Saku Ilvonen ja Jesse Virtanen

Robotiikan hyödyntäminen teollisuudessa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Automaatiotekniikka

Insinöörityö

8.4.2016

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Saku Ilvonen, Jesse Virtanen Robotiikan hyödyntäminen teollisuudessa 61 sivua + 3 liitettä 8.4.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Automaatiotekniikka
Suuntautumisvaihtoehto	
Ohjaaja(t)	Lehtori Timo Tuominen

Tämä insinöörityö tehtiin Metropolia Ammattikorkeakoululle. Työn aihe, robotiikan hyödyntäminen teollisuudessa, oli laaja ja se mahdollisti robotiikkaan perehtymisen monipuolisesti.

Käytännönosion tavoitteena oli ohjelmoida koulun käyttöön mahdollisimman realistinen lastausohjelma, jota voitaisiin käyttää myös messutarkoituksiin. Opittujen asioiden perusteella oli myös tarkoitus tehdä laboratorioharjoitus koulun käyttöön. Se esittelisi oppilaille sellaisia puolia robottien toiminnasta, joita nykyisissä harjoituksissa ei käydä läpi. Näiden uusien asioiden olisi tarkoitus viedä oppilaat lähemmäs sitä käytännön työtä, jota robottien parissa tehdään.

Robotin käyttöä harjoiteltiin runsaasti ennen varsinaisen työn alkua. Useamman ohjelmointiharjoituksen jälkeen valmistui kaksi lastausohjelmaa, joista toinen, automaattisesti toimivana, soveltuu messutarkoitukseen, ja toinen antaa käyttäjälle monipuolisia mahdollisuuksia varastoinnin suhteen.

Ohjelmointiharjoitusten aikana selvisi, että muuttujien ja rutiinien käyttö helpottaa ohjelmointityötä merkittävästi. Tästä syystä koululle tehty laboratorioharjoitus keskittyi juuri näihin aiheisiin.

Insinöörityön lopputuloksena syntyi laboratorioharjoitus, jonka ymmärtäminen monipuolistaa opiskelijan näkemystä robotiikan antamista mahdollisuuksista. Lisäksi syntyi kaksi lastausohjelmaa robotille, jotka ovat realistisia teollisuuden robottiohjelmia, ja koulu voi hyödyntää näitä messutilaisuuksissa ja muuten haluamallaan tavalla.

Avainsanat	Robotti, robotiikka

Author(s) Title Number of Pages Date	Saku Ilvonen, Jesse Virtanen Industrial Usage of Robotics 61 pages + 3 appendices 8 April 2016
Degree	Bachelor of Engineering
Degree Programme	Automation Technology
Specialisation option	
Instructor(s)	Timo Tuominen, Senior Lecturer

This Bachelor`s thesis was made for Helsinki Metropolia University of Applied Sciences. The topic, industrial usage of robotics, was comprehensive and it allowed a versatile approach to robotics.

The practical objective was to create a storage program as realistic as possible for the institute's use, which also could be used in trade fairs. Another objective was to create a laboratory exercise related to the learning outcomes during the process. It would introduce new ways of using robots that had not been considered in the existing exercises. The new exercise was to enhance students' practical approach to using industrial robots.

A lot of practise was done with robots before the actual work. After several programming exercises, two storage programs were made. The other is more automatic and suitable for fairs and the other one gives the operator multiple choices of storage.

During the program exercises, it was found that by using robot variables and routines, programming is made easier. For this reason the laboratory exercise focuses on these matters.

The result of this Bachelor`s thesis was a laboratory exercise, the understanding of which gives the students more versatile view of what robotics has to offer. In addition, during this study, two realistic storage programs were created, which can be used for educational or fair purposes.

Keywords	Robot, robotics

Sisällys

Lyhenteet

NPV	Net Present Value. Nettonykyarvo on tulojen ja menojen nykyarvojen erotus.
WACC	Weighted average cost of capital. Kertoo yrityksen pääoman keskimääräisen kustannuksen.

1 Johdanto

Tämä insinöörityö tehtiin Metropolia Ammattikorkeakoulussa kiinnostuksena robotiikkaa kohtaan. Opinnäytetyön tarkoituksena oli perehtyä robottien käyttöön teollisuudessa.

Teoriaosuudessa mietimme laajemmin teollisuusrobottien taloudellista kannattavuutta ja teollisuusrobotin toimintaa. Teoriaosuus jaettiin kahteen osaan, laskennalliseen ja kinemaattiseen -osioon, sekä robottien rakenne ja ohjausten toteutus -osioon. Pääsääntöisesti opinnäytetyö käsittelee kiertyvänivelisiä teollisuusrobotteja.

Käytännön harjoituksessa simuloitiin yksinkertaista varastointiohjelman toteutusta, joka vastaa teollisuudessa paljon käytettyä robottisovellusta. Työnjaollisesti insinöörityö oli tasapuolinen. Käytännönharjoitus tehtiin yhdessä ja teoriaosuus saatiin jaettua tasapuolisesti. Vaikka tekstin kirjoittaminen jaettiin tasan työn tekijöiden kesken, molemmat perehtyivät aihealueeseen kokonaisuudessaan.

2 Robotiikan määritelmä ja historia

Robotti käsitteenä sai alkuunsa novellisti Karel Capekin novellista vuonna 1921 ”Rossums`s Universal Robots”, jossa hän loi mielikuvituksellisen koneen väsymättömästä ihmisen kaltaisesta koneesta, robotista. Robotilla hän tarkoitti epäinhimillistä, toistuvaa ja orjallista työn tekemistä. [1, s. 10; 2, s. 2.]

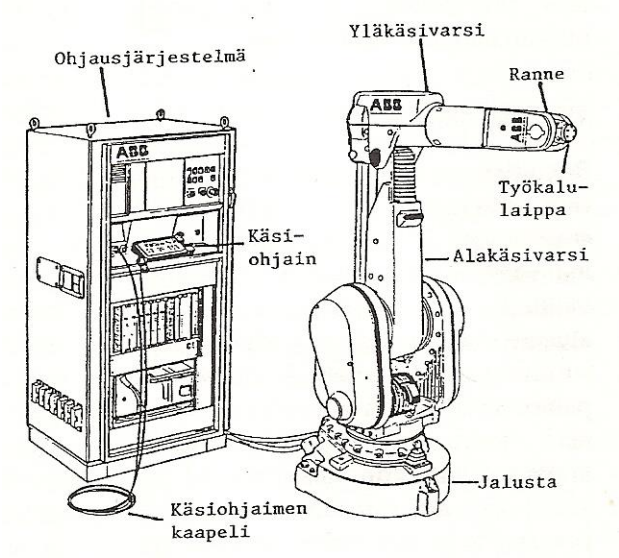
Ensimmäinen teollisuusrobotin ”Unimates” kehittivät George Devol ja Joseph Engelberger 1950-luvulla. Tietotekniikan ja mikroprosessorien kehitys mahdollisti kaupalliset, ensimmäiset mikroprosessoriohjatut robotit vasta vuonna 1974. Ilmestyttäessään robottien kehitys lähti nopeasti nousuun ja roboteista tuli suurenluokan liiketoiminta. [1, s. 10; 2, s. 6.]

Tieteisfiktiokirjailija Isaac Asimov määritti teoksissaan robotiikan lait. Hän määritteli robotin lakeihin ominaisuuksia. Robotti ei esimerkiksi saa vahingoittaa ihmiskuntaa. [1, s. 10.] Nykyisin robotin määritelmä kuuluu seuraavasti:

> Robotti on uudelleen ohjelmoitavissa oleva monipuolinen vähintään kolminivelinen mekaaninen laite, joka on suunniteltu liikuttamaan kappaleita, osia, työkaluja tai erikoislaitteita ohjelmoitavin liikkein monenlaisten tehtävien suorittamiseksi teollisuuden sovelluksissa [3, s. 13].

Määritelmän mukaan voidaan yksinkertaisesti todeta teollisuusrobotin koostuvan toistensa suhteen liikkuvista tukivarsista, joita kutsutaan niveliksi. Niveliä liikuttavat servotoimilaitteet etukäteen määriteltyjen liikeratojen mukaan erillisten tai ulkoisten tuntoelimien perusteella [3, s. 13–14].

Teollisuusrobotti muodostuu tyypillisesti kahdesta pääkomponentista, ohjausjärjestelmästä ja mekaanisesta manipulaattorista (kuva 1) [3, s. 34].



Kuva 1. Teollisuusrobotin peruskomponentit [3, s. 13].

Ohjausjärjestelmä pitää tyypillisesti sisällään esimerkiksi robotin keskusyksikön tietojenkäsittelyä varten ja nivelkohtaiset servo-ohjaimet [3, s. 34].

3 Perusteet robotin hankinnalle

Kun mietitään teollisuusrobotin hankinnan taloudellista aspektia, on hyvä muistaa, että on syitä, jotka puoltavat robotin hankintaa, vaikka se ei olisi taloudellista. Ihmisen voi korvata robotilla myös turvallisuus- ja terveyssyistä. [4, s. 23.]

Kun verrataan ihmisen toimintakykyä robottiin, huomataan ihmisen toimivan pienillä painoilla ja lyhyillä etäisyyksillä jopa robottia nopeammin. Etäisyyden ja kuorman kasvaessa etu kääntyy robotille. [4, s. 23–24.] Myös tämä on huomioitava sijoituspäätöstä tehtäessä.

Robotin hankkimista suunniteltaessa on huomioitava useita asioita. Robotin nopeuden on oltava sellainen, että se pystyy suoriutumaan annetuista tehtävistä. Sen täytyy myös pystyä käsittelemään haluttuja kuormia haluttujen etäisyyksien päässä käyttäen sopivaa työkalua. Näiden lisäksi on mietittävä, millaiseen tarkkuuteen robotin on kyettävä, kuinka monta niveltä se toimintaansa tarvitsee ja miten sen ohjelmoiminen tultaisiin suorittamaan. Kaikki nämä asiat liittyvät robotin valintaan hinnan ja käyttökustannusten lisäksi. [4, s. 25–26.] Joissain tapauksissa teollisuudessa ollaan jopa päädytty vähentämään robottien käyttöä, koska ne eivät pysty toimimaan yhtä joustavasti kuin ihmiset. Tästä esimerkkinä Mercedes Benzin ilmoitus vähentää robottiensa määrää autonvalmistuksessa huomattavasti [5].

Kun kaikki muut kriteerit robotin hankinnalle on täytetty, voidaan miettiä sen taloudellista vaikutusta. Robotin hankkimiseen tulee yrityksessä suhtautua kuin mihin tahansa investointiin. Tästä syystä robotin hankinnassa käytetään normaaleja investointilaskennan menetelmiä. Näitä ovat takaisinmaksuaika, sijoitetun pääoman tuotto sekä nykyarvomenetelmä. [4, s. 153.]

Yksinkertaisimmillaan takaisinmaksuajan voi laskea kaavalla:

$$P = \frac{I}{L - M} \tag{1}$$

jossa

P on takaisinmaksuaika

I on kokonaisinvestoinnin määrä

L on vuosittaiset säästöt nykyiseen verrattuna

M on robotista vuosittain aiheutuvat kulut

Tässä laskutavassa ei ole otettu huomioon inflaation vaikutusta tulevien vuosien säästöihin ja robotista aiheutuviin kuluihin [4, s. 153].

Jos robotti maksaa asennuksineen ja käyttöönottoineen kokonaisuudessa 30 000 euroa ja siitä ennustetaan tulevan vuosittain 2 500 euron kustannukset varaosien ja huoltojen muodossa, pitää vielä arvioida vuosittaiset kokonaissäästöt, mitä robotti tuo mukanaan, kun ihmisen ei enää tarvitse tehdä kyseisiä töitä. Tässä esimerkissä nämä säästöt arvioidaan 15 000 euron arvoisiksi. Takaisinmaksuajaksi saadaan kaavan 1 mukaan:

$$P = \frac{30000}{15000-2500} = 2{,}4$$

Takaisinmaksuajaksi saadaan siis 2,4 vuotta. Lasku on helppo laskea. Tärkeämpää onkin saada luvut oikein. Mikäli ostopäätöstä tehtäessä ei toimittajan kanssa olekaan ollut esisopimusta hintoineen tehtynä, kokonaishinta saattaakin olla 33 000 euroa. Säästöjen suhteen ei osattukaan ottaa kaikkia asioita huomioon ja todellisuudessa säästöjä saadaankin vain 12 000 euroa vuodessa. Robotin vuosittaiset kustannuksetkin ovat 2800 euroa. Tällöin kaava 1 antaa tulokseksi:

$$P = \frac{33000}{12000-2800} = 3{,}6$$

Ensimmäinen arvio takaisinmaksuajasta oli 2,4 vuotta ja jälkimmäinen päivitetyillä arvoilla oli 3,6 vuotta. Eroa on kolmannes. Takaisinmaksuaika on hyvä mittari päätettäessä investoinnista, mutta siihen käytetyt arvot on oltava mahdollisen totuudenmukaisesti arvioitu. Muuten laskun lopputulos on harhaanjohtava.

Sijoitetun pääoman tuotolla voidaan laskea, kuinka monta prosenttia saa sijoituksestasi takaisin vuosittain [4, s. 154].

$$R = \frac{Työkustannussäästöt - ylläpito - arvonalennus}{Kokonaisinvestointi} \quad (2)$$

jossa

R on sijoitetun pääoman tuottoprosentti

Esimerkkinä otetaan säästöt työkustannuksista 5 €/h ja töitä tehtiin yhdessä vuorossa vuoden ympäri ja nyt robotti hoitaa näiden ihmisten työt. Työkustannussäästöt ovat 5*8*365=14 600. Ylläpitoon kuluu 2 000 € ja arvonalennusta 30 000 € maksaneelle robotille tulee 3 000 €. Sijoitetaan luvut kaavaan.

$$R = \frac{14600 - 2000 - 3000}{30000} = 32\ \%$$

Kyseissä esimerkissä sijoittaja saa vuodessa jokaista sijoittamaansa euroa kohden 0,32 euroa takaisin. Täytyy siis olla varma, että sijoitus tuottaa useana vuonna, jotta siitä tulee kannattava.

Investoinnin kannattavuutta voidaan monipuolisemmin tarkastella laskemalla ensin koko yrityksen pääoman keskimääräinen kustannus, WACC. Tämän avulla voidaan laskea kannattavuus nykyarvomenetelmällä.

$$WACC = \frac{E}{D+E} * R_e + \frac{D}{D+E} * R_d * (1 - t) \quad (3)$$

jossa

WACC on yrityksen pääoman keskimääräinen kustannus

E on oma pääoma

D on vieras pääoma

Re on tuottovaade

Rd on vieraan pääoman hinta

t on tuloveroprosentti

Nykyarvomenetelmä tarvitsee diskonttauskertoimen toimiakseen. Diskonttauskerroin lasketaan kaavalla:

$$\frac{1}{(1+i)^n} \quad (4)$$

jossa

i on laskentakorkokanta

n on aikajakso vuosissa

Kaavalla 3 saatua WACC-arvoa käytetään laskentakorkokantana laskettaessa nettonykyarvoa.

Parhaiten nykyarvomenetelmä avautuu esimerkin avulla. Yrityksen oma pääoma on 7 miljoonaa euroa ja vieras pääoma on 3 miljoonaa euroa. Yrityksen johto vaatii 15 % tuottoa. Vieraan pääoman korko on 10 % ja tuloveroprosentti on 29 %. Nämä tiedot sijoitetaan kaavaan 3, ja lasketaan WACC.

$$WACC = \frac{7}{3+7} * 0{,}15 + \frac{3}{3+7} * 0{,}10 * (1 - 0{,}29) = 0{,}1263 \approx 12{,}6\,\%$$

Yritys harkitsee robotin hankkimista, ja on arvioitu sen hankkimishinnan olevan 30 000 €. Robotin vuosittaiseksi säästöksi on arvioitu 7 000 €. Sen käyttöiäksi on arvioitu 10 vuotta, jonka jälkeen se myydään arviolta 5 000 euron hintaan. Ensimmäisen kolmen vuoden ajan robotin käyttökustannuksiksi on arvioitu 1 000 €, seuraavien kolmen vuoden ajan 1 500 € ja viimeisen neljän vuoden ajan 2 000 €. Diskonttauskerrointa laskettaessa laskentakorkokantana käytetään WACC-arvoa. Näillä arvioiduilla ja lasketuilla tiedoilla voidaan laskea investoinnin kannattavuus. Esimerkkitapauksessa, jonka arvot esitetään taulukossa 1, nettonykyarvo, NPV, on 2 267 euroa. Koska luku on positiivinen, investointi on kannattava.

Taulukko 1. Esimerkki nykyarvolaskennasta

Vuosi	Investointi	Säästöt	Kulut	Netto-tulot	Jäännösarvo	Diskonttauskerroin	Nykyarvo
0	-30000					1	-30000
1		7000	1000	6000		0,8879	5327
2		7000	1000	6000		0,7883	4730
3		7000	1000	6000		0,6999	4199
4		7000	1500	5500		0,6214	3418
5		7000	1500	5500		0,5517	3035
6		7000	1500	5500		0,4899	2694
7		7000	2000	5000		0,4349	2175
8		7000	2000	5000		0,3862	1931
9		7000	2000	5000		0,3429	1714
10		7000	2000	5000	5000	0,3044	3044
						NPV	**2267**

4 Robottityypit ja -rakenteet

4.1 Robotin tarkkuus ja vapausasteet

Absoluuttinen tarkkuus määrittää robotin tarkkuuden peruskoordinaatistossa. Absoluuttinen tarkkuus ei huomioi ulkoisten voimien tai robotin tukivarsien aiheuttamaa taipumista. Tällä tarkoitetaan sitä, että esimerkiksi robotin paikka työkalupisteessä saattaa muuttua työkalun painosta johtuen. Absoluuttinen tarkkuus tulisi ottaa huomioon ohjelmointivaiheessa. Muuttuvista tekijöistä johtuen absoluuttinen tarkkuus voi olla ± 50 mm. [3, s. 14, 21–22.]

Toistotarkkuus määrittää robotin tilastollisen tarkkuuden palata pisteeseen, johon se on määritelty kulkemaan [3, s. 185].

Vapausasteeksi (DOF, Degree of Freedom) kutsutaan kiertyviä tai suoria niveliä. Vapausaste määrittyy käytännössä robotin nivelien määrästä. Lähes kaikissa kiertyvänivelisissä roboteissa vapausasteita on kuusi. Robotti voi myös menettää yhden tai useamman vapausasteen, kun työkalua ei voida mekaanisten tai liikeavaruuden reunoilla sijoittaa mielivaltaiseen asentoon. Tätä kutsutaan singulaaripisteeksi. [3, s. 15, 28, 37.]

4.2 Robottityypit

Valmistajat kehittävät jatkuvasti eri robottityyppejä. Vakiintuneet robottimallit esitetään Standardissa ISO 8373 niitten rakenteen, kinemaattisen kaavion ja työalueen mukaan. Vakiintuneita robottityyppejä ovat muuan muassa [3, s. 12–13]

- suorakulmainen robotti
- Scara-robotti
- kiertyvänivelinen robotti
- rinnakkaisrakenteinen robotti

Suorakulmaisessa robotissa (kuva 2) eli niin sanotussa portaalirobotissa tulee olla vähintään kolme vapausastetta, joista ensimmäiset lineaarisia x-, y-, z-akseleita. Tyypillisesti portaalirobotit kiinnitetään työalueen päälle toimimaan ylösalaisin. [3, s.16.]

Kuva 2. Suorakulmaisen robotin rakenne, kinemaattinen kaavio ja työalue [3, s. 12].

Scara-robotti (kuva 3) on vähintään neljän vapausasteen robotti, jossa on kolme kiertyvänivelistä niveltä ja yksi pystysuuntainen nivel [3, s. 16].

Kuva 3. Scara-robotin rakenne, kinemaattinen kaavio ja työalue [3, s. 12].

Kiertyvänivelinen robotti (kuva 4) on tyypillisesti kuuden vapausasteen robotti, jossa kaikki nivelet ovat kiertyviä. Robotin avoimen rakenteen ja vapausasteiden avulla sen työkalu voidaan määrittää mihin kohtaan vain työalueen rajoissa. [3, s. 18.]

Kuva 4. Kiertyvänivelisen robotin rakenne, kinemaattinen kaavio ja työalue [3, s. 12].

Rinnakkaisnivelinen robotti (kuva 5) on suljettuun kinemaattiseen rakenteeseen perustuva robotti. Rinnakkaisnivelisissä roboteissa, joissa käytetään kevyitä rakenteita, voidaan päästään suureen työskentelynopeuteen. [3, s. 17.]

Kuva 5. Rinnakkaisrakenteisen robotin rakenne, kinemaattinen kaavio ja työalue [3, s. 12].

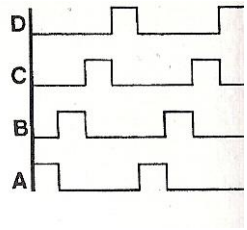
5 Robotin toiminta

5.1 Moottorit ja voimansiirto

Mekaanisen manipulaattorin nivelten liikuttamiseen toistensa suhteen voidaan käyttää esimerkiksi tasavirtaservomoottoreita, vaihtovirtaservomoottoreita tai askelmoottoreita.

