

Upravljanje industrijskim robotom Toshiba TV-1000 definiranjem putanje u .dxf datoteci

Kraljić, Luka

Undergraduate thesis / Završni rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Bjelovar University of Applied Sciences / Veleučilište u Bjelovaru**

Permanent link / Trajna poveznica: <https://um.nsk.hr/um:nbn:hr:144:601895>

Rights / Prava: [In copyright](#) / [Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2025-03-13**



Repository / Repozitorij:

[Digital Repository of Bjelovar University of Applied Sciences](#)



VELEUČILIŠTE U BJELOVARU
PREDDIPLOMSKI STRUČNI STUDIJ MEHATRONIKA

**UPRAVLJANJE INDUSTRIJSKIM ROBOTOM
TOSHIBA TV-1000 DEFINIRANJEM PUTANJE U
.DXF DATOTECI**

Završni rad br. 01/MEH/2019

Luka Kraljić

Bjelovar, ožujak 2019.



Veleučilište u Bjelovaru

Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Kandidat: **Kraljić Luka**

Datum: 05.02.2019.

Matični broj: 001347

JMBAG: 0314013698

Kolegij: **OSNOVE ROBOTIKE**

Naslov rada (tema): **Upravljanje industrijskim robotom Toshiba TV-1000 definiranjem putanje u .dxf datoteci**

Područje: **Tehničke znanosti**

Polje: **Strojarstvo**

Grana: **Proizvodno strojarstvo**

Mentor: **Tomislav Pavlic, mag.ing.mech.**

zvanje: **viši predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. **Zoran Vrhovski, mag.ing.el.techn.inf., predsjednik**
2. **Tomislav Pavlic, mag.ing.mech., mentor**
3. **dr.sc. Alan Mutka, član**

2. ZADATAK ZAVRŠNOG RADA BROJ: 01/MEH/2019

Potrebno je opisati ulogu .dxf datoteka u kontekstu upravljanja i vođenja numerički upravljanih strojeva i industrijskih robota. Potrebno je prilagoditi 3D CAD model robota Toshiba TV-1000. Uvesti prilagođeni CAD model robota u programski alat RoboDK, prema zahtjevima dimenzija, kinematike i orijentacija koordinatnih sustava baze robota i potrebnih alata robota. Objasniti ulogu postprocesiranja kod upravljanja numerički upravljanih strojeva i industrijskih robota. Na osnovu simulacije u programskome alatu RoboDK generirati kod za 6-osnog rotacijskog robota Toshiba TV-1000. Testirati kretanje robota na nekoliko primjera.

Zadatak uručen: 05.02.2019.

Mentor: **Tomislav Pavlic, mag.ing.mech.**



Zahvala

Zahvaljujem se mentoru Tomislavu Pavlic, mag.ing.mech., na stručnim savjetima, potpori i vođenju kroz cijeli proces izrade završnog rada u tvrtki Data Link d.o.o kao i kroz pisanje ovog završnog rada, Zoranu Vrhovski, mag.ing.el.techn.inf., na podršci i uloženom trudu oko dogovora s Tvrtkom Data Link d.o.o. Zahvaljujem se tvrtki Data Link d.o.o i svim njenim zaposlenicima na ustupljenoj opremi i radnom prostoru za izradu završnog rada. Također, velika zahvala profesorima, profesoricama i cijelom osoblju Veleučilišta u Bjelovaru na uloženom trudu i korektnom odnosu prema studentima.

Sadržaj

1. Uvod	1
2. OPIS PROBLEMA I RJEŠENJA ZA VOĐENJE ROBOTA DXF DATOTEKOM	2
2.1 <i>Primjena vođenja DXF datotekom u drugim područjima industrije</i>	5
2.2 <i>Slična rješenja trenutno dostupna na tržištu</i>	7
3. KORIŠTENA OPREMA I PROGRAMSKI ALATI	8
3.1 <i>6- osni robot Toshiba TV-1000</i>	8
3.2 <i>TS-3100 kontroler robota</i>	10
3.3 <i>CNC vreteno</i>	11
3.4 <i>TSPC programski alat</i>	13
3.5 <i>Programsko simulacijsko okruženje RoboDK</i>	14
4. UPRAVLJAČKI KOD I UPRAVLJANJE ROBOTOM TOSHIBA TV-1000	16
4.1 <i>Definiranje točaka putanje u kartezijском prostoru</i>	17
4.2 <i>Konfiguracija robota</i>	19
4.3 <i>Struktura upravljačkog koda robota Toshiba TV-1000</i>	23
5. DXF DATOTEKA	28
5.1 <i>Uloga DXF datoteke u upravljanju numerički upravljanih strojeva</i>	28
5.2 <i>Sadržaj DXF datoteke</i>	29
6. IZRADA SIMULACIJSKOG OKRUŽENJA U PROGRAMU ROBODK	33
6.1 <i>Izrada mehanizma robota</i>	33
6.2 <i>Izrada radnog okruženja robota</i>	36
6.3 <i>Dodavanje alata robota</i>	37
7. RAZVOJ POSTPROCESORA ZA PREVOĐENJE DXF DATOTEKE U KOD ROBOTA	40
7.1 <i>Razvoj skripte post procesora</i>	41
7.1.1 <i>Post procesiranje varijabli</i>	42
7.1.2 <i>Post procesiranje upravljačkih naredbi</i>	44
7.1.3 <i>Post procesiranje ulazno / izlaznih podataka</i>	45
7.1.4 <i>Post procesiranje podataka o konfiguraciji</i>	46
8. ISPITIVANJE RADA ROBOTA UPRAVLJANOG PUTANJOM IZ DXF DATOTEKE	48
8.1 <i>Izrada brtve</i>	48
8.2 <i>Bušenje provrta na sjenilima LED reflektora</i>	55
9. ZAKLJUČAK	58
10. LITERATURA	59
11. OZNAKE I KRATICE	60
12. SAŽETAK	61
13. ABSTRACT	62

14. PRILOZI	63
--------------------------	-----------

1. Uvod

Potreba industrije za bržom i jednostavnijom prilagodbom robota različitim zadacima sve je veća, pogotovo u tvrtkama koje se primarno bave razvojem. Takve tvrtke imaju potrebu za stalnom izmjenom upravljačkih kodova robota ovisno o zadacima koji se konstantno nameću prilikom razvoja proizvoda. Ponekad tvrtke posjeduju mnogo različitih robota raznih proizvođača i modela što dodatno otežava zadatak programiranja. Potrebna su programska rješenja koja mogu omogućiti brzo programiranje različitih robota na temelju željene putanje bez gubitka vremena na izradu složenih upravljačkih kodova. Takvi programsko-simulacijski alati samostalno generiraju upravljački kod robota na temelju željene putanje prethodno nacrtane u CAD alatu i spremljene u DXF format. Primjenom navedenog postupka znatno se smanjuje vrijeme programiranja robota te je moguća vrlo brza korekcija koda ili potpuna izmjena koda u slučaju potrebe.

Završni rad je baziran na industrijskom robotu Toshiba, model TV-1000 sa šest stupnjeva slobode gibanja. Potrebno je ubaciti 3D model robota u programsko-simulacijski alat RoboDK i unijeti dimenzije robota, poziciju koordinatnog sustava baze i alata na temelju kojih se određuje kinematika robota unutar simulacije. Nakon postavljanja 3D modela te kreiranja robota u programsko-simulacijskom okruženju RoboDK potrebno je izraditi post procesor za potrebe generiranja upravljačkog koda robota Toshiba TV-1000. Potrebno je navesti nekoliko primjera upravljanja robotom putanjom dobivenom iz DXF datoteke .

U završnom radu je opisan postupak prevođenja DXF datoteke u upravljački kod industrijskog robota Toshibe TV-1000 korištenjem kombinacije post procesorskih alata i simulacijskog okruženja RoboDK. Opisana je izrada simulacijskog okruženja robota Toshiba TV-1000 u programskom alatu za simuliranje i *off-line* programiranje RoboDK. U prvom dijelu rada navedena su slična rješenja koja se upotrebljavaju u drugim područjima industrije. Navedeni su svi korišteni alati i oprema korištena za izradu rada. Opisan je upravljački kod robota i post procesora te su dani primjeri na kojima je ispitan post procesor.

2. OPIS PROBLEMA I RJEŠENJA ZA VOĐENJE ROBOTA DXF DATOTEKOM

Izrada upravljačkog koda za industrijske robote klasičnim metodama programiranja dug je i kompliciran proces. Programer mora dobro poznavati programske jezike različitih robota i tehničkim karakteristikama svakog pojedinog robota. Vrlo je teško planirati i određivati putanje u radnom prostoru robota, posebno neke kompleksnije interpolacije kao što je kružna. Putanje robota za neke radnje mogu biti vrlo složene. Programiranje klasičnim metodama povećava vjerojatnost pojave pogrešaka u upravljačkom kodu te samim time povećava i utrošeno vrijeme na analizu i traženje pogrešaka ako je do njih došlo. Problem nastaje u trenutku kada se pojavi potreba za učestalom izmjenom upravljačkog koda robota. Izrada novog upravljačkog koda za novi posao može trajati danima i za to vrijeme cijeli se pogon mora zaustaviti. Kod velikih robotskih postrojenja to uzrokuje goleme troškove za tvrtku. Upravo zbog utroška vremena i potrebe za učestalim reprogramiranjem robota potreban je drugačiji pristup programiranju industrijskih robota.

Jedno od mogućih rješenja, koje se danas najviše upotrebljava u industriji je korištenje bilo kojeg CAD alata za crtanje putanje robota. Putanja iz CAD alata može se spremirati u DXF format te se kao takva može koristiti za daljnju obradu s nekim CAM alatom ili alatom za *off-line* programiranje robota. U sljedećem koraku radi se određena analiza DXF datoteke. Datoteka spremljena u DXF format je vektorska 2D slika CAD crteža koja se može koristiti kao putanja alatnoga stroja ili robota.

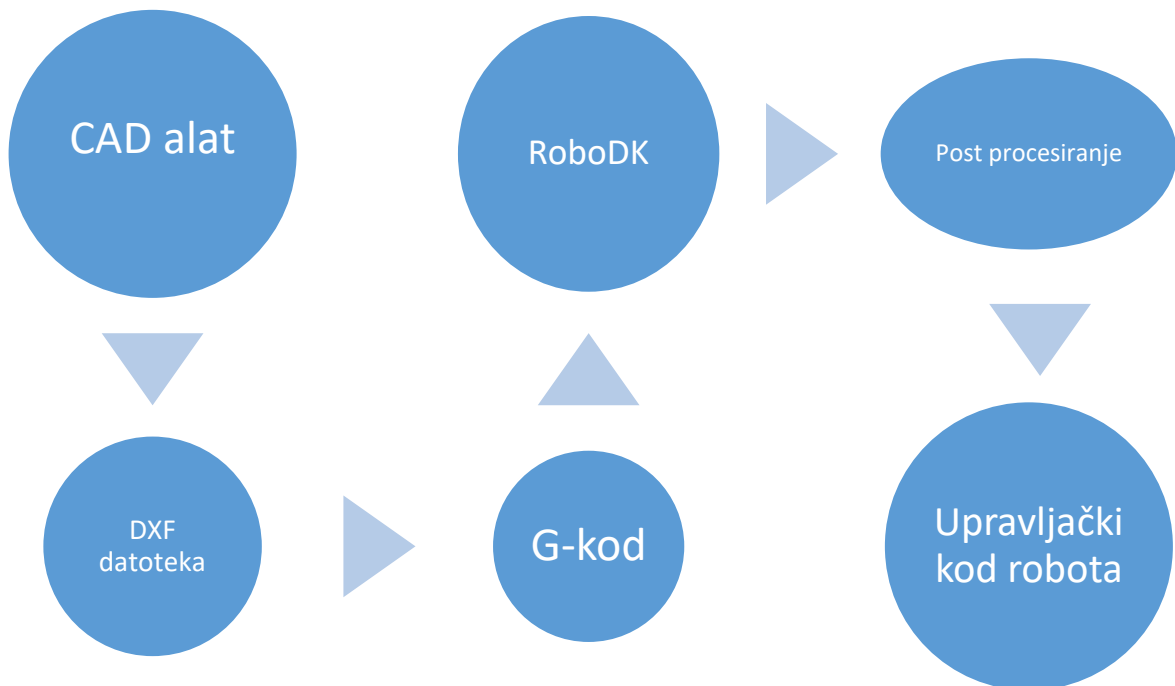
Da bi se DXF datoteka mogla koristiti kao putanja robota potreban je programski alat koji će vektorsku 2D sliku DXF datoteke prevesti u kod industrijskog robota. Problemu prevođenja u upravljački kod robota moguće je pristupiti na više načina, direktnim prevođenjem DXF datoteke u kod robota ili korištenjem CAM alata ili simulacijskog okruženja za *off-line* programiranje. Teži način bio bi direktno prevođenje vektorske 2D slike u kod robota zato što je za izradu upravljačkog koda robota potreban niz drugih parametara kao što su pozicije referentnih koordinatnih sustava baze i alata, konfiguracija robota, brzine i ubrzanja robota.

Sama DXF datoteka ne sadrži te podatke nego samo vektorske podatke o CAD crtežu. Upravo iz tog razloga nije moguće izravno prevođenje datoteke u upravljački kod robota te se treba poslužiti međukorakom. Međukorak može biti post procesorski alat kao zaseban računalni program u sklopu CAM alata ili alata za *off-line* programiranje robota koji će vršiti analizu DXF datoteke i omogućiti programeru podešavanje ostalih upravljačkih parametara robota.

Tvrtki Data Link d.o.o ponuđeno je univerzalno programsko-simulacijsko okruženje RoboDK koje posjeduje niz funkcija a jedna od njih je simuliranje putanje iz DXF datoteke i prevođenje iste u upravljački kod robota. Kada se putanja robota potpuno simulira, simulacijsko okruženje RoboDK generira jedinstveni niz naredbi i varijabli koje se zatim pomoću post procesora prevode u upravljački kod bilo kojeg industrijskog robota. Simulacijsko okruženje pruža programeru dodatno prilagođavanje putanje robota po potrebi. Osim dodatnog prilagođavanja putanje RoboDK pruža mogućnost *off-line* programiranja.

Nakon što se putanja potpuno definira u simulacijskom okruženju, potrebno je koristiti post procesorski alat u sklopu istog simulacijskog okruženja koji će simuliranu putanju prevesti u upravljački kod robota. Post procesorski alat u sklopu simulacijskog okruženja tvori jedinstveni skup raznih programskih rješenja te olakšava posao programiranja robota i potpuno eliminira potrebu za poznavanjem upravljačkog koda robota.

Na slici 2.1 slikovno su prikazani koraci prevođenja putanje iz datoteke DXF formata u upravljački kod industrijskog robota Toshiba TV-1000. Proces prikazan na slici 2.1 ujedno je i rješenje koje je ponuđeno tvrtki Data Link d.o.o.



Slika 2.1: Koraci prevođenja DXF datoteke u upravljački kod industrijskog robota

Opis sustava prevođenja DXF datoteke u kod robota vidljivog na slici 2.1:

Programer u CAD alatu izrađuje 2D putanju po kojoj robot mora obaviti zadanu radnju. Nakon što je putanja nacrtana sprema se u DXF vektorski format i zatim se predaje simulacijskom okruženju RoboDK. U simulacijskom okruženju programer izrađuje simulaciju na temelju putanje iz DXF datoteke. Kada je programer zadovoljan simulacijom pokreće prevođenje simulacije u upravljački kod robota. G-kod je međukorak koji obavlja RoboDK te programeru taj korak nije vidljiv na zaslonu u procesu prevođenja. RoboDK može koristiti neki javno dostupan alat za dobivanje G-koda iz DXF datoteke ako se on instalira na točno određeno mjesto na računalu gdje je instaliran i programski alat RoboDK.

2.1 Primjena vođenja DXF datotekom u drugim područjima industrije

Mnogo numerički upravljanih alatnih strojeva koristi upravo DXF datoteke za upravljanje. Zbog zahtijeva industrije takvi strojevi moraju imati brz, jednostavan i učinkovit način upravljanja te su oni prilagođeni isključivo operaterima. Kod razvoja stroja programeri razvijaju korisničko sučelje i post procesor koji na temelju DXF datoteke iz nekog CAD alata samostalno generira upravljački kod stroja. Danas se u industriji upotrebljava kombinacija CAD i CAM sustava.

„CAD je u pojednostavljenom smislu elektronička crtača daska jer se umjesto olovkom i papirom konstruira na kompjuterskom terminalu. U složenijem značenju CAD se može povezati s proizvodnjom, pri čemu se prenose specifikacije i proces izrade konstruiranog proizvoda. CAM je pojednostavljena kompjutorizirana proizvodnja koja obuhvaća numerički upravljanje alatne strojeve, industrijske robote i fleksibilne proizvodne sisteme“. [1]

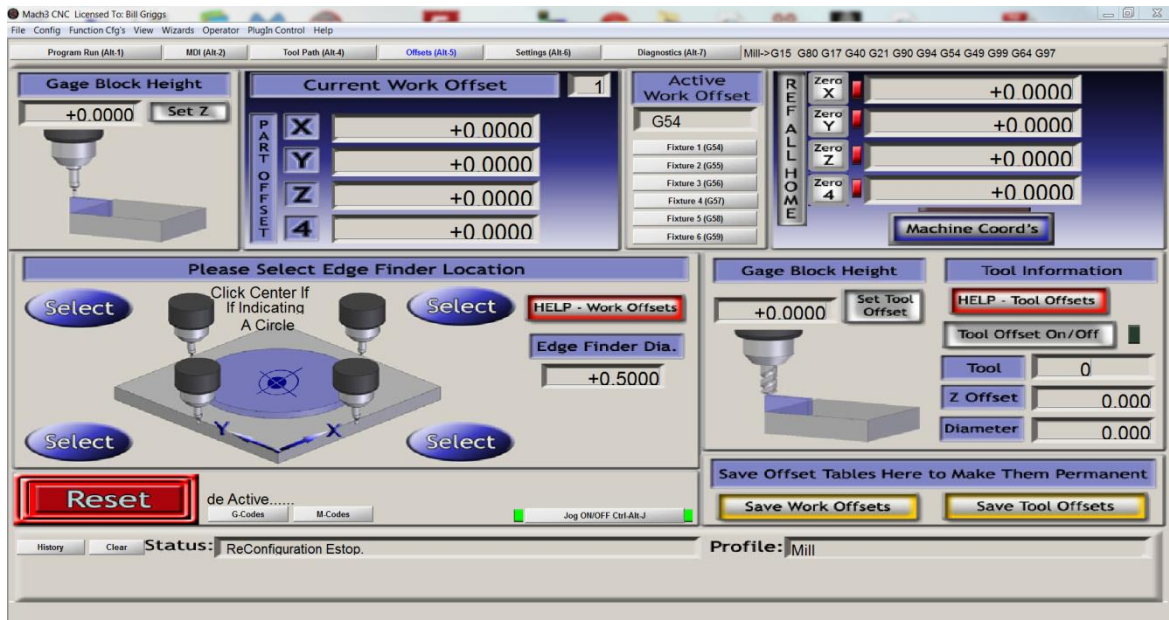
Neke od operacija koje se upravljaju DXF ili drugim CAD datotekama:

- Rezanje
- Glodanje
- Tokarenje
- Bušenje
- Nanašanje brtvila
- 3D printanje
- Lasersko rezanje i graviranje

Obavljanje gore navedenih operacija programiranjem alatnog stroja bio bi dug proces jer bušenje kao i svaka druga gore navedena operacija zahtijeva stalnu izmjenu upravljačkog koda ovisno o strojnom elementu koji se obrađuje.

Svi su CNC strojevi upravljani DXF datotekama jer je njihova zadaća da obavljaju različite zadatke koji se kontinuirano izmjenjuju pa bi ručno programiranje putem G-koda ili bilo kojeg programskog jezika bio složen i dugotrajan proces. Takvi CNC strojevi kao ulazni podatak primaju DXF datoteku te unutar vlastitog programskog alata za upravljanje, prevode DXF datoteku u G-kod ili neki drugi kod kojeg koriste te se zatim taj kod

izvršava. Neki strojevi zahtijevaju dodatno post procesiranje pa se koristi CAM kompjutorizirana proizvodnja koja daje stroju dodatne instrukcije.



Slika 2.1.1: Mach 3 program za upravljanje CNC strojem korištenjem DXF datoteke [2]

Na slici 2.1.1 prikazan je primjer programa Mach3 za upravljanje CNC strojem. Takvi i slični programi primaju DXF datoteku kao ulazni podatak te zatim tu datoteku dodatno prilagođava operator stroja. Nakon toga programski alat samostalno generira upravljački kod stroja.

Takva rješenja omogućuju vrlo brzo programiranje i reprogramiranje alatnog stroja bez da operator mora poznavati strukturu upravljačkoga koda numerički upravljana stroja. Drugi numerički upravljani strojevi koriste slične programske alate, razlika je samo u dodatnoj prilagodbi parametara ovisno o radnji koja se obavlja i o vrsti stroja. Glavni je cilj ovakvih programskih rješenja vrlo brzo dobivanje upravljačkog koda na temelju željene putanje iz DXF datoteke.

