Ilmari Ali-Jaakkola

Hitsausprosessin 3D-simulointi PLC-rajapinnalla

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Kone- ja tuotantotekniikka

Insinöörityö

6.5.2016

Metropolia

Tekijä Otsikko Sivumäärä Aika	Ilmari Ali-Jaakkola Hitsausprosessin 3D simulointi PLC rajapinnalla 52 sivua + 2 liitettä 6.5.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Kone- ja tuotantotekniikka
Suuntautumisvaihtoehto	Koneautomaatio
Ohjaajat	Lehtori Heikki Paavilainen Tekninen myyntipäälikkö Samuli Ahonen

Tämän insinöörityön tilaajana oli Visual Components Oy, joka tuottaa tehdassimulointi-ohjelmia. Työn tarkoituksena oli simuloida hitsausprosessi, jota ohjataan PLC:llä, ja tehdä tutoriaali PLC:n liittämisestä simulaatioon.

Työssä tarkastellaan tehdassimulaatioita sekä niihin usein yhdistettyä etäohjelmointia, virtuaalista käyttöönottoa ja PLC:tä. Lisäksi tutustutaan 3DAutomate tehdassimulaatio-ohjelman käyttämiseen, josta käydään läpi kappaleen luominen ja muokkaaminen sekä simulaation rakentaminen ja yhdistäminen PLC:hen.

Tulokseksi saatiin simulaatio taka-akselin hitsausprosessista, jossa työntekijä tuo akselin kääntyvälle hitsauspöydälle ja jossa robotti hitsaa akselin päälisuojan ja sivukiinnikkeet. Simulaatiosta saatiin toimivat versiot sekä PLC:llä että ilman. Lisäksi PLC:n yhdistämisestä saatiin selkeä tutoriaali.

Työssä valmistunutta simulaatioita käytetään esimerkkikappaleena simulaatio-ohjelman käyttömahdollisuuksista ja PLC:n liittämisestä. Tutoriaalia käytetään koulutustarkoitukseen.

Avainsanat	Tehdassimulointi, PLC, OLP, virtuaalinen käyttöönotto

Author Title Number of Pages Date	Ilmari Ali-Jaakkola Making a Simulation of a Welding Process with PLC Interface 52 pages + 2 appendices 6 May 2016
Degree	Bachelor of Engineering
Degree Programme	Mechanical Engineering
Specialisation option	Machine Automation
Instructors	Heikki Paavilainen, Senior Lecturer Samuli Ahonen, Technical Account Manager

This Bachelor's thesis was commissioned by Visual Components Ltd, which produces factory simulation programs. The purpose of this thesis was to make a simulation of a welding process, which is controlled by PLC, and make a tutorial for connecting the PLC to simulation.

This thesis was carried out using the 3DAutomate factory simulation program. With the program it was studied how to create and modify components and how to connect simulation to PLC. Also factory simulation, offline programming, virtual commissioning and PLC were studied.

As a result, a simulation of a process to weld a rear axle was made. In the process a worker brings the axle to a turning welding table, where a robot welds the axle's cover and side fasteners. The simulation was made to work with and without PLC. Also a clear tutorial for connecting the PLC was made. The simulation is used to demonstrate the accessibility of the simulation program and its PLC connectivity. The tutorial is used as training material.

Keywords	Factory simulation, PLC, OLP, Virtual Commissioning

Sisällys

Lyhenteet

FDB	Function Block Diagram. PLC:n ohjelmointikieli.
IL	Instruction List. PLC:n ohjelmointikieli.
LD	Ladder Diagram. PLC:n ohjelmointikieli.
OEM	Original Equipment Manufacturer. Alkuperäislaitevalmistaja.
OLP	Offline programming. Robotin etäohjelmointi.
PLC	Programmable logic controller. Ohjelmoitava logiikka.
PnP	Plug and Play. 3DAutomate-ohjelman työkalu.
SCF	Sequential Function Chart. PLC:n ohjelmointikieli.
ST	Structured Text. PLC:n ohjelmointikieli.

1 Johdanto

Tämän insinöörityön tilaajana on Visual Components Oy, joka tuottaa tehdassimulointiohjelmia. Tehdassimulointiohjelmalla luodaan tarkkoja malleja tuotantolinjoista, roboteista ja robottien prosesseista.

Työn tarkoituksena on simuloida hitsausprosessi Visual Components Oy:n tehdassimulointiohjelmalla. Prosessissa tulee olla ohjelmoitava ja kääntyvä hitsauspöytä sekä robotti, joka suorittaa hitsauksen. Hitsauspöytää ja robottia on tarkoitus ohjata PLC:n avulla. PLC:nä käytetään Beckhoff Oy:n TwinCat 3 -ohjelmaa. Tarkoitus on myös tehdä tutoriaali PLC:n ja simulaation yhdistämisestä.

Simulaatiota on tarkoitus käyttää esimerkkikappaleena simulaatio-ohjelman käyttömahdollisuuksista ja PLC:n liitettävyydestä. Tutoriaali tulee koulutustarkoitukseen opettamaan PLC:n yhdistäminen simulaatioon PLC Add-On -lisäosalla.

Visual Components Oy on vuonna 1999 kokeneiden 3D-simulaatioasiantuntijoiden perustama suomalais-yhdysvaltalainen tehdassimulointiohjelmia valmistava yritys. Yrityksen toimitusjohtajana toimii Juha Renfors. Tällä hetkellä Visual Components Oy:ssa työskentelee 29 työntekijää, jotka ovat jakautuneet hallintoon, myyntiin ja markkinointiin, tekniseen tukeen sekä tuotekehitykseen. [1]

Yrityksen pääkonttori sijaitsee Leppävaarassa Espoossa ja sillä on toimipiste Michiganissa Yhdysvalloissa sekä Münchenissä Saksassa. Tämän lisäksi yrityksellä on yhteistyökumppaneita alkuperäislaitevalmistajina (OEM), insinöörikonsultteina ja jälleenmyyjinä Euroopassa, Yhdysvalloissa ja Aasiassa. Merkittävinä asiakkaina yrityksellä ovat Kuka Oy, Flexlink Oy ja Delfoi Oy. Näistä yrityksistä Visual Components toimittaa alustan Kuka Oy:n Kuka sim - ja Flexlink Oy:n Design Tool -simulointiohjelmille. Visual Components julkaisee aktiivisesti tutoriaaleja yhteisönsä verkkosivuilla ja pitää jatkuvasti asiakkailleen koulutuksia ohjelmiensa käytöstä. [1]

2 Tehdassimulointi

2.1 Periaate

Tehdassimulointi on 3D-simulaation luomista tehtaasta, tuotantolinjasta tai yksittäisestä robottiprosessista. Toisin kuin normaalin 3D-mallin tai piirroksen tekemisessä, jossa pyritään luomaan vain havainnollistava malli, tehdassimuloinnissa simulaatiosta pyritään tekemään mahdollisimman tarkka, jotta nähtäisiin toimiiko prosessi ennen kuin isoja hankintoja oikeista osista on tehty. Tarkkuuden saavuttamiseksi kaikkien simulaation osien on toimittava mahdollisimman todenmukaisesti. Kun riittävä tarkkuus on saavutettu simulaatiosta saatava statistiikka auttaa näkemään eri osien käyttöasteen, jaksonajat, ulottuvuudet, törmäyskohdat, pullonkaulat, yms. Tehdassimulointia käytetään muun muassa myyntitarkoitukseen näyttämään asiakkaille valmiita ratkaisumalleja, virtuaalisena käyttöönottona (Virtual Comissioning) ja robotin etäohjelmointiin (Offline Programming, OLP). [2]

2.2 Virtuaalinen käyttöönotto

Virtuaalisen käyttöönoton tarkoituksena on yhdistää tehtaasta tehty simulaatio oikeisiin tehtaan ohjausjärjestelmiin, esimerkiksi PLC:n avulla. Näin voidaan testata suunnitelma ja ohjausjärjestelmä ennen kuin yhtään fyysistä asennusta on tehty. Lisäksi tehtaan yleisiä toimintoja voidaan varmistaa toimiviksi ja huomataan virheitä, jotka muuten tulevat esille vasta, kun oikeaa käyttöönottoa tehdään. Tällä pyritään vähentämään käyttöönottoon kuluvaa aikaa ja ylimääräisiä kustannuksia. [3]

2.3 Robotin etäohjelmointi

Robotin etäohjelmointi tarkoittaa robottien ohjelmointia simulaation avulla tuotannon ulkopuolella tuotantoa pysäyttämättä. Etäohjelmoinnin aikana saadaan korjattua ohjelman virheet, jotka muuten jouduttaisiin korjaamaan ohjelman käyttöönotosta johtuvan tuotantoseisokin aikana. Lisäksi koska ohjelma on valmiiksi tehty, seisokkiin kuluva aika menee lähinnä ohjelman syöttämiseen ja viimeistelytesteihin. Yleisesti etäohjelmointia käytetään luomaan robotin työkalun liikeratoja, havaitsemaan mahdollisia yhteentörmäyksiä sekä tarkistamaan robotin nivelten rajoja, nopeuksia ja kiihtyvyyksiä. [4]

2.4 PLC

PLC, eli ohjelmoitava logiikka, on teollisuuskäyttöinen tietokoneohjain, joka jatkuvasti tarkkailee tulolaitteiden tilaa ja tekee PLC:hen asetetun ohjelman perusteella päätöksen lähtölaitteiden tilasta. Suurin hyöty PLC:n käytöstä on sen ominaisuus muuttua ja toistaa operaatio tai prosessi samalla keräten tärkeää tietoa. Toinen PLC-järjestelmän etu on, että se on modulaarinen, eli sen voi muokata ja yhdistää haluttuun käyttötarkoitukseen parhaiten sopiviin tulo- ja lähtölaitteisiin. [5]

PLC koostuu tulosta, prosessorista ja lähdöstä. Prosessori sisältää sisäisen ohjelman, joka kertoo, kuinka suorittaa seuraavat tehtävät:

- Suorittaa käyttäjän ohjelman sisällä olevat hallintaohjeet. Tämä ohjelma tallennetaan pysyvään muistiin, eli ohjelma ei häviä, jos laitteen virta kytketään päältä.