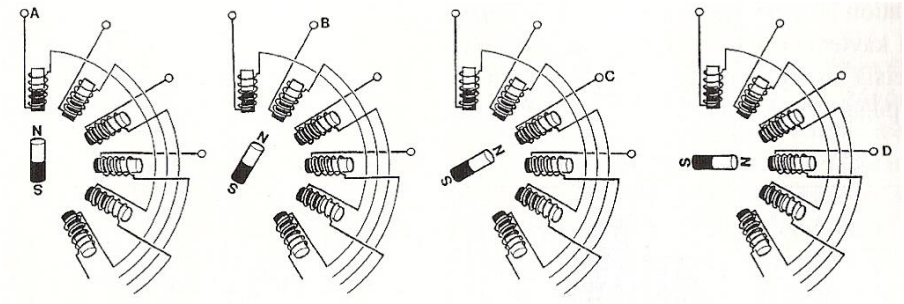
Servo-nimitys tulee latinankielen sanasta ”servus” eli orja. Tällä tarkoitetaan sitä, että servomoottoreista saattavan takaisinkytkentä paikanmittausantureiden avulla, sille voidaan määrittää tarkka käynnistyshetki, kiihdytysaika, liikenopeus, käynnissä oloaika, hidastusaika ja tarkka pysäytyspaikka. [3, s. 15, 19; 1, s. 43, 52.]

Askelmoottorin toiminta perustuu yhteen askelmaiseen liikkeeseen yhdellä ohjauskerralla, ja sen pyörimiskulman muutos on yleensä 0,9–15 astetta riippuen esimerkiksi moottorin hammaspyörästöistä. Tavanomaiset askelmoottorit ovat kaksi- tai neljäkäämisiä. Esimerkiksi neljäkääminen askelmoottori vaatii neljä kaksitilaista ohjauspulssia, joten ohjauksen toteuttamiseen vaaditaan elektronista ohjausta, esimerkiksi logiikkaohjaus. Ohjaus voidaan suorittaa kuvan 6 mukaisella logiikalla. Pyörimissuunta määrittyy käämien ohjausjärjestyksen vaihtamisella. [1, s. 44–45; 6, s. 138.]



Kuva 6. Neljäkäämisen askelmoottorin ohjauspulssit [6, s. 138].

Perinteisen askelmoottorin rakenne (kuva 7) perustuu kestomagneettiankkuriin ja staattorikäämeihin [6, s. 138].



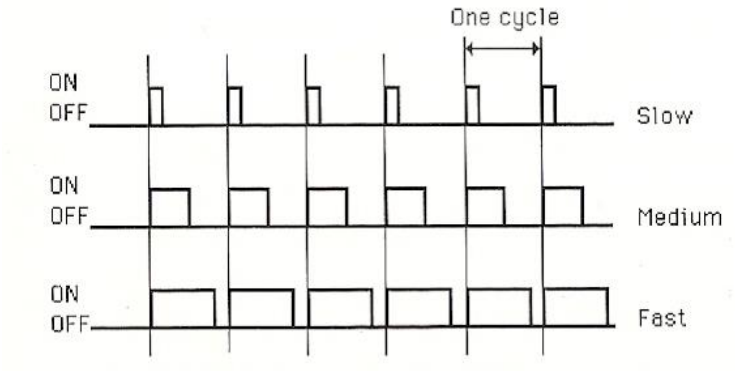
Kuva 7. Neljä käämisen askelmoottorin toimintaperiaate [6, s. 138].

Askelmoottoreiden asemanmittaustieto voidaankin laskea ohjauskertojen perusteella. Haittana on kumminkin esimerkiksi mahdollisten ylikuormien aiheuttamat luistot. Tämä aiheuttaa laskennallisesti vääriä paikoitustietoja. [1, s. 44–45.]

Tasavirtamoottoreissa moottorin nopeutta kontrolloidaan syötetyllä tehon määrällä ja nopeuteen vaikuttaa myös moottoriin vaikuttavat ulkoiset kuormat. Takaisinkytkennän ansiosta moottorille syötettävän tehon määrää voidaan korjata halutun nopeuden mukaan satoja kertoja sekunnissa.

Pääsääntöisesti on kaksi tapaa ohjata elektronisesti moottorin nopeutta. Ensimmäinen tapa on säädellä syötettävää jännitettä analogisesti, esimerkiksi 0–10 volttia.

Toinen tapa on ohjata vakiojännitteellä moottoria pulssimaisesti eli digitaalisesti (kuva 8). Digitaalinen ohjaus perustuu moottorille syötettävän pulssinpituuteen eli toisin sanoen tehon määrään yhdellä ohjelmakierrolla. Tapaa kutsutaan pulssinleveysmodulaatioksi (Pulse Width Modulation). [1, s. 52–53.]



Kuva 8. PWM:n pulssin toimintaperiaate [1, s. 53].

Moottorit voidaan asentaa lähelle haluttua niveltä tai käyttää hyödyksi voimansiirtomekanismeja. Voimansiirron tarkoituksena on välittää moottorin voimaa ja liikettä halutulle nivelelle. Voimansiirto voidaan toteuttaa esimerkiksi hammashihnoilla tai hammaspyörästöillä (kuva 9). [3, s. 19.]

Kuva 9. Erään robotin voimansiirto hammashihnalla.

Mekaanisilta rakenteilta teollisuusrobotit voidaan jakaa kahteen kinemaattiseen ryhmään. Tukivarsia, jotka on kytketty toisiinsa nähden peräkkäin, kutsutaan *avoimeksi kinemaattiseksi rakenteeksi*. Tukivarsia, jotka on kytketty rinnakkain, kutsutaan *suljetuksi kinemaattiseksi rakenteeksi.* Näitä rakenteita soveltaen saadaan erilaisia robotteja. [3, s. 16.]

Avoimessa rakenteessa etuna on suurempi ulottuvuus, mutta rakenne heikentää robotin käsittelykykyä työalueen reunoilla. Suljettu rakenne taas mahdollistaa kuorman jakamista useimmille rinnankytketyille nivelille, mutta robotin ulottuvuusalue heikkenee. [3, s. 16–17.]

5.2 Anturit

Teollisuusrobotin laskennalliseen paikkatietokoordinaatistoon tarvitaan jatkuvaa tietoa robotin nivelten asennoista ohjausjärjestelmälle. Ohjausjärjestelmässä paikkatieto voidaan esittää käyttäjälle muun muassa tukivarsien nivelkulmien arvona tai millimetreinä. Asematietoja voidaan lukea erilaisilla paikanmittausantureilla. Kun paikkatietoa luetaan hyvin nopealla syklillä, saadaan nivelen nopeustieto ja kiihtyvyys myös laskettua. Robotin paikanmittaus ja ohjaustaajuutta kutsutaan servointervalliksi. [3, s. 25, 30, 38.]

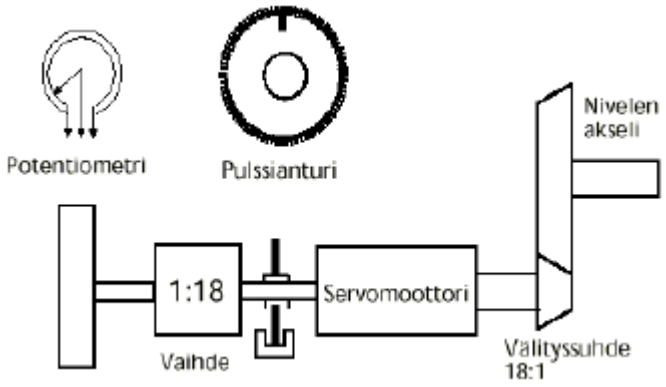
Tyypillisiä paikanmittausantureita:

- potentiometrit
- pulssianturit
- absoluuttianturit.

5.2.1 Pulssianturi

Pulssianturia voidaan soveltaa useammalla tavalla. Pelkkä pulssitieto ei tyypillisesti riitä esimerkiksi paikkatiedon hävitessä sähkökatkon yhteydessä. Ohjausjärjestelmälle voidaan ilmoittaa nollapiste kaksitilaisen rajakytkimen avulla. Tämä tarkoittaa sitä, että robotin nivelen liikkeen tulisi ohittaa rajakytkin mahdollisimman useasti ohjelmakierron aikana, jolloin ohjausjärjestelmä saisi aina riittävän tarkan tiedon asemasta. Pelkkä

yhden kierroksen synkronointipulssi ei riitä, vaan eräissä sovelluksissa käytetään myös potentiometriä määrittämään servomoottorin kierrosmäärä (kuva 10). [3, s. 30.]

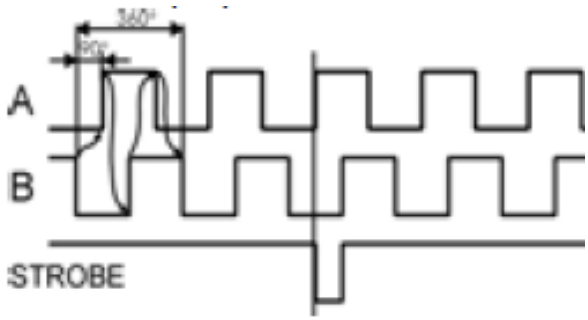


Kuva 10. Nivelen aseman mittausperiaate [3, s. 30].

5.2.2 Inkrementtianturi

Kulmaa mittaava inkrementtianturi on useamman kanavan pulssianturi, tavallisesti 3-kanavainen. Inkrementtianturia käyttäessä tulee ohjausjärjestelmän tietää nivelen aloitusasema järjestelmää käynnistäessä [3, s. 31]. Tämä tarkoittaa sitä, että esimerkiksi sähkökatkon aikana tapahtuvat robotin nivelten siirrot eivät rekisteröidy ohjausjärjestelmään, joten nivelet joudutaan kalibroimaan ohjausjärjestelmän käynnistysvaiheessa [3, s. 31].

Inkrementtianturi perustuu kahteen toisiinsa 90 asteen vaihe-erossa olevaan pulssiin, A- ja B-kanavaan (kuva 11). Lisäksi yksi kanava ilmoittaa yhden kierroksen täyttyneen. Anturin tarkkuuteen vaikuttaa sen yhden kierroksen aikana antama pulssimäärä/kanava. [3, s. 31.] Esimerkiksi 250 pulssia/kierros.



Kuva 11. Inkrementtipulssit [3, s. 31].

Kuvan 11 mukaisesti 90 asteen vaihe-erolla saadaan bittikombinaation perusteella selvitettyä inkrementtianturin pyörimissuunta.

Taulukko 2. Bittikombinaatio katsottuna kuvasta 11 vasemmalta oikealle.

A-kanava	B-kanava	A- kanavan muutos	B- kanavan muutos
0	0	1	0
0	1	0	0
1	1	0	1
1	0	1	1

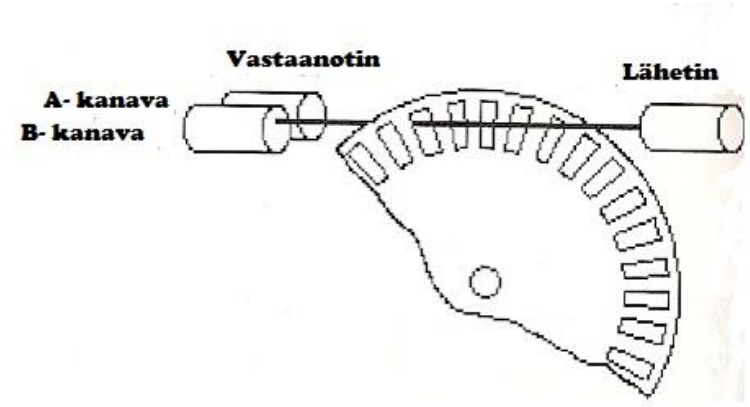
Taulukosta 2 voidaan todeta totuustaulukon mukaiset ehdot pyörimissuunnan varmistamiseksi.

Taulukko 3. Bittikombinaatio katsottuna kuvasta 11 oikealta vasemmalle.

A-kanava	B-kanava	A- kanavan muutos	B- kanavan muutos
0	0	0	1
0	1	1	1
1	1	1	0
1	0	0	0

Taulukosta 3 voidaan todeta totuustaulukon mukaiset ehdot pyörimissuunnan varmistamiseksi.

Inkrementtianturin fyysinen rakenne perustuu kahteen 90 astetta toisiinsa eroavaan koodikiekkoon, joissa kummassakin on läpinäkyviä tai läpinäkymättömiä viivoja. Pulssit saadaan aikaiseksi lukemalla kummankin kiekon viivatietoa infrapunalukuhaarukalla (kuva 12). [3, s. 31; 7.]

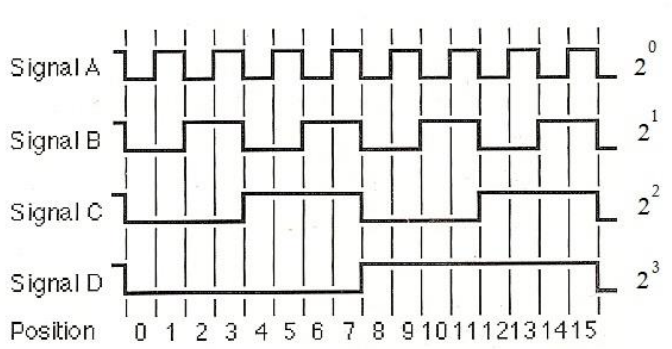


Kuva 12. Inkrementtianturin toimintaperiaate [1, s. 50].

5.2.3 Absoluuttianturi

Absoluuttianturin etuna on antaa ohjausjärjestelmälle reaaliaikaista paikkatietoa. Toisin sanoen esimerkiksi sähkökatkon aiheuttamat vaikutukset eivät vaadi erillistä nollapistetietoa. Absoluuttianturin toteutustapoja on useita esimerkiksi yksikierroksiset, monikierroksiset ja kenttäväylään soveltuvat. [3, s. 32; 7.]

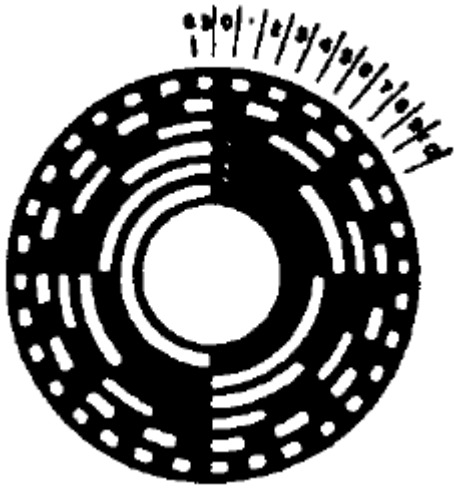
Yksinkertaisuudessaan yksikierroksiset absoluuttianturit perustuvat rinnakkaismuotoisiin binäärikoodeihin, joissa absoluuttianturin yhden johtimen viesti osoittaa sen bittiarvon (kuva 13).



Kuva 13. Neljäbittisen absoluuttianturin toimintaperiaate [1, s. 51].

Absoluuttianturin resoluutiosta riippuen kierrosta kohden voidaan saada esimerkiksi 1 024 (10 bittiä), 2 048 (11 bittiä) tai 4 096 (12 bittiä) asematietoa [3, s. 32].

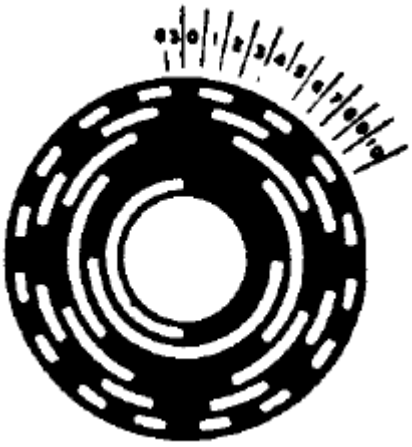
Absoluuttianturin rakenne perustuu inkrementtianturin tavoin koodikiekkoihin (kuva 14). Kiekot saadaan pyörimään välitysten avulla eri nopeutta, jolloin valoantureilla luettavat kiekot kertovat anturin nivelkulman [3, s. 32].



Kuva 14. Binäärinen koodikiekko [8, s. 10].

Binäärikoodauksessa saattaa esiintyä virheitä, johtuen usean bitin samanaikaisesta muutoksesta. Esimerkiksi desimaaliarvosta seitsemän (0000 0111) vaihtuessa kahdeksaan (0000 1000) vaihtaa neljä eri bittiä tilaansa. [8, s. 10.]

Ongelman voi ratkaista vaihtamalla binäärikoodaus Gray-koodaukseksi (kuva 15). Gray-koodauksessa vain yksi bitti kerrallaan vaihtaa tilaa, joten mahdollisten lukuvirheiden määrä pienenee [8, s. 10].

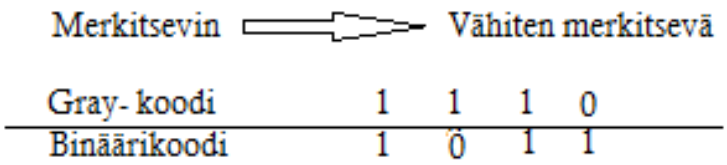


Kuva 15. Gray-koodikiekko [8, s. 10].

Gray-koodi on muutettavissa binäärikoodiksi [8, s. 10].

Koodin voi muuttaa helposti muutaman perussäännön avulla (kuva 16).

- Gray-koodin merkitsevin bitti on aina sama kuin binäärikoodin merkitsevin bitti.
- Jos seuraava Gray-bitti on nolla, pysyy seuraava binääri bitti samana.
- Jos seuraava Gray-bitti on yksi, vaihtaa binääri bitti tilaansa.



Kuva 16. Esimerkki Gray-binäärimuutoksesta.

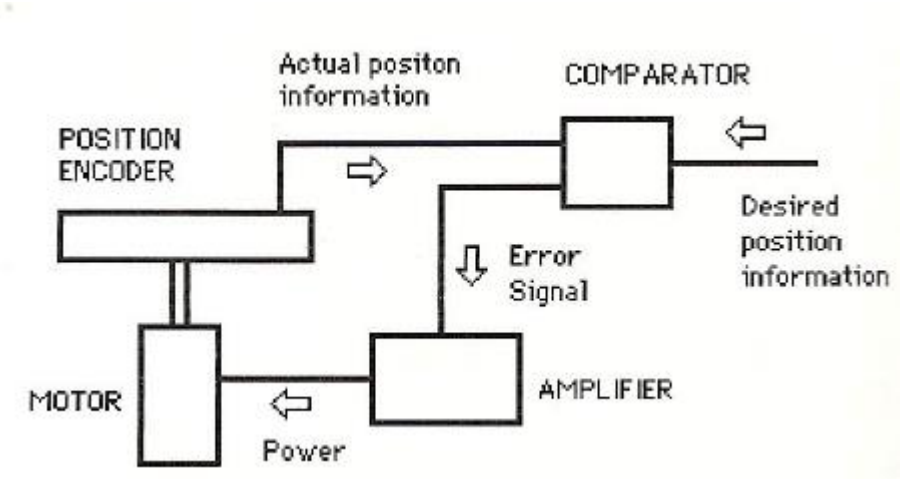
Seuraavaksi esitetään esimerkkinä ensimmäisen tavun desimaali, binäärikoodi ja Gray-koodi. Kuten taulukosta 4 voidaan todeta Gray-koodi muuttaa vain yhden bitin tilaa desimaaliarvon noustessa.

Taulukko 4. Taulukko yhden tavun koodimuutoksesta

Desimaali	Binäärikoodi	Gray- koodi
0	000 000	000 000
1	000 001	000 001
2	000 010	000 011
3	000 011	000 010
4	000 100	000 110
5	000 101	000 111
6	000 110	000 101
7	000 111	000 100

5.3 Robotin ohjaus

Robotin moottoreiden, toisin sanoen nivelten, liikuttamiseen tarvitaan servo-ohjainta. Servo-ohjaimen tarkoituksena on käyttää takaisinkytkettyjä paikkatietoja ja analysoida ja laskea tarvittavat moottorin ohjaukseen liittyvät parametrit. Yksinkertaisuudessaan robotin ohjaus vaatii jonkinasteisen takaisinkytkennän (kuva 17). On siis tiedettävä robotin aloitus- ja lopetuspiste. Ohjaus tapahtuu ohjelmakierrollisella vertailulla takaisinkytkennän ja edellisen ohjelmakierron paikkatiedosta, jonka avulla voidaan laskea moottoreille syötettävän tehonmäärä. Takaisinkytkennän laskennallinen toimitus voidaan toteuttaa PID-säätimellä. PID-säädin koostuu *proportional*-, *integrative*- ja *derivative*-termeistä, joiden yksinkertainen tarkoitus on korjata asetusarvon ja oloarvon laskennallista virhettä. [1, s. 56; 2, s. 91.]

