Univerzitetna založba
Univerze v Mariboru

Timi
KARNER

Janez
GOTLIH

PROGRAMIRANJE INDUSTRIJSKIH ROBOTOV

Univerza v Mariboru

Fakulteta za strojništvo

Programiranje industrijskih robotov

Avtorja

Timi Karner

Janez Gotlih

Marec 2022

Naslov **Programiranje industrijskih robotov**
Title *Industrial Robot Programming*

Avtorja Timi Karner
Authors (Univerza v Mariboru, Fakulteta za strojništvo)

Janez Gotlih
(Univerza v Mariboru, Fakulteta za strojništvo)

Recenzija Marko Munih
Review (Univerza v Ljubljani, Fakulteta za elektrotehniko)

Karl Gotlih
(Univerza v Mariboru, Fakulteta za strojništvo)

Jezikovni pregled Danica Gotlih
Language edeting

Tehnična urednika Jan Perša (Univerza v Mariboru, Univerzitetna založba); Janez Čep (Univerza v Mariboru, Fakulteta za strojništvo)
Technical editors

Oblikovanje ovitka Jan Perša
Cover designer (Univerza v Mariboru, Univerzitetna založba)

Grafike na ovitku Industrial robot stock photo, Stock photo ID 852524230, avtor: zssp, istockphoto.com, nakup, 2022
Cover graphics

Grafične priloge Avtorja, 2022. Slika 3.2, Slika 3.3, Slika 3.4, Slika 3.5,Slika 3.7, Slika 3.11, Slika 3.14, Slika 3.15, Slika 4.1, Slika 4.2, Slika 4.3, Slika 4.4, Slika 4.5, Slika 4.6, Slika 4.7, Slika 4.8, Slika 4.10, Slika 4.11, Slika 4.12, Slika 5.1, Slika 8.1, Slika 8.2, Slika 13.1, Slika 14.1, Slika 14.2, Slika 14.3, Slika 14.5 so uporabljene z dovoljenjem podjetja ABB Switzerland Ltd © [2022].
Graphic material

Založnik **Univerza v Mariboru**
Published by **Univerzitetna založba**
Slomškov trg 15, 2000 Maribor, Slovenija
https://press.um.si, zalozba@um.si

Izdajatelj **Univerza v Mariboru**
Issued by **Fakulteta za strojništvo**
Smetanova ulica 17, 2000 Maribor, Slovenija
https://www.fs.um.si, fs@um.si

Izdaja Prva izdaja **Izdano** Maribor, marec 2022
Edition *Published at*

Vrsta publikacije E-knjiga **Dostopno na** https://press.um.si/index.php/ump/catalog/book/652
Publication type *Available at*

CIP - Kataložni zapis o publikaciji
Univerzitetna knjižnica Maribor

621.865.8:004.42(075.8)(0.034.2)

KARNER, Timi
Programiranje industrijskih robotov [Elektronski vir] / avtorja Timi Karner, Janez Gotlih. - 1. izd. - E-publikacija. - Maribor : Univerza v Mariboru, Univerzitetna založba, 2022

Način dostopa (URL): https://press.um.si/index.php/ump/catalog/book/652
ISBN 978-961-286-570-2
doi: 10.18690/um.fs.2.2022
COBISS.SI-ID 97319171

ISBN 978-961-286-570-2 (pdf) **DOI** https://doi.org/10.18690/um.fs.2.2022

Cena Brezplačni izvod **Odgovorna oseba založnika** prof. dr. Zdravko Kačič,
Price *For publisher* rektor Univerze v Mariboru

Citiranje Karner, T. in Gotlih, J. (2022). *Programiranje industrijskih robotov*. Maribor: Univerzitetna založba. doi: 10.18690/um.fs.2.20222
Attribution

ZAHVALA

Avtorja se zahvaljujeva podjetju ABB Inženiring d.o.o., predvsem g. Karlu Jermanu, ki vsako leto poskrbi za brezplačne univerzitetne licence za program RobotStudio. Prav tako bi se rada zahvalila g. Žigu Majdiču za strokovno pomoč.

Posebna zahvala gre tudi strokovnima recenzentoma, red. prof. dr. Marku Munihu ter izr. prof. dr. Karlu Gotlihu, za dobrodošle komentarje in nasvete.

Prav tako bi se rada zahvalila tehničnemu recenzentu g. Janezu Čepu, ki je poskrbel za skladnost učbenika kot e-gradiva.

Posebna zahvala gre tudi ge. prof. Danici Gotlih, ki je poskrbela za jezikovno pravilnost učbenika.

Univerzitetna založba
Univerze v Mariboru

Kazalo

Univerzitetna založba
Univerze v Mariboru

Uporabljene kratice

ABB	–	Asea Brown Boveri
ACIS	–	Format Alan, Charles, Ian's System
ADC	–	Analog to Digital Conversion
AI/AO	–	Analog Input/Analog Output
ASCII	–	American Standard Code For Information Interchange
CAD	–	Computer Aided Design
DAC	–	Digital to Analog Conversion
DI/DO	–	Digitalni vhod oz. izhod
ES	–	Emergency Stop
GSD	–	General Station Description
GSDML	–	General Station Description in HTML
HMI	–	Human Machine Interface
ISO	–	International Organization for Standardization
IP	–	Ingress Protection – stopnja zaščite
IP	–	Internet Protocol
IRC	–	Industrial Robot Controller
KRC	–	Kuka Robot Controller
KUKA	–	Keller Und Knappich Augsburg
LSB	–	Least significant bit
MSB	–	Most significant bit
NC	–	Normally closed
PLC	–	Programmable Logical Controller

PLK – Programabilni logični krmilnik
PTP – Point to Point gib robota
TCP – Tool Center Point, koordinatni sistem orodja
XML – Extensible Markup Language

1 Uvod

1.1 Izkušnje, ki so privedle do nastanka učbenika

Industrijska robotika se na Fakulteti za strojništvo v Mariboru poučuje že kar nekaj let. Pri tem je bil poudarek predvsem na teoretičnem delu, ki je odlična podlaga za razumevanje prakse ter principov delovanja robotov nasploh. Vendar zgolj teorija v industrijskem okolju ne zadošča, tam je glavni poudarek na praktičnem znanju, ki ga je potrebno kar se da hitro implementirati. Pri tem ne gre pričakovati, da boste s preučevanjem tega učbenika postali strokovnjak programiranja industrijskih robov, boste pa prav gotovo korak bližje. Pridobili boste dovoljšno podlago, na kateri boste nadalje gradili svoje znanje. Tako kot pri gradnji hiš, kjer morajo biti temelji trdni, tako moramo znanje najprej osvojiti pri temeljih, šele potem lahko znanje nadgrajujemo.

1.2 Komu je učbenik namenjen?

Učbenik je prvotno namenjen študentom, ki obiskujejo enega izmed predmetov Industrijska robotika, Robotizacija ali predmet Roboti in robotizacija, kot tudi vsem drugim, ki bi želeli obnoviti, osvežiti ali poglobiti znanje programiranja industrijskih robotov. Učbenik naj služi kot splošen uvod v nastavitve in programiranje industrijskih robotov različnih proizvajalcev. Na trgu obstaja več različnih proizvajalcev, ki ponujajo različne tipe industrijskih robotov tako po velikosti oz. dosegu, nosilnosti kot tudi namembnosti posameznih robotov. Vendar imajo vsi roboti tudi svoje skupne značilnosti. Učbenik je sestavljen tako, da bodo zagotovljene vse temeljne zahteve nastavljanja in

programiranja industrijskih robotov. Ker imamo na Fakulteti za strojništvo v Mariboru na voljo industrijskega robota podjetja ABB, bodo vse praktične vaje prikazane na tem robotu. Vendar to ne pomeni, da tega znanja ne bo možno uporabiti tudi na drugih industrijskih robotih. Učbenik je na voljo v obliki e-gradiva in je dostopen vsem. **Če bo vsebina učbenika uporabljena, je potrebno vir ustrezno citirati.**

1.3 Kaj učbenik ponuja?

Najprej bo na kratko podana definicija industrijskih robotov, podani bodo standardi, ki jim morajo industrijski roboti zadostovati, ter standardi, ki jim mora zadostovati industrijska robotska celica. Predstavljeni bodo ključni sestavni deli industrijskega robota, s katerimi se sooča končni uporabnik, ter podani bodo vsi koordinatni sistemi, ki se uporabljajo v robotiki. Prav tako bodo predstavljene karakteristike, s pomočjo katerih določamo lastnosti industrijskih robotovo in na podlagi katerih se odločamo za pravilno izbiro. Predstavljene bodo vse lastnosti, na katere moramo biti pozorni, da bomo z izbiro robota zagotovili nemoteno delovanje proizvodnega procesa in zagotavljali predpisano zmogljivost ter doseg.

V nadaljevanju bo predstavljeno, kako postopamo v primeru nabave novega industrijskega robota, kako robota pritrdimo na podlago in na kaj vse moramo biti pri tem pozorni. Dostavljen robot prav tako še nima okoli sebe postavljene varnostne ograje in tudi ne dosega varnostnih standardov industrijske robotske celice. Zato mora programer v tem primeru biti posebej pozoren na okolico robota, druge naprave ter morebitne delavce v okolici dosegljivosti robota.

Po uspešnem zagonu in zagotovitvi varnosti bo predstavljen splošen postopek priprave robota na uporabo digitalnih vhodno-izhodnih signalov za namene uporabe prijemal oz. drugih komponent. V tem primeru bo podan postopek konfiguriranja DI/DO-enote na primeru ABB-robota.

Ko imamo pripravljene signale za uporabo prijemala oz. orodja, lahko nadaljujemo s konfiguracijo orodja. Pri tem imamo v mislih predvsem določevanje TCP (Tool Center Point) oz. koordinatnega sistema orodja. Za določevanje koordinatnih sistemov imamo na voljo več postopkov, ki pa so odvisni od tega, katere podatke imamo na voljo. Prav tako bo potrebno definirati težišče orodja, masne vztrajnostne momente in maso obdelovanca. Več o tem bo predstavljeno v poglavju 6.

Po uspešni konfiguraciji TCP točke orodja lahko pričnemo z nastavljanjem koordinatnega sistema objektov. V poglavju 7 bodo predstavljeni vsi koordinatni sistemi, ki se uporabljajo v industrijski robotiki. Predstavili bomo tudi koordinatni sistem orodja, ki ni pritrjen na robotsko prirobnico, temveč je fiksno nameščen nekje v okolici dosegljivega prostora robota.

V večini primerov robot ne deluje kot samostojna enota, temveč deluje v sodelovanju z nadzornim sistemom. Najpogosteje je to PLK (programabilni logični krmilnik) oz. angl. PLC (Programmable Logic Controller) enota, ki komunicira preko serijske komunikacije z industrijskim robotom. V poglavju 8 bo predstavljena najpogosteje uporabljena serijska komunikacija, ki se trenutno pojavlja v industrijskem okolju, to je PROFINET protokol. Prav tako bo podana konfiguracija PROFINET protokola na robotu ABB.

V poglavju 9 bodo predstavljene osnove programiranja industrijskih robotov, predstavljeni bodo osnovni robotski gibi, kako jih robot izvaja, kaj pomeni nastavljena hitrost izvajanja gibov in ali so to prehodne točke ali so to točke zaustavitve. Predstavljene bodo tudi spremenljivke, ki se uporabljajo pri programiranju, vsi odločitveni stavki, ponavljajoče ter druge zanke.

V poglavju 10 bo predstavljen program pošiljanja pozicije preko PROFINET komunikacije. En robot IRB 1200 bo konfiguriran kot gospodar oz. »master«, ki bo po vsaki zaključeni inštrukciji poslal koordinate drugemu robotu IRB 1200, ki bo konfiguriran kot suženj oz. »slave«. Ta bo prebral koordinate in izvedel zahtevan gib. Pri tej nalogi bo bistven poudarek na pošiljanju realnih števil zaokroženih na drugo decimalko preko protokola PROFINET.

V poglavju 11 bo predstavljen primer programiranja štiriosnega robota za namene paletizacije. Podane bodo zahteve glede zlaganja kosov na evro paleto, kakšna mora biti orientacija le-teh in koliko slojev mora biti na paleti. Program bo napisan s pomočjo »for« zanke.

Po uspešnem programiranju robota glede podane naloge za naročnika pa delo robotskega programerja še ni končano. Potrebno je preveriti tudi, kaj se zgodi, če nekdo spremeni režim delovanja robota iz avtomatskega v ročni, ga premakne iz programirane trajektorije in potem spremeni režim zopet nazaj v avtomatskega. Pri tem je potrebno preprečiti premik robota s trenutne točke na prvo točko programa, saj je velika možnost nastanka kolizije. Prav tako je potrebno zagotoviti vračanje robota iz katerekoli točke v varno oz.

»Home« pozicijo. To vračanje mora biti z zmanjšano hitrostjo, varno in pri izvajanju vračanja se mora robot izogniti vsem kolizijam. Na žalost robotski krmilnik ne ve, kje je ovira in kje je ni, zato je to naloga robotskega programerja. Prikazani bosta rutini, kako izvesti vrnitev robotov v varno pozicijo.

Zasuk posameznega industrijskega robota znaša približno +/–170° v prvi osi. Glede na to, za kakšno aplikacijo gre in v kakšni robotski celici robot deluje, je potrebno programsko določiti meje zasukov posameznih osi. Če imamo še možnost kolizije z določenimi objekti ali pa imamo robote, ki delujejo v prostoru, ki je presek več robotov, potem lahko uporabimo tudi omejitve, kjer robotu omejimo gibanje s pomočjo navideznega kvadra. Takšna funkcija je trenutno serijsko vgrajena v krmilnike KUKA-robotov, pri ABB pa je potrebno doplačilo. Prikazani bodo postopki programskega omejevanja in omejevanja s pomočjo navideznih kvadrov.

Za vsakim poglavjem bodo podana tudi vprašanja, s pomočjo katerih bo bralec utrdil svoje znanje. Vsi uporabljeni primeri bodo na voljo študentom Industrijske robotike, Robotizacije in tudi študentom predmeta Roboti in robotizacija. Ti bodo dostopni na eŠtudiju pod ustreznim predmetom. Programska koda, s pomočjo katere smo ustvarili primere, je na voljo v poglavju Priloge 17.

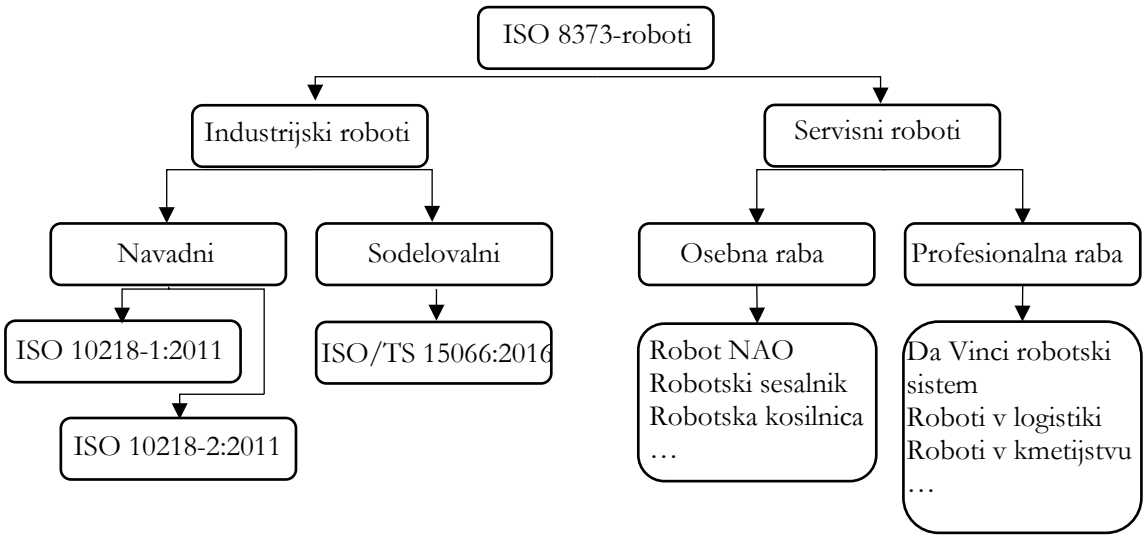
Univerzitetna založba
Univerze v Mariboru

2 Definicija industrijskih robotov ter standardi, ki jih morajo upoštevati

Današnje življenje si brez standardov težko predstavljamo, saj je že skoraj vse standardizirano. Nosimo različne velikosti majic, od S, M, L pa do XXL. Vse te številke so standardizirane, da lahko isto velikost majice kupujemo tako v Sloveniji, Evropi kot tudi širom sveta.

Podobno morajo roboti ustrezati določenim standardnim predpisom. Tako je definicija industrijskega robota podana po standardu **ISO 8373:2012** [1]. Industrijski robot je avtomatsko krmiljen, prosto programabilen večnamenski manipulator z najmanj tremi programabilnimi osmi, ki ga uporabljamo za industrijske aplikacije, bodisi na fiksnem mestu ali pa na mobilni podlagi. Standard **ISO 8373:2012** podaja tudi, da kolaborativni oz. sodelovalni roboti spadajo k industrijskim robotom. Razlika je samo v načinu integracije v proizvodnjo.

Na sliki 2.1 je prikazana delitev robotov po **ISO 8373** še na servisne in industrijske robote ter na robote, ki se uporabljajo v osebni rabi, kot so npr. robotski sesalec oz. robotska kosilnica, in tudi servisni roboti, ki se uporabljajo v profesionalni rabi.



Slika 2.1: Delitev robotov po ISO 8373 [1]

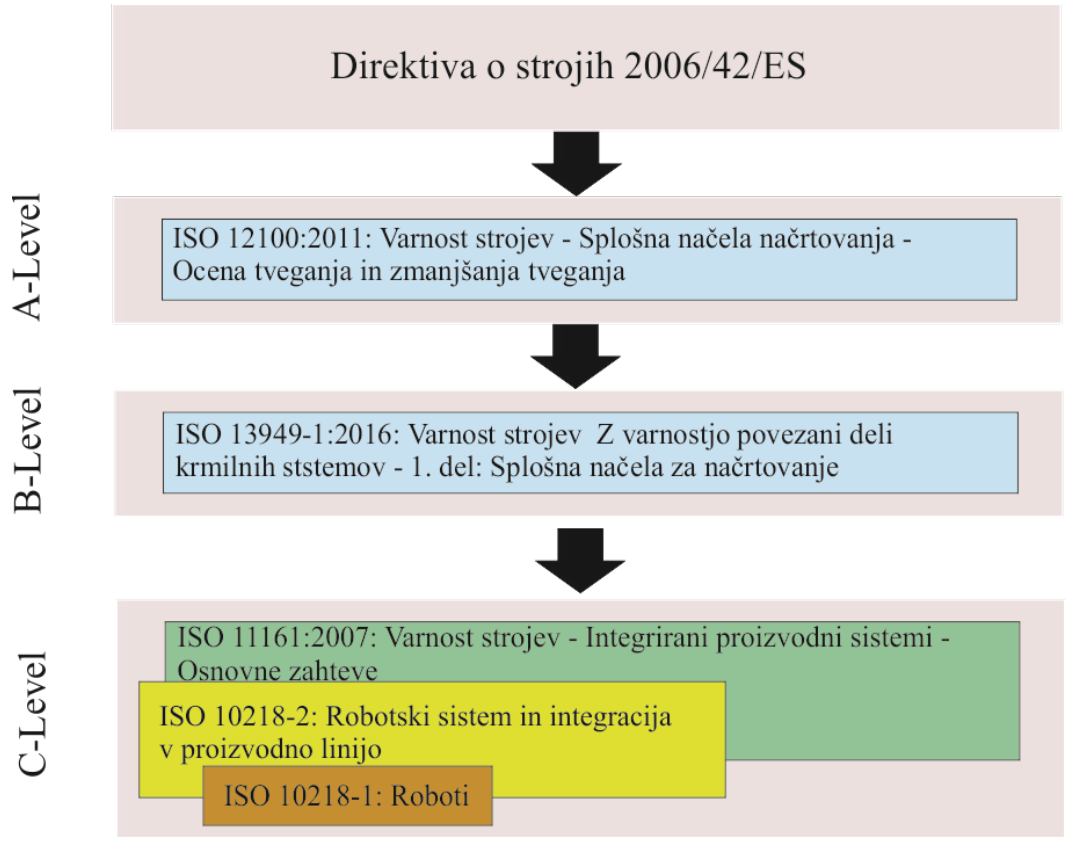
Pri tem je potrebno za navadni industrijski robot upoštevati varnostne predpise za robota po **ISO 10218-1:2011** [2]. Ta standard podaja varnostne zahteve, ki jih mora dosegati robot kot stroj, in sicer nadzor hitrosti gibanja robota, zaustavitve robota, še posebej zaustavitve v sili, ki morajo biti znotraj predpisanega časa, nadzor simultanega gibanja posameznih osi, pravilno delovanje zavor posameznega motorja in med drugim tudi elektrifikacija robota ter uporabo učne konzole oz. »teach pendant«. Tega standarda se morajo držati predvsem proizvajalci industrijskih robotov. Za integratorja in integracijo robota v industrijsko okolje je potrebno upoštevati predpise **ISO 10218-2:2011** [3]. Standard podaja smernice za zagotavljanje varnosti v avtomatskem režimu delovanja, uporabo tipk izklopa v sili, uporabo varnostne ograje, varnostnih vrat in varnostnih ključavnic, varnostnih optičnih zaves, uporabo varnostnih con, programskega omejevanja gibanja robota, namestitve vseh opozorilnih znakov kot tudi zvočnih in optičnih opozoril v primeru avtomatskega režima delovanja oz. napake. Ker se industrijski robot integrira v proizvodnji proces, kjer sodeluje več naprav, je pri integraciji potrebno zadostiti tudi predpisom **ISO 11161:2007** [4]: Varnost strojev – Integrirani proizvodni sistemi – Osnovne zahteve. Standard določa varnostne zahteve za integrirane proizvodne sisteme dveh ali več strojev. Standard **ISO 13949-1:2016** [5]: Varnost strojev – Z varnostjo povezani deli krmilnih sistemov – 1. del: Splošna načela za načrtovanje, zagotavlja varnostne zahteve ter smernice za načrtovanje in integracijo z varnostjo povezanih delov krmilnih sistemov, vključno z načrtovanjem programske opreme oz. programiranja.

Že v fazi načrtovanja robotske celice je smiselno vključiti ter oceniti vse potencialne nevarnosti in se jim že takoj na začetku izogniti. Za ta namen je bil spisan standard **ISO 12100:2011**[6]: Varnost strojev – Splošna načela načrtovanja – Ocena tveganja in zmanjšanje tveganja, ki podaja smernice za ocenitev tveganja in možnosti, da se izognemo le-tem že v fazi načrtovanja proizvodnega procesa.

Po končani integraciji robotske celice v proizvodnji proces je potrebno urediti še oceno tveganja za posamezne sklope in tudi za celoto, kar podaja standard **ISO 12100:2011** [6]. Za pridobitev uporabnega dovoljenja proizvodnega procesa je potrebno predložiti nadzornemu organu oceno tveganja. Pri tem je potrebno podati vsa tveganja, ki se lahko zgodijo, in kako se jim izognemo. Če smo opisali vsa mogoča tveganja in možnosti izogibanja le-teh, podali vsa opozorila, česa operaterji proizvodnega procesa ne smejo početi, namestili vse oznake za nevarnosti, potem smo zagotovili minimalno možnost, da pride do nesreče pri opravljanju avtomatiziranega procesa. Vse naštete standarde prikazuje slika 2.2, katerim je krovni zakon **Direktiva o strojih 2006/42/ES** [7].

Pred predajo proizvodne linije končnemu uporabniku je potrebno napisati navodila za uporabo, ki morajo vsebovati tudi primer, kako ravnati, če sistem preide v napako, ter kaj vse je potrebno storiti, da proizvodnji proces lahko ponovno varno zaženemo v avtomatskem režimu.

Poseben poudarek je potrebno podati uporabi standardov. Standardi so plačljivi in niso prenosljivi. Vsako podjetje jih potrebuje zase in za svoje zaposlene. Če ni tako, lahko ISO-organizacija oglobi podjejte. Prav tako se je potrebno zavedati, da je potrebno zadostiti vsem standardom, ne samo tem, ki pokrivajo industrijske robote in njihovo integracijo, če ste integrator proizvodnega procesa v neko obstoječo linijo ali pa proizvodnji proces postavljate na novo.



Slika 2.2: Hierarhično zaporedje standardov pri integraciji industrijskih robotov [8]

Generalno gledano so si industrijski roboti med sabo zelo podobni. Vendar imajo različni proizvajalci različne pristope k nastavljanju in programiranju robotov. Prav tako so pri istem proizvajalcu roboti z različnimi karakteristikami. Tako je vedno priporočljivo najprej prebrati varnostna navodila za varno uporabo robota, pregledati posebnosti priključitve robota in varnostne predpise, šele nato se lahko lotimo programiranja industrijskega robota. Pri tem je zopet potrebno poudariti, da se programski jeziki, ki jih uporabljajo proizvajalci robotov, med seboj razlikujejo. Razlike so večinoma v sintaksi programiranega jezika in tudi v samem načinu izvajanja robotskega programa. Tukaj se zopet obračamo na navodila posameznih proizvajalcev robotov oz. – kar je še bolje – da se kočni uporabnik udeleži enega izmed šolanj pri proizvajalcu oz. zastopniku.

2.1 Vprašanja za utrjevanje snovi

1. Zakaj uporabljamo standarde?

2. Kakšna je razlika med navadnim in sodelovalnim industrijskim robotom?

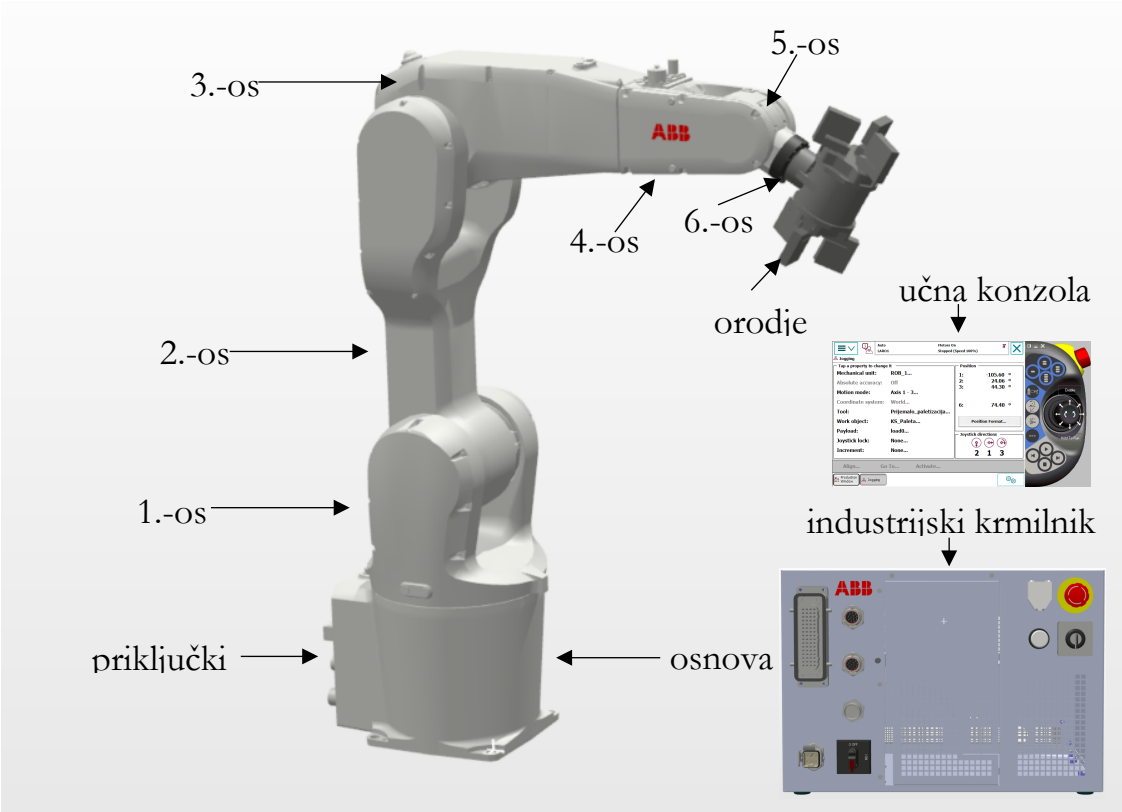
3. Katere standarde je potrebno upoštevati pri implementaciji industrijskih robotov v proizvodno celico?

ZAPISKI

Univerzitetna založba
Univerze v Mariboru

3 Ključni sestavni deli industrijskega robota in njegove karakteristike

Standardni industrijski robot ima šest prostostnih stopenj. Povedano drugače, ima šest servomotorjev, ki skrbijo, da lahko robot doseže vse točke v njegovem delovnem prostoru z različnimi možnimi orientacijami. Prvi trije servomotorji skrbijo za doseganje vseh možnih pozicij, zadnji trije pa za doseganje več možnih orientacij. Na žalost znotraj njegovega delovnega območja obstajajo tudi točke, kjer se robotu bistveno zmanjša gibljivost. Tem točkam pravimo singularne točke. Slika 3.1 prikazuje ključne sestavne dele robota, kot jih vidi končni uporabnik.



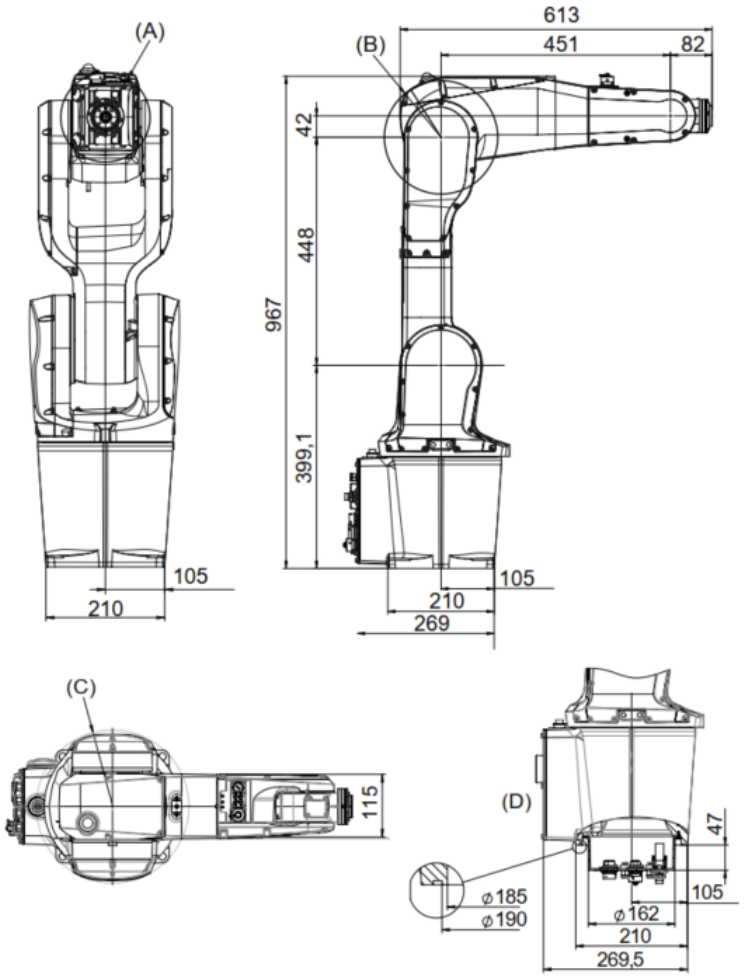
Slika 3.1: Zgradba industrijskega robota, kot ga vidi končni uporabnik.
Vir: lasten.

Za končnega uporabnika je najpomembneje, da zna izbrati najprimernejšega robota za svojo aplikacijo. Pri tem mislimo predvsem na karakteristike industrijskega robota.

3.1 Karakteristike industrijskih robotov oz. priporočila glede dosegljivosti in nosilnosti

Dve bistveni karakteristiki industrijskih robotov, po katerih se roboti tudi označujejo in tržijo, sta dosegljivost in nosilnost robota. Dosegljivost robota nam podaja doseg pete osi robota. Nosilnost pa podaja podatek, s kakšno maso lahko obremenimo robota na končni prirobnici oz. na šesti osi, da še kljub teži zagotavlja predpisano ponovljivost. Ponovljivost je tretji najpogostejši podatek, s katerim se karakterizirajo industrijski roboti. Ponovljivost nam pove, s kakšnim odstopanjem se bo robot vrnil v predpisano točko po tem, ko je to točko zapustil. To so tri glavne karakteristike, s katerimi prodajalec trži industrijske robote. A vendar to še zdaleč ni dovolj, da si zagotovimo brezhibno in nemoteno delovanje industrijskih robotov pri maksimalni obremenitvi in maksimalni možni hitrosti.

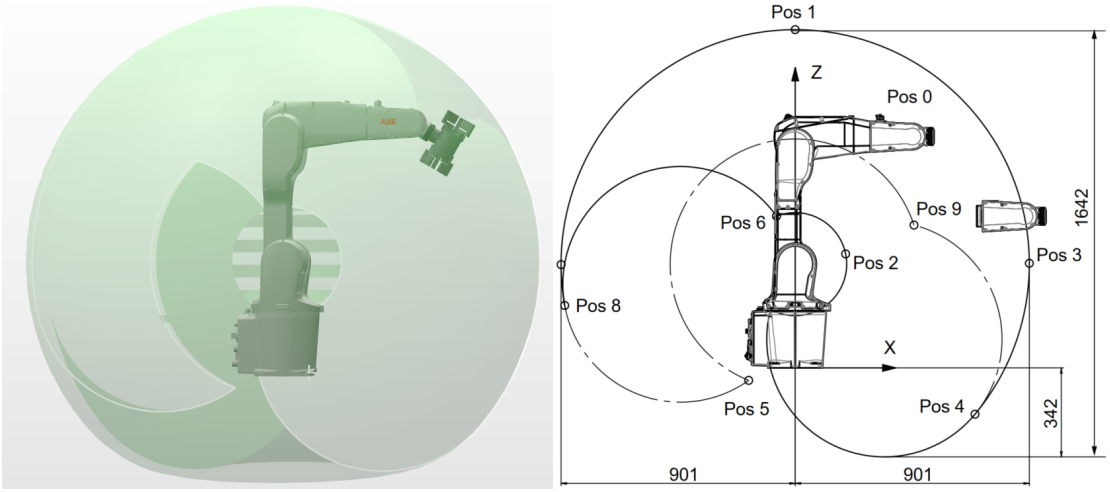
Na Fakulteti za strojništvo v Mariboru imamo ABB-jevega industrijskega robota z oznako IRB 1200. Zanimivost ABB-jevih oznak je, da sama oznaka ne pove bistva o robotu, temveč je to zgolj oznaka. Uporabnik mora to poiskati na spletu. KUKA na drugi strani ima svoje robote označene z nosilnostjo npr. KUKA KR 40 PA, kar pomeni, da gre za robota z nosilnostjo 40 kg, ki je namenjen za paletizacijo. Industrijski robot IRB 1200 ima dve možni izvedbi, in sicer 5 kg nosilnosti pri 0.9 m dosegljivosti ter 7 kg nosilnosti pri 0.7 m dosegljivosti. Na fakulteti imamo robota s prvo karakteristiko. Slika 3.2 prikazuje dimenzije IRB 1200 5 kg/0.9 m.



Slika 3.2: Dimenzije industrijskega robota ABB IRB 1200 [9]

Slika 3.3 prikazuje dosegljivost robota IRB 1200. Na levem delu je prikazana dosegljivost istega robota v programu RobotStudio [10]. S temno zeleno barvo je označen del prostora, ki za robota ni dosegljiv. Na desnem delu pa je prikazana dosegljivost robota po navodilih proizvajalca. Program RobotStudio je namenski program za »offline« programiranje ABB-jevih robotov. Program bo predstavljen v poglavju 6.1.1.

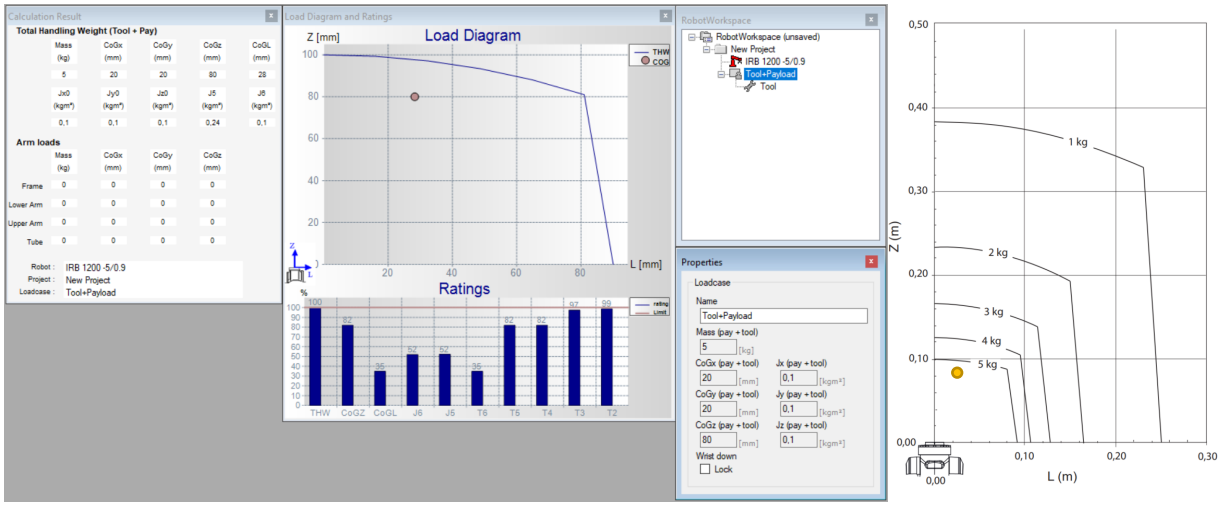
Pri dosegljivosti je potrebno poudariti še, da se le-ta poveča, ko na robota namestimo orodje, kar pa še ne pomeni, da bomo oddaljene točke dosegli v katerikoli orientaciji. Primerno dosegljivost je najlažje testirati v namenskih programih za industrijske robote, kot je npr. RobotStudio, še preden se dejansko odločimo za nakup robota.



Slika 3.3: Dosegljivost robota IRB 1200 5 kg/0.9 m. Levo v programu RobotStudio, desno po navodilih proizvajalca [9].

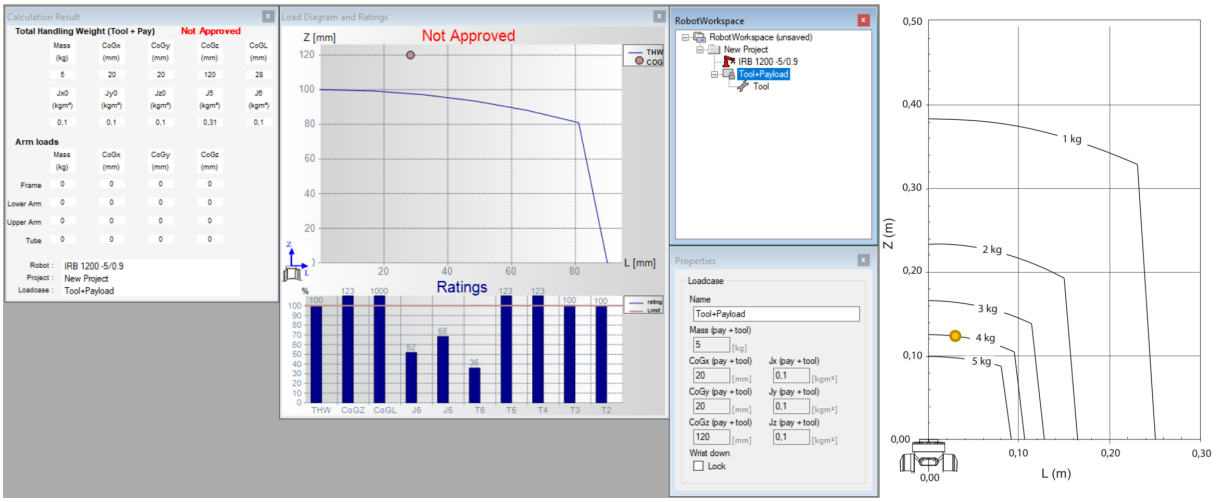
3.1.1 Obremenitveni diagram industrijskega robota

Proizvajalci nosilnost robotov vedno podajajo v eni sami številki, kot je npr. 5 kg za IRB 1200. Vendar je potrebno vedeti, kakšno orodje bo pritrjeno na robota. Ali bo to eno prijemalo ali bosta dve prijemali, pod kotom 45 ali 90°. Ali bo prijemalo imelo podaljšek ipd. Vsak proizvajalec robota poleg dosegljivosti podaja tudi **obremenitveni diagram**, ki pa je namenjen predvsem osi 5, saj je v osi 5 manjši servomotor, kot je v oseh 1–3. S tem je tudi najšibkejši člen industrijskega robota. V navodilih posameznega proizvajalca je posebej poudarjeno, da proizvajalec ne jamči zagotovljene ponovljivosti, ne jamči za garancijo motorjev in zobniških zglobov, če se ne upošteva obremenitvenih diagramov. Že s tega vidika je smotrno upoštevati obremenitvene diagrame, poleg tega bo tudi delovanje robota bolj zvezno. Tako podjetje ABB kot tudi KUKA ponujata brezplačne programe za kalkulacijo ustreznosti obremenitvenih diagramov. Pri ABB je to program RobotLoad [11], pri KUKA pa KUKALoad [12]. Slika 3.4 prikazuje primerno obremenitev robota IRB 1200. Pri določevanju obremenitvenega diagrama je potrebno upoštevati maso prijemala, pozicijo težišča, merjeno od prirobnice, ter vse tri glavne vztrajnostne momente. Najpomembnejša podatka sta zagotovo masa in pozicija težišča. A pravilno definirani vztrajnostni momenti bistveno pripomorejo k izboljšanju zveznosti gibanja industrijskega robota.

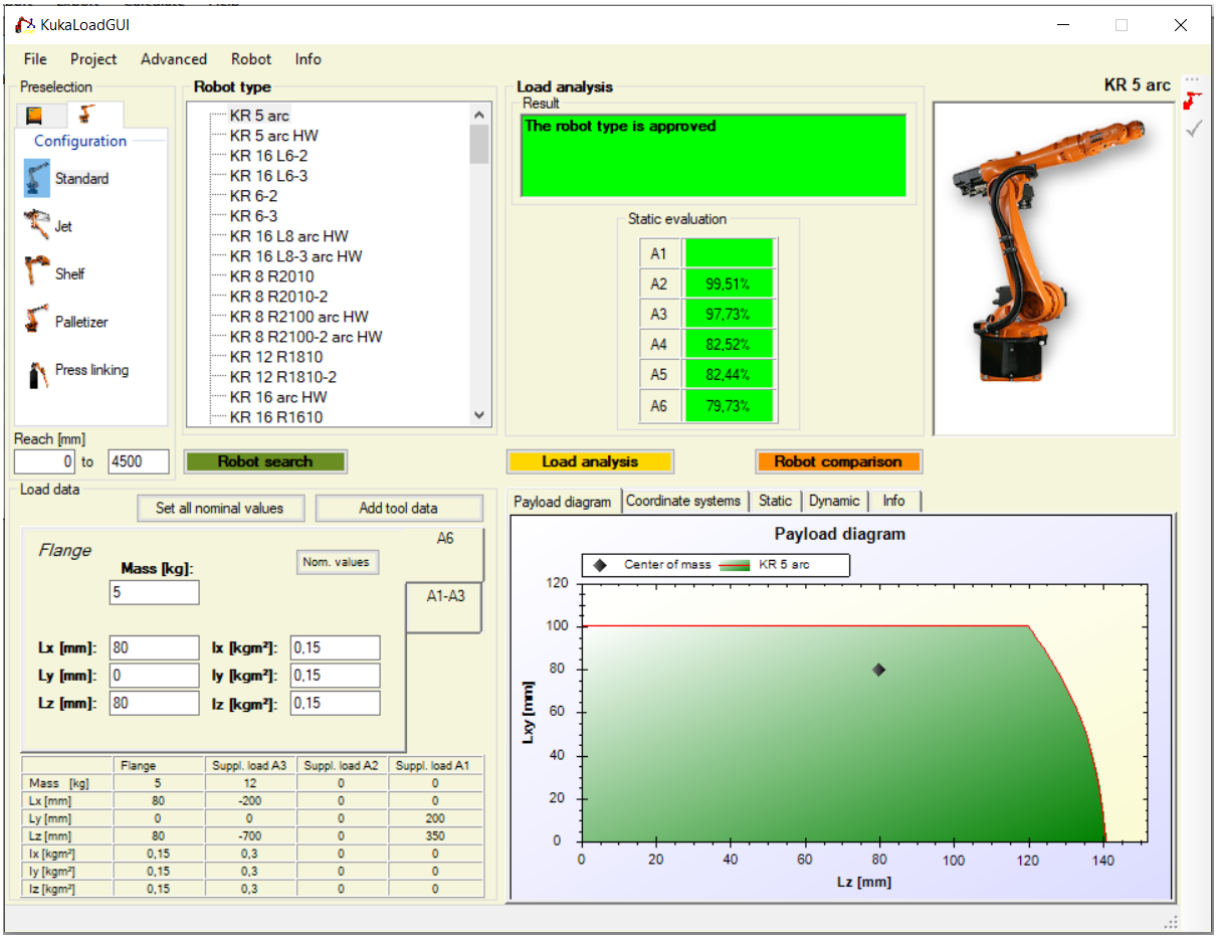


Slika 3.4: Primerna obremenitev robota IRB 1200 v peti osi. Levo: določevanje obremenitvenega diagrama s pomočjo programa RobotLoad. Desno: ročno določevanje obremenitvenega diagrama. [9]

Slika 3.5 prikazuje nepravilno obremenitev robota IRB 1200. Problem nastane, ker je težišče orodja, ki ima maso 5 kg, zunaj predpisanega območja. S tem se na motorju 5 ustvari prevelik navor, ki bistveno skrajša življenjsko dobo motorja. Robot IRB 1200 bi s takšnim prijemalom še vedno deloval, vendar bi lahko prihajalo do pregrevanja motorja v 5. osi, prav tako pa bi se skrajšala njegova življenjska doba. Vprašljiva bi bila tudi ponovljivost robota. Slika 3.6 prikazuje določevanje obremenitvenega diagrama s pomočjo programa KUKALoad.



Slika 3.5: Neprimerna obremenitev robota IRB 1200 v peti osi. Levo: določevanje obremenitvenega diagrama s pomočjo programa RobotLoad. Desno: ročno določevanje obremenitvenega diagrama. [9]



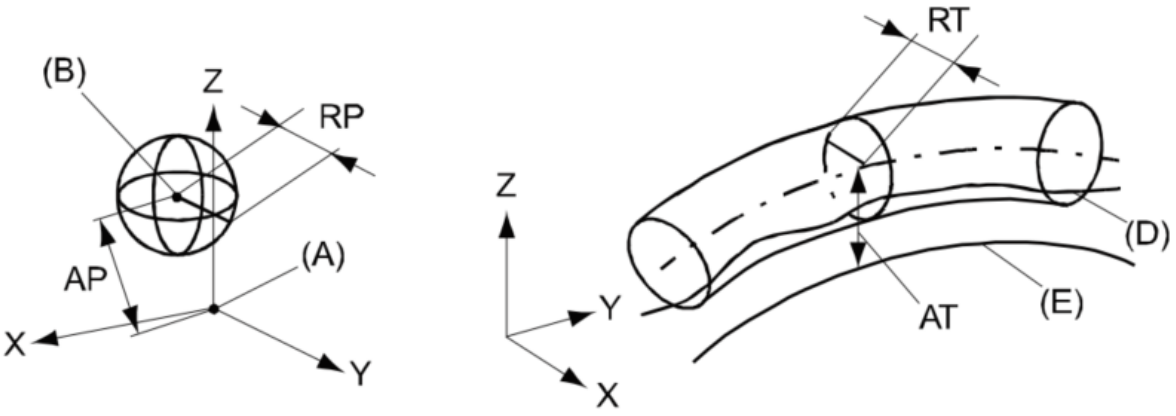
Slika 3.6: Primer določevanja obremenitvenega diagrama s pomočjo KUKA Load
Vir: lasten.

V navodilih posameznega robota so podane tudi omejitve glede maksimalnega navora na motorju na 5. in 6. osi. Ta podatek je potrebno še posebej upoštevati v aplikacijah, kjer robot izvaja razna opravila, ki zahtevajo togost robota, npr. vijačenje. Pri vijačenju se mora robot upirati navoru električnega vijačnika. Če je ta navor prevelik, lahko zopet prihaja do okvar oz. zmanjšanja življenjske dobe robota.

3.1.2 Ponovljivost industrijskega robota

Tretji podatek, ki nas najbolj zanima pri karakteristiki robotov, je poleg dosegljivosti in nosilnosti še ponovljivost. Ponovljivost je definirana kot odstopanje robota od programirane točke, ko je robot to točko zapustil z maksimalno hitrostjo in obremenitvijo ter se potem ponovno vrnil v to točko. Ponovljivost je izvedena tudi na ponavljanju predpisane trajektorije. Ponovljivost določamo s pomočjo standarda ISO 9283:1998 [13]. Slika 3.7 prikazuje ponovljivostni kriterij po prej omenjenem standardu. Po tem standardu se industrijskim robotom določa ponovljivost. Tabela 3.1 podaja opis posameznih oznak

za ponovljivostni kriterij. Tabela 3.2 podaja podatke o ponovljivosti robota IRB 1200 5kg/0.9m.



Slika 3.7: Ponovljivostni kriterij po ISO 9283:1998 [9]

Tabela 3.1: Opis oznak ponovljivostnega kriterija po ISO 9283:1998 [9]

Pozicija	Opis	Pozicija	Opis
A	Programirana pozicija	E	Programirana pot
B	Dosežena pozicija robota	D	Dejanska pot robota
AP	Razdalja od robotske do dejanske pozicije	AT	Maksimalno odstopanje dejanske poti robota od programirane
RP	Toleranca ponovljivosti točke B	RT	Toleranca ponovljivosti poti roboti

Tabela 3.2: Podatki o ponovljivosti robota IRB 1200 5kg/0.9m [9]

Opis	Vrednosti IRB 1200 –5/0.9
Ponovljivost pozicije, RP (mm)	0.025
Točnost pozicije, AP (mm)	0.02
Ponovljivost linearne poti, RT (mm)	0.07
Točnost linearne poti, AT (mm)	0.53
Stabilizacija v točki (s), znotraj 0.1 mm pozicije	0.113

3.2 Zaščita industrijskih robotov glede na IP-standard

Industrijski roboti delujejo v različnih industrijskih okoljih. Lahko delujejo v čistejših okoljih, kjer prah in umazanija nista prisotna, v večini industrijskih okolij pa brez tega ne gre. Če proizvajalci želijo doseči ustrezno življenjsko dobo robotov, jih morajo opremiti z ustrezno zaščito. Pri tem zaščito robotov podajajo glede na IP-standard, ki je določen z EN 60529:1997 [14]. IP-standard ima dve številki, ki določujeta zaščito stroja/robota pred umazanijo in vodo. Pri tem prva številka sporoča zaščito stroja pred velikostjo umazanih

delcev, druga številka pa sporoča, kolikšna je zaščita pred vodnimi kapljicami oz. vodo. Tabela 3.3 prikazuje razčlenitev IP-standarda.

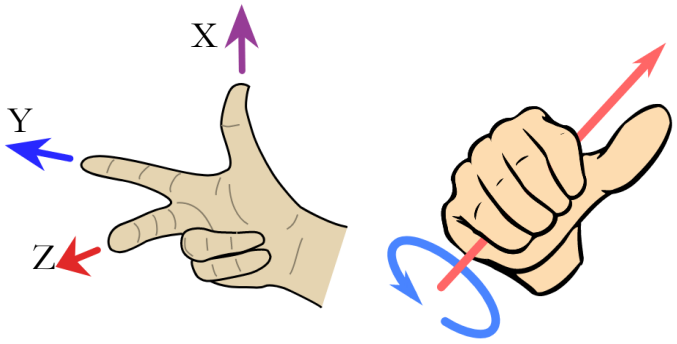
Tabela 3.3: Razčlenitev IP-standarda [15]

IP …	Prva številka: prepustnost delcev	Druga številka: prepustnost tekočin
0	Brez zaščite	Brez zaščite
1	Zaščita pred delci velikosti nad 50 mm: roka, klešče	Zaščita pred padajočimi kapljicami oz. kondenzom
2	Zaščita pred delci velikosti nad 12.5 mm: roka, klešče	Zaščita pred padajočimi kapljicami pod kotom do ±15° od vertikale
3	Zaščita pred delci velikosti nad 2.5 mm: žica, vijak	Zaščita pred padajočimi kapljicami pod kotom do ±60° od vertikale
4	Zaščita pred delci velikosti nad 1 mm: žica	Zaščita pred razpršeno vodo z vseh strani
5	Omejena zaščita pred delci, ni nevarnih mest odlaganja	Zaščita pred vodnim curkom z majhnim pritiskom z vseh strani. Majhna prepustnost je dovoljena.
6	Popolna zaščita pred delci	Zaščita pred vodnim curkom z velikim pritiskom z vseh strani. Majhna prepustnost je dovoljena
7	/	Zaščita pred kratkoročno potopitvijo v vodo
8	/	Zaščita pred dolgoročno potopitvijo v vodo
9k	/	Zaščita pred izpostavitvijo vodnim curkom pod visokim pritiskom in vročo paro

Osnovna IP-zaščita je pri različnih tipih ABB-robotov različna. Robot IRB 1200 ima zaščito IP 40, kar pomeni, da so roboti zaščiteni pred delci velikosti nad 1 mm, kar se tiče tekočin, pa nimajo nobene zaščite. Če pričakujemo, da bo robot deloval v okolju, kjer bi lahko bil izpostavljen tekočinam, potem je potrebno pri izbiri industrijskega robota izbrati ustrezno zaščito. Podjetje ABB tako ponuja še IP 67 zaščito robotov. To pomeni, da ima robot popolno zaščito pred delci in da je lahko izpostavljen tekočini. Ponujajo tudi posebno zaščito robotov, ki so namenjeni delu v livarski industriji. Ti imajo posebno zaščito, ki ni klasificirana po IP-standardu, temveč je klasificirana po internih merilih podjetja. Ti roboti so lahko izpostavljeni vročini, obrizgom in pari.

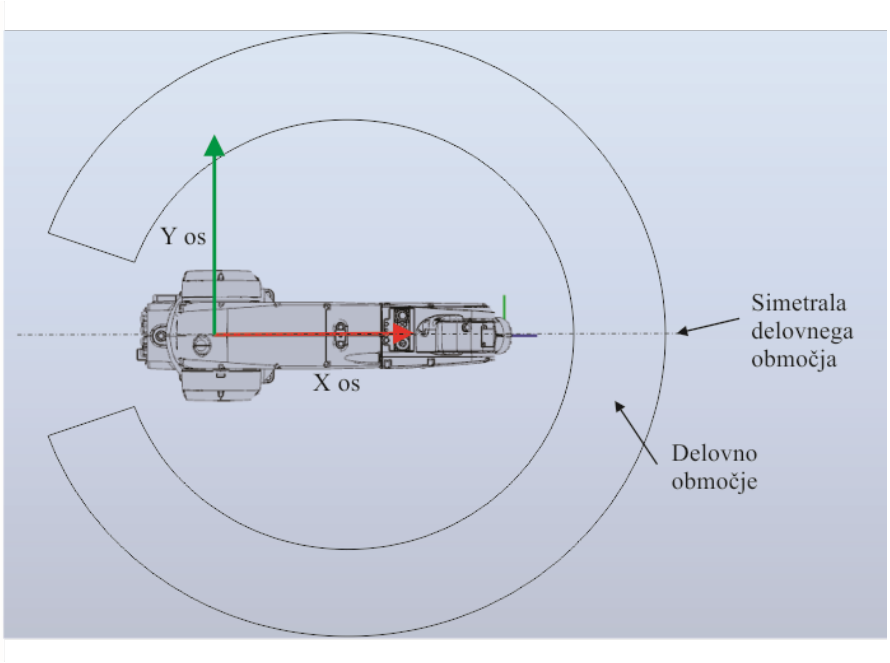
3.3 Koordinatni sistemi industrijskih robotov

V industrijski robotiki se uporabljajo standardni kartezični koordinatni sistemi, ki so desnosučni. Na teh velja pravilo desne roke. Palec kaže v smer osi X, kazalec v smeri osi Y in sredinec v smeri osi Z. Pri rotacijah koordinatnih sistemov si prav tako pomagamo z desno roko, da določimo smer pozitivne rotacije. Tukaj velja matematična pozitivna smer, ki je pozitivna kontra smeri potovanja urinega kazalca. Pravilo pravi, da palec kaže v smeri pozitivne osi, drugi prsti pa kažejo pozitivno rotacijo. Slika 3.8 prikazuje obe pravili.



Slika 3.8: Pravilo desne roke pri koordinatnem sistemu levo in določevanju pozitivne smeri rotacij desno [16, 17]

Industrijski roboti imajo več koordinatnih sistemov, ne po vrsti koordinatnih sistemov, temveč glede na to, kje se nahajajo. Osnovni koordinatni sistem industrijskega robota je bazni koordinatni sistem (BASE). Tega imenujemo tudi robotski koordinatni sistem. Definicijo baznega koordinatnega sistema podaja standard ISO 9787:2013 [18]. Nahaja se na prirobnici od osnove robota, kjer je robot pričvrščen na podlago ali podstavek. Os X baznega koordinatnega sistema leži na preseku delovnega območja, kot to prikazuje slika 3.9. Po pravilu desne roke hitro določimo še os Y in os Z. Pri določevanju baznega koordinatnega sistema si lahko pomagamo tudi na drug način. Večina robotov ima os X robotskega ali baznega koordinatnega sistema, določenega v smeri priključkov (napajanje, pnevmatika ipd.). Ko imamo enkrat določeno os X, potem uvedemo pravilo desne roke, kjer vemo, da os Z gleda navzgor. S tem določimo še os Y.



Slika 3.9: Definicija baznega koordinatnega sistema robota po standardu ISO 9787:2013
Vir: lasten.

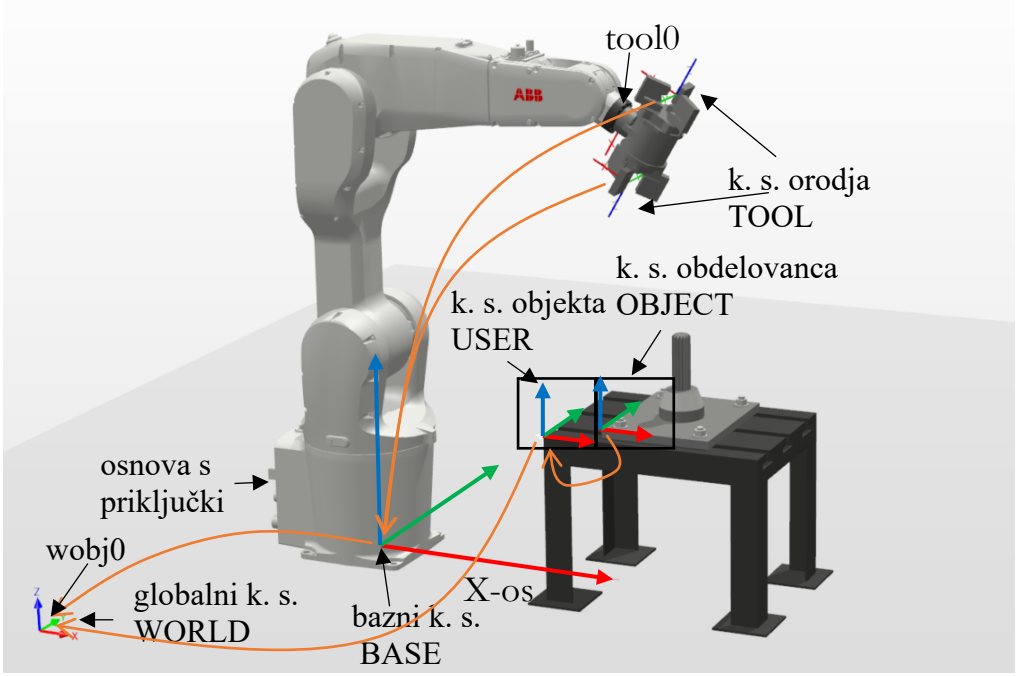
Industrijski robot ima tudi globalni koordinatni sistem (WORLD), ki se **nahaja na istem mestu** kot koordinatni sistem robota (BASE), razen če je globalni koordinatni sistem namensko prestavljen.

Najpogosteje pa se uporabljata koordinatni sistem orodja (TOOL) in koordinatni sistem objekta (WORKOBJECT; OBJECT). Koordinatni sistem objekta je na primeru ABB-robotov razdeljen na dva koordinatna sistema: prvega, ki definira referenco koordinatnega sistema objekta glede na globalni koordinatni sistem, in drugega, ki definira koordinatni sistem obdelovanca na koordinatni sistem objekta. To je specifično samo za ABB-jeve robote. Le-ti ločijo koordinatni sistem objekta na dva ločena koordinatna sistema. Ta sistem ima svoje prednosti. Če je potrebno spremeniti pozicijo objekta, potem spreminjamo nastavitve koordinatnega sistema objekta in pozicija in orientacija obdelovanca se glede na objekt ohranita. Če rotira obdelovanec, se spremeni samo koordinatni sistem obdelovanca, koordinatni sistem objekta pa ostane na istem mestu.

Pri novih robotih se bazni in globalni koordinatni sistem nahajata v isti točki, koordinatni sistem orodja pa se nahaja na vrhu prirobnice 6. osi. Ko pritrdimo orodje, je potrebno ta koordinatni sistem temu primerno spremeniti. En razlog je že bil podan v prejšnjem podpoglavju 3.1.1, da povečamo ponovljivost in zanesljivost delovanja robota. Drugi razlog je praktične narave. Z natančno definiranim koordinatnim sistemom orodja oz. točke TCP bomo bistveno lažje manipulirali z robotom in bomo lažje vodili robota do

željene točke. Prav tako se predpisana hitrost pomikanja robota navezuje na TCP-točko orodja. Če TCP ni v redu definiran, bo tudi odstopanje hitrosti veliko in robot ne bo dosegal predpisanih zahtev, kar pa ni napaka proizvajalca, temveč robotskega programerja. Slika 3.10 prikazuje položaje različnih koordinatnih sistemov, uporabljenih v robotiki. Povemo naj še, da koordinatni sistem orodja ni nujno pritrjen na robotsko prirobnico. Robot lahko določen izdelek brusi ali pa nanaša lepilo na izdelek, pri čemer je šoba nanašanja fiksirana, robot pa mora objekt premikati po neki tirnici.

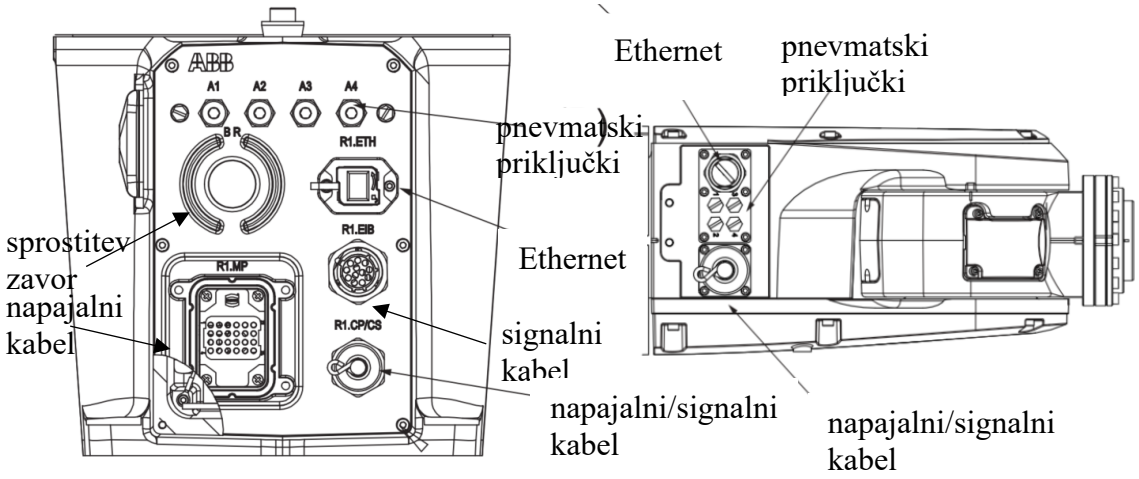
Prav tako lahko koordinatni sistem objekta vežemo na gibljiv objekt. Npr. izdelek se pripelje po tračnem transporterju in v trenutku, ko se prekine fotocelica, se na izdelek pritrdi gibljiv koordinatni sistem, ki potuje z isto hitrostjo, kot potuje izdelek. Tako bo robot točno vedel, kam na gibajoč izdelek mora nanesti lepilo, ali pa odložiti drugi izdelek. Določevanje in spreminjanje posameznih koordinatnih sistemov bo podrobneje razloženo v poglavjih 6 in 7.



Slika 3.10: Položaji različnih koordinatnih sistemov v robotiki
Vir: lasten.

3.4 Priključki industrijskih robotov

Industrijski roboti se uporabljajo v različnih aplikacijah, kjer imajo na prirobnici 6. osi priključena različna orodja. Ta orodja potrebujejo napajanje, pnevmatiko, napajanje različnih senzorjev, komunikacijo s posebnimi merilci razdalje itd. Vse te žice, kable oz. cevi je potrebno pripeljati do orodja. Če že vnaprej vemo, kaj vse bomo potrebovali pri orodju, lahko to proizvajalec vgradi v ogrodje robota. Prednosti tega so, da imamo manj kablov, žic in cevi, ki visijo okoli robota. S tem se izognemo pritrjevanju teh kablov, izognemo se morebitnim nesrečam, da bi te kable potrgali pri kakšnem gibu, ter izognemo se ovijanju kablov okoli robota pri izvajanju nalog v avtomatskem režimu. Vendar je potrebno to željo pri proizvajalcu izraziti že prej. Priključki za kable, žice oz. cevi se namestijo poleg obstoječih na osnovi in speljejo skozi ogrodje do 3. ali pa 4. osi, od koder se potem dalje namestijo na orodje robota. Slika 3.11 prikazuje dodatne priključke tako na osnovi robota kot na 4. osi.



Slika 3.11: Dodatni priključki na robotu: levo na osnovi in desno na 4. osi [9]

3.5 Krmilnik industrijskega robota

Industrijski robot je zgolj mehanski sestav s servomotorjem, ki ga krmili krmilnik robota. Glavna naloga krmilnika je krmiljenje vseh šestih motorjev, da vrh robota oz. TCP točka dosega željene točke, predpisane trajektorije s predpisano hitrostjo ter da izvaja gibe, ki smo mu jih predpisali. Poleg izvajanja robotskega programa krmilnik skrbi tudi za varnost, za komunikacijo z drugimi napravami, kot so senzorji, vhodno/izhodnimi signali DI/DO ter nadzornim sistemom preko serijske komunikacije. Slika 3.12 prikazuje 4 različne tipe robotskih krmilnikov podjetja ABB. Vsi ABB-jevi krmilniki so tipa IRC5, različne so samo konfiguracije. Vse konfiguracije pa uporabljajo isti procesor.

Podjetje ponuja kompaktno verzijo oz. Compact Controller, ki je namenjena manjšim robotom, ima samo eno fazo priključne napetosti ter integriranih 16 DI/DO-priključkov. Integriran ima tudi DeviceNet [19] in pa RS232 [20] komunikacijski protokol. Vse to je seveda na voljo tudi opcijsko. Vsi drugi komunikacijski protokoli so opcijski, prav tako tudi dodatne servoosi. Pri tem je potrebno poudariti majhnost kompaktne verzije.