- Keskustelee muiden laitteiden kanssa, kuten I/O-laitteet, ohjelmointilaitteet (tietokone), tietoverkot ja muut PLC:t.

- Suorittaa ylläpitotehtäviä, kuten sisäistä diagnosointia.

2.4.1 PLC:n tehtäväsykli

PLC suorittaa neljä tehtävää aina ennalta määritetyn ajan välein:

1. Tulojen skannaus: havaitsee kaikkien tulolaitteiden tilat, jotka on kiinnitetty PLC:hen.

2. Ohjelman skannaus: suorittaa käyttäjän luoman ohjelman logiikan.

3. Lähtöjen skannaus: lähettää lähtölaitteille ohjelmassa luodut tiedot.

4. Ylläpito: ylläpitää muun muassa viestintää ohjelmointipäätteiden kanssa ja sisäistä diagnostiikkaa. [5]

2.4.2 PLC:n ohjelmointikielet

PLC:n ohjelmat yleensä kirjoitetaan tietokoneelle asennetulle sovelluksella, josta ohjelma tallennetaan PLC:hen. Yleisesti PLC:n ohjelmointikielinä käytetään standardin IEC 61131-3 alaisia kieliä, joista yleisimmät käydään seuraavaksi läpi.

Ladder Diagram (LD), eli perinteinen tikapuulogiikka, on graafinen ohjelmointikieli. Alun perin ohjelmoitu yksinkertaisilla yhteyksillä, jotka näyttivät releiden avautumiset ja sulkeutumiset. LD on myöhemmin laajentunut sisältämään myös laskureita, ajastimia ja matemaattisia operaatioita.

Function Block Diagram (FDB) on graafinen ohjelmointikieli, joka kuvaa signaali- ja tietovirtoja uudelleenkäytettävien toimintopalikoiden kautta. FDB on hyödyllinen näyttämään ohjussysteemin algoritmien ja logiikoiden yhteyksiä.

Structured Text (ST) on korkean tason tekstipohjainen ohjelmointikieli, jolla on PASCAL-ohjelmointikieltä muistuttava sanastorakenne. ST tukee laajan valikoiman perusfunktioita ja -operaattoreita.

Instruction List (IL) on tekstipohjainen alhaisen tason Assembly-ohjelmointikielen tyylinen kieli.

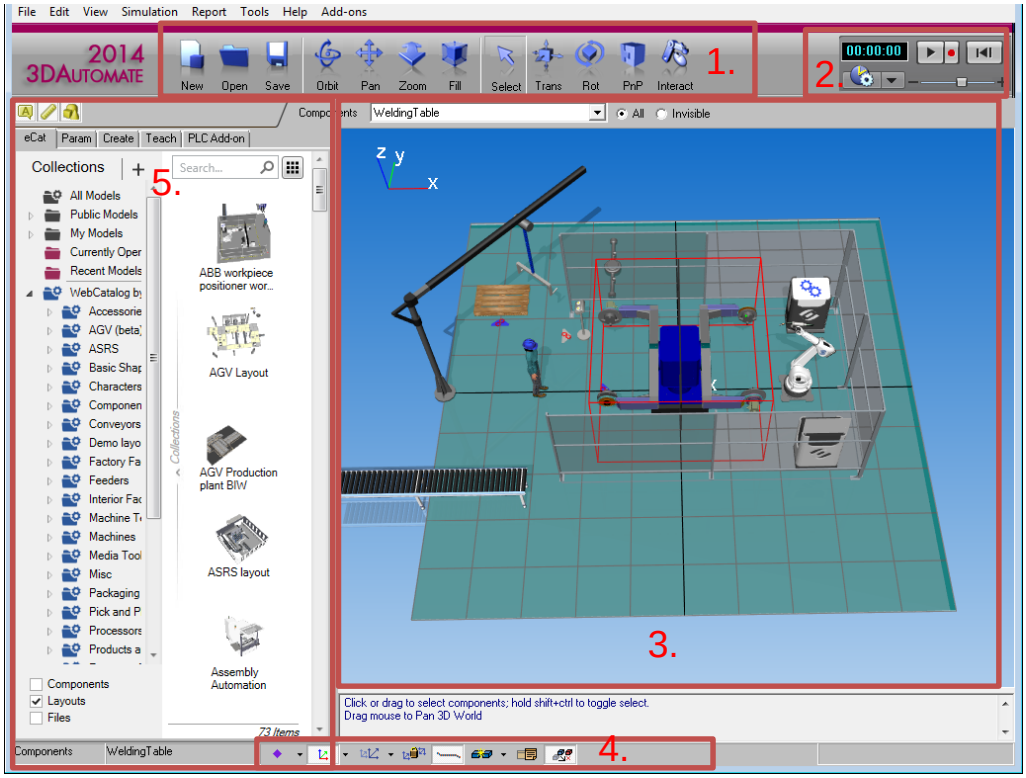
Sequential Function Chart (SCF) on yleisnäkymä ohjausjärjestelmästä, jossa perus rakennuspalikat ovat kokonaisia ohjelmatiedostoja. Jokainen ohjelmatiedosto on luotu käyttäen muun tyyppistä ohjelmointikieltä. SCF:ssä ohjataan laajat ja monimutkaiset ohjelmointitehtävät pienemmiksi helpommin hallittaviksi tehtäviksi. [5]

3 3DAutomate

Visual Componentsilla on viisi erilaista tehdassimulaatio-ohjelmaa, joiden ominaisuuksia on rajoitettu käyttötarkoituksien mukaan. Tässä insinöörityössä käsitellään kolmannen sukupolven vuonna 2014 julkaistua 3DAutomate-ohjelmaa, jossa on kaikki ominaisuudet. Lisäksi ohjelmaan lisättiin PLC Add-On -lisäosa PLC-rajapintojen luomiseen.

3DAutomaten yleisnäkymä on kuvassa 1. Yleisnäkymän ylälaidassa on pikatyökalut tiedostojen, näkymän ja komponenttien muokkaamiseen. Oikeassa yläkulmassa on ohjaimet simulaation ohjaamiseen. Keskellä on simulaatiopohja, johon simulaatiot rakennetaan. Alhaalla on painikkeita erilaisten osien näkyvyyden säätöön. Vasemmalla on toimintovälilehtiä suorittamaan erilaisia toimintoja.

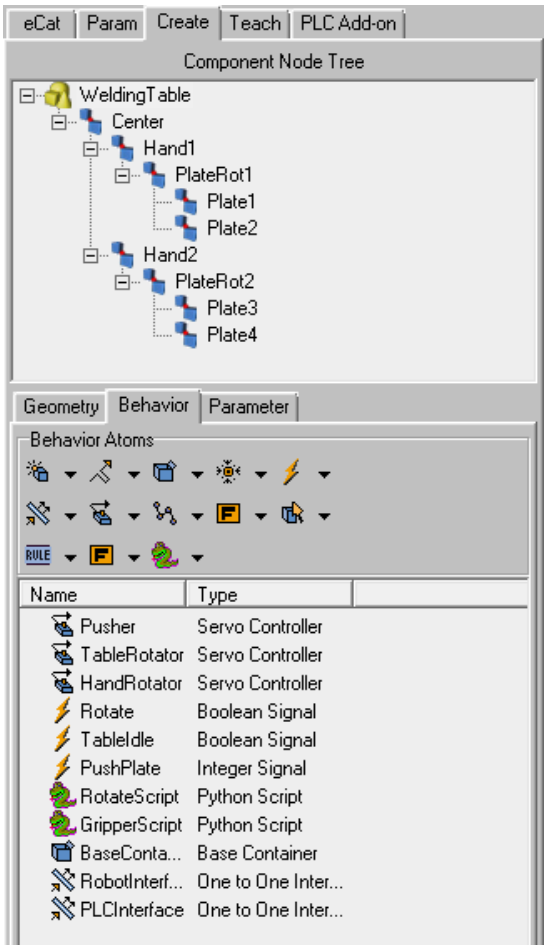
Toimintovälilehdistä eCat-välilehdellä on kirjastoja valmiiksi luoduista komponenteista ja asetteluista, joita voi tuoda simulaatiopohjaan ja yhdistää toisiinsa.



Kuva 1. 3DAutomate yleisnäkymä: 1. pikatyökaluja 2. simulaation ohjaimet 3. simulaatiopohja 4. osien näkyvyyksien säätö 5. toimintovälilehdet.

Param-välilehdellä voi muokata yksittäisten komponenttien ominaisuuksia. Pelkästään eCat- ja Param-välilehteä käyttäen saadaan luotua toimivia simulaatioita yleisistä prosesseista.

Create-välilehdellä (kuva 2) voi luoda uusia komponentteja tai muokata valmiita. Välilehdessä olevassa Component Node Treessä ylimpänä on päänode, jonka alirakenteeseen luodaan linkkejä. Nämä erotellaan, jotta voitaisiin määritellä eri tehtäviä eri osille. Component Node Treen alla on Geometry-, Behavior, ja Parameter-välilehdet, joista Geometry-välilehdellä luodaan muotoja, piirteitä, frameja eli pisteitä joihin voidaan määritellä tapahtumia, sekä liikutellaan muotoja. Behavior-välilehdellä luodaan komponenteille tehtäviä, liikuttajia, rajapintoja, signaaleja, yms. Parameter-välilehdellä komponentille luodaan ominaisuuksia, joita voi muokata Param-välilehdellä. Teach-välilehdellä luodaan ja toteutetaan roboteille ohjelmia. PLC Add-on -välilehdellä luodaan rajapinta PLC-ohjelman kanssa.



