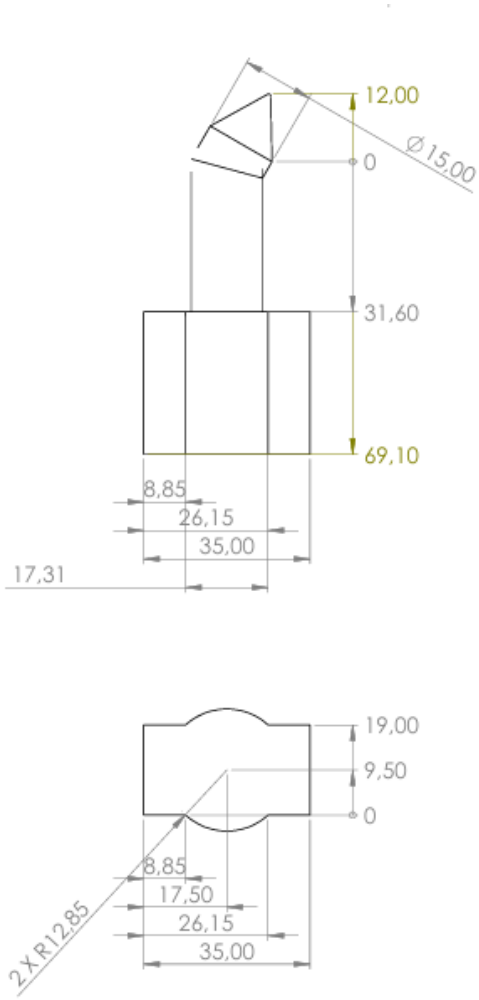
Tukaj sem naredil konico za varjenje, ki je vpeta v glavo robota.

Slika 16: Popravljena konica za ležaj, na kateri ni zastavice



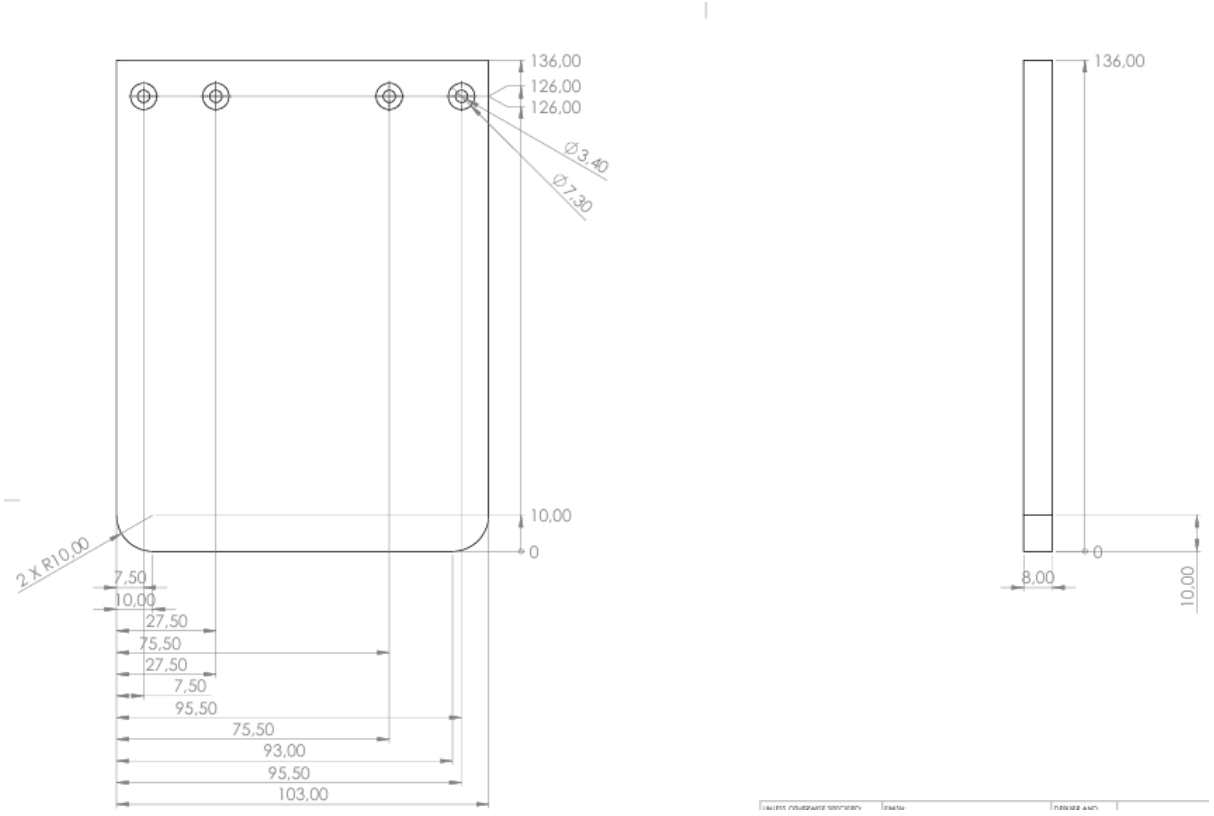
Ta konica je narejena enako kot tista z zastavico, vendar je pa ta nima.

Slika 17: Dobljena varilna pištola



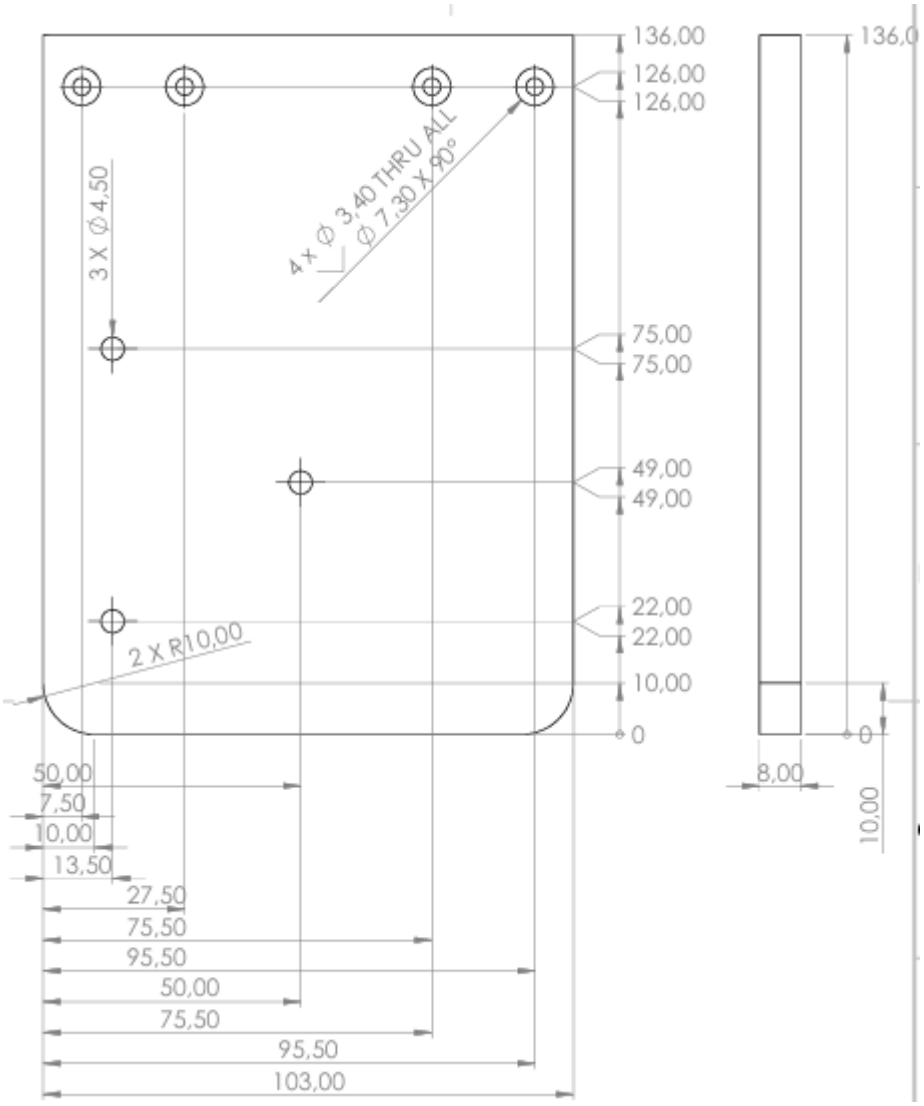
Na tej sliki je prikazana varilna pištola, ki je malo krajša. Iz načrta je razvidno, da je konica pod kotom, tako tudi dosežemo, da lažje pridemo do samega materiala oz. do točke, kjer je to potrebno za varjenje.

Slika 18: Dobljena plošča za nosilec



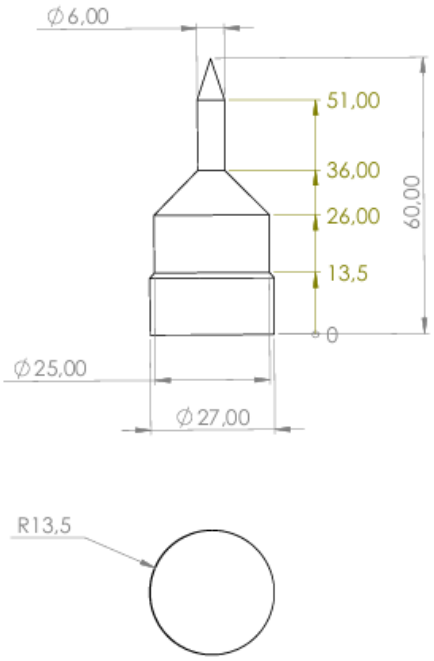
Izdelal sem tudi ploščo za nosilca z vnaprej pripravljenimi luknjami, kamor se bodo privili vijaki in nosilci, v katerih bodo kasneje različna orodja, da si jih bo robot lahko vzel.

Slika 19: Dobljena plošča za nosilec, ki sem jo popravil



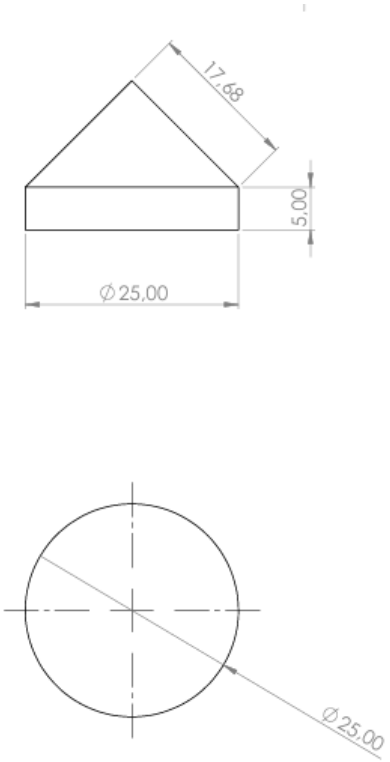
Dodal sem luknje za nogice in popravil luknje, kjer se bosta pri vijačila držala za orodje.

Slika 20: Nova komponenta brez popravljanja



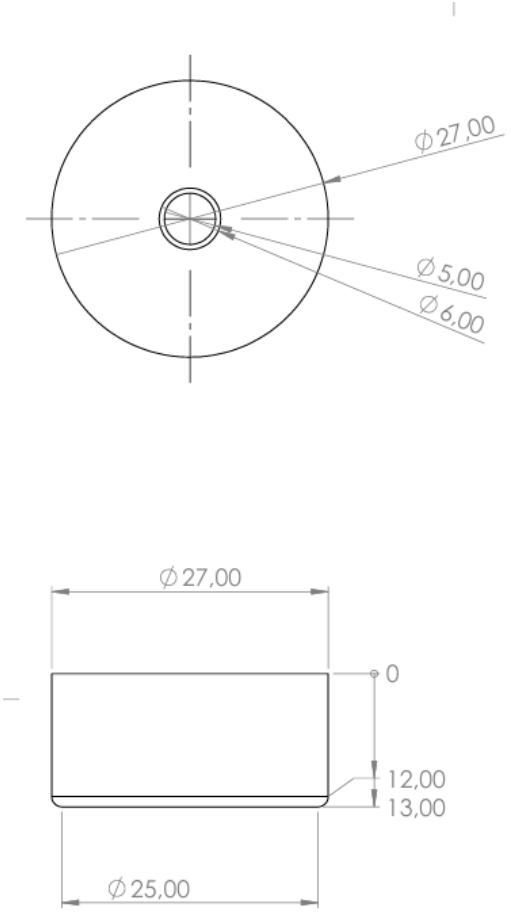
Nova komponenta bo primerna za v prijemalo orodja in bo lahko uporabljena za varjenje.

Slika 21: Novo narejen pripomoček za določanje TCP v prijemalu



Nova komponenta, ki je enostavna in se bo uporabila za v prijemalo.

Slika 22: Novo narejena nogica za pritrditev na različne komponente



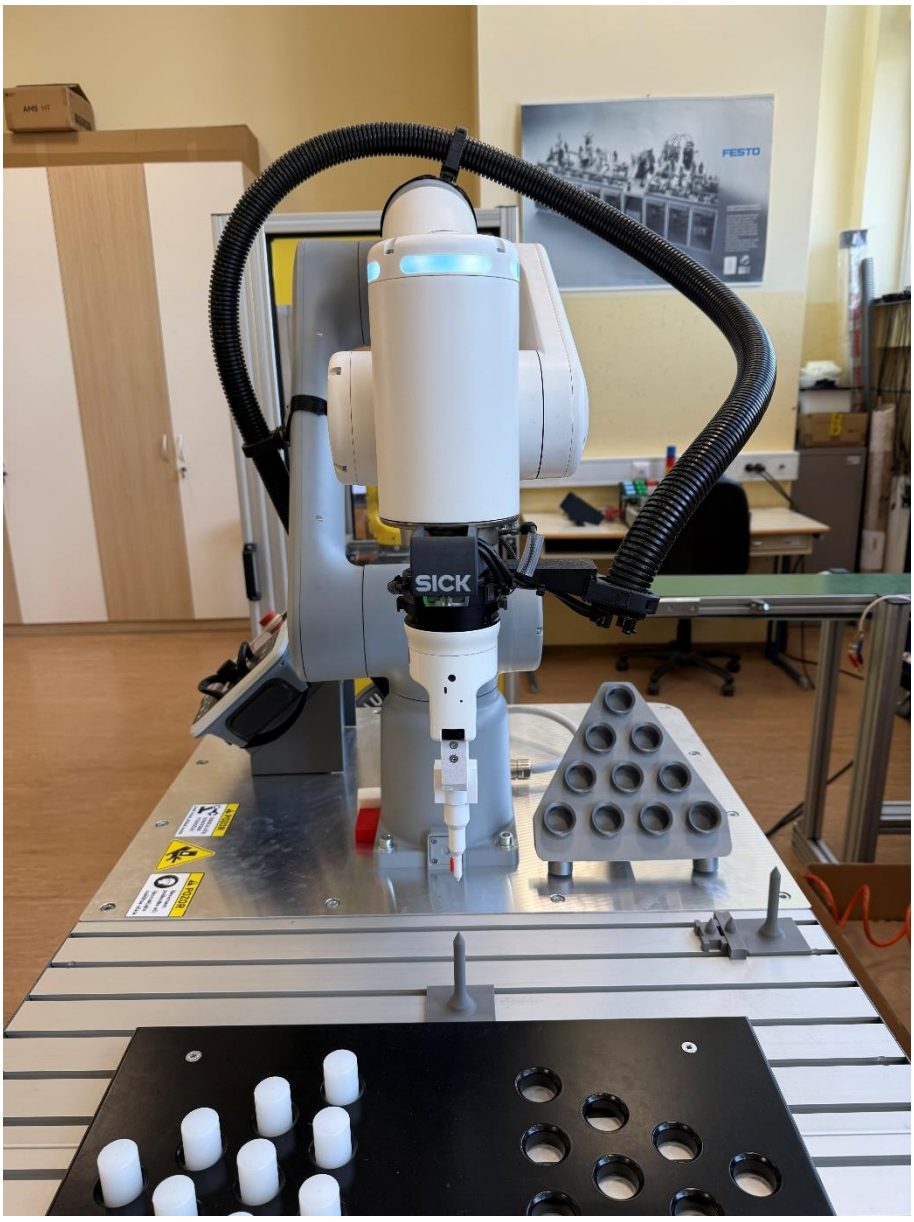
Izdelal sem nogico, ki se bo privijačila na komponente, ki potrebujejo nogico.

3.3 VAJA 1: ROČNO VODENJE ROBOTA

V vaji 1 se boste naučili, kako se ročno vodi robota, kakšen je namen tipke DMS ter E – STOP. Pridobili boste potrebno znanje za ročno vodenje robota, kako se odpre in zapre prijemalo. Razloženo boste imeli, kakšni so vse načini za ročno vodenje robota.

Vaja 1 vsebuje, kako pravilno delovati z robotom. Robot ima več integriranih varnostnih funkcij. Pri kontaktu z drugim predmetom se robot varnostno ustavi, tako tudi velja, če pride v stik z osebo. Robot je sestavljen z zaobljeno konstrukcijo in to omogoča, da ni nevarnih ostrih robov. Robot ima tudi varnostno lučko. Če sveti rdeča barva, to pomeni, da robot ni v stanju premika in da nakazuje na nekakšno napako, ki pa jo je potrebno ročno popraviti na učni enoti.

Slika 23: Prikaz lučke na robotu



3.3.1 Pregled robota pred obratovanjem

Robota je pred delovanjem potrebno pregledati. Za nastavitev varnostnih funkcij pri ročnem vodenju robota in postopnem programiranju je potrebno robota premikati z zmanjšano oziroma varno hitrostjo. Tak način dela zmanjšuje tveganje za trk robota z delovno mizo, obdelovanci ali drugimi predmeti, ki se nahajajo v delovnem prostoru robota. Počasnejše premikanje omogoča tudi boljši nadzor nad gibanjem robota ter večjo natančnost pri določanju programskih točk.

Pred začetkom ročnega vodenja je potrebno preveriti pravilno delovanje vseh varnostnih funkcij robota, saj te zagotavljajo varno interakcijo med operaterjem in robotom. Poseben pomen ima varnostno stikalo, ki se nahaja na upravljalni enoti robota. Operater mora stikalo držati v aktivnem položaju, kar pomeni, da ga pritisne do srednje stopnje. Le v tem položaju je omogočeno premikanje robota glede na operaterjeve ukaze.

Slika 24: Prikaz tipke DMS

(ABB, 2024)

Če operater stikalo DMS pritisne premočno ali ga popolnoma spusti, se robot takoj ustavi, saj sistem to zazna kot potencialno nevarno situacijo. Na ta način je zagotovljena dodatna stopnja varnosti pri ročnem upravljanju robota.

V primeru nujne situacije je na upravljalni enoti na voljo tudi gumb za zasilno zaustavitev (Emergency Stop), ki je običajno izveden kot velika rdeča tipka. Aktivacija tega gumba povzroči takojšnjo in popolno zaustavitev robota, s čimer se prepreči morebitna nevarnost za operaterja ali okolico.

Slika 25: Prikaz tipka E – STOP

(ABB, 2024)

Takšen način upravljanja predstavlja pomemben del varnostnih postopkov pri programiranju in upravljanju industrijskih robotov, saj zmanjšuje možnost poškodb operaterja ter materialne škode na opremi in delovnem okolju.

3.3.2 Nevarnosti na robotu

Med obratovanjem industrijskega robota je potrebno dosledno upoštevati varnostna pravila, saj nepravilno ravnanje lahko povzroči poškodbe operaterja ali poškodbe opreme. Operater ne sme posegati z roko ali s katerim koli drugim predmetom v delovno območje robota, kadar je robot v gibanju. Takšno ravnanje predstavlja tveganje za mehanske poškodbe, saj lahko pride do trka z robotsko roko ali do ujetja med gibljive dele robota in okoliško opremo.

Barvna signalizacija omogoča hitro prepoznavanje morebitnih težav ali sprememb v načinu delovanja. Rdeča barva običajno pomeni, da je prišlo do napake ali varnostne zaustavitve, zaradi katere se robot ne more premikati, dokler napaka ni odpravljena na upravljalni enoti oziroma Teach Panelu (TP). Bela barva označuje, da so napake odpravljene in da je robot pripravljen na delovanje. Zelena barva pa pomeni, da robot deluje v avtomatskem načinu in izvaja program, ki je naložen na upravljalni enoti.

V primeru, da robot v katerem koli načinu delovanja izvede nenaden ali nepričakovan gib, je potrebno njegovo delovanje takoj zaustaviti in preveriti vzrok za nepravilno delovanje. Po potrebi je treba program ustrezno popraviti ali ponovno preveriti parametre gibanja.

Robot naj med delovanjem vedno nadzoruje usposobljena oseba, ki je seznanjena z upravljanjem robota in varnostnimi postopki. Stalni nadzor omogoča hitro ukrepanje v primeru nepravilnosti ter zmanjšuje možnost nastanka poškodb ali materialne škode.

Pri delu z industrijskimi roboti obstaja več potencialnih nevarnosti, na katere mora biti operater posebej pozoren. Med najpogostejše spadajo nenaden premik robota, možnost ujetja med

robotsko roko in okoliško opremo, zlasti pri večjih robotih, ter visoke temperature površin, ki lahko povzročijo opekline. Dodatno nevarnost predstavlja tudi nepričakovan premik robota ob sprostitvi zavor, ki se lahko pojavi med servisiranjem ali ročnim premikanjem robota. Zaradi navedenih tveganj je nujno dosledno upoštevanje varnostnih postopkov in stalna pozornost operaterja med obratovanjem robota.

3.3.3 Ročna načina vodenja robota

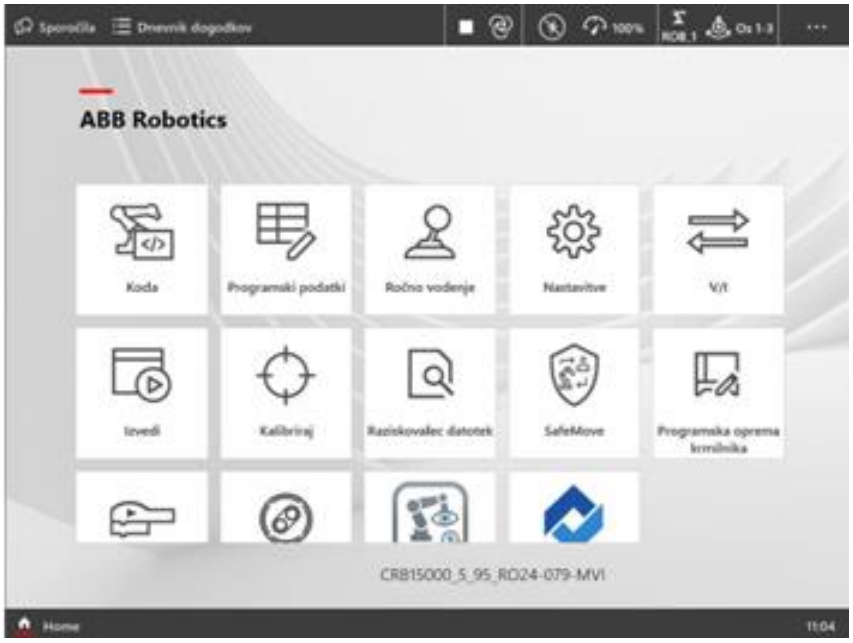
Pri robotu ABB GoFa CRB 15000 obstaja več načinov ročnega vodenja robota. Ti načini omogočajo operaterju, da robota premika počasi in natančno pri učenju točk, testiranju programa ali nastavljanju orodja.

Slika 26: Krmilna palica

(ABB, 2024)

S to krmilno palico boste dosegli premikanje robota v smer, v katero si vi želite. Potrebno je, da veste, v katerem načinu ročnega vodenja imate robota. Možnosti za ročno vodenje imate:
Os 1-3,
Os 4-6,
Linearno in
Reorient.

Slika 27: Domač zaslon

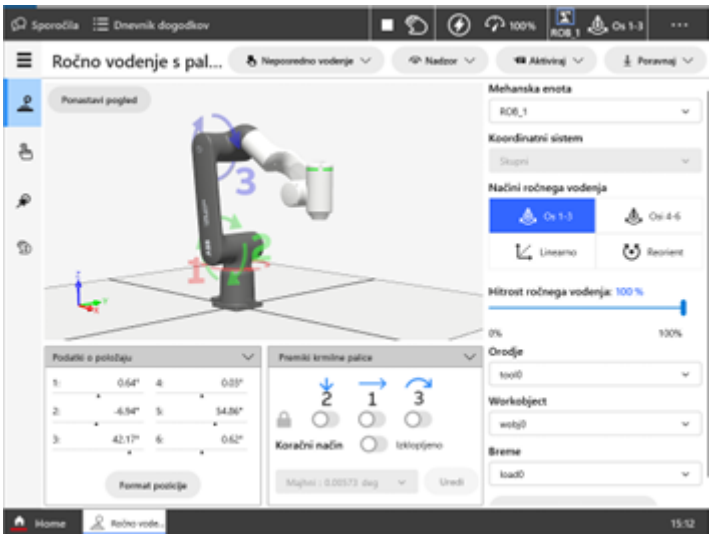


Če želite ročno voditi robota, boste morali na home zaslonu pritisniti zavihek Ročno vodenje.

Pri tem načinu operater premika posamezne osi robota (J1–J6). Robot ima šest rotacijskih osi, ki omogočajo gibanje celotne robotske roke. Na učnem panelu je mogoče izbrati premikanje osi **1–3** ali **4–6**, kar omogoča lažje upravljanje posameznih delov robota.

- Osi 1–3 premikajo osnovni del robota (baza, rama in komolec).
- Osi 4–6 upravljajo orientacijo zapestja robota in orodja.

Slika 28: Prikaz načina ročnega vodenja



Na fotografiji je prikazano, v katerem načinu ročnega vodenja je robot nastavljen. Trenutno boste ročno vodili osi od 1-3.

Pri tem načinu se robot premika tako, da se vrti posamezni sklep robota, zato se položaj orodja lahko spreminja po kompleksni poti.

Ta način se najpogosteje uporablja pri:

- začetni nastavitvi robota,
- servisiranju ali diagnostiki,
- učenju posameznih položajev robota.

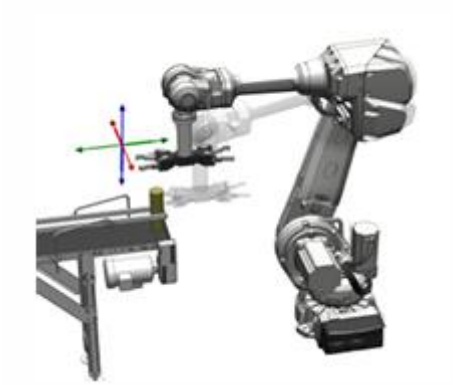
Dokumentacija proizvajalca navaja, da se pri takšnem premikanju robot premika v pozitivni ali negativni smeri posamezne osi, odvisno od izbranega ukaza na učnem panelu.

Pri linearnem načinu se robot premika glede na kartezični koordinatni sistem v smereh:

- X – levo / desno
- Y – naprej / nazaj
- Z – gor / dol

Te smeri veljajo, če operater stoji pred robotom in robota gleda.
Pri tem načinu se TCP (Tool Center Point) premika po ravni liniji v prostoru, medtem ko robot samodejno prilagaja položaj svojih osi, da ohrani linearno pot gibanja.

Slika 29: Prikaz premika po oseh



(ABB, 2024)

Ta način je zelo uporaben pri:

- natančnem določanju točk v prostoru,
- programiranju poti robota,
- premikanju robota okoli delovnega predmeta.

Prednost linearnega gibanja je, da se orodje robota premika po predvidljivi in ravni poti, kar omogoča večjo natančnost programiranja.

Pri orientacijskem načinu robot ne spreminja položaja TCP v prostoru, temveč spreminja orientacijo orodja.

Orodje se lahko vrti okoli treh rotacijskih osi:

- Rx – rotacija orodja okrog X koordinatne osi.
- Ry – rotacija orodja okrog Y koordinatne osi.
- Rz – rotacija orodja okrog Z koordinatne osi.

Pri tem načinu robot:

- nagiba orodje naprej in nazaj,
- nagiba orodje levo in desno,
- vrti orodje okoli lastne osi.

Ta način se uporablja predvsem pri:

- nastavljanju orientacije orodja,
- varjenju,
- lepljenju ali nanašanju materialov,
- pozicioniranju prijemal.

Pri slovensko sodelujočih robotih, kot je ABB GoFa, je mogoče robota premikati tudi tako, da operater fizično prime robotsko roko in jo premakne v želeni položaj. Ta funkcija se imenuje lead-through programming.

Postopek poteka tako, da:

1. operater aktivira funkcijo Lead-through na učnem panelu,
2. robot preklopi v način, kjer so motorji aktivni,
3. operater nežno premakne robotsko roko v želeni položaj.

Robot nato shrani ta položaj kot programsko točko.

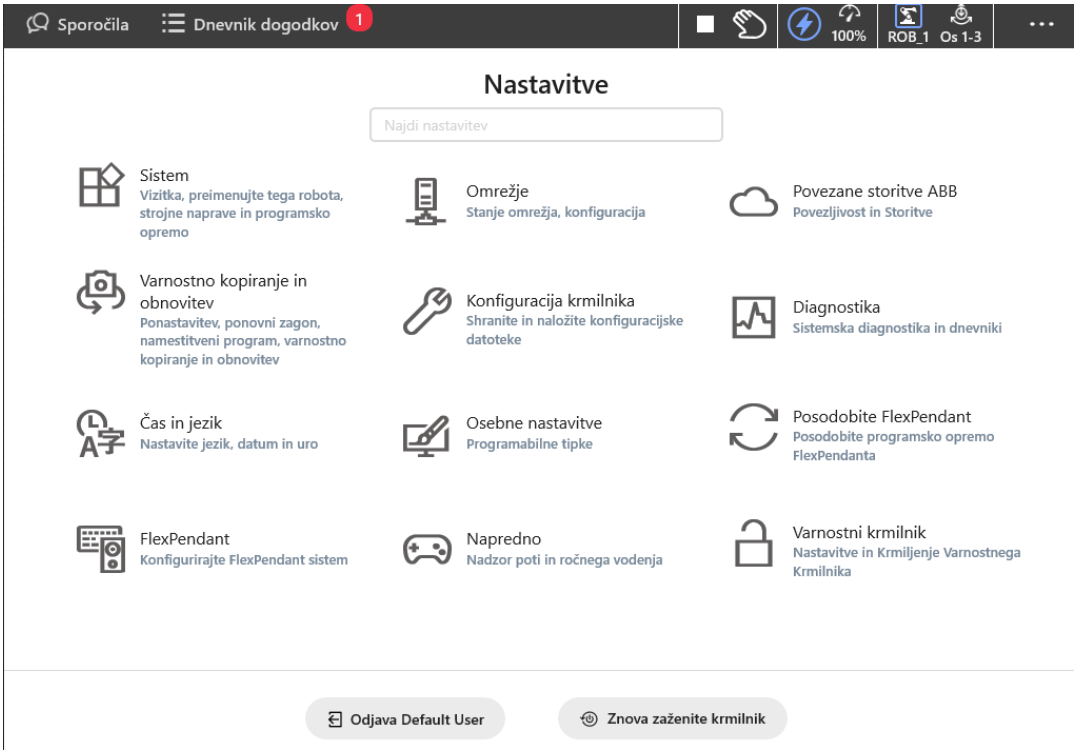
Ta metoda je posebej uporabna pri:

- hitrem učenju poti,
- enostavnem programiranju brez kompleksnih nastavitev,
- slovenskih aplikacijah.

3.3.4 Uporaba prijemala

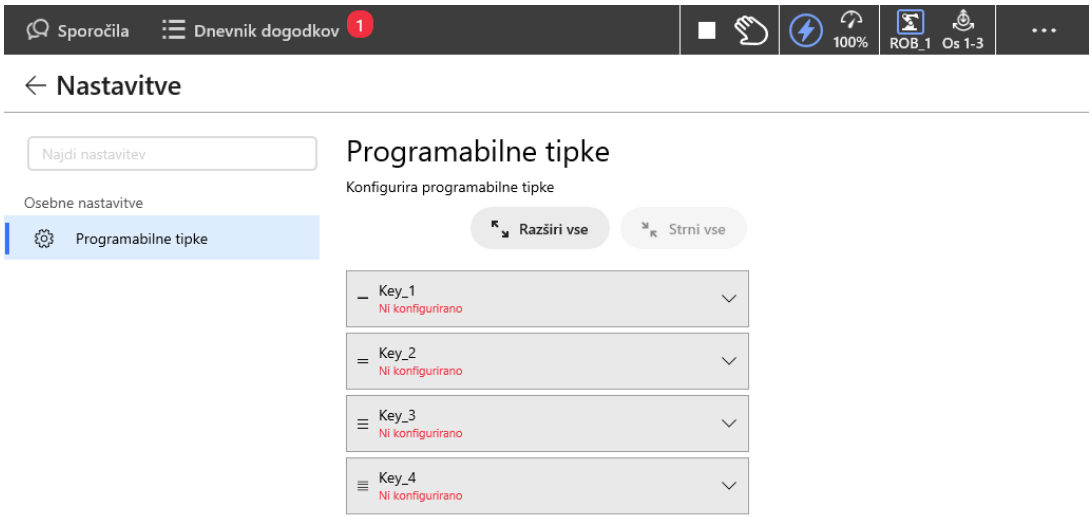
Uporaba prijemala se lahko na robotski roki vklopi ročno ali pa z napisanim programom, kjer robot dobi signal, da se prijemalo zapre ali odpre. Če je potrebno prijemalo ročno vklopiti, sta na učni enoti dve tipki za stisk in izpust prijemala na robotu.

Slika 30: Nastavitve na učni enoti



Na učni enoti je potrebno v glavnem meniju pritisniti tipko Nastavitve. Po pritisku na tipko nastavitve se vam bo odprlo to okence. Nato izberete Osebne nastavitve.

Slika 31: Programiranje tipk



Po pritisku na osebne nastavitve se vam bo odprlo to okence, kjer pa lahko konfigurirate tipke po vaših željah. Imate možnost spremembe imena in ali bo tipka bila uporabljena kot vhodni signal ali izhodni signal.

V pod programu pa je potrebno dati signal na 1 in takrat se doseže, da se prijemalo zapre, če pa želimo, da se prijemalo odpre, pa se da signal nazaj na 0.

Slika 32: openGripper in closeGripper

```
443   PROC Pick_and_place_1()
444       ! Zacetna pozicija
445       MoveAbsJ jt_Home,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_1;
446       ! Premik nad valj 1
447       MoveJ p1_1_Pick_Above,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_1;
448       ! Spust do valja
449       MoveL p1_2_Pick,v1000,fine,t_Prijemalo\WObj:=wobj_plosca_1;
450       ! Zapri prijemalo
451       closeGripper;
452       WaitTime 1;
453       ! Dvig valja
454       MoveL p1_3_Pick_Up,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_1;
455       ! Pomik nad oviro
456       MoveJ p1_4_Over_Obstacle,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_1;
457       ! Pomik nad odlagalno mesto
458       MoveJ p1_5_Place_Above,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_1;
459       ! Spust valja
460       MoveL p1_6_Place,v1000,fine,t_Prijemalo\WObj:=wobj_plosca_1;
461       ! Odpri prijemalo
462       openGripper;
463       WaitTime 1;
464       ! Dvig prijemala
465       MoveL p1_7_Place_Up,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_1;
466       ! Zacetna pozicija
467       MoveAbsJ jt_Home,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_1;
468   ENDPROC
```

V vrstici 451 in 462 sta programa za zapiranje in odpiranje prijemala.

Slika 33: Gumba za ročno odpiranje in zapiranje prijemala

(ABB, 2024)

Na sliki sta prikazani tipki, s katero lahko ročno odpirate in zapirate prijemalo.

Po pritisku levega gumba, ki je obkrožen na sliki, se vam bo prijemalo odprlo, po pritisku desnega gumba, pa se vam bo prijemalo zaprlo.

3.3.5 Osno gibanje

Pomen osnega gibanja je premikanje posameznih osi robota neodvisno od ostalih osi. Robot premika (rotira) samo izbrano os, vse ostale osi so na miru. Pri 6 osnih robotih, kot je GoFa CRB15000, ima robot 6 rotacijskih osi (J1-J6). Te osi skupaj omogočajo popolno orientacijo orodja v prostoru.

Slika 34: Premiki vseh osi

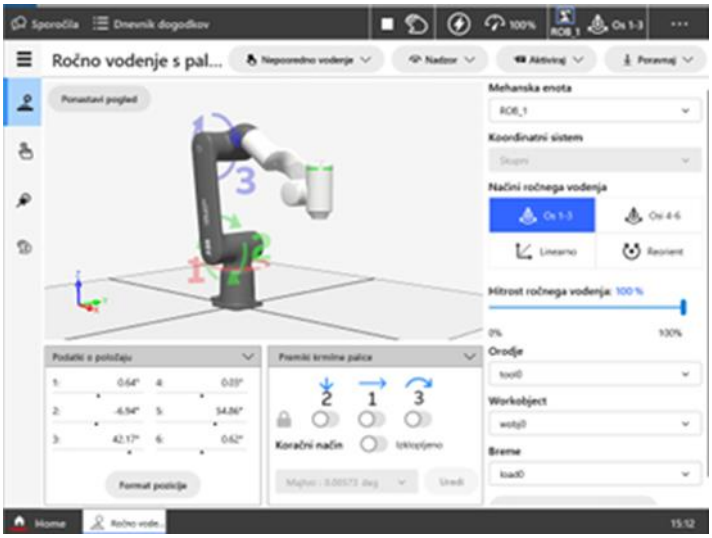
(ABB, 2024)

Osno premikanje v praksi pomeni, da se izbere ena os, premakne se samo ta os in posledično ostale osi mirujejo. To se uporabi predvsem pri ročnem upravljanju, ko se robota nastavlja, ko se ga servisira in ko se na njem izvaja učenje pozicij.

Osi na robotu in njihove funkcije:
OS 1 – funkcija prve osi je, da je to baza robota, robot se na prvi osi vrti celoten v levo ali desno stran.
OS 2 – druga os predstavlja ramo robota. Ta os omogoča dvig ali spust roke naprej ali nazaj.
OS 3 – tretja os je pri tem robotu rama. Rama omogoča, da se roka raztegne ali krči.
OS 4 – zapestje 1. S to osjo dosežemo vrtenje tega zapestja.
OS 5 – zapestje 2. S to osjo dosežemo nagib zapestja.
OS 6 – zapestje 3.S to osjo dosežemo, da se vrti orodje.

