Matti Pelkonen

SEURANTAOMINAISUUDEN KÄYTTÖÖNOTTO KÄSIVARSIROBOTISSA

Tekniikka ja merenkulku Pori
Sähkötekniikan koulutusohjelma
2009

SEURANTAOMINAISUUDEN KÄYTTÖÖNOTTO KÄSIVARSIROBOTISSA

Pelkonen, Matti
Satakunnan ammattikorkeakoulu
Sähkötekniikan koulutusohjelma
huhtikuu 2009
Tuomela, Jorma
UDK: 007.52, 621.865, 681.586
Sivumäärä: 38

Asiasanat: automaatio, robotiikka, konenäkö

Tämän opinnäytetyön tarkoituksena oli ottaa käyttöön seurantaominaisuus Satakunnan ammattikorkeakoulun automaatiolaboratoriossa sijaitsevassa Mitsubishin käsivarsirobotissa. Robotti poimi erivärisiä alumiinisia kappaleita kuljettimelta, ja lajitteli ne värin perusteella eteenpäin. Kamera sai koordinaatit ja kappaleen värin tietoonsa älykameralta, joka oli asennettu kuljettimen yläpuolelle.

Aikaisemmin kappale jouduttiin pysäyttämään poimimisen ajaksi, mutta seurannan ansiosta robotti kykenee poimimaan ne suoraan liikkeestä. Seuranta tarvitsee toimiakseen tietoonsa kappaleen paikkatiedon poimimisen aikana. Tätä varten kuljettimeen asennettiin pulssianturi, jonka avulla voidaan laskea kuljettimen liikenopeus. Nopeuden perusteella määritetään kappaleen paikkatieto ja välitetään se robotille. Aikaisempaan ohjelmaan jouduttiin tekemään huomattavia muutoksia ja tarkentamaan liikkeiden ja kameran toimintaa.

Robottijärjestelmää tullaan myöhemmin käyttämään opetustarkoituksessa laboratoriotöiden yhteydessä. Tämän takia opinnäytetyön raporttiin lisättiin ohjeet, jotka auttavat robotin käyttämisessä ja ohjelmien tekemisessä ja muokkaamisessa.

INTRODUCTION OF TRACKING FEATURE TO ARM ROBOT

Pelkonen, Matti
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Electrical Engineering
April 2009
Tuomela, Jorma
UDC: 007.52, 621.865, 681.586
Number of pages: 38

Key words: automation, robotics, machine vision

The purpose of this thesis was to introduce a tracking feature to the Mitsubishi arm robot in the Automation Laboratory of Satakunta University of Applied Sciences. The robot picked aluminium pieces of a different colour from the conveyor and sorted them forward by the colour of the piece. The robot recieved the coordinates and the colour of the piece from the smart camera mounted above the conveyor.

Previously the piece had to be stopped during the picking but with the tracking feature the robot is able to pick up the pieces while the conveyor is running. For the tracking to work, data is needed about the current position of the piece and the speed of the conveyor. That data was calculated with pulses sent by a pulse sensor. The pulse sensor was installed on the conveyor shaft. The previous program had to be edited and significant changes were made. Besides, the commands for the movements and the operations of the camera were adjusted.

This robot system will later be used as a teaching aid in practical automation laboratory work. Therefore the specifications and instructions were added to the report to assist in using the robot and creating and editing programs.

SISÄLLYS

1 JOHDANTO

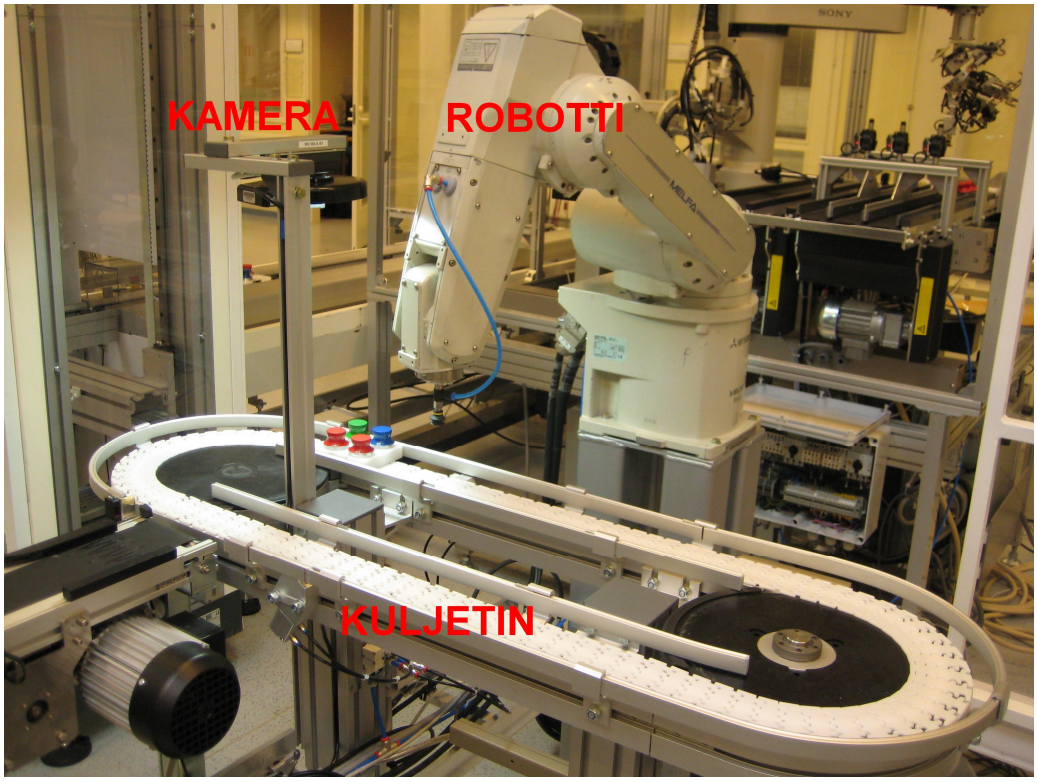
Työn kohteena oli Satakunnan ammattikorkeakoulun automaatiolaboratoriossa sijaitseva robottijärjestelmä, jossa alumiinisia kappaleita poimitaan kuljettimelta värin perusteella. Alkutilanteessa robottijärjestelmä osasi poimia kappaleet silloin, kun ne olivat paikallaan. Paletti, jossa kappaleet olivat, pysäytettiin erillisellä ”stopparilla” tiettyyn kohtaan, kuljettimen liikkuessa sen alla jatkuvasti. Tämä johti usein ruuhkatilanteeseen, kun kaikki kuljettimella olevat paletit kasautuivat pysäytettynä olevan paletin perään. Ruuhkatilanteissa paletit saattoivat pyörähtää kuljettimen mutkissa ja jäädä vinottain jumiin, jolloin koko järjestelmä ei enää edennyt. Työn tarkoituksena oli siis korjata tilanne ja saada robotti poimimaan kappaleet vauhdissa, ilman paletin pysäyttämistä. Ominaisuutta kutsutaan seurannaksi. Tämä ominaisuus siis estää ruuhkatilanteet ja myös nopeuttaa järjestelmän toimintaa.

Robottijärjestelmää tullaan käyttämään myöhemmin automaatiolaboratorion harjoitustöissä. Työstä tehty raportti tulee olemaan keskeisin ohjemateriaali robotin ja ohjelmiston käyttämisessä, koska siihen on koottu käytön kannalta tärkeimmät tiedot ja toimintatavat kuvilla varustettuna.

2 ROBOTTIJÄRJESTELMÄ

Robottijärjestelmä on osana suurempaa kokonaisuutta, jossa erivärisiä alumiinisia kappaleita lajitellaan värin perusteella eri varastointilinjoille. Tältä linjalta toinen robotti lajittelee tilauksen mukaisen väriyhdistelmän kappaleiden kuljetukseen käytettävälle paletille. Paletti kuljetetaan välivarastoon, josta se lähtevät myöhemmin tilaajalle.

Tässä opinnäytetyössä käytetty robotti poimii paineilmakäyttöisen imukupin avulla kuljettimella olevasta paletista kappaleet niiden värin perusteella. Kappaleita on kolmea eri väriä: punainen, vihreä ja sininen. Kun paletti tulee optisen anturin kohdalle, älykamera ottaa kuvan paletista. Kamera kertoo robotin ohjaukselle, missä kohtaa paletissa on tuotteita ja sen, minkä värisiä ne ovat. Jos paletti on tyhjä, poimii robotti paletin erilliselle paletteja käsittelevälle kuljettimelle. Kameran antamat paikkatiedot muutetaan robotin koordinaateiksi, joiden avulla robotti osaa nostaa kappaleen pois paletista. Alkuperäisessä kokoonpanossa kuljettimella oli ”stoppari”, joka pysäytti paletin kohtaan, joka on 140mm kamerasta eteenpäin kuljettimella. Tämä siitä syystä, ettei robotti joutuisi menemään kameran alle, jolloin on suuri riski kameran vaurioitumiseen. Tavoitteena oli poistaa tämä stoppari ja ohjelmoida robotti poimimaan kappaleet kuljettimelta suoraan liikkeestä. Tähän vaadittiin kuljettimen nopeustietoja, joka saatiin asentamalla kuljettimen akselille pulssianturi.

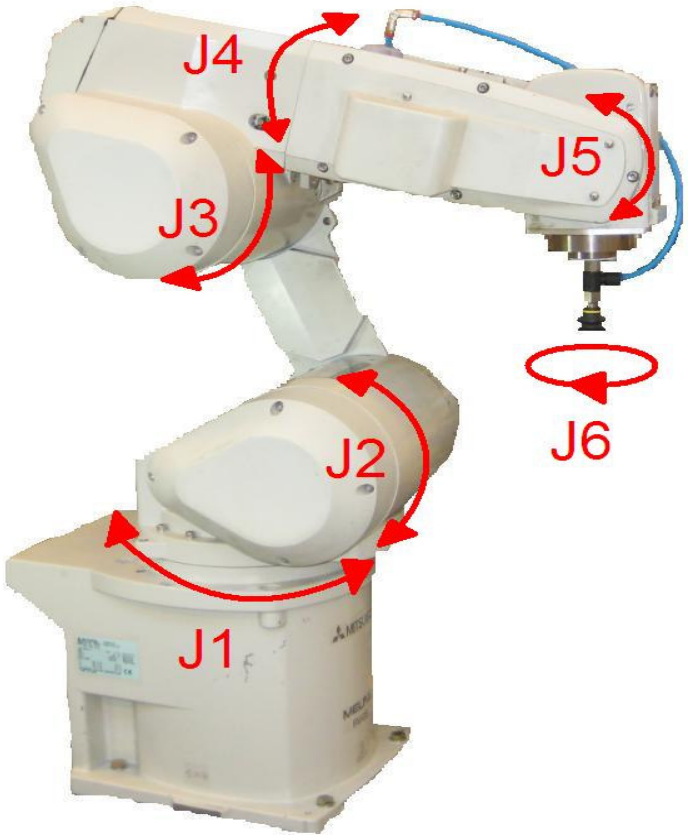


Kuva 1. Robottijärjestelmä.

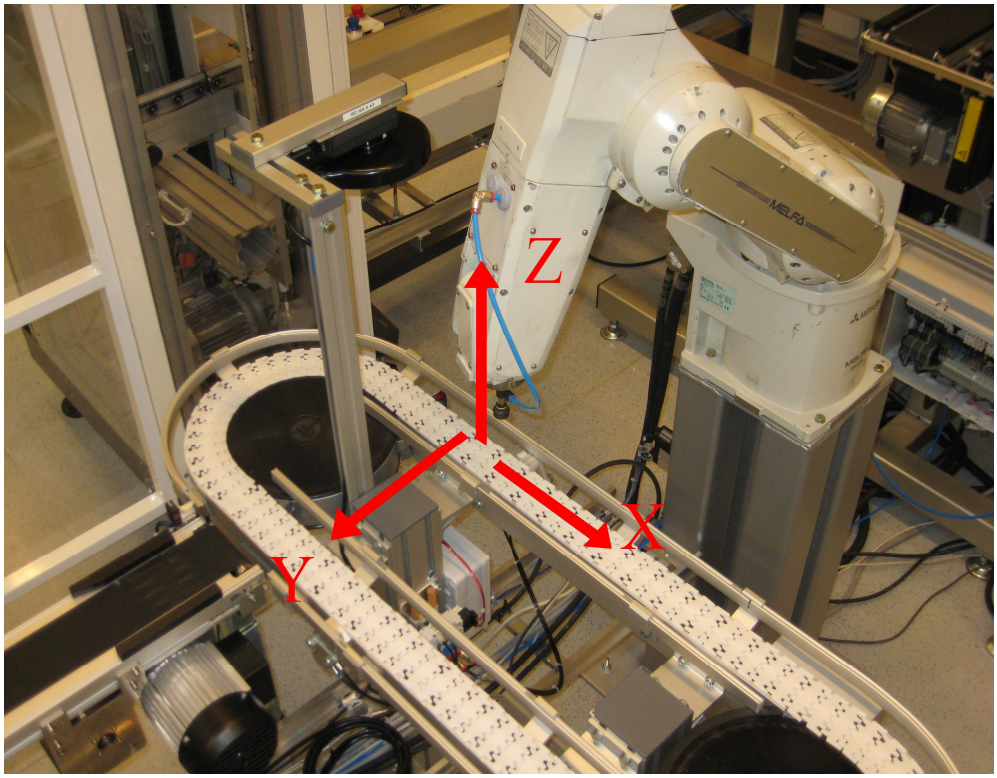
Robottijärjestelmä koostuu viidestä osasta: Robotti, ohjainyksikkö, älykamera, käsiohjain ja tietokone, jotka kaikki olivat yhteydessä toisiinsa. Tietokone ja ohjainyksikkö ovat hieman sivummalla, eivät siis näy tässä kuvassa.