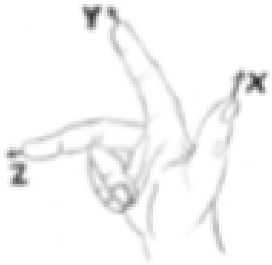


Kuva 17. Periaatteellinen säätöpiiri [1, s.56].

Todellisuudessa servo-ohjainten laskenta pitää sisällään myös muita ohjaukseen liittyviä komponentteja takaisinkytkentätiedon lisäksi. Tarvitaan esimerkiksi myös momentti- ja nopeussäätö.

5.4 Koordinaatistot

Standardin ISO 9787-1990 määrittelee teollisuusrobottien koordinaatistot. Pääsääntöisesti ne noudattavat suorakulmaisia oikeakätisiä koordinaatistoja (kuva 18).



Kuva 18. Suorakulmaisen oikeankäden muistisääntö [9].

Peruskoordinaatisto on sidottu robotin alustaan, jossa tyypillisesti Z-akseli on robotin ensimmäisen nivelen akselinsuuntaisesti ja XY-akselit alustantasolla niin, että X-akseli osoittaa työalueen keskikohtaan. Tällä tarkoitetaan sitä, että robotin alustan liikkuessa koordinaatisto seuraa perässä. Maailmakoordinaatisto on robotin ulkopuolinen koordinaatisto, jonka voi sitoa robotin työskentely-ympäristöön. Esimerkiksi laatikoiden lavauksessa koordinaatisto voidaan sitoa lavan reunaan. Työkalukoordinaatiston avulla voidaan työkalupiste muuttaa robotin työkalulaippaan sidotusta koordinaatistosta. [3, s. 20–21.]

5.5 Position ja orientaation ilmaiseminen

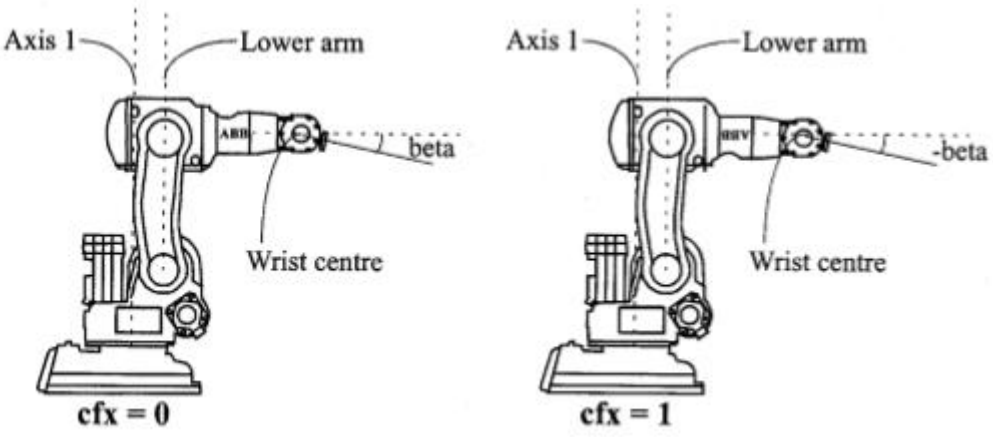
Positiolla tarkoitetaan robotin paikkatiedon esitystä X, Y ja Z valitussa koordinaatistosta. Orientaatiolla taas tarkoitetaan työkalun asentoa halutussa pisteessä [10, s. 1150, 1160].

5.6 Konfiguraatiot ja kaksoismerkitysongelma

Monissa roboteissa voi ilmetä ns. kaksoismerkitysongelma. Työkalun paikka voidaan määrittää useilla eri nivelkulmakombinaatioilla, johtuen joidenkin nivelten kyvystä pyöriä yli 360 astetta. Samaan työkalun paikkaan voidaan päätyä myös jonkin nivelen 180 asteen käännöksellä ja korjaamalla muiden nivelten asentoa. Ongelma voidaan ratkaista konfiguroimalla jokaiselle nivelelle kulma-alue, johon työkalun paikka halutaan määrittää. [3, s. 26.]

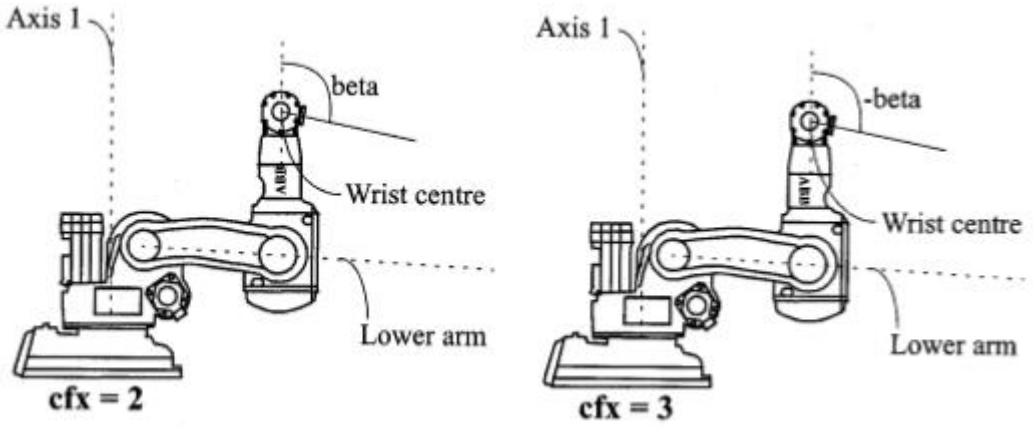
Esimerkkinä kuuden nivelen kiertyvänivelisellä robotilla samaan pisteeseen voidaan robotti konfiguroida kahdeksalla eri tavalla.

Kuvasta 19 voidaan todeta neljännen nivelen 180 asteen muutos.



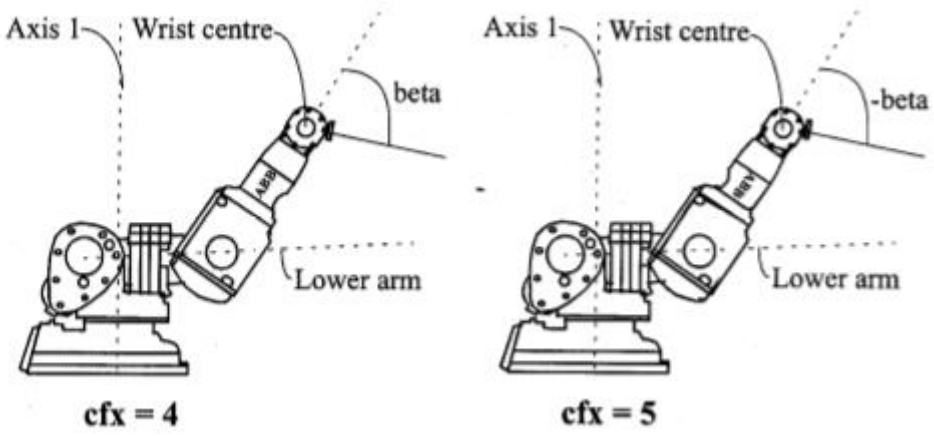
Kuva 19. Ensimmäinen ja toinen konfiguraatioasento [9].

Kuvasta 20 voidaan todeta toisen akselin muutos edellisiin konfiguraatioihin ja neljännen nivelen 180 asteen muutokset.



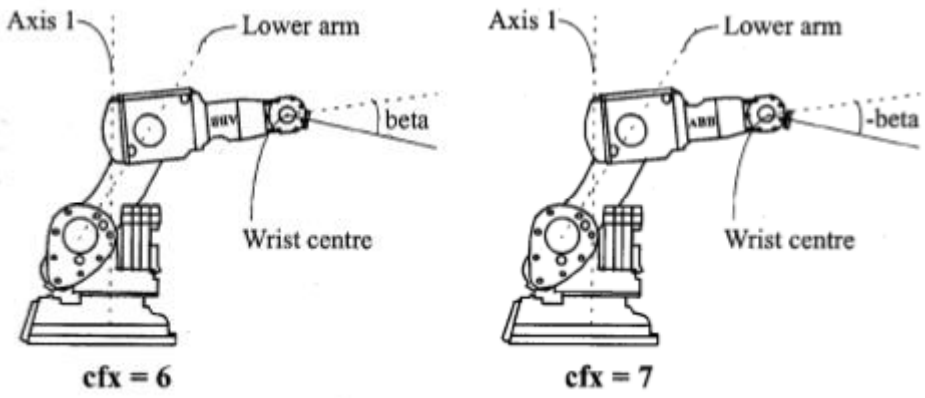
Kuva 20. Kolmas ja neljäs konfiguraatioasento [9].

Kuvasta 21 voidaan todeta ensimmäisen nivelen 180 asteen muutos edellisiin konfiguraatioihin, toisen nivelen 180 asteen muutos ja neljännen nivelen muutos.



Kuva 21. Viides ja kuudes konfiguraatioasento [9].

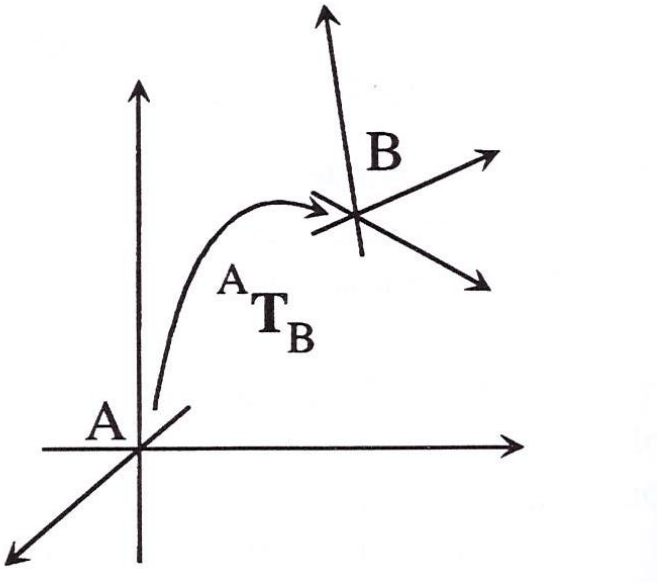
Kuvasta 22 voidaan todeta toisen, kolmannen ja neljännen nivelen muutos.



Kuva 22. Seitsemäs ja kahdeksas konfiguraatioasento [9].

5.7 Koordinaatistomuutokset

Kahden koordinaatiston välistä suhdetta voidaan esittää siirtomatriisien avulla (kuva 23). Tämä suhde vastaa esimerkiksi työkalun asemaa peruskoordinaatistossa. Koordinaatistomuutoksia voidaan ketjuttaa suorittamalla niitä peräkkäin. Niistä voidaan myös tehdä silmukka, jolloin silmukan sisäiset koordinaatistomuutokset voidaan ilmaista silmukan koordinaatistomuunnosten tai käänteismuunnosten avulla. [3, s. 22–23.]



Kuva 23. Kahden koordinaatiston välinen suhde [3, s. 22].

Esimerkkinä koordinaatistomuutoksesta otetaan koordinaatisto B, joka on kiertynyt A-koordinaatiston suhteen x-akselin ympäri 40° ja siirtynyt x-akselin suhteen 5 yksikköä sekä z-akselin suhteen 10 yksikköä. Näiden tietojen perusteella selvitetään, mikä on ${}^{A}p$, kun ${}^{B}p = [\ 2.0\ 4.0\ 1.0\ ]^{T}$.

$$ {}^{A}p = {}^{A}T_{B}\ {}^{B}p \tag{5} $$

jossa

${}^{A}p$ on pisteen koordinaatit koordinaatistossa A

${}^{A}T_{B}$ on homogeeninen siirtomatriisi

${}^{B}p$ on pisteen koordinaatit koordinaatistossa B

Homogeeninen siirtomatriisi johdetaan lausekkeesta, jossa huomioidaan liikkeet jokaisen akselin suhteen:

$$ {}^{A}T_{B} = \begin{bmatrix} x_x & y_x & z_x & p_x \\ x_y & y_y & z_y & p_y \\ x_z & y_z & z_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \vec{x} & \vec{y} & \vec{z} & \vec{p} \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (6) $$

Esimerkkitapauksessa lopullinen siirtomatriisi on

$$ {}^{A}T_{B} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7) $$

Sijoittamalla arvot siirtomatriisiin saadaan selville pisteen koordinaatit kaavan 5 mukaisesti.

$$ {}^{A}p = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & \cos 40 & -\sin 40 & 0 \\ 0 & \sin 40 & \cos 40 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix} [2.0 \quad 4.0 \quad 1.0]^T = \begin{bmatrix} 7 \\ -3{,}413 \\ 12{,}313 \end{bmatrix} $$

Oikean tuloksen saamiseksi on huomioitava, että 4*4-matriisin kertominen 1*3-matriisilla ei onnistu ilman skaalauskerrointa, joka robotiikassa on usein 1.

5.8 Aistimet

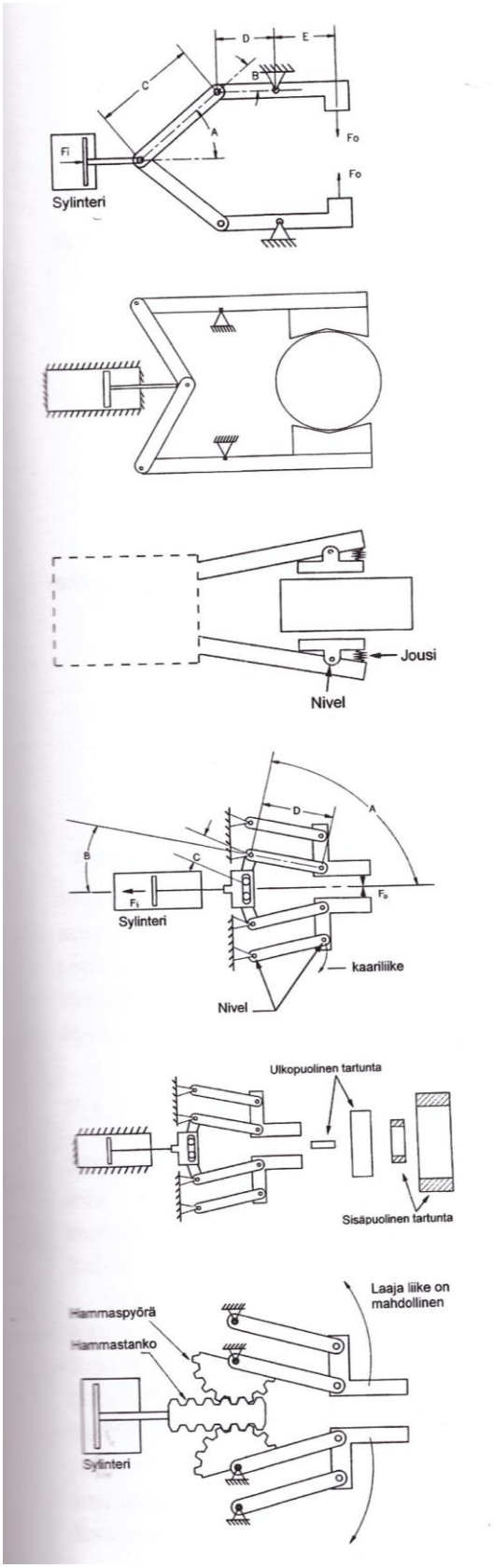
Aistimilla tarkoitetaan tuntoelintä, esimerkiksi analoginen anturi, joka antaa lisätietoa robotille sen työalueen tai ympäristöön liittyvissä asioissa. Aistimien avulla voidaan robotin paikkatietoa muuttaa kesken työnsuoritusta, jolloin voidaan välttää mahdolliset törmäykset tai väärät robotin liikkeet työstökohteeseen nähden. Aistimien avulla saadaan toiminnallisesti robotista hieman älykkäämpi. [6, s. 319–320.]

6 Tarraimet ja työkalut

6.1 Yleistä

Jotta robotti pystyisi työstämään haluttua kappaletta, se tarvitsee tarraimen tai mahdollisesti muun työkalun. Tällaisia työkaluja ovat muun muassa ruuvaustyökalut, valukauhat, polttoleikkaimet, jyrsimet, hiomalaitteet, liimaus-, sauma- ja ruiskumaalaussuuttimet sekä erilaiset hitsauspäät. Työkalua valittaessa pyritään huomioimaan työstettävien kappaleiden mahdolliset muotopoikkeamat. Muotopoikkeaman aiheuttamia ongelmia voidaan vähentää valitsemalla joustavia työkaluja. [3, s. 76–77.] Työkaluista merkittävämmässä roolissa robotiikassa ovat tarraimet. Tarraimet voidaan jakaa ryhmiin niiden tyypin ja tartuntatavan mukaan [3, s. 60].

Robotiikassa voidaan simuloida ihmisen sormien liikkeitä mekaanisten tarrainten avulla. Mekaaniset tarraimet voivat olla nivelmekanismisia, hammaspyörä ja hammastankoisia, epäkeskoja, ruuveja, vaijeriväkipyöriä tai sekalaisia kinemaattiselta rakenteeltaan [3, s. 60]. Mekaaniset tarraimet rakentuvat sormista, kynsistä, toimilaitteesta sekä mekanismista (kuva 24) [3, s. 63].



Kuva 24. Esimerkkejä tarrainmekanismeistä [3, s.61].

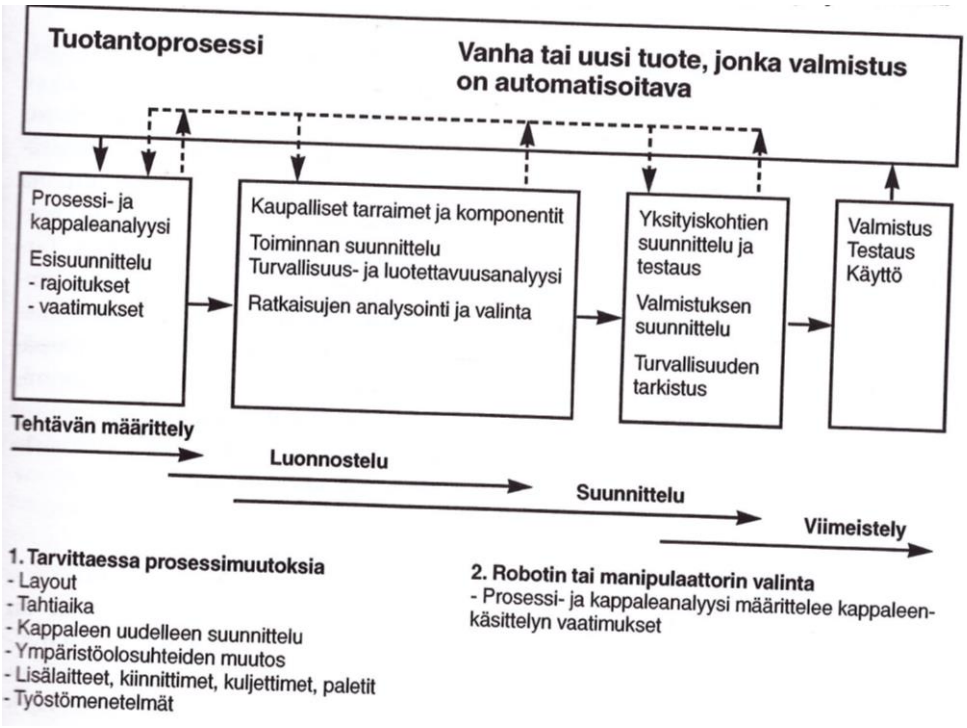
Mikäli työstettävä kappale ei sovellu mekaaniselle tarraimelle, voidaan käyttää alipaineeseen perustuvia tarraimia. Tällöin kappaleeseen tartutaan kumisilla tai muovisilla imukupeilla, jotka eivät aiheuta kappaleen pinnalle naarmuja. Paine-eroa ja imupinta-alaa kasvattamalla imutehoa voidaan nostaa tarvittavalle tasolle. Alipaine synnytetään yleensä alipainepumpulla tai venturilla/ejektorilla. [3, s. 63.]

Magneettisille kappaleille voidaan käyttää magneettitarraimia. Kappaleesta on löydyttävä sopivan kokoinen tasainen alue, johon voidaan tarttua. Magneetintarraimen toimivuuteen vaikuttavat kappaleen materiaali, pinnanlaatu, muoto, ilmarako sekä magneetin lämpötila. [3, s. 64.]

Joissain tilanteissa normaaleista tarraimista ei löydy ratkaisua ja käyttöön otetaan erikoistarraimia. Tällaisessa tapauksessa esimerkiksi lamelli tai muu muotoutuva elementti toimii tarraimena. [3, s. 64.]

6.2 Valinta ja suunnittelu

Kun lähdetään miettimään tarraimen valintaa, on pyrittävä välttämään ihmisen toimintojen matkimista. Valintaa tehtäessä on myös otettava huomioon koko prosessi, eikä vain siihen liittyvää tarrainta (kuva 25). Tällä tavalla voidaan saavuttaa optimaalinen tulos valinnassa. Toiminnan luotettavuus taataan huomioimalla vielä tarraimen luoksepäästävyys ja kunnossapitonäkökohdat sekä toleranssianalyysi, tartuntamenetelmä ja robotin hyötykuorma. [3, s. 65.]