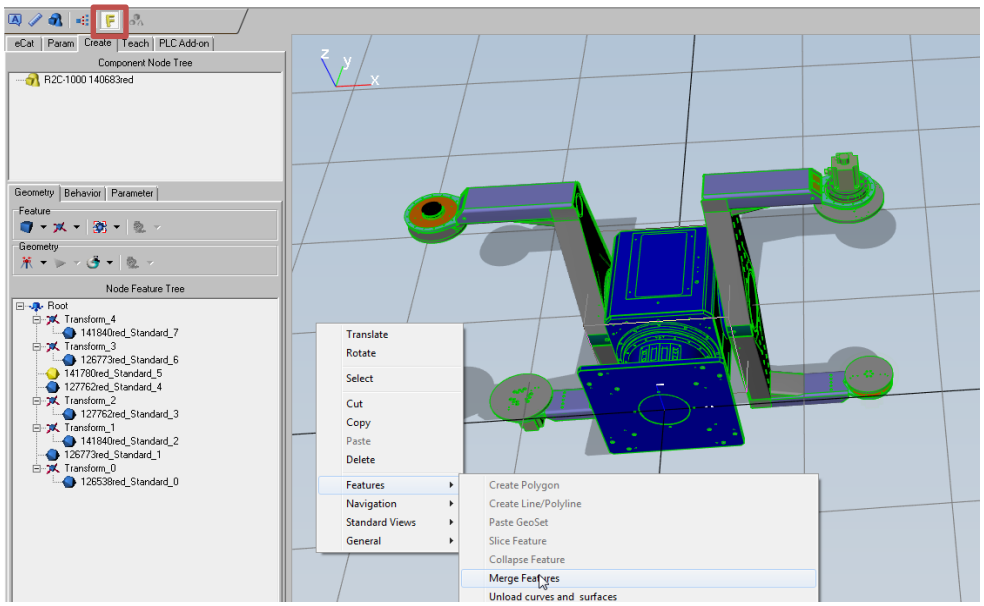
Kuva 2. Create-välilehti, jossa Component Node Tree käsittää päänoden (WeldingTable), jonka alirakenteessa on linkit.

4 Hitsauspöydän simulointi

4.1 3D-mallin tuominen simulaatiopohjaan ja muokkaus

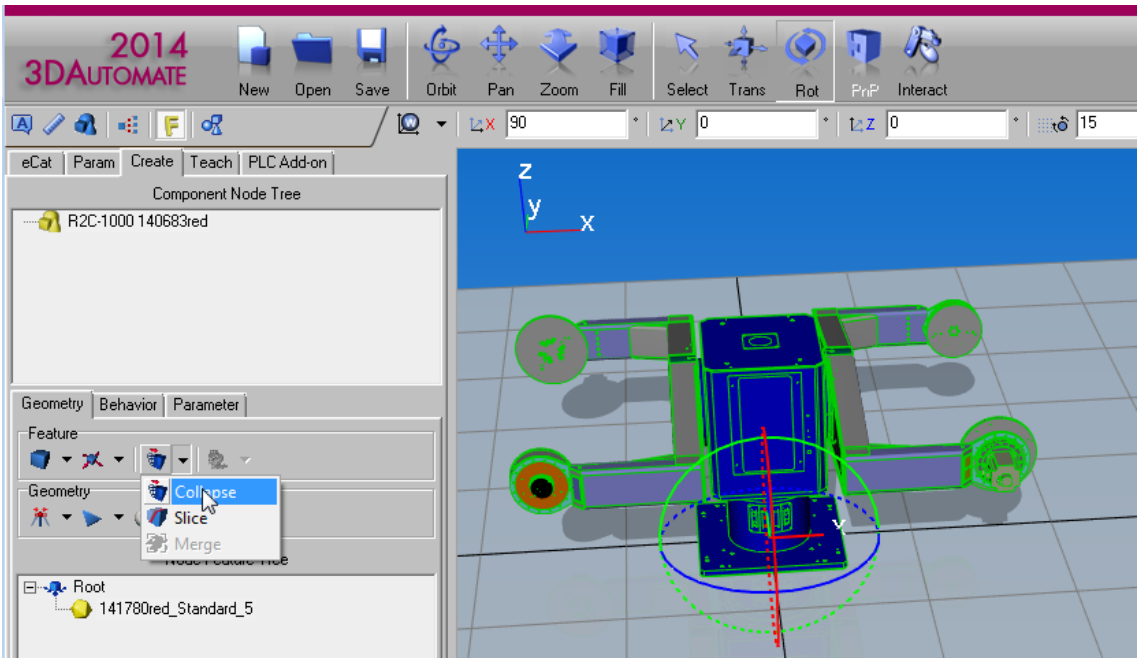
Hitsausprosessia lähdettiin rakentamaan ensin valitsemalla hitsauspöytä ja muokkaamalla sitä Create-välilehdessä. Mallina käytettiin Motomanin R2C-1000 kaksiasemaista pöytää, jonka 3D-malli saatiin suoraan motomanin nettisivuilta [6]. 3D-mallin tiedoston on oltava STEP-muodossa, jotta se olisi lukukelpoinen simulointiohjelmalle. Malli tuodaan ohjelmaan raahaamalla (drag & drop) tiedosto simulointipohjaan. Usein mallien koordinaatistot ovat erilaisia kuin ohjelmassa, joten malli saattaa tulla pohjaan ei-toivotussa asennossa, mikä voi aiheuttaa ongelmia prosessin muissa vaiheissa. Tämä pystytään parhaiten välttämään asettamalla koordinaatistojen suunta 3D-mallinnusohjelmassa samansuuntaiseksi kuin simulointiohjelmassa. Koordinaatit pystytään muuttamaan jälkikäteen 3DAutomatessa, mutta se usein vaatii komponenttien osien yhdistämistä ja erottamista uudelleen sekoittaen alkuperäisesti tarkoitetut säädöt.

Koska hitsauspöytä oli valmiiksi tehty malli, olivat myös tässä koordinaatit sattuneet ei-toivotusti kyljelleen ja kaikki osat yhdistettiin yhdeksi piirteeksi Node Feature Tree -kehyksessä kuvan 3 mukaisesti.



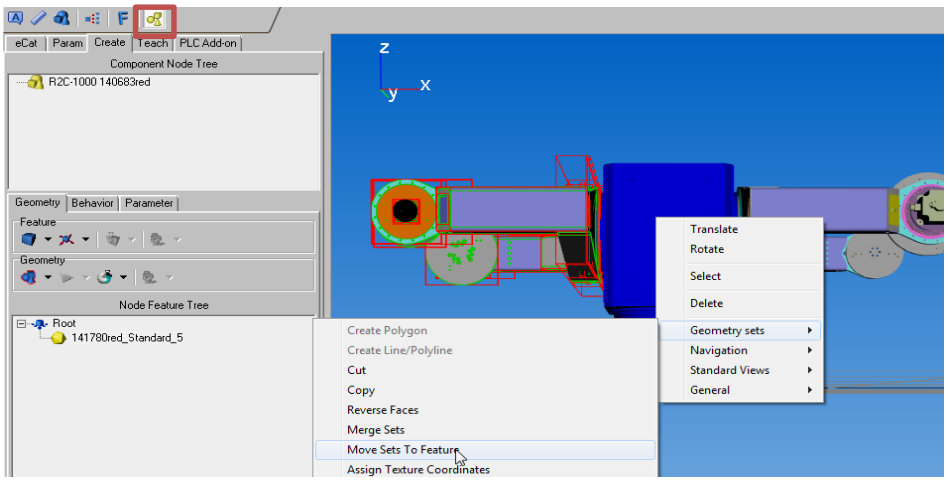
Kuva 3. Mallin osat yhdistettiin valitsemalla Features vasemmasta yläkulmasta, valitsemalla kaikki osat ja klikkaamalla Merge Featuresia.

Yhdistämisestä jäi Transform-piirteitä, jotka poistettiin. Pöytää käännettiin Rot-työkalulla x-suunnassa 90 astetta ja painettiin Collapse-toimintoa kuvan 4 mukaisesti. Tämä nollaa pöydän koordinaatiston asetetun asennon mukaisesti, eli pöydän Z-akseli saatiin osoittamaan ylöspäin.



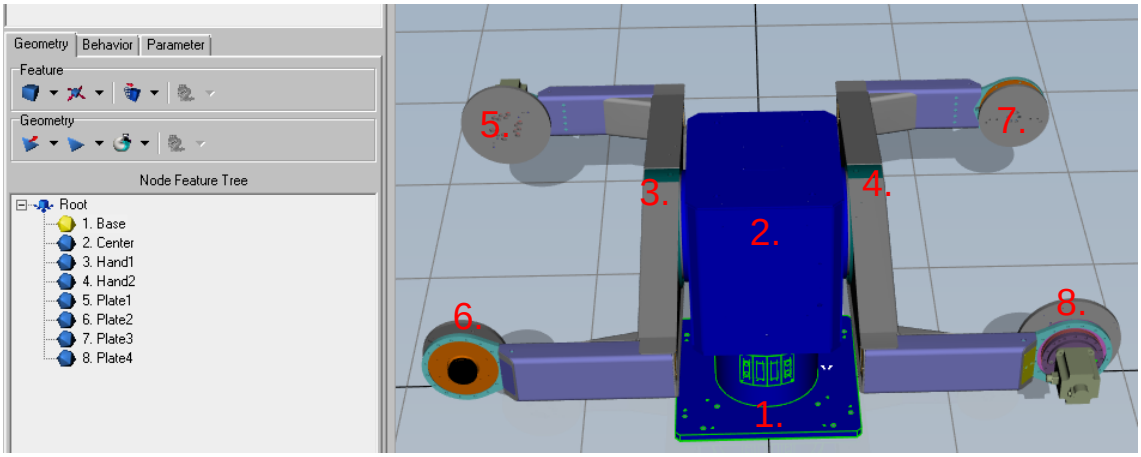
Kuva 4. Pöydän koordinaatisto saatiin nollattua Collapse-toiminnolla.

Osat saatiin erotettua valitsemalla Geometry sets vasemmasta yläkulmasta, valitsemalla osat, jotka halutaan piirteeseen ja painamalla Move Sets To Feature -toimintoa kuvan 5 mukaisesti.



Kuva 5. Osat eroteltiin Move Sets To Feature –toiminnon avulla.

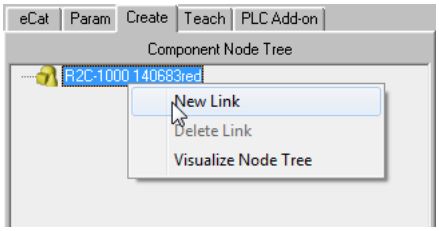
Hitsauspöydästä eroteltiin lopulta jalka, keskiosa, molemmat kädet ja käsien levyt kuvan 6 mukaisesti.



Kuva 6. Hitsauspöydästä eroteltiin lopulta jalka (1), keskiosa (2), molemmat kädet (3 – 4) ja käsien levyt (5 – 8).