2.1 Robotti: MELFA RV-6S

RV-6S on tavallinen 6-akselinen robotti. Robotin kaikkia akseleita voidaan liikuttaa samaan aikaan, jolloin liikkeet ovat sulavia ja käyttö tehokasta. Robottia voidaan ajaa käsiajolla muuttamalla XYZ-koordinaatteja tai voidaan ajaa akseleita erikseen. Robotille voidaan antaa myös reittipisteet, joiden kautta se kulkee määrätyssä järjestyksessä. Nämä pisteet voidaan ohjelmoida syöttämällä ne käsin tai käyttäen opetustoimintoa, jolloin robotti liikutetaan haluttuihin reittipisteisiin ts. positioihin ja tallennetaan piste. Nämä positiot tallentuvat positiolistaan, josta ohjelma hakee koordinaatit erikseen jokaiseen liikkeeseen./1/



Kuva 2. Robotin eri nivelien liikkeet.



Kuva 3. Robotin koordinaattiakselit piirrettynä kuvaan.

2.1.1 Tekniset tiedot RV-6S

Vapausasteita:	6
Suurin kuormitettavuus:	6 kg
Ohjainyksikkö:	CR2B
Toistotarkkuus:	±0,02 mm
Ulottuvuus:	696 mm
Maksiminopeus:	9 300 mm/s
Paino:	58 kg

/1/

2.2 Ohjainyksikkö: MELFA CR2B-574

CR2B-574 on robotin ohjainyksikkö, jossa tapahtuu varsinainen robotin ohjaus. Tietokoneella tehdyt ja muokatut ohjelmat ja positiolistat lähetetään ohjainyksikölle, johon ne tallentuvat.

Kuva 4. Ohjainyksikkö.

Ohjainyksikkö keskustelee tietokoneen ja kameran kanssa Ethernet-yhteyden välityksellä. Ohjainyksikön etupaneelissa on tärkeimpiä toimintoja käyttävät painikkeet sekä avain, jolla vaihdetaan ohjainyksikön tilaa. Tiloja on kolme: Auto(External): yksikköä ohjataan jollakin ulkoisella ohjaimella(esim. tietokone), Auto(Op): yksikköä ohjataan sen omilla näppäimillä ja Teach, joka mahdollistaa robotin ajon ja ohjelmamuokkaukset käsiohjaimella.. Paneelissa on myös liitäntä käsiohjaimelle R46TB ja hätäseis-kytkin./2/

2.3 Käsiohjain R46TB

Ohjainyksikköön saatiin liitettyä kädessä pidettävä opetusyksikkö. Ohjaimessa oli suurikokoinen kosketusnäyttö, josta eri toimintoja oli helppo käyttää. Ohjaimella pystyi määrittämään robotille paikkapisteitä eli ns. positioita, sekä ajamaan robottia manuaalisesti. Myös ohjelmien ja muuttujien monitorointi onnistui käsiohjaimella. Käsiohjain on todella kätevä käyttää silloin, kun tarvitsee päästä robotin lähelle tarkkailemaan ja samalla ohjata sitä./2/

Käsiohjaimella tapahtuvaan robotin ajoon tulee ohjainyksikön avain kääntää ”Teach”-asentoon. Tämän lisäksi tulee painaa ohjaimesta näppäin ”Teach”, jolloin siihen syttyy valo, ja ohjain aktivoituu. Servot käynnistetään painamalla ohjaimen sivulla olevan ”kuolleen miehen kytkimen” pohjaan ja painaa Servo-näppäintä. Kuolleen miehen kytkin tulee pitää pohjassa jatkuvasti ajettaessa robottia tai muuten servot sammuvat. Se on eräänlainen varotoimenpide, ettei robotti liiku vahingossa painettaessa nappuloita./2/

Kuva 5. Käsiohjain R46TB.

2.4 Älykamera

Älykameratekniikka, eli toisin sanoen konenäkötekniikka, on robotin tavoin korvaamassa ihmistä. Älykameralla voidaan korvata ihmissilmä todella kustannustehokkaasti. Lisäksi oikein säädettynä älykamera on tarkempi, luotettavampi ja nopeampi, eikä se väsy ihmisen tavoin./4/

Konenäköä käytetään laajenevassa määrin teollisuudessa monessa eri tehtävässä. Älykameroita tarvitaan mm. laadunvalvontaan esim. pinnoitevirheet, lajitteluun koon ja värin perusteella esim. hedelmät, mittaukseen esim. kappaleiden pituus tai pinta-ala, sekä liikkeenohjaukseen esim. paikkatiedot robotille.

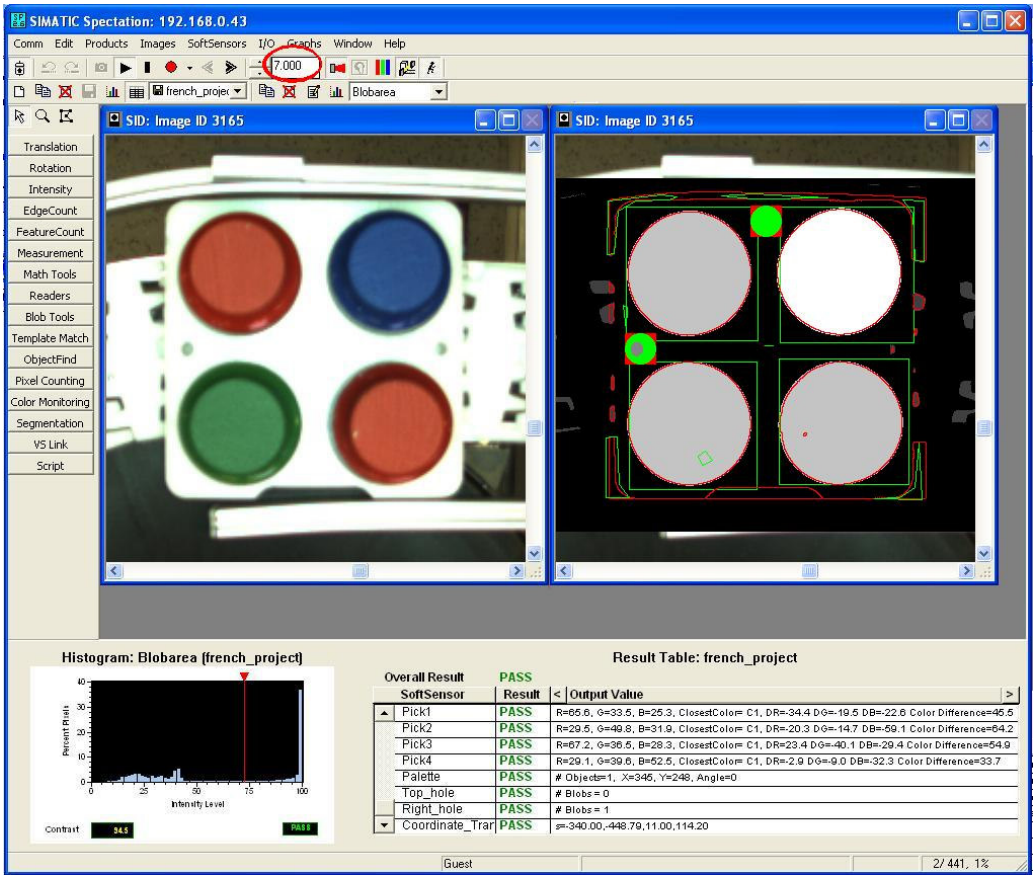
Älykamera oli merkiltään ja malliltaan Siemens VS 725 ja se oli varustettu Stocker Yalen kameravalaisimella. Kamera oli sijoitettu kuljettimen yläpuolelle ja se oli ohjelmoitu ottamaan valokuva paletista silloin, kun paletin reuna ilmestyi valokennon eteen. Kameran ja robotin ohjainyksikön välinen liikenne kulki Ethernet-väylää pitkin.

Kameraan oli ennalta määritetty kuvausalue, josta se havaitsi paletin ja siinä mahdollisesti olleet kappaleet. Kamera kertoi myös kappaleiden värit, joiden mukaan lajittelu tapahtui.

Kuva 6. Kamera ja valaisin alhaalta päin kuvattuna.

Kameran kappaleiden muotojen tunnistus perustuu samanväristen tai lähes samanväristen pikseleiden yhtenevään joukkoon. Se osaa määrittää ääriviivat värialueiden rajaviivoille. Kamera etsii riittävän suurta yhtenäistä värialuetta(Blob) ja laski tälle alueelle keskipisteen. Kamera ilmoitti kappaleen keskipisteen avulla, missä kohtaa palettia nostettavia kappaleita on. Kameran tunnistuskyky ei ollut kuitenkaan kovin hyvä. Tämä johtui hyvin pitkälle valaistuksen vaihtelusta ja valon heijastelusta kappaleiden pinnalta. Heijastuminen johti siihen, ettei kamera aina tunnistanut paletilla olevaa kappaletta lainkaan. Joskus kamera kyllä tunnisti kappaleen, mutta antoi robotille vääristyneet koordinaatit. Joskus jopa niin paljon, että robotti ei osunut kappaleeseen lainkaan.

Kameran ja robotin ohjainyksikön välinen liikenne kulki myöskin Ethernet-väylää pitkin. Kameraa käytettiin Siemensin omalla Simatic Spectation ohjelmalla, jossa on näkyvissä viimeisin kameran ottama kuva, sekä tiedot kameran näkemistä kappaleista ja niiden väreistä.



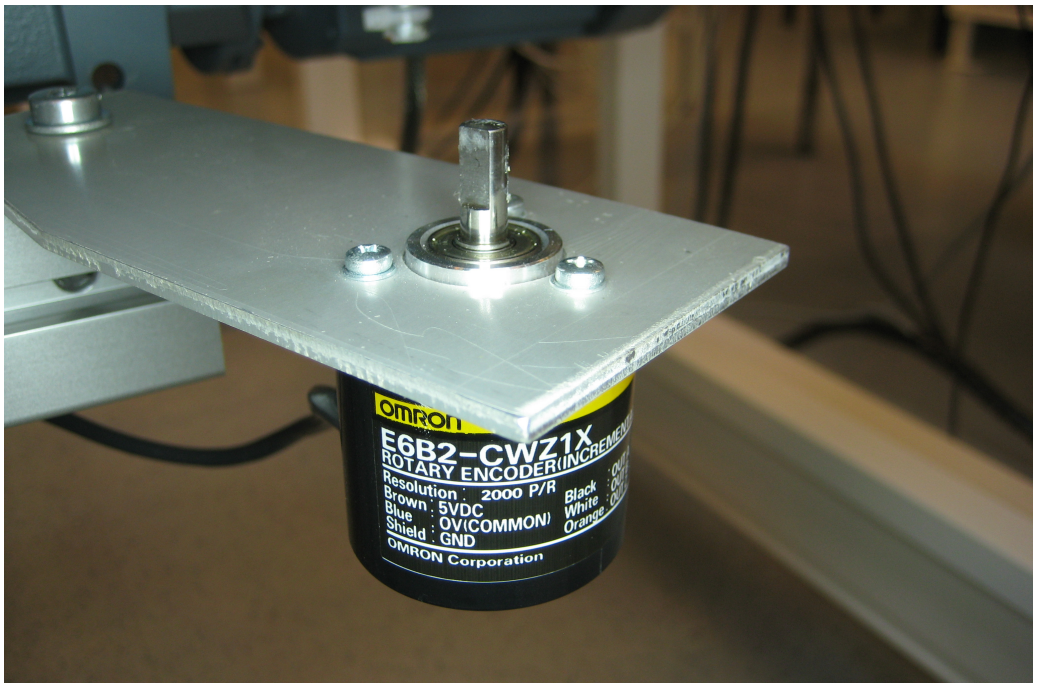
Kuva 7. Simaticin näkymä.

