Upravljanje robotom temeljem asinkronog prenošenja programske podrške

Rebić, Filip

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Filip Rebić

Zagreb, 2024. godina.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor: Student:

Izv. prof. dr. sc. Tomislav Stipančić, dipl. ing. Filip Rebić

Zagreb, 2024. godina.

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

Zahvaljujem se mentoru na podršci i pomoći tijekom izrade ovog rada.

Posebna zahvala ide poštovanoj Piji Rebić na motivaciji.

Filip Rebić

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

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 24 – 06 / 1**	
Ur.broj: **15 – 24 –**	

ZAVRŠNI ZADATAK

Student: **Filip Rebić** JMBAG: **0035233293**

Naslov rada na hrvatskom jeziku: **Upravljanje robotom temeljem asinkronog prenošenja programske podrške**

Naslov rada na engleskom jeziku: **Control of a robot based on asynchronous program transmission**

Opis zadatka:

Koncept digitalnog blizanca se temelji na modelu robota unutar virtualnog svijeta koji je integriran s fizičkim robotom u stvarnom okruženju. Robotom se u tome slučaju može upravljati koristeći sinkrona programska rješenja direktno iz virtualne okoline na temelju koncepta digitalnog blizanca. Osim toga, moguće je naknadno, asinkrono prenošenje i izvođenje programa koji su razvijeni u virtualnom svijetu na fizičkom robotu.

Cilj ovog rada je uspostaviti poveznicu između virtualnog modela robota i njegove fizičke implementacije, te razviti i evaluirati programsko rješenje koje omogućuje efikasno upravljanje fizičkim robotom na temelju asinkronog prenošenja. Rad je potrebno temeljiti na RoboDK simulacijskom okruženju za modeliranje i programiranje robota, te Pythonu kao programskom jeziku za razvoj upravljačkog softvera. Rad treba sadržavati kritički osvrt na dobiveno rješenje.

U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
24. 4. 2024.	**2. rok (izvanredni):** 11. 7. 2024. **3. rok:** 19. i 20. 9. 2024.	**2. rok (izvanredni):** 15. 7. 2024. **3. rok:** 23. 9. – 27. 9. 2024.

Zadatak zadao: Predsjednik Povjerenstva:

izv. prof. dr. sc. Tomislav Stipančić prof. dr. sc. Damir Godec

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
x	m	Udaljenost točke P od ishodišta O u smjeru x osi
ρ	m	Euklidska udaljenost između točke P i osi z
φ	Rad	Kut između osi x i vektora usmjerenog od ishodišta prema projekciji točke P na referentnu ravninu
y	m	Udaljenost točke P od ishodišta O u smjeru y osi
z	m	Udaljenost točke P od ishodišta O u smjeru z osi
θ	Rad	Kut između osi z i vektora usmjerenog od ishodišta prema točki P

SAŽETAK

Ovaj rad istražuje primjenu robota UR5 u simulacijskom okruženju RoboDK, s naglaskom na automatizaciju, programiranje i integraciju Python skripti. Rad se bavi osnovama koordinatnih sustava robota, koji su ključni za precizno upravljanje i simulaciju. Također, obrađuje primjenu hvataljki koje se koriste za manipulaciju predmetima. Opisuje se RoboDK, njegov način rada, ključne značajke i prednosti koje donosi u simulaciji robotskih procesa. Posebna pažnja posvećena je asinkronom prijenosu podataka i konceptu digitalnog blizanca, omogućujući naprednu integraciju virtualnih simulacija s fizičkim robotima. Na kraju, naglašava se uloga Pythona u RoboDK-u, koji omogućuje brzu prilagodbu, automatizaciju procesa i razvoj složenih algoritama, čime se značajno unapređuje funkcionalnost simulacija i njihova primjena u industriji. Kao završni dio rada, uspješno je izrađena simulacija pokreta robota UR5 u RoboDK-u, koja je potom testirana na stvarnom robotu, čime su potvrđene točnost i učinkovitost razvijenih algoritama.

Ključne riječi: UR5 robot, RoboDK, Python skripte, asinkroni prijenos podataka, digitalni blizanac, simulacija, integracija

SUMMARY

This thesis explores the application of the UR5 robot within the RoboDK simulation environment, focusing on automation, programming, and the integration of Python scripts. It addresses the fundamentals of robot coordinate systems, which are crucial for precise control and simulation. The paper also examines the use of end-effectors for object manipulation. It provides a comprehensive description of RoboDK, detailing its operational principles, key features, and the benefits it brings to robot process simulation. Special attention is given to asynchronous data transfer and the digital twin concept, facilitating advanced integration between virtual simulations and physical robots. Additionally, the role of Python in RoboDK is highlighted, showcasing its capability for rapid adaptation, process automation, and development of complex algorithms, thereby significantly enhancing the functionality of simulations and their industrial applications. As a concluding part of the work, a simulation of the UR5 robot's movements was successfully created in RoboDK and subsequently tested on the actual robot, confirming the accuracy and effectiveness of the developed algorithms.

Keywords: UR5 robot, RoboDK, Python scripts, asynchronous data transfer, digital twin, simulation, integration

1. UVOD

Programiranje robota igra ključnu ulogu u mnogim industrijama, od automatizacije u proizvodnji do naprednih medicinskih zahvata. Roboti omogućuju izvršavanje repetitivnih, zahtjevnih i često opasnih zadataka s većom učinkovitošću i preciznošću nego što bi to bilo moguće ljudskim radom. U industrijskom sektoru, roboti automatiziraju proizvodne linije, smanjujući troškove i povećavajući kvalitetu proizvoda. U medicini, koriste se za precizne kirurške zahvate i rehabilitaciju, dok u istraživanju omogućuju eksperimente u ekstremnim uvjetima. Roboti također imaju važnu ulogu u vojnoj tehnologiji, zabavi i obrazovanju, gdje služe kao platforma za inovacije i razvoj.

U okviru ovog završnog rada, fokusirat ćemo se na koncept digitalnog blizanca robota i njegovo povezivanje s fizičkom implementacijom robota pomoću asinkronog upravljanja. Koncept digitalnog blizanca omogućuje simulaciju robota u virtualnom okruženju, dok asinkrono upravljanje omogućuje prenos i izvođenje programa između virtualnog modela i fizičkog robota u stvarnom vremenu.

Detaljnije, cilj rada je uspostaviti poveznicu između virtualnog modela robota, kreiranog u RoboDK simulacijskom okruženju, i njegove fizičke verzije. Kroz razvoj i evaluaciju programskog rješenja, rad će se fokusirati na efikasno upravljanje fizičkim robotom temeljenom na asinkronom prenošenju programa razvijenog u virtualnom svijetu. Koristit ćemo Python kao programski jezik za upravljanje robotom, a rad će uključivati kritički osvrt na implementirano rješenje, izazove i potencijalna poboljšanja.

2. POVIJEST

Razvoj programiranja robota složen je proces koji se proteže kroz nekoliko desetljeća, od prvih eksperimenata s robotikom do modernih sustava koji omogućuju sofisticiranu autonomiju i integraciju. Iako su prvotni koncepti programiranja robota bili ograničeni i specifični za to vrijeme, razvoj tehnologije omogućio je značajan napredak u ovom području.

Povijest programiranja robota započinje sredinom 20. stoljeća, s prvim značajnim napretkom koji se dogodio nakon Drugog svjetskog rata. U tom periodu, industrija je počela prepoznavati potrebu za automatizacijom kako bi smanjila proizvodne troškove i poboljšala učinkovitost. Jedan od ključnih koraka bio je razvoj prvog industrijskog robota, Unimate, koji je 1961. godine uveden u proizvodnju automobila tvrtke General Motors, a prikazan je na Slici 1. Unimate je razvio Joseph Engelberger u suradnji s Georgom Devolom, a patentiran je 1954. godine. Ovaj robot je predstavljao početak prve generacije programiranja robota, koristeći primitivne metode za učenje i manipulaciju vrućim komadima lijevanog željeza putem ručnog postavljanja u željene pozicije.

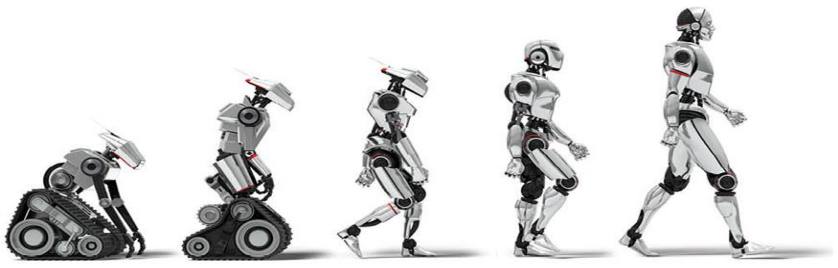
Slika 1. Unimate - prvi industrijski robot [5]

Prvi pravi programski jezik za robote bio je MHI, razvijen 1960. godine, ali je to bio specijaliziran jezik s ograničenim primjenama. Sredinom 1970-ih, razvoj se ubrzao s pojavom općih robotskih jezika. VAL, razvijen 1973. godine na Sveučilištu Stanford, označava početak druge generacije robotskih jezika. Ovi jezici omogućavali su osnovno programiranje robota i temeljili su se na postojećim programskim jezicima poput COBOL-a, ALGOL-a i Fortrana. Iako su bili inovativni za svoje vrijeme, njihova sposobnost interakcije sa senzorima i drugim komponentama robota bila je ograničena.

S razvojem jezika VAL II 1982. godine, druga generacija robotskih jezika donijela je naprednije mogućnosti u kontroli pokreta i sučeljavanju sa senzorima. Ova generacija jezika omogućila je robote da bolje upravljaju složenim zadacima i okruženjima, iako je industrija robotike i dalje bila oprezna u njihovom usvajanju. Ipak, jezici ove generacije postavili su temelje za daljnje inovacije u robotskoj tehnologiji.

Treća generacija robotskih programskih jezika nije se više fokusirala na specifične jezike, već na razvoj koncepata i ideja za napredne robotske aplikacije. Ova generacija uključuje tehnologije poput automatskih 3D modela okoline, dinamičkog razumijevanja okoline, te naprednih metoda učenja, uključujući strojno učenje i poučavanje demonstracijom. Ove inovacije omogućuju robotima da autonomno uče i prilagođavaju se novim situacijama, što predstavlja značajan napredak u razvoju robotskih sustava.

Gledajući prema budućnosti, trendovi u robotskom programiranju ukazuju na sve veću autonomiju robota i integraciju s naprednim tehnologijama. Robotski sustavi postaju sve sposobniji za samostalno učenje i proširivanje svojih funkcionalnosti, što stvara nove izazove i prilike za korisnike i proizvođače. Ova evolucija ne samo da mijenja način na koji se roboti koriste, već i oblikuje budućnost industrije robotike u cjelini, kao i našu maštu o budućnosti robotike što se može vidjeti na Slici 2.