Najpogosteje se uporablja enojni krmilnik oz. Single Cabinet Controller, ki ima dovolj prostora za vse nadaljnje nadgradnje ter omogoča s pomočjo funkcije MultiMove [21] upravljanja do 4 robotov simultano.

Podjetje prav tako ponuja vgradni krmilnik oz. Panel Mounted Controller, ki ga lahko vgradimo v elektroomaro. Krmilnik nima ogrodja, ima samo krmilni in močnostni del ter vse druge priključke, ki so potrebni za krmiljenje robota.

Zadnji krmilnik, ki ga podjetje ponuja, je krmilnik, namenjen robotom, ki nanašajo barvo oz. Paint Controller. To je namenski krmilnik, ki ima že integriran program za nanašanje barve. Pri tem je potrebno zadostiti tudi določenim varnostnim zahtevam.

Vse razširitvene kartice, ki jih želimo integrirati **znotraj** ABB-jevega krmilnika, morajo biti od podjetja ABB. Pri tem ne moremo uporabiti kartic tujega proizvajalca. Zunanje razširitvene kartice pa so lahko od drugih podjetij.

Podobno kot podjetje ABB tudi KUKA ponuja različne velikosti krmilnikov. KUKA imenuje svoje krmilnike KRC4, ki pa se med seboj razlikujejo po konfiguraciji. Vse konfiguracije uporabljajo isti procesor. Slika 3.13 prikazuje pet različnih konfiguracij krmilnikov KRC4.

Kompaktna verzija je zopet najmanjša in ima samo enofazno napajalno napetost. Vse druge verzije imajo trifazno priključno napetost, pri čemer se povečuje tudi enofazno število dodatnih servoosi, ki jih lahko krmilnik krmili. KUKI-ni krmilniki nimajo integriranih DI/DO, temveč je potrebno definirati komunikacijski protokol, s katerim želimo dostopati do razširitvene kartice, ki omogoča DI/DO-izhode. Najpogosteje se za to uporablja komunikacija PROFINET [22]. Če želimo imeti v krmilniku na voljo PROFINET komunikacijo, je to potrebno specificirati pri nabavi. Prav tako je potrebno povedati, ali bo robot gospodar, suženj ali oboje. S to opcijo imamo proste roke pri nabavi razširitvenih kartic, saj lahko integriramo kartice poljubnih proizvajalcev, ki delujejo na principu PROFINET komunikacije.

a) Compact Controller

b) Single Cabinet Controller

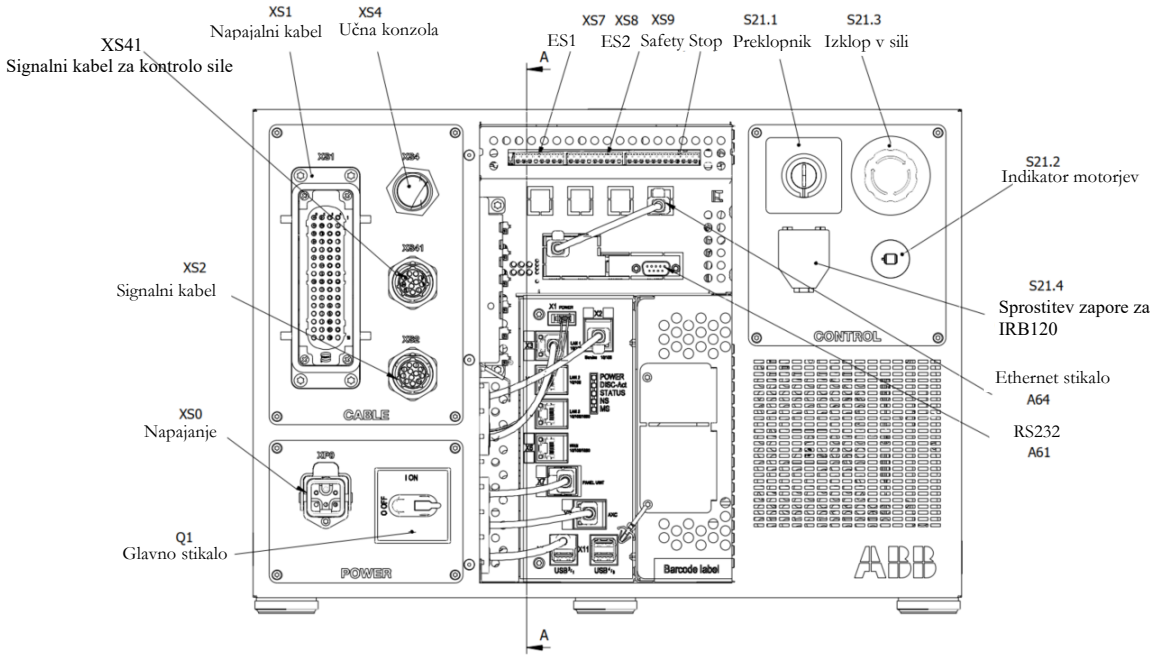
c) Panel Mounted Controller

d) Paint Controller

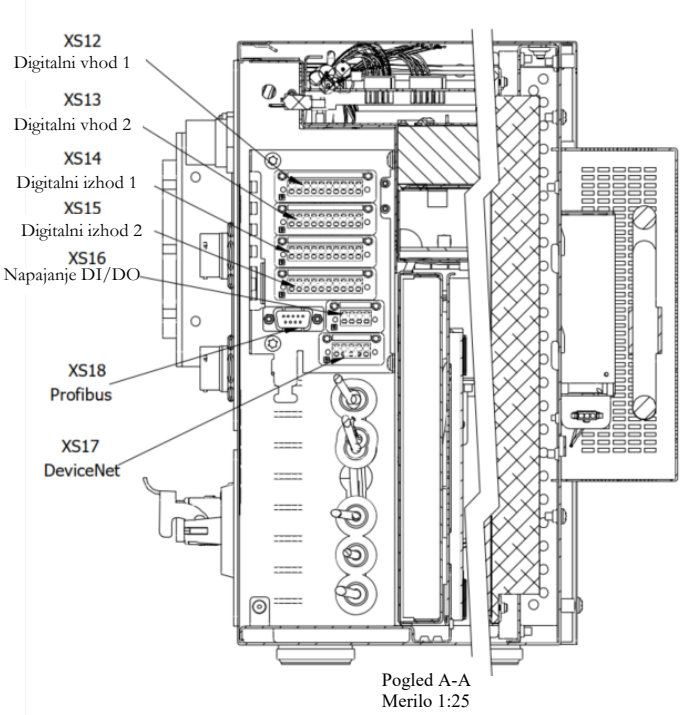
Slika 3.12: Vrste industrijskih krmilnikov pri ABB: a) kompaktni, b) enojni oz. standardni, c) vgradni, d) krmilnik za aplikacije barvanja [23]

Slika 3.13: KUKA krmilniki: a) KRC4 compact, b) KRC4 smallsize-2, c) KRC4, d) KRC4 midsize, e) KRC4 extended [24]

V nadaljevanju bodo na kratko predstavljeni priključki in gumbi na krmilniku IRC5 Compact na sliki 3.13. Krmilnik je kompaktne izvedbe za majhne robote, saj za napajanje potrebuje zgolj eno fazo. Krmilnik je opremljen s stikalom za izklop v sili (S21.3), preklopnim stikalom za preklop režima delovanja robota (S21.1) in gumbom, ki je hkrati tudi indikator za vklop motorjev (S21.2). Poleg tega ima še glavno stikalo za vklop krmilnika in robota (Q1). Krmilnik ima štiri glavne priklope, priklop za napajanje (XS0), priklop za napajanje motorjev oz. napajalni kabel (XS1), priklop za signale iz motorjev oz. signalni kabel (XS2) ter priklop za ročno konzolo (XS4). Krmilnik ima integriranih 16 DI/DO (XS12-15), ki s pomočjo glavnega računalnika komunicira preko DeviceNet protokola. Zaradi galvanske ločitve DI/DO je le-te potrebno **napajati ločeno** (XS16). Krmilnik vsebuje še varnostne vhode za dodatne tipke izklopa v sili (XS7–8). Za komunikacijo z nadzornim sistemom skrbi področno vodilo DeviceNet (XS17) ali pa Profibus (XS18). Na krmilnik robota se povežemo preko »Service Port« priključka z oznako X2 s pomočjo omrežnega kabla.



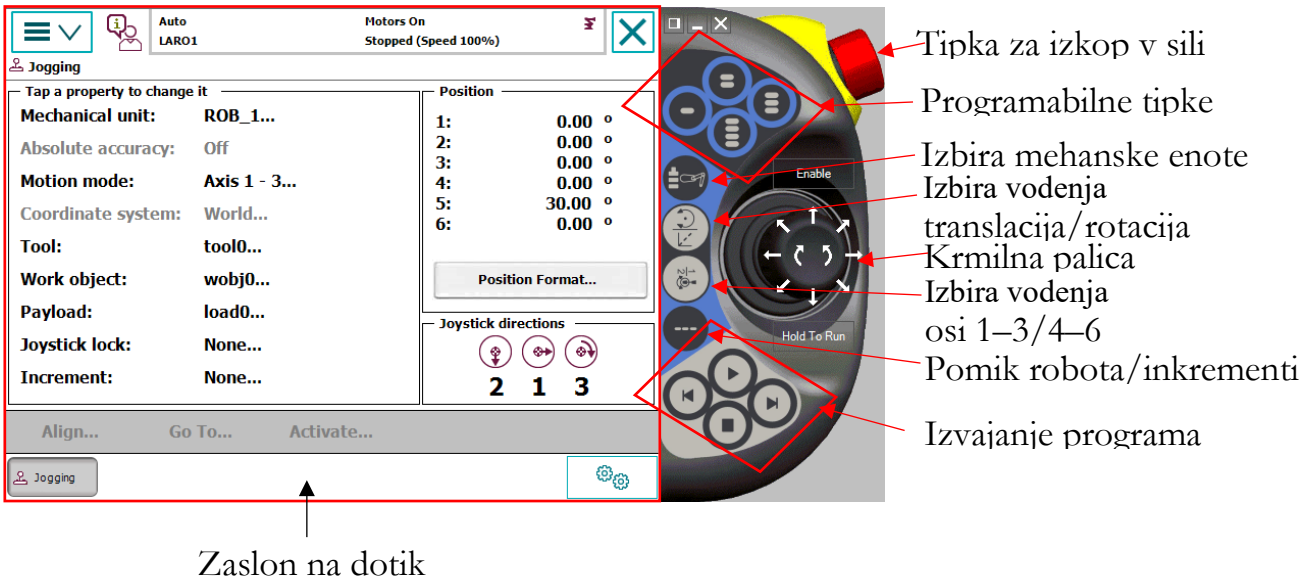
Slika 3.14: Priključki industrijskega krmilnika IRC5 Compact [25]



Slika 3.15: Priključki za periferijo krmilnika IRC5 Compact [25]

3.6 Učna konzola industrijskega robota

Vsak industrijski robot ima svojo učno konzolo, preko katere robota programiramo, vodimo in nastavljamo določene parametre. Najpogosteje so te konzole fizično ožičene in povezane s krmilnikom. Zadnje čase se na tržišču že pojavljajo konzole, ki so brezžične in lahko s pomočjo ene konzole upravljamo več robotov, vendar ne hkrati, temveč posamično. Prav tako so konzole različnih proizvajalcev različno ergonomsko oblikovane. Slika 3.16 prikazuje ročno konzolo ABB-jevih robotov. Konzola je oblikovana tako, da jo enakovredno lahko uporabljajo levičarji in desničarji. V nastavitvah se lahko zaslon obrne za 180° in v tem primeru se jo drži z desno roko. Je ena izmed najlažjih na tržišču, ima zaslon na dotik, majhno število gumbov in krmilno palico. Krmilna palica zaznava velikost odklona tako, da se robot temu primerno hitro premika. Na hrbtni strani ima še pašček, s katerim si pomagamo, da lažje držimo konzolo, ter varnostno tipko za vklop motorjev. Ta tipka ima dva nivoja. Prvi je delovni, pri čemer motorji dobijo napajanje. Drugi pa je mirovni, kjer se napajanje motorjev prekine. To je dodatna varnost v primeru, kadar se robot nekam zaleti, se ustrašimo in pri tem dodatno stisnemo tipko. Stikalo pade na drugi nivo in motorjem prekine napajanje ter posledično ustavi robota.



Slika 3.16: Učna konzola ABB robotov

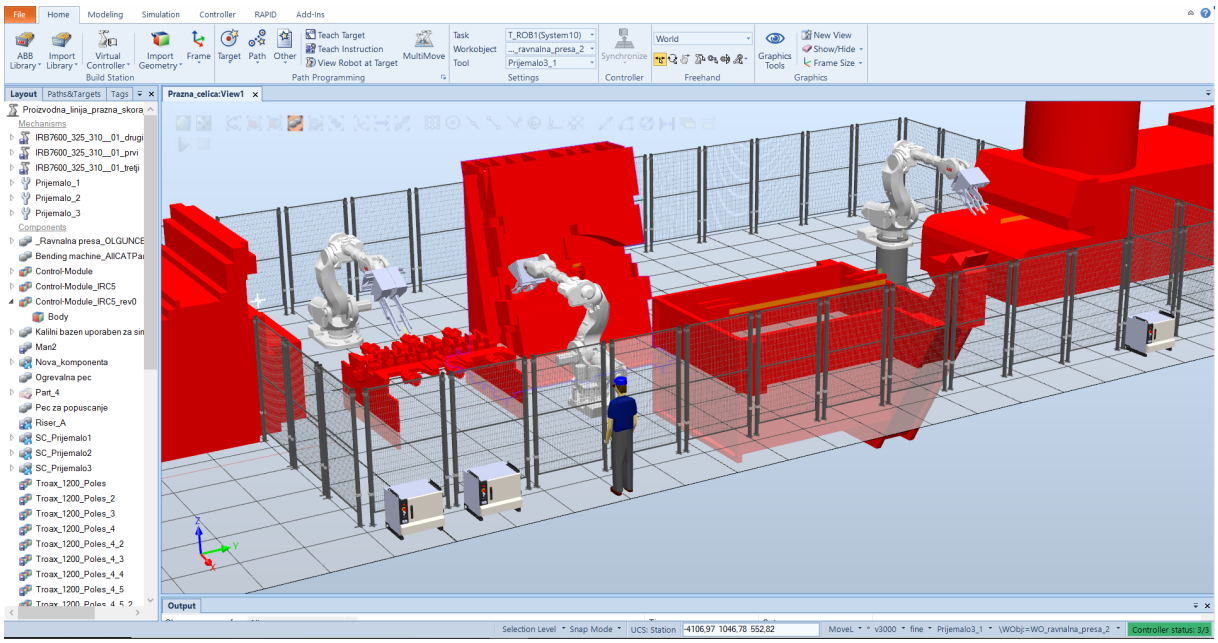
Vir: lasten.

3.7 Program za »offline« programiranje industrijskih robotov

Vsak proizvajalec že ponuja svojo verzijo računalniškega programa za programiranje svojih industrijskih robotov. Programiranju robotov preko računalnika, ko računalnik ni povezan na krmilnik robota, pravimo »offline« programiranje. Ker nimamo pravilnega slovenskega izraza, bomo v nadaljevanju uporabljali kar to besedo. Prav tako ima vsak proizvajalec robotov svoj programski jezik, s katerim programira. Podjetje ABB za svoje robote uporablja programski jezik RAPID, ki temelji na programskem jeziku C.

Podjetje ABB ponuja program RobotStudio [26] za »offline« programiranje njihovih robotov. Preko progama lahko že v fazi snovanja robotske celice testiramo dosegljivost robota, pripravimo program, testiramo delovni cikel izvajanja programa ter določimo vse DI/DO signale, ki jih za to potrebujemo. Slika 3.17 prikazuje primer industrijske robotske celice s tremi roboti, narejene v programu RobotStudio. Na uradni spletni strani podjetja ABB RobotStudio lahko najdete program, ki je na voljo v poskusni verziji za 30 dni ter veliko uporabnih videov. Prav tako so navodila dostopna tudi v diplomskem in magistrskem delu **Luka Kumerja** [27, 28]. V nadaljevanju bo prikazana konfiguracija DI/DO-signalov, poglavje 5.1.1, in priprava PROFINET komunikacije, poglavje 8, preko programa RobotStudio za robote ABB.

V zadnjem času se na tržišču pojavljajo tudi univerzalni programi, ki vsebujejo bazo robotov vseh proizvajalcev. Na eni strani je tukaj Siemens s programom PLM Automation [29], čigar del je namenjen tudi programiranju industrijskih robotov. Drugi takšen program pa je RoboDK [30], namenski program za programiranje industrijskih robotov različnih proizvajalcev. Ti programi so namenjeni predvsem podjetjem, ki se ukvarjajo z integracijo robotov različnih proizvajalcev. Vsak tak program je plačljiv in po navadi imajo podjetja letne licence za vsakega proizvajalca posebej. Tukaj gre za ekonomsko upravičenost nakupa takšnega programa.



Slika 3.17: Primer proizvodne celice s tremi industrijskimi roboti, narejene v programu RobotStudio.
Vir: lasten.

3.8 Vprašanja za utrjevanje snovi

1. Na spletu poiščite specifikacije industrijskega robota ABB IRB 7600 325/3.10. S pomočjo programa RobotLoad od podjetja ABB preverite, ali nosilnost robota zadostuje za naslednje orodje:

Masa:	275 kg
Težišče:	x = 0 mm
	y = 13 mm
	z = 385 mm
Vztrajnostni momenti:	Ix = 141 kgm2
	Iy = 273 kgm2
	Iz = 310 kgm2
TCP (Tool center point):	x = 0 mm
	y = –120,692 mm
	z = 1777,185 mm

Zapišite svoje ugotovitve:

2. Kaj se zgodi z ustreznostjo izbranega robota, če vemo, da bo robot prekladal izdelke z maso 30 kg? Pri tem je prijemališče izdelka na koncu orodja. Da lahko preverite še to stanje, je potrebno danim specifikacijam v programu RobotLoad dodati še eno obremenitev, in sicer pod imenom »Payload«. S tem se specifikacija orodja spremeni. Kaj ugotovite? Kako je sedaj z ustrezno nosilnostjo izbranega robota? Kaj predlagate za rešitev?

4 Priprava in zagon novega industrijskega robota

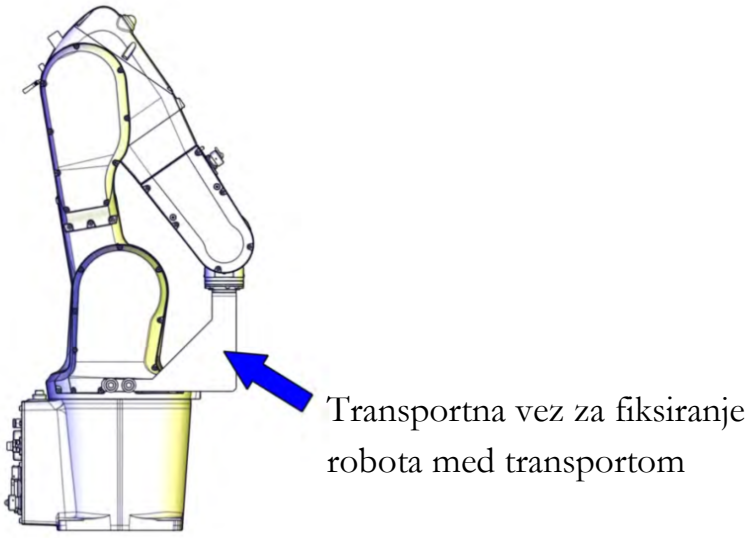
S točno določenimi podatki o uporabljenem orodju, o teži izdelka, ki ga bomo premikali in o zahtevani dosegljivosti lahko natanko določimo tip robota, ki ga potrebujemo. Prav tako moramo pomisliti na vse razširitvene module oz. kartice, ki jih bomo potrebovali. Pri tem imamo v mislih predvsem na število DI/DO, ali bomo potrebovali tudi kakšne analogno-digitalne pretvornike oz. ADC ali pa DAC. Z drugimi besedami, ali bomo potrebovali tudi kakšne analogne razširitvene kartice, kako bomo komunicirali z nadzornim sistemom in kakšen protokol bomo izbrali. Ali bomo potrebovali kakšne dodatne varnostne funkcije v smislu preprečevanja kolizije, omejevanja gibanja robota, da mu predpišemo dovoljeno območje gibanja? Vse to je smiselno vnaprej predvideti, da se lahko te funkcije integrirajo že pri proizvajalcu ter da se že vnaprej kalkulirajo vsi predvideni stroški. Seveda so nadgradnje možne tudi pozneje, a pri teh nadgradnjah po navadi prihaja do zamud, ki pa jih med načrtovanjem nismo predvidevali.

4.1 Transport in pritrjevanje industrijskega robota

Po nakupu industrijskega robota je potrebno robota pritrditi na bodisi fiksno podlago, kot so tla, najpogosteje se ga pritrdi na podstavek ali pa tudi na mobilno podlago. Pri tem je potrebno robota transportirati na to predvideno mesto. Postopek transporta mora biti izpeljan po navodilih proizvajalca, saj ima večina industrijskih robotov vgrajene servomotorje, ki imajo inkrementalne dajalnike položaja. Ti niso absolutni, temveč so

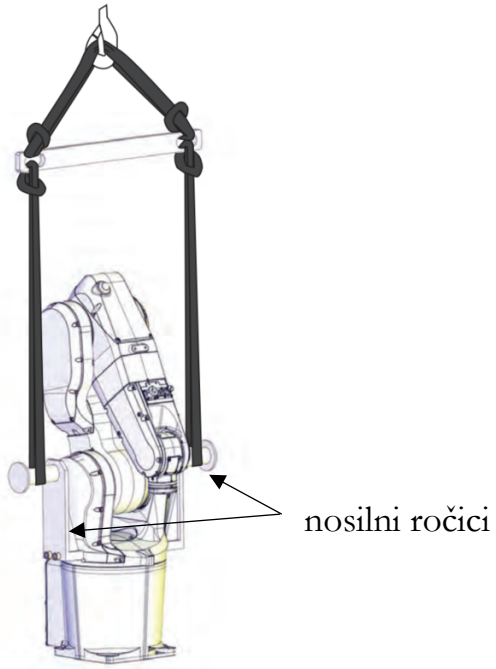
relativni in kot taki so občutljivi na udarce. Po vsakem transportu industrijskega robota je le-tega smiselno na novo kalibrirati. S tem se izognemo morebitnim posledicam preskoka inkrementalnega dajalnika, kar se v fazi prvotnega programiranja ne bi poznalo. Problem bi nastal, ko bi po končanem programu ponovno kalibrirali robota. Takrat bi se lahko zgodilo, da program ne bi več pravilno deloval.

Vsak industrijski robot ima svojo transportno pozicijo. To je pozicija, kjer je robot v stabilnem stanju in se brez zunanje sile ne more prevrniti. Slika 4.1 prikazuje transportno pozicijo robota IRB 1200. V tej poziciji naročnik prejme robota. Potem je potrebno robota transportirati na njegovo predvideno mesto.



Slika 4.1: Transportna pozicija robota IRB 1200 5 kg/0.9 m [9]

Če transportiramo majhne robote, to so roboti, ki imajo maso do 50 kg, transport ni pretirano zahteven, saj lahko robota dvigneta dva delavca in ga postavita na predvideno mesto. Pri večjih in težjih robotih pa je pravilen transport toliko bolj pomemben. Slika 4.2 prikazuje pravilen transport robota IRB 1200. Na vsako stran robota je potrebno namestiti namenski ročici, ki sta ustrezno dimenzionirani. Ti ročici je potrebno naročiti pri proizvajalcu. S pomočjo teh ročic lahko potem ustrezno transportiramo robota na njegovo predvideno mesto.

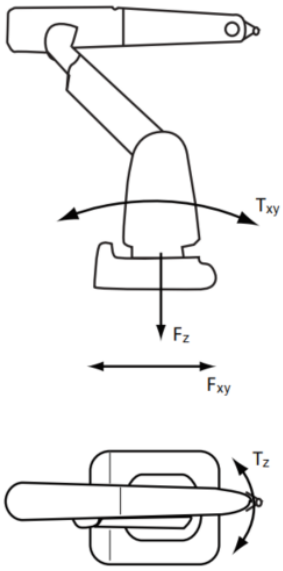


Slika 4.2: Pravilen transport industrijskega robota IRB 1200 5 kg/0.9 m [9]

4.1.1 Pritrjevanje industrijskega robota na podlago

Preden lahko robota transportiramo na njegovo ustrezno mesto, je potrebno zagotoviti, da je to pripravljeno v skladu z navodili proizvajalca. Ker je robot stroj z odprto zančno kinematično strukturo, se na stroju pojavljajo določene obremenitve, še posebej, ko robot opravlja gibe pri polni obremenitvi in polni hitrosti. Hitrost ni toliko problematična, kot so pospeški in pojemki. Ti ustvarjajo največji navor in sile na robotu. Prav tako moramo ločiti generirane sile na podlago, ko robot deluje v avtomatskem režimu, ter primere, ko deluje v avtomatskem režimu z največjo obremenitvijo in z največjo hitrostjo ter nekdo pritisne tipko za izklop v sili. Takrat se mora robot zaustaviti v najkrajšem možnem času. Slika 4.3 prikazuje zunanje sile in navore, ki delujejo na podlago industrijskega robota. Tabela 4.1 podaja vrednosti sil in navorov, ki jih generira robot IRB 1200. Z zagotavljanjem pravilnega pritrjevanja na podlago proizvajalec jamči za ustrezno delovanje industrijskega robota. Če robota ne pritrjujemo direktno na tla, temveč na podlago, lahko od proizvajalca kupimo namenske podlage, namesto da jih dimenzioniramo sami. S tem se izognemo marsikateri nevšečnosti.

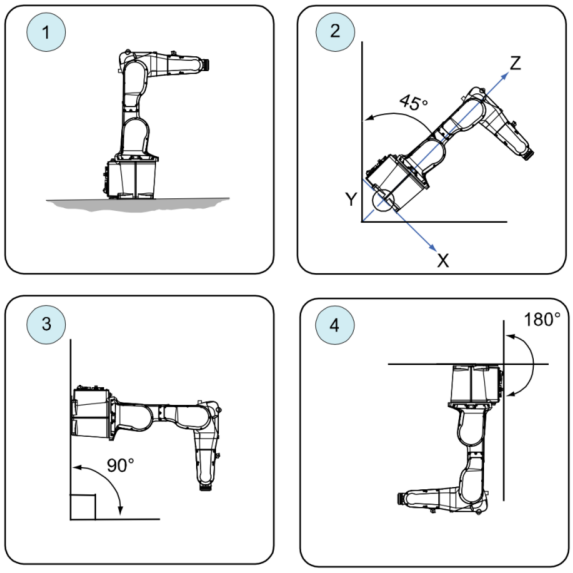
Posebej je potrebno poudariti, da je nujno potrebno prebrati navodila ali na **novo definirati smer delovanja gravitacije v krmilniku**, če robota montiramo na zid ali na strop. S tem se zagotovi pravilno delovanje krmilnika tudi, če je robot pričvrščen na stropu. Slika 4.4 prikazuje možne montažne položaje.



Slika 4.3: Prikaz delovanja zunanjih sil/navorov na industrijskega robota, ki je pritrjen na tla ali podlago in ne visi s stropa [9].

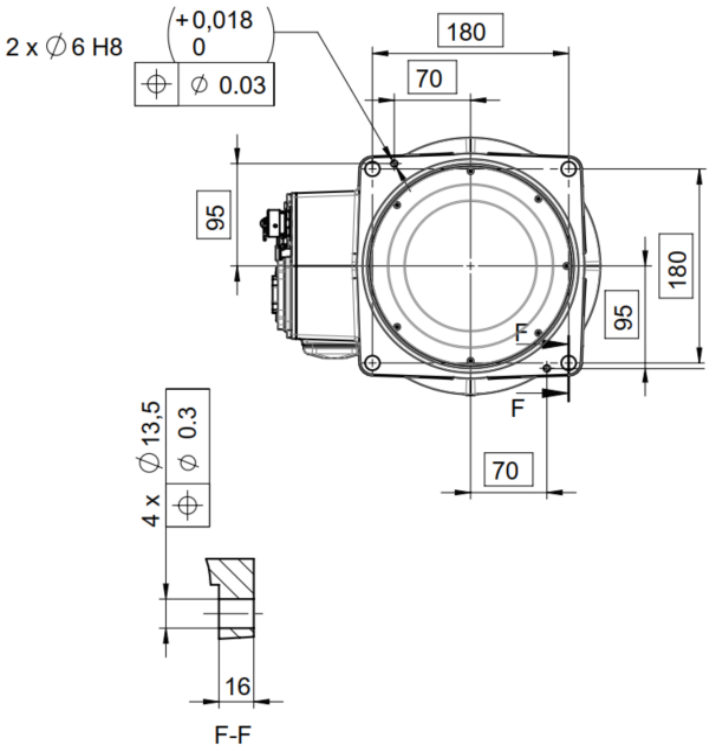
Tabela 4.1: Opis sil in navorov, ki se pojavijo pri normalnem delovanju in izklopu v sili pri robotu IRB 1200 [9].

Sila/Navor	Opis	Obremenitev med normalnim delovanjem	Max. obremenitev (izklop v sili)
Sila F_{xy}	Sila v katerikoli smeri ravnine XY	± 910 N	± 1620 N
Sila F_z	Sila v smeri Z	-550 ± 980 N	-550 ± 1610 N
Navor T_{xy}	Navor okoli osi X ali Y	± 570 Nm	± 1550 Nm
Navor T_z	Navor okoli osi Z	± 280 Nm	± 580 Nm



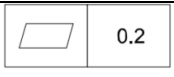
Slika 4.4: Pritrjevanje robota IRB 1200: 1) tla (angl. Floor), 2) poševno (angl. Tilted), 3) stena (angl. Wall), 4) strop (angl. Suspended) [9]

Spodnja prirobnica robotske osnove je izdelana na način, da omogoča natančno pritrjevanje robota na podlago. Slika 4.5 prikazuje načrt spodnje prirobnice. Prav tako spodnja prirobnica vsebuje dve izvrtini za zatiče, s pomočjo katerih lahko robota natančno umestimo.



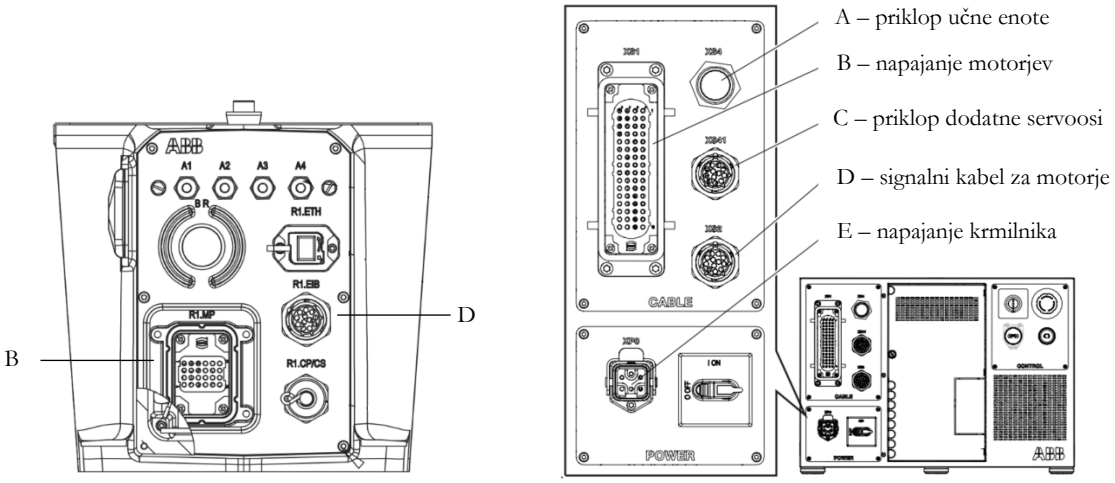
Slika 4.5: Načrt spodnje prirobnice osnove robota IRB 1200 [9]

Tabela 4.2: Zahteve za vijake, vodilno os in ravnost nosilne površine [9]

Primerni vijaki	**M12 x 35 (če se robot direktno pritrdi na nosilno podlago in ne na tla)**
Količina	**4 x**
Kvaliteta vijakov	**8.8**
Primerne podložke	**13 x 20 x 2, trdote 300 HV**
Izvrtini za zatiče	**2 x, D6 x 20, ISO 2338 – 6 m 6 x 20 – A1**
Zategniti z navornim ključem	**55 Nm ± 5 Nm**
Zahteva za ravnost nosilne površine	 **Tolerirana površina mora ležati med vzporednima ravninama, ki sta vzporedni z idealno ravnino, njun medsebojni razmik je 0.02 mm.**

4.2 Priklop napajalne napetosti in povezava robota s krmilnikom

Po uspešni namestitvi industrijskega robota na določeno mesto je potrebno zagotoviti primerno napajanje krmilnika in priklopiti napajalni in signalni kabel od krmilnika do robota ter priklopiti učno enoto. Pri tem se je vedno potrebno držati navodil proizvajalca, saj lahko ob neupoštevanju le-teh pride do nepopravljive škode. Slika 4.6 prikazuje priklop kablov na krmilnik IRC5 Compact in robota IRB 1200. Pri prvem priklopu so glavni trije kabli. Napajalni kabel za krmilnik (E), saj s tem napajamo tudi robota. Napajalni kabel za motorje robota (B), ki teče iz krmilnika na priključno ploščo robota, ter signalni kabel (D), ki skrbi za podajanje pozicije servomotorjev. Prav tako je potrebno priklopiti učno enoto oz. »teach pendant« (A) na krmilnik in s pomočjo te enote lahko nastavljamo in programiramo robota.

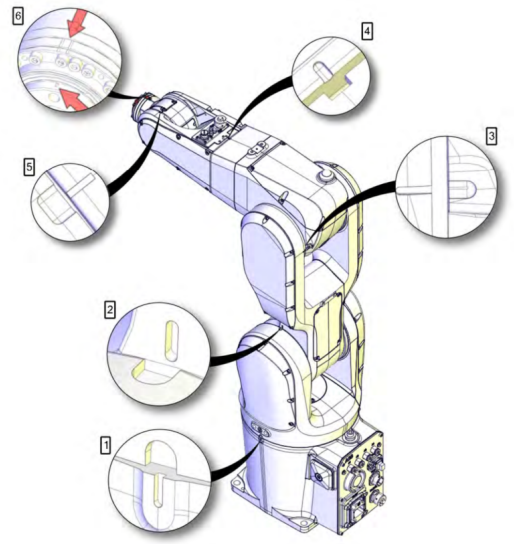


Slika 4.6: Priklop kablov na krmilnik IRC5 Compact in robota IRB 1200 [9]

4.3 Zagon, kalibracija industrijskega robota in priklop varnostnih funkcij

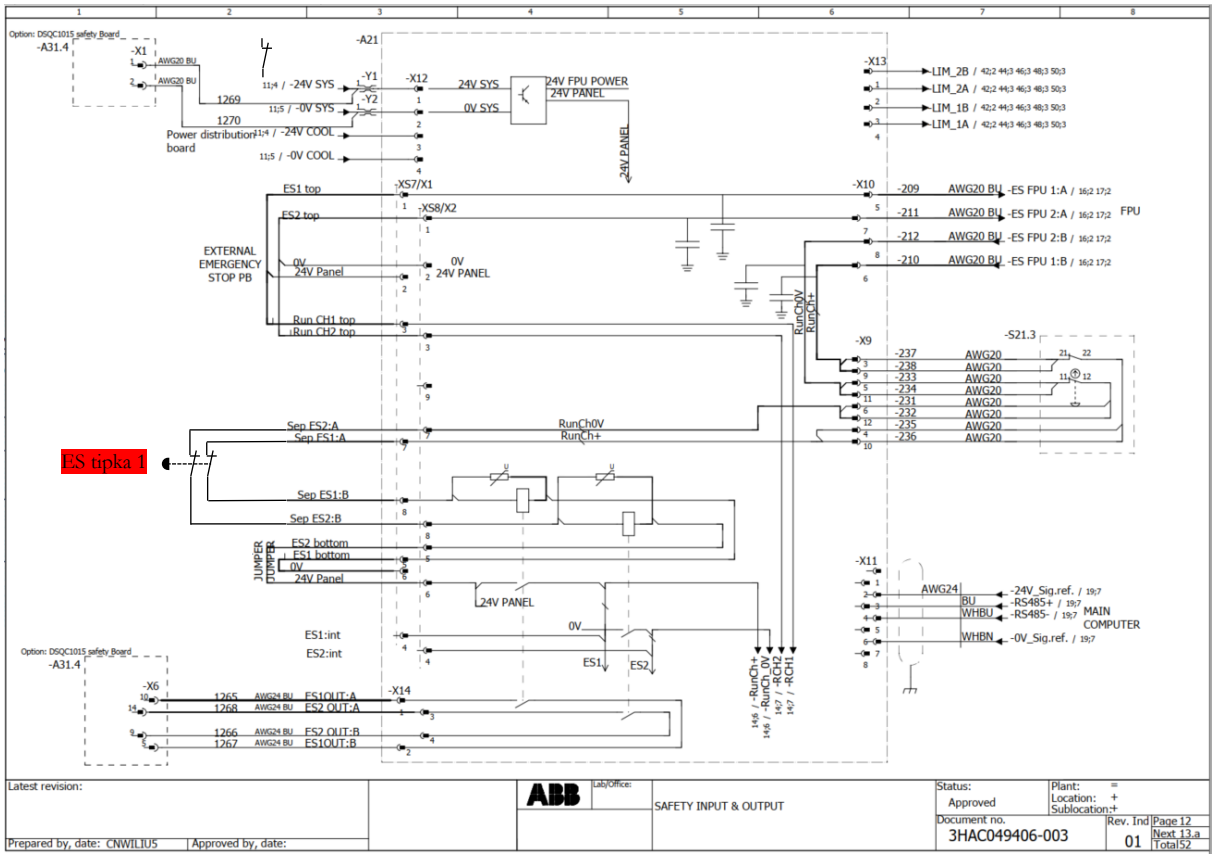
Po uspešnem priklopu napajalne napetosti in kablov med krmilnikom in robotom lahko zaženemo krmilnik. Prvi zagon industrijskega robota se razlikuje od posameznega proizvajalca in je priporočeno, da sledimo navodilom proizvajalca. Pri prvem zagonu industrijskega robota ABB IRB 1200 dobimo sporočilo, da števec motorjev ni posodobljen. To je standardni postopek prvega zagona robota podjetja ABB. Pri tem je potrebno postaviti robota v referenčno pozicijo in posodobiti števce. Tu ne gre za kalibracijo, temveč za posodobitev števcev, kjer se robotu definira ničti obrat. ABB tako ne zahteva kalibracije, ta je potrebna samo v primeru menjave posameznega motorja.

Prav tako ima robot IRB 1200 vse dodatne varnostne vhode, kot je tipka izklop v sili, premostičene oz. "brikane" (XS7, XS8), tako da lahko robota že takoj ob prvem zagonu premikamo. Naloga integratorja robota je naknadna vgradnja varnostnih funkcij.



Slika 4.7: Pozicije posamezne osi za sinhronizacijo in posodobitev števcev [9]

Za priklop dodatnih varnostnih funkcij, kot je tipka izklop v sili, je potrebno pregledati električno shemo krmilnika ter tipko priklopiti na ustrezno mesto konektorjev XS7 in XS8. Standardne varnostne funkcije so v industriji vedno dvokanalne (dodatna varnost), kar pomeni, da mora tudi tipka za izklop v sili imeti dve stikali NC 'Normally Closed'. Pri tem morata oba signala biti hkrati ali aktivna ali neaktivna. Če sta aktivna, to pomeni normalno delovanje, če sta neaktivna, pomeni, da je tipka pritisnjena in je tokokrog prekinjen. Če želimo na krmilnik IRC5 Compact priklopiti dodatno tipko za izklop v sili, je potrebno glede na elektro načrt uporabiti dvokanalno tipko, ki ima dve mirovni stikali, na konektor XS7 ali XS8. Pri konektorju XS7 sta to priključka XS7/X5 in XS7/X6 ter XS7/X7 in XS7/X8. Pri tem je potrebno žičke za premostičenje odstraniti in na ta mesta ustrezno priklopiti tipko za izklop v sili.



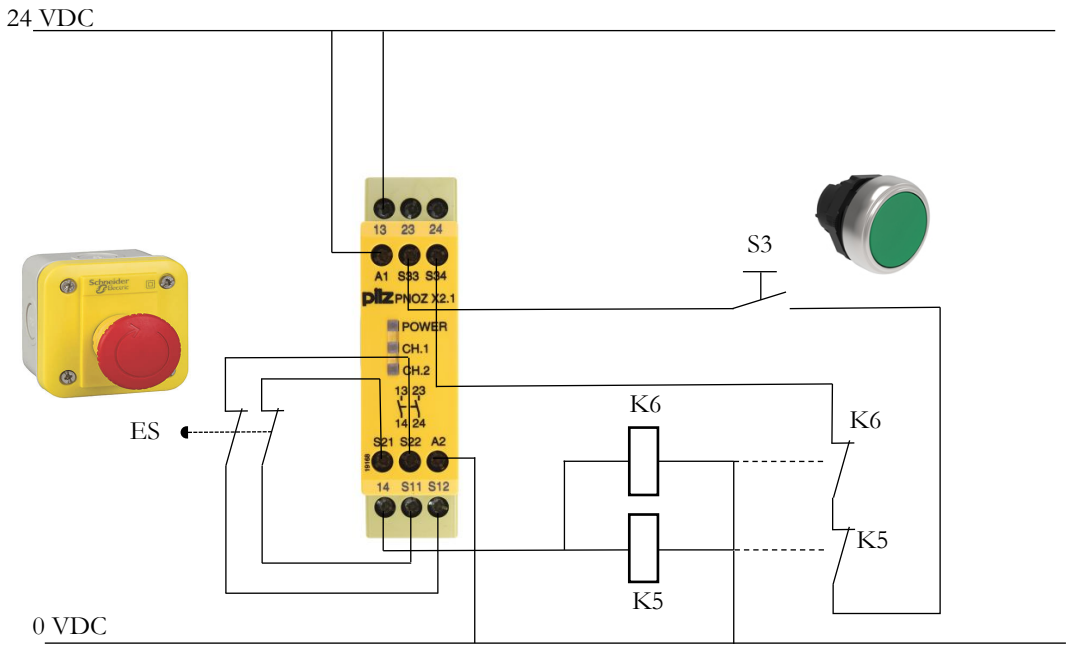
Slika 4.8: Elektro načrt vezave dodatnih tipk za izklop v sili [25]

Ker je v večini primerov industrijski robot integriran v robotsko celico, ki je del večjega proizvodnega procesa, se za varnost uporabljajo varnostni releji ali varnostni PLK-ji. V industriji so vse varnostne komponente označene z rumeno barvo.

Varnostni releji so posebni releji, namenjeni za nadzor varnostnih funkcij, kot so npr. tipka za izklop v sili, svetlobne zavese, varnostna vrata z zaklepom, varnostne blazine ter dvokanalni ročni vklopi. Slika 4.9 prikazuje primer uporabe PILZ varnostnega releja [31] za izklop v sili ter reset tipko za resetiranje releja po sprostitvi tipke za izklop v sili.

Varnostni PLK-ji so nadgradnja standardnih PLK-jev. Poleg navadnih funkcij PLK-ja imajo še dodatno varnostno komponento, preko katere nadzorujejo vse varnostne funkcije proizvodnega procesa, poleg tega pa nadzorujejo še pravilno izvajanje programa ter komunikacije. Če varnostni PLK zazna kakršno koli odstopanje PLK programa od dejanskega ali pa izpad komunikacije, se sproži alarm, ki zaustavi celoten proizvodni proces na varen način [32]. Varnostni PLK-ji prav tako potrebujejo posebne DI/DO kartice, ki so ločene od navadnih tako po priključkih kot tudi po rumeni barvi. Vsaka varnostna funkcija je tako določena z vhodom, klasificirana v bazi in takoj vidna na HMI

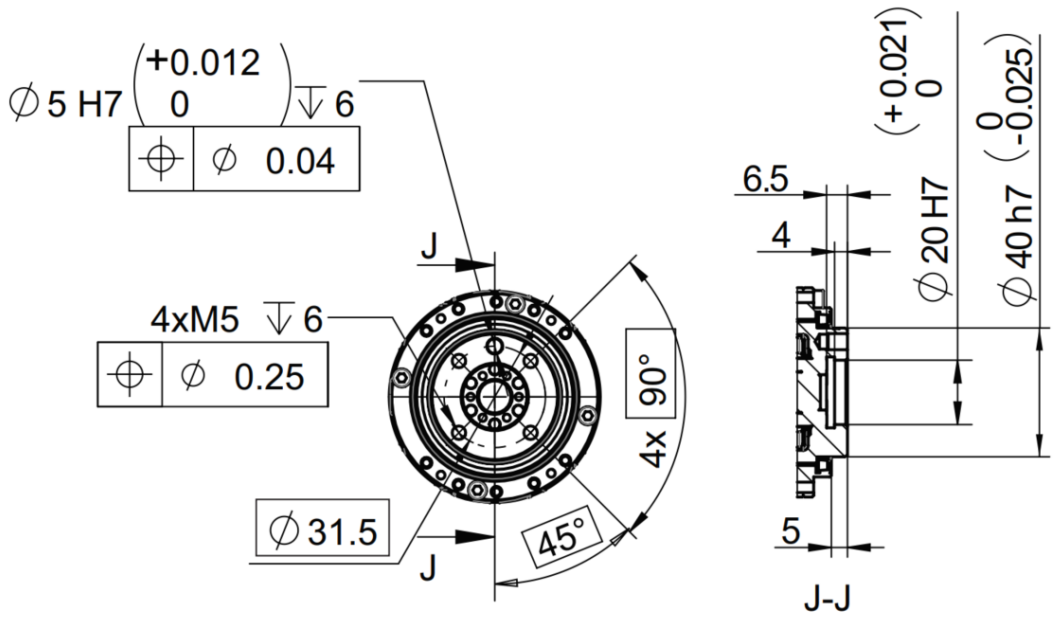
(Human Machine Interface) panelu. Prav tako program določi, kaj se zgodi, če se pojavi posamezna napaka ter kako PLK na to napako reagira.



Slika 4.9: Primer uporabe Pilz PNOZ X2.1 varnostnega releja za izklop v sili s potrditveno tipko za resetiranje [33–36]

4.4 Namestitev orodja na prirobnico robota

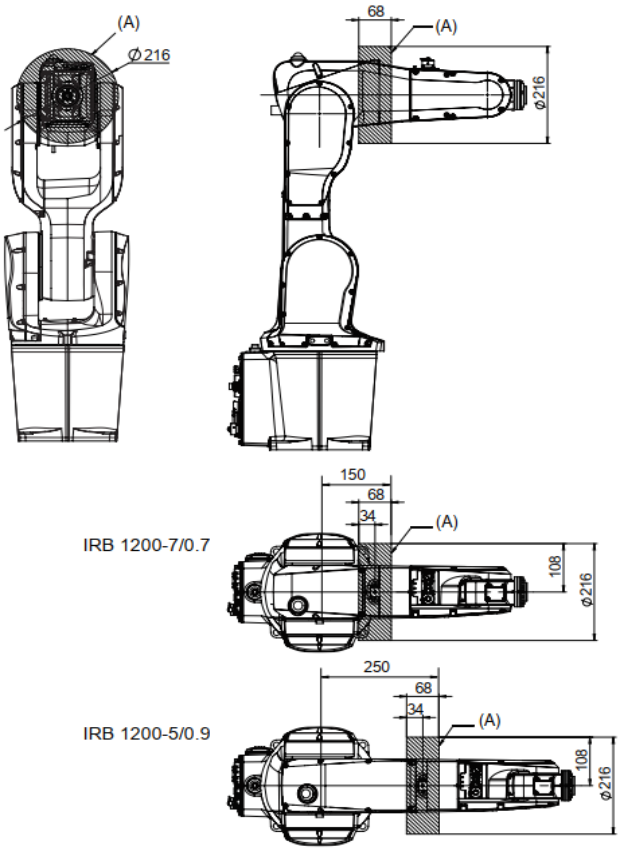
Preden lahko namestimo orodje na prirobnico robota, je potrebno izdelati ustrezno vmesno ploščo. Pri tem je potrebno pregledati navodila proizvajalca posameznega robota. Slika 4.10 prikazuje dimenzije in tolerance prirobnice na vrhu robota IRB 1200 5 kg/0.9 m, kjer imamo na voljo 4 navojne luknje M5 ter eno luknjo za zatič, s pomočjo katere lahko orodje centriramo na sredino. Prirobnice industrijskih robotov so standardizirane, tako da je zagotovljena zamenljivost orodij med posameznimi proizvajalci. Za to skrbi standard ISO 9409-1:2004 [37]. Pri dimenzioniranju vmesne plošče je potrebno paziti na globino naseda, saj ta ne sme znašati več kot 6.5 mm. V nasprotnem primeru bo vmesna plošča nasedla na ogrodje prirobnice, ki pa se ne vrti. S tem se bo motor 6. osi pregreval ali pa se sploh ne bo vrtel.



Slika 4.10: Dimenzije in tolerance prirobnice vrha robota IRB 1200 [9]

4.5 Namestitev dodatnih obremenitev na robota

Pogosto se v industriji dogaja, da je potrebno poleg orodja na robota namestiti še dodaten del opreme. Ta oprema se lahko namesti na za to določena mesta, zato je potrebno pogledati navodila proizvajalca za specifičen model robota. Tukaj bodo podane možnosti pritrditve dodatne opreme na konkretnem robotu IRB 1200 5 kg/0.9 m. Te so povzete iz navodil »IRB1200 Product Specification« [9]. Slika 4.11 prikazuje možnost pritrjevanja dodatne obremenitve na industrijskega robota IRB 1200. Kot je iz slike opazno, se lahko težišče dodatne obremenitve nahaja znotraj osenčenega območja v okolici pritrjevanja. Težišče se lahko nahaja znotraj območja navideznega valja z dimenzijami ϕ 216 mm x 68 mm. Dodatno obremenitev je potrebno definirati v krmilniku. Te obremenitve ni možno določiti s pomočjo procedure »LoadIdentity«, ki lahko določa ali maso in težišče orodja ali izdelka. Ta procedura je podrobno predstavljena v poglavju 6.1. Dodatno obremenitev najlažje definiramo kar preko programa RobotStudio.

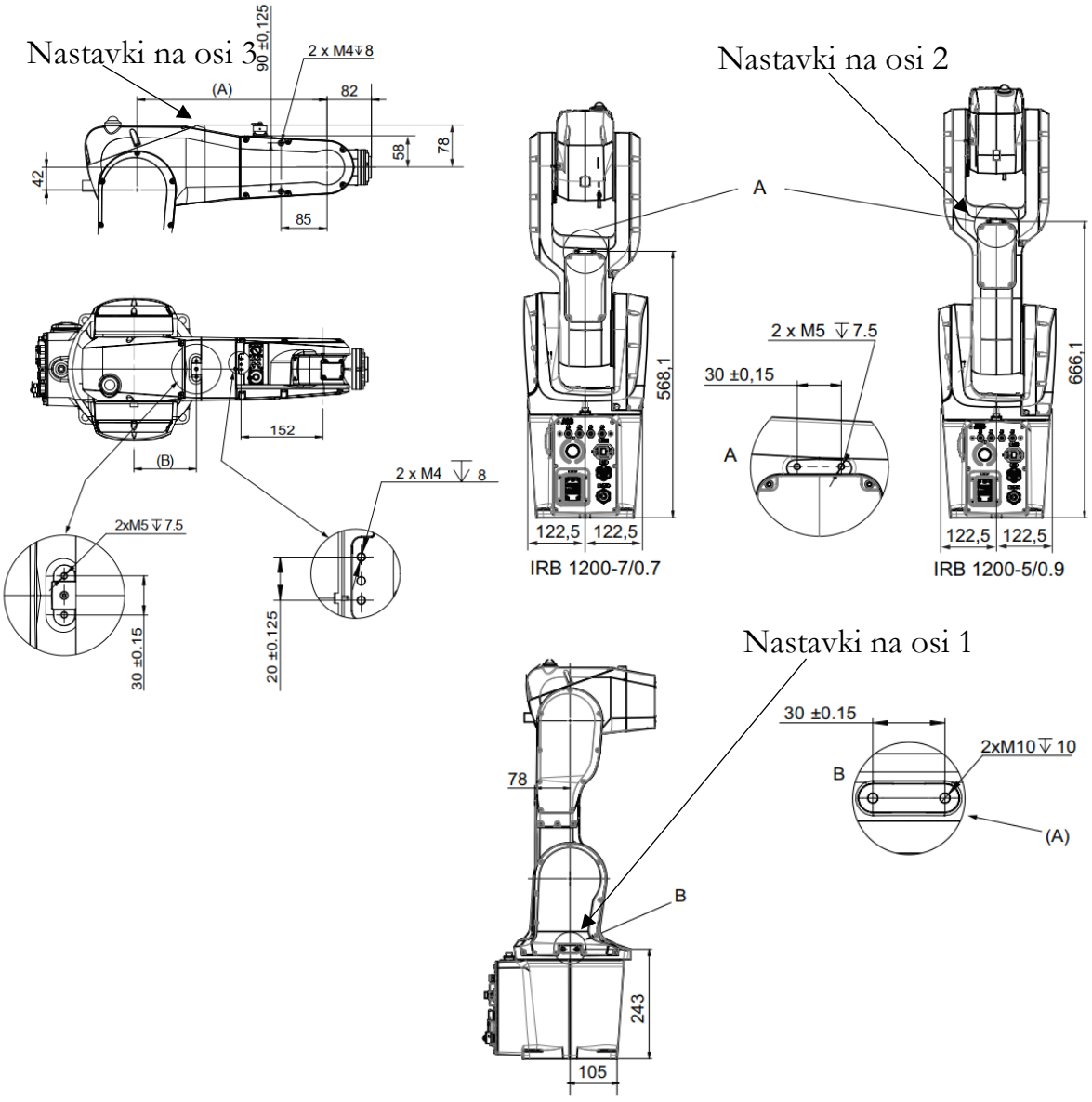


Slika 4.11: Možnost pritrjevanja dodatne obremenitve na industrijskega robota IRB1200 [9]

Tabela 4.3: Maksimalna dodatna obremenitev industrijskega robota IRB 1200 5 kg/0.9 m

Obremenitveno območje A	Maksimalna obremenitev
IRB 1200-5/0.9	0.3 kg

Robot IRB 1200 omogoča nameščanje tudi dodatnih priključkov, vezanih na uporabo orodja, kot so razni nastavki/nosila/vodila za kable, žice ali cevi, ki jih ne moremo uporabiti na obstoječih konektorjih. Pri tem je potrebno poudariti, da maso teh nastavkov ni potrebno definirati, saj je zanemarljivo majhna. Po navodilih proizvajalca težjih priključkov na drugo in pa prvo os ni dovoljeno pritrjevati.



Slika 4.12: Možnost pritrjevanja dodatnih priključkov opreme na tretjo, drugo in prvo os [9]

4.6 Vprašanja za utrjevanje snovi

1. Na kaj moramo biti pozorni pri pripravljanju podlage za pritrjevanje robota?

2. Na kaj moramo biti pozorni pri transportu robota do mesta za pritrjevanje na podlago?

3. Na kaj mora biti pozoren robotski programer, ko začne s programiranjem industrijskega robota? Kako je pri tem poskrbljeno za varnost?

4. Zakaj v industriji uporabljajo varnostne releje in varnostne PLK-je? Kaj je njihova bistvena naloga/prednost?

5. Kam na industrijskem robotu lahko pritrdimo dodatno opremo in kaj to pomeni za krmilnik?

ZAPISKI

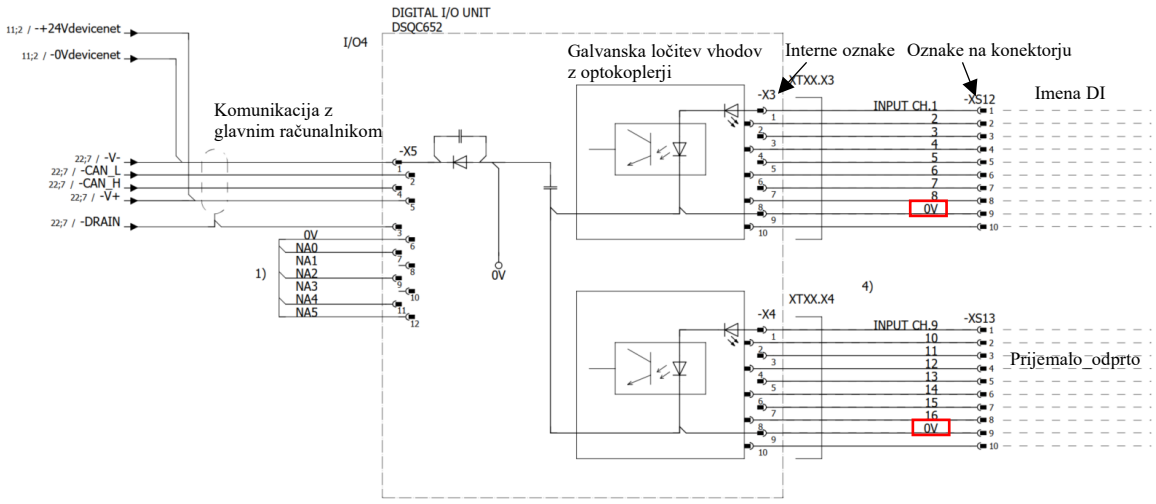
5 Konfiguracija DI/DO-signalov

Po postavitvi robota, priklopu varnosti in namestitvi orodja je potrebno konfigurirati uporabo DI/DO-signalov. Ti signali niso popolnoma integrirani. Integriranost DI/DO-kartice je zopet odvisna od posameznega proizvajalca. Podjetje ABB ponuja pri krmilniku IRC5 Compact integriranih 16 DI/DO-signalov, ki z glavnim računalnikom komunicirajo preko DeviceNET protokola [19]. Signale je za uporabo potrebno še konfigurirati. Če potrebujemo več DI/DO-signalov, je to potrebno specificirati že pri nabavi robota. To se da sicer urediti tudi naknadno z integracijo dodatne vhodno/izhodne kartice poljubnega proizvajalca, ki mora imeti ustrezen komunikacijski protokol.

Pri podjetju KUKA nimajo integriranih lastnih kartic za DI/DO-signale, temveč je potrebno specificirati, s pomočjo katere serijske komunikacije robotski krmilnik dostopa do zunanje kartice. Krmilnik KUKA komunicira z zunanjo kartico najpogosteje preko PROFINET komunikacije ter posredno dostopa do signalov. Prednost tega je, da nismo omejeni s strani proizvajalca robota ter imamo dovolj prostora za nadgradnjo, če imamo navadni krmilnik KRC4. Pri tem lahko uporabimo razširitveno kartico različnega proizvajalca.

5.1 Konfiguracija DI/DO-signalov na krmilniku ABB IRC5 Compact

Krmilnik IRC5 Compact ima integriranih 16 DI/DO-signalov, ki so na voljo na kartici DSQC652. Ti konektorji so na krmilniku označeni z oznakami XS12 in XS13 kot digitalni vhodi ter XS14 in XS15 kot digitalni izhodi. XS16 je oznaka za napajalni konektor. Posamezni vhodi so označeni z XS12.1. Te vhode je potrebno definirati, jih ustrezno oštevilčiti in poimenovati. To lahko storimo tako na robotu preko učne konzole kot tudi preko programa RobotStudio. Slika 5.1 prikazuje električno shemo digitalnih vhodov. Pri tem ne smemo pozabiti, da je nujno potrebno na vhod XS12.9 in XS13.9 priključiti 0 V DC. Brez tega na vhodih ni razlike v napetostih in kartica ne bo zaznala digitalnega vhoda. Pri digitalnih izhodih pa poleg 0 V DC potrebujemo še napajalno napetost 24 V DC. Če uporabljamo več različnih kartic različnih proizvajalcev, moramo vse priklope za 0 V DC povezati skupaj. S tem so vse kartice na istem potencialu električne napetosti.

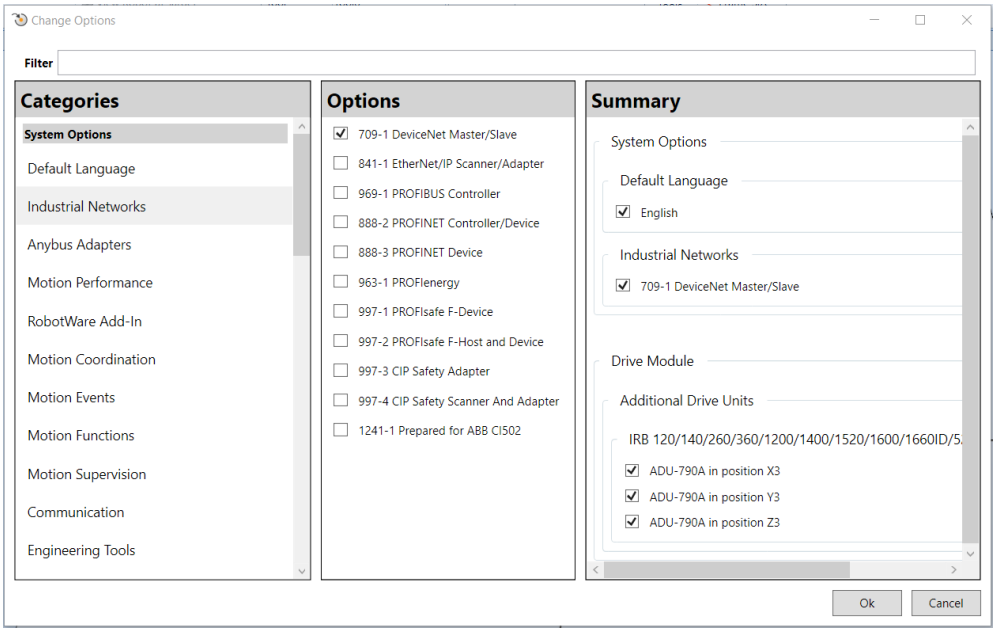


Slika 5.1: Električna shema DI-konektorjev [25]

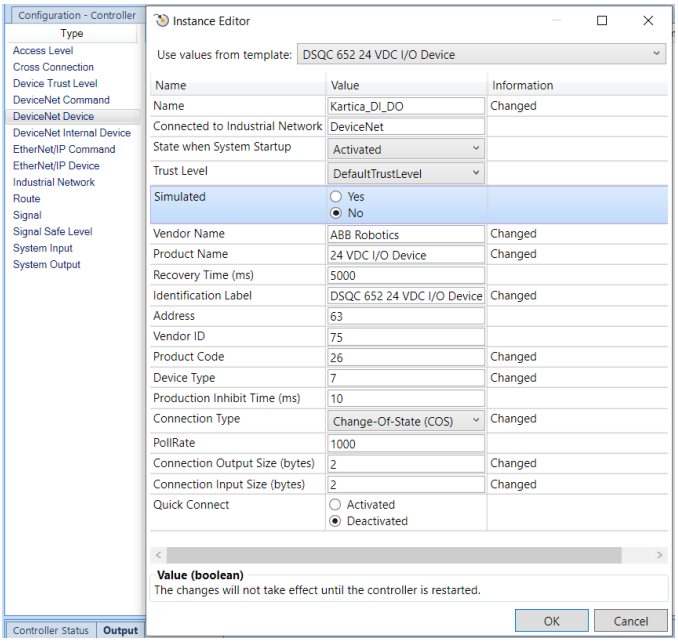
5.1.1 Konfiguracija DI/DO s pomočjo programa RobotStudio

Pri konfiguraciji DI/DO s pomočjo programa RobotStudio je priporočljivo, da ima uporabnik že nekaj izkušenj s programom. V nasprotnem primeru bo uporabnik imel kar nekaj težav pri iskanju določenih ukazov. Najprej je potrebno izbrati željenega robota in verzijo krmilnika. Nato je potrebno dodati opcijo 709-1 DeviceNet Master/Slave, ki je potrebna za konfiguracijo DI/DO-signalov. Slika 5.2 prikazuje dodatno opcijo. Če želimo dostopati do DI/DO-signalov, je najprej potrebno krmilniku dodati kartico DSQC652. Pod zavihkom »Controller« je potrebno izbrati »Configurations« in potem »I/O systems«. Najprej dodamo manjkajočo kartico k »DeviceNet Device«, kar prikazuje slika 5.3. Kartici

lahko poljubno poimenujemo, pri čemer pa ne smemo delati presledkov. Vseh drugih nastavitev pa ni potrebno spreminjati. Pri tem je potrebno poudariti, da je dodajanje opcije »DeviceNet« in dodatne kartice DSQC652 potrebno zgolj v programu RobotStudio zato, da si ustvarimo digitalno verzijo realnega krmilnika robota. Na realnem krmilniku so te opcije že naložene. Do njih dostopamo tako, da se preko programa RobotStudio povežemo na realni krmilnik in z istimi ukazi, ki so zapisani v nadaljevanju, kreiramo ali urejamo signale.



Slika 5.2: Dodajanje opcije DeviceNet Master/Slave v programu RobotStudio



Slika 5.3: Dodajanje manjkajoče kartice krmilniku v programu RobotStudio

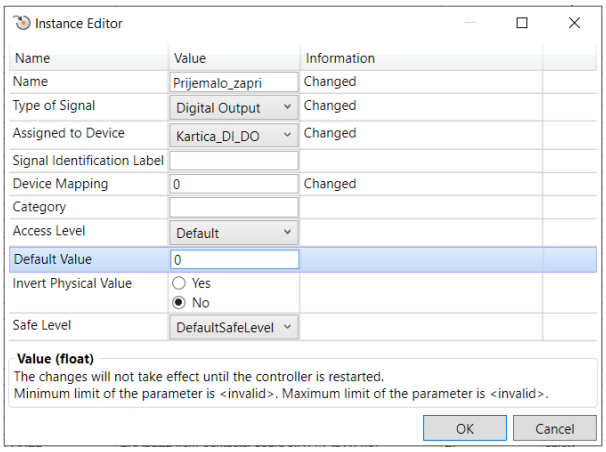
Z dodano kartico lahko dodamo posamezne DI/DO-signale. Ostanemo kar na istih nastavitvah in z desnim klikom na miško na »Signal« dodamo nov signal. Slika 5.4 prikazuje dodajanje novega signala. Signal je potrebno smiselno poimenovati, da bomo takoj vedeli, čemu je namenjen. Potrebno je definirati, ali je to digitalni vhod ali izhod, kateri kartici pripada (kartici, ki smo jo prej definirali) ter nastaviti privzeto vrednost. Prav tako je potrebno nastaviti tudi naslov, na katerem se ta signal na kartici nahaja. Pri tem je pomembno, da **ločimo fizične in programske oznake signalov**. V tem konkretnem primeru je fizični naslov na XS14.1, kar pa programsko pomeni, da je to prvi izhod iz kartice DSQC652. Programsko se vedno naslovi začnejo šteti z 0, tako da je ta naslov pri »Device Mapping« enak 0.

Če je potrebno signal zaščititi pred ročnim spreminjanjem signala, potem je potrebno spremeniti »Acces Level« na »ReadOnly«.

»Device Trust Level« pogojuje varnost, če pride od izklopa kartice z DI/DO. Privzeta vrednost javi napako, če se kartica izklopi. Kadar je nastavljena vrednost na »SafetyTrustLevel«, ob izklopu kartice ne dobimo nobenega sporočila.

Če pogledamo primer določevanja digitalnega vhoda na XS13.4, moramo za ta signal določiti digitalni vhod, ime in programski naslov »Device Mapping«, ki je 11. Podobno potem nastavimo še ostale signale, ki jih potrebujemo pri naši aplikaciji. S temi signali po navadi krmilimo orodje, ki je pritrjeno na robotu. Ko imamo vse potrebne signale nastavljene, je potrebno virtualni krmilnik resetirati, saj smo spreminjali sistemske nastavitve. Ti signali so potem na voljo pri pripravi robotskega programa v RAPID-u.