Kuvassa 7. on ylhäällä ympyröity punaisella kenttä, josta valotusaikaa voidaan säätää. Tätä aikaa muuteltiin moneen kertaan valaistuksen muuttuessa työn aikana.

Ohjelma ilmoitti kaikille eri tunnistuskentille(ts. sektoreille) niiden sisältämät punaisen, vihreän ja sinisen värien määrät. Ohjelma vertaili näitä lukuja ja niiden perusteella päätteli, minkä värinen kappale sektorissa on, vai onko siinä kappaletta ollenkaan.

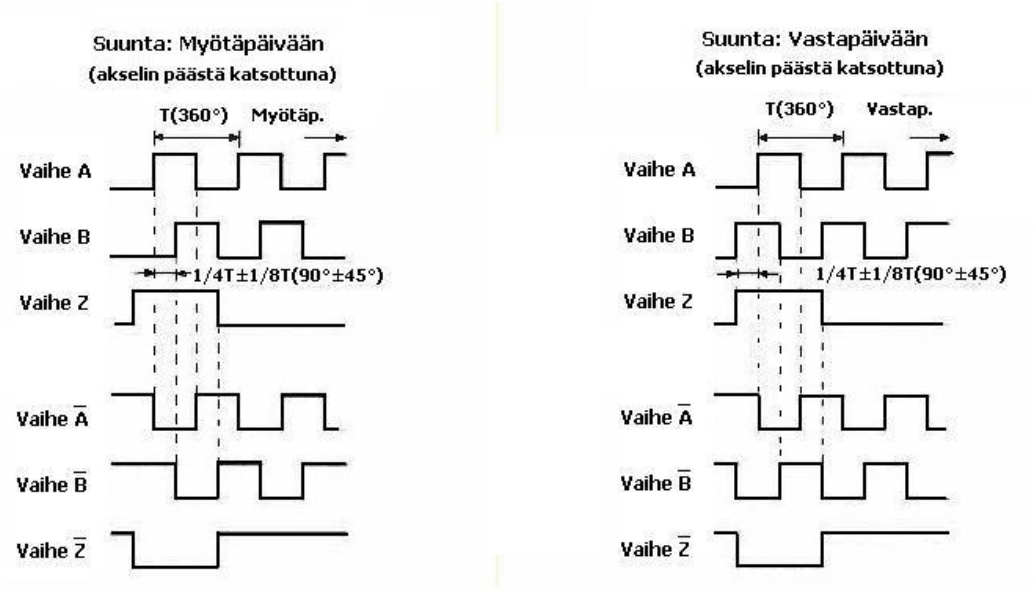
2.5 Pulssianturi

Pulssianturia käytetään yleisesti liikkeen ja sen nopeuden määrittämisessä. Pulssianturin toimintaperiaatteena on pulssien lähettäminen tarkasti akselinsa kiertymiskulmaan verrannollisesti. Antureita on eri resoluutiolla, jolla tarkoitetaan sitä, kuinka monta pulssia jokaista anturin kierrosta kohti se lähettää. Näiden pulssien avulla voidaan koneellisesti laskea, kuinka nopeasti ja mihin suuntaan esim. jokin kuljetin liikkuu. /5/



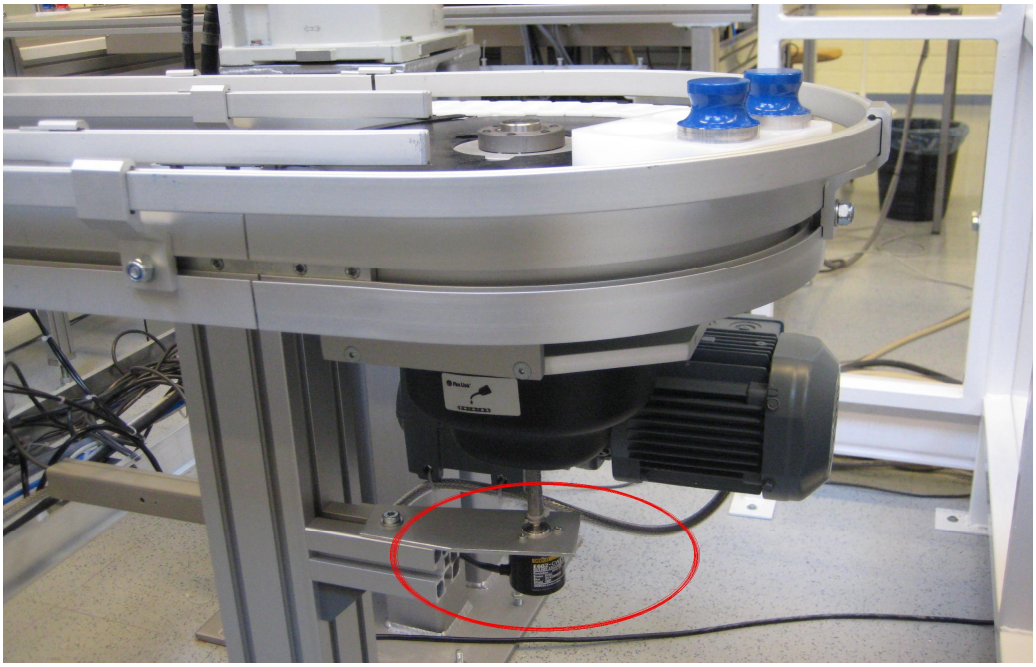
Kuva 8. Pulssianturi akseli kuljettimesta irrotettuna.

Pyörimissuunta määritetään antamalla pulsseja kahteen eri vaiheeseen 1/4T vaiheerolla ja ±1/8T toleranssilla(Kuva 9). Jos A-vaihe on ennen B-vaihetta, pyörimissuunta on myötäpäivään(akselin päästä katsottuna) ja jos B-vaihe on ennen A-vaihetta, pyörimissuunta on vastapäivään. Pyörimissuunta näkyy pulsseissa kasvavana ja laskevana pulssimääränä. Z-vaihe on niin kutsuttu nollapulssi, joka esiintyy joka kierroksella. Nollapulssia voidaan käyttää mm. kiertymisen referenssipisteenä tai pulssien nollaukseen joka kierroksella./5/



Kuva 9. Aikakaavio

Käytetyn anturin tyyppi oli E6B2-CWZ1X. Pulssianturi oli kiinnitetty suoraan kuljettimen akseliin, jolloin niiden pyörimisnopeudet ovat yhtä suuret. Anturin ja kuljettimen akselit liitettiin yhteen joustavalla välikappaleella, jottei anturiin kohdistuisi minkäänlaisia vääntäviä voimia. Ne voisivat ajan kanssa vahingoittaa sitä./5/



Kuva 10. Pulssianturi kiinnitettynä kuljettimen akseliin.

2.5.1 Liitäntä ja yhteydet

Pulssianturi on liitetty ohjausyksikössä olevaan liitäntäkorttiin anturin omalla kaapelilla. Ohjainyksikössä on korttipaikka, johon pulssianturin kortti on kiinnitetty.

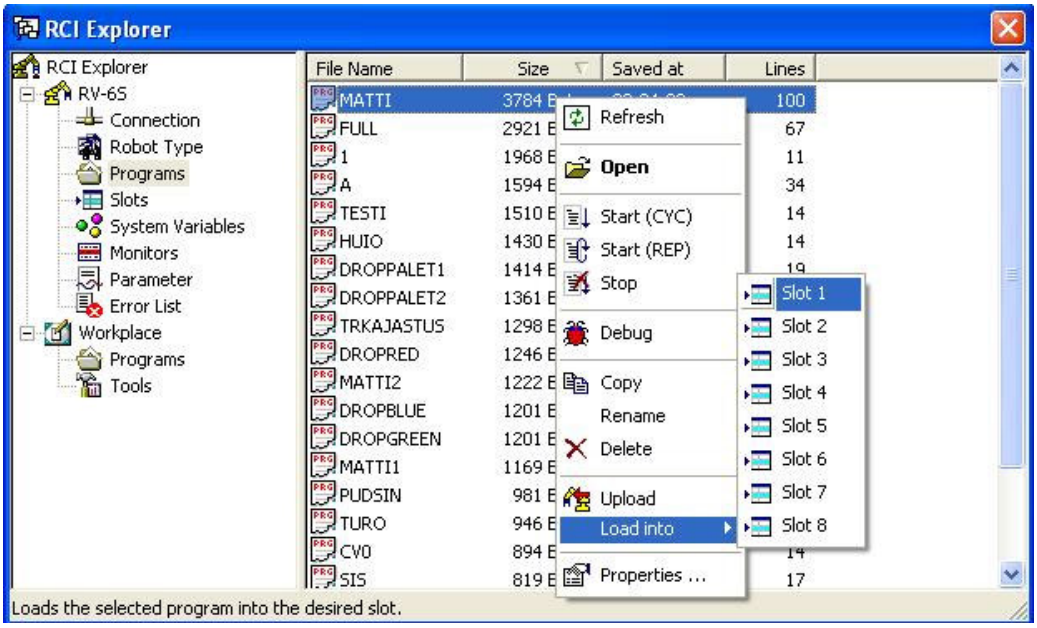
2.6 Ohjelmat

Ohjelmia tehtiin, muokattiin ja hallittiin ohjelmalla nimeltä Cosirob. Cosirob on kehittynyt ohjelmisto, jonka avulla voi hallita täydellisesti myös robottia ja sen toimilaitteita. Ohjelmassa on työkalut ohjelmointia, projektinhallintaa, virheentarkistusta ja monitorointia varten. Robottia voi myös ajaa ohjelman avulla ja luoda sen avulla positiolistat.

Ohjelmat toteutettiin ohjelmointikielellä, jossa jokainen komentorivi on numerojärjestyksessä. Näin pystyttiin halutessa kesken ohjelman hypätä jollekin tietylle riville numeron perusteella. Tästä on hyötyä lähinnä ehtolauseissa, joissa kysytään jotain tuloa tai parametriä, ja jos ehto ei täyty, hypätään jollekin määritetylle riville.

Ohjelmointikieli on konekielestä kehittyneempi ja yksinkertaistetumpi kieli. Siinä on yhdistetty useampi konekielen komento yhdeksi ohjelmointikielen komennoksi. Komennoista on tehty mahdollisimman loogisia ja toimivia, juuri nopeaa ohjelmointia ajatellen. Ohjelmointikielen ja konekielen etuna bittipohjaiseen ohjelmointiin verrattuna on virheiden väheneminen, koska komennot pysyvät aina samoina. Yksittäiset bittivirheet eivät siis ole ongelmana. Lisäksi ohjelmointikielessä virheet on helpompi havaita ja korjata. Tässä työssä käytetty ohjelmointikieli oli Mitshubishin oma MELFA BASIC IV. /3/

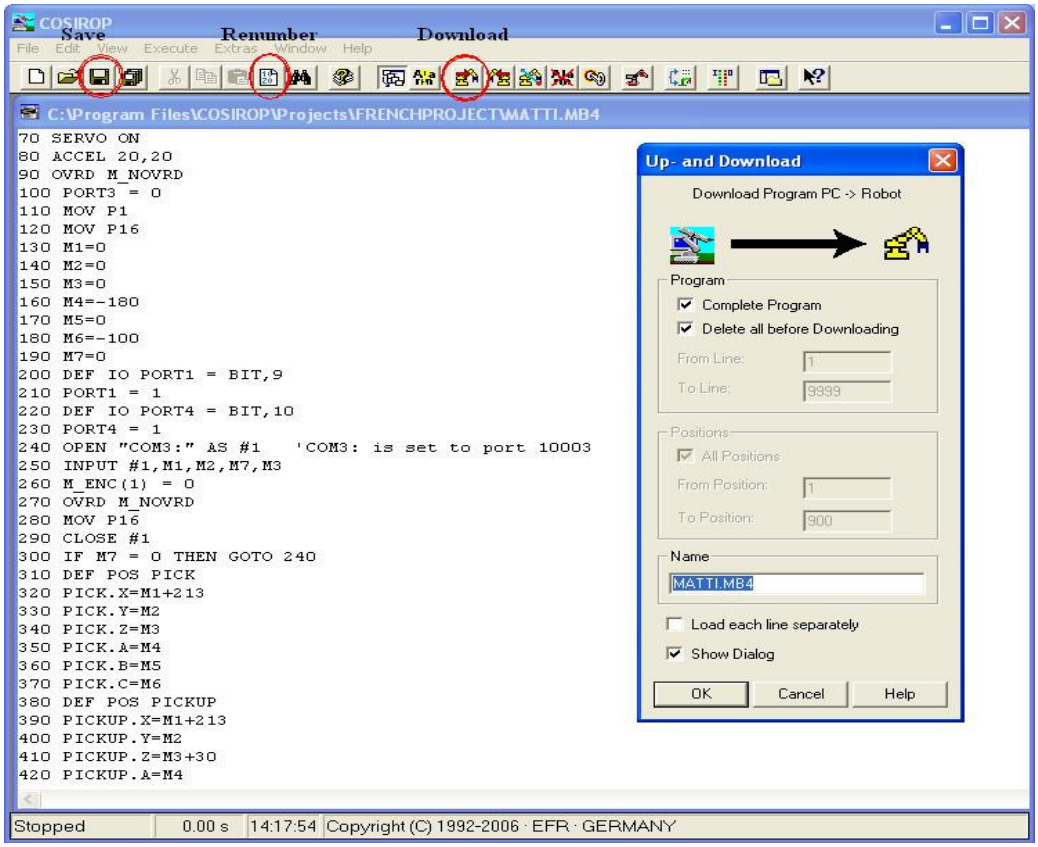
Ohjelmat tehtiin tietokoneella ja siirrettiin sitten ohjainyksikköön. Ohjainyksikössä ohjelma piti ladata aina johonkin ohjelmapaikkaan, slottiin, josta robotti sitä käytti.