2.2 Slična rješenja trenutno dostupna na tržištu

Trenutno je na tržištu dostupno niz različitih programskih rješenja za *off-line* programiranje i simuliranje radnog okruženja robota. Takva programska okruženja znatno su ubrzala i pojednostavila programiranje industrijskih robota. Neki proizvođači robota nude vlastita programska rješenja dok postoje i tvrtke koje se bave isključivo izradom simulacijskih programskih okruženja za programiranje industrijskih robota. Tvrtke koje nude samo programska rješenja ona su obično univerzalna i u njih je moguće ubaciti bilo koji 3D model robota i njemu prikladan post procesor.

Programska rješenja slična alatu RoboDK:

- Octopuz
- Robotmaster

Octopuz:

Program za *off-line* programiranje industrijskih robota. Nudi mogućnost kreiranja vlastitih robota te pruža korisniku opciju uređivanja *Python* skripti koje se koriste za post procesiranje.

Robotmaster:

Program za CAD/CAM sustave u industrijskoj robotici koji posjeduje mogućnosti *off-line* programiranja i simuliranja, generiranje upravljačkih kodova robota, optimiziranje putanja i brojne druge mogućnosti.

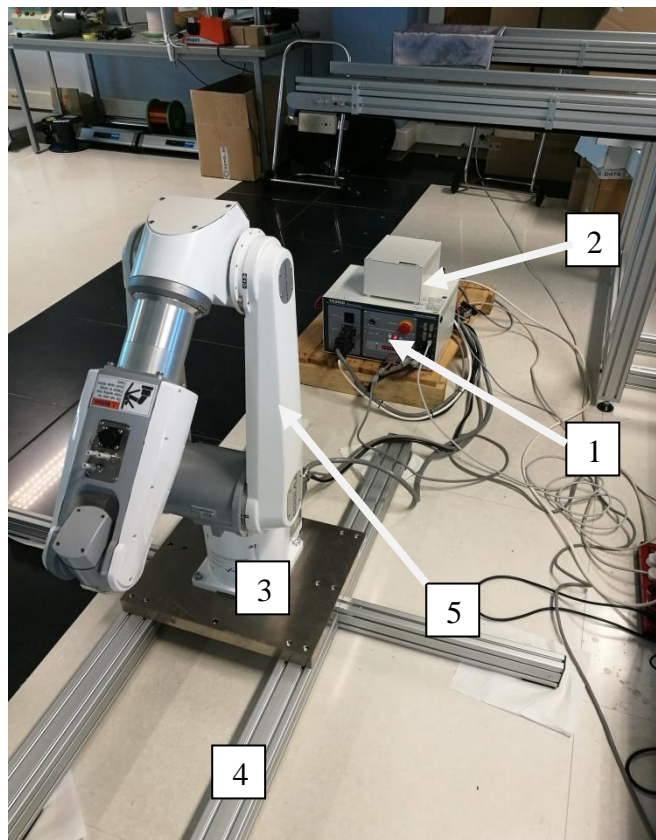
Svim gore navedenim programskim alatima zajednička je jedna stvar, koriste simulacijsko okruženje u kojem se robot programira *off-line* te se zatim taj program prevodi u upravljački kod robota pomoću integriranih post procesora. Integrirani post procesori obično su pisani u nekom od nižih programskih jezika kao što su *Python*, *C*, *C++* ili drugi. Također nude opciju dodavanja vlastitih 3D modela robota u simulacijsko okruženje.

3. KORIŠTENA OPREMA I PROGRAMSKI ALATI

Za izradu projekta korištena je *softverska* i *hardverska* oprema navedena u pod poglavljima. Navedene su neke od karakteristika opreme i tehnički podaci popraćeni slikama.

3.1 6- osni robot Toshiba TV-1000

Robot Toshiba TV-1000 6-osni je robot japanske korporacije Toshiba namijenjen za uporabu u industrijskoj robotici. Vrlo je robustan i male mase što mu omogućava velika ubrzanja i brzine kretanja, pogodan je za manipuliranje lakšim predmetima kod montaže. Spada u kategoriju robota s rotacijskim zglobovima tako zvane revolutne strukture te ima mogućnost nadogradnje još 2 stupnja slobode gibanja. Na slici 3.1.1. prikazan je industrijski robot Toshiba TV-1000 u svome radnom okruženju.



Slika 3.1.1: Robot Toshiba TV-1000 u radnom okruženju

Na slici 3.1.1. nalazi se:

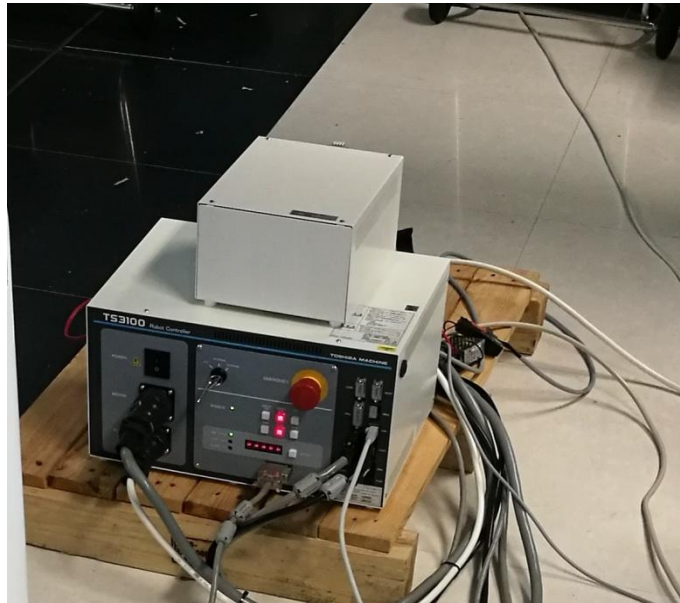
1. TS3100 kontroler robota
2. Relej za sigurnosnu zaštitu robota
3. Baza robota s utegom
4. Nosiva konstrukcija robota
5. Robot Toshiba TV-1000

Tablica 3.1.1: Tehnički podaci industrijskog robota Toshiba TV-1000

Dužina ruke	Ukupna dužina	1000 mm
	Prva ruka	480 mm
	Druga ruka	520 mm
	Doseg	1090 mm
Radni volumen	Zglob 1	$\pm 170^\circ$
	Zglob 2	$-100 \sim +150^\circ$
	Zglob 3	$-127 \sim +167^\circ$
	Zglob 4	$\pm 190^\circ$
	Zglob 5	$\pm 120^\circ$
	Zglob 6	$\pm 360^\circ$
Maksimalna brzina	Zglob 1	$237^\circ/s$
	Zglob 2	$240^\circ/s$
	Zglob 3	$288^\circ/s$
	Zglob 4	$350.5^\circ/s$
	Zglob 5	$484^\circ/s$
	Zglob 6	$576^\circ/s$
	Ukupna brzina	9.61 m/s
Teret	Maksimalna nosivost	5 kg
Ponovljivost pozicioniranja	X,Y,Z kod konstantne vanjske temperature	$\pm 0.03\text{mm}$
Metoda određivanja pozicije		Apsolutni enkoder
Ukupna masa robota		47 kg

3.2 TS-3100 kontroler robota

Kontroler TS-3100 koristi se za upravljanje SCARA robota i ostalih Toshiba robota s rotacijskim zglobovima. Podržava do 8 pojedinih rotacijskih osi. Posjeduje mnogo različitih funkcija a neke od njih su samo-dijagnostiku, aritmetičke operacije, ograničavanje momenta, komunikacija tijekom upravljanja robotom i brojne druge navedene u tehničkim karakteristikama kontrolera. Na slici 3.2.1 nalazi se kontroler.



Slika 3.2.1: TS-3100 Upravljačka jedinica robota

Tehnički podaci:

- Broj mogućih osi: maksimalno 6 + 2 dodatne
- Modovi kretanja: PTP (od točke do točke), CP (linearno, kružno), Short cut (najbrži put)
- Detekcija pozicije: Apsolutni enkoder
- Kapacitet pohrane: 12800 točaka + 25600 koraka, 1 program 2000 točaka + 3000 koraka.
- Može pohraniti: 256 programa
- Programski jezik: SCOL
- Broj ulaza/izlaza: 32 ulaza, 32 izlaza
- Komunikacija s računalom: RS-232C: 2 porta, Ethernet 1 port, USB
- Programski alat : TSPC, mogu se koristiti i neki drugi
- Dimenzije i masa: (D x Š x V) 420 x 241 x 298 mm , 17 kg

3.3 CNC vreteno

Vreteno je korišteno za bušenje provrta. Vrlo je male mase te je pogodno za korištenje na robotu Toshiba TV-1000. Uz malu masu i snažan motor pruža veliku izlaznu snagu. Uz vreteno korištena je i upravljačka elektronika za regulaciju broja okretaja vretena. Na slici 3.3.1. prikazano je vreteno.



Slika 3.3.1: EWL-4026 CNC vreteno

Tehničke karakteristike vretena:

- Marka: EWL-SF-Spindeln
- Model: 4026
- Napon: 3 do 40V AC
- Frekvencija: 250 – 1000 Hz
- Broj okretaja: 5000 – 60 000 o/min

Upravljačka jedinica vretena (kontroler)

Koristi se za upravljanje CNC vretenom. Daje varijabilni izlazni napon ovisno o željenom broju okretaja vretena.

Tehničke karakteristike:

- Ulazni napon: 220V 50/60 Hz
- Struja: 1.5 A
- Izlazni napon za vreteno: 5-26 V frekvencije 84-1000 Hz
- Broj okretaja vretena: 5000 - 60 000 o/min



Slika 3.3.2: Upravljačka jedinica CNC vretena

Na slici 3.3.2 nalazi se upravljačka jedinica CNC vretena na kojoj se može vidjeti potenciometar za namještanje brzine vrtnje vretena, prekidač za paljenje i gašenje i izlaz za napajanje vretena.

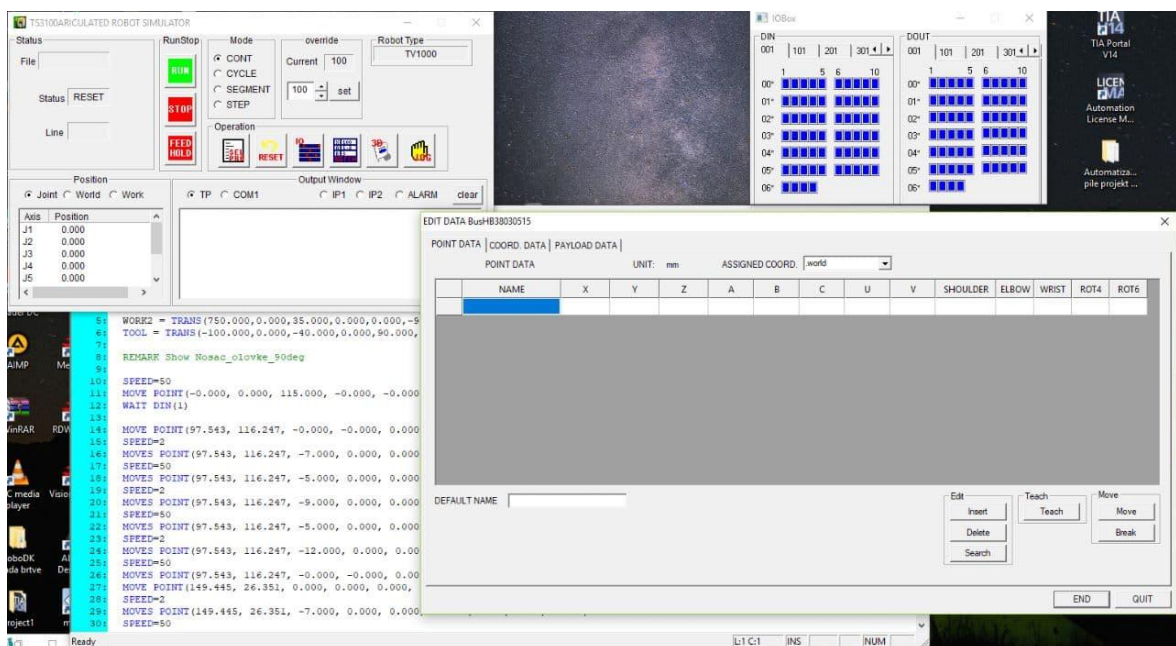
3.4 TSPC programski alat

TSPC programski je alat koji se koristi za programiranje Toshiba robota, neke verzije posjeduju mogućnost simuliranja. Vrlo je jednostavan za korištenje čak i za inženjere koji se prvi puta susreću s programiranjem industrijskih robota.

Karakteristike TSPC programskog alata:

- Jednostavan programski jezik (SCOL)
- Povezivanje putem Etherneteta
- Spremanje povijesti pozicija i alarma
- Provjera programskog koda na greške

Na slici 3.4.1 prikazano je sučelje programa TSPC koje se sastoji od više pojedinačnih prozora koji služe za podešavanje komunikacije s računalom, nadgledanje ulazno/izlaznih podataka, pisanje i uređivanje koda.



Slika 3.4.1: Prozori TSPC programskog alata

3.5 Programsko simulacijsko okruženje RoboDK

RoboDK programsko je rješenje za simuliranje radnog okruženja bilo kojeg industrijskog robota. Vrlo je jednostavan za korištenje, posjeduje bazu različitih robota te pruža mogućnost stvaranja vlastitog robota, ako on ne postoji u bazi podataka. Osim robota, u bazi podataka nalazi se niz druge popratne opreme da bi simulacija bila što potpunija. Pruža mogućnost ubacivanja vlastitih 3D modela za potpunu simulaciju radnog okruženja robota.

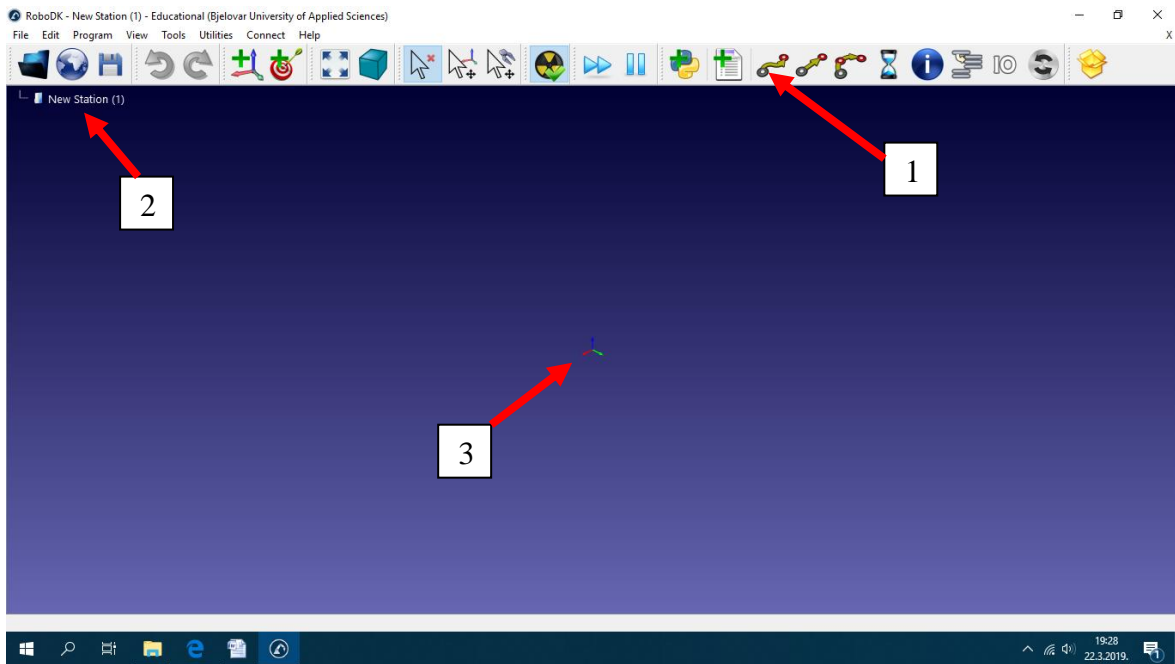
RoboDK omogućuje:

- 5 osnovno glodanje
- 3D printanje
- Pretvorbu NC programa (G-koda ili APT-CLS datoteke) u kod robota
- Offline programiranje
- Kalibriranje robota

Neke od dodatnih funkcija offline programiranja su:

- Detekcija kolizije
- Automatsko optimiziranje putanje
- Limitiranje pokreta svakog zgloba robota (engl. Axis Limit)

RoboDK posjeduje opciju kreiranja vlastitog robota u simulacijskom okruženju ubacivanjem 3D modela robota, definiranjem dimenzija robota te ostalih parametara robota. Osim kreiranja vlastitog robota, programeru nudi mogućnost uređivanja *Python* skripte post procesora.

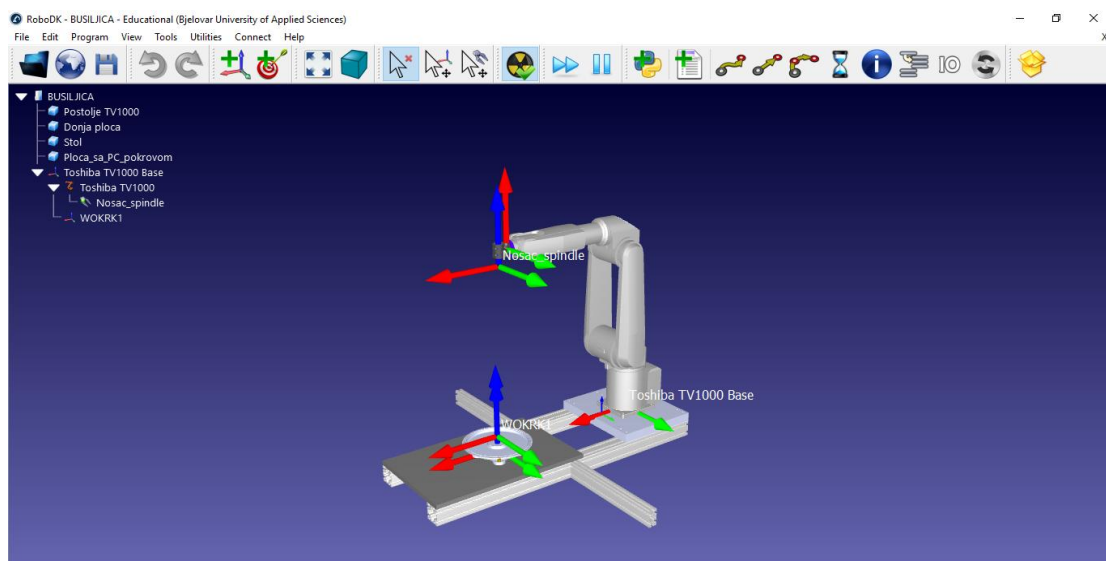


Slika 3.5.1: programsko simulacijsko okruženje RoboDK

Na slici 3.5.1. prikazano je programsko simulacijsko okruženje RoboDK koje se sastoji od:

1. Alatna traka
2. Hijerarhijsko stablo projekta
3. Referentni koordinatni sustav simulacijskog okruženja

Na slici 3.5.2. prikazan je primjer radnog okruženja u alatu RoboDK



Slika 3.5.2: Robot Toshiba TV-1000 u simulacijskom okruženju RoboDK

4. UPRAVLJAČKI KOD I UPRAVLJANJE ROBOTOM TOSHIBA TV-1000

Prvi korak u realizaciji zadatka je upoznavanje s načinima upravljanja 6-osnih industrijskih robota te upoznavanje sa strukturom upravljačkog koda industrijskog robota Toshiba TV-1000. U ovom poglavlju opisan je način pozicioniranja robota u prostoru, navedeni su neki od mogućih načina upravljanja, opisana je struktura koda industrijskog robota i moguće konfiguracije robota.

Industrijski roboti mogu zamijeniti ljude i raditi teške poslove bez umaranja, no potrebno ih je programirati za određeni zadatak. Programiranje robota je davanje niza instrukcija upravljačkom sučelju robota što i kako mora činiti. [3] Te instrukcije slijedno se izvršavaju liniju po liniju i tek kada je robot pozicioniran u točku koja je opisana jednom linijom koda izvršava se sljedeća linija. Robot će izvršiti točno onu naredbu koja mu se zada zato je vrlo bitno znati kako pravilno zadati robotu onu naredbu koja će obaviti željenu radnju. Postoje različiti načini programiranja robota.

Jedan od načina je da se vođenjem robot kroz radne točke u modu za učenje (engl. Teaching mode) određena radnja nauči robota i zatim robot ponavlja naučenu radnju. Taj način programiranja koristi se za programiranje jednostavne manipulacije i točkastog zavarivanja. Drugi i najčešće korišten način je pisanje upravljačkog koda u programskom jeziku robota. U nastavku će se govoriti samo o drugom načinu upravljanja pomoću programskog jezika robota.

Programiranje robota zahtijeva dobro poznavanje načina gibanja slobodnog tijela u kartezijskom prostoru, konfiguracije robota, definiranje raznih parametara kao što je masa alata koji robot koristi. To su vrlo bitne stavke čije poznavanje znatno olakšava programiranje robota.