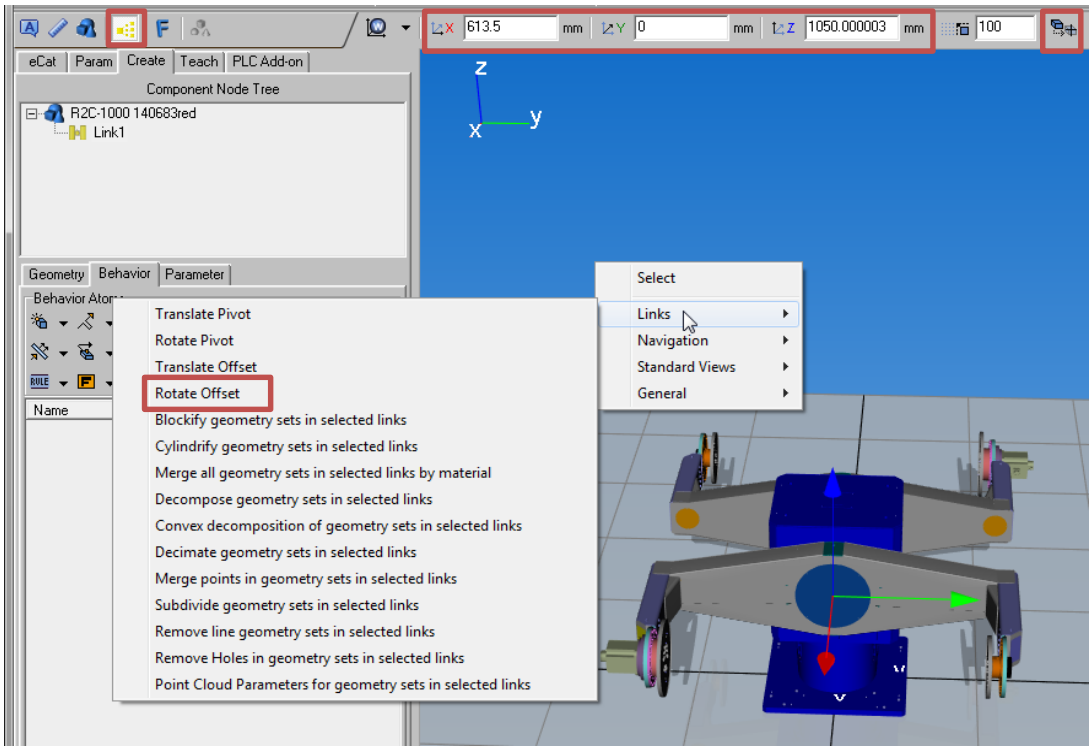
4.2 Linkkien luominen

Koska hitsauspöydässä on useita eri niveliä, niille täytyy erotella ja määritellä uudet koordinaatistot, joiden mukaan ne kääntyvät. Uusi linkki Component Node treehin luodaan kuvan 7 mukaisesti.



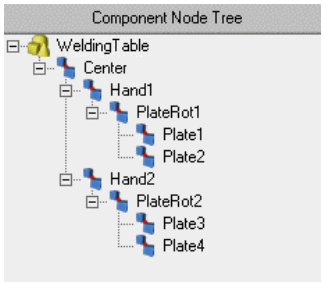
Kuva 7. Linkin luominen.

Linkin koordinaatisto vaihdetaan painamalla Links-näppäintä yläreunassa (kuva 8), valitsemalla Links valikosta Translate Offset. Nyt linkin koordinaatin voi asettaa joko syöttämällä arvot suoraan X-,Y- ja Z-laatikoihin tai valitsemalla oikeassa yläreunassa olevan Snap Position to Frame / Vertex / Surface Point -työkalun (kuva 8) ja painamalla kohtaa, johon koordinaatin origon haluaa sijoittaa.



Kuva 8. Linkin koordinaatiston origon muuttaminen.

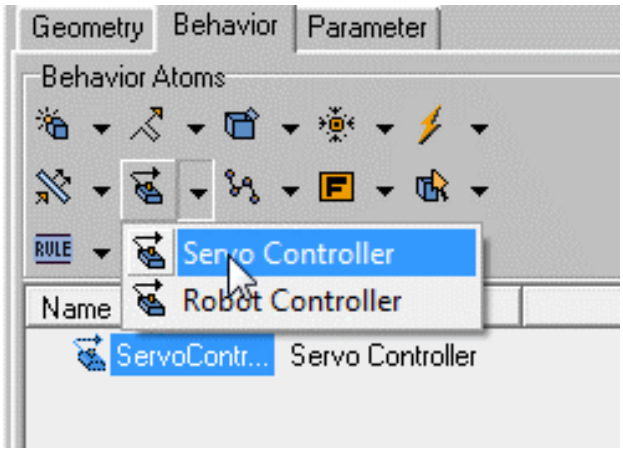
Hitsauspöytään luotiin luotiin yhdeksän linkkiä: yksi pöydän kääntämiseen, kaksi käsien kääntämiseen, kaksi pyörittämään kiinnityslevyjä ja neljä työntämään kiinnityslevyjä molemmilta puolilta. Linkit tulee jaotella alirakenteisiin kuvan 9 mukaisesti, jotta alirakenteen osat liikkuisivat samalla, kun ylärakenne liikkuu. Lopuksi erotetut osat viedään pitämällä shiftiä pohjassa linkkeihin, näin osien paikka ei muutu linkin koordinaatin mukaan.



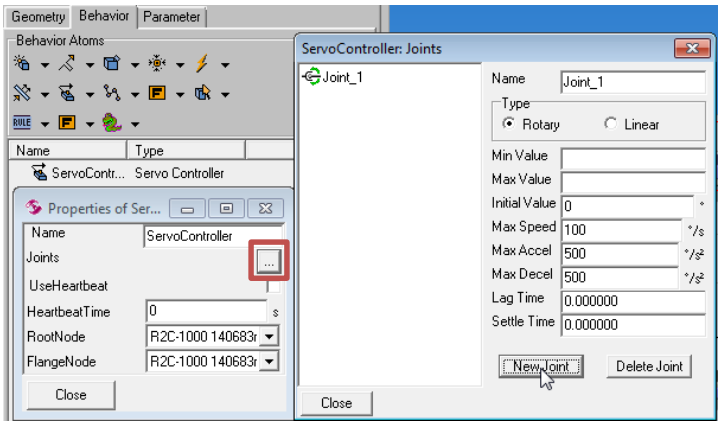
Kuva 9. Hitsauspöydän linkit.

4.3 Servojen ja nivelten luominen

Varsinainen nivel linkeille luodaan servon avulla, joka myös liikuttaa niveliä. Nivelet voidaan määrittää joko ympyräliikkeisiksi tai lineaarisiksi. Servo luodaan Behavior-välilehdessä kuvan 10 mukaisesti ja sille määritellään nivelet kuvan 11 mukaisesti. Kaikki luodut servot tulee sijoittaa päänodeen käytettävyyden helpottamiseksi. Kuvassa 12 päänodena on WeldingTable, jonka alirakenteena on linkkejä.

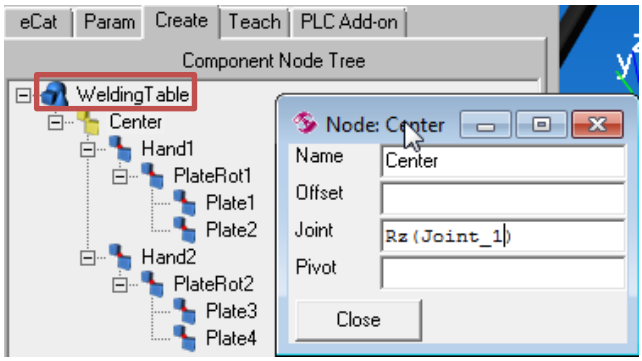


Kuva 10. Servon luominen.



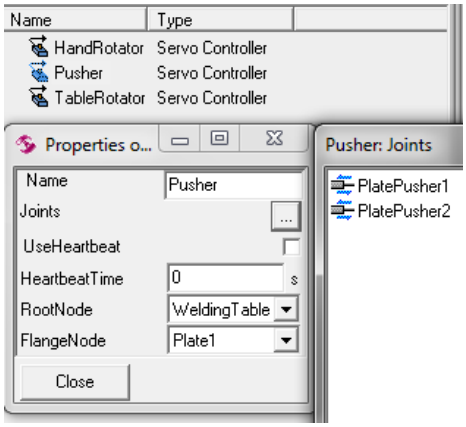
Kuva 11. Nivelen luominen.

Luotu nivel määritetään haluttuun linkkiin kuvan 12 mukaisesti kaksoisnapauttamalla linkkiä ja asettamalla Joint kehykseen, minkä akselin mukaan nivel liikkuu ja mistä nivelestä on kyse. Kuvan 12 nivelellä halutaan kääntää koko pöytää, eli sen on käännyttävä Z-akselin ympäri.



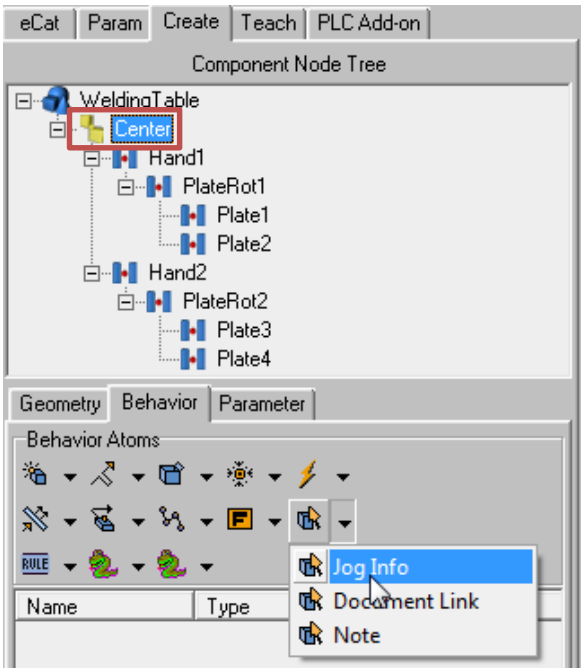
Kuva 12. Päänode ja linkin nivelen määrittäminen.