Za ročno vodenje robota na učni enoti pojdite v meni ročno vodenje, kjer se vam prikaže robot, kot je na spodnji sliki.

Slika 35: Premikanje osi za ročno vodenje robota



Ko boste prišli v meni za ročno vodenje robota, lahko izbirate, v kakšnem načinu boste vodili robota.

Možnosti za ročno vodenje so:

- Os 1-3
- Os 4-6
- Linearno
- Reorient

Lahko tudi nastavljate hitrost, kako hitro želite, da se bo robot gibal v ročnem načinu.

Slika 36: Ročica za premikanje robota

S premikanjem ročice boste premikali robota v smeri, katero si izberete pri ročnem načinu vodenja robota. Da se bo robot premaknil in da bodo motorji delovali, pa morate držati stikalo DMS.

Slika 37: Prikaz tipke DMS

3.3.6 Kartezično gibanje robota

Kartezično gibanje robota pomeni, da se orodje oz. TCP premika ali rotira glede na kartezični koordinatni sistem (X, Y, Z + rotacije RX, RY, RZ).

Translacije (linearni pomiki)
Po X smeri se orodje oz. TCP premika naprej in nazaj, če se stoji točno pred robotom. Po Y smeri se premika orodje oz. TCP levo in desno. Po Z smeri pa se robot premika gor in dol.

Rotacije (vrtenje orodja):

Rotacije orodja pomeni, da se orodje rotira okoli X, Y ali Z osi.
Rotacija RX – tukaj se orodje rotira okoli X osi (nagib naprej/nazaj).
Rotacija RY – tukaj se orodje rotira okoli Y osi (nagib levo/desno).
Rotacija RZ – tukaj se orodje rotira okoli Z osi (zasuk orodja).
TCP – Tool Center Point.

TCP je virtualna točka na orodju, je referenčna točka za gibanje robota ter točka, ki jo robot premika v prostoru. V primeru da TCP ne bi bil pravilno nastavljen, bi povzročilo napačno gibanje, poti ne bodo natančne ker lahko pri translaciji in hkrati rotaciji orodja pride do trka orodja z drugimi komponentami.

S kartezičnim gibanjem so prednosti, da je to naravno za operaterja, natančno sledenje poti, primerno je tudi za procesna gibanja in omogoča konstantno orientacijo orodja. Najpogosteje se kartezično gibanje uporablja pri pobiranju in odlaganju, lepljenju, varjenju, vstavljanju komponent, poliranju in manipulacija materiala.

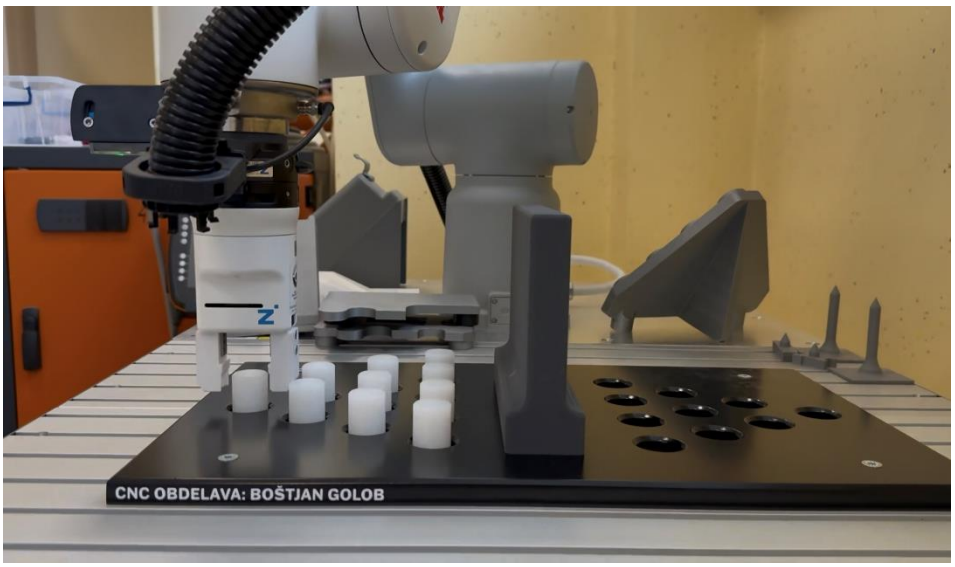
Razlika med translacijo in rotacijo je ta, da s translacijo dobimo pomik po prostoru, položaj se spremeni in orientacija ostane enaka. Pri rotaciji pa dosežemo vrtenje na mestu, položaj TCP ostane in orientacija orodja se spremeni.

TCP je pa pri različnih orodjih drugačen, tukaj imate postopek, kako nastavite TCP. Oz. KS orodja, tool …

Z manjšo hitrostjo tudi dosežemo to, da ko testiramo robota in program, če vidimo napačne premike ali možnost, da se bo robot zaletel, ga lahko dovolj hitro ustavimo. Robot se lahko v auto načinu ustavi tako, da se pritisne močno DMS, da se spusti DMS ali pa da se pritisne E-STOP.

Navodila za ročno vodenje robota in premik valja iz ene strani na drugo stran:

Slika 38: Približanje do prvega valja v ročnem načinu



Ko boste v meniju za ročno vodenje robota, izberite Linear premik robotske roke. Z držanjem DMS stikala se vam bodo zagnali motorji in nato z ročico premaknite robota v položaj, da boste imeli valj v sredini prijemala orodja.

Slika 39: Tipka za prijem prijemala

Po pritisku na levo tipko se vam bo prijemalo zaprlo.

Slika 40: Prijemalo zagrabi valj

Po zaprtju prijemala lahko z ročico dvignete robotsko roko in jo ročno odpeljete na drugo stran plošče preko ovire, ki je na sredini.

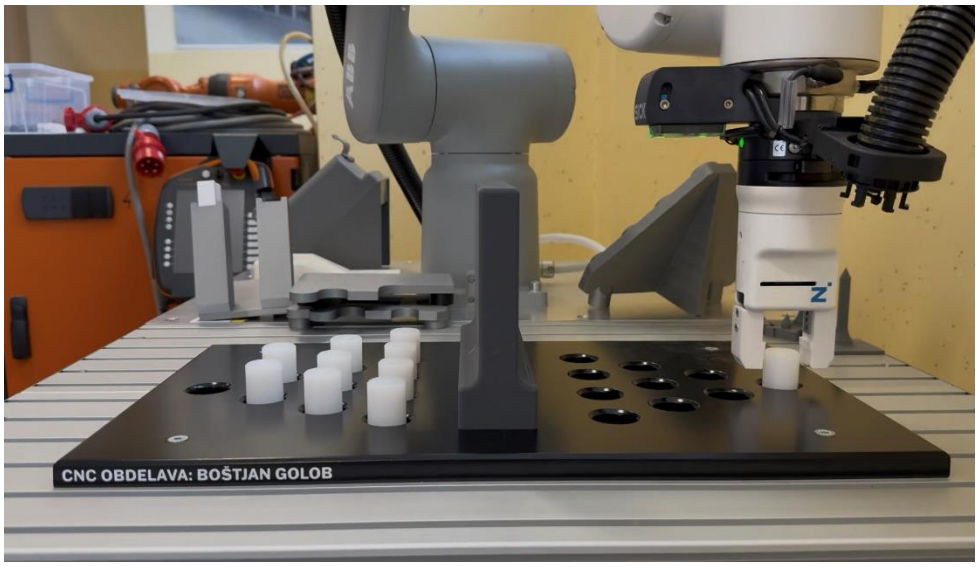
Slika 41: Prenos valja v ročnem načinu vodenja v drugo odprtino preko ovire

Preko ovire poljubno odložite valj.

Slika 42: Pritisk leve tipke, ki je obkrožena, da se prijemalo odpre

Po pritisku na desno tipko, ki je obkrožena, se vam bo odprlo prijemalo.

Slika 43: Odprto prijemalo



Ko boste pritisnili na desno tipko, se vam bo odprlo prijemalo in takrat lahko dvignete ponovno robotsko roko in jo premaknete do naslednjega valja, ki ga želite pobrati.

3.3.7 Tool Data

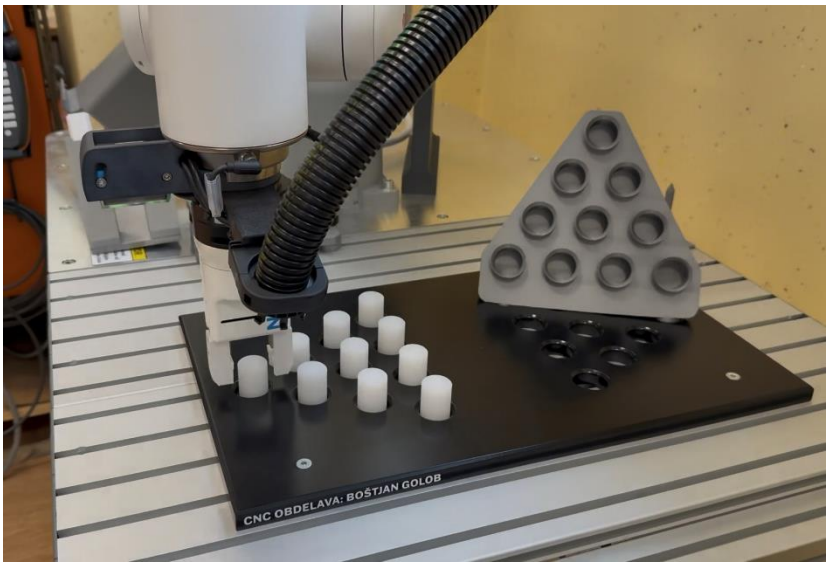
Tool Frame je koordinatni sistem orodja, ki definira, kje je koordinatno izhodišče (TCP) in kakšne so smeri orodja X, Y in Z ter kako je orodje orientirano glede na prirobnico robota.

Tool Frame je pomemben, da je pravilno nastavljen, če pa ni, so lahko poti napačne, slabo vstavljanje kosov in možne kolizije.

Navodila za ročno vodenje robotske roke in pobiranja valja na eni strani in prenos valja na poševnino na drugi strani:

Na začetku premaknite robotsko roko nad valj.

Slika 44: Prikaz pozicije robota za pobiranje valja

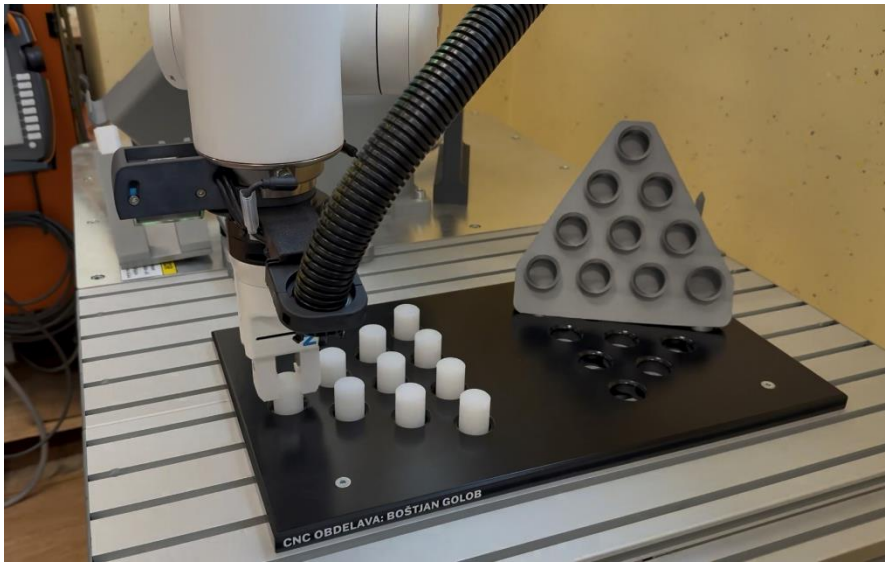


Po premiku robotske roke nad valj, ki ga želite prenesti, pritisnete levo tipko, kot je prikazano na sliki spodaj.

Slika 45: Tipka za zapiranje prijemala

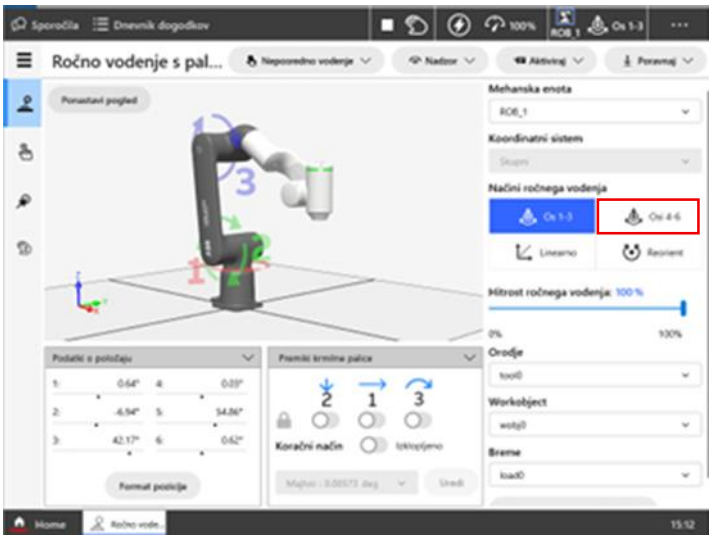
Po zaprtju prijemala lahko premaknete robotsko roko proti poševnini na drugi strani.

Slika 46: Pobiranje valja



Ker se mora robotska roka zasukati, namesto linearnih premikov uporabite drugačen način vodenja.

Slika 47: Izbire drugačnega načina vodenja robotske roke

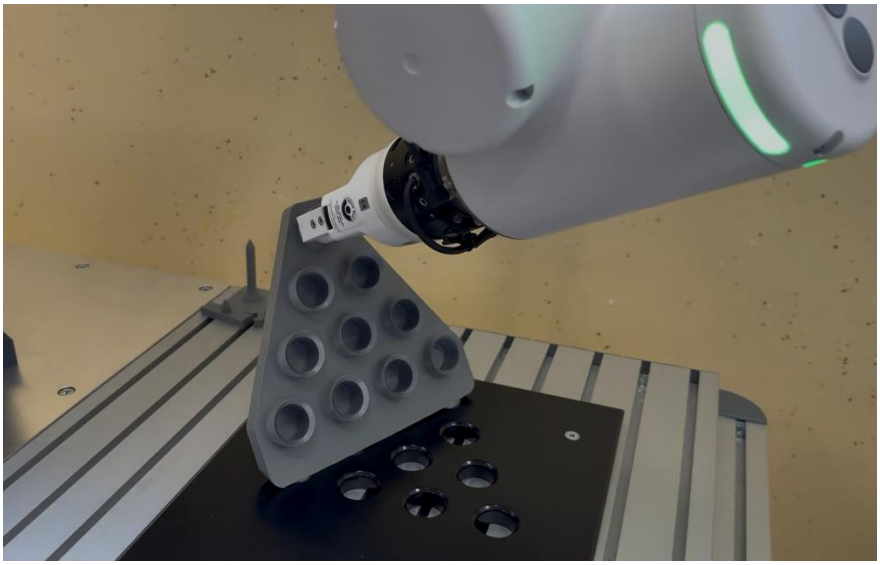


(ABB, 2024)

Uporabite način Os 4-6. S tem boste dosegli rotacijo robotske roke v smer, ki vam ustreza. Lahko tudi poskusite z načinom Reorient.

Po premiku robotske roke v pozicijo, ki vam bo ustrezala, lahko ponovno daste v način linearnega ročnega vodenja in premaknete robotsko roko do poševnine in do tiste luknje, kamor želite odložiti vaš valj.

Slika 48: Odlaganje valja na poševnino



Ko odložite valj natančno v odprtino na poševnini, lahko s pritiskom na desno tipko na učni enoti odprete prijemalo.

Slika 49: Pritisk na desno tipko, ki je obkrožena, da se prijemalo odpre

Ko boste imeli odprto prijemalo, lahko umaknete robota iz položaja za odlaganje valja in nato nadaljujete na enak način s pobiranjem ostalih valjev.

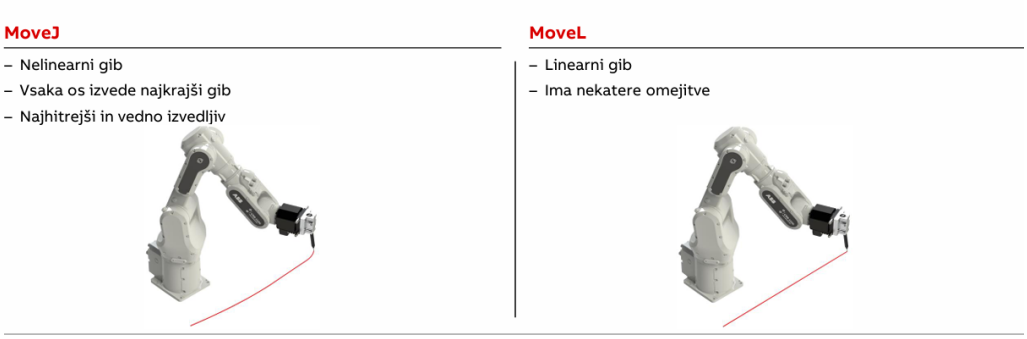
3.4 VAJA 2: POBIRANJE IN ODLAGANJE ENEGA VALJA PREKO OVIRE

Ustvarite program, kjer bo robotska roka iz ene strani prenesla valj preko ovire na drugo stran in se na to vrnila v začetni položaj. Robotska roka ima začetno pozicijo takrat, ko so vse osi postavljene na 0°. Vsebovati mora točko nad valjem, točko pobiranja, točko, ko se dvigne z valjem, točko nad oviro, točko, kjer se robot približa točki, kjer bo odložil valj, točko odlaganja, točko vračanja in začetno točko. V programu je še potrebno dodati zapiranje in odpiranje prijemala.

Po koncu je potrebno vajo testirati koračno, kontinuirano in nato v avtomatskem načinu. Ta način testiranja je potreben pri vseh vajah.

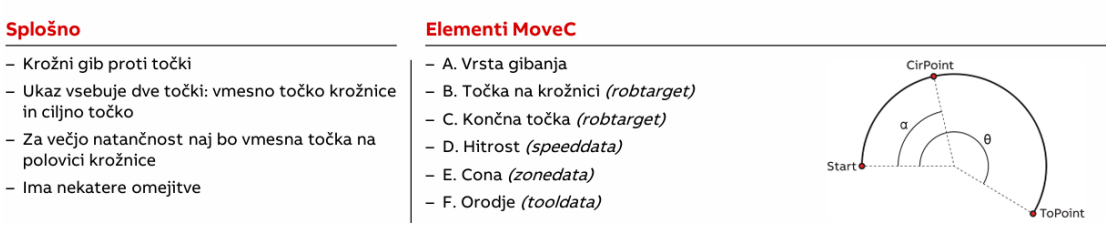
Pred začetkom programiranja je potrebno vedeti, kakšne gibe je potrebno imeti za premik robota.

Slika 50: MoveJ in Move L



(ABB, 2024)

Slika 51: MoveC

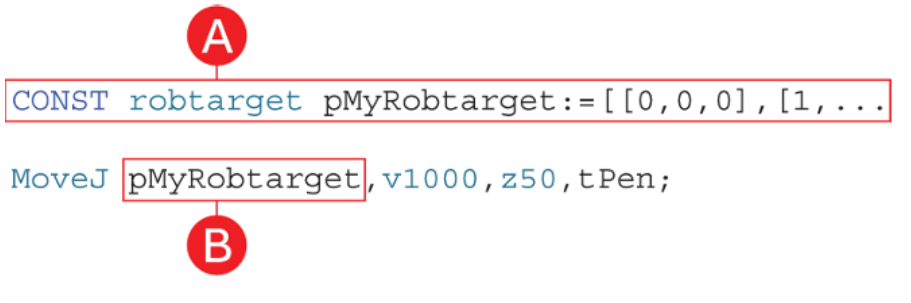


(ABB, 2024)

Za programiranje na TP je potrebno vedeti nekatere osnove, kaj nekatere stvari pomenijo. Potrebno je vedeti, kakšen gib želimo, da robot izvede, ko bomo napisali vrstico.
Move L – je linearni gib, uporablja se predvsem v trenutkih, ko gre robot ravno navzgor, navzdol, levo ali desno.
Move J – je osni gib gib, katerega lahko uporabimo pri gibih, kjer ga z linearnim gibom ne bi mogel doseči. To se predvsem uporablja v delih programa, kjer se na robotu uporablja rotacija osi.

Move C – je kartezični gib, uporablja se predvsem v delih, kjer mora robot uporabiti krožni premik. V vrstici s kartezičnim premikom uporabimo dve točki.
Ukaz za gibanje:

Slika 52: Ukaz za gibanje in deklaracija točke



(ABB, 2024)

Podatki so informacije, ki so shranjene v pomnilniku, ki pa jih poiščete po imenu.
Podatkovna deklaracija navaja ime, vrsto in začetno vrednost podatkov.
Vsako sklicevanje na podatke bo priklicalo povezano vrednost shranjeno v pomnilniku.
A – podatkovna deklaracija
B – referenca podatkov
Nekateri podatki, ki so pomembni v vrstici, so opredeljeni že v naprej.
Speeddata (v100, v500, v1000 itd.)
Tooldata (tool0)
Zonedata (fine, z1, z50)
Kot uporabnik pa lahko spreminjate določene podatke.
Robtargets (položaj za robota)
Numerični podatki (spremljanje števila izdelanih delov itd.)
Speeddata

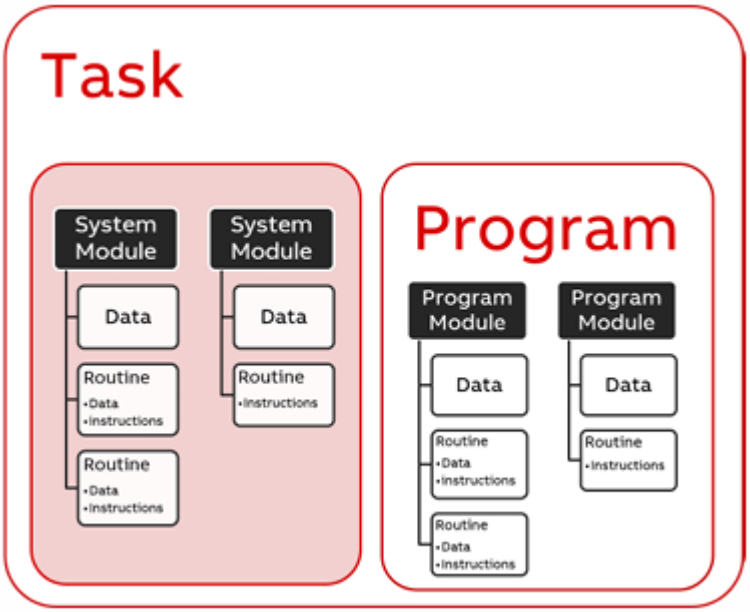
Ukazi so navodila, ki jih lahko podate krmilniku. Primer teh so:

- MoveJ
- MoveL
- Set
- WaitDI
- Stop
- + še mnogi drugi

Da ustvarite nov program, je potrebno ustvariti nov programski modul in nov sistemski modul. Premakniti je potrebno rutine v nov modul. Svojim rutinam dodajte priklice postopkov iz main. Poskusno lahko zaženete program iz main. Potrebno je nastaviti PK na main. PK pomeni klic procedure.

Struktura programa je sestavljena iz modula, programa, podatkov, rutine ter ukazov.

Slika 53: Struktura programa



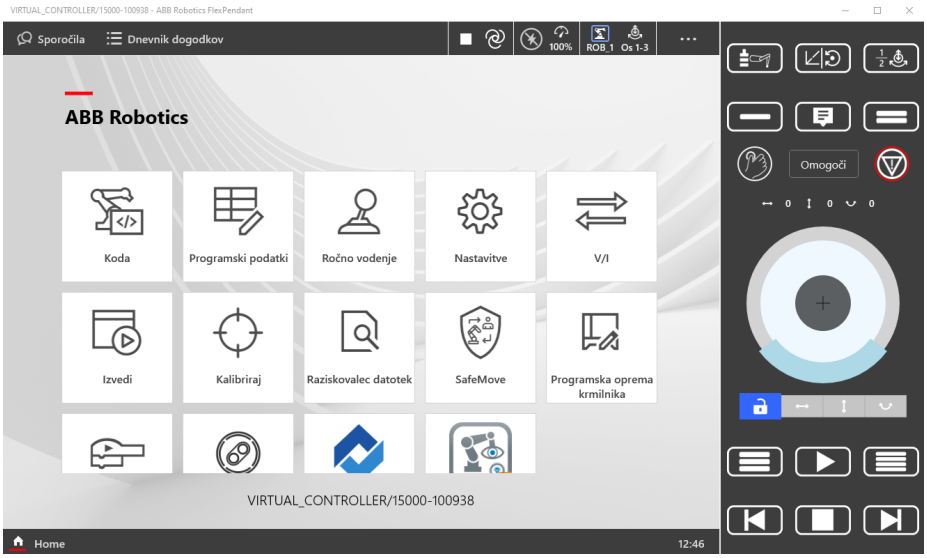
(ABB, 2024)

Pojasnila besed na sliki;
Task → *Naloga*
System Module → *Sistemski modul*
Program → *Program*
Program Module → *Programski modul*
Data → *Podatki*
Routine → *Rutina* (ali *postopek*)
Instructions → *Navodila*
Moduli: Besedilne datoteke za strukturiranje podatkov in rutin.
Program: Vsi naloženi moduli programa.
Podatki: Podatkovne deklaracije, npr. položaji, orodja in delovni objekt.
Rutine: Nabori ukazov.
Ukazi: Ukazi za krmilnik, npr. MoveJ/MoveL/MoveC.
Moduli so besedilne datoteke, ki lahko vsebujejo podatke in/ali rutine (nabore ukazov).
– Moduli se uporabljajo za strukturiranje vašega programa.
– Različna podjetja uporabljajo različne metode strukturiranja.
– Število uporabljenih modulov ni omejeno.

Postopek, kako se naredi modul in vrstica:

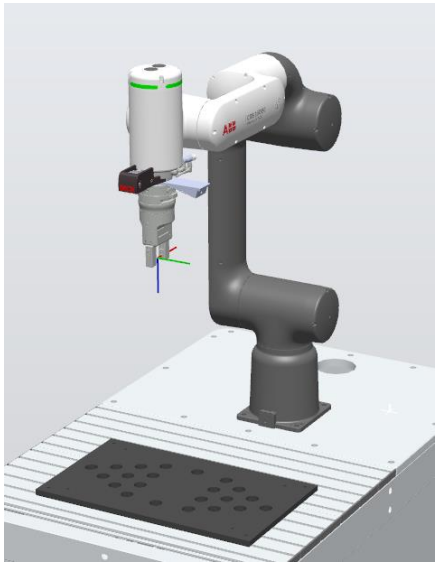
1. Prikaz domačega zaslona na krmilniku

Slika 54: Prikaz domačega zaslona na učnem panelu



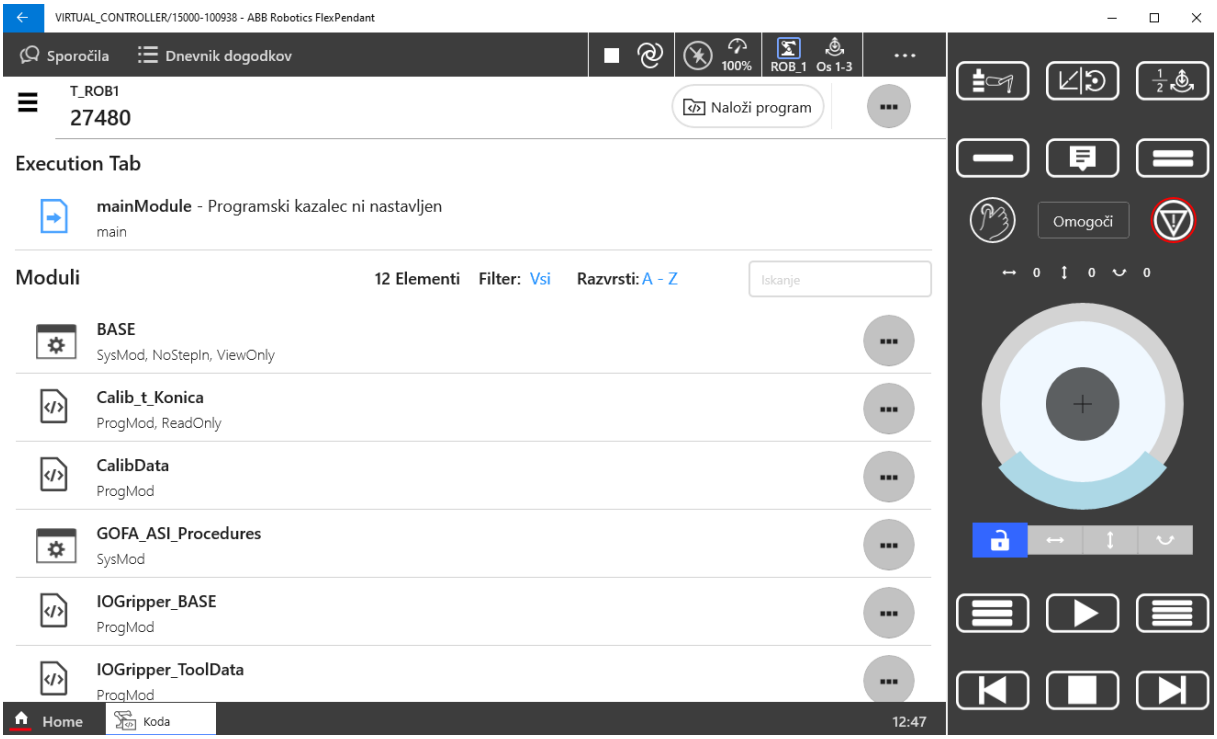
2. Pritisnite na zavihek Koda

Slika 55: Pozicija robota



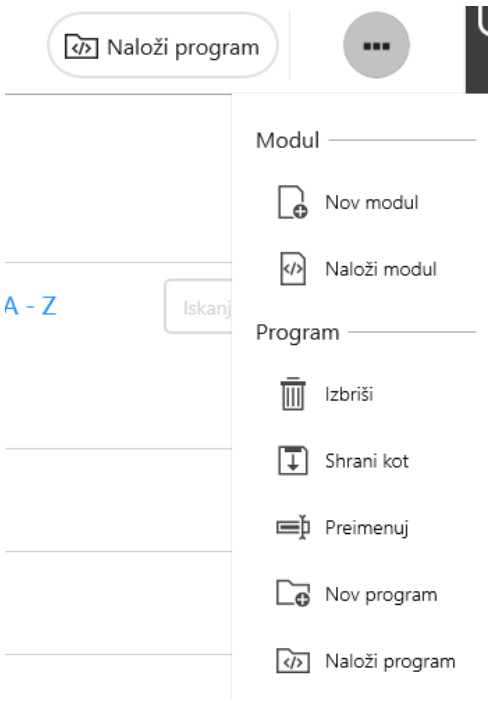
3. Prikazana je pozicija robota, v kateri imejte postavljenega robota. To pozicijo boste dosegli tako, da premaknete 5 os za -90°. Ostale osi je potrebno imeti na 0°

Slika 56: Prikaz zavihka Koda



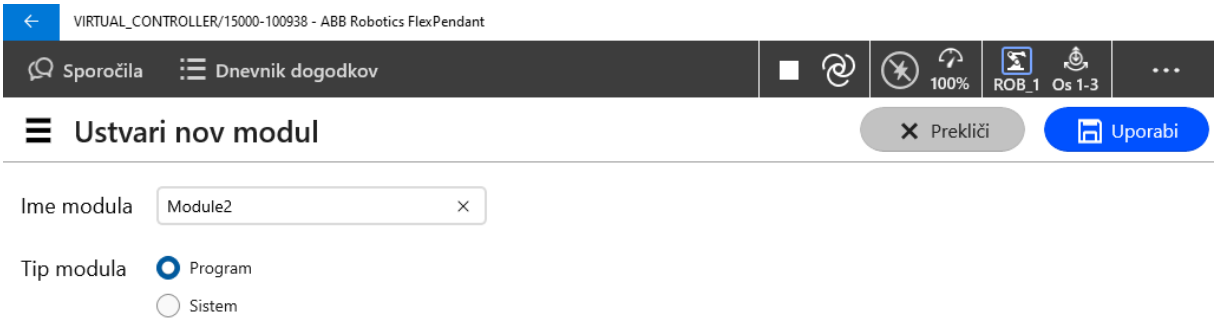
4. Po pritisku na gumb Koda lahko pritisnete tri pikice v zgornjem desnem kotu,

Slika 57: Prikaz po pritisnjenih treh pikah



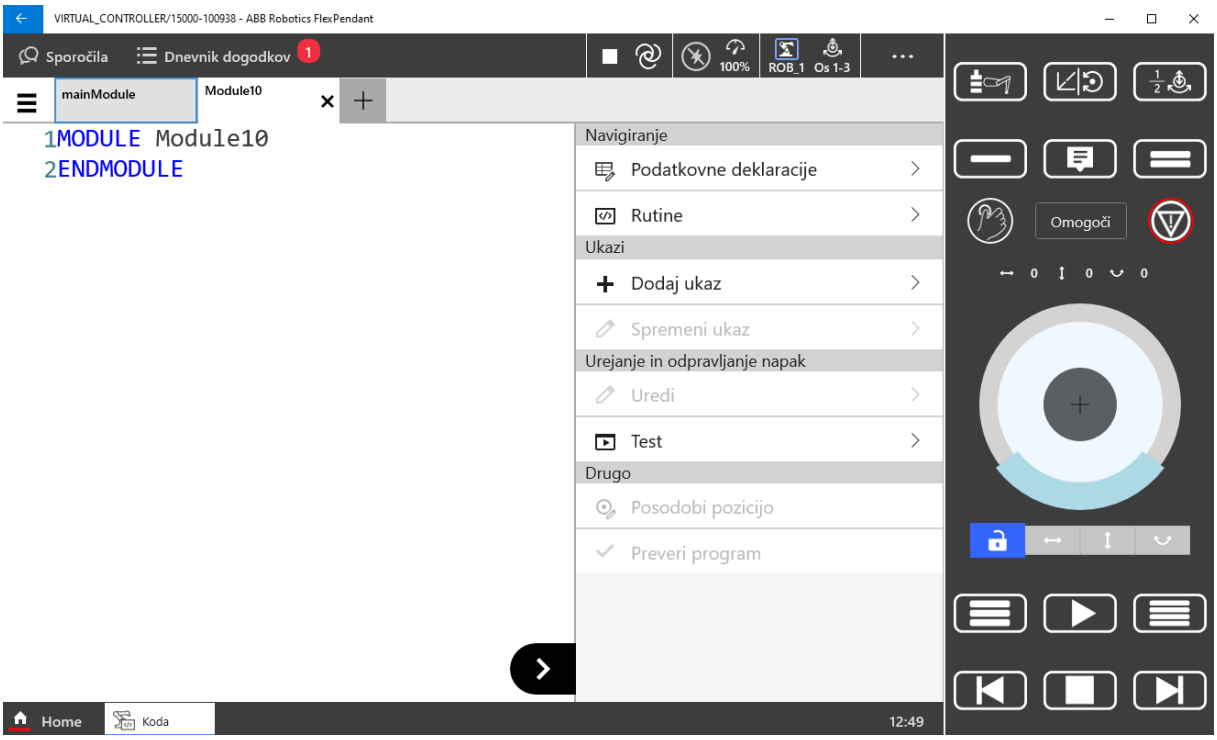
5. Ko pritisnete tri pikice, pritisnete kasneje zavihek Nov modul.

Slika 58: Prikaz, kako se ustvari nov modul



6. Po pritisku na Nov modul se vam odpre ta zavihek, kjer pa napišete ime modula in tip modula ali boste imeli program ali sistem. V vašem primeru pustite program in pritisnite na tipko zgoraj desno Uporabi.

Slika 59: Prikaz, ko se ustvari nov modul

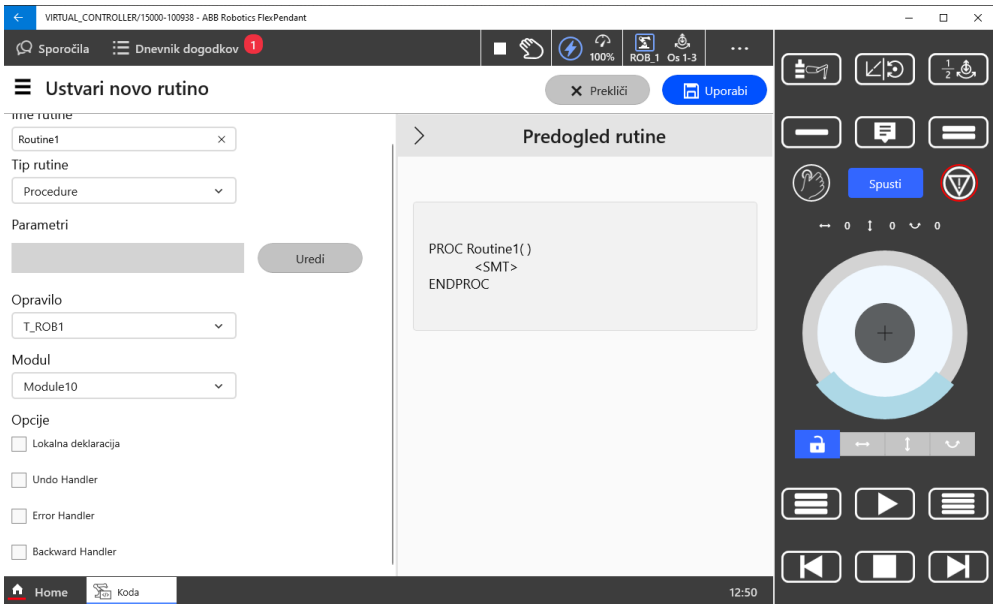


7. Po pritisku gumba Uporabi se vam bo odprlo to okence. Ustvariti morate novo rutino.

Slika 60: Prikaz nove rutine

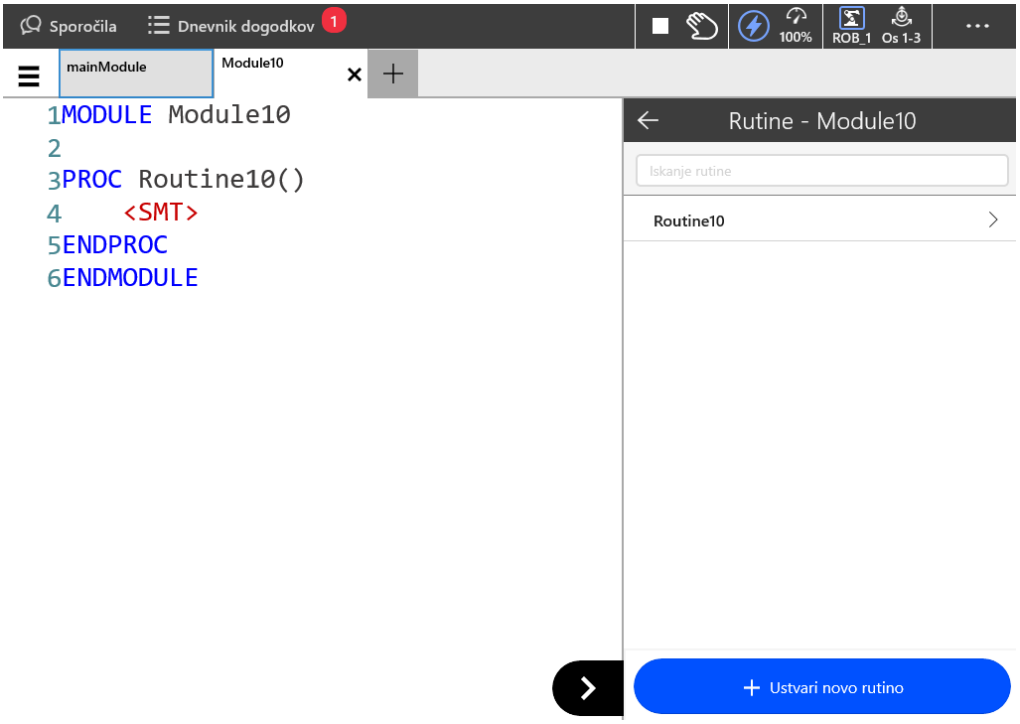
8. Tukaj pritisnete Ustvari novo rutino.