4.1 Definiranje točka putanje u kartezijskom prostoru

Točke putanje 6 osnog robota definiraju se u kartezijskom desnokretnom koordinatnom sustavu koji je smješten u radnom prostoru robota. Ishodište koordinatnog sustava robota smješteno je u samu bazu robota te ono služi za određivanje pozicija ostalih referentnih koordinatnih sustava.

Stupnjevi slobode gibanja 6-osnih robota:

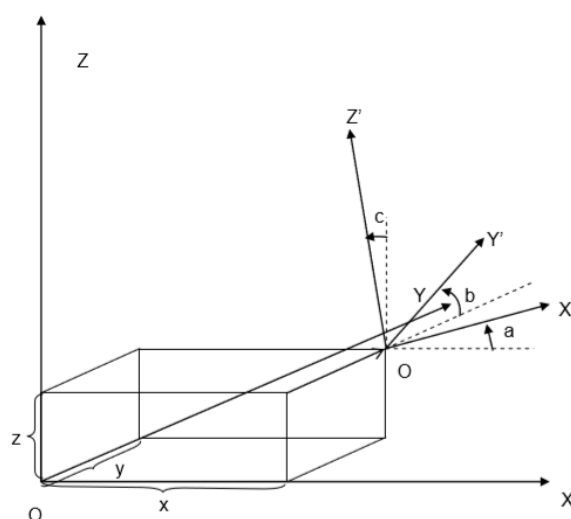
- 3 translacijska stupnja slobode gibanja
- 3 rotacijska stupnja slobode gibanja

Da bi uspješno naučili programirati bilo kojeg industrijskog robota potrebno je poznavati način na koji se definira točka putanje za robote sa 6 stupnjeva slobode gibanja. Postoje 2 osnovna koraka u definiranju nove točke u radnom prostoru robota.

Određivanje nove točke u radnom prostoru robota:

- Translacija referentnog koordinatnog sustava - pozicioniranje
- Rotacija novonastalog koordinatnog sustava - orijentacija

Na slici 4.1.1 prikazana je transformacija referentnog koordinatnog sustava u 2 koraka



Slika 4.1.1: Transformacija koordinatnog sustava [3]

Opis transformiranja koordinatnog sustava prikazanog na slici 4.1.1:

Translacija koja se sastoji od:

- Translacija referentnog koordinatnog sustava uzduž X osi
- Translacija referentnog koordinatnog sustava uzduž Y osi
- Translacija referentnog koordinatnog sustava uzduž Z osi

Referentni koordinatni sustav O smješten je u bazi robota. Translacijska komponenta definirana je vektorom pozicije u prostoru koji sadrži 3 komponente a to su udaljenost od referentnog koordinatnog sustava po X, Y i Z koordinatnoj osi.

Translacijom koordinatnog sustava uzduž sve 3 osi kartezijevog koordinatnog sustava nastaje novi koordinatni sustav s ishodištem u točki O' te koordinatnim osima X', Y' i Z' . Hvatište vektora pozicije nalazi se u ishodištu robota u točki O , kraj vektora nalazi se ishodištu koordinatnog sustava novonastalog ishodišta novog koordinatnog sustava.

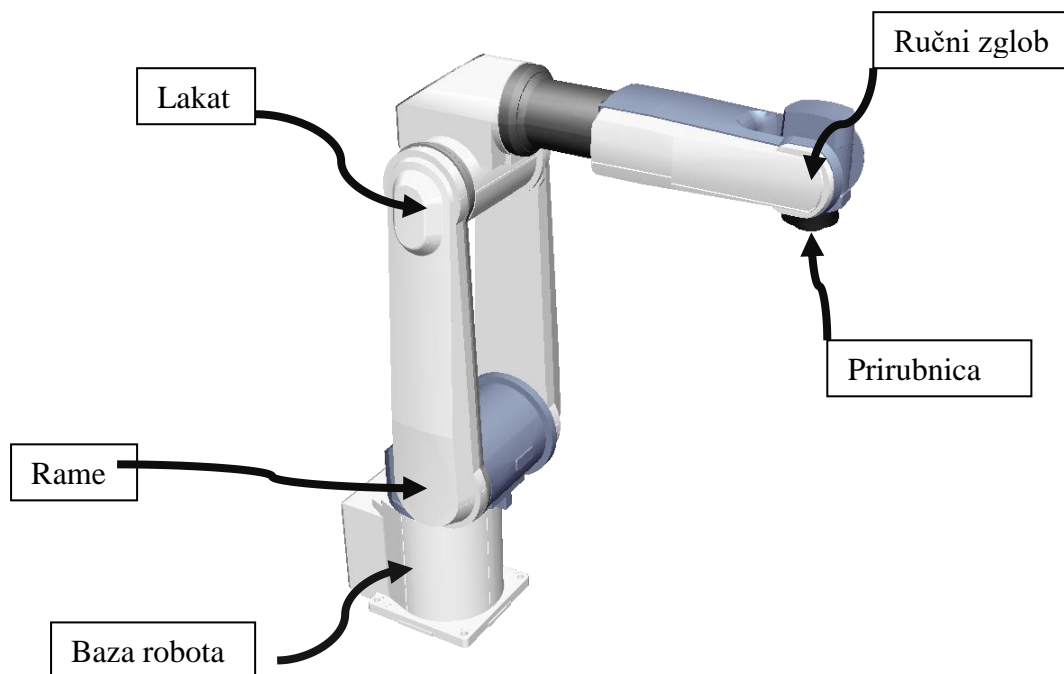
Rotacija koja se sastoji od:

- Rotacija referentnog koordinatnog sustava oko X osi za vrijednost kuta A
- Rotacija referentnog koordinatnog sustava oko Y osi za vrijednost kuta B
- Rotacija referentnog koordinatnog sustava oko Z osi za vrijednost kuta C

Novonastali koordinatni sustav O' rotira oko svake pojedine osi za iznose kuta A, B, C. Iznos kuta može se interpretirati na više načina. Svaki roboti koriste drugačiju interpretaciju rotacije. Kod robota Toshiba koristi se $X \rightarrow Y \rightarrow Z$ interpretacije.

4.2 Konfiguracija robota

6 osni industrijski robot može se postaviti u određenu točku na više mogućih načina. Ti se načini definiraju konfiguracijom pojedinog zglobova robota. Slično kao i ljudska ruka 6 osni robot posjeduje rame, lakat, ručni zglob i prirubnicu. Postoji razlika između zglobova robota i zglobova ljudske ruke. Lakat robotske ruke ima slobodan kut gibanja od 250° a neki zglobovi i 360° i tu se razlikuje od ljudske ruke. Da bi znali odrediti konfiguraciju robota treba poznavati dijelove 6 - osnoga robota (slika 4.2.1).



Slika 4.2.1: Dijelovi 6 osnog robota Toshiba TV-1000 [3]

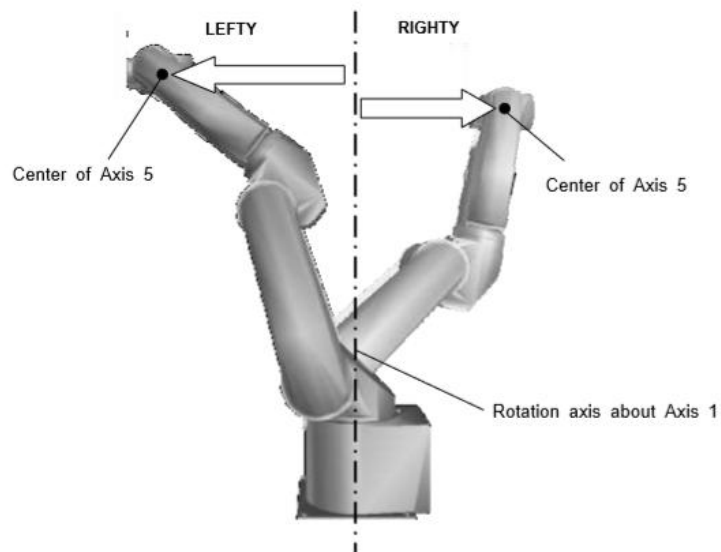
Konfiguracija robota definirana je konfiguracijama svakog pojedinog zgloba:

- Konfiguraciju ramena
- Konfiguraciju lakta
- Konfiguraciju ručnog zgloba
- Konfiguracija četvrte i pete osi

Konfiguracija ramena može biti:

- **Lefty**- Kada se centar pete osi rotacije robota(ručnog zgloba) nalazi ispred centra prve osi rotacije tada se konfiguracija ramena naziva LEFTY
- **Righty**- Kada se centar pete osi rotacije robota(ručnog zgloba) nalazi iza prve osi rotacije tada se konfiguracija ramena naziva RIGHTY

Na slici 4.2.2 prikazana je konfiguracija LEFTY (lijevo na slici) i RIGHTY (desno na slici)

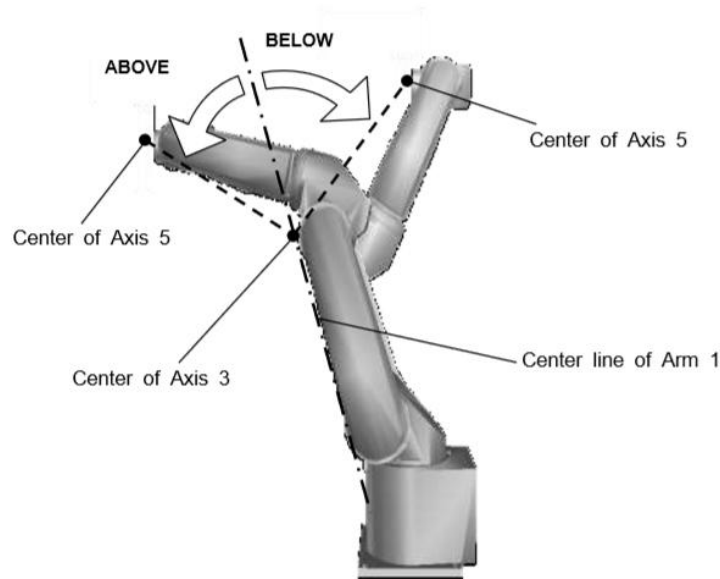


Slika 4.2.2: Konfiguracija ramena robota [3]

Konfiguracija lakta može biti:

- **Above** -Kada je centar pete osi robotske ruke ispod središnje linije prve ruke robota tada se konfiguracija lakta naziva ABOVE.
- **Below**-Kada je centar pete osi robotske ruke iznad središnje linije prve ruke robota tada se konfiguracija lakta naziva BELOW.

Na slici 4.2.3 prikazana je Above i Below konfiguracija

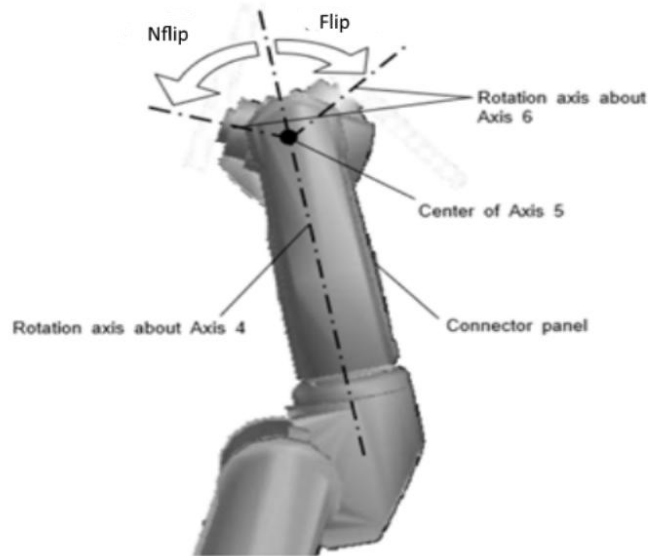


Slika 4.2.3: konfiguracija lakta robota [3]

Konfiguracija ručnog zgloba može biti:

- **Nflip**- Kada je kut pete osi rotacije jednak nuli ili veći od nule tada se konfiguracija pete osi naziva NFLIP.
- **Flip** Kada je kut pete osi rotacije manji od nule tada se konfiguracija pete osi rotacije naziva FLIP.

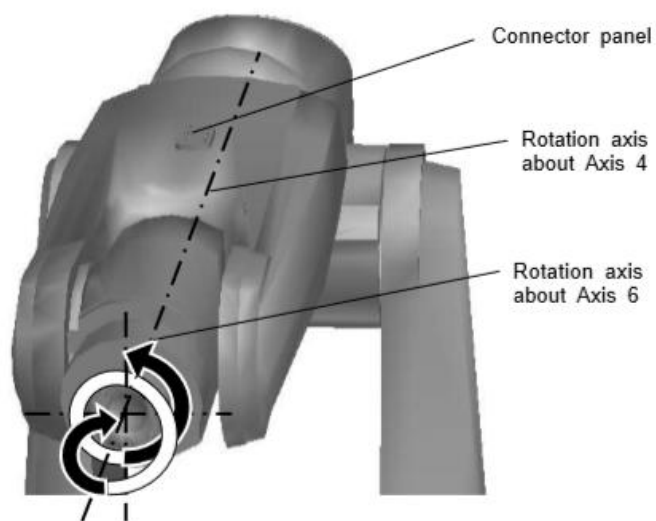
Konfiguracije pete osi Flip i Nflip prikazane su na slici 4.2.4



Slika 4.2.4: konfiguracija ručnog zgloba [3]

Konfiguracija 4. i 6. osi:

- SINGLE 4 / SINGLE 6 – Ako je apsolutna pozicija četvrte ili šeste osi manja od 180° tada se konfiguracija naziva SINGLE 4 ili 6, ili oboje.
- DOUBLE 4 / DOUBLE 6 – $180^\circ \leq$ Apsolutna pozicija četvrte i šeste osi $< 360^\circ$ tada je konfiguracija DOUBLE:



Slika 4.2.5: Konfiguracija četvrte i šeste osi [3]

4.3 Struktura upravljačkog koda robota Toshiba TV-1000

Za razvoj post procesora koji će prevoditi DXF datoteku u kod robota potrebno je poznavati upravljački kod robota Toshiba TV-1000. U prethodnim pod poglavljima objašnjen je način definiranja nove točke u radnom prostoru robota te razne konfiguracije robota. U ovom će se pod poglavljju analizirati struktura upravljačkog koda robota čije poznavanje je nužno za razvoj post procesora.

Toshiba TV-1000 koristi SCOL programski jezik koji posjeduje 6 vrsti različitih naredbi:

- **Naredbe upravljanja pokretom** – naredbe koje se koriste za upravljanje pokretima robota, u tu kategoriju spadaju i naredbe za pauziranje, zaustavljanje ili resetiranje robota.
- **Naredbe za upravljanje programom** – Ove naredbe upravljaju izvršavanjem programa izvršavanjem samo jednog dijela programa, čekanje vanjskih senzora kao uvjet za daljnje izvršavanje.
- **I/O (Input/Output) naredbe** – Ove naredbe se koriste za čitanje ulaznih signala ili za postavljanje izlaznih signala.
- **Naredbe o stanju pokreta** – Ove naredbe se koriste za podešavanje konfiguracije, brzina različitih zglobova, brzine robota dok se nalazi u pokretu.
- **Naredbe za matematičke proračune** – Ove naredbe koriste se za proračun matematičkih funkcija kao što su trigonometrijske funkcije, kvadriranje, korjenovanje itd...
- **Naredbe za određivanje pokreta** – koriste se za provjeru pokreta naprimjer, ako želimo znati koliki je postotak pokreta izvršen, koliko je ciklusa robot izvršio i druge.

Sve ove naredbe koriste se u kombinaciji sa drugim naredbama, pravilnim rasporedom mogu se stvoriti kompleksni upravljački programi za sve potrebe industrije. Znanjem svih komandi programer ima potpunu kontrolu nad pokretima robota. U tablici 4.3.1 nalaze su sve vrste naredbi u programskom jeziku SCOL.

Tablica 4.3.1: Naredbe SCOL programskog jezika

VRSTA	SVRHA	NAREDBA
Naredbe upravljanja pokretom	Pomicanje robota	MOVE,MOVES,MOVEC,MOVEA,MOVEI
	Privremeno zaustavljanje robota	DELAY
	Pomicanje prihvatnice	OPEN1,OPENI1,OPEN2,OPENI2,CLOSE1,CLOSEI1,CLOSE2,CLOSEI2
	Operacija prekida ili resetiranja	BREAK,PAUSE,RESUME
Naredbe upravljanja programom	Nadzor izlaznih signala	ON~DO, IGNORE, IF~THEN~ELSE, WAIT, TIMER
	Upravljanje izvođenja programa	PROGRAM, END, GOTO, RCYCLE, RETURN, FOR, TO, STEP, NEXT, STOP, TASK, KILL, SWITCH, TID, MAXTASK, REMARK
	Komentari u programu	REMARK
Naredbe za upravljanje ulazom/izlazom	Čitanje i postavljanje ilaznih varijabli	DIN, DOUT, PULOUT, RESET, BCDIN, BCDOUT
	Čitanje i postavljanje komunikacijskih podataka	PRINT, INPUT
Naredbe za postavljanje uvjetia kretanja	Postavljanje uvjeta za upravljanje pokretima robota	CONFIG, ACCUR, ACCEL, DECEL, SPEED, PASS, ENABLE, DISABLE, NOWAIT, PAYLOAD, SWITCH, MOVESYNC
Naredbe za paletizaciju	Učitaj datoteku	LOADLIB
	Inicijaliziraj paletu	INITPLT
	Pomakni paletu na određenu poziciju	MOVEPLT
Naredbe za matematičke proračune	Proračuni za realne brojeve	SIN, COS, TAN, ASIN, ACOS, ATAN, ATAN2, SQRT, ABS, SGN, INT, REAL, LN, MOD, LOG10, EXP, AND , OR , NOT
	Izvođenje kalkulacija bez korištenja podataka o poziciji	HERE, DEST, POINT, TRANS
	Korištenje niza	DIM, AS
Naredbe za provjeru sanja pokreta	Provjera kretanja robota	MOTION, MOTIONT, REMAIN, REMAINT
	Provjera ciklusa pokreta	MODE, CONT, CYCLE, SEGENT
	Definiranje koordinatnog sustava	TOOL, BASE, WORK
Ostale	Definiranje varijable	GLOBAL, DANA, END
	Resetiraj i učitaj vrijednosti u programski file	RESTORE
	Spremi podatke i isključi napajanje	SAVEEND

Upravljački program sastoji se od:

- Početka programa
- Sadržaj programa
- Završetka programa

Početak svakog upravljačkog koda u programskom jeziku SCOL započinje naredbom PROGRAM i nazivom programa. Nakon deklariranja programa u nastavku slijede sadržaj programa koji se sastoji od niza upravljačkih naredbi. Kraj programa definira se naredbom koja kontrolira izvođenje programa a ona se naziva END. Na slici 4.3.1 prikazana je osnovna struktura SCOL programskog jezika.

```
PROGRAM <program name>  
  Contents of your program  
END
```

Slika 4.3.1: Osnovna struktura upravljačkog koda SCOL jezika [3]

Na slici 4.3.1 vidi se da svaki program započinje naredbom PROGRAM i završava naredbom END a u sredini se nalazi sadržaj, na slici 4.3.2 prikazan je primjer sadržaja programa.

```
PROGRAM SAMPLE      Program name "SAMPLE"  
REMARK SAMPLE      Comment  
SPEED=20            Set the movement speed to 20% of the maximum  
                    speed.  
MOVE A1             Move the robot to position A1.  
DELAY 0.5           Wait for 0.5 sec.  
MOVE A2             Move the robot to position A2.  
DELAY 0.5           Wait for 0.5 sec.  
END                 End of program
```

Slika 4.3.2. Primjer sadržaja upravljačkog koda [3]

Naredba MOVE daje robotu vektorski podatak u koju točku da se pomakne, one se još nazivaju i naredbe interpolacije. Robot može izvršiti pomicanje iz jedne točke u drugu na više načina. Vrsta kretanja deklarira se kombinacijom različitih naredaba pokreta ili interpolacijom. U tablici 4.3.2 prikazane su naredbe pokreta i značenje svake naredbe.

Tablica 4.3.2: Naredbe upravljanja pokretom

Naredba	Svrha
MOVE	Sinkroni pokret
MOVES	Linearna interpolacija
MOVEC	Kružna interpolacija
MOVEA	Apsolutni pokret jedne osi
MOVEI	Relativni pokret jedne osi
MOVEJ	Pokret na putanji kružnog luka

Struktura naredbe MOVE je u suštini vektor pozicije s 9 pojedinih podataka. Taj vektor se sastoji od koordinatnih podataka i podataka o konfiguraciji. Koordinatni podaci definiraju točku u prostoru a to su X,Y,Z,A,B,C, točnije njenu poziciju i orijentaciju a podaci o konfiguraciji definiraju se realnim brojevima od 0-2 i predstavljaju konfiguraciju robota.