Kuva 25. Tarraimen suunnittelukonsepti [3, s.65].

Jotta varsinaiseen suunnitteluvaiheeseen (kuva 25) päästään, on ensin tehtävä analyysi käsiteltävästä kappaleesta ja prosessista, johon sen käsittely kuuluu. Joskus on jopa tarpeen muuttaa prosessia ja kappaletta, jotta ne toimisivat paremmin, kun prosessiin otetaan mukaan robotti. Tässä vaiheessa on myös syytä pitää taloudelliset asiat koko ajan esillä. [3, s. 66.]

Tehtyjen analyysien perusteella päästään esisuunnitteluvaiheeseen, jossa luonnostellaan mahdollisuuksia tulevaksi tarraimeksi [3, s. 66]. Tämä vaihe aloitetaan perehtymällä itse tartuntatapahtumaan, joka perustuu joko puristusvoimaan tai kappaleen tarjoamiin muotoihin. Puristusvoima on mietittävä sopivaksi, jotta kappale ei putoa, mutta ei myöskään vaurioidu liian suuren puristusvoiman johdosta. [3, s. 67.] Joissain tapauksissa myös alipaineisen tai magneettisen tartuntatavan käyttämistä prosessissa voidaan harkita. Tarraimen mekaaninen rakenne on syytä tehdä mahdollisimman yksinkertaiseksi toimintahäiriöiden vähentämiseksi. [3, s. 68.]

Tarraimen ja kappaleen välinen kitkavoima on pintojen välisen kitkakertoimen ja kappaleeseen kohdistuvan puristusvoiman tulo. Tuon kitkavoiman on oltava suurempi kuin gravitaation ja robotin liikkeiden aiheuttamat voimat, jotta kappale pysyy tarraimessa. Mitoitusta ei kannata tehdä kovin tiukasti tätä rajaa ajatellen, vaan mekanismille ja toi-

milaitteelle on syytä antaa varmuuskerroin, joka on noin 1,5–2. [3, s. 68] Mitoituksen kanssa täytyy kuitenkin olla tarkkana. Liian suuret varmuuskertoimet johtavat painaviin rakenteisiin [3, s. 69].

Toimilaitteen on oltava helposti ohjattavissa, tarpeeksi pieni sekä helposti liitettävissä mekanismiin. Näillä toimilaitteilla voidaan sitten ohjata tarrainvoimaa ja tarraimen avautumaa. Käyttölajeina ovat hydrauliikka, jolla voidaan saada suuri tartuntavoima, servopneumatiikka, jonka käyttö on lisääntynyt, sekä sähköenergia. [3, s. 69.]

Tarraimista tehdään älykkäitä lisäämällä niihin antureita, jotka kertovat tarraimen toiminnasta ja ympäristöstä [3, s. 69]. Antureiden vähimmäisvaatimus on kertoa tehtävän epäonnistumisesta. Ne voivat mitata tarraimen sisäistä tai ulkoista tilaa. Sisäisellä tilalla tarkoitetaan itse tarraimen toiminnan tilaa ja ulkoisella tilalla käsiteltäviä kappaleita ja toimintaympäristöä. [3, s. 70.] Kuvassa 26 näytetään millaisia erilaisia tehtäviä tarraimen antureilla on eri tilanteissa.

	Tartuntavaihe	**Siirtovaihe**	**Prosessivaihe**
Tarraimen tarkkailu	-tarraimen tila -tartuntavoima -törmäys	-tartuntavoima -törmäys	-tartuntavoima -törmäys
Kappaleen tarkkailu	-läsnäolo -tunnistus -asema ja asento -kappaleen tarkistus -tartunta	-luiston valvonta -kappaleen tarkistus -törmäys	-luiston valvonta -törmäys
Prosessin tarkkailu			-läsnäolo -tunnistus -asema ja asento -ympäristön tarkistus -asennusvoima -asennussyvyys

Kuva 26. Tarraimen tehtäviä [3, s.70].

Esisuunnittelun tietoja viedään eteenpäin kehittelyvaiheessa. Hyvä lähtökohta on valita jokin vakiotarrain ja muokata sitä haluttuun suuntaan. Tällä tavalla saadaan niin taloudellista kuin toiminnallistakin hyötyä. Kehitysvaiheessa ryhdytään kiinnittämään huomiota myös järjestelmän luotettavuuteen sekä turvallisuuteen. Tarrain ei esimerkiksi saa päästää kappaletta irti, vaikka sähkönsyöttö laitteelle katkeaisi. Luotettavuuteen lähdetään vaikuttamaan hyvin valmistellun vaatimusmäärittelyn avulla. On myös muis-

tettava, että tarraimeen liittyviä seikkoja ei voi suunnitella ottamatta huomioon myös robottia. Robotti ja tarrain muodostavat kokonaisuuden, joka on pidettävä suunnittelussa mukana koko ajan. [3, s. 70.]

6.3 Viimeistely

Tarraimen viimeistelyssä on pidettävä huolta, että sille aiemmin asetetut vaatimukset pystytään täyttämään. Tarraimesta on pyrittävä saamaan kevyt, ja se onnistuu käyttämällä muovia ja alumiinia. Keveyttä saadaan lisättyä myös tekemällä tarraimeen kevennysreikiä. Keveyden lisäksi tarraimesta on pyrittävä saamaan puristusvoimainen, jäykkä ja helposti huollettava. Tarraimen anturoinnissa on pyrittävä suojaamaan anturit ja sen johdot mahdollisimman hyvin. Kaiken suunnittelun jälkeen seuraa tarraimen valmistus. Valmis tarrain testataan kunnolla ennen käyttöönottoa. [3, s. 71–72.]

7 Robotin ohjelmointi

Rapid-ohjelmointi on ABB:n kehittämä ohjelmointikieli erityisesti roboteille. Tässä luvussa tullaan käymään peruskomennot ja muuttujat RobotWare 5.0:lla, joita harjoituksessa tullaan käyttämään.

Robotin ohjelmoinnin perustavoite on robotin työkalun liikkeiden tai haluttujen liikeratojen ohjaus. Toinen merkittävä tehtävä on olla vuorovaikutuksessa työalueen tai ympäristön kanssa erilaisin kommunikaatiomenetelmin. Tämä saattaa tarkoittaa esimerkiksi kommunikointia logiikan kanssa. [6, s. 313.]

Robotille pitää määritellä ainakin seuraavat komennot pisteestä pisteeseen liikkumiseen (kuva 27):

- liiketapa
- piste, johon liikutaan koordinaatistossa
- nopeusohje, jolla pisteeseen liikutaan

- paikoitustarkkuus pisteeseen

- työkalupiste.

```
MoveL p10, v1000, fine, tool0;
```

Kuva 27. Kuva eräästä robotin liikekäskystä [11, s. 23].

Perusliiketapoja on kolme:

- MoveL, tarkoittaa robotin siirtymistä lineaarisesti pisteestä pisteeseen [11, s. 25].

- MoveJ, ajaa robotin suorinta tietä pisteestä pisteeseen. Komentoa ei suositella käytettävän, jos robotin kanta (BASE) on haluttujen pisteiden välissä, jolloin syntyy törmäysvaara [11, s. 30].

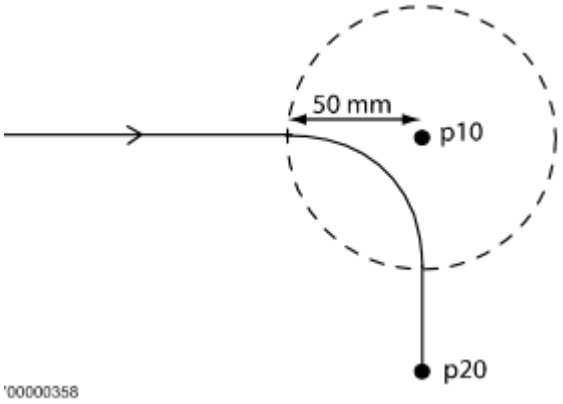
- MoveC, liikuttaa robottia kaarenmuotoisesti pisteestä pisteeseen [11, s. 50].

Rapid-ohjelmointikieli käsittää myös lukuisia muita liiketapoja esimerkiksi liikkeen ja samanaikaisen digitaalilähdön ohjauksen.

Koordinaatistossa sijaitseva piste, esimerkiksi ”P10”, voidaan nimetä uudestaan. Online-ohjelmoinnissa piste voidaan määrittää ajamalla robotti haluttuun kohtaa ja tallentaa piste. Ohjelma määrittää automaattisesti nivelten positiot, tällä tarkoitetaan koordinaatiston X-, Y- ja Z-pisteitä työskentelyavaruudessa.

Nopeusohjeella tarkoitetaan robotin liikkumisnopeutta mm/s, esimerkkitapauksessa v1000 nopeus on 1000 mm/s [11, s. 25].

Paikoitustarkkuudella tarkoitetaan haluttua tarkkuutta, jolla robotti pysähtyy pisteeseen. Esimerkkitapauksessa (kuva 28) ”fine” tarkoittaa sitä, että robotti ajaa juuri siihen koordinaattiin, johon se on määritelty kulkemaan. Robotin liike voidaan myös määritellä oikaisemaan ennen haluttua pistettä esimerkiksi komennolla z50. [11, s. 26.]



Kuva 28. Paikoitustarkkuuden vaikutus arvolla z50 [11, s. 26].

Työkalupiste tool0 osoittaa robotin työkalulaipan keskipisteeseen. Robotin sijainti koordinaatistossa ja liikkeet ovat aina sidoksissa työkalupisteeseen. Työkalupisteitä voidaan myös itse määritellä työkalun mukaan. [10, s. 121.]

RAPID-ohjelmointikielessä on myös kolme päädatatyyppiä. Numeeriset arvot voidaan esittää kokonaislukuina tai desimaalilukuina, tekstiä voidaan kirjoittaa merkkijonojen avulla ja totuusarvoilla tosi- tai epätosi. Lisäksi on ennalta määriteltyjä datatyyppejä, esimerkiksi nopeusdata. [11, s. 14.]

Datatyyppejä voidaan määritellä kolmelle muuttujatyypille, perusmuuttuja, pysyvä muuttuja ja vakiomuuttuja (taulukko 5). Pääsääntöisesti muuttujat pitää ensin määrittää ja alustaa ohjelman alussa. Perusmuuttaja pysyy ohjelman muistissa hetkellisesti ohjelmakierron aikana, mutta ohjelmasta poistuttaessa se ottaa käyttöönsä alustusarvon. Pysyvää muuttujaa voidaan taas muuttaa ohjelmassa ja sen alustusarvoa voidaan muokata. Pysyvää muuttujaa voidaan soveltaa esimerkiksi laskennallisissa tehtävissä. Vakiomuuttuja säilyttää arvonsa samana eikä sitä voi muokata ohjelmankierrossa. Näin se sopii hyvin esimerkiksi piin vakioarvoksi. [11, s. 14–16.]

Taulukko 5. RAPID-kielen muuttujat.

Muuttujatyyppi	Rapid- lyhenne
Perusmuuttuja	VAR
Pysyvä muuttuja	PERS
Vakiomuuttuja	CONST

Muuttujien muokkaaminen voidaan toteuttaa RAPID-ohjelmointikielessä olevien numeeristen, vertailu ja merkkijono operaatioiden avulla (taulukko 6) [11, s. 17].

Taulukko 6. RAPID-operaatiot

RAPID-OPERAATIOT					
Numeeriset		Vertailu		Merkkijonot	
+	summa	=	yhtä suuri	+	summa
-	erotus	<	pienempi kuin		
*	kertolasku	>	suurempi kuin		
/	jakolasku	<=	pienempi tai yhtä suuri kuin		
		>=	suurempi tai yhtä suuri kuin		
		<>	erisuuri		

Ohjelmakierron hallintaa voidaan toteuttaa esimerkiksi käyttämällä loogisia komentoja. Muutamia komentoja ovat IF, ELSE, ELSEIF ja FOR. Käytännössä RAPID-ohjelmaa suoritetaan ulkopuolisten signaalien avulla. Signaalit voivat olla esimerkiksi digitaalinen tosi- tai epätosi-ehto, analogiaviesti tai digitaalinen viestikehys. [11, s. 18, 19, 33.]

8 Robotin liitynnät

Robotti sovelluksissa robotin ohjaamiseen voidaan käyttää esimerkiksi logiikoita (Programmable logic controller). Logiikka on uudelleen ohjelmoitavissa oleva digitaalinen tietokone. Logiikan perusajatuksena on loogisten operaatioiden ja aritmeettisten laskusuoritusten toteuttaminen mikroprosessorien avulla.

Loogisia operaatioita ja laskusuorituksia voidaan suorittaa logiikan binääristen kaksitilaisten tulojen avulla esimerkiksi sulkeutuvilla koskettimilla, analogisilla viesteillä esimerkiksi paineenmittaus 4-20 mA tai logiikan sisäisillä apumuuttujilla.

Logiikan kaksitilaisilla tai analogisilla lähdöillä voidaan ohjata esimerkiksi toimilaitteita tai robotteja. Lähdöt toimivat logiikan suorittamien toimintojen jälkeen ehtojen toteutuessa. [6, s. 249, 254.]

9 Robotin käyttäminen

Robotin käyttämistä voidaan hallita esimerkiksi kahdella eri tavalla, joko suoraan käsiohjaimen avulla online-tilassa tai robottiohjelmistolla offline-tilassa.

Online-tilalla tarkoitetaan ohjelmointia tai ohjelman muuttamista suoraan robottia hyväksi käyttäen. Usein keskeyttää tuotannon ja suurten kokonaisuuksien luonti vie tuotantoaikaa.

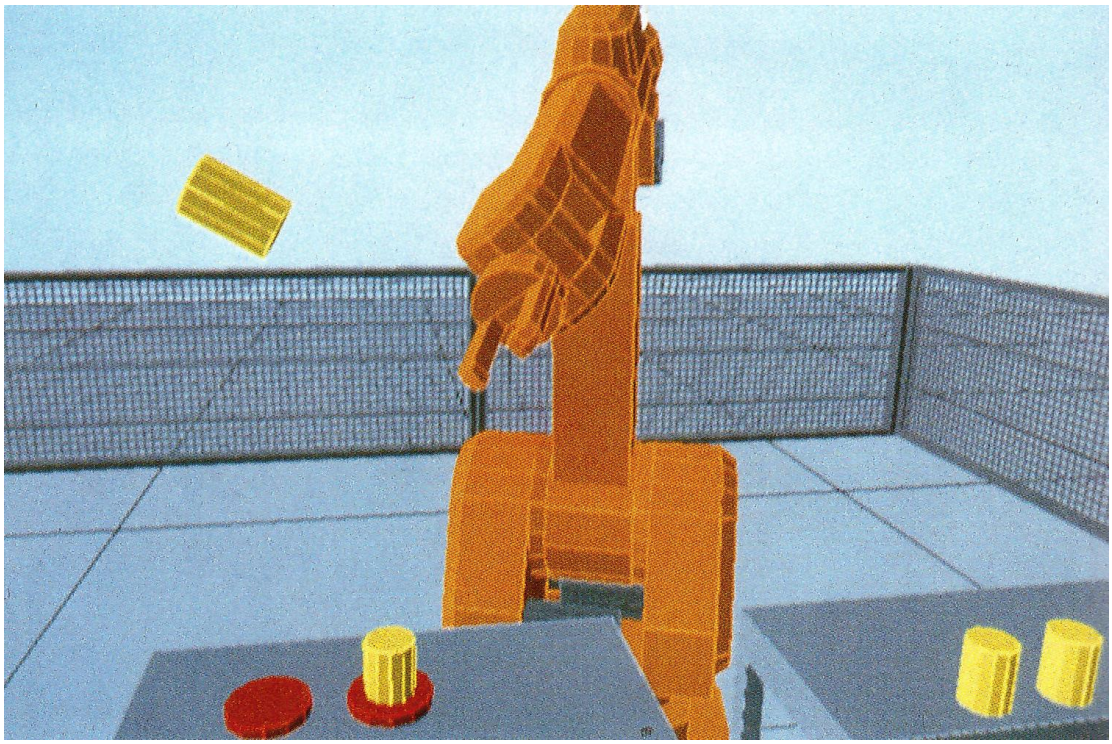
Offline-tilalla tarkoitetaan ohjelman suunnittelua ja rakentamista robotille soveltuvan ohjelmiston avulla. Etuna on, että tuotantoa ei tarvitse pysäyttää. [6, s. 316.]

Esimerkkinä ABB:n RobotStudio on PC-pohjainen ohjelmisto. RobotStudiossa voidaan haluttu prosessi simuloida ja valmistella valmiiksi ennen varsinaista käyttöönottoa. Myös laajojen ohjelma kokonaisuuksien hallinta on helpompaa.

10 Robottiturvallisuus

10.1 Riskitekijät

Teollisuusrobotin käyttöönottoa suunniteltaessa turvallisuus on otettava huomioon. Vaikka robotille ilmoitettaisiin käsittelykyvyksi 10 kg, se pystyy todellisuudessa nostamaan jopa yli 100 kg kuorman. Näin suuri voima voi aiheuttaa vakavan tapaturman. Tapaturman riski kasvaa, mikäli robotin käyttämä työkalu on ihmiselle vaarallinen. Robotin ja siinä kiinni olevan työkalun tai tarttujan lisäksi vaaraa kasvattaa mahdollisuus siitä, että tarttujassa kiinni oleva kappale irtoaa ja osuu työntekijään (kuva 29). [12, s. 13–14.]



Kuva 29. Kappale irtoaa robotin tarraimesta [12, s. 14].

Useilla roboteilla liikerata on laaja. Robotti saattaa olla työntekijästä melko kaukana, ja silti työntekijä on vaara-alueella. Robotin työskentelyssä ei voida täysin luottaa siihen, että se tekisi samaa liikerataa toistuvasti. Jo yhden ohjelmoidun pisteen poistaminen muuttaa liikerataa, kun robotti pyrkii löytämään lyhimmän reitin ohjelmoitujen pisteiden välillä. Jos työntekijä on lähellä robotti ja muistelee sen liikeratoja, hän on vaarassa, mikäli joku on muokannut ohjelmaa vähääkään. [12, s. 15–16.]

Robotit voivat liikkua jopa 3–4 m/s. Robotit eivät ole myöskään kovin nopeita pysähtymään. Vanhalla robotilla pysähtymismatka on mitattu jopa metrin mittaiseksi. Ihmisen reaktionopeus ei riitä väistämään nopeaa robottia, ja sen pitkä pysähtymismatka tekee tilanteesta entistä vaarallisemman. [12, s. 17–18.]

Robottiturvallisuutta ajateltaessa pyritään tässä luvussa mainittujen riskitekijöiden vaikutus minimoimaan. Sekään ei aina riitä, sillä työntekijät eivät halua pysäyttää koneita, mikäli niiden uudelleenkäynnistämisestä koituu prosessille edes hetkellistä haittaa. Pysäytystoimintojen kehityksellä on pyrittävä vaikuttamaan työntekijöiden halukkuuteen niiden käyttöön. [12, s. 21.]

10.2 Riskejä sisältävät työtehtävät

Teollisuusrobotin käyttöön liittyy useita työtehtäviä, jotka sisältävät turvallisuusriskejä. Järjestelmän asennus ja käyntiinajo sisältävät sekä riskejä että mahdollisuuksia. Nämä liittyvät lähinnä toimittajan tarjoamaan perehdytykseen ja työntekijöiden saamaan turvallisuuskoulutukseen. [12, s. 8.] Mikäli nämä asiat hoidetaan hyvin, turvallisuusasioissa ollaan jo pitkällä, mutta niiden laiminlyönti on suuri turvallisuusriski.

Robotin ohjelmointia suoritetaan harvemmin offline-tilassa erillisellä tietokoneella. Yleensä ohjelmointi tapahtuu kannettavalla ohjelmointipaneelilla, jonka avulla robotti opetetaan liikkumaan pisteestä toiseen. Tällä tavalla ohjelmoitaessa ollaan yleensä lähellä robottia. Tästä syystä robotin tulisi liikkua huomattavasti tuotantonopeuttaan hitaammin, maksimissaan 250 mm/s. Robotin liikkumista rajoitetaan myös toiminnolla, jonka perusteella robotti ei voi liikkua kuin ohjauspaneelin painikkeita painettaessa. Sormen nostaminen painikkeelta pysäyttää robotin. Viimeisenä varotoimena on ns. sallintakytkin, joka tunnetaan myös nimellä kuolleen miehen kytkin. Kun tuo kytkin on pohjassa, ohjelmointia voidaan tehdä. Kun ote irrotetaan, robotti pysähtyy. [12, s. 8–9.] Jos robotti löisi ohjelmoijaa ja tämän ote irtoisi kytkimeltä, robotti pysähtyisi eikä pystyisi tekemään enempää vahinkoa.