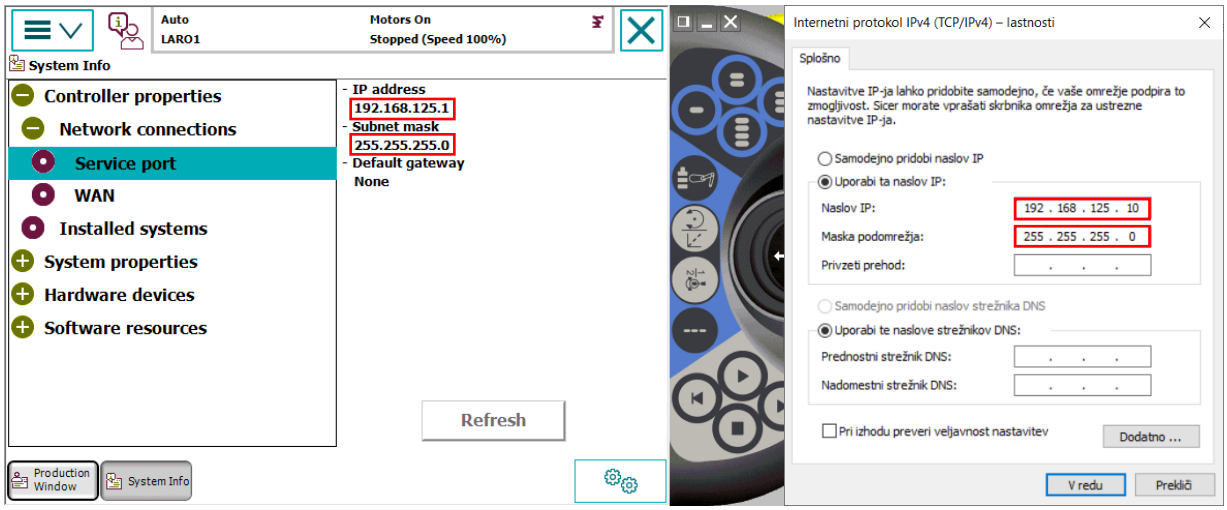
.



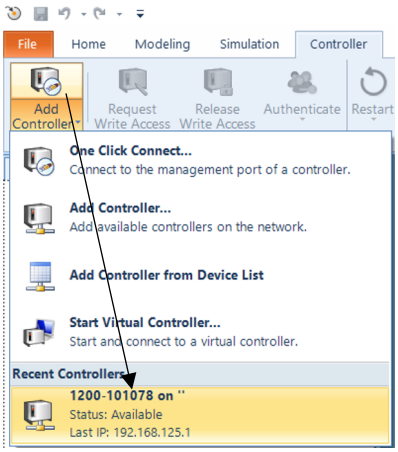
Slika 5.4: Dodajanje novega signala v programu RobotStudio

5.1.2 Konfiguracija DI/DO na realnem krmilniku s pomočjo programa RobotStudio

Če želimo dodajati oz. urejati signale s pomočjo programa RobotStudio na realnem krmilniku, moramo vzpostaviti komunikacijo z realnim krmilnikom preko omrežnega kabla. Najprej moramo preveriti, kakšen IP-naslov ima krmilnik. To storimo preko učne konzole z ukazom »System Info«, »Controller properties«, »Network connections« in »Service port«. Ugotovimo, da ima krmilnik naslednji IP: 192.168.125.1 ter podmasko omrežja 255.255.255.0. Na drugi strani moramo nastaviti IP-naslov na omrežni kartici v računalniku, ki bo imel prva tri števila enaka, zadnje pa različno. To si lahko predstavljamo takole: Ulica vsebuje več hiš, vsaka hiša pa ima unikatno številko. Tako točno vemo, za katero hišo gre, če bi imeli dve hiši isto številko, bi prihajalo do konflikta pri dostavi pošte. Slika 5.5 prikazuje preverjanje IP-naslova na krmilniku in nastavljanje IP-naslova na omrežni kartici računalnika. Za dostop do realnega krmilnika je le-tega potrebno dodati v programu RobotStudio. To storimo s pomočjo ukaza »Add Controller«, ki se nahaja pod zavihkom »Controller«. Če se je komunikacija že vzpostavila, je realen krmilnik avtomatsko viden, v nasprotnem pa kliknemo na ukaz »One Click Connect«. Po uspešno opravljenih spremembah je potrebno krmilnik robota resetirati.



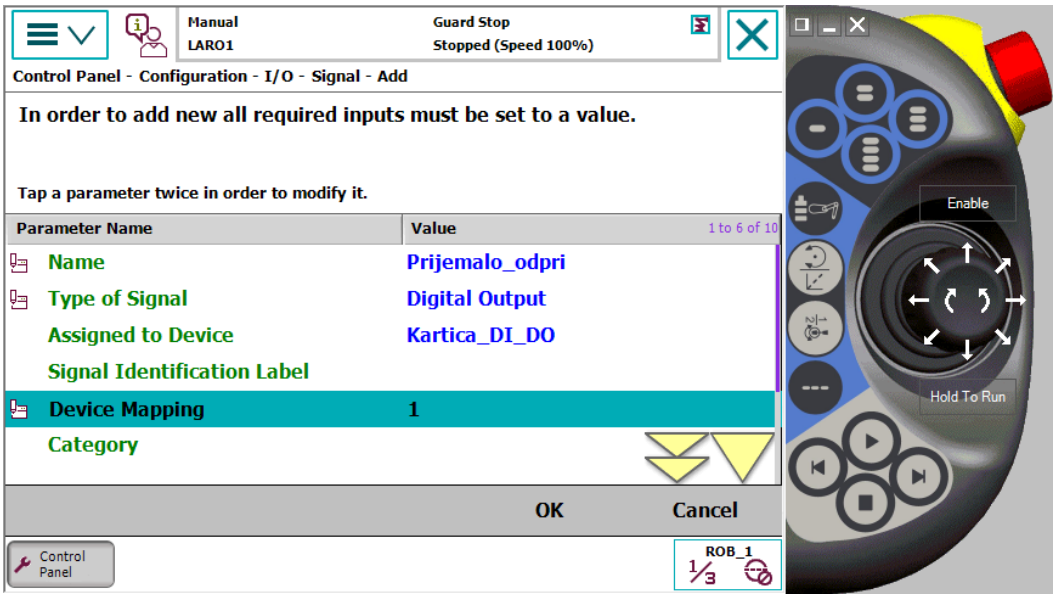
Slika 5.5: Preverjanje IP-naslova na krmilniku in nastavljanje IP-naslova na omrežni kartici računalnika



Slika 5.6: Dodajanje realnega krmilnika

5.1.3 Konfiguracija DI/DO s pomočjo učne konzole

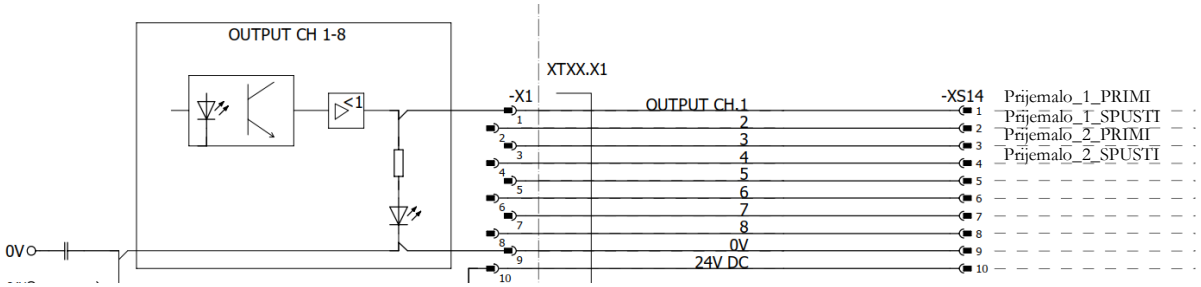
Če smo pri nabavi robota naročili kartico z DI/DO-signali, je ta že integrirana in pripravljena za uporabo. Signale, ki jih želimo uporabljati, je potrebno konfigurirati. Na učni konzoli izberemo nadzorno ploščo »Control Panel« in izberemo nastavitve »Configuration«. Preverimo, ali imamo kartico DSQC652 že integrirano in konfigurirano, ter izberemo ukaz »Signal«, kjer dodamo signale. Slika 5.7 prikazuje dodajanje novega signala preko krmilnika IRC5. Postopek je v bistvu enak postopku dodajanja ukaza preko programa RobotStudio. Po dodanih vseh novih ukazih je potrebno na krmilniku robota izvesti ponovni zagon, da se spremembe aktivirajo.



Slika 5.7: Definiranje signalov preko krmilnika IRC5

5.2 Vprašanja za utrjevanje snovi

1. V programu RobotStudio kreirajte nov robotski sistem, kjer izberete robota IRB 1200 5/0.9 z verzijo RobotWare-a 6.10. Še pred kreiranjem sistema krmilniku dodajte opcijo »709-1 DeviceNet Master/Slave«. Pod zavihkom »Controller« in ukazom »Configurations« dodajte kartico z 16 DI/DO z oznako DSQC652. Konfigurirajte uporabo naslednjih digitalnih izhodov. Ti izhodi bodo uporabljeni pri izvedbi vaj na realnem robotu. Zapišite fizične in programske naslove, ki ste jih pri tem uporabili.



ZAPISKI

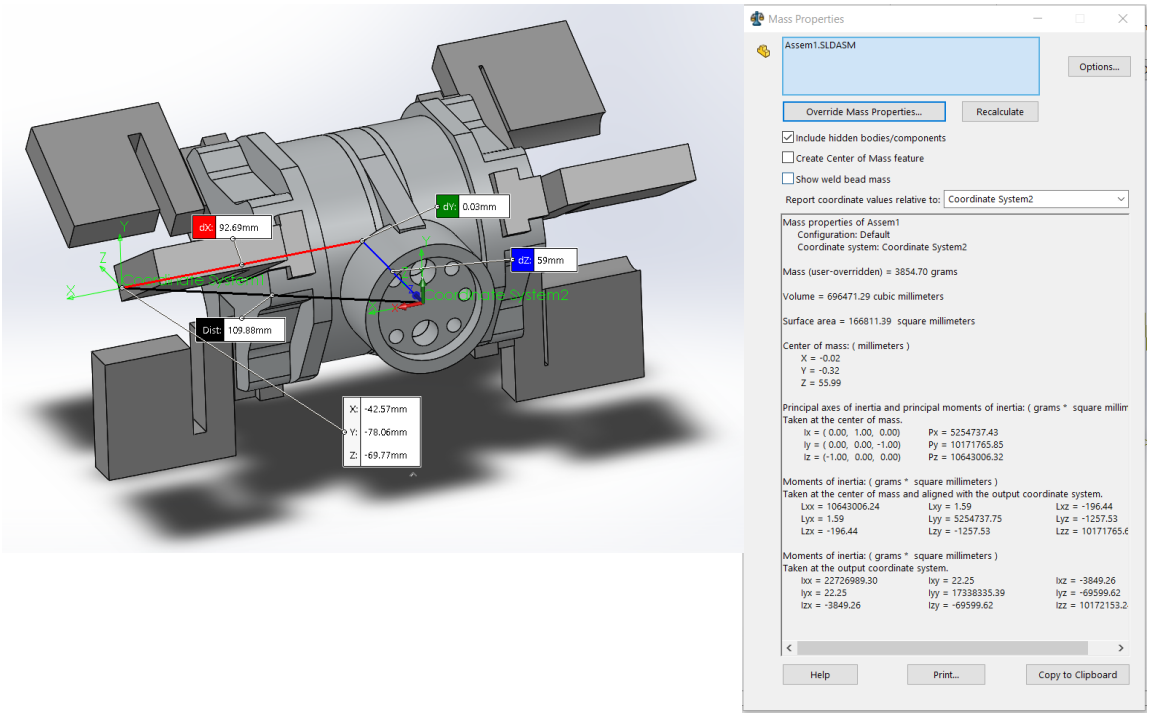
6 Konfiguracija orodja

Z uspešnim konfiguriranjem signalov, ki jih potrebujemo za upravljanje orodja na robotu, konfiguracija orodja še ni končana. Uporabljenemu orodju je potrebno definirati še TCP točko, definirati težišče orodja in vztrajnostne momente. Pravilno definirano orodje ima velik pomen pri izvajanju robotskega programa. Ko robotu predpišemo hitrost izvajanja nekega giba, ta hitrost velja za TCP-točko. Torej, če želimo uporabiti hitrost 100 mm/s pri nanašanju lepila, je ta hitrost definirana na koncu šobe in ne na prirobnici robota. V primeru napačno definiranega orodja bo tudi hitrost izvajanja giba napačna in nanos lepila ne bo ustrezen. Prav tako pravilno definirano težišče in vztrajnostni momenti orodja vplivajo na izvajanje giba, saj krmilnik vse te vrednosti upošteva pri kalkulaciji pospeškov, hitrosti in pojemkov. Natančnost definiranja težišča in vztrajnosti je pomembna pri vseh tipih in velikostih robotov. Majhna odstopanja niso problematična, tukaj imamo v mislih predvsem popolnoma zgrešene podatke o težišču in vztrajnosti.

6.1 Določevanje TCP-ja, mase, težišča in vztrajnosti orodja

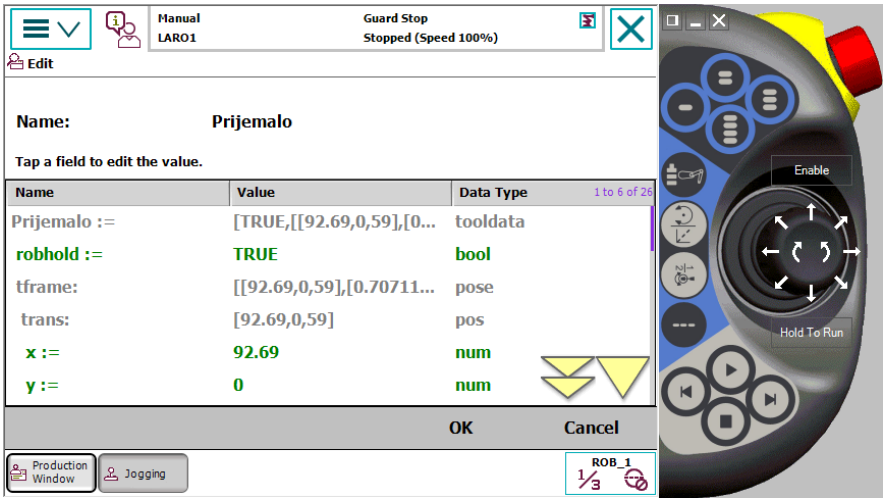
Če sami načrtujemo orodje, ki ga bomo namestili na vrh robota, potem lahko s pomočjo CAD-modela v modelirnem programu, kot je npr. SolidWorks [38], pridobimo vse podatke o orodju. Definiramo lahko tako TCP kot tudi težišče in masne vztrajnostne momente. Slika 6.1 prikazuje princip določevanja TCP-točke, mase, težišča in masne vztrajnostne momente v programu SolidWorks. Pri tem je potrebno poudariti, da je potrebno definirati nove koordinatne sisteme, enega v središče prirobnice, ki se pritrdi na robota, drugega pa v točko, kjer želimo imeti TCP. Po tem izmerimo razdaljo med njima,

kot tudi poiščemo težišče in masne vztrajnostne momente glede na koordinatni sistem prirobnice. Če želimo to orodje uporabiti tudi v programu RobotStudio, moramo model shraniti pod format .sat in pri shranjevanju izbrati koordinatni sistem prirobnice. S tem si zagotovimo nemoteno namestitev orodja na prirobnico robota v programu RobotStudio.



Slika 6.1: Določevanje TCP-točke, teže, težišča in masnih vztrajnostnih momentov orodja v programu SolidWorks

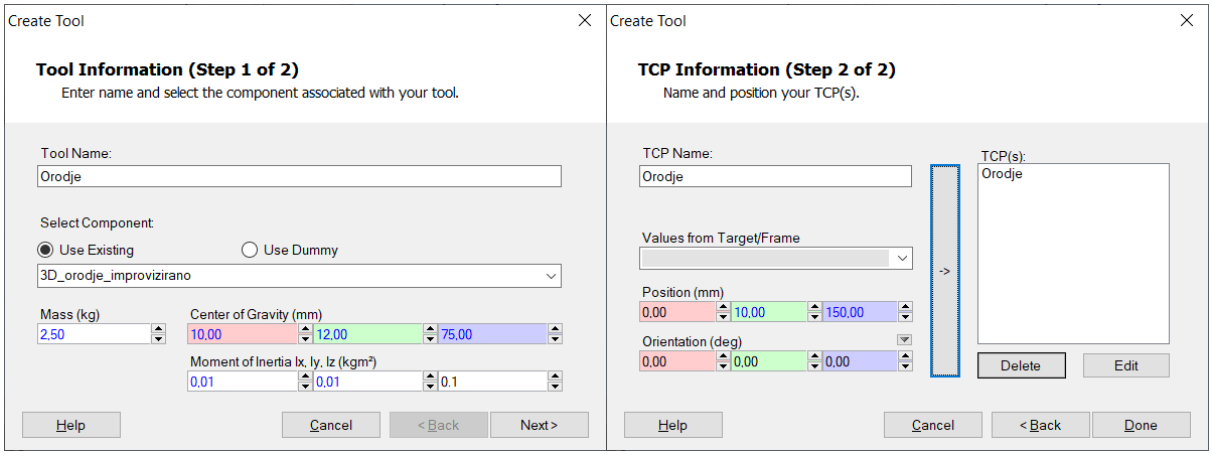
Pridobljene podatke lahko ročno vnesemo pri nastavitvi novega orodja preko učne konzole. Pri tem je potrebno podatke o TCP in težišču vnesti v mm, podatek o masi v kg, podatke o masnih vztrajnostnih momentih pa v kgm^2. Podatke o orientaciji je potrebno vnesti v zapisu kvaternionov [39]. To je samo drugačen zapis orientacije. Splošno orientacijo zapisujemo s pomočjo Eulerjevih kotov, v primeru uporabe ABB-robotov pa v obliki kvaternionov. Ta zapis pretvori kote v števila, s pomočjo katerih pri računanju orientacije ne operiramo s kotnimi funkcijami, temveč samo s preprostim množenjem. Takšen način računanja je hitrejši, a za vizualizacijo orientacije ne preveč posrečen. Pri računanju orientacije si tako pomagamo s pretvornikom iz Eulerjevih kotov v kvaternione [40]. Slika 6.2 prikazuje kreiranje novega orodja z ročnim vnosom podatkov.



Slika 6.2: Kreiranje novega orodja z ročnim vnosom podatkov o orodju

6.1.1 Kreiranje novega orodja v programu RobotStudio

Če imamo CAD-model, hitro pridemo do vseh podatkov o orodju. Kreiranje orodja v programu RobotStudio tako poteka zelo enostavno. Najprej je potrebno željen CAD-model uvoziti. Pri tem mora biti model v pravilnem formatu. Najbolj primeren format je ACIS, ki ima končnico .sat. Pod zavihkom »Modelling« izberemo ukaz »Create Tool«. Izberemo uvožen CAD-model, ki mu določimo maso, težišče in vztrajnostne momente ter dodamo položaj enega ali več TCP-jev.



Slika 6.3: Primer kreiranja novega orodja v programu RobotStudio

Po uspešnem kreiranju orodja je le-tega potrebno pritrditi na prirobnico robota. Z desnim klikom na kreirano orodje izberemo ukaz »AttachTo«, izberemo robota in na vprašanje, če želimo posodobiti položaj orodja, kliknemo »Da«.

6.1.2 Določevanje TCP točke orodju, kjer nimamo na voljo CAD-modela

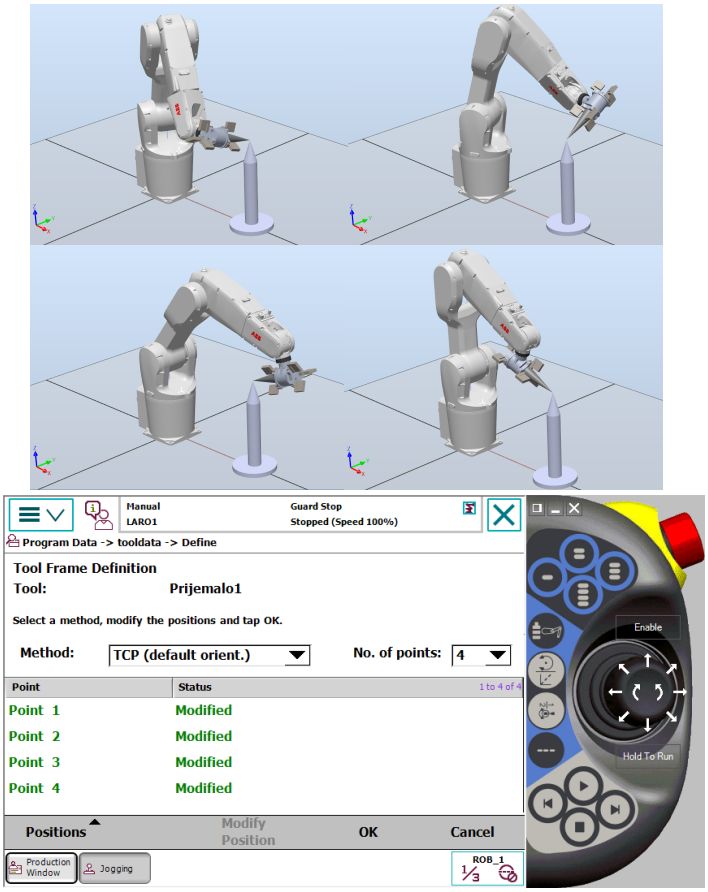
Pogosto se dogaja, da nimamo vseh podatkov o orodju. V tem primeru imamo na voljo tri metode določevanja TCP-točke željenega orodja s pomočjo realnega robota. Izbira metode je odvisna od postavitve TCP in njene orientacije. Tabela 6.1 opisuje različne metode. Najpogosteje se uporablja metoda TCP&Z, saj nam je poleg določitve pozicije najpomembnejša še določitev orientacije orodja v osi Z.

Tabela 6.1: Možne metode določevanja TCP-orodja

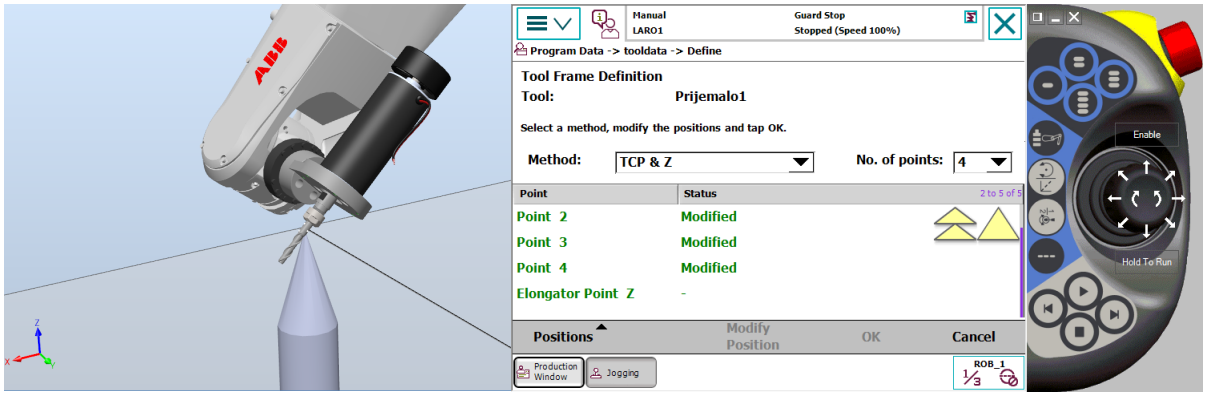
Kaj želimo določiti?	Metoda
Želimo imeti isto orientacijo orodja, kot ga ima prirobnica vrha robota, spremeniti želimo samo translacijo TCP-točke.	**TCP (default)**
Določiti želimo translacijo TCP-točke in novo orientacijo v osi Z.	**TCP&Z**
Določiti želimo translacijo TCP-točke in novo orientacijo v osi X in osi Z.	**TCP&Z,X**

Slika 6.4 prikazuje privzeto metodo TCP(default), pri čemer določimo samo pozicijo TCP, orientacija pa se prevzame po orientaciji prirobnice. Pri tej metodi je najprej potrebno določiti novo prijemalo, jo poimenovati, to okno zapreti, izbrati prijemalo, ki smo ga kreirali, ter izbrati ukaz »Edit«, kjer potem izberemo »Define«.

Slika 6.5 prikazuje določevanje TCP in orientacije po osi Z. Pri tem potrebujemo dodatno točko, ki jo določimo tako, da ko končamo četrto točko določevanja pozicije TCP, potem orientacije orodja ne spreminjamo več, temveč se samo zapeljemo vzdolž orodja v smeri, kjer želimo imeti os Z. Peta točko tako določa orientacijo Z novega orodja, pri čemer nas X in Y ne zanimata. Če želimo določiti še orientacijo, se moramo poslužiti metode TCP&Z,X, pri čemer je postopek enak prejšnjemu, s tem da določimo še dodatno točko, ki določuje smer osi X.



Slika 6.4: Postopek določevanja TCP pri standardni metodi TCP(default)



Slika 6.5: Določevanje TCP-točke in orientacije v smeri Z po metodi TCP&Z

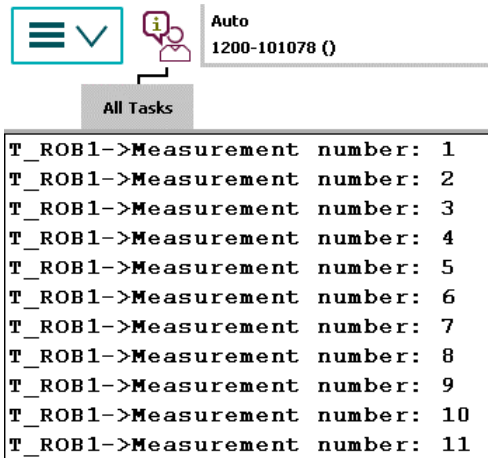
6.1.3 Določevanje mase, težišča in vztrajnostnih momentov, ko nimamo podatkov o orodju

Pri določevanju mase, težišča in vztrajnosti obstoječega orodja, kjer nimamo dostopa do CAD-modela, nam pomaga implementirana funkcija »LoadIdentify«. S pomočjo te funkcije, ki se izvede s pritrjenim orodjem na prirobnici 6. osi, se preračunajo vsi manjkajoči podatki. Te preračunane podatke krmilnik avtomatsko dopiše podatku o

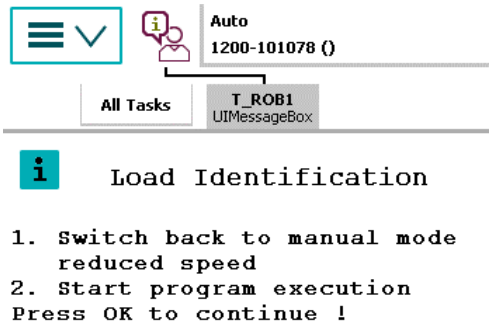
orodju. Funkcijo »LoadIdentify« najdemo pod »ProgramEditor«, kjer kliknemo na »Debug« in izberemo »CallRoutine«. Pri tem se odpre novo okno, kjer izberemo rutino oz. funkcijo »LoadIdentify«. Pri izvajanju funkcije je potrebno izbrati pravo ime orodja, če jih imamo več.

Ob zagonu funkcije mora biti robot v varni točki. Funkcija sproži pomikanje 3., 5. in 6. osi, pri čemer se 3. os premika za +/– 3 °, 5. os se pomika med +/– 30 ° in 6. os se pomika za +/– 90 °, pri čemer se uporabnik odloči glede smeri vrtenja 6. osi. To je potrebno smiselno izbrati, saj se lahko zgodi, da imamo na orodju cevi oz. žice, ki omejujejo gibanje orodja.

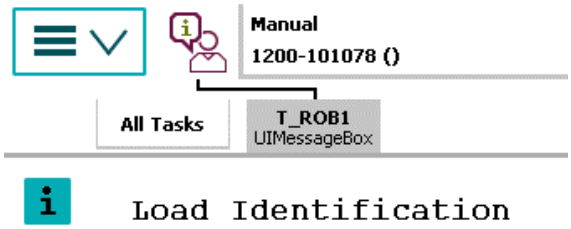
Pri tem je priporočljivo funkcijo najprej izvesti v ročnem režimu, da se preverijo vsi položaji, ko pa zagotovo vemo, da je izvedba funkcije varna, jo izvedemo še v avtomatskem režimu pri 100-% hitrosti. Kljub temu da je robot v avtomatskem režimu, se ne premika hitro, a vseeno pazljivost ni odveč. Slika 6.6 prikazuje postopek pridobivanja podatkov o uporabljenem orodju. Pri tem lahko rutino »LoadIdentity« izberemo za identifikacijo podatkov o orodju ali pa za identifikacijo mase izdelka »Payload«, ki ga premikamo s pomočjo orodja kot dodatno obremenitev. Postopka sta identična s to razliko, da je pri identifikaciji mase izdelka potrebno imeti prej natančno določeno orodje, pa naj bo to z ročnim vnosom ali pa z rutino »LoadIdentity«. Pri tem je potrebno poudariti, da je potrebno maso transportiranega izdelka kot tudi njegovo težišče in vztrajnostne momente natančno definirati. Nepravilnosti se kar hitro poznajo pri natančnosti gibanja robota. Slika 6.7 prikazuje odvisnost koordinatnih sistemov orodja, prirobnice ter težišča orodja in težišča izdelka.



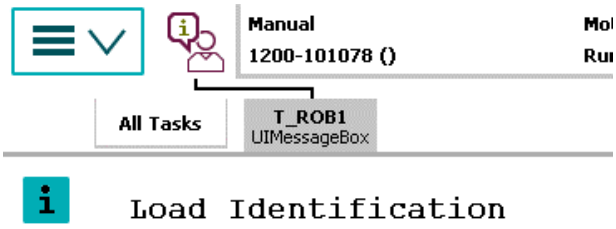
a) Izvajanje meritev



b) Preklop nazaj v ročni režim

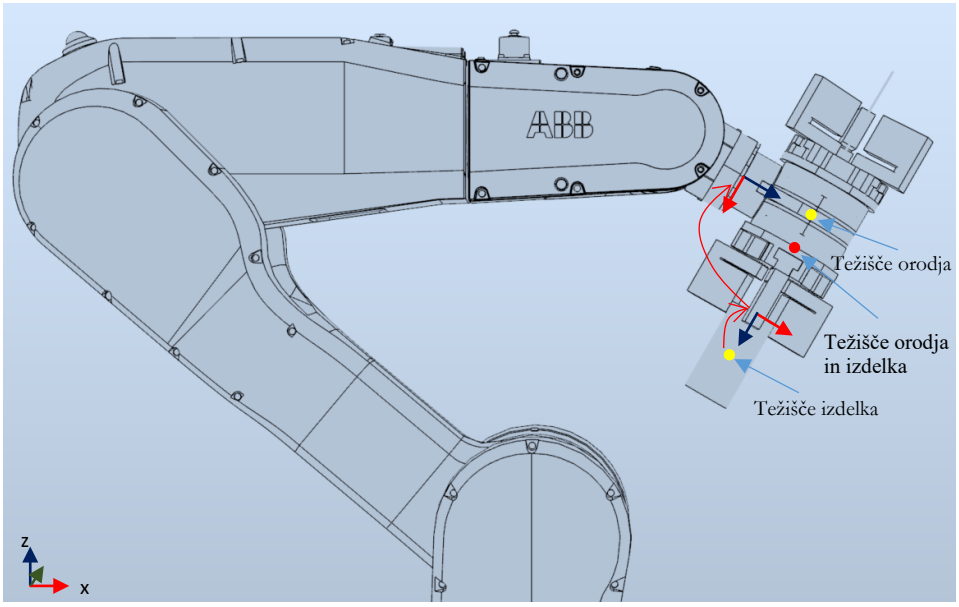


c) Izračunani podatki za orodje



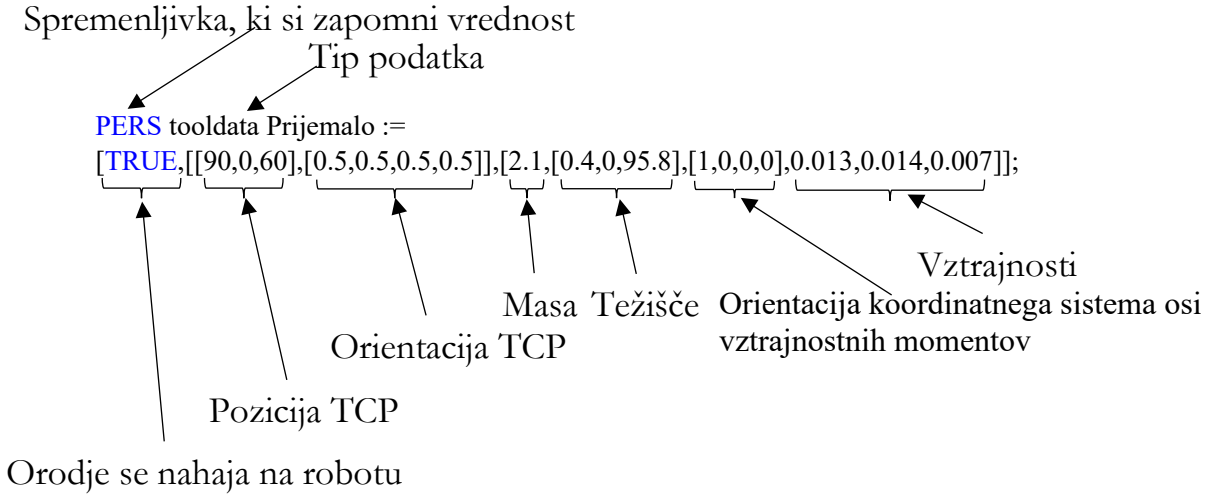
d) Izračunani podatki za maso izdelka

Slika 6.6: Postopek izračuna mase, težišča in masnih vztrajnostnih momentov



Slika 6.7: Prikaz odvisnosti koordinatnih sistemov orodja, prirobnice, težišča orodja in težišča izdelka

Podatki, ki se izračunajo s pomočjo rutine, se vnesejo na mesto v programu, kjer je orodje definirano.



Podobno se podatki posodobijo za breme oz. izdelek, ki je definiran v programu. Slika 6.8 prikazuje uporabo ukaza za upoštevanje dodatnega bremena pri prijemanju in odlaganju.

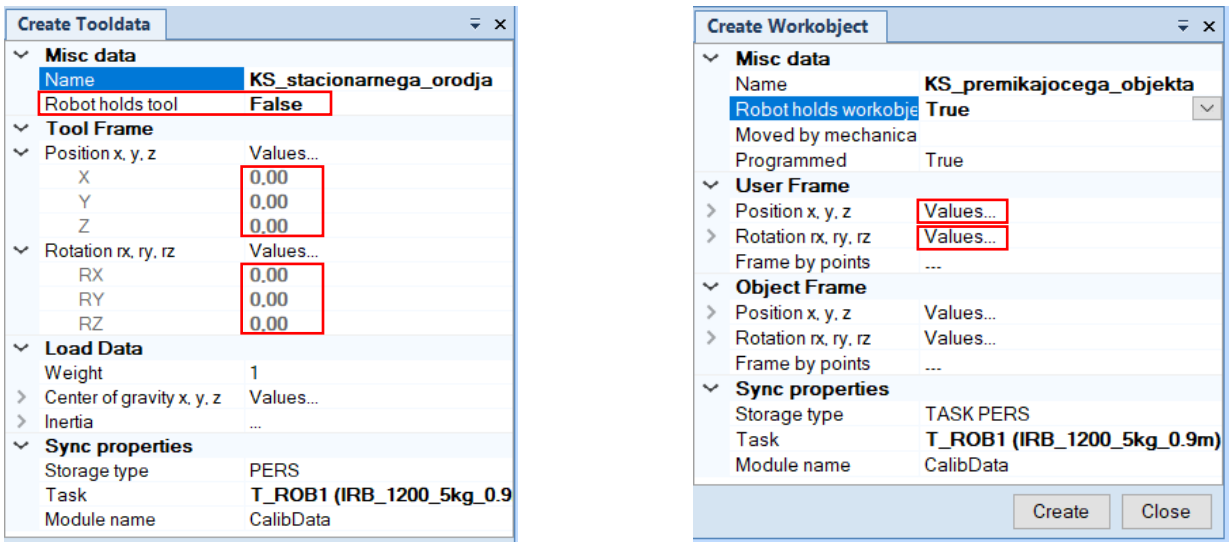
```
PERS loaddata Komad1:=[0.8,[270,0,80],[1,0,0,0],0,0,0];

PulseDO \High \PLength:=2,DO_Prijemalo1_primi;
WaitTime 0.5;
GripLoad Komad1;          Aktivacija bremena
!.
!.
!.
PulseDO \High \PLength:=2,DO_Prijemalo1_spusti;
WaitTime 0.5;
GripLoad load0;           Brez bremena
```

Slika 6.8: Primer ukaza za upoštevanje bremena »Komad1« pri prijemanju in odlaganju

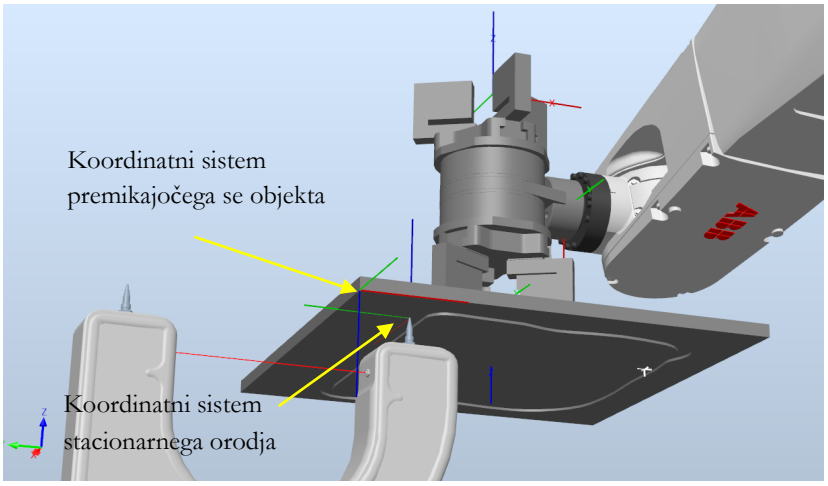
6.1.4 Konfiguriranje stacionarnega orodja in premikajočega se objekta v programu RobotStudio

Stacionarna orodja se v industrijski robotiki kar pogosto pojavljajo. Pojavljajo se pri varjenju, kjer imamo ostrenje varilne žice po vsakem končanem varjenju. Prav tako se uporabljajo kot luknjači, za luknjanje izdelka, ki ga drži robot. Ali pri aplikaciji nanosa lepila pod stacionarno šobo, kjer robot premika izdelek po predvideni trajektoriji, šoba pa je na fiksnem mestu.

Kreiranje novega stacionarnega orodja izvedemo s pomočjo ukaza »Other«, ki se nahaja pod zavihkom »Home«. Pri tem izberemo ukaz »Create Tooldata«. Pri kreiranju novega orodja je potrebno spremeniti nastavitev za »Robhold« na »False«, saj v tem konkretnem primeru orodje ni pritrjeno na robota. Za konfiguracijo TCP-točke stacionarnega orodja uporabljamo iste metode kot pri določevanju orodja, ki je pritrjeno na prirobnico robota. Se pa ta koordinatni sistem orodja kreira glede na **globalni koordinatni sistem** in ni vezan na koordinatni sistem robota, razen če se oba nahajata v isti točki. Pri takšnem načinu konfiguracije je potrebno kreirati orodje tako, da izberemo nastavitev »Robhold« na »False«. Pri kreiranju koordinatnega sistema objekta je potrebno izbrati nastavitev za »Robhold« na »True«, več o tem v poglavju 7. S tema dvema nastavitvama zagotovimo pravilno razmerje med obema koordinatnima sistemoma. Takšno razmerje nam omogoča zelo preprosto programiranje željene trajektorije v programu RobotStudio. Slika 6.9 prikazuje kreiranje koordinatnega sistema stacionarnega orodja in gibajočega se objekta. Slika 6.10 pa prikazuje koordinatna sistema stacionarnega orodja in premikajočega se objekta, ki ga drži robot. Kot priporočilo naj velja, da bo os Z koordinatnega sistema stacionarnega orodja obrnjena v nasprotno smer, kot je os Z koordinatnega sistema gibajočega se objekta.



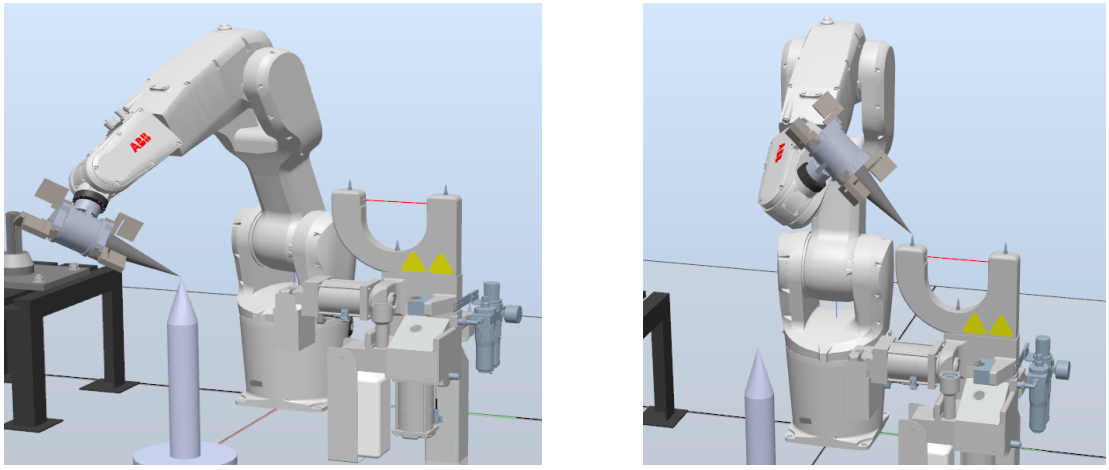
Slika 6.9: Kreiranje koordinatnega sistema stacionarnega orodja in gibajočega se objekta



Slika 6.10: Določitev koordinatnih sistemov stacionarnega orodja in premikajočega se objekta

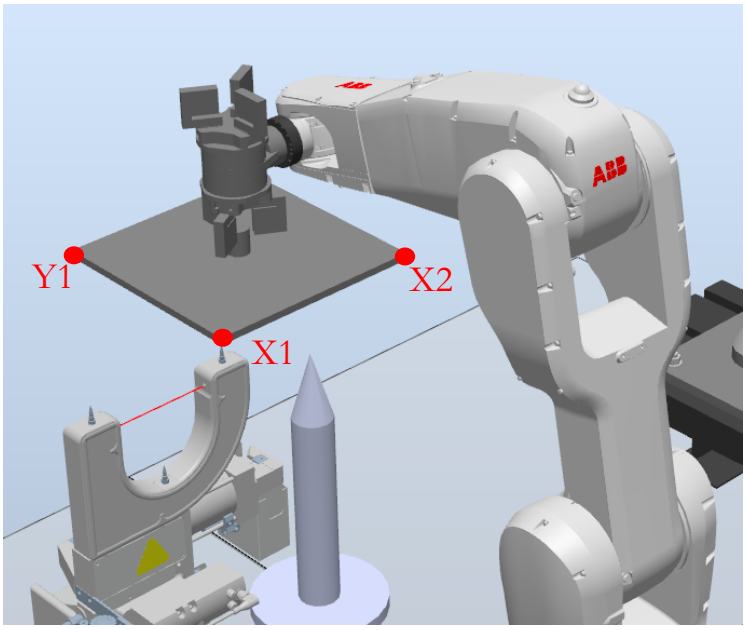
6.1.5 Konfiguriranje stacionarnega orodja in premikajočega se objekta na realnem robotu

Pogoj pri konfiguraciji stacionarnega orodja na realnem krmilniku je, da imamo že definirano koničasto orodje na robotu. **Brez TCP-točke na koničastem orodju, ki se nahaja na robotu, ni možno dovolj natančno definirati TCP-točke stacionarnega orodja.** V primeru triprstnega prijemala lahko v prijemalo vstavimo koničasti valj. V nasprotnem pa na orodje namestimo tanko ravno palico, ki lahko služi kot vrh TCP. S pomočjo na novo določene TCP-točke definiramo stacionarno orodje z novim TCP. Pri tem ne smemo pozabiti izbrati možnosti »Robot hold« na »False«! S pomočjo kreiranega stacionarnega TCP-ja določimo še koordinatni sistem objekta, ki ga drži robot v svojem prijemalu. Pri tem ne smemo pozabiti izbrati možnosti »Robot hold« na »True«! Postopek določevanja TCP je v obeh primerih enak in je opisan v prejšnjem podpoglavju.



a) Določevanje TCP koničastega orodja b) Določevanje TCP stacionarnega orodja

Slika 6.11: Določevanje TCP koničastega in TCP stacionarnega orodja



Slika 6.12: Določevanje koordinatnega sistema objekta, ki ga drži robot, glede na stacionarni TCP.

6.2 Vprašanja za utrjevanje snovi

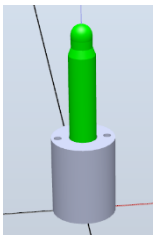
1. Zakaj je potrebno natančno definirati maso, težišče in vztrajnostne momente uporabljenega orodja?

2. Na kaj se navezuje hitrost izvajanja linearnega giba? Kako se to odraža pri nepravilno konfiguriranem orodju?

3. Kako postopamo pri določevanju mase, težišča in masnih vztrajnostnih momentov orodja in kako pri določevanju obremenitve transportiranih izdelkov, kadar teh podatkov ni na voljo?

4. V programu RobotStudio ustvarite nov robotski sistem z robotom IRB 1200 5/0.9 s programsko verzijo 6.10.XX. Na spletni strani Estudij, kjer se nahajajo predmeti Industrijska robotika, Robotizacija ali Roboti in robotizacija, iz datoteke RobotStudio prenesite CAD-model preprostega orodja pod imenom pisalo. Le-tega uvozite v RobotStudio kot preprosto geometrijo. Kreirajte novo orodje z naslednjimi karakteristikami. Po kreiranju orodja, le-tega pritrdite na prirobnico robota in ustvarite poljubno trajektorijo z novim orodjem.

Masa:	0.8 kg
Težišče:	x = 0 mm
	y = 0 mm
	z = 65 mm
Vztrajnostni momenti:	Ix = 0.001 kgm2
	Iy = 0.001 kgm2
	Iz = 0.01 kgm2
TCP (Tool center point):	x = 0 mm
	y = 0 mm
	z = 125 mm



5. Na spletni strani Estudij, kjer se nahajajo predmeti Industrijska robotika, Robotizacija ali Roboti in robotizacija, iz datoteke RobotStudio prenesite »Stacionarno_orodje_komad_v_prijemalu.rspag«. Ta datoteka je »Pack&Go« datoteka za program RobotStudio. Z dvojnim klikom na datoteko se zažene postopek »Unpack&Work«. Po zagonu datoteke dodajte na prirobnico robota že obstoječe orodje »Prijemalo_2«, ki ima izdelek z utorom. Na novo kreirajte koordinatni sistem stacionarnega orodje, ki se nahaja na stroju z oznako TSC2013, ter na novo kreirajte koordinatni sistem gibajočega se izdelka. Po uspešnem kreiranju sprogramirajte robota, da se bo z utorom premikal po stacionarnem orodju. Pri tem si pomagajte z ukazom »AutoPath« ter pazite na pravilno izbiro trenutno aktivnega orodja in koordinatnega sistema objekta.

Ugotovite:

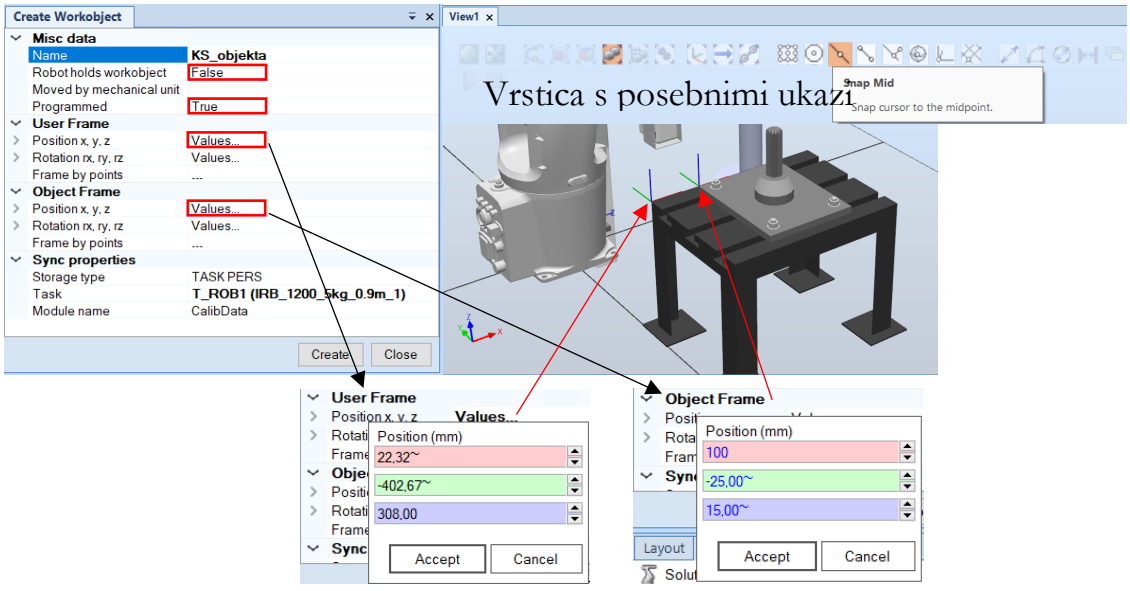
Univerzitetna založba
Univerze v Mariboru

7 Konfiguracija koordinatnih sistemov objektov

V novejšem času se večino industrijskih robotskih aplikacij razvija »offline« preko namenskih programov. Ta način omogoča, da že v fazi načrtovanja ugotovimo ustreznost uporabljenega robota, ustreznost postavitve robota do strojev in določimo cikel. Ker pa še vedno ostaja odstopanje virtualnega od realnega sistema, še posebej pri velikih robotih, je pri programiranju industrijskih robotov smiselno uporabljati koordinatne sisteme objektov. Uporaba koordinatnih sistemov objektov se ni razvila zaradi razvoja »offline« programov, ampak ima predvsem praktičen pomen. Predstavljajte si, da ste robotski programer, ki razvija programe za avtomobilsko industrijo. Pri tem razvijate aplikacijo za nanos lepila na notranji del sprednjega avtomobilskega pokrova, na katerega potem pride nalepljena notranja zaščita proti hrupu. Pri tem vse točke tega programa vežete na koordinatni sistem robota oz. bazni koordinatni sistem. Pri tem ustvarite program s približno 500–600 točkami. Ko zaključite z delom, nadzornik ugotovi, da postavitev naprave za nanos lepila ni na ustreznem mestu in jo je potrebno prestaviti. To pomeni, da morate vsako točko nanosa lepila ustrezno zamakniti. Z uporabo koordinatnega sistema objekta bi bilo potrebno samo ustrezno premakniti koordinatni sistem in bi se vse točke, ki so vezane na ta koordinatni sistem, premaknile z njim. Na koncu je potrebno samo še enkrat preveriti celoten program. To bistveno olajša delo programerja.

7.1 Definiranje koordinatnih sistemov v programu RobotStudio

V programu RobotStudio je definiranje koordinatnega sistema zelo enostavna, saj ima mnogo odličnih pomagal, ki nam olajšajo izbiro robov, kotov, točk, zaokrožitev in še kaj na objektu, kjer želimo ustvariti nov koordinatni sistem. Za kreiranje novega koordinatnega sistema moramo v zavihku »Home« izbrati ukaz »Other« in potem »WorkObject«. Pri tem moramo ločiti koordinate objekta »User Frame« in pa koordinate obdelovanca »Object Frame«. Kot je bilo rečeno že v poglavju 3.3, je koordinatni sistem objekta »User Frame« vezan na globalni koordinatni sistem »World«, koordinatni sistem obdelovanca »Object Frame« pa je vezan na koordinatni sistem objekta »User Frame«. Ta specifika je značilna samo za ABB-jeve robote. Takšna postavitev koordinatnih sistemov ima tudi svojo prednost, predvsem pri premikanju postavitve obdelovanca do mize oz. objekta. Slika 7.1 prikazuje kreiranje koordinatnega sistema objekta in obdelovanca v programu RobotStudio. Če je koordinatni sistem fiksiran na objekt, ki ga robot ne premika, je potrebno izbrati »Robot holds workobject« na »False«. Robove v programu RobotStudio izbiramo s pomočjo ukaza »SnapEnd«, ki označi najbližji rob, kjer se kurzor nahaja.

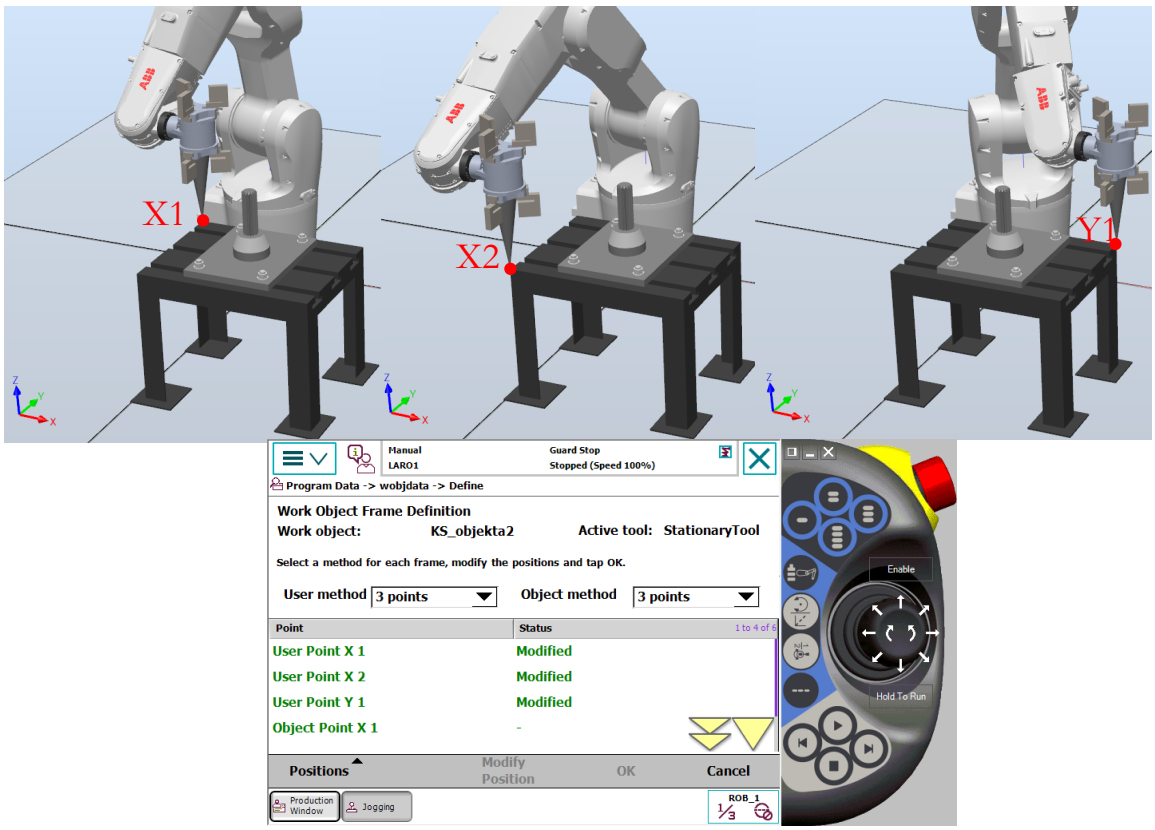


Slika 7.1: Kreiranje koordinatnega sistema objekta in obdelovanca v programu RobotStudio

7.2 Definiranje koordinatnih sistemov na krmilniku IRC5

Če želimo kreirati koordinatni sistem objekta in obdelovanca preko krmilnika IRC5, je nujno potrebno, da imamo najprej pravilno definiran koordinatni sistem orodja. V nasprotnem primeru bo prihajalo do odstopanj, še posebej, če bomo v prihodnje zamenjali orodje in ga natančno definirali. Postopek kreiranja koordinatnega sistema je naslednji:

Najprej v meniju izberemo možnost »Jogging«, kjer kliknemo na »Work object« in kreiramo nov koordinatni sistem. Pri tem zapišemo ime in okno zapremo. Izberemo kreiran koordinatni sistem in kliknemo na »Edit« ter potem »Define«. Slika 7.2 prikazuje kreiranje koordinatnega sistema objekta na krmilniku IRC5. Pri tem nam krmilnik ponudi možnosti za kreiranje ali koordinatnega sistema objekta ali pa tudi obdelovanca. Kot je že bilo povedano, se koordinatni sistem objekta shrani glede na globalni koordinatni sistem, koordinatni sistem obdelovanca pa glede na koordinatni sistem objekta. Prav tako lahko koordinatni sistem definiramo tako, da je pritrjen na prirobnico robota. Pri tem je potrebno izbrati opcijo »Robot hold workobject« na »True«.



Slika 7.2: Kreiranje koordinatnega sistema objekta in obdelovanca na krmilniku IRC5

7.3 Vprašanja za utrjevanje snovi

1. Naštejte razloge, zakaj uporabljamo koordinatne sisteme objektov v industrijski robotiki. Kakšne so prednosti pred uporabo zgolj baznega koordinatnega sistema? Kako definiramo dodatni koordinatni sistem na primeru robota ABB? Kakšna je razlika med koordinatnima sistemoma »User« in »Object«?

2. Na spletni strani Estudij, kjer se nahajajo predmeti Industrijska robotika, Robotizacija ali Roboti in robotizacija iz datoteke RobotStudio – vaje prenesite »Stacionarno_orodje_komad_v_prijemalu.rspag«. Ta datoteka je »Pack&Go« datoteka za program RobotStudio. Z dvojnim klikom na datoteko se zažene postopek »Unpack&Work«. Na prirobnico robota pritrdite orodje z imenom »Dvojni_Gimatic_stozec«. Na vogalu podporne mize z imenom »Table_and_fixture« kreirajte nov koordinatni sistem. Pri tem definirajte najprej podatke za koordinatni sistem objekta oz. »User«, potem pa na obdelovancu kreirajte še koordinatni sistem obdelovanca oz. »Object«. Po tem sprogramirajte robota, da se sprehodi po robovih obdelovanca, kjer naj bodo koordinate točk vezane na novokreirani koordinatni sistem.

Ugotovitve:

ZAPISKI

8 Konfiguracija PROFINET komunikacije

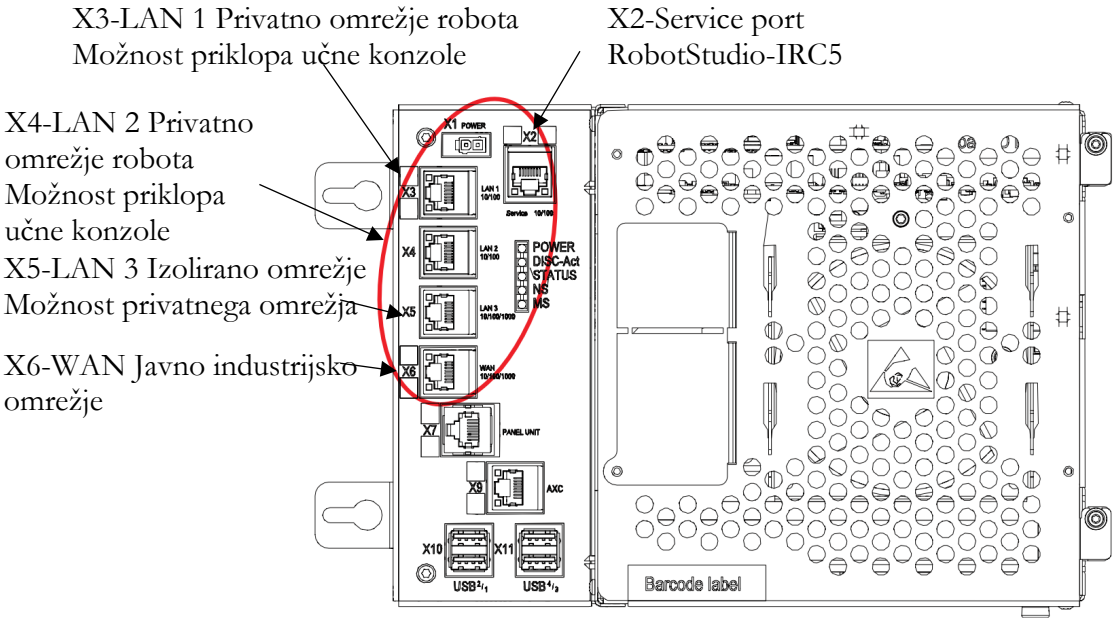
8.1 Kaj je PROFINET?

Danes si življenje brez interneta težko predstavljamo. Prav tako je internet že povsod in skoraj vsi imamo dostop do njega. Vendar se nam dogaja, da se hitrost prenosa upočasni, da se povezava z internetom prekine in se čez nekaj časa obnovi. Pri spremljanju novic ali gledanju kakšnega posnetka to ni kritično. Takšni dogodki pa se v industrijskem okolju ne smejo zgoditi. V ta namen je bil razvit PROFINET [22, 41]. To je industrijski ethernet [42], ki deluje v realnem času in skrbi za 100-% prenos podatkov od pošiljatelja do naslovnika. PROFINET tako definira celotno izmenjavo vseh podatkov med krmilniki, ki jih imenujejo »IO-Controllers« in napravami, ki jih imenujejo »IO-Devices«. »Controller« je tudi znan pod imenom »Master« oz. gospodar, »Device« pa kot »Slave« oz. suženj. »IO-Controllers« so v večini primerov PLK-ji, lahko pa je to tudi robot, ki ima DI/DO ali AI/AO-kartico, ki komunicira preko PROFINET protokola. Roboti so v industriji najpogosteje definirani kot »Devices«, kjer prejemajo ukaze od nadzornega krmilnika.

8.2 Nastavitev PROFINET komunikacije na krmilniku IRC5

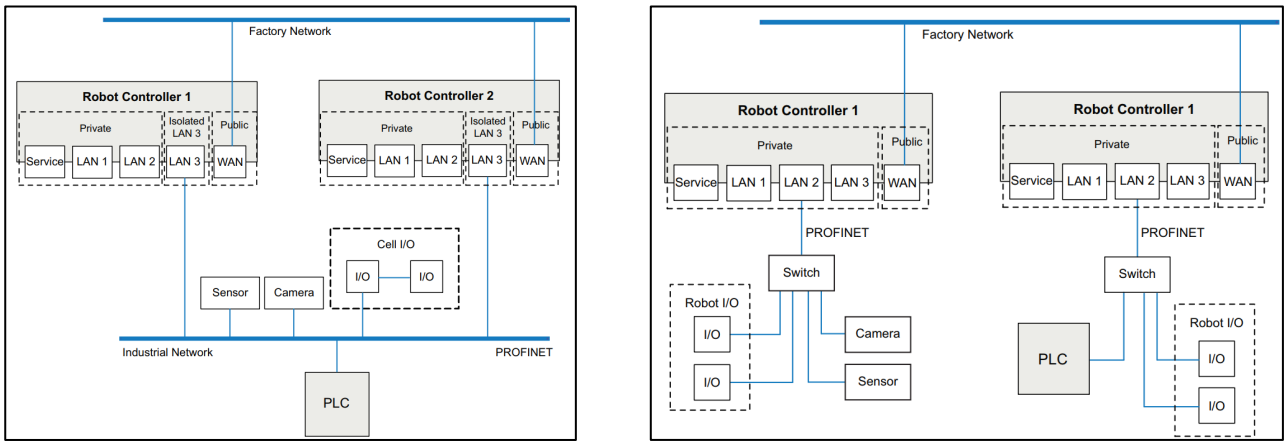
V tem poglavju bodo predstavljene poglavitne informacije, ki so tudi splošne narave pri nastavljanju PROFINET komunikacije na industrijskih robotih. Miselni postopek je v večini primerov enak. Prva stvar, ki se je moramo zavedati, je, ali bo robot imel vlogo »Controller-ja« ali bo deloval v načinu »Device«. Od tega je odvisna tudi nastavitev robota in temu primerno pri proizvajalcu naročimo ustrezno opcijo. Če bo robot deloval

izključno samo kot »Device«, je dovolj, da izberemo opcijo 888-3 PROFINET Device. Če bo deloval kot »Controller« ali pa »Controller / Device«, je nujno izbrati opcijo 888-2 PROFINET Controller/Device. Izbrana opcija je ob prihodu robota že avtomatsko inštalirana. Slika 8.1 prikazuje vse možne internetne priključke krmilnika IRC5 Compact.