Slika 2. Predviđanje budućnosti robotike [6]

3. KOORDINATNI SUSTAVI U ROBOTICI

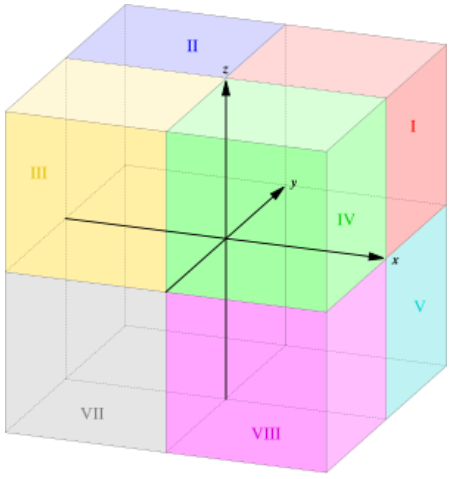
Koordinatni sustavi igraju ključnu ulogu u matematičkom opisivanju i analizi točaka i objekata unutar različitih geometrijskih konfiguracija. Oni omogućuju precizno predstavljanje lokacija i kretanja u prostoru koristeći numeričke vrijednosti, što je od suštinskog značaja za mnoge aplikacije, uključujući robotsku tehnologiju. Razumijevanje i primjena koordinatnih sustava omogućuju robotima da interpretiraju svoje okruženje i izvršavaju zadatke s visokom točnošću.

U trodimenzionalnom prostoru, najčešće korišteni koordinatni sustavi uključuju kartezijev, cilindrični i sferni sustav. Svaki od ovih sustava pruža jedinstvene prednosti i primjene, a izbor pravog sustava ovisi o specifičnim zahtjevima zadatka i prirodi okruženja u kojem se robot koristi. Razumijevanje kako različiti koordinatni sustavi utječu na preciznost i funkcionalnost robota omogućuje inženjerima da optimiziraju performanse i učinkovitost robotskih sustava.

3.1. Kartezijev koordinatni sustav

Kartezijev koordinatni sustav koristi tri međusobno ortogonalne osi – X, Y i Z – koje se sijeku u zajedničkom ishodištu, označenom kao O. Ovaj sustav omogućuje opisivanje točaka u trodimenzionalnom prostoru pomoću trojke realnih brojeva (x, y, z). Ovdje, apscisa (x) označava udaljenost duž X-osi, ordinata (y) udaljenost duž Y-osi, a aplikata (z) udaljenost duž Z-osi.

Kartezijev sustav je posebno važan u robotici jer omogućuje precizno modeliranje i upravljanje pozicijama robota i njegovih dijelova. Ovaj sustav pojednostavljuje proces definicije i kontrole kretanja robota, što je ključno za postizanje visoke preciznosti u automatskim procesima. Na Slici 3. prikazan je primjer kartezijevog koordinatnog sustava.



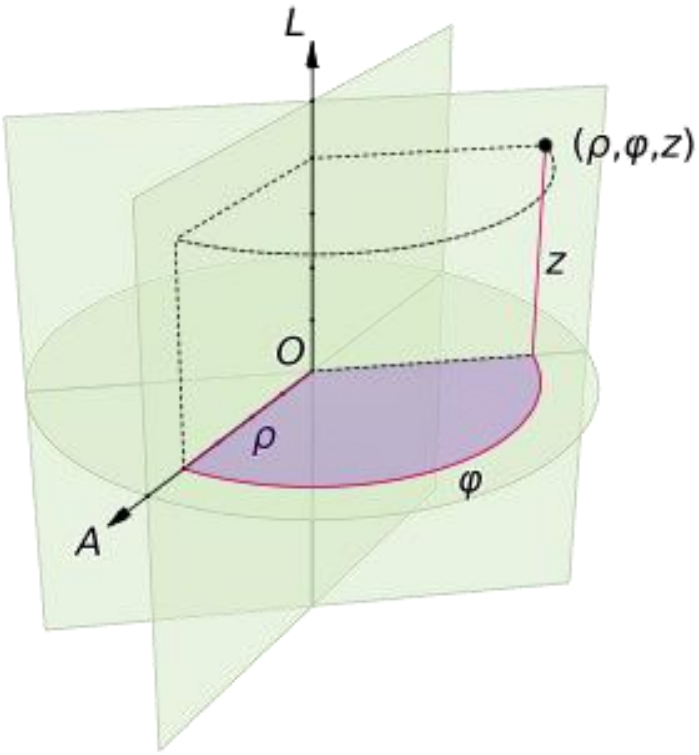
Slika 3. Kartezijev koordinatni sustav [7]

3.2. Cilindrični koordinatni sustav

Cilindrični koordinatni sustav definiran je pomoću ishodišta O, radijusne zrake p i pravca z koji je okomit na zraku p i prolazi kroz ishodište. U ovom sustavu, pozicija točke P određuje se s tri koordinate: ρ, φ i z.

- ρ označava udaljenost od točke P do osi z, mjerenu okomito na pravac z.
- φ je kut između projekcije vektora OP na ravninu okomitu na pravac z i zrake p.
- z predstavlja udaljenost duž osi z od točke P do ishodišta O.

Cilindrični koordinatni sustav je koristan za opisivanje objekata i kretanja u okruženjima koja su prirodno cilindrična, poput rotirajućih dijelova ili valjkastih struktura. Na Slici 4. prikazan je primjer cilindričnog koordinatnog sustava, koji ilustrira kako se ove koordinate koriste za određivanje položaja u trodimenzionalnom prostoru.



Slika 4. Cilindrični koordinatni sustav [7]

Pretvorba iz kartezijevih u cilindrične koordinate izvodi se pomoću sljedećih formula:

$$\rho = \sqrt{x^2 + y^2}$$

$$\boldsymbol{\varphi = arctan\frac{y}{x}}$$

$$\boldsymbol{z = z} \quad .$$

Pretvorba iz cilindričnih u kartezijeve koordinate vrši se pomoću sljedećih formula:

$$\boldsymbol{x = \rho \cdot cos\varphi}$$

$$\boldsymbol{y = \rho \cdot sin\varphi}$$

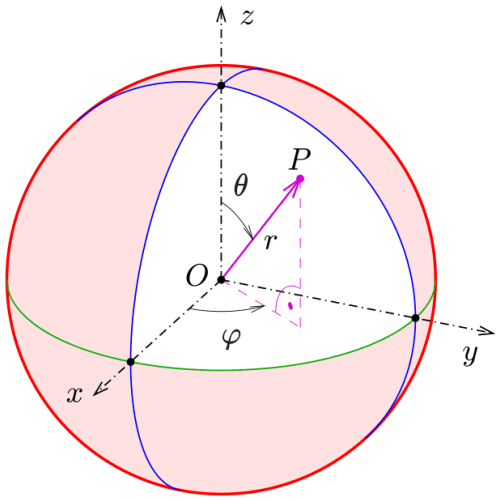
$$\boldsymbol{z = z}$$

3.3. Sferni koordinatni sustav

Sferni koordinatni sustav koristi ishodište O i dvije međusobno okomite polupravce: radijalnu polupravcu p i vertikalnu polupravcu z. Pozicija točke P u ovom sustavu određuje se s tri koordinate: r, φ i θ.

- r predstavlja udaljenost od ishodišta O do točke P.
- φ je azimutni kut između projekcije vektora OP na ravninu okomitu na poluos z i radijalne polupravce p.
- θ označava kut koji vektor OP zatvara s poluos z.

Ovaj sustav je koristan za opisivanje objekata u sfernim ili zakrivljenim strukturama. Na Slici 5. možete vidjeti kako se ove koordinate koriste za određivanje položaja u trodimenzionalnom prostoru.



Slika 5. Sferni koordinatni sustav [7]

Formule za pretvorbu iz kartezijevih u sferne koordinate su sljedeće:

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$\theta = arccos\frac{z}{\sqrt{x^2 + y^2 + z^2}} = arccos\frac{z}{r}$$

$$\varphi = arctan\frac{y}{x}$$

Pretvorba iz sfernih u kartezijeve koordinate vrši se pomoću sljedećih izraza:

$$x = r \cdot sin\theta \cdot cos\rho$$

$$y = r \cdot sin\theta \cdot sin\rho$$

$$z = r \cdot cos\theta$$

3.4. Koordinatni sustavi robota

3.4.1. Globalni koordinatni sustav

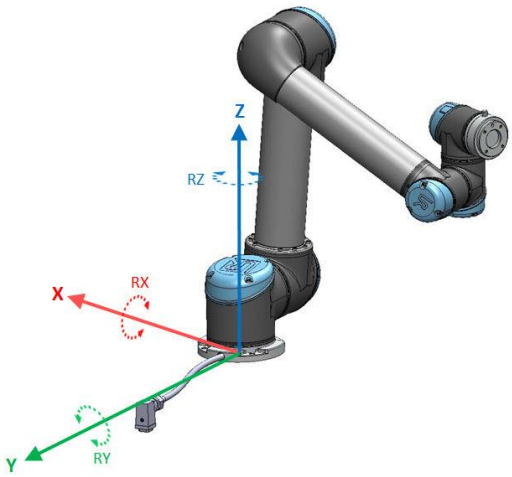
Globalni koordinatni sustav fiksiran je u prostoru i služi kao referentna točka koja može biti zajednička za više uređaja ili robotskih sustava. Ovaj sustav često se podudara s drugim koordinatnim sustavima, poput onih definiranih za pojedine robotske ruke ili alate, omogućujući preciznu koordinaciju i međusobno djelovanje različitih uređaja. U kontekstu robotike, globalni koordinatni sustav omogućuje robotima da dosljedno interpretiraju svoje položaje i kretanja u odnosu na cijelu radnu okolinu, što je ključno za precizno izvršavanje zadataka i međusobnu interakciju više robota.

3.4.2. Lokalni koordinatni sustav

Lokalni koordinatni sustav definira se u odnosu na globalni koordinatni sustav i specifičan je za pojedine dijelove mehanizma robota koji se kreću u odnosu na ostatak sustava. Ovaj koordinatni sustav omogućuje precizno praćenje i kontrolu položaja i orijentacije tih dijelova u okviru šire radne okoline. U robotici, lokalni koordinatni sustavi olakšavaju upravljanje pokretima i interakcijama između različitih dijelova robota, kao i prilagodbu kretanja u odnosu na promjene u globalnom koordinatnom sustavu.

3.4.3. Koordinatni sustav baze

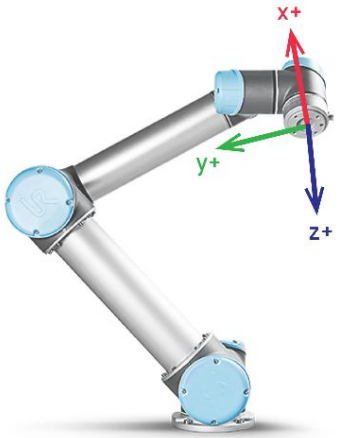
Koordinatni sustav baze robota predstavlja zadani referentni sustav koji određuje položaj robota u odnosu na globalni koordinatni sustav. Njegovo ishodište je fiksirano za bazu robota, što ga čini osnovom za definiranje svih ostalih koordinatnih sustava povezanih s robotom. Ovaj sustav omogućuje precizno praćenje i kontrolu kretanja robota u radnom okruženju. Na Slici 6. prikazan je koordinatni sustav baze robota.