Kuva 11. Ohjelman lataus ohjelmapaikkaan.

Ohjelma täytyy uudelleen numeroida(renumber) aina, kun siihen lisätään tai poistetaan komentorivejä. Uudelleen numerointi tarkoittaa sitä, että ohjelma tarkastaa jokaisen komentorivin numeron ja tarvittaessa muuttaa ne peräkkäisiksi, jos näin ei ole.

Uudelleen numeroinnin jälkeen ohjelma tallennetaan ja tämän jälkeen se ladataan ohjainyksikköön. Ohjelman täytyy olla Stop-tilassa, jotta sitä voidaan muuttaa.



Kuva 12. Ohjelman uudelleen numerointi, tallennus ja lataus.

Jokaisella ohjelmalla on myös oma positiolistansa(Kuva 12), joka myös pitää lähettää ohjainyksiköllе muutoksien jälkeen. Ohjainyksikön Multitasking-ominaisuuden ansiosta ohjelmia voi olla päällä useampi samaan aikaan ja ne voivat olla yhteyksissä toisiinsa. Ohjelmassa voidaan myös kutsua toista ohjelmaa, aliohjelmaa. Tällöin kutsuttu ohjelma siirtyy kyseiseen ohjelmapaikkaan. /6/

2.6.1 Pääohjelma

Pääohjelmassa hoidetaan laitteiston ylösajo(robotti, kuljetin, ym.), sekä tarvittavien tietoliikenneporttien avaaminen ja tarvittavien aliohjelmien kutsuminen. Ohjelmassa määritetään myös kappaleiden nostokohtien koordinaatit ja tehdään tarvittavat korjaukset niihin. Seuranta olisi ollut mahdollista liittää järjestelmään kokonaan omana ohjelmana, mutta liittämällä se pääohjelmaan saatiin se toimimaan paremmin halutulla tavalla. Liitteessä 1. on pääohjelman ohjelmakoodi kommentoituna.

2.6.2 Aliohjelmat

Pääohjelman kanssa käytettiin aliohjelmia, joita kutsuttiin siinä vaiheessa, kun kappale oli nostettu pois paletilta. Niitä käytettiin siis kappaleiden lajitteluun oikealle linjalle seuraavalla kuljettimella. Aliohjelmien nimet olivat DROPBLUE, DROPGREEN, DROPRED ja DROPPALETTE. Kameran antaman tiedon perusteella pääohjelma osasi kutsua oikean aliohjelman. Aliohjelmat ovat hyvin yksinkertaisia, sisältäen lähinnä vain liikekäskyjä ja tarttujan ohjauksen. Ohjelmat ovat keskenään samanlaisia, mutta niillä on jokaisella omanlaisensa positiolista, jonka perusteella lajittelu tapahtuu.

Kun aliohjelma saatiin ajettua loppuun, siirryttiin takaisin pääohjelmaan. Myös nämä ohjelmat olivat pääosin valmiit, joskin joitakin muokkauksia jouduttiin tekemään paikkatietoihin ja liikekäskyihin.

2.6.3 Apuohjelmat

Apuohjelmat olivat tilapäisiä ohjelmia, joita käytettiin muuttujien ja parametrien määrittämiseen, sekä eri toimintojen testaukseen. Ne eivät siis olleet jatkuvassa käytössä.

2.6.3.1 Kalibrointi

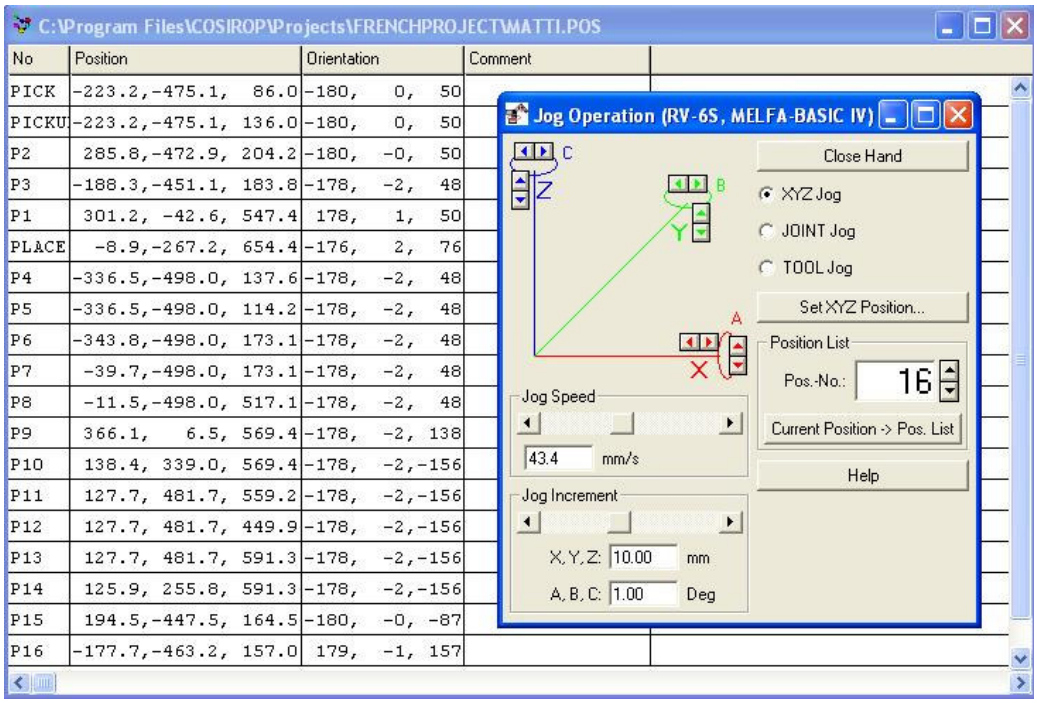
Kalibrointiohjelmaa(Liite 2) käytettiin pulssianturin antaman pulssimäärän muuttamiseen robotin liikettä vastaavaksi matkaksi. Kalibrointidata tallentui parametriin P_ENCDLT/7/.

2.6.3.2 Testi

Testiohjelmaa käytettiin nimensä mukaan erilaisten komentojen ja toimintojen testaamiseen. On paljon helpompaa testata uusia asioita ohjelmassa, jossa ei ole mitään muita erikoisia toimintoja. Tällöin testaaminen on nopeampaa ja uusien komentojen vaikutus ja toiminta on helpompi havaita. Lisäksi näin saatiin varmuus, ettei vahingossa pilata varsinaista ohjelmaa tai epähuomiossa pyyhitä pois siitä jotain olennaista. Testiohjelma osoittautui todella hyödylliseksi juuri nopean ja selkeän käytettävyytensä ansiosta.

2.6.4 Positiolistat

Ohjelmat tarvitsevat rinnalleen positiolistan, jossa on kaikkien robotin käyttämien paikkapisteiden koordinaatit. Eri ohjelmat tarvitsevat eri määrän paikkapisteitä, riippuen itse ohjelman laajuudesta.



No	Position	Orientation	Comment
PICK	-223.2,-475.1, 86.0	-180, 0, 50	
PICKU	-223.2,-475.1, 136.0	-180, 0, 50	
P2	285.8,-472.9, 204.2	-180, -0, 50	
P3	-188.3,-451.1, 183.8	-178, -2, 48	
P1	301.2, -42.6, 547.4	178, 1, 50	
PLACE	-8.9,-267.2, 654.4	-176, 2, 76	
P4	-336.5,-498.0, 137.6	-178, -2, 48	
P5	-336.5,-498.0, 114.2	-178, -2, 48	
P6	-343.8,-498.0, 173.1	-178, -2, 48	
P7	-39.7,-498.0, 173.1	-178, -2, 48	
P8	-11.5,-498.0, 517.1	-178, -2, 48	
P9	366.1, 6.5, 569.4	-178, -2, 138	
P10	138.4, 339.0, 569.4	-178, -2,-156	
P11	127.7, 481.7, 559.2	-178, -2,-156	
P12	127.7, 481.7, 449.9	-178, -2,-156	
P13	127.7, 481.7, 591.3	-178, -2,-156	
P14	125.9, 255.8, 591.3	-178, -2,-156	
P15	194.5,-447.5, 164.5	-180, -0, -87	
P16	-177.7,-463.2, 157.0	179, -1, 157	

Kuva 13. Positiolista ja Jog-operaattori.

Listassa on aina yksi positio yhdellä rivillä. Ensimmäinen luku on position X-koordinaatti, toinen Y-koordinaatti ja kolmas on Z-koordinaatti. Kolme viimeistä lukua ovat robotin kolmen uloimman akselin kiertymiskulmat A, B ja C, joista C on uloin, B toiseksi uloin ja A kolmanneksi uloin akseli/7,s.16/.

3 KEHITYSHANKE

Tämä osio pitää sisällään varsinaisen työn osuuden, jossa vanha järjestelmä muokataan oikeanlaiseksi ja seuranta lisätään ohjelmaan. Myös tarvittavat hienosäädöt on kerrottu tässä osiossa.

3.1 Pulssianturi

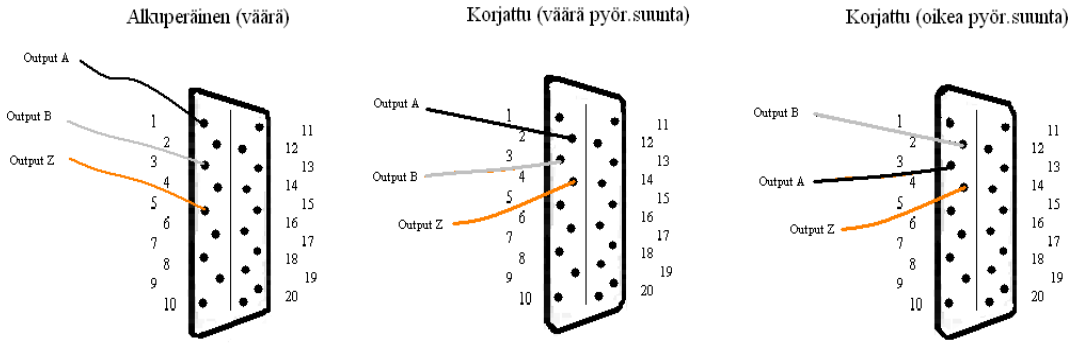
Pulssianturi on sinänsä yksinkertainen ja varmatoiminen anturi. Kuitenkin sen asentaminen ja käyttöönotto on usein työläämpää ja tarkempaa, kuin monilla muilla antureilla. Tässäkin tapauksessa pulssianturin kanssa oli ongelmia, jotka kuitenkin saatiin ratkaistua selvittämällä tarkkaan kyseisen pulssianturin toiminta.

3.1.1 Ongelmatilanne ja ratkaisu

Kun työ aloitettiin, oli tiedossa, että pulssianturi ei toimi oikealla tavalla. Ensin testattiin Cosirob-ohjelmassa, että yhteys pelaa. Pulssit tallentuvat M_ENC(1) muuttujaan, joka on kasvava tai pienenevä luku pyörimissuunnasta riippuen. Kun ohjelmalla tarkasteli lukemaa ja samalla pyöritti anturia, pulssiluku vaihteli vain yhden pulssin verran suurempaan ja pienempään.

Ratkaisua lähdettiin hakemaan liitosjohdosta ja sen liittimestä. Johto näytti päälisin puolin ehjältä, joten siirryttiin tutkimaan liitintä ja sen kytkentää. Pulssianturin manuaalista löytyy liittimen ja anturikaapelin kytkentäkuva, jossa kerrotaan tarkasti johtimien värikoodit ja liittimen nasta./5/ Lähempi tarkastelu osoitti, että johtimet oli juotettu alun perin epähuomiossa vääriin nastoihin, eikä anturin oikeanlainen toiminta täten ollut mahdollista. Johtimet juotettiin manuaalin määräämään järjestykseen, liitin kasattiin ja laitettiin paikoilleen. Kun nyt anturin akselia pyöritti, pulssiluku kasvoi ja pieneni aivan niin kuin pitikin. Anturi kiinnitettiin paikoilleen ja liitettiin kuljettimen akseliin. Kun kuljetinta alettiin pyörittämään, selvisi, että halutulla pyörimissuunnalla pulssiluku onkin laskeva. Ohjelmallisesti pulsseja on huomattavasti helpompi käsitellä, jos pulssiluku kasvaa oikeassa pyörimissuunnassa. Ongelma saa-

tiin ratkaistua avaamalla liitin vielä kerran ja vaihtamalla A- ja B-kanavien paikkaa, jolloin niiden järjestys muuttuu ja anturi luulee pyörivänsä eri suuntaan.