Servoon voi määrittää useampia niveliä, mutta mikäli niveliä on paljon, on hyvä luoda myös useampia servoja. Hitauspöytään luotiin kolme servoa (kuva 13) - niihin määrätyistä nivelistä levyjä työntävät nivelet määrättiin lineaarisiksi ja muut ympyräliikkeisiksi.

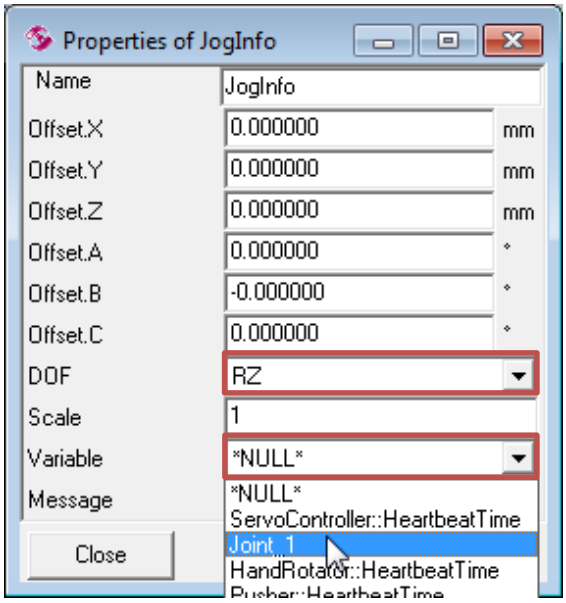


Kuva 13. Hitauspöydän servot ja levyjä työntävän servon nivelet.

Kappaleen osat tehdään liikuteltaviksi Jog-behaviorin avulla, joka luodaan siihen linkkiin, jossa kyseinen osa on. Kuvassa 14 Jog luodaan Center-linkkiin ja sille annetaan kuvassa 15 akseli, jonka mukaan sitä liikutellaan, sekä linkkiin tarkoitettu nivel (joint_1).



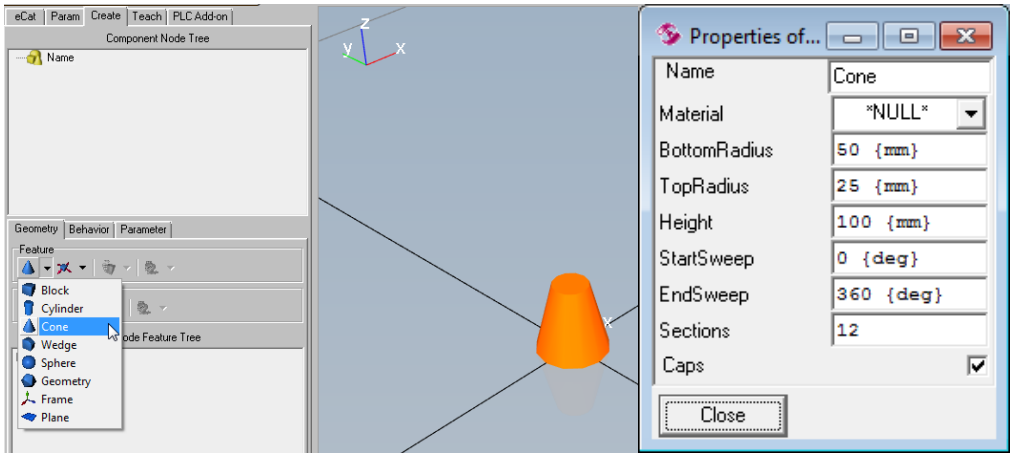
Kuva 14. Jogin luominen.



Kuva 15. Akselin ja nivelen määrittäminen.

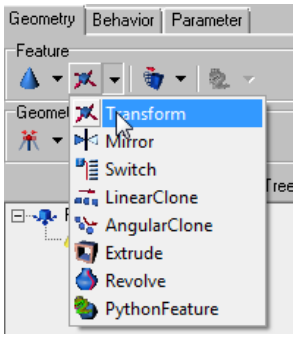
4.4 Osien luonti, pöydän korotus ja hitsaussuojan luominen

Osia voidaan luoda Create-välilehden Geometry-välilehdessä joko lisänä kappaleeseen tai kokonaan uutena osana. Feature-kehyksen voi kuvan 16 mukaisesti luoda erilaisia yksinkertaisia muotoja.



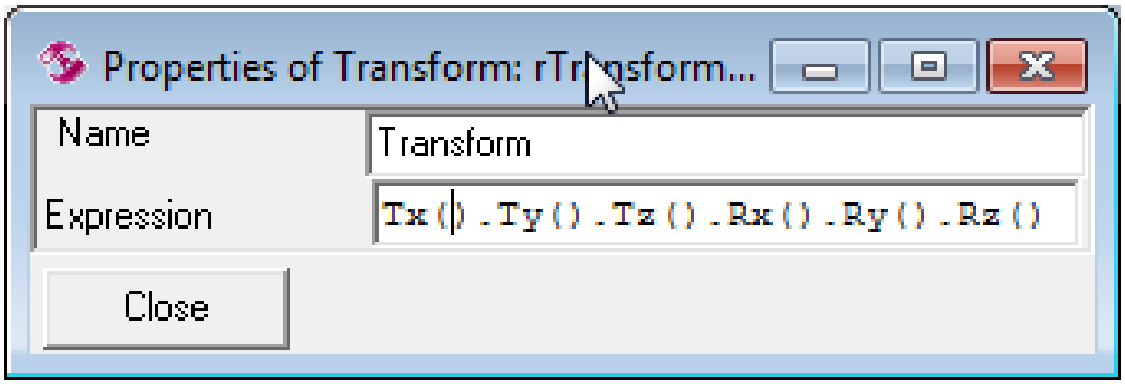
Kuva 16. Muodon luominen ja sen arvot.

Muotojen ominaisuuksia ja sijainteja voi muokata muotojen vieressä olevilla työkaluilla (kuva 17), joista Transform-työkalu on käytetyin.



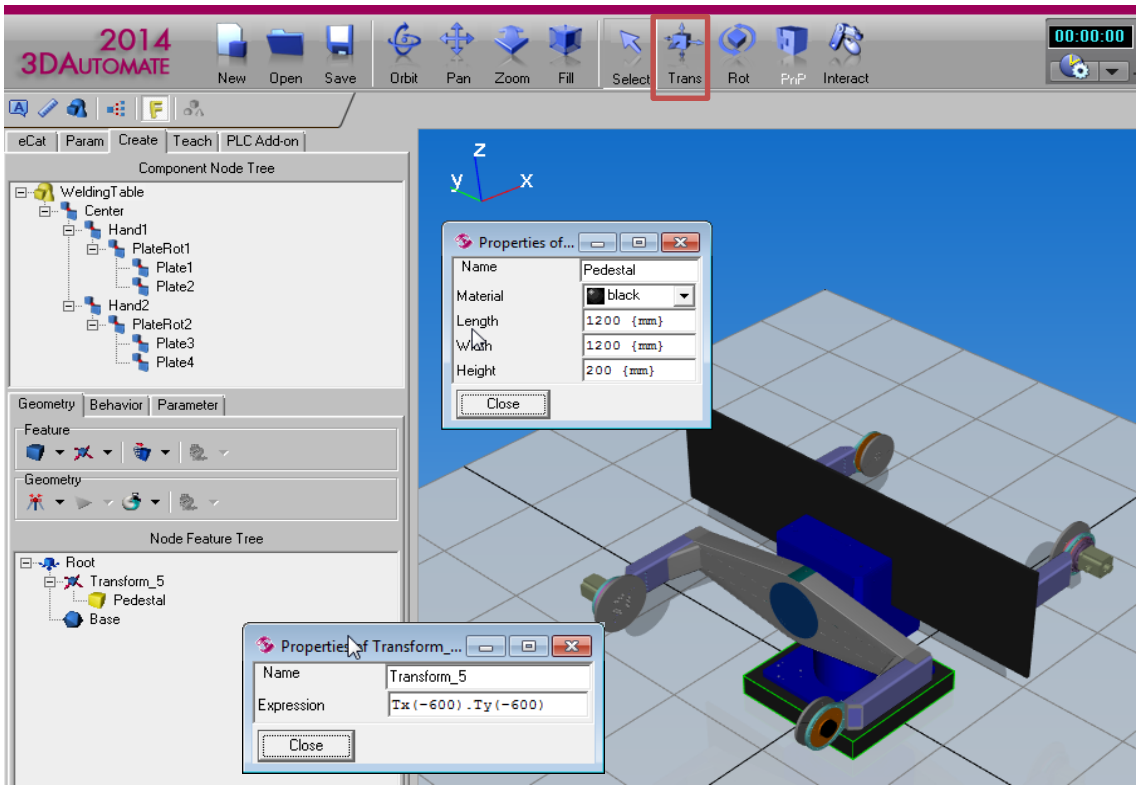
Kuva 17. Muotojen muokkaustyökalut.

Transformilla voi siirtää ja kääntää kappaletta kuvan 18 mukaisesti. T, eli translate, siirtää ja R, eli rotate, kääntää, minkä jälkeen määrätään, minkä akselin mukaan muoto siirtyy tai kääntyy. Komennot erotellaan pisteellä.



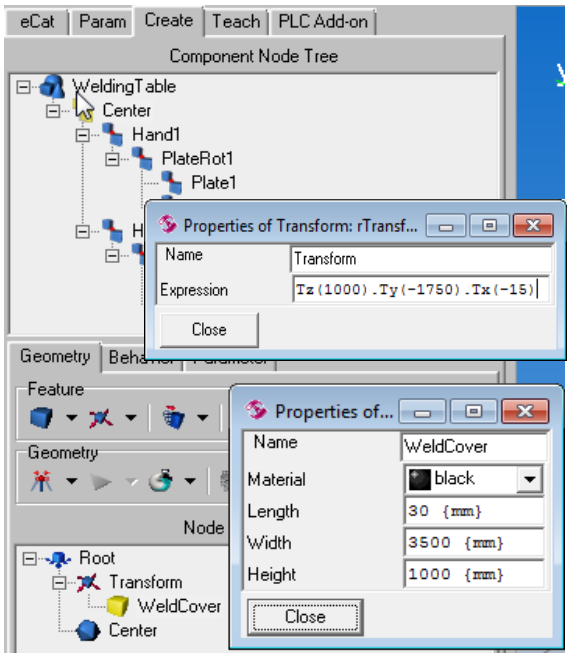
Kuva 18. Transform-työkalu.