Slika 61: Prikaz, kako se ustvari nova rutina



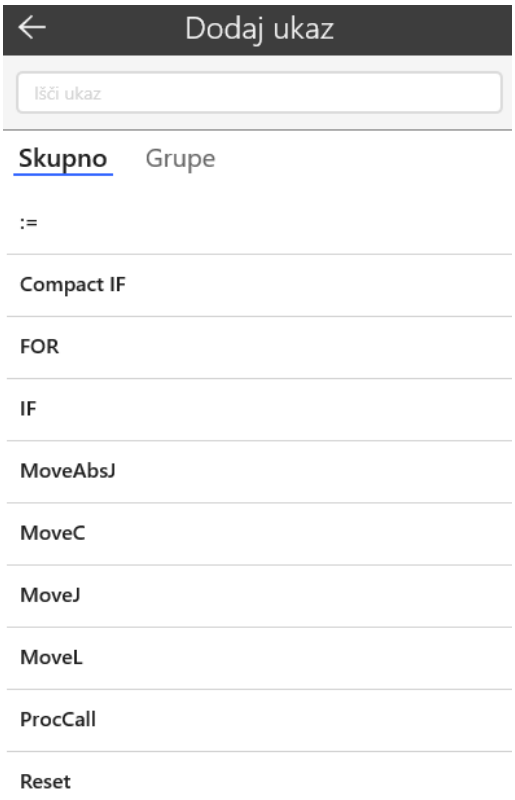
9. To se vam bo prikazalo ob pritisku na Ustvari novo rutino. Tukaj je potrebno napisati ime vaše rutine, tip rutine, opravilo ter modul, v katerem boste imeli to rutino, torej v vašem primeru tisto rutino, ki ste jo prej ustvarili. Nato pa pritisnite gumb uporabi, ki je zgoraj desno.

Slika 62: Prazen program, ki prikazuje uporabljen modul in rutino



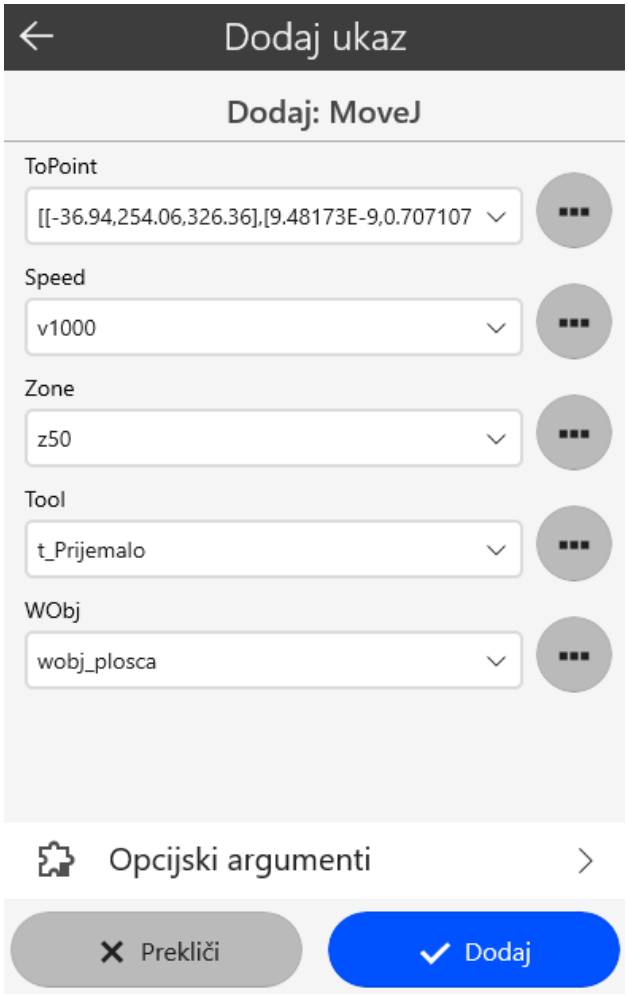
10. Ko boste uporabili rutino, ki ste jo prej ustvarili, se vam bo v vašem programu prikazalo vaša procedura.

Slika 63: Prikaz, kako se doda ukaz



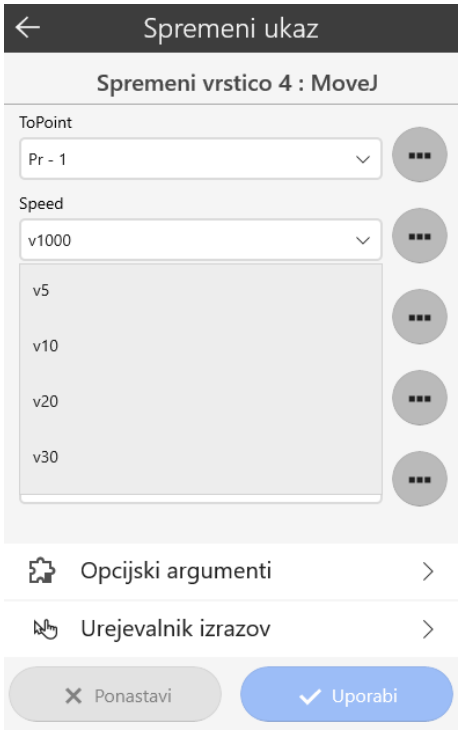
11. Sedaj je naslednji korak, da v zavihku Ukazi dodate nov ukaz. Tukaj lahko izberete, s katerim premikom želite, da se bo robot premaknil v točko, ki jo boste imeli označeno.

Slika 64: Dodajanje ukaza



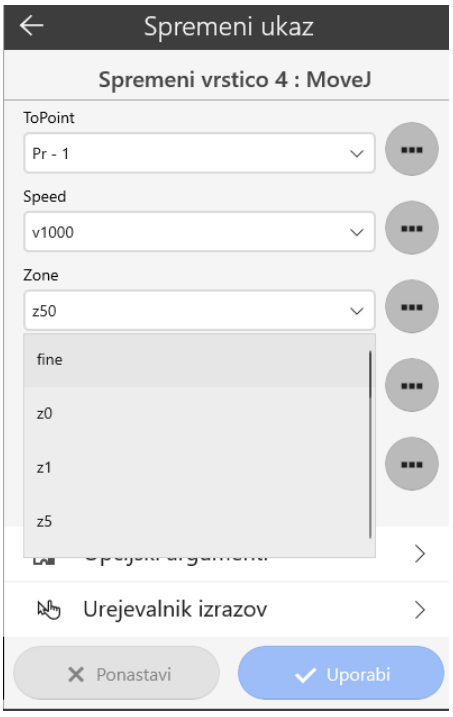
12. Za primer je bil izbran MoveJ. ToPoint pomeni, kam se bo vaš robot premaknil točno v katero točko. Na začetku boste imeli prikazane te številke, vendar se te številke da spremeniti in zapisati z besedo, tako boste lahko točno vedeli, v kateri točki ste.

Slika 65: Prikaz, kako se lahko spremeni hitrost



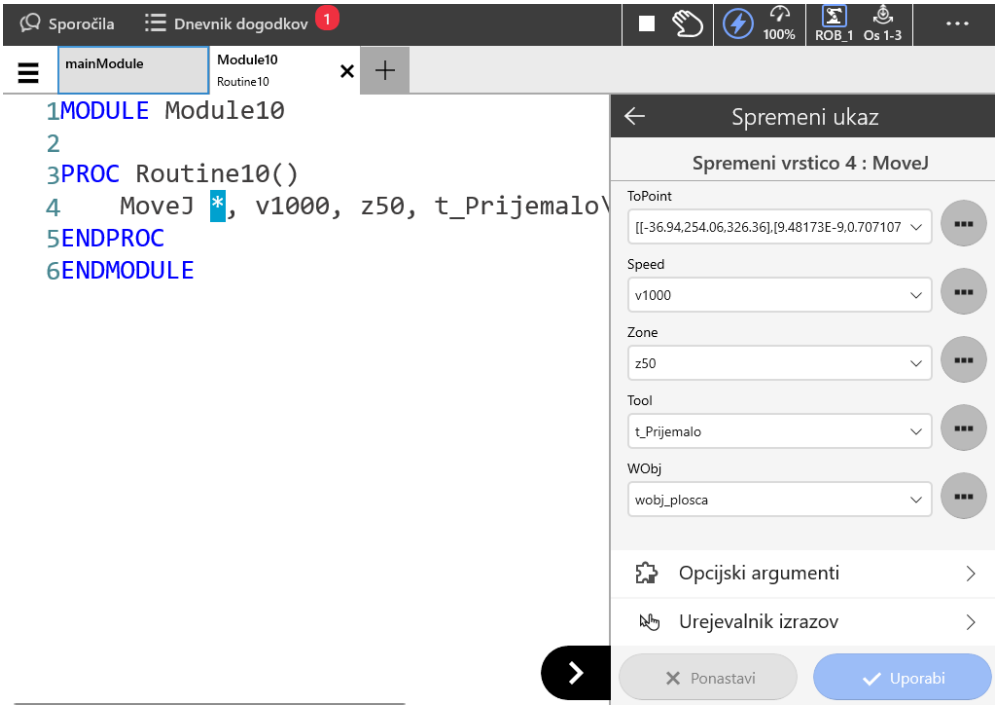
13. Tukaj lahko spremenite vašo hitrost, s katero se bo robotska roka premikala.

Slika 66: Prikaz, kako se lahko spremeni zone



14. Tukaj lahko tudi spremenite, ali se bo robot v tej točki ustavil, ali pa bo šel skozi njo.

Slika 67: Vrstica je zapisana v programu

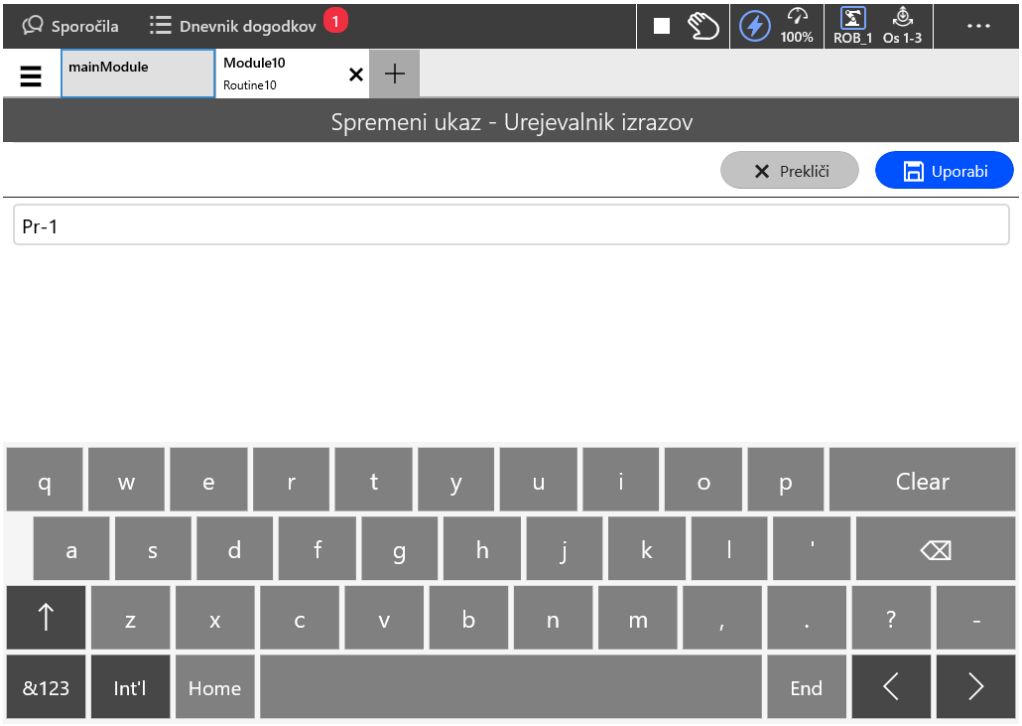


15. Po pritisku gumba Uporabi boste imeli vašo vrstico vpisano v vaš glavni program. Če se želite rešiti te zvezdice, je potrebno pritisniti na njo in izbrati zavihek Urejevalnik izrazov.

Slika 68: Sprememba imena točke v vrstici

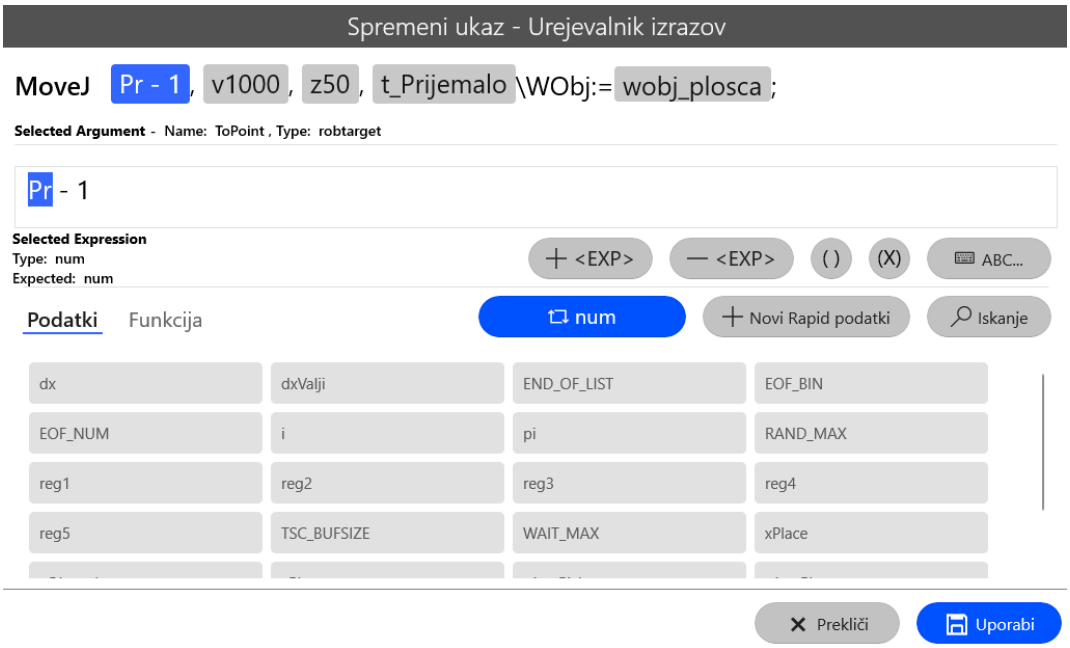
16. Takrat se vam bo odprlo to okence, kjer pa lahko poljubno vpišete, kakšno ime bi želeli imeti za to točko.

Slika 69: Potrditev spremembe imena v točki



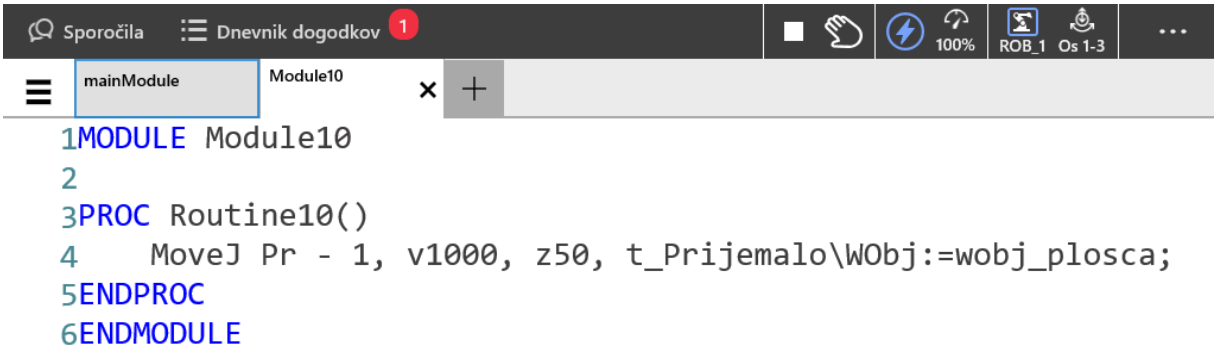
17. Primer, kako lahko zapišete vašo točko, vendar bo enostavnejše, če si točko zapišete na čim lažji način in da uporabite nekakšen sistem, ki vam bo logičen.

Slika 70: Uporabljeno spremenjeno ime v vrstici



18. Po pritisku na Uporabi vam bo tiste številke spremenilo v to, kar ste napisali, nato pa še lahko ponovno pritisnete gumb Uporabi in se vam bo shranila točka tako, kot si jo želite imeti.

Slika 71: Vrstica napisana in dodelana



19. Tako bo na koncu izgledala vaša vrstica, ki ste jo zapisali.

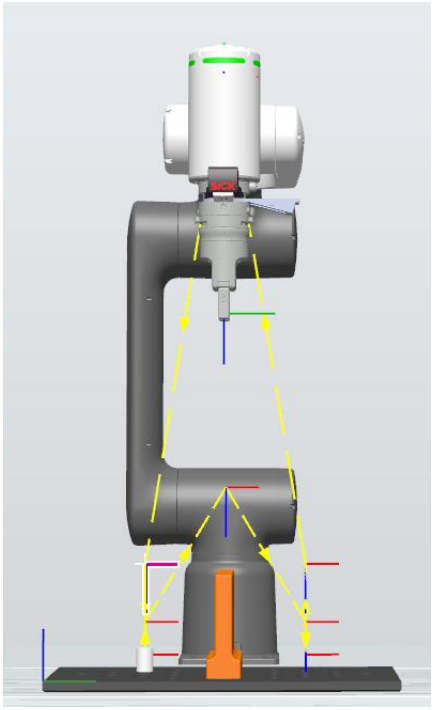
Po zapisu te vrstice lahko nadaljujete z istim načinom, kot pa ste zapisali to prvo vrstico.

Slika 72: Program za prenos enega valja

```
443  PROC Pick_and_place_1()
444      ! Zacetna pozicija
445      MoveAbsJ jt_Home,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_1;
446      ! Premik nad valj 1
447      MoveJ p1_1_Pick_Above,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_1;
448      ! Spust do valja
449      MoveL p1_2_Pick,v1000,fine,t_Prijemalo\WObj:=wobj_plosca_1;
450      ! Zapri prijemalo
451      closeGripper;
452      WaitTime 1;
453      ! Dvig valja
454      MoveL p1_3_Pick_Up,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_1;
455      ! Pomik nad oviro
456      MoveJ p1_4_Over_Obstacle,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_1;
457      ! Pomik nad odlagalno mesto
458      MoveJ p1_5_Place_Above,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_1;
459      ! Spust valja
460      MoveL p1_6_Place,v1000,fine,t_Prijemalo\WObj:=wobj_plosca_1;
461      ! Odpri prijemalo
462      openGripper;
463      WaitTime 1;
464      ! Dvig prijemala
465      MoveL p1_7_Place_Up,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_1;
466      ! Zacetna pozicija
467      MoveAbsJ jt_Home,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_1;
468  ENDPROC
```

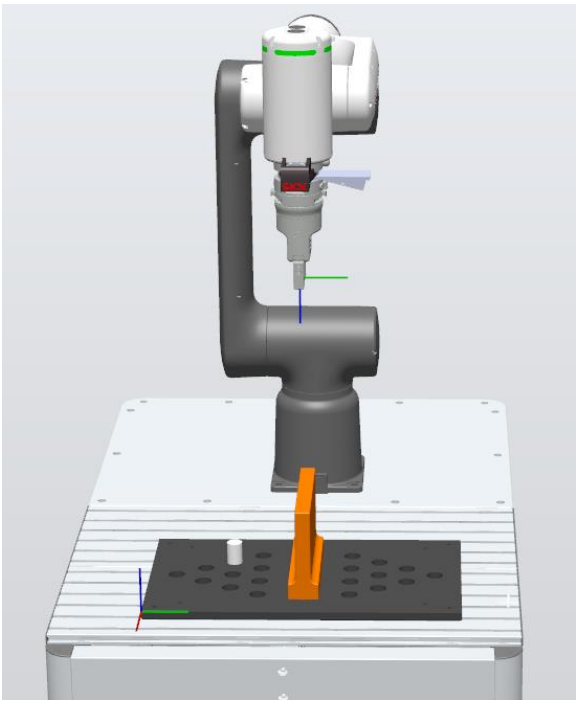
Vrstica 443 pove, da se kliče program Pick_and_place_1.
Vsaka vrstica, ki ima v ospredju ! pomeni, da je ta vrstica komentar in po tem lahko vidimo, kaj pomeni naslednja vrstica.

Slika 73: Pot robota v programu



Vrstica 445 pove, da je robot v HOME točki, ki je bila sprogramirana tako, da robot ve, kje mu je domača točka, kje začne in tudi, kam se robot vrne. V tej vrstici nam v1000 pove hitrost robota, z100 nam pove, da se robot ne bo ustavil v tej poziciji, saj bi v tem primeru namesto z100 moralo pisati FINE. Kljub temu, da ni zapisan FINE, se robot vrne v prvotno pozicijo oz. štarta iz te pozicije kot je razvidno v programu.

Slika 74: Home pozicija robota



Vrstica 447 pove, da je premik robota v Joint gibu, to se lahko razbere iz MoveJ. Točka, ki je poimenovana p1_2_Pick_Above, nam pove, da je to druga točka v programu in se robot premakne v to pozicijo, ki je nad valjem. Do te točke se premakne s hitrostjo v1000, v tej točki se ne ustavi, saj ima z100. Robot gre blizu te točke, vendar naredi premik skozi njo v drugo točko. Je pa razvidno iz vrste, v katerem work object-u deluje robot in to je wobj_plosca_1. Razvidno je tudi, da se v tem programu uporablja prijemalo.

Vrstica 449 pove, da se robot premakne v Linear gibu, to lahko razberemo iz MoveL. Ta točka, poimenovana p1_2_Pick, nam pove, da se tukaj pobere valj. Hitrost robota je še vedno v1000, vendar pa je tukaj za razliko od prejšnjih dveh točk zapisan fine. Tukaj pa se robot mora ustaviti. Vrstica 451 kliče pod program closeGripper. V tem podprogramu so točke, ki robotu povejo, da se mora prijemalo zapreti. To se doseže tako, da se signal prestavi na 1, kar posledično zapre prijemalo (dodaj sliko podprograma za lažjo razlago). Ko se podprogram zaključi, robot nadaljuje njegovo pot.

Vrstica 452 nam pove, da bo robot po prijemu valja moral malo počakati. To dosežemo tako, da se zapiše v program WaitTime 1. To nam pove, da bo robot po zaprtju prijemala počakal 1 sekundo, nato bo nadaljeval s premikom.

Vrstica 454 pove, da se robot premakne v točko p1_3_Pick_Up. Hitrost do te točke je v1000, v tej točki se tudi uporabi z100, saj ni nobene potrebe, da bi se robot v tej točki ustavil in s tem posledično bi program trajal dosti dlje. Ker je gib navpično navzgor, je bil uporabljen Linear gib.

Vrstica 456 pove, da se robot premakne v točko p1_4_Over_Obstacle s hitrostjo v1000 ter z100, saj je nepotrebno robota ustavljati v tej točki. Robot s tem, da je zapis z100, rahlo zaobide to točko, se ji približa, vendar ne gre direktno skozi njo. To bi dosegli tako, če bi dali Z1, takrat bi robot šel zelo blizu točki in bi se že tudi skoraj ustavil. Ker ta gib ni raven in mora robot narediti določeno pot, se uporabi Joint gib, saj je Joint gib dosti bolj primeren za vse gibe razen takrat, ko se robot premakne ravno v določeno točko.

Vrstica 458 pove, da se robot premakne v točko p1_5_Place_Above s hitrostjo v1000 ter z100. Robot gre skozi to točko do naslednje točke. Je pa ta točka Joint gib, saj gib robota in njegova pot nista naravnost.

Vrstica 460 nam pove, da bo robot odložil valj, robot se bo v točki p1_6_Place ustavil, saj ima zahtevo fine. Se bo pa robot v to točko premaknil s hitrostjo v1000. Robot bo opravil linearni gib, saj se bo iz prejšnje točke spustil ravno navzdol.

Vrstica 462 je podprogram openGripper s, katerim prijemalu povemo, da se odpre, to dosežemo tako, da pošljemo signal 0 in se bo vrnil v prvotni odprt položaj. Ko robot odpre prijemalo, bo zaradi WaitTime počakal 1 sekundo preden bo šel naprej po programu.

Vrstica 465 p1_7_Place_Up pove, da se robot premakne v to pozicijo, premik je narejen z Linear gibom s hitrostjo v1000 in z100, kar pomeni, da se robot iz prejšnje v to točko dvigne in gre skozi njo, v njej se ne ustavi in zato tudi ne upočasnjuje postopka gibanja robota.

Vrstica 467 nam pove, da se robot v Joint gibu vrne v jt_Home točko. Vrne se s hitrostjo v1000 in če tudi ima z100 in se ne konča s fine, bo robot v tej točki počakal, saj nato sledi ENDPROC, kar naznanja konec tega programa, ki ga je robot naredil.

3.4.1 Kako se naredi vrstica v programu

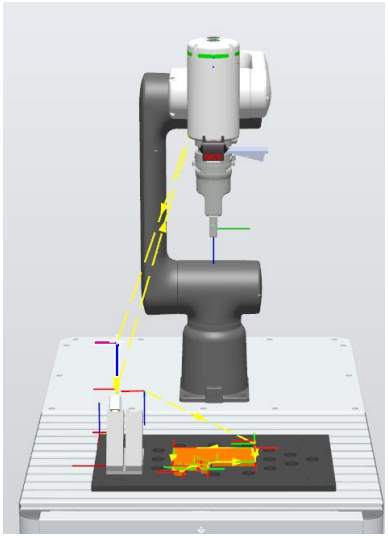
Hitrost robota se označuje s črko v. V praksi je lažje videti, kakšno hitrost želimo, saj se pri robotu, ko nima v prijemalu ničesar ali pa ko ne vari tako kot v našem primeru, lahko uporabi višja hitrost, da se program hitreje izvede. Ko pa ima robot v prijemalu varilno pištolo ali pa valj, je potrebno hitrost robota zmanjšati.

Kdaj se uporabi fine, kdaj se uporabi z. Potrebno je razumeti, kdaj v programu si želimo, da se robot popolnoma ustavi v tisti točki. Če se v vrstici uporabi z, se s tem doseže, da se robot ne ustavi v točki, kjer je zapisan z in robot to točko zaobide. Večji kot je z, dlje gre robot od točke. Primer: če bi z bil 1, se bi robot popolnoma približal tej točki, vendar se ne bi ustavil, ampak bi kar nadaljeval pot do naslednje točke. V primeru pa, da je z 100, pa se bo robot premaknil mimo te točke za 100 mm oziroma je možna tolikšna toleranca.

3.5 VAJA 3: VARJENJE 2D ENOSTAVNE KONTURE Z VARILNO PIŠTOLO

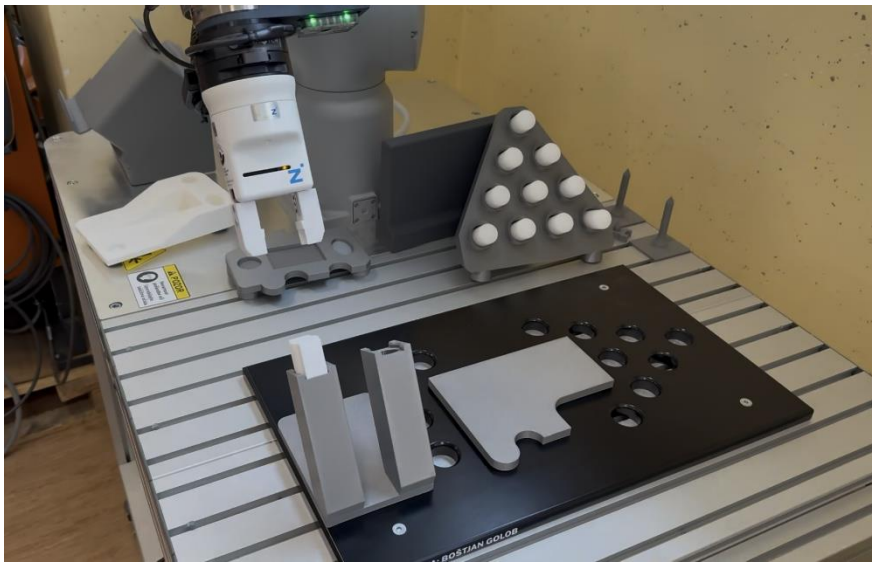
Ustvarite program, ki se začne tako, da je robotska roka v home poziciji. Nato se premakne do držala, v katerem je varilna pištola, ki jo pobere.

Slika 75: Pot, ki jo robot opravi



Ko robot pobere varilno pištolo, za katero je potrebno imeti posebej napisan program, gre robot proti začetni točki varjenja te enostavne 2D konture. Ko konča s sledenjem po konturi, se robot v zadnji točki malo dvigne in nato začne pot proti mestu, kjer je pobral varilno pištolo in jo odloži. Ko jo odloži, gre robot na domač položaj. Tukaj je potrebno upoštevati, da je hitrost varjenja prilagojena.

Slika 76: Pozicija robota, preden pobere varilno pištolo



Slika 77: Začetek programa za 2D varjenje

```
995     PROC p_Welding_2D()
996         !MoveAbsJ jt_Varjenje_HOME,v1000,z100,t_Varilna_Pistola\WObj:=wobj_2D_kontura;
997         MoveJ RT3, v1000, z50, t_Varilna_Pistola\WObj:=wobj_2D_kontura;
998         MoveJ AP1, v1000, z50, t_Varilna_Pistola\WObj:=wobj_2D_kontura;
999 
1000        MoveL k2D_1,v20,fine,t_Varilna_Pistola\WObj:=wobj_2D_kontura;
1001        MoveL k2D_2,v20,fine,t_Varilna_Pistola\WObj:=wobj_2D_kontura;
1002        MoveL k2D_3,v20,fine,t_Varilna_Pistola\WObj:=wobj_2D_kontura;
1003        MoveL k2D_4,v20,fine,t_Varilna_Pistola\WObj:=wobj_2D_kontura;
1004        MoveL k2D_5,v20,fine,t_Varilna_Pistola\WObj:=wobj_2D_kontura;
1005        MoveL k2D_6,v20,z10,t_Varilna_Pistola\WObj:=wobj_2D_kontura;
1006        MoveC k2D_7,k2D_8,v20,z10,t_Varilna_Pistola\WObj:=wobj_2D_kontura;
1007        !MoveL k2D_8,v1000,z5,t_Varilna_Pistola\WObj:=wobj_2D_kontura;
1008        MoveC k2D_9,k2D_10,v20,fine,t_Varilna_Pistola\WObj:=wobj_2D_kontura;
1009        MoveL k2D_11,v20,fine,t_Varilna_Pistola\WObj:=wobj_2D_kontura;
1010        MoveL k2D_12,v20,z1,t_Varilna_Pistola\WObj:=wobj_2D_kontura;
1011        MoveL k2D_13, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura;
1012        MoveL k2D_14, v20, z10, t_Varilna_Pistola\WObj:=wobj_2D_kontura;
1013        MoveC k2D_15,k2D_16,v20,z10,t_Varilna_Pistola\WObj:=wobj_2D_kontura;
1014        MoveL k2D_17,v20,fine,t_Varilna_Pistola\WObj:=wobj_2D_kontura;
1015        MoveJ RT1, v20, z50, t_Varilna_Pistola\WObj:=wobj_2D_kontura;
1016 
1017        !MoveAbsJ jt_Varjenje_HOME,v1000,z100,t_Varilna_Pistola\WObj:=wobj_2D_kontura;
1018    ENDPROC
```

Ko zazna, da je prišel skozi to točko, nadaljuje naprej do točke, ki je napisana v programu p_Welding_2D, saj je to naslednji program v mainModule. Do točke se premakne z veliko hitrostjo in tudi v točko AP1 se premakne z veliko hitrostjo. Ta točka je approach točka, ki je malo dvignjena nad začetno točko. Ko pa se robot premakne v točko k2D_1, takrat robot zmanjša hitrost na v20 in se v tej točki ustavi, saj mora tukaj zelo natančno začeti variti. V primeru, da bi namesto fine tukaj bil zapisan z100, se robot ne bi ustavil v tej točki, vendar bi jo zaobšel in bi zgrešil točko. Robot ima točke na koncu vsake stranice te konture, zato je v točkah od 1001 do 1004 zapisan fine.

Robot v vrstici 1005 gre skozi točko zato, da izgleda delovanje robota in postopek varjenja lepši in hitrejši brez ustavljanja. Glede na to, da je ta točka narejena preden robot naredi cirkularni gib, se mora upoštevati z kot boljša alternativa.

Robot izvede cirkularni gib z dvema točkama k2D_7, k2D_8. Zaradi z10 dosežemo to, da gre robot skozi točko brez vmesnega ustavljanja in da je še vedno dovolj natančen, da naredi lep polkrog.

V nadaljevanju uporabi naslednji cirkularni gib s točkama k2D_9, k2D_10. Ker je na koncu te vrstice zapisan fine, se robot ustavi samo v točki k2D_10 in ne v točki, ki je prej.

Robot kasneje z linearnimi točkami potuje vse do točk k2D_14.
Nato nadaljuje s cirkularnim gibom v točkah k2D_15 ter k2D_16. Ko se premakne do točke k2D_17 pride do konca konture. Nato se z Joint gibom dvigne v RT1 točko, malo se lahko tudi že poviša hitrost, saj ne vari več in z je lahko višji, saj mora iti samo skozi to točko. Tukaj je konec glavnega programa p_Welding_2D.

Slika 78: Klic programa za pobiranje varilne pištole

```
PROC p_Drzalo_orodja()
    ! Zacetna pozicija
    MoveAbsJ jt_Home, v1000, z100, t_Prijemalo\WObj:=wobj_drzalo_orodja;
    ! Postavitev nad orodje
    MoveJ O_1_Pick_Above, v1000, z10, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    ! Pobiranje orodja
    MoveL O_2_Pick, v500, fine, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    ! Zapri prijemalo
    closeGripper;
    WaitTime 1;
    ! Dvig prijemala
    MoveL O_3_Pick_Up, v500, z1, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    ! Koncna pozicija
    !MoveAbsJ jt_Varjenje_HOME,v1000,z100,t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
ENDPROC
```

Pri programu za varjenje enostavne 2D konture ima robot več opravkov. Robot mora najprej pobrati varilno pištolo, ki jo uporabi pri varjenju te konture. Program p_Drzalo_Orodja je v mainModule zapisan pred p_Welding_2D. Pri izvedbi programa za pobiranje varilne pištole se uporabljajo točke jt_Home. Ta točka pove, da je robot v domači home točki, kjer se nato premakne v O_1_Pick_Above. Premakne se v naslednjo točko O_2_Pick, kjer robot že malo zmanjša hitrost in se v tej točki tudi ustavi. Nato robot zapre prijemalo ter počaka eno sekundo, nato se dvigne v točko O_3_Pick_Up. V tej točki se robot ne ustavi popolnoma, gre skozi točko.

Slika 79: Klic programa za odlaganje varilne pištole

```
PROC p_drzalo_orodjak()
    MoveJ O_1_Pick_Above10, v1000, z50, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
    MoveL O_1_Pick_Above, v1000, z100, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    MoveL O_2_Pick, v500, fine, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    openGripper;
    WaitTime 1;
    MoveL O_3_Pick_Up, v500, z10, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    MoveAbsJ jt_Varjenje_HOME,v1000,z100,t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
ENDPROC
```

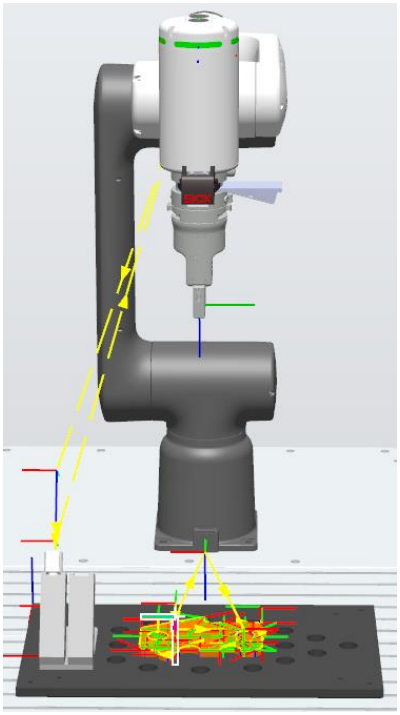
Nato bo robot začel z branjem vrstic v programu p_drzalo_orodjak. Tukaj se robot v Joint gibu premakne v točko O_1_Pick_Above10, robot do te točke pospeši in ima hitrost v1000, nato se v Linear gibu premakne v točko O_1_Pick_Above z isto hitrostjo ter z100. Robot gre skozi to točko do točke O_2_Pick, kjer se ustavi. Točke v tem programu so iste kot točke v programu p_Drzalo_orodja, vendar pa je v programu, kjer robot odloži orodje, še vedno nekaj razlik.