Kuva 14. Liittimen kytkentä eri tilanteissa.

3.2 Ohjelmointi

Ohjelmien rungot olivat aiemman työryhmän jäljiltä jokseenkin valmiit. Muutokset ja hienosäätö kuitenkin lisäsivät rivimäärää melkoisesti varsinkin pääohjelmassa. Pelkästään vanhan ohjelman eri toimintojen ja termien opetteleminen kesti huomattavan kauan. Manuaaleja ja erilaisia ohjeita oli käytössä useita ja ne kaikki olivat englanninkielisiä. Tästä johtui, että jokaista komentoa joutui etsimään kauan ja useasta paikasta. Sanakirja oli myös usein käytössä vaikeahkon ammattisanaston johdosta.

Koska X-akseli oli ainoa akseli, johon haluttiin muutosta seurannan toimesta, piti ensin selvittää, täytyykö pienentää vai suurentaa X-koordinaattia. Ohjelman rivillä 320 PICK.X=M1+210 määritetään PICK-paikkatiedon X-koordinaatti ja lisätään siihen 210mm. PICK tarkoittaa kappaleiden nostopaikkaa. Kun tätä millimetrilukemaa kasvatettiin, siirtyi nostopaikka kamerasta poispäin, samaan suuntaan kuin kuljetin liikkui. Tämä tarkoitti siis sitä, että seurannalta tulevan pulssimäärän tulee kasvattaa X-akselin koordinaattia.

Ohjelmointi aloitettiin kokeilemalla lisätä pulssianturin pulssiluku suoraan muuttujasta komentoriville ja jakaa se pulssisuhteella, joka oli aiemmin manuaalisesti laskettu. PICK.X=M1+(M_ENC(1)/8,5). Tämä ei kuitenkaan vaikuttanut millään lailla

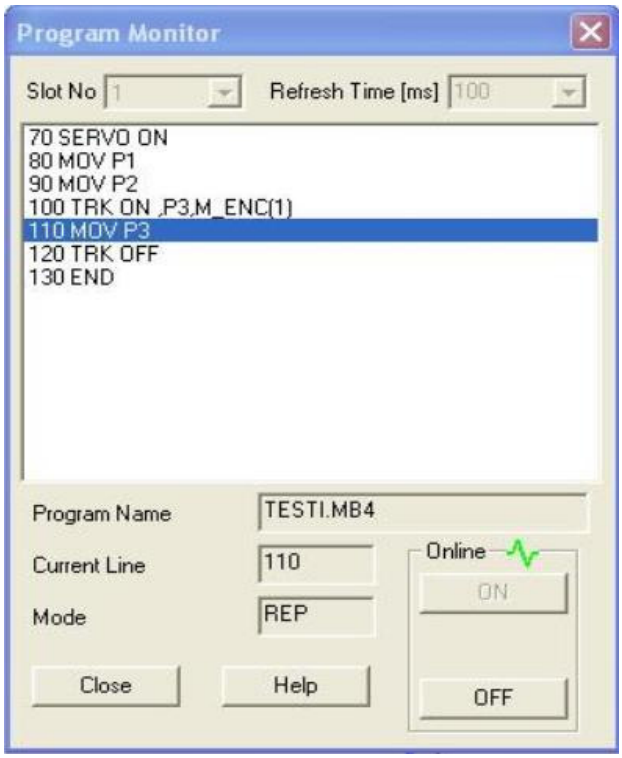
ohjelman toimintaan. Pulssitieto täytyy siis saada lisättyä jollakin muulla tavalla ohjelmaan. /7/

Manuaaleja selatessa löytyi ”Conveyor trackin funktion”-manuaalista valmis kalibrointiohjelma, joka kirjoitettiin paperiversiosta tietokoneelle./8, s.29/ Kalibrointi on selitetty myöhemmin kohdassa 3.2.4.

Kun kalibrointi oli saatu kuntoon, yritettiin lisätä kalibroitu pulssitieto edelliseen tapaan PICK-paikkatietoon, siinä kuitenkaan onnistumatta. Seuranta ei siis ollutkaan niin yksinkertainen asia, kuin ensin luultiin. Tracking-manuaalista löytyi ohjeet, miten seuranta pitäisi saada toimimaan./8, s.22/

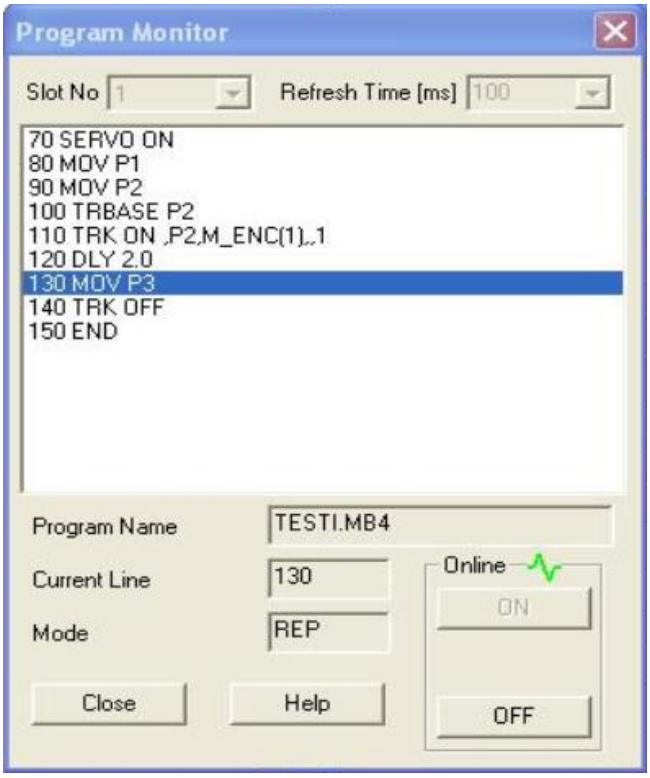
3.2.1 Tutustuminen seurantaan.

Seurantaa testattiin Testi-ohjelmassa, johon laitettiin toiminnan kannalta vain välttämättömimmät komennot. Tällä ohjelmalla Cosirob antoi virheilmoituksen jo latausvaiheessa, joten jokin ei ollut kunnossa.



Kuva 15. Toimimaton seuranta Testi-ohjelmassa.

Kun seurantaa ei omin avuin saatu toimimaan, otettiin yhteyttä robotin maahantuojaan Beijer Electronics Oy:hyn. He lähettivät erään ohjelmaesimerkin, jossa seurantaa oli käytetty hieman erilaisen robotin kanssa. Ohjelmaa tutkiessa huomattiin, että ainoat eroavaisuudet ohjelmissa seurannan osalta olivat komentojen TRBASE, jota ei oltu aiemmin käytetty ja TRK, jonka asetteluissa oli väärä paikkatieto, loppuosasta puuttui pakolliset pilkut, sekä pulssianturin järjestysnumero./8, s.22-23/



Kuva 16. Korjattu seuranta Testi-ohjelmassa.

Nyt seuranta saatiin ensimmäistä kertaa toimimaan edes jossain määrin. Havaittiin myös, että kalibrointi oli onnistunut täydellisesti. Robotti kulki seurannan aikana hyvin tarkasti kuljettimen kanssa samalla nopeudella. Seuraavana haasteena on saada seuranta toimimaan myös varsinaisessa ohjelmassa ja ohjaamaan robottia kameran antamien koordinaattien avulla.

3.2.2 Seurannan liittäminen pääohjelmaan

Seuranta lisättiin pääohjelmaan ja se alkoi toimimaan heti ensimmäisellä kerralla, kun ohjelma käynnistettiin. Tässä vaiheessa ei vielä keskitytty kappaleiden poimimiseen, vaan ainoastaan seurannan liittämiseen pääohjelmaan. Käytössä ei siis ollut ollenkaan poimittavia kappaleita, vaan toimintaa kokeiltiin pelkän robotin ja kuljettimen avulla. Täysin ongelmitta ei tämäkään osio sujunut, vaan robotin lopettaessa seurantaa ja lähtiessä kohti seuraavaa paikkapistettä, se meni vikatilaan(kohta 3.3.2). Kun ongelmat oli ratkaistu, lisättiin seuranta myös paletin poimivaan ohjelman osioon.

3.2.3 Hienosäätö

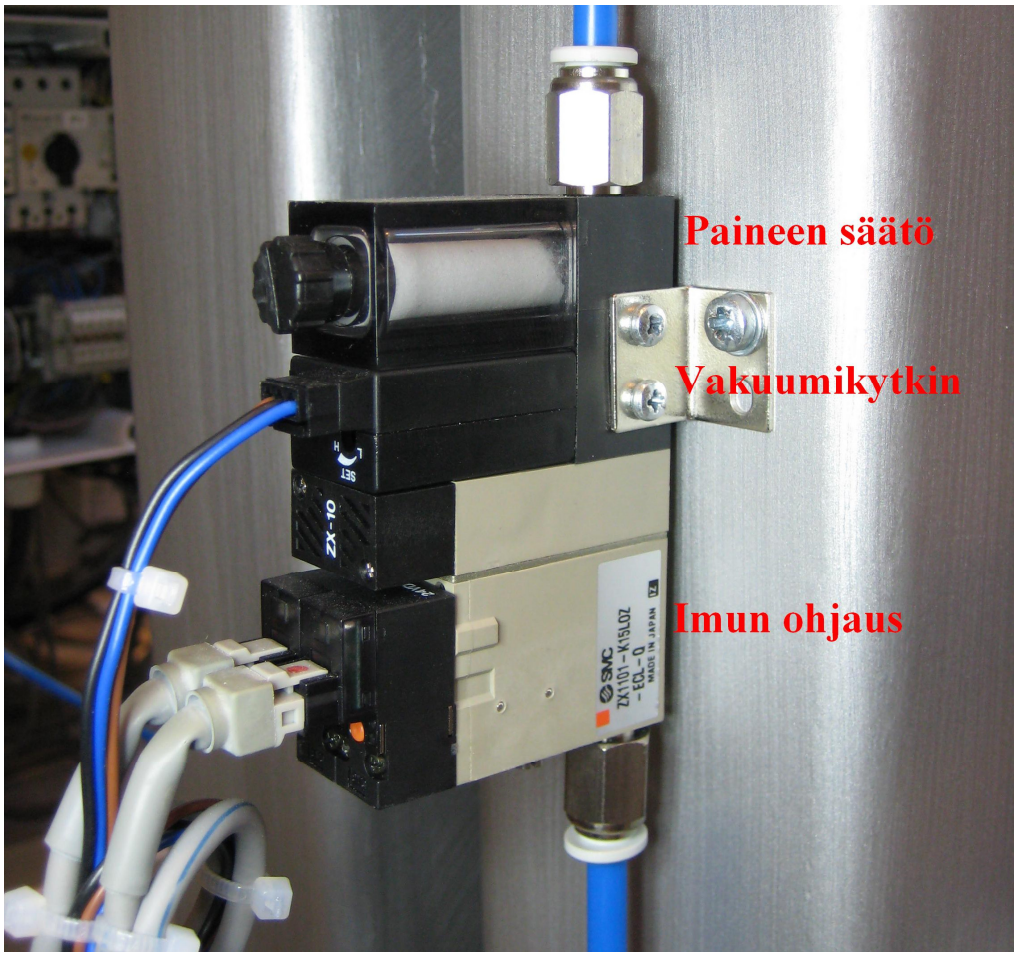
Tässä vaiheessa seuranta toimi ohjelmassa jotenkuten, mutta tarkoitus oli saada ohjelma hiottua sulavaliikkeiseksi ja muutenkin toimimaan mahdollisimman tehokkaasti.

3.2.3.1 Odotuspaikan muuttaminen.

Tähän asti robotti oli odottanut liikekäskyä kameralta positiossa P1, joka sijaitsee valkoisen kuljettimen ja mustan lajittelukuljettimen puolivälissä. Havaittiin kuitenkin, että vie liian kauan aikaa, kun robotti liikkuu nostoalueelle, joten luotiin uusi positio Jog-operaatiolla(kuva 13). Uusi positio oli nimeltään P16 ja se sijaitsi heti kuljettimen yläpuolella lähellä nostopaikkaa. Nyt nosto voidaan aloittaa nopeammin, koska robotti on jo valmiiksi lähellä.