Slika 8.1: Možna uporaba internetnih priključkov [43]

Pri tem uporabljamo WAN (Wide Area Network) priključek za priklop na internetno omrežje podjetja, pri čemer potrebujemo javni IP, ki nam ga dostavi administrator podjetja. Če podjetje uporablja PROFINET protokol na industrijskem omrežju, se na to omrežje priklopimo preko priključka WAN. Če uporabljamo PROFINET protokol za komunikacijo z zunanjimi napravami, kot so kamere ali senzorji, bomo uporabili ta protokol na privatnem omrežju znotraj krmilnika robota na portu npr. LAN 2. Preko WAN pa potem komuniciramo z industrijskim omrežjem. Po navadi ima podjetje ločeno industrijsko omrežje z omrežjem, na katerem se nahajajo vsi stroji. Pri tem iz industrijskega omrežja do robota zopet dostopamo preko WAN, za komunikacijo do naprav in nadzornih PLK-jev pa uporabljamo LAN 3, ki velja za izolirano omrežje. Slika 8.2 prikazuje vse možne kombinacije.



a) PROFINET na ločenem industrijskem omrežju

b) PROFINET na privatnem omrežju

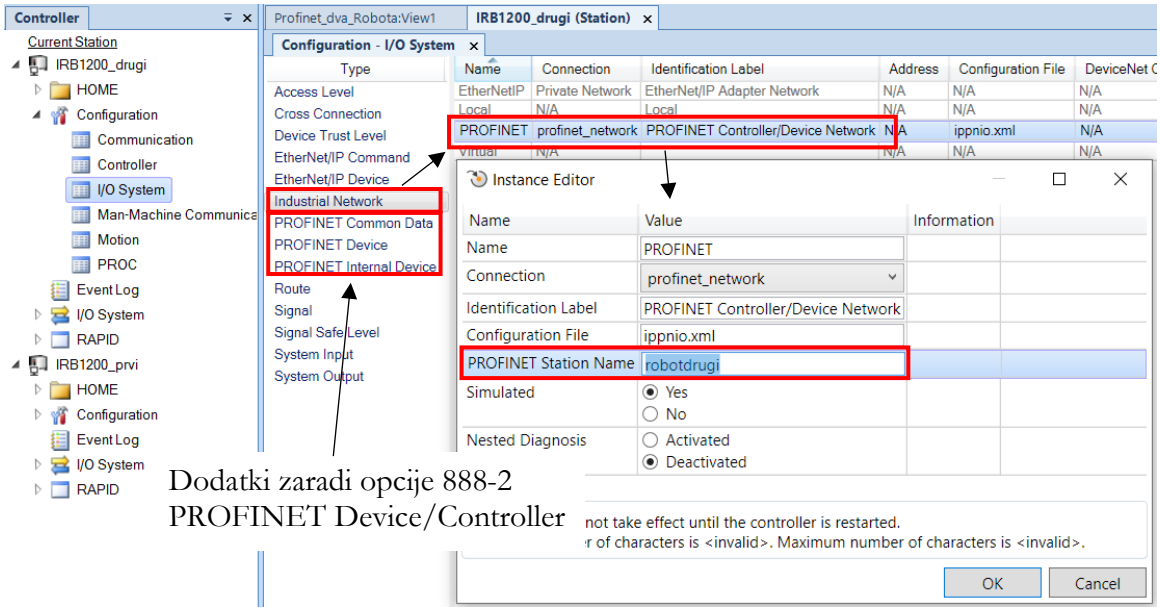
c) PROFINET na glavnem industrijskem omrežju

Slika 8.2: Načini priklopa krmilnika na industrijsko omrežje in na PROFINET [43]

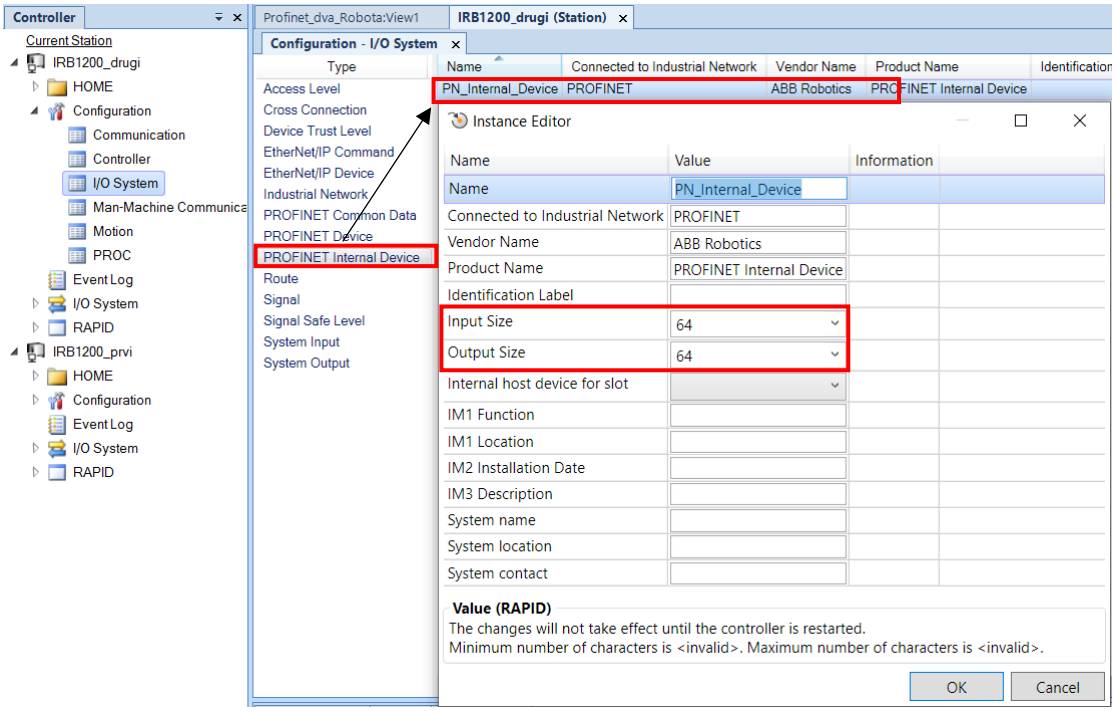
8.2.1 Nastavitev IP-naslova in vhodno/izhodnih signalov z nadzornim PLK-jem

Ko vemo, na kateri priključek bomo namestili PROFINET komunikacijo, lahko preko programa RobotStudio konfiguriramo vse potrebne nastavitve na krmilniku IRC5. Najprej je potrebno nastaviti in poimenovati industrijsko omrežje znotraj krmilnika IRC5. Najpomembnejše je, **kako poimenujemo krmilnik robota**. Ta mora biti enolično definiran z malimi črkami in brez presledkov. Slika 8.3 prikazuje nastavitve industrijskega omrežja. Z vključitvijo funkcije 888-2 PROFINET Device/Controller se v drevesni strukturi pojavijo tri dodatne opcije. Pri tem je najpomembnejša PROFINET Internal Device, ki skrbi za prenos podatkov med zunanjim krmilnikom, ki služi kot »Controller«, in krmilnikom robota, ki služi kot »Device«. V naslednjem koraku je potrebno nastaviti preko PROFINET Internal Device, koliko bajtov potrebujemo za komunikacijo z nadzornim sistemom, ki je po navadi PLK. Pri tem je potrebno poudariti, da je en bajt sestavljen iz 8 bitov. PROFINET Internal Device nam tako omogoča izmenjavo do 256 bajtov, kar je 256 x 8 = 2048 signalov tako za pošiljanje kot za prejemanje. Slika 8.4 prikazuje določevanje števila bajtov za izmenjavo signalov.

Če želimo imeti na PROFINET komunikaciji še dodatno kartico, kjer je robotski krmilnik »Controller«, je to potrebno dodati k »PROFINET Device«, saj je ta kartica »Device«. Dodajanje kartice preko »I/O Configurator« bo podano v 8.2.3.



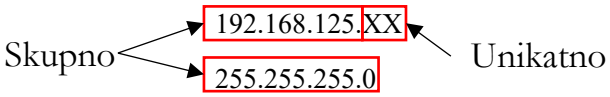
Slika 8.3: Poimenovanje omrežne povezave in krmilnika



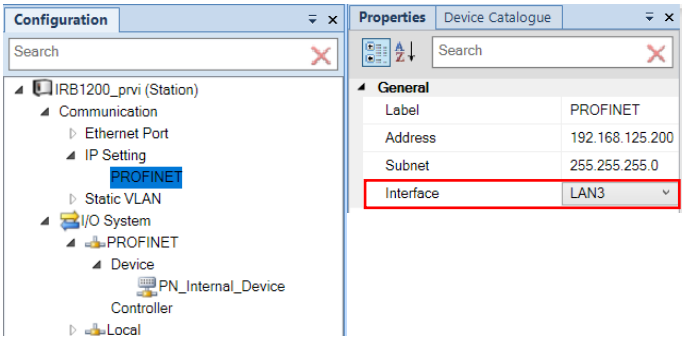
Slika 8.4: Nastavljanje števila potrebnih bajtov (byte) za komunikacijo s PLK

Do sedaj smo dodali vse naprave, s katerimi komuniciramo, a še nismo nastavili IP-naslovov. Vsaka naprava ima svoj IP-naslov, preko katerega komunicira z drugo napravo.

Pri tem je potrebno upoštevati, da so prve tri številke IP-naslova identične, zadnja pa je unikatna za vsako napravo posebej.

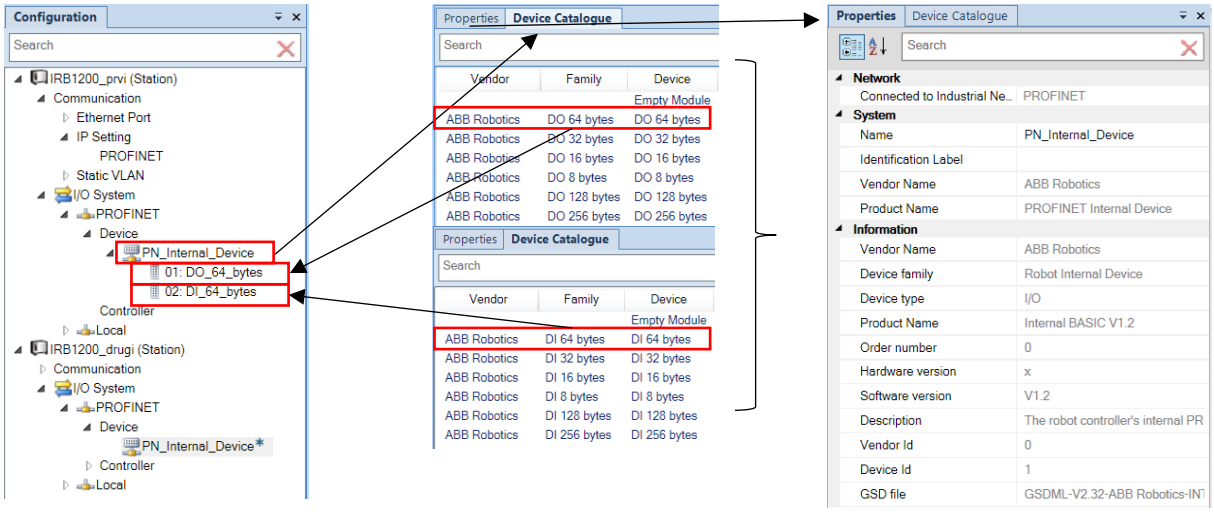


Nastavljanje IP-naslovov izvedemo preko ukaza »IO Engineering Tool«, ki se v RobotStudio nahaja pod zavihkom »Controller« ter pod ukazom »Configuration«. Najprej je potrebno nastaviti IP-naslov robotskega krmilnika. V drevesni strukturi izberemo »IP Settings« ter PROFINET. Slika 8.5 prikazuje nastavljanje IP-naslova za PROFINET. Pri tem je potrebno nastaviti tudi, na kateri port bo PROFINET priključen.



Slika 8.5: Nastavljanje IP-naslova ter porta za PROFINET

Naslednje, kar moramo nastaviti, je interna naprava PN_Internal_Device za komunikacijo z nadzornim PLK-jem. Določili smo že število vhodno/izhodnih signalov. Sedaj moramo določiti še, h kateremu podsklopu spadajo. Slika 8.6 prikazuje nastavljanje vhodno/izhodnih signalov PN_Internal_Device. Ko nastavimo signale, nam možnost, da sami izbiramo velikost vhodno/izhodnih signalov, ni več omogočena, saj smo jih s tem eksplicitno nastavili. Tako imamo na voljo za komunikacijo 64 bajtov oz. 512 vhodnih in 512 izhodnih signalov.

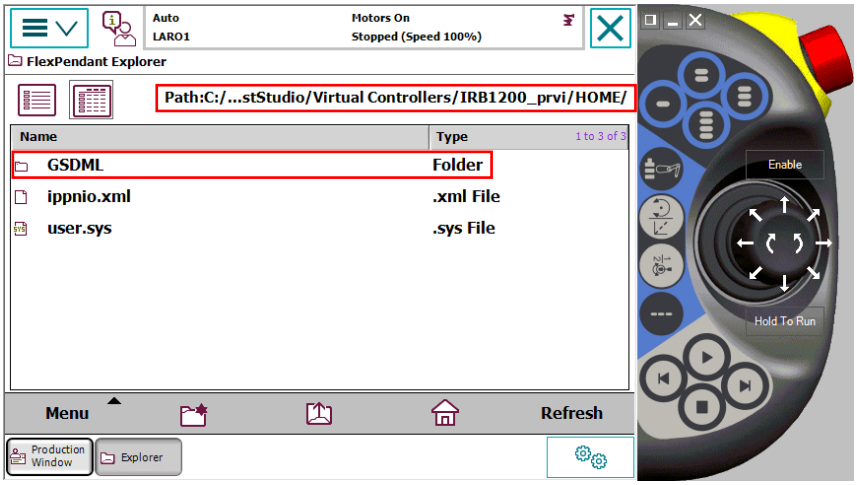


Slika 8.6: Nastavljanje vhodno/izhodnih signalov PN_Internal_Device

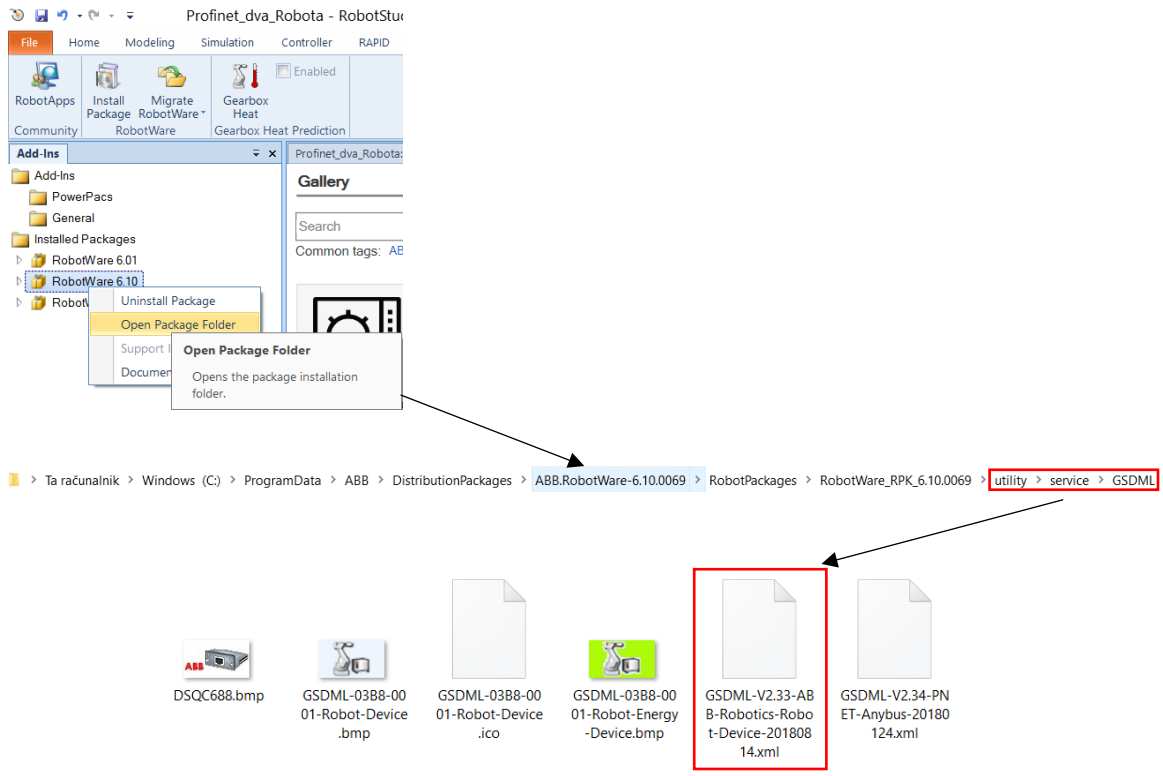
8.2.2 Potrebne nastavitve na PLK za komunikacijo s krmilnikom IRC5

Ker programiranje PLK-jev ni stvar robotskih programerjev, temu podpoglavju ne bomo posvečali preveč pozornosti. Pri konfiguraciji PROFINET komunikacije na PLK-ju, ki deluje v načinu »Controller«, je potrebno nastaviti IP-naslov naprave, s katero komunicira, ter vnesti datoteko GSD oz. GSDML naprave [44], s katero komunicira. Datoteka GSD oz. General Station Description vsebuje opis naprave proizvajalca. Pri tem datoteka zajema konfiguracijo, parametre, module, diagnostiko in alarme ter unikatno ID-številko oz. »vendor ID« in identifikacijo naprave oz. »device ID«. Pri tem unikatno ID-številko oz. »vendor ID« poda organizacija PI (PROFIBUS & PROFINET International) [45], ki skrbi, da imajo vse naprave unikatno ID. GSD-datoteka je namenjena PROFIBUS protokolu, ki uporablja tekstovne datoteke ASCII. PROFINET pa uporablja format XML, zato za ta protokol potrebujemo datoteke GSDML [44]. Pri tem velja opozorilo, da je potrebno preveriti, katera verzija je kompatibilna z uporabljenim krmilnikom.

Za uspešno komunikacijo med PLK (»Controller«) in krmilnikom robota (»Device«) je potrebno na strani PLK vnesti GSDML-datoteko uporabljene naprave. GSDML-datoteka robota se nahaja ali na krmilniku IRC5 ali pa v datoteki, kjer je naložen program RobotStudio. Slika 8.7 prikazuje pridobitev GSDML-datoteke preko učne konzole. Slika 8.8 pa prikazuje pridobitev GSDML-datoteke preko programa RobotStudio.



Slika 8.7: Pridobitev GSDML-datoteke preko učne konzole

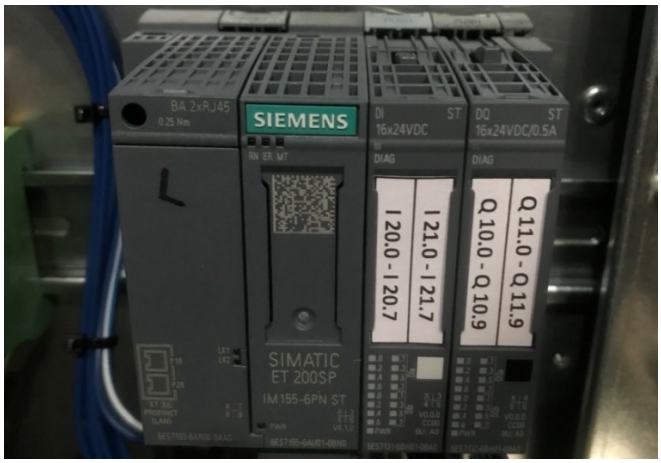


Slika 8.8: Prikaz pridobitve GSDML-datoteke za ABB-jevega robota v programu RobotStudio

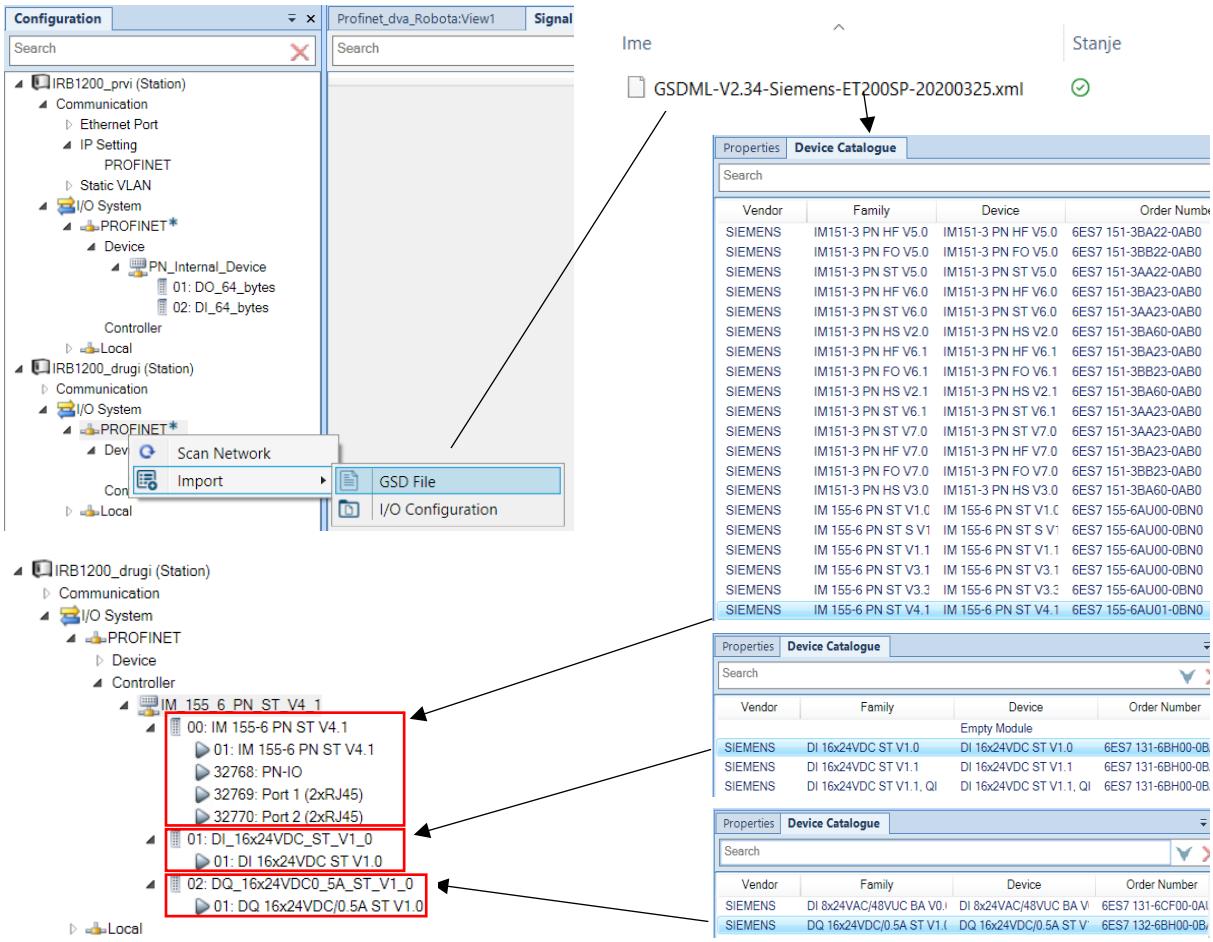
8.2.3 Dodajanje naprave v načinu »Device« na krmilnik robota

Če imamo namen dodati napravo, kot je dodatna kartica z DI/DO-signalov, moramo preko programa RobotStudio v »I/O Engineering Tool« uvoziti datoteko GSDML. Slika 8.10 prikazuje uvoz GSDML-datoteke. V tem primeru bomo dodali Siemens kartico ET 200SP [46] s 16 DI/DO-signalov. Slika 8.9 prikazuje Siemens ET 200SP kartico. Ta datoteka nam omogoča, da preko »Device Catalogue« vnesemo izbrano napravo, ki je vnaprej konfigurirana. Pri tem konfiguracija zavzema PROFINET vmesnik in napajalni

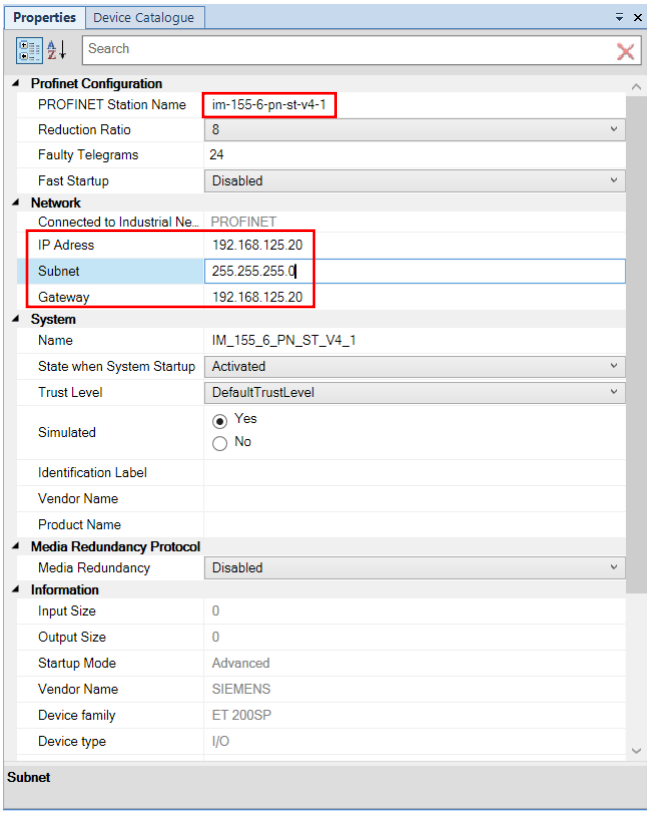
modul. Koliko uporabljenih vhodno/izhodnih signalov imamo priklopljenih na kartico, je potrebno posebej definirati. Pri tem je potrebno tudi poudariti, da napravo dodajamo pod »Controller«, saj je razširitvena kartica »Device« robotskemu krmilniku. Slika 8.10 prikazuje vnos GSDML-datoteke in željene konfiguracije vhodno/izhodnih signalov. Tej napravi nato vnesemo samo še IP-naslov. Slika 8.11 prikazuje nastavitev IP-naslova.



Slika 8.9: Siemens ET 200SP kartica s 16 DI/DO



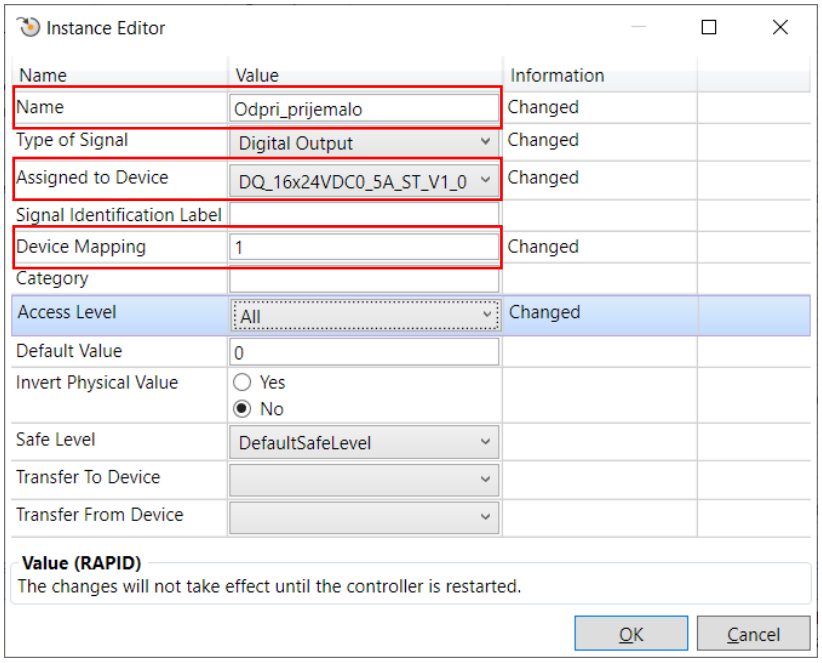
Slika 8.10: Vnos GSDML-datoteke in izbira naprave kot »Device«



Slika 8.11: Nastavitev imena in IP-naslova »Device« naprave

8.2.3 Kreiranje izhodno/vhodnih signalov, ki potekajo preko PROFINET

Signale, ki potekajo preko PROFINET komunikacije, kreiramo podobno, kot je to bilo storjeno v poglavju 5. Potrebno je izbrati zavihek »Controller« ter »Configuration«, kjer izberemo »I/O System« za dodajanje novega signala. Pri tem signalu izberemo ime, določimo tip signala in kateri kartici pripada. Ker smo v prejšnjem poglavju definirali Siemensovo kartico ET 200SP z DI/DO-signalov, je treba ta signal povezati s to kartico ter mu določiti naslov oz. »Device Mapping«, ki pa je zaporedna številka izhoda. Slika 8.12 prikazuje konfiguracijo izhodnega signala. Tukaj je smiselno poudariti, da naslov oz. »Device Mapping« signalov, ki potekajo preko PROFINET s Siemensovo kartico, nimajo nobene povezave s signali, ki potekajo preko DeviceNet protokola. To sta med sabo ločena sistema, tako da lahko v obeh primerih uporabljamo iste »Device Mapping« naslove. Če se vrnemo k primerjavi IP-naslovov z naslovi ulic in hišnih številk, imamo v različnih mestih v Sloveniji iste ulice z istimi hišnimi številkami, a to Pošto Slovenije nič ne moti, saj se nahajajo v različnih mestih oz. krajih.



Slika 8.12: Konfiguracija izhodnega signala, do katerega dostopamo preko PROFINET komunikacije.

V nadaljevanju bo prikazana osnovna struktura programov ABB-jevih robotov, kako je program strukturiran, kateri so njegovi podsklopi ter kako so med seboj povezani. Po tem bo predstavljen princip pošiljanja trenutne pozicije robota preko PROFINET komunikacije do drugega robota, pri čemer je vrednost pozicije realno število, ki je lahko tako pozitivno kot tudi negativno z decimalnim mestom.

8.3 Vprašanja za utrjevanje snovi

1. Zakaj v industriji uporabljamo PROFINET komunikacijo? Kaj je tisto, kar mu daje prednost pred navadnim ethernet protokolom?

2. V programu RobotStudio kreirajte nov robotski sistem z robotom IRB 1200 5/0.9. Pred klikom na gumb »Create« izberite še opcijo »Customize options«. S tem se po pritisku na »Create« odpre okno z dodatnimi opcijami. Pri tem izberite opcijo »PROFINET Controller/Device«, ki se nahaja pod »Industrial Networks«. Po kreiranju robotskega sistema je potrebno dodati razširitveno kartico ET200SP s 16 DI/DO-signalov. GSDML-datoteka se nahaja na spletni strani Estudij, kjer se nahajajo predmeti Industrijska robotika, Robotizacija ali Roboti in robotizacija, v datoteki RobotStudio – vaje.

Ugotovitve:

3. Pod zavihkom »Controller« izberite ukaz »Configuration« in »I/O Engineering Took«. S tem se odpre I/O Configurator. Nastaviti je potrebno IP-naslov krmilnika robota in mu dati unikatno ime. Za IP uporabite naslednje nastavitve: 192.168.125.1, ter 255.255.255.0. Prav tako je potrebno krmilniku dodati unikatno ime, ki bo vidno v PROFINET omrežju. To storimo s klikom na »PROFINET« v drevesni strukturi in na desnem okencu pod »Properties« spremenimo ime. Krmilnik poimenujte »abbirb1200«. Da se spremembe trajno shranijo, je potrebno resetirati robotski krmilnik.

Ugotovitve:

4. Ko smo končali z nastavitvami IP-naslova in poimenovanjem naprave, lahko dodamo razširitveno kartico ET 200SP. Z desnim klikom na »PROFINET«, ki se nahaja v drevesni strukturi, izberemo »Import« in »GSD File«. Poiščemo direktorij, kamor smo shranili datoteko GSDML in jo uvozimo. V drevesni strukturi kliknemo na »Controller«, nato pa v desnem oknu izberemo »Device Catalogue«, kjer se nam pokaže družina vseh naprav ET 200SP. Izberemo »IM 155-6 PN ST V4.1« kot glavno napravo. Preimenujemo napravo iz »IM_155_6_PN_ST_V4_1« na »et200sp«. Tej napravi je potrebno dodati še kartico DI in kartico DO s 16 signali. Tako najprej dodamo še »DI_16x24VDC_ST_V1_0« in nato še »DQ_16x24VDC0_5A_ST_V1_0«. Po vseh nastavitvah je potrebno resetirati robotski krmilnik, da spremembe postanejo trajne.

Ugotovitve:

5. Po uspešni konfiguraciji razširitvene kartice lahko kreiramo nov signal. Na razširitveni kartici kreirajte 2 DI- in 2 DO-signala. Pri tem ne pozabiti, da je potrebno signal povezati na razširitveno kartico. V tabelo dopolnite zaporedno številko, ki ste jo priredili v nastavitvah »Device Mapping«.

Signal	Ime signala	Oznaka na ET200	Device Mapping
Digitalni vhod	Vrata_odprta	DI1.0	
Digitalni vhod	Vrata_zaprta	DI1.1	
Digitalni izhod	Vrata_odpri	DQ2.0	
Digitalni izhod	Vrata_zapri	DQ2.1	

Ugotovitve:

ZAPISKI

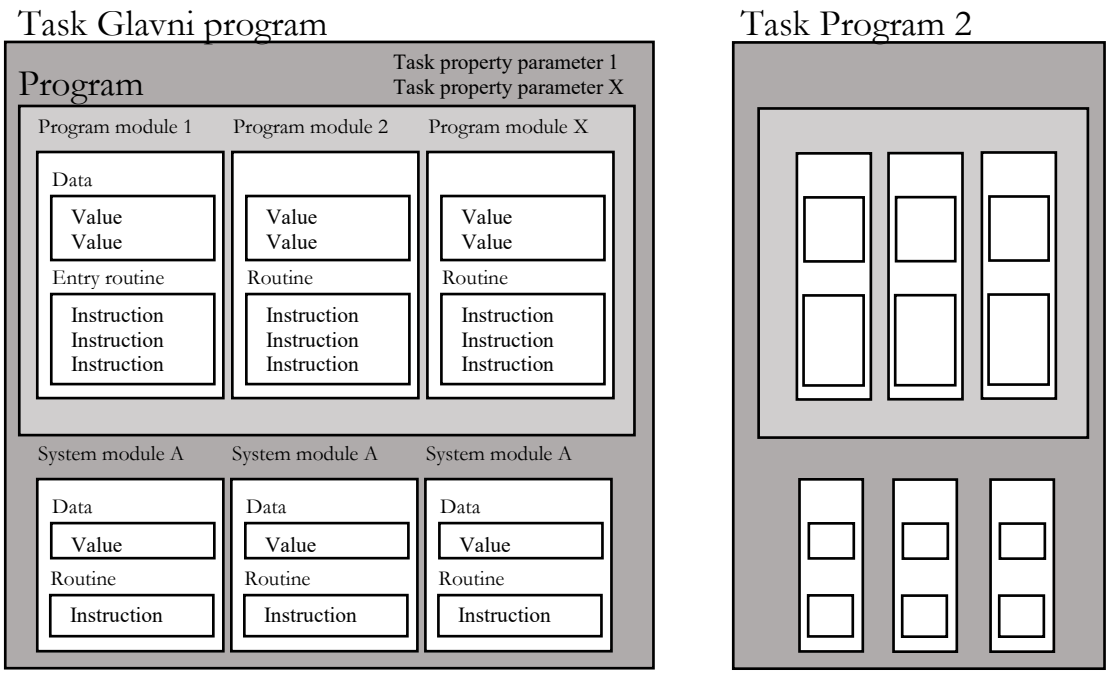
Univerzitetna založba
Univerze v Mariboru

9 Osnove programiranja industrijskega robota

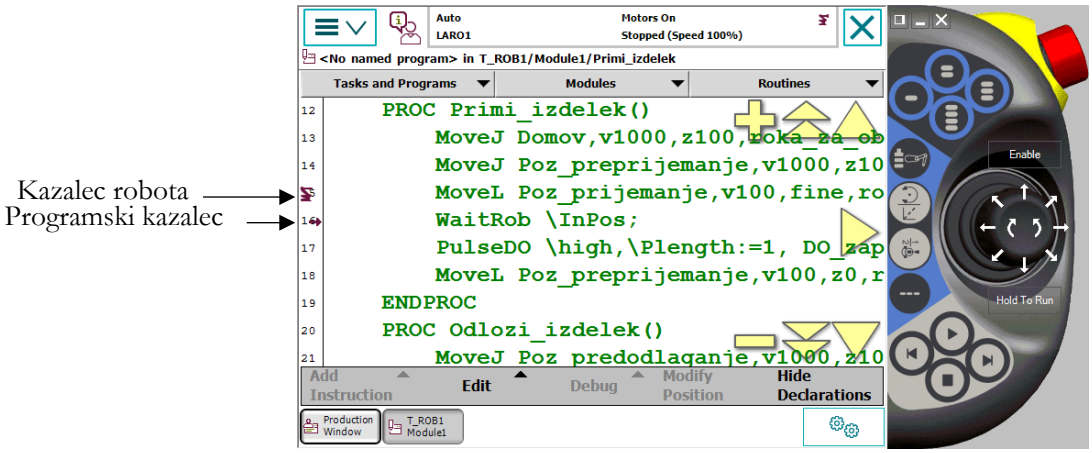
Programiranje industrijskega robota ni zgolj znanje o vodenju robota do določene pozicije in shranjevane le-te, temveč je mnogo več. Kot je bilo že v prejšnjih poglavjih rečeno, je potrebno postoriti kar nekaj nastavitev, preden lahko sploh začnemo programirati robota in pripravljati željeni program. Industrijski robot je v večini primerov integriran v nek proizvodnji proces in tako se mora tudi obnašati. V veliki večini ima robot vlogo podrejenega stroja in si z nadzornim sistemom izmenjuje določene podatke, kot je npr. trenutna pozicija robota, v okolici katerega stroja se nahaja, kdaj ima dovoljenje za izvajanje programa, kateri program izvaja, kdaj se vrne na začetno varno pozicijo ter spreminjanje hitrosti izvajanja programa. Vse te podatke si izmenjuje preko serijske komunikacije. V tem poglavju bo predstavljana zgradba in uporaba programov na ABB-jevem krmilniku, predstavljeni bodo osnovni ukazi programiranja in pisanja programov in podprogramov ter definiranje globalnih in lokalnih spremenljivk. Ker je sintaksa programiranja ABB-jevih robotov drugačna od drugih in ker je veliko možnih spremenljivk, bodo tukaj predstavljene samo najnujnejše, za vse druge pa priporočamo uporabo navodil proizvajalca.

9.1 Osnovna delitev robotskega programa na krmilniku ABB IRC5

V osnovi je robotski program razdeljen na glavni del »Main«, v katerem se program začne izvajati, čemur pravijo tudi »Entry routine«, ter na podprograme, ki jih izvajamo posamično po potrebi. Prav tako določeni signali povzročajo prekinitve trenutnega programa in izvedbo prekinitvene rutine. En takšen primer zasledimo, kadar robot želi odložiti vrečo pšenice na paleto. Vemo, da višina vreče ni točno določena, saj se pšenica v vreči razleze. V tem primeru imamo na prijemalo nameščen laserski merilec razdalje, ki nam sporoča, kdaj smo 100 mm nad spodnjo vrečo. V ta namen uporabimo signal iz laserskega merilca razdalje, ki prekine program približevanja, si zapomni pozicijo, ko se je prekinitev sprožila, ter z zmanjšano hitrostjo nadaljuje z odlaganjem za 100 mm nižje.
Struktura ABB-jevih programov na krmilnikih IRC5 je sestavljena iz »Tasks«, pri čemer je lahko samo en »Task« aktiven oz. se izvaja. Če želimo imeti več takšnih programov, ki potekajo vzporedno, potrebujemo na krmilniku opcijo »MultiTasking«, kar pa je mogoče samo z doplačilom. Slika 9.1 prikazuje razdelitev programa na krmilniku IRC5. Vsak »Task« ima svoje programske module, ki pa jih je lahko več. Vsak modul vsebuje najprej podatke ter posamezne rutine. Od tega mora en modul – po navadi je to prvi modul – vsebovati rutino z imenom »main«, saj to predstavlja »Entry routine« oz. rutino, kjer se bo program začel izvajati. Robotski program ima programski kazalec, ki kaže na trenutni ukaz, ki se bo izvedel, ter kazalo robota, kjer se robot trenutno nahaja. Povedano drugače, programski kazalec kaže, kam bo šel robot oz. kateri del programa se bo izvedel. Slika 9.2 prikazuje primer uporabe programskega kazalca in kazalca robota. Programski kazalec vedno kaže nekaj ukazov naprej, da se izvajanje programa in/ali izvajanje gibanja robota nemoteno izvaja.

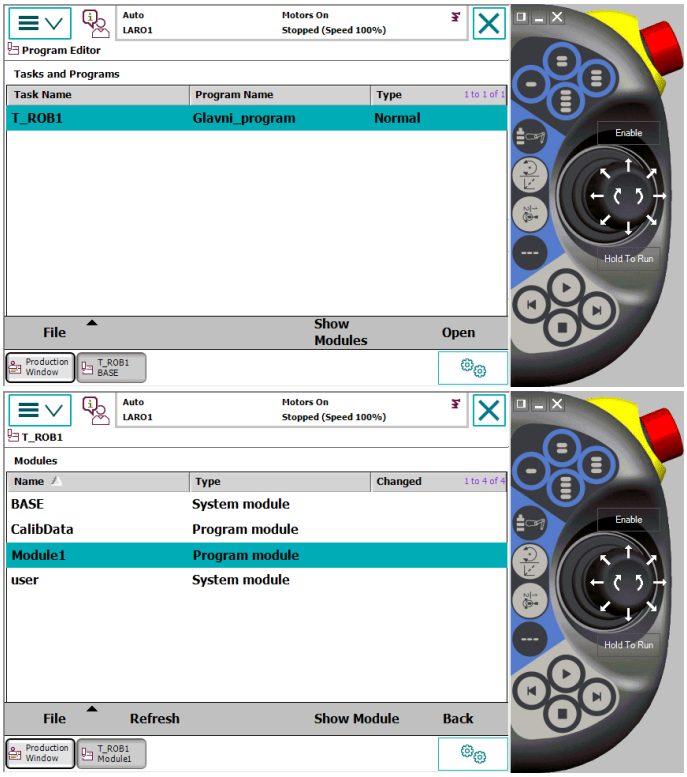


Slika 9.1: Shematska razdelitev programa na krmilniku IRC5

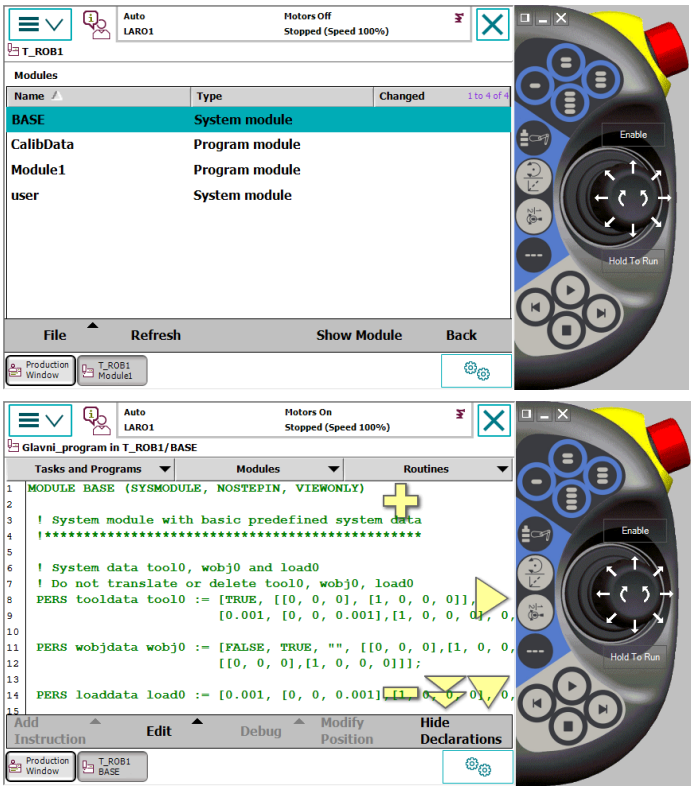


Slika 9.2: Programski kazalec in kazalec robota

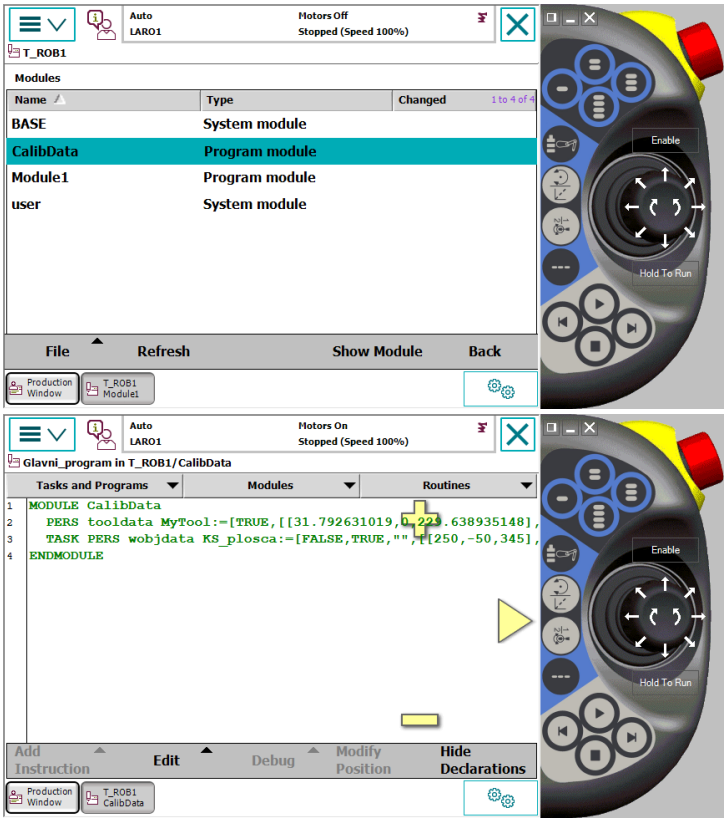
Program vsebuje tudi sistemske module »System module«, ki vsebujejo podatke o robotu, kot je npr. podatek, kje se nahaja osnovni koordinatni sistem delovnega objekta »wobj0« ter kje se nahaja TCP-robota na prirobnici brez orodja. Slika 9.3 prikazuje razdelitev programa na krmilniku robota. Ti moduli se ne shranjujejo avtomatsko, ko shranimo program, kar pomeni, da bo kakršnakoli sprememba sistemskega modula imela vpliv na program, napisan v programskem modulu. Ne glede na to, kateri program je naložen, so sistemski moduli nenehno prisotni v spominu robotskega krmilnika.



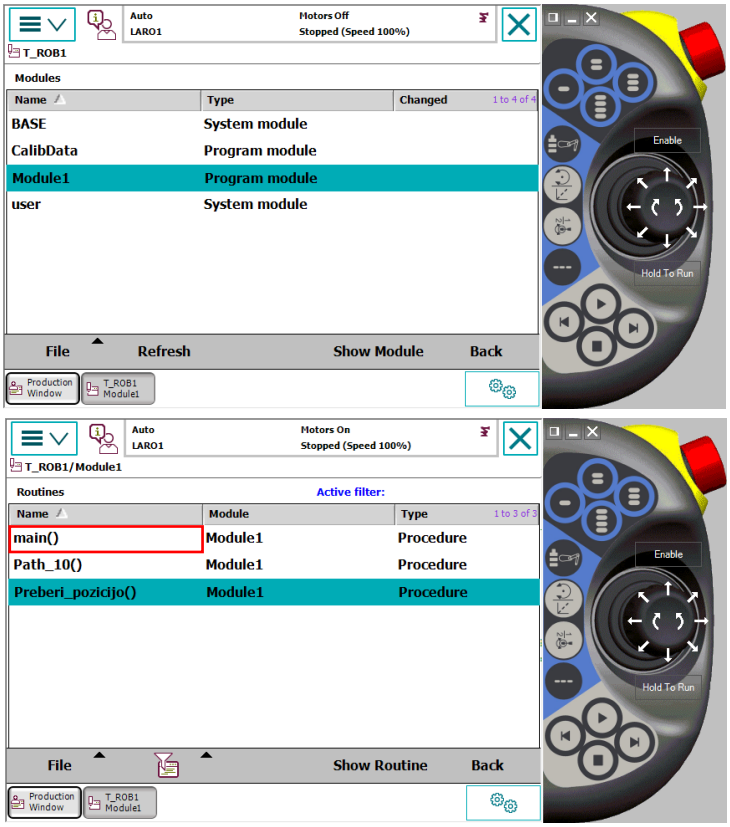
Slika 9.3: Razdelitev programa na krmilniku IRC5



Slika 9.4: Razdelitev programa na krmilniku IRC5-BASE



Slika 9.5: Razdelitev programa na krmilniku IRC5-CalibData



Slika 9.6: Razdelitev programa na krmilniku IRC5-Module1, kjer se nahaja kreiran program.

Program je namenoma razdeljen na več sklopov oz. modulov ter rutin, saj s primernim programiranjem naredimo še tako kompleksen program zelo pregleden. Vzemimo na primer program prijemanja in odlaganja. Zelo preprost program, ki ga lahko napišemo v enem modulu s tremi rutinami. Glavno rutino oz. »main« ter dvema podrutinama, prijemanje in odlaganje, kot je prikazano v poglavju 9.3. Preden razčlenimo posamezne dele programa, je potrebno definirati možne gibe, ki jih lahko robot opravi, ter kako so definirani.

9.2 Sintaksa ukazov za izvajanje gibanja

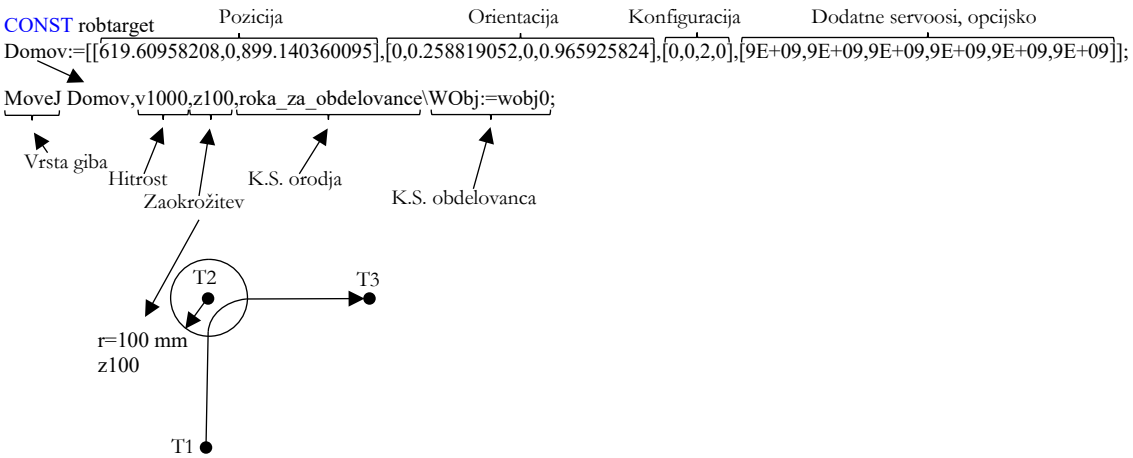
Kot je bilo že rečeno, ima vsak proizvajalec industrijskih robotov svojo sintakso. Industrijski robot lahko izvaja PTP »Point-to-Point« gibe. To so gibi, kjer se robot premakne iz trenutne točke v naslednjo, pri čemer izbere najhitrejšo možno trajektorijo. Po navadi je ta gib neka krivulja oz. zaokrožitev. Ta gib ni vnaprej predvidljiv zato tukaj velja opozorilo, da ga uporabljamo samo v primeru, kadar ne more priti do kolizije med robotom in okolico. ABB-jeva sintaksa teh gibov je »MoveJ«.

Naslednji gib, ki ga poznamo, je linearni gib. Tukaj gre za premaknitev robota iz trenutne pozicije v naslednjo po točno določeni ravni liniji. Ta gib je počasnejši, ker zahteva usklajeno gibanje vseh šestih osi hkrati. ABB-jeva sintaksa teh gibov je »MoveL«.
Za izvajanje krožnih gibov imamo na voljo ukaz »MoveC«, pri čemer ukaz vsebuje dve točki: srednjo točko krožnice ter končno točko krožnice. Prva točka je definirana s prejšnjim ukazom, ki pa je lahko »MoveJ«, »MoveL« ali »MoveC«. Tukaj velja opozorilo, da robot z enim ukazom ne more izvesti krožnega giba, večjega od 270 °.

Vsi ukazi gibanja, ki so bili do sedaj predstavljeni, so definirani glede na trenutno uporabljeno orodje in koordinatni sistem. Če ne želimo, da je določena pozicija robota vezana na kakršen koli koordinatni sistem, uporabimo ukaz »MoveAbsJ«, ki trenutno pozicijo shrani glede na posamezne zasuke motorja. Takšne inštrukcije najpogosteje uporabljamo za definiranje varne oz. »Home« pozicije, saj se ta s spreminjanjem koordinatnih sistemov ne spreminja.

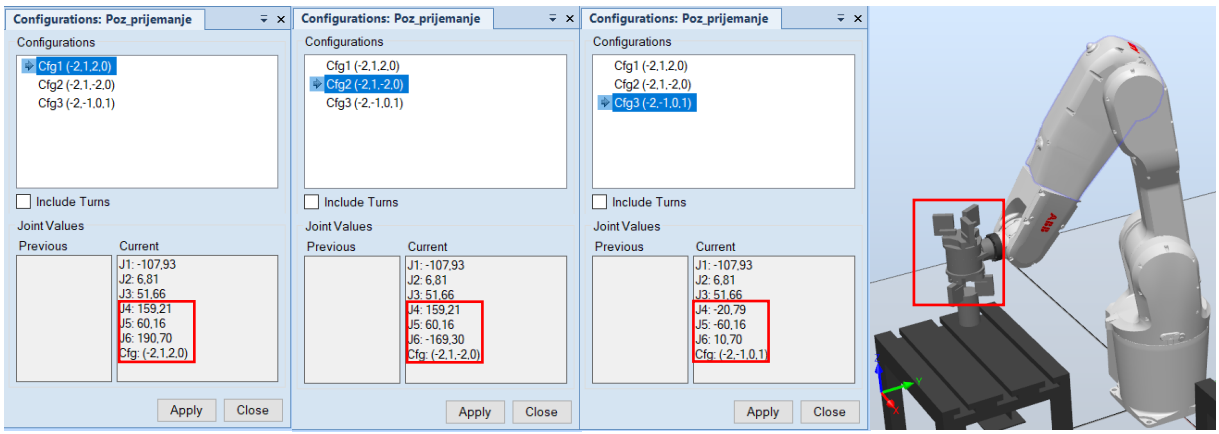
Vsak ukaz za izvajanje gibanja vsebuje vrsto giba, pozicijo in orientacijo željene točke, hitrost, s katero želimo doseči to točko, kako natančno jo želimo doseči, s katerim koordinatnim sistemom orodja ter v katerem koordinatnem sistemu se ta točka nahaja. Sledi prikaz ukaza za gibanje robota. Vsaka točka v prostoru, ki jo določimo, je lahko točka prehoda ali točka zaustavitve. Pri prehodnih točkah se robot med izvajanjem tej točki

približa in potem začne zavijati proti novi točki. Temu zavijanju pravimo zaokrožitev oz. »Zone data«. Ta vrednost predstavlja radij zaokrožitve. Slika 9.7 prikazuje sestavo ukaza gibanja in podaja razlago za zaokrožitev v določeni točki.



Slika 9.7: Razčlenitev ukaza gibanja in zaokrožitve v določeni točki

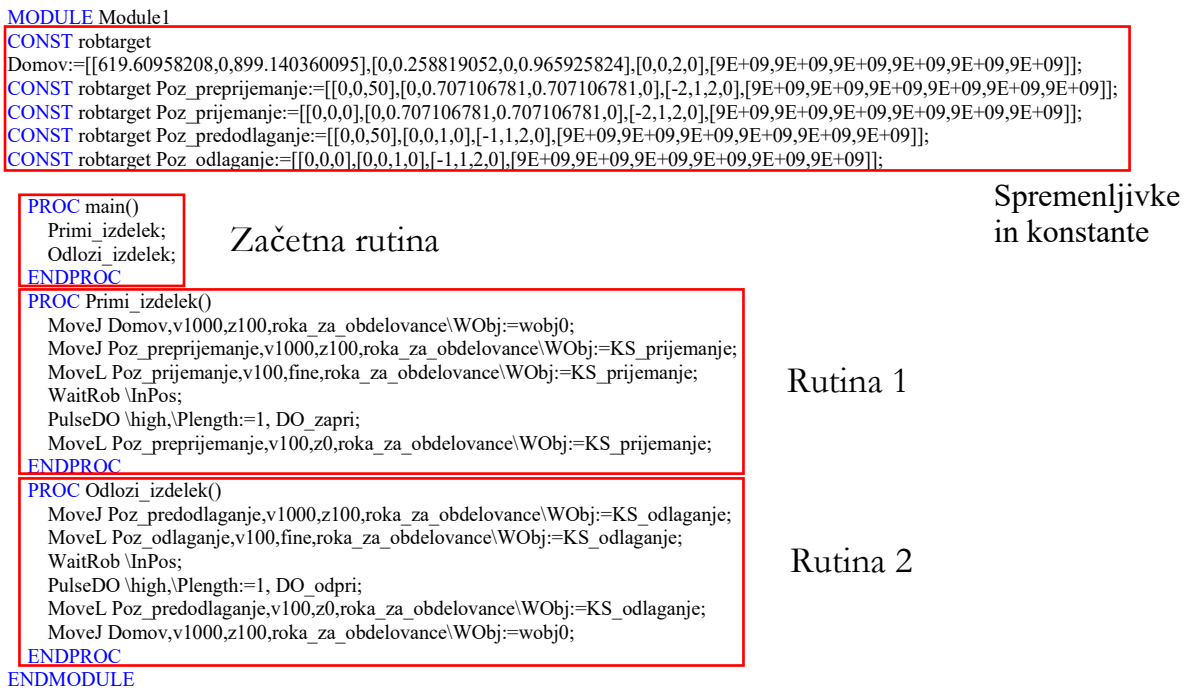
Pri kreiranju točk v prostoru, kjer ima točka podatek o poziciji in orientaciji uporabljenega orodja, se poleg tega shrani tudi konfiguracija industrijskega robota. Konfiguracija robota v določeni točki pomeni točno določeno postavitev sklepov osi 1, 4, 5 in 6. V določeni točki se lahko zgodi, da ima robot na voljo več možnih postavitev sklepov, torej imamo na voljo več konfiguracij robota. Pri programiranju v programu RobotStudio konfiguracija robota ni vedno natančno določena. Tako moramo izbrati pravo konfiguracijo. Slika 9.8 prikazuje več možnih konfiguracij postavitve robota v programu RobotStudio. V tej konkretni točki imamo na voljo 3 možne konfiguracije, kjer se spreminja postavitev 4., 5. in 6. osi, pri čemer se pozicija in orientacija prijemala **ohranjata**.



Slika 9.8: Primer izbire konfiguracije postavitve robota v eni točki

9.3 Zgradba posameznega modula

Znotraj »Tasks« lahko imamo več modulov, a samo eden vsebuje vstopno rutino oz. »Entry routine«, ki mora biti poimenovana z »main«. Ta rutina definira začetek programa. Robotski program na krmilniku vedno zaženemo z ukazom »PP to Main«. S tem postavimo programski kazalec na vstopno rutino, iz katere se program začne varno izvajati. Tukaj velja podati opozorilo. Ne glede na to, kje se robot nahaja, bomo z ukazom »PP to Main« povzročili, da se bo robot pomaknil na prvo točko programa. Pri tem moramo biti pozorni, da ta gib **ne povzroči kolizije**, saj je **po navadi prvi gib programa definiran kot PTP**. Zato je v tem primeru smiselno uporabiti preverjanje, ali se robot pri ukazu »PP to Main« nahaja v željeni točki ali ne. V nasprotnem primeru je potrebno napisati program varnega vračanja robota na začetno točko izvajanja programa iz poljubne točke v prostoru. Ta miselni proces in preprost program bosta predstavljena v poglavju 11.4. Slika 9.9 prikazuje zgradbo posameznega modula. Ta je sestavljen iz imena modula, podatkov, ki jih potrebujemo v tem modulu, (po navadi so to definirane pozicije, spremenljivke ali konstante), iz vstopne oz. glavne rutine in iz posameznih rutin, ki jih kličemo iz glavne rutine.

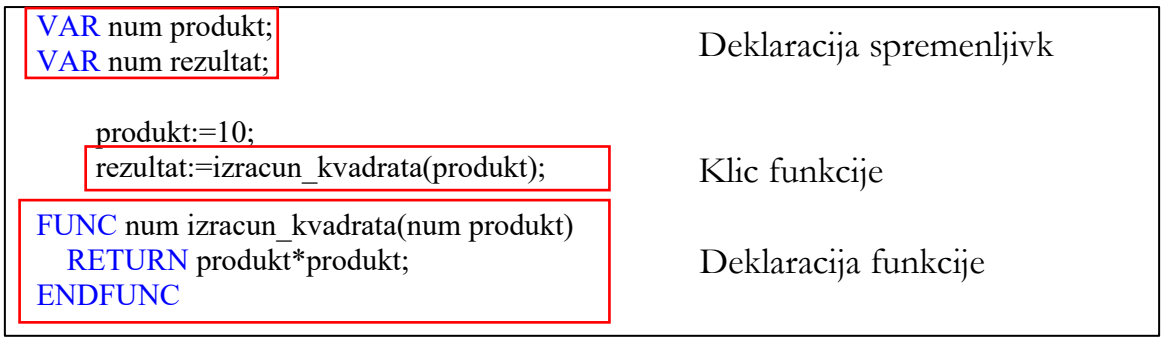


Slika 9.9: Zgradba posameznega modula

9.4 Vrste rutin

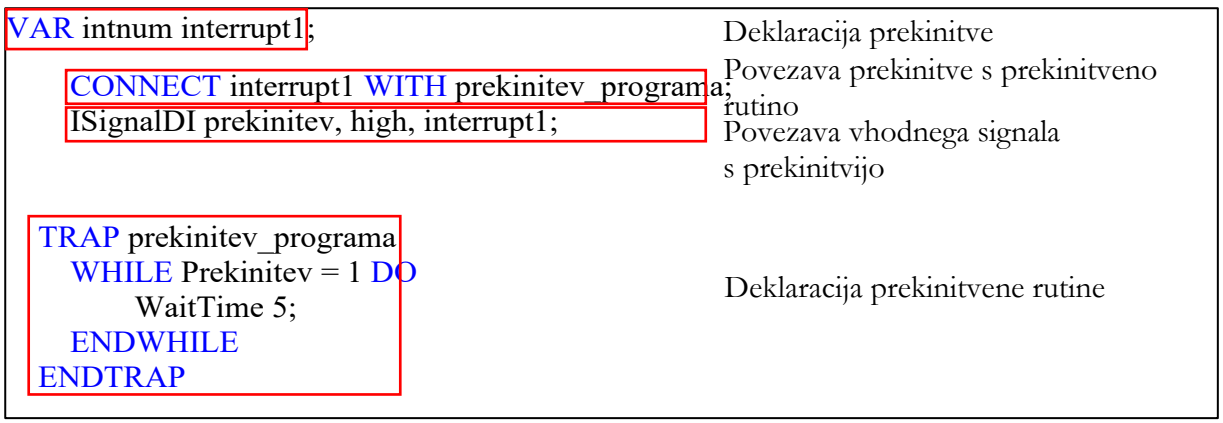
Kot smo že dejali, potrebujemo vstopno rutino oz. glavno rutino, iz katere zaženemo robotski program. Rutina je lahko definirana na tri različne načine: kot procedura oz. »procedures«, to je standardna rutina, ki vsebuje tudi ukaze gibanja robota. Slika 9.9 prikazuje primer vhodne rutine.

Rutina je lahko definirana tudi kor funkcija, kjer robot izvede določeno operacijo na vhodnih podatkih, kot je množenje, deljenje ipd. ter vrne rezultat. V tej funkciji ni mogoče izvajati ukazov gibanja robota. Slika 9.10 prikazuje primer uporabe funkcijske rutine.



Slika 9.10: Primer uporabe funkcijske rutine

Velikokrat se dogaja, da moramo prekiniti izvajanje trenutnega programa, saj se lahko zgodi, da je npr. zmanjkalo komprimiranega zraka. V tem primeru bomo uporabili prekinitveno rutino oz. »Trap routine«. Prekinitvena rutina je povezana s prekinitvami oz. »Interrupts«, ki se lahko zgodijo kadarkoli. Vsaka prekinitev ima svojo številko, ki ji jo dodeli krmilnik sam. Je pa prekinitev potrebno kreirati oz. deklarirati in to prekinitev povezati s prekinitveno rutino. Prav tako je potrebno definirati, kateri signal predstavlja prekinitev. Slika 9.11 prikazuje primer uporabe DI za prekinitev.



Slika 9.11: Primer uporabe prekinitvene rutine

9.5 Vrste podatkov »Data objects«

Znotraj programa lahko imamo definiranih več različnih podatkov. Ti so lahko konstantni »constants«, spremenljivi »variables« in persistentni oz. globalni »persistent«. Vrednost konstant ne moremo spreminjati znotraj programa, ko jim določimo vrednost v deklaraciji. Spremenljive podatke lahko znotraj programa spreminjamo poljubnokrat. Ti podatki so vedno znova naloženi ob vsakem zagonu programa. Podatki z oznako »persistent« po navadi definirajo orodja in koordinatne sisteme. Ti ohranjajo svojo vrednost tudi, če spremenimo program in so na voljo v vseh »Tasks«. Slika 9.12 prikazuje vrsto podatkov.

```
MODULE Spremenljivke
        CONST robtarget Poz_odlaganje:=[[0,0,0],[0,0,1,0],[-1,1,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        PERS tooldata roka_za_izdelke:=[TRUE,[[-93,0,59],[0,-0.707106781,0,0.707106781]],[3,[0,0,1],[1,0,0,0],0,0,0]];
        TASK PERS wobjdata KS_prijemanje:=[FALSE,TRUE,"",[[-161.738,-360.073,308],[1,0,0,0]],[[61.738,-
        89.927,80],[1,0,0,0]]];
        VAR num rezultat;
        VAR intnum interrupt1;
ENDMODULE
```

Slika 9.12: Vrste podatkov

9.6 Življenjska doba podatkov

Od tega, kje definiramo podatke, je odvisna njihova »življenjska« doba; z drugimi besedami – trajanje shrambe podatka v pomnilniku krmilnika. Slika 9.13 prikazuje različne načine definiranja podatkov ter njihovo dobo shranjevanja v pomnilniku krmilnika. Če je podatek definiran v glavi modula, bo ta podatek na voljo skozi ves »Task«, ki je trenutno aktiven. Če je podatek izrecno definiran kot »local«, bo veljal samo za ta modul. Če pa je podatek definiran znotraj rutine, bo aktiven in veljaven zgolj v definirani rutini.

```
MODULE Module2
   !Veljavnost spodnje spremenljivke je globalna in je dostopno znotraj celotnega programa v
trenutnem TASK-u
   VAR num global_var:=321;
   !Veljavnost spodnje spremenljivke je lokalna in je dostopna samo znotraj Module2
   LOCAL VAR num local_var:=123;

   PROC main2()
   !Veljavnost spodnje spremenljivke je lokalna in je dostopna samo znotraj rutine main2
   VAR num local_var2:=456;

   ENDPROC
ENDMODULE
```

Slika 9.13: Trajanje shrambe podatka v pomnilniku

9.7 Podatkovni tipi »Data types«

Pri deklaraciji podatkov ni dovolj, da povemo, ali bo ta podatek konstanta, spremenljivka, ali pa bo ta podatek persistentni, temveč moramo programu povedati, kakšen podatkovni tip je. Podatek je lahko samo eden, lahko pa je sestavljen iz več podatkov, pri čemer so ti deli lahko samo številke ali pa so to lahko tudi črke. Pri tem poznamo enostavne podatkovne tipe, kot so cela ali realna števila, ki so označena kot »num« ali pa »dnum«, če potrebujemo večje vrednosti. Med enostavne podatkovne tipe spadajo še podatki za logično vrednost oz. »Boolean«, ki so definirani kot »True« in pa »False«, ter podatki z nizom znakov oz. »String«. Tabela 9.1 prikazuje enostavne podatkovne tipe in njihovo deklaracijo.

Tabela 9.1: Enostavni podatkovni tipi

Podatkovni tip	Vrednost	Deklaracija	Primer
num	+/-8388607	VAR num stevec;	stevec:= 150;
dnum	+/- 4503599627370496	VAR dnum vrednost;	vrednost:= 1E+10
bool	True/False	VAR bool aktivno;	avtivno:= False;
string	0-80 znakov	VAR string ime;	ime:= "Bruce Wayne"

Podatkovni tipi so lahko sestavljeni tudi iz več podatkov. Tabela 9.2 prikazuje nekaj primerov sestavljenih podatkovnih tipov. Pri tem sta najosnovnejša podatkovna tipa podatek o poziciji oz. »pos«, ki vsebuje tri podatke, ter podatek o orientaciji oz. »orient«, ki pa vsebuje štiri podatke, ker ABB-jev krmilnik operira s kvaternioni. Naslednji takšen podatek je podatek o pozi oz. »pose«, ki vsebuje tako pozicijo in orientacijo, torej je v bistvu sestavljen iz obeh prejšnjih podatkovnih tipov. Naslednji tak primer sestavljenih podatkov so podatki o orodju oz. »tooldata«, o koordinatnih sistemih oz. »wobjdata« in pa o točki v prostoru, ki je definirana kot »robtarget« in vsebuje podatke o poziciji, orientaciji, konfiguraciji in pa podatek o dodatnih servooseh, če so na voljo. Slika 9.7 prikazuje primer uporabe točke v prostoru ter podaja razčlenitev posameznega dela sestavljenega tipa podatka.