Hitsauspöydän käsivartta käännellessä huomattiin sen osuvan maahan, joten pöytää korotettiin ensin nostamalla Trans-työkalulla 200 mm Z-akselissa ylöspäin ja luomalla jalusta Block-featurena päänodeen kuvan 19 mitoilla ja sijainnilla.



Kuva 19. Jalusta arvoineen ja hitsaussuoja.

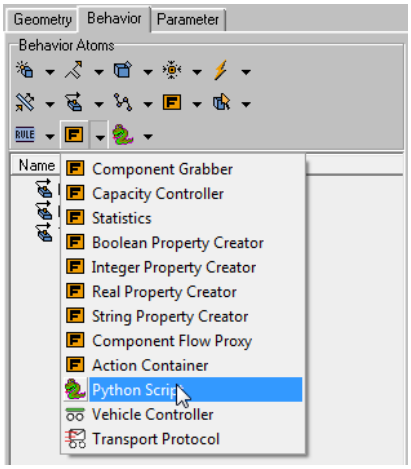
Kuvassa 19 näkyy myös hitsaussuoja, jonka mitat ja sijainti näkyy kuvassa 20.



Kuva 20. Hitsaussuojan arvot.

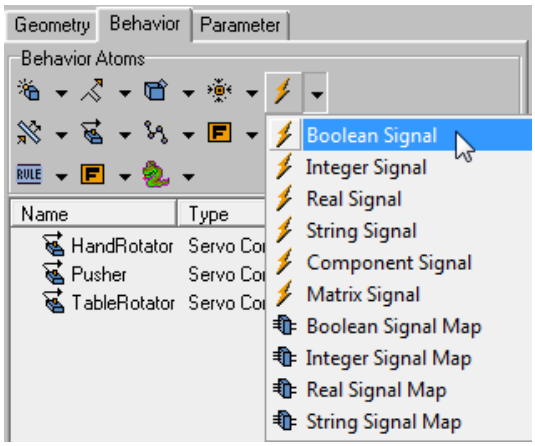
4.5 Python-koodi ja signaalit

Servoja ohjataan Python-koodin ja signaalien avulla. Signaaleja käytetään saamaan komentoja simulaation muista osista, joita käsitellään Python-koodin avulla. Python-ohjelma, johon koodi kirjoitetaan, luodaan kuvan 21 mukaisesti.



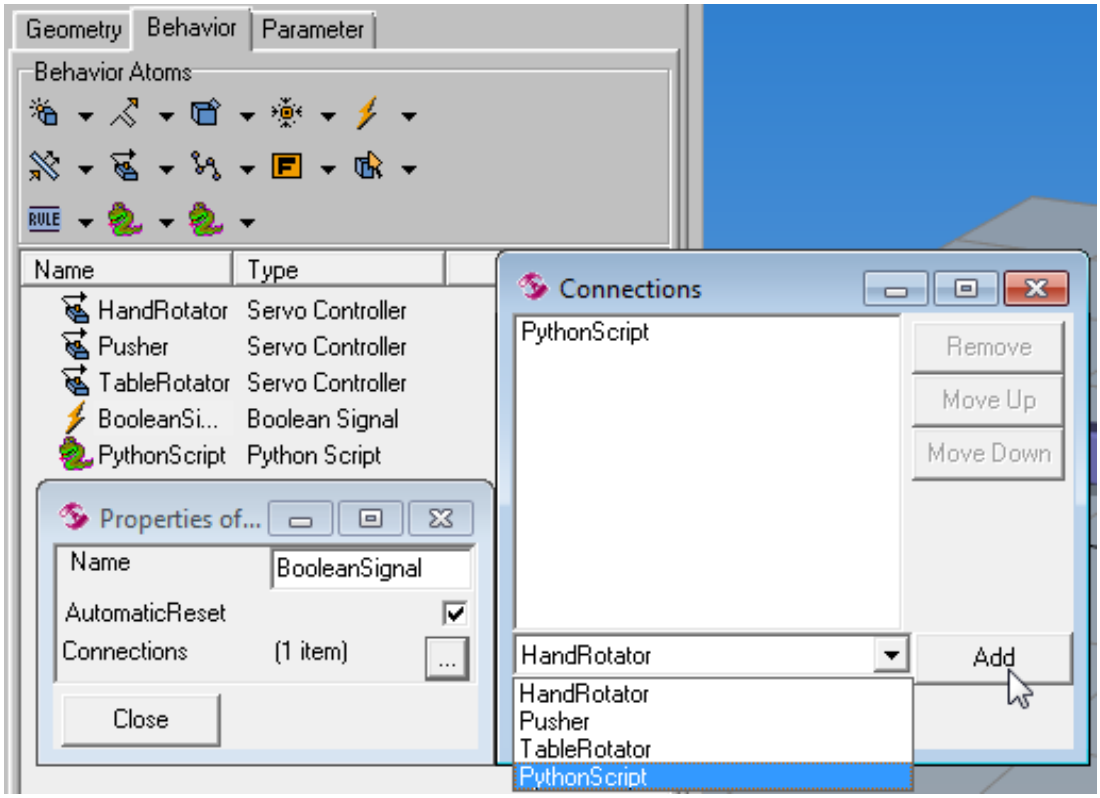
Kuva 21. Python ohjelman luominen.

Signaali luodaan kuvan 22 mukaisesti.



Kuva 22. Signaalin luominen.

Signaalit jaotellaan tyypin mukaan, joista yleisimmin käytetään boolean- (on/off), integer- (kokonaisluku) ja string-signaaleja (teksti). Hitsauspöytään luotiin kaksi boolean-signaalia, yksi integer-signaali ja yksi string-signaali. Signaalit yhdistetään Python-ohjelmaan kuvan 23 mukaisesti.



Kuva 23. Signaalin yhdistäminen Python-ohjelmaan.

Hitsauspöytään luotiin kaksi Python-ohjelmaa – toinen ohjaamaan pöydän kääntymistä ja toinen ohjaamaan levyjen työntämistä. Pöydän kääntymistä ohjaavaassa ohjelmassa (kuva 24) vaiheessa yksi (1) etsitään servot ja signaalit sekä sijoitetaan ne yksinkertaisimpiin muuttujiin. Vaiheessa kaksi (2) luodaan funktio, joka simulaation käynnistyttyä palauttaa pöydän alkuasentoon. Yksinkertaistettuna vaiheessa kolme (3) funktio menee silmukkaan, jossa task-signaalin on saatava kaksi käskyä kääntää pöytä ennen kuin funktio kääntää pöydän. Nämä kaksi käskyä ilmoittavat, että sekä työntekijä on asettanut uuden tuotteen hitsauspöytään että robotti on valmis. Käskyt saatuaan jos pöytä on perusasennossa, se kääntyy 180 astetta tehtävälle määrätyssä ajassa, jos taas pöytä on jo kääntynyt, se palaa perusasentoon. Kääntymisen jälkeen funktio lähettää tiedon siitä, mikä tehtävä on tehty ja silmukka alkaa alusta.

Script 'WeldinTable::RotateScript'

```
1  from vcScript import *
2
3  def OnRun():
4    global comp, servo
5    comp = getComponent()
6    stats = comp.findBehaviour("Statistics")
7    servo = comp.findBehaviour("TableRotator")
8    task = comp.findBehaviour("task")
9
10   servo.setJointTarget(0,0)
11   servo.moveImmediate()
12   r = 0
13
14   reqs = []
15   tasks = []
16   while True:
17     stats.State = "Idle"
18     for i in range(comp.ParallelRequests):
19       triggerCondition(lambda: getTrigger() == task)
20       vals = task.Value.split(",")
21       processtime = float(vals[0])
22       othercomp = getApplication().findComponent(vals[1])
23       reqs.append(othercomp)
24       tasks.append(task.Value)
25     stats.State = "Busy"
26     dtime = processtime
27     if r == 0.0:
28       r = 180.0
29     else:
30       r = 0.0
31     servo.setJointTarget(0,r)
32     servo.setMotionTime(dtime)
33     servo.move()
34     for i in range(len(reqs)):
35       sig = reqs[i].findBehaviour("TaskDone")
36       sig.signal('%s,%s,'%(vals[0],reqs[i].Name))
37       delay(0.0)
38     reqs = []
```

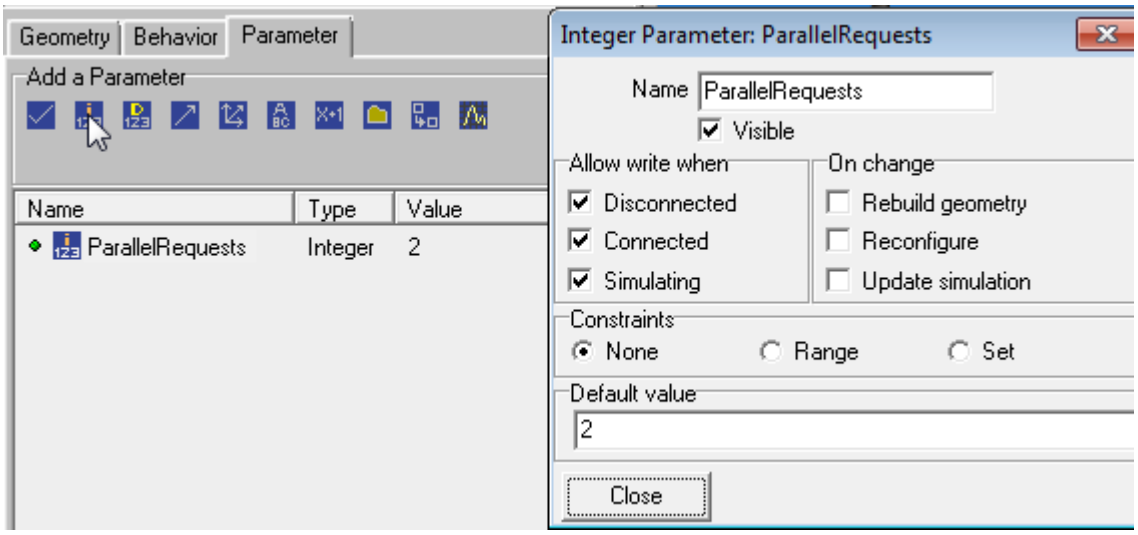
1.