Tuotantokäytössä ihmisen harvemmin tarvitsee mennä robotin työskentelyalueelle. Tällöin robotti ei aiheuta kenellekään vaaraa. On kuitenkin prosesseja, joita täytyy tarkkailla erittäin läheltä niiden laadunvarmistamisen takia. Tällaiset prosessit ovat turvallisuuden kannalta hankalia. [12, s. 12.]

Häiriönpoiston ja huollon aikana ollaan lähes aina vaara-alueella robotin suhteen. Työntekijöiden turvallisuus vaarantuu, mikäli robotti ei ole pysäytystilassa. Näin saattaa käydä, mikäli työntekijät pyrkivät tapahtumaa nopeuttaakseen toimimaan prosessia pysäyttämättä. Turvaratkaisujen sivuuttaminen aiheuttavat suuren osan robotteihin liittyvistä onnettomuuksista. [12, s. 12.]

10.3 Tuotannon turvallisuustekniikka

Turvallisuuden suunnittelun lähtökohtana on pidettävä tavoitetta suunnitella laitteet mahdollisimman vaarattomiksi. Ne vaarat mitä kuitenkin jää jäljelle, poistetaan erilaisilla

suoja- ja turvalaitteilla. Tämänkin jälkeen joitain vaaroja saattaa jäädä jäljelle. Työturvallisuuslaki määrää, että näistä vaaroista on varoitettava. Näiden lisäksi tulee käyttää mm. hätäpysäyttimiä. [12, s. 21.] Hätäpysäyttimillä vaaratilanteen nähnyt työtoveri voi pysäyttää robotin.

Jokainen prosessi jossa robotteja käytetään, on erilainen, ja robotin toimintaympäristö vaihtelee tapauskohtaisesti. Tästä syystä täysin yksityiskohtaisten turvallisuusohjeiden antaminen on lähes mahdotonta. Yleisperiaatteita on sovellettava jokaisessa tilanteessa tilanteen vaatimalla tavalla. [12, s. 21.]

Turvallisuuden suunnittelua hankaloittaa entisestään se, että robotit työskentelevät harvemmin yksin. Ne on liitetty muuhun prosessiin. Robotille tehdyt turvallisuusratkaisut vaikuttavat siis muihinkin laitteisiin ja koko prosessiin. Koko prosessin ja siihen liittyvien laitteiden turvallisuus on kokonaisratkaisu, joka on hankala suunnitella, mutta joka on myös tuotannon keskeytyksettömän toimimisen ja turvallisuuden kannalta saatava kuntoon. [12, s. 22.]

10.4 Turvallisuusratkaisut

10.4.1 Pysäytystoiminnot

Robotti voidaan pysäyttää eri tavoin riippuen tarpeesta. Tuotantopysäytys on prosessin kannalta paras vaihtoehto. Siinä robotti siirtyy pysäytystilaan siinä vaiheessa, kun se on suorittanut työkierron loppuun. [12, s. 22.] Robotti pysähtyy siis prosessin kannalta sopivaan kohtaan, ja se on helppo käynnistää uudestaan.

Turvapysäytys tapahtuu, kun työntekijä astuu vaara-alueelle. Energiansyöttö robottiin katkaistaan ja kaikki vaaralliset toiminnot estetään. Pysäytyksessä käytetään usein releteknііkkaa sen luotettavuuden takia. [12, s. 23.]

Vakava vaaratilanne saatetaan joutua estämään käyttämällä hätäpysäytystä (kuva 30). Koko laiteyksikön energiansyöttö katkaistaan kuitenkin niin, että tällä ei aiheuteta muuta vaaraa. Hätäpysäytyksen jälkeen prosessin uudelleenkäynnistäminen voi olla hankalaa. [12, s. 24.]

Kuva 30. Hätäseis-painike [12, s. 28].

10.4.2 Aitaratkaisut

Prosessin aitaaminen on hyvä keino turvallisuuden parantamiseen. Kun työntekijät pidetään erossa järjestelmästä, he eivät joudu vaaraan. Aitaratkaisusta on tehtävä korkea, ja sen ylittämisestä sekä läpi kurottamisesta on tehtävä hankalaa. [12, s. 25.]

Aitauksen kulkuaukossa oleva tunnistin pysäyttää robotin sen työskentelyalueelle mentäessä. Kulkuaukon voi varustaa portilla, jonka avaaminen aiheuttaa turvapysäytyksen käyttäen rajakytkintä, valokennoa, valoverhoa tai tuntomattoa. Mekaanisen turvakytkimen etu muihin vaihtoehtoihin nähden on se, että sitä ei tarvitse kahdentaa. [12, s. 25.]

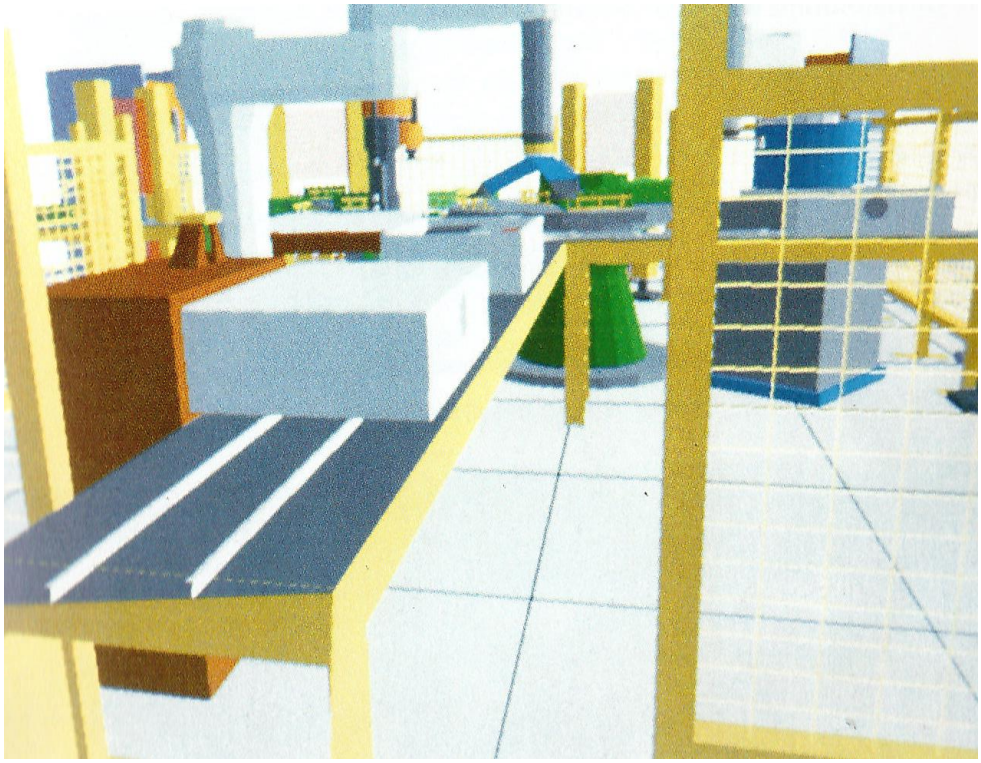
10.4.3 Materiaalivirtojen hallinta

Vaikka työntekijää ei saa päästää robotin työalueelle sen ollessa toiminnassa, sinne on kuitenkin saata materiaalit, joita robotti työstää. Materiaalien liikuttamiseen tarvitaan siis omat turvajärjestelynsä. [12, s. 26.]

Ihmisen ja robotin välille voidaan asentaa kääntöpöytä. Työntekijä suorittaa oman työosuutensa ja kun hän on sen kuitannut, robotti voi kääntää pöydän ja suorittaa omansa. Kuittaus suoritetaan pöydän kääntöalueen ulkopuolella. [12, s. 26.]

Portteja ja veräjiä voidaan käyttää siten, että niiden ollessa auki robotti ei tule alueelle [12, s. 27]. Työntekijä voi työskennellä alueella turvassa ja poistuessaan sulkee portin ja kuittaa työvaiheen tehdyksi. Tällöin robotti tule suorittamaan omaa työvaihettaan.

Materiaaleja robotille tuovassa kuljetinjärjestelmässä on omat ongelmansa (kuva 31). Järjestelmän on pystyttävä varmistamaan, että työntekijä ei tule robotin aiheuttamalle vaara-alueelle kuljetinjärjestelmää käyttäen. Ongelmakohtana on myös kuljettimen ja aidan väliin jäävä tila, josta työntekijä saattaa pyrkiä sisään. Jos aukkoa valvomassa on antureita, jotka eivät tarvitse kosketusta toimiakseen, ne saattavat luulla työntekijää materiaaliksi ja turvapysäytys jää tekemättä. Koska kuljetinjärjestelmät ovat hyvin erilaisia, niiden turvaratkaisut ovat tapauskohtaisia. [12, s. 27.]



Kuva 31. Kuljettimen aukko pitää huomioida turvaratkaisuissa [12, s. 27].

10.4.4 Anturitekniikka

Ihmisen turvaaminen robotilta aidalla ei aina ole paras mahdollinen ratkaisu. Jotta tuotanto voidaan pitää joustavana, turvaratkaisuja kannattaa tehdä anturitekniikan avulla. Ihmistä anturit eivät voi pitää poissa robotin läheisyydeltä, joten niiden tehtäväksi jää robotin vaarallisten liikkeiden estäminen silloin, kun ihminen on sen vaara-alueella. Tällaisten turvatoimintojen on täytettävä korkeat laatuvaatimukset. [12, s. 29.] Osa antureista, joita mainostetaan turva-antureina, eivät sovellu kuitenkaan tämäntyyppisiin tehtäviin [12, s. 30].

Turvaratkaisut ovat usein erilaisten anturitekniikkavaihtoehtojen sovellutuksia. Antureilla voidaan vahtia ihmisten menoa vaara-alueelle ja estää vaaralliset toiminnot näin tapahtuessa. Antureilla voidaan vahtia myös koko vaara-aluetta, jolloin ihmisen liikkeistä saadaan havainto myös alueen sisällä. Jos halutaan valvoa tarkasti jotain tiettyä vaara-aluetta, voidaan anturit asentaa liikkuvaan koneeseen kiinni. [12, s. 29.]

Tällaiset anturiratkaisut sopivat hyvin tilanteisiin, joissa toiminta-alueella joudutaan prosessin toimivuuden takia käymään usein. Antureilla alue voidaan jakaa osiin, ja näin se toimii joustavammin kuin perinteinen aitaratkaisu. [12, s. 29–30.]

Turvatarkoituksiin on olemassa erilaisia antureita. Kosketusanturi voi esimerkiksi pysäyttää robotin, kun ihminen astuu tietylle alueelle. Tällaisia antureita mietittäessä on huomioitava robotin pysähtymismatka ja -aika. Tästä syystä anturia ei voi sijoittaa esimerkiksi robotin työkaluun. [12, s. 30.]

Optisessa anturissa valonsäde lähetetään kohti vastaanotinta. Jos vastaanotin ei tunnista sädettä, turvatoiminto voidaan laukaista. Säteen katkeamisen oletetaan johtuvan siitä, että ihminen pyrkii vaara-alueelle. Erilaisiin tarkoituksiin on olemassa valokennoja, jotka ovat yksisäteisiä optisia antureita, sekä valoverhoja, joissa on useampia säteitä. [12, s. 31.]

On olemassa myös muun tyyppisiä läsnäoloantureita. Näitä ovat mikroaalto- ja ultraäänianturit sekä passiiviset infrapunailmaisimet. Näistä ei kuitenkaan ole ensisijaisiksi turva-antureiksi, mutta ne voivat tuoda lisäturvaa haluttaessa. [12, s. 33.]

11 Käytännön harjoituksen toteutus

Käytännön harjoitus aloitettiin siitä ajatuksesta, että koululle voitaisiin tehdä laboratorioharjoitus, jonka tyylistä ei tällä hetkellä ole olemassa. Robotin käyttöä harjoiteltaessa huomattiin, että pystyttiin tekemään paljon muutakin. Uudeksi tavoitteeksi asetettiin lastausohjelman ohjelmointi koulun IRB120-robotille.

Harjoituksessa käytettiin ABB:n robottia. Robotin komponentit:

- ohjausjärjestelmä IRC5 M2004
- manipulaattori IRB120 (liite 1)
- käsi-ohjain FlexPendant.

Harjoitustyön ensimmäisessä vaiheessa ohjelmoitiin ABB:n IRB120-robotille varastointiohjelma. Ohjelmassa käyttäjä saa valita, käsitelläänkö isoja vai pieniä kappaleita. Tämän jälkeen käyttäjän tehtäväksi jää määritellä, suoritetaanko varastonkasaus vai -purku. Kummassakin tapauksessa käyttäjä vielä määrittelee varaston koon vaaka-, pysty- ja korkeusakseleilla. Käyttäjän ohjeet saatuaan robotti suorittaa halutun varastointitehtävän.

Harjoitustyön toisessa vaiheessa tehtäväksi saatiin jatkuvatoimisen robottiohjelman luominen messutarkoituksiin. Ratkaisuksi tehtiin kahdeksan kappaleen kasaus ja purku rutiini. Kasausvaiheessa kappaleet ovat kahdessa neljän kappaleen rivissä, jonka jälkeen kappaleet kasataan kaksi kertaa pysty-, vaaka- ja korkeusakselin mukaisesti. Purkuvaiheessa toiminta on päinvastainen.

Toisen vaiheen toteutuksessa päätettiin hyödyntää ensimmäisen vaiheen loogisia operaatioita. Ohjelman alkuun luotiin vaihtoehto, jossa voidaan valita ensimmäisen- tai toisen vaiheen tehtävä.

Robotin työkaluna toimi tarttuja (kuva 32), jolla kappaleita siirrettiin paikasta toiseen. Tarttuja oli paineilmalla toimiva sylinteri, jota ohjasi pneumatiikkaventtiili. Venttiiliä ohjattiin robotin ohjausjärjestelmän lähdöillä kahdella 24 VDC-kelalla.

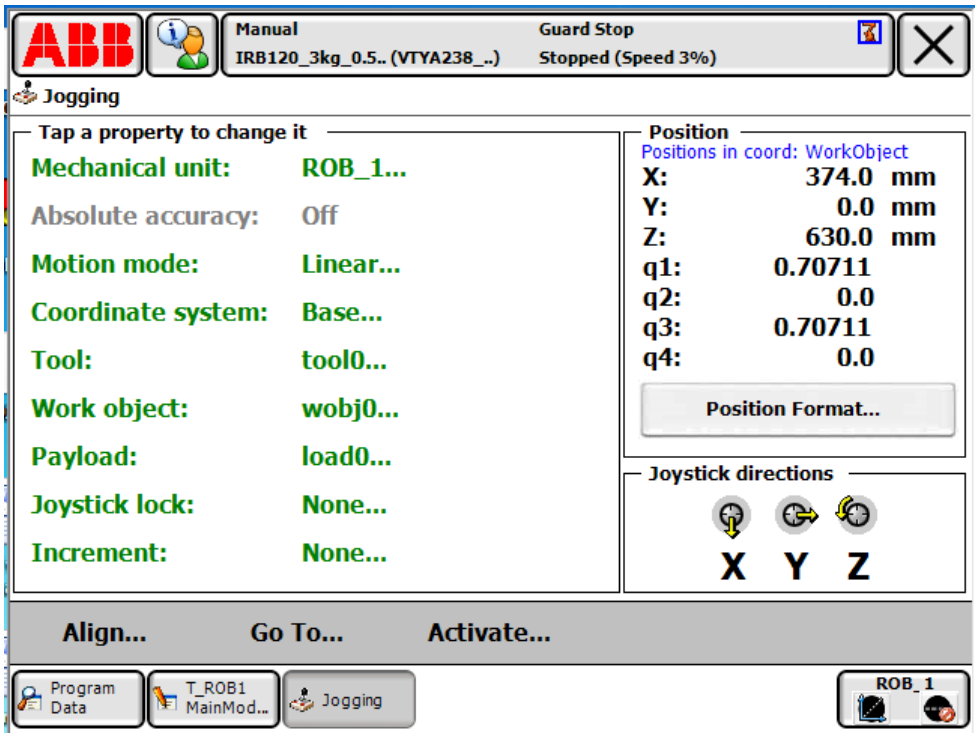
Kuva 32. Robotin tarttuja.

Varastointityyppisen ohjelman looginen ajatus on siirtää x-, y- ja z- akseleiden paikkaa aina haluttaessa. Tämä onnistuu helposti RAPID-ohjelmoinnissa Offs-komennolla (kuva 33) ja numeeristen muuttujien avulla. komentoon määritellään yksikertaisuudessaan aloituspiste, josta tulee akseleiden laskennallinen nollapiste. Aloituspisteen jälkeen määritellään x-, y-, ja z-akseleiden numeerinen arvo millimetreissä. Esimerkiksi kun x-akselille annetaan arvo 10, se tarkoittaa robotin 10 mm:n siirtymistä x-akselin mukaisesti. Arvoiksi voidaan myös asettaa numeerinen muuttuja, jota voidaan ohjelmakierron aikana muuttaa aina haluttaessa.

```
Offs (Point XOffset YOffset ZOffset)
```

Kuva 33. Offs-komennon asetettavat parametrit [13].

Oletuksena käytetään työkalupistettä tool0, joka on sidottu työkalupisteen keskipisteeseen. Koordinaatistona käytetään peruskoordinaatistoa (BASE), joka on sidottu robotin alustan keskipisteeseen (kuva 34).



Kuva 34. Liikeohjaukseen liittyvät parametrit.

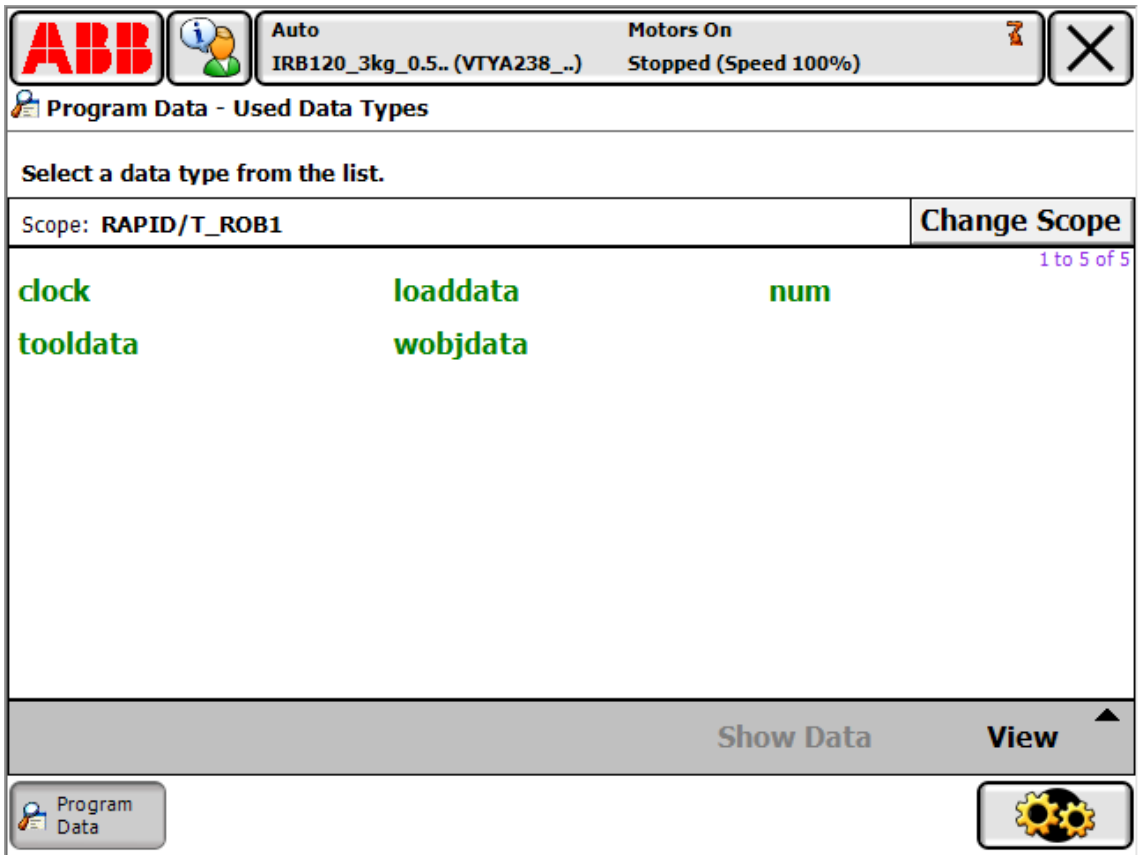
Harjoituksessa käytettävät datatyypit:

- nopeusdata (speeddata)
- robotin paikkadata (robtarget)
- numeeriset arvot (num).

Harjoituksessa käytettävät robotin paikkapisteet määriteltiin online-tilassa. Paikkapisteitä voidaan luoda kahdella eri tavalla. Ensimmäinen tapa on luoda paikkapiste ohjelmaeditorissa liikekäskyn valinnalla halutussa pisteessä tai ohjelmadatassa luomalla datatyyppi robtarget halutussa pisteessä.