3.2.3.2 Vakuumin tunnistus

Robotin tarttujassa oli vakuumin eli alipaineen tunnistava anturi, jonka avulla tiedettiin, milloin tarttuja on saanut kappaleesta kiinni. Tätä käytettiin hyväksi kahdessa kohtaa.



Kuva 17. Vakuumikytkin ja imun ohjauslaitteet.

Alun perin oli määritetty tarttumisaika, jonka jälkeen robotti nosti kappaleen ylös paletilta, huolimatta siitä, oliko kappaletta saatu kiinni vai ei. Koska eri paletin sektoreista nostaminen vie robotilta eri ajan, jouduttiin tämä aika asettamaan pisimmän ajan vievän sektorin mukaan. Tämä hidasti jälleen ohjelmaa ja täten koko järjestelmän toimintaa. Parannuksena tähän käytettiin vakuumin tunnistusta, joka antaa robotille tiedon heti, kun vakuumi on saatu. Tämän jälkeen robotti nostaa kappaleen välittömästi, eikä viiveitä synny muiden sektoreiden takia.

Vakuumintunnistusta käytettiin myös siinä, että robotti yrittää nostaa kappaletta vain tietyn ajan ja jos sinä aikana ei vakuumia ole saatu, palaa robotti takaisin odotuspaikkaan. Tästä on hyötyä siinä vaiheessa, jos jostain syystä robotti ei osu kunnolla kappaleen päälle, tai muusta syystä kunnollista tartuntaa ei saavuteta. Ilman tunnistusta robotti käy läpi koko ohjelman, vaikka sillä ei olisikaan kappaletta kuljetettavanaan ja taas menisi aikaa hukkaan.

3.2.3.3 Seurannan aloitus ja lopetus pulssien mukaan.

Jotta seuranta toimisi kaikilla kuljettimen pyörimisnopeuksilla, ei seurannan ajoittamista voi tehdä ajastimien kanssa, vaan kaikki odotukset tulee pohjautua pulsseihin. Ensinäkin seuranta aloitetaan vasta, kun 70 pulssia on kulunut kuvan ottamisesta. Tuon pulssimäärän jälkeen paletti on ehtinyt tulla tarpeeksi paljon pois kameran alta. Seuraavan kerran pulsseja käytetään, kun tiedustellaan vakuumikytkimeltä onko kappale saatu kiinni. Seuranta myös lopetetaan pulssiluvun saavutettua asetetun suuruinen arvo.

3.2.4 Pulssien käsittely

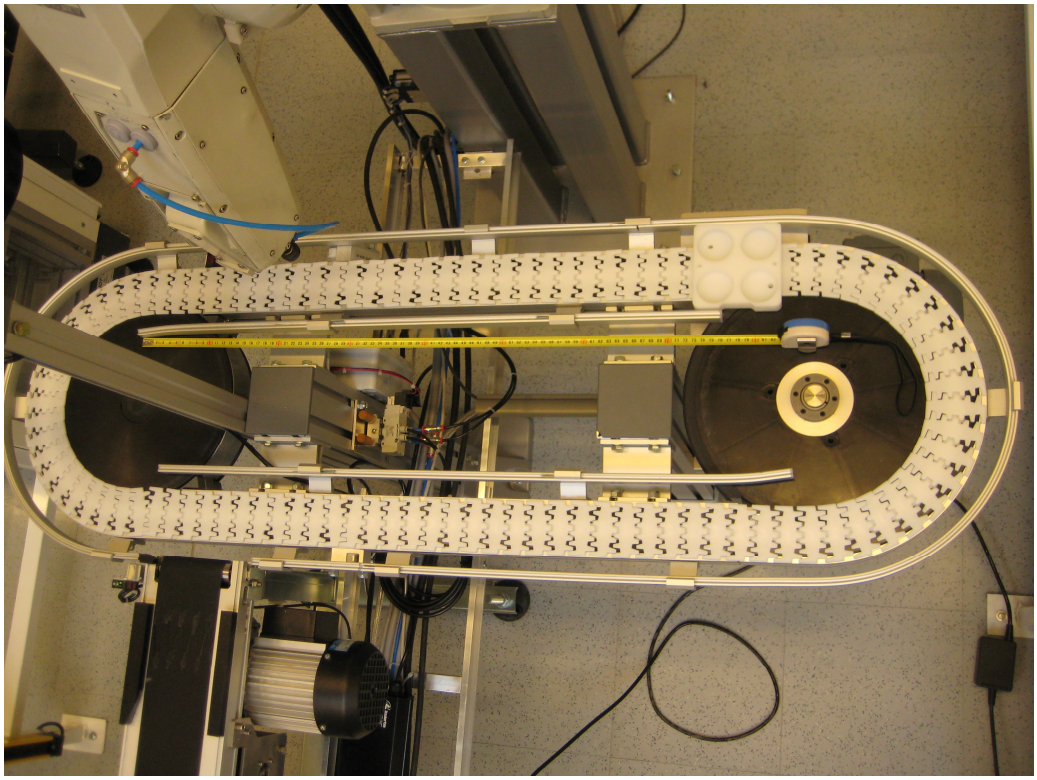
Koska pulssianturi antaa liiketiedon pulssien määrän kasvuna tai pienenemisenä ja robotti tarvitsee liikkuakseen millimetritiedon, täytyi pulssit ja millimetrit saada skaalattua yhtenäiseksi. Skaalauksen voi laskea itse käsin mitattujen tulosten perusteella, tai voi käyttää erillistä kalibrointiohjelmaa.

3.2.4.1 Manuaalinen kalibrointi

Aluksi määritettiin manuaalisesti, kuinka monta pulssia vastaa yhden millimetrin kuljettua matkaa. Paletti asetettiin kuljettimen suoran alkupäähän valokennoa lähtöviivana käyttäen. Sen jälkeen tarkistettiin alkutilanteen pulssilukema M_ENC(1) muuttujasta ja kirjattiin ylös. Kuljetinta ajettiin aivan kuljettimen suoran loppuun. Mahdollisimman pitkä ajettu matka pienentää mittavirheen merkitystä ja antaa täten luotettavamman tuloksen. Kun paletti oli suoran lopussa, pysäytettiin kuljetin ja mi-

tattiin rullamitalla paletin kulkema matka. Tämän hetkisestä pulssilukemasta vähennettiin alkutilanteen lukema, jolloin saatiin pulssien määrä kuljetulle matkalle. Kun kuljetun matkan pulssilukema jaettiin kuljetun matkan millimetreillä, saatiin tuloksena kerroin, joka muuttaa pulssit millimetreiksi kuljettimella.

Esimerkkilasku: $\frac{6163\,pulssia}{725mm} \approx 8{,}5\,{}^{pulssia}\!/_{mm}$

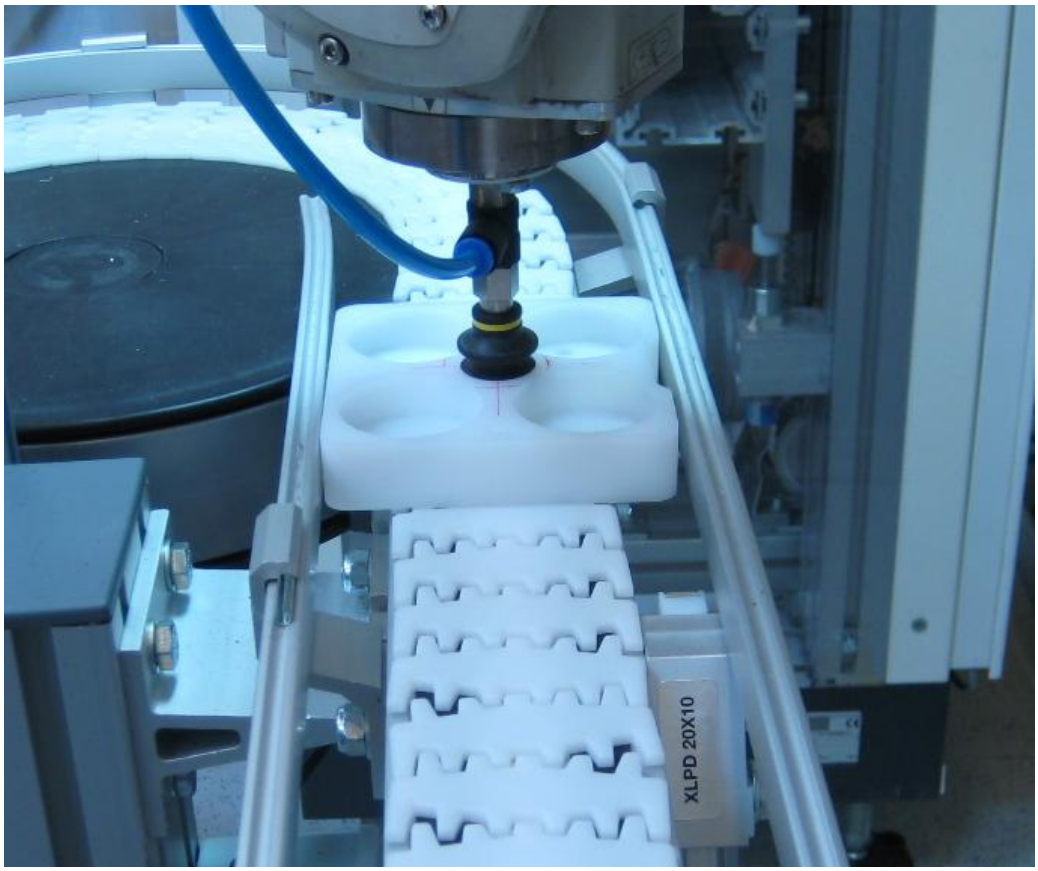


Kuva 18. Kuljetun matkan mittaus.

3.2.4.2 Ohjelmallinen kalibrointi

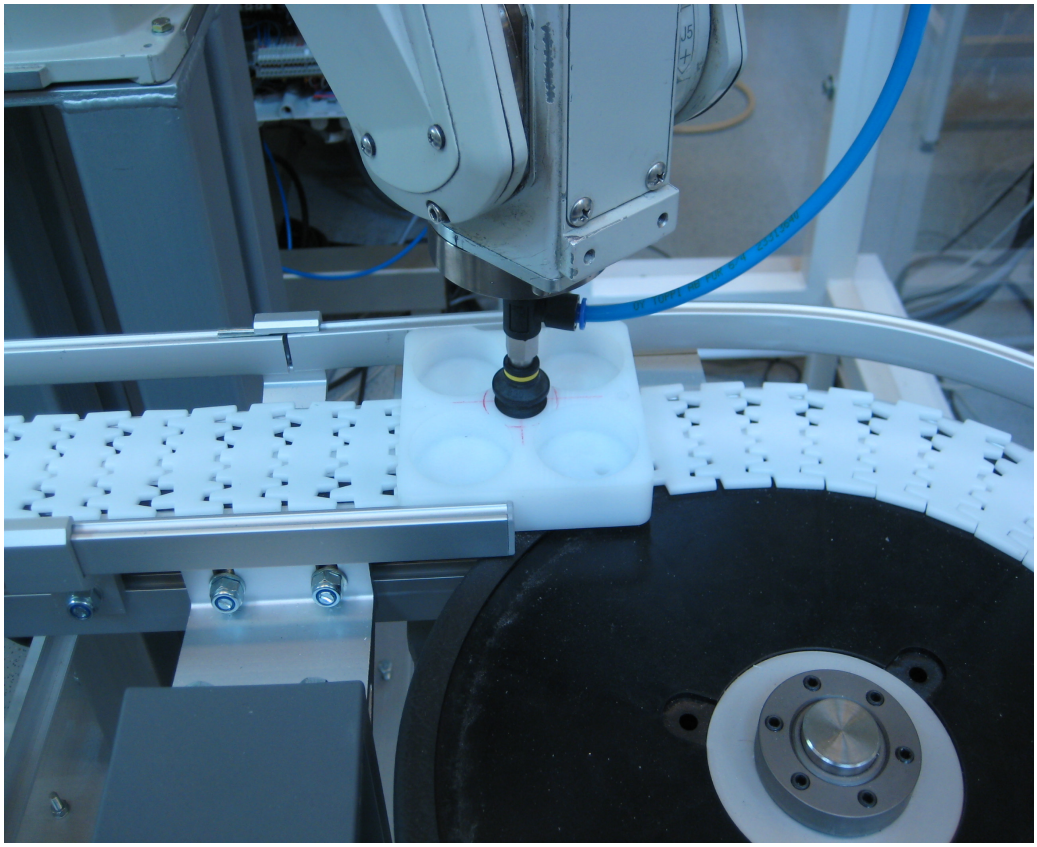
Kalibroinnin voi suorittaa myös erillisen kalibrointiohjelman avulla, joka laskee kaikkien akseleiden muutoksen erikseen. Tällöin tulos on luotettavampi ja käytettävissä myös niissä tilanteissa, joissa poikkeamaa tulee myös muiden, kuin X-akselin suhteen.