Slika 6. Koordinatni sustav baze robota [8]

3.4.4. Koordinatni sustav prirubnice

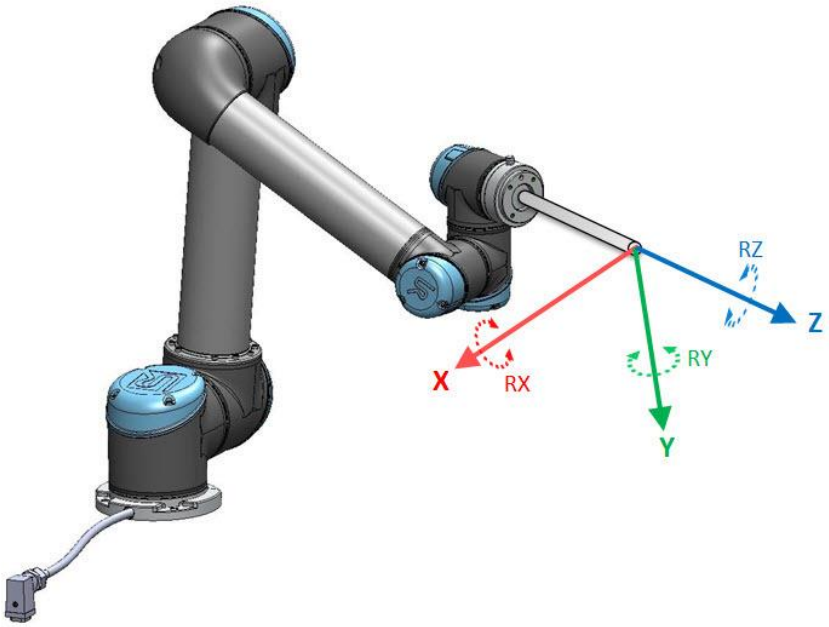
Koordinatni sustav prirubnice smješten je na prirubnici robota, iznad hvataljke, s ishodištem u geometrijskom središtu prirubnice. Ovaj sustav je tvornički postavljen i ne može se mijenjati, a njegova primarna svrha je precizno definiranje pozicije i orijentacije koordinatnog sustava alata. Na sljedećoj slici prikazan je koordinatni sustav prirubnice.



Slika 7. Koordinatni sustav prirubnice robota [8]

3.4.5. Koordinatni sustav alata

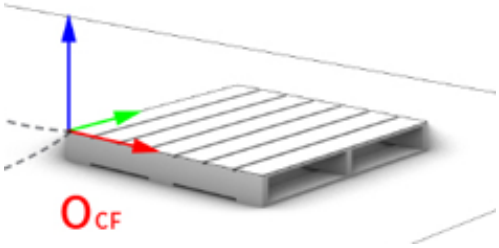
Koordinatni sustav alata definira se u odnosu na koordinatni sustav prirubnice i može se prilagoditi prema specifičnim zahtjevima korisnika. Najčešće se postavlja u TCP točki (središnjoj točki alata). Na Slici 8. prikazan je ovaj koordinatni sustav.



Slika 8. Koordinatni sustav alata [8]

3.4.6. Koordinatni sustav objekta

Koordinatni sustav objekta definiran je na objektima s kojima robot manipulira, omogućujući precizno određivanje njihovog položaja i orijentacije u prostoru. Ovaj sustav je ključan za točno hvatanje i manipulaciju, posebno kada se radi o složenim zadacima poput sklapanja ili preciznog pozicioniranja. Na Slici 9. prikazan je taj koordinatni sustav.



Slika 9. Koordinatni sustav objekta kojim robot manipulira [10]

4. PODJELA ROBOTA

Roboti se klasificiraju prema vrsti pogona, geometriji radnog prostora i metodama upravljanja kretanjem. Ove kategorije omogućuju razlikovanje robota na temelju njihove funkcionalnosti i primjene u različitim okruženjima.

4.1. Vrsta pogona

- Električni motor

Električni motori, uključujući istosmjerni (DC), izmjenični (AC) i koračne motore, najčešće se koriste za pogon robota. Ovi motori su popularni zbog svoje pristupačnosti, visoke brzine i preciznosti. Električni pogon omogućuje implementaciju složenih kontrolnih algoritama, što povećava fleksibilnost i sposobnost robota za izvođenje različitih zadataka s visokom točnošću.

- Hidraulički pogon

Roboti s hidrauličkim pogonom idealni su za manipulaciju velikim teretima zbog svoje iznimne snage i sposobnosti da održavaju stabilan položaj, zahvaljujući nestlačivosti hidrauličkog ulja. Iako su učinkoviti i mogu pružiti zadovoljavajuću brzinu rada, ovi roboti mogu uzrokovati onečišćenje okoliša zbog mogućeg istjecanja ulja, te su često bučni i skupi za održavanje i instalaciju.

- Pneumatski pogon

Pneumatski roboti koriste zrak pod pritiskom za pokretanje, što im omogućuje veliku brzinu rada i nisku cijenu. Ovi roboti su idealni za laboratorijski rad i manipulaciju lomljivim predmetima jer ne zagađuju okoliš i imaju relativno jednostavno održavanje. Međutim, njihova upotreba za rad s velikim teretima je ograničena zbog stlačivosti zraka, što otežava održavanje stabilne pozicije. Također, pneumatski roboti mogu proizvoditi značajnu buku i zahtijevaju dodatne sustave za filtraciju i sušenje zraka.

4.2. Radni prostor – geometrija

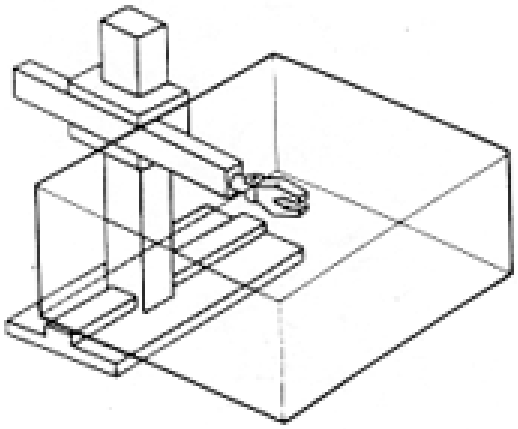
Radni prostor robota čini skup točaka u trodimenzionalnom prostoru do kojih robot može doći pomoću svojih zglobova. Veličina i oblik radnog prostora ovise o broju i vrsti zglobova, duljinama članaka te fizičkim ograničenjima specifičnim za konstrukciju i dizajn robota.

U industrijskoj robotici, postoje dva glavna tipa zglobova: rotacijski, koji omogućuju rotaciju oko osi, i translacijski, koji omogućuju linearno gibanje duž osi. Kombiniranjem rotacijskih i translacijskih zglobova za osnovne osi, mogu se postići različite konfiguracije robota. Svaka konfiguracija omogućuje robota da dosegne specifične radne prostore i izvršava raznolike zadatke s različitim stupnjevima slobode i fleksibilnosti. Te konfiguracije su:

- Pravokutna
- Cilindrična
- Sferna
- Rotacijska
- Robot tipa SCARA

4.2.1. Pravokutna konfiguracija robota

Kod robota s pravokutnom konfiguracijom koriste se tri translacijska zgloba, što rezultira radnim prostorom u obliku pravokutnog prizme. Na Slici 10. i 11. vidimo prikaze robota s pravokutnom konfiguracijom.

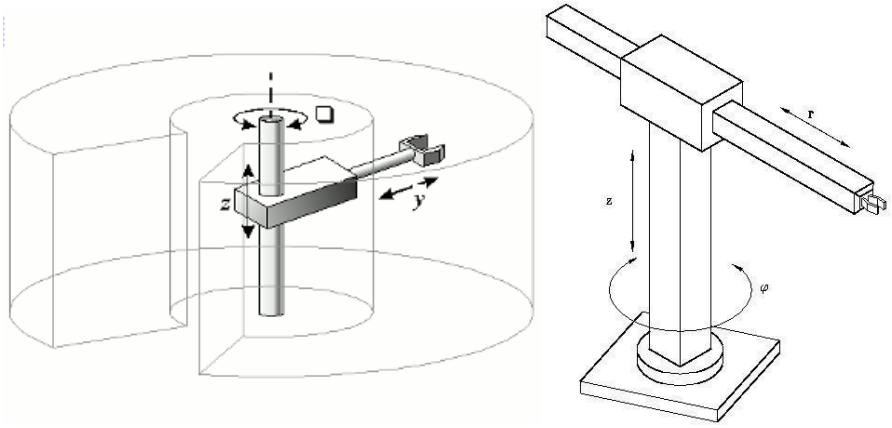


Slika 10. Shematski prikaz pravokutne konfiguracije robota [4]

Slika 11. Industrijski robot s pravokutnom konfiguracijom [4]

4.2.2. Cilindrična konfiguracija robota

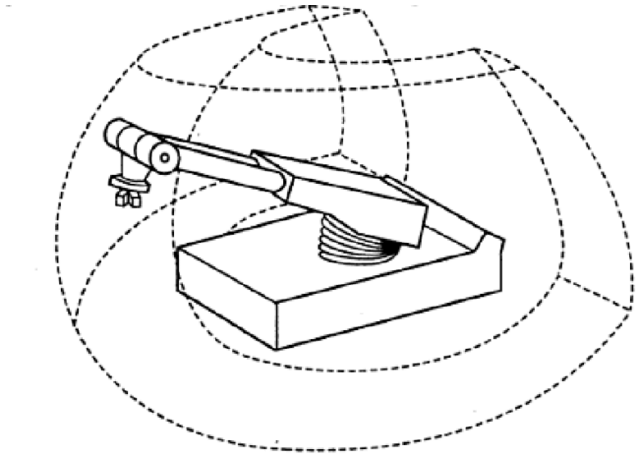
Da bi se postigla cilindrična konfiguracija robota, koristi se jedan rotacijski zglob zajedno s dva translacijska zgloba. U ovoj konfiguraciji, translacijsko gibanje robota je ograničeno, stvarajući radni prostor koji se može opisati kao volumen smješten između dva koncentrična valjka s vertikalnim osi. Ova struktura omogućuje robota da manipulira unutar tog valjkastog volumena. Slika 12. prikazuje robota u cilindričnoj konfiguraciji, koja je pogodna za aplikacije koje zahtijevaju rad u specifičnom radnom prostoru s definiranim ograničenjima gibanja.



Slika 12. Shematski prikaz cilindrične konfiguracije robota [4]

4.2.3. Sferna konfiguracija robota

Robot sa sfernom konfiguracijom koristi dva rotacijska zgloba i jedan translacijski zglob. Ako je translacijsko gibanje ograničeno, radni prostor robota oblikuje se kao volumen između dvije koncentrične sfere. Ako su pak sva gibanja ograničena, radni prostor robota postaje segment volumena između tih dviju koncentričnih sfera. Ova konfiguracija omogućava robotu da pokriva određeni trodimenzionalni prostor s karakteristikama sfernog oblika. Na sljedećim slikama prikazan je robot sa sfernom konfiguracijom, koji demonstrira ove karakteristike u praksi.