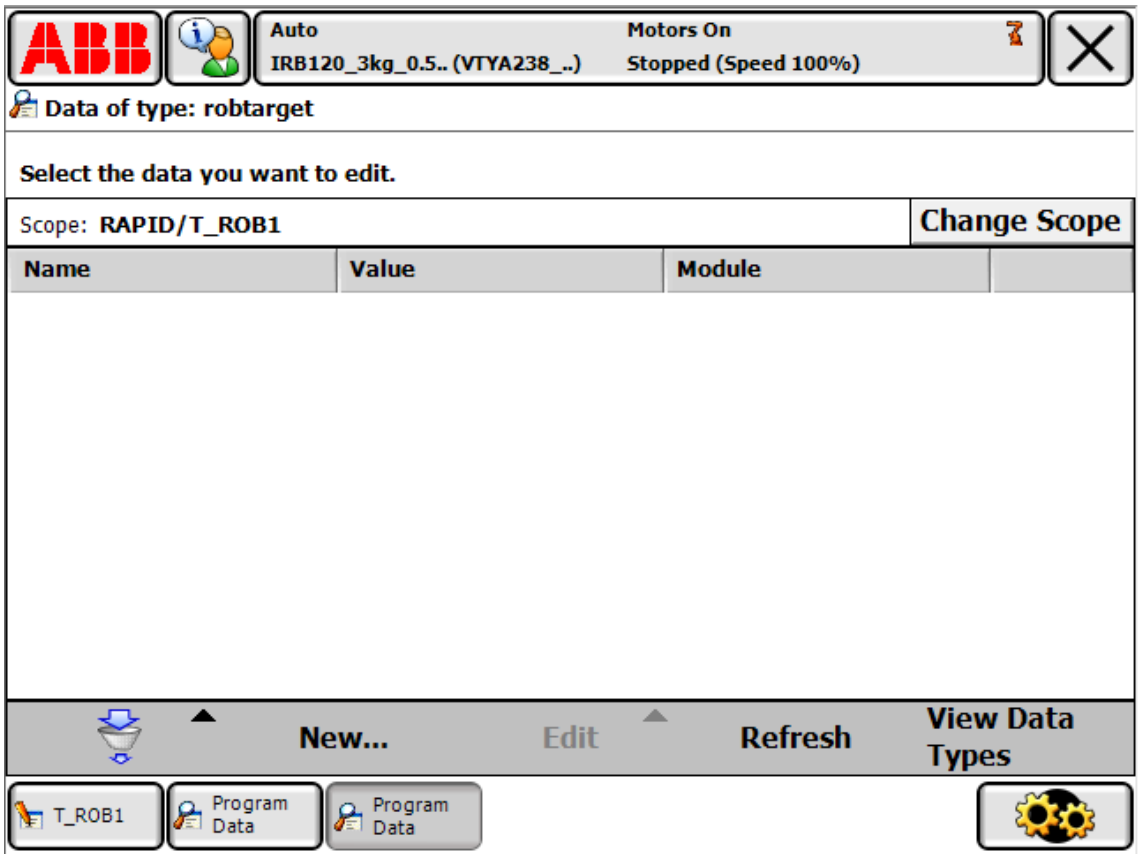
Kummatkin tavat ovat yhtä päteviä, ja samojen parametrien määritys onnistu kumpaakin polkua pitkin.

Robtarget-datatyypin määritys tapahtuu ABB-ikonin painamisesta ja valikosta *Program data* eli datatyypit (kuva 35). Ikkunaan avautuu sen hetkisen käytettävien datatyyppien luettelo.



Kuva 35. Käytössä olevat datatyypit.

Lisää datatyyppejä saadaan esille valitsemalla ”Näytä kaikki datatyypit”. Luettelosta valitaan robtarget. Tehdään uusi datatyyppi (kuva 36).



Kuva 36. Datatyyppikohtainen luonti-ikkuna.

Robotin paikkadataan sisällytetään tarvittaessa neljä robotin paikanmäärittämiseen tarvittavaa parametria (kuva 37).

- trans, työkalupisteen x-, y-, ja z-koordinaatit millimetreissä suhteessa käytettävään koordinaatistoon
- rot, työkalun suunnanmääritysparametrit käytettävän koordinaatiston suhteen
- robconf, määrittää 1-, 4-, ja 6-akseleiden konfiguraatioarvot, neljäs muuttuja on robottityyppikohtainen.
- extax, ulkoisten akseleiden paikanmääritykseen liittyvät arvot.

```
CONST robtarget p15 := [ [600, 500, 225.3], [1, 0, 0, 0], [1, 1,
     0, 0], [ 11, 12.3, 9E9, 9E9, 9E9, 9E9] ];
```

Kuva 37. robtarget-parametrit [11].

Pisteen voi määrittää, joko asettamalla itse halutut robtarget-parametrit tai ajaa robotti haluttuun pisteeseen ja luomalla uusi piste. Tällöin parametrit saavat kyseisen pisteen arvot.

Varastointi harjoituksessa paikkapisteitä määriteltiin yhdeksän:

- Aloitusp, paikkapiste, josta jokainen rutiini alkaa ja päättyy
- Hakuapup, kappaletta haettaessa hakupisteen kohtisuora apupiste
- Hakup, kappaleen hakupiste
- Jattopapu, varastointipaikan haku- ja jättöpisteen kohtisuora apupiste
- Purkuapup, purettavien kappaleiden purkupaikan kohtisuora apupiste
- Purkup, purettavien kappaleiden jättöpiste
- Hakusimulointip, jatkuvatoimisen ohjelman hakupiste
- Hakusimulointiapu, jatkuvatoimisen ohjelman hakuapupiste
- Jattop, varastointipaikan x-, y-, ja z-koordinaattien nollapiste

Harjoitukseen luotiin myös kolme nopeusdataa.

- Hidas, työkalupisteen nopeus 50 mm/s ja työkalun kohdistusnopeus 50 astetta/s
- Keskinopeus, työkalupisteen nopeus 300 mm/s ja työkalun kohdistusnopeus 300 astetta/s

- Nopea, työkalupisteen nopeus 600 mm/s ja työkalun kohdistusnopeus 600 astetta/s

Nopeusdata luodaan samankaltaisesti kuin robtarget-datatyyppi mutta datatyyppivalikosta valitaan speeddata.

Nopeusdataan sisällytetään tarvittaessa neljä nopeusparametria (kuva 38):

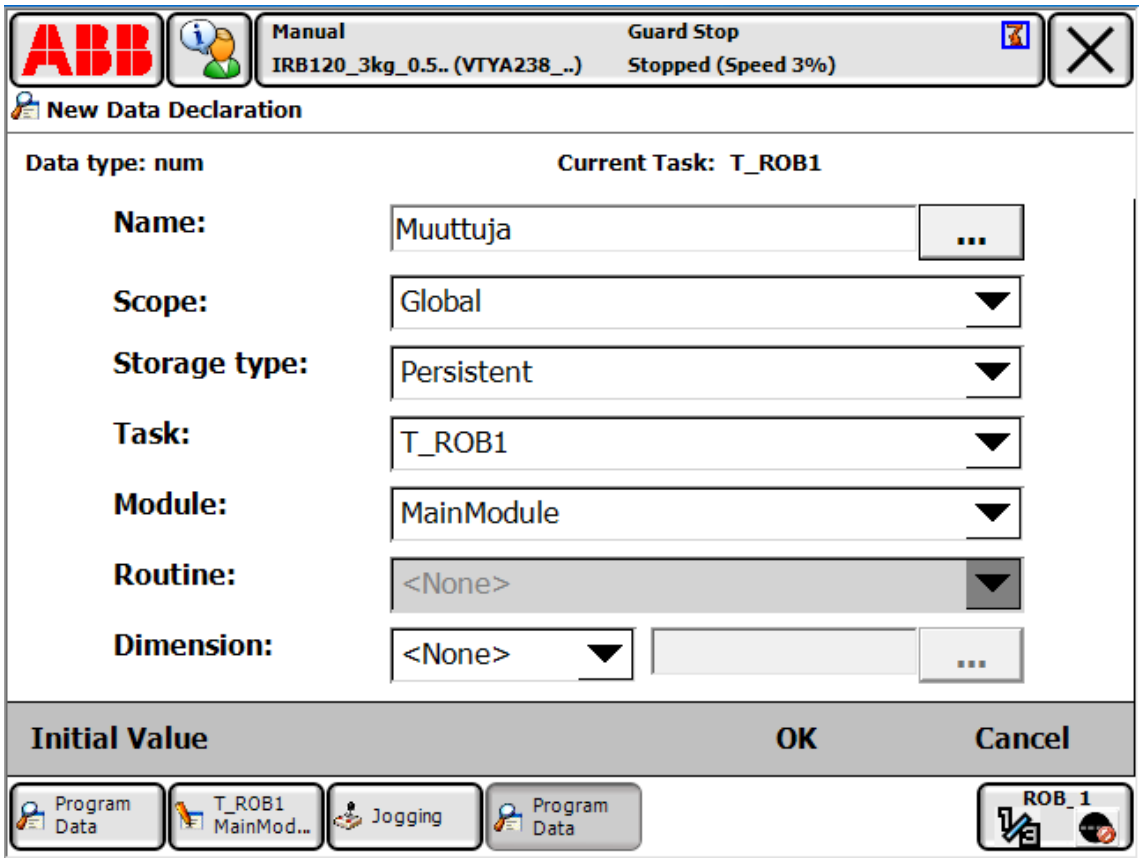
- v_tcp, työkalupisteen nopeus mm/s
- v_ori, työkalun uudelleen kohdistuminen astetta/s
- v_leax, ulkoisen akselin lineaarinen nopeus mm/s
- v_reax, ulkoisen akselin pyörimisnopeus mm/s.

```
VAR speeddata vmedium := [ 1000, 30, 200, 15 ];
```

Kuva 38. speeddata parametrit [11].

Numeerisia arvoja ovat kokonaisluvut ja desimaaliluvut. Kokonaislukujen käyttö rajoittuu Num-datatyypillä välille -8388607…8388608. Tarvittaessa voidaan myös käyttää dnum-datatyyppiä, joka mahdollistaa suuremman kokonaislukualueen.

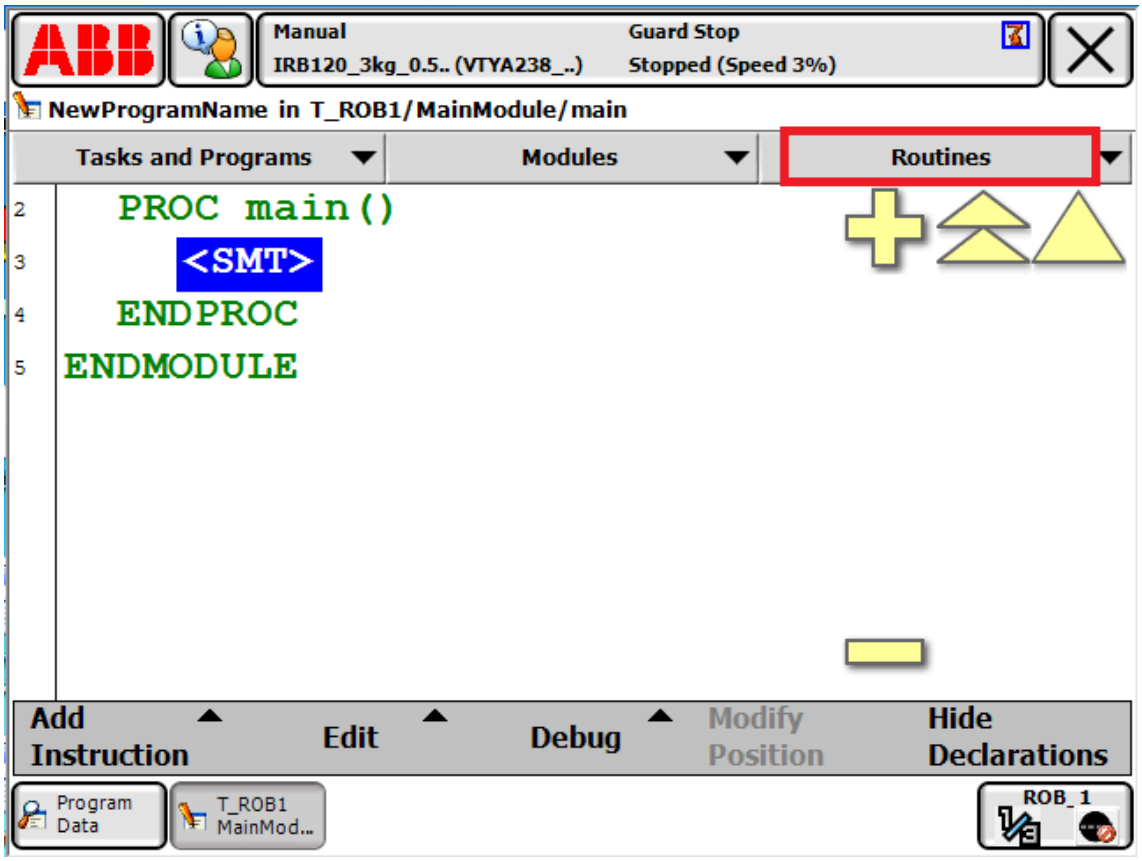
Harjoitusta varten tehtiin Num-datatyyppejä muuttujiin. Muuttujilla esimerkiksi laskettiin varastointipisteen paikanmääritystä akseleille. Num-datatyypin luonti tapahtuu datatyyppivalikosta (kuva 39). Harjoituksessa käytettiin ainoastaan pysyvää muuttujaa (Persistent).



Kuva 39. Num-datatyypin parametrit.

Rutiineiden eli aliohjelmien käyttö ohjelmassa hyödyttää tapauksissa, joissa joudutaan käyttämään samoja komentoja useasti. Esimerkkinä tarttujan ohjaus -aliohjelmassa ohjataan vain robotin lähtöjä päälle ja pois päältä. Onkin ohjelman selkeyden kannalta helpompaa luoda siitä oma rutiininsa ja suorittaa se aina haluttaessa.

Uuden rutiinin tekeminen tapahtuu ABB-valikon Program Editorissa eli ohjelmaeditorissa (kuva 40).



Kuva 40. Kuva ohjelmaeditorista ja rutiini-valikosta.

Harjoituksessa päätettiin luoda aina ensin rutiini halutulle robotin tehtävälle. Taulukossa 7 näkyy ohjelmakierrollinen järjestys kolmelle robotin suorittamalle tehtävälle. Taulukko ei pidä sisällään useasti toistuvia rutiinikutsuja, kuten *Tartuntakiinni-* ja *Tartuntaauki* -rutiineita.

Taulukko 7. Rutiineiden periaatteelliset ohjelmapolut.

Rutiineiden ohjelmapolut		
main		
Messuvalintarut		
Parametritmaaritys		Varastopaikoitussimulointi
Purkuparametrit	Kasausparametrit	Hakusimulointi
VARASTONPURKU	Varastopaikoitus	KappaleenJatto
PURKUHAKU	Haku	Varastonpurkusimulointi
PURKUJATTO	KappaleenJatto	PURKUHAKU
main	main	Purkusimulointi
		Varastopaikoitussimulointi

Tarkastellaan rutiineiden toimintaa ja niiden merkitystä kokonaisuudessa.

Pääohjelma *Main* (kuva 41) alkaa aina robotin siirtymisestä aloituspisteeseen, jonka jälkeen suoritetaan *Tartuntaauki*-rutiini. Alustustoimintojen jälkeen suoritetaan varsinaiset ohjelmarutiinit. Suoritetaan *Messuvalintarut*-rutiini.

```
PROC main()
    MoveJ Aloitusp, Nopea, z50, tool0;
    Tartuntaauki;
    Messuvalintarut;
ENDPROC
```

Kuva 41. main-rutiini.

Messuvalintarut-rutiinissa (kuva 42) FlexPendant tulostaa TPReadFK-komennolla ”PurkuHaku vai Messu”. Vastaus purkuhaku antaa muuttujalle *Messuvalinta* arvon yksi ja *Messu* arvon kaksi. Jos muuttujan arvo on yksi, suoritetaan *Parametrimaarity*s-rutiini. Tai jos muuttuja on kaksi, asetetaan kuvan mukaiset muuttujat nolliksi ja suoritetaan *Varastopaikoitussimulointi*-rutiini. Muuten palataan *main*-rutiiniin.

```
PROC Messuvalintarut()
    TPReadFK Messuvalinta, "PurkuHaku vai Messu", "purkuhaku", "Messu", stEmpty, stEmpty, stEmpty;
    IF Messuvalinta = 1 THEN
        Parametritmaaritys;
    ELSEIF Messuvalinta = 2 THEN
        apuhakuX := 0;
        apuhakuY := 0;
        apuX := 0;
        apuY := 0;
        apuZ := 0;
        kplkoko := 0;
        Varastopaikoitussimulointi;
    ELSE
        main;
    ENDIF
ENDPROC
```

Kuva 42. Messuvalintarut-rutiini.

Tartuntakiinni-rutiini (kuva 43) ohjaa lähtöyksikön lähtöjä DO10_1 ja DO_2. Rutiini suorittaa tarttujan venttiilin ohjauksen niin, että ensin resetoidaan tarttujanohjaus auki DO10_2, jonka jälkeen asetetaan venttiilinohjaus kiinni DO10_1. Puolen sekunnin ajastuksen jälkeen resetoidaan venttiiliohjaus kiinni. Näin varmistetaan, ettei lähtöjä jää päälle.

```
PROC Tartuntakiinni()
    Reset DO10_2;
    Set DO10_1;
    WaitTime 0.5;
    Reset DO10_1;
ENDPROC
```

Kuva 43. Tartuntakiinni-rutiini.

Tartuntaauki-rutiini (kuva 44) ohjaa lähtöyksikön lähtöjä DO10_1 ja DO_2. Rutiini suorittaa tarttujan venttiilin ohjauksen niin, että ensin resetoidaan tarttujanohjaus kiinni DO10_1, jonka jälkeen asetetaan venttiilinohjaus auki DO10_2. Puolen sekunnin ajastuksen jälkeen resetoidaan venttiiliohjaus auki. Näin varmistetaan, ettei lähtöjä jää päälle.

```
PROC Tartuntaauki()
    Reset DO10_1;
    Set DO10_2;
    WaitTime 0.5;
    Reset DO10_2;
ENDPROC
```

Kuva 44. Tartuntaauki-rutiini.

*Parametrimaarity*s-rutiinissa (kuva 45) määritellään ensiksi kappaleenkorkeus. Flexpendant tulostaa näytölle TPreadFK-komennolla ”Pieni 35 mm tai iso 70 mm?” ja ”PIENI”- ja ”ISO” -painikkeet. ”PIENI”-painike asettaa luvun ”1” apumuuttujaan *apuhakuZ ja* ”ISO”-painike asettaa luvun ”2”.

Jos apumuuttuja *apuhakuZ* on yksi, asetetaan apumuuttujaan *kplkoko arvo ”0”*. Tai jos *apuhakuZ* on kaksi, asetetaan apumuuttujan *kplkoko* arvo ”35”. Muuten siirrytään main-pääohjelmaan.

Seuraavaksi tulostetaan "Haluatko purkaa vai kasata?". Jos apumuuttuja *PurkuKasaus* on yksi, siirrytään rutiiniin *Purkuparametrit.* Tai jos apumuuttujan *Purkukasaus* arvo on kaksi, siirrytään rutiiniin *Kasausparametrit.* Muuten siirrytään main-pääohjelmaan.

```
PROC Parametritmaaritys()
    TPReadFK apuhakuZ, "Pieni 35 mm tai iso 70 mm?", "PIENI", "ISO", stEmpty, stEmpty, stEmpty;
    IF apuhakuZ = 1 THEN
        kplkoko := 0;
    ELSEIF apuhakuZ = 2 THEN
        kplkoko := 35;
    ELSE
        main;
    ENDIF
    TPReadFK PurkuKasaus, "Haluatko purkaa vai kasata?", "PURKU", "KASAUS", stEmpty, stEmpty, stEmpty;
    IF PurkuKasaus = 1 THEN
        Purkuparametrit;
    ELSEIF PurkuKasaus = 2 THEN
        Kasausparametrit;
    ELSE
        main;
    ENDIF
ENDPROC
```

Kuva 45. Parametritmaaritys-rutiini.

Purkuparametrit-rutiinissa (kuva 46) määritellään varastoon purkuun vaikuttavat muuttujat. Komennolla *TPreadNum* Flexpendant tulostaa näytölle ”Kuinka monta pinoa on pystyakselilla” ja numeronäppäimistön. Numeronäppäimistön arvo asetetaan apumuuttujaan *PystyvastausX*. *PystyvastausX*-arvo asetetaan *apuX*-apumuuttujaan. Kysymysten vastataukset asetetaan apumuuttujiin *apuX*, *apuY* ja *apuZ.* Asetusten jälkeen siirrytään rutiiniin *VARASTONPURKU*.

```
PROC Purkuparametrit()
    TPReadNum PystyvastausX, "Kuinka monta pinoa on pystyakselilla?";
    apuX := PystyvastausX;
    TPReadNum VaakavastausY, "Kuinka monta pinoa on vaakaakselilla?";
    apuY := VaakavastausY;
    TPReadNum KorkeusvastausZ, "Kuinka monta palikkaa on pinoissa?";
    apuZ := KorkeusvastausZ;
    VARASTONPURKU;
ENDPROC
```

Kuva 46. Purkuparametrit-rutiini.