2.

3.

Kuva 24. Pöydän kääntymistä ohjaava Python-ohjelma.

Jotta pöytä tietäisi kuinka monta käskyä sen tarvitsee saada ennen kuin se kääntyy, luotiin Parameter-välilehdellä ParallelRequests integer-parametri kuvan 25 mukaisesti, ja asetettiin sen perusarvoksi kaksi.



Kuva 25. Parametri kuinka monta käskyä pöydän tulee saada ennen se kuin kääntyy.

Levyjä työntämään luodussa ohjelmassa (kuva 26) vaiheissa yksi (1) ja kaksi (2) on sama periaate kuin pöydän kääntymistä ohjaavassa ohjelmassa, mutta vaiheessa kolme (3) funktio jää odottamaan, että push-signaali saa uuden arvon. Signaalista riippuen servo joko työntää levyt kappaleeseen kiinni tai avaa ne määrätyltä puolelta.

Script 'WeldingTable::GripperScript'

```
1  from vcScript import *
2
3  comp = getComponent()
4  servo = comp.findBehaviour("Pusher")
5  push = comp.findBehaviour("PushPlate")
6
7  def OnRun():
8
9    servo.setJointTarget(0,0)
10   servo.setMotionTime(0.001)
11   servo.move()
12
13   servo.setJointTarget(1,0)
14   servo.setMotionTime(0.001)
15   servo.move()
16
17   while True:
18
19     triggerCondition(lambda: getTrigger() == push)
20
21     if push.Value == 11:
22       servo.setJointTarget(0,40)
23       servo.setMotionTime(1)
24       servo.move()
25
26     elif push.Value == 12:
27       servo.setJointTarget(0,0)
28       servo.setMotionTime(1)
29       servo.move()
30
31     elif push.Value == 21:
32       servo.setJointTarget(1,-40)
33       servo.setMotionTime(1)
34       servo.move()
35
36     elif push.Value == 22:
37       servo.setJointTarget(1,0)
38       servo.setMotionTime(1)
39       servo.move()
```

1.

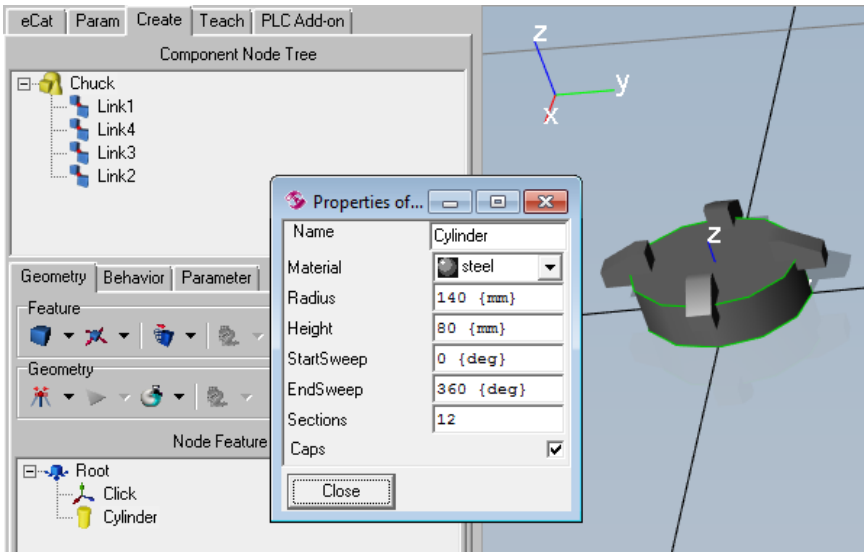
2.

3.

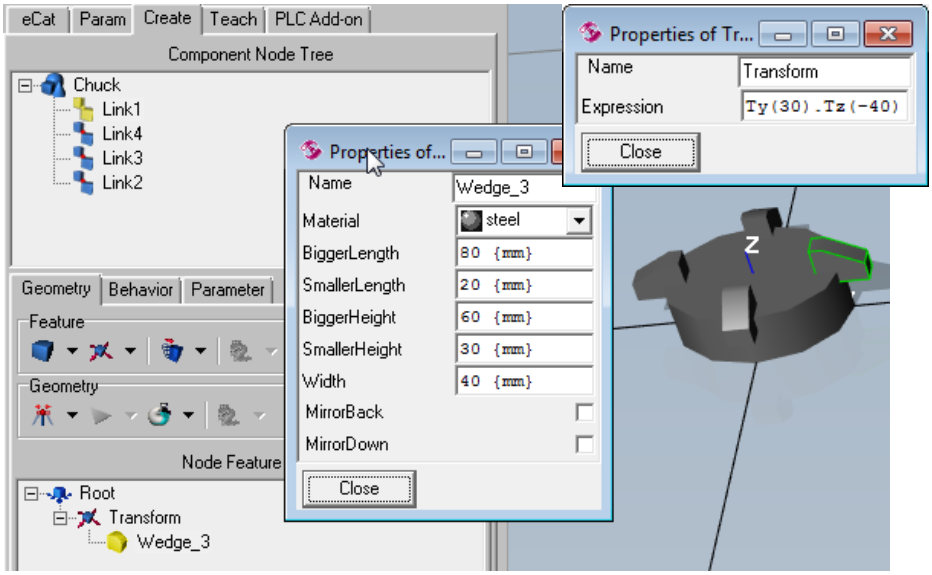
Kuva 26. Levyjen työntämistä ohjaava Python-ohjelma.

4.6 Istukan luominen

Tässä vaiheessa levyissä ei ollut varsinaista kiinnitystä, johon kappaleen voisi kiinnittää, vaan se olisi vain työntäjien paineen varassa. Tämän vuoksi luotiin erikseen hitsauspöytään kiinnittyvä istukka Create-välilehdessä kuvien 27 ja 28 mukaisesti. Istukka on periaatteessa vain sylinteri, jonka sormina toimii neljä suorakulmaista puolisuunnikasta. Lisäksi luotiin kuvassa 27 näkyvä Click-frame, joka on yhteenliittymää varten - tämä meni suoraan oikeaan paikkaan istukan pohjaan.

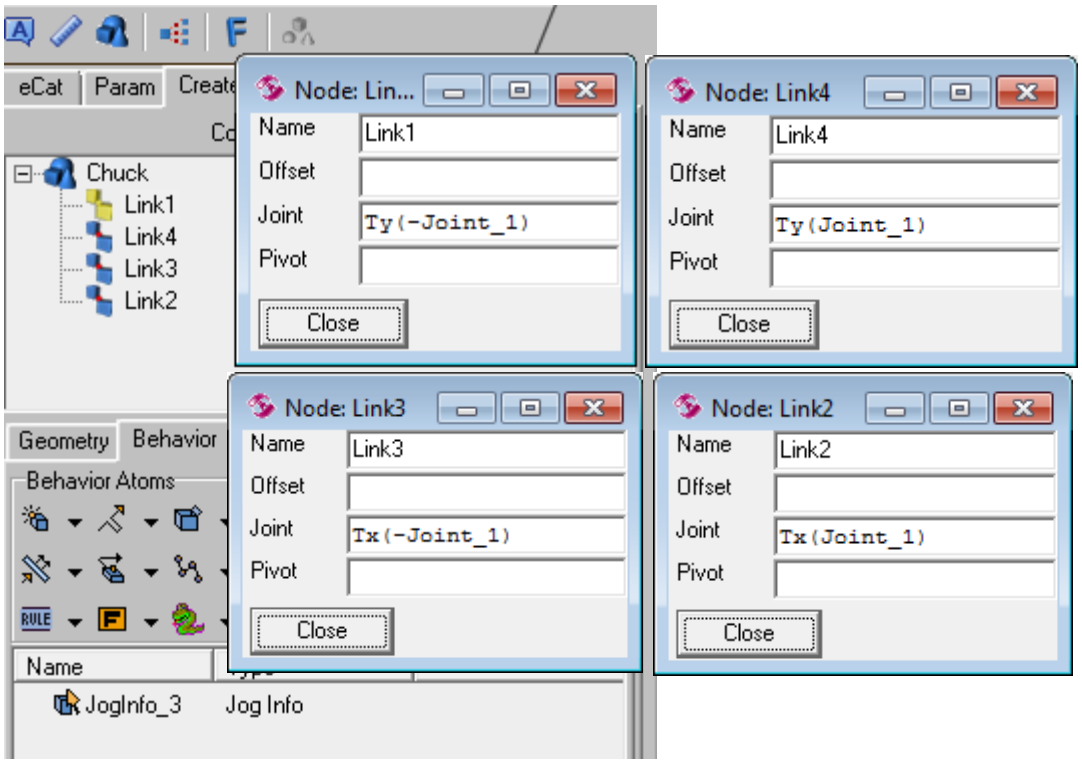


Kuva 27. Istukka.



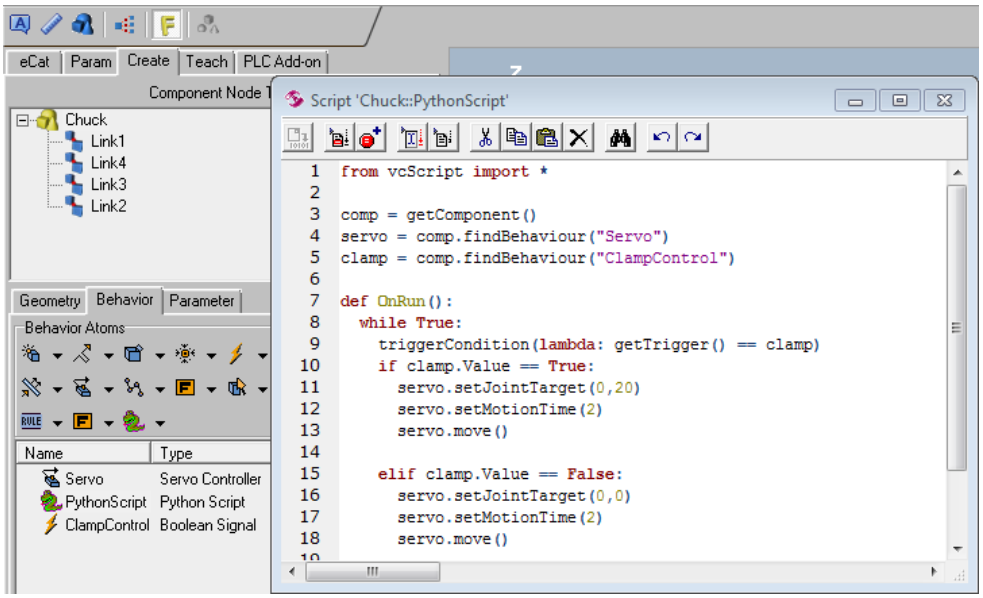
Kuva 28. Istukan sormi.