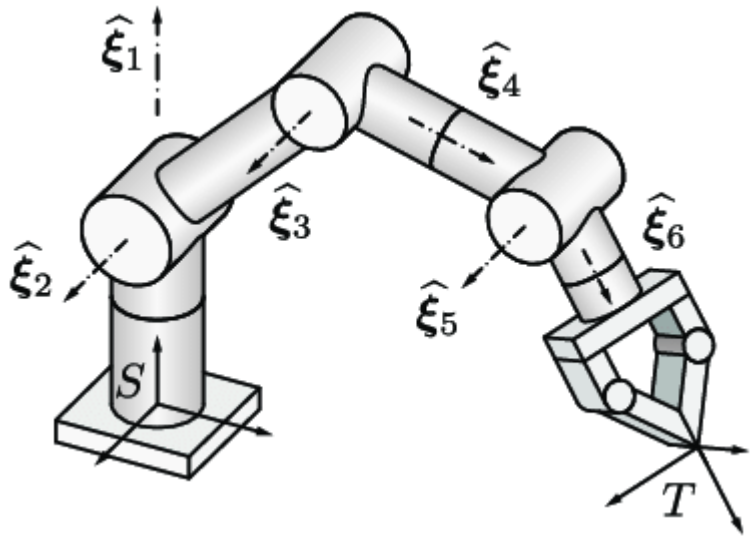


Slika 13. Shematski prikaz sferne konfiguracije robota [4]

Slika 14. Industrijski robot sferne konfiguracije [4]

4.2.4. Rotacijska konfiguracija robota

Robot s ovom konfiguracijom koristi tri rotacijska zgloba. Ova vrsta konfiguracije često se naziva laktasta, antropomorfna ili zglobna. Kada nisu prisutna ograničenja gibanja, radni prostor robota ima oblik kugle. U slučajevima kada postoje ograničenja, radni prostor robota postaje dio kugle složenog oblika. Na slici 15. prikazan je robot s rotacijskom konfiguracijom, koja ilustrira ove karakteristike u praksi.



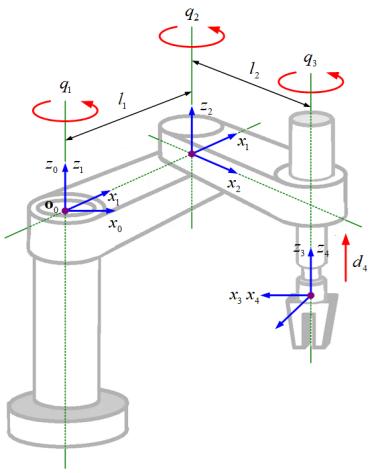
Slika 15. Prikaz robota s rotacijskom konfiguracijom [4]

4.2.5. Robot tipa SCARA

Ovaj tip robota uključuje dva rotacijska zgloba i jedan translacijski zglob. Ovisno o rasporedu tih zglobova, razlikujemo tri glavne konfiguracije: RTR (rotacijski-translacijski-rotacijski), TRR (translacijski-rotacijski-rotacijski) i RRT (rotacijski-rotacijski-translacijski).

SCARA roboti su popularni zbog svoje sposobnosti da obavljaju visok brzi rad u automatizaciji montaže i pakiranja, zahvaljujući svojoj stabilnosti i preciznosti u horizontalnom kretanju. Ovi roboti su često korišteni u industrijama poput elektronike i automobilske proizvodnje, gdje je potrebna visoka preciznost u montaži i manipulaciji.

Na sljedećim slikama prikazane su sve tri konfiguracije SCARA robota, koje ilustriraju različite načine kombiniranja zglobova za postizanje specifičnih radnih prostora i funkcionalnosti.



Slika 16. Shematski prikaz robota tipa SCARA [4]

Slika 17. Robot tipa SCARA (RRT konfiguracija) [4]

4.3. Načini kretanja završnog mehanizma robota

Završni mehanizam robota može se kretati na dva osnovna načina:

- Kretanje točka – točka

  Ova metoda koristi se kada je ključna točnost pozicioniranja robota, dok putanja između točaka nije od velike važnosti. Tipične primjene uključuju točkasto zavarivanje, kao i podizanje i spuštanje predmeta, gdje se robot pomiče direktno između unaprijed definiranih pozicija.

- Kontinuirano gibanje po putanji

  Ovaj način kretanja primjenjuje se kada su i točnost pozicioniranja i trajektorija kretanja robota podjednako važni. Idealno je za zadatke poput bojanja, šavnog zavarivanja i lijepljenja, gdje je potrebno održavati konstantan kontakt s obradkom ili površinom uz precizno praćenje putanje.

5. IZVRŠNI ELEMENTI ROBOTA

Izvršni elementi robota su komponente koje izravno provode zadatke ili operacije zadane u programskom kodu robota. Njihova uloga je ključna u fizičkom izvođenju pokreta, manipulaciji predmetima, ili obavljanju specifičnih zadataka koje robot mora ispuniti. Izbor izvršnih elemenata ovisi o prirodi zadatka koji robot obavlja. Uobičajeni izvršni elementi uključuju hvataljke, magnete, kamere, bušilice, odvijače, alate za zavarivanje, pištolje za bojenje, senzore sile, alate za rezanje, alate za brušenje, i druge specijalizirane alate. Među njima, hvataljke su najčešće korišteni izvršni elementi, a mogu se podijeliti na nekoliko vrsta:

- Krute hvataljke

  Ove hvataljke dizajnirane su za hvatanje i manipulaciju čvrstim i stabilnim predmetima. Njihova konstrukcija od izdržljivih materijala omogućava nošenje težih predmeta poput kutija, paleta ili metalnih dijelova. Krute hvataljke pružaju visoku stabilnost i snagu, što ih čini prikladnim za rukovanje objektima koji zahtijevaju čvrsto držanje.

- Mekane hvataljke

  Mekane hvataljke idealne su za manipulaciju predmetima različitih oblika, veličina i tekstura. Izrađene su od fleksibilnih materijala poput gume, silikona ili elastomera, što im omogućava prilagodbu obliku predmeta i smanjuje rizik od oštećenja krhkih ili osjetljivih objekata. Ova vrsta hvataljki često se koristi u aplikacijama gdje je potrebno nježno rukovanje, poput pakiranja osjetljivih proizvoda.

- Posebne hvataljke

  Posebne hvataljke dizajnirane su za specifične zadatke ili manipulaciju posebnim vrstama predmeta. Ove hvataljke često posjeduju jedinstvene značajke ili funkcionalnosti koje ih razlikuju od standardnih hvataljki. Primjerice, mogu imati prilagođene oblike, specijalizirane hvataljke za određene materijale, ili integrirane senzore za preciznije rukovanje.

Hvataljke se također mogu klasificirati prema načinu rada:

- Vakuumske hvataljke

  Koriste se za podizanje i držanje objekata koristeći vakuumsku silu.

- Pneumatske hvataljke

  Pokreću se pomoću zraka pod tlakom, što omogućava brzo i precizno hvatanje.

- Hidrauličke hvataljke

  Pokreću se pomoću tekućine pod tlakom, što ih čini prikladnima za hvatanje težih predmeta.

- Električne hvataljke

  Pokreću se pomoću električnih motora, nudeći visoku preciznost i kontrolu nad hvatanjem.

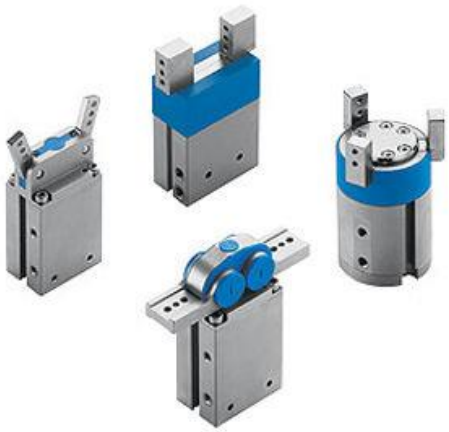
5.1. Vakuumske hvataljke

Vakuumske hvataljke robota su specijalizirani izvršni elementi koji koriste vakuumsku silu za hvatanje i manipulaciju predmetima. Ove hvataljke stvaraju vakuum tako što snižavaju tlak unutar hvataljke u odnosu na vanjski tlak, čime se stvara sila koja čvrsto drži predmet na mjestu. Najčešće se koriste za rukovanje predmetima s ravnim površinama i pravilnim oblicima, poput stakla, keramike, ili elektroničkih komponenti. Zbog svoje sposobnosti da nježno, ali sigurno drže osjetljive i lomljive predmete, vakuumske hvataljke idealne su za aplikacije gdje je potrebna preciznost i delikatnost. Na sljedećoj slici možete vidjeti prikaz vakuumske hvataljke.

Slika 18. Vakuumska hvataljka [11]

5.2. Pneumatske hvataljke

Pneumatske hvataljke robota koriste komprimirani zrak kao izvor energije za pogon prstiju ili čeljusti koje hvataju i manipuliraju predmetima. Ove hvataljke su popularne u industrijskim postrojenjima zbog svoje jednostavnosti, brzine i pouzdanosti, a također se mogu prilagoditi različitim oblicima i veličinama predmeta. Iako su vrlo efikasne, njihov rad zahtijeva stalni pristup izvoru komprimiranog zraka. Zbog visoke brzine i snage, upotreba pneumatskih hvataljki zahtijeva strogo pridržavanje sigurnosnih mjera kako bi se spriječile ozljede i oštećenja predmeta. Na Slici 19. prikazane su pneumatske hvataljke.



Slika 19. Pneumatske hvataljke [11]

5.3. Hidrauličke hvataljke

Hidrauličke hvataljke robota koriste hidraulični sustav za pokretanje prstiju ili čeljusti, omogućujući im hvatanje i manipulaciju predmetima. Ove hvataljke koriste hidraulički tlak kako bi generirale snažnu silu potrebnu za precizno pokretanje i držanje, što ih čini idealnim za rukovanje teškim i masivnim predmetima. Međutim, zahtijevaju pristup hidrauličnom izvoru energije, a sustav također zahtijeva redovito održavanje kako bi se osigurala pouzdanost i dugotrajna učinkovitost. Na sljedećoj slici prikazana je hidraulička hvataljka.

Slika 20. Hidraulička hvataljka [12]

5.4. Električne hvataljke

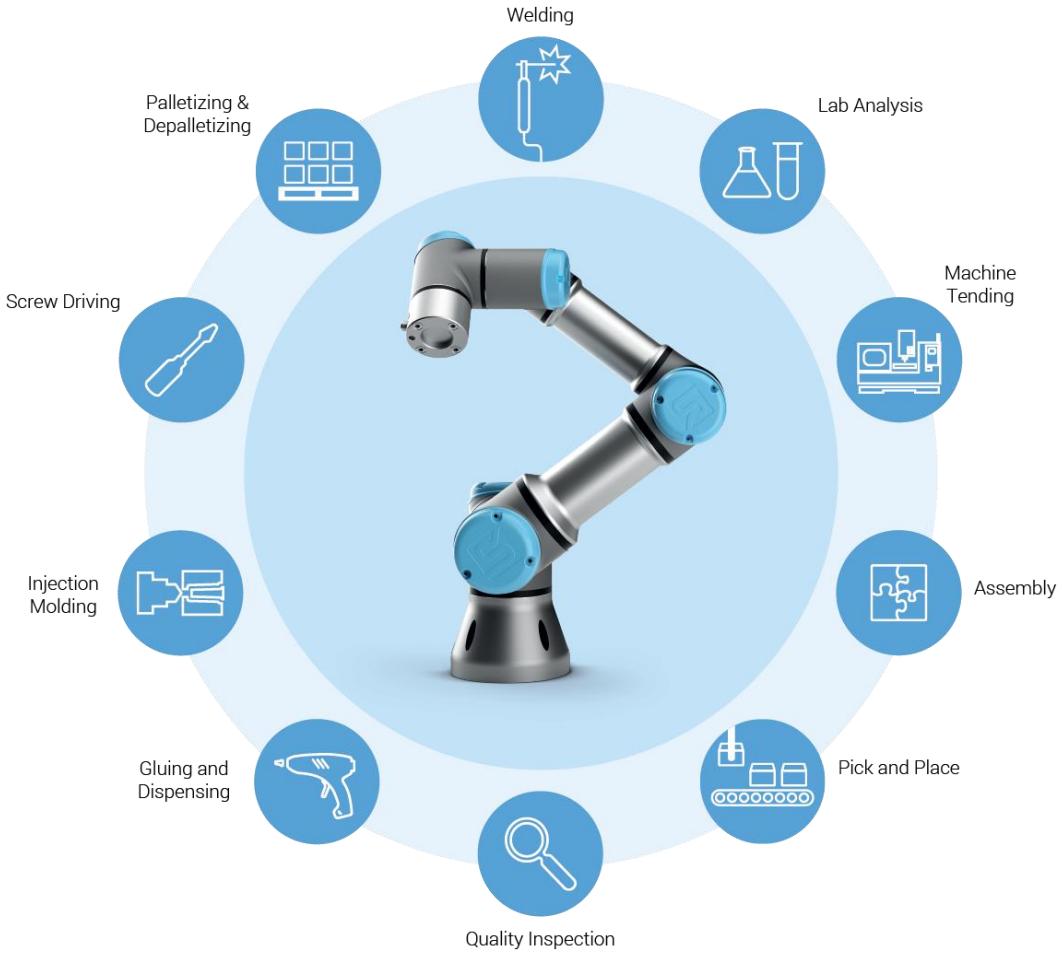
Električne hvataljke robota koriste električnu energiju za pokretanje prstiju ili čeljusti, omogućujući im hvatanje i manipulaciju predmetima. Ove hvataljke oslanjaju se na elektromotore, elektromagnete ili druge električne aktuatore kako bi generirale potrebnu silu za precizno hvatanje i držanje predmeta. Zbog toga pružaju visoku preciznost, kontrolu pokreta i značajnu snagu, a istovremeno su prilagodljive za različite zadatke. Na Slici 21. prikazane su električne hvataljke.