Kasausparametrit-rutiinissa (kuva 47) määritellään varastoon kasaukseen vaikuttavat muuttujat. Komennolla *TPreadNum* Flexpendant tulostaa näytölle ”Kuinka monta pystyrivia haluat?” ja numeronäppäimistön. Numeronäppäimistön arvo asetetaan apumuuttujaan *PystyvastausX*. Jokaisen kysymyksen jälkeen asetetaan apumuuttujille *PystyvastausX*, *VaakavastausY* ja *KorkeusvastausZ* vastausten arvot. Apumuuttujille *apuX*, *apuY* ja *ApuZ* asetetaan arvo ”0”. Asetusten jälkeen siirrytään rutiiniin *Varastopaikoitus*.

```
PROC Kasausparametrit()
    TPReadNum PystyvastausX, "Kuinka monta pystyrivia haluat?";
    apuX := 0;
    TPReadNum VaakavastausY, "Kuinka monta vaakarivia haluat?";
    apuY := 0;
    TPReadNum KorkeusvastausZ, "Kuinka monta palikkaa haluat pinoon?";
    apuZ := 0;
    Varastopaikoitus;
ENDPROC
```

Kuva 47. Kasausparametrit-rutiini.

HAKU-rutiinissa (kuva 48) robotti liikkuu aloituspisteen kautta Hakuapupisteeseen. Hakupiste toteutetaan Offs-komennolla. Hakupisteen Z koordinaatti on sidottu apumuuttujaan *kplkoko.* Näin robotin hakukorkeus on kappaleen mukainen. Suoritetaan tarttujan *Tartuntakiinni*-rutiini. Tartunnan jälkeen robotti palaa samaa polkua pitkin takaisin aloituspisteeseen.

```
PROC HAKU()
    MoveJ Aloitusp, Nopea, z50, tool0;
    MoveJ Hakuapup, Keskinopeus, z50, tool0;
    MoveJ Offs(Hakup,0,0,kplkoko), Hidas, fine, tool0;
    Tartuntakiinni;
    MoveJ Hakuapup, Keskinopeus, z50, tool0;
    MoveJ Aloitusp, Nopea, z50, tool0;
ENDPROC
```

Kuva 48. HAKU-rutiini.

Hakusimulointi-rutiini (kuva 49) tehtiin messusimulointia varten. Rutiini suorittaa kappaleen haun samankaltaisesti kuin *HAKU*-rutiinissa. Pisteille on määritelty vain eri Offs-aloituspiste. Muuttujilla HakupaikkaX ja HakupaikkaY määritellään kappaleen noutopiste.

```
PROC Hakusimulointi()
    MoveJ Aloitusp, Nopea, z50, tool0;
    MoveJ Offs(Hakusimulointiapu,HakupaikkaX,HakupaikkaY,0), Keskinopeus, z50, tool0;
    MoveJ Offs(Hakusimulointip,HakupaikkaX,HakupaikkaY,0), Hidas, fine, tool0;
    Tartuntakiinni;
    MoveJ Offs(Hakusimulointiapu,HakupaikkaX,HakupaikkaY,0), Keskinopeus, z50, tool0;
    MoveJ Aloitusp, Nopea, z50, tool0;
ENDPROC
```

Kuva 49. Hakusimulointi-rutiini.

Purkusimulointi-rutiinissa (kuva 50) käytetään edellisen rutiinin pohjaa, mutta sitä käytetään ainoastaan kappaleen purkuvaiheessa.

```
PROC Purkusimulointi()
    MoveJ Aloitusp, Nopea, z50, tool0;
    MoveJ Offs(Hakusimulointiapu,HakupaikkaX,HakupaikkaY,0), Keskinopeus, z50, tool0;
    MoveJ Offs(Hakusimulointip,HakupaikkaX,HakupaikkaY,0), Hidas, fine, tool0;
    Tartuntaauki;
    MoveJ Offs(Hakusimulointiapu,HakupaikkaX,HakupaikkaY,0), Keskinopeus, z50, tool0;
    MoveJ Aloitusp, Nopea, z50, tool0;
ENDPROC
```

Kuva 50. Purkusimulointi-rutiini.

Varastopaikoitus-rutiini (kuva 51) suorittaa varaston kasauksen. Oikea paikka palikan jättämiselle haetaan kolmen sisäisen FOR-lauseen avulla. X-, y- ja z-akseleille on jokaiselle oma FOR-lause. Lauseiden avulla varmistetaan, että varaston koko on halutunlainen, ja samalla määritellään jokaiselle kappaleelle oikeat koordinaatit. Itse kappaleen hakeminen ja sen jättäminen tapahtuu kutsutuilla rutiineilla. Jokainen FOR-lause suoritetaan vähintään kerran. Tämä tapahtuu silloin, kun käyttäjä antaa varaston kooksi joka suuntaan arvon 1. Tässä tapauksessa robotti hakee yhden kappaleen, vie sen paikoilleen ja rutiini päättyy tähän.

Rutiinin toimintaa on helpoin tarkastella esimerkin avulla. Otetaan tilanne, jossa käyttäjä on antanut jokaiselle akselille arvon 2 ja kappaleen koko on pieni. Koska FOR-lauseet ovat muotoa FOR i FROM 1 TO *käyttäjänarvo* DO, jokainen lause suoritetaan kaksi kertaa. Laskuri i alkaa arvosta 1 ja päättyy käyttäjän antamaan arvoon, eli tässä tapauksessa kahteen.

Esimerkkitapauksessa varastoon laitetaan 2*2*2 = 8 palikkaa. Näiden koordinaatit suhteessa varastoinnille annettuun jättöpisteeseen lasketaan rutiinissa seuraavasti. Lähtökohtana jokaisen koordinaatin apumuuttujat on nollattu ja *kplkoko* on pienellä kappaleella 0. Lähtökohtien perusteella FOR-lauseen X:n ollessa 1, apuX on 0 ja X:n ollessa 2, apuX on 1 ja niin edelleen.

PaikkaX := apuX * (-60);

PaikkaY := apuY * 80;

PaikkaZ := apuZ * 35 + kplkoko * (apuZ + 1);

Taulukko 8 kertoo kappaleen paikan jokaiselle kahdeksalle palikalle. Esimerkiksi viimeisen palikan paikka on 60 mm x-akselin, 80 mm y-akselin ja 35 mm z-akselin suuntaisesti ensimmäisen palikan paikasta

Taulukko 8. Akseleiden arvot eri kertoimilla.

X	Y	Z	Paikka X	Paikka Y	Paikka Z
1	1	1	0	0	0
2	1	1	-60	0	0
1	2	1	0	80	0
2	2	1	-60	80	0
1	1	2	0	0	35
2	1	2	-60	0	35
1	2	2	0	80	35
2	2	2	-60	80	35

```
PROC Varastopaikoitus()
    FOR i FROM 1 TO KorkeusvastausZ DO
        FOR i FROM 1 TO VaakavastausY DO
            FOR i FROM 1 TO PystyvastausX DO
                HAKU;
                PaikkaX := apuX * (SiirtoX);
                PaikkaY := apuY * SiirtoY;
                PaikkaZ := apuZ * 35 + kplkoko * (apuZ + 1);
                KappaleenJatto;
                apuX := apuX + 1;
            ENDFOR
            apuY := apuY + 1;
            apuX := 0;
        ENDFOR
        apuZ := apuZ + 1;
        apuX := 0;
        apuY := 0;
    ENDFOR
    apuZ := 0;
    PystyvastausX := 0;
    VaakavastausY := 0;
    KorkeusvastausZ := 0;
    main;
ENDPROC
```

Kuva 51. Varastopaikoitus-rutiini.

KappaleenJatto-rutiinissa (kuva 52) aloituspisteen jälkeen kuljetaan Jättöapupisteeseen Offs-komennolla. X- ja y-koordinaatit määritellään apumuuttujissa *PaikkaX* ja *PaikkaY*. Z-koordinaatti on ”0”. Jättöpiste toimii nollapisteenä, ja kaikki koordinaatit ovat apumuuttujissa *PaikkaX*, *PaikkaY* ja *PaikkaZ*. Jättöpisteen jälkeen suoritetaan *Tartuntaauki*-rutiini, jonka jälkeen robotti kulkee saman reitin takaisin aloituspisteeseen.

```
PROC KappaleenJatto()
    MoveJ Aloitusp, Keskinopeus, z50, tool0;
    MoveJ Offs(Jattopapu,PaikkaX,PaikkaY,0), Keskinopeus, z50, tool0;
    MoveJ Offs(Jattop,PaikkaX,PaikkaY,PaikkaZ), Hidas, fine, tool0;
    Tartuntaauki;
    MoveJ Offs(Jattopapu,PaikkaX,PaikkaY,0), Keskinopeus, z50, tool0;
    MoveJ Aloitusp, Nopea, z50, tool0;
ENDPROC
```

Kuva 52. KappaleenJatto-rutiini.

PURKUHAKU-rutiinissa (kuva 53) aloituspisteen jälkeen kuljetaan Jättöapupisteeseen Offs-komennolla. X- ja y- koordinaatit määritellään apumuuttujissa *PaikkaX* ja *PaikkaY*.

Z-koordinaatti on ”0”. Jättöpiste toimii nollapisteenä ja kaikki koordinaatit ovat apumuuttujissa *PaikkaX*, *PaikkaY* ja *PaikkaZ*. Jättöpisteen jälkeen suoritetaan *Tartuntakiinni*-rutiini, jonka jälkeen robotti kulkee saman reitin takaisin aloituspisteeseen.

```
PROC PURKUHAKU()
    MoveJ Aloitusp, Nopea, z50, tool0;
    MoveJ Offs(Jattopapu,PaikkaX,PaikkaY,0), Keskinopeus, z50, tool0;
    MoveJ Offs(Jattop,PaikkaX,PaikkaY,PaikkaZ), Hidas, fine, tool0;
    Tartuntakiinni;
    MoveJ Offs(Jattopapu,PaikkaX,PaikkaY,0), Keskinopeus, z50, tool0;
    MoveJ Aloitusp, Nopea, z50, tool0;
ENDPROC
```

Kuva 53. Purkuhaku-rutiini.

PURKUJATTO-rutiinissa (kuva 54) robotti liikkuu aloituspisteen kautta purkuapupisteeseen. Purkupiste toteutetaan Offs-komennolla. Hakupisteen z-koordinaatti on sidottu apumuuttujaan *kplkoko.* Näin robotin hakukorkeus on kappaleen mukainen. Suoritetaan tarttujan *Tartuntaauki*-rutiini. Tartunnan vapautuksen jälkeen robotti palaa samaa polkua pitkin takaisin aloituspisteeseen.

```
PROC PURKUJATTO()
    MoveJ Aloitusp, Nopea, z50, tool0;
    MoveJ Purkuapup, Keskinopeus, z50, tool0;
    MoveJ Offs(Purkup,0,0,kplkoko), Hidas, fine, tool0;
    Tartuntaauki;
    MoveJ Purkuapup, Keskinopeus, z50, tool0;
    MoveJ Aloitusp, Nopea, z50, tool0;
ENDPROC
```

Kuva 54. Purkujatto-rutiini.

Varaston purkaminen toimii lähes samalla periaatteella, kuin sen kasaaminen (kuva 55). Erona tässä on se, että purkaminen on aloitettava varastoon viimeiseksi laitetusta palikasta ja lopetettava sinne ensimmäisenä laitettuun. Tämä on huomioitava ohjelmassa. Kun kasausvaiheessa apumuuttujia kasvatettiin ja koordinaatteja muutettiin sitä kautta, niin purkuvaiheessa toimitaan päinvastoin. Apumuuttujia vähennetään ja kierroksen päätteeksi niille annetaan käyttäjän antama arvo. Kasausvaiheessa kierroksen jälkeen muuttujat nollattiin. Nyt nollaus tapahtuu vasta koko FOR-lausekierron jälkeen.

```
PROC VARASTONPURKU()
    FOR i FROM 1 TO KorkeusvastausZ DO
        FOR i FROM 1 TO VaakavastausY DO
            FOR i FROM 1 TO PystyvastausX DO
                PaikkaX := (apuX - 1) * (SiirtoX);
                PaikkaY := (apuY - 1) * SiirtoY;
                PaikkaZ := (apuZ - 1) * 35 + kplkoko * apuZ;
                PURKUHAKU;
                PURKUJATTO;
                apuX := apuX - 1;
            ENDFOR
            apuY := apuY - 1;
            apuX := PystyvastausX;
        ENDFOR
        apuZ := apuZ - 1;
        apuX := PystyvastausX;
        apuY := VaakavastausY;
    ENDFOR
    apuX := 0;
    apuY := 0;
    apuZ := 0;
    PystyvastausX := 0;
    VaakavastausY := 0;
    KorkeusvastausZ := 0;
    main;
ENDPROC
```

Kuva 55. Varastonpurku-rutiini.

Varastoinpaikoitussimulointi-rutiini (kuva 56) on tehty toisen harjoitusvaiheen kappaleen kasausta. Se noudattaa samankaltaista logiikkaa kuin aikaisemmat FOR-lauseen tehtävät. Eroten siitä, että muuttujia *HakupaikkaX* ja *HakupaikkaY* asetellaan kappaleiden hakupaikan mukaisesti. FOR-lauseen jälkeen muuttujat asetellaan seuraavaa *Varastonpurkusimulointi*-rutiinia varten halutuiksi.

```
PROC Varastopaikoitussimulointi()
    FOR i FROM 1 TO 2 DO
        FOR i FROM 1 TO 2 DO
            FOR i FROM 1 TO 2 DO
                HakupaikkaX := apuhakuX * 60;
                HakupaikkaY := apuhakuY * 80;
                Hakusimulointi;
                PaikkaX := apuX * (-60);
                PaikkaY := apuY * 80;
                PaikkaZ := apuZ * 35 + kplkoko * (apuZ + 1);
                KappaleenJatto;
                apuX := apuX + 1;
                apuhakuX := apuhakuX + 1;
            ENDFOR
            apuY := apuY + 1;
            apuX := 0;
        ENDFOR
        apuZ := apuZ + 1;
        apuX := 0;
        apuY := 0;
        apuhakuY := apuhakuY + 1;
        apuhakuX := 0;
    ENDFOR
    apuZ := 0;
    apuhakuX := 3;
    apuhakuY := 1;
    Varastonpurkusimulointi;
ENDPROC
```

Kuva 56. Varastopaikoitussimulointi-rutiini.

Varastonpurkusimulointi-rutiinissa (kuva 57) taas puretaan kappaleet päinvastaisessa järjestyksessä, niin että purkupisteet noudattavat kappaleen hakupisteiden päinvastaista järjestystä. Rutiinin lopuksi aloitetaan *Varastonkasaussimulointi*-rutiini. Toisin sanoen saadaan aikaiseksi tehtävän mukainen haluttu, jatkuvatoiminen ohjelma.

```
PROC Varastonpurkusimulointi()
    apuX := 1;
    apuY := 1;
    apuZ := 1;
    FOR i FROM 1 TO 2 DO
        FOR i FROM 1 TO 2 DO
            FOR i FROM 1 TO 2 DO
                HakupaikkaX := apuhakuX * 60;
                HakupaikkaY := apuhakuY * 80;
                PaikkaX := apuX * (-60);
                PaikkaY := apuY * 80;
                PaikkaZ := apuZ * 35 + kplkoko * (apuZ + 1);
                PURKUHAKU;
                Purkusimulointi;
                apuX := apuX - 1;
                apuhakuX := apuhakuX - 1;
            ENDFOR
            apuY := apuY - 1;
            apuX := 1;
        ENDFOR
        apuZ := apuZ - 1;
        apuX := 1;
        apuY := 1;
        apuhakuY := apuhakuY - 1;
        apuhakuX := 3;
    ENDFOR
    apuZ := 0;
    apuX := 0;
    apuY := 0;
    apuhakuX := 0;
    apuhakuY := 0;
    Varastopaikoitussimulointi;
ENDPROC
```

Kuva 57. Varastonpurkusimulointi-rutiini.

12 Yhteenveto

Tässä Metropolia Ammattikorkeakoululle tehdyssä insinöörityössä lähtökohtaisena ajatuksena oli tehdä koululle laboratorioharjoitus, joka olisi erilainen kuin koululla tällä hetkellä robotiikan tunneilla tehtävät harjoitukset. Insinöörityön edetessä kävi kuitenkin selväksi, että työn pääpaino keskittyisi robotilla tehtäviin ohjelmointiharjoituksiin.

Robotiikan teorian opiskelun jälkeen insinöörityössä siirryttiin robotin peruskäytön harjoitteluun ja ohjelmointiin. Perusteellisen harjoittelun jälkeen edettiin tekemään lastausohjelmaa, joka olisi periaatteiltaan mahdollisimman yhtenevä teollisuudessa käytettävien vastaavien ohjelmien kanssa. Suoraan teollisuuteen sopivaa ohjelmaa ei koulun robotilla pystynyt tekemään, mutta samoilla periaatteilla pystyisi ohjelmoimaan toimivan systeemin myös yritysten käyttöön.

Ohjelmoitaessa robottia joudutaan usein etenemään yrityksen ja erehdyksen kautta. Niin kävi nytkin. Hyvin sujuneen perehtymisen jälkeen nuo erehdykset jäivät kuitenkin

pieniksi. Ohjelmien suunnittelussa ei ollut suuria virheitä, ja aikaa kului hienosäätöön suurien virheiden etsimisen sijaan.

Mikäli koululla halutaan kehittää ohjelmoimaamme järjestelmää, niin se voisi tapahtua kuljetinjärjestelmän liittämisellä systeemiin. Tällöin robotti voisi ottaa kappaleita kuljettimelta ja lastata ne haluttuun paikkaan.

Kokonaisuutena arvioisimme käytännön harjoituksen sujuneen erittäin hyvin. Saimme siitä paljon oppia tulevaa varten. Toivomme myös, että tekemämme laboratorioharjoitus auttaa tulevia oppilaita omaksumaan samoja, mielestämme tärkeitä, asioita robottien ohjelmoinnista.

Lähteet

1 Sandhu Harprit. 1997. An intruduction to robotics. Hertfordshire: Nexus Special Interests.

2 Pires J. Norberto. 2007. Industrial Robots Programming. Springer.

3 Kuivanen, Risto. 1999. Robotiikka. Vantaa: Talentum Oyj/MetalliTekniikka.

4 Colestock Harry. 2004. Industrial Robotics. The McGraw-Hill Companies, Inc.

5 Mercedes luopuu roboteista tehtaillaan. Verkkoartikkeli. <http://www.hs.fi/talous/a1456547709229> Luettu 11.3.2016.

6 Keinänen, Kärkkäinen, Metso & Putkonen. 2000. Logiikat ja ohjausjärjestelmät. Vantaa: WSOY KONETEKNIIKKA

7 Pulssianturien teoriaa. Inkrementtianturi, Absoluuttianturi. Verkkodokumentti. <http://www.oem.fi/Tuotteet/Anturi/Pulssianturit/Yleista/Pulssianturien_teoriaa/825723-526144.html> Luettu 3.3.2016.

8 Koivuviita, K. J. 1997. Anturitekniikan Perusteet. Verkkodokumentti. <http://personal.inet.fi/yritys/kkov.eduserver/yhteinen/anturitekniikka2_27_53.pdf > Luettu 3.3.2016.

9 Jokelainen, Jouni. Robotiikka 1. Luentomateriaali. Metropolia Ammattikorkeakoulu. Luettu 3.3.2016.

10 RAPID Reference Manual. Verkkodokumentti <http://rab.ict.pwr.wroc.pl/irb1400/overviewrev1.pdf> Luettu 3.3.2016.

11 Introduction to RAPID. Verkkodokumentti. <http://www.oamk.fi/~eeroko/Opetus/Tuotantoautomaatio/Robotiikka/RapidProgrammering_start.pdf> Luettu 3.3.2016.

12 Suominen, Jyrki & Kuivanen, Risto. 1992. Robottiturvallisuus. Tampere: Työsuojelurahasto.

13 RAPID Instructions, Functions and Data types. Verkkodokumentti. <https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf> Luettu 3.3.2016.

IRB 120 tekniset tiedot

Robotics

IRB 120
Industrial Robot
ABB's smallest robot – for flexible and compact production

The IRB 120 robot is the latest addition to ABB's new fourth-generation of robotic technology and ABB's smallest robot ever produced. Ideal for material handling and assembly applications, the new IRB 120 robot provides an agile, compact and lightweight solution with superior control and path accuracy.

Compact and lightweight
As the smallest robot from ABB, the IRB 120 offers all the functionality and expertise of the ABB range in a much smaller package. Its reduced weight of only 25kg and compact design enables it to be mounted virtually anywhere, whether it is inside a cell, on top of a machine or close to other robots on the production line.