Behavior-välilehdessä istukkaan luotiin sormia ohjaava servo yhdellä lineaarisella nivelellä. Nivel syötettiin istukan linkkeihin kuvan 29 mukaisesti ja jokaiselle linkille luotiin Jog behavior. Näin riittää että kun käytetään yhtä komentoa tai liikutellaan yhtä sormea, niin kaikki sormet liikkuvat.



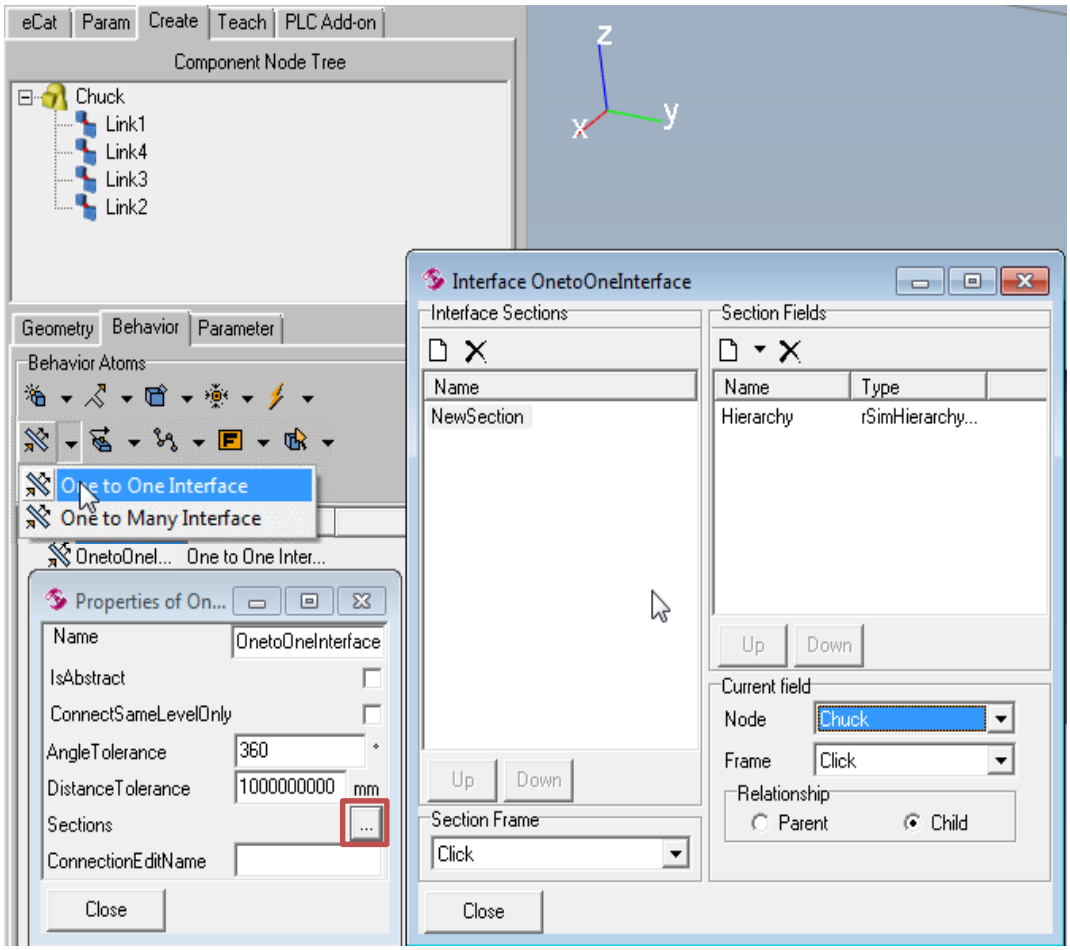
Kuva 29. Istukan sormien liikkumissuuntien määrittäminen.

Lisäksi luotiin Python-ohjelma ohjaamaan servoa, sekä Boolean-signaali sormien ohjausta varten kuvan 30 mukaisesti. Python-ohjelma (kuva 30) käskee servoa joko sulkemaan tai avaamaan sormet, kun signaali saa arvon.



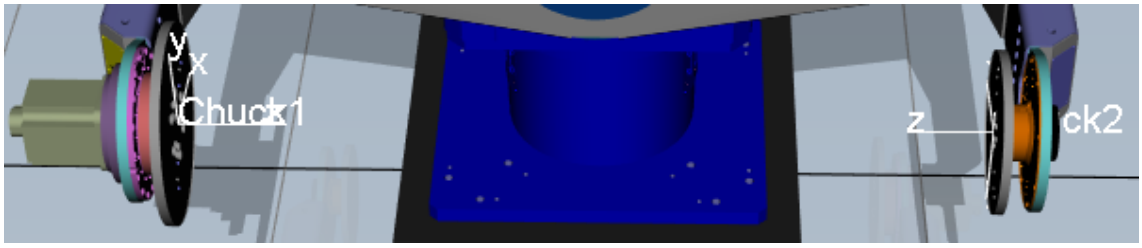
Kuva 30. Istukan behaviorit ja Python-ohjelma.

Istukan rajapinta hitsauspöydän levyjä varten luotiin kuvan 31 mukaisesti One to One Interface behavioria käyttäen. Se tehtiin luomalla Interface Sections -kehyksessä New-Section ja Section Fieldsissä Hierarchy Field, jossa Current field -kehyksessä asetettin Nodeksi Chuck ja Frameksi aiemmin luotu Click-frame. Lisäksi valittiin istukan olevan kiinnittyvä osa Relationship-kehyksessä.



Kuva 31. Rajapinnan luominen.

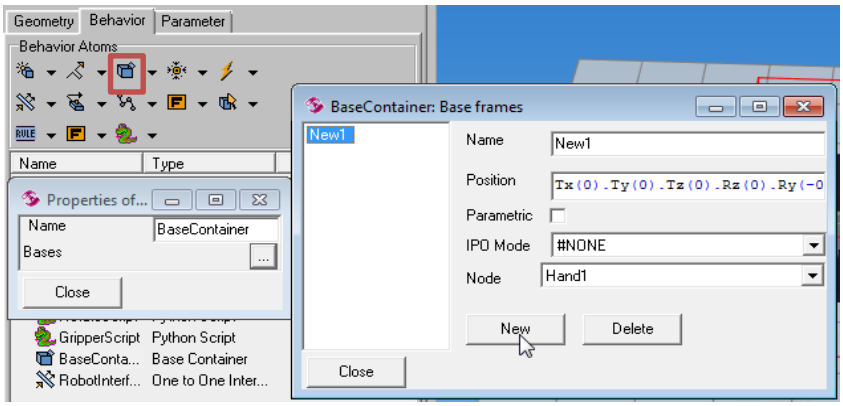
Myös hitsauspöydän levyihin luotiin rajapinta samalla tavalla kuin istukka, mutta Relationshipiksi valittiin Parent, eli siihen kiinnittäydytään, ja huomioitiin että levyn framen z-akseli osoittaa toisella puolella olevaan levyyn kuvan 32 mukaisesti.



Kuva 32. Hitsauspöydän levyjen rajapinta-framet.

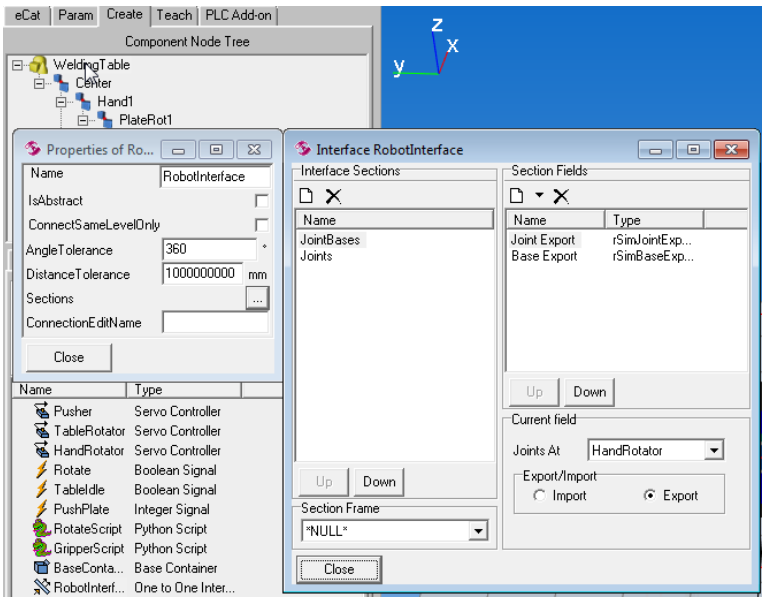
4.7 Hitsauspöydän käsien yhdistäminen robotin ohjelmointiin

Jotta saataisiin robotin liikkeitä opettavassa teach-välilehdessä opetettua myös hitsauspöydän käsien liikkeet, niin luodaan hitsauspöytään robottirajapinta. Rajapinnan luomiseksi tehtiin Base Container ja One to One Interface Behavior-välilehdessä. Base container tallentaa asetetun noden tai linkin sijaintitiedot. Hitsauspöydästä nodeksi asetettiin Hand1-linkki kuvan 33 mukaisesti.



Kuva 33. Base Container.

One to One Interfacessa määritellään minkä servon nivelten ja Base Containerin tiedot tallennetaan robotin prosessiin kuvan 34 mukaisesti.