Slika 21. Električne hvataljke [8]

6. ROBOT UR5

Robot UR5 pripada skupini kolaborativnih robota, poznatih i kao coboti (eng. cobot). Za razliku od tradicionalnih industrijskih robota, coboti su robotske ruke male težine, dizajnirane za siguran rad u zajedničkom prostoru s ljudima. Dok su industrijski roboti često izolirani ili zahtijevaju posebne sigurnosne mjere, kolaborativni roboti su namjerno sporiji i manje snažni kako bi osigurali sigurnost ljudi s kojima surađuju. Ova prilagodba nije nedostatak, već ključna karakteristika koja im omogućuje primjenu u novim područjima gdje bi tradicionalni roboti bili previše zahtjevni ili skupi.

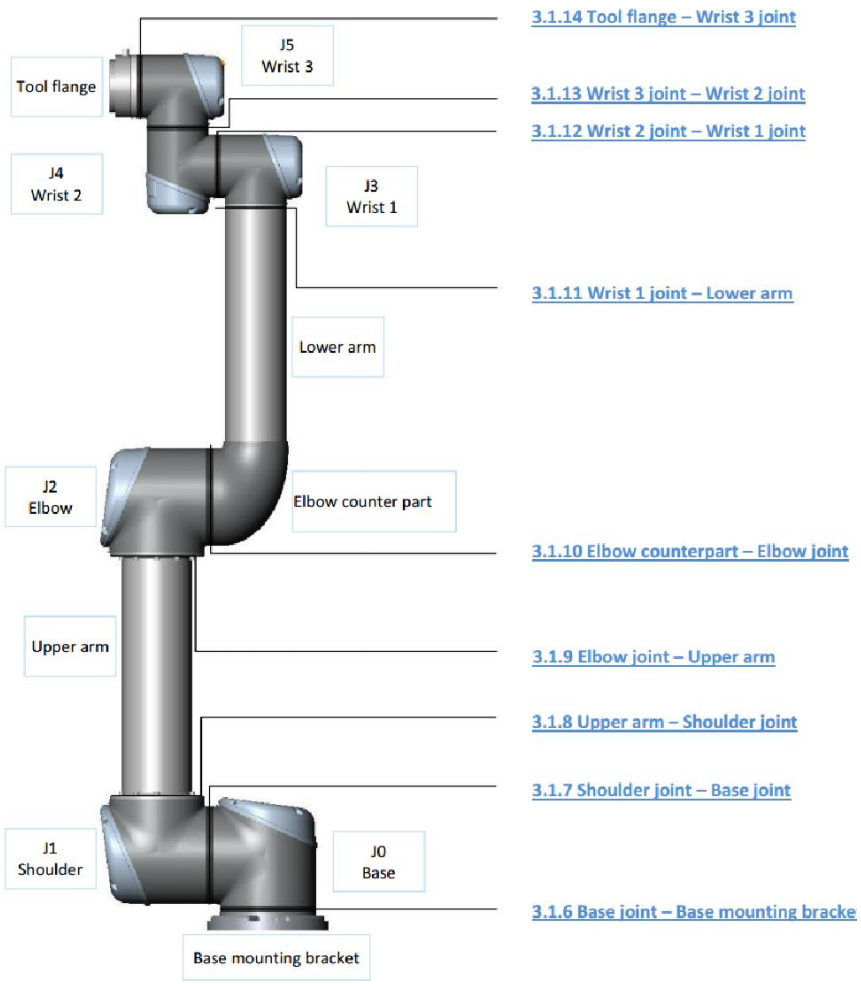
Kolaborativni roboti se često koriste za zadatke poput odabira i umetanja komponenti, sastavljanja, upravljanja CNC strojevima, paletiranja, poliranja, lijepljenja, i drugih aplikacija u bliskoj interakciji s ljudima. Slika 22. prikazuje neke od tih primjena za robota UR5, demonstrirajući njegovu svestranost i sigurnost u radu u neposrednoj blizini ljudi.



Slika 22. Universal robot - različite primjene [8]

Glavne karakteristike kolaborativnih robota uključuju sigurnost, jednostavnost korištenja i pristupačnost. Sigurnost ovih robota postiže se upotrebom laganih materijala, zaobljenim rubovima, ograničenjem brzine i sile, te sofisticiranim senzorima i softverom koji omogućuju sigurno ponašanje robota u blizini ljudi.

Robot UR5 posebno se ističe svojom malom težinom i fleksibilnošću. Izrađen je od aluminijskih cijevi i karika, što ga čini laganim i prikladnim za zadatke koji ne zahtijevaju velika opterećenja. Može podizati terete do 5 kg i ima 6 osi koje mu omogućuju postizanje željenih položaja i orijentacija. Opremljen je senzorom sile koji se može prilagoditi putem grafičkog korisničkog sučelja, što dodatno povećava njegovu prilagodljivost za razne aplikacije. Zbog ovih karakteristika, UR5 je idealan za obavljanje ponavljajućih zadataka u industrijskim okruženjima. Na sljedećoj slici prikazan je sklop robotske ruke s označenim pomičnim zglobovima robota.



Slika 23. Robotska ruka – UR5 manipulator [8]

Sljedeća tablica prikazuje ključne tehničke karakteristike robota UR5 koje su bitne za njegovo razumijevanje i korištenje.

Tablica 1. Tehničke karakteristike robota UR5

Karakteristika	Vrijednost
Težina	18,4 kg
Nosivost	5 kg
Doseg	850 mm
Broj osi (stupnjevi slobode gibanja)	6
Ponovljivost	± 0,1 mm
Potrošnja energije	200 W
Kompatibilnost	Integracija s raznim alatima
Radna temperatura	0 - 50°C

U nastavku je prikazana tablica koja detaljno opisuje dimenzije, brzine i mogućnosti zglobova robota UR5, ključne za simulaciju i programiranje.

Tablica 2. Detalji o zglobovima i mehaničkim mogućnostima robota UR5

Zglob (os)	Raspon kretanja	Max. brzina	Kapacitet momenta
J0 (Baza)	± 360°	180°/s	150 Nm
J1 (Rame)	± 360°	180°/s	150 Nm
J2 (Lakat)	± 360°	180°/s	150 Nm
J3 (Zglob 1)	± 360°	180°/s	28 Nm
J4 (Zglob 2)	± 360°	180°/s	28 Nm
J5 (Zglob 3)	± 360°	180°/s	28 Nm
Senzor sile	Integriran	-	Podesiv
Materijali konstrukcije	Aluminij, plastika PP	-	Lagano i izdržljivo

7. PROGRAMIRANJE ROBOTA

Robot UR5 može se programirati na tri različita načina, svaki sa specifičnim prednostima i primjenama:

- Izravno ili online programiranje

  Robot se programira direktno pomoću ručnog programatora s intuitivnim grafičkim korisničkim sučeljem nazvanim Polyscope, koje je vrlo popularno među korisnicima UR robota. Prednost ove metode je što programer stoji pokraj robota, omogućujući praćenje reakcija robota u realnom vremenu i brzo zaustavljanje u slučaju grešaka ili neočekivanih situacija. Ova metoda je posebno korisna za složenije primjene, kao što je zavarivanje pod teško dostupnim kutovima, jer omogućuje precizno ručno upravljanje robotom do željenih točaka.

- Neizravno ili offline programiranje

  Programiranje robota također je moguće putem robotskog jezika URScript, koji se koristi putem Ethernet TCP/IP veze u različitim programskim okruženjima. Često se koristi kombinacija s RPN-om, gdje se određeni dijelovi programa uređuju na računalu (npr. u Notepadu) i zatim importiraju u RPN. Naredbe URScripta mogu se slati direktno robotu preko mreže, bez potrebe za fizičkom intervencijom. Prednost ove metode je što se program može razvijati i mijenjati dok robot obavlja druge zadatke, omogućujući nesmetan rad proizvodnje i efikasno korištenje vremena.

- Simulacijski programi

  Simulacijski programi postaju sve popularniji jer omogućuju planiranje kompletnih proizvodnih linija i integraciju robota u njih. Ovi programi omogućuju offline programiranje, a neki, poput ArtiMinds RPS i RoboDK, omogućuju istovremeno izvođenje simulacije i stvarnog programa na robotu. ArtiMinds RPS, posebno certificiran za UR robote, nudi dodatne module za korištenje senzora sile i robotskih hvataljki. S druge strane, RoboDK je univerzalniji simulator koji podržava različite robote i omogućuje sofisticirano planiranje virtualnih simulacija, pružajući fleksibilnost u razvoju i testiranju robotskih programa.

8. RoboDK

RoboDK je napredni softverski alat za simulaciju koji omogućuje dizajniranje, programiranje i testiranje industrijskih robota u virtualnom okruženju. Objavljen u siječnju 2015. godine, RoboDK pruža ključne prednosti u modernom industrijskom okruženju, omogućujući korisnicima da unaprijede svoje proizvodne procese bez potrebe za fizičkim testiranjem strojeva.