Multipurpose
Ideal for a wide range of industries including the electronic, food and beverage, machinery, solar, pharmaceutical, medical and research sectors, the IRB 120 joins ABB's fourth-generation of new robotic technology.

A white finish Clean Room ISO class 5 version enhances this versatility by making it suitable for environment with stringent cleanliness standard.

The six-axis robot handles a payload of up to 3kg (4kg with its wrist down) and with a reach of 580 mm, the robot is able to carry out a series of operations using flexible rather than hard automated solutions. The IRB 120 is the perfect building block to design cost effective applications – especially when space is at a premium.

Easy to integrate
Weighing in at only 25kg, this robot arm is truly the most portable and easy to integrate on the market. It can be mounted at any angle without any restriction. The smooth surfaces are easy to clean and the cables for air and customer signals are internally routed, all the way from the foot to the wrist, ensuring that integration is effortless.

Optimized working range
In addition to a horizontal reach of 580 mm, the robot has best in class stroke and the ability to reach 112 mm below its base. Furthermore, the IRB 120 has a very compact turning radius, which is enabled by the robots symmetric architecture, without offset on axis 2. This ensures the robot can be mounted close to other equipment and the slim wrist enables the arm to reach closer to its application.

Fast, accurate and agile
Designed with a light, aluminum structure, the powerful compact motors ensure the robot is enabled with a fast acceleration, and can deliver accuracy and agility in any application. Using the IRB 120T variant, cycle-times can be reduced up to 25% where the work piece needs extensive re-orientation and axis 4, 5 and 6 are predominantly used. This faster versions is well suited for pick and packing applications and guided operations together with PickMaster 3™.

IRC5 Compact controller – optimised for small robots
ABB's new IRC5 Compact controller takes the capabilities of the extremely powerful IRC5 controller and presents them in a truly compact format. The new Compact controller brings accuracy and motion control to applications, which previously had been exclusive to large installations.

In addition to space saving benefits, the new controller also enables easy commissioning through one phase power input, external connectors for all signals and a built-in expandable 16 in, 16 out, I/O system.

RobotStudio for offline programming enables manufacturers to simulate a production cell to find the optimal position for the robot, and provide offline programming to prevent costly downtime and delays to production.

Reduced footprint
For applications where floor space is crucial, the combination of the new compact, lightweight architecture of the IRB 120 with the new IRC5 Compact controller introduces a significantly reduced footprint.

Power and productivity
for a better world™

IRB 120

Specification

Variants	Reach	Payload	Armload
IRB 120-3/0.6	580 mm	3 kg (4kg)*	0.3 kg

Features

Integrated signal supply	10 signals on wrist
Integrated air supply	4 air on wrist (5 bar)
Position repeatability	0.01 mm
Robot mounting	Any angle
Degree of protection	IP30
Controllers	IRC5 Compact / IRC5 Single cabinet

Movement

Axis movements	Working range	Maximum speed	
		IRB 120	IRB 120T
Axis 1 Rotation	+165° to -165°	250 °/s	250 °/s
Axis 2 Arm	+110° to -110°	250 °/s	250 °/s
Axis 3 Arm	+70° to -110°	250 °/s	250 °/s
Axis 4 Wrist	+160° to -160°	320 °/s	420 °/s
Axis 5 Bend	+120° to -120°	320 °/s	590 °/s
Axis 6 Turn	+400° to -400°	420 °/s	600 °/s

Performance

	IRB 120	IRB 120T
1 kg picking cycle		
25 x 300 x 25 mm	0.58 s	0.52 s
25 x 300 x 25 with 180° axis 6 reorientation	0.92 s	0.69 s
Acceleration time 0-1 m/s	0.07 s	0.07 s

Electrical connections

Supply voltage	200-600 V, 50/60 Hz
Rated power	
Transformer rating	3.0 kVA
Power consumption	0.25 kW

Physical

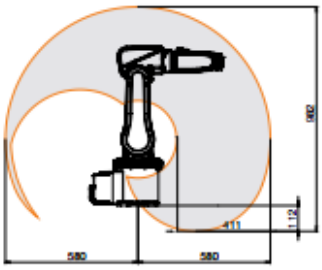
Dimension robot base	180 x 180 mm
Dimension robot height	700 mm
Weight	25 kg

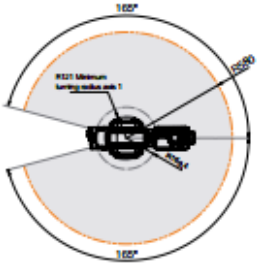
Environment

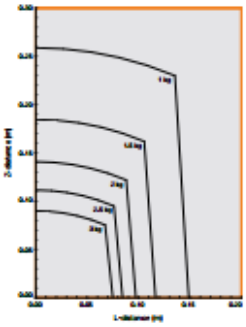
Ambient temperature for Robot manipulator:	
During operation	+5°C (41°F) to +45°C (122°F)
Relative transportation and storage	-25°C (-13°F) to +55°C (131°F)
For short periods	up to +70°C (158°F)
Relative humidity	Max 95%
Options	Clean Room ISO class 5 (certified by IPA)**
Noise level	Max 70 dB (A)
Safety	Safety and emergency stops 2-channel safety circuits supervision 3-position enabling device
Emission	EMC/EMI-shielded

* With vertical wrist
** ISO class 4 can be reached under certain conditions
Data and dimensions may be changed without notice

Working range at wrist center & load diagram







© Copyright ABB Robotics. ROBX-IDEN_O May 2012

www.abb.com/robotics

Power and productivity for a better world™

RAPID-ohjelma

Tässä liitteessä on kokonaisuudessaan tehdyn lastausohjelman ohjelmakoodi.

```
MODULE MainModule
    CONST robtarget Aloitusp:=[[300.55,22.08,400.41],
[0.00952569,0.00578763,0.999935,-0.00221095],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget Hakup:=[[173.98,-457.60,82.40],
[0.00952717,0.00579336,0.999935,-0.00221108],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget Hakuapup:=[[173.98,-457.59,165.69],
[0.00952852,0.00581044,0.999935,-0.00220031],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget Jattop:=[[405.56,-137.24,81.09],
[0.00953168,0.005796,0.999935,-0.00221199],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget Jattopapu:=[[405.56,-137.24,246.04],
[0.00953568,0.00579597,0.999935,-0.00221279],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget Purkup:=[[184.15,317.89,80.45],
[0.0095286,0.0057915,0.999935,-0.00221045],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget Purkuapup:=[[184.15,317.89,183.84],
[0.0095281,0.00579385,0.999935,-0.00221015],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    PERS num apuX:=0;
    PERS num apuY:=0;
    PERS num apuZ:=0;
    PERS num apuhakuZ:=1;
    PERS num apupurkuZ:=0;
    PERS num VaakavastausY:=0;
    PERS num PystyvastausX:=0;
    PERS num KorkeusvastausZ:=0;
    PERS num PurkuKasaus:=2;
    PERS num kplkoko:=0;
    PERS num PaikkaX:=0;
    VAR num PaikkaZ:=0;
    VAR num PaikkaY:=0;
    CONST speeddata Hidas:=[100,90,0,0];
    CONST speeddata Keskinopeus:=[300,90,0,0];
    CONST speeddata Nopea:=[600,90,0,0];
    PERS num apuhakuX:=0;
    PERS num apuhakuY:=0;
    CONST robtarget Hakusimulointip:=[[172.78,-408.39,82.38],
[0.00952354,0.00582467,0.999935,-0.00218856],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget Hakusimulointiapu:=[[172.78,-408.39,139.29],
[0.00952447,0.00582305,0.999935,-0.00218701],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    PERS num HakupaikkaX:=0;
    PERS num HakupaikkaY:=0;
    PERS num Messuvalinta:=2;
    PROC main()
        MoveJ Aloitusp, Nopea, z50, tool0;
        Tartuntaauki;
        Messuvalintarut;
    ENDPROC
    PROC Tartuntakiinni()
```

```
        Reset DO10_2;
        Set DO10_1;
        WaitTime 0.5;
        Reset DO10_1;
    ENDPROC
    PROC Tartuntaauki()
        Reset DO10_1;
        Set DO10_2;
        WaitTime 0.5;
        Reset DO10_2;
    ENDPROC
    PROC Parametritmaaritys()
        TPReadFK apuhakuZ, "Pieni 35 mm tai iso 70 mm?", "PIENI", "ISO", stEmpty, stEmpty,
stEmpty;
        IF apuhakuZ = 1 THEN
            kplkoko := 0;
        ELSEIF apuhakuZ = 2 THEN
            kplkoko := 35;
        ELSE
            main;
        ENDIF
        TPReadFK PurkuKasaus, "Haluatko purkaa vai kasata?", "PURKU", "KASAUS", stEmpty,
stEmpty, stEmpty;
        IF PurkuKasaus = 1 THEN
            Purkuparametrit;
        ELSEIF PurkuKasaus = 2 THEN
            Kasausparametrit;
        ELSE
            main;
        ENDIF
    ENDPROC
    PROC Purkuparametrit()
        TPReadNum PystyvastausX, "Kuinka monta pinoa on pystyakselilla?";
        apuX := PystyvastausX;
        TPReadNum VaakavastausY, "Kuinka monta pinoa on vaakaakselilla?";
        apuY := VaakavastausY;
        TPReadNum KorkeusvastausZ, "Kuinka monta palikkaa on pinoissa?";
        apuZ := KorkeusvastausZ;
        VARASTONPURKU;
    ENDPROC
    PROC Kasausparametrit()
        TPReadNum PystyvastausX, "Kuinka monta pystyrivia haluat?";
        apuX := 0;
        TPReadNum VaakavastausY, "Kuinka monta vaakarivia haluat?";
        apuY := 0;
        TPReadNum KorkeusvastausZ, "Kuinka monta palikkaa haluat pinoon?";
        apuZ := 0;
        Varastopaikoitus;
    ENDPROC
    PROC HAKU()
        MoveJ Aloitusp, Nopea, z50, tool0;
        MoveJ Hakuapup, Keskinopeus, z50, tool0;
        MoveJ Offs(Hakup,0,0,kplkoko), Hidas, fine, tool0;
        Tartuntakiinni;
        MoveJ Hakuapup, Keskinopeus, z50, tool0;
```

```
    MoveJ Aloitusp, Nopea, z50, tool0;
ENDPROC
PROC KappaleenJatto()
    MoveJ Aloitusp, Keskinopeus, z50, tool0;
    MoveJ Offs(Jattopapu,PaikkaX,PaikkaY,0), Keskinopeus, z50, tool0;
    MoveJ Offs(Jattop,PaikkaX,PaikkaY,PaikkaZ), Hidas, fine, tool0;
    Tartuntaauki;
    MoveJ Offs(Jattopapu,PaikkaX,PaikkaY,0), Keskinopeus, z50, tool0;
    MoveJ Aloitusp, Nopea, z50, tool0;
ENDPROC
PROC Varastopaikoitus()
    FOR i FROM 1 TO KorkeusvastausZ DO
        FOR i FROM 1 TO VaakavastausY DO
            FOR i FROM 1 TO PystyvastausX DO
                HAKU;
                PaikkaX := apuX * (-60);
                PaikkaY := apuY * 80;
                PaikkaZ := apuZ * 35 + kplkoko * (apuZ + 1);
                KappaleenJatto;
                apuX := apuX + 1;
            ENDFOR
            apuY := apuY + 1;
            apuX := 0;
        ENDFOR
        apuZ := apuZ + 1;
        apuX := 0;
        apuY := 0;
    ENDFOR
    apuZ := 0;
    PystyvastausX := 0;
    VaakavastausY := 0;
    KorkeusvastausZ := 0;
    main;
ENDPROC
PROC Varastopaikoitussimulointi()
    FOR i FROM 1 TO 2 DO
        FOR i FROM 1 TO 2 DO
            FOR i FROM 1 TO 2 DO
                HakupaikkaX := apuhakuX * 60;
                HakupaikkaY := apuhakuY * 80;
                Hakusimulointi;
                PaikkaX := apuX * (-60);
                PaikkaY := apuY * 80;
                PaikkaZ := apuZ * 35 + kplkoko * (apuZ + 1);
                KappaleenJatto;
                apuX := apuX + 1;
                apuhakuX := apuhakuX + 1;
            ENDFOR
            apuY := apuY + 1;
            apuX := 0;
        ENDFOR
        apuZ := apuZ + 1;
        apuX := 0;
        apuY := 0;
        apuhakuY := apuhakuY + 1;
```

```
        apuhakuX := 0;
    ENDFOR
    apuZ := 0;
    apuhakuX := 3;
    apuhakuY := 1;
    Varastonpurkusimulointi;
ENDPROC
PROC Varastonpurkusimulointi()
    apuX := 1;
    apuY := 1;
    apuZ := 1;
    FOR i FROM 1 TO 2 DO
        FOR i FROM 1 TO 2 DO
            FOR i FROM 1 TO 2 DO
                HakupaikkaX := apuhakuX * 60;
                HakupaikkaY := apuhakuY * 80;
                PaikkaX := apuX * (-60);
                PaikkaY := apuY * 80;
                PaikkaZ := apuZ * 35 + kplkoko * (apuZ + 1);
                PURKUHAKU;
                Purkusimulointi;
                apuX := apuX - 1;
                apuhakuX := apuhakuX - 1;
            ENDFOR
            apuY := apuY - 1;
            apuX := 1;
        ENDFOR
        apuZ := apuZ - 1;
        apuX := 1;
        apuY := 1;
        apuhakuY := apuhakuY - 1;
        apuhakuX := 3;
    ENDFOR
    apuZ := 0;
    apuX := 0;
    apuY := 0;
    apuhakuX := 0;
    apuhakuY := 0;
    Varastopaikoitussimulointi;
ENDPROC
PROC PURKUHAKU()
    MoveJ Aloitusp, Nopea, z50, tool0;
    MoveJ Offs(Jattopapu,PaikkaX,PaikkaY,0), Keskinopeus, z50, tool0;
    MoveJ Offs(Jattop,PaikkaX,PaikkaY,PaikkaZ), Hidas, fine, tool0;
    Tartuntakiinni;
    MoveJ Offs(Jattopapu,PaikkaX,PaikkaY,0), Keskinopeus, z50, tool0;
    MoveJ Aloitusp, Nopea, z50, tool0;
ENDPROC
PROC PURKUJATTO()
    MoveJ Aloitusp, Nopea, z50, tool0;
    MoveJ Purkuapup, Keskinopeus, z50, tool0;
    MoveJ Offs(Purkup,0,0,kplkoko), Hidas, fine, tool0;
    Tartuntaauki;
    MoveJ Purkuapup, Keskinopeus, z50, tool0;
    MoveJ Aloitusp, Nopea, z50, tool0;
```

```
    ENDPROC
    PROC VARASTONPURKU()
        FOR i FROM 1 TO KorkeusvastausZ DO
            FOR i FROM 1 TO VaakavastausY DO
                FOR i FROM 1 TO PystyvastausX DO
                    PaikkaX := (apuX - 1) * (-60);
                    PaikkaY := (apuY - 1) * 80;
                    PaikkaZ := (apuZ - 1) * 35 + kplkoko * apuZ;
                    PURKUHAKU;
                    PURKUJATTO;
                    apuX := apuX - 1;
                ENDFOR
                apuY := apuY - 1;
                apuX := PystyvastausX;
            ENDFOR
            apuZ := apuZ - 1;
            apuX := PystyvastausX;
            apuY := VaakavastausY;
        ENDFOR
        apuX := 0;
        apuY := 0;
        apuZ := 0;
        PystyvastausX := 0;
        VaakavastausY := 0;
        KorkeusvastausZ := 0;
        main;
    ENDPROC
    PROC Hakusimulointi()
        MoveJ Aloitusp, Nopea, z50, tool0;
        MoveJ Offs(Hakusimulointiapu,HakupaikkaX,HakupaikkaY,0), Keskinopeus, z50, tool0;
        MoveJ Offs(Hakusimulointip,HakupaikkaX,HakupaikkaY,0), Hidas, fine, tool0;
        Tartuntakiinni;
        MoveJ Offs(Hakusimulointiapu,HakupaikkaX,HakupaikkaY,0), Keskinopeus, z50, tool0;
        MoveJ Aloitusp, Nopea, z50, tool0;
    ENDPROC
    PROC Purkusimulointi()
        MoveJ Aloitusp, Nopea, z50, tool0;
        MoveJ Offs(Hakusimulointiapu,HakupaikkaX,HakupaikkaY,0), Keskinopeus, z50, tool0;
        MoveJ Offs(Hakusimulointip,HakupaikkaX,HakupaikkaY,0), Hidas, fine, tool0;
        Tartuntaauki;
        MoveJ Offs(Hakusimulointiapu,HakupaikkaX,HakupaikkaY,0), Keskinopeus, z50, tool0;
        MoveJ Aloitusp, Nopea, z50, tool0;
    ENDPROC
    PROC Messuvalintarut()
        TPReadFK Messuvalinta, "PurkuHaku vai Messu", "purkuhaku", "Messu", stEmpty,
  stEmpty, stEmpty;
        IF Messuvalinta = 1 THEN
            Parametritmaaritys;
        ELSEIF Messuvalinta = 2 THEN
            apuhakuX := 0;
            apuhakuY := 0;
            apuX := 0;
            apuY := 0;
            apuZ := 0;
            kplkoko := 0;



            Varastopaikoitussimulointi;
        ELSE
            main;
        ENDIF
    ENDPROC
ENDMODULE
```

Laboratorioharjoitus

Tässä liitteessä on koulun käyttöön tehty laboratorioharjoitus.

IRB-120 Rutiinit ja muuttujien käyttäminen

Sisällys

1. Yleistä

Muuttujia RAPID- ohjelmointikielessä on kolme, muuttuja pysyvä muuttuja ja vakiomuuttuja. Perusmuuttuja ”VAR” asettaa tilansa aina asetusarvoonsa, joka on määritelty muuttujaa luodessa kun ohjelmakierto kulkee main-rutiinin kautta. Pysyvä muuttuja ”PERS” säilyttää aina arvonsa, johon se asetetaan. Vakiomuuttuja ”Const” ei ole muokattavissa ohjelmakierron aikana.

Muuttujia voivat olla esimerkiksi Num-tyyppinen eli numeerinen muuttuja. Niiden voidaan käyttää esimerkiksi RAPID-operaatioissa tai paikkapisteiden muokkaaminen.

Rutiinit toisin sanoen aliohjelmat, hyödyttävät ohjelmakokonaisuudessa, joissa joudutaan käyttämään jatkuvasti samankaltaista robotin toimintoja tai liikkeitä esimerkiksi tarttujan ohjausta.

2. Muuttujien tekeminen

Muuttujia voi tehdä, joko suoraan ohjelmaeditorissa tai sitten ohjelmadatassa. ABB-ikoni → ohjelmadata → näytä kaikki datatyypit → Valitse NUM- datatyyppi → luo uusi.

3. Rutiinin tekeminen

Rutiinin voi luoda ohjelmaeditorissa valitsemalla rutiini → uusi rutiini.

4. Tehtävät

4.1

Ensimmäinen harjoitus aloitetaan tekemällä uusi NUM-tyyppinen **pysyvä** muuttuja. Muuttujalle annetaan nimeksi XSiirto.

Ohjelmaeditorissa tehdään kaksi paikkapistettä. Ensimmäinen paikkapiste voi olla vapaasti valittava.

Toinen paikkapiste luodaan pöydäntasolle. Tee siitä aloituspiste Offs-komennolla. Anna x-akselille tekemäsi muuttuja XSiirto. Anna y- ja z-akseleille arvo ”nolla”.

Määrittele vielä kolmas paikkapiste, esimerkiksi sama kuin ensimmäinen paikkapiste. Liikekomentojen jälkeen lisää funktio ”:=”, jossa summataan Xsiirto-muuttujaan, jokin arvo (mm).

Aja muutaman kerran ohjelma läpi. Voit todeta x-akselin siirtyvän.

HUOM. Mitä liiketapaa käytät? Muuttujan arvon voi asetella käsin ohjelmadatassa ja muuttujan arvoa muokkaamalla.

4.2

Toinen tehtävä on tehdä kaksi uutta rutiinia, jossa toisessa robotin tarttuja avataan ja toinen, jossa tarttuja suljetaan.

Tee main- rutiiniin kolme paikkapistettä ensimmäinen vapaasti valittava paikkapiste. Tee toinen paikkapiste

pöydäntasolle ja kolmas pöydäntasolle, jonnekin muualle kuin piste kaksi.

Aloita ohjelma ensimmäisestä pisteestä jonka jälkeen toinen piste. Kutsu rutiinia, jossa tarttuja menee kiinni (ProcCall). Määrittele robotti kulkemaan ensimmäisen pisteen kautta kolmanteen pisteeseen, jossa kutsutaan tarttujan avaus rutiinia.

4.3

Kolmannessa harjoituksessa yhdistä opitut asiat ja tee ohjelma, jossa kappaleen hakupaikka muuttuu aina ohjelmakierroksen jälkeen.

Pohdittavaa

1. Minkälaisissa käytännön sovelluksissa muuttujia voidaan käyttää?
2. Mitä hyötyä on rutiineiden tekemisessä?