Ohjelmakoodi(Liite 2) löytyi Conveyor tracking funktion-manuaalista, josta se kopioitiin Cosirob:iin. Ohjelma aloitettiin asettamalla tyhjä paletti kuljettimen suoran alkupäähän. Tähän käytettiin erikoista palettia, jonka keskikohta on tarkasti mitattu ja merkitty palettiin. Seuraavaksi ajettiin käsiajolla robotin tarttuja tarkalleen keskelle palettia. Ohjelmaa ajettiin rivi kerrallaan ja pulssianturin sen hetkinen pulssiluku ja robotin tarkka paikkatieto tallentuivat ohjelmaan.



Kuva 19. Kalibroinnin alkutilanne.

Tämän jälkeen nostettiin robotti irti paletista ja ajettiin kuljetinta niin, että paletti liikkui kuljettimen suoran loppupäähän. Robotti ajettiin jälleen paletin keskelle ja laskettiin alas kiinni palettiin.



Kuva 20. Kalibroinnin lopputilanne

Ohjelmaan tallennettiin jälleen sen hetkinen pulssimäärä ja robotin paikkatieto. Ohjelma laski kalibrointisuhteen kuljetun matkan ja tallentuneiden pulssien perusteella jokaiselle akselille erikseen erilliseen kalibrointiparametriin P_ENCDLT.

3.3 Havaintoja ja ongelmia

Projektin aikana ilmeni useita ongelmia ja puutteita, joiden korjaaminen vei usein paljon aikaa. Aivan kaikkia ongelmia, kuten valaistusominaisuuksien vaihtelut, ei saatu tämän projektin puitteissa täysin hallintaan.

3.3.1 Kameran ongelmat

Kameraan liittyvät ongelmat johtuivat lähes poikkeuksetta valaistukseen ja valaistuksen vaihteluihin. Kirkkaalla säällä ei kameran lamppuja saanut pitää päällä lainkaan, tai muuten kamera näki jatkuvasti linjalla jotain ”mielenkiintoista” ja antoi robotille koordinaatteja olemattomista kappaleista. Tällöin oli myös ongelmia tyhjän paletin tunnistamisessa.

Parannuksen tähän ongelmaan voisi tuoda ”suoja”, joka estää ulkopuolelta tulevan valon pääsemisen kuvausalueelle. Tällöin valaistusominaisuudet säilyisivät lähes muuttumattomina ja toiminta olisi varmempaa.

Kun ulkoa tuli vähemmän valoa, kameran lamppujen täytyi olla päällä, jotta saatiin riittävä valon määrä kuvan ottamiseen. Tällöin ongelmana oli se, että kun valonlähde oli kameran ympärillä, valo heijastui kiiltävistä maalipinnoista takaisin ylivalottaen osan kuvasta. Varsinkin kamerasta katsottuna vasemmalla alhaalla olleessa paletin sektorissa ollut kappale kärsi tästä kaikkein eniten. Ylivalottuminen aiheutti sen, että kamera ei nähnytkään isoa yhtenäistä samanväristä aluetta, vaan osa tästä alueesta oli valkoinen. Tällöin kamera ei välttämättä huomioinut kappaletta lainkaan. Eniten vikoja havaittiin punaisen värisillä kappaleilla.

Korjauksen asiaan voisi tuoda kappaleiden maalaaminen mattapintaisiksi, tai valonlähteen siirtäminen niin, ettei heijastuksia tapahdu.

Välillä kesken ohjelman, kamera ei yhtäkkiä havainnutkaan määrityksen mukaisia kappaleita paletilla, vaikka niitä kulki kameran alta aivan samalla tavalla, kuin aikaisemminkin. Vika korjaantui, kun kameran lamput sammutettiin, ja sytytettiin uudelleen.

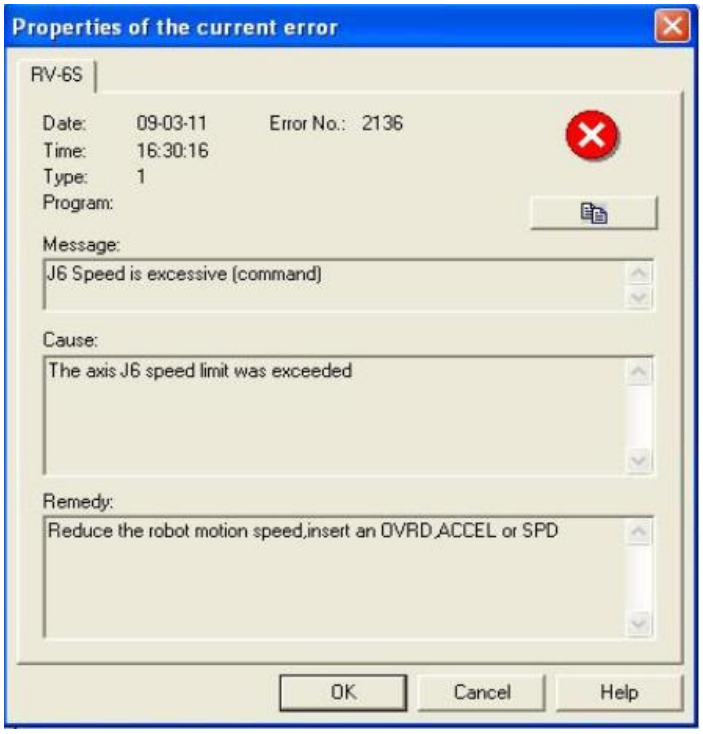
Työn ollessa muuten pääosiltaan valmis, kutsuttiin Provendorilta ohjelmoija auttamaan kameran säätämisessä. Hänen kanssaan kameran säätöjä ja ohjelmia tutkiessa löytyi kameran ohjelmasta kirjoitusvirhe. Kamerasta katsottuna vasemman alakulman sektorin tunnistusvirheeksi paljastui virhe kameran ohjelmakoodissa, jossa oli viittaus väärään muistipaikkaan. Tämän takia kamera vertaili keskenään vääriä luku-

ja, eikä niistä tullut joka kerta järkevää tulosta. Tämän takia kamera ei aina tunnistanut siinä sektorissa olevaa kappaletta.

Lisäksi ongelmia oli paletin tunnistamisessa. Koska paletti ja kuljetin ovat molemmat valkoisia, on kameran vaikea löytää paletin tarkkaa sijaintia**.** Korjauksena säädettiin kameran asetuksista valotusaikaa, sekä pikselijoukkojen reunojen terävyyttä. Lisäksi muutettiin kameran ohjelmaa siten, ettei paletin koordinaatteja enää määritetä joka kuvauksen yhteydessä, vaan ne pysyvät aina samana. Paletin sijaintihan ei voi vaihdella kuljettimella lähestulkoon ollenkaan kuvauskertojen välillä.

3.3.2 Robotin liikeongelma

Seurannan yhteydessä havaittiin ongelma, jonka vuoksi robotti meni vikatilaan aina seurannan päätyttyä. Kun kappale nostettiin pois paletilta ja annettiin heti tämän jälkeen käsky liikkua seuraavaan positioon, meni robotti vikatilaan ja antoi seuraavanlaisen virheraportin:



Kuva 21. Robotin virheraportti.

Ohjelma ilmoitti, että J6-akseli, eli tarttujan imukuppia pystyakselinsa ympäri kiertävän akselin liikenopeus on liian suuri. Akseli ei kuitenkaan silmämääräisesti näyttänyt liikkuvan lainkaan. Ohjelma antoi korjausehdotukseksi vähentää robotin nopeutta komennoilla OVRD ja SPD tai kiihtyvyyttä komennolla ACCEL, mutta mikään näistä komennoista ei tuonut muutosta tilanteeseen./7,s.(119, 199 ja 213) /

Ohjelma saatiin toimimaan, kun jatkettiin seurantaa vielä hetki nostotapahtuman jälkeenkin. Ilmeisesti kahden erisuuntaisen paikkapisteen välinen nopeahko liike, yhdistettynä seurannan lopettamiseen saa aikaan liian nopeita, hallitsemattomia liikkeitä ja tämän takia robotti menee vikatilaan.

4 YHTEENVETO

Työn valmistuttua voidaan sanoa, että se oli haasteellisempi ja monipuolisempi, kuin osattiin odottaa. Aihe oli sinällään entuudestaan lähes tuntematon ja kaiken joutui opettelemaan manuaaleista, ennen kuin mitään pystyi edes kokeilemaan onnistuneesti. Kun alun ongelmat pulssianturin kanssa saatiin ratkaistua ja seurannan peruskomennot löytyivät, alkoi työ vasta todella edetä.

Ohjelma olisi saatu toimimaan myös hieman yksinkertaisemminkin, mutta koska tarkoitus oli saada se toimimaan mahdollisimman tehokkaasti ja varmasti, jouduttiin ohjelmaan lisäämään joitakin lisäominaisuuksia: mm. vakuumintunnistus ja pulssien perusteella tahdistettu seuranta.

Lopputulos on kaikin puolin toimiva ja toteutustapa järkevä. Myös ne alueet, jotka eivät alun perin kuuluneet tähän opinnäytetyöhön, toimivat työn jälkeen moitteitta, muutamaa valaistukseen liittyvää yksityiskohtaa lukuun ottamatta, jotka voidaan korjata jonkun toisen projektin puitteissa.

LÄHTEET

/1/ BEIJER Oy, Robottimallit [verkkodokumentti]. [viitattu 16.4.2009]. Saatavissa: http://www.beijer.fi/web/web_aut_fi.nsf/AllDocuments/C125701A003AA919C1256F3A006E2941

/2/ BEIJER Oy, Ohjainyksiköt [verkkodokumentti]. [viitattu 16.4.2009]. Saatavissa:http://www.beijer.fi/web/web_aut_fi.nsf/AllDocuments/C125701A003AA919C1256F3A006EB610

/3/ Kerttu Pollari-Malmi, Ohjelmoinnin peruskurssi Y1 [verkkodokumentti]. [viitattu 23.4.2009]. Saatavissa: http://users.tkk.fi/~t106216/pruju2004/pruju2004/node4.html

/4/ PROVENDOR Oy, Konenäköjärjestelmät [verkkodokumentti]. [viitattu 16.4.2009].Saatavissa: http://www.provendor.fi/index.php?page=siemensvisyleista

/5/ OMRON, Omron Rotary Encoder E6B2-C Manual, Saatavissa: Satakunnan ammattikorkeakoulu, automaatiolaboratorio.

/6/ Robottijärjestelmän käyttöönottoraportti: Automatisation on production line, Potoczny Gaël, Thuilliez Yann. Saatavissa: Satakunnan ammattikorkeakoulu, automaatiolaboratorio.

/7/ MITSUBISHI, Mitsubishi CR2 Controller Instruction Manual, Saatavissa: Satakunnan ammattikorkeakoulu, automaatiolaboratorio.

/8/ MITSUBISHI, Mitsubishi conveyor tracking function manual, Saatavissa: Satakunnan ammattikorkeakoulu, automaatiolaboratorio.