Ključne prednosti RoboDK-a uključuju:

- Povećanje produktivnosti: RoboDK omogućuje simulaciju i testiranje robotskih aplikacija bez stvarnog pokretanja stroja, što značajno smanjuje vrijeme i troškove proizvodnje. Ovaj pristup eliminira potrebu za zaustavljanjem proizvodnje, čime se poboljšava produktivnost i smanjuje vrijeme potrebna za ispravke.
- Jednostavnost korištenja: Softver se ističe korisnički prijateljskim sučeljem i obuhvaća opsežnu biblioteku robota i alata, što ga čini pristupačnim i za manje iskusne korisnike. Programiranje u RoboDK-u moguće je koristeći različite programske jezike poput Pythona, C# i C++, čime se omogućuje pristup širokom spektru korisnika.
- Simulacija složenih procesa: RoboDK podržava rad s više robota istovremeno, što omogućuje simulaciju kompleksnih industrijskih procesa. Ovaj softver može simulirati aktivnosti kao što su glodanje, crtanje, bojanje, zavarivanje i 3D ispis.
- Sigurnost i optimizacija: RoboDK omogućuje prepoznavanje potencijalnih sudara i drugih problema prije nego što se dogode u stvarnom okruženju. Ovo poboljšava sigurnost stroja i smanjuje rizik od nesreća ili oštećenja opreme. Softver generira detaljne izvještaje za svaki korak simulacije, olakšavajući brzo rješavanje problema i optimizaciju procesa.
- Integracija s CAD/CAM softverima: RoboDK nudi podršku za integraciju s različitim CAD i CAM softverima, što pojednostavljuje proces izrade simulacije. Korisnici mogu koristiti virtualne modele u formatima .step, .iges ili .stl za izgradnju kompletnih radnih okruženja robota.

- Opsežna biblioteka robota: RoboDK uključuje više od osamsto robota različitih proizvođača, što omogućuje stvaranje virtualnih simulacija s raznovrsnim robotskim modelima. U lokalnoj knjižnici programa nalazi se mnogo primjera simulacija koje korisnici mogu preuzeti i proučiti.

Korištenjem RoboDK-a, korisnici mogu unaprijediti svoje robotske aplikacije, smanjiti troškove i vrijeme potrebno za implementaciju, te poboljšati ukupnu učinkovitost proizvodnih procesa. Na sljedećoj slici može se vidjeti logo tvrtke RoboDK te primjer izgleda njegovog sučelja.

Slika 24. RoboDK - logo tvrtke i izgled sučelja [13]

U programu RoboDK, kretanje robota može se programirati korištenjem tri osnovna tipa pokreta, svaki s različitim karakteristikama i primjenama:

- Zglobni pomak (PTP – Point to Point): Ovo je najosnovniji i najčešće korišteni način kretanja. Primjenjuje se kada nema značajnih prepreka u radnom okruženju i kada precizna putanja između točaka nije kritična. Robot se pomiče iz točke A u točku B bez brige o putanji koju prelazi. Ovaj način kretanja označava se s MoveJ u RoboDK.
- Linearni pomak (LIN – Linear): Ovaj način omogućuje robota da se kreće ravno duž određene osi, što je posebno korisno za precizne zadatke kao što su približavanje ili udaljavanje od objekta koji treba uhvatiti hvataljkom. Linearni pomak osigurava točno praćenje pravca, a oznaka u RoboDK za ovu vrstu gibanja je MoveL.
- Kružni pomak (CIRC – Circular): Ova vrsta kretanja koristi se za kreiranje kružnih putanja. Za primjenu kružnog pomaka, potrebno je definirati tri ključne točke: početnu točku, središnju točku i krajnju točku kruga. Ova metoda je manje česta i koristi se za specifične zadatke koji zahtijevaju kretanje u krugu. Oznaka za kružni pomak u RoboDK-u je MoveP.

Svaka od ovih metoda omogućuje različite načine upravljanja robotskim pokretima, ovisno o zahtjevima aplikacije i specifičnostima radnog okruženja.

Postoji aktivna zajednica korisnika RoboDK-a na forumima i društvenim mrežama gdje korisnici mogu postavljati pitanja, dijeliti iskustva i tražiti savjete. Ove zajednice pružaju dodatnu podršku i resurse za rješavanje problema i optimizaciju rada u RoboDK-u.

9. ASINKRONO PRENOŠENJE I PRINCIP DIGITALNOG BLIZANCA U ROBOTSKIM SUSTAVIMA

U današnjoj industriji, efikasno upravljanje robotskim sustavima predstavlja ključ za postizanje optimalnih performansi i produktivnosti. Dva ključna koncepta koja omogućuju napredne metode upravljanja robotima su asinkrono prenošenje i princip digitalnog blizanca. Ovi koncepti, iako kompleksni, igraju ključnu ulogu u integraciji virtualnih simulacija s stvarnim robotskim sustavima, kao što je slučaj u RoboDK-u.

9.1. Asinkrono prenošenje: dinamička sinkronizacija za optimizaciju performansi

Asinkrono prenošenje odnosi se na metodu komunikacije i upravljanja koja omogućuje simultano slanje i primanje podataka između različitih sustava bez potrebe za potpunom sinkronizacijom u realnom vremenu. U kontekstu robotskih sustava, asinkrono prenošenje omogućuje robotima i njihovim upravljačkim sustavima da procesuiraju i odgovore na podatke bez stalne potrebe za usklađivanjem vremena između različitih komponenti.

- Primjer: U laboratoriju, robot može obavljati kompleksne zadatke poput montaže ili zavarivanja dok simultano šalje podatke o svojim statusima i senzorskim očitanjima u stvarnom vremenu. Asinkrono prenošenje omogućuje robotskom sustavu da procesira ove informacije bez kašnjenja, čime se poboljšava reakcija i prilagodljivost robota.

9.2. Princip digitalnog blizanca: virtualna replika za realno praćenje i optimizaciju

Princip digitalnog blizanca (eng. digital twin) uključuje kreiranje virtualne replike stvarnog robota ili sustava, koja se koristi za simulaciju, praćenje i optimizaciju u stvarnom vremenu. Digitalni blizanac je kompleksan model koji replicira fizičke karakteristike i ponašanje stvarnog robota, omogućujući analizu i testiranje različitih scenarija bez potrebe za fizičkim eksperimentiranjem.

- Primjer: U RoboDK-u, digitalni blizanac robota omogućuje korisnicima da kreiraju virtualnu simulaciju robota kako bi testirali različite operacije i strategije u sigurnom virtualnom okruženju. Ova simulacija omogućuje precizno planiranje i prilagodbu operacija prije nego što se implementiraju u stvarnom robotskom sustavu.

9.3. Integracija virtualnih i stvarnih robota

Integracija između virtualnog robota u simulaciji i stvarnog robota u laboratoriju koristi prednosti oba svijeta. Digitalni blizanac omogućuje korisnicima da unaprijed simuliraju i optimiziraju performanse robota, dok asinkrono prenošenje omogućuje dinamičku komunikaciju i prilagodbu u stvarnom vremenu. Ova kombinacija omogućuje:

1. Optimizaciju performansi: Korištenjem simulacija u RoboDK-u, korisnici mogu unaprijed identificirati i riješiti probleme prije nego što utječu na stvarne operacije. Ovo smanjuje vrijeme potrebno za testiranje u fizičkom okruženju i omogućuje brže postizanje optimalnih performansi.
2. Povećanje sigurnosti: Digitalni blizanac omogućuje testiranje i validaciju novih scenarija i operacija u virtualnom okruženju, čime se smanjuje rizik od nesreća i oštećenja opreme u stvarnom laboratoriju.
3. Prilagodba u realnom vremenu: Asinkrono prenošenje omogućuje robotima da se prilagođavaju promjenama u radnom okruženju u stvarnom vremenu, poboljšavajući njihovu učinkovitost i fleksibilnost.
4. Učinkovito praćenje i održavanje: Praćenjem performansi robota putem digitalnog blizanca, korisnici mogu pravovremeno uočiti i riješiti probleme, što vodi ka smanjenju zastoja i povećanju ukupne produktivnosti.

Kombiniranjem asinkronog prenošenja s principom digitalnog blizanca, industrijski roboti mogu biti efikasnije upravljani i optimizirani. Ova integracija ne samo da poboljšava preciznost i brzinu, već također doprinosi većoj sigurnosti i smanjenju troškova u industrijskoj proizvodnji. U konačnici, ove tehnologije omogućuju stvaranje naprednih, prilagodljivih i produktivnih robotskih sustava, što predstavlja značajan korak prema budućnosti automatizacije.

10. MODELIRANJE I SIMULACIJA

10.1. Uvod u modeliranje dijelova

Prije nego što se započne simulacija robota u RoboDK-u, može biti potrebno modelirati specifične dijelove radnog okruženja ili opreme. Ovisno o složenosti aplikacije i dostupnosti modela u RoboDK-ovim bibliotekama, mogu se koristiti vanjski CAD alati za izradu modela. U ovom slučaju nije bilo potrebno specifično modelirati dijelove kako bi se simulirale osnovne kretnje robota u programu. Sve potrebno već je dostupno u RoboDK-ovim bibliotekama, pa možemo preskočiti fazu vanjskog modeliranja i odmah započeti sa simulacijom.

10.2. Postavljanje radnog okruženja

- Odabir radne stanice
- Postavljanje radnog stola

10.3. Postavljanje robota

- Odabir robota
- Pozicioniranje robota

10.4. Kreiranje i konfiguriranje simulacije

- Definiranje kretanja
- Programiranje
- Simulacija i testiranje

10.5. Analiza i optimizacija

Ako je potrebno pregledaju se rezultati simulacije i identificiraju se mogući problemi ili područja za poboljšanje. RoboDK omogućuje vizualizaciju i analizu performansi robota. Na temelju analiza, izvrše se potrebne prilagodbe u programu ili konfiguraciji robota kako bismo poboljšali učinkovitost i preciznost.

10.6. Kratak zaključak

Prethodno modeliranje dijelova može biti ključno za točnost simulacije, dok RoboDK omogućuje jednostavan početak simulacije s dostupnim modelima i radnim stanicama. Korištenjem ovog pristupa, možemo učinkovito planirati, testirati i optimizirati robotske aplikacije u virtualnom okruženju prije nego što se implementiraju u stvarnom svijetu.

11. PYTHON i RoboDK: INTEGRACIJA ZA AUTOMATIZACIJU I RAZVOJ SLOŽENIH ALGORITAMA

Python je jedan od najpopularnijih programskih jezika koji se koristi zbog svoje jednostavnosti, svestranosti i bogate biblioteke. Njegova uloga u RoboDK-u, vodećem softverskom alatu za simulaciju robota, je ključna za unapređenje funkcionalnosti i prilagodljivosti simulacija. Ovaj jezik omogućuje korisnicima da koriste moćne skripte za automatizaciju, prilagodbu simulacija i razvoj složenih algoritama.

11.1. Automatizacija procesa

Jedna od ključnih prednosti korištenja Pythona u RoboDK-u je mogućnost automatizacije robotskih procesa. Python skripte omogućuju korisnicima da napišu programe koji automatski izvode složene zadatke, smanjujući potrebu za manualnim unosom i ponavljajućim radnjama. Automatizacija se može koristiti za:

- Kreiranje i uređivanje simulacija: Python omogućuje korisnicima da programski generiraju i modificiraju simulacijske scenarije. Na primjer, može se automatski postaviti robote, definirati putanje i konfigurirati radne stanice bez potrebe za ručnim radom.
- Upravljanje zadacima: Skripte mogu upravljati rasporedom i izvršenjem različitih robotskih zadataka, kao što su paletizacija, zavarivanje ili bojanje, prema unaprijed definiranim pravilima i uvjetima.

11.2. Prilagodljivost i fleksibilnost

Python doprinosi visokoj prilagodljivosti RoboDK-a, omogućujući korisnicima da prilagode simulacije specifičnim potrebama i zahtjevima. Ova prilagodljivost uključuje:

- Interakcija s 'RoboDK API': Python omogućuje pristup RoboDK-ovom API-ju (eng. Application Programming Interface), što korisnicima omogućuje da integriraju RoboDK s drugim sustavima i aplikacijama. To uključuje prilagodbu simulacija za specifične robote, alate i radne stanice.
- Razvoj složene logike: Python skripte mogu implementirati složene logičke uvjete i algoritme koji mogu biti previše kompliciran za postizanje putem standardnih grafičkih sučelja. Ovo omogućuje kreiranje naprednih rješenja koja mogu optimizirati radne procese i učinkovitost robota.