Robot spusti varilno pištolo z ukazom openGripper, kjer po odprtju počaka 1 sekundo, nato se premakne v točko pomika nazaj in kasneje preide v home točko.

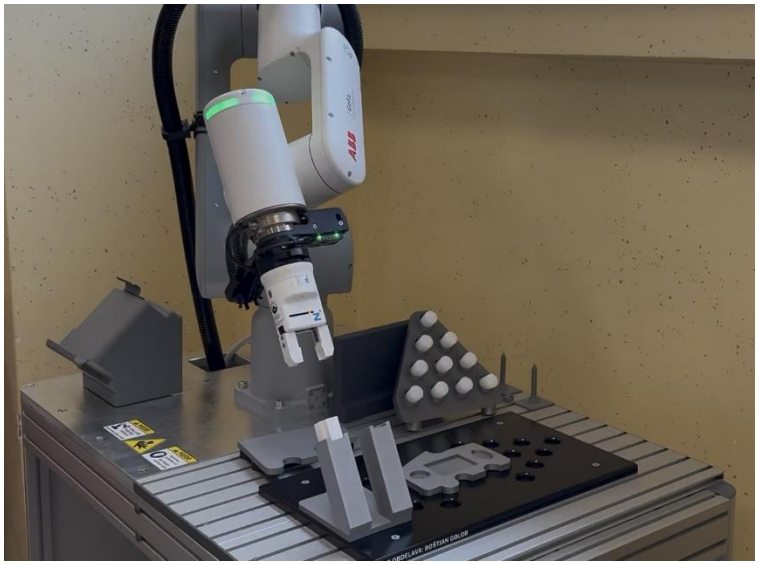
3.6 VAJA 4: VARJENJE 2D NAPREDNEJŠE KONTURE

Ustvarite program, kjer bo začetek programa v home poziciji. V tem programu boste imeli nov 2D lik, vendar ta ima drugačno zunanjo pot in tri oblike v notranjosti tega predmeta. Potrebno je za vsak lik uporabiti svoj program, lahko pa imate tudi vse točke v enem programu, vendar je bolj pregledno, če imate vsako obliko v svojem programu. Zaporedje oblik, ki ga sledite je, da naprej začnete na notranjih oblikah, in sicer po zaporedju: pravokotnik, krog in elipsa, nato pa napreduje z varjenjem po zunanjem robu.

Slika 80: Pot, ki jo robot opravi



Slika 81: Pozicija robota, preden pobere varilno pištolo



Slika 82: Klic vseh programov v glavnem programu

```
! Klic Welding 2D z oblikami
p_Drzalo_orodja;
p_Welding_2D_oblike;
p_Welding_2D_oblike_2;
p_Welding_2D_oblike_3;
p_Welding_2D_oblike_4;
p_drzalo_orodjak;
WaitTime 1;
```

Slika 83: Program za pobiranje orodja

```
PROC p_Drzalo_orodja()
    ! Zacetna pozicija
    MoveAbsJ jt_Home, v1000, z100, t_Prijemalo\WObj:=wobj_drzalo_orodja;
    ! Postavitev nad orodje
    MoveJ O_1_Pick_Above, v1000, z10, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    ! Pobiranje orodja
    MoveL O_2_Pick, v500, fine, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    ! Zapri prijemalo
    closeGripper;
    WaitTime 1;
    ! Dvig prijemala
    MoveL O_3_Pick_Up, v500, z1, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    ! Koncna pozicija
    !MoveAbsJ jt_Varjenje_HOME,v1000,z100,t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
ENDPROC
```

S programom p_Drzalo_orodja dosežemo, da robot preden izvede program, pobere varilno pištolo in nato se nadaljuje branje programa. Program za pobiranje varilne pištole lahko uporabite pri vsaki vaji, kjer je to potrebno.

Prva oblika v programu je pravokotnik. Pri tem programu je pomembno, da je hitrost nizka in da se uporabi znanje, kdaj se uporabi ukaz fine. V tem primeru je potrebno uporabiti ukaz fine, saj boste s tem dosegli, da se bo robot v točki, do katere mora priti, tudi ustavil in zarotiral v naslednjo točko in kljub temu ostal na isti točki. Pri nekaterih točkah se bo spremenila samo rotacija orodja, da bo robot lahko pravilno sledil obliki.

Slika 84: Prva oblika je pravokotnik

```
! Pravokotnik
MoveJ k2Do_2e_20, v20, z5, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_3p_1, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_3p_2, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_3p_3, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_3p_4, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_3p_5, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_3p_6, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_3p_7, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_3p_8, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_3p_18, v20, z50, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_3p_9, v20, z100, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
```

Naslednja oblika je krog. Krog se naredi tako, da se uporabita dva cirkularna premika. Cirkularni premik boste dosegli tako, da boste dali dve točki dovolj narazen, da bo robot lahko naredil ta gib. Pri tem ni potrebno imeti ukaza fine, saj si želimo, da bo robot naredil lep krog brez ustavljanja.

Slika 85: Naslednja oblika je krog

```
! Krog
MoveJ k2Do_3p_19, v20, z30, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_4k_1, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveC k2Do_4k_2, k2Do_4k_3, v20, z10, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveC k2Do_4k_4, k2Do_4k_5, v20, z10, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_4k_5, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_4k_15, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
```

Po krogu bo robot začel z varjenjem v elipsi. Postopek izdelave elipse je enak kot pri krogu, vendar se uporabi več cirkularnih premikov. Pri zadnjem cirkularnem premiku je potrebno imeti ukaz fine, saj se more robot v tisti točki ustavit saj sledi linearni premik navzgor, saj bo robot nadaljeval s svojim programom in ga umaknemo od točke, kjer je končal z elipso.

Slika 86: Primer programa za elipso

```
! Elipsa
MoveJ k2Do_1r_32, v20, z100, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_42, v20, z10, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_2e_1, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveC k2Do_2e_2, k2Do_2e_3, v20, z10, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveC k2Do_2e_4, k2Do_2e_5, v20, z10, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveC k2Do_2e_6, k2Do_2e_7, v20, z10, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveC k2Do_2e_8, k2Do_2e_9, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_2e_19, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_2e_10, v20, z100, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
```

Robot nato nadaljuje z zunanjim robom. Potrebno je razumeti, kdaj se uporabi ukaz fine in kdaj se uporabi ukaz z. Gibanje robota je potrebno razumeti, saj bo tako programiranje lažje. Princip programiranja je enak kot pri oblikah pred tem. Če je na poti robota kakšen oster rob, se uporabi ukaz fine, saj se mora robot v tistem delu obrniti in nadaljevati s programom. Če je na poti robota kakšna zaobljena oblika in se na to nadaljuje z ravnim premikom in ne z ostrim robom, se lahko uporabi ukaz z. Hitrost varjenja skozi ostaja enaka – pri varjenju je vedno potrebno imeti nizko hitrost, saj se vari počasi.

Slika 87: Primer programa za rob

```
! Rob
MoveJ APP1, v500, z10, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_1, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_2, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_3, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_4, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_5, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_6, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_7, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveC k2Do_1r_8, k2Do_1r_9, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_10, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_11, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_12, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveC k2Do_1r_13, k2Do_1r_14, v20, z10, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveC k2Do_1r_15, k2Do_1r_16, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_17, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_18, v20, z10, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveC k2Do_1r_19, k2Do_1r_20, v20, z10, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_21, v20, z10, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveC k2Do_1r_22, k2Do_1r_23, v20, z10, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_24, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_25, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_26, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_27, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveC k2Do_1r_28, k2Do_1r_29, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_30, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL k2Do_1r_31, v20, fine, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
MoveL RTT1, v20, z10, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
```

Slika 88: Klic programa za odlaganje orodja

```
PROC p_drzalo_orodjak()
    MoveJ O_1_Pick_Above10, v1000, z50, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
    MoveL O_1_Pick_Above, v1000, z100, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    MoveL O_2_Pick, v500, fine, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    openGripper;
    WaitTime 1;
    MoveL O_3_Pick_Up, v500, z10, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    MoveAbsJ jt_Varjenje_HOME,v1000,z100,t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
ENDPROC
```

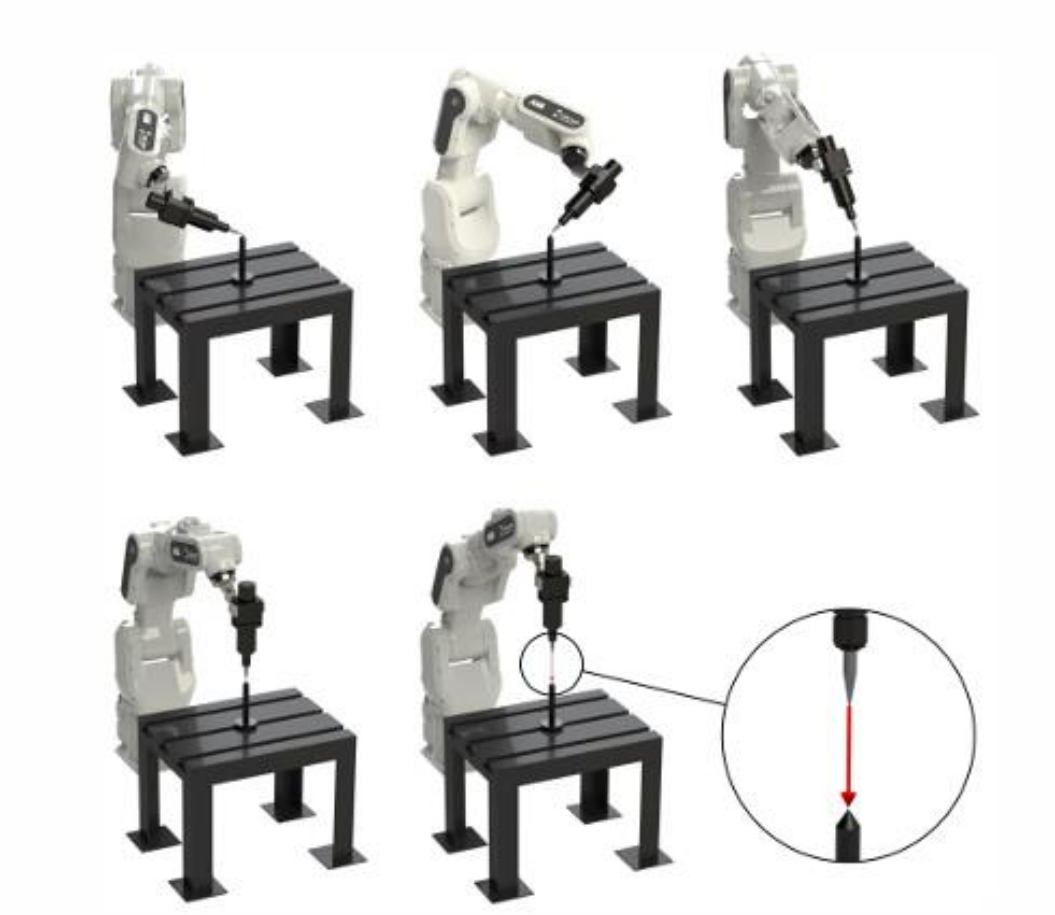
S tem programom boste dosegli, da bo robot vrnil varilno pištolo na mesto, kjer jo je vzel na začetku programa. Program je identičen programu za pobiranje orodja, vendar je tu uporabljen openGripper. Ta program lahko uporabite v vsakem programu, kjer potrebujete pobiranje in odlaganje varilne pištole.

3.7 VAJA 5: NASTAVITEV KOORDINATNEGA SISTEMA ORODJA

Nastavitev koordinatnega sistema orodja za varilno pištolo ali za konico z zastavico je načeloma enako.

Poznamo več metod, kako se nastavi TCP v obeh primerih.
Tool koordinatni sistem je definiran glede na zapestni koordinatni sistem robota. TCP je koordinatno izhodišče pa točka na orodju, ki jo mora robot natančno voditi (tu noter pa spadata varilna pištola in konica z zastavico). Če pa se zamenja orodje, se spremeni tudi Tool. Program običajno ostane isti.

Slika 89: Opredelitev TCP s 3 točkami in Z



(ABB, 2024)

3.7.1 Postopek na učni enoti

Menu – Data – Tool data – New
Potem se nastavi masa orodja in težišče.

3.7.2 Metode kalibracije TCP (najpogostejše)

Uporablja se največkrat 4 točkovna metoda, ki pa je hkrati tudi najbolj standardna. Rezultat te metode je, da ima vsako orodje svoj TCP. Postopek, kako se to opravi, je pa sledeč:

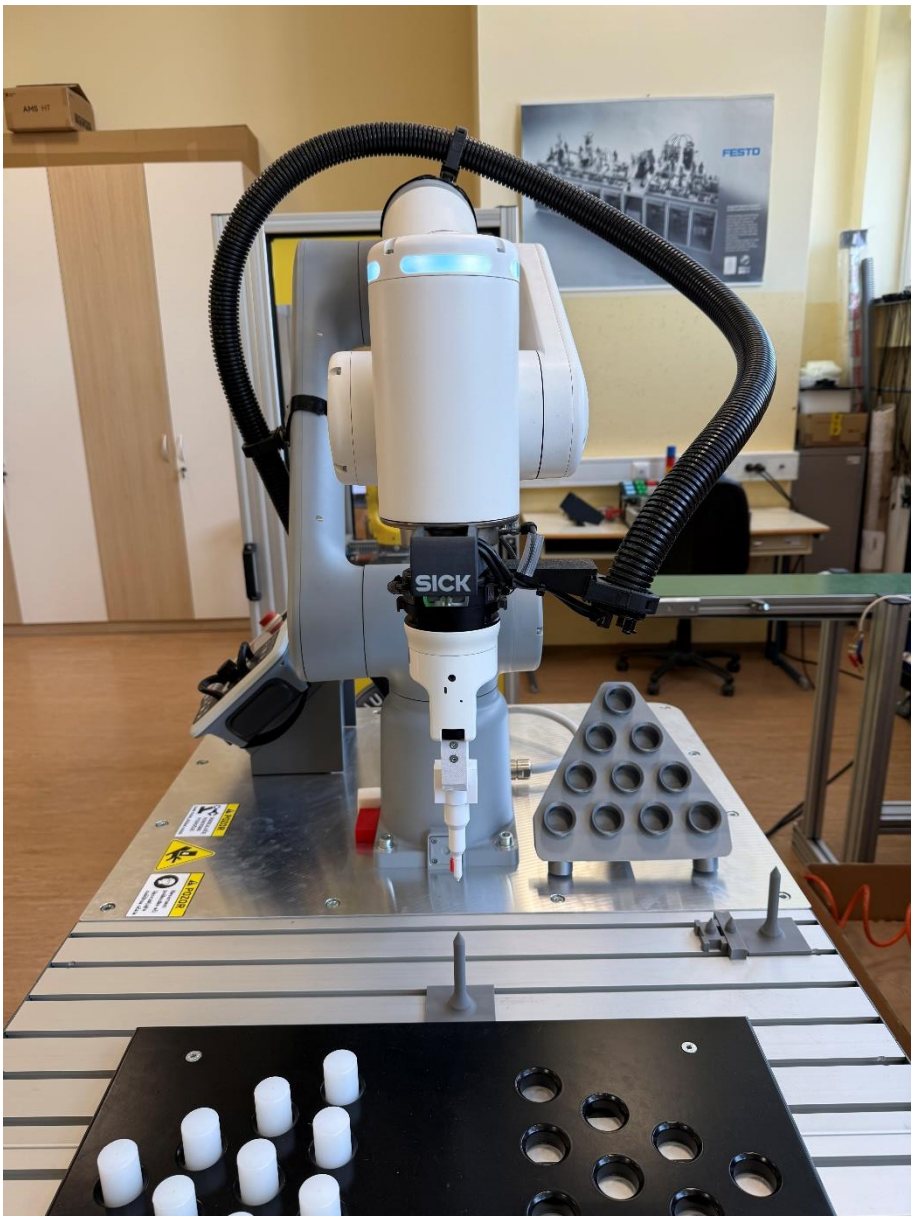
Izbereš en fiksni oster referenčni point v celici. Z orodjem se ga nato dotakneš. Spremeniš rotacijo orodja. Nato se ponovno dotakneš orodja in to ponoviš 4 krat, da imaš 4 različne orientacije in tako potem robot izračuna TCP.

Slika 90: Robot v poziciji za nastavljanje TCP

Slika 91: Robot v poziciji za nastavljanje TCP

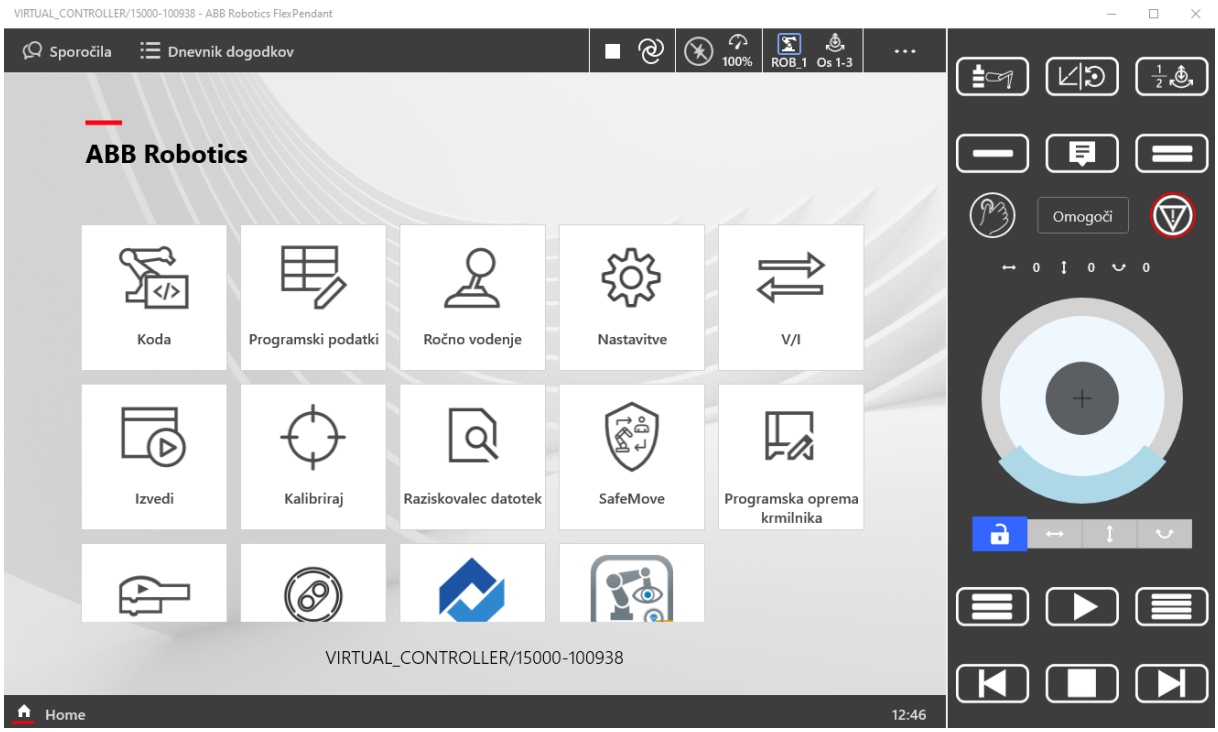
Slika 92: Robot v poziciji za nastavljanje TCP

Slika 93: Postavitev robota višje po Z osi



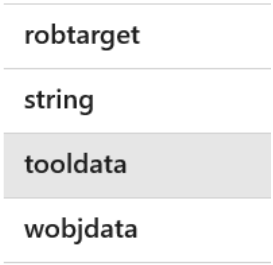
Ta metoda definira samo TCP, orientacija ostane isto kot zapestje.
Lahko se še uporabi metoda 5 točk, kjer pa se še definira Z smer orodja.
Na robotu je potrebno vedno imeti isto fizično točko, spremeni se samo orientacija orodja. Za metodo 5 ali 6 točk se spremeni samo orodje v smeri Z ter v smeri X.
Da se prepričamo, da smo naredili prav, lahko testiramo z Jog v Tool koordinatah. Vrti orodje – konica mora ostati na mestu. Premik Z – mora iti naravnost po osi orodja.
V primeru napake bo konica risala krog, tako se bo videlo, da TCP ni pravilno nastavljen.
Tipične napake, ki se lahko zgodijo so, da se premikajo XYZ med TCP točkami. Da imamo premalo različne orientacije, slab referenčni point in da je narobe tudi Tool masa – slab motion.

Slika 94: Slika domačega zaslona



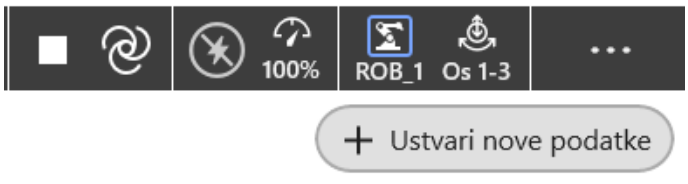
1. Ko ste na domačem zaslonu, izberete zavihek Programski podatki.

Slika 95: Izbira tooldata



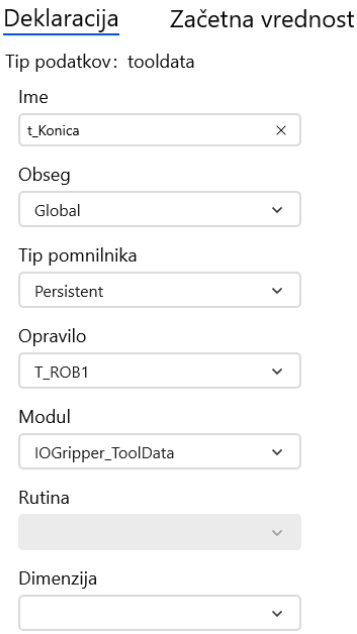
2. Nato izberete element tooldata

Slika 96: Ustvari nove podatke



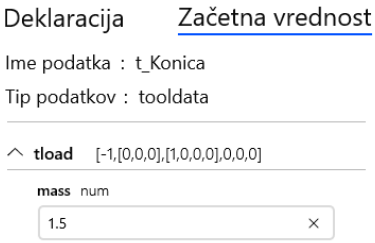
3. Izberete zavihek Ustvari nove podatke.

Slika 97: Izpolni potrebne stvari in poimenujte svoj tooldata



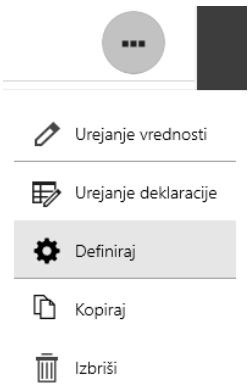
4. Nato v zavihku Deklaracije izpolnite potrebne podatke po vrsti, ki so zapisani.

Slika 98: Tload mass na 1,5



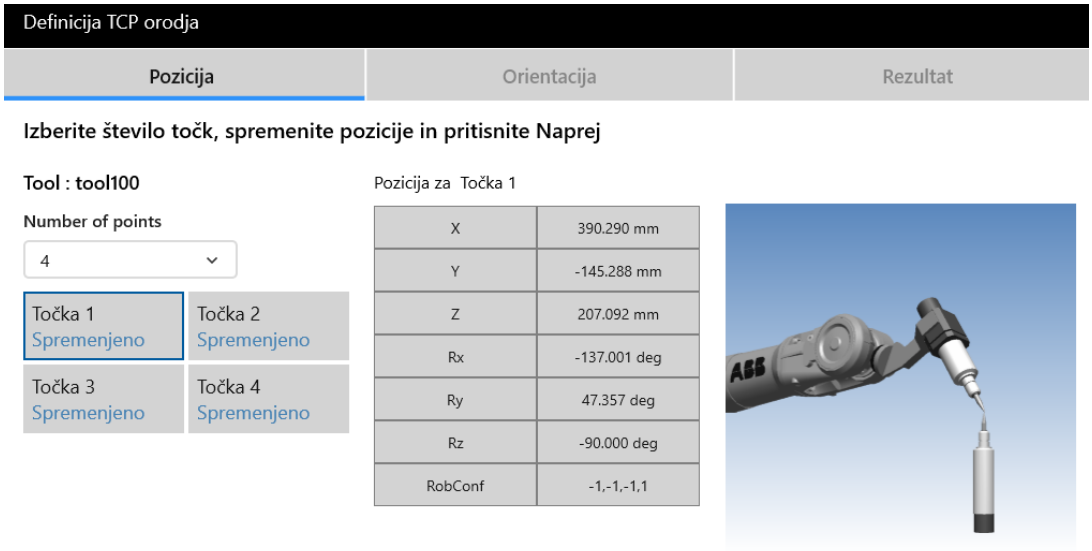
5. V meniju začetna vrednost je potrebno spremeniti samo tload mass na 1,5, nato vse to potrdite v desnem kotu z gumbom Uporabi. Prikazal se vam bo na novo narejen tooldata.

Slika 99: Definiraj svoj tooldata



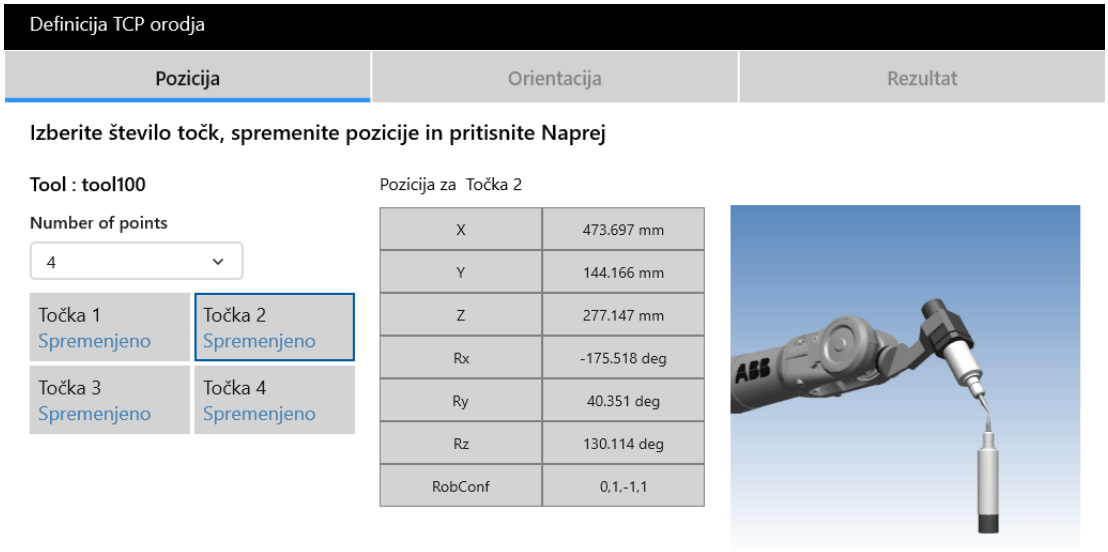
6. Na desni strani boste videli tri pikice, pritisnite jih in izberite zavihek Definiraj.

Slika 100: Metoda 4 točk



7. Tukaj kasneje definirate vaš TCP z metodo s 4 točkami. Lahko si izbirate, s koliko točkami si boste naredili vaš TCP. Imate pa tudi v razpredelnici vrednosti vaše 1. točke.

Slika 101: Metoda 4 točk - rezultat 2. točke

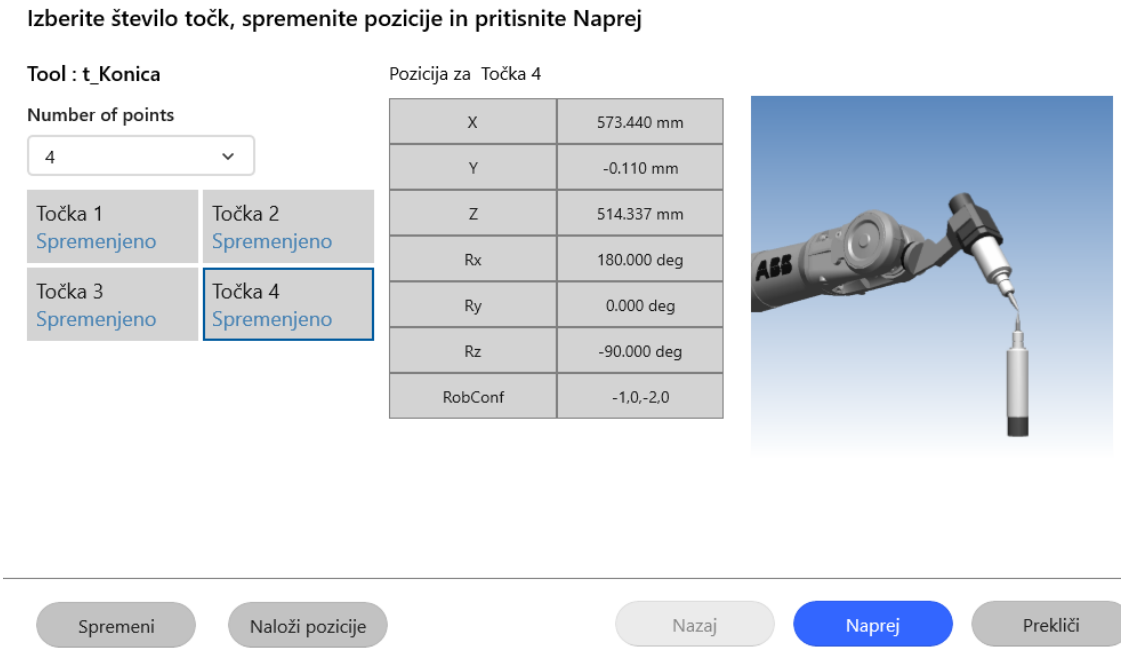


8. Po potrditvi točke se vam izpišejo vrednosti za vašo točko v 2. poziciji

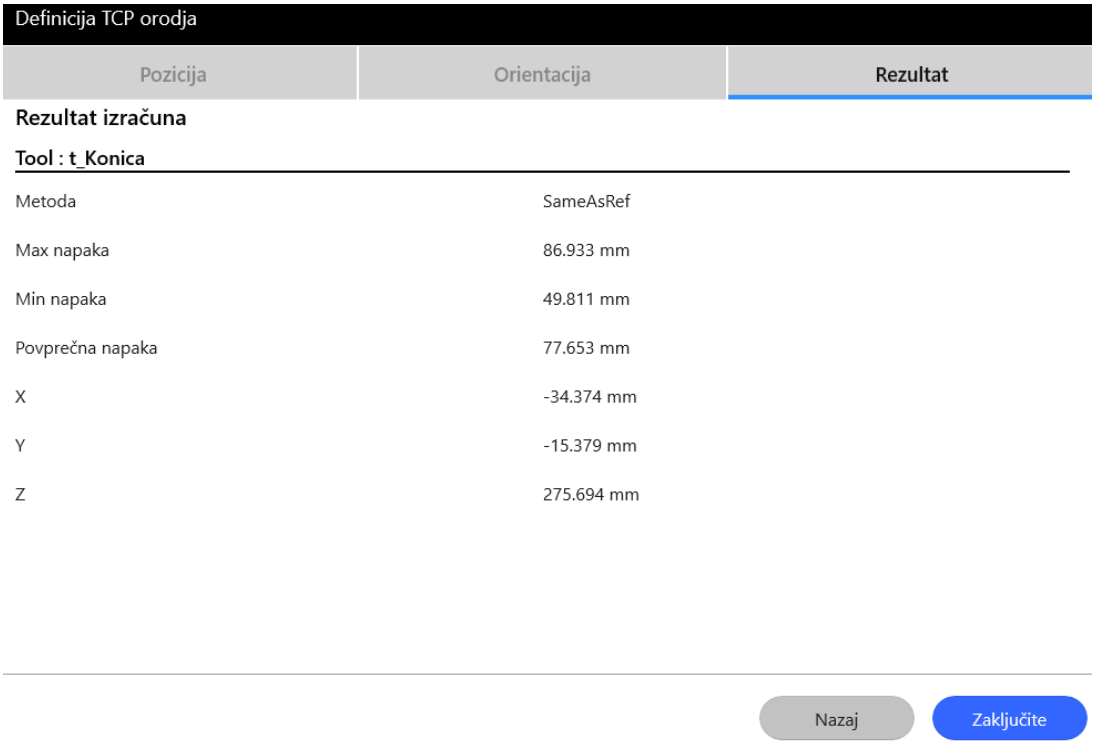
Slika 102: Metoda 4 točk - rezultat 3. točke

9. Po potrditvi 3. točke dobite rezultat enako kot ste dobili v prvih dveh primerih.

Slika 103: Spremenjene točke in rezultat



Slika 104: Rezultat izračuna



10. Prikaz izračuna vaših vrednosti po spremenjenih pozicijah.

3.8 VAJA 6: NASTAVITEV DELOVNEGA KOORDINATNEGA SISTEMA

Nastavitev delovnega koordinatnega sistema se nastavlja, saj so vsi naučeni robotični položaji vezani na WObj. Napačen WObj povzroči slabo gibanje po X/Y ravnini kosa.
Wobj se običajno kalibrira glede na že definiran tool koordinatni sistem.

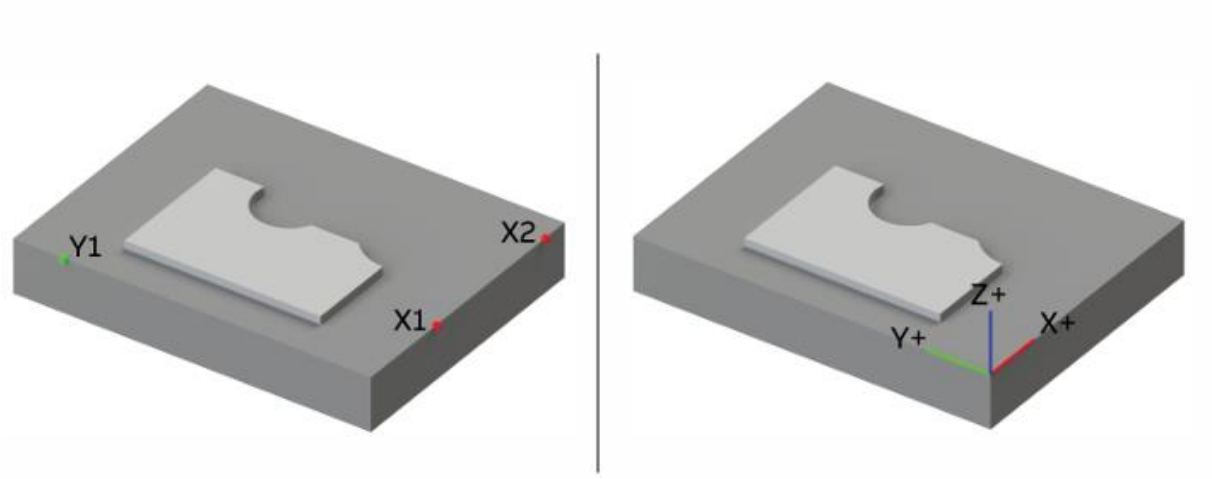
3.8.1 Postopek na učni enoti

ABB meni – Jogging- Work Object – New – OK. To ustvari nov WObj. Je pa priporočljivo, da se ga preimenuje v nekaj smiselnega. Naj bo tudi global in persistent, saj bo le tako na voljo v programu.

3.8.2 Nastavi koordinatni sistem v WObj

WObj se definira tako, da robot pokaže lokacijo z več točkami. Tipično je, da sta 2 točki za X os in 1 točka za Y smer.

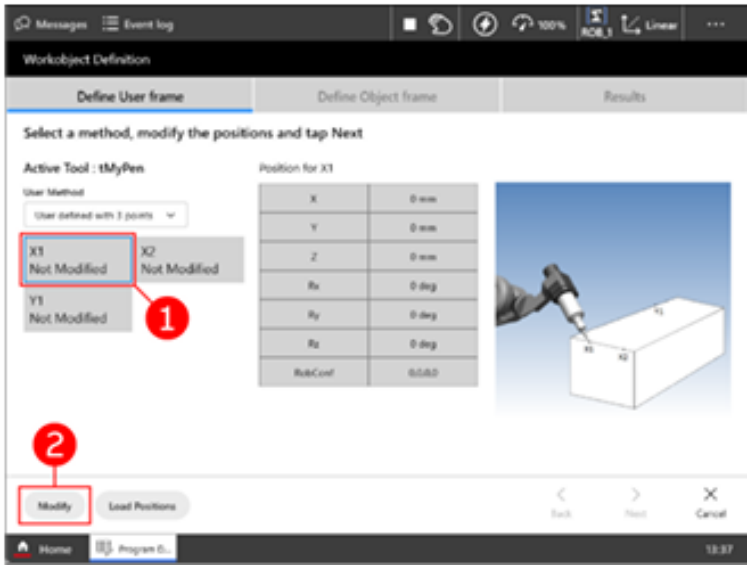
Slika 105: Primer za nastavitev work objecta



(ABB, 2024)

Lahko pa se uporabi tudi ročni vnos, lahko se vpišejo X, Y, Z položaj in orientacija. WObj vsebuje user frame position + rotation, object frame position + rotation.

Slika 106: Nastavitev work object



Na home zaslonu boste pritisnili zavihek Programski podatki.
Nato boste začeli z ustvarjanjem delovnega objekta. Opredelili ga boste s tremi točkami kot je prikazano na fotografiji. Ustvarili boste rutino za risanje lika v delovnem objektu, nato boste lahko izvedli preizkus.
Premaknite papir in na novo opredelite delovni objekt, nato pa izvedite preizkus.