Tabela 9.2: Sestavljeni podatkovni tipi

Sestavljeni podatkovni tip	Podatki	Podatkovni tip		Deklaracija	Primer 1	Primer 2
pos (pozicija)	x, y, z	num		VAR pos t1;	t1:=[10,10,10]	t1.x:=10;
orient (orientacija)	q1,q2,q3,q4	num		VAR orient o1;	o1:=[0,1,0,1];	o1.q1:=-1;
pose (pozicija in orientacija)	trans	pos	num	VAR pose p1;	p1:=[[10,10,10],o 1]	P1.trans:=t1;
	rot	orient	num			

9.8 Programske zanke, odločitve in matrike

Pri programiranju industrijskih robotov se večkrat poslužujemo raznih zank in odločitev. Še posebej, če imamo kompleksen program. Standardni zanki, ki se najpogosteje uporabljata v robotiki, sta FOR in WHILE. Pri tem gre za podobni, a vendarle različni zanki. Pri FOR zanki to zanko ponavljamo tolikokrat, kolikorkrat smo definirali. Torej, če smo definirali deset ponovitev, se bo zanka desetkrat ponovila. Na drugi strani se zanka WHILE ponavlja tako dolgo, dokler pogoj na začetku zanke ne bo izpolnjen. Ko je ta pogoj izpolnjen, se zanka prekine. Slika 9.14 prikazuje primer zanke WHILE in FOR. Na koncu izvedbe programa bosta obe spremenljivki, pri tem konkretnem primeru, imeli isto vrednost.

```
VAR num st_ponovitev_w;
VAR num st_ponovitev_f;
VAR num real_st;
PROC main()
  WHILE st_ponovitev_w<5 DO
    st_ponovitev_w:=st_ponovitev_w+1;
  ENDWHILE

  FOR i FROM 1 TO 5 DO
    st_ponovitev_f:=st_ponovitev_f+1;
  ENDFOR
ENDPROC
```

Slika 9.14: Primer zanke WHILE in FOR

Najpogosteje uporabljena odločitev je s pomočjo IF in SWITCH vejitve. IF vejitev uporabljamo pri navadnih odločitvah, kadar imamo na voljo dve možnosti ali več. IF vejitev se lahko pojavi tudi znotraj IF vejitve. Če imamo na voljo več možnosti, potem raje izberemo SWITCH vejitev, saj bo v tem primeru program bolj pregleden. Slika 9.15 prikazuje primer IF vejitve.

```
IF st_ponovitev_w<1 THEN
  WHILE st_ponovitev_w<5 DO
    st_ponovitev_w:=st_ponovitev_w+1;
  ENDWHILE
ELSE
  FOR i FROM 1 TO 5 DO
    st_ponovitev_f:=st_ponovitev_f+1;
  ENDFOR
ENDIF
```

Slika 9.15: Primer uporabe IF vejitve

Če želimo uporabiti SWITCH vejitev na krmilniku ABB, moramo uporabiti njihovo sintakso. ABB za to uporablja TEST … ENDTEST sintakso, znotraj katere so potem definirani CASE 1…n. Slika 9.16 prikazuje primer SWITCH vejitve.

Pogosto se dogaja, da poleg vejitev in odločitev potrebujemo tudi polja z določenimi vrednostmi. Tem poljem pravimo tudi matrike, strokovni izraz v angleščini pa je »Array«. Pri tem je lahko polje enodimenzionalno ali večdimenzionalno, odvisno od tega, kaj potrebujemo. Enodimenzionalno polje ima npr. samo eno vrstico in tri stolpce. Dvodimenzionalno polje ima najmanj dve vrstici in dva stolpca. Tridimenzionalno polje pa je sestavljeno iz najmanj dveh dvodimenzionalnih polj.

Enodimenzionalno polje matrika $\{1,3\}$ - $[1 \quad 2 \quad 3]$

Dvodimenzionalno polje matrika $\{2,2\}$ - $\begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix}$

Tridimenzionalno polje matrika $\{3,3,2\}$ - $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$

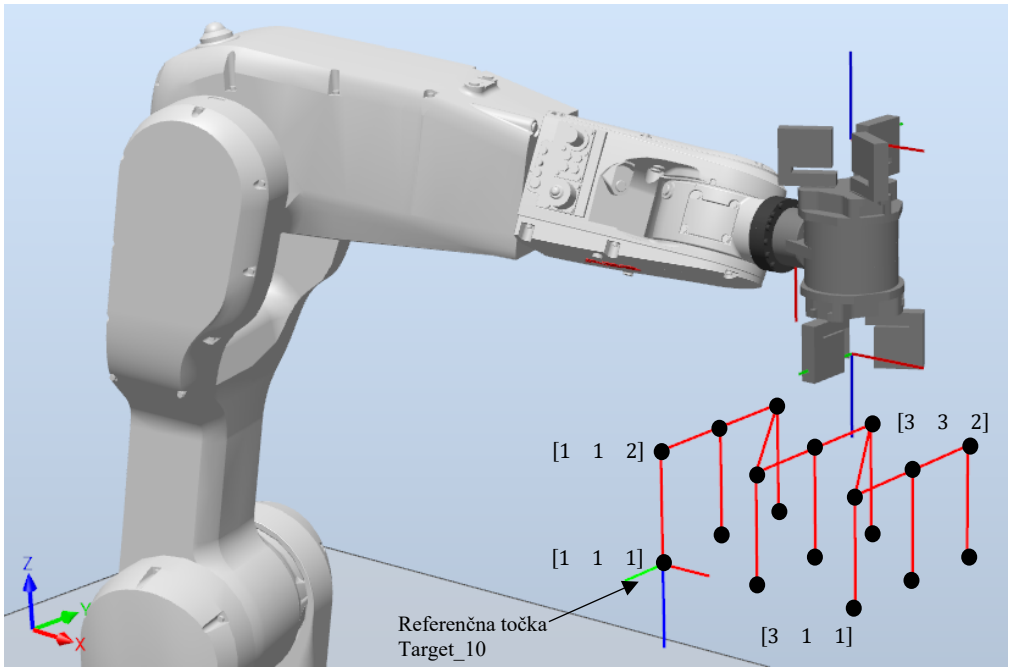
$m * n * 1$

$m * n * 2$

```
TEST program
CASE 1:
  IF st_ponovitev_w<1 THEN
    WHILE st_ponovitev_w<5 DO
      st_ponovitev_w:=st_ponovitev_w+1;
    ENDWHILE
  ELSE
    FOR i FROM 1 TO 5 DO
      st_ponovitev_f:=st_ponovitev_f+1;
    ENDFOR
  ENDIF
CASE 2:
  IF st_ponovitev_f<1 THEN
    FOR i FROM 1 TO 5 DO
      st_ponovitev_f:=st_ponovitev_f+1;
    ENDFOR
  ELSE
    WHILE st_ponovitev_w<5 DO
      st_ponovitev_w:=st_ponovitev_w+1;
    ENDWHILE
  ENDIF
DEFAULT:
  st_ponovitev_f:=1;
  st_ponovitev_w:=1;
ENDTEST
```

Slika 9.16: Primer SWITCH vejitve

Slika 9.17 prikazuje točke pozicije, ki jih želimo kreirati glede na referenčno točko. Slika 9.18 prikazuje primer željenega programa s tridimenzionalno matriko, ki vsebuje 18 spremenljivk (3 x 3 x 2) tipa »robtarget«. Pozicije so ustvarjene s pomočjo treh FOR zank, kjer referenčno točko ustrezno zamikamo. Po uspešnem kreiranju pozicij pa te točke kličemo v dveh FOR zankah. Globino pomikanja robota določamo kar z ustrezno številko tretje dimenzije v matriki. Druga matrika je zamaknjena za 100 mm navzgor nad referenčno točko. Slika 9.18 prikazuje primer programa, kjer s pomočjo treh FOR zank kreiramo matriko točk pozicije glede na referenčno točko. Te točke lahko potem kličemo in izvajamo linearne gibe.



Slika 9.17: Željene točke pozicije

```
VAR robtarget tocke{3,3,2};        Definicija matrike {3,3,2}.
CONST robtarget
Target_10:=[[552.577457107,0,564.973066959],[0,1,0,0],[0,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

PROC main()
   matrika_3_3_2;
   izvedba_matrike_3_3_2;
ENDPROC

PROC matrika_3_3_2()
   FOR i FROM 1 TO 2 DO ! Globina matrike
      FOR j FROM 1 TO 3 DO    ! Vrstica matrike
         FOR k FROM 1 TO 3 DO    ! Stolpec matrike
            tocke{j,k,i}:=Offs(Target_10,100*(j-1),80*(k-1),100*(i-1));
         ENDFOR
      ENDFOR
   ENDFOR
ENDPROC

PROC izvedba_matrike_3_3_2()
      FOR j FROM 1 TO 3 DO
         FOR k FROM 1 TO 3 DO
            MoveL tocke{j,k,2},V100, fine,Prijemalo_2\WObj:=wobj0;
            MoveL tocke{j,k,1},V100, fine,Prijemalo_2\WObj:=wobj0;
            MoveL tocke{j,k,2},V100, fine,Prijemalo_2\WObj:=wobj0;
         ENDFOR
      ENDFOR
ENDPROC
```

Slika 9.18: Primer kreiranja točk pozicije glede na referenčno točko s tridimenzionalno matriko

9.9 Vprašanja za utrjevanje snovi

1. V programu RobotStudio kreirajte nov robotski sistem z robotom IRB 1200 5/0.9. V tem primeru ne potrebujemo nobenih dodatnih opcij. Z ukazom »Import Library« uvozite orodje »MyTool« ter ga namestite na prirobnico robota. Ustvarite referenčno točko oz. prvo inštrukcijo programa ter program prenesite na virtualni krmilnik z ukazom »Synchronize to RAPID«. V zavihku »RAPID« v drevesni strukturi odprete »RAPID« -> »Module1« in z dvojnim klikom na »main« se odpre robotski program. Program dopolnite z naslednjimi spremenljivkami.

Vrsta spremenljivke	Ime spremenljivke
Lokalno celo število	st_produktov
Realno število	cena
Logična vrednost	napaka
Niz znakov (string)	zapis_napake

Ugotovitve:

2. Napišite funkcijo, ki bo izračunala vrednost produktov tako, da število produktov pomnožite s ceno 5.3 €. To funkcijo pokličite in izračunajte vrednost. Potem preverite, če je cena vseh produktov manjša od 53 €. Če je cena manjša od 53 €, je potrebno spremenljivko »napaka« označiti za pravilno oz. postaviti na ena. S pomočjo ukaza »TPWrite« izpišite niz znakov »zapis_napake«, ki naj ima vrednost »Cena izdelka je prenizka!« na zaslon učne konzole. Pri tem si pomagajte z navodili »Techical reference manual – RAPID overview«, ki so dostopna preko programa RobotStudio.

Ugotovitve:

3. S pomočjo FOR zank in referenčne točke ustvarite matriko pozicij (3 x 3 x 3), ki bodo po osi X zamaknjene za 80 mm, po osi Y za 120 mm in po Z za 50 mm. Pri tem si pomagajte s prej opisanimi navodili. Pazite, da bo orientacija orodja v referenčni točki navpično navzdol.

Ugotovitve:

4. S pomočjo FOR zank pripravite izvajanje gibanja robota z linearnimi gibi, da bo obšel vse točke v prej kreirani matriki pozicij tako, da bo začel pri najvišji in potem šel do najnižje in nazaj gor ter se pomaknil v naslednjo vertikalo. S pomočjo ukaza »TCPTrace«, ki se nahaja pod zavihkom »Simulation«, vklopite sledenje TCP-točki ter opazujte izvajanje gibanj robota. Prav tako s pomočjo ukaza »TPWrite« in »TPErase« pripravite izpis, v kateri vrstici, stolpcu in globini matrike se trenutno nahaja robot. Pri tem si pomagajte z navodili »Techical reference manual – RAPID overview«, ki so dostopna preko programa RobotStudio.

Ugotovitve:

ZAPISKI

10 Uporaba PROFINET komunikacije za pošiljanje podatkov

V industrijskem okolju robot komunicira z nadzornim sistemom, s katerim si izmenjuje razne podatke. V večini primerov gre za preproste podatke, kot je to bit. Če želimo robotu sporočiti številko programa, ki ga mora izvesti, pa en bit ni več dovolj, temveč potrebujemo več bitov, da lahko dobimo pozitivno celoštevilčno vrednost. Prav tako lahko dobimo zahtevo, da robot pošilja trenutno pozicijo PLK-ju, ki jo ta zapisuje na HMI-zaslon. Trenutna pozicija robota ni več celo število, temveč je realno število z desetinkami ali celo stotinkami.

V nadaljevanju bo predstavljen princip pošiljanja trenutne pozicije industrijskega robota preko PROFINET komunikacijskega protokola do drugega robota, ki bo prebral poslane podatke, jih zapisal v podatkovni tip za pozicijo in izvedel zahtevan gib.

10.1 Bitna logika

Pri komunikaciji preko PROFINET protokola uporabljamo bitno logiko oz. dvojiški sistem, kar pomeni, da lahko naslavljamo posamezni bit ali več bitov skupaj. Pri tem je en bit najmanjši podatek, ki ga lahko pošljemo. Bit ima lahko samo dve stanji oz. vrednosti, to je 0 ali 1. PROFINET protokol nam omogoča naslavljanje 2048 vhodnih in 2048 izhodnih bitov. Osem združenih bitov nam predstavlja bajt ali zlog oz. »byte«. Najvišja vrednost enega bajta v celoštevilčnem zapisu tako znaša 255, saj 0 predstavlja že eno

vrednost. Dva združena bajta ali zloga, torej 16 bitov, pa nam predstavlja »besedo« oz. »word«, katere vrednost znaša 65535, saj 0 predstavlja že eno vrednost. Pri preračunu vrednosti iz bitnega oz. dvojiškega zapisa v desetiški je pomembno, da vemo, kje se nahaja najmanj pomembni bit oz. »LSB-Least significant bit« ter kje se nahaja najpomembnejši bit oz. »MSB-Most significant bit«. Po navadi se LSB nahaja na desni strani zapisa bajta. Ta podatek je na voljo v navodilih proizvajalca. Če dva stroja nimata LSB na isti strani, je potrebno podatek, ki se pošilja, obrniti na strani prejemnika, saj se v nasprotnem primeru poslani in prejeti podatek ne ujemata. Slika 10.1 prikazuje primer bita, bajta in besede ter podaja vrednost LSB in MSB. Pri zamenjavi lahko prejemnik prejme namesto vrednosti 1 vrednost 128. Slika 10.2 podaja primer izračuna posameznih vrednosti iz bitne vrednosti v desetiško.

0 / 1 – bit

0 0 0 0 0 0 0 0 – bayt (vrednost od 0 – 255)

0 0 0 0 0 0 0 1 = 0 0 0 0 0 0 0 2^1 = 1 - LSB

1 0 0 0 0 0 0 0 = 2^7 0 0 0 0 0 0 0 = 128 - MSB

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 – word –(vrednost od 0 – 65535)

Slika 10.1: Primer bita, bajta in besede ter LSB in MSB

Izračun posameznih vrednosti:

0 0 0 0 0 0 0 1 = 0 0 0 0 0 0 0 2^1 = 1

0 0 0 0 0 0 1 0 = 0 0 0 0 0 0 2^2 0 = 2

0 0 0 0 0 1 0 0 = 0 0 0 0 0 2^3 0 0 = 4

0 0 0 0 0 1 1 1 = 0 0 0 0 0 2^3 2^2 2^1 = 7

Slika 10.2: Primer pretvorbe iz bitne v desetiško vrednost

10.2 Uporaba negativnih števil s pomočjo bitne logike

V prejšnjem poglavju smo spoznali, kako deluje bitna logika za pošiljanje podatkov preko PROFINET komunikacije. Pri tem se pozitivno celo število pretvori v bitno število in pošlje po komunikaciji do prejemnika, ki mora to bitno število zopet pretvoriti nazaj v pozitivno celo število. Kako pa sedaj pošiljamo negativna števila preko PROFINET komunikacije?

10.2.1 Dvojiški komplement

Če želimo poslati negativno število preko PROFINET komunikacije, moramo uporabiti tako imenovani dvojiški komplement za zapis negativnih celih števil s pomočjo bitne logike [47, 48]. Pri dvojiškem komplementu si izberemo en bit za predznak. Npr., če imamo na voljo en bajt, potem imamo na voljo števila od 0 do $(2^{8-1}-1)=(2^7-1)=127$ za pozitivna števila ter od $-2^{8-1}=-2^7=-128$ do -1 za negativna cela števila. Skupaj to znaša ravno 255. Gledano iz bitne logike števila od 0–127 predstavljajo pozitivna števila, števila od 128–255 pa negativna števila, pri čemer število 128 predstavlja -128 in število 255 predstavlja -1. Tabela 10.1 prikazuje osembitni zapis negativnih števil in dvojiški komplement enega bajta.

Tabela 10.1: Osembitni zapis negativnih števil

Bitna vrednost	Dvojiški komplement	Celoštevilčni zapis
0000 0000	0	0
0000 0001	1	1
…	…	…
0111 1110	126	126
0111 1111	127	127
1000 0000	-128	128
1000 0001	-127	129
1000 0010	-126	130
…	…	…
1111 1110	-2	254
1111 1111	-1	255

Pretvorba negativnega števila v bitni zapis tako poteka po naslednjem postopku za dvojiški komplement. Najprej pretvorimo število v dvojiški zapis, potem invertiramo 1 v 0 in 0 v 1, nato pa dobljenemu bitnemu zapisu prištejemo vrednost 1. Pretvorimo npr. negativno celo število -127 v bitni zapis.

Najprej pretvorimo število 127 v bitni zapis:

127→0111 1111

Dobljeni bitni zapis negiramo:

0111 1111
1000 0000

Negiranemu bitnemu zapisu prištejemo 1:

1000 0000 + 1=1000 0001

Pretvorimo nazaj v desetiški zapis:

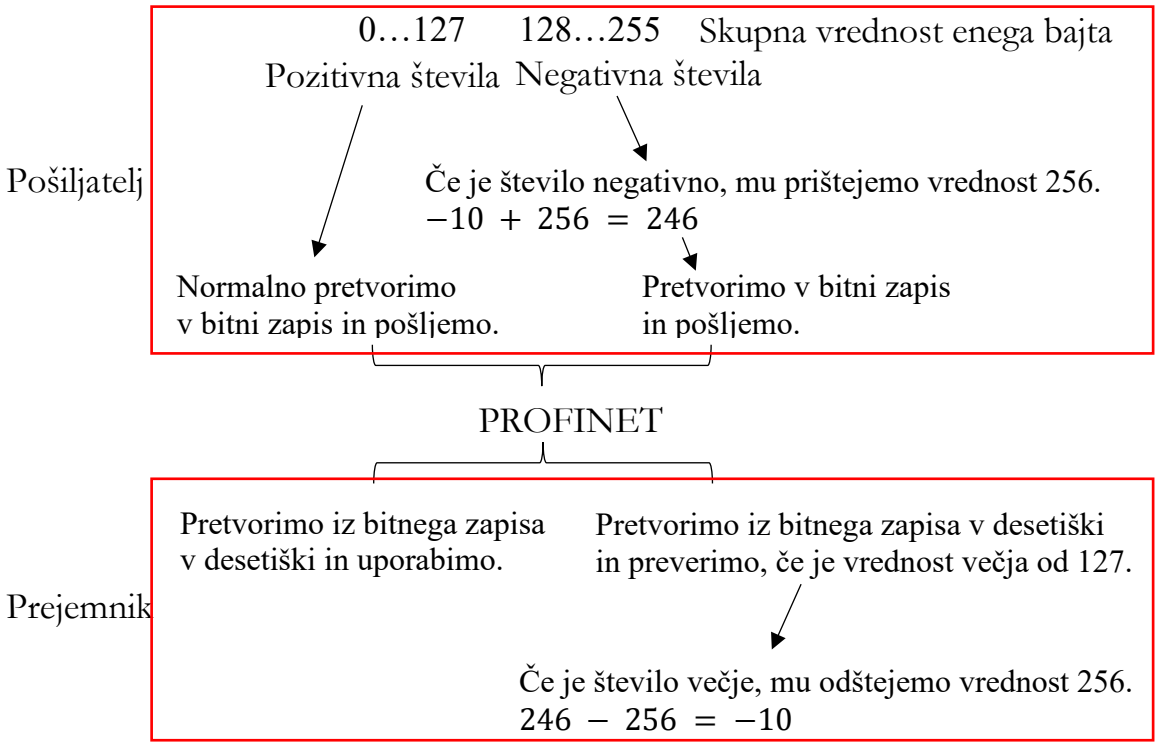
1000 0001→128

Tako dobimo, da negativno celo število -127 v bitnem zapisu predstavlja število 128, kar je lepo razvidno v tabeli 10.1.

10.2.2 Uporaba v praksi

En bajt ima vrednost od 0 do 255, torej skupaj 256 možnosti. Ta bajt razdelimo in tako dobimo 128 možnosti za pošiljanje pozitivnih števil od 0 do 127 in 128 števil za pošiljanje negativnih števil od -1 do -128. Vrednosti, ki so pozitivne, normalno pretvorimo in pošljemo. Vrednostim, ki so negativne, pa na pošiljateljevi strani prištejemo vrednost $2^8 = 256$. S tem negativno vrednost pripravimo za pretvorbo v bitni zapis, ki je popolnoma identičen dvojiškemu komplementu. Na prejemnikovi strani vrednost, ki je večja od 127, predstavlja negativno število, zato prejeti bajt pretvorimo v desetiški zapis in od njega odštejemo 256. Tako zopet dobimo negativno število.

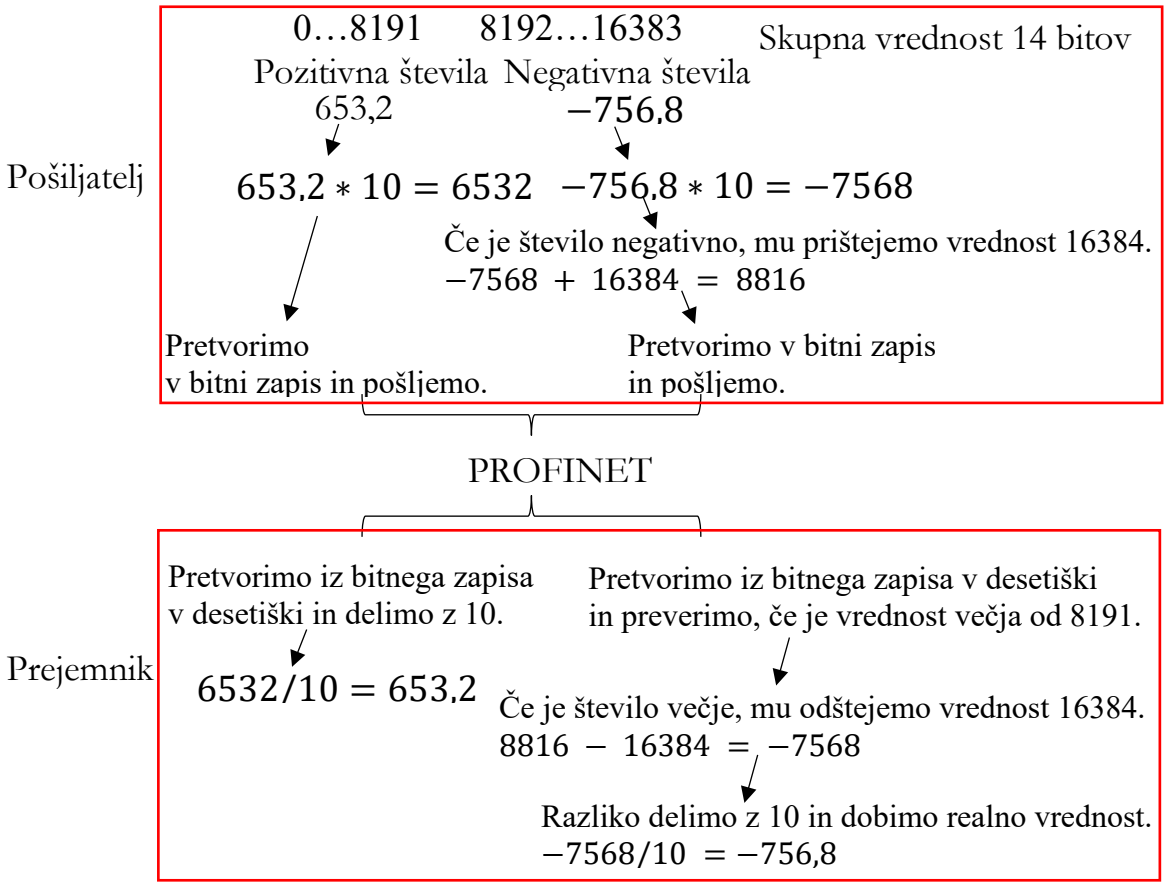
Slika 10.3 podaja primer pošiljanja negativnega števila. Pošiljati želimo števila do +/–127. Za to izberemo en bajt oz. 8 bitov. Skupna vrednost vseh 8 bitov je 256 oz., če štejemo poleg še stanje 0, je skupna vrednost 255. To pomeni, da lahko od tega porabimo 127 števil za pozitivna števila in 128 za negativna števila. Če imamo pozitivno število manjše od 127, ga normalno pretvorimo v bitni zapis, pošljemo in na prejemnikovi strani pretvorimo nazaj v celo število. Če pa imamo negativno število, npr. –10, potem moramo na pošiljateljevi strani temu številu prišteti maksimalno vrednost izbranega bajta, to je 256, in dobimo 246. To število potem pretvorimo v bitni zapis in pošljemo. Na prejemnikovi strani to število najprej pretvorimo v celi zapis in potem preverimo, če je večje od 127. Če je večje, temu številu odštejemo prej prišteto vrednost, torej 256, in dobimo zopet vrednost –10.



Slika 10.3: Primer pošiljanja negativnega števila preko PROFINET

10.3 Uporaba realnih števil s pomočjo bitne logike

Če želimo preko PROFINET pošiljati realna števila, pa potrebujemo več bitov, kot jih je v enem bajtu, še posebej, če so realna števila tudi negativna. Najprej moramo vedeti, kolikšna je maksimalna vrednost podatka ter število decimalnih mest. Več kot bo decimalnih mest, več bitov bo potrebnih. Če bomo potrebovali realna števila v vrednosti do +/– 800.0 z enim decimalnim mestom, potem bomo potrebovali število vrednosti največ 2*800*10 = 16000. To vrednost bomo dosegli s 14 biti. Največja vrednost, ki jo lahko dosežemo s 14 biti, je 16384 oz., če odštejemo stanje 0, je to 16383. Od tega bomo vrednosti do vključno 8191 namenili pozitivnim številom, vrednostim od 8192 do 16383 pa negativnim številom. Če želimo poslati število –756,8, je to realno število potrebno najprej pretvoriti v celo število. To pretvorbo naredimo s pomočjo množenja tega števila z 10 in dobimo –7568. Ker gre za negativno število, moramo temu številu prišteti maksimalno vrednost 14-ih bitov, to je 16384, in dobimo 8816. Na prejemnikovi strani pa dobljen bitni zapis pretvorimo v desetiški in število primerjamo, če je večje od 8191. Če je večje, potem od števila odštejemo 16384 in dobljeno število delimo z 10, tako ponovno dobimo vrednost –756,8. Slika 10.4 prikazuje opisani primer.

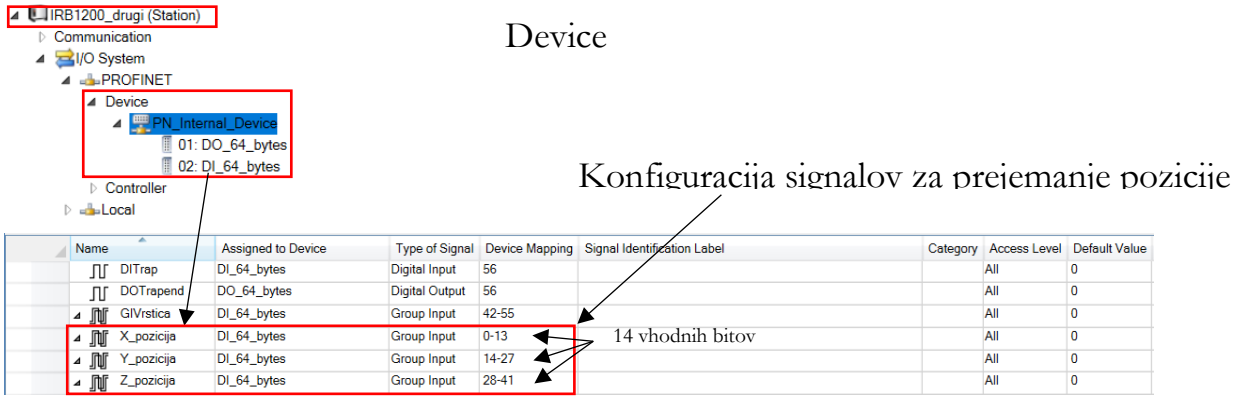


Slika 10.4: Primer pretvorbe realnega števila z enim decimalnim mestom v bitni zapis

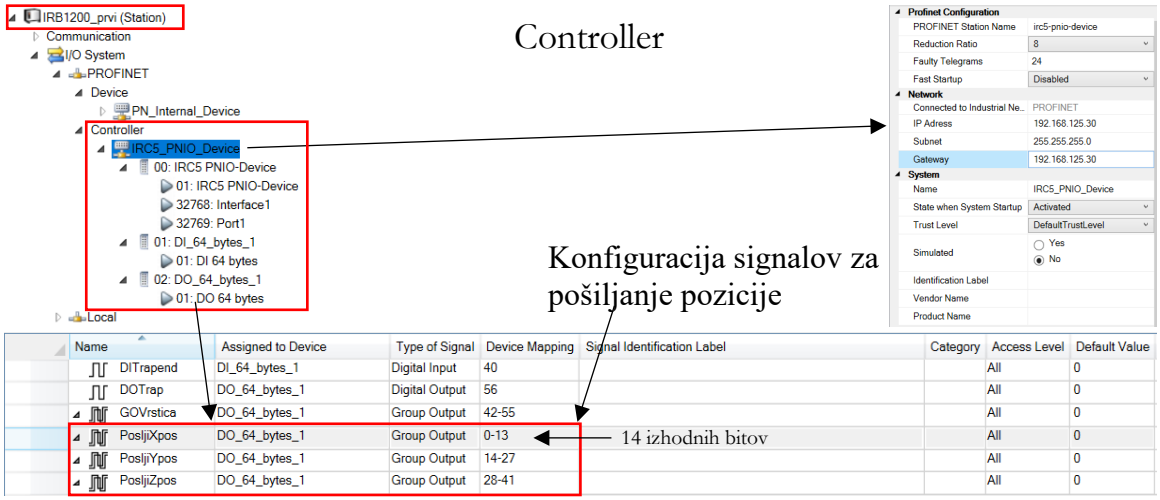
10.4 Pošiljanje pozicije robota preko PROFINET protokola

V nadaljevanju bo prikazan primer pošiljanja pozicije industrijskega robota IRB 1200 preko PROFINET komunikacije drugemu robotu v programu RobotStudio. Prvi robot bo tako veljal za »Controller«, drugi pa za »Device«. Pošiljali bomo pozicijo na eno decimalno mesto, ki pa ne bo večje od +/–800,0 mm. Pri tem bomo pošiljali pozicijo za smeri X, Y in Z tako, da bomo potrebovali tri 14-bitne zloge. V RobotStudio je potrebno konfigurirati dva IRB 1200 robota, ju postaviti nasproti drug drugemu na razdalji X = 1500 mm, Y = 0 mm in Z = 0 mm, pri čemer je drugi robot rotiran okoli Rz = 180 °. Oba robota konfiguriramo posamezno s pomočjo ukaza »Virtual Controller« ter izberemo ukaz »From Layout«. Pri tem pri obeh izberemo dodatno opcijo »PROFINET Controller\Device« za komunikacijo preko PROFINET protokola. Drugega robota je potrebno konfigurirati v načinu »Device«, prvega pa v načinu »Controller«. Pri drugem izberemo isti princip, kot je opisan v poglavju 8.2.1. Pod »I/O Configurator« dodamo na »Device« opcijo s 64 bajtov vhodno/izhodnih signalov. Slika 10.5 prikazuje konfiguracijo drugega robota. Pri prvem je potrebno pod »Controller« dodati »Device« napravo, ki je v tem konkretnem primeru krmilnik drugega robota, tako tukaj dodamo ABB Robotics

IRC5 PNIO-Device. Poleg glavne kartice moramo definirati še velikost vhodno/izhodnih bajtov. Pri tem izberemo možnost DI 64 bytes in DO 64 bytes. Tako imamo na voljo $64 * 8 = 512$ vhodno/izhodnih signalov. Slika 10.6 prikazuje konfiguracijo prvega robota.



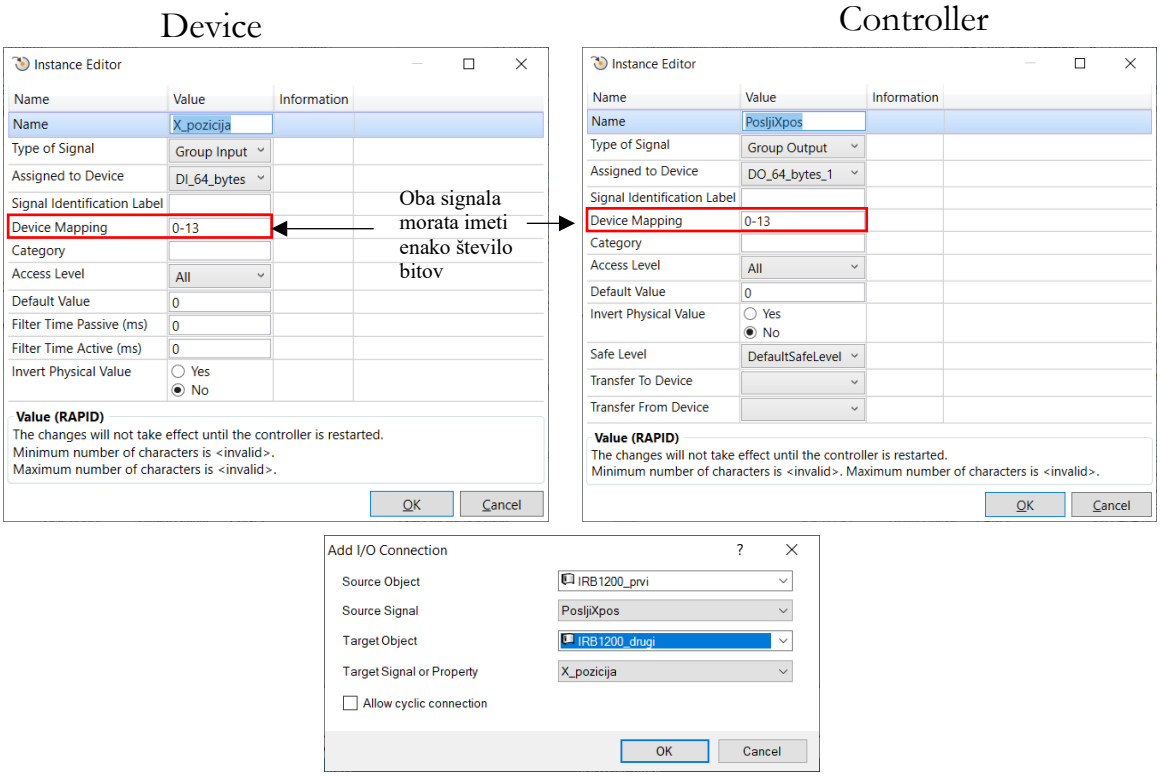
Slika 10.5: Konfiguracija drugega robota v »Slave« načinu



Slika 10.6: Konfiguracija prvega robota v »Controller« načinu

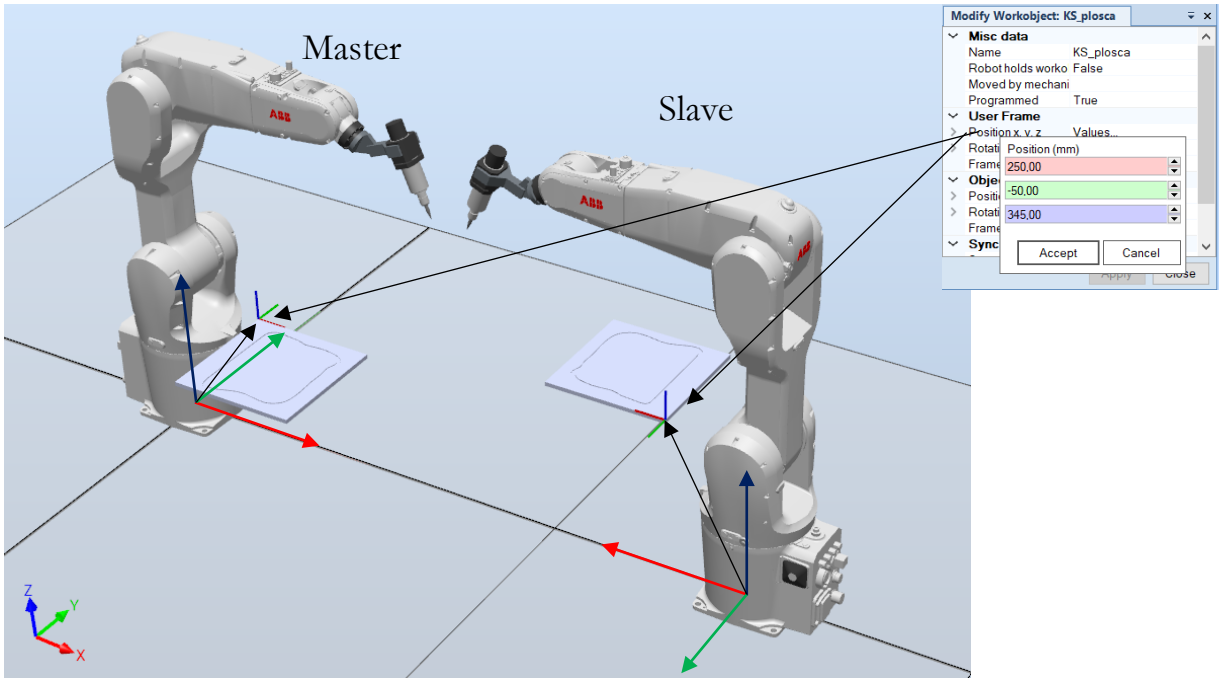
Za pošiljanje pozicije preko PROFINET protokola moramo kreirati skupino signalov in jih združiti pod enim imenom. Kreiranje vhodno/izhodnih signalov poteka preko ukaza »Configuration« -> »I/O System« in z desnim klikom na »Signal« dodamo nov signal. Pri tem izberemo opcijo »GroupOutput« ter to opcijo povežemo z »DO_64_bytes«. S tem zagotovimo povezljivost teh signalov s PROFINET kartico. Slika 10.7 prikazuje kreiranje skupine signalov na obeh straneh ter povezavo teh signalov. Ker komunikacija poteka v virtualnem okolju, je potrebno definirati, kateri signali so medsebojno odvisni, saj v virtualnem okolju ne moremo priklopiti omrežnega kabla. Za povezavo signalov moramo na zavihku »Simulation« izbrati ukaz »Station logic« in nadalje pod zavihkom »Signals and Connections« dodati povezavo, kot je prikazana na sliki 10.7. **Pri tem moramo paziti, da**

je velikost skupine signala tako na pošiljateljevi strani kot na strani prejemnika enaka. V nasprotnem primeru prenos ne bo pravilno deloval.



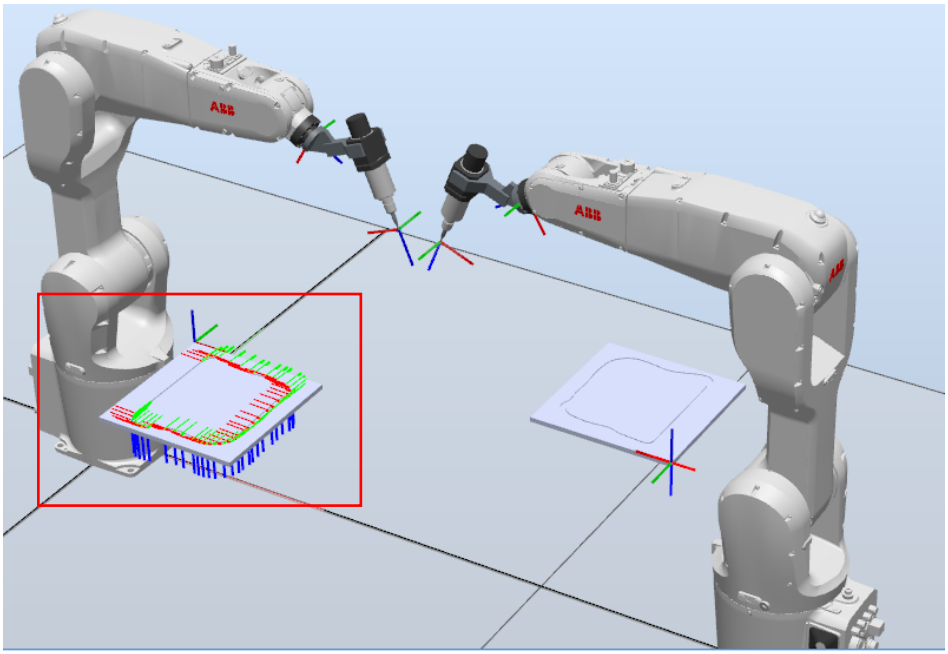
Slika 10.7: Kreiranje skupine signalov na strani »Device« in »Controller« ter povezava teh signalov med krmilnikoma

Obema robotoma dodamo orodje »MyTool«, ki se nahaja v »Import Library« pod »Equipment«. Orodje pritrdimo na prirobnico robota. Poleg robota vnesemo ploščo z izrezom, okoli katerega se morata robota gibati. Za lažje programiranje bomo na ploščo z izrezom pritrdili koordinatni sistem tako, da bomo definirali »User Frame«. Plošči z izrezom postavimo zrcalno-diagonalno tako, da je razmerje med koordinatnim sistemom robota in ploščo enako pri obeh robotih. Slika 10.8 prikazuje pravilno postavitev plošč z izrezom.



Slika 10.8: Pozicija postavitev plošč z izrezom mora biti enaka pri obeh robotih.

Ko imamo postavljen koordinatni sistem na obeh ploščah, je potrebno kreirati referenčne točke, ki jih bo prvi robot pošiljal k drugemu robotu. Slika 10.9 prikazuje kreirane tečke, ki jih ustvarimo s pomočjo ukaza »AutoPath«.



Slika 10.9: Kreiranje referenčnih točk s pomočjo ukaza »AutoPath«

Po kreiranju točk je potrebno modificirati program, da bo ta po vsakem premiku v kreirano točko poslal trenutno pozicijo preko PROFINET komunikacije do drugega robota. Ta pa bo po prejemu izvedel gib v prejete koordinate.

Najprej bomo pripravili program na prvem robotu. Dodati je potrebno nekaj spremenljivk vrste VAR tipa »num« in pa »pos«. Nato je potrebno pripraviti program, ki zapiše trenutno pozicijo TCP glede na izbrani koordinatni sistem in jo pripravi za pošiljanje preko PROFINET komunikacije. Krmilnik robota pozicijo prebere na 3 decimalna mesta natančno. Ker pa nas zanima natančnost samo na eno decimalko, bomo to vrednost pomnožili z 10 in nato odrezali decimalni del tako, da nam ostane celoštevilčni zapis. Pri tem bomo uporabili ukaz »Trunc«. Slika 10.10 prikazuje program pošiljanja pozicije na prvem robotu.

```
VAR pos Pozicija_robota;
VAR num Poz_x:=0;
VAR num Poz_y:=0;
VAR num Poz_Z:=0;
```

Deklariranje dodatnih spremenljivk

```
PROC Path_10()
        MoveJ Home,v1000,z100,MyTool\WObj:=wobj0;
        MoveJ Target_10,v1000,fine,MyTool\WObj:=KS_plosca;
        SetDO Start,1;
        Poslji_pozcijio;
        MoveL Target_20,v100,fine,MyTool\WObj:=KS_plosca;
        .
        .
        .
        Poslji_pozcijio;
        MoveJ Home,v1000,z100,MyTool\WObj:=wobj0;
        WaitRob \InPos;
        SetGO PosljiXpos, 0;
        SetGO PosljiYpos, 0;
        SetGO PosljiZpos, 0;
        SetGO GOVrstica, 0;
        SetDO Start,0;
ENDPROC
```

Glavni program

Start drugega robota
Klic funkcije za pošiljanje pozicije

Čakanje robota, da je v pozicij

Resetiranje vseh vrednosti

Rutina pošiljanja pozicije

```
PROC Poslji_pozcijio()
        vrstica:=vrstica+1;                          ! Po vsakem posiljanju povecamo vrednost vrstice
    Pozicija_robota:=CPos(\Tool:=MyTool \WObj:=KS_plosca);! Preberemo trenutno pozicijo z izbranim orodjem in KS
    Poz_x:=Trunc(Pozicija_robota.x*10);              ! Prebrano pozicijo X pomnozimo z 10 in odrezemo decimalni del
    IF Poz_x<0 THEN                                  ! Ce je pozicija negativna, potem pristejemo 16384
       Poz_x:=Poz_x+16384;
    ENDIF
    Poz_y:=Trunc(Pozicija_robota.y*10);              ! Ponovimo zgoraj opisani postopek za Y
    IF Poz_y<0 THEN
       Poz_y:=Poz_y+16384;
    ENDIF
    Poz_z:=Trunc(Pozicija_robota.z*10);              ! Ponovimo zgoraj opisani postopek za Y
    IF Poz_z<0 THEN
       Poz_z:=Poz_z+16384;
    ENDIF
    SetGO PosljiXpos, Poz_x;                         ! Izracunano vrednost posjlemo preko grupiranega signala na PROFINET
    SetGO PosljiYpos, Poz_y;                         ! Izracunano vrednost posjlemo preko grupiranega signala na PROFINET
    SetGO PosljiZpos, Poz_z;                         ! Izracunano vrednost posjlemo preko grupiranega signala na PROFINET
    SetGO GOVrstica, vrstica;                        ! Posjlemo st. vrstice preko grupiranega signala na PROFINET
    WHILE (DITrapend=0) DO                           ! Dokler ne dobimo signala, da je drugi robot prebral pozicijo, robot ne
!nadaljuje
       PulseDO \high \Plength:=0.2,DOTrap;
    ENDWHILE
ENDPROC
```

Slika 10.10: Program pošiljanja pozicije na prvem robotu

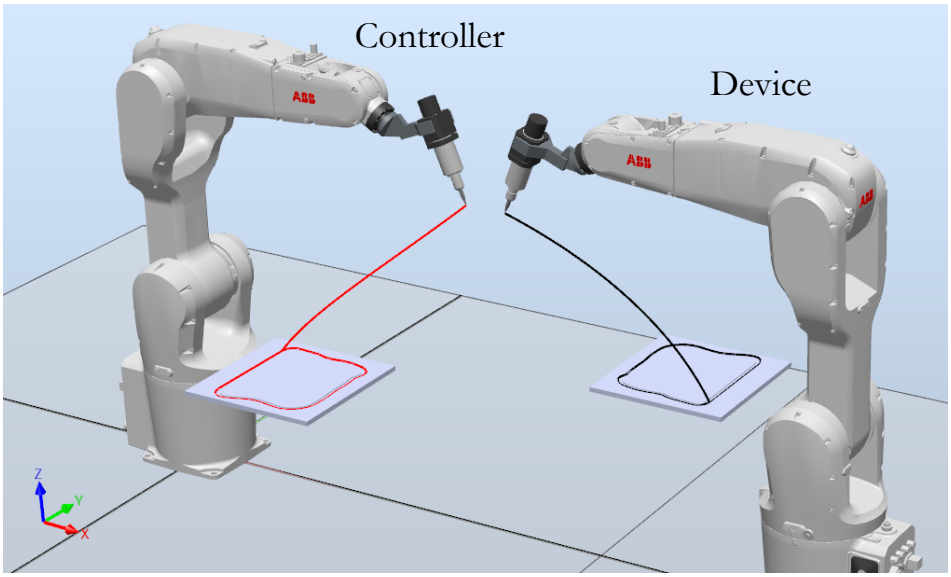
Celotni program za prvega robota lahko najdete v prilogi 17.1.

Pri pripravi programa na **drugem robotu** potrebujemo referenčno točko, kateri bomo potem prištevali oz. odštevali prejeto pozicijo. Referenčno točko potrebujemo z namenom, da robotu eksplicitno povemo, kako naj pride v prejeto točko. Če robot nima nobene referenčne točke, ne ve, s kakšno konfiguracijo se bo v to točko zapeljal. Več o konfiguraciji robota lahko preberete v poglavju 9.2. Robot bo program začel izvajati v varni točki, pri hkratnem zagonu počakal 1 sekundo, da prvi robot izvede prvo pošiljanje, potem pa bo drugi robot začel z branjem pozicije in izvajanjem gibov.

```
VAR robtarget Target_robota;
VAR num Poz_x:=0;
VAR num Poz_y:=0;                                                    Kreiranje dodatnih spremenljivk
VAR num Poz_Z:=0;
VAR num vrstica:=0;
CONST robtarget Target_10:=[[0,0,0],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];   Kreiranje referenčne točke

PROC Path_10()                                                       Glavni program
   MoveJ Home,v1000,z100,MyTool\WObj:=wobj0;      ! Postavitev robota v varno tocko
   WaitDI Start,1;                                ! Cakanje prvega ukaza za branje
   WaitTime 0.5;                                  ! Zakasnitev branja
   Preberi_pozicijo;                              ! Klic funkcije za branje prejete pozicije
   ConfL\Off;                                     ! Ne preverja konfiguracije linearnih gibov
   MoveJ Target_robota,V1000,z0,MyTool\WObj:=KS_plosca;! Izvedba prvega giba na podlagi prejetih podatkov
   WHILE vrstica<43 DO                            ! Ponavljanje branja in izvajanja giba robota, dokler prvi robot
   Preberi_pozicijo;                              ! ne poslje vseh podatkov
   ConfL\Off;
   MoveL Target_robota,V100,z0,MyTool\WObj:=KS_plosca;
   WaitRob \InPos;                                ! Robot mora doseci pozicijo, prej se program ne sme izvajati dalje
   ENDWHILE
   MoveJ Home,v1000,z100,MyTool\WObj:=wobj0;      ! Vrnitev robota nazaj domov
   vrstica:=0;
ENDPROC

PROC Preberi_pozicijo()                                              Branje pozicije
   Poz_x:=GInput(X_pozicija);                     ! Branje pozicije po X smeri
   IF Poz_x>8191 THEN                             ! Ce je prejeta pozicija vecja od 8191, potem od tega odstejemo 16384
      Poz_x:=Poz_x-16384;
   ENDIF
   Poz_x:=Poz_x/10;                               ! Da dobimo decimalko, prejeto vrednost delimo z 10
   Poz_y:=GInput(Y_pozicija);                     ! Ponovimo postopek za Y os
   IF Poz_y>8191 THEN
      Poz_y:=Poz_y-16384;
   ENDIF
   Poz_y:=Poz_y/10;
   Poz_z:=GInput(Z_pozicija);                     ! Ponovimo postopek za Z os
   IF Poz_z>8191 THEN
      Poz_z:=Poz_z-16384;
   ENDIF
   Poz_z:=Poz_z/10;
   Target_robota:=Offs(Target_10,Poz_x,Poz_y,Poz_Z);! V referencno tocko Target_10 zapisemo zamike po posameznih oseh
   vrstica:=GInput(GIVrstica);                    ! Preberemo stevilko vrstice
   PulseDO \High \PLength:=0.1, DOTrapend;        ! Posljemo kratek signal prvemu robotu, da smo prebrali pozicijo
   RETURN;
ENDPROC
```

Slika 10.11: Program prejemanja pozicije na drugem robotu

Po kreiranju obeh programov in zagonu simulacije lahko vključimo še funkcijo sledenja TCP-točki. Pri tem lahko za posameznega robota nastavimo različno barvo. Slika 10.12 prikazuje zagon programa in sledenje TCP-točki pri obeh robotih. Opazimo lahko, da so na strani drugega robota majhna odstopanja, saj so tudi točke določene zgolj na eno decimalno vrednost, vendar je to dovolj dober primer za dokazovanje principa pošiljanja realnih podatkov preko PROFINET protokola.



Slika 10.12: Zagon programa in sledenje TCP-točki.

10.5 Vprašanja za utrjevanje snovi

1. Naslednja desetiška števila pretvorite v dvojiški zapis ter dvojiška števila pretvorite v desetiški zapis.

11 _____

123 _____

5113 _____

0010 0001 _____

1000 1111 _____

0001 0111 1111 0100 _____

Ugotovitve:

_____

_____

2. Zapišite logiko programa za pošiljanje realnih števil velikosti do ± 200.0 preko PROFINET protokola. Pri tem izberite ustrezno velikost bitov in jih združite v ustrezno skupino signalov. Ne pozabite, da lahko pošiljamo tako pozitivna kot negativna števila.

Ugotovitve:

3. Na spletni strani eStudij, kjer se nahajajo predmeti Industrijska robotika, Robotizacija ali Roboti in robotizacija, iz datoteke RobotStudio – vaje prenesite »PROFINET_dva_robota_vaja«. Z dvojnim klikom se odpre postopek »Unpack&Work«, kjer določite, kam boste shranili datoteko. Pri tem potrebujete »robotware« verzijo 6.10.10, v nasprotnem primeru bo program javljal napako. Sistem vsebuje dva robota in dve plošči z izrezom. Posameznemu robotu je potrebno določiti koordinatni sistem plošče z izrezom. Pri tem si pomagajte z opisom v tem poglavju.

Ugotovitve:

4. Po kreiranju koordinatnih sistemov na obeh robotih je potrebno preveriti, ali so signali, ki se pošiljajo iz enega robota k drugemu, pravilno definirani ali jih je za potrebe pošiljanja željene vrednosti potrebno spremeniti. Prav tako poglejte, ali je prvi robot pravilno konfiguriran kot »Controller« ter drugi robot kot »Device«. Poglejte in zapišite tudi, kolikšen je IP-naslov prvega in kolikšen je IP-naslov drugega. Pri tem si pomagajte z ukazom »I/O Engineering Tool«.

IP-naslov prvega robota	IP-naslov drugega robota

Ugotovitve:

5. Na prvem robotu ustvarite program preprostega sledenja majhnemu delu utora na plošči. Ta program naložite v virtualni krmilnik in posodobite program tako, da bo po vsakem zaključenem gibu poslal svojo pozicijo na drugega robota. Pri tem pozicijo robota zaokrožite na eno decimalno mesto. Za start prenosa in branje pozicije na drugem robotu uporabite signal START, ki ga proži prvi robot.

Ugotovitve:

ZAPISKI

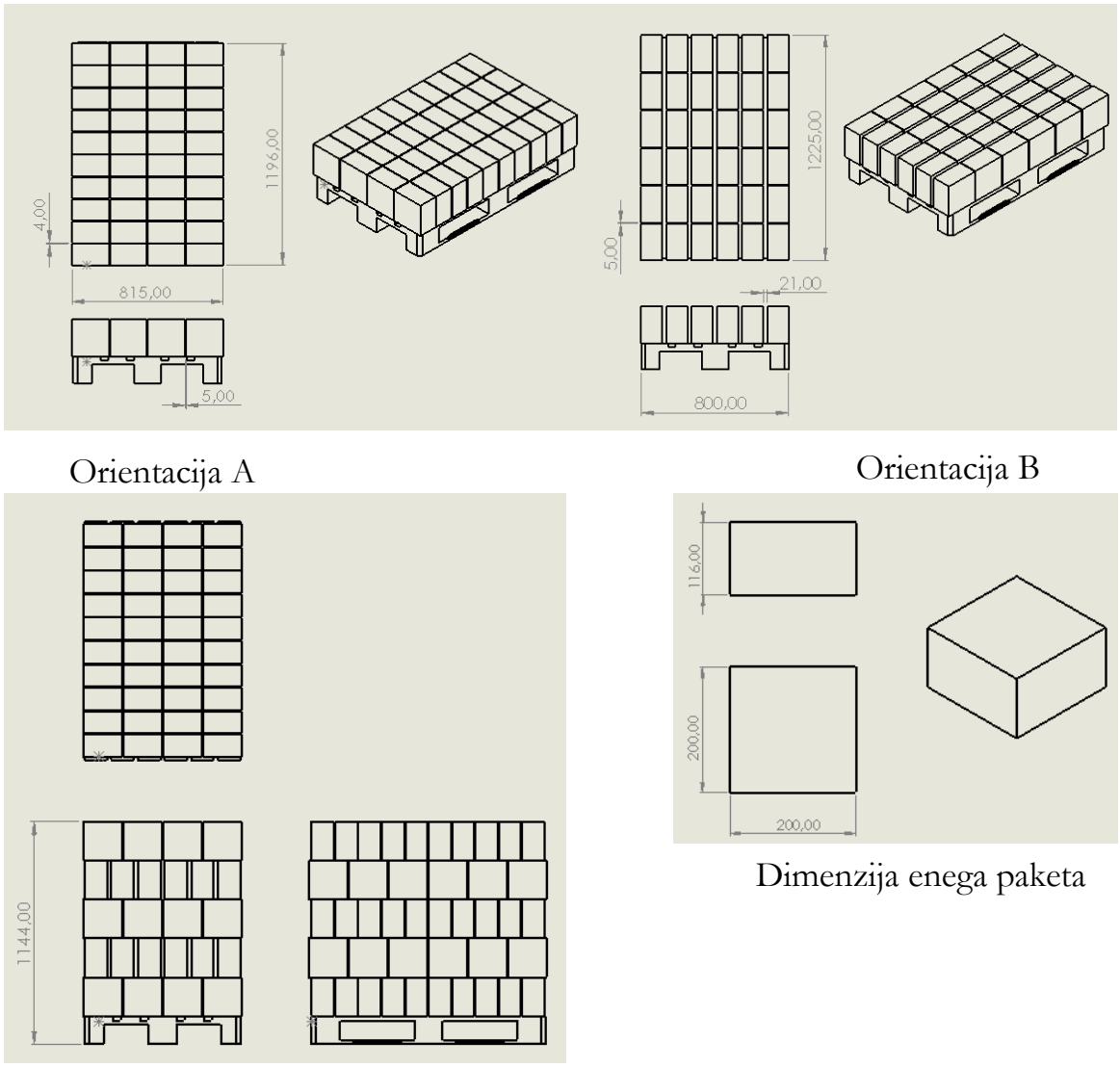
Univerzitetna založba
Univerze v Mariboru

11 Program paletizacije

V pripravi programa za paletizacijo izdelkov s pomočjo industrijskega robota bodo zajeti podatkovni tipi, ki so bili opisani v prejšnjih poglavjih. Prav tako bo v programu prikazana uporaba prekinitev, prekinitvenih rutin, način vračanja v začetno varno pozicijo, preverjanja prijemala v slučaju izgube izdelkov kot tudi prekinitev izvajanja programa in vrnitev v varno pozicijo.

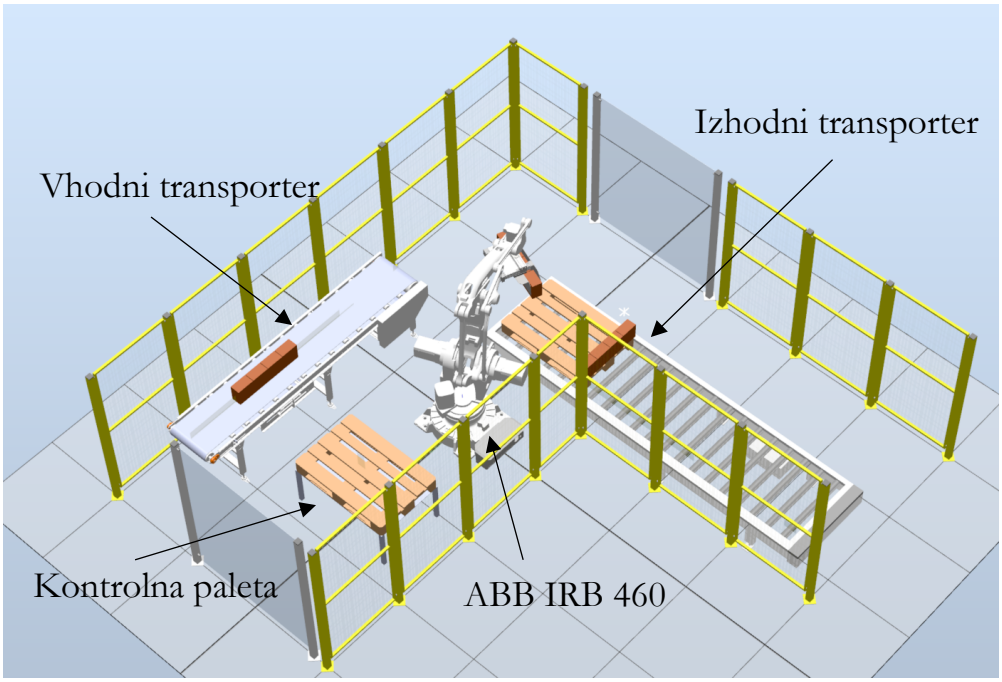
11.1 Zahteve za zlaganje izdelkov

Program paletizacije je sestavljen iz vhodnega transporterja, iz katerega prihajajo izdelki. Ti bodo prihajali po štirje oz. po trije skupaj vzdolžno, odvisno katera orientacija zlaganja bo trenutno aktualna. Njihove dimenzije so (200 x 116 x 200) mm^3 in se zlagajo na euro paleto dimenzije (1200 x 800) mm^2. Pri tem se zlagata dva različno orientirana sloja, da se izdelki bolje prepletajo in je paleta stabilnejša. Skupno je na paleti zloženih pet slojev. Slika 11.1 prikazuje orientacijo in dolžino slojev.



Slika 11.1: Zahteva zlaganja izdelkov na euro paleto

Robotska celica vsebuje tudi izhodni transporter, na katerem se nahaja euro paleta. Levo od robota se nahaja miza z dodatno euro paleto, na kateri se odlagajo izdelki za preverjanje. Ta program se izvede pod pogojem, da je operater podal zahtevo za kontrolo. Nalogo paletizacije opravlja 4-osni robot podjetja ABB z oznako IRB 460, nosilnostjo 110 kg in dosegom 2.4 m. Slika 11.2 prikazuje postavitev posameznih komponent proizvodne celice.



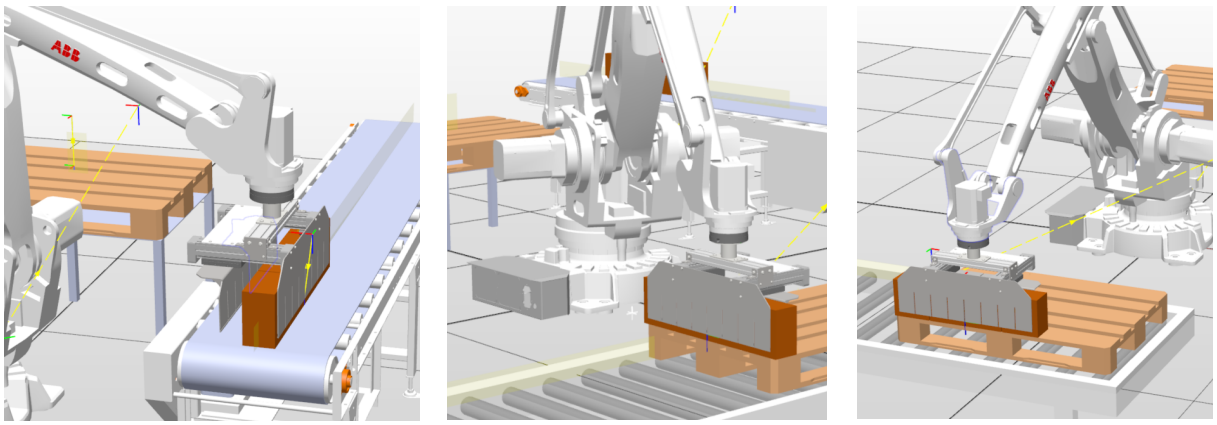
Slika 11.2: Postavitev komponent proizvodne celice

Proces je zastavljen tako, da najprej po vhodnem transporterju prihajajo štirje izdelki hkrati. Ti se potem zložijo prečno na paleto. Ta način zlaganja predstavlja orientacijo A. V tem načinu imamo po štiri pakete v vrsto in deset vrstic. Na sloj tako naložimo 40 izdelkov. Po tem se sloj zamenja in po vhodnem transporterju pričnejo prihajati izdelki po trije skupaj. Ti se potem zlagajo vzdolžno na paleto, kjer se v eni vrstici zloži šest izdelki. V celotnem sloju imamo tako šest vrstic. Ta način zlaganja je označen kot orientacija B in na en sloj zložimo 36 izdelkov. Orientaciji se tako menjujeta, na paleto pa zložimo skupno pet slojev. Orodje, ki bo skrbelo za premikanje izdelkov, ima približno 40 kg ter dva pnevmatska cilindra. Vsak cilinder zapira svojo polovico. Ko imamo popolnoma definirano in pritrjeno prijemalo na prirobnici robota, lahko začnemo z določevanjem koordinatnih sistemov in referenčnih točk.

11.2 Kreiranje koordinatnih sistemov in referenčnih točk

Prvi koordinatni sistem, ki ga moramo določiti, je koordinatni sistem na vhodnem transporterju. Ta določa referenčno točko prijemanja izdelkov. Pri tem koordinatnem sistemu je dovolj, da definiramo samo »User Frame« vrednosti. Pri tem ima točka prijemanja koordinate 0,0,0 glede na novo kreirani koordinatni sistem. Od te točke definiramo točko pred prijemanjem in točko za prijemanjem. Tako za prijemanje izdelkov potrebujemo tri točke. Te točke so vezane na prej kreiran koordinatni sistem ter na aktivno prijemalo. Slika 11.3 a) prikazuje referenčno točko za prijemanje izdelkov.

Nato se lotimo kreiranja koordinatnega sistema za odlaganje izdelkov, najprej za orientacijo A. Točka odlaganja za orientacijo A na paleto tako predstavlja koordinate za koordinatni sistem odlaganja. Slika 11.3 b) prikazuje referenčno točko za odlaganje izdelkov glede na orientacijo A. Za odlaganje izdelkov v orientaciji B pa uporabimo isti koordinatni sistem, kot je za orientacijo A, s to razliko, da kreiramo referenčno točko za odlaganje. Slika 11.3 c) prikazuje kreiranje referenčne točke za odlaganje izdelkov v B orientaciji sloja. V obeh primerih potrebujemo še dve točki, ki sta dvignjeni za 220 mm od referenčnega koordinatnega sistema. Slika 11.4 prikazuje kreiranje referenčnih točk odlaganja in točk pred odlaganjem. S tem imamo kreirane vse potrebne referenčne točke, da lahko začnemo s pisanjem glavnega programa.