Vektor pozicije sastoji se od podataka: (X, Y, Z, A, B, C, U, V, <konfiguracija>) gdje su U i V dodatne osi rotacije koje se trenutno ne koriste. Svaka pojedina konfiguracija definira se jednom integer vrijednošću od 0 do 2 kao što je prikazano u tablici 4.3.3. i ukupna kombinacija konfiguracije sastoji se od 5 znakova.

Tablica 4.3.3: Integer vrijednosti za konfiguracije robota

Rame	Lakat	Ručni zglobovi	Os 4	Os 6
0:Nedefinirano	0:Nedefinirano	0:Nedefinirano	0:Nedefinirano	0:Nedefinirano
1:LEFTY	1:ABOVE	1:NFLIP	1:SINGLE4	1:SINGLE6
2:RIGHTY	2:BELOW	2:FLIP	2:DOUBLE4	2:DOUBLE6

Iz tablice 4.3.3 vidljivo je da postoji 5 pojedinih konfiguracija te će ona biti zapisana kao niz *intera* na kraju naredbe MOVE ili definirana naredbom CONFIG. U tablici 4.3.4 na nekoliko je primjera prikazan rezultat nekoliko različitih kombinacija konfiguracije.

Tablica 4.3.4: Primjeri *integer* zapisa konfiguracije robota

Konfiguracija robota	Integer zapis konfiguracije
LEFTY,BELOW,NFLIP,DOUBLE4,DOUBLE6	12122
RIGHTY,BELOW,FLIP,DOUBLE4,SINGLE6	22221

Ustanovljeno je da se u SCOL programskom jeziku naredba interpolacije sastoji od podataka pozicije i orijentacije koji su vektorski i izražavaju se u milimetrima ili stupnjevima i od podataka o konfiguraciji robota koji se izražavaju u *integer* vrijednosti od 0 do 2. Iza naredbe koja određuje vrstu interpolacije slijedi naredba POINT koja kao podatak prima 9 parametara a to su parametri pozicije i konfiguracije. Naredba POINT ujedno je dio naredbe MOVE. U slučaju da se ne koristi naredba POINT potrebno je prethodno deklarirati točku nekom varijablom te ju zatim pozivati iza naredbe MOVE. Na slici 4.3.3 prikazan je primjer upravljačkog koda robota Toshiba TV-1000 s naredbom interpolacije i konfiguracijom za svaku pojedinu točku.

```
PROGRAM Završni_Rad
REMARK Program generated by RoboDK v3.4.5 for Toshiba TV1000 on 05/04/2019 16:14:38
REMARK Using nominal kinematics.

BASE = TRANS(0.000,0.000,0.000,-0.000,0.000,-0.000)
TOOL = TRANS(-80.000,0.000,-40.000,0.000,90.000,0.000)

MOVE POINT(720.000, 0.000, 430.000, -0.000, -0.000, -0.000,0.000,0.000, 11211) WITH WORK = BASE
MOVES POINT(880.000, 0.000, 430.000, -0.000, 0.000, 0.000,0.000,0.000, 11211) WITH WORK = BASE

END
```

Slika 4.3.3. Primjer upravljačkog koda robota s *integer* vrijednostima konfiguracije robota

Na slici 4.3.3 također je vidljiva linija koda BASE i TOOL koja definira referentni koordinatni sustav i alata robota. Kod definiranja baze i alata koristi se naredba TRANS te zatim pozicija koordinatnog sustava alata.

5. DXF DATOTEKA

DXF datoteka je datoteka CAD alata koja služi za razmjenu podataka o crtežu između različitih računalnih programa. Datoteka je zapravo vektorski podatak o crtežu te se iz nje mogu iščitati točni vektorski podaci o crtežu. Upravo zbog tog razloga vrlo je pogodna za upravljanje numerički upravljanih strojeva i robota. U ovom poglavlju će se analizirati DXF datoteka i naglasiti njena uloga u upravljanju numerički upravljanih strojeva.

5.1 Uloga DXF datoteke u upravljanju numerički upravljanih strojeva

Sa brzim rastom moderne ekonomije, integracija CAD/CAM tehnologija postaje sve bitnija za poduzeća.[4] Uloga DXF datoteke u upravljanju numerički upravljanih strojeva može se povezati s pojmom mosta. Ona povezuje osnovne podatke o crtežu iz CAD alata s nekim drugim CAD alatom ili CNC strojem. Uz poboljšanje naprednih proizvodnih procesa rastu zahtjevi za integraciju više sustava, dobivanje grafičkih podataka iz DXF datoteka postaje sve potrebnije.[4] Danas je potpuno nezamislivo ručno programiranje G-kodom numerički upravljanoj stroja. Strojevi koji zahtijevaju stalno reprogramiranje koriste DXF datoteke kao bazu za kreiranje putanje stroja koja se kasnije optimizira post procesorom. DXF nije jedini vektorski format koji se danas koristi, no on je najkompatibilniji za razmjenu podataka između svih CAD alata ili CNC strojeva.

Ostali vektorski formati za razmjenu CAD podataka:

- DWG
- DWF
- SVG
- EPS

Svi gore navedeni formati datoteka također su vektorski, no razlika je u tome što je puno lakše analizirati DXF datoteku koja je u ASCII tekstualnom formatu nego neku binarnu datoteku kao što je DWG.

Prednosti korištenja DXF datoteke za upravljanje:

- Vrlo je jednostavna za analizirati
- Omogućuje brzu izmjenu upravljačkog koda
- Brzo uređivanje postojeće putanje

5.2 Sadržaj DXF datoteke

Datoteka DXF formata je *open-source* datoteka CAD alata u vektorskom obliku. DXF format je standardni format ASCII tekstualne datoteke.[5] DXF format je metoda prikaza podataka koja sadrži označene podatke svih informacija iz AutoCAD DWG datoteke. DXF datoteka je *open-source* i podaci o tome od čega se sastoji i kako ju analizirati javno su dostupni.

Podaci u DXF datoteci podijeljeni su u grupe, svaka varijabla pripada određenoj grupi podataka. Svaka grupa se sastoji od kodova a kodovi svake grupe su *integer* brojevne vrijednosti i mogu se kretati u rasponu od 0 do 1071 (slika 5.2.1). Svaki se kod grupe koristi u određenoj situaciji i predstavlja razne tipove podataka. Važno je naglasiti da se varijable raznih podataka u DXF datoteci reprezentiraju kao *integer*, *float* ili kao tekst odnosno *string* tip podatka.

Group code value types	
Code range	Group value type
0-9	String (with the introduction of extended symbol names in AutoCAD 2000, the 255-character limit has been increased to 2049 single-byte characters not including the newline at the end of the line)
10-39	Double precision 3D point value
40-59	Double-precision floating-point value
60-79	16-bit integer value
90-99	32-bit integer value
100	String (255-character maximum; less for Unicode strings)
102	String (255-character maximum; less for Unicode strings)
105	String representing hexadecimal (hex) handle value
110-119	Double precision floating-point value
120-129	Double precision floating-point value
130-139	Double precision floating-point value
140-149	Double precision scalar floating-point value
160-169	64-bit integer value
170-179	16-bit integer value
210-239	Double-precision floating-point value
270-279	16-bit integer value

Slika 5.2.1: Kod grupe i tip podatka koji ta grupa predstavlja [5]

Iz slike 5.2.1. vidljivo je da svaki kod grupe predstavlja broj od 0 – 1071 te su kodovi podijeljeni u neke raspone brojeva. Svaki raspon brojeva definira određeni tip podatka. Raspon od 0 do 9 definira *String* tip podatka, odnosno tekstualni.

DXF datoteka sastoji se od više odjeljaka:

- **Zaglavlje**
 - Predstavlja odjeljak u kojem se nalaze podaci o DXF datoteci, podaci koji se formiraju samim kreiranjem datoteke. U zaglavlju se mogu pronaći podaci o tome kojom verzijom Auto CAD alata je stvorena DXF datoteka, tko je stvorio datoteku i brojni drugi.

- **Razredi**
 - Odjeljak RAZREDI sadrži informacije za razrede definirane aplikacijama čiji se dijelovi pojavljuju u odjeljcima blokovi, entiteti i objekti.

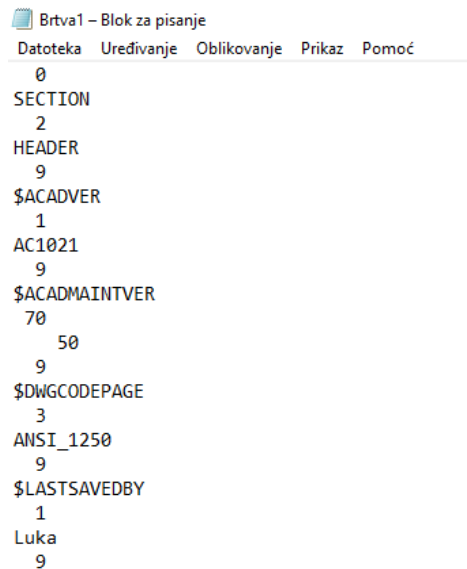
- **Tablice**
 - Ovaj odjeljak sadrži definicije za različite tablice i svaka tablica sadrži određene simbole. Ovdje se većinom deklariraju različiti simboli, stilovi teksta u DXF datoteci, vrste linija itd..

- **Blokovi**
 - Ovaj odjeljak sadrži grafičke objekte i crteže

- **Entiteti**
 - Ovaj odjeljak sadrži podatke o crtežu, objektu i o sastavnim dijelovima objekta. Krug se opisuje svojim radijusom, debljinom linije kojom je opisan te ako je 3D objekt svojom debljinom.

- **Objekti**
 - U ovome odjeljku nalaze se dijelovi crteža koji nemaju grafičko ili geometrijsko značenje u crtežu. Svi objekti koji nisu entiteti ili tablice zapisani su ovdje.

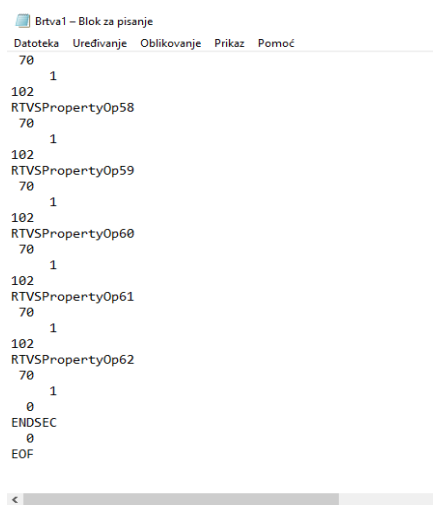
Svi odjeljci sastoje se od grupa a grupe su definirane prethodno spomenutim kodovima grupe. Na slici 5.2.2. nalazi se kod DXF datoteke.



```
Brтва1 – Blok za pisanje
Datoteka Uređivanje Oblikovanje Prikaz Pomoć
0
SECTION
2
HEADER
9
$ACADVER
1
AC1021
9
$ACADMAINTVER
70
50
9
$DWGCODEPAGE
3
ANSI_1250
9
$LASTSAVEDBY
1
Luka
9
```

Slika 5.2.2: Struktura DXF datoteke

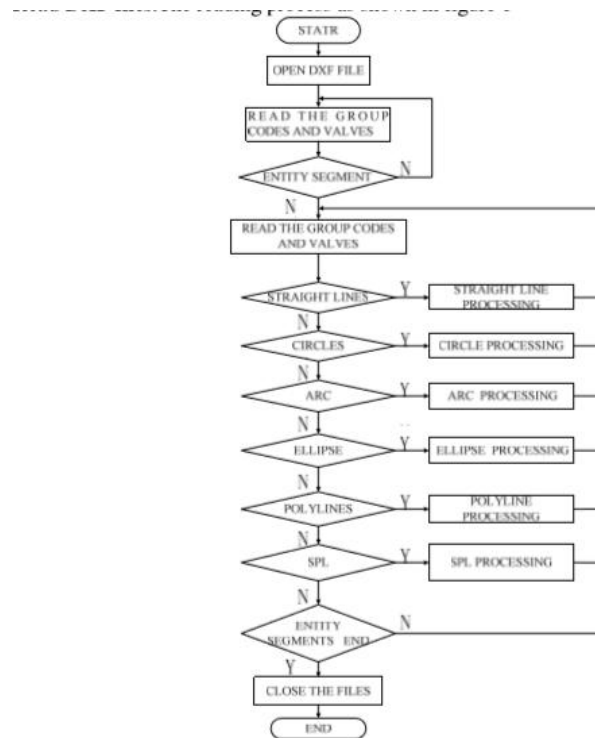
Na slici 5.2.2. vidljivo je da datoteka započinje kodom grupe 0 te slijedi *string* tip podatka pod nazivom SECTION. Kraj DXF datoteke također završava kodom grupe 0 i *stringom* ENDSEC. Iza ENDSEC slijedi kod grupe 0 i string EOF koji označava kraj datoteke slika (5.2.3) Vidljivo je da se sve svrstava u kodove grupe što dodatno olakšava snalaženje u DXF datoteci.



```
Brтва1 – Blok za pisanje
Datoteka Uređivanje Oblikovanje Prikaz Pomoć
70
1
102
RTVSPPropertyOp58
70
1
102
RTVSPPropertyOp59
70
1
102
RTVSPPropertyOp60
70
1
102
RTVSPPropertyOp61
70
1
102
RTVSPPropertyOp62
70
1
0
ENDSEC
0
EOF
```

Slika 5.2.3: Završetak DXF datoteke

Uz poznavanje strukture DXF datoteke korisno je napomenuti i način analiziranja datoteke s nekim drugim računalnim programom. Proces čitanja DXF datoteke prikazan je na slici 5.2.4



Slika 5.2.4: Funkcijski dijagram analiziranja DXF datoteke [6]

Na slici 5.2.4. s funkcijskog dijagrama može se vidjeti redosljed čitanja DXF datoteke. Vidljivo je da se prvo čitaju kodovi grupa te zatim entiteti. Ako ne postoje entiteti petlja se vraća na čitanje kodova grupa. U sljedećem koraku ako postoje segmenti nekog objekta tako se oni čitaju redom prvo ravne linije , krugovi, elipse te se svaka od njih procesira i vraća petljom nazad na čitanje. Kada su svi dijelovi crteža procesirani datoteka se zatvara.

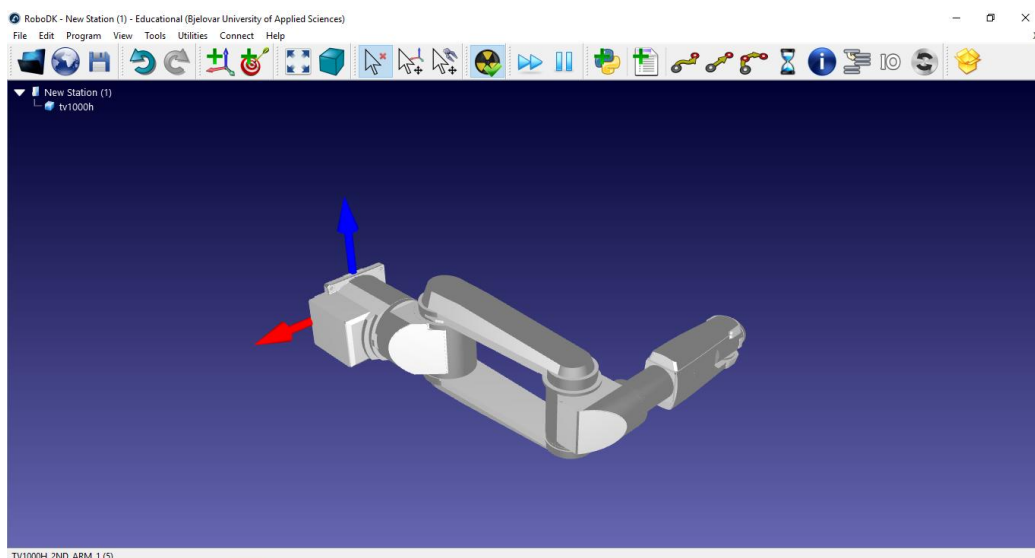
6. IZRADA SIMULACIJSKOG OKRUŽENJA U PROGRAMU ROBODK

Za realizaciju zadatka koristi se simulacijsko okruženje koje posjeduje opciju *off-line* programiranja. RoboDK je programsko simulacijsko okruženje koje nudi baš tu opciju i univerzalno je za sve robote. U simulacijskom okruženju prvo se kreira robot, zatim njegovo radno okruženje i alat kojim će se koristiti.

6.1 Izrada mehanizma robota

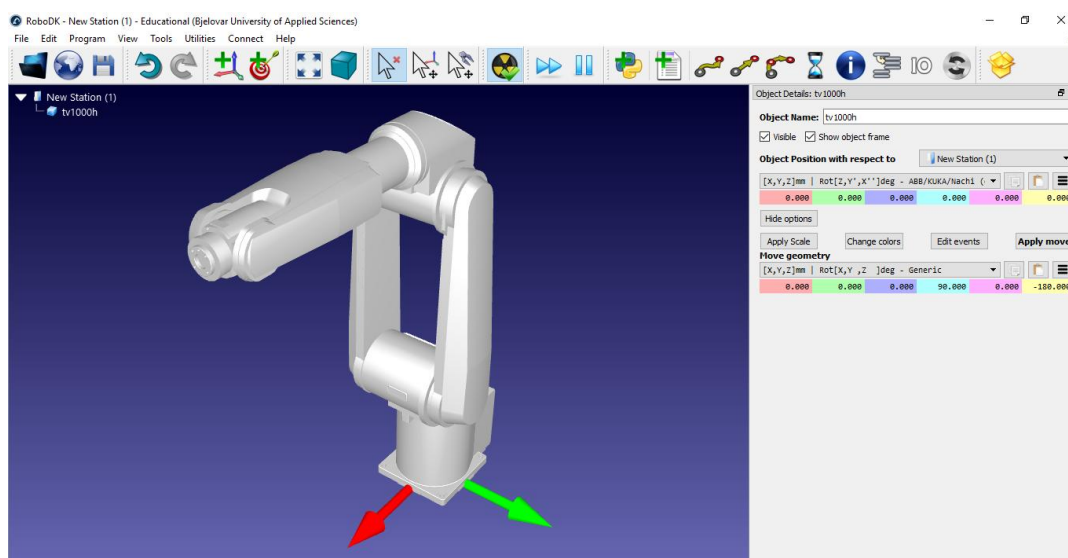
RoboDK posjeduje bazu robota, no ne nalaze se svi roboti u bazi podataka te je potrebno ubaciti 3D model robota i definirati novi mehanizam. 3D model može biti u bilo kojem 3D formatu kao što su IGES, STL, STP, OBJ ili drugi.

Kreiranje mehanizma robota kreće od ubacivanja 3D modela koji može biti u jednoj datoteci u dijelovima ili u više datoteka u dijelovima. Pritom je nužno da se 3D model sastoji od sastavnih dijelova robota te da svaka komponenta posjeduje svoj referentni koordinatni sustav. Na alatnoj traci odabiremo dio pod nazivom File → Open i ubacujemo 3D model. Na slici 6.1.1 prikazan je ubačeni 3D model robota toshiba TV-1000 u STP formatu.



Slika 6.1.1: Ubačeni 3D model robota u alat RoboDK

Nakon što je 3D model ubačen potrebno pravilno orijentirati robota u odnosu na referentni koordinatni sustav. Ovaj korak obavlja se sljedećim slijedom radnji: desni klik miša na objekt u projektnom stablu → Options → More options te se kao interpretacija rotacije odabire Generic u padajućem izborniku.

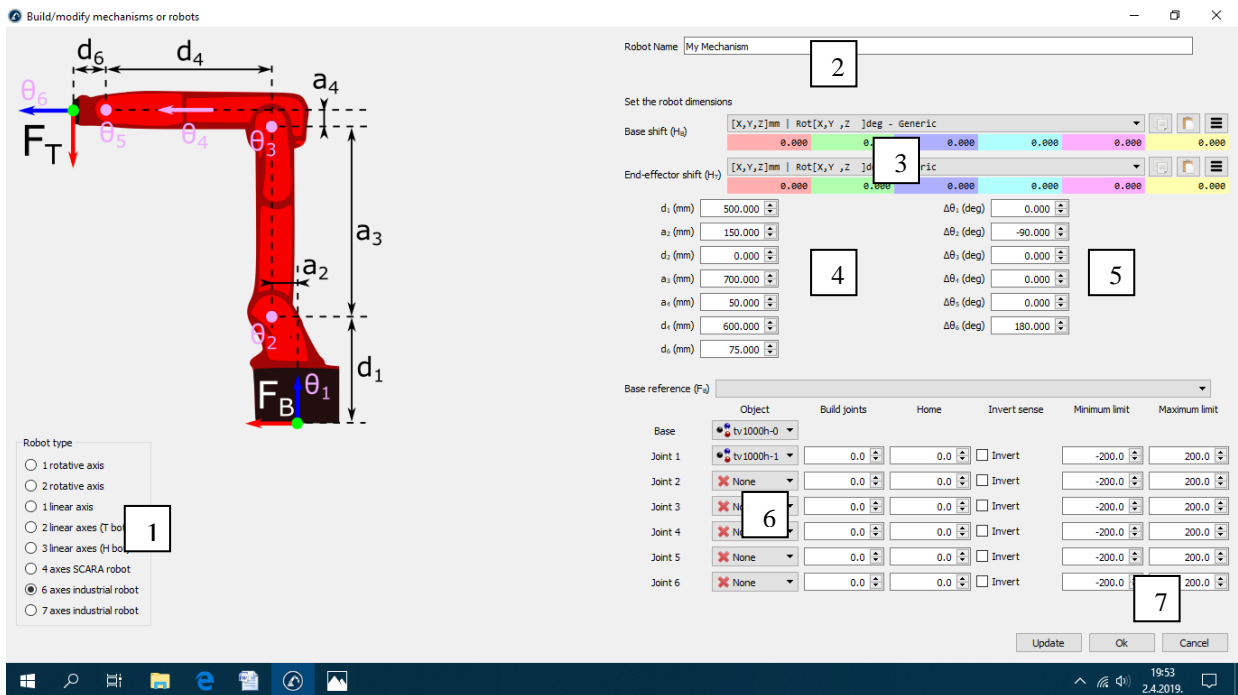


Slika 6.1.2: Pravilno orijentirani koordinatni sustav robota

Na slici 6.1.2 s desne strane nalaze se detalji o 3D modelu. Pod dijelom „Move geometry“ potrebno je zarotirati robota oko referentnog koordinatnog sustava tako da on bude jednako orijentiran kao na stvarnom robotu. Kada je orijentacija robota pravilno postavljena potrebno je potvrditi promjenu pritiskom na „Apply move“ da se trenutna orijentacija koordinatnog sustava robota postavi kao apsolutna.