3.8.3 Tipičen work flow na robotu

Kreiraš Tool tako, da se naredi TCP + masa + orientacija.
Kreiraš WObj ter naučiš referenčno točko

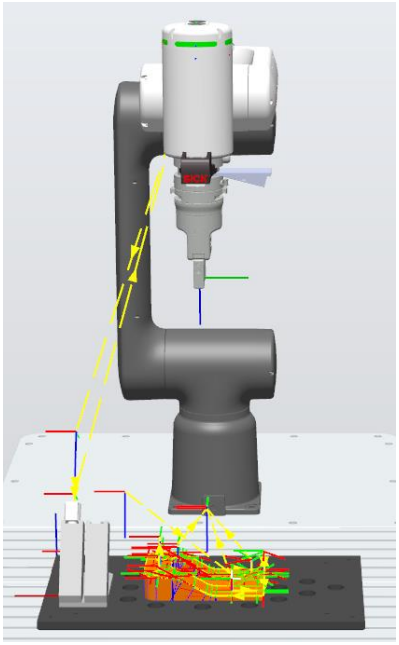
3.8.4 Primer uporabe WObj v programu

MoveL p1_2_Pick, v1000, z100, t_Prijemalo\WObj: =wobj_plosca_1;

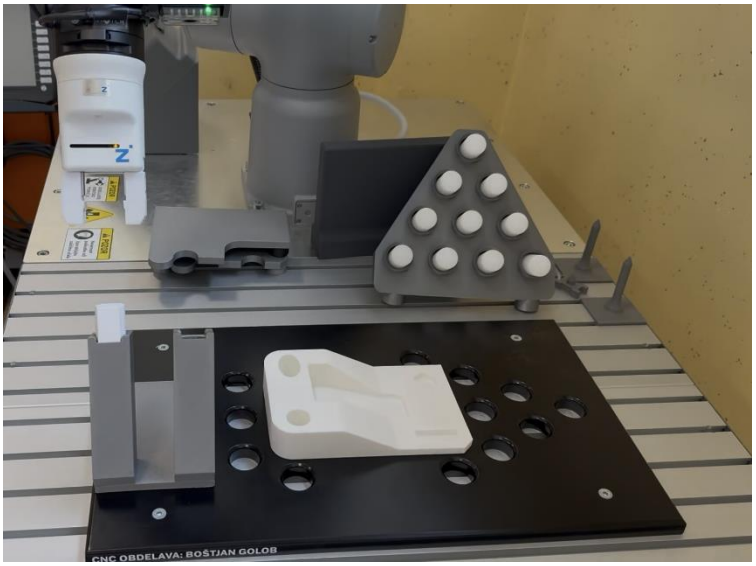
3.9 VAJA 7: SLEDENJE 3D KONTURI

Ustvarite program, ki ga je potrebno začeti v začetni home poziciji. Na začetku se robotska roka po programu pomakne proti držalu orodja, saj mora robot pobrati varilno pištolo, da bo z njo lahko varil. Ko jo dvigne iz pobiralnega mesta, se premakne v prvo točko, kjer bo začel slediti obliki. Zaporedje oblik je določeno tako, da robot začne v pravokotniku, nato nadaljuje v heksagon, valj in krog. Ko zaključi s temi oblikami, nadaljuje na notranji rob, nato pa še na zunanji rob. Ko konča s sledenjem vseh oblik in zunanjega roba, se robot ponovno premakne proti držalu orodja in odloži varilno pištolo tam, kjer jo je pobral. Nato se robot premakne v domačo pozicijo.

Slika 107: Pot, ki jo robot opravi



Slika 108: Točka, preden robot pobere varilno pištolo



Ta program je napisan malo drugače, in sicer tako, da so vsi programi napisani v enem večjem programu, vendar so razvidno ločeni s presledki in komentarjem, da se točno ve, v katero obliko se bo premaknil robot.

Slika 109: Program za pobiranje varilne pištole

```
PROC p_Drzalo_orodja()
    ! Zacetna pozicija
    MoveAbsJ jt_Home, v1000, z100, t_Prijemalo\WObj:=wobj_drzalo_orodja;
    ! Postavitev nad orodje
    MoveJ O_1_Pick_Above, v1000, z10, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    ! Pobiranje orodja
    MoveL O_2_Pick, v500, fine, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    ! Zapri prijemalo
    closeGripper;
    WaitTime 1;
    ! Dvig prijemala
    MoveL O_3_Pick_Up, v500, z1, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    ! Koncna pozicija
    !MoveAbsJ jt_Varjenje_HOME,v1000,z100,t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
ENDPROC
```

Robot začne na začetku s prihodom do pobiranja varilne pištole, ki je praktično enak način kot v prejšnjih primerih. Lahko se uporabi enak program, saj pobere robot varilno pištolo na istem mestu.

Slika 110: Pravokotnik v 3D konturi

```
1156        ! Pravokotnik
1157        MoveJ APPP1, v500, z50, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1158        MoveL k3D_2s_32, v500, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1159        MoveL k3D_3p_1, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1160        MoveL k3D_3p_2, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1161        MoveL k3D_3p_3, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1162        MoveL k3D_3p_4, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1163        MoveL k3D_3p_5, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1164        MoveL k3D_3p_6, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1165        MoveL k3D_3p_7, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1166        MoveL k3D_3p_8, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1167        MoveL k3D_3p_18, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1168        MoveL k3D_3p_9, v20, z100, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
```

Pri 3D konturi robot začne z varjenjem pravokotnika. Robot se približa po aproach točki, ki je malo višje nad začetno točko. V tej točki ni potrebno imeti fine nastavitve, saj ni potrebe po ustavljanju, zato je tudi z50, saj je točka dovolj visoko, da jo lahko robot malo zaobide.

V vsaki točki varjenja robota v pravokotniku je potrebno v vsaki vrstici imeti zapisan fine, saj se mora robot v vsaki točki tudi ustaviti. Program je zapisan z linearnimi gibi, saj ni potrebe po joint gibu, saj se robot nikjer ne nagiba. Hitrost varjenja mora biti vedno enaka, saj je s tem doseženo, da je rezultat v lepem zvaru.

Robot se na koncu programa umakne v retrive točko, kjer je lahko uporabljen z, saj ni potrebe po ustavljanju, ker bi robot za dokončanje programa potreboval dlje časa in je tako bolj optimizirana uporaba robota.

Slika 111: Heksagon v 3D konturi

```
1170        ! Heksagon
1171        MoveL k3D_3p_19, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1172        MoveL k3D_4h_1, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1173        MoveL k3D_4h_2, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1174        MoveL k3D_4h_3, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1175        MoveL k3D_4h_4, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1176        MoveL k3D_4h_5, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1177        MoveL k3D_4h_6, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1178        MoveL k3D_4h_7, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1179        MoveL k3D_4h_8, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1180        MoveL k3D_4h_9, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1181        MoveL k3D_4h_10, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1182        MoveL k3D_4h_11, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1183        MoveL k3D_4h_12, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1184        MoveL k3D_4h_22, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1185         MoveL k3D_4h_13, v20, z100, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
```

Heksagon je naslednja oblika, katero začne robot variti. Program je zasnovan enako kot je program za pravokotnik. Potrebno je imeti samo Linear gibe, saj so poti robota samo po ravni liniji. V vsakem kotu se robot ustavi, se zarotira in kasneje odpravi do naslednje točke. Hitrost je nastavljena na v20, saj je to primerna hitrost za varjenje.

Slika 112: Krog v 3D konturi

```
1187        ! Krog
1188        MoveL k3D_4h_23, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1189        MoveL k3D_5k_1, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1190        MoveC k3D_5k_2, k3D_5k_3, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1191        MoveC k3D_5k_4, k3D_5k_5, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1192        MoveL k3D_5k_5, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1193        MoveL k3D_5k_15, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1194        MoveL k3D_5k_6, v20, z100, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
```

Krog je naslednja oblika za heksagonom. V krog se približamo z določeno točko, ki je malo višje od začetne. V začetni točki k3D_5k_1 se robot ustavi, saj je v vrstici zapisane fine. Program se lahko izvede z dvema cirkularnima giboma. Najlažje je, da si krog razdelimo na dva pol kroga. Eno točko damo na polovico pol kroga, naslednjo na konec polkroga. Naslednji cirkularni gib se naredi na polovici naslednjega pol kroga, zadnja točka pa je na koncu kroga, oz. tam, kjer se je začel program. Ko pride robot v končno točko, se ta točka zapiše tudi v novo vrstico, takrat pa naj bo zapisan fine, saj se mora program v tisti točki ustaviti. Nato se še samo doda točka, s katero se robot dvigne nad končno točko in napreduje tudi proti drugemu programu. V tem primeru je dodana še ena točka za premik robota proti novemu delu, kjer mora variti. To drugače ni vedno potrebno narediti, vendar se s tem lahko doseže, da se robot odvrti. Potrebno je razumeti, v kolikšni rotaciji je robot in kaj si želimo, da bo robot delal v programu,

ki prihaja in kako se bo vrtel. Če se mora robot vrteti v eno stran, v katero se je že, bo imel omejene stopinje, ki se še bodo lahko za rotirale. Zato je potrebno poznati rotacije robota, da nam kasneje robot ne bo javljal napak.

Slika 113: Elipsa v 3D konturi

```
1196        ! Elipsa
1197        MoveL k3D_5k_16, v20, z50, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1198        !MoveL k3D_6e_5, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1199        !MoveC k3D_6e_4, k3D_6e_5, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1200        !MoveL k3D_6e_3, v20, z1, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1201        !MoveC k3D_6e_2, k3D_6e_3, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1202        MoveL k3D_6e_1, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1203        MoveC k3D_6e_2, k3D_6e_3, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1204        !MoveL k3D_6e_3, v20, z1, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1205        MoveC k3D_6e_4, k3D_6e_5, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1206        !MoveL k3D_6e_5, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1207        MoveL k3D_6e_15, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1208        ! Sredinski rob
1209        !MoveL k3D_2s_1, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1210        !MoveL k3D_2s_2, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1211        !MoveL k3D_2s_3, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1212        MoveJ k3D_1r_44, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1213        MoveL k3D_2s_4, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1214        MoveL k3D_2s_5, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1215        MoveL k3D_2s_7, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1216        MoveL k3D_2s_8, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1217        MoveL k3D_2s_10, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1218        !MoveC k3D_2s_9, k3D_2s_10, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1219        MoveL k3D_2s_11, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1220        MoveL k3D_2s_14, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1221        !MoveC k3D_2s_13,k3D_2s_14,v150,z10,t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1222        MoveL k3D_2s_15, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1223        MoveL k3D_2s_16, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1224        MoveL k3D_2s_17, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1225        MoveC k3D_2s_18, k3D_2s_19, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1226        MoveL k3D_2s_20, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1227        MoveL k3D_2s_22, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1228        !MoveC k3D_2s_21,k3D_2s_22,v150,z10,t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1229        MoveL k3D_2s_23, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1230        !MoveL k3D_2s_24, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1231        !MoveC k3D_2s_24,k3D_2s_25,v150,z10,t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1232        MoveL k3D_2s_1, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1233        MoveL k3D_2s_2, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1234        MoveL k3D_2s_42, v20, z20, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1235        !MoveL k3D_2s_26,v1000,z10,t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1236        !MoveC k3D_2s_27,k3D_2s_28,v150,z10,t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1237        !MoveL k3D_2s_28,v1000,fine,t_Varilna_Pistola\WObj:=wobj_3D_kontura;
1238        !MoveL k3D_2s_29,v1000,fine,t_Varilna_Pistola\WObj:=wobj_3D_kontura;
```

Elipsa je naslednji program, ki sledi za krogom. K elipsi se približamo v approach točki k3D_5k_16. Te točke se lahko poljubno poimenujejo. Za boljšo pregledno je dobro imeti točke poimenovane tako, da se bo vedelo, kaj ta točka pomeni in v katerem delu programa je, lahko pa se tudi pusti tako kot predlaga TP. V vrstici 1202 je točka, v kateri začne robot z varjenjem. Kot vse točke do zdaj, kjer začne robot variti, je obvezno potrebno imeti v tej vrstici fine oznako. Program se naredi po principu enako kot program za krog. Potrebno je narediti dva kartezična premika, enako narejene 4 točke kot v programu za krog. Program se konča na enak princip kot pa program za krog.

Slika 114: Sredinski rob v 3D konturi

```
! Sredinski rob
! MoveL k3D_2s_1, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 !MoveL k3D_2s_2, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 !MoveL k3D_2s_3, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveJ k3D_1r_44, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_4, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_5, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_7, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_8, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_10, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 !MoveC k3D_2s_9, k3D_2s_10, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_11, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_14, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 !MoveC k3D_2s_13,k3D_2s_14,v150,z10,t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_15, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_16, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_17, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveC k3D_2s_18, k3D_2s_19, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_20, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_22, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveC k3D_2s_21,k3D_2s_22,v150,z10,t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_23, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_24, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 !MoveC k3D_2s_24,k3D_2s_25,v150,z10,t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_1, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_2, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_42, v20, z20, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_26,v1000,z10,t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 !MoveC k3D_2s_27,k3D_2s_28,v150,z10,t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_28,v1000,fine,t_Varilna_Pistola\WObj:=wobj_3D_kontura;
 MoveL k3D_2s_29,v1000,fine,t_Varilna_Pistola\WObj:=wobj_3D_kontura;
```

Naslednji program po vrsti je program za sredinski rob. Ta program je med bolj zapletenimi, saj se mora robot rotirati ter uporabljati tudi cirkularne premike. Robot se približa notranjemu robu skozi approach točko do začetne točke, kjer se ustavi. Robot se v vsakem robu zasuče primerno varjenju. Ko robot pride do dela, kjer se mora spustiti dol po robu, se lahko uporabita dva načina, kako bi robot nadaljeval pot po premiku navzdol. Ker ta del deluje, kot da bi robot moral uporabiti cirkularni premik, je ena možnost ta. Vendar sem pri tem programu uporabil linearni premik, saj je bil del, pri katerem bi moral uporabiti cirkularni gib, minimalen. Na začetku se lahko proba s cirkularnim premikom, vendar se bo hitro opazilo, da gre robot preveč naokrog in ne bo prišlo do lepih zvarov. Če pa se dve točki postavljeni blizu ena drugi, in se uporabi linearni premik in robota tudi pravilno zarotiramo, bo robot šel po liniji in naredil odličen zvar. V teh točkah se lahko uporabi z način, saj tukaj fine ni potreben, ker robota ne bomo ustavljali, saj je boljše, če nadaljuje z varjenjem in bo tako program deloval boljše. Cirkularni gib se uporabi samo na polkrožnem delu notranjega roba. Robot nato po cirkularnem gibu ponovno potuje po začrtani poti, ki jo naredimo s točkami. Ko pride robot do pregiba na konturi, se lahko ponovno uporabi linearni gib, saj je pregib tako majhen, da cirkularni premik ne bo naredil lepega premika in bo zato zvar malo zgrešen. Ko je konec programa, se z linearnim premikom premakne robot v točko, ki je višje kot pa zadnja točka. Pri tej točki je priporočljivo uporabiti ukaz z, saj lahko tak robot napreduje v drug program in se v vmesni točki ne zaustavi.

Slika 115: Zunanji rob v 3D konturi

```
! Rob
MoveJ k3D_1r_31, v200, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveJ k3D_1r_1, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveL k3D_1r_2, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveL k3D_1r_3, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveL k3D_1r_27, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveL k3D_1r_26, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveL k3D_1r_4, v20, z5, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveL k3D_1r_6, v20, z5, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
!MoveC k3D_1r_5, k3D_1r_6, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveL k3D_1r_7, v20, z5, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
!MoveL k3D_1r_8, v20, z5, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveC k3D_1r_9, k3D_1r_10, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveL k3D_1r_11, v20, z5, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveC k3D_1r_12, k3D_1r_13, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveL k3D_1r_14, v20, z5, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveC k3D_1r_15, k3D_1r_16, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveL k3D_1r_17, v20, z5, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
!MoveL k3D_1r_19, v20, z5, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveC k3D_1r_18, k3D_1r_19, v20, z5, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveL k3D_1r_20, v20, z5, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
!MoveL k3D_1r_21,v1000,z5,t_Varilna_Pistola\WObj:=wobj_3D_kontura;
!MoveL k3D_1r_22, v20, z10, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveL k3D_1r_23, v20, z5, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveL k3D_1r_24, v20, fine, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
MoveL k3D_1r_34, v20, z50, t_Varilna_Pistola\WObj:=wobj_3D_kontura;
```

Ko je robot zaključil z notranjim robom, napreduje naprej do zunanjega roba. To je pa tudi zadnji del programa, ki bo robot dejansko varil. Pri tem programu se uporabijo linearni in cirkularni premiki okoli obdelovanca. V tem programu je pa že potrebno znati, kdaj se lahko uporabi fine, kdaj pa se lahko uporabi ukaz z. V primeru, da je na poti robota kakšen rob oz. konec ene linije in se nadaljuje pot v drugo linijo, takrat se robot ustavi s fine ukazom, se zarotira in v tej točki, ko se robot zarotira, je tudi potrebno uporabiti fine ukaz. Če je rob zavit tako, da se lahko uporabi cirkularni gib, se uporabi cirkularni gib. V programu probamo doseči to, da se bo robot lepo in tekoče gibal po zastavljeni poti, zato bo tudi poznavanje programa in vednost, kako pot mora robot opraviti, ključnega pomena. Programiranje je za vsak program dejansko enak postopek. Če pa testiramo program, ko ga naredimo in vidimo, da ne dela tako kot bi moral, pa ga lahko tudi popravimo na enostaven način, tako da nekatere točke zakomentiramo, vendar pa se moramo zavedati, da takrat robot ne bo prebral te točke oz. je ne bo upošteval. Na koncu programa bi se robot lahko vrnil tudi domov v home pozicijo robota, vendar še moramo upoštevati, da mora robot pred tem še odložiti varilno pištolo na mesto, kjer jo je vzel.

Slika 116: Klic podprograma za odlaganje varilne pištole

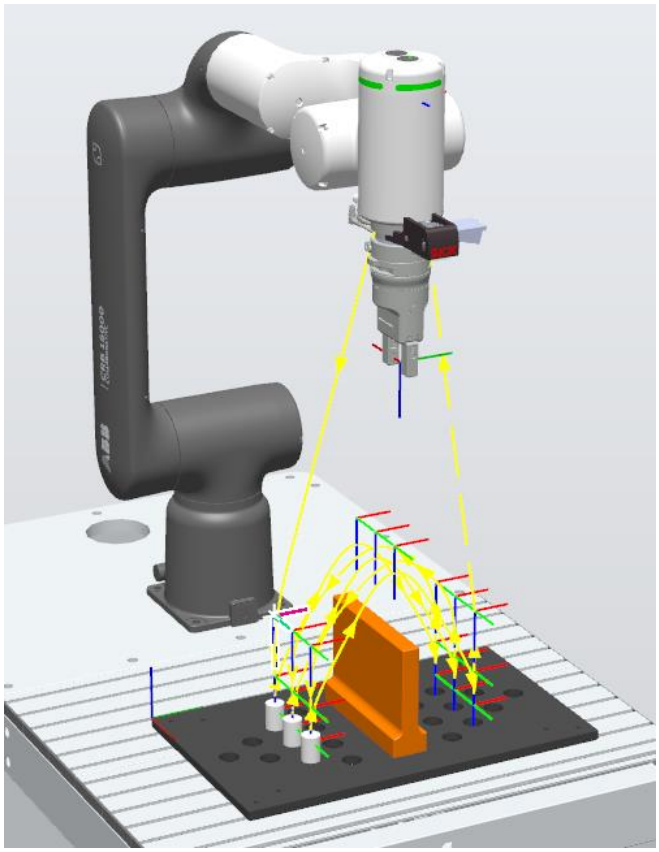
```
PROC p_drzalo_orodjak()
    MoveJ O_1_Pick_Above10, v1000, z50, t_Varilna_Pistola\WObj:=wobj_2D_kontura_oblike;
    MoveL O_1_Pick_Above, v1000, z100, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    MoveL O_2_Pick, v500, fine, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    openGripper;
    WaitTime 1;
    MoveL O_3_Pick_Up, v500, z10, t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
    MoveAbsJ jt_Varjenje_HOME,v1000,z100,t_Varilna_Pistola\WObj:=wobj_drzalo_orodja;
ENDPROC
```

To je zadnji program, ki ga robot opravi na poti do home pozicije. Program je zasnovan enako kot program za pobiranje prijemala. Razlika je samo v tem, da v tem programu uporabimo ukaz openGripper tako, da robot ve, da odpre prijemalo, nato pa se odpravi v home pozicijo.

3.10 VAJA 8A: PROGRAM ZA PRENOS 3 VALJEV

Ustvarite program, kako z računanjem pozicij oziroma z offsetom naredite program za prenos treh valjev preko ovire. Obstaja tudi druga možnost, kako se lahko naredi ta program, in sicer na način, ki je uporabljen pri varjenju, to pa je s klasičnimi točkami, ki jih določimo na robotu.

Slika 117: Pot, ki jo robot opravi



Slika 118: Program za prenos treh valjev

```
 ! Zacetek programa za 3 valje
PROC Pick_and_place_3()
    ! Zacetna pozicija
    MoveAbsJ jt_Home,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_3;
    ! Premik nad valj 1
    MoveJ v1_1_Pick_Above,v300,z100,t_Prijemalo\WObj:=wobj_plosca_3;
    ! Spust do valja
    MoveL v1_2_Pick,v300,z100,t_Prijemalo\WObj:=wobj_plosca_3;
    ! Zapri prijemalo
    closeGripper;
    WaitTime 1;
    ! Dvig valja
    MoveL v1_3_Pick_Up,v200,z100,t_Prijemalo\WObj:=wobj_plosca_3;
    ! Pomik nad oviro
    MoveJ v1_4_Over_Obstacle,v200,z100,t_Prijemalo\WObj:=wobj_plosca_3;
    ! Pomik nad odlagalno mesto
    MoveJ v1_5_Place_Above,v200,z100,t_Prijemalo\WObj:=wobj_plosca_3;
    ! Spust valja
    MoveL v1_6_Place,v200,z100,t_Prijemalo\WObj:=wobj_plosca_3;
    ! Odpri prijemalo
    openGripper;
    WaitTime 1;
    ! Dvig prijemala
    MoveL v1_7_Place_Up,v300,z100,t_Prijemalo\WObj:=wobj_plosca_3;
    ! Pomik nad oviro
    MoveJ v1_8_Over_Obstacle,v300,z100,t_Prijemalo\WObj:=wobj_plosca_3;

    ! Premik nad valj 2
    MoveJ v2_1_Pick_Above,v300,z100,t_Prijemalo\WObj:=wobj_plosca_3;
    ! Spust do valja
    MoveL v2_2_Pick,v300,z100,t_Prijemalo\WObj:=wobj_plosca_3;
```

Tukaj je najprej primer programa, ki je bolj klasičen in tudi enostavnejši za razumevanje in uporabo predvsem za začetnike, in za tiste, ki se želijo na programu učiti, ki se ne bojijo napak in ki si želijo, da se naučijo napake popraviti.

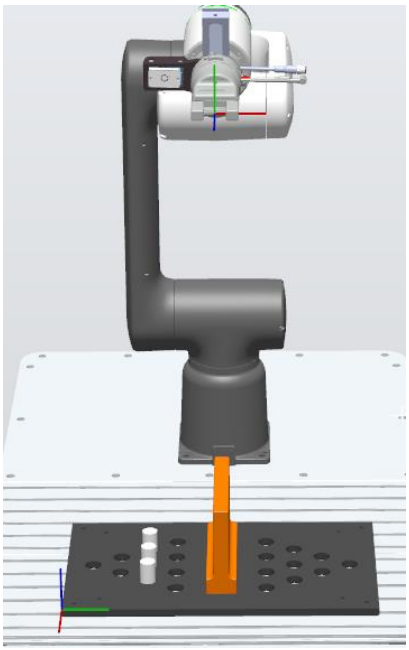
Robot začne v home poziciji.
Podrobna razlaga vrstice, ki jo boste uporabili:
t_Prijemalo – to pomeni, katero orodje izberemo, kateri TCP upošteva,
WObj:=wobj_plosca_3 pomeni, v katerem delovnem okolju se točke shranjujejo, v kateri koordinatni sistem okolja.
z100 pomeni, da gre proti točki, vendar jo zaobide.

Potrebno je robota premakniti v želene točke, ki jih potrebujete za izvedbo programa. Iz domače pozicije premaknite robota nad valj in shranite njegovo točko, določite hitrost in vaš z, saj se robot v tej točki ne rabi ustaviti. Robota nato premaknete v točko, kjer bo dovolj nizko in bo lahko zagrabil valj. Robot v tej točki tudi ustavite. Če boste namesto ukaza fine uporabili ukaz z, se bo prijemalo preden bo prišlo do tiste določene točke že zaprlo in bo robot udaril v valj, s tem pa posledično lahko pride do poškodb robota, tako da je v tej točki zelo potrebno uporabiti izraz fine. Ko boste pobrali valj, se boste premaknili malo višje kot na točko pobiranja in boste robot preko ovire odpeljali na drugo stran do točke, kjer boste robota lahko naravnost navzdol premaknili v točno določeno polje, kjer bo robot odložil valj. Ko bo robot odložil valj, ga boste ponovno premaknili nad ta valj in kasneje preko ovire in boste pobrali naslednji valj na vaši mizi. To zaporedje uporabite trikrat in boste imeli pravilno napisan program. Robota nato po koncu odlaganja zadnjega valja premaknite nad tisti določen valj in ga z naslednjo točko pošljite v začetno pozicijo.

3.11 VAJA 8B: POBIRANJE IN ODLAGANJE TREH VALJEV S POMOČJO UKAZA OFFSET

Ustvarite program, v katerem se bo robotska roka premaknila iz ene strani in bo prenesla valj na drugo stran. Program je potrebno začeti v domači poziciji robota. Vsak valj mora vsebovati točke za približanje valju, točko pobiranja, točko vračanja, točko nad oviro, točko, kjer se približa odlagalnemu mestu in točko, v kateri se vrne nad odlagalno mesto. Ne smete pa pozabiti tudi programa za odpiranje in zapiranje prijemala, ki ste ga že uporabili v prejšnjem programu.

Slika 119: Robot v domači poziciji



Ustvarite nov Work Object in v njem shranite točko, v kateri prijemalo valj pobere. Vse ostale točke pa boste ustvarili z zamikom začetne točke.

Slika 120: Vrednosti za offset program za 3 valje

```
443     CONST num yPlace       := 240.33;
444     CONST num zAppPick     := 130;
445     CONST num zAppPlace    := 130;
446     CONST num zObstacle    := 250;
447     CONST num yObstacle    := 120;
448     CONST num zPickUp      := 80;
449     CONST num zPlaceUp     := 80;
450
451     CONST num dxValji{3}   := [0, 53, 106];
452     CONST num xPlace       := -300;
453
454     VAR num i;
455     VAR num dx;
```

Slika 121: Program za 3 valje z offset računanjem

```
! Program za racunanje premika za 3 valje
PROC p_Valj_3()
        ! Zacetna pozicija
        MoveAbsJ jt_Home,v1000,z100,t_Prijemalo\WObj:=Valj_3;
    FOR i FROM 1 TO 3 DO
        dx := dxValji{i};
        !PickAbove
        MoveJ Offs(pPick_Ref, dx + xPlace, 0, zAppPick), v300, z50, t_Prijemalo \WObj:=Valj_3;
        !Pick
        MoveL Offs(pPick_Ref, dx + xPlace, 0, 0), v300, fine, t_Prijemalo \WObj:=Valj_3;
        !Zapre prijemalo
        closeGripper;
        WaitTime 0.5;
        !PickUp
        MoveL Offs(pPick_Ref, dx + xPlace, 0, zPickUp), v200, z50, t_Prijemalo \WObj:=Valj_3;
        !OverObsatcle
        MoveJ Offs(pPick_Ref, dx + xPlace, yObstacle, zObstacle), v200, z50, t_Prijemalo \WObj:=Valj_3;
        !PlaceAbove
        MoveJ Offs(pPick_Ref, dx + xPlace, yPlace, zAppPlace), v200, z50, t_Prijemalo \WObj:=Valj_3;
        !Place
        MoveL Offs(pPick_Ref, dx + xPlace, yPlace, 0), v200, fine, t_Prijemalo \WObj:=Valj_3;
        !Odpre prijemalo
        openGripper;
        WaitTime 0.5;
        !PlaceUp
        MoveL Offs(pPick_Ref, dx + xPlace, yPlace, zPlaceUp), v300, z50, t_Prijemalo \WObj:=Valj_3;
        !OverObsatcle
        MoveJ Offs(pPick_Ref, dx + xPlace, yObstacle, zObstacle), v300, z50, t_Prijemalo \WObj:=Valj_3;
    ENDFOR
        ! Zacetna pozicija
        MoveAbsJ jt_Home,v1000,z100,t_Prijemalo\WObj:=Valj_3;
ENDPROC
```

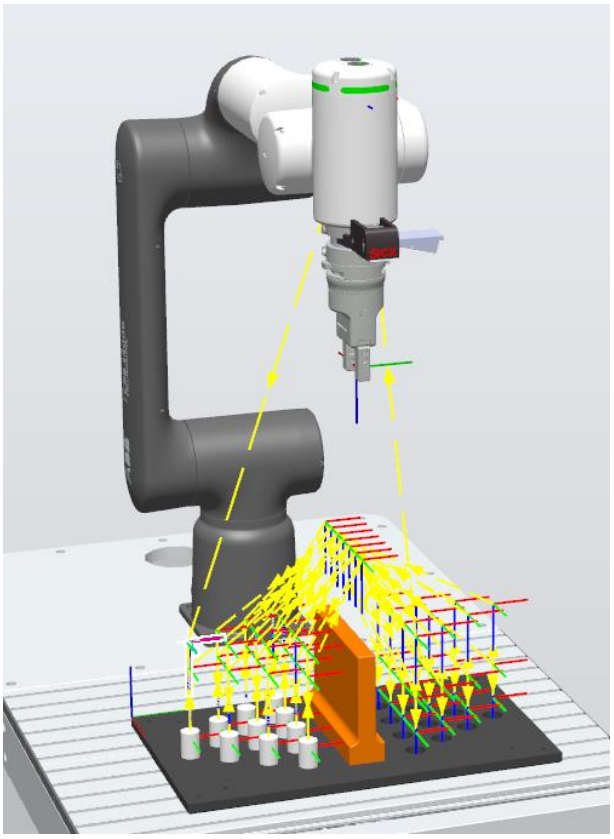
Offset računanje je drugačen način programiranja, ki je lahko uporabljena na robotu. Na začetku je potrebno zapisati vse vrednosti, ki bodo potem v enačbah, da bo robot naredil program pravilno. V tem programu je tudi for zanka, ki se začne s for in konča z endfor. Program se ponovi trikrat, saj je po for zanki v isti vrstici zapisano FROM 1 TO 3 DO. Programiranje je načeloma enako, premiki so enaki, vendar se razlikuje, kaj je zapisano po delu, kjer se določi ali bo joint gib ali linear gib. Vse, kar je zapisano v oklepajih, ima svoje vrednosti, vendar je razlika ta, da so tokrat zapisane besede namesto številk, ki bi veljale za koordinate. Te besede, kot je razvidno v sliki 58, imajo svoje vrednosti, zato je to tudi pregledno. Če robot ne naredi pravilno programa, je lahko napaka v vrednostih, če ne vemo, kje bi se moral robot ustaviti, da bi lahko pobral valj iz luknje v plošči. Tukaj je tudi spremenjena hitrost. Ko ima robot valj v prijemalu, je hitrost zmanjšana na v200, ko ga nima, je povečana na v300. Robot se bo premaknil v home pozicijo, takrat ko se bo for zanka ponovila trikrat. Po premiku v home pozicijo pa se bo zaključil celoten program.

Največje težave pri tem programu so lahko nerazumevanje programa, neznanje zapisa pravilnih vrednosti, ki so konstantne, nepravilen izračun. Ta program je težji za razumevanje, vendar je pa krajši od tistega, ki se naredi s točkami.

3.12 VAJA 9A: POBIRANJE IN ODLAGANJE 10 VALJEV PREKO OVIRE

Ustvarite program, v katerem bo robotska roka prenesla deset valjev iz ene strani na drugo stran. Potek programiranja je popolnoma enak kot pri programu za tri točke, vendar s tistim programom, ki je bolj klasičen s točkami.

Slika 122: Pot, ki jo robot opravi



Ustvarite toliko točk, da boste s tem prenesli vseh deset valjev na drugo stran. Ustvariti je potrebno nov Work Object, točke programa, pot gibanja in program.

Uporabite enak program za zapiranje in odpiranje prijemala kot ste ga že uporabili.

Slika 123: Začetek programa za 10 valjev

```
! Zacetek programa za 10 valjev
PROC Pick_and_place_10()
    ! Zacetna pozicija
    MoveAbsJ jt_Home,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Premik nad valj 1
    MoveJ d1_1_Pick_Above,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Spust do valja
    MoveL d1_2_Pick,v1000,fine,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Zapri prijemalo
    closeGripper;
    WaitTime 1;
    ! Dvig valja
    MoveL d1_3_Pick_Up,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Pomik nad oviro
    MoveJ d1_4_Over_Obstacle,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Pomik nad odlagalno mesto
    MoveJ d1_5_Place_Above,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Spust valja
    MoveL d1_6_Place,v1000,fine,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Odpri prijemalo
    openGripper;
    WaitTime 1;
    ! Dvig prijemala
    MoveL d1_7_Place_Up,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Pomik nad oviro
    MoveJ d1_8_Over_Obstacle,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Premik nad valj 2
    MoveJ d2_1_Pick_Above,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Spust do valja
    MoveL d2_2_Pick,v1000,fine,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Zapri prijemalo
    closeGripper;
    WaitTime 1;
    ! Dvig valja
    MoveL d2_3_Pick_Up,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Pomik nad oviro
    MoveJ d2_4_Over_Obstacle,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Pomik nad odlagalno mesto
    MoveJ d2_5_Place_Above,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Spust valja
    MoveL d2_6_Place,v1000,fine,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Odpri prijemalo
    openGripper;
    WaitTime 1;
    ! Dvig prijemala
    MoveL d2_7_Place_Up,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
```

Slika 124: Nadaljevanje programa za 10 valjev

```
! Dvig prijemala
MoveL d2_7_Place_Up,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
! Pomik nad oviro
MoveJ d2_8_Over_Obstacle,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
! Premik nad valj 3
MoveJ d3_1_Pick_Above,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
! Spust do valja
MoveL d3_2_Pick,v1000,fine,t_Prijemalo\WObj:=wobj_plosca_10;
! Zapri prijemalo
closeGripper;
WaitTime 1;
! Dvig valja
MoveL d3_3_Pick_Up,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
! Pomik nad oviro
MoveJ d3_4_Over_Obstacle,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
! Pomik nad odlagalno mesto
MoveJ d3_5_Place_Above,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
! Spust valja
MoveL d3_6_Place,v1000,fine,t_Prijemalo\WObj:=wobj_plosca_10;
! Odpri prijemalo
openGripper;
WaitTime 1;
! Dvig prijemala
MoveL d3_7_Place_Up,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
! Pomik nad oviro
MoveJ d3_8_Over_Obstacle,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
! Premik nad valj 4
MoveJ d4_1_Pick_Above,v1000,fine,t_Prijemalo\WObj:=wobj_plosca_10;
! Spust do valja
MoveL d4_2_Pick, v1000, fine, t_Prijemalo\WObj:=wobj_plosca_10;
! Zapri prijemalo
closeGripper;
WaitTime 1;
! Dvig valja
MoveL d4_3_Pick_Up,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
! Pomik nad oviro
MoveJ d4_4_Over_Obstacle,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
! Pomik nad odlagalno mesto
MoveJ d4_5_Place_Above,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
! Spust valja
MoveL d4_6_Place, v1000, z100, t_Prijemalo\WObj:=wobj_plosca_10;
! Odpri prijemalo
openGripper;
WaitTime 1;
! Dvig prijemala
MoveL d4_7_Place_Up,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
! Pomik nad oviro
MoveJ d4_8_Over_Obstacle,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
```

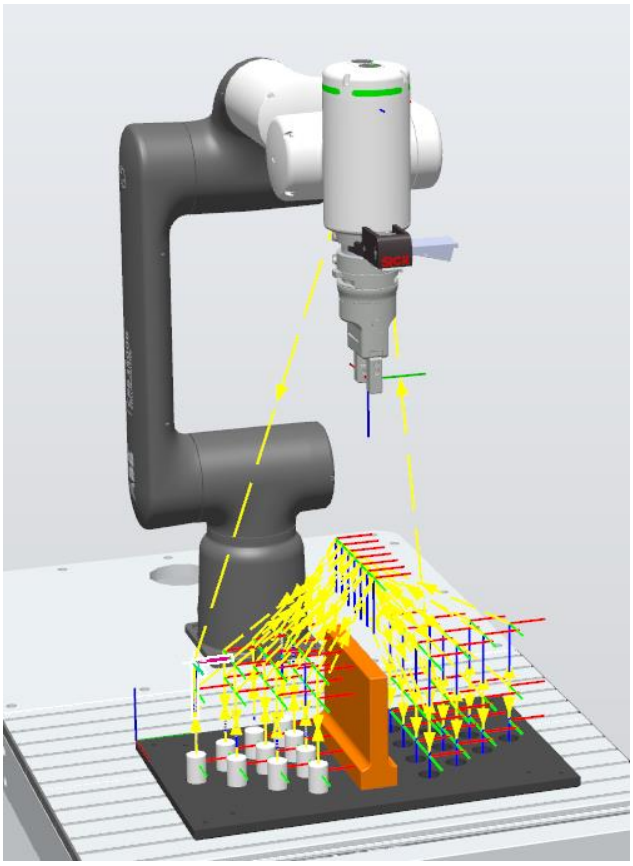
Ta program z 10 valji je zelo podoben programu s 3 ali z 1 valjem.
Točke lahko malo razdelimo s komentarji, ki so postavljeni po vsaki vrstici in veljajo za vrstico pod njimi, ali nad njimi. Pri tem je pa tudi pomembno, da je predvsem pri točkah, kjer se pobira in odlaga, pravilno zapisan fine, saj če je v programu z, potem bo robot prehitro spustil valj, saj bo program šel do close ali openGripper in tako bo program napačno deloval. V nobeni drugi točki ni potrebno imeti zapisa fine razen v točkah, kjer pobere in odloži valj.

Za nadgradnjo programa lahko zmanjšate hitrost takrat, ko ima robot prijemalo polno z valjem.

3.13 VAJA 9B: PRENOS 10 VALJEV PREKO OVIRE IN NATO NAZAJ NA PRVOTNO MESTO

Ustvarite program, kjer boste dodelali program tako, da bo robot iz ene strani prenesel valje na drugo stran, kasneje pa te iste valje prenesel na začetno stran v njihovo začetno pozicijo.