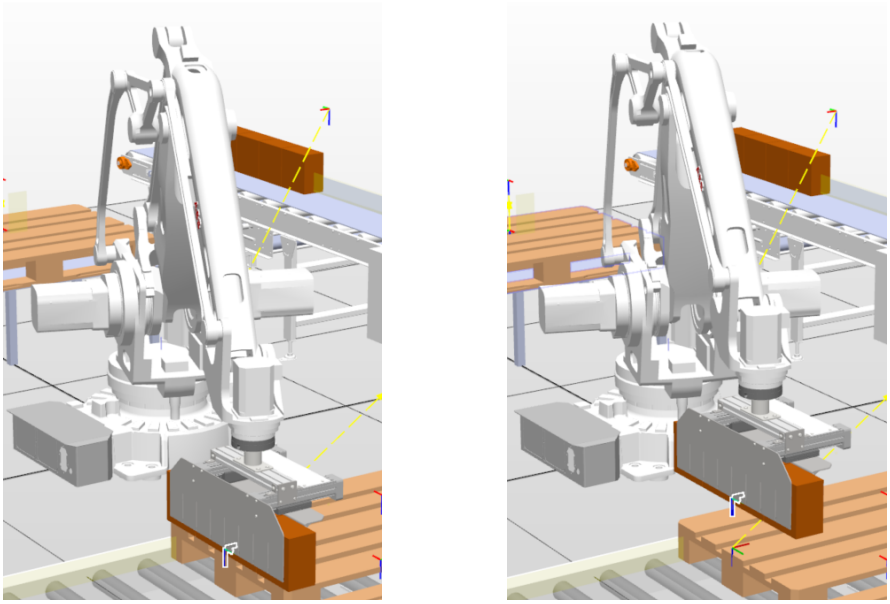


a) Referenca vhodni transporter

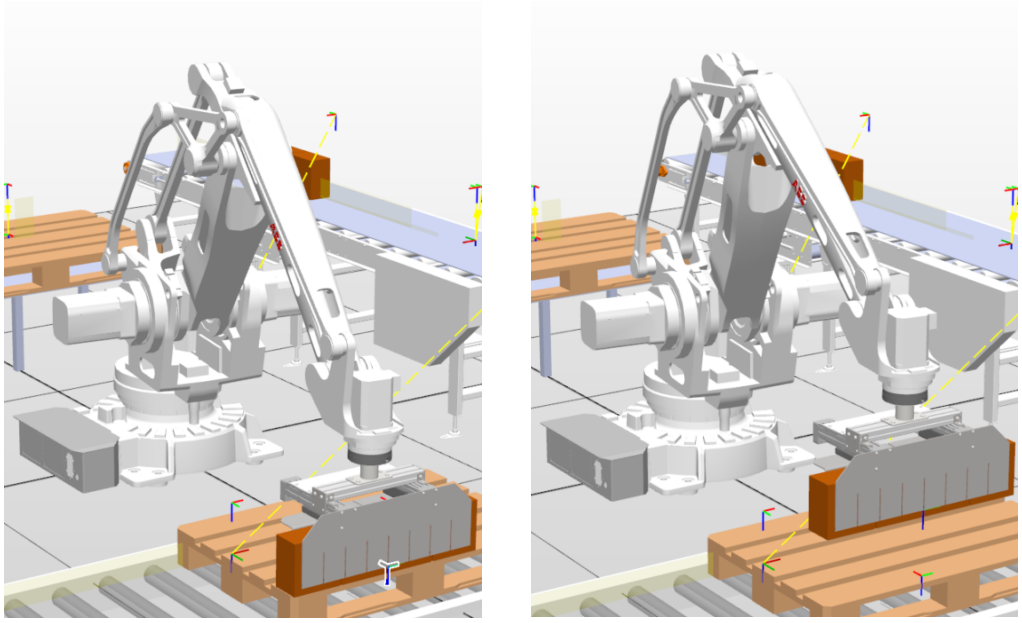
b) Referenca odlaganje orientacija A

c) Referenca odlaganje orientacija B

Slika 11.3: Določevanje referenčnih koordinatnih sistemov in točk za prijemanje in odlaganje izdelkov



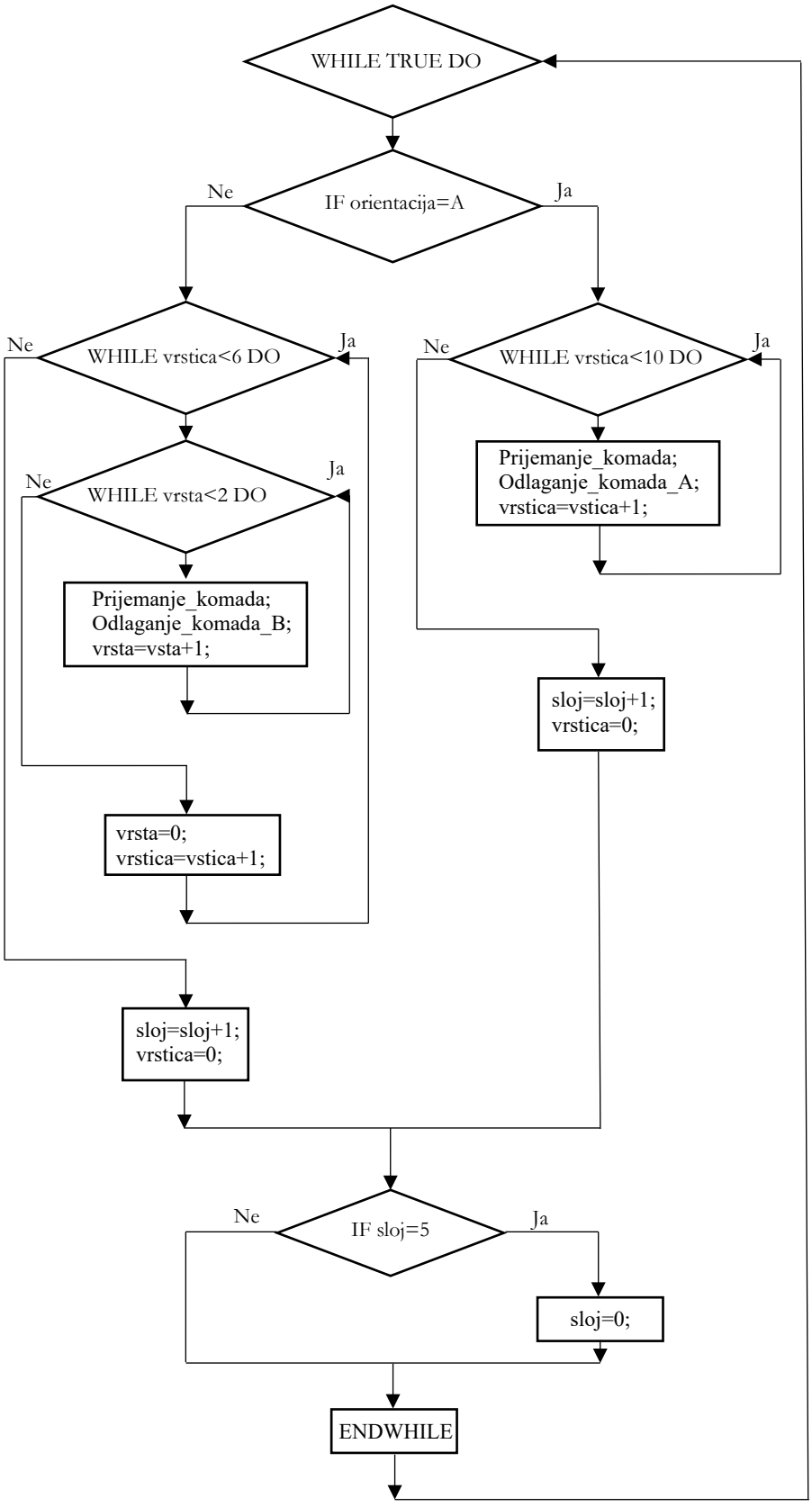
Slika 11.4: Kreiranje referenčne točke odlaganja in točke pred odlaganjem pri orientaciji A



Slika 11.5: Kreiranje referenčne točke odlaganja in točke pred odlaganjem pri orientaciji B

11.3 Kreiranje glavnega programa paletizacije

Za namene paletizacije bomo uporabili samo en TASK znotraj krmilnika ter samo en modul, ki pa bo imel več rutin, med katerimi bodo tudi prekinitvene in funkcijske rutine. Prva stvar, ki se je moramo zavedati pri pisanju programa, je preverjanje stanja robota, ali se robot nahaja v varni poziciji, iz katere lahko varno štarta program in začne z izvajanjem operacije. Ta rutina bo razložena v nadaljevanju v podpoglavju 11.4. Najprej bomo kreirali program zlaganja izdelkov v željene orientacije in na željeno število slojev. Pri tem bomo uporabili odločitveni zanki IF in WHILE DO. Ker bo robot opravljal ponavljajoče se opravilo, bo odločitvena zanka WHILE TRUE DO poskrbela za neskončno ponavljanje programa. Odločitvena zanka IF bo določevala orientacijo sloja, WHILE DO pa bo skrbela za štetje vrstic v določenem sloju. Glavni program je tako sestavljen iz preverjanja orientacije sloja, potem pa se deli na zlaganje A in B sloja. Pri zlaganju A orientacije pobiramo in odlagamo 4 izdelke hkrati v eno vrstico. Ta postopek je potrebno ponoviti 10-krat. S tem zaključimo sloj. Pri odlaganju B orientacije pa pobiramo in odlagamo po 3 izdelke hkrati. Da pokrijemo celotno dolžino palete, je potrebno v eno vrstico odložiti 6 izdelkov. Tako moramo v eno vrstico dvakrat odložiti pakete. Pri odlaganju izdelkov v B orientacijo sta tako potrebni dve odločitveni zanki WHILE DO, kjer prva šteje število vrstic, druga pa število vrst znotraj ene vrstice. Slika 11.6 prikazuje celoten miselni postopek v obliki diagrama poteka.



Slika 11.6: Diagram poteka glavnega programa paletizacije

Znotraj glavnega programa se nahajajo podprogrami prijemanja in odlaganja izdelkov. Tako se ne glede na orientacijo najprej izvede podprogram Prijemanje_komadov, nato pa podprogram odlaganja ali orientacije A ali B. Slika 11.7 prikazuje RAPID kodo glavnega programa. Slika 11.8 pa prikazuje podprograme prijemanja in odlaganja. Celoten program je v obliki kode zapisan v prilogi 17.3.

```
PROC main()
    WHILE TRUE DO
      IF Orientacija_sloja=1 THEN
        WHILE vrstica<10 DO
          Prijemanje_komada;
          Odlaganje_komada_A;
          vrstica:=vrstica+1;
        ENDWHILE
        vrstica:=0;
        sloj:=sloj+1;
      ENDIF
      IF Orientacija_sloja=0 THEN
        WHILE vrstica<6 DO
          WHILE vrsta<2 DO
            Prijemanje_komada;
            Odlaganje_komada_b;
            vrsta:=vrsta+1;
          ENDWHILE
          vrsta:=0;
          vrstica:=vrstica+1;
        ENDWHILE
        vrstica:=0;
        sloj:=sloj+1;
      ENDIF
      IF sloj=5 THEN !
        sloj:=0;
      ENDIF
    ENDWHILE
ENDPROC
```

Slika 11.7: RAPID koda glavnega programa

```
PROC Prijemanje_komada()
  MoveJ Pred_prijemanje,v2000,z1,Prijemalo_paletizacija_1\WObj:=KS_tranposrter;
  Preverjanje_prijemala;
  WaitDI Komadi_pripravljeni,1; ! V kolikor komadi niso pripravljeni, robot ne sme pobirati!
  MoveL Prijemanje,v500,fine,Prijemalo_paletizacija_1\WObj:=KS_tranposrter;
  ZapriPrijemalo;
  MoveL Prijemanje_gor,v500,z1,Prijemalo_paletizacija_1\WObj:=KS_tranposrter;
ENDPROC
```

```
PROC Odlaganje_komada_A()
  PredOdlaganje_paleta:=Pred_odlaganje;
  Odlaganje_paleta:=Odlaganje;
  visina_off:=sloj*visina_paketa;
  sirina_off:=vrstica*sirina_paketa;
  PredOdlaganje_paleta:=Offs(PredOdlaganje_paleta,sirina_off,0,-visina_off);
  Odlaganje_paleta:=Offs(Odlaganje_paleta,sirina_off,0,-visina_off);
  MoveJ PredOdlaganje_paleta,V2000,z10,Prijemalo_paletizacija_1\WObj:=KS_Paleta;
  WaitDI Paleta_v_poziciji,1;          ! V kolikor paleta ni pripravljena, robot ne sme odlagati komadov!
  MoveL Odlaganje_paleta,V500,fine,Prijemalo_paletizacija_1\WObj:=KS_Paleta;
  OdpriPrijemalo;
  MoveL PredOdlaganje_paleta,V500,z10,Prijemalo_paletizacija_1\WObj:=KS_Paleta;
  MoveJ Vmesna,v2000,z100,Prijemalo_paletizacija_1\WObj:=wobj0;   ! Dodatna točka za izogibanje napravam!
ENDPROC

  VAR robtarget PredOdlaganje_paleta;
  VAR robtarget Odlaganje_paleta;
  VAR num visina_off;
  VAR num sirina_off;
```

```
PROC Odlaganje_komada_B()
  PredOdlaganje_paleta:=Pred_odlaganje_b;
  Odlaganje_paleta:=Odlaganje_b;
  visina_off:=sloj*visina_paketa;
  sirina_off:=vrstica*sirina_paketa_b;
  dolzina_off:=vrsta*dolzina_paketa;
  PredOdlaganje_paleta:=Offs(PredOdlaganje_paleta,dolzina_off,-sirina_off,-visina_off);
  Odlaganje_paleta:=Offs(Odlaganje_paleta,dolzina_off,-sirina_off,-visina_off);
  MoveJ PredOdlaganje_paleta,V2000,z10,Prijemalo_paletizacija_1\WObj:=KS_Paleta;
  WaitDI Paleta_v_poziciji,1; ! V kolikor paleta ni pripravljena, robot ne sme odlagati komadov!
  MoveL Odlaganje_paleta,V500,fine,Prijemalo_paletizacija_1\WObj:=KS_Paleta;
  OdpriPrijemalo;
  MoveL PredOdlaganje_paleta,V500,z10,Prijemalo_paletizacija_1\WObj:=KS_Paleta;
  MoveJ Vmesna,v2000,z10,Prijemalo_paletizacija_1\WObj:=wobj0; ! Dodatna točka za izogibanje napravam!
ENDPROC
```

Slika 11.8: Podprogrami prijemanja in odlaganja

V podprogramu prijemanja imamo najprej ukaz za izvedbo gibanja v točko pred prijemanjem, nakar se izvrši podprogram preverjanja odprtja prijemala. Ta podprogram preveri, ali sta končni stikali aktivni ali ne. Če končni stikali za odprti prijemali nista aktivni, je potrebno prijemalo odpreti. Ta ukaz se izvaja tako dolgo, dokler stikali nista aktivni. Naslednje kar moramo preveriti, je prisotnost izdelkov. Če izdelki niso prisotni, se robot ne sme premakniti v pozicijo prijemanja. Če so vsi pogoji aktivni, robot prime izdelke in se dvigne nad vhodni transporter. Po tem se program premakne v podprogram odlaganja. Pri orientaciji A imamo eno odlaganje na vrstico in 10 vrstic, pri orientaciji B pa imamo dve odlaganji na vrstico in 6 vrstic. Podprogram odlaganja pri orientaciji A je sestavljen tako, da se referenčni točki Pred_odlaganje in Odlaganje prepišeta v spremenljivki, izračuna se potreben premik referenčnih točk tako po višini kot po dolžini glede na zaporedno številko vrstice in sloja. Ta premik se izvede v obliki »Offs« na prej določeni spremenljivki referenčnih pozicij. Nato se izvede gib pred odlaganjem, zatem robot preveri, ali je paleta v poziciji. Če palete ni na poziciji, potem robot ne sme odlagati. Ko so vsi pogoji izpolnjeni, robot izvede odlaganje na mesto, ki se izračuna iz trenutne vrstice in sloja. Podobno logiko uporabimo pri odlaganju izdelkov pri orientaciji B s to razliko, da moramo pri tem v eni vrstici odložiti dvakrat po tri izdelke. Tako moramo pri tem upoštevati poleg širine in višine izdelkov še dolžino. Končni mesti odlaganja in predodlaganja se tako izračunajo iz upoštevanja trenutne vrstice, vrste in sloja.

Podprogrami Preverjanje_prijemala, OdpriPrijemalo in ZapriPrijemalo pa so funkcijski podprogrami, saj ne vsebujejo nobene inštrukcije gibanja, temveč imajo samo funkcijo preverjanja odprtosti prijemala, odpiranja in zapiranja le-tega.

```
PROC Preverjanje_prijemala()
  WHILE (Prijemalo_D_odprto=0) OR (Prijemalo_L_odprto=0) DO
    SetDO Zapri_prijemalo,0;
    SetDO Odpri_prijemalo,1;
  ENDWHILE
  SetDO Odpri_prijemalo,0;
ENDPROC

PROC OdpriPrijemalo()
  SetDO Zapri_prijemalo,0;
  SetDO Odpri_prijemalo,1;
  WaitTime 0.6;
ENDPROC

PROC ZapriPrijemalo()
  SetDO Odpri_prijemalo,0;
  SetDO Zapri_prijemalo,1;
  WaitTime 0.6;
ENDPROC
```

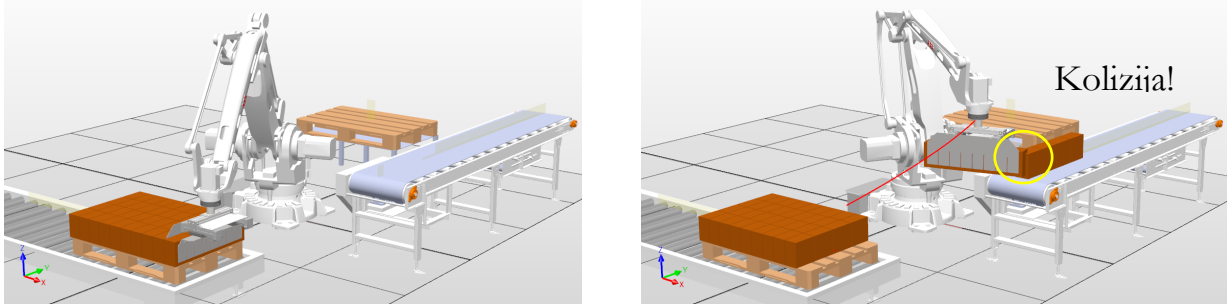
Slika 11.9: Podprogram preverjanja, odpiranja in zapiranja prijemala

11.4 Podprogram varnega vračanja v začetno točko po resetiranju programskega kazalca

Opisani program predstavlja osnovo programa za paletizacijo. S tem programom bi robot opravljal svojo funkcijo, a bi bil za neveščega operaterja zelo nevaren. Predstavljajmo si, da podjetje ugotovi, da so izdelki, ki jih je robot že zložil na paleto, poškodovani in ne morejo iti v prodajo. Tako se celoten proces paletizacije ustavi, pri čemer se robot ustavi nekje nad paleto odlaganja. Na pol naložena paleta se odstrani, na njeno mesto se postavi nova paleta, v robotskem prijemalu pa ostanejo izdelki še od prejšnje serije. Po ponovnem zagonu se robotski program vrne na začetek (PP to Main), a ker robot ni začel iz varne točke, je možnost kolizije robota z ostalimi komponentami precejšnja. Tako je vedno potrebno poskrbeti, da se **po resetiranju programskega kazalca** na začetek programa izvede podprogram **varnega vračanja robota** v varno pozicijo, če robot ni v varni poziciji. Prav tako po resetiranju programskega kazalca uporabljene spremenljivke obdržijo zadnje vrednosti. Zato je nujno potrebno, da te spremenljivke postavimo na začetne vrednosti. Ta program mora biti izveden še pred začetkom izvajanja glavnega programa, tako je diagram poteka na sliki 11.5 potrebno dopolniti.

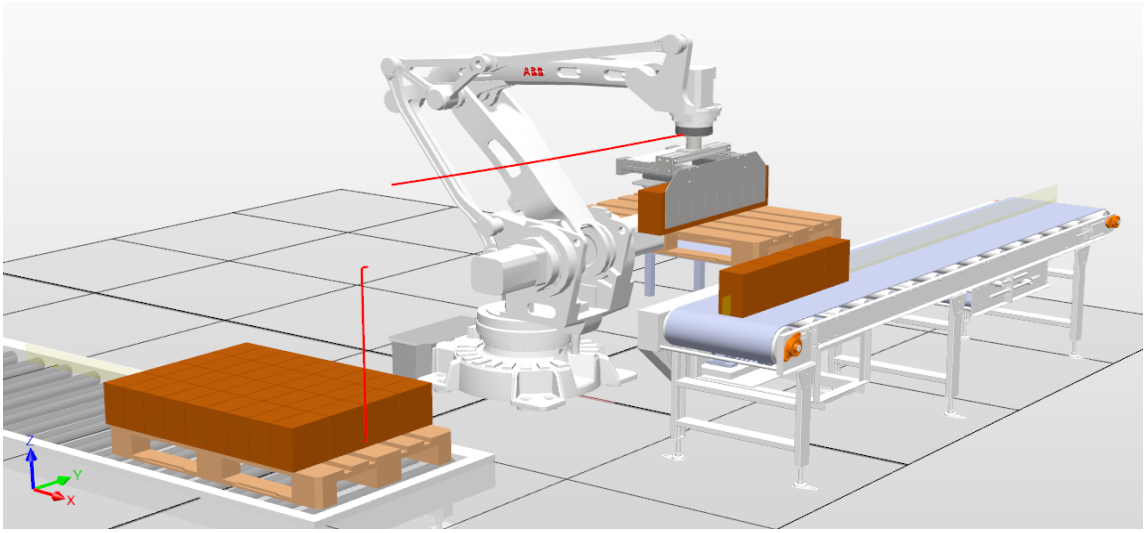
Najprej v glavnem programu izvedemo inicializacijo spremenljivk, v naslednjem koraku pa preverimo, ali se robot nahaja v varni poziciji. Za izvedbo inicializacije lahko ustvarimo podprogram, znotraj katerega resetiramo spremenljivke na začetno vrednost.

V naslednjem koraku je potrebno preveriti, ali se robot nahaja v varni poziciji. Če ni tako, se mora do varne pozicije vrniti z zmanjšano hitrostjo in brez kolizij. V tem konkretnem primeru imamo samo nekaj primerov možnih situacij, kjer lahko pride do kolizije robota s transporterjem. To je primer, kadar je robot v fazi odlaganja izdelkov na najnižji sloj in na začetku palete. Če ga v tej poziciji pošljemo domov, se bo na trajektoriji zaletel v vhodni transporter in izdelke, ki čakajo na njem. Slika 11.10 prikazuje najslabši možni scenarij za vračanje robota v varno pozicijo.



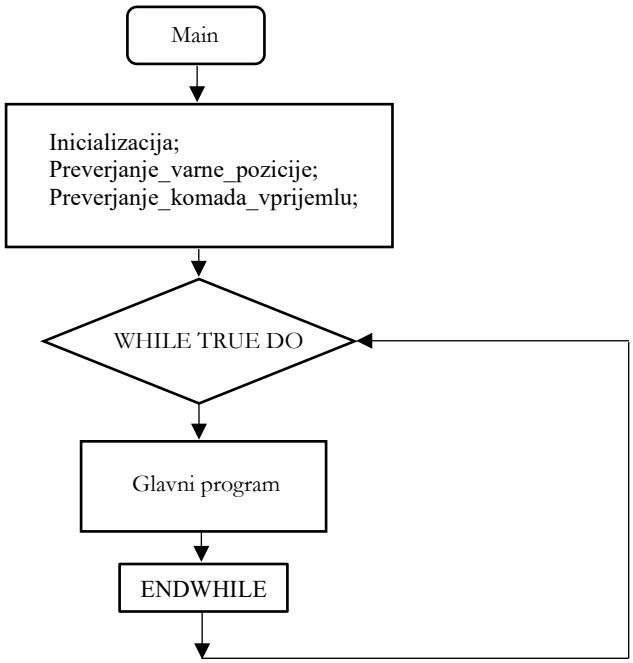
Slika 11.10: Najslabši možni scenarij za vračanje robota v varno pozicijo, pri čemer bo zagotovo prišlo do kolizije.

Ta scenarij je potrebno preprečiti, kar bomo storili s pomočjo preverjanja položaja prijemala. Če bo prijemalo nižje od 350 mm eksperimentalno določene višine od K.S. robota, potem bomo robota dvignili za 500 mm in ga nato pomaknili v varno pozicijo. Če bo prijemalo med 350 in 900 mm, potem bomo robota dvignili za 300 mm in ga nato pomaknili v varno pozicijo. Če pa bo višje od 900 mm, pa naj gre direktno v varno pozicijo, saj v tem primeru ni nevarnosti kolizije. Slika 11.11 prikazuje princip vračanja robota brez kolizije v varno pozicijo iz najslabšega možnega scenarija.



Slika 11.11: Preprečevanje kolizije pri izvajanju vračanja robota v varno pozicijo

Kot lahko opazimo v sliki 11.11, se pri vračanju v varno pozicijo lahko pojavi nov problem. In sicer imamo v prijemalu še vedno izdelke, ki pa jih moramo odložiti, če želimo nadaljevati s programom paletizacije. Splošen postopek je tak, da je za to predvideno posebno mesto, kamor se odlagajo takšni izdelki. Tako potrebujemo še program za preverjanje izdelkov v prijemalu in odlaganje le-teh za na to predvideno mesto. Ta program se mora prav tako izvršiti še pred glavnim programom. Slika 11.12 prikazuje posodobitev diagrama poteka. Slika 11.13 prikazuje kodo programa varne vrnitve, inicializacijo spremenljivk ter program preverjanja prisotnosti izdelka v prijemalu še pred izvedbo glavnega programa.



Slika 11.12: Posodobitev diagrama poteka

```
PROC Preverjanje_varne_pozicije()
    Trenutna_poz:=CRobT(\Tool:=tool0\WObj:=wobj0);       ! V spremenljivko shrani trenutno spremenljivko glede na tool0 in k.s. wobj0
    IF Trenutna_poz<>Varna_pozicija THEN                 ! Ce je trenutna pozicija razlicna od varne, potem se vrni v varno pozicijo
        Trenutna_poz:=CRobT(\Tool:=Prijemalo_paletizacija_1\WObj:=wobj0);   ! V spremenljivko shrani trenutno spremenljivko glede na aktivno orodje in k.s. wobj0
        IF Trenutna_poz.trans.z<350 THEN                 ! Ce je robot nizje od 350 mm po Z osi, potem se najprej pomakni za 500 mm gor
            Trenutna_poz:=Offs(Trenutna_poz,0,0,500);
            MoveL Trenutna_poz,v200,z10,Prijemalo_paletizacija_1\WObj:=wobj0;
        ENDIF
        IF Trenutna_poz.trans.z>350 AND Trenutna_poz.trans.z<900 THEN ! Ce je robot visje od 350 mm in nizje od 900 po Z osi, potem se najprej pomakni za 300 mm gor
            Trenutna_poz:=Offs(Trenutna_poz,0,0,300);
            MoveL Trenutna_poz,v200,z10,Prijemalo_paletizacija_1\WObj:=wobj0;
        ENDIF
        MoveJ Varna_pozicija,v200,z10,tool0\WObj:=wobj0;   ! Vrni se v varno pozicijo
    ELSE
    ! Ne naredi nicesar
    ENDIF
ENDPROC
```

```
PROC Inicializacija()
    vrstica:=0;
    sloj:=0;
    stetje_komadov:=0;
ENDPROC
```

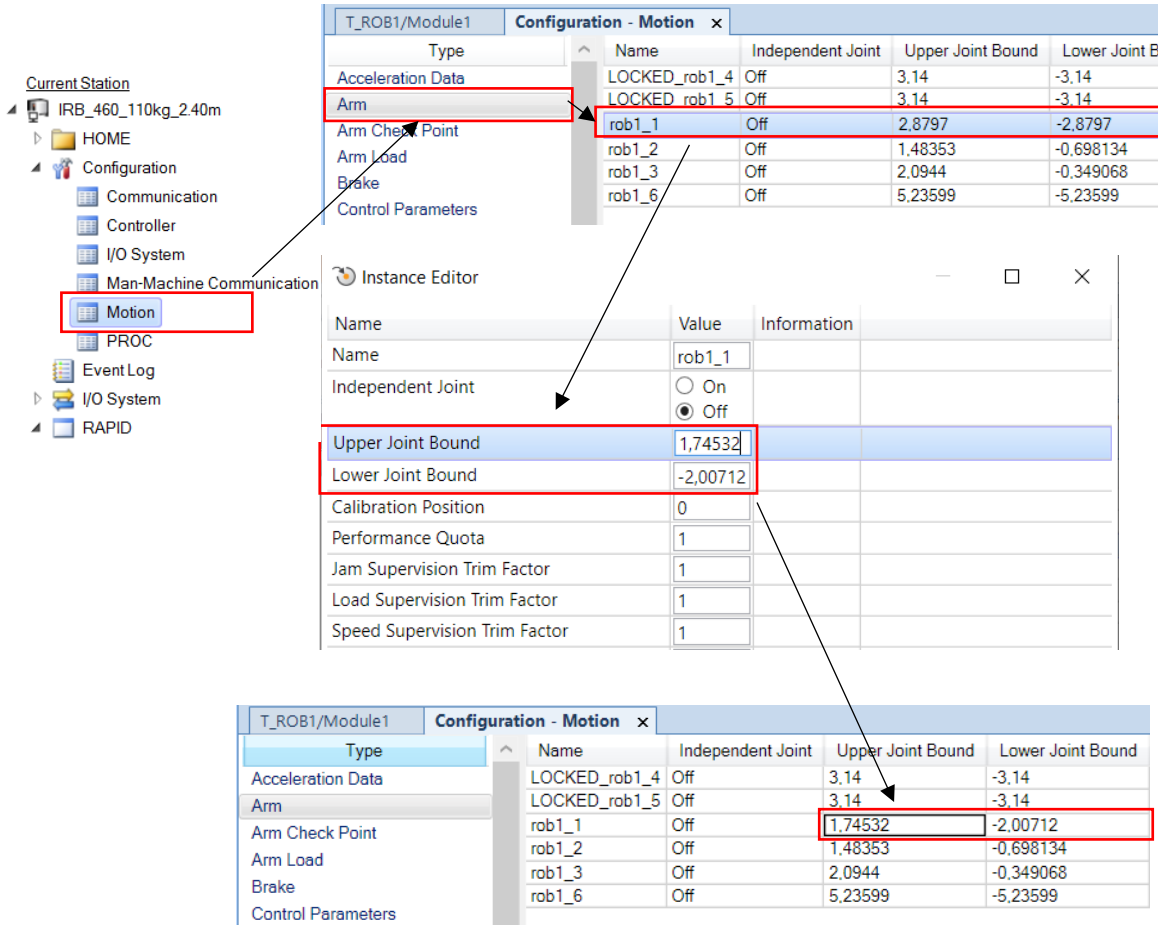
```
PROC Preverjanje_komada_vprijemalu()
    IF Prijemalo_D_odprto=0 OR Prijemalo_L_odprto=0 THEN    ! Če je vsaj eno prijemalo zaprto, pojdi in odloži komade na za to določeno mesto
        Preverjanje_odlaganje_komada;
    ENDIF
ENDPROC
```

```
PROC Preverjanje_odlaganje_komada()
    MoveJ PredOdlaganje_kom_v_prij,v1000,z1,Prijemalo_paletizacija_1\WObj:=wobj0;
    MoveL Odlaganje_kom_v_prij,v500,fine,Prijemalo_paletizacija_1\WObj:=wobj0;
    OdpriPrijemalo;
    MoveL PredOdlaganje_kom_v_prij,v500,z1,Prijemalo_paletizacija_1\WObj:=wobj0;
    MoveJ Varna_pozicija,v1000,z10,tool0\WObj:=wobj0;
ENDPROC
```

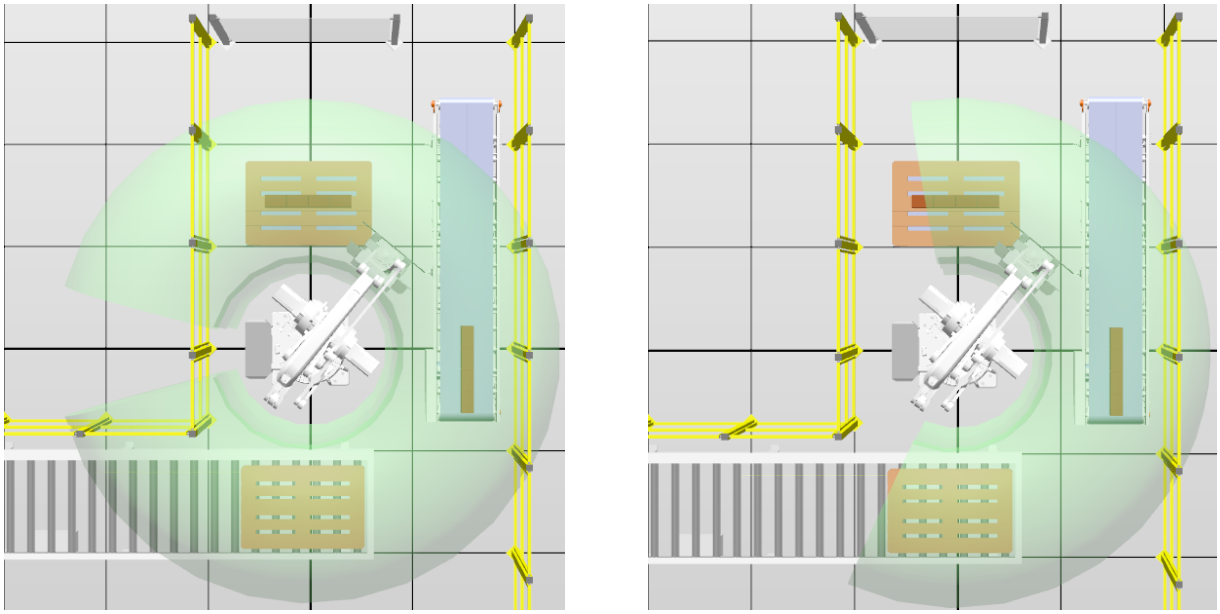
Slika 11.13: Program preverjanja varne pozicije robota, inicializacije in preverjanja izdelka v prijemalu

11.5 Omejevanje zasuka posameznih osi industrijskega robota

Industrijski robot redko dela v okolju, kjer ima dovolj prostora, da se lahko neomejeno giblje glede na podane specifikacije. Pogosto se dogaja, da je robot integriran v okolje, kjer je zelo omejen s svojimi gibi. V ta namen je potrebno omejiti njegovo območje gibanja, še posebej za namene preprečevanja kolizije, saj v večini primerov operater ni posebej izšolan za upravljanje z robotom. Pri tem lahko omejimo posamezno os ali pa več osi. Vse to je odvisno od posameznega okolja, v katerega je robot integriran. V konkretnem primeru paletizacije je robot omejen zgolj na gibanje po prvi osi. Neomejen robot se lahko po prvi osi giblje $\pm$ 165 °. Če tega ne omejimo, lahko pride do kolizije z varnostno ograjo pri nepravilni manipulaciji z robotom. Slika 11.14 prikazuje postopek omejevanja dosega posamezne osi na industrijskem robotu IRB 460. Slika 11.15 pa prikazuje dosegljivost neomejenega in omejenega robota.



Slika 11.14: Omejevanje dosega prve osi na ABB IRB 460



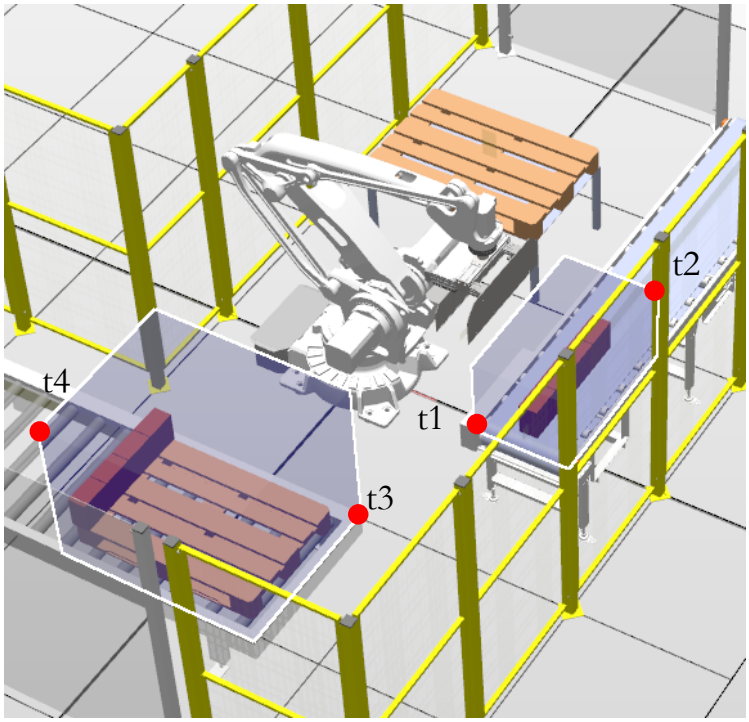
Slika 11.15: Doseg neomejenega robota – levo, doseg omejenega robota – desno

11.6 Definiranje delovnih območij robota-WorldZones

Proizvajalci industrijskih robotov ponujajo tudi kreiranje delovnih območij okoli robota. Delovno območje je definirano kot območje okoli nekega stroja, lahko je tudi območje okoli robota. Pri tem imamo na voljo več različnih oblik delovnih območij, kot je kvader, sfera ali valj. Zakaj sploh potrebujemo delovna območja v delovnem prostoru robota? Predstavljajmo si, da si dva robota delita skupno delovno območje. Oba pobirata izdelke z istega zalogovnika. Pri tem obstaja velika verjetnost kolizije med njima. V tem primeru definiramo delovno območje, ki je oblike kvadra. Kadar se trenutno aktivni TCP-robot 1 nahaja v tem območju, robot 2 nima dovoljenja za vstop v to delovno območje in obratno. Delovno območje lahko definiramo tudi tako, da preprečimo robotu, da vstopi v to delovno območje. Želimo npr. zavarovati en del delovnega stroja. Na tem delu kreiramo delovno območje z možnostjo preprečitve gibanja robota. Kadar trenutno aktivni TCP-robot preide to območje, se robot avtomatsko ustavi in javi napako. Prav tako se pogosto dogaja, da robot javlja nadzornemu PLK-ju območje, kje se nahaja. To je zelo enostavna rešitev. Ko se trenutno aktiven TCP-robot nahaja znotraj delovnega področja, potem je digitalni izhod aktiven. Delovna območja so aktivna tako v avtomatskem kot tudi v ročnem načinu delovanja. To je še posebej dobrodošlo, kadar imamo na voljo zelo malo prostora in moramo zaščititi delovne stroje in robota. S tem preprečimo kolizijo, če z robotom upravlja manj vešč operater. Kot opozorilo pa naj velja, da **ta funkcija ni namenjena varovanju človeka pred robotom**! Prav tako je potrebno pri ponudniku industrijskega robota preveriti, ali je ta funkcija že vgrajena ali jo je potrebno dodatno

naročiti. Glede na izkušnje je bolje vse funkcije, ki jih potrebujemo, naročiti že v fazi naročanja robota kot pa kasneje, saj lahko dobimo nižjo ceno. Ta funkcija pri ponudniku ABB ni avtomatsko vgrajena v industrijske robote, tako jo je potrebno posebej naročiti. Funkcija ima oznako »608-1 World Zones« in če jo želimo uporabljati v programu RobotStudio, jo je potrebno definirati že pri kreiranju robotskega sistema, lahko pa jo vključimo naknadno.

Pri kreiranju delovnih območij v programu RobotStudio si pomagamo s kreiranjem kvadrov. Kvader kreiramo na mestu, kjer želimo imeti delovno območje, potem pa s pomočjo kreiranja točk »CreateTarget« odčitamo poziciji diagonale tega kvadra. Te točke nam predstavljajo delovno območje. Tako smo na konkretnem primeru paletizacije kreirali dve delovni območji. Prvo predstavlja delovno območje vhodnega transporterja, drugo pa predstavlja delovno območje izhodnega transporterja. Slika 11.16 prikazuje kreiranje delovnih območij nad vhodnim in izhodnim transporterjem. Slika 11.17 prikazuje programsko kodo za kreiranje delovnih območij. Pri kreiranju delovnih območij moramo izbrati ali želimo imeti stacionarno ali začasno delovno območje. Razlika med njima je bistvena. Stacionarno delovno območje je namenjeno varovanju objektov, zato ga lahko definiramo samo enkrat, in še to po navadi samo pri zagonu robota. Prav tako stacionarnega delovnega objekta ne moremo deaktivirati znotraj programa RAPID. Začasna delovna območja pa uporabljamo tam, kjer ni namenjeno varovanju, temveč samo sporočanju, v katerem območju se nahaja robot. Najprej moramo kreirati spremenljivki »wztemporary«, ki opisujeta trenutni delovni območji. Ti spremenljivki kreiramo v glavnem programu, da sta dostopni znotraj celotnega programa. V podprogramu »def_zone« pa najprej definiramo spremenljivki »shapedata«, kjer izberemo volumen, ki bo opisoval delovno območje. Nato definiramo prvo točko kvadra in njegovo diagonalo. Isto storimo za kvader nad izhodnim transporterjem. Ko imamo definirane točke, kreiramo kvader s pomočjo ukaza »WZBoxDef«. Pri tem moramo izbrati način »\Inside«, saj s tem omogočimo zaznavanje aktivnega TCP-orodja znotraj definiranega območja. Prav tako izberemo ime območja »volumen1« in točki, ki ga definirata »t1 in t2«. V naslednjem koraku je potrebno definirati, da bo prečenje delovnega območja z aktivnim TCP-jem prožilo digitalni izhod. To storimo s pomočjo ukaza »WZDOSet«, ki ga definiramo kot začasnega »\Temp«, definiramo mu območje »volumen1«, ki se bo aktiviral znotraj tega območja »\Inside«. Prav tako mu moramo definirati, kateri DO je potrebno prožiti »Poz_vh_trn«, ter definirati vrednost na 1.



Slika 11.16: Kreiranje delovnih območij nad vhodnim in izhodnim transporterjem

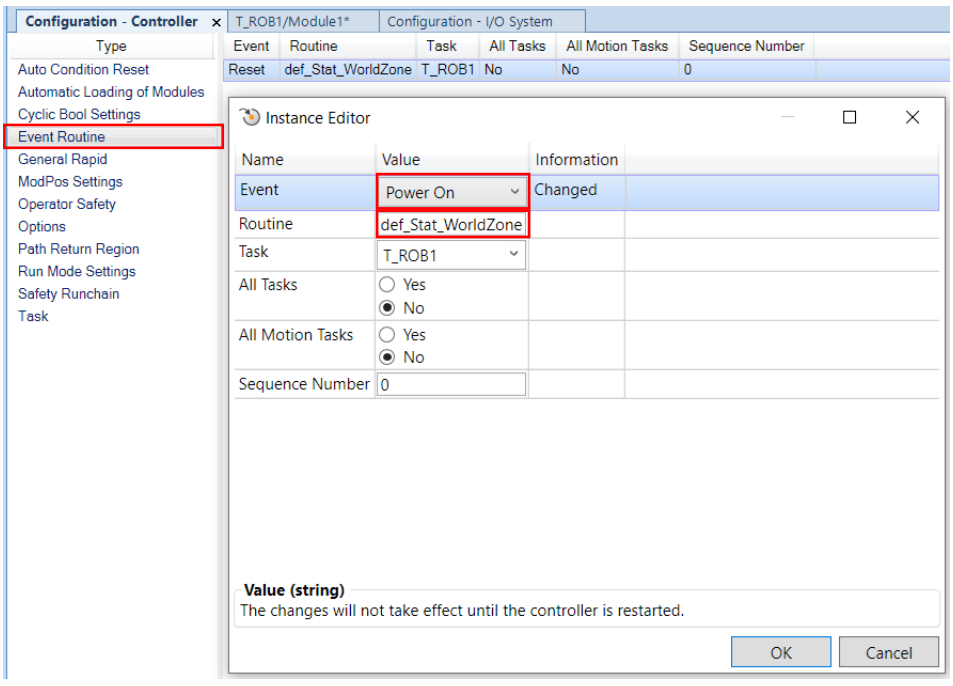
Če bi želeli ustvariti delovno območje, ki je namenjeno varovanju opreme, je potrebno delovno območje definirati kot stacionarno. Postopek je identičen, kot je to opisano v predhodnem primeru. Edina razlika je, da stacionarno delovno območje lahko definiramo samo enkrat. To najlažje storimo tako, da rutino, ki definira stacionarno območje, povežemo z rutino dogodka oz. »Event Routine«. Ko zaženemo krmilnik, ta izvede določene funkcije, med drugimi izvede tudi rutine, ki jih povežemo z zagonom. Slika 11.18 prikazuje primer kreiranja stacionarnih delovnih območij. Da povežemo rutino z rutino dogodka, je potrebno v zavihku »Configuration« izbrati ukaz »Controller« ter »Event Routine«. Z desnim klikom dodamo novo rutino dogodka in jo povežemo s prej kreirano rutino. Slika 11.19 prikazuje kreiranje in povezovanje rutine z rutino dogodka.

```
    VAR wztemporary zona_transporter_vhodni;
    VAR wztemporary zona_transporter_izhodni;

PROC def_zone()
        ! Oblika omejitve bo volumen
        VAR shapedata volumen1;
        VAR shapedata volumen2;
        CONST pos t1 :=[1208.3,-675,737.8]; ! Definiraj zacetno tocko kvadra1
        CONST pos t2 :=[1808.3,325,1237.8]; ! Definiraj diagonalno tocko kvadra1
        CONST pos t3 :=[620,-990,35]; ! Definiraj zacetno tocko kvadra1
        CONST pos t4 :=[-680,-2060,1035]; ! Definiraj diagonalno tocko kvadra1
        ! Definiranje varne pozicije
        WZBoxDef \Inside,volumen1,t1,t2;     ! Definiraj kvader nad vhodnim transp
        WZBoxDef \Inside,volumen2,t3,t4;     ! Definiraj kvader nad izhodnim transp
        ! Ce je TCP znotraj, aktiviraj DO
        WZDOSet \Temp,zona_transporter_vhodni\Inside,volumen1,Poz_vh_trn,1;
        WZDOSet \Temp,zona_transporter_izhodni\Inside,volumen2,Poz_izh_trn,1;
ENDPROC
```

Slika 11.17: Program kreiranja začasnih delovnih območij

```
    PROC def_Stat_WorldZone()
        VAR shapedata volumen1;
        ! Oblika omejitve bo volumen
        CONST pos t1:=[1300,-600,700];
        ! Definiraj zacetno tocko kvadra1
        CONST pos t2:=[1800,300,1000];
        ! Definiraj diagonalno tocko kvadra1
        WZBoxDef\Inside,volumen1,t1,t2;
        ! Definiraj kvader nad vhodnim transporterjem
        WZDOSet\Temp,zona_transporter_vhodni\Inside,volumen1,Robot_prep_obm,1;
        ! Ce je TCP znotraj, aktiviraj DO
    ENDPROC
```

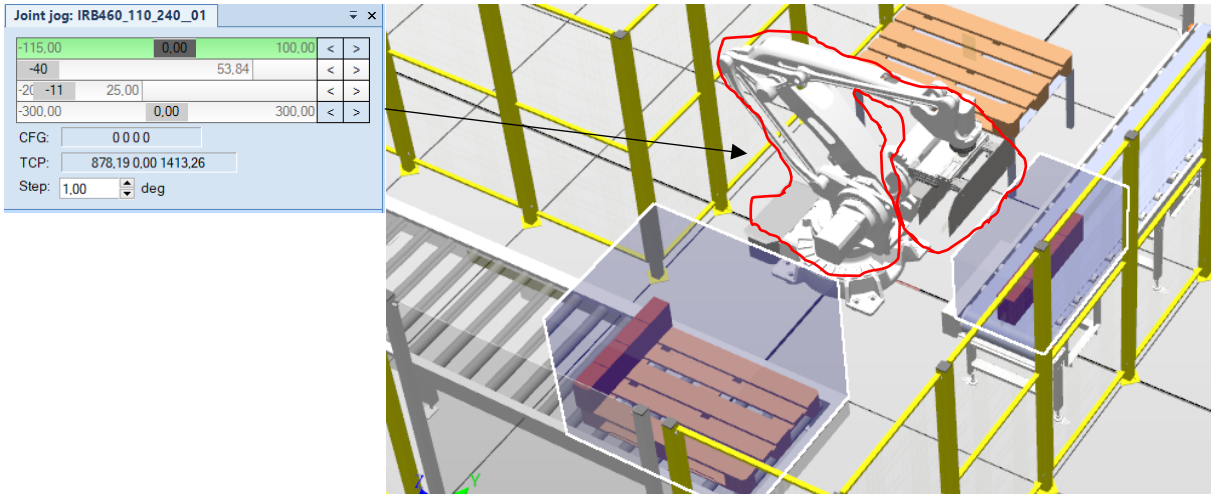
Slika 11.18: Primer programa kreiranja stacionarnih delovnih območij



Slika 11.19: Povezovanje rutine z rutino dogodka

11.7 Kreiranje signala, da se robot nahaja v varni poziciji

Dejali smo že, da je izrednega pomena, kako se robot vrne v varno pozicijo. Pri tem je potrebno paziti na vse možne ovire, ki se lahko nahajajo na tej trajektoriji. Ko se robot nahaja v varni poziciji, je potrebno to sporočiti tudi nadzornemu sistemu oz. PLK-ju. Kako to storimo, je odvisno od proizvajalca industrijskih robotov. Določeni industrijski roboti imajo to funkcijo že vgrajeno, kot je npr. pri KUKA robotih, pri drugih pa moramo to sami sprogramirati. Pri industrijskih robotih ABB je ena izmed možnosti uporaba funkcije WorldZones, kjer definiramo delovno območje okoli zasukov sklepov v varni poziciji. Podobno kot pri definiranju delovnega območja v obliki kvadra bomo tudi tukaj definirali volumen, ki bo v tem primeru definiran kot območje okoli sklepov. Slika 11.20 prikazuje varno pozicijo robota. To pozicijo je potrebno zapisati v obliki »jointtarget«, kar pomeni, da zapišemo trenutno pozicijo zasukov posameznih motorjev. Tako smo na primeru paletizacije določili varno pozicijo, kjer so sklepi robota na (0, –40, –11, 158, **0, 0,** 0)°. Ne smemo pozabiti, da gre tukaj za štiriosnega robota, ki nima sklepov 4 in 5. Tako ta dva sklepa avtomatsko zapišemo z 0. V naslednjem koraku moramo definirati odstopanja po posameznih oseh od varne pozicije. Pri tem so odstopanja zapisana v °, če imamo »jointtarget«, ter v mm, če imamo navaden »robtarget«. V primeru paletizacije določimo, da je odstopanje po posameznih oseh ± 1 °. To odstopanje prav tako zapišemo v obliki »jointtarget«. V naslednjem koraku s pomočjo ukaza »WZHomeJointDef« kreiramo varno pozicijo in njeno možno odstopanje od idealne. Po tem je potrebno definirati še signal, ki se proži, ko je robot v dani poziciji s pomočjo ukaza »WZDOSet«.




```
    VAR wztemporary Varna_poza;
PROC def_varne_poz()
    ! Oblika bo obmocje sklepov
    VAR shapedata obmocje_sklepov;
    ! Definiranje varne pozicije v zapisu sklepov posameznih kotov
    CONST jointtarget j_varna_poza:=[[0,-39,-11.158,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
    ! Definiraj mozno odstopanje od varne pozicije
    CONST jointtarget delta_varna_poza:=[[1,1,1,1,1,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
    ! Definiraj varno pozicijo okoli sklepov
    WZHomeJointDef \Inside,obmocje_sklepov,j_varna_poza,delta_varna_poza;
    ! Ce je TCP znotraj, aktiviraj DO
    WZDOSet \Temp,Varna_poza \Inside,obmocje_sklepov,Varna_poz,1;
ENDPROC
```

Slika 11.20: Varna pozicija robota in kreiranje območja varne pozicije

11.8 Preverjanje kakovosti transportiranih izdelkov

V avtomatizaciji je potrebno preverjati kakovost izdelkov. V primeru paletizacije gre za preverjanje pravilnih dimenzij škatel in nalepk, ki so nalepljene na škatle. To preverjanje se mora izvajati že pred procesom paletizacije. Dogaja se, da naročnik izrazi željo po možnosti odlaganja transportiranih izdelkov na mesto kontrole. V tem primeru za programiranje tega potrebujemo prekinitveno rutino, ki bo to izvedla. Uporabili bomo prekinitveni signal oz. »interrupt«. Tega program konstantno preverja, ali je prožen ali ne. V trenutku, ko se pojavi prekinitev, se izvede prekinitvena rutina. Ker preverjanje kakovosti izdelkov ni prioriteta, pri tem gibanje robota ni potrebno zaustaviti. S pomočjo prekinitvenega signala bomo prožili signal za preverjanje izdelkov. Ko bo robot izdelek prijel, pa se bo preverjala vrednost signala za preverjanje izdelkov. Če bo ta na 1, se bo izvedla rutina odlaganja izdelka na mesto preverjanja, v nasprotnem se bo program normalno izvajal.

Če uporabljamo prekinitev, je potrebno najprej definirati prekinitveni signal, ga povezati s prekinitveno rutino in aktivirati sledenje prekinitvenemu signalu. Slika 11.21 prikazuje definiranje, povezavo in kreiranje prekinitvenih signalov in rutine.

```
VAR intnum test_prekinitev;
VAR bool test1;
```

Definiranje prekinitvenih signalov

```
PROC init_interrupt()
    CONNECT test_prekinitev WITH Test_komadov;
    ISignalDI Prekinitev_test,1,test_prekinitev;
    !Aktivacija interruptov
    IEnable;
ENDPROC
```

Povezovanje prekinitvenih signalov s prekinitveno rutino in aktivacija prekinitve

```
TRAP Test_komadov
    ISleep test_prekinitev;
    test1:=true;
ENDTRAP
```

Definiranje prekinitvene rutine ter postavitev signala test1. Prekinitvenega signala potem ne spremljamo več.

Slika 11.21: Definiranje, povezava in aktivacija prekinitvenih signalov s prekinitveno rutino

S tem ko postavimo signal test1 na vrednost 1, je potrebno spremeniti glavni program. Preden pri orientaciji zlaganja A ali B nadaljujemo z zlaganjem, preverimo vrednost signala test1. Če je ta na 1, izvršimo rutino odlaganja izdelka na kontrolno mesto. V nasprotnem se program normalno izvaja. Po odlaganju izdelka na kontrolno mesto je potrebno resetirati signal test1 na 0 in ponovno aktivirati spremljanje prekinitvenega signala. Slika 11.22 prikazuje modifikacijo glavnega programa, resetiranje signala test1 in ponovno aktivacijo spremljanja prekinitvenega signala.

```
WHILE vrstica<10 DO
    nadaljujA:
    Program_prijemanja;
    IF test1=True THEN
        MoveL Prijemanje_gor_test,v500,z50,Prijemalo_paletizacija_1\WObj:=KS_Prijemanje;
        Izmet_komadov_v_prijemalu;
        test1:=FALSE;
        IWatch test_prekinitev;
        !Po prekinitvi nadaljuj zgoraj
        GOTO nadaljujA;
    ENDIF
    Odlaganje_A_orientacija;
ENDWHILE
PROC Izmet_komadov_v_prijemalu()
    MoveJ Pred_Odlaganje_izmet,v500,z50,Prijemalo_paletizacija_1\WObj:=wobj0;
    MoveL Odlaganje_izmet,v400,fine,Prijemalo_paletizacija_1\WObj:=wobj0;
    OdpriPrijemalo;
    MoveL Pred_Odlaganje_izmet,v500,z50,Prijemalo_paletizacija_1\WObj:=wobj0;
    MoveAbsJ Domov,v400,z10,Prijemalo_paletizacija_1\WObj:=wobj0;
ENDPROC
```

Preverjanje signala test1

Resetiranje signala test1 in ponovna aktivacija prekinitvenega signala

Po končanju nadaljuj pri ukazu nadaljujA.

Slika 11.22: Modifikacija glavnega programa, primer za orientacijo A ter kreiranje rutine za kontrolo izdelkov

11.9 Preverjanje prisotnosti izdelkov v prijemalu

V večini primerov se za prijemanje izdelkov uporabljajo pnevmatski cilindri. Teoretično se lahko zgodi, da zaradi vibracij, ki jih povzroča robot, izpade dovodna cevka za zrak in tako med prenašanjem izgubimo izdelek. Za zaznavanje prisotnosti izdelkov oz. izgubo le-teh obstaja več možnih načinov. Na prijemalo se lahko namesti dodatna fotocelica, ki skrbi za preverjanje prisotnosti. Če fotocelica ni nameščena, si pomagamo s čitalci pozicije pnevmatskih cilindrov oz. tako imenovanih »reed switch« ali REED stikal [49]. To so stikala, ki zaznavajo prisotnost batnice cilindra in jih namestimo v utore na ohišju pnevmatskega cilindra. Z njimi zaznavamo skrajne lege cilindrov, torej kdaj je cilinder odprt in kdaj zaprt. Tako lahko za preverjanje prisotnosti izdelka preverjamo, da REED stikalo ni vklopljeno oz. da prijemalo med prenašanjem ni zaprto. Prijemalo bi se zaprlo v primeru izgube izdelka, saj je cilinder še vedno prožen. Za to potrebujemo dva dodatna prekinitvena signala, s pomočjo katerih preverjamo zaprtost prijemala. Zaprtost prijemala preverjamo samo med prenašanjem izdelkov. Prej ali po tem nas zaprtost prijemala ne zanima.

Takoj za deklaracijo prekinitvenih signalov in rutin, prekinitvena signala začasno onemogočimo. Signala omogočimo šele potem, ko izdelke v prijemalu primemo. Med prenašanjem izdelkov tako spremljamo, ali pride do kakšne izgube izdelka. Po odlaganju izdelkov do ponovnega prijemanja zopet začasno onemogočimo spremljanje prekinitvenih signalov. Če se prekinitev zgodi, se mora gibanje robota v trenutku zaustaviti. V prekinitveni rutini je potrebno zaustaviti robota in potem javiti nadzornemu sistemu (PLK) napako. Prav tako je pri izgubi paketa potrebno posredovanje operatorja. Tako je potrebno z vizualnim in zvočnim signalom preko signalnega semaforja opozoriti na napako. Kako reagiramo potem, je odvisno od proizvodnega procesa in od odločitve operatorja. Ali je dejansko prišlo do izgube paketa ali pa je mogoče prišlo samo do okvare REED stikala. Slika 11.23 prikazuje kreiranje prekinitvenih signalov in rutin za preverjanje izgube izdelkov med prenašanjem.

```
PROC init_interrupt()
    CONNECT test_prekinitev WITH Test_komadov;
    ISignalDI Prekinitev_test,1,test_prekinitev;
    !Deklaracija interrupta za levi cilinder
    CONNECT Prijemalo_reedL WITH Paket_izgubljen;
    ISignalDI Prijemalo_L_zaprto,1,Prijemalo_reedL;
    ! V času deklaracije prekinitveni signal še ni potrebeno spremljati
    ISleep Prijemalo_reedL;
    !Deklaracija interrupta za desni cilinder
    CONNECT Prijemalo_reedD WITH Paket_izgubljen;
    ISignalDI Prijemalo_D_zaprto,1,Prijemalo_reedD;
    ISleep Prijemalo_reedD;
    !Aktivacija interruptov
    IEnable;
ENDPROC

TRAP Paket_izgubljen
    StopMove \Quick; ! Hitra zaustavitev robota
    ! Preko določenega signala je potrebno javiti napako nadzornemu PLK-ju.
    ! Potrebno je poslati zahtevo po posredovanju operaterja, vizualni in zvočni signal.
ENDTRAP

PROC OdpriPrijemalo()
    SetDO Zapri_prijemalo,0;
    SetDO Odpri_prijemalo,1;
    WaitTime 0.5;
    ISleep Prijemalo_reedD;
    ISleep Prijemalo_reedL;
ENDPROC

PROC ZapriPrijemalo()
    SetDO Odpri_prijemalo,0;
    SetDO Zapri_prijemalo,1;
    WaitTime 1;
    IWatch Prijemalo_reedD;
    IWatch Prijemalo_reedL;
ENDPROC
```

Deklaracija dveh novih prekinitvenih signalov

Prekinitveni signal začasno onemogočimo.

Za oba prekinitvena signala uporabimo isto prekinitveno rutino.

Začasno onemogočimo spremljanje prekinitvenega signala.

Omogočimo spremljanje prekinitvenega signala.

Slika 11.23: Kreiranje in definiranje prekinitvenih signalov in rutin za preverjanje izgube izdelkov

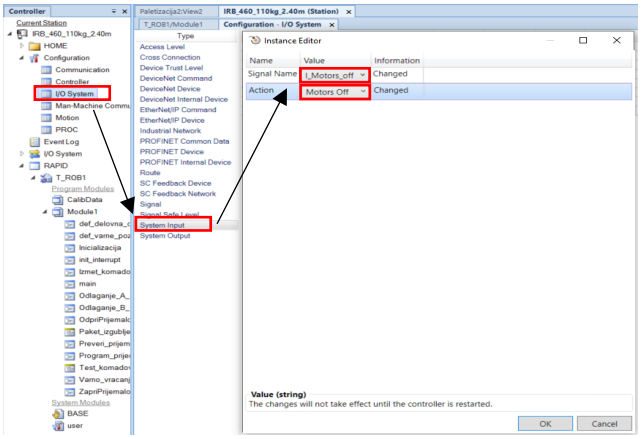
11.10 Povezava sistemskih signalov robota z nadzornim sistemom (PLK)

Do sedaj smo ustvarili program, ki zagotavlja delovanje industrijske robotske celice v avtomatskem režimu. Če ne prihaja do izgub paketov, bo linija delovala brez posredovanja operaterja. A v takšni konfiguraciji nadzorni PLK nima prav nobene kontrole nad robotom. Včasih se zgodi, da bi bilo potrebno robotski program pognati od začetka, ga resetirati. Za to je potrebno robota postaviti v ročni način delovanja in ročno podati zahtevo za resetiranje programa. Če želimo to avtomatizirati, pa je potrebno uporabiti za to namenjene signale. Signali, ki so na voljo, so zopet odvisni od posameznega proizvajalca industrijskih robotov, zato je nujno preveriti in prebrati navodila proizvajalca industrijskega robota. V tem konkretnem primeru si bomo pogledali sistemske signale za ABB-jeve industrijske robote. Na kaj je tukaj potrebno opozoriti že takoj na začetku, je, **da preklopa med avtomatskim in ročnim režimom ne moremo avtomatizirati** oz. ne moremo pooblastiti PLK, da to stori. Ta funkcija je onemogočena zaradi varnostnih razlogov. V večini primerov lahko avtomatiziramo resetiranje programa, resetiranje napak,

resetiranje oz. potrjevanje napake izklopa v sili, vklop in izklop motorjev, hitre zaustavitve robota v primeru izklopa v sili ter zaustavitve in zagon izvajanja programa. Večino teh stanj lahko tudi spremljamo. Spremljamo lahko, ali se je program resetiral, ali je robot v ročnem ali v avtomatskem načinu delovanja, ali ima motorje pod napetostjo ali ne ter ali je v napaki. Zelo pomembna je tudi funkcija spremljanja, ali **se robot ne nahaja na programirani trajektoriji**. Če je ta signal neaktiven, smo lahko prepričani, da se pri izvajanju programa ne bo zaletel, razen če smo naredili napako že v snovanju programa. Če pa je ta signal aktiven, je nekdo premaknil robota izven programirane trajektorije. V tem primeru lahko pri nadaljnjem izvajanju programa pride do trka z okoliško opremo. Nadzorni PLK tako robotu ne sme dati dovoljenja za nadaljevanje izvajanja programa, razen v primeru, da to potrdi PLK-operater. Tabela 11.1 prikazuje najpogosteje uporabljene sistemske vhode in izhode za industrijskega robota proizvajalca ABB. S temi ukazi lahko PLK prevzame »nadzor« nad industrijskim robotom. Če želimo uporabljati sistemske signale, jih je potrebno povezati s signali, ki jih robot prejme po serijski povezavi, najpogosteje preko PROFINET protokola. Signal, ki ga robot prejme preko PROFINET, normalno definiramo, tako kot je to bilo že storjeno v poglavju 8.3. Ta signal je potrebno povezati s sistemskim signalom. Slika 11.24 prikazuje povezovanje sistemskega signala in signala, uporabljenega v PROFINET komunikaciji.

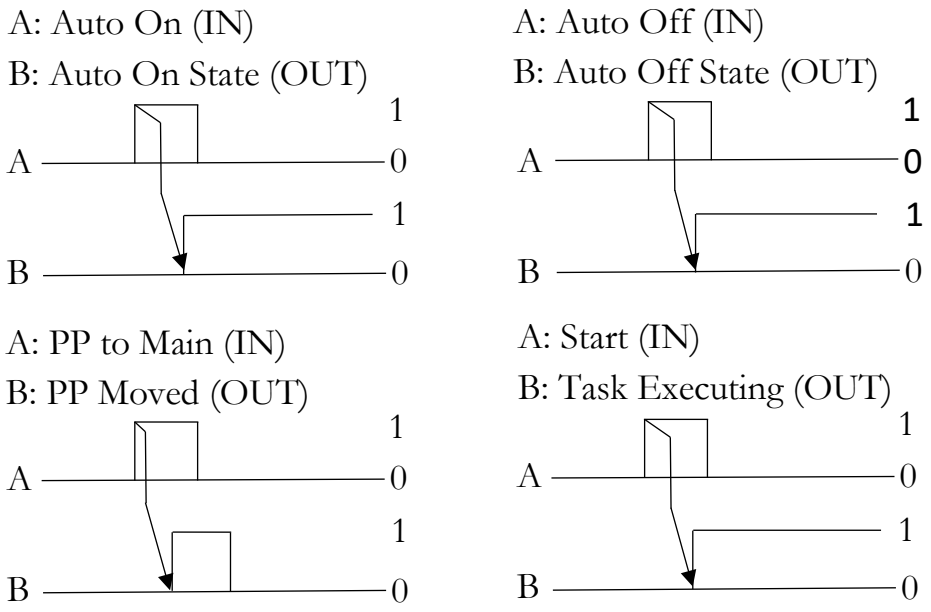
Tabela 11.1: Sistemski vhodi in izhodi

Sistemski vhodi	Opis	Sistemski izhodi	Opis
Motors Off		Motors Off State	
Motors On		Motors On State	
PP to Main	Resetiranje programa	PP Moved	Program resetiran
Quick Stop		Robot not on Path	Robot ni na programirani poti
Reset Emergency Stop		Emergency Stop	
Reset Execution Error		Execution Error	
Stop		Auto On	Avtomatski režim
Start		Task Executing	Program se izvaja

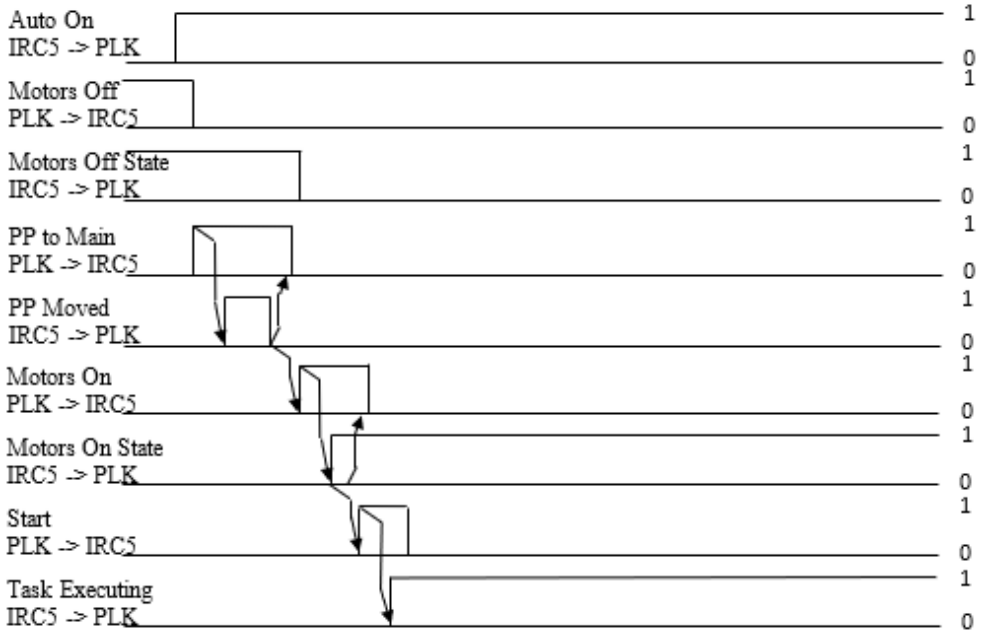


Slika 11.24: Povezovanje sistemskih signalov s signali PROFINET-a

Uporaba sistemskih signalov zahteva upoštevanje tudi časovnih zamikov med preklapljanjem določenih stanj. Uporaba npr. sistemskega signala »PP to Main« ni možna, če motorji robota niso izklopljeni. Prav tako je potrebno upoštevati, da se za vklop in izklop motorjev uporablja zgolj signal z neko končno dolžino, nekaj 100 ms. Stanje motorjev je potem dokončno oz. traja do preklopa. Slika 11.25 prikazuje časovni potek glavnih sistemskih signalov posamično. Slika 11.26 prikazuje kronološko zaporedje vklopa in izklopa sistemskih signalov v avtonomnem režimu delovanja industrijskega robota.