LIITELUETTELO

LIITE 1

Pääohjelma

```
70 SERVO ON                    'Käynnistetään servot
80 ACCEL 20,20                 'Asetetaan kiihtyvyys 20% maksimista
90 OVRD M_NOVRD                'Asetetaan robotin liikenopeus normaaliarvoon
100 PORT3 = 0                  'Suljetaan tarttujan imua ohjaava portti
110 MOV P1                     'Liike pisteeseen P1
120 MOV P16                    'Liike pisteeseen P16
130 M1=0                       'Asetetaan muistipaikka M1=0(vastaa X-koordinaattia)
140 M2=0                       'Asetetaan muistipaikka M2=0(vastaa Y-koordinaattia)
150 M3=0                       'Asetetaan muistipaikka M3=0(vastaa Z-koordinaattia)
160 M4=-180                    'Asetetaan muistipaikka M4=-180(vastaa A-koordinaattia)
170 M5=0                       'Asetetaan muistipaikka M5=0   (vastaa B-koordinaattia)
180 M6=-100                    'Asetetaan muistipaikka M6=-100(vastaa C-koordinaattia)
190 M7=0                       'Asetetaan muistipaikka M7=0(kertoo nostettavan kappaleen värin ja
                               sijainnin paletilla)
200 DEF IO PORT1 = BIT,9       'Määritetään PORT1 komento ohjaamaan bittiä 9(Kuljettimen hidas
                               nopeus)
210 PORT1 = 1                  'Asetetaan PORT1=1(Käynnistetään kuljetin hitaalla nopeudella)
220 DEF IO PORT4 = BIT,10      'Määritetään PORT4 komento ohjaamaan bittiä 10(Kuljettimen nopea
                               nopeus)
230 PORT4 = 1                  'Asetetaan PORT4=1(Asetetaan kuljettimen nopeus normaaliksi)
240 OPEN "COM3:" AS #1         'Avataan COM3, joka vastaanottaa SIMATIC:in STRING-muotoisen
                               tiedoston numero 1

 250 INPUT #1,M1,M2,M7,M3      'Luetaan tiedosto 1 ja asetetaan sen ensimmäinen arvo
                               muistipaikkaan M1, toinen M2:n, kolmas M7:n ja neljäs M4:n
260 M_ENC(1) = 0               'Nollataan pulssianturin muuttuja
270 OVRD M_NOVRD               'Asetetaan robotin liikenopeus normaaliarvoon
280 MOV P16                    'Liike pisteeseen P16
290 CLOSE #1                   'Suljetaan tiedosto 1
300 IF M7 = 0 THEN GOTO 240    'Jos M7=0, siirrytään riville 240
```

```
310 DEF POS PICK                    'Määrietään positio PICK(akseli kerrallaan) kameran antamien
                                    koordinaattien avulla
320 PICK.X=M1+210                   'Asetetaan X-akselin arvoksi M1 ja lisätään siihen 210mm
330 PICK.Y=M2                       'Asetetaan Y-akselin arvoksi M2
340 PICK.Z=M3                       'Asetetaan Z-akselin arvoksi M3
350 PICK.A=M4                       'Asetetaan A-kulman arvoksi M4
360 PICK.B=M5                       'Asetetaan B-kulman arvoksi M5
370 PICK.C=M6                       'Asetetaan C-kulman arvoksi M6
380 DEF POS PICKUP                  'Määritetään positio PICKUP(akseli kerrallaan) kameran
                                    antamien koordinaattien avulla
390 PICKUP.X=M1+210                 'Asetetaan X-akselin arvoksi M1 ja lisätään siihen 210mm
400 PICKUP.Y=M2                     'Asetetaan Y-akselin arvoksi M2
410 PICKUP.Z=M3+30                  'Asetetaan Z-akselin arvoksi M3 ja lisätään siihen 30mm
420 PICKUP.A=M4                     'Asetetaan A-kulman arvoksi M4
430 PICKUP.B=M5                     'Asetetaan B-kulman arvoksi M5
440 PICKUP.C=M6                     'Asetetaan C-kulman arvoksi M6
450 'DEF IO PORT2 = BIT,5           'Määritetään PORT2 komento ohjaamaan bittiä 5(stoppari)
                                    (ei käytössä)
460 'PORT2 =1                       'Asetetaan stoppari päälle (ei käytössä)
 470 DEF IO PORT3 = BIT,6           'Määritetään PORT3 komento ohjaamaan bittiä 6(imu)
 480 IF M7 = 13 THEN GOTO 730       'Jos M7=13, siirrytään riville 730
 490 IF M7 = 14 THEN GOTO 730       'Jos M7=14, siirrytään riville 730
 500 MOV P16                        'Liike pisteeseen P16
 510 WAIT M_ENC(1)>70               'Odotus, kunnes M_ENC(1) (pulssianturin lukema) >70
 520 MOV PICKUP   'P2               'Liike pisteeseen PICKUP
 530 TRBASE PICKUP                  'Asetetaan seurannan lähtöasemaksi piste PICKUP
  540 J10=PTOJ(PICKUP)              'Hajotetaan pisteen PICKUP paikkatieto akselitiedoksi J10
  550 J10.J6=J_CURR.J6              'Määritetään J10.J6 akselin arvoksi sen nykyinen arvo
  560 TRK ON ,PICKUP,M_ENC(1),,1 'Seuranta päälle, PICKUP=aloituspiste,
                                    M_ENC(1)=pulssitiedon lähde, 1=pulssianturin järjestysnume-
                                    ro
  570 M_ENC(1)=0                    'Nollataan pulssien määrä
  580 MOV J10                       'Siirrytään pisteeseen J10
  590 MOV PICK                      'Siirrytään pisteeseen PICK
```

```
600 PORT3 = 1                          'Asetetaan PORT3=1 (imu päälle)
610 WAIT M_ENC(1)>1300                 'Odota, kunnes pulssiluku >1300
620 IF M_IN(6)=0 THEN GOTO 70          'Jos tulobitti 6=0, siirrytään riville 70
650 MOV J10                            'Liike pisteeseen J10
670 WAIT M_ENC(1)>2400                 'Odota, kunnes pulssiluku >2400
680 TRK OFF                            'Lopetetaan seuranta
700 PORT2 = 0                          'Asetetaan PORT2=0 (lopeta imu)
710 MOV P1                             'Liike pisteeseen P1
720 GOTO 910                           'Siirry riville 910
730 MOV P16                            'Liike pisteeseen P16
740 WAIT M_ENC(1)>70                   'Odota, kunnes pulssiluku >70
750 MOV PICKUP                         'Liike pisteeseen PICKUP
760 TRBASE PICKUP                      'Asetetaan seurannan lähtöasemaksi piste PICKUP
770 J10=PTOJ(PICKUP)                   'Hajotetaan pisteen PICKUP paikkatieto akselitiedoksi J10
780 J10.J6=J_CURR.J6                   'Määritetään J10.J6 akselin arvoksi sen nykyinen arvo
790 TRK ON ,PICKUP,M_ENC(1),,1 'Seuranta päälle, PICKUP=aloituspiste,
                               M_ENC(1)=pulssitiedon lähde, 1=pulssianturin järjestysnume-
                               ro
800 M_ENC(1)=0                         'Nollataan pulssien määrä
810 MOV J10                            'Siirrytään pisteeseen J10
820 MOV PICK                           'Siirrytään pisteeseen PICK
830 PORT3 = 1                          'Asetetaan PORT3=1 (imu päälle)
840 WAIT M_ENC(1)>1300                 'Odota, kunnes pulssiluku >1300
850 IF M_IN(6)=0 THEN GOTO 70          'Jos tulobitti 6=0, siirrytään riville 70
860 MOV J10                            'Liike pisteeseen J10
870 WAIT M_ENC(1)>2400                 'Odota, kunnes pulssiluku >2400
880 TRK OFF                            'Lopetetaan seuranta
890 PORT2 = 0                          'Asetetaan PORT2=0 (lopeta imu)
900 MOV P1                             'Liike pisteeseen P1
910 IF M7 = 1 THEN CALLP"DROPRED"          'Jos M7=1, kutsu aliohjelma "DROPRED"
920 IF M7 = 2 THEN CALLP"DROPBLUE"         'Jos M7=2, kutsu aliohjelma "DROPBLUE"
930 IF M7 = 3 THEN CALLP"DROPGREEN"        'Jos M7=3, kutsu aliohjelma "DROPGREEN"
940 IF M7 = 4 THEN CALLP"DROPRED"          'Jos M7=4, kutsu aliohjelma "DROPRED"
950 IF M7 = 5 THEN CALLP"DROPBLUE"         'Jos M7=5, kutsu aliohjelma "DROPBLUE"
```

```
960 IF M7 = 6 THEN CALLP"DROPGREEN"      'Jos M7=6, kutsu aliohjelma "DROPGREEN"
970 IF M7 = 7 THEN CALLP"DROPRED"        'Jos M7=7, kutsu aliohjelma "DROPRED
980 IF M7 = 8 THEN CALLP"DROPBLUE"       'Jos M7=8, kutsu aliohjelma "DROPBLUE"
990 IF M7 = 9 THEN CALLP"DROPGREEN"      'Jos M7=9, kutsu aliohjelma "DROPGREEN"
1000 IF M7 = 10 THEN CALLP"DROPRED"      'Jos M7=10, kutsu aliohjelma "DROPRED
1010 IF M7 = 11 THEN CALLP"DROPBLUE"     'Jos M7=11, kutsu aliohjelma "DROPBLUE"
1020 IF M7 = 12 THEN CALLP"DROPGREEN"    'Jos M7=12, kutsu aliohjelma "DROPGREEN"
1030 IF M7 = 13 THEN CALLP"DROPPALET1"   'Jos M7=13, kutsu aliohjelma "DROPPALET1"
1040 IF M7 = 14 THEN CALLP"DROPPALET2"   'Jos M7=14, kutsu aliohjelma "DROPPALET2"
1050 GOTO 240                            'Siirry riville 240
1060 END                                 'Lopeta ohjelma
```

LIITE2

Kalibrointiohjelma

```
70 '(1)STICK THE MARKING SEAL TO THE UPPER COURSE OF THE CONVEYOR
  'Laitetaan kohdistuspaletti kuljettimen alkupäähän

80 '(2)MOVE THE ROBOT TO CENTER OF THE SEAL
  'Ajetaan käsiajolla tarttuja paletin keskelle

90 MX10EC1# = M_ENC(1) 'GET THE ENCODER DATA 1
  'Otetaan 1. pulssitieto muistiin

100 PX10PS1=P_FBC   'GET THE POSITION 1
'Otetaan robotin 1.paikkatieto muistiin

120 '(3)MOVE ROBOT UP
'Nosta robotti ylös paletin päältä

130 '(4)MOVE CONVEYOR FORWARD
'Aja kuljetinta suoran loppuun

140 '(5)MOVE THE ROBOT TO CENTER THE SEAL AGAIN
  'Aja käsiajolla robotti jälleen paletin keskelle

150 MX10EC2#=M_ENC(1) 'GET THE ENCODER DATA 2
  'Otetaan 2.pulssitieto muistiin

160 PX10PS2=P_FBC 'Get the position 2
'Otetaan robotin 2.paikkatieto muistiin

170 P_101(1)=PX10PS2
'Siirretään paikkatieto toiseen muistipaikkaan
```

```
180 '(6)Move the robot up
'Nosta robotti ylös

190 '(7)Executeby step execution till END
'Aja ohjelma askel kerrallaan loppuun

200 GOSUB *S10ENC 'Calculation processing of P_ENCDLT
  'Siirrytään aliohjelmaan, jossa lasketaan P_ENCDLT (kalibrointiparametri)

210 P_ENCDLT=PY10ENC
'Asetetaan laskutoimituksen arvo kalibrointiparametriin

220 END
'Lopetus

240 '##### Calculation processing on P_ENCDLT#####

250       'MX10EC1:Encoder data 1
'Nimetään 1.pulssitieto

260       'MX10EC2:Encoder data 2
'Nimetään 2.pulssitieto

270       'PX10PS1:Position 1
'Nimetään 1.paikkatieto

280       'PX10PS2:Position 2
'Nimetään 2.paikkatieto

290       'Py10ENC:Value of P_ENCDLT
'Nimetään kalibrointiparametri

300 *S10ENC
```

```
'Aliohjelma alkaa

310  M10ED#=MX10EC2#-MX10EC1
'Lasketaan pulssien määrä

320  IF M10ED#>800000000.0 THEN MD10ED#=M10ED#-1000000000.0
  'Jos pulssimäärä>800000000, vähennetään siitä 1000000000 pulssia

330  IF M10ED#<-800000000.0 THEN MD10ED#=M10ED#+1000000000.0
  'Jos pulssimäärä<800000000, lisätään siihen 1000000000 pulssia

340 PY10ENC.X=(PX10PS2.X-PX10PS1.X)/M10ED#
'Lasketaan X-akselitieto

350 PY10ENC.Y=(PX10PS2.Y-PX10PS1.Y)/M10ED#
'Lasketaan Y-akselitieto

360 PY10ENC.Z=(PX10PS2.Z-PX10PS1.Z)/M10ED#
'Lasketaan Z-akselitieto

370 PY10ENC.A=(PX10PS2.A-PX10PS1.A)/M10ED#
'Lasketaan A-kulmatieto

380 PY10ENC.B=(PX10PS2.B-PX10PS1.B)/M10ED#
'Lasketaan B-kulmatieto

390 PY10ENC.C=(PX10PS2.C-PX10PS1.C)/M10ED#
'Lasketaan C-kulmatieto

400 END
'Lopetus
```

LIITE3

Kappaleen pudotusohjelma

```
70 SERVO ON                    'Käynnistetään servot
80 MOV P1                      'Liike pisteeseen P1
90 MOV P9                      'Liike Pisteeseen P9
100 MOV P10                    'Liike Pisteeseen P10
110 MOV P11                    'Liike Pisteeseen P11
120 MOV P12                    'Liike Pisteeseen P12
130 DLY 0.5                    'Viive 0,5s
140 DEF IO PORT1 = BIT,6       'Määritetään PORT1 komento ohjaamaan bittiä 6
150 PORT1=2                    'Asetetaan PORT1 = 2
160 MOV P13                    'Liike Pisteeseen P13
170 MOV P14                    'Liike Pisteeseen P14
180 MOV P1                     'Liike Pisteeseen P1
190 MOV P16                    'Liike Pisteeseen P16
200 END                        'Lopetus
```