Slika 125: Pot, ki jo bo robot opravil



Pri tem programu lahko uporabite isti WorkObject, vendar samo spremenite ime programa tako, da boste videli razliko med obema programoma, saj si bosta zelo podobna.

Slika 126: Program za 10 valjev v obe smeri

```
PROC Pick_and_place_10_1()
! Zacetna pozicija
    MoveAbsJ jt_Home,v1000,z100,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Premik nad valj 1
    MoveJ d1_1_Pick_Above,v300,z100,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Spust do valja
    MoveL d1_2_Pick,v300,fine,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Zapri prijemalo
    closeGripper;
    WaitTime 1;
    ! Dvig valja
    MoveL d1_3_Pick_Up,v200,z100,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Pomik nad oviro
    MoveJ d1_4_Over_Obstacle,v200,z100,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Pomik nad odlagalno mesto
    MoveJ d1_5_Place_Above,v200,z100,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Spust valja
    MoveL d1_6_Place,v200,fine,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Odpri prijemalo
    openGripper;
    WaitTime 1;
    ! Dvig prijemala
    MoveL d1_7_Place_Up,v300,z100,t_Prijemalo\WObj:=wobj_plosca_10;
    ! Pomik nad oviro
    MoveJ d1_8_Over_Obstacle,v300,z100,t_Prijemalo\WObj:=wobj_plosca_10;
```

Ta program je enak programu prenos 10 valjev preko ovire. Razlika je samo v tem, da ko robot prenese še zadnji valj preko ovire, gre na prvi valj na strani, kamor je prenesel valje in jih odnese nazaj na začetno točko, kjer jih je na začetku dvignil. Razlika je samo v tem, da se vrstice skoraj enako ponavljajo. Zamenja se samo vrstni red, če gredo točke pred tem od 1 – 8, gredo sedaj od 8-1. Zamenjata se tudi pod programa openGripper in closeGripper. S tem se doseže, da bo robot, ko bo že enkrat prenesel 10 valjev iz desne na levo stran, kasneje valje prenesel iz leve na desno stran.

Slika 127: Del programa, ki prikazuje premikanje valjev nazaj na prvotno stran

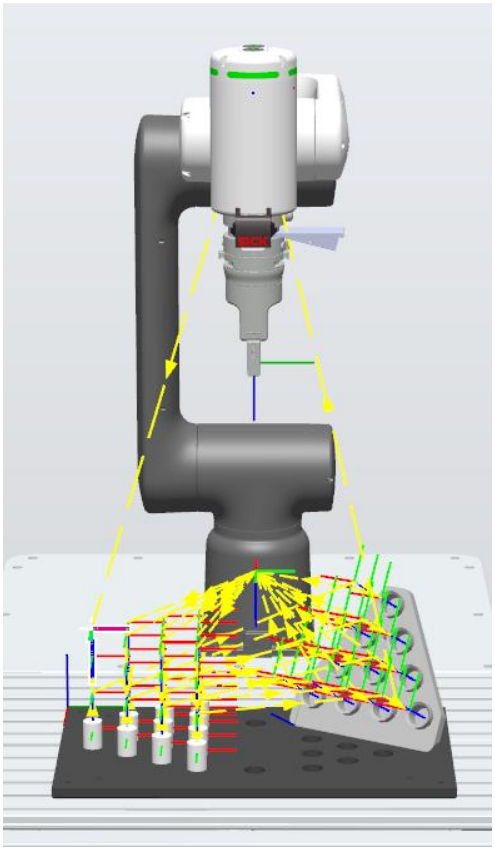
```
! Premik nad valj 1
MoveJ d1_7_Place_Up,v300,z100,t_Prijemalo\WObj:=wobj_plosca_10;
! Spust do valja
MoveL d1_6_Place,v300,fine,t_Prijemalo\WObj:=wobj_plosca_10;
! Zapri prijemalo
closeGripper;
WaitTime 1;
! Dvig valja
MoveL d1_5_Place_Above,v200,z100,t_Prijemalo\WObj:=wobj_plosca_10;
! Pomik nad oviro
MoveJ d1_4_Over_Obstacle,v200,z100,t_Prijemalo\WObj:=wobj_plosca_10;
! Pomik nad odlagalno mesto
MoveJ d1_3_Pick_Up,v200,z100,t_Prijemalo\WObj:=wobj_plosca_10;
! Spust valja
MoveL d1_2_Pick,v200,fine,t_Prijemalo\WObj:=wobj_plosca_10;
! Odpri prijemalo
openGripper;
WaitTime 1;
! Dvig prijemala
MoveL d1_1_Pick_Above,v300,z100,t_Prijemalo\WObj:=wobj_plosca_10;
! Pomik nad oviro
MoveJ d1_8_Over_Obstacle,v300,z100,t_Prijemalo\WObj:=wobj_plosca_10;
```

Te strani se gledajo tako, da se postavimo za robota in ga opazujemo od zadaj, tako določimo, katera je desna in katera je leva stran ovire.
Pri tem programu, ko se ponavlja toliko število točk, je pomembno, da se ne zameša, katere točke imajo ukaz z ali fine. V primeru napake in nepravilnega zapisa bo robot prehitro odprl prijemalo.
Kar se tiče programiranja, je isti postopek kot pri programiranju za 1 ali pa 3 valje, postopek je isti kot pri teh valjih, vendar je to daljši program z več valji.

3.14 VAJA 10A: ODLAGANJE 10 VALJEV IZ ENE STRANI NA POŠEVNINO NA DRUGI STRANI

Ustvarite program, kjer bo robotska roka prenesla deset valjev na drugo stran na poševnino. Program boste začeli in končali v home poziciji robota. Program mora vsebovati točke, kjer se robot približa valju, točko pobiranja, točko dviga, točko nad sredino delovne plošče, točko približanja odlagalnemu mestu in točko vračanja. Te točke ustvarite iz desetih valjev, dodajte še zapiranje in odpiranje prijemala ter uredite pravilne gibe.

Slika 128: Pot, ki jo bo robot opravil



Ustvarite nov Workobject za pobiranje valjev na delovni plošči, za odlaganje valjev na poševnini in ustvarite nov program, pot gibanja in program.

Slika 129: Program za 10 valjev na poševnino

```
! Zacetek programa za 10 valjev na posevnino
PROC Pick_and_place_10_posevnina()
   ! Zacetna pozicija
    MoveAbsJ jt_Home,v1000,z100,t_Prijemalo\WObj:=wobj_posevna_plosca;
    ! Premik nad valj 1
    MoveJ k1_1_Pick_Above, v200, z100, t_Prijemalo\WObj:=wobj_plosca_posevnina;
    ! Spust do valja
    MoveL k1_2_Pick, v200, fine, t_Prijemalo\WObj:=wobj_plosca_posevnina;
    ! Zapri prijemalo
    closeGripper;
    WaitTime 1;
    ! Dvig valja
    MoveL k1_3_Pick_Up, v100, z100, t_Prijemalo\WObj:=wobj_plosca_posevnina;
    ! Pomik nad oviro
    MoveJ k1_4_Position, v100, z100, t_Prijemalo\WObj:=wobj_plosca_posevnina;
    ! Pomik nad odlagalno mesto
    MoveJ k1_5_Place_Above, v100, z100, t_Prijemalo\WObj:=wobj_posevna_plosca;
    ! Spust valja
    MoveL k1_6_Place, v100, fine, t_Prijemalo\WObj:=wobj_posevna_plosca;
    ! Odpri prijemalo
    openGripper;
    WaitTime 1;
    ! Dvig prijemala
    MoveL k1_7_Place_Up, v200, z100, t_Prijemalo\WObj:=wobj_posevna_plosca;
    ! Pomik nad oviro
    MoveJ k1_8_Position, v200, z100, t_Prijemalo\WObj:=wobj_posevna_plosca;
```

Ta program je med bolj zapletenimi, saj se mora robota sprogramirati zelo točno glede na poševnino, da ne bo robot šel predaleč pri poševnini in bo prišlo do udarca. Prve štiri točke, ki so home pozicija, approach (približati) točka, točka, kjer pobere valj in točka, ki je malo nad pobiranjem valja, so lahko enake kot pa točke na prejšnjih programih za pobiranje 10 valjev. Home točka se upošteva samo preden se pobere prvi valj.

Slika 130: Nadaljevanje programa

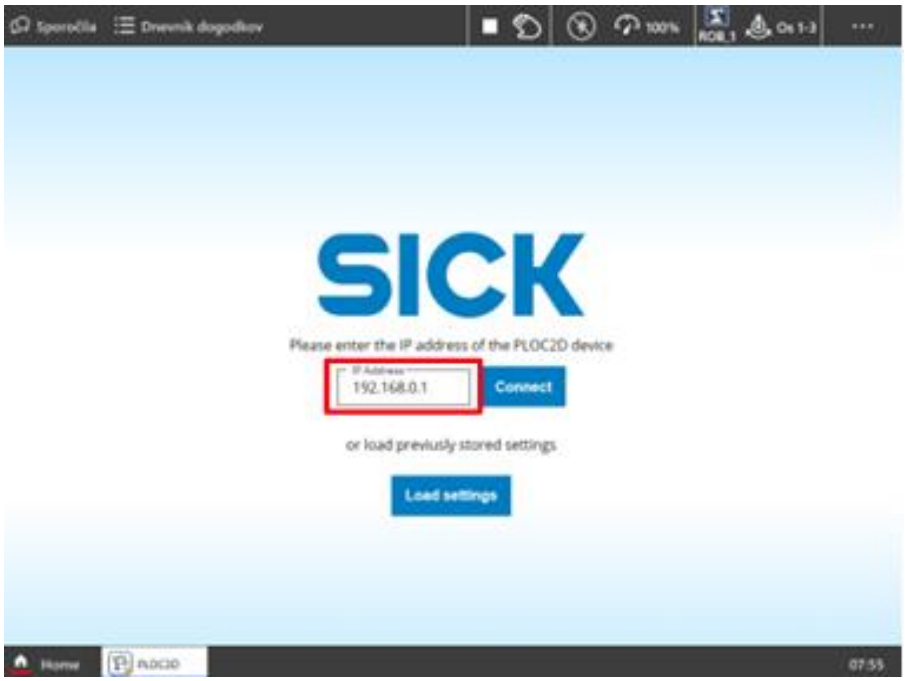
```
! Premik nad valj 2
MoveJ k2_1_Pick_Above, v200, z100, t_Prijemalo\WObj:=wobj_plosca_posevnina;
! Spust do valja
MoveL k2_2_Pick, v200, fine, t_Prijemalo\WObj:=wobj_plosca_posevnina;
! Zapri prijemalo
closeGripper;
WaitTime 1;
! Dvig valja
MoveL k2_3_Pick_Up, v100, z100, t_Prijemalo\WObj:=wobj_plosca_posevnina;
! Pomik nad oviro
MoveJ k2_4_Position, v100, z100, t_Prijemalo\WObj:=wobj_plosca_posevnina;
! Pomik nad odlagalno mesto
MoveJ k2_5_Place_Above, v100, z100, t_Prijemalo\WObj:=wobj_posevna_plosca;
! Spust valja
MoveL k2_6_Place, v100, fine, t_Prijemalo\WObj:=wobj_posevna_plosca;
! Odpri prijemalo
openGripper;
WaitTime 1;
! Dvig prijemala
MoveL k2_7_Place_Up, v200, z100, t_Prijemalo\WObj:=wobj_posevna_plosca;
! Pomik nad oviro
MoveJ k2_8_Position, v200, z100, t_Prijemalo\WObj:=wobj_posevna_plosca;
```

V tem programu na sredini ni ovire, zato se robotska roka ne rabi premakniti tako visoko kot je to potrebno pri programih, kjer mora robot iti nad oviro. Premikanje robota tik pred tem ko bo odložil valj v poševnino, je potrebno, da je počasno oz. da je hitrost znižana, saj se s tem doseže večja varnost.

3.15 VAJA 11: PREMIK ROBOTA S STROJNIM VIDOM IN POBIRANJE RAZLIČNIH OBLIK NA PLOŠČI

Ustvarite program, kjer boste ustvarili povezavo s kamero. Nato boste sprogramirali kamero in senzorja. Robotu boste nato napisali program zelo podoben, vendar boste uporabili stavek if.

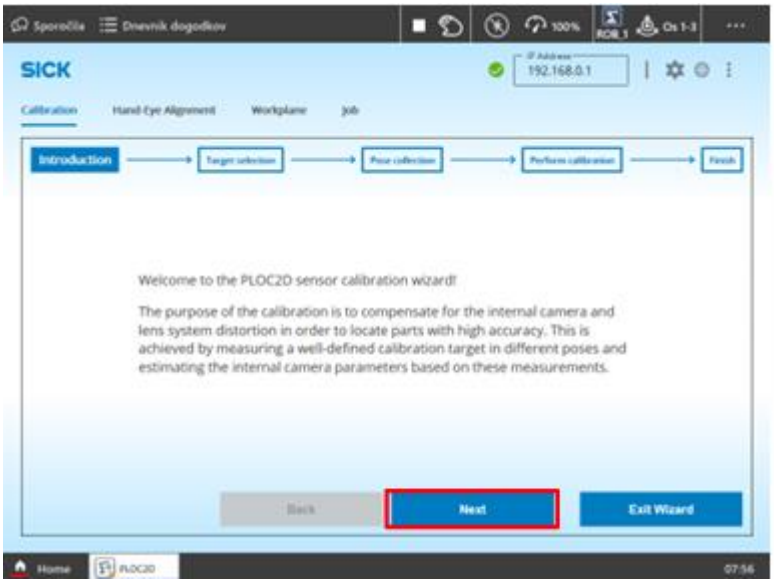
Slika 131: IP naslov kamere



Na začetku programiranja robota s strojnim vidom je potrebno slediti navodilom, ki sledijo bodo pomagale pospešiti učenje, kako se naredi strojni vid.
Prvotno je potrebno vpisati IP naslov kamere: 192.168.0.2. Ta IP naslov se napiše v okence, ki je označeno z rdečim kvadratom

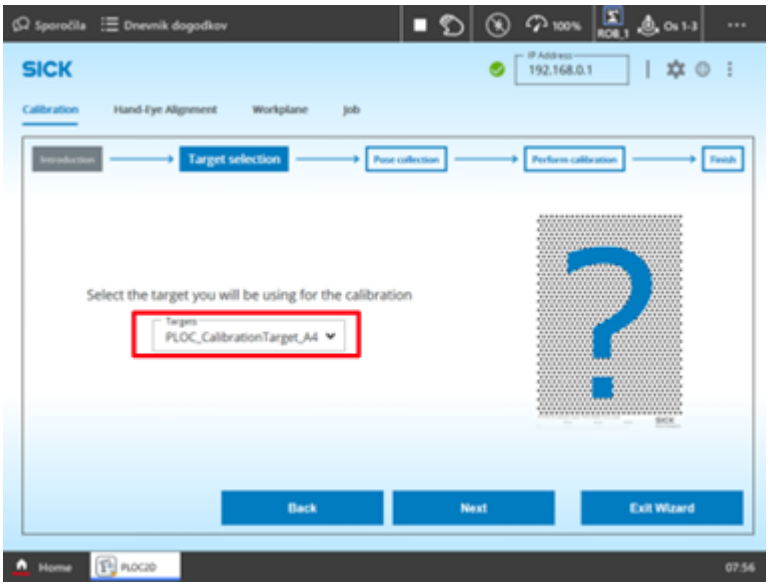
Nato se sledi navodilom, ki so vam napisane podrobno in razločno v nadaljevanju.

Slika 132: Callibration



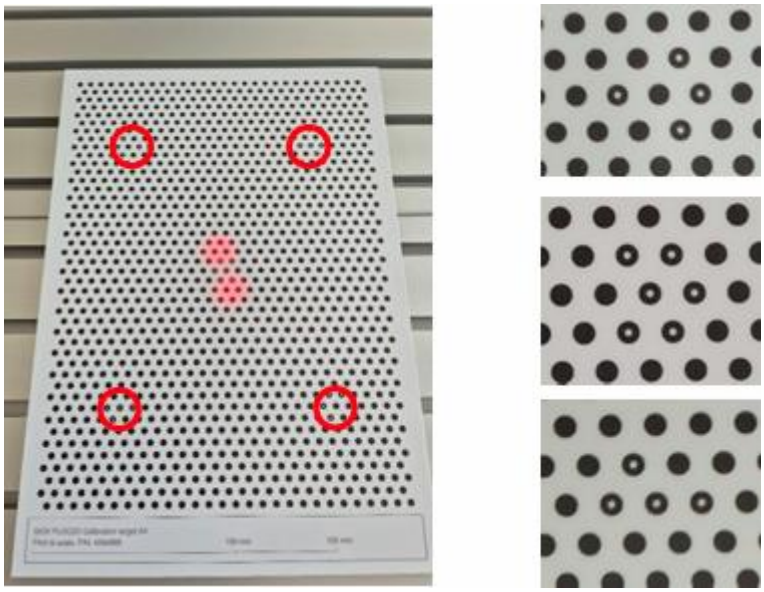
Z rutino boste izvedli kompenzacijo popačenosti slike tako, da boste pritisnili tipko NEXT.

Slika 133: Izbira plošče



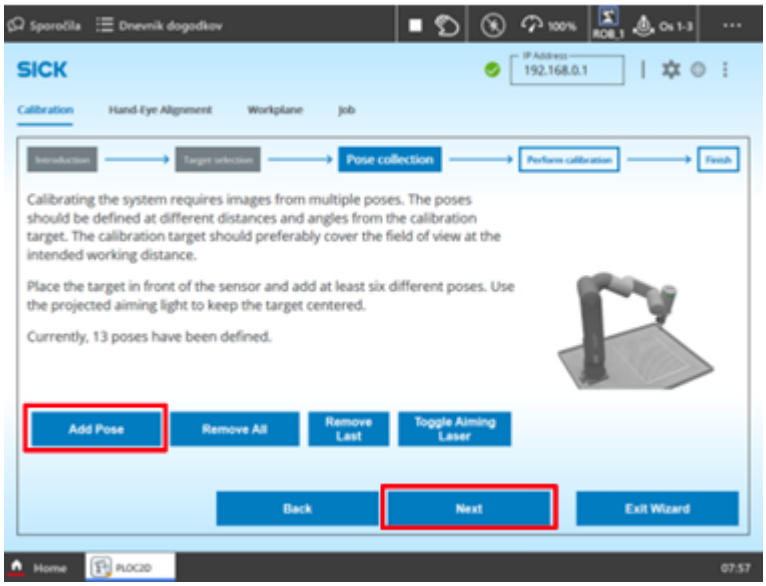
Tukaj je potrebno izbrati ustrezno ploščo, v vašem primeru je to A4.

Slika 134: Točke, kjer bo svetil laser



Robota je potrebno postaviti na položaj tako, da laser sveti točno v sredinski kalibracijski vzorec. Na plošči je 5 kalibracijskih vzorcev.

Slika 135: Dodajanje kalibracijskih pozicij



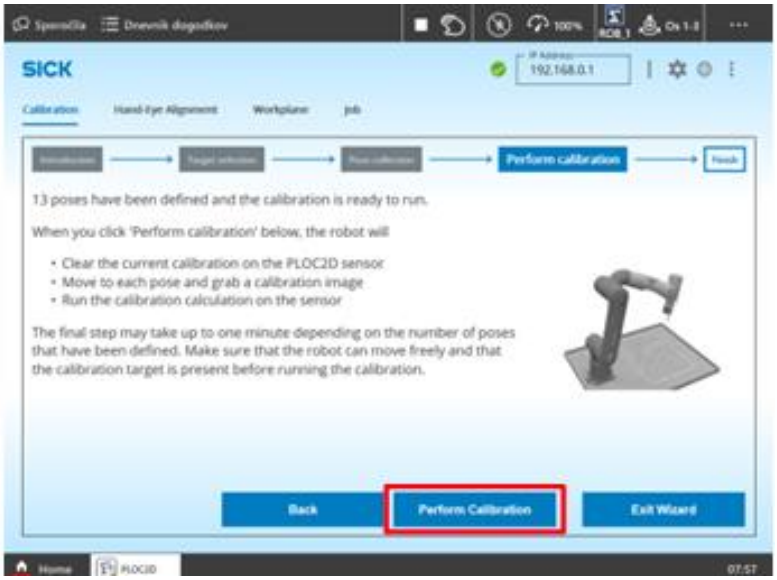
Dodati boste morali 6 kalibracijskih pozicij. V vsaki poziciji mora laser svetiti v enega izmed kalibracijskih vzorcev.

Slika 136: Robot v različni poziciji

Slika 137: Robot v različni poziciji

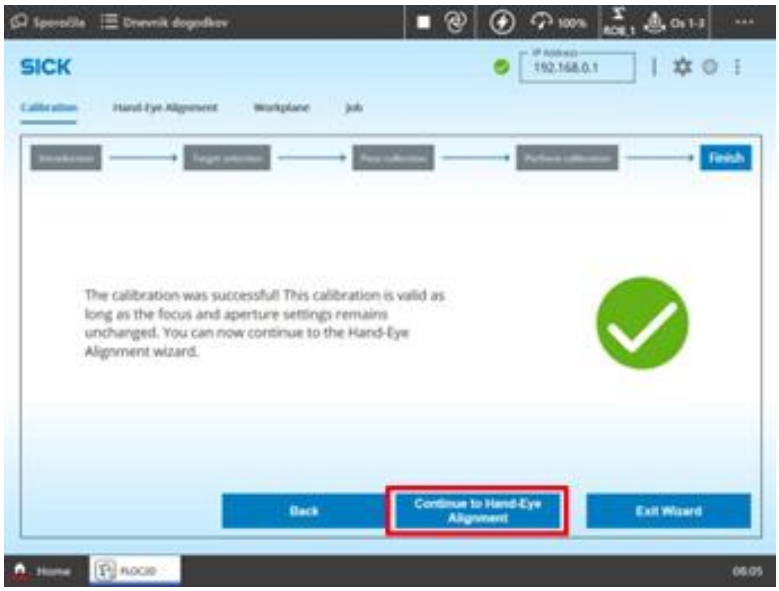
Orientacija robota (kamere) naj se med pozicijami spreminja kot je razvidno iz slik. Kamera naj bo oddaljena 300 mm.

Slika 138: Zagon kalibracije



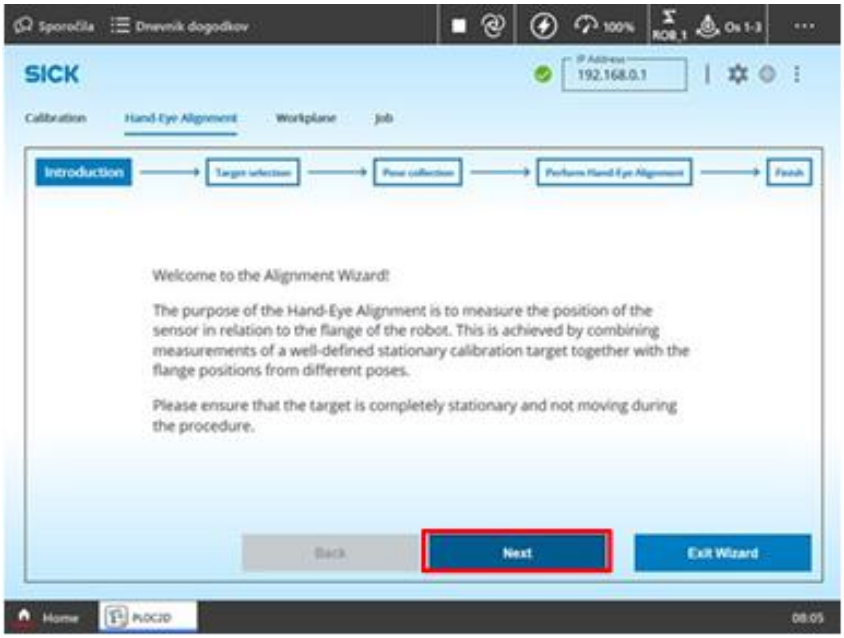
Nato zaženete rutino za kalibracijo. Med postopkom morajo biti vključeni motorji. Zagnali boste kalibracijo tako, da boste pritisnili na tipko, ki je označena z rdečim kvadratom.

Slika 139: Nadaljevanje na Hand-Eye Alignment



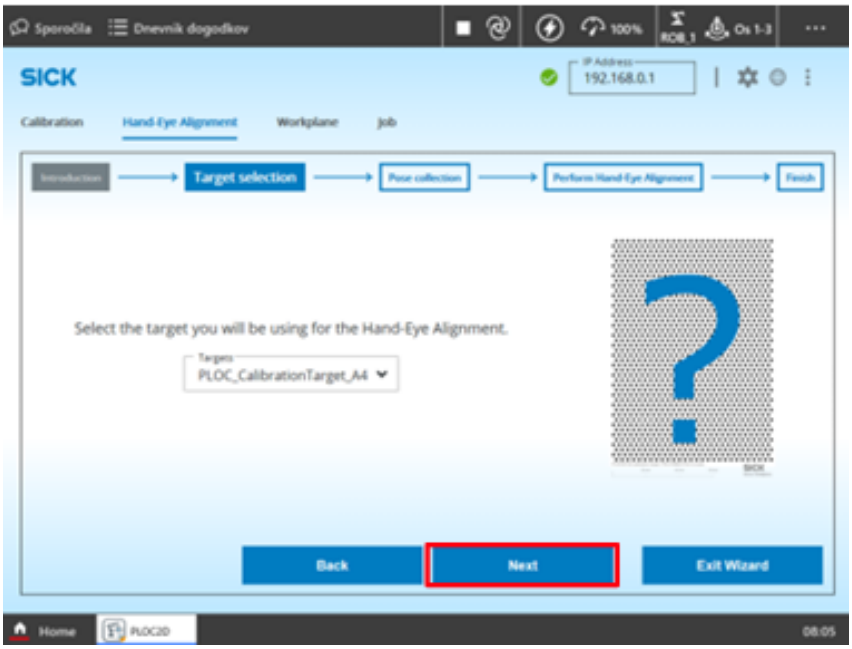
Nadaljujete na postopek določitve orodja kamere tako, da pritisnete na tipko, ki je označena z rdečim kvadratom.

Slika 140: Začetek Hand-Eye Alignment



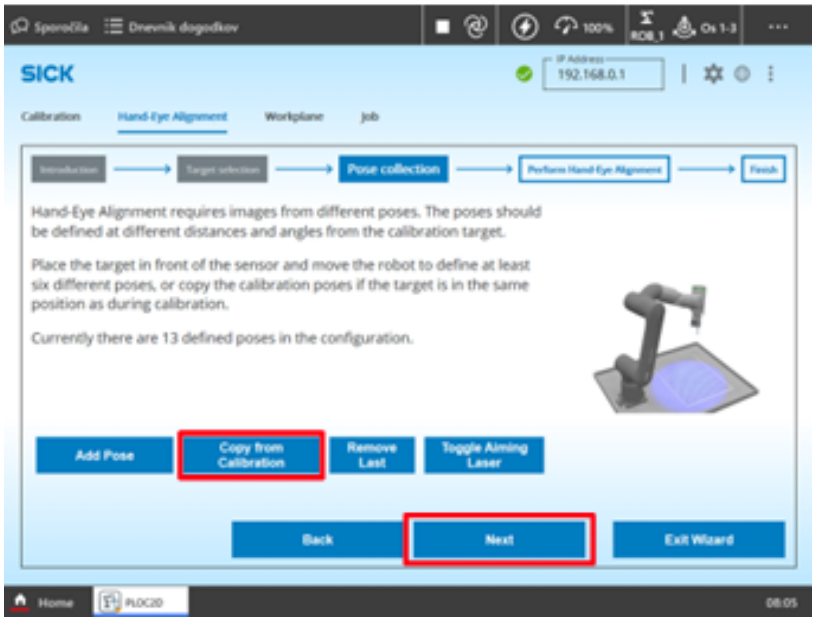
Z izvedbo rutine boste določili pozicijo slikovnega zaznavala v koordinatnem sistemu robota.

Slika 141: Ponovna izbira plošče



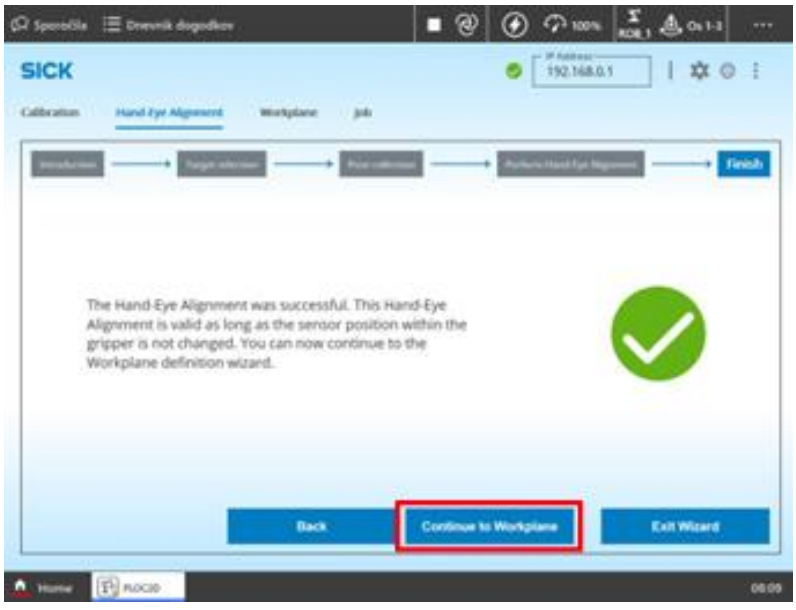
Ponovno izberete ustrezno kalibracijsko ploščo, v vašem primeru bo to A4 plošča.

Slika 142: Kopiranje točk od Calibration



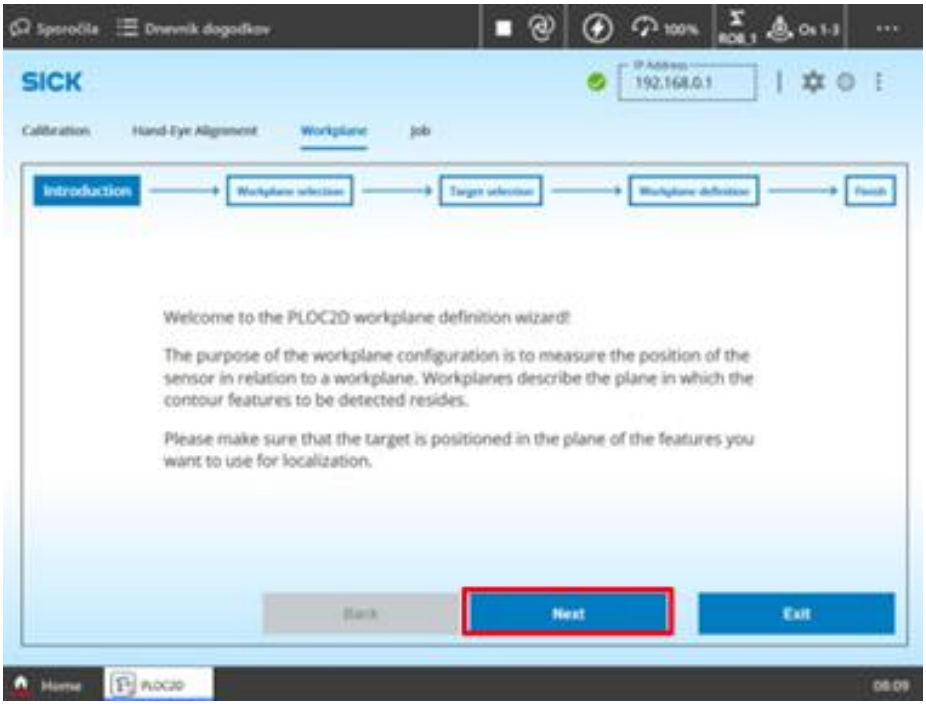
Dodate vsaj 6 poziciji, enako kot pri postopku kalibracije. Da boste dosegli enake točke, jih lahko samo skopirate in boste imeli enake točke. Pritisnite na tipko Add Pose, in lahko dodate 6 različnih pozicij in jih potrdite s Finish.

Slika 143: Nadaljevanje postopka



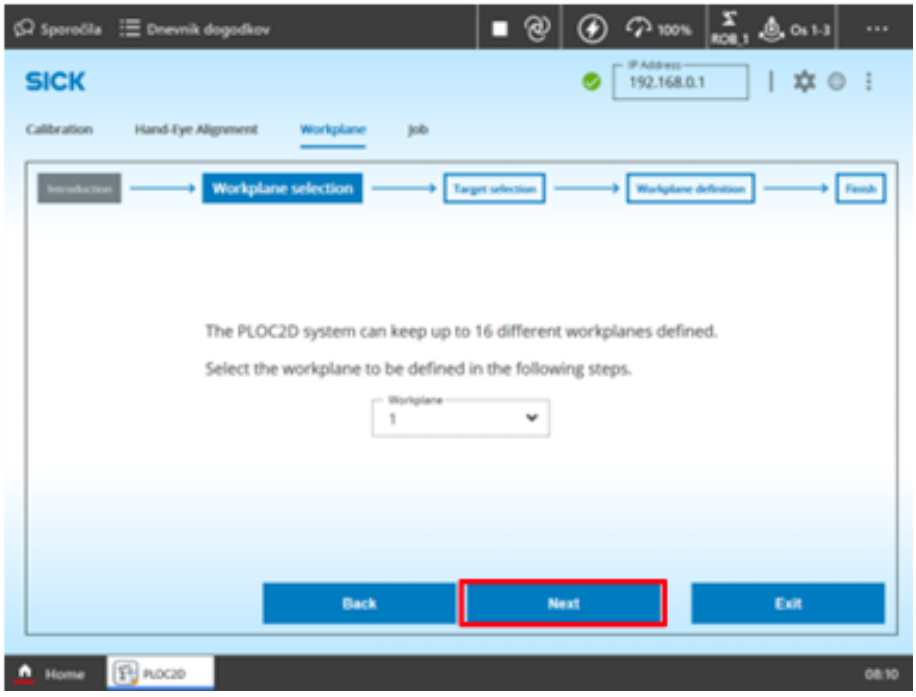
Nadaljujete postopek določitve prvega delovnega območja tako, da pritisnete na tipko, ki je označena z rdečim kvadratom.

Slika 144: Začetek workplane



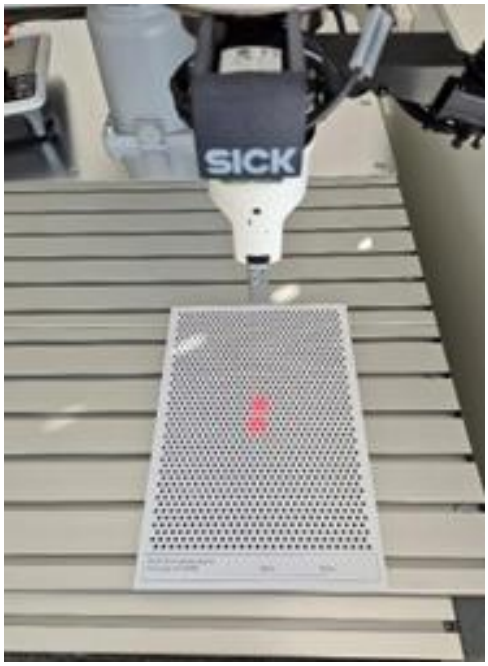
Delovno območje definira slikovno polje, v katerem boste definirali delovne rutine.

Slika 145: Število delovnega območja



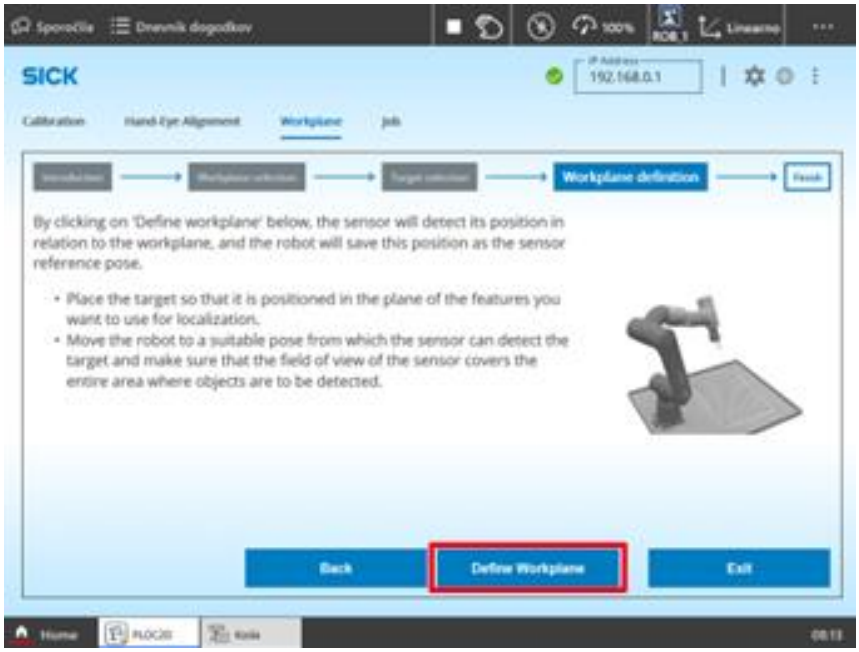
Izbrali boste številko delovnega območja, ki ga boste želeli definirati tako, da boste v kvadratek izbrali številko Workplace – a in nato potrdili s tipko Next, ki je označena z rdečim kvadratom.