11.3. Prilagodljivost i fleksibilnost

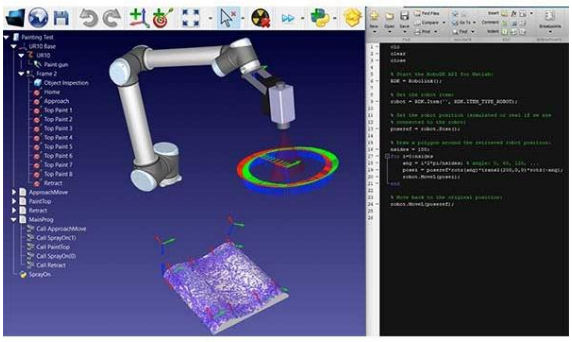
Korištenje Pythona u RoboDK-u također doprinosi bržem razvoju i implementaciji robotskih aplikacija:

- Brza iteracija i testiranje: Python omogućuje brzu promjenu i testiranje različitih skripti i scenarija. Ovo omogućuje korisnicima da brzo provjere različite konfiguracije i optimiziraju simulacije bez dugog čekanja na promjene u stvarnom vremenu.
- Poboljšanje razvoja algoritama: Razvoj složenih algoritama postaje učinkovitiji jer Python pruža bogat skup biblioteka i alata za rad s podacima, matematičkim izračunima i optimizacijom. Ovo omogućuje korisnicima da implementiraju napredne strategije za upravljanje i optimizaciju robotskih zadataka.

11.4. Prilagodljivost i fleksibilnost

Python je izuzetno koristan za razvoj složenih algoritama koji mogu značajno poboljšati performanse robota u simulacijama:

- Algoritmi za planiranje putanje: Korištenjem Pythona, mogu se razviti i implementirati algoritme za napredno planiranje putanje, omogućujući robotima da efikasnije i preciznije kreiraju putanje za izvršenje zadataka.
- Optimizacija performansi: Python skripte mogu se koristiti za analizu performansi robota i radnih procesa te za implementaciju optimizacijskih algoritama koji poboljšavaju brzinu i učinkovitost robotskih operacija.



Slika 25. Primjer korištenja Python-a u RoboDK

Ukratko, Python skripte su ključne za poboljšanje funkcionalnosti i prilagodljivosti simulacije u RoboDK-u, omogućujući korisnicima da učinkovito automatiziraju procese, prilagode simulaciju prema svojim potrebama i razvijaju složene algoritme. Ovaj jezik pruža fleksibilnost i moćne alate koji omogućuju brži razvoj, optimizaciju performansi i unapređenje robotskih aplikacija. Integracija Pythona s RoboDK-om omogućuje stvaranje sofisticiranih i prilagođenih robotskih rješenja koja unapređuju učinkovitost i produktivnost u industrijskim okruženjima.

12. PYTHON SKRIPTA

U ovom zadatku bilo je potrebno osmisliti i evaluirati rješenje za efikasno kretanje robota UR5 koristeći Python skriptu. Skripta omogućava povezivanje robota s Python okruženjem te zadaje niz naredbi koje upravljaju pokretima robota prema unaprijed definiranim točkama u prostoru. Na taj način osigurana je precizna i automatizirana simulacija različitih smjerova kretanja robota, što je ključno za uspješno izvršenje zadatka. Na Slici 26. može se vidjeti spomenuta skripta.

Program.py 2

C: > Users > T14s > Desktop > zavrsni > Program.py > ...

```python

# You can also use the new version of the API:
from robodk import robolink    # RoboDK API
from robodk import robomath    # Robot toolbox
RDK = robolink.Robolink()

# poveznica s robotom UR5
robot = RDK.Item('UR5')

# targeti
target_home = RDK.Item('Home')
target_1 = RDK.Item('Pozicija 1')
target_2 = RDK.Item('Pozicija 2')
target_3 = RDK.Item('Pozicija 3')
target_4 = RDK.Item('Pozicija 4')
target_5 = RDK.Item('Pozicija 5')
target_6 = RDK.Item('Pozicija 6')
target_7 = RDK.Item('Pozicija 7')
target_8 = RDK.Item('Pozicija 8')

# glavni program
robot.MoveJ(target_home)
robot.MoveJ(target_1)
robot.MoveJ(target_2)
robot.MoveJ(target_3)
robot.MoveJ(target_4)
robot.MoveJ(target_5)
robot.MoveJ(target_6)
robot.MoveJ(target_home)
robot.MoveJ(target_7)
robot.MoveJ(target_8)
robot.MoveJ(target_home)

print('Robot je zavrsio s kretanjem i vratio se u home poziciju')

```

Slika 26. Python skripta

```
from robodk import robolink     # RoboDK API
from robodk import robomath     # Robot toolbox
RDK = robolink.Robolink()

# poveznica s robotom UR5
robot = RDK.Item('UR5')
```

Slika 27. Python program - inicijalni dio

U početnom dijelu koda uvoze se potrebni moduli 'robolink' i 'robomath' iz RoboDK-a. 'robolink' se koristi za povezivanje s RoboDK-om i kontrolu robota, dok 'robomath' pruža matematičke funkcije za rad s robotima. Kreira se objekt 'RDK' koji predstavlja instancu veze s RoboDK aplikacijom. Ovaj objekt omogućava komunikaciju između Pythona i RoboDK softvera. Skripta pronalazi i povezuje se s robotom UR5 unutar RoboDK-a. 'robot' postaje referenca na ovaj robot, omogućujući daljnje manipulacije i upravljanje njegovim pokretima.

```
# targeti
target_home = RDK.Item('Home')
target_1 = RDK.Item('Pozicija 1')
target_2 = RDK.Item('Pozicija 2')
target_3 = RDK.Item('Pozicija 3')
target_4 = RDK.Item('Pozicija 4')
target_5 = RDK.Item('Pozicija 5')
target_6 = RDK.Item('Pozicija 6')
target_7 = RDK.Item('Pozicija 7')
target_8 = RDK.Item('Pozicija 8')
```

Slika 28. Python program – targeti

Ovdje se definiraju ciljne pozicije koje je robot UR5 prethodno naučio. Skripta koristi 'RDK.Item' za dohvaćanje referenci na ove pozicije u RoboDK-u, koje su nazvane 'Home', 'Pozicija 1', itd.

```
21  # glavni program
22  robot.MoveJ(target_home)
23  robot.MoveJ(target_1)
24  robot.MoveJ(target_2)
25  robot.MoveJ(target_3)
26  robot.MoveJ(target_4)
27  robot.MoveJ(target_5)
28  robot.MoveJ(target_6)
29  robot.MoveJ(target_home)
30  robot.MoveJ(target_7)
31  robot.MoveJ(target_8)
32  robot.MoveJ(target_home)
```

Slika 29. Python program - glavni program

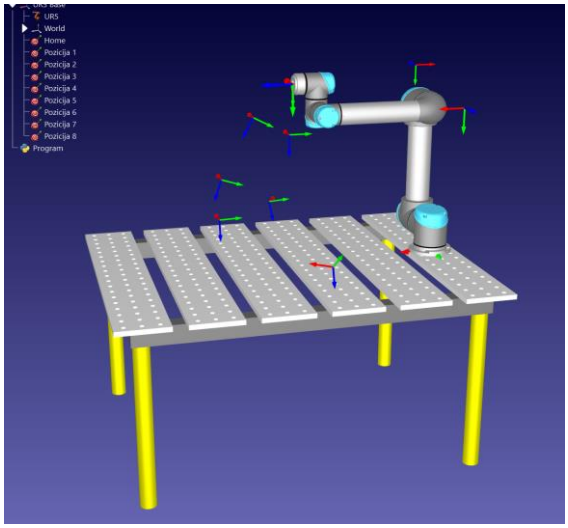
Ovo je niz naredbi koje zadaju robotu UR5 da se kreće između definiranim pozicijama. Funkcija 'MoveJ()' pokreće zglobno kretanje (eng. joint movement) robota, što znači da će robot od trenutne pozicije prijeći u definiranu ciljnu poziciju putem najmanjeg mogućeg puta. Robot se prvo pomiče na 'target_home', zatim redom prolazi kroz 'Pozicija 1' do 'Pozicija 6', vraća se u 'target_home', nastavlja na 'Pozicija 7' i 'Pozicija 8', i na kraju se ponovno vraća u početnu, 'target_home' poziciju.

Nakon što robot završi s izvođenjem svih pokreta, skripta ispisuje poruku u konzoli koja obavještava korisnika da je robot završio sve pokrete i da se vratio u početnu poziciju. Zadnja linija služi tome da se provjeri hoće li se sve naredbe uredno izvršiti kako je zadano.

Ova skripta jednostavno i učinkovito prikazuje kako kontrolirati robota UR5 kroz Python koristeći već naučene pozicije u RoboDK-u.

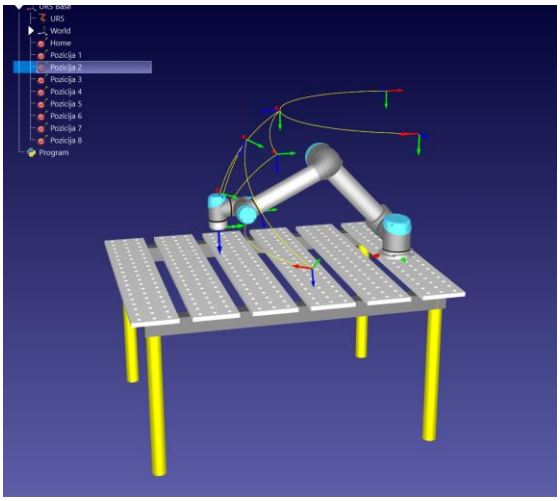
13. SIMULACIJA U RoboDK

Za pokretanje simulacije u RoboDK-u, prvi korak je učitavanje robota UR5 iz programske knjižnice. Zatim se iz dostupne ponude odabire odgovarajući radni stol na kojem će robot biti smješten. Tijekom ovog procesa, posebna pažnja posvećuje se pravilnom usklađivanju koordinatnih sustava između robota i radnog stola kako bi se osigurala preciznost u simulaciji. Na sljedećoj slici prikazani su robot i radni stol.



Slika 30. Robot UR5 i radni stol

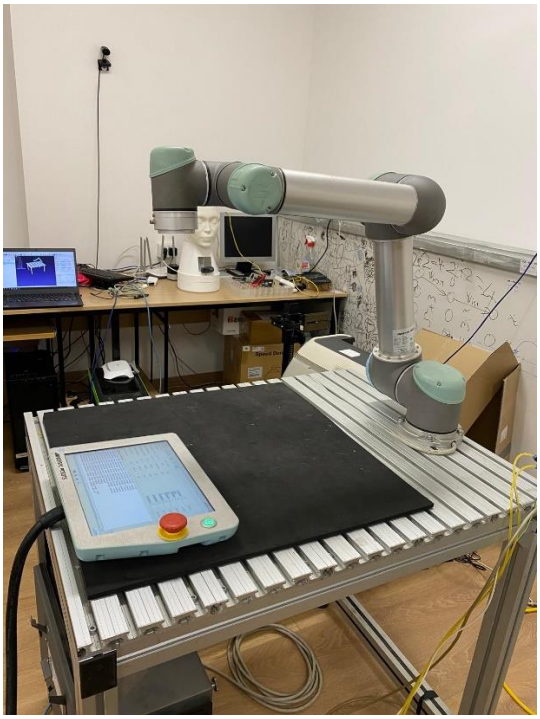
Potom sam ručno postavio sve potrebne pozicije u programu, koje su prethodno objašnjene zajedno s Python skriptom. Nakon toga, izradio sam skriptu koja je detaljno opisana iznad i pokrenuo simulaciju. Na Slici 31. prikazan je robot u jednoj od zadanih pozicija, što potvrđuje uspješno izvršenje skripte i pokretanje robota.



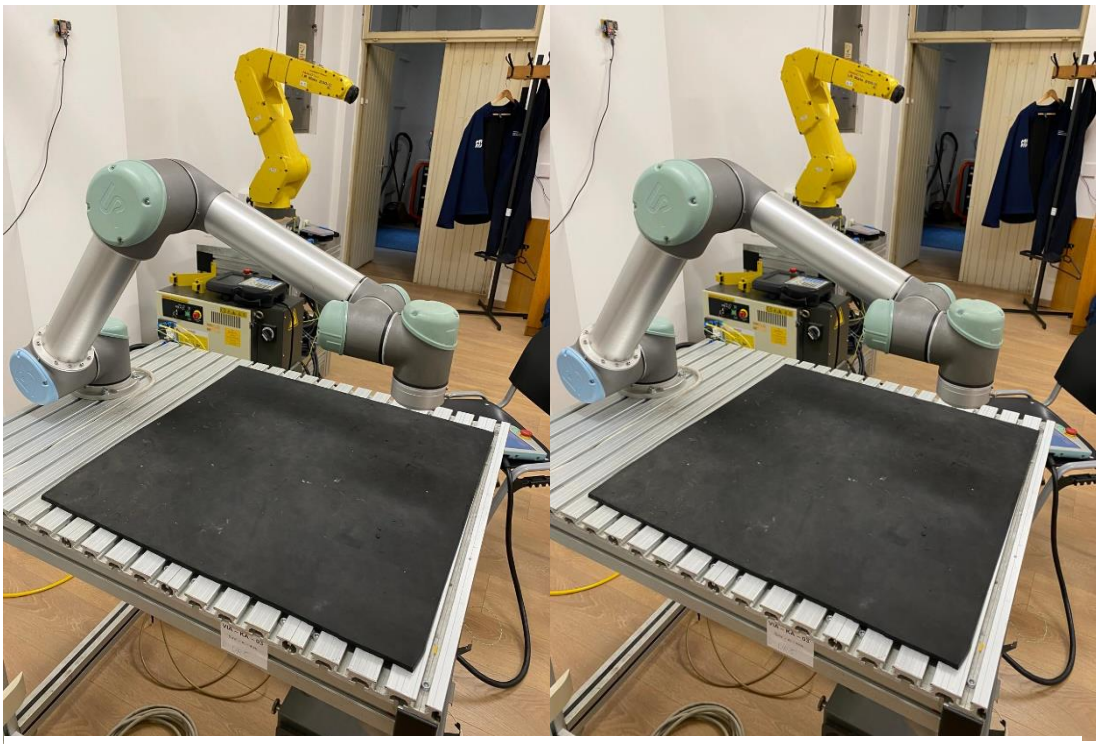
Slika 31. Robot u poziciji 2 (prikazana putanja)

14. POVEZIVANJE S ROBOTOM

Nakon uspješno pokrenute i testirane simulacije u RoboDk, potrebno je učitati simulaciju na stvarnog robota. Naredne slike prikazuju početni položaj te poziciju 2 i 4 u stvarnosti.



Slika 32. Početni položaj robota



Slika 33. Pozicija 2 i 4 u stvarnosti

15. KRITIČKI OSVRT

Prvi izazov s kojim sam se suočio bio je osigurati da se virtualno okruženje precizno podudara sa stvarnim uvjetima rada robota UR5. To je zahtijevalo pažljivu kalibraciju koordinatnih sustava, postavljanje robota i svih potrebnih elemenata na radni stol, te definiranje pozicija koje će robot zauzimati tijekom simulacije. Iako je RoboDK moćan alat, prijenos virtualnih parametara u stvarni svijet uvijek nosi sa sobom potencijalne probleme, kao što su nesavršenosti u modeliranju ili razlike u ponašanju robota u simuliranom i stvarnom okruženju.

Osmišljavanje i implementacija Python skripte za kontrolu robota također nije proteklo bez izazova. Iako je skripta omogućila automatizaciju kretanja robota kroz definirane pozicije, trebalo je osigurati da su svi ciljevi (targeti) pravilno postavljeni i da robot može glatko prelaziti s jedne pozicije na drugu. Svaka greška u definiranju pozicija ili kodiranju skripte mogla je dovesti do nepredviđenih poteškoća, poput kolizije robota s objektima, singularnosti ili nepravilnog kretanja, što bi zahtijevalo dodatno prilagođavanje i ispravljanje.

Potencijalni problemi poput neusklađenosti koordinatnih sustava, pogrešnog programiranja sekvence pokreta ili nepreciznosti u postavljanju početnih pozicija mogli su predstavljati ozbiljne prepreke. No, ovi izazovi uspješno su savladani kroz temeljito testiranje i iterativni pristup razvoju simulacije. Svaka faza simulacije bila je pažljivo praćena i prilagođavana kako bi se osigurala što veća točnost i pouzdanost rezultata.

Sve u svemu, iako nije bilo bez poteškoća, rad na ovoj simulaciji omogućio je duboko razumijevanje upravljanja robotom putem Python skripti i važnosti preciznog planiranja i postavljanja virtualne robotske stanice. Kroz kritički pristup i stalno poboljšavanje, postignuti su ciljevi simulacije, što je pokazalo važnost metodičkog pristupa u rješavanju složenih zadataka u robotici.

16. ZAKLJUČAK

Glavni cilj bio je istražiti mogućnosti i prednosti koje pruža integracija robotskih sustava unutar virtualnog okruženja RoboDK koristeći pritom Python, kao i primjena koncepta digitalnog blizanca za napredno upravljanje i simulaciju robotskih procesa.

Kroz rad smo se uvjerili u važnost pažljivog planiranja i postavljanja svih elemenata u simulacijskom okruženju. Postavljanje koordinatnih sustava, definiranje ciljeva (targeta) te programiranje kretanja robota zahtijevalo je metodički pristup i preciznost kako bi se osigurala točnost simulacije i njezina uspješna primjena u stvarnom okruženju. Python skripte pokazale su se kao ključan alat za automatizaciju procesa, omogućujući fleksibilnost i brzinu u razvoju robotskih aplikacija.

Unatoč izazovima, poput potencijalnih neusklađenosti i poteškoća u prijenosu simulacijskih parametara u stvarni svijet, uspjeli smo razviti i testirati funkcionalno rješenje koje je pokazalo visoku razinu preciznosti i učinkovitosti. Simulacija kretanja robota UR5 uspješno je izvedena i potvrđena na stvarnom robotu, što potvrđuje vrijednost korištenih metoda i alata.

Ovaj rad naglašava važnost simulacije u suvremenoj robotici, posebno u kontekstu optimizacije proizvodnih procesa i smanjenja rizika. Integracija robota s naprednim softverskim alatima poput RoboDK-a i programskim jezicima poput Pythona predstavlja značajan korak prema učinkovitijem i prilagodljivijem upravljanju robotskim sustavima u industriji. Rezultati ovog rada pružaju temelj za daljnje istraživanje i primjenu sličnih pristupa u složenijim i zahtjevnijim robotskim aplikacijama.

LITERATURA

[1] Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). *Robotics: Modelling, Planning and Control*. Springer London, https://doi.org/10.1007/978-1-84628-642-1, pristupljeno: 28.8.2024.

[2] Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*. MIT press, https://doi.org/10.1017/s0263574705221628, pristupljeno: 29.8.2024.

[3] Mataric, M. J. (2007). *The robotics primer*. MIT press.

[4] Z. Kovačić, S. Bogdan, V. Krajči, Osnove robotike, Zagreb, 2002.

[5] Unimate – Robots, https://robotsguide.com/robots/unimate, pristupljeno: 28.8.2024.

[6] Robohub, https://robohub.org/envisioning-the-future-of-robotics/, pristupljeno: 28.8.2024.

[7] Koordinatni sustav – Wikipedija, https://www.wikiwand.com/hr/articles/Koordinatni_sustav, pristupljeno; 28.8.2024.

[8] Universal Robots, https://forum.universal-robots.com/t/base-coordinate-system-documentation/140, pristupljeno: 28.8.2024.

[9] Why use Offline Robot Programming Software and how to get started – Robotmaster, http://www.robotmaster.com/en/newsroom/offline-robot-programming-software, pristupljeno: 29.8.2024.

[10] COMPAS FAB, https://gramaziokohler.github.io/compas_fab/latest/examples/01_fundamentals/02_coordinate_frames.html, pristupljeno: 29.8.2024.

[11] MSITEC, https://msitec.com/product/vacuum-gripper-kit-for-universal-robots/, pristupljeno: 30.8.2024.

[12] Mobile Hydraulic Tips, https://www.mobilehydraulictips.com/hydraulic-grippers/, pristupljeno: 30.8.2024.

[13] UR5 – RoboDK, https://robodk.com/robot/Universal-Robots/UR5, pristupljeno: 30.8.2024.

[14] Robotika - Kobot vs Robot – Mreža, https://mreza.bug.hr/robotika/robotika-kobot-vs-robot-32282, pristupljeno: 31.8.2024.

[15] A History of Robot Programming Languages – Robotiq, https://blog.robotiq.com/the-history-of-robot-programming-languages, pristupljeno: 27.8.2024.

[16] RoboDK API – RoboDK, https://robodk.com/doc/en/RoboDK-API.html#RoboDKAPI, pristupljeno: 31.8.2024.

[17] Python API – RoboDK, https://robodk.com/doc/en/RoboDK-API-Python-API.html, pristupljeno: 31.8.2024.

PRILOZI

I. Programski kod za manipulaciju robotom

I. Programski kod za manipulaciju robotom

```
# You can also use the new version of the API:
from robodk import robolink    # RoboDK API
from robodk import robomath    # Robot toolbox
RDK = robolink.Robolink()

# poveznica s robotom UR5
robot = RDK.Item('UR5')

# targeti
target_home = RDK.Item('Home')
target_1 = RDK.Item('Pozicija 1')
target_2 = RDK.Item('Pozicija 2')
target_3 = RDK.Item('Pozicija 3')
target_4 = RDK.Item('Pozicija 4')
target_5 = RDK.Item('Pozicija 5')
target_6 = RDK.Item('Pozicija 6')
target_7 = RDK.Item('Pozicija 7')
target_8 = RDK.Item('Pozicija 8')

# glavni program
robot.MoveJ(target_home)
robot.MoveJ(target_1)
robot.MoveJ(target_2)
robot.MoveJ(target_3)
robot.MoveJ(target_4)
robot.MoveJ(target_5)
robot.MoveJ(target_6)
robot.MoveJ(target_home)
robot.MoveJ(target_7)
robot.MoveJ(target_8)
robot.MoveJ(target_home)

print('Robot je zavrsio s kretanjem i vratio se u home poziciju')
```