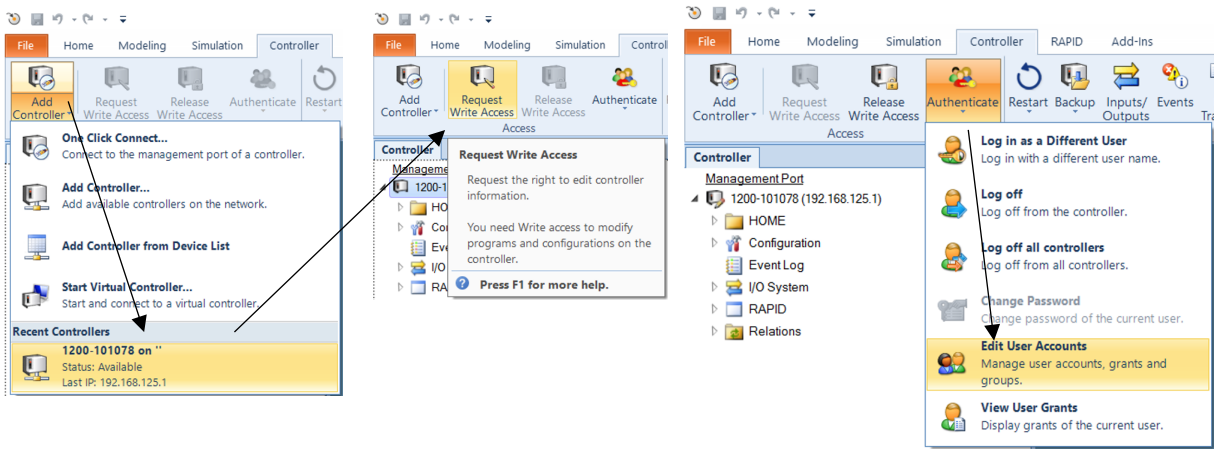
Slika 11.25: Časovni potek glavnih sistemskih signalov posamično



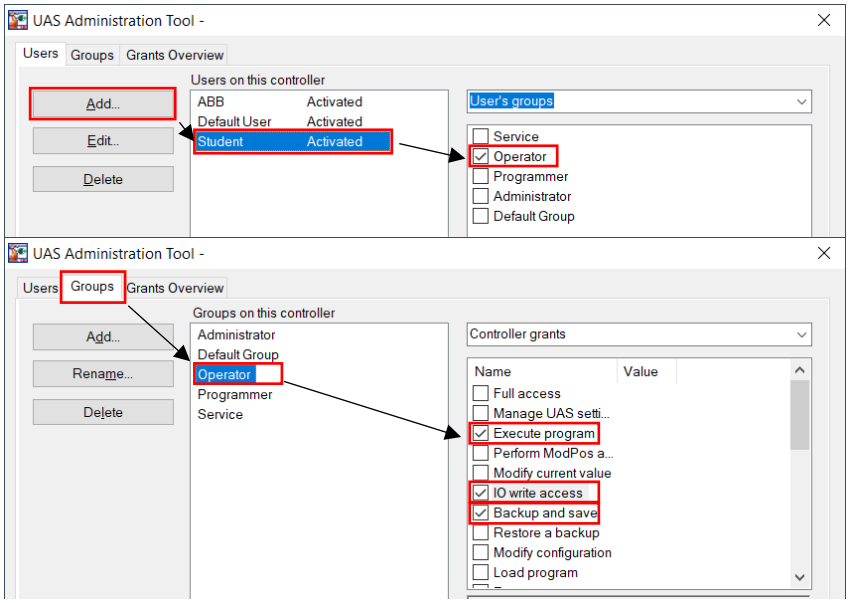
Slika 11.26: Kronološko zaporedje vklopa in izklopa sistemskih signalov

11.11 Spreminjanje uporabnikov in njihovih dovoljenj na krmilniku industrijskega robota

Ko je programer industrijskih robotov zaključil s svojim delom, pri čemer je upošteval vse možne scenarije, ki se lahko zgodijo, zavzel najbolj optimalno trajektorijo vračanja robota iz vseh predvidljivih pozicij v varno točko in pripravil program, ki je pregleden, smiseln in dovolj dobro komentiran, mora kreirati še uporabnika, ki ima omejen dostop do programa. Koliko pravic urejanja programa, dostopa do DI/DO-signalov, shranjevanja programa ter popravljanja točk bo dodelil, je odvisno tudi od naročnika. Načeloma je končni uporabnik robotski operater s pomanjkljivim znanjem programiranja industrijskih robotov. Tako mu je potrebno omejiti dostop do določenih funkcij. Na primeru ABB industrijskem robotu to storimo s pomočjo programa RobotStudio. Pri tem vzpostavimo povezavo z robotskim krmilnikom in zaprosimo za dovoljenje za dostop. Preko ukaza »Authenticate« in ukaza »Edit User Accounts« kreiramo novega uporabnika ter ga aktiviramo. Uporabniku predpišemo tudi geslo. Temu uporabniku moramo določiti še uporabniško skupino. Uporabniška skupina predpisuje uporabniška dovoljenja. Slika 11.27 prikazuje postopek povezave z realnim krmilnikom in odpiranje okna za urejanje uporabnikov. Slika 11.28 prikazuje kreiranje novega uporabnika in dodeljevanje pravic.



Slika 11.27: Povezava z realnim krmilnikom in odpiranje okna za urejanje uporabnikov



Slika 11.28: Kreiranje novega uporabnika ter dodeljevanje pravic

11.12 Vprašanja za poglobljen študij

1. Na spletni strani Estudij, kjer se nahajajo predmeti Industrijska robotika, Robotizacija ali Roboti in robotizacija, iz datoteke RobotStudio prenesite datoteko z imenom »Paletizacija.rspag«. Datoteko odprite s pomočjo programa RobotStudio. Zahteve za zlaganje so podane v poglavju 11.1. Definirajte vse potrebne koordinatne točke, varno pozicijo robota in vse vmesne točke.

Ugotovitve:

2. Napišite program, ki bo najprej zlagal orientacijo A in potem orientacijo B s šestimi sloji. Pri tem si pomagajte z diagramom poteka iz podpoglavja 11.3.

Ugotovitve:

3. Ko boste končali s programiranjem glavnega programa je potrebno urediti preverjanje stanja prijemala, ali je prijemalo odprto pred začetkom izvajanja glavnega programa. Ne sme se zgoditi, da gre robot z zaprtim prijemalom po izdelke, saj jih bo s tem uničil. V nadaljevanju v poglavju 12, si lahko pogledate princip uporabe upravljalca napak za preverjanje stanja prijemala.

Ugotovitve:

4. V proizvodnji se zgodi, da je potrebno program resetirati zaradi napake, ki se pojavi v proizvodnem procesu. Zagotovite, da se bo robot, ne glede na to, kje v svojem delovnem območju se nahaja, varno vrnil v varno oz. začetno pozicijo. Pri tem si pomagajte s podpoglavjem 11.4.

Ugotovitve:

5. Standardni industrijski roboti imajo na prvi osi kot zasuka omejen na približno +/– 170 °. S takšno omejitvijo in nepazljivim operaterjem se lahko robot zaleti v varnostno ograjo. Z omejevanjem zasuka poskrbite, da se to ne more zgoditi. Pri tem si pomagajte s podpoglavjem 11.5.

Ugotovitve:

6. V marsikaterih aplikacijah se zahteva, da nadzorni krmilnik oz. PLK "ve", kje se nahaja industrijski robot. V tem primeru najpogosteje uporabljamo delovna območja. Kreirajte dve delovni območji nad vhodnim transporterjem in nad izhodno paleto. Pri tem si pomagajte s funkcijo WorldZones in podpoglavjem 11.6.

Ugotovitve:

7. Podobno kot z delovnimi območji je smiselno nadzornemu krmilniku javiti, kdaj se robot nahaja v varni poziciji. S tem nadzorni krmilnik ve, da je varno zagnati izvajanje programa. S pomočjo funkcije WorldZones definirajte varno točko robota in kreirajte signal, ki se bo pri tem sprožil. Pomagajte si s podpoglavjem 11.7.

Ugotovitve:

8. V proizvodnih linijah je marsikdaj prisotno tudi preverjanje kakovosti izdelkov. Izdelki, ki jih robot zlaga na paleto, bi morali biti že testirani, a za vajo naredite kontrolo kakovosti izdelkov. Ko se sproži signal »Prekinitev_test«, ni potrebno prekiniti delovanja programa, ampak ob naslednjem prijemanju izdelkov se naj ti odložijo na kontrolno mesto. Pri tem ustvarite prekinitveno rutino in prekinitveni signal. Postavite spremenljivko na 1 in poskrbite, da se bodo izdelki ob naslednjem prijemanju odložili na kontrolno mesto. Pri tem si pomagajte s podpoglavjem 11.8.

Ugotovitve:

9. Večina robotskih prijemal je opremljena zgolj s senzorjema, ki javljata končni poziciji prijemala. Ali je prijemalo odprto ali je zaprto? Napišite program, ki bo s pomočjo prekinitev in prekinitvenih rutin preverjal prisotnost izdelka v prijemalu in bo v trenutku izgube izdelka zaustavil gibanje robota in javil napako na učni konzoli. Pri tem si pomagajte s podpoglavjem 11.9 in ukazom »TPWrite«, s pomočjo katerega boste napisali napako na učni konzoli.

Ugotovitve:

10. Delovanje industrijskih robotov v aplikacijah mora biti popolnoma avtonomno. To pomeni, da ima nadzorni krmilnik popolni nadzor nad robotom, če obstaja. Tudi nad delovanjem motorjev, resetiranjem programa, resetiranjem napak, resetiranje signala izklop v sili, ustavljanjem programa, zagonom programa ter preverjanjem, ali se robot nahaja na programirani trajektoriji. S pomočjo podpoglavja 11.10 kreirate naslednje signale in jih povežite s sistemskimi signali industrijskega robota.

Digitalni vhodi	DeviceMapping	Sistemski vhodi	Digitalni izhodi	DeviceMapping	Sistemski izhodi
DIMotorsOff	0	MotorsOff	DOMotorsOffState	0	MotorsOffState
DIMotorsOn	1	MotorsOn	DOMotorsOnState	1	MotorsOnState
DIPPtoMain	2	PPtoMain	DOPPMoved	2	PPMoved
DIQuickStop	3	QuickStop	DORobotNotOnPath	3	RobotNotOnPath
DIResetEmergencyStop	4	ResetEmergencyStop	DOEmergencyStop	4	EmergencyStop
DIResetExecutionError	5	ResetExecutionError	DOExecutionError	5	ExecutionError
DIStop	6	Stop	DOAutoOn	6	AutoOn
DIStart	7	Start	DOTaskExecuting	7	TaskExecuting

Ugotovitve:

11. Ko je robotski programer zaključil s svojim delom, robotsko celico preda v roke robotskih operaterjev. Na žalost imajo ti v večini primerov pomanjkljivo znanje o programiranju in delovanju industrijskih robotov. Zadnja naloga robotskega operaterja je spreminjanje uporabnikov in njihovih dovoljenj. Na krmilnik robota dodajte novega uporabnika z imenom Robotski_operater in mu dodelite pravice za izvajanje programa, pisanje vhodno/izhodnih spremenljivk, ustvarjanja varnostne kopije in shranjevanje programa, obnavljanje varnostne kopije ter nalaganje programa. Pri tem si pomagajte s podpoglavjem 11.11.

Ugotovitve:

Univerzitetna založba
Univerze v Mariboru

12 Upravljanje z napakami »error handler« pri izvajanju programa

Kadar imamo opravka s preprostimi aplikacijami, kot so npr. »pick and place« aplikacije, preverjamo orodje na način, kot je to bilo predstavljeno v poglavju 11.3. Pri začetnem zagonu programa preverimo, ali je prijemalo odprto, če ni, ga odpremo in začnemo z izvajanjem programa. Pri tem avtomatsko privzamemo, da se bo prijemalo po proženju signala odprlo. Res je, da robot ne nadaljuje, če se prijemalo ne odpre, in tako dolgo "vztraja" pri proženju pnevmatskega cilindra, dokler REED stikalo ni aktivno. Kako pa naj odreagiramo, kadar se kljub "vztrajanju" robota pnevmatski cilinder ne premakne oz. kadar REED stikalo ne da signala, da je cilinder v željeni poziciji? V tem konkretnem primeru se poslužujemo upravljalcev napak oz. ang. »error handler«. Prav gotovo se lahko zgodi, da se je pnevmatski cilinder ustavil nekaj milimetrov pred REED stikalom ravno pri preverjanju prijemala. V tem primeru ni priporočljivo robota pustiti v situaciji "vztrajanja" za odpiranje prijemala, temveč je potrebno definirati, kako naj robot odreagira. Potrebno je npr. definirati, koliko poizkusov želimo, da izvede, po preteku števila preizkusov pa nadzorni krmilnik obvesti o napaki na prijemalu. To je primer napake, ki jo definira programer. Vsaki napaki, ki jo definira programer, je potrebno predpisati tudi unikatno število. S tem programu določimo številko napake ter v nadaljevanju, kako naj ob tej napaki reagira, ne da se izvajanje programa zaustavi, če je to seveda možno.

Znotraj programa se lahko pojavijo tudi napake pri izvajanju programa ang. »execution error«, ki predstavljajo abnormalno situacijo, povezano na izvajanje določenega dela programske kode. Vsaka takšna napaka predstavlja potencialno nevarnost za nadaljnje

izvajanje programa, zato se izvajanje prenese v program upravljanja napak oz. »error handler«, ki poskrbi za to, kako program tretira napake. Koncept upravljalca napak omogoča odziv na napake ter obnovitev v stanje brez napak, ne da se program zaustavi, seveda, če je to možno. Če to ni možno, upravljalec napak omogoča elegantno zaustavitev oz. prekinitev izvajanja programa.

Če ne uporabljamo upravljalca napak, se ob pojavu napake sproži interni upravljalec napak, ki avtomatsko zaustavi program v točki, kjer je prišlo do napake. Takšne zaustavitve so nezaželene, saj potrebujejo posredovanje operaterja robota, prekinjajo proizvodnjo ter ustvarjajo izgubo.

Ob pojavu napake se aktivira rutina, kjer je definiran upravljalec napake. Trenutna napaka se potem tretira znotraj upravljalca napake. Veliko napak je tako že vnaprej definiranih. Tabela 12.1 prikazuje samo nekaj od njih [50, 51]. Vendar definiranost napake še ne pomeni, da bo ta tretirana znotraj upravljalca napake. Tako je potrebno še definirati rutino upravljalca specifične napake. Ta napaka je lahko že vnaprej definirana ali pa jo lahko definiramo sami. Na voljo imamo števila od 1–90 za definiranje specifičnih napak v programu. Če želimo definirati specifično napako, uporabimo ukaz »RAISE«, ki specifični napaki priredi definirano število. V nadaljevanju bo podan primer uporabe vnaprej definiranih napak ter primer kreiranja specifičnih napak.

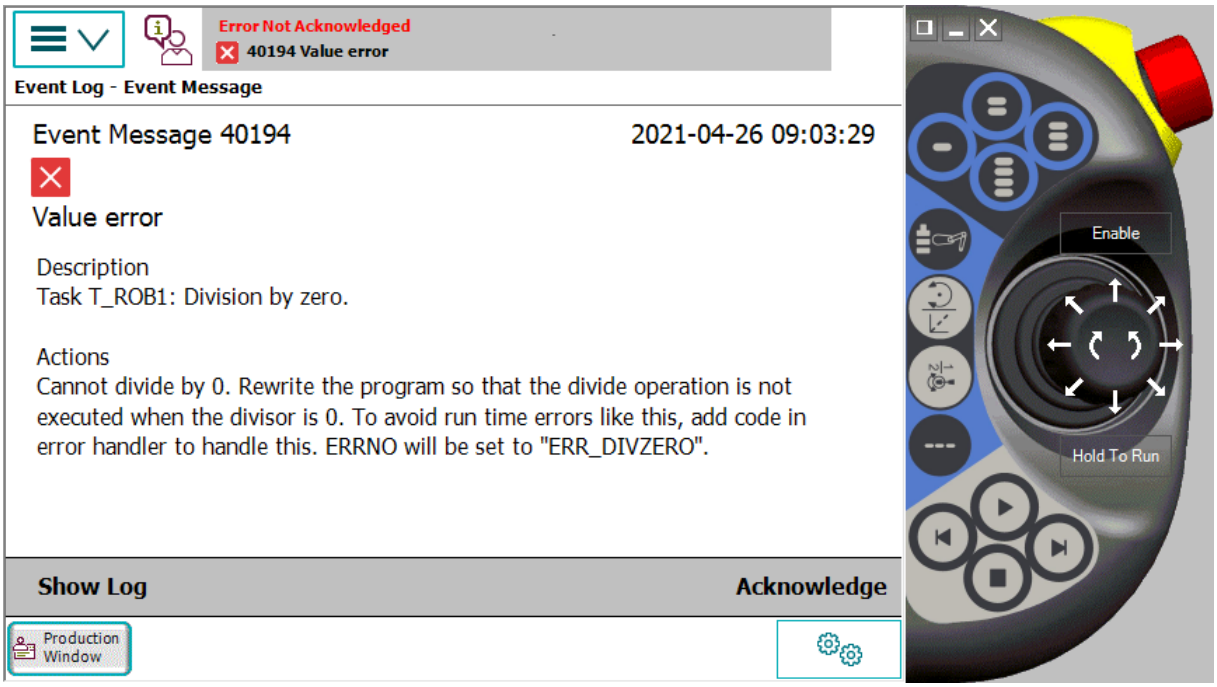
Tabela 12.1: Primer v naprej kreiranih napak

Vrsta napake:	Opis napake:
ERR_AO_LIM	Analogni signal je izven limite
ERR_DIVZERO	Deljenje z 0
ERR_COLL_STOP	Prekinitev gibanja zaradi kolizije
ERR_SIGSUPSEARCH	Signal že aktiven v primeru iskanja pozitivnega signala
ERR_WHLSEARCH	Ni pozitivnega signala v primeru iskanja pozitivnega signala
ERR_WAIT_MAXTIME	Časovni interval presežen pri signalih WaitDI, WaitDO …

12.1 Uporaba vnaprej definiranih napak v upravljalcu napak »error handler«

V nadaljevanju bo prikazana uporaba vnaprej definirane napake za deljenje s številom 0. Vemo, da takšno deljenje ni dovoljeno. Z uporabo upravljalca napak definiramo, kaj se bo zgodilo v primeru deljenja s številom 0. Za to bomo potrebovali funkcijo, znotraj katere bomo dve števili med sabo delili. Kreiramo funkcijo z imenom deljenje in ji dodelimo dva argumenta. V tem konkretnem primeru bomo s pomočjo funkcije »TPWrite« zapisali rezultat deljenja na zaslon učne konzole robota, z ukazom »TPErase« pa bomo zbrisali prejšnji zapis na zaslonu. Če pri deljenju dveh števil ne uporabimo deklariranega

upravljalca napak, bomo pri deljenju dobili naslednjo napako in zaustavitev izvajanja robotskega programa. Slika 12.1 prikazuje izpis napake. Takšna zaustavitev programa ni zaželena. Zato je potrebno uporabiti upravljalca napak in deklarirati, kako naj se program odzove na takšno napako.



Slika 12.1: Izpis napake pri deljenju z nič, kadar upravljalec napak ni posebej deklariran.

V tem primeru gre za primarno vnaprej deklarirano napako v krmilniku robota. Napaka ima oznako »ERR_DIVZERO« in ima svojo unikatno število. Kadar pričakujemo napako, moramo v programu, podprogramu ali pa znotraj funkcije na koncu zapisati ukaz »ERROR«. Zatem deklariramo, kako naj se program odzove na to napako. V primeru deljenja števila z nič želimo izpisati na zaslonu učne konzole, da je prišlo do deljenja s številom nič. To storimo z ukazom »TPWrite«. S tem opozorimo robotskega operaterja, da je prišlo do napake, za nadaljnje izvajanje programa pa imamo več možnosti. Program lahko prekinemo, ponovno izvedemo isti ukaz, izvedemo ukaz za tem, ki je povzročil napako, ali pa privzamemo privzeto vrednost in se vrnemo v rutino, ki je povzročila napako. Tabela 12.2 prikazuje uporabljene ukaze za nadaljevanje izvajanja programa po pojavu napake. Kaj bomo izbrali, je seveda odvisno od aplikacije, kjer sodeluje industrijski robot. V tem konkretnem primeru bomo uporabili ukaz »RETURN«. Torej program se naj vrne v rutino, ki je povzročila napako z rezultatom, ki je enak deljencu. Slika 12.2 prikazuje programsko kodo za uporabo vnaprej definirane napake za deljenje s številom nič ter princip nadaljevanja programa po pojavu napake.

Tabela 12.2: Uporabljeni ukazi za nadaljnje izvajanje programa po pojavu napake

Izvajanje programa po pojavu napake	Opis ukaza
EXIT	V primeru napake zaustavimo izvajanje programa.
RETRY	Ponovno izvedemo ukaz, ki je sprožil napako. Privzeto število ponovitev je 4.
TRYNEXT	Izvedemo ukaz za ukazom, ki je povzročil napako.
RETURN	Privzamemo privzeto vrednost in se vrnemo v rutino, ki je povzročila napako, če je upravljalec napak v funkciji. V nasprotnem se vrnemo v rutino, iz katere je bila klicana napaka.

```
PROC main()
    TPErase;
    rezultat:=deljenje(10,0);
ENDPROC

FUNC num deljenje(num x,num y)
    RETURN x/y;
ERROR
    IF ERRNO=ERR_DIVZERO THEN
        TPWrite "Stevilo ni mozno deliti z 0";
        RETURN x;
    ENDIF
ENDFUNC
```

Brisanje zapisa na zaslonu

Kreiranje funkcije deljenja dveh števil

Definiranje potencialne napake in deklariranje, kako naj robot reagira.

Slika 12.2: Primer uporabe vnaprej deklarirane napake za deljenje z nič

12.2 Uporaba specifično kreiranih napak v upravljalcu napak »error handler«

V prejšnjem poglavju je bil prikazan primer uporabe vnaprej definiranih napak proizvajalca robotov. V nadaljevanju bo prikazan primer uporabe specifično kreiranih napak, ki jih kreira robotski programer. Primer bo demonstriran na preverjanju prijemala. Po navadi ima prijemalo nameščeno REED stikala, ki odčitavajo pozicijo batnice znotraj pnevmatskega cilindra. Tako vemo, ali je prijemalo odprto ali zaprto. Pred začetkom izvajanja programa moramo preveriti, ali je prijemalo odprto. Ta primer je bil uporabljen že v poglavju 11.3. Vendar smo takrat avtomatsko privzeli, da se z aktivacijo signala prijemalo odpre, ničesar pa nismo definirali, kaj se zgodi v primeru, da se prijemalo ne odpre. V programu bi robot avtomatsko čakal na signal, da so čeljusti prijemala odprte. Tako bi celotna robotska celica čakala, ne da bi se kjerkoli izpisala napaka. V tem konkretnem primeru bo uporabljen ukaz »RETRY«, da program ponovi odpiranje prijemala. Če v štirih poizkusih program ne bo uspel odpreti prijemala, se bo na zaslonu učne konzole izpisala napaka za odpiranje orodja ter informacijo, da je število poizkusov preseglo limit. Program je sestavljen tako, da simulira odpiranje prijemala s pomočjo

ukazov preko učne konzole. Na zaslonu se bo prikazalo vprašanje, ali je prijemalo v varni poziciji. Na voljo imamo dva odgovora: DA in NE. Vsak odgovor ima svoje število, DA ima 2 in NE ima 4. Na podlagi našega odgovora se potem generira napaka ter kako naj robotski program odreagira na to napako. Pri tem nastopi upravljalec napak, ki ga mora zopet definirati robotski programer. V tem primeru nas program vpraša, ali želimo ponoviti odpiranje prijemala ali pa želimo prekiniti s programom. Zopet s pomočjo ukazov preko učne konzole odgovorimo. Odgovoru PONOVI se dodeli število 1 in odgovoru PREKINI se dodeli število 5. Glede na naš odgovor se program ponovi ali pa prekine. Slika 12.3 prikazuje celoten program specifično kreiranih napak ter kako se na te napake program odzove. Slika 12.4 pa prikazuje, kako se program izvaja na učni konzoli industrijskega robota.

```
PROC main()
    TPErase;
    TestPrijemala;
ENDPROC
PROC TestPrijemala()
    VAR num Rezultat:=0;
    ! -- Preveri prijemalo, ali je v zacetni poziciji?
    Test_prijemala;
    IF TestPrijemalaRezultat=4 RAISE 10;
ERROR
    IF (errno=10) THEN
        WHILE (Rezultat<>5) DO
            TPErase;
            TPWrite "-- Prijemalo ni v zacetni poziciji --";
            TPWrite "";
            TPWrite "Lahko ponovno poizkusite";
            TPWrite "ali pa prekinete test prijemala ";
            TPWrite "in premaknete robota v varno pozicijo.";
            TPReadFK Rezultat,"Izberi zeljeno:","Ponovi","","","","Prekini";
            IF (Rezultat=1) THEN
                !Ponovi test prijemala!
                Test_prijemala;
                IF TestPrijemalaRezultat=4 RETRY;
            ENDIF
            IF (Rezultat=5) or TestPrijemalaRezultat=2 RETURN;
        ENDWHILE
    ENDIF
ENDPROC

PROC Test_prijemala()
    !Procedura premika prijemala, simulirano;
    TPWrite "-- Simulacija testiranja prijemala --";
    TPReadFK TestPrijemalaRezultat,"Prijemalo v varni poziciji?:","","Da","","Ne","";
ENDPROC
```

Če je TestPrijemalaRezultat enak 4, aktiviramo napako s številko 10.

Preverjanje številke napake

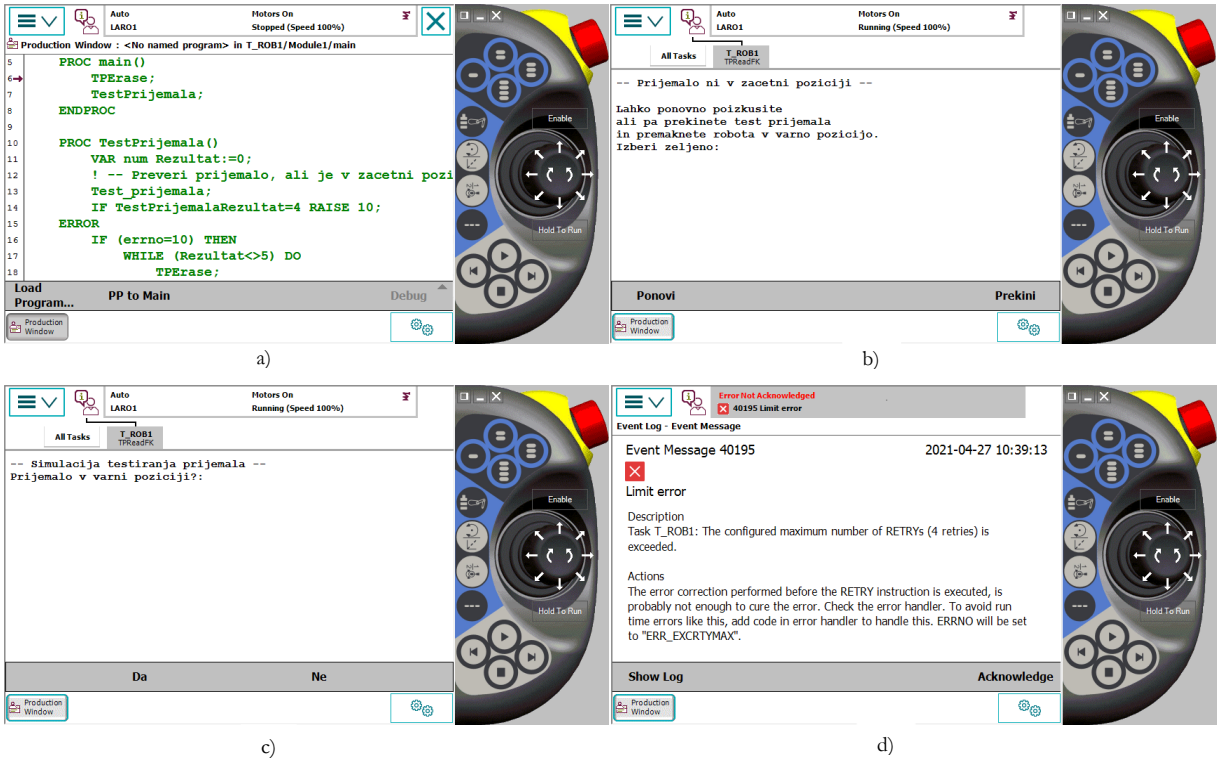
Ponovi = 1; Prekini = 5

Ponovi izvajanje preverjanja prijemala.

Vračanje programskega kazalca na mesto vstopa v rutino preverjanja

Simuliranje prijemala v varni poziciji
Da = 2; Ne = 4

Slika 12.3: Primer uporabe specifično kreiranih napak na primeru odpiranja prijemala

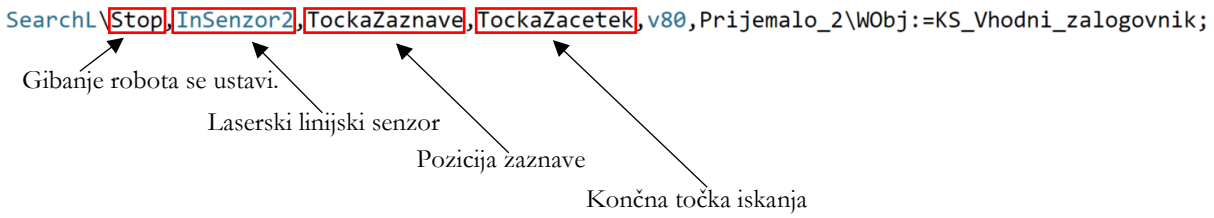


Slika 12.4: a) Začetek izvajanja programa, b) Obvestilo, da prijemalo ni odprto, c) Simulacija odpiranja prijemala in d) Napaka po preteku omejenega števila ponovitev

12.3 Primer industrijske aplikacije in uporabe upravljalca napak

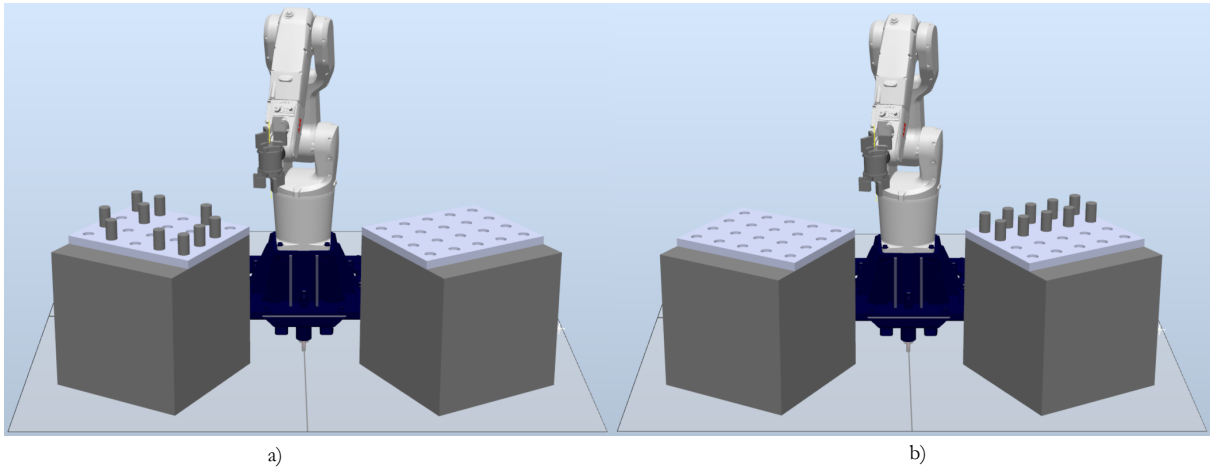
V nadaljevanju bo prikazan primer industrijske aplikacije, kjer bo uporabljen industrijski robot ABB IRB 1200, ki ima dve triprstni prijemali. Prijemali sta opremljeni z laserskim linijskim senzorjem za merjenje prisotnosti izdelkov. Izhod iz laserjev je tako samo 0 ali 1. Na desni in na levi strani industrijskega robota se nahajata zalogovnika z 20 utori za stožčaste izdelke, ki so med sabo zamaknjeni za 100 mm po obeh smereh. Valji so dimenzije premera 40 mm in višine 80 mm. Na vhodnem zalogovniku so valji poljubno nameščeni v utore. Naloga robota je, da avtomatsko poišče valje, jih pobere in na drugi strani zaporedno odlaga na izhodni zalogovnik. Pri tem je potrebno uporabiti linijski laser za zaznavanje valjev in ukaz »SearchL«. Ta ukaz je sicer specifičen za industrijske robote ABB. Pri drugih proizvajalcih bi uporabili navaden linearni pomik in s pomočjo prekinitev zaznali, v kateri poziciji je prišlo do zaznave. Ukaz »SearchL« že vsebuje prekinitveni signal, ki ga je seveda potrebno definirati. Ukaz je namenjen iskanju pozicije (x,y,z) pri izvajanju linearnega giba glede na trenutno uporabljen TCP. V trenutku, ko se aktivira opazovani senzor, si robotski krmilnik zapomni pozicijo. Pogoj, ki ga je tukaj potrebno upoštevati, je, da se prenos aktivacije signala s senzorja prenese na robota v čim krajšem času. Uporaba DSQC 652 (Digitalni I/O signali) kartice pri industrijskih robotih ABB to

omogoča, saj se aktivira v trenutku, ko je na vhodu zaznana sprememba stanja (0->1) [51]. Slika 12.5 prikazuje strukturo ukaza »SearchL«. Glavna omejitev pri iskanju je predvsem hitrost pomikanja robota. Optimalna hitrost je 100 mm/s ali manj, čeprav je hitrost odvisna od vrste aplikacije, zahtevane natančnosti in uporabljenega senzorja. Hitrost se lahko naknadno prilagodi glede na odzivnost robotske aplikacije v fazi testiranja. Pri uporabi ukaza »SearchL« lahko pri zaznavi senzorja robota zaustavimo v najkrajšem možnem času z ukazom »\Stop«. Pri tej opciji je potrebno paziti, saj robot zapusti programirano trajektorijo. Lahko izvedemo mehko zaustavitev z ukazom »\PStop«, ki poskrbi za zaustavitev robota, ne da bi zapustil programirano trajektorijo. Obstaja še en ukaz za zaustavitev robota, in sicer »\SStop«, ki je hitrejši od ukaza »\PStop« a počasnejši od ukaza »\Stop«. Z uporabo ukaza »\SStop« lahko robot zapusti programirano trajektorijo, a pri hitrostih, ki so večje od 100 mm/s. Ukaz »SearchL« tako potrebuje še digitalni signal, preko katerega linijski laserski senzor sporoči spremembo stanja. Tretji argument ukaza »SearchL« je spremenljivka, v katero se zapiše pozicija ob aktivaciji digitalnega vhoda. Naslednji argument je točka, do katere želimo preverjati digitalni vhod. Ostali argumenti ukaza »SearchL« so enaki, kot so pri navadnem linearnem ukazu pomika.



Slika 12.5: Struktura ukaza »SearchL«

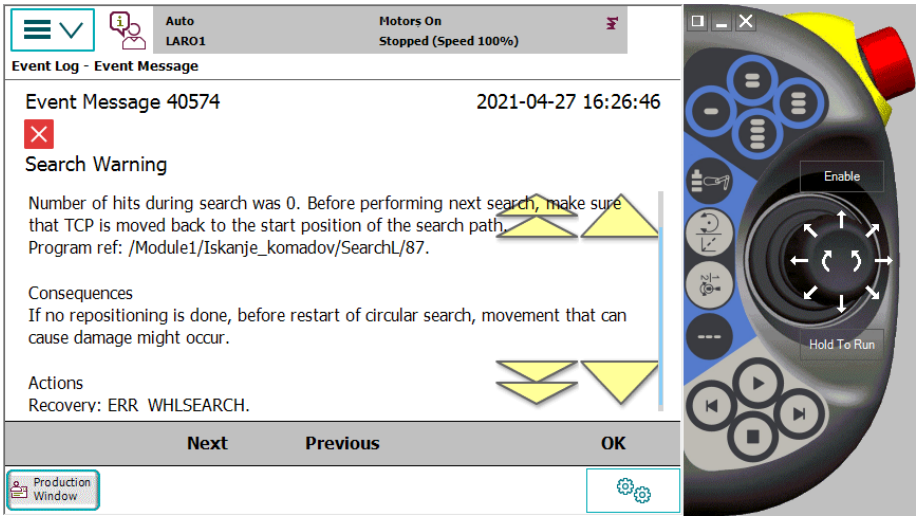
Slika 12.6 prikazuje začetno in končno stanje robotske aplikacije z uporabo ukaza »SearchL«. Pri tem mora iskanje izdelkov potekati avtomatsko.



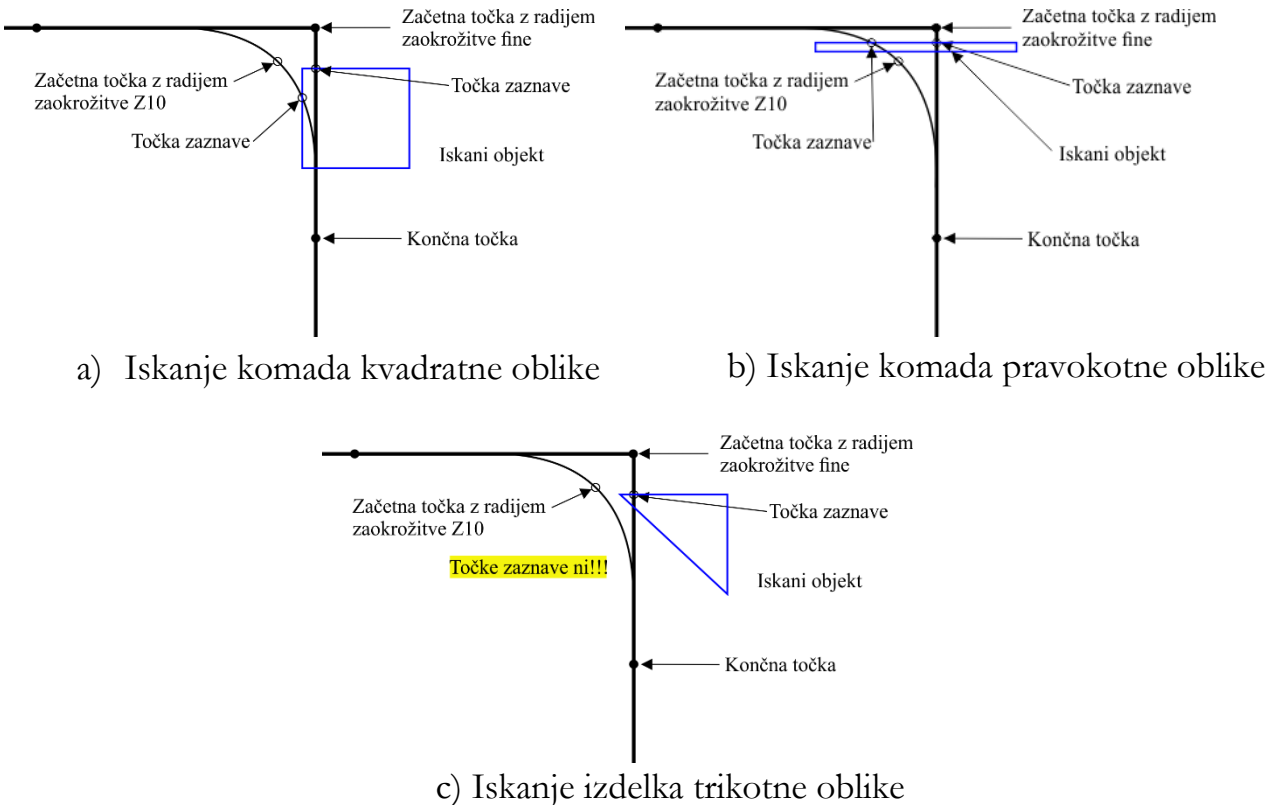
Slika 12.6: a) Začetna postavitev stožcev in b) Končna postavitev stožcev

Po navadi uporabljamo ukaz »SearchL« za iskanje izdelkov v isti liniji, npr. aplikacija pobiranje kartonov, še posebej če so kartoni različnih debelin, kar po navadi niso. V primeru različno debelih kartonov se mora robot približati kartonu na različni globini. V tem primeru si pomagamo z linijskim laserskim senzorjem in prej omenjenim senzorjem. Najprej se kartonom približujemo z namenom odčitavanja pozicije, potem pa se spustimo še za dodatnih nekaj centimetrov, da z vakuumskimi seski primemo karton. Pri tem imamo tako začetno in končno točko, znotraj katere z ukazom »SearchL« iščemo izdelek. V primeru prelaganja valjev pa moramo ta ukaz uporabiti za vsako potencialno mesto, kjer bi se lahko nahajal valj. Spreminjati moramo začetno in končno točko iskanja potencialnega mesta valja. Ker so vmes prazna mesta, se dogaja, da ukaz »SearchL« ne dobi nobenega signala od senzorja. Če ta ukaz znotraj predpisanega območja iskanja ne dobi signala, se sproži napaka, ki jo aktivira interni upravljalec napak in ustavi izvajanje programa. Takšen program, ki bo za vsako prazno mesto v vhodnem zalogovniku javil napako in zahteval posredovanje operaterja, nam kaj dosti ne koristi. Zato je potrebno napako zaznati in ustrezno odreagirati, da se bo program nemoteno izvajal dalje. Slika 12.7 prikazuje izpis napake na učni konzoli, kadar ukaz »SearchL« ne dobi nobenega signala od senzorja znotraj predpisanega iskalnega območja. Kot lahko vidimo, je v naslednjem opisu napake predlagana tudi rešitev. Uporaba upravljalca napake in napake »ERR_WHLSEARCH«. Znotraj te rutine je potrebno definirati, kako naj robot odreagira, ko se napaka pojavi.

Preden začnemo kreirati program in uporabljati upravljalca napak, naj opozorimo na uporabo ukaza »SearchL« in uporabo radijev zaokrožitve iskane točke oz. »ZoneData«, kot temu pravijo pri ABB. Z največjo gotovostjo bomo pri uporabi te funkcije najnatančnejši, če bomo uporabili radij zaokrožitve »fine«. Ta zagotovi, da se bo robot premaknil natanko v definirano točko. Vsakršna druga uporaba radija zaokrožitve, kot je npr. »Z10«, nam lahko bistveno zmanjša natančnost definiranja pozicije izdelka. Slika 12.8 prikazuje, kaj se lahko zgodi, če pri različnih geometrijskih oblikah uporabljamo zaokrožitev »fine« oz. zaokrožitev »Z10«.



Slika 12.7: Napaka, ki se pojavi, kadar ukaz »SearchL« znotraj predpisanega iskalnega območja ne dobi signala od senzorja.



Slika 12.8: Primer iskanja točke zaznave pri različnih geometrijskih oblikah objekta in različnih radijih zaokrožitve v iskani točki

Glavni program bo zasnovan tako, da bo vseboval dve FOR zanki. Znotraj prve bomo povečevali vrstice, znotraj druge pa stolpce. Za vsak stolpec posebej definiramo začetno in končno točko iskanja izdelka. Ti dve točki se morata prilagajati trenutni vrstici in stolpcu. Če je mesto prazno, se glede na to, kje se nahaja, ustrezno poveča stolpec in/ali

vrstica. Tako za iskanje potrebujemo eno samo referenčno točko, ki jo potem glede na to, v kateri vrstici in stolpcu se nahajamo, ustrezno zamikamo. V primeru zaznave valja moramo k točki zaznave prišteti vrednost polmera oz. vrednost polmera in še dodatek, ki je odvisen od hitrosti zaznave digitalnega signala iz senzorja. V virtualnem okolju RobotStudio to znaša 0.5 mm. To zamaknjeno točko uporabimo za postavitev robota nad valjem in za prijemanje valja. Ostali postopek je enak postopku, ki je opisan že v prejšnjih nalogah. Slika 12.9 prikazuje začetni del programa zaznave valjev s pomočjo ukaza »SearchL« in ustreznega zamika referenčne točke. Ko program pride do konca vrstice, se iskanje naj nadaljuje na isti strani, a v drugi vrstici.

```
PROC Iskanje_komadov()
    MoveAbsJ JointHome,v1000,z100,tool0\WObj:=wobj0;
    MoveJ Referencna_tocka_zaznave,v200,z30,Prijemalo_2\WObj:=KS_Vhodni_zalogovnik;        Referenčna točka
    FOR I FROM 1 TO 4 DO        Trenutna vrstica
        FOR J FROM 1 TO 5 DO        Trenutni stolpec
            II:=I;
            TockaZaznave:=PraznaTocka;        Resetiranje točke zaznave
            TockaZacetek:=Referencna_tocka_zaznave;
            TockaKonec:=Referencna_tocka_zaznave;
            !Program za prvo in tretjo vrstico
            IF I=1 OR I=3 THEN
                TockaZacetek:=Offs(TockaZacetek,(I-1)*100,(J-1)*100,0);        Ustrezno zamikanje začetne točke iskanja
                TockaKonec:=Offs(TockaKonec,(I-1)*100,J*100,0);        Ustrezno zamikanje končne točke iskanja
                IF KomadVPrijemalu2=False THEN
                    MoveJ TockaZacetek,v300,fine,Prijemalo_2\WObj:=KS_Vhodni_zalogovnik;
                    SearchL\Stop,InSenzor2,TockaZaznave,TockaKonec,v50,Prijemalo_2\WObj:=KS_Vhodni_zalogovnik;
                    IF TockaZaznave<>PraznaTocka THEN        Preverjanje ali so koordinate različne!
                        TockaZaznave:=Offs(TockaZaznave,0,20.5,0);        Prištevanje 20.5 mm po smeri Y k točki zaznave
                        TockaZaznavePrijemanje:=Offs(TockaZaznave,0,0,50);
                        MoveL TockaZaznave,v100,z0,Prijemalo_2\WObj:=KS_Vhodni_zalogovnik;
                        MoveL TockaZaznavePrijemanje,v100,fine,Prijemalo_2\WObj:=KS_Vhodni_zalogovnik;
                        PulseDO\High,\PLength:=2,Zapri_prijemalo2;
                        KomadVPrijemalu2:=TRUE;
                        WaitTime 1;
                        TockaZaznave:=Offs(TockaZaznave,0,0,-50);
                        MoveL TockaZaznave,v300,z0,Prijemalo_2\WObj:=KS_Vhodni_zalogovnik;
                        TockaZaznave:=PraznaTocka;        Resetiranje točke zaznave za uporabo drugega prijemala
                    ENDIF
                ENDIF
```

Slika 12.9: Začetni del programa iskanja valjev na vhodnem zalogovniku z drugim prijemalom

Uporaba opisanega programa bi na mestu, kjer ni valja, povzročila napako, ki je bila opisana v sliki 12.7. S pomočjo upravljalca napak se je potrebno ustrezno odzvati na napako. Ukaz »SearchL« uporabljamo tako pri iskanju izdelka s pomočjo senzorja 1 in senzorja 2. Napaka, ki jo prazno mesto povzroči, je ista pri obeh senzorjih. Tako je dovolj, če imamo samo en upravljalec napak. Prvi ukaz, ki ga potrebujemo, je ukaz »ERROR«. Ta se mora nahajati na koncu in znotraj našega programa iskanja izdelkov. Takoj zatem postavimo pogojno zanko, kjer preverjamo napako, ali je le-ta enaka »ERR_WHLSEARCH«. Ko se napaka pojavi, takoj shranimo trenutno generirano trajektorijo robota ter izvedemo pomik robota v naslednjo začetno točko iskanja izdelka.

Če se nahajamo v prvi ali tretji vrstici, iščemo izdelke iz leve proti desni, gledano v smeri robota. V tem primeru je začetna točka naslednjega iskanja kar končna točka prejšnjega iskanja. Ravno obratno je v primeru iskanja izdelkov iz desne proti levi. Po postavitvi robota v novo točko iskanja obnovimo prej shranjeno generirano trajektorijo, jo počistimo in ponovno štartamo program. Zadnji ukaz, ki po vsem tem sledi, je »TRYNEXT«. S tem ukazom program nadaljuje za ukazom, kjer se je pojavila napaka. Tako program nadaljuje v FOR zanki in poveča stolpec in/ali vrstico, odvisno je od tega, kje se nahaja. Slika 12.10 prikazuje program z upravljalcem napak ter celoten prej opisan postopek, zapisan v RAPID kodi. V prilogi 17.4 lahko najdete celoten program.

```
ERROR
    IF ERRNO=ERR_WHLSEARCH THEN
        StorePath;  !Shrani trenutno generirano trajektorijo robota
        IF II=1 or II=3 THEN    !Za vrstici 1 in 3
            Trenutno_orodje:=CTool();   !Shrani trenutno aktivno orodje v spremeljivko
            MoveL TockaKonec,v100,fine,Trenutno_orodje\WObj:=KS_Vhodni_zalogovnik;
        ENDIF
        IF II=2 OR II=4 THEN    !Za vrstici 2 in 4
            Trenutno_orodje:=CTool();   !Shrani trenutno aktivno orodje v spremeljivko
            MoveL TockaZacetek,v100,fine,Trenutno_orodje\WObj:=KS_Vhodni_zalogovnik;
        ENDIF
        RestoPath;  !Obnovi shranjeno generirano trajektorijo robota
        ClearPath;  !Počisti shranjeno generirano trajektorijo robota
        StartMove;  !Začni z nadaljnjim izvajanjem programa
        TRYNEXT;    !Nadaljuj za ukazom, kjer se je pojavila napaka
    ENDIF
```

Slika 12.10: Uporaba upravljalca napak v primeru uporabe ukaza »SearchL«

12.4 Vprašanja za poglobljen študij

1. Kaj je smiselnost uporabe upravljalca napak oz. »error handlerja«? Opišite praktični primer uporabe.

2. Napišite rutino, ki bo preverjala, ali je poljubno število celo število. Pri tem si za testiranje celega števila pomagajte z ukazom »TryInt«. Če izbrano število ni celo število, ta ukaz sproži napako »ERR_INT_NOTVAL«. S pomočjo upravljalca napak poskrbite, da poljubno število postane celo število. Pri tem si pomagajte z ukazom »Round« in ponovite ukaz, ki je sprožil napako.

3. Na spletni strani Estudij, kjer se nahajajo predmeti Industrijska robotika, Robotizacija ali Roboti in robotizacija, iz datoteke RobotStudio prenesite datoteko z imenom »IRB1200Iskanje_komadov_prazna_celica«. Odprite datoteko s pomočjo RobotStudia in sprogramirajte industrijskega robota tako, da bo iskal valje v vhodnem transporterju v smeri naprej/nazaj. Ker so vmes prazna mesta, je za nemoteno delovanje potrebno uporabiti upravljalca napak. Prav tako naj robot zlaga izdelke v smeri naprej/nazaj.

Ugotovitve:

ZAPISKI

Univerzitetna založba
Univerze v Mariboru

13 Uporaba funkcij nadzora gibanja in snemanja že izvedene trajektorije

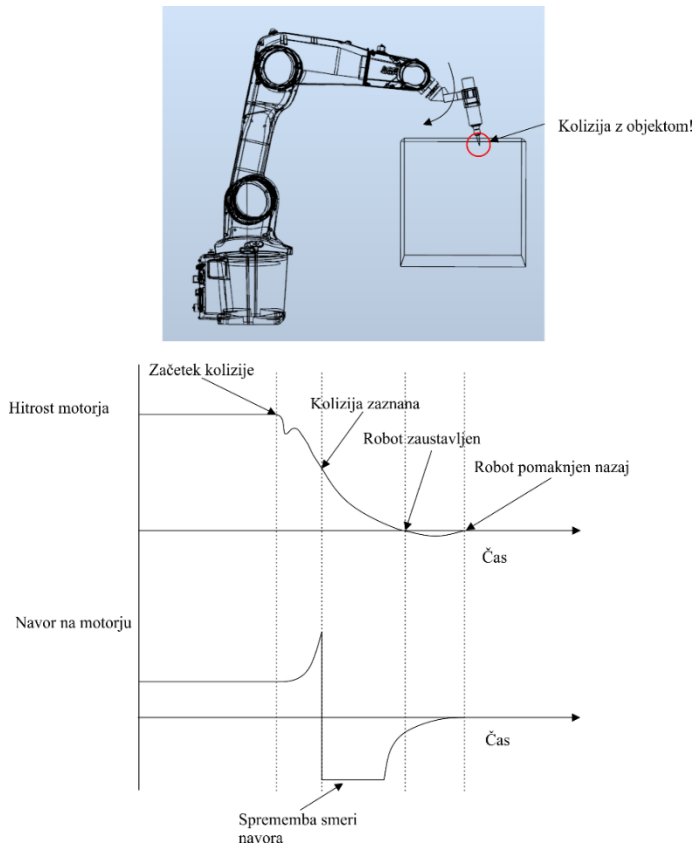
Trki industrijskega robota s periferijo oz. opremo so izredno nezaželeni, vendar se pogosto dogajajo, še posebej v fazi programiranja in testiranja robotske celice. Če se trk zgodi pri majhni hitrosti testiranja, ne bo ogromne materialne škode, mogoče samo kakšna praska, stisnjena škatla, razlit zaboj mleka, raztrošena mačja hrana in podobno. Pri večji hitrosti testiranja pa sta energija in vztrajnost robota bistveno večji in posledično tudi škoda v primeru trka. Kadar samo sumimo na to, da so trki s periferijo možni, uporabljamo dodatno funkcijo nadzora gibanja. Ta nam žal ne preprečuje trkov, zagotovo pa zmanjša nastalo škodo.

13.1 Funkcija nadzora gibanja »Motion Supervision/Collision Detection«

Kot je že bilo rečeno, različni proizvajalci ponujajo serijsko različne rešitve. Proizvajalec KUKA funkcijo nadzora ponuja serijsko. Proizvajalec ABB funkcijo ponuja opcijsko, zato tukaj še enkrat velja opozorilo. Preden se odločite za nakup določenega industrijskega robota, preučite tudi, kaj vse proizvajalec ponuja serijsko in kaj opcijsko. Vsaka dodatna opcija ima svojo vrednost. Proizvajalec robotov ABB to opcijo ponuja opcijsko in jo je potrebno pri naročilu robota posebej specificirati. Funkcija nadzora gibanja se tako imenuje »Motion Supervision«. Pri nakupu tako dobimo tudi opcijo »613-1 Collision Detection«. Funkcija skrbi za spremljanje nadzora navora v posameznih oseh. Pri trku navor v motorju oz. motorjih naraste. Če navor naraste nad dovoljeno mejo, ki je

nastavljiva, se gibanje robota ustavi in motorji izvedejo kratek povratni gib za sprostitev navora oz. zunanjih sil na konstrukcijo robota [50]. V večini primerov se trk robota zgodi s prijemalom. Slika 13.1 prikazuje princip delovanja nadzora gibanja oz. funkcije »Collision Detection«. Če uporabljamo funkcijo »Collision Detection«, je še toliko bolj pomembno, da pravilno nastavimo podatke o orodju in o teži, ki jo robot prenaša. V primeru nepravilno nastavljenih podatkov lahko zaradi navorov, ki se pojavljajo pri premagovanju vztrajnosti orodja, funkcija prekine gibanje robota, saj je prišlo do prekoračitve navorov. Funkcija uporablja spremenljivo vrednost zaznave povečanega navora. To pomeni, da je pri manjših hitrostih bolj občutljiva kot pri večjih. S tem robotskemu programerju pri navadnih aplikacijah ni potrebno posebej nastavljati občutljivosti. Privzeta občutljivost je 100 %, če želimo občutljivost povečati, to številko zmanjšamo na npr. 80 %. Če želimo manj občutljiv sistem, potem to število povečamo na npr. 150 % [50]. Sledi prikaz primera vklopa funkcije »Collision Detection« s spremembo občutljivosti, preden začnemo izvajati trajektorijo, kjer je možnost kolizije večja. Potem lahko to funkcijo izklopimo. V praksi funkcije ne izklapljamo, temveč raje zmanjšamo občutljivost (povečamo številko).

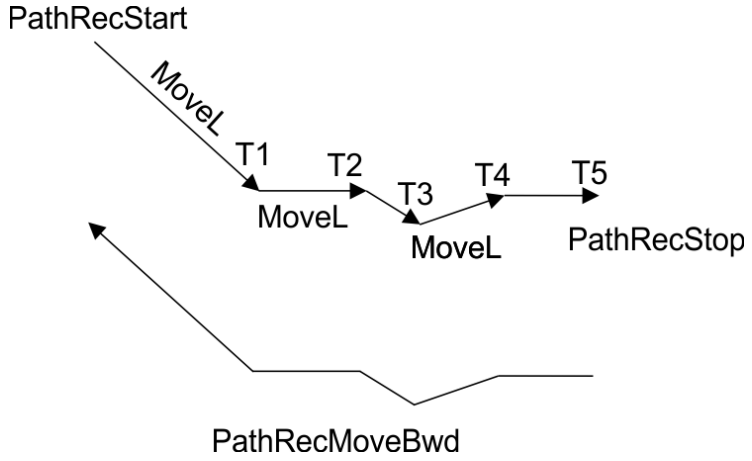
```
MotionSup \On \TuneValue:= 80;   !Sprememba obcutljivosti funkcije Motion Supervision na 80 %
!Trajektorija z možnimi kolizijami
MotionSup \Off;
!Trajektorija brez kolizij
```



Slika 13.1: Prikaz delovanja funkcije »Collision Detection« [50]

13.2 Funkcija snemanja že izvedene trajektorije »Path Recording«

Industrijski roboti delujejo v različnih industrijskih aplikacijah, kjer so ene manj zahtevne, druge pa imajo zelo kompleksne trajektorije. V vsakem primeru moramo zagotoviti varno vračanje robota v začetno, varno ali neko specifično točko. Da si olajšamo delo, imajo določeni proizvajalci na voljo snemanja oz. shranjevanja že izvedene trajektorije. Zopet velja, da je treba preveriti, kateri proizvajalci to opcijo ponujajo serijsko in pri katerih je to treba dokupiti. V primeru uporabe robota ABB je to opcijo potrebno dokupiti. Opcija se imenuje »611-1 Path Recovery«, omogoča pa, da pred začetkom izvajanja zahtevne trajektorije začnemo s snemanjem izvedene trajektorije. Trajektorija snemamo, dokler ne zaustavimo snemanja. Po tem se lahko odločimo, ali posneto trajektorijo shranimo ali zavržemo. Predstavljajte si robotsko varjenje notranjega dela tlačne posode. Pri tem je prehod robota v tlačno posodo sprogramiran izredno precizno. V primeru napake, kot je izpad zaščitnega plina, se izvajanje trajektorije ustavi in robot se mora vrniti v izhodiščno točko. V tem primeru bi bilo zelo kompleksno programirati povratno pot, saj lahko pride do izpada zaščitnega plina kadarkoli in bi morali imeti več možnih scenarijev. Opcija snemanja trajektorije nam omogoča, da uporabimo posneto trajektorijo za vrnitev robota v začetno pot snemanja. Slika 13.2 prikazuje princip delovanja snemanja izvedene trajektorije.

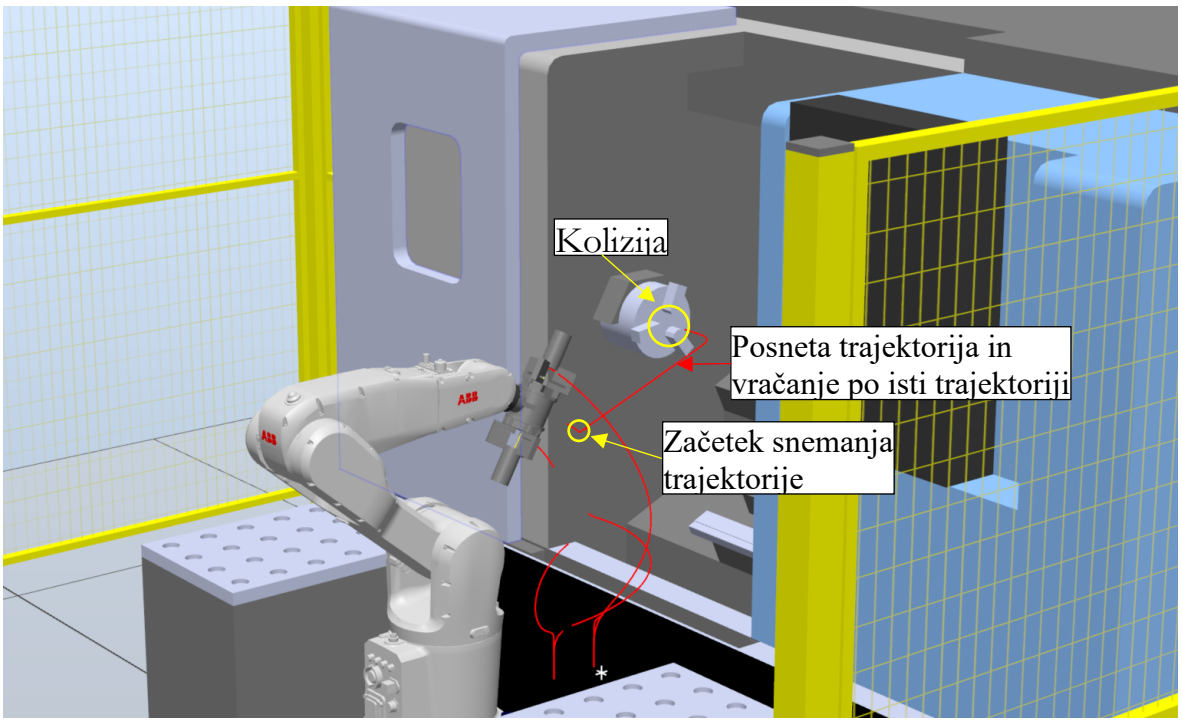


Slika 13.2: Princip delovanja snemanja izvedene trajektorije z opcijo »Path Recording«

13.3 Primer uporabe funkcij nadzora gibanj in snemanja že izvedene trajektorije

V nadaljevanju bo podan praktičen primer predstavitve obeh funkcij. Predstavljena bo robotska aplikacija z robotom IRB 1200, ki vstavlja izdelke v obdelovalni stroj CNC. Gre za enostavnejšo aplikacijo, na kateri bosta predstavljeni funkciji nadzora gibanj in snemanje že izvedene trajektorije. Teoretično se lahko zgodi, da bo prišlo do trka pri

vstavljanju surovca v triprstno prijemalo stroja CNC. Na začetku programa je potrebno zmanjšati občutljivost funkcije nadzora gibanj oz. »Collision Detection«. Z začetkom vstavljanja surovca v stroj CNC je potrebno začeti s snemanjem trajektorije vstavljanja, da bomo lahko v primeru trka robota varno vrnili v začetno točko snemanja. Pri tem bo seveda uporabljen tudi upravljalec napak oz. »error handler«, saj bo v trenutku kolizije sprožena napaka, ki jo je potrebno ustrezno obravnavati. Ker trka trenutno še ni možno simulirati v programskem okolju RobotStudio, bo to storjeno s pomočjo prekinitve in prekinitvene rutine. Slika 13.3 prikazuje praktičen primer delovanja snemanja trajektorije »Path Recording«. V nadaljevanju bo predstavljen še program.



Slika 13.3: Praktičen primer delovanja snemanja trajektorije »Path Recording«

Slika 13.4 prikazuje primer uporabe funkcij »Collision Detection«, kjer se občutljivost za kolizijo poveča na 80 % (povečanje občutljivosti) in »Path Recording«.

```
VAR pathrecid Vstavljanje1;
VAR pathrecid Vstavljanje2;
VAR pathrecid Pobiranje1;
VAR pathrecid Pobiranje2;
VAR intnum Prekinitev_kolizija;
PERS tooldata TrenutnoOrodje;
```

Imena posnetih trajektorij

```
PROC Trajektorija()
    MotionSup \On \TuneValue:= 80;   !Sprememba obcutljivosti funkcije Motion Supervision na 80 %
    MoveAbsJ Domov,v500,z50,tool0\WObj:=wobj0;
    ConfL\Off;
    MoveJ Prijemanje1_Gor,v1000,z50,Prijemalo_1\WObj:=KS_vhodni_zalogovnik;
                .
                .
    MoveJ Vmesna,v1000,z50,Prijemalo_2\WObj:=wobj0;
```

Začetek in konec snemanja trajektorije

```
    PathRecStart Vstavljanje1;   !Zacetek shranjevanja izvedene trajektorije
    MoveL CNC_vstavljanje_pred,v100,z50,Prijemalo_1\WObj:=KS_CNC;
    MoveL CNC_vstavljanje,v100,fine,Prijemalo_1\WObj:=KS_CNC;
    PathRecStop \Clear; !V kolikor se na tej trajektoriji kolizija ni zgodila, potem to pocistimo
    PulseDO \High \PLength:=2,DO_Prijemalo1_spusti;
    WaitTime 0.5;
    MoveL CNC_vstavljanje_pred,v100,z10,Prijemalo_1\WObj:=KS_CNC;
    MoveL Vmesna,v100,z10,Prijemalo_2\WObj:=wobj0;
```

Upravljalec napake

```
    ERROR (10)
IF ERRNO=10 THEN    !V primeru realnega robota in uporabe Collision Detection bi bilo tukaj
                    ! IF ERRNO=ERR_COLL_STOP THEN ... znotraj posameznega IF stavka pa bi morali definirati
    IF (PathRecValidBwd (\ID:=Vstavljanje1)) THEN       !nekaj podobnega, kot je definirano v prekinitveni
        PathRecMoveBwd \ID:=Vstavljanje1 \Speed:=v200;  !rutini
    ENDIF
                .
                .
    TrenutnoOrodje:=CTool();
    MoveAbsJ Domov,v500,fine,tool0\WObj:=wobj0;
    TPErase;
    TPWrite ("Zgodila se je kolizija!");
    TPWrite ("Robot se je vrnil v zacetno pozicijo.");
    EXIT;
ENDIF
```

Vračanje robota iz začetne točke snemanja v varno točko Domov

```
PROC Init_interrupt()
    CONNECT Prekinitev_kolizija WITH Kolizija;      !Simuliranje kolizije s pomočjo prekinitvene rutine
    ISignalDI DI_Prekinitev_kolizija,1,Prekinitev_kolizija;
    IEnable;
ENDPROC
TRAP Kolizija
    StopMove;   !Prekini z izvajanjem programa
    StorePath;  !Shrani trenutno pozicijo
    RAISE 10;   !Aktiviraj napako 10
    ERROR       !Ukaz, da napako prenese v visji nivo
    RAISE;      !Ukaz, da napako prenese v visji nivo
UNDO            !Preden kazalec zapusti prekinitveno rutino
    ClearPath;  !pocisti shranjeno pozicijo in
    StartMove;  !nadaljuje z izvajanjem programa
ENDTRAP
```

Simuliranje kolizije s prekinitvijo ter prenos napake v višji nivo

Slika 13.4: Prikaz delovanja funkcij »Collision Detection« in »Path Recording«

13.4 Vprašanja za poglobljen študij

1. Kaj je smiselnost uporabe funkcij »Collision Detection« in »Path Recording«? Opišite praktičen primer uporabe.

2. Na spletni strani eStudij, kjer se nahajajo predmeti Industrijska robotika, Robotizacija ali Roboti in robotizacija, iz datoteke RobotStudio – vaje prenesite datoteko z imenom »CollisionDetection_PathRecovery_prazna«. Odprite datoteko s pomočjo RobotStudia in sprogramirajte industrijskega robota tako, da bo v fazi odlaganja surovca v stružnico CNC snemal opravljeno trajektorijo. Prav tako naj posname opravljeno trajektorijo pri prijemanju izdelka iz stroja CNC. Prijemanje in odlaganje mora imeti ločeno ime posnete trajektorije. To storite za oba izdelka. Če se v času snemanja ne zgodi kolizija, posneto trajektorijo izbrišite. Kolizijo simulirate s pomočjo aktivacije digitalnega vhoda »DI_Prekinitev_kolizija«. Ta signal je potrebno povezati s prekinitvenim signalom in prekinitveno rutino, kjer je potrebno sprožiti napako 10. To napako je potrebno prenesti iz prekinitvene rutine v glavni program. Pri tem si pomagajte s sliko 13.4.