Slika 146: Kalibracija plošče



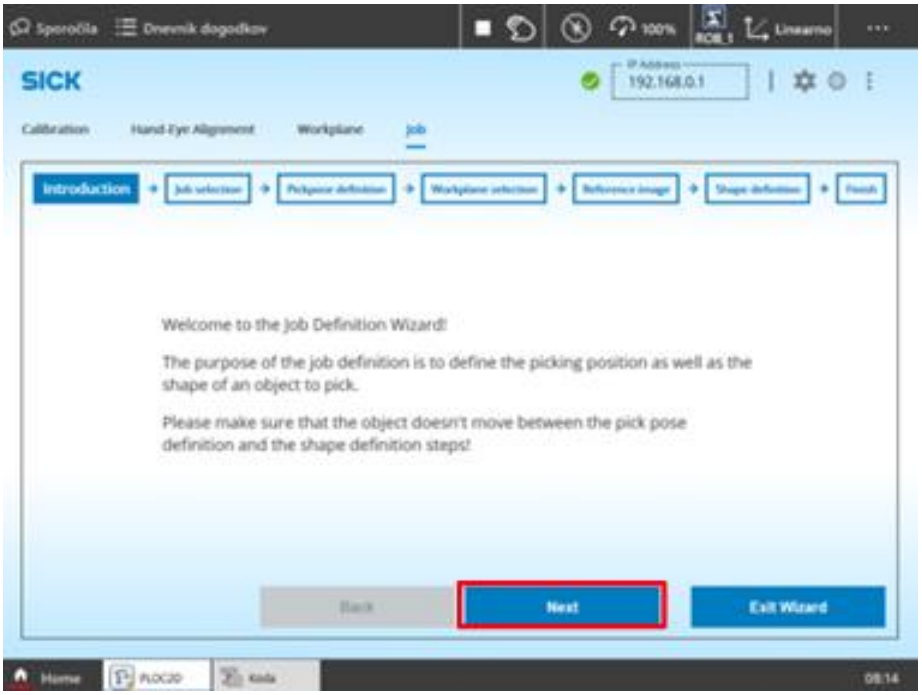
Za določitev delovnega območja je potrebna uporaba kalibracijske plošče.

Slika 147: Definirati workplane



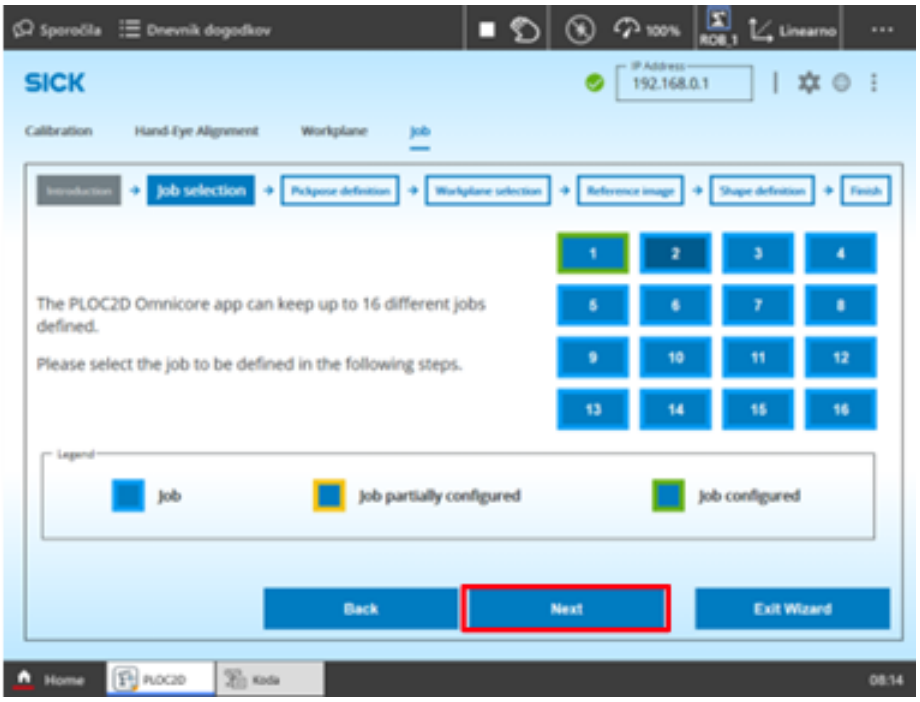
Robota je potrebno postaviti v pozicijo, da laser sveti točno v sredino kalibracijske plošče. Kamera naj bo usmerjena pravokotno na površino.

Slika 148: Nadaljevanje programiranja



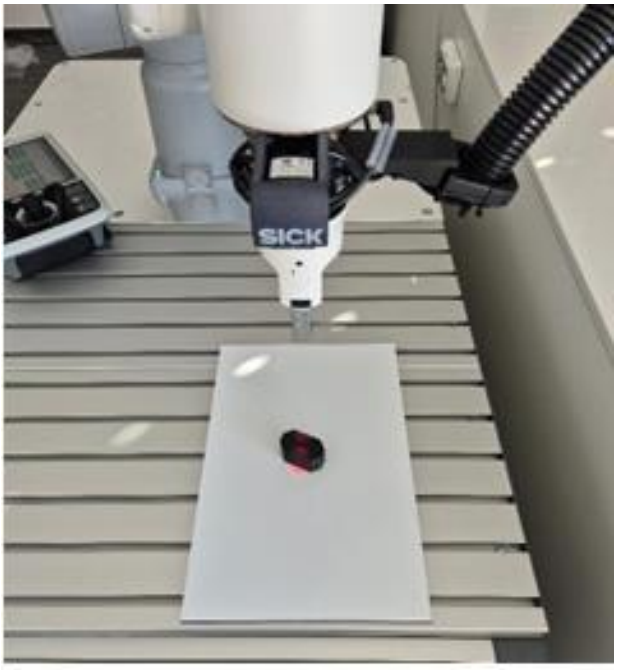
Z definicijo delovne rutine določite konturo, ki jo boste iskali v delovnem območju. V tem primeru, pritisnite tipko Next, ki je označena z rdečim kvadratkom.

Slika 149: Število delovne rutine



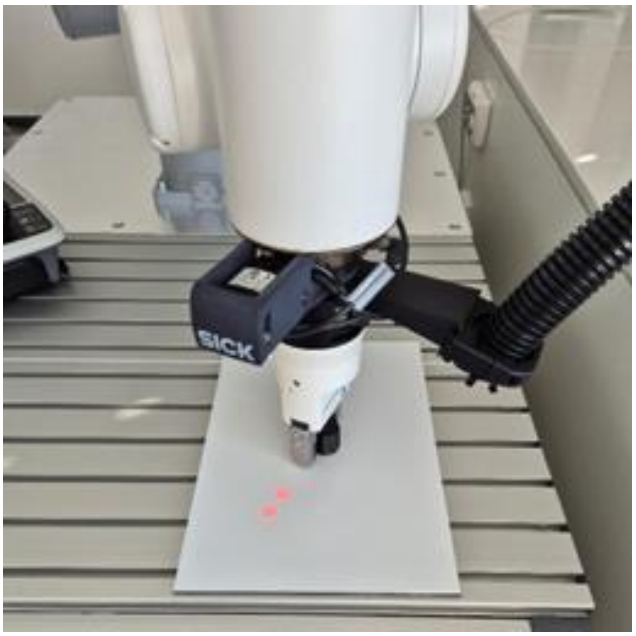
Izbrati je potrebno številko delovne rutine, ki jo želite definirati tako, da izberete številko na desni strani in nato pritisnete tipko Next.

Slika 150: Postavitev robota



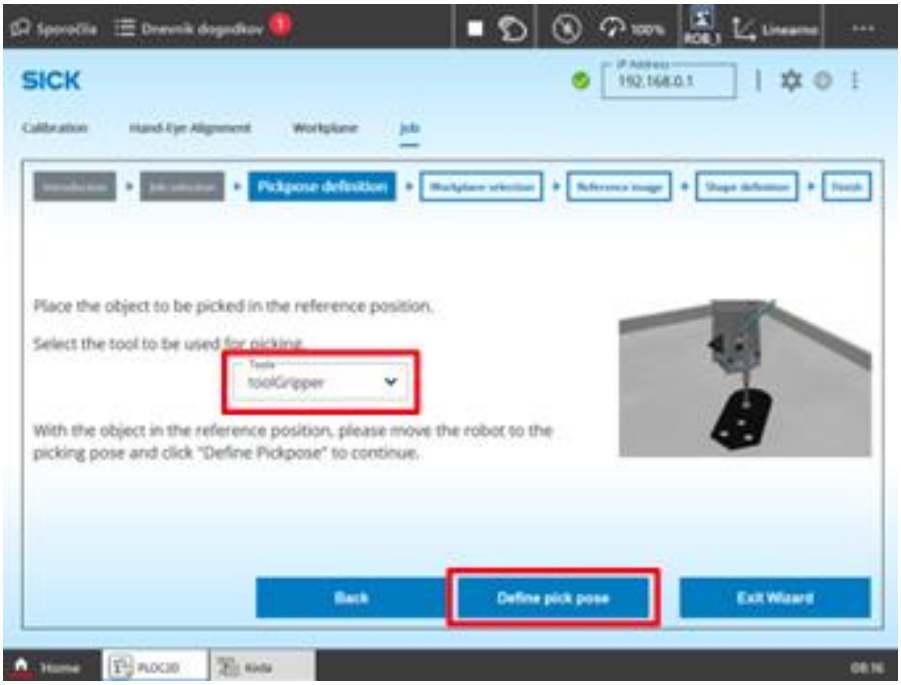
Robota postavite v pozicijo za slikanje delovnega območja.
Izdelek boste postavili točno pod kamero.

Slika 151: Postavitev robota



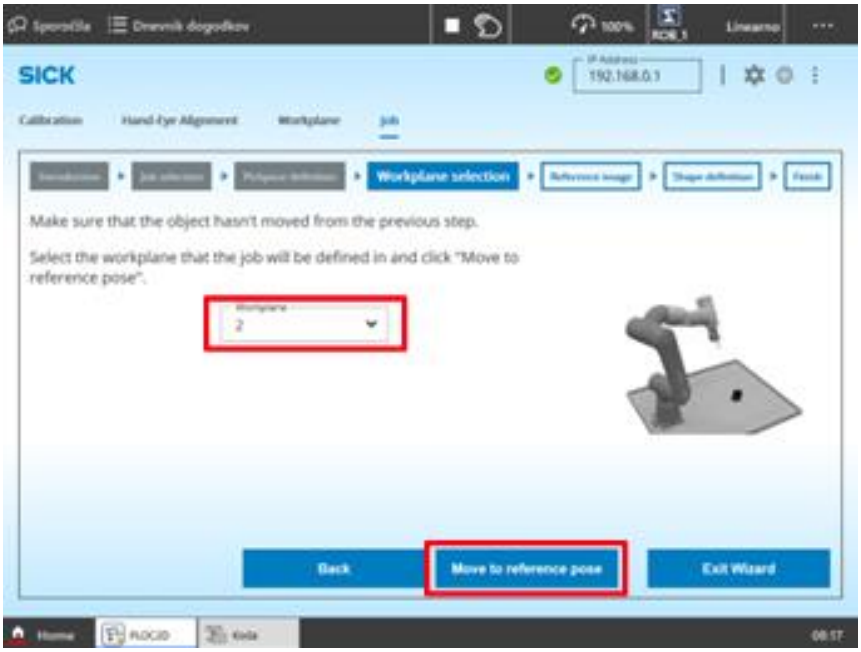
Nato boste robota peljali v pozicijo, kjer bo robot izdelek pobral.
Postavite ozadje, ki ga boste uporabljali.

Slika 152: Izbira orodja



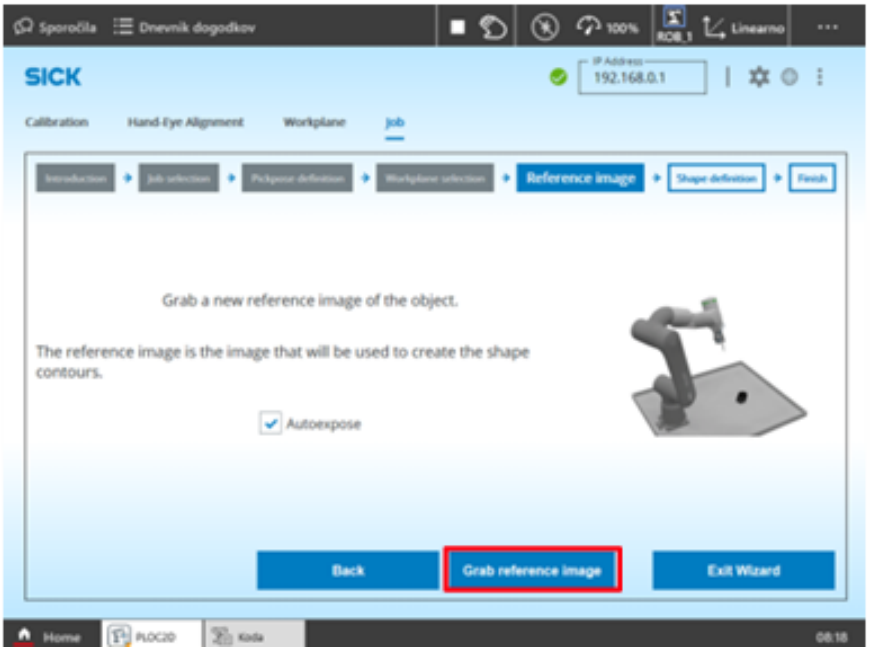
Izbrali boste ustrezno orodje v zavihku na sredini zaslona, nato pa pritisnili tipko Next.

Slika 153: Izbira delovnega območja



Izbrali boste ustrezno delovno območje. Nato boste pa potrdili premik pozicije za slikanje Izberite številko v sredinskem kvadratku, nato pritisnite tipko Move to reference pose, ki je označena z rdečim okvirjem.

Slika 154: Slika delovnega območja



Potrdili boste zajem slike, ki vsebuje izdelek.

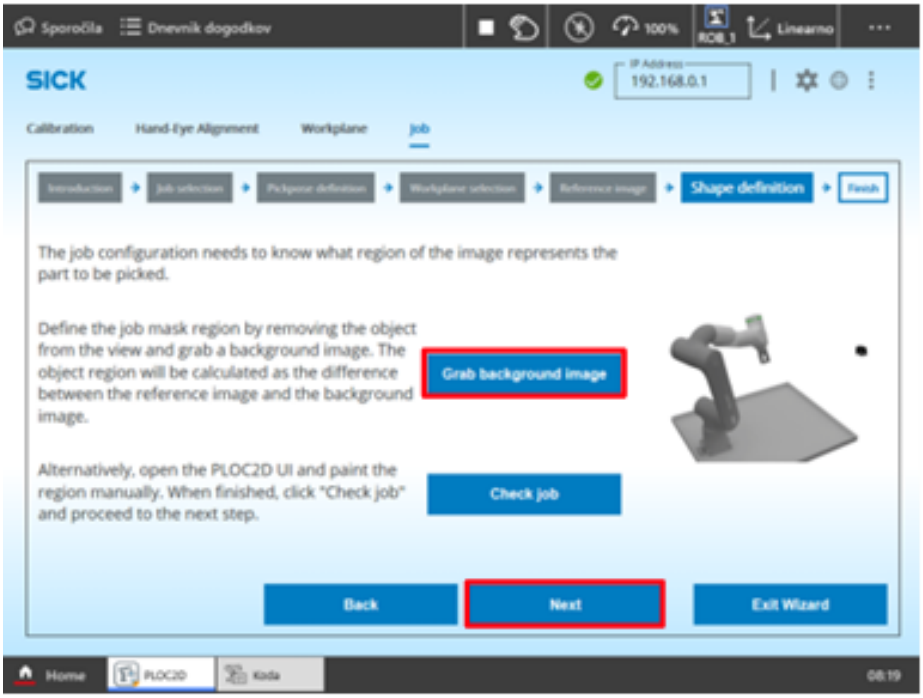
Slika 155: Pozicija robota za zajem slike

Robot je v poziciji zajema slike glede na izbrano območje

Slika 156: Odstranitev izdelka

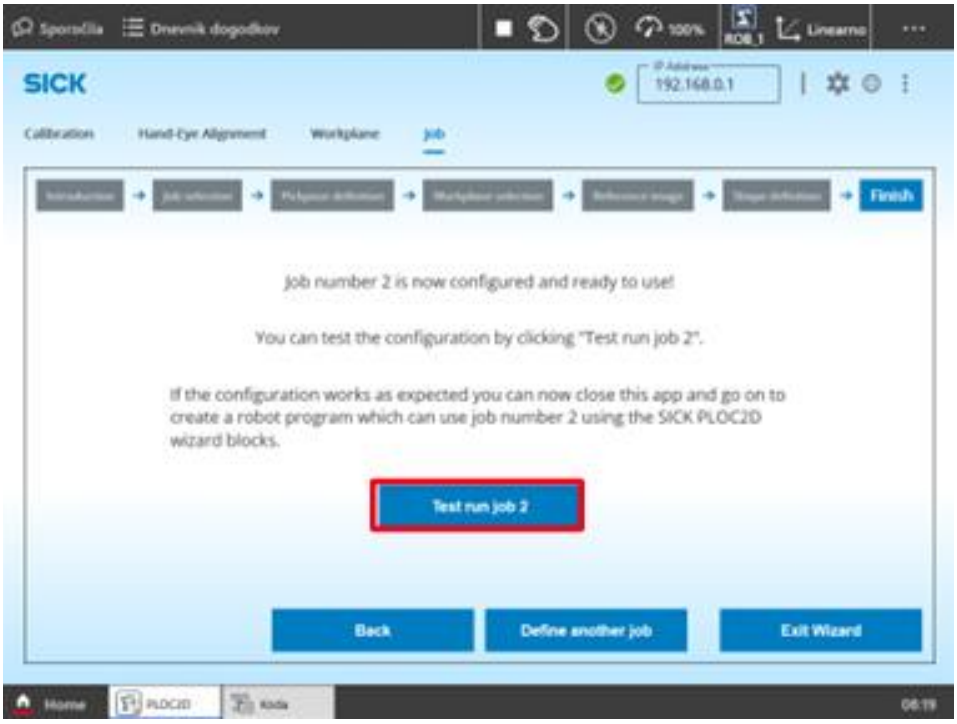
Nato boste odstranili izdelek iz plošče.

Slika 157: Nadaljevanje programa



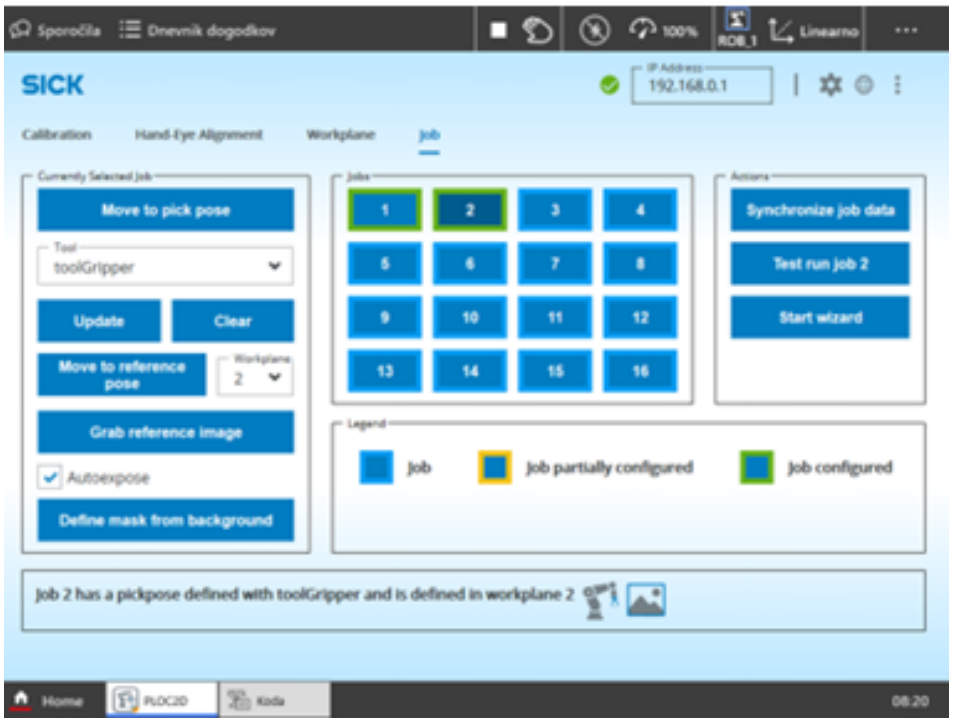
Potrdite zajem slike iz ozadja in nadaljujete. Najprej pritisnite gumb na sredini v rdečem okvirju, nato pa tipko Next.
Iz razlike med fotografijama bo program določil konturo izdelka.

Slika 158: Testiranje



Izdelek boste postavili v delovno območje in nato testirate delovno rutino.

Slika 159: Pregled definiranih rutin



V meniju si lahko ogledate informacije o definiranih delovnih rutinah.

3.15.1 Programiranje blok programa

Slika 160: Program za robota

```
24      PROC main()
25  locate_hidden 1;
26  IF bPartFound THEN
27      MoveJ Location4, v200, fine, t_Prijemalo\WObj:=wobj_plosca_1;
28      MoveL Location2, v50, fine, t_Prijemalo\WObj:=wobj_plosca_1;
29      closeGripper ;
30      MoveJ Location20, v200, fine, t_Prijemalo\WObj:=wobj_plosca_1;
31      MoveJ Location21, v200, fine, t_Prijemalo\WObj:=wobj_plosca_1;
32      MoveJ Location6, v50, fine, t_Prijemalo\WObj:=wobj_plosca_1;
33      openGripper ;
34      MoveJ Location7, v500, fine, t_Prijemalo\WObj:=wobj_plosca_1;
35      locate_hidden 2;
36      MoveJ Location8, v200, fine, t_Prijemalo\WObj:=wobj_plosca_1;
37      MoveJ Location9, v50, fine, t_Prijemalo\WObj:=wobj_plosca_1;
38      closeGripper ;
39      MoveJ Location10, v200, fine, t_Prijemalo\WObj:=wobj_plosca_1;
40      MoveJ Location22, v200, fine, t_Prijemalo\WObj:=wobj_plosca_1;
41      MoveJ Location11, v20, fine, t_Prijemalo\WObj:=wobj_plosca_1;
42      openGripper ;
43      MoveJ Location12, v200, fine, t_Prijemalo\WObj:=wobj_plosca_1;
44      locate_hidden 3;
45      MoveJ Location13, v200, fine, t_Prijemalo\WObj:=wobj_plosca_1;
46      MoveJ Location14, v50, fine, t_Prijemalo\WObj:=wobj_plosca_1;
47      closeGripper ;
48      MoveJ Location15, v50, fine, t_Prijemalo\WObj:=wobj_plosca_1;
49      MoveJ Location16, v50, fine, t_Prijemalo\WObj:=wobj_plosca_1;
50      MoveJ Location17, v50, fine, t_Prijemalo\WObj:=wobj_plosca_1;
51      openGripper ;
52      MoveJ Location18, v200, fine, t_Prijemalo\WObj:=wobj_plosca_1;
53  ENDIF
54      ENDPROC
55  ENDMODULE
```

Na začetku programiranja se izbere, kateri locate job je potrebno izbrati. V mojem primeru je bil na začetku locate job 1. Nato se naredi IF stavek.

IF stavek deluje tako, da če se v programu potrdi IF stavek, se nato nadaljuje program. V tem primeru mora robot poiskati v točki, v kateri je tisti predmet, ki ga mora robot kasneje pobrati. Če ga robot najde, se začne izvajanje IF stavka.
V eni vrstici je potrebno izbrati, ali bo robot naredil joint ali linear premik. Nato je potrebno izbrati, s čim bo robot prijel ta predmet, v mojem primeru je to t_prijemalo. Nato se izbere hitrost, ki je razdeljena na 5 hitrosti od very quickly do very slowly. Nato se mora robotu povedati, v katero lokacijo gre oz. kam se bo premaknil. Po premiku se tudi lahko izbere, ali bo se v tej točki ustavil, takrat se bo uporabil fine, v primeru če pa želimo, da se robot v tej točki ne ustavi, pa se uporabi z. Nato pa še je samo potrebno določiti work object, ki pa je že bil določen v predhodnih nalogah.
Robot dela po principu premikanja od točke do točke. Moj program je napisan samo z enim IF – DO stavkom. Lahko bi program bil tudi napisan s tremi IF – DO stavki, vendar je to poljubno po izbiri, kako je všeč vsakemu posamezniku. V programu se še uporabita oznake grip – release.

4 ZAKLJUČEK

4.1 SKLEPI

V diplomskem delu so bili uspešno zastavljeni vsi cilji. Pripravljene so bile učne situacije za realnega robota ABB ter simulacijo robota. Te simulacije bodo omogočale, da bo robot pokazal, česa je zmožen. Razvite vaje vključujejo celoten postopek priprave pravega robota ter nastavitev robota, gibanje robota. Pripravljena navodila bodo omogočala samostojno izvedbo vaj, tudi za tiste, ki se pred tem niso nikoli srečali s programiranje robota.

5 VIRI

ABB. 2025. ABB. *Software downloads.* [Elektronski] 2025. [Navedeno: 10. junij 2025.] https://new.abb.com/products/robotics/software-and-digital/downloads.

—. 2025. ABB IRB. *IRB.* [Elektronski] 2025. [Navedeno: 11. 19 2025.] https://www.abb.com/global/en/areas/robotics/products/robots/articulated-robots/medium-robots/irb-2600.

—. 2025. ABB IRB 2600. *IRB 2600.* [Elektronski] 2025. [Navedeno: 19. 11 2025.] https://www.abb.com/global/en/areas/robotics/products/robots/articulated-robots/medium-robots/irb-2600.

—. 2025. ABB logo. [Elektronski] 2025. [Navedeno: 19. 11 2025.] https://www.abb.com/global/en.

—. 2025. ABB Manual activation. [Elektronski] 2025. [Navedeno: 19. 11 2025.] http://manualactivation.e.abb.com/.

—. 2025. ABB Robotics. *ABB Robotics.* [Elektronski] 2025. [Navedeno: 19. 11 2025.] https://www.abb.com/global/en/areas/robotics.

—. 2025. ABB Robotics GoFa. *GoFa.* [Elektronski] 2025. [Navedeno: 19. 11 2025.] https://www.abb.com/global/en/areas/robotics/products/robots/collaborative-robots/gofa.

—. 2025. ABB RobotStudio. *RobotStudio.* [Elektronski] 2025. [Navedeno: 19. 11 2025.] https://www.abb.com/global/en/areas/robotics/products/software/robotstudio-suite/robotstudio-desktop.

—. 2025. ABB Šolanje navodila. *RobotStudio tečaj.* 2025.

—. 2024. Izobraževanje za ABB šolsko robotsko celico. 2024.

Jerman, Karl. 2024. *ABB aktivacija licence.* 2024.

Sigh, Aman. 2025. eLearning Industry. *Scenario-Based Learning.* [Elektronski] 6. 7 2025. [Navedeno: 3. 3 2026.] https://elearningindustry.com/scenario-based-learning-transforming-corporate-ld-with-real-world-context.