Kada je 3D model pravilno orijentiran potrebno je rastaviti model robota na njegove sastavne komponente te se zatim može pristupiti dijaloškom okviru za kreiranje novog mehanizma. Model se rastavlja na dijelove desnim klikom na model robota → Split object.

Na alatnoj traci odabiremo Utilities → Model mechanism or Robot, otvara se dijaloški okvir (slika 6.1.3) za kreiranje novog mehanizma robota te u sljedećem koraku slijedi kreiranje robota.



Slika 6.1.3: Dijaloški okvir za kreiranje mehanizma robota

Dijaloški okvir za kreiranje robota sastoji se od:

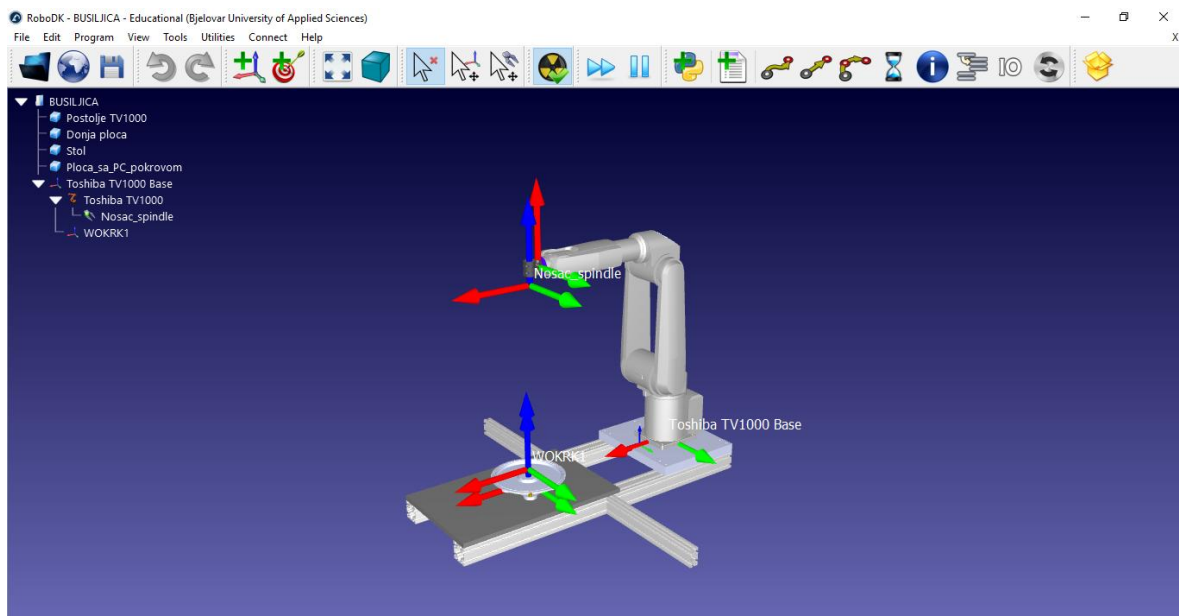
1. Izbornika za odabir vrste robota
2. Polje za unasanje imena robota
3. Polja za definiranje koordinatnih sustava baze i prirubnice
4. Polje za unasanje dimenzija robota
5. Polja za unasanje početnog kuta pojedinih osi robota
6. Polja za definiranje fizičkog 3D modela zgloba robota
7. Polje za definiranje opsega kretanja svakog pojedinog zgloba

Sve podatke potrebno je uskladiti s onima na stvarnom modelu robota da bi simulacija bila potpuna. Vrlo je važno naglasiti pravilnu orijentaciju koordinatnih sustava baze robota i prirubnice robota te je bitno dobro pročitati korisnički priručnik (engl. *User manual*) robota. Ako podaci nisu pravilno uneseni može doći do odstupanja kod puštanja programa u rad na pravom robotu ili čak do neželjenih i neplaniranih grešaka u kodu.

6.2 Izrada radnog okruženja robota

Simulacija radnog okruženja robota vrlo je bitna kod *off-line* programiranja. Potrebno je simulirati realni prostor robota ubacivanjem raznih 3D modela objekata dimenzija i oblika istovjetnih stvarnima. RoboDK posjeduje mogućnost provjere kolizije robota u slučaju da robot na svojoj putanji dodiruje neki statični objekt u radnom okruženju te će kod puštanja simulacije u rad RoboDK izbaciti pogrešku.

Podržani su svi oblici 3D datoteka, postupak ubacivanja jednak je kao postupak ubacivanja 3D modela robota. Na alatnoj traci odabire se File → Open te se ubacuje željeni objekt. Vrlo je bitno paziti na orijentaciju pojedinih objekata jer svaki novo ubačeni objekt posjeduje vlastiti koordinatni sustav te je moguće koordinatni sustav tog objekta uzimati kao referencu kod *off-line* programiranja robota. O odabiru referentnih koordinatnih sustava odlučuje programer.



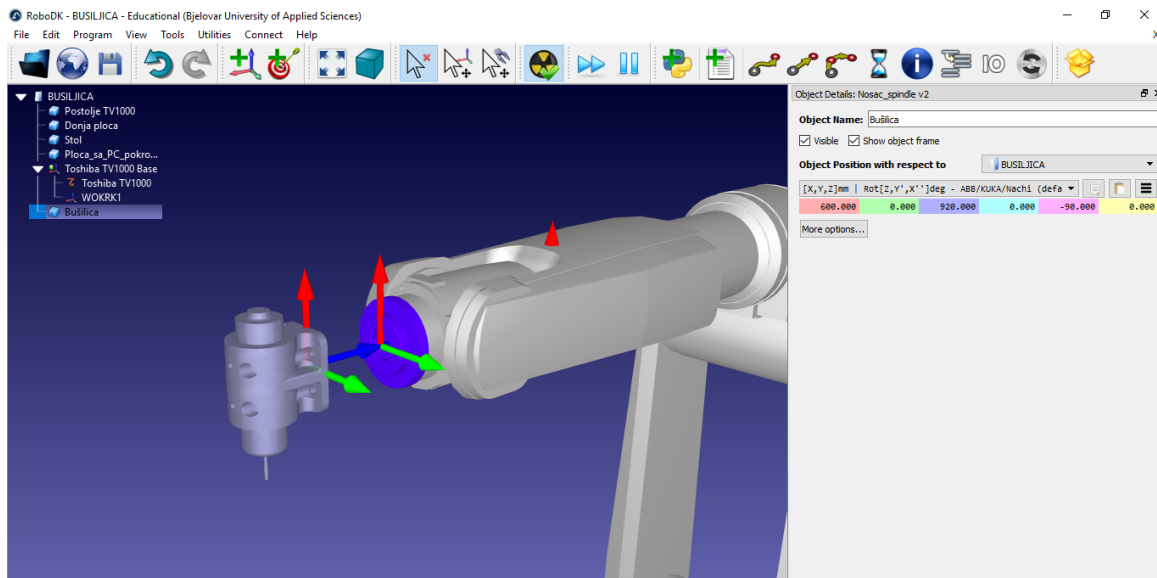
Slika 6.2.1: Radno okruženje robota sa svim koordinatnim sustavima

Na slici 6.2.1 nalazi se radno okruženje robota Toshiba TV1000 sa stolom za montiranje radnog komada za bušenje, postoljem, utegom robota i alat robota. U gornjem lijevom kutu nalazi se projektno stablo na kojem se može vidjeti hijerarhija koordinatnih sustava baze robota, WORK sustava i alata robota.

6.3 Dodavanje alata robota

Ako robot obavlja neki zadatak, koji god on bio mora postojati alat koji će obaviti željeni zadatak. Kod dodavanja alata u simulacijsko okruženje RoboDK vrlo je važno pravilno postaviti koordinatni sustav alata (TCP) i znati koju točku na alatu uzeti kao referentnu. Alat se kao i ostali objekti može učitati u više raznih 3D formata te se on kod samog ubacivanja ne razlikuje od ostalih 3D modela. Kod ubacivanja potrebno je i deklarirati taj 3D objekt kao alat.

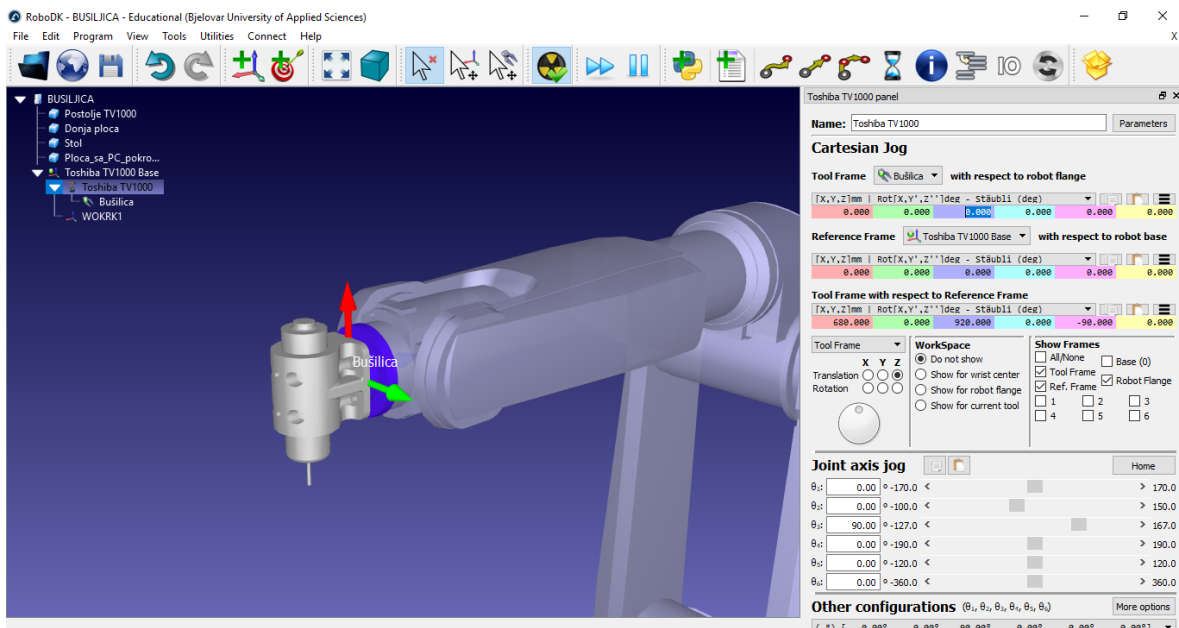
Referentni sustav za određivanje koordinatnog sustava alata je uvijek onaj od prirubnice robota na koju se montira alat robota. Na slici 6.3.1 prikazan je model alata te koordinatni sustav alata i prirubnice.



Slika 6.3.1: Koordinatni sustav prirubnice i alata

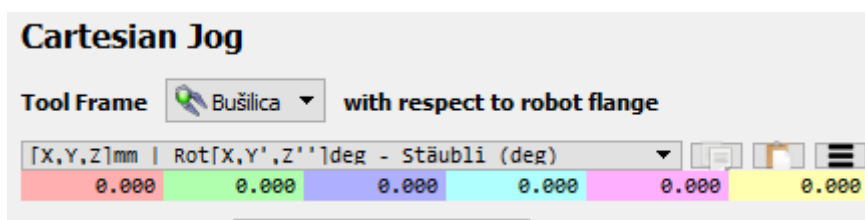
Desno na slici vidi se da je koordinatni sustav alata u odnosu na koordinatni sustav baze robota zarotiran za 90° oko Y osi.

Na hijerarhijskom stablu projekta nalazi se 3D model alata, da bi se taj model mogao koristiti kao alat potrebno ga je deklarirati kao alat. Taj korak obavlja se na način da se pomicanjem 3D modela u hijerarhijskom stablu na ikonu robota alat automatski postavlja na prirubnicu robot na način da se koordinatni sustav alata poklopi sa koordinatnim sustavom prirubnice kao što je vidljivo na slici 6.3.2.



Slika 6.3.2: Alat postavljen na prirubnici robota

Kada je alat postavljen potrebno je pravilno odrediti TCP alata. Alat koji se koristi je bušilica te je potrebno postaviti koordinatni sustav na sam vrh svrdla za bušenje. Koordinatni sustav alata (TCP) postavlja se u prozoru Tool Frame prikazanom na slici 6.3.3.



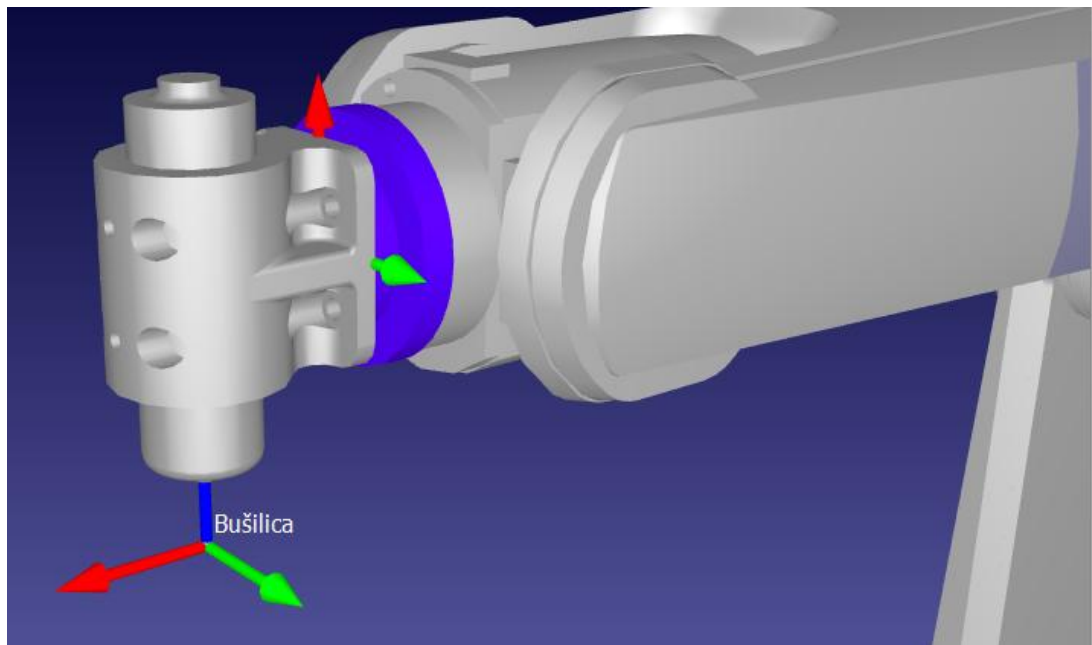
Slika 6.3.3: Polje za postavljanje koordinatnog sustava alata

Redom s desna na lijevo se nalaze podaci o poziciji X, Y i Z te podaci o orijentaciji A, B i C. Potrebno je postaviti koordinatni sustav na vrh svrdla bušilice, dakle translirati koordinatni sustav od pozicije prirubnice do vrha svrdla te zatim rotirati koordinatni sustav ako je potrebno. U ovome je koraku potrebno izmjeriti udaljenost vrha svrdla od središta prirubnice.

Tablica 6.3.1: Pozicija koordinatnog sustava alata u odnosu na prirubnicu robota

Točka	Pozicija					
	X	Y	Z	A	B	C
Točka 1 (prirubnica)	0	0	0	0	0	0
Točka 2 (vrh alata)	-80	0	-40	0	90	0

U tablici 6.3.1 prikazane su koordinate vrha alata u odnosu na referentni koordinatni sustav prirubnice robota. Translacija je primjenjena samo na X i Y osi, rotacija oko Y osi za iznos kuta B 90 °. Na slici 6.3.4 prikazan je koordinatni sustav alata posmaknut za iznose navedene u tablici 6.3.1.



Slika 6.3.4: Koordinatni sustava alata i prirubnice robota.

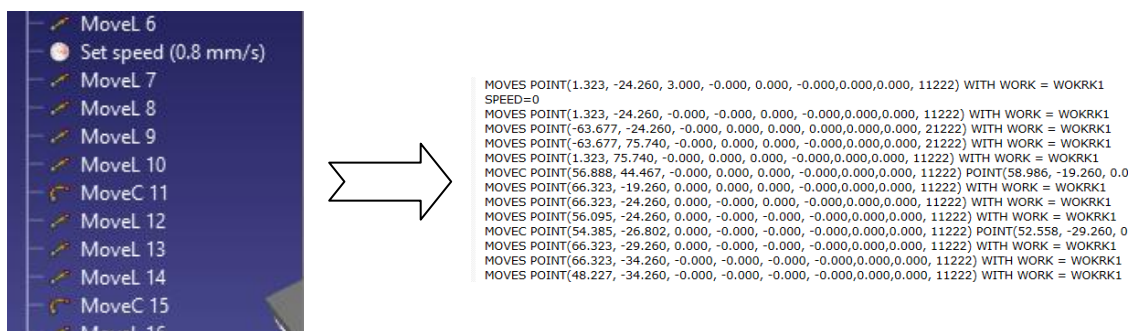
Na slici je vidljiva transformacija koordinatnog sustava prirubnice dvjema translacijama uzduž X i Z osi te jednom rotacijom oko Y osi. Kada je alat postavljen tada on predstavlja poziciju robota u prostoru, dakle njegova udaljenost od referentnog koordinatnog sustava reprezentira se kao pozicija robota u prostoru.

7. RAZVOJ POSTPROCESORA ZA PREVOĐENJE DXF DATOTEKE U KOD ROBOTA

Post procesor je skripta koja služi za prevođenje DXF datoteka ili drugih CAD modela u kod CNC stroja ili robotskog mehanizma. Uvijek se koristi zajedno s programom za off-line programiranje ili kao CAM alat iz razloga što se G-kod ili kod robota može razlikovati od stroja do stroja. Nekada je potrebno definirati posebne alate i režime rada stroja pa bi obavljanje takvog prevođenja u kod stroja, bez nekog dodatnog simulacijskog alata bilo nemoguće izvesti.

RoboDK simulacijski je alat u čijem je sklopu post procesor za prevođenje simulacije u kod bilo kojeg industrijskog robota. Svaki robot posjeduje svoj post procesor, ako on ne postoji u bazi podataka potrebno je na temelju predložka prilagoditi post procesor željenom modelu robota.

Programski jezik u kojem su napisani svi post procesori u RoboDK je Pythone. RoboDK omogućava uređivanje svih postojećih post procesora ili stvaranje novih. Post procesor je kao prevoditelj koji razumije jezik svih robota i može izvršiti prijevod jednog univerzalnog jezika u jezik bilo kojeg robota. RoboDK na temelju DXF datoteke stvara simulaciju te kao rezultat simulacije poseban niz naredbi unutar RoboDK koji se potom prevodi u upravljački kod robota kao što je prikazano na slici 7.1.



Slika 7.1: Prevedeni niz naredbi simulacije u upravljački kod robota

7.1 Razvoj skripte post procesora

Post procesor mora na temelju simulacije u RoboDK generirati upravljački kod robota. Simulacija kreira program koji se sastoji od niza naredbi za deklariranje koordinatnih sustava, vrste interpolacija, brzina robota, ubrzanja itd.. Sve te podatke potrebno je obuhvatiti post procesorom i prevesti ih u programski jezik razumljiv robotu. RoboDK u nizu upravljačkih instrukcija također generira varijable te inicijalizira razne podatke.

RoboDK generira sljedeće osnovne naredbe:

- Deklarira referentni koordinatni sustav
- Deklarira koordinatni sustav alata
- Kreira interpolacije:
 - Linearna interpolacija
 - Kružna interpolacija
 - Slobodna
- Ulazne/izlazne varijable
- Brzina robota
- Konfiguracija robota

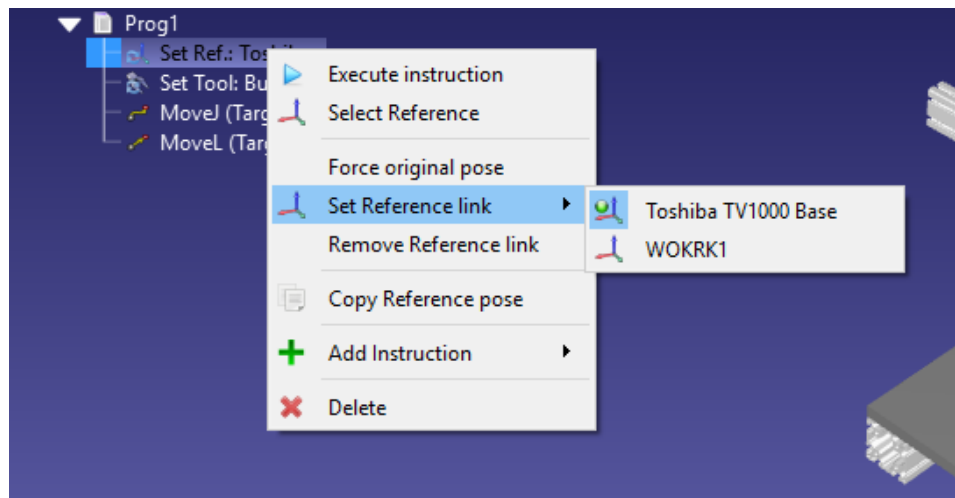
Sve gore navedene parametre generira RoboDK tokom *off-line* programiranja i zatim ih svrstava u programski kod koji se nalazi u hijerarhijskom stablu projekta. Taj niz instrukcija univerzalan je za simulaciju bilo kojeg robota, no ne i za upravljački kod realnog robota. Zato se kod *off-line* programiranja prevodi neki upravljački kod specifičan za taj programski alaz u kod bilo kojeg modela robota pomoću post procesora.

Prevođenje koda može se podijeliti na:

- Prevođenje definiranih varijabli
- Prevođenje naredbi interpolacije
- Prevođenje ulazno/izlaznih varijabli
- Prevođenje podataka o konfiguraciji

7.1.1 *Post procesiranje varijabli*

Prva stvar koju je potrebno prevesti su varijable jer se u njih upisuju podaci koji su kasnije potrebni u izvršavanju upravljačkoga koda. Neke osnovne varijable koje generira RoboDK su pozicija referentnog koordinatnog sustava te pozicija alata, ako on postoji. Kada robot mora stalno mijenjati poziciju na kojoj nešto obavlja svaka se pozicija deklarira svojim referentnim radnim koordinatnim sustavom. U post procesoru je potrebno svaki koordinatni sustav (ako ih postoji više) spremi u novu varijablu.



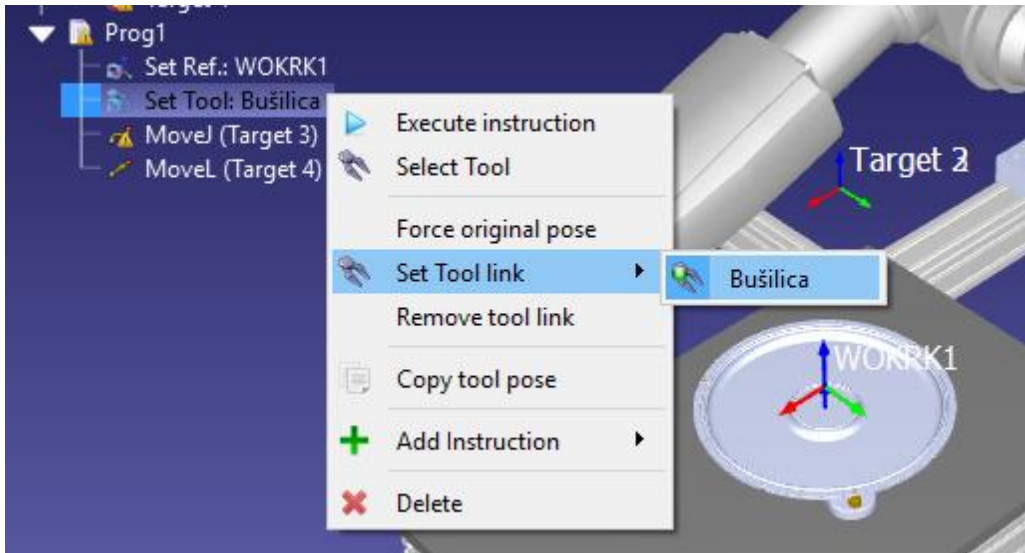
Slika 7.1.1.1 Postavljanje referentnog koordinatnog sustava robota u RoboDk

Na slici 7.1.1.1 prikazan je način postavljanja referentnog koordinatnog sustava u RoboDK alatu. U post procesoru taj dio koda mora prevesti naredbu za postavljanje referentnog koordinatnog sustava u naredbu SCOL programskog jezika. SCOL programski jezik prima poziciju koordinatnog sustava u varijablu koja se zatim upotrebljava u upravljačkom kodu .

```
#SetReffFrame
def setFrame(self, pose, frame_id=None, frame_name=None):
    """Change the robot reference frame"""
    global work_name      "Globalna varijabla u koju se sprema ime koordinatnog sustava"
    work_name = frame_name
    self.addline('%s = %s' % (frame_name, pose_2_str(pose)))
```

Slika 7.1.1.2 Dio koda koji ispisuje liniju u kojoj deklarira poziciju koordinatnog sustava

Uz deklariranje pozicija i orijentacija referentnih koordinatnih sustava u SCOL programskom jeziku potrebno je deklarirati i alat robota. Alat robota deklarira se u RoboDK programskom linijom Set Tool kao što je prikazano na slici 7.1.1.3.



Slika 7.1.1.3: Odabir alata robota u RoboDK

Nakon što je odabran alat potrebno je prilagoditi taj dio post procesora da tu liniju koda prevede u onu prikladnu SCOL programskom jeziku i robotu Toshiba. Na slici 7.1.1.4 prikazan je dio skripte post procesora koji zapisuje TCP alata u TOOL varijablu za programski jezik SCOL

```
#SetTool
def setTool(self, pose, tool_id=None, tool_name=None):
    """Change the robot TCP"""
    self.addline('TOOL = %s' % pose_2_str(pose))
```

Slika 7.1.1.4: Dio skripte post procesora koji prevodi deklariranje alata robota

7.1.2 Post procesiranje upravljačkih naredbi

Dio koda koji upravlja robotom određuju naredbe interpolacije. One opisuju pokrete robota te koordinate točaka u prostoru. Od naredbi interpolacija sastoji se veći dio upravljačkog koda. RoboDK omogućuje 3 osnovne interpolacije kojima se mogu opisati sve putanje robota.

3 osnovne interpolacije u simulacijskom okruženju roboDK su:

- Linearna interpolacija
- Kružna interpolacija
- Slobodna interpolacija

RoboDK interpretira interpolacije na način koji je prikazan u tablici 7.1.2.1

Tablica 7.1.2.1 Usporedba naredbi interpolacije u RoboDK i u SCOL jeziku

Vrsta interpolacije	RoboDK	Toshiba TV-1000
Linearna	MoveL(pozicija)	MOVES
Kružna	MoveC(pozicija)	MOVEC
Slobodna	MoveJ(pozicija)	MOVE

```
1
#Move Synchronous
def MoveJ(self, pose, joints, conf_RLF=None):
    """Add a joint movement"""
    self.addline('MOVE %s WITH WORK = %s' % (target_2_str(pose, joints),work_name))

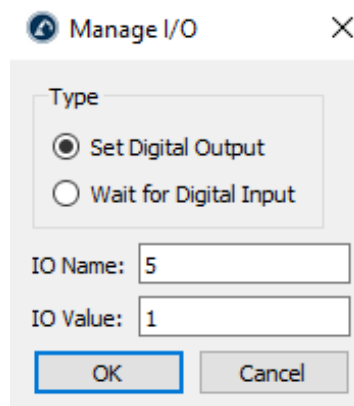
#MOVE Linear
def MoveL(self, pose, joints, conf_RLF=None):
    """Add a linear movement"""
    self.addline('MOVES %s WITH WORK = %s' % (target_2_str(pose, joints),work_name))

#MOVE Circular
def MoveC(self, pose1, joints1, pose2, joints2, conf_RLF_1=None, conf_RLF_2=None):
    """Add a circular movement"""
    self.addline('MOVEC %s %s WITH WORK = %s' % (target_2_str(pose1, joints1), target_2_str(pose2, joints2),work_name))
```

Slika 7.1.2.1. Dio skripte post procesora koji prevodi interpolacije koda robota

7.1.3 Post procesiranje ulazno / izlaznih podataka

Važan dio upravljanja izvršavanjem upravljačkog koda su ulazno/izlazni podaci. To u fizičkom svijetu mogu biti tipkala kao ulazne, razni podaci senzora ili kao izlazne signali da je neka linija koda izvršena. Robo DK omogućuje korisniku da doda u liniju naredbi dio koji deklarira čekanje na neki ulazni podatak ili postavljanje nekog izlaza. Na slici 7.1.3.1. prikazan je dijaloški okvir za dodavanje I/O naredbe u Robo DK.



Slika 7.1.3.1. Dodavanje I/O naredbe u RoboDK

```
Python 3.4.1: Toshiba_TV1000.py - C:\RoboDK\Posts\Toshiba_TV1000.py
File Edit Format Run Options Windows Help

#self.addlog('setZoneData not defined (%.1f mm)' % zone_mm)

#*****Digitalni ulazi/izlazi

def setDO(self, io_var, io_value):
    """Sets a variable (output) to a given value"""
    if type(io_var) != str: # set default variable name if io_var is a number
        io_var = "OUT[%s]" % str(io_var)
    if type(io_value) != str: # set default variable value if io_value is a number
        if io_value > 0:
            io_value = 'TRUE'
        else:
            io_value = 'FALSE'

    # at this point, io_var and io_value must be string values
    self.addline("%s=%s" % (io_var, io_value))

def waitDI(self, io_var, io_value, timeout_ms=1):
    """Waits for an input io_var to attain a given value io_value. Optionally, a timeout can be provided."""
    if type(io_var) != str: # set default variable name if io_var is a number
        io_var = "DIN[%s]" % str(io_var)
    if type(io_value) != str: # set default variable value if io_value is a number
        if io_value > 0:
            io_value = 'TRUE'
        else:
            io_value = 'FALSE'

    # at this point, io_var and io_value must be string values
    if timeout_ms <= 0:
        self.addline("WAIT %s" % (io_var))
    else:
        self.addline("WAIT %s" % (io_var))
        self.addline("DELAY %s.If" % (timeout_ms*0.001))

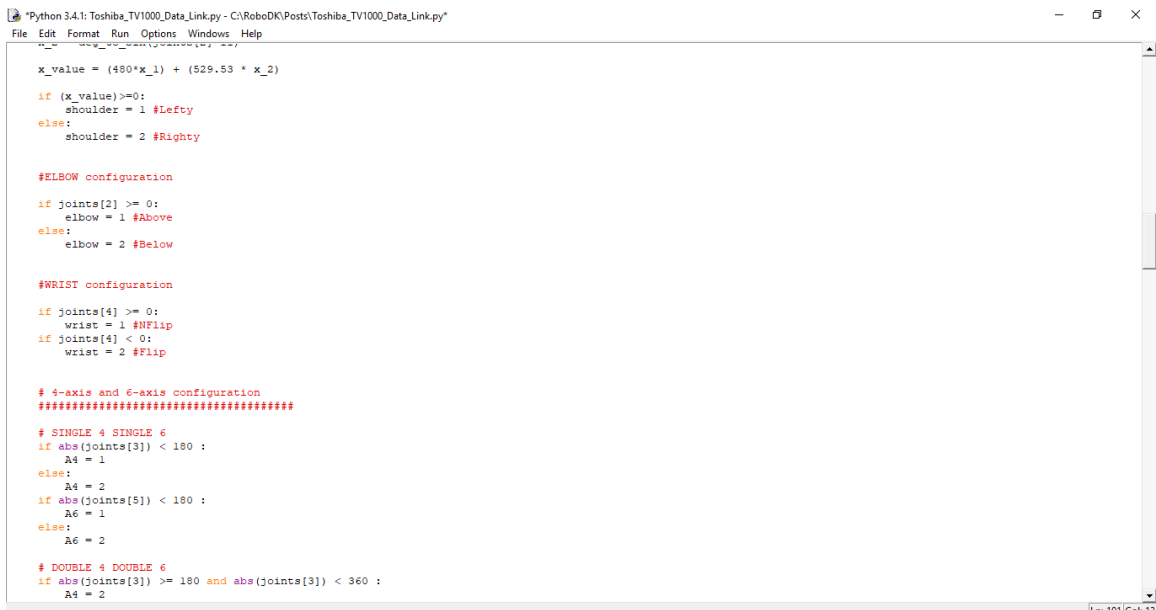
def RunCode(self, code, is_function_call = False):
    """Adds code or a function call"""
    if is_function_call:
        code.replace(' ', '')
        if code.find('(') < 0:
            code = code + '()'

    self.addline(code)
```

Slika 7.1.3.2. Dio skripte post procesora koji prevodi naredbe za ulazno / izlaznih naredbi

7.1.4 Post procesiranje podataka o konfiguraciji

Konfiguracijom robota dajemo robotu naredbu kako da se postavi u određenu točku. Svaki zglob robota ima svoju konfiguraciju. Nešto proizvoljnoj točki robot može pristupiti u raznim konfiguracijama. Na primjeru će se analizirati konfiguracija četvrte i šeste osi robota. Na slici 7.1.4.1 prikazane su neke od konfiguracija a na slici 7.1.4.2. prikazana je dio koda koji određuje konfiguraciju četvrte i šeste osi robota.



```
*Python 3.4.1: Toshiba_TV1000_Data_Link.py - C:\RoboDK\Posts\Toshiba_TV1000_Data_Link.py
File Edit Format Run Options Windows Help

x_value = (480*x_1) + (529.53 * x_2)

if (x_value)>=0:
    eShoulder = 1 #Lefty
else:
    shoulder = 2 #Righty

#ELBOW configuration
if joints[2] >= 0:
    elbow = 1 #Above
else:
    elbow = 2 #Below

#WRIST configuration
if joints[4] >= 0:
    wrist = 1 #NFlip
if joints[4] < 0:
    wrist = 2 #Flip

# 4-axis and 6-axis configuration
#####
# SINGLE 4 SINGLE 6
if abs(joints[3]) < 180 :
    A4 = 1
else:
    A4 = 2
if abs(joints[5]) < 180 :
    A6 = 1
else:
    A6 = 2
# DOUBLE 4 DOUBLE 6
if abs(joints[3]) >= 180 and abs(joints[3]) < 360 :
    A4 = 2
```

Slika 7.1.4.1: Dio post procesora koji određuje konfiguracije robota

```
# 4-axis and 6-axis configuration
#####

# SINGLE 4 SINGLE 6
if abs(joints[3]) < 180 :
    A4 = 1
else:
    A4 = 2
if abs(joints[5]) < 180 :
    A6 = 1
else:
    A6 = 2

# DOUBLE 4 DOUBLE 6
if abs(joints[3]) >= 180 and abs(joints[3]) < 360 :
    A4 = 2
if abs(joints[5]) >=180 and abs(joints[5]) < 360:
    A6 = 2
#####
```

Slika 7.1.4.2: Dio post procesora koji prevodi konfiguraciju 4. i 6. osi robota

Ako se prisjetimo poglavlja o konfiguracijama, konfiguracija 4. i 6. osi glasi:

- SINGLE 4 / SINGLE 6 – Ako je apsolutna pozicija četvrte ili šeste osi manja od 180° tada se konfiguracija naziva SINGLE 4 ili 6, ili oboje.
- DOUBLE 4 / DOUBLE 6 – $180^\circ \leq$ Apsolutna pozicija četvrte i šeste osi $< 360^\circ$ tada je konfiguracija DOUBLE:

Sa slike 7.1.4.2 u prvome dijelu se nalazi dio koda koji uzima apsolutnu poziciju četvrte i šeste osi rotacije i uspoređuje ih s uvjetima koji određuju da li je SINGLE ili DOUBLE konfiguracija. Sa slike se vidi da j+se uzimaju podaci iz joints[3] i joints [5] a razlog tomu je što se prva os rotacije gleda kao joints[0]. Rezultat konfiguracije zapisuje se u varijablu A, zatim se na kraju postavlja ukupna konfiguracija robota što je vidljivo na slici 7.1.4.3.

```
# DOUBLE 4 DOUBLE 6
if abs(joints[3]) >= 180 and abs(joints[3]) < 360 :
    A4 = 2
if abs(joints[5]) >=180 and abs(joints[5]) < 360:
    A6 = 2
#####

#Configuration array
CONFIG[0] = shoulder
CONFIG[1] = elbow
CONFIG[2] = wrist
CONFIG[3] = A4
CONFIG[4] = A6

# RETURN POINT
return ('POINT(%3f, %3f, %3f, %3f, %3f, %3f,0.000,0.000, %s)' % (x,y,z,r,p,w,''.join(map(str, CONFIG))))
```

Slika 7.1.4.3: Dio koda post procesora koji sprema konfiguraciju u string

Na slici 7.1.4.3 vidljivo je da se zaključna konfiguracija sprema u niz(string) od 5 znakova. Pod dijelom koda koji sadrži funkciju return ispisuje se linija koda koja sadrži sve post procesirane podatke. U liniji return ispisuju se podaci o poziciji, orijentaciji i konfiguraciji robota. Time je završeno post procesiranje u upravljački kod robota.

8. ISPITIVANJE RADA ROBOTA UPRAVLJANOG PUTANJOM IZ DXF DATOTEKE

Cilj projekta bio je na zahtjev tvrtke Data Link d.o.o. osmisliti sustav koji će na temelju putanje iz CAD alata spremljene u DXF format generirati upravljački kod robota za nacrtanu putanju. U ovom poglavlju navedena su dva primjera vođenja robota putanjom iz DXF datoteke.

8.1 Izrada brtve

Izrada brtvila *dispensanjem* mješavine korištenjem robota, metoda je koja se primjenjuje u auto industriji. Za potrebe tvrtke Dana Link d.o.o izrađen je sustav generiranja upravljačkog koda robota Toshiba TV-1000 na temelju bilo kojeg CAD dvodimenzionalnog crteža putanje. Takav sustav vrlo je fleksibilan, moguće je vrlo brzo izraditi složene upravljačke kodove za brtvu bilo kojega oblika. Za potrebe ispitivanja sustava putanja je nacrtana na papiru olovkom.

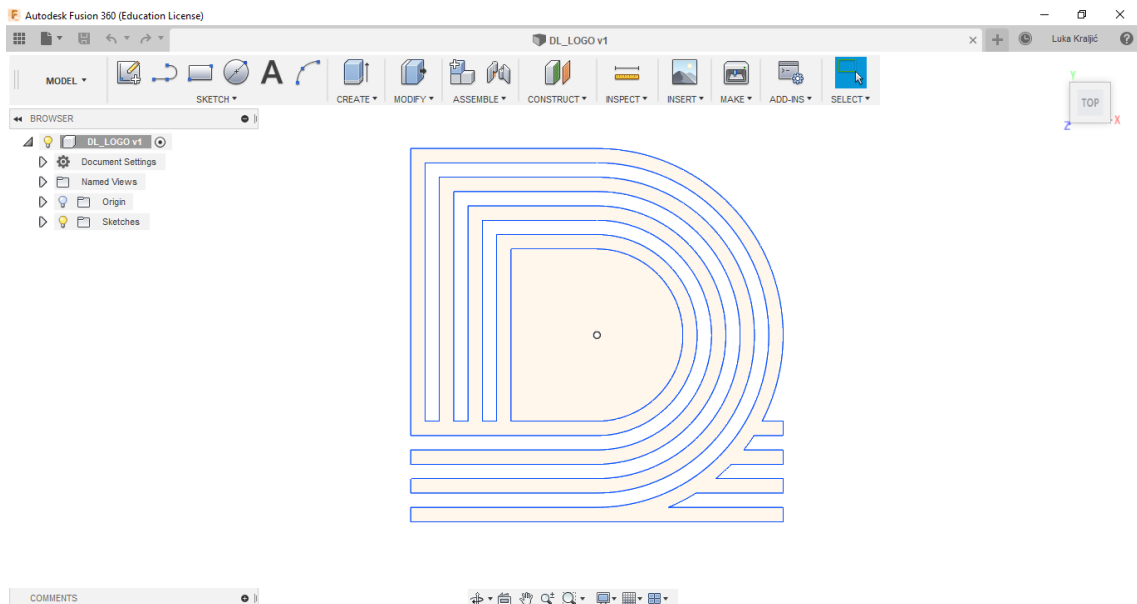
Sam proces kreiranja upravljačkog koda provodi se u RoboDK programsko - simulacijskom alatu, a upravljački kod robota generira post procesor koji je razvijen posebno za robot Toshiba TV-1000 i potrebe tvrtke Data Link d.o.o.

Proces se sastoji od nekoliko koraka:

- Izrada konture brtve u CAD alatu
- Nacrtana brtva sprema se u DXF format
- RoboDK kreiranje simulacije na temelju DXF putanje
- Post procesiranje putanje na temelju simulacije
- Upravljački kod robota se testira na robotu

Izrada konture brtve u CAD alatu

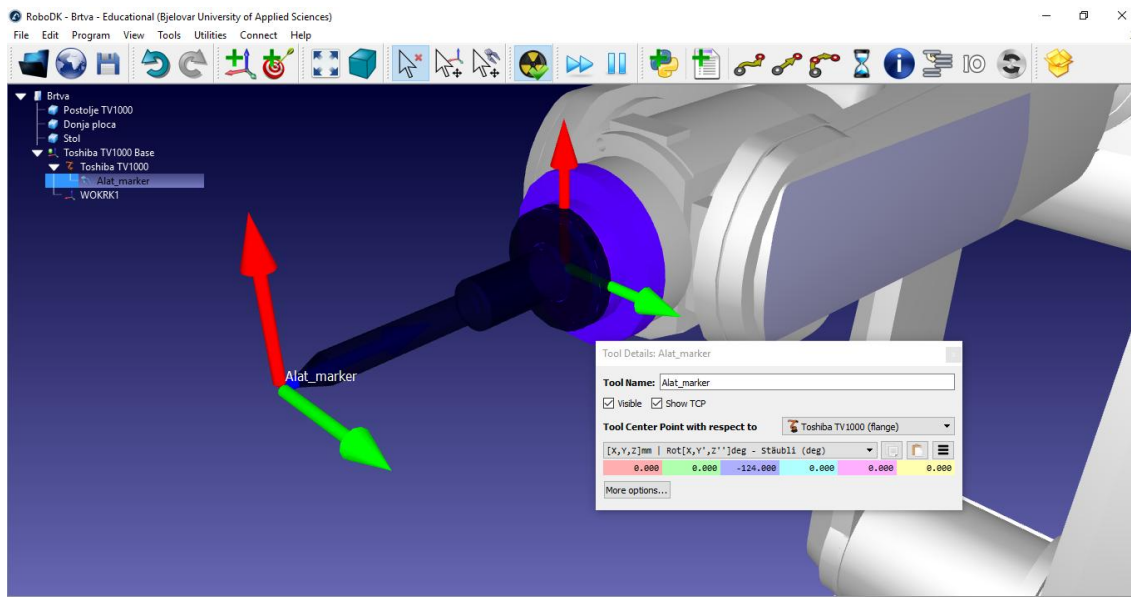
Za potrebe ispitivanja sustava nacrtana je proizvoljna putanja u CAD alatu te je spremljena u DXF format. Na slici 8.1.1. prikazana je kontura putanje u CAD alatu koja se potom sprema u DXF datotečni format i kao takva koristit će se za generiranje putanje.



Slika 8.1.1: Proizvoljna putanja u CAD alatu

Nakon što je putanja nacrtana sprema se u DXF format. Sljedeći korak je dodavanje alata koji će se koristiti za ispitivanje. Za svrhu ispitivanja korišten je flomaster koji je pričvršćen za nosač izrađen na 3D printeru.

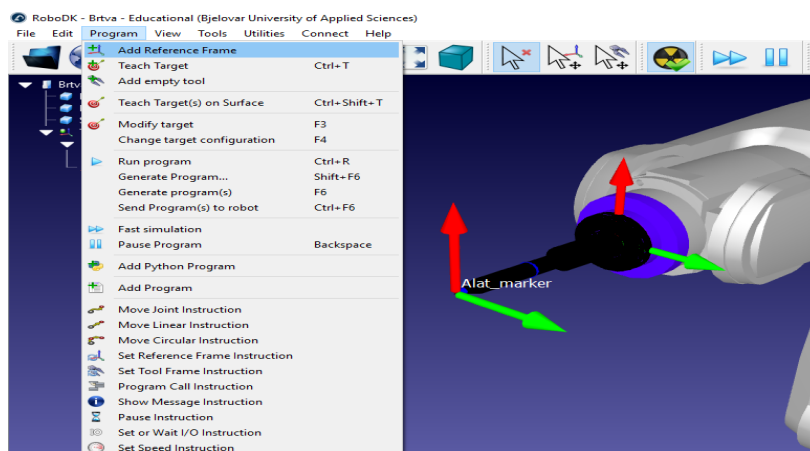
Sljedeći korak je prilagodba alata u simulacijskom okruženju RoboDK. Potrebno je postaviti koordinatni sustav alata na sam vrh flomastera koji će se potom koristiti za crtanje vizualnog prikaza putanje iz CAD alata. Mjerenjem pomičnim mjerilom utvrđuje se realna udaljenost vrha alata od prirubnice robota. U simulacijskom okruženju RoboDK potrebno je koordinatni sustav alata posmaknuti za točno one iznose izmjerenih vrijednosti na pravome robotu. Na slici 8.1.2. prikazan je alat u simulacijskom okruženju te pozicija njegovog koordinatnog sustava u odnosu na koordinatni sustav prirubnice.



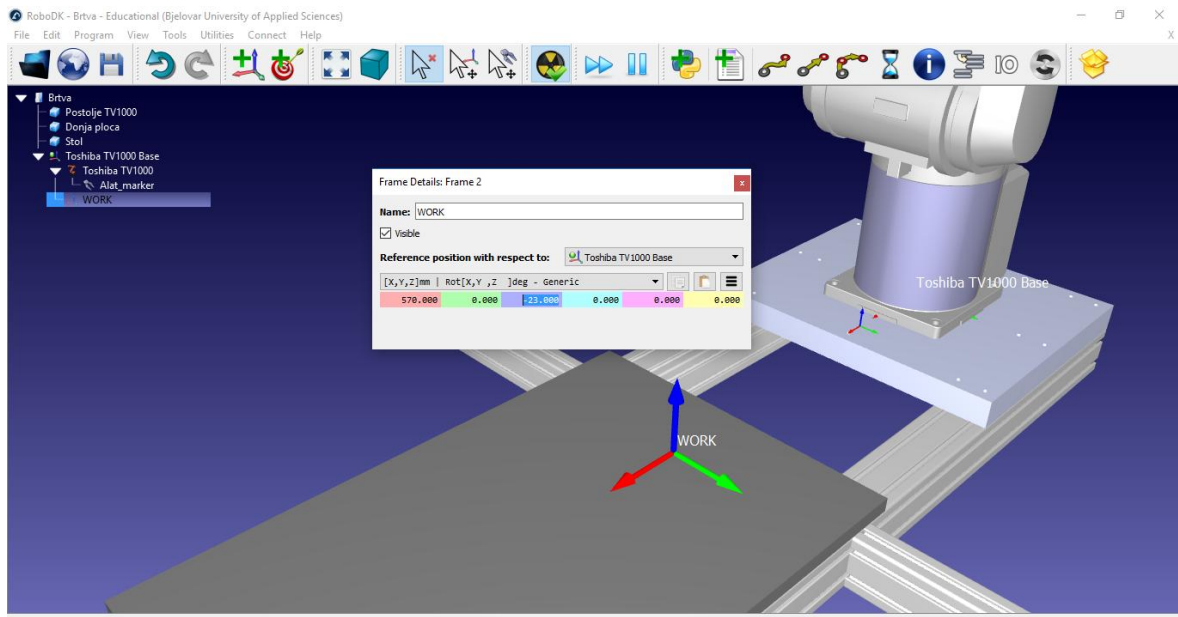
Slika 8.1.2: Koordinatni sustav alata i prirubnice

Iz slike 8.1.2. uočljivo je da je koordinatni sustav alata robota u odnosu na koordinatni sustav prirubnice pomaknut samo uzduž Z osi za vrijednost dužine alata. Kada je alat definiran potrebno je definirati radni WORK koordinatni sustav robota.

Novi koordinatni sustav dodaje se na način da se na alatnoj traci odabere izbornik Program → Add Reference Frame (Slika 8.1.3). Na projektnome stablu pojavljuje se novi koordinatni sustav kojeg je potrebno postaviti na željenu poziciju i dati mu ime WORK (Slika 8.1.4).



Slika 8.1.3: Dodavanje novog koordinatnog sustava

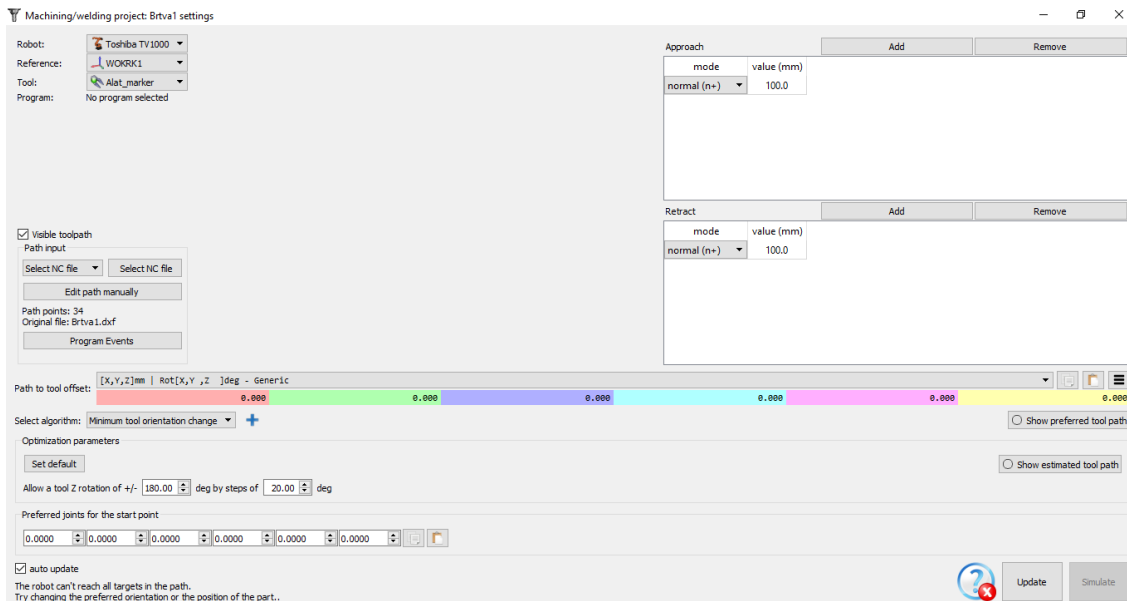


Slika 8.1.4: WORK koordinatni sustav

Na slici 8.1.4. vidljivo je da se WORK koordinatni sustav spušta se po Z osi za iznos od 23 mm u odnosu na bazu robota iz razloga što je ploča stola niža za točno taj iznos. Također se pomiče po X osi za vrijednost od 570 mm dabi robot mogao pristupiti radnome prostoru koji je definiran WORK koordinatnim sustavom.

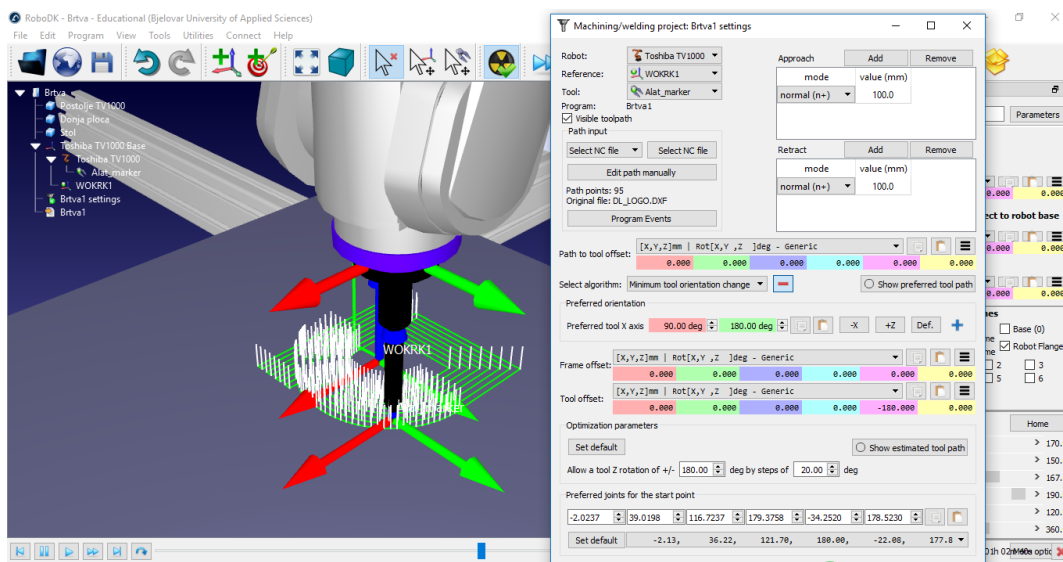
Izrada simulacije:

Kada je alat robota postavljen i WORK koordinatni sustav pozicioniran izrada simulacije može započeti. Ubacuje se DXF datoteka s konturom željene brtve. Odabire se FILE→ OPEN te se odabire DXF datoteka brtve. Kada se datoteka učita otvara se dijaloški okvir za kreiranje projekta simulacije na osnovu DXF datoteke prikazanog na slici 8.1.5.



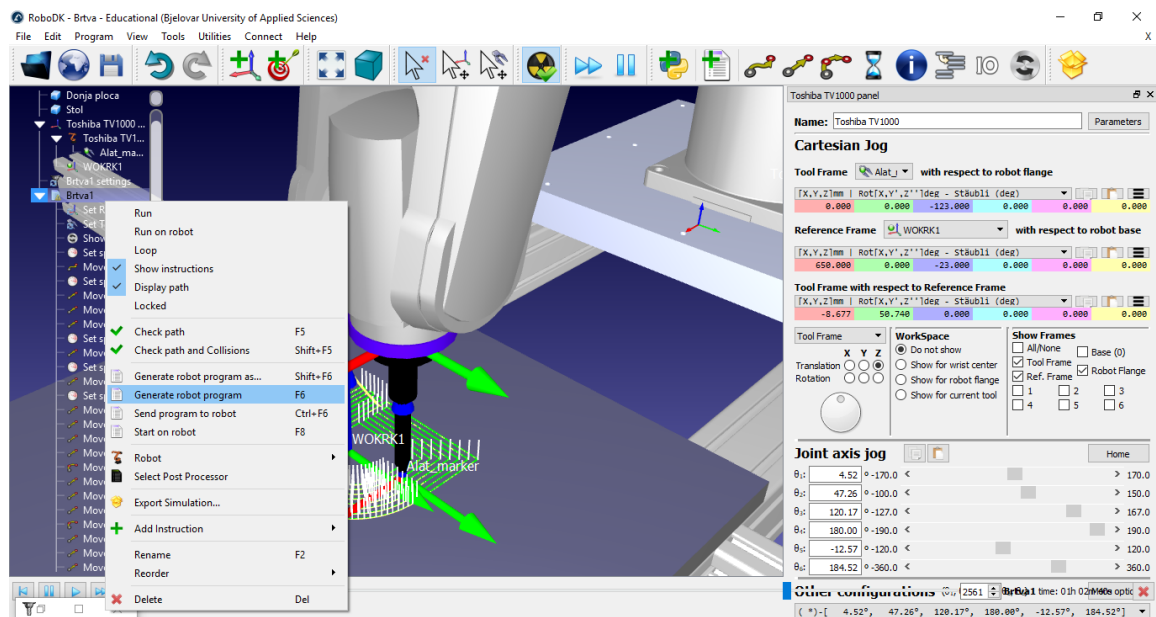
Slika 8.1.5. Okvir za kreiranje projekta praćenja DXF datoteke

Kao referentni koordinatni sustav potrebno je odabrati WORK1 te se može prilagoditi *offset* alata ukoliko je to potrebno. U ovome slučaju nije, sve što je potrebno je postaviti WORK koordinatni sustav i konfiguraciju robota. Pritiskom na tipku Update robot kreira putanju koja se nalazi na WORK koordinatnom sustavu prikazanom na slici 8.1.6.



Slika 8.1.6. Generirana putanja DXF datoteke

Na slici 7.1.6. vidljivo je da se na lijevoj strani projektnoga stabla pojavio program Brtva1. RoboDK omogućuje pregled svih instrukcija programa u simulaciji desnim klikom na program u projektne stablu te zatim Show instructions. Svaku liniju koda moguće je dodatno uređivati prije post procesiranja te raditi dodatne radnje na putanji robota ukoliko je to potrebno. Kada je putanja dovršena pokreće se generiranje upravljačkoga koda robota. Desnim klikom na program u projektne stablu otvara se padajući izbornik na kojem se odabire „Generate robot program“ kao što je vidljivo na slici 8.1.7.



Slika 8.1.7. Pokretanje prevođenja simulacije u upravljački kod robota

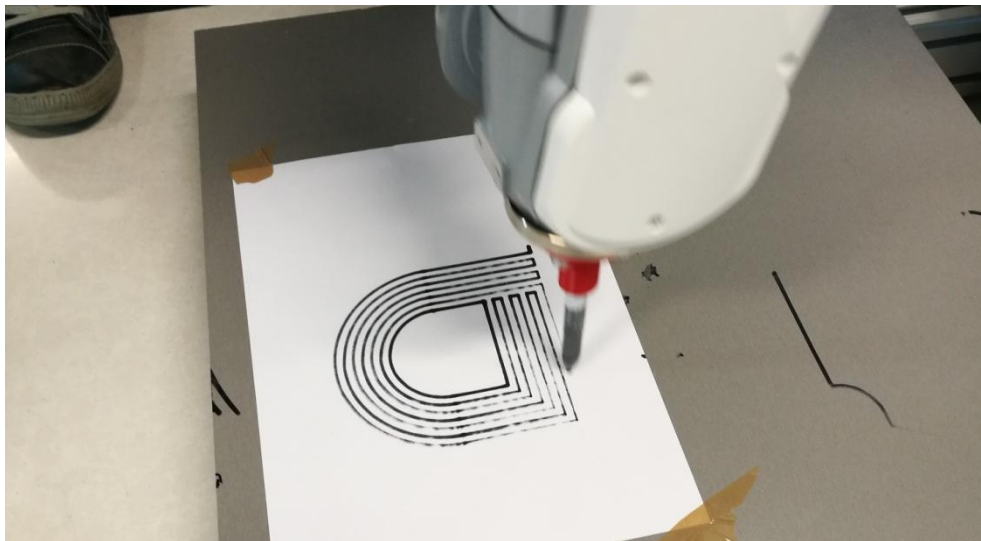
Na temelju simulacije RoboDK pokreće prevođenje simulacije u upravljački kod robota. Nakon što prevođenje simulacije završi upravljački se kod učitava na robot pomoću TSPC programskog alata. Na slici 8.1.8. prikazan je upravljački kod DXF putanje za proizvoljnu putanju koja je nacrtana u svrhu testiranja.

```
PROGRAM Brtva1
REMARK Program generated by RoboDK v3.4.5 for Toshiba TV1000 on 07/04/2019 09:48:13
REMARK Using nominal kinematics.
WOKRK1 = TRANS(650.000,0.000,-23.000,-0.000,0.000,-0.000)
TOOL = TRANS(0.000,0.000,-123.000,-0.000,0.000,-0.000)
REMARK Show Alat_marker

SPEED=100
MOVE POINT(1.323,-24.260,103.000,-0.000,0.000,0.000,0.000,11222) WITH WORK = WOKRK1
SPEED=100
MOVES POINT(1.323,-24.260,103.000,-0.000,0.000,0.000,0.000,11222) WITH WORK = WOKRK1
MOVES POINT(1.323,-24.260,3.000,-0.000,0.000,-0.000,0.000,11222) WITH WORK = WOKRK1
SPEED=100
MOVES POINT(1.323,-24.260,3.000,-0.000,0.000,-0.000,0.000,11222) WITH WORK = WOKRK1
SPEED=100
MOVES POINT(1.323,-24.260,3.000,-0.000,0.000,-0.000,0.000,11222) WITH WORK = WOKRK1
SPEED=0
MOVES POINT(1.323,-24.260,-0.000,-0.000,0.000,-0.000,0.000,11222) WITH WORK = WOKRK1
MOVES POINT(-63.677,-24.260,-0.000,0.000,0.000,0.000,0.000,21222) WITH WORK = WOKRK1
MOVES POINT(-63.677,75.740,-0.000,0.000,0.000,-0.000,0.000,21222) WITH WORK = WOKRK1
MOVES POINT(1.323,75.740,-0.000,0.000,0.000,-0.000,0.000,11222) WITH WORK = WOKRK1
MOVEC POINT(56.888,44.467,-0.000,0.000,0.000,-0.000,0.000,0.000,11222) POINT(58.986,-19.260,0.000,-0.000,-0.000,-0.000,0.000,11222) WITH WORK = WOKRK1
MOVES POINT(66.323,-19.260,0.000,0.000,0.000,-0.000,0.000,11222) WITH WORK = WOKRK1
MOVES POINT(66.323,-24.260,0.000,-0.000,0.000,-0.000,0.000,11222) WITH WORK = WOKRK1
MOVES POINT(56.095,-24.260,0.000,-0.000,-0.000,0.000,0.000,11222) WITH WORK = WOKRK1
MOVEC POINT(54.385,-26.802,0.000,-0.000,-0.000,-0.000,0.000,11222) POINT(52.558,-29.260,0.000,0.000,0.000,-0.000,0.000,11222) WITH WORK = WOKRK1
MOVES POINT(66.323,-29.260,0.000,-0.000,-0.000,-0.000,0.000,11222) WITH WORK = WOKRK1
MOVES POINT(66.323,-34.260,-0.000,-0.000,-0.000,-0.000,0.000,11222) WITH WORK = WOKRK1
MOVES POINT(48.227,-34.260,-0.000,-0.000,-0.000,-0.000,0.000,11222) WITH WORK = WOKRK1
MOVEC POINT(45.612,-36.836,-0.000,-0.000,-0.000,-0.000,0.000,11222) POINT(42.856,-39.260,-0.000,0.000,0.000,-0.000,0.000,11222) WITH WORK = WOKRK1
MOVES POINT(66.323,-39.260,0.000,-0.000,-0.000,-0.000,0.000,11222) WITH WORK = WOKRK1
MOVES POINT(66.323,-44.260,0.000,0.000,-0.000,-0.000,0.000,11222) WITH WORK = WOKRK1
MOVES POINT(35.964,-44.260,-0.000,0.000,0.000,0.000,0.000,11222) WITH WORK = WOKRK1
MOVEC POINT(31.248,-46.962,-0.000,-0.000,-0.000,-0.000,0.000,11222) POINT(26.323,-49.260,-0.000,0.000,0.000,0.000,0.000,11222) WITH WORK = WOKRK1
MOVES POINT(66.323,-49.260,0.000,-0.000,-0.000,-0.000,0.000,11222) WITH WORK = WOKRK1
MOVES POINT(66.323,-54.260,-0.000,-0.000,-0.000,-0.000,0.000,11222) WITH WORK = WOKRK1
MOVES POINT(-63.677,-54.260,-0.000,-0.000,-0.000,0.000,-0.000,21222) WITH WORK = WOKRK1
```

Slika 8.1.8. Dio upravljačkog koda robota generiranog iz DXF datoteke

Kada je upravljački kod generiran on se još jednom provjerava na pogreške u TSPC programskom alatu te se učitava na kontroler robota. Na slici 8.1.9. prikazan je rezultat ispitivanja generiranja putanje iz DXF datoteke.

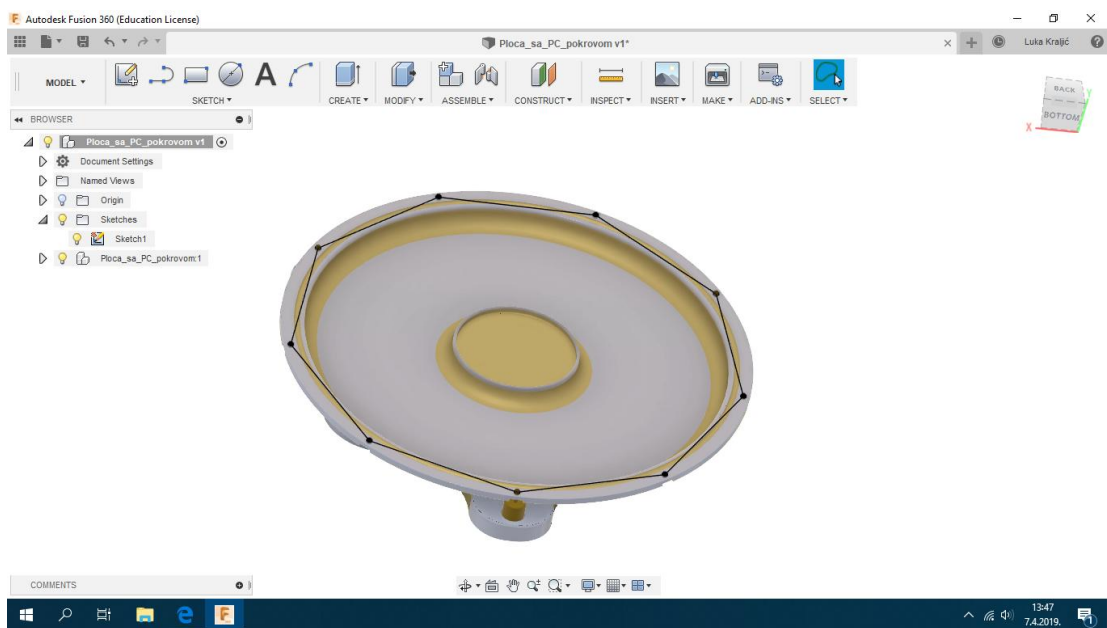


Slika 8.1.9. Rezultat ispitivanja prevođenja DXF datoteke u kod robota

8.2 Bušenje provrta na sjenilima LED reflektora

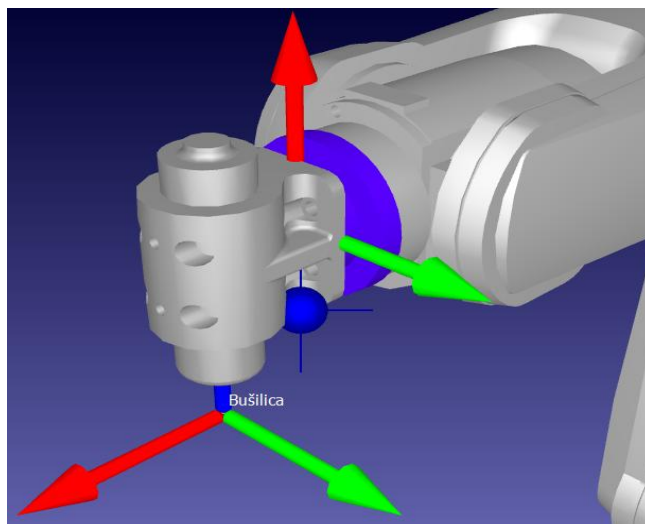
Bušenje provrta korištenjem industrijskog robota vrlo je zastupljena metoda u industriji. Robot kao bušilica pruža niz mogućnosti, pogotovo 6 osni roboti koji mogu pristupiti nekoj točki na 6 različitih načina za razliku od kartezijskih bušilica.

Proces bušenja je testiran na razvijenom post procesoru te je ubacivanje putanje DXF datoteke potpuno isto kao ubacivanje putanje za prethodno prikazan primjer izrade brtve. Na slici 8.2.1. prikazana je putanja za bušenje provrta na sjenilima LED reflektora.



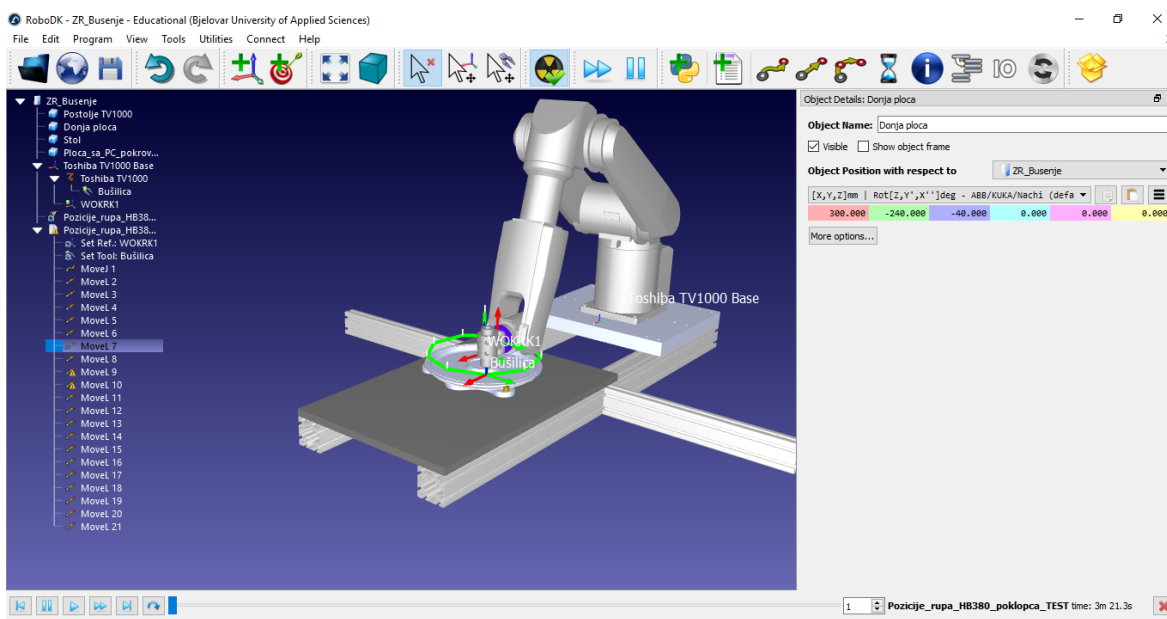
Slika 8.2.1: Putanja za bušenje provrta na sjenilima LED reflektora

Sa slike je vidljivo da je putanja nacrtana točno na 3D modelu na kojem će biti izvršeno bušenje. Kao alat za bušenje koristi se CNC vreteno te je potrebno u RoboDK alatu postaviti 3D model i TCP koordinatni sustav alata. Na slici 8.2.2. nalazi se 3D model bušilice u simulacijskom alatu RoboDK s postavljenim TCP koordinatnim sustavom.

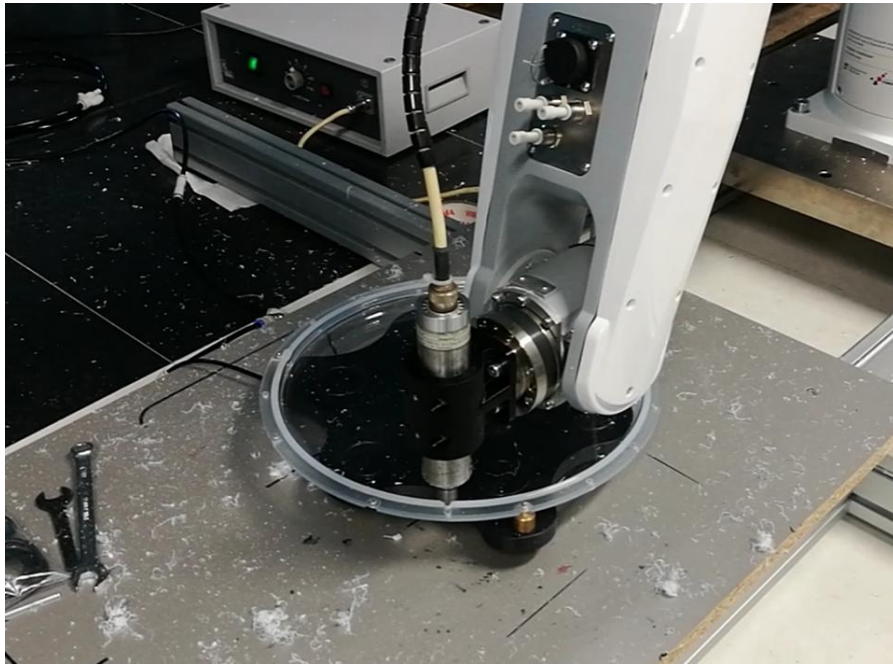


Slika 8.2.2: 3D model CNC vretena za bušenje s TCP koordinatnim sustavom

Nakon postavljanja alata za bušenje ubacuje se putanja te se izrađuje simulacija za bušenje provrta. Na slici 8.2.3. Prikazana je simulacija putanje za bušenje provrta.



Slika 8.2.3: Simulacija bušenja provrta



Slika 8.2.4. Bušenje provrta putanjom iz DXF datoteke

Na slici 8.2.4. prikazan je poklopac LED reflektora u procesu bušenja. Testiranjem post-procesora uočeno je da ispravno generira upravljački kod robta te može prevesti bilo koju putanju iz CAD alata u upravljački kod robota.

Jedina stavka koja bi se dodatno mogla usavršiti je post procesiranje postavki bušenja. Na ovome primjeru u *off-line* programiranju je originalna putanja iz DXF datoteke dodatno prilagođena zbog razloga što nije moguće unijeti parametre bušenja. Na ovom primjeru ručno je metodom *off-line* programiranja u RoboDK alatu nadograđena izvorna putanja iz DXF datoteke.

Ishod ispitivanja sustava je pozitivan i post procesor se može koristiti za prevođenje bilo koje DXF datoteke u upravljački kod robota Toshiba TV-1000 i time je glavni cilj izrade rada ispunjen.

9. ZAKLJUČAK

Tema ovog završnog rada je Upravljanje industrijskim robotom Toshiba TV-1000 definiranjem putanje u DXF datoteci. Projekt je u potpunosti izrađen za tvrtku Data Link d.o.o. i tema je odabrana za završni rad iz kolegija Osnove robotike. Bilo je potrebno pronaći rješenje koje će generirati upravljački kod robota na temelju putanje iz DXF datoteke. Takav sustav znatno bi olakšao i ubrzao sam proces programiranja robota. Projekt je započeo s istraživanjem dostupnih programskih rješenja za realizaciju. Kako to uvijek biva na tržištu postoje rješenja no ona nikada ne mogu u potpunosti pokriti potrebe sve brže rastuće industrije. Sa željom za realizaciju projekta počele su se razvijati i ideje. U prvim fazama projekta bilo je to istraživanje i učenje o problematici samog upravljanja robota na temelju neke zamišljene putanje. Najveći izazov bilo je upoznavanje s osnovama o upravljanju 6-osnim robotima.

Za realizaciju projekta pronađen je programski alat RoboDK koji bi mogao obaviti funkciju prevođenja DXF datoteke u upravljački kod robota no on nije sadržavao potreban model robota i prikladan post procesor u svojoj bazi podataka. Učenjem i istraživanjem same problematike pronađeno je rješenje te je robot ubačen u RoboDK i razvijen post-procesor za prevođenje datoteke u kod robota Toshiba TV-1000. Testiranjem post-procesora na robotu on je usavršen u potpunosti te je time ispunjen i glavni cilj projekta koji je bio postavljen u samome početku.

Testiranjem se pokazalo da je moguće izraditi kompleksne upravljačke kodove za složene putanje robota u samo nekoliko minuta. Ni jedno područje industrijske robotike nije izuzetak te bi ovakav sustav našao primjenu posvuda u industriji. Potpuno se eliminira potreba za poznavanjem upravljačkog koda robota a moguće je na ovaj način upravljati bilo kojim robotom pa čak i onim vlastite izrade za edukacijske svrhe. Neupitna je primjena ove metode programiranja no sustav se uvijek može usavršiti. Moguće je povezati simulacijsko okruženje s kontrolerom robota te koristiti samo jedno simulacijsko okruženje kao jedinstven alat za programiranje svih modela robota. Po mišljenju autora sustav je moguće unaprijediti doradom post procesora da bi bio sposoban generirati upravljačke kodove Vizijski sustav s kamerama. RoboDK ne pruža mogućnost unašanja parametara za operacije bušenja te su i u tom dijelu moguće dorade.

10. LITERATURA

- [1] Šurina T. Crneković M. Industrijski roboti. U: Zagreb: Školska knjiga; 1990.
- [2] Mach 3 informacije o programu [Online] Dostupno na:
<http://www.themakersguide.com/home/products/mach3>
- [3] Toshiba Machines co, .Ltd. Robot Language Manual for Vertical Articulated robot system; 2014.
- [4] Qisheng Liu, Dechen Huang, Xueli Tai, Tao Han, Bingbing Yan : The Research of NC Programming Method Based on DXF File [Online]. Jiamusi University, Jiamusi, Heilongjiang, 154007, China; 2015. dostupno na:
<https://www.atlantis-press.com/proceedings/icmmcce-15/25845075> (prosinac, 2015.)
- [5] Autodesk, Inc. DXF Reference [Online] ; 2011 dostupno na:
https://images.autodesk.com/adsk/files/autocad_2012_pdf_dxf-reference_enu.pdf
(Veljača 2011.)
- [6] Zhaoxin Chen, Jiming Yi, Jun Liu, Yanfeng Lin: Off-Line Programming Design of Robots Based on DXF Files [Online]. Xiamen, China: School of Mechanical and Automotive Engineering, Xiamen University of Technology; 2014. dostupno na:
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1029.5408&rep=rep1&type=pdf>
(31.10.2014.)

11. OZNAKE I KRATICE

ASCII – American Standard Code for Information Interchange

APT – Automatically Programmed Tool

CAD – Computer Aided Design

CAM – Computer Aided Manufacturing

CNC – Computer Numerical Control

CP – Circullar pattern

EPS – Encapsulated

IGES- Initial Grapshic Exchange Specification

Integer- cijeli broj

OBJ – Object

Python – Programski jezik

SSG – Stupnjevi Slobode Gibanja

STP – Tool Center Point

SVG – Scalable Vector Graphics

SCOL – Symbolic Code Language for Robots

String – tekst

TCP – Tool Center Point

2D – dvije dimenzije

3D – tri dimenzije

12. SAŽETAK

Naslov: Upravljanje industrijskim robotom Toshiba TV-1000 definiranjem putanje u DXF datoteci.

Ovaj rad je razvijen u suradnji s tvrtkom Data Link d.o.o. koja je ustupila svu potrebnu opremu autoru ovog rada. Cijeli proces od razvoja do testiranja sustava obavljao se u tvrtki. Glavni cilj bio je ponuditi programsko rješenje za generiranje upravljačkog koda robota na temelju putanje iz DXF datoteke te ga potom i realizirati. Tvrtki je ponuđeno rješenje koje se temelji na kombinaciji simulacijskog okruženja RoboDK i post-procesora razvijenog od strane autora. Ustrajnim radom cijeli je sustav zaživio i uspješno je testiran. Utrošak vremena na izradu upravljačkog koda robota znatno se smanjio što je i bio glavni cilj. Tema približava čitatelju problematiku upravljanja 6-osnih robota klasičnim programiranjem i prikazuje prednosti *off-line* metoda programiranja na temelju pred definirane putanje u CAD alatu. Opisan je način upravljanja 6-osnim robotom Toshiba TV-1000, analiziran je upravljački kod robota te post procesor koji se koristi za prevođenje DXF datoteke u upravljački kod robota. Proces izrade i ispitivanja detaljno je opisan u nadi da će to znanje biti od koristi čitateljima u realizaciji sličnih ideja.

Ključne riječi: Robot, post procesor, Toshiba, DXF, CAD, simulacija, putanja, RoboDK, off-line programiranje.

13. ABSTRACT

Title: Control of industrial robot Toshiba TV-1000 by defining the path in DXF file

This project is developed in coöperation with company Data Link d.o.o. who has provided all the necessary equipment to the author of this work. The whole process of development to system testing has done in the company. The primary goal was to offer a software solution to generate a robot code based on a DXF path, and then to realize that idea. By persistent work, the system has finished and successfully tested. The time necessary to build a robot code much decreased what was the primary goal. The theme approaches the reader with the problem of managing 6-axis robots by classical programming and demonstrates the advantages of off-line programming methods based on a predefined path in CAD tool. This work described the ways of controlling the 6-axis robots, the robot code and the post processor code used to translate the DXF file into the robot code are analyzed in this work. The process of designing and testing is in detail described in the hope that this knowledge will be of use to readers in the realization of similar ideas.

Keywords: Robot, Post processor, Toshiba, DXF, CAD, simulation, path, RoboDK, off-line programming.

14. PRILOZI

U prilogu na CD-u dostavljam video zapise i slike s rezultatima cjelokupnog testiranja obavljenog na robotu i post procesoru. Na CD-u dostavljam i funkcionalnu inačicu rada.

IZJAVA O AUTORSTVU ZAVRŠNOG RADA

Pod punom odgovornošću izjavljujem da sam ovaj rad izradio/la samostalno, poštujući načela akademske čestitosti, pravila struke te pravila i norme standardnog hrvatskog jezika. Rad je moje autorsko djelo i svi su preuzeti citati i parafraze u njemu primjereno označeni.

Mjesto i datum	Ime i prezime studenta/ice	Potpis studenta/ice
U Bjelovaru, <u>14.4.2019.</u>	Luka Kraljić	Luka Kraljić

Prema Odluci Veleučilišta u Bjelovaru, a u skladu sa Zakonom o znanstvenoj djelatnosti i visokom obrazovanju, elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru bit će pohranjene i javno dostupne u internetskoj bazi Nacionalne i sveučilišne knjižnice u Zagrebu. Ukoliko ste suglasni da tekst Vašeg završnog rada u cijelosti bude javno objavljen, molimo Vas da to potvrdite potpisom.

Suglasnost za objavljivanje elektroničke inačice završnog rada u javno dostupnom nacionalnom repozitoriju

Luka Kraljić

ime i prezime studenta/ice

Dajem suglasnost da se radi promicanja otvorenog i slobodnog pristupa znanju i informacijama cjeloviti tekst mojeg završnog rada pohrani u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu i time učini javno dostupnim.

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, 14.4.2019.

Luka Kraljić

potpis studenta/ice