Ugotovitve:

ZAPISKI

Univerzitetna založba
Univerze v Mariboru

14 Vzdrževanje industrijskih robotov

Kot je bilo že v uvodu rečeno, je industrijski robot delovni stroj. Kot takega ga moramo obravnavati tudi glede vzdrževanja. Industrijski robot je sestavljen iz več mehanskih sklopov. Med njimi so tako elektromotorji, prestavna razmerja oz. reduktorji, jermenski prenosi in podobno. Posamezni deli tako zahtevajo svoj časovni interval vzdrževanja. S pravilnim in rednim vzdrževanjem lahko industrijskega robota maksimalno izkoristimo tudi že po tem, ko je preživel kar nekaj časa v industriji. Tukaj se sedaj zopet navezujemo na posameznega proizvajalca industrijskih robotov, saj ima vsak svoja navodila za vzdrževanje. Prav tako ima tudi vsak tip industrijskega robota svoja navodila vzdrževanja. V nadaljevanju bodo predstavljene osnove vzdrževanja mehanike na industrijskem robotu IRB 1200.

4.1 Intervali vzdrževanja industrijskih robotov

Vzdrževanje industrijskih robotov je razdeljeno na različne intervale za različne mehanske dele. Ti so zopet odvisni od proizvajalca in tipa robota.

Tabela 14.1: Časovni intervali aktivnosti vzdrževanja [9]

Aktivnost vzdrževanja	Redno	Vsakih 12 mesecev	Vsakih 36 mesecev	Opomba
Čiščenje robota	X			Odvisno od okolja
Preventivni pregledi				
Pregled robota	X			Iščejo se obrabe in kontaminacija, še posebej v čistih prostorih
Pregled robotskih kablov	X oz. po potrebi			Vizualni pregled kablov
Pregled informacijskih tablic		X		Pregledati je potrebno čistost in berljivost tablic.
Pregled mehanskih stopov prve, druge, tretje in četrte osi	X oz. po tem, ko se robot nasloni na mehanski stop.			Pregledati je potrebno, ali je mehanska omejitev še ustrezno pričvrščena.
Preverjanje jermenov			X	Preveriti je potrebno jermen na tretji in peti osi, če je pravilno napet.
Menjava baterije v robotu	Ni časovne omejitve po tem, ko se pojavi napaka »38213 Battery charge low«.			Naročiti je treba originalno baterijo in jo zamenjati po navodilih.

4.2 Čiščenje industrijskega robota

Industrijski roboti delajo v različnih industrijskih okoljih pri različnih pogojih. Nekateri delajo v čistih okoljih, kot je to farmacija, kjer so standardi za zagotavljanje čistosti izredno visoki in je zahteva po čiščenju robota visoka. Roboti, ki delajo v livarski industriji, so bistveno bolj izpostavljeni umazaniji, vendar kljub temu da umazanija zmanjšuje življenjsko dobo robotov, tam ni takšne potrebe po čistosti. Obstaja tudi možnost, da robote prevlečemo z zaščitno prevleko, ki jih ščiti pred umazanijo.

Frekvenca čiščenja industrijskih robotov je odvisna od okolja, v katerem obratuje, od števila ur, ki jih opravlja, in od zasedenosti robota. Pri postopku čiščenja je vedno potrebno poskrbeti, da je robot ugasnjen in da ga ni možno premikati. Robota ni priporočljivo čistiti s stisnjenim zrakom! Odvisno od standarda zaščite robota je tudi odvisno, s čim ga lahko čistimo.

Tabela 14.2: Načini čiščenja industrijskega robota [9]

IP zaščita	Metoda čiščenja			
	Industrijski sesalec	Brisanje s krpo	Škropljenje z vodo	Čiščenje z vodo pod visokim pritiskom in paro
Standard IP 40	Da	Da, z nežnim detergentom	Ne!	Ne!
IP 67 (opcijsko)	Da	Da, z nežnim detergentom	Da. Pri tem je priporočljivo, da voda vsebuje sredstvo proti rjavenju. Po tem je potrebno manipulator posušiti.	Ne!
Zaščita za livarska okolja (foundry plus)	Da	Da, z nežnim detergentom	Da. Pri tem je priporočljivo, da voda vsebuje sredstvo proti rjavenju. Po tem je potrebno manipulator posušiti.	Da. Pri tem je priporočljivo, da voda in para vsebujeta sredstvo proti rjavenju brez detergenta.
Roboti za čista okolja (clean room)	Da	Da, z nežnim detergentom, alkoholom ali izopropilnim alkoholom	Ne	Ne

14.2.1 Brisanje umazanij z industrijskega robota

Pri brisanju umazanij z industrijskega robota je potrebno paziti, da uporabljena tekočina detergenta ne zaide v členke robota ter da se tekočina ne zadržuje na površini oz. v raznih vboklinah.

14.2.2 Čiščenje z uporabo vode

Čiščenje industrijskega robota z vodo je mogoče samo pri robotih, ki imajo zaščito IP 67 oz. imajo dodatno zaščito za livarsko industrijo. Pri tem je potrebno upoštevati naslednja priporočila [9]:

- Maksimalni vodni pritisk curka znaša 7 barov.
- Pri tem naj bo uporabljena šoba, ki vodo razprši pod kotom 45 °.
- Minimalna priporočena razdalja čiščenja je 0.4 m.
- Maksimalni pretok vode je 20 l/min.

14.2.3 Čiščenje z uporabe vodne pare in vode pod pritiskom

Industrijski roboti, ki so namenjeni uporabi v livarski industriji, imajo posebno zaščito, ki se imenuje »Floundry Plus«. Le-ti so namenjeni čiščenju s pomočjo vodne pare in vode pod pritiskom. Pri tem je potrebno upoštevati naslednja priporočila [9]:

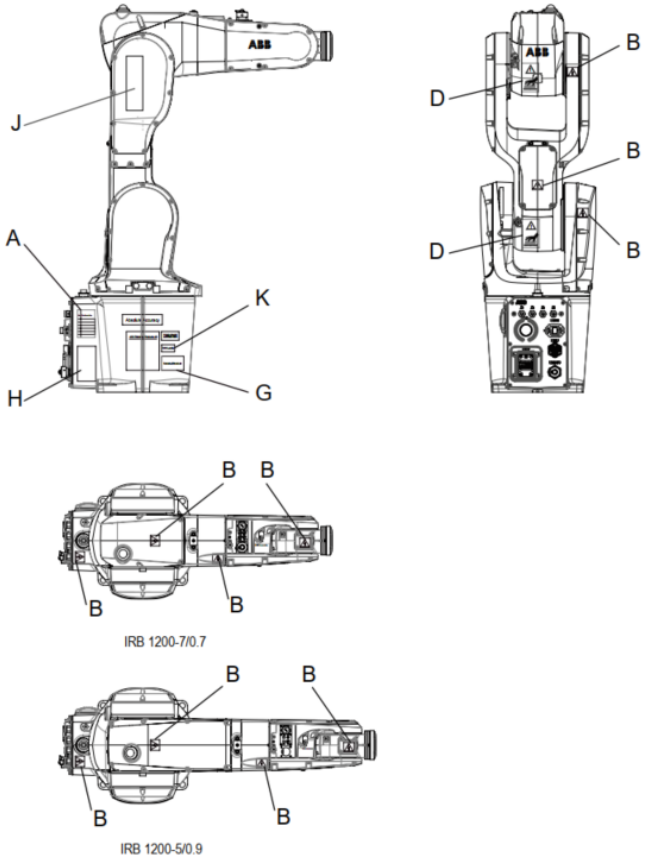
- Maksimalni vodni pritisk lahko znaša 25 barov.
- Pri tem naj bo uporabljena šoba, ki vodo razprši pod kotom 45 °.
- Minimalna priporočena razdalja čiščenja je 0.4 m.
- Maksimalna temperatura vode je 80 °C.

14.3 Pregled robotskih kablov

Pri pregledovanju robotskih kablov moramo biti pozorni na to, da na kablih ni vidnih poškodb, da je izolacija cela, da so konektorji čisti in čvrsto pričvrščeni. Prav tako moramo pregledati kable, ki potujejo do orodja. Ti kabli se lahko med sabo drgnejo, kar lahko privede do obrabe zaščite in v končni fazi do kratkega stika. To se lahko zgodi še posebej v prostorih z veliko prahu in umazanije, zato je takšna kontrola še toliko bolj smiselna. Pri tem je smiselno upoštevati umazanijo in prah že v fazi snovanja priključkov, kablov in cevi, ki bodo vodile do orodja robota.

14.4 Pregled informacijskih tablic, ki so nameščene na robotu

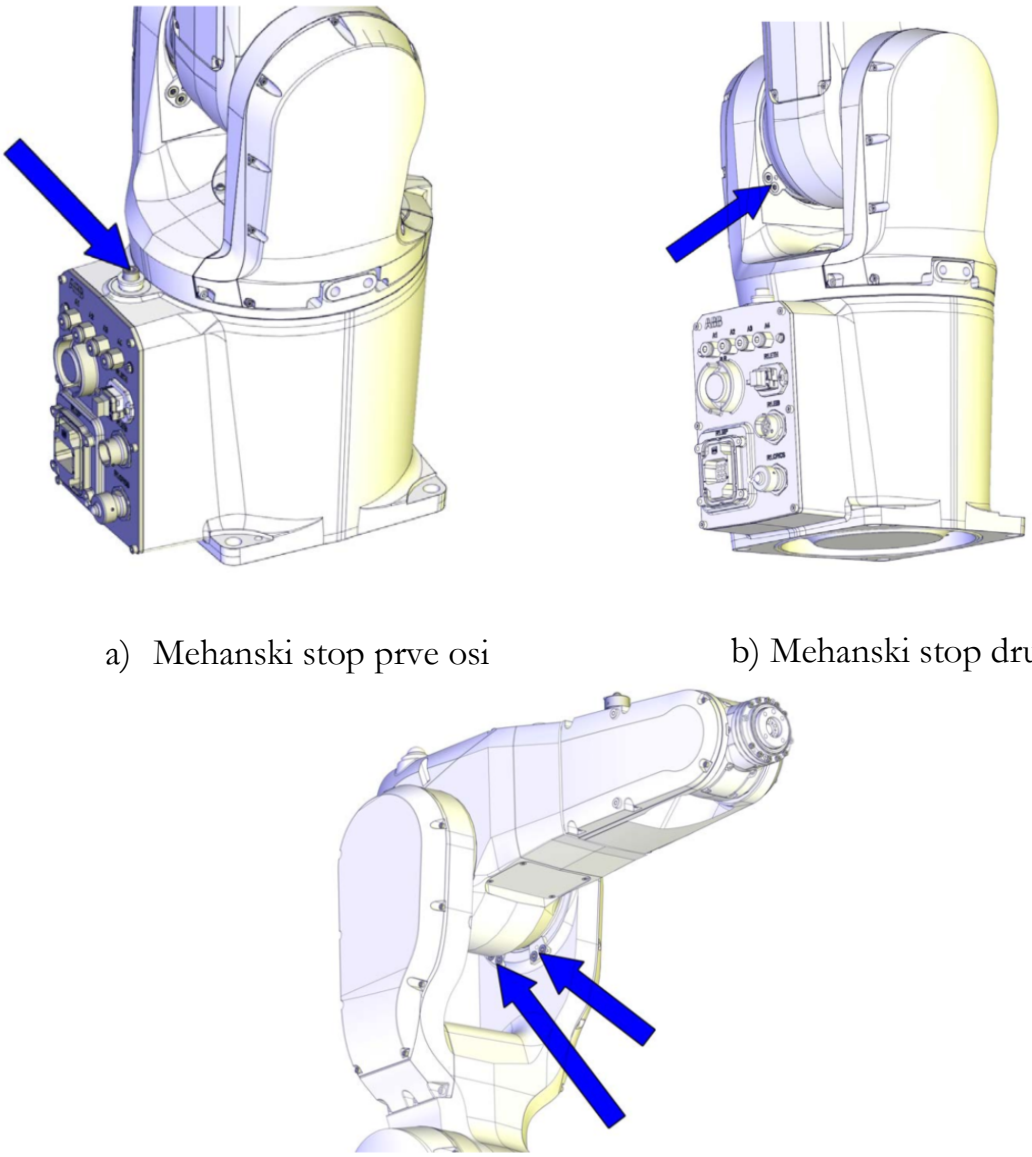
Vsak industrijski robot je opremljen z informacijskimi tablicami, ki so tako informacijske kot tudi opozorilne narave. Nekatere so splošnega značaja, nekatere pa opozarjajo na nevarnost. Kakorkoli, te tablice morajo biti nepoškodovane in ne smejo biti zamazane. Slika 14.1 prikazuje mesta kontrole informacijskih tablic.



Slika 14.1: Mesta kontrole informacijskih tablic na robotu IRB 1200

14.5 Kontrola mehanskih stopov

Vsak robot ima poleg programskih omejitev še mehanske stope, ki preprečujejo zasuk posameznih motorjev izven svojega območja. Ti mehanski stopi ne varujejo motorjev, saj se motorji lahko vrtijo neskončnokrat, temveč varujejo mehanske sklope robota, kot so členi, ter notranje kable, ki imajo končno dolžino, da se ne pretrgajo. Mehanske stope je potrebno preverjati samo v primeru, ko se robot nasloni, zaleti ali kakorkoli drugače dotakne stopa. Pri tem se preverja ustreznost stopa, njegova pričvrščenost in vizualna kontrola.



a) Mehanski stop prve osi

b) Mehanski stop druge osi

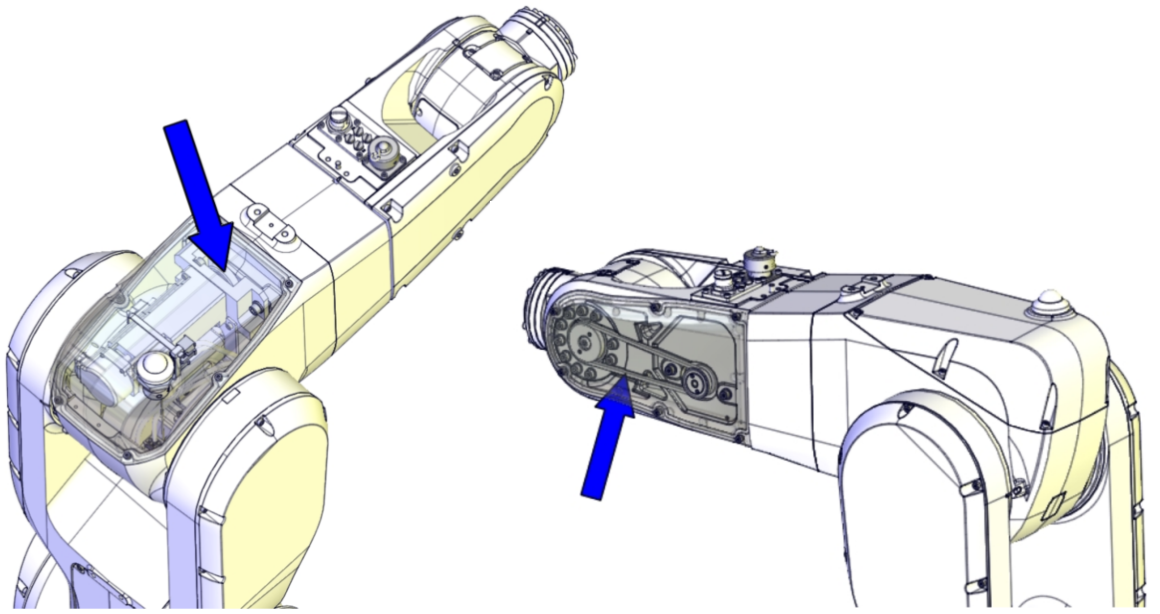
c) Mehanski stop tretje osi

Slika 14.2: Mehanski stopi prve, druge in tretje osi [9]

14.6 Kontrola jermenov na industrijskem robotu

V določenih zgibih industrijskega robota se za prenos gibanja med motorjem in sklepom uporablja jermen. Da je prenos natančen, je potrebno jermen prednapeti na določeno silo. Po navodilih proizvajalca robotov je pri robotu IRB 1200 potrebno preveriti prednapetje jermena vsake tri leta. Preverja se jermen v četrtem in v petem sklepu. Pri tem preverjamo jermen vizualno, da nima nikakršnih razpok, obrabe oz. mehanskih poškodb, ter preverimo ohlapnost jermena. Prav tako vizualno preverimo jermene. Če je jermen ohlapen na dotik, ga je potrebno prednapeti. Pri tem proizvajalec ne podaja konkretne sile,

potrebne za prednapetje jermena, temveč samo opozarja na to, da jermen ne sme biti ohlapen.



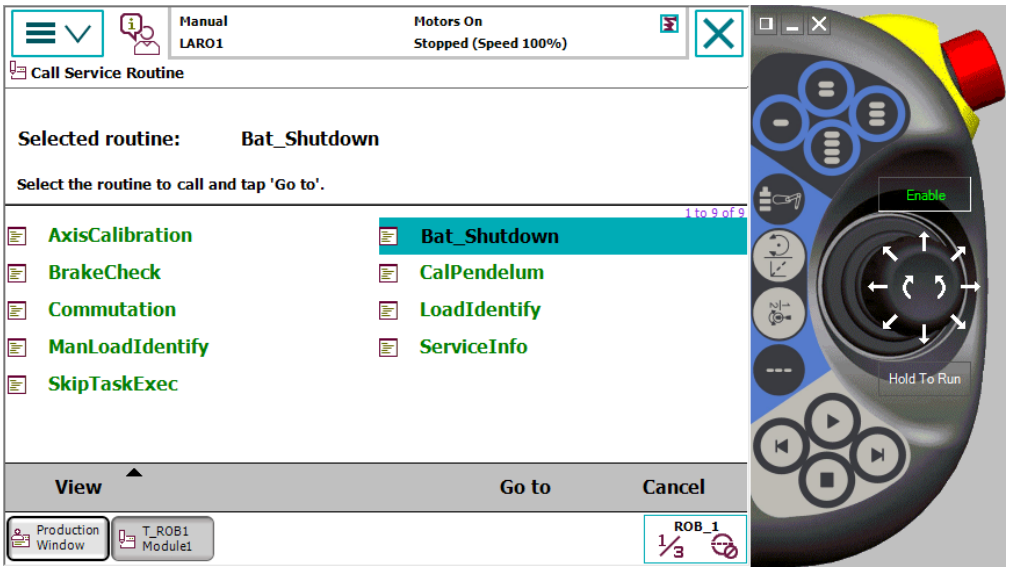
a) Jermen v četrti osi b) Jermen v peti osi

Slika 14.3: Pozicije jermenov v četrti in peti osi robota IRB 1200 [9]

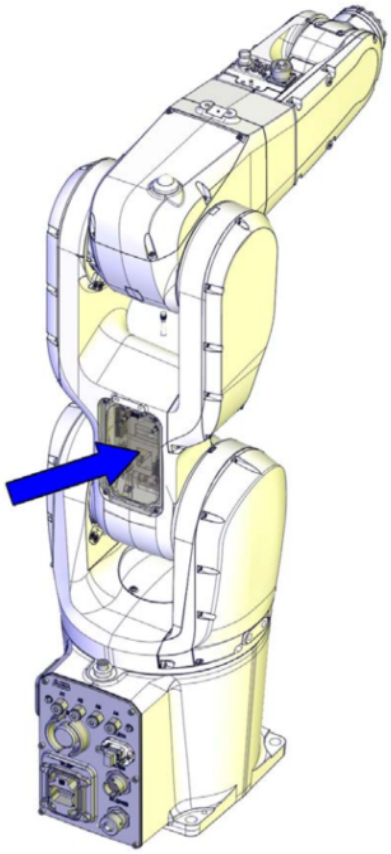
14.7 Menjava baterije na industrijskem robotu

Robot mora vedno vedeti, kje se nahaja njegova TCP-točka. Pri tem si pomaga s čitalci pozicije na posameznem servomotorju. Nekateri proizvajalci uporabljajo absolutne dajalnike pozicij. Njihova prednost je, da je pozicija enolično določena in robot vedno ve, kje se nahaja. Večino proizvajalcev pa uporablja inkrementalne dajalnike pozicije. Pri takšnem načinu robot potrebuje referenčno točko, od katere potem odšteva oz. prišteva signale inkrementalnega dajalnika. Roboti s takšnim načinom potrebujejo baterijo, ki vzdržuje trenutno število prebranih signalov. To število se ob zagonu krmilnika prebere z robota in krmilnik avtomatsko ve, na katerih stopinjah so posamezni motorji ter kje se nahaja TCP-točka. A takšen način pomeni, da se baterija slej ko prej izprazni. Kdaj se baterija izprazni, je odvisno predvsem od uporabe robota. Tipična življenjska doba nove baterije je 36 mesecev, pri čemer se upošteva, da je robot ugasnjen 2 dni na teden, ali 18 mesecev, če je robot ugasnjen 16 ur na dan. Življenjska doba baterije se lahko podaljša, če se občasno izvede rutina »Batt_Shutdown« [9]. Ta rutina se nahaja v istem direktoriju kot rutina »LoadIdentify«. Slika 14.4 prikazuje mesto, kjer se nahaja procedura »Batt_Shutdown«. Ko je življenjska doba baterije približno še na dveh mesecih, krmilnik začne javljati napako »Battery charge low«. Po tem je potrebno pri podjetju ABB naročiti

novo baterijo in jo zamenjati. Pri tem je potrebno slediti navodilom menjave baterije ter paziti na kakršen koli kratek stik, ki bi se lahko zgodil, da se ne uniči električnega vezja.



Slika 14.4: Klicanje procedure »Batt_Shutdown«



Slika 14.5: Mesto na robotu, kjer se nahaja baterija [9].

14.8 Vprašanja za utrjevanje snovi

1. Od katerih pogojev je odvisno pregledovanje in čiščenje industrijskih robotov? Na kaj moramo biti pri tem pozorni?

2. Ali je čiščenje industrijskih robotov dovoljeno s pomočjo vodnega curka? V katerih primerih da in v katerih primerih ne? Kakšnim zahtevam mora robot zadostovati, da ga lahko čistimo s pomočjo vodne pare?

3. Zakaj industrijski robot potrebuje baterijo?

ZAPISKI

Uporabljeni viri

[1] "ISO 8373:2012 Robots and robotic devices — Vocabulary." https://www.iso.org/standard/55890.html Dostopno dne: 17. 03. 2020.
[2] "ISO 10218-1:2011 Robots and robotic devices — Safety requirements for industrial robots — Part 1: Robots." https://www.iso.org/standard/51330.html Dostopno dne: 17. 03. 2020.
[3] "ISO 10218-2:2011 Robots and robotic devices — Safety requirements for industrial robots — Part 2: Robot systems and integration." https://www.iso.org/standard/41571.html Dostopno dne: 17. 03 2020.
[4] "ISO 11161:2007 Safety of machinery — Integrated manufacturing systems — Basic requirements." https://www.iso.org/standard/35996.html Dostopno dne: 21.07.2021.
[5] "ISO 13849-1:2016 Safety of machinery — Safety-related parts of control systems — Part 1: General principles for design." https://www.iso.org/standard/69883.html Dostopno dne: 21.07.2021.
[6] "ISO 12100:2010 Safety of machinery — General principles for design — Risk assessment and risk reduction." https://www.iso.org/standard/51528.html Dostopno dne: 22. 07. 2020
[7] "Direktiva o strojih 2006/42/ES " https://eur-lex.europa.eu/legal-content/SL/TXT/?uri=celex%3A32006L0042 Dostopno dne: 21.07.2021.
[8] B. Matthias. "Industrial Safety Requirements for Collaborative Robots and Applications." https://www.academia.edu/36358543/Industrial_Safety_Requirements_for_Collaborative_Robots_and_Applications Dostopno dne: 21.07.2021.
[9] *IRB 1200 Product Specification*, 2017.
[10] "ABB RobotStudio " https://new.abb.com/products/robotics/robotstudio Dostopno dne: 23. 07. 2020.
[11] "ABB RobotLoad " https://abb-robotload.software.informer.com/download/ Dostopno dne: 23. 07. 2020.
[12] "KUKA Load." https://www.kuka.com/en-de/services/downloads?terms=Language:en:1;&q=kuka%20load Dostopno dne: 23. 07. 2020.
[13] "ISO 9283:1998(en) Manipulating industrial robots — Performance criteria and related test methods." https://www.iso.org/obp/ui/#iso:std:iso:9283:ed-2:v1:en Dostopno dne: 23. 07. 2020.
[14] SIST. "SIST EN 60529:1997." http://ecommerce.sist.si/catalog/project.aspx?id=8a9f21d2-ef8b-4a66-9cbd-1b398a95c247 Dostopno dne: 1. 10. 2020.
[15] MPL. "IP Ratings (Ingress Protection)." https://www.mpl.ch/info/IPratings.html Dostopno dne: 1. 10. 2020.
[16] "Right hand rule, Wikipedia." https://upload.wikimedia.org/wikipedia/commons/d/d2/Right_hand_rule_cross_product.svg Dostopno dne: 23. 07. 2020.

[17] "Right hand screw rule, Wikipedia." https://commons.wikimedia.org/wiki/File:Right-hand_screw_rule.svg Dostopno dne: 23. 07. 2020.
[18] "ISO 9787:2013 Robots and robotic devices — Coordinate systems and motion nomenclatures." https://www.iso.org/standard/59444.html Dostopno dne: 26.07.2021.
[19] "DeviceNet protocol." https://www.rtautomation.com/technologies/devicenet/ Dostopno dne: 24. 07. 2020.
[20] "RS232 Protocol." https://www.electronicshub.org/rs232-protocol-basics/ Dostopno dne: 24. 07. 2020.
[21] "ABB MultiMove Coordinated." https://new.abb.com/products/3HAC054355-001/multimove-coordinated Dostopno dne: 24. 07. 2020.
[22] "PROFINET protokol." https://www.profibus.com/technology/profinet/ Dostopno dne: 24. 07. 2020.
[23] ABB. "IRC5 Industrial Robot Controller." https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiO9bavvO3qAhXK0qYKHaJKCKEQFjAAegQIAxAB&url=https%3A%2F%2Flibrary.e.abb.com%2Fpublic%2Fd6f68ade0cb24d6aa0fe79220321d187%2FIRC5_ROB0295EN-Rev.D.pdf&usg=AOvVaw2t92nWMJIe9kPCL9AsVzBa Dostopno dne: 27. 07. 2020.
[24] "KUKA KRC4 Controllers." https://www.kuka.com/en-de/products/robot-systems/robot-controllers/kr%C2%A0c4 Dostopno dne: 24. 07. 2020.
[25] *Electric circuit 3HAC049406-003 IRC5 Compact*, 2014.
[26] ABB. "RobotStudio " https://new.abb.com/products/robotics/robotstudio Dostopno dne: 28. 07. 2020.
[27] L. Kumer, "SIMULACIJA PROIZVODNEGA SISTEMA ROBOTSKE ROKE V POVEZAVI S STROJEM DOOSAN CNC V PROGRAMSKEM ORODJU ROBOTSTUDIO ABB," L. Kumer, 2016. [Online]. Dostopno dne:: https://dk.um.si/Dokument.php?lang=slv&id=98666&dn=
[28] L. Kumer, "Avtomatizacija proizvodnega procesa med DOOSAN CNC in ABB robotom IRB 1200 : magistrsko delo," [L. Kumer], Maribor, 2019. [Online]. Dostopno dne:: https://dk.um.si/Dokument.php?lang=slv&id=137777&dn=
[29] Siemens. "PLM Automation, robot programming." https://www.plm.automation.siemens.com/global/en/products/tecnomatix/robotics-automation-programming.html Dostopno dne: 28. 07. 2020.
[30] RoboDK. "Simulating program." https://robodk.com/index Dostopno dne: 28. 07. 2020.
[31] PILZ. "Safety Relays " https://www.pilz.com/en-INT/products/relay-modules/safety-relays-protection-relays Dostopno dne: 28. 07. 2020.
[32] P. Design. "Safety PLC." http://plcdesign.xyz/en/whats-safety-plc/ Dostopno dne: 28. 07. 2020.
[33] PILZ. "Pilz PNOZ X2.1." https://media.rs-online.com/t_large/F2391055-01.jpg Dostopno dne: 28. 07. 2020.
[34] S. Electric. "Emergeny stop button." https://media.screwfix.com/is/image//ae235?src=ae235/615HV_P&$prodImageMedium$ Dostopno dne: 28. 07. 2020.
[35] L. Electric. "Push button." https://www.lovatoelectric.com/HandlerPhoto.ashx?s=LPCB103.jpg&d=ProductsMiddle Dostopno dne: 28. 07. 2020.
[36] PILZ. "PILZ Varnostni rele primer ožičenja." https://www.youtube.com/watch?v=SYkYQCfPVPI Dostopno dne: 11. 05. 2021.
[37] "ISO 9409-1:2004 Manipulating industrial robots — Mechanical interfaces — Part 1: Plates." https://www.iso.org/standard/36578.html Dostopno dne: 26.07.2021.
[38] SolidWorks. https://www.solidworks.com/ Dostopno dne: 29. 07. 2020.
[39] Wikipedia. "Quaternions to Euler." https://en.wikipedia.org/wiki/Conversion_between_quaternions_and_Euler_angles Dostopno dne: 30. 07. 2020.
[40] Quaternions. "Visualization and conversion." https://quaternions.online/ Dostopno dne: 30. 07. 2020.
[41] Wikipedia. "Profinet." https://en.wikipedia.org/wiki/PROFINET Dostopno dne: 03. 08. 2020.
[42] Wikipedia. "Industrial Ethernet." https://en.wikipedia.org/wiki/Industrial_Ethernet Dostopno dne: 03. 08. 2020.
[43] *Application manual: PROFINET Controller/Device with IO Configurator*, 2018.

[44] P. University. "Profinet GSD file basics." https://profinetuniversity.com/profinet-basics/profinet-gsd-file-basics/ Dostopno dne: 06. 08. 2020.
[45] "Profibus Profinet International." https://www.profibus.com/pi-organization/ Dostopno dne: 06. 08. 2020.
[46] Siemens. "ET200SP I/O Distributor." https://support.industry.siemens.com/cs/document/58649293/simatic-et-200sp-distributed-i-o-system?dti=0&lc=en-US Dostopno dne: 07. 08. 2020.
[47] "Two's complement." https://en.wikipedia.org/wiki/Two%27s_complement Dostopno dne: 28.09.2021.
[48] "Signed number representations." https://en.wikipedia.org/wiki/Signed_number_representations Dostopno dne: 28.09.2021.
[49] "Reed stikalo." https://www.explainthatstuff.com/howreedswitcheswork.html Dostopno dne: 01.04.2021.
[50] *RAPID Overview*, 2018.
[51] *RAPID Instructions, Functions and Data types*, 2018.

Univerzitetna založba
Univerze v Mariboru

Priloge

Priloga 1: Program pošiljanja pozicije prvega robota preko PROFINET komunikacije
Priloga 2: Program prejemanja pozicije drugega robota preko PROFINET komunikacije
Priloga 3: Program paletizacije
Priloga 4: Program iskanja izdelkov, uporaba »SearchL« in upravljalca napak

Priloga 1:

Program pošiljanja pozicije prvega robota preko PROFINET komunikacije

```
MODULE Module1
    CONST robtarget
Home:=[[704.990919137,0,705.74730631],[0.190808996,0,0.981627183,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_10:=[[51.644297026,-
48.844067712,0],[0,0.016213219,0.999868557,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_20:=[[54.411442727,-35.859297965,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_30:=[[62.627378003,-27.05694332,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_40:=[[73.439866244,-23.032880485,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_50:=[[101.819546768,-24.746914058,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_60:=[[128.84288659,-26.699393253,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_70:=[[152.680729576,-33.771401765,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_80:=[[166.016193983,-34.02298943,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_90:=[[190.352575869,-31.056797287,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_100:=[[205.786996804,-30.624713772,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_110:=[[216.350686816,-31.425885408,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_120:=[[240.819660685,-36.185615145,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_130:=[[252.664979425,-40.614584146,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_140:=[[263.350300491,-49.424300181,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_150:=[[270.275269718,-62.038683621,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_160:=[[270.565813603,-73.666614805,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_170:=[[265.156696225,-88.980990126,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_180:=[[267.497216675,-110.586632669,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_190:=[[267.847255198,-126.157999788,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_200:=[[266.992904172,-154.590018468,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_210:=[[267.999976068,-172.840755759,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_220:=[[269.656068406,-192.398723345,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_230:=[[269.856117539,-202.520994341,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_240:=[[268.778998735,-222.562871417,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_250:=[[267.49846648,-232.948614539,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_260:=[[263.123765606,-247.083209931,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_270:=[[254.242309576,-259.534416461,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_280:=[[239.993750357,-270.345075,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_290:=[[229.328710552,-274.894177077,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_300:=[[218.531811202,-277.145471202,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_310:=[[208.005067895,-277.535403535,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_320:=[[197.843099653,-276.675359551,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_330:=[[169.050618916,-271.835618928,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_340:=[[150.726422315,-270.840461013,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_350:=[[131.761463759,-274.574957459,0],[0,0.016213219,0.999868557,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
        CONST robtarget Target_360:=[[100.528178443,-285.612485608,0],[0,0.016213219,0.999868557,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Target_370:=[[90.141515082,-288.350576806,0],[0,0.016213219,0.999868557,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Target_380:=[[79.590579304,-289.306853113,0],[0,0.016213219,0.999868557,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Target_390:=[[68.740117834,-286.873584825,0],[0,0.016213219,0.999868557,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Target_400:=[[59.406938375,-279.39501298,0],[0,0.016213219,0.999868557,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Target_410:=[[53.884273575,-267.998451306,0],[0,0.016213219,0.999868557,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Target_420:=[[51.946455609,-257.454591663,0],[0,0.016213219,0.999868557,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Target_430:=[[51.644297026,-48.844067712,0],[0,0.016213219,0.999868557,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    VAR pos Pozicija_robota;
    VAR num Poz_x:=0;
    VAR num Poz_y:=0;
    VAR num Poz_Z:=0;
    VAR num vrstica:=0;

    PROC main()
        Path_10;
    ENDPROC
        PROC Path_10()
            MoveJ Home,v1000,z100,MyTool\WObj:=wobj0;
            MoveJ Target_10,v1000,fine,MyTool\WObj:=KS_plosca;
        SetDO Start,1;
        Poslji_pozcijio;
            MoveL Target_20,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_30,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_40,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_50,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_60,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_70,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_80,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_90,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_100,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_110,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_120,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_130,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_140,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_150,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_160,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_170,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_180,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_190,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_200,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_210,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_220,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_230,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_240,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_250,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
            MoveL Target_260,v100,fine,MyTool\WObj:=KS_plosca;
        Poslji_pozcijio;
```

```
            MoveL Target_270,v100,fine,MyTool\WObj:=KS_plosca;
    Poslji_pozcijio;
            MoveL Target_280,v100,fine,MyTool\WObj:=KS_plosca;
    Poslji_pozcijio;
            MoveL Target_290,v100,fine,MyTool\WObj:=KS_plosca;
    Poslji_pozcijio;
            MoveL Target_300,v100,fine,MyTool\WObj:=KS_plosca;
    Poslji_pozcijio;
            MoveL Target_310,v100,fine,MyTool\WObj:=KS_plosca;
    Poslji_pozcijio;
            MoveL Target_320,v100,fine,MyTool\WObj:=KS_plosca;
    Poslji_pozcijio;
            MoveL Target_330,v100,fine,MyTool\WObj:=KS_plosca;
    Poslji_pozcijio;
            MoveL Target_340,v100,fine,MyTool\WObj:=KS_plosca;
    Poslji_pozcijio;
            MoveL Target_350,v100,fine,MyTool\WObj:=KS_plosca;
    Poslji_pozcijio;
            MoveL Target_360,v100,fine,MyTool\WObj:=KS_plosca;
    Poslji_pozcijio;
            MoveL Target_370,v100,fine,MyTool\WObj:=KS_plosca;
    Poslji_pozcijio;
            MoveL Target_380,v100,fine,MyTool\WObj:=KS_plosca;
    Poslji_pozcijio;
            MoveL Target_390,v100,fine,MyTool\WObj:=KS_plosca;
    Poslji_pozcijio;
            MoveL Target_400,v100,fine,MyTool\WObj:=KS_plosca;
    Poslji_pozcijio;
            MoveL Target_410,v100,fine,MyTool\WObj:=KS_plosca;
    Poslji_pozcijio;
            MoveL Target_420,v100,fine,MyTool\WObj:=KS_plosca;
    Poslji_pozcijio;
            MoveL Target_430,v100,fine,MyTool\WObj:=KS_plosca;
    Poslji_pozcijio;
            MoveJ Home,v1000,z100,MyTool\WObj:=wobj0;
    WaitRob \InPos;
    SetGO PosljiXpos, 0;
    SetGO PosljiYpos, 0;
    SetGO PosljiZpos, 0;
    SetGO GOVrstica, 0;
    SetDO Start,0;
        ENDPROC
  PROC Poslji_pozcijio()
    vrstica:=vrstica+1;                         ! Po vsakem posiljanju povecamo vrednost vrstice
    Pozicija_robota:=CPos(\Tool:=MyTool \WObj:=KS_plosca);    ! Preberemo trenutno pozicijo z izbranim orodjem in KS
    Poz_x:=Trunc(Pozicija_robota.x*10);              ! Prebrano pozicijo X pomnozimo z 10 in odrezemo decimalni del
    IF Poz_x<0 THEN                             ! Ce je pozicija negativna, potem pristejemo 16384
      Poz_x:=Poz_x+16384;
    ENDIF
    Poz_y:=Trunc(Pozicija_robota.y*10);              ! Ponovimo zgoraj opisani postopek za Y
    IF Poz_y<0 THEN
      Poz_y:=Poz_y+16384;
    ENDIF
    Poz_z:=Trunc(Pozicija_robota.z*10);              ! Ponovimo zgoraj opisani postopek za Y
    IF Poz_z<0 THEN
      Poz_z:=Poz_z+16384;
    ENDIF
    SetGO PosljiXpos, Poz_x;                    ! Izracunano vrednost posjlemo preko grupiranega signala na PROFINET
    SetGO PosljiYpos, Poz_y;                    ! Izracunano vrednost posjlemo preko grupiranega signala na PROFINET
    SetGO PosljiZpos, Poz_z;                    ! Izracunano vrednost posjlemo preko grupiranega signala na PROFINET
    SetGO GOVrstica, vrstica;                   ! Posjlemo st. vrstice preko grupiranega signala na PROFINET
    WHILE (DITrapend=0) DO                      ! Dokler ne dobimo signala, da je drugi robot prebral pozicijo, robot ne nadaljuje
    PulseDO \high \Plength:=0.2,DOTrap;
    ENDWHILE
  ENDPROC
ENDMODULE
```

Priloga 2:

Program prejemanja pozicije drugega robota preko PROFINET komunikacije

```
MODULE Module1
    CONST robtarget Home:=[[704.990919137,0,705.74730631],[0.190808996,0,0.981627183,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    VAR robtarget Target_robota;
    VAR num Poz_x:=0;
    VAR num Poz_y:=0;
    VAR num Poz_Z:=0;
    VAR num vrstica:=0;
    CONST robtarget Target_10:=[[0,0,0],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    PROC main()
        vrstica:=0;
        Poz_x:=0;
        Poz_y:=0;
        Poz_Z:=0;
        Path_10;
    ENDPROC
    PROC Path_10()
        MoveJ Home,v1000,z100,MyTool\WObj:=wobj0;              ! Postavitev robota v varno točko
        WaitDI Start,1;                                       ! Čakanje prvega ukaza za branje
        WaitTime 0.5;                                         ! Zakasnitev branja
        Preberi_pozicijo;                                     ! Klic funkcije za branje prejete pozicije
        ConfL\Off;                                            ! Ne preverja konfiguracije linearnih gibov
        MoveJ Target_robota,V1000,z0,MyTool\WObj:=KS_plosca;    ! Izvedba prvega giba na podlagi prejetih podatkov
        WHILE vrstica<43 DO                                   ! Ponavljanje branja in izvajanja giba robota, dokler prvi robot
        Preberi_pozicijo;                                     ! ne pošlje vseh podatkov
        ConfL\Off;
        MoveL Target_robota,V100,z0,MyTool\WObj:=KS_plosca;
        WaitRob \InPos;                                       ! Robot mora doseči pozicijo, prej se program ne sme izvajati dalje
        ENDWHILE
        MoveJ Home,v1000,z100,MyTool\WObj:=wobj0;              ! Vrnitev robota nazaj domov
        vrstica:=0;
    ENDPROC
    PROC Preberi_pozicijo()
        Poz_x:=GInput(X_pozicija);                            ! Branje pozicije po X smeri
        IF Poz_x>8191 THEN                                    ! Če je prejeta pozicija večja od 8192, potem od tega odštejemo 16384
            Poz_x:=Poz_x-16384;
        ENDIF
        Poz_x:=Poz_x/10;                                      ! Da dobimo decimalko, prejeto vrednost delimo z 10
        Poz_y:=GInput(Y_pozicija);                            ! Ponovimo postopek za Y os
        IF Poz_y>8191 THEN
            Poz_y:=Poz_y-16384;
        ENDIF
        Poz_y:=Poz_y/10;
        Poz_z:=GInput(Z_pozicija);                            ! Ponovimo postopek za Y os
        IF Poz_z>8191 THEN
            Poz_z:=Poz_z-16384;
        ENDIF
        Poz_z:=Poz_z/10;
        Target_robota:=Offs(Target_10,Poz_x,Poz_y,Poz_Z);      ! V referenčno točko Target_10 zapišemo zamike po posameznih oseh
        vrstica:=GInput(GIVrstica);                           ! Preberemo številko vrstice
        PulseDO \High \PLength:=0.1, DOTrapend;               ! Pošljemo kratek signal prvemu robotu, da smo prebrali pozicijo
        RETURN;
    ENDPROC
ENDMODULE
```

Priloga 3:
Program paletizacije

```
MODULE Module1
    CONST robtarget Pred_prijemanje:=[[-30,0,-220],[1,0,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Prijemanje_komada:=[[0,0,0],[1,0,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Prijemanje_gor:=[[0,0,-120],[1,0,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Pred_odlaganje:=[[0,0,-220],[1,0,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Odlaganje:=[[0,0,0],[1,0,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    VAR robtarget PredOdlaganje_paleta;
    VAR robtarget Odlaganje_paleta;
    VAR num sirina_off;
    VAR num vrstica;
    VAR num sirina_paketa:=120;
    VAR num stolpec;
    VAR num dolzina_paketa3:=612.5;
    VAR num stolpec_off;
    VAR num sloj;
    VAR num visina_off;
    VAR num visina_paketa:=200;
    VAR robtarget Vmesna_off;
    VAR robtarget Vmesna_nazaj_off;
    VAR robtarget Vracanje_domov_off;
    VAR intnum test_prekinitev;
    VAR bool test1;
    !Spremenljivke za delovna obmocja
    VAR wztemporary zona_vhodni_transporter;
    VAR wztemporary zona_izhodni_transporter;
    VAR wztemporary Varna_poza;
    CONST robtarget Vmesna:=[[1128.572292448,-1037.004921121,911.751561213],[0,0.363078838,0.931758422,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Vmesna_nazaj:=[[1128.572292448,-1037.004921121,961.751561213],[0,0.363078838,0.931758422,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Pred_Odlaganje_B:=[[379.85,400,-220],[0.707106781,0,0,-0.707106781],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Odlaganje_B:=[[379.85,400,0],[0.707106781,0,0,-0.707106781],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Vmesna_2:=[[981.302789512,-901.684215001,911.751561213],[0,0.363078838,0.931758422,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST jointtarget Domov:=[[0,-30,-15,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    !Odstopanje od tocke domov
    CONST jointtarget del_Domov:=[[1,1,1,1,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Pred_Odlaganje_izmet:=[[0,1450,979],[0,-0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Odlaganje_izmet:=[[0,1450,679],[0,-0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Prijemanje_gor_test:=[[0,0,-220],[1,0,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];


    PROC main()
        WaitDI Start_of_program,1;
        Inicializacija;
        init_interrupt;
        def_delovna_obmocja;
        def_varne_poz;
        Varno_vracanje_domov;
        !Ali je prijemalo odprto in drzi komade?
        Preveri_prijemalo;
        WHILE TRUE DO
            IF Orientacija_sloja=1 THEN
                WHILE vrstica<10 DO
                    nadaljujA:
                    Program_prijemanja;
                    IF test1=True THEN
                        MoveL Prijemanje_gor_test,v500,z50,Prijemalo_paletizacija_1\WObj:=KS_Prijemanje;
                        Izmet_komadov_v_prijemalu;
                        test1:=FALSE;
                        IEnable;
                        !Po prekinitvi nadaljuj zgoraj
                        GOTO nadaljujA;
                    ENDIF
                    Odlaganje_A_orientacija;
                ENDWHILE
                vrstica:=0;
                sloj:=sloj+1;
            ENDIF

            IF Orientacija_sloja=0 THEN
                WHILE vrstica<6 DO
                    WHILE stolpec<2 DO
                        nadaljujB:
```

```
                Program_prijemanja;
                IF test1=True THEN
                    MoveL Prijemanje_gor_test,v500,z50,Prijemalo_paletizacija_1\WObj:=KS_Prijemanje;
                    Izmet_komadov_v_prijemalu;
                    test1:=FALSE;
                    IEnable;
                    !Po prekinitvi nadaljuj zgoraj
                    GOTO nadaljujB;
                ENDIF
                Odlaganje_B_orientacija;
                stolpec:=stolpec+1;
            ENDWHILE
            stolpec:=0;
            vrstica:=vrstica+1;
        ENDWHILE
        vrstica:=0;
        sloj:=sloj+1;
    ENDIF
    IF sloj=4 THEN
        sloj:=0;
    ENDIF
ENDWHILE

ENDPROC

PROC Inicializacija()
    sirina_off:=0;
    vrstica:=0;
    sloj:=0;
    stolpec:=0;
    stolpec_off:=0;
    visina_off:=0;
ENDPROC

PROC Program_prijemanja()
    MoveJ Pred_prijemanje,v2000,z50,Prijemalo_paletizacija_1\WObj:=KS_Prijemanje;
    !Cakanje na komade!
    WaitDI Komadi_pripravljeni,1;
    MoveL Prijemanje_komada,v500,fine,Prijemalo_paletizacija_1\WObj:=KS_Prijemanje;
    ZapriPrijemalo;
    MoveL Prijemanje_gor,v500,z50,Prijemalo_paletizacija_1\WObj:=KS_Prijemanje;
ENDPROC

PROC Odlaganje_A_orientacija()
    PredOdlaganje_paleta:=Pred_odlaganje;
    Odlaganje_paleta:=Odlaganje;
    sirina_off:=vrstica*sirina_paketa;
    !Offset visine
    visina_off:=sloj*visina_paketa;
    PredOdlaganje_paleta:=Offs(PredOdlaganje_paleta,sirina_off,0,-visina_off);
    Odlaganje_paleta:=Offs(Odlaganje_paleta,sirina_off,0,-visina_off);
    !Offsetiraj vmesno tocko pred odlaganjem, da preprecis trk s paleto!
    Vmesna_off:=Vmesna;
    IF sloj>2 THEN
        Vmesna_off:=Offs(Vmesna_off,0,0,(100*(sloj-2)));
    ENDIF
    MoveJ Vmesna_off,v2000,z50,Prijemalo_paletizacija_1\WObj:=wobj0;
    MoveJ PredOdlaganje_paleta,v2000,z50,Prijemalo_paletizacija_1\WObj:=KS_odlaganje;
    ! Preverjanje, ali je paleta pripravljena?
    WaitDI Paleta_v_poziciji,1;
    MoveL Odlaganje_paleta,v200,fine,Prijemalo_paletizacija_1\WObj:=KS_odlaganje;
    OdpriPrijemalo;
    MoveL PredOdlaganje_paleta,v500,z10,Prijemalo_paletizacija_1\WObj:=KS_odlaganje;
    vrstica:=vrstica+1;
    !Offsetiraj vmesno tocko pred prijemanjem, da preprecis trk s paleto!
    Vmesna_nazaj_off:=Vmesna_nazaj;
    IF sloj>2 THEN
        Vmesna_nazaj_off:=Offs(Vmesna_nazaj_off,0,0,(100*(sloj-2)));
    ENDIF
    MoveJ Vmesna_nazaj_off,v2000,z50,Prijemalo_paletizacija_1\WObj:=wobj0;
ENDPROC

PROC Odlaganje_B_orientacija()
    PredOdlaganje_paleta:=Pred_Odlaganje_B;
    Odlaganje_paleta:=Odlaganje_B;
    !offsetiramo sirino
    sirina_off:=vrstica*(sirina_paketa+21);
    !offsetiramo dolzino
    stolpec_off:=stolpec*dolzina_paketa3;
    !offsetiramo visino
```

```
    visina_off:=sloj*visina_paketa;
    PredOdlaganje_paleta:=Offs(PredOdlaganje_paleta,stolpec_off,-sirina_off,-visina_off);
    Odlaganje_paleta:=Offs(Odlaganje_paleta,stolpec_off,-sirina_off,-visina_off);
    !Offsetiraj vmesno tocko pred odlaganjem, da preprecis trk s paleto!
    Vmesna_off:=Vmesna;
    IF sloj>2 THEN
        Vmesna_off:=Offs(Vmesna_off,0,0,(100*(sloj-2)));
    ENDIF
    MoveJ Vmesna_off,v2000,z50,Prijemalo_paletizacija_1\WObj:=wobj0;
    MoveJ PredOdlaganje_paleta,v2000,z50,Prijemalo_paletizacija_1\WObj:=KS_odlaganje;
    MoveL Odlaganje_paleta,v200,fine,Prijemalo_paletizacija_1\WObj:=KS_odlaganje;
    OdpriPrijemalo;
    MoveL PredOdlaganje_paleta,v500,z10,Prijemalo_paletizacija_1\WObj:=KS_odlaganje;
    !Offsetiraj vmesno tocko pred prijemanjem, da preprecis trk s komadom!
    Vmesna_nazaj_off:=Vmesna_nazaj;
    IF sloj>2 THEN
        Vmesna_nazaj_off:=Offs(Vmesna_nazaj_off,0,0,(100*(sloj-2)));
    ENDIF
    MoveJ Vmesna_nazaj_off,v2000,z50,Prijemalo_paletizacija_1\WObj:=wobj0;
ENDPROC

PROC Preveri_prijemalo()
    IF Prijemalo_D_odprto=0 OR Prijemalo_L_odprto=0 THEN
        Izmet_komadov_v_prijemalu;
    ENDIF
ENDPROC

PROC OdpriPrijemalo()
    SetDO Zapri_prijemalo,0;
    SetDO Odpri_prijemalo,1;
    WaitTime 0.5;
ENDPROC

PROC ZapriPrijemalo()
    SetDO Odpri_prijemalo,0;
    SetDO Zapri_prijemalo,1;
    WaitTime 1;
ENDPROC

PROC Varno_vracanje_domov()
    Vracanje_domov_off:=CRobT(\Tool:=Prijemalo_paletizacija_1\WObj:=wobj0);
    IF Vracanje_domov_off.trans.z<600 THEN
        Vracanje_domov_off:=Offs(Vracanje_domov_off,0,0,500);
    ENDIF
    IF Vracanje_domov_off.trans.z>600 THEN
        Vracanje_domov_off:=Offs(Vracanje_domov_off,0,0,310);
    ENDIF
    MoveJ Vracanje_domov_off,v400,z50,Prijemalo_paletizacija_1\WObj:=wobj0;
    MoveAbsJ Domov,v400,z10,Prijemalo_paletizacija_1\WObj:=wobj0;
ENDPROC

PROC Izmet_komadov_v_prijemalu()
    MoveJ Pred_Odlaganje_izmet,v500,z50,Prijemalo_paletizacija_1\WObj:=wobj0;
    MoveL Odlaganje_izmet,v400,fine,Prijemalo_paletizacija_1\WObj:=wobj0;
    OdpriPrijemalo;
    MoveL Pred_Odlaganje_izmet,v500,z50,Prijemalo_paletizacija_1\WObj:=wobj0;
    MoveAbsJ Domov,v400,z10,Prijemalo_paletizacija_1\WObj:=wobj0;
ENDPROC

PROC init_interrupt()
    CONNECT test_prekinitev WITH Test_komadov;
    ISignalDI Prekinitev_test,1,test_prekinitev;
    !Aktivacija interrupta
    IEnable;
ENDPROC

TRAP Test_komadov
    IDisable;
    test1:=true;
ENDTRAP

PROC def_delovna_obmocja()
    VAR shapedata volumen1;
    VAR shapedata volumen2;
    !Definiraj zacetno tocko prvega kvadra
    CONST pos t1:=[1208.3,-675,737.8];
    !Definiraj diagonalno tocko prvega kvadra
    CONST pos t2:=[1808.3,325,1237.8];
    CONST pos t3:=[620,-990,35];
    CONST pos t4:=[-680,-2060,1035];
```

```
    !Definiran kvader nad vhodnim transporterjem
    WZBoxDef \Inside,volumen1,t1,t2;
    !Definiran kvader nad izhodnim transporterjem
    WZBoxDef \Inside,volumen2,t3,t4;
    WZDOSet \Temp,zona_vhodni_transporter\Inside,volumen1,Poz_vh_trn,1;
    WZDOSet \Temp,zona_izhodni_transporter\Inside,volumen2,Poz_izh_trn,1;
  ENDPROC

  PROC def_varne_poz()
    VAR shapedata obmocje_sklepov;
    !Def varno pozicijo okoli tocke Domov
    WZHomeJointDef \Inside,obmocje_sklepov,Domov,del_Domov;
    !Ce je robot znotraj, aktiviraj DO
    WZDOSet \Temp,Varna_poza \Inside,obmocje_sklepov,Varna_poz,1;
  ENDPROC

ENDMODULE
```

Priloga 4:

Program iskanja izdelkov, uporaba »SearchL« in upravljalca napak

```
MODULE Module1
    CONST jointtarget JointHome:=[[0,0,0,0,30,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    VAR robtarget TockaZacetek;          !Tocka na zacetku zalogovnika
    VAR robtarget TockaKonec;            !Tocka na koncu zalogovnika
    VAR robtarget TockaZaznave;
    VAR robtarget TockaZaznavePrijemanje;
    VAR robtarget PraznaTocka;
    PERS tooldata Trenutno_orodje;
    VAR num I;
    VAR num J;
    VAR num K;
    VAR num L;
    VAR num II;
    VAR bool KomadVPrijemalu1;
    VAR bool KomadVPrijemalu2;
    CONST robtarget Tocka_odlaganja:=[[50,-50,-50],[1,0,0,0],[-1,-1,-2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Vmesna:=[[476.19119858,-65.597875063,832.345802868],[0.047048162,0.705539842,-0.047048214,0.705539841],[-1,-1,-
2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    VAR robtarget OdlaganjeOffs1;
    VAR robtarget OdlaganjeOffs2;
            CONST robtarget VmesnaNazaj:=[[470.469942006,-110.277022795,802.910519136],[0.700685988,0.103737122,0.701011438,-0.082829454],[-
1,0,-1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Referencna_tocka_zaznave:=[[50,0,-100],[1,0,0,0],[-1,0,-1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    PROC main()
        Init;
        Iskanje_komadov;
    ENDPROC

    PROC Init()
        I:=1;
        J:=1;
        K:=1;
        L:=1;
    ENDPROC

    PROC Iskanje_komadov()
        MoveAbsJ JointHome,v1000,z100,tool0\WObj:=wobj0;
        MoveJ Referencna_tocka_zaznave,v200,z30,Prijemalo_2\WObj:=KS_Vhodni_zalogovnik;
        FOR I FROM 1 TO 4 DO
            FOR J FROM 1 TO 5 DO
                II:=I;
                TockaZaznave:=PraznaTocka;
                TockaZacetek:=Referencna_tocka_zaznave;
                TockaKonec:=Referencna_tocka_zaznave;
                !Program za prvo in tretjo vrstico
                IF I=1 OR I=3 THEN
                    TockaZacetek:=Offs(TockaZacetek,(I-1)*100,(J-1)*100,0);
                    TockaKonec:=Offs(TockaKonec,(I-1)*100,J*100,0);
                    IF KomadVPrijemalu2=False THEN
                        MoveJ TockaZacetek,v300,fine,Prijemalo_2\WObj:=KS_Vhodni_zalogovnik;
                        SearchL\Stop,InSenzor2,TockaZaznave,TockaKonec,v50,Prijemalo_2\WObj:=KS_Vhodni_zalogovnik;
                        IF TockaZaznave<>PraznaTocka THEN
                            TockaZaznave:=Offs(TockaZaznave,0,20.5,0);
                            TockaZaznavePrijemanje:=Offs(TockaZaznave,0,0,50);
                            MoveL TockaZaznave,v100,z0,Prijemalo_2\WObj:=KS_Vhodni_zalogovnik;
                            MoveL TockaZaznavePrijemanje,v100,fine,Prijemalo_2\WObj:=KS_Vhodni_zalogovnik;
                            PulseDO\High,\PLength:=2,Zapri_prijemalo2;
                            KomadVPrijemalu2:=TRUE;
                            WaitTime 1;
                            TockaZaznave:=Offs(TockaZaznave,0,0,-50);
                            MoveL TockaZaznave,v300,z0,Prijemalo_2\WObj:=KS_Vhodni_zalogovnik;
                            TockaZaznave:=PraznaTocka;
                        ENDIF
                    ENDIF
                    IF KomadVPrijemalu1=FALSE AND KomadVPrijemalu2=TRUE THEN
                        MoveJ TockaZacetek,v300,fine,Prijemalo_1\WObj:=KS_Vhodni_zalogovnik;
                        SearchL\Stop,InSenzor1,TockaZaznave,TockaKonec,v50,Prijemalo_1\WObj:=KS_Vhodni_zalogovnik;
                        IF TockaZaznave<>PraznaTocka THEN
                            TockaZaznave:=Offs(TockaZaznave,0,20.5,0);
                            TockaZaznavePrijemanje:=Offs(TockaZaznave,0,0,50);
                            MoveL TockaZaznave,v100,z0,Prijemalo_1\WObj:=KS_Vhodni_zalogovnik;
                            MoveL TockaZaznavePrijemanje,v100,fine,Prijemalo_1\WObj:=KS_Vhodni_zalogovnik;
                            PulseDO\High,\PLength:=2,Zapri_prijemalo1;
```

```
                KomadVPrijemalu1:=TRUE;
                WaitTime 1;
                TockaZaznave:=Offs(TockaZaznave,0,0,-50);
                MoveL TockaZaznave,v300,z0,Prijemalo_1\WObj:=KS_Vhodni_zalogovnik;
                TockaZaznave:=PraznaTocka;
            ENDIF
        ENDIF
        IF KomadVPrijemalu1=TRUE and KomadVPrijemalu2=TRUE THEN
            Odlaganje;
        ENDIF
    ENDIF
    !Program za drugo in četrto vrstico
    IF I=2 OR I=4 THEN
        TockaZacetek:=Offs(TockaZacetek,(I-1)*100,(5-J)*100,0);
        TockaKonec:=Offs(TockaKonec,(I-1)*100,(6-J)*100,0);
        IF KomadVPrijemalu2=FALSE THEN
            MoveL TockaKonec,v300,fine,Prijemalo_2\WObj:=KS_Vhodni_zalogovnik;
            SearchL\Stop,InSenzor2,TockaZaznave,TockaZacetek,v80,Prijemalo_2\WObj:=KS_Vhodni_zalogovnik;
            IF TockaZaznave<>PraznaTocka THEN
                TockaZaznave:=Offs(TockaZaznave,0,-21,0);
                TockaZaznavePrijemanje:=Offs(TockaZaznave,0,0,50);
                MoveL TockaZaznave,v100,z0,Prijemalo_2\WObj:=KS_Vhodni_zalogovnik;
                MoveL TockaZaznavePrijemanje,v100,fine,Prijemalo_2\WObj:=KS_Vhodni_zalogovnik;
                PulseDO\High,\PLength:=2,Zapri_prijemalo2;
                KomadVPrijemalu2:=TRUE;
                WaitTime 1;
                MoveL TockaZaznave,v300,z0,Prijemalo_2\WObj:=KS_Vhodni_zalogovnik;
                TockaZaznave:=PraznaTocka;
            ENDIF
        ENDIF

        IF KomadVPrijemalu1=FALSE AND KomadVPrijemalu2=TRUE THEN
            MoveL TockaKonec,v300,fine,Prijemalo_1\WObj:=KS_Vhodni_zalogovnik;
            SearchL\Stop,InSenzor1,TockaZaznave,TockaZacetek,v80,Prijemalo_1\WObj:=KS_Vhodni_zalogovnik;
            IF TockaZaznave<>PraznaTocka THEN
                TockaZaznave:=Offs(TockaZaznave,0,-21,0);
                TockaZaznavePrijemanje:=Offs(TockaZaznave,0,0,50);
                MoveL TockaZaznave,v100,z0,Prijemalo_1\WObj:=KS_Vhodni_zalogovnik;
                MoveL TockaZaznavePrijemanje,v100,fine,Prijemalo_1\WObj:=KS_Vhodni_zalogovnik;
                PulseDO\High,\PLength:=2,Zapri_prijemalo1;
                KomadVPrijemalu1:=TRUE;
                WaitTime 1;
                MoveL TockaZaznave,v300,z0,Prijemalo_1\WObj:=KS_Vhodni_zalogovnik;
                TockaZaznave:=PraznaTocka;
            ENDIF
        ENDIF
        IF KomadVPrijemalu1=TRUE and KomadVPrijemalu2=TRUE THEN
            Odlaganje;
        ENDIF
    ENDIF
  ENDFOR
 ENDFOR
ERROR
 IF ERRNO=ERR_WHLSEARCH THEN
    StorePath;  !Shrani trenutno generirano trajektorijo robota
    IF II=1 or II=3 THEN    !Za vrstici 1 in 3
        Trenutno_orodje:=CTool();   !Shrani trenutno aktivno orodje v spremeljivko
        MoveL TockaKonec,v100,fine,Trenutno_orodje\WObj:=KS_Vhodni_zalogovnik;
    ENDIF
    IF II=2 OR II=4 THEN    !Za vrstici 2 in 4
        Trenutno_orodje:=CTool();   !Shrani trenutno aktivno orodje v spremeljivko
        MoveL TockaZacetek,v100,fine,Trenutno_orodje\WObj:=KS_Vhodni_zalogovnik;
    ENDIF
    RestoPath;  !Obnovi shranjeno generirano trajektorijo robota
    ClearPath;  !Počisti shranjeno generirano trajektorijo robota
    StartMove;  !Začni z nadaljnjim izvajanjem programa
    TRYNEXT;    !Nadaljuj za ukazom, kjer se je pojavila napaka
 ENDIF
ENDPROC

PROC Odlaganje()
 MoveJ Vmesna,v500,z50,tool0\WObj:=wobj0;
 IF KomadVPrijemalu1=TRUE THEN
    OdlaganjeOffs1:=Tocka_odlaganja;
    OdlaganjeOffs2:=Tocka_odlaganja;
    OdlaganjeOffs1:=Offs(OdlaganjeOffs1,(K-1)*100,-(L-1)*100,-100);
    MoveJ OdlaganjeOffs1,v300,z10,Prijemalo_1\WObj:=KS_Izhodni_zalogovnik;
    OdlaganjeOffs2:=Offs(OdlaganjeOffs2,(K-1)*100,-(L-1)*100,0);
    MoveL OdlaganjeOffs2,v80,fine,Prijemalo_1\WObj:=KS_Izhodni_zalogovnik;
    KomadVPrijemalu1:=FALSE;
```

```
            PulseDO\high,\PLength:=2,Odpri_prijemalo1;
            WaitTime 1;
            OdlaganjeOffs1:=Offs(OdlaganjeOffs1,0,0,-50);
            MoveL OdlaganjeOffs1,v300,z10,Prijemalo_1\WObj:=KS_Izhodni_zalogovnik;
        ENDIF
        IF L<5 THEN
            L:=L+1;
        ELSE
            K:=K+1;
            L:=1;
        ENDIF
        IF K=4 THEN
            K:=1;
        ENDIF
        IF KomadVPrijemalu2=TRUE THEN
            OdlaganjeOffs1:=Tocka_odlaganja;
            OdlaganjeOffs2:=Tocka_odlaganja;
            OdlaganjeOffs1:=Offs(OdlaganjeOffs1,(K-1)*100,-(L-1)*100,-100);
            MoveJ OdlaganjeOffs1,v300,z10,Prijemalo_2\WObj:=KS_Izhodni_zalogovnik;
            OdlaganjeOffs2:=Offs(OdlaganjeOffs2,(K-1)*100,-(L-1)*100,0);
            MoveL OdlaganjeOffs2,v80,fine,Prijemalo_2\WObj:=KS_Izhodni_zalogovnik;
            KomadVPrijemalu2:=FALSE;
            PulseDO\high,\PLength:=2,Odpri_prijemalo2;
            WaitTime 1;
            OdlaganjeOffs1:=Offs(OdlaganjeOffs1,0,0,-50);
            MoveL OdlaganjeOffs1,v300,z10,Prijemalo_2\WObj:=KS_Izhodni_zalogovnik;
        ENDIF
        IF L<5 THEN
            L:=L+1;
        ELSE
            K:=K+1;
            L:=1;
        ENDIF
        IF K=4 THEN
            K:=1;
        ENDIF
        MoveJ VmesnaNazaj,v500,z50,tool0\WObj:=wobj0;
    ENDPROC
ENDMODULE
```

Univerzitetna založba
Univerze v Mariboru

Stvarno kazalo

PROGRAMIRANJE INDUSTRIJSKIH ROBOTOV

TIMI KARNER IN JANEZ GOTLIH
Univerza v Mariboru, Fakulteta za strojništvo, Maribor, Slovenija.
E-pošta: timi.karnere@um.si, janez.gotlih@guest.um.si

Povzetek Učbenik zajema osnove programiranja, pravilne izbire industrijskih robotov, postavitve, zagon, prve nastavitve, nastavitve koordinatnega sistema orodja, obdelovancev, uporaba odločitvenih vej, ponovitvenih zank, prekinitvenih rutin, serijske komunikacije z uporabo PROFINET protokola ter podaja smernice za varno vračanje robota v izhodiščno točko. Prav tako podaja smernice za pripravo robota za operaterja.

Ključne besede:
programiranje industrijskih robotov, konfiguracija, koordinatni sistemi, PROFINET, varnost

Univerzitetna založba Univerze v Mariboru
DOI https://doi.org/10.18690/um.fs.2.2022
ISBN 978-961-286-570-2

Univerza v Mariboru

Fakulteta za strojništvo