PRILOGE

PRILOGA A: 2D-KONTURA_ADAM-V3

PRILOGA B: 3D_KONTURA_PT3

PRILOGA C: KONICA_ZA_LEZAJ_BREZ_ZASTAVICE

PRILOGA D: KONICA_ZA_LEZAJ_Z_ZASTAVICO

PRILOGA E: KONTURA_2D

PRILOGA F: ORODJE_VARILNA_PISTOLA_KRATKA

PRILOGA G: OVIRA

PRILOGA H: PLOSCA_SOLITARE:POSEVNINA_V8_VALJ_PODROBNI

PRILOGA I: PLOSCA_SOLITARE_V5_PT2

PRILOGA J: PLOSCA_ZA_NOSILEC

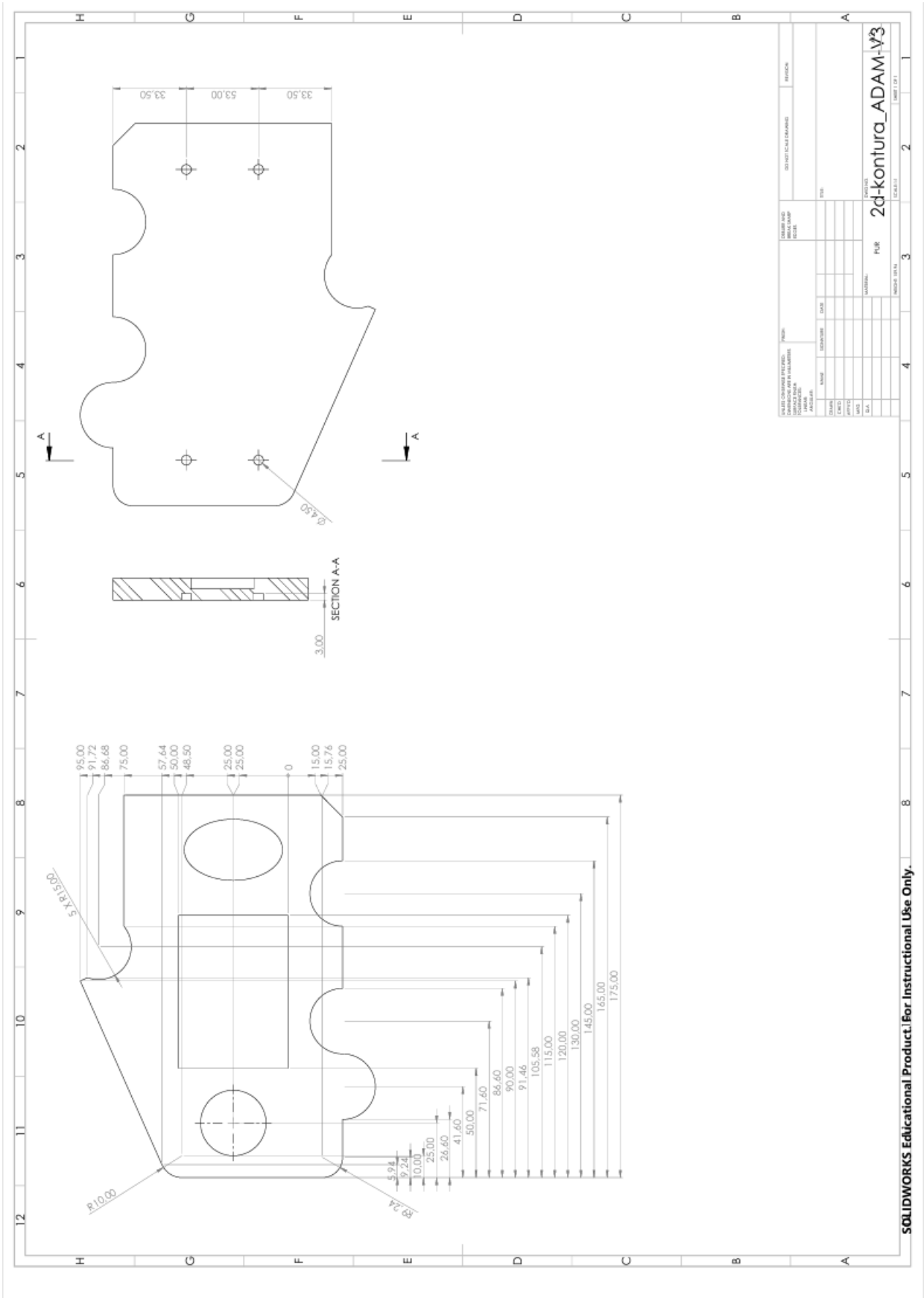
PRILOGA K: POZ.1

PRILOGA L: POZ.3

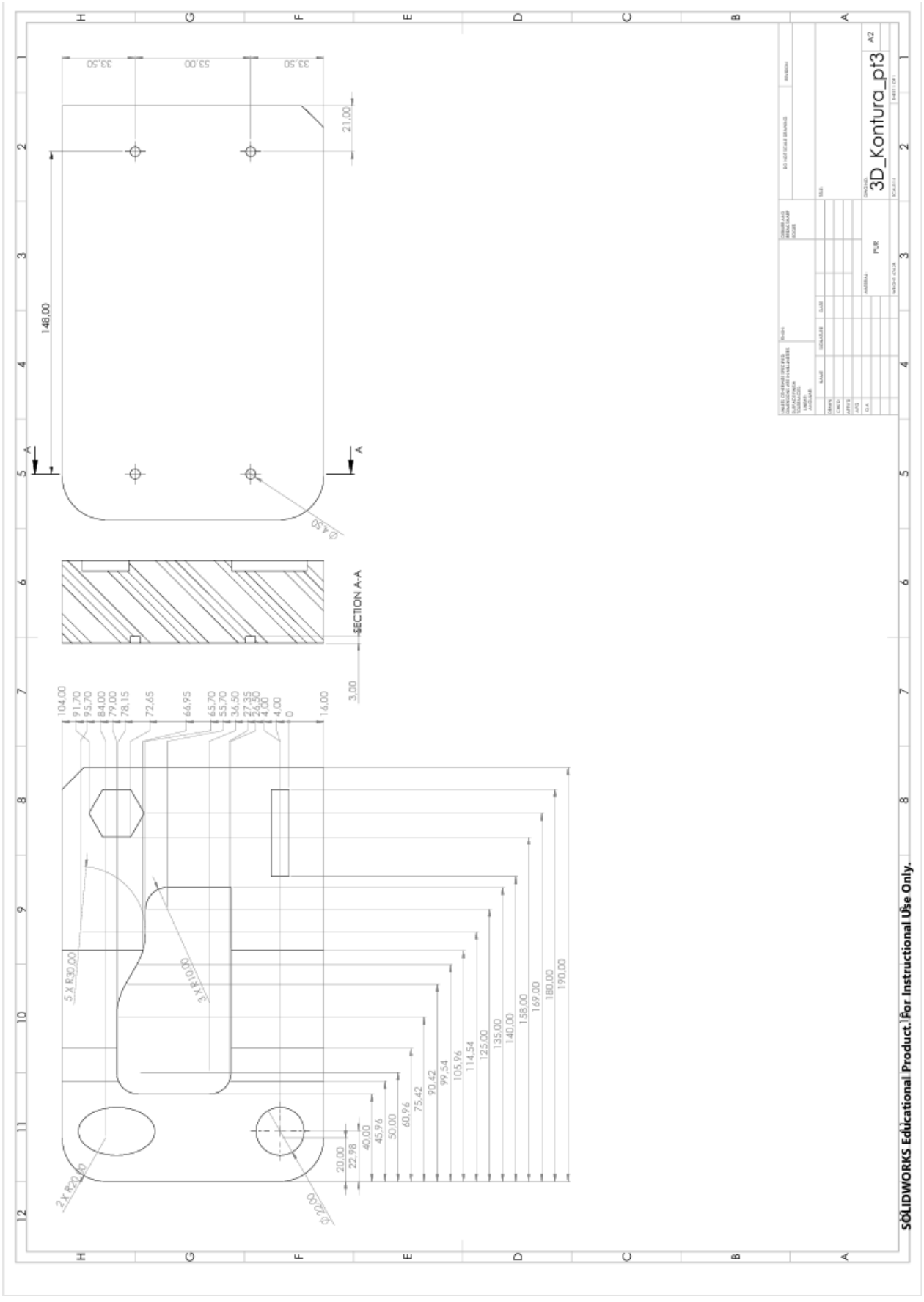
PRILOGA M: PRSTI_KRENKER_KOLAR – V4

PRILOGA N: PART 3

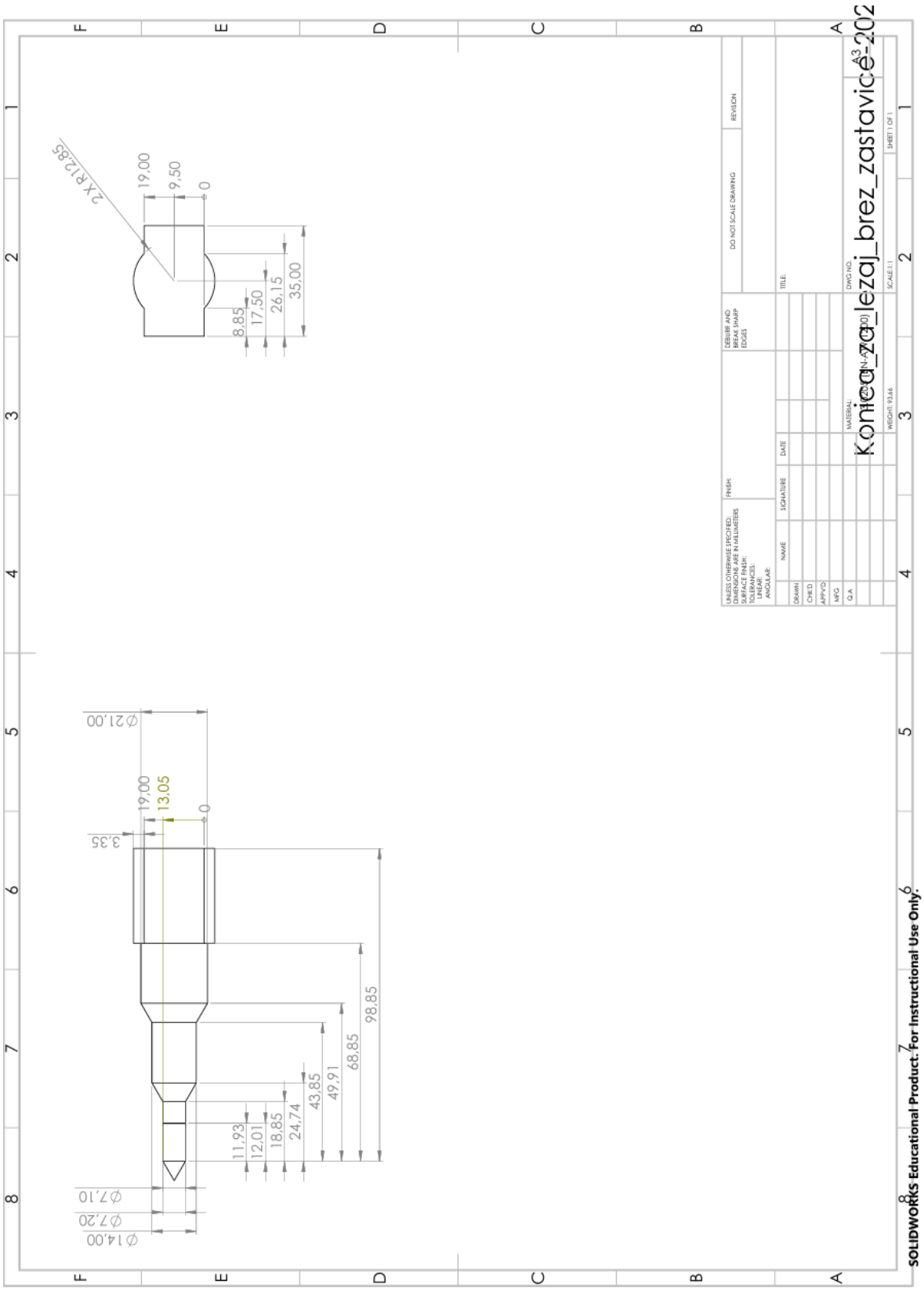
PRILOGA A: 2D-KONTURA_ADAM-V3



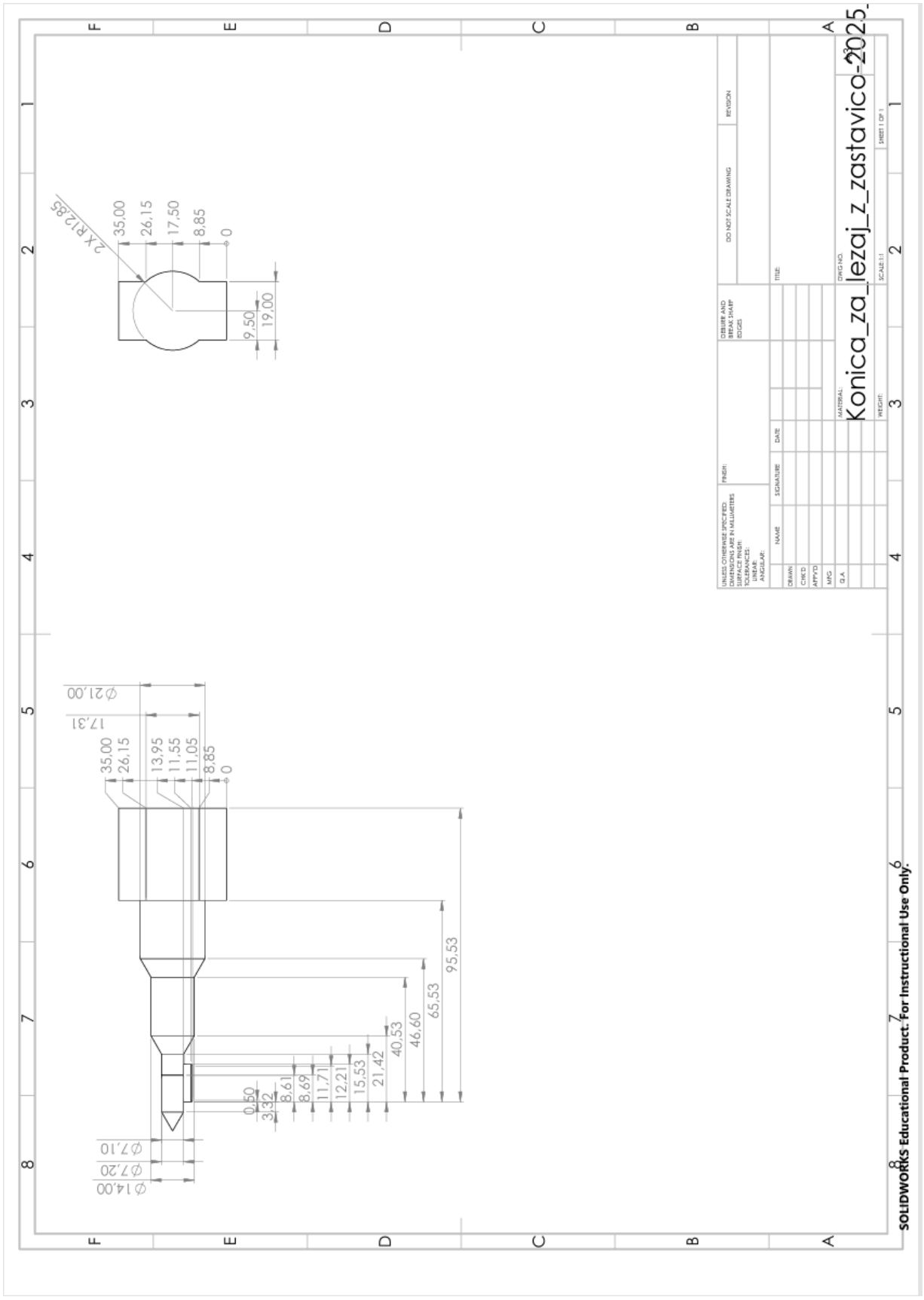
PRILOGA B: 3D_KONTURA_PT3



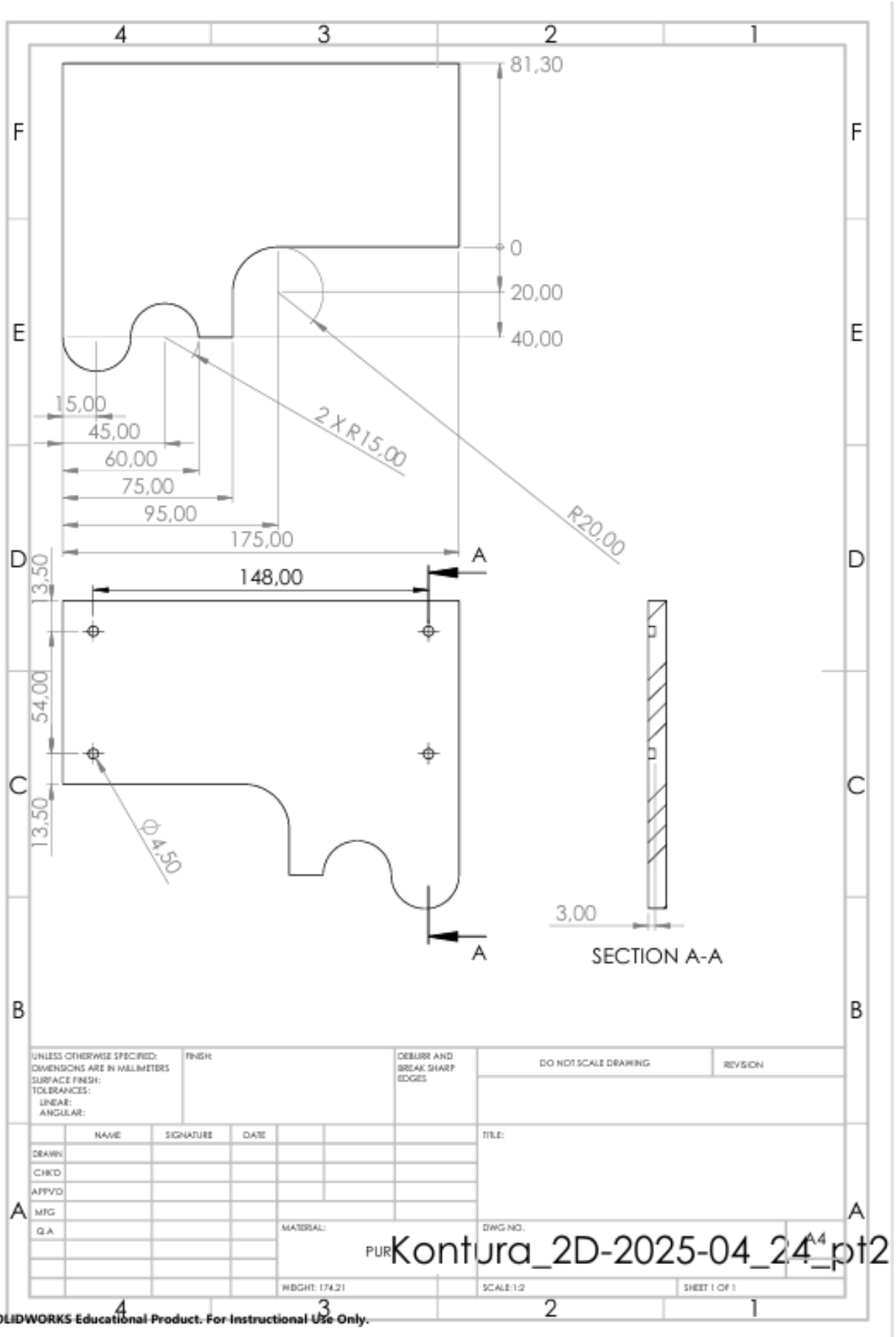
PRILOGA C: KONICA_ZA_LEZAJ_BREZ_ZASTAVICE



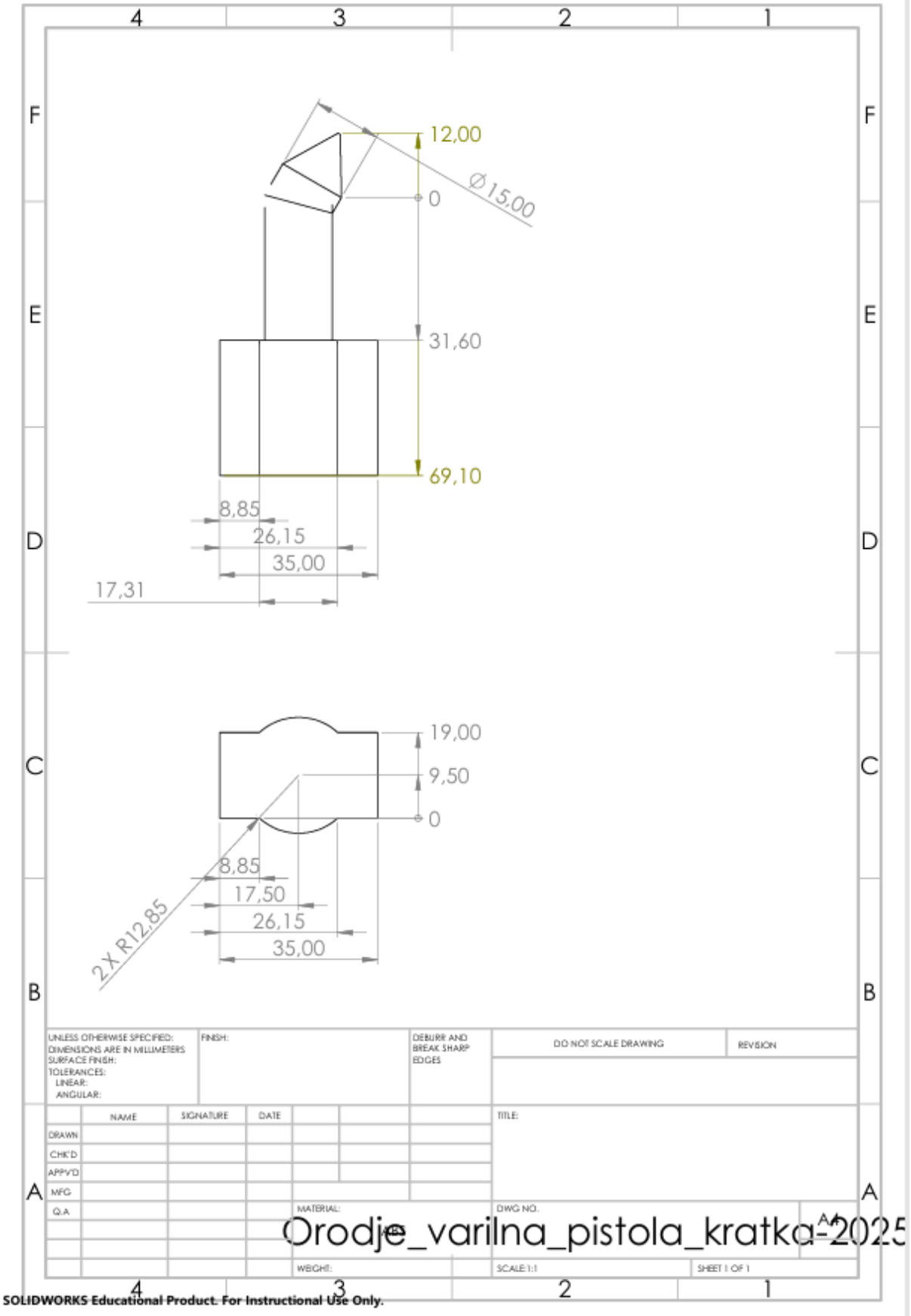
PRILOGA D: KONICA_ZA_LEZAJ_Z_ZASTAVICO



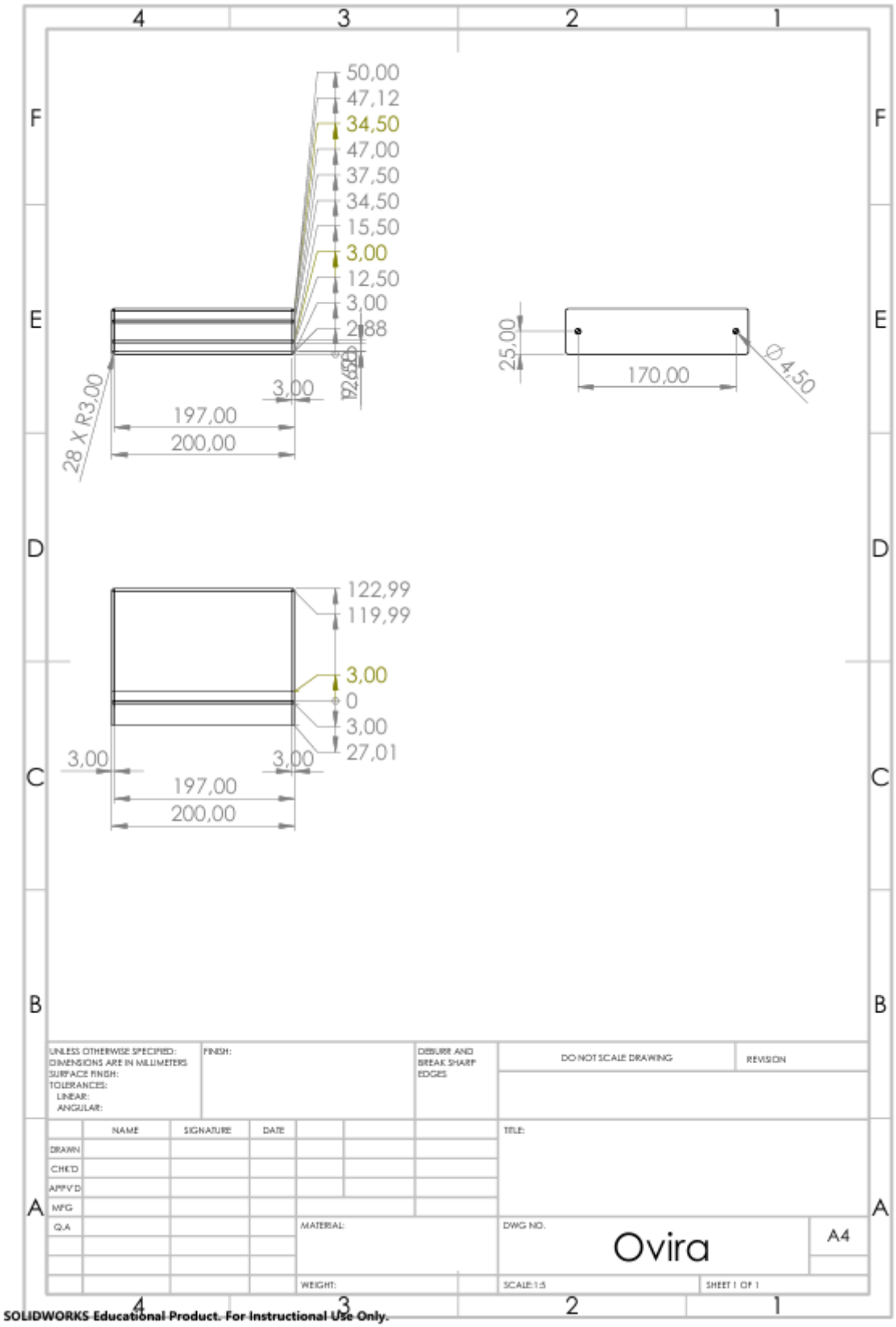
PRILOGA E: KONTURA_2D



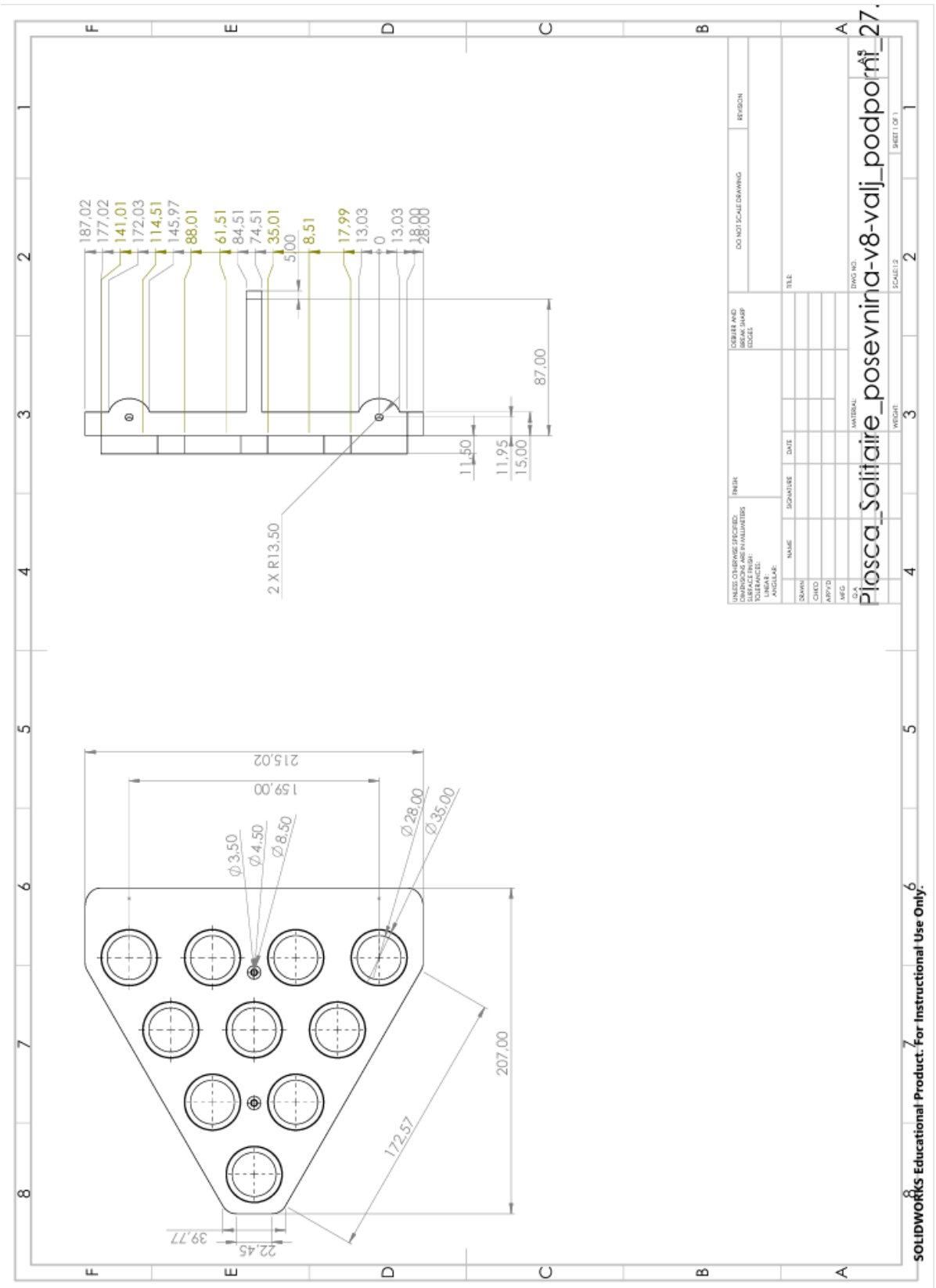
PRILOGA F: ORODJE_VARILNA_PISTOLA_KRATKA



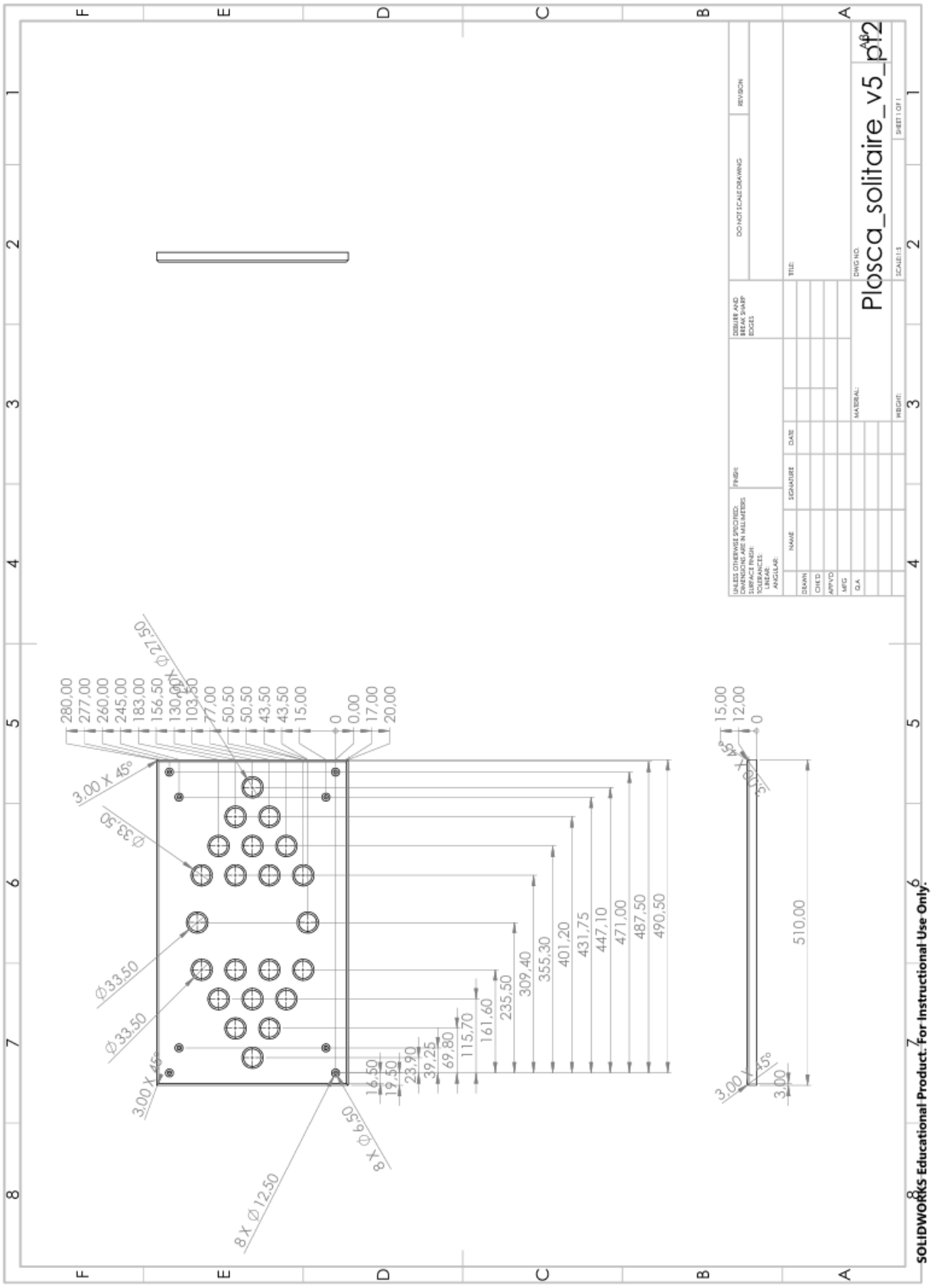
PRILOGA G: OVIRA



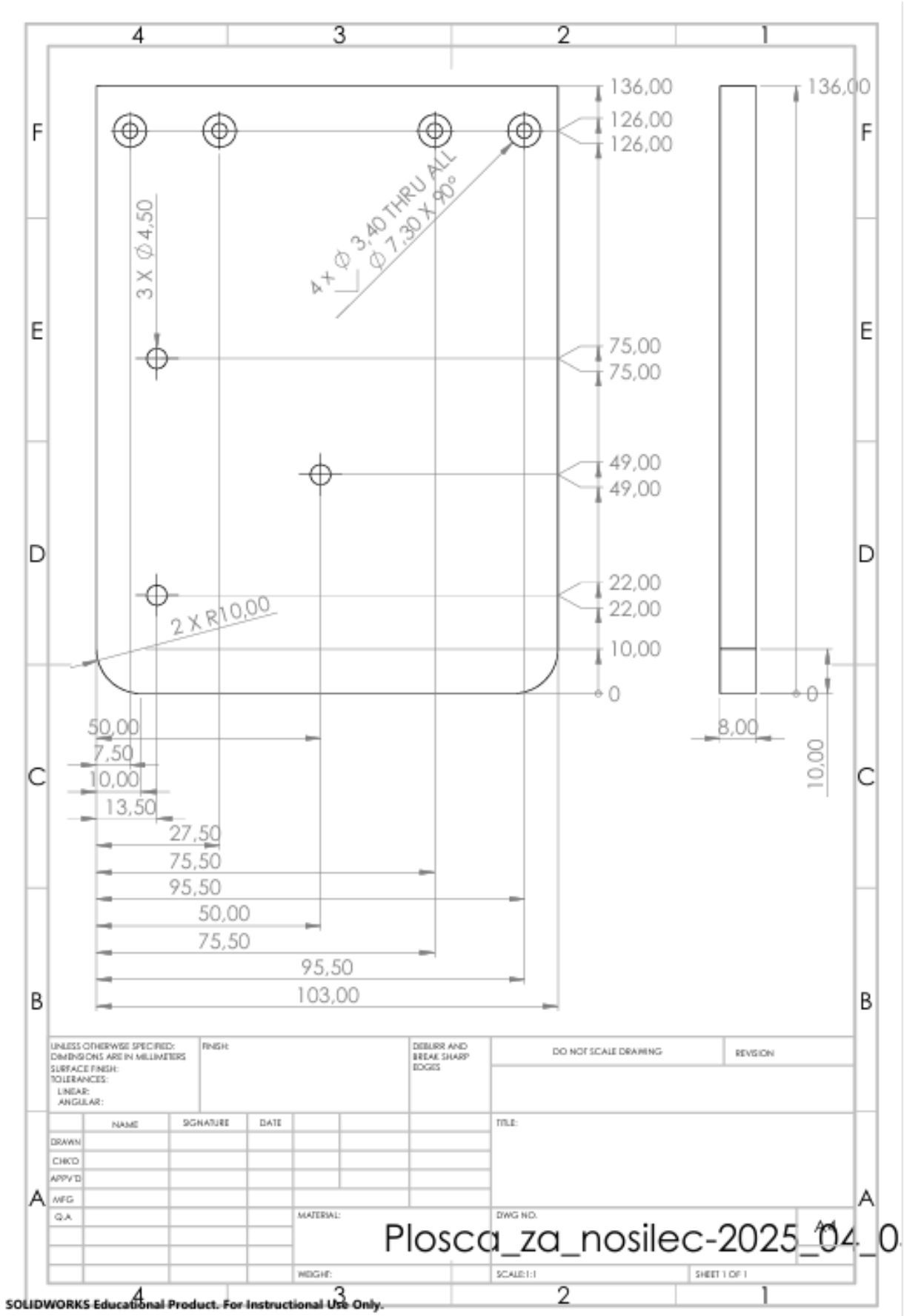
PRILOGA H: PLOSCA_SOLITARE:POSEVNINA_V8_VALJ_PODROBNI



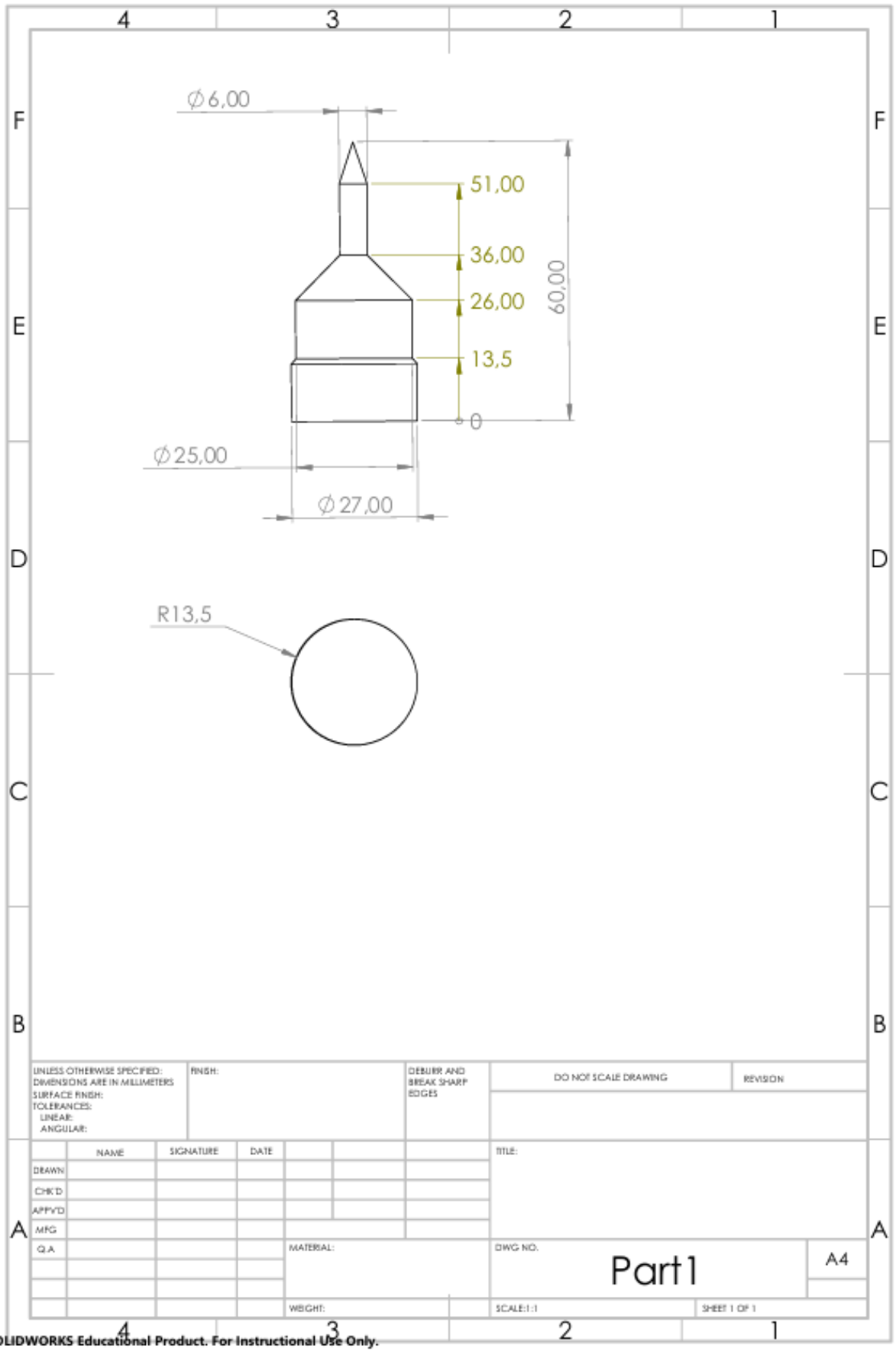
PRILOGA I: PLOSCA_SOLITARE_V5_PT2



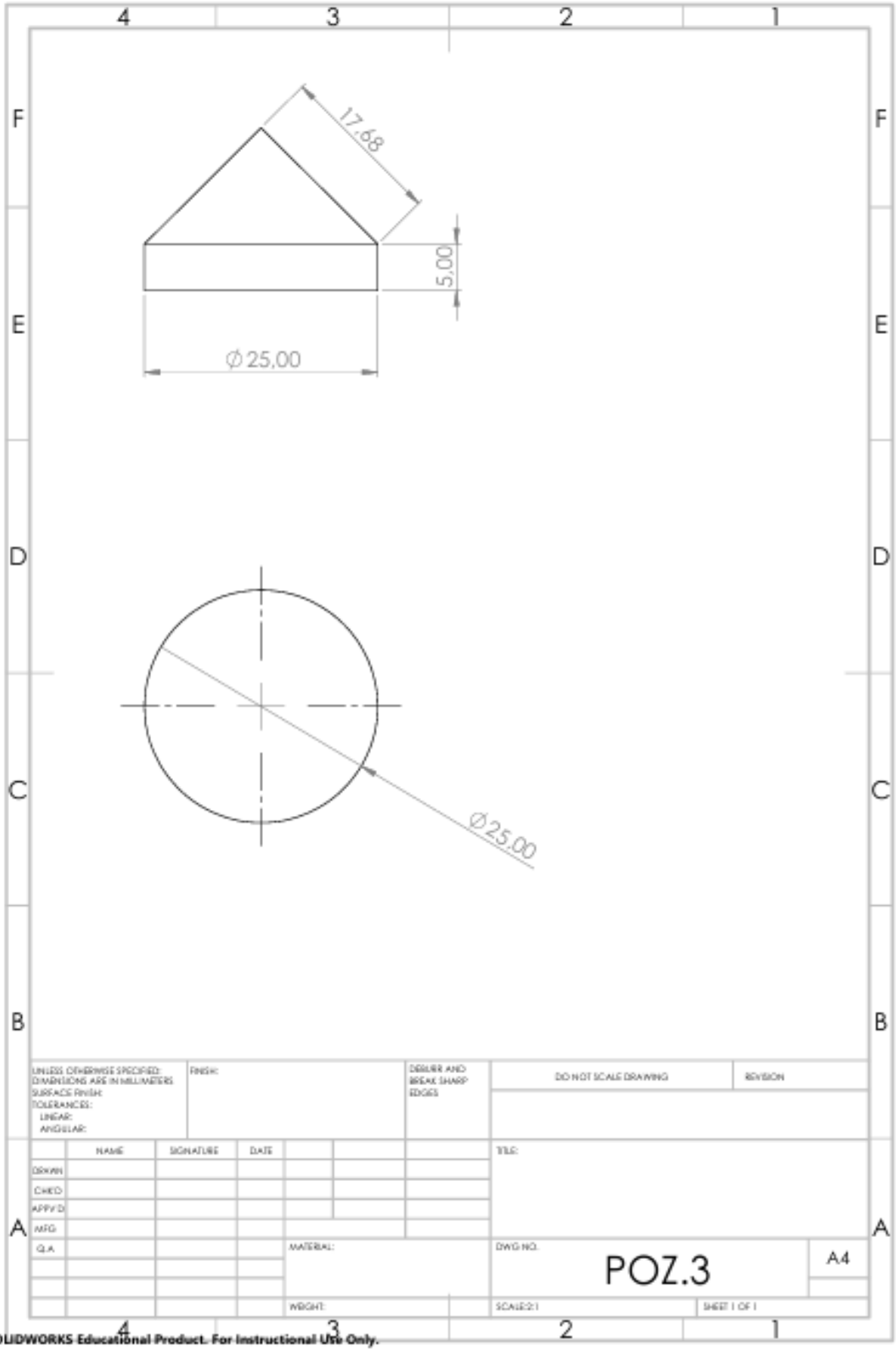
PRILOGA J: PLOSCA_ZA_NOSILEC



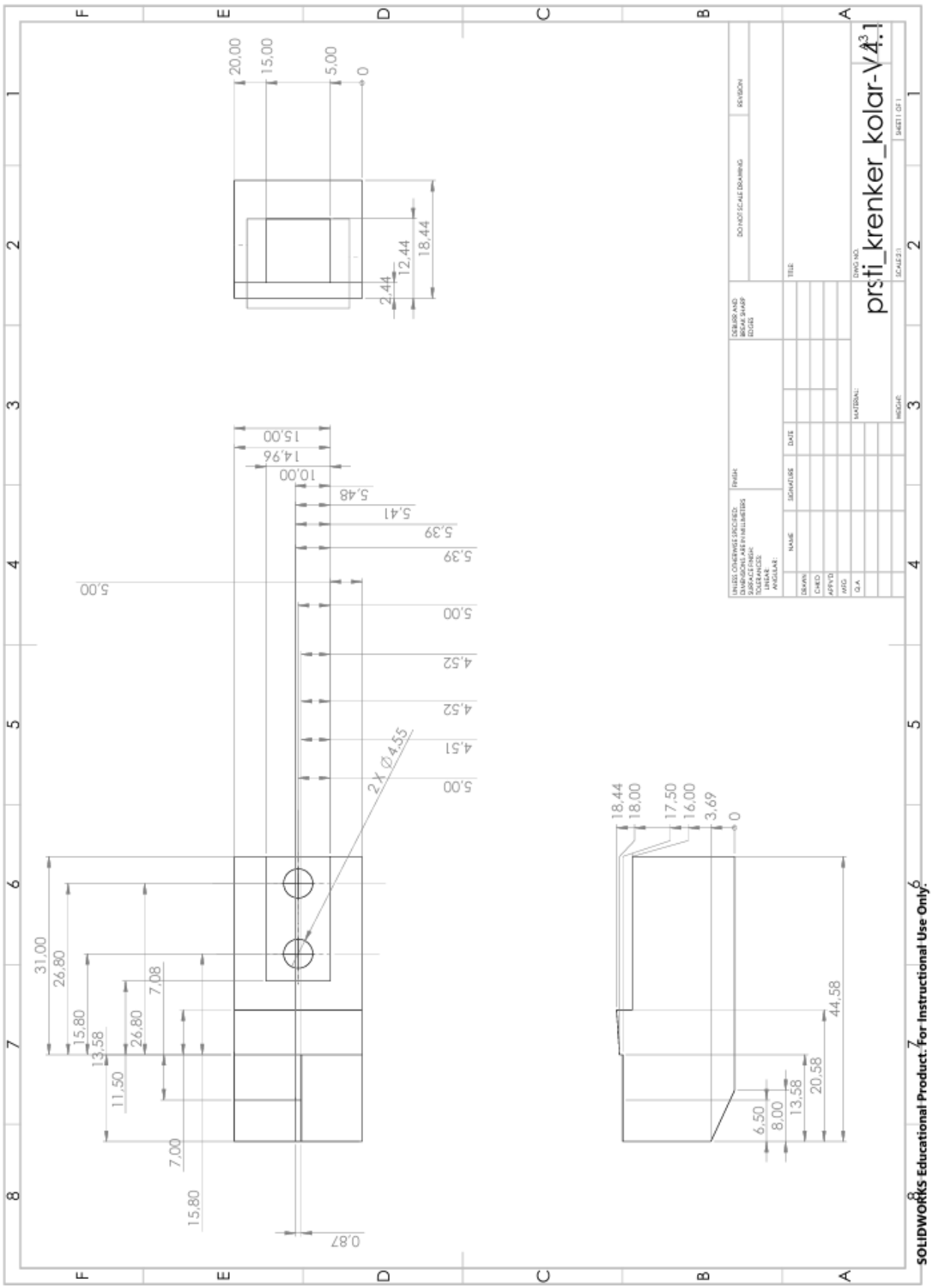
PRILOGA K: POZ.1



PRILOGA L: POZ.3



PRILOGA M: PRSTI_KRENKER_KOLAR – V4



PRILOGA N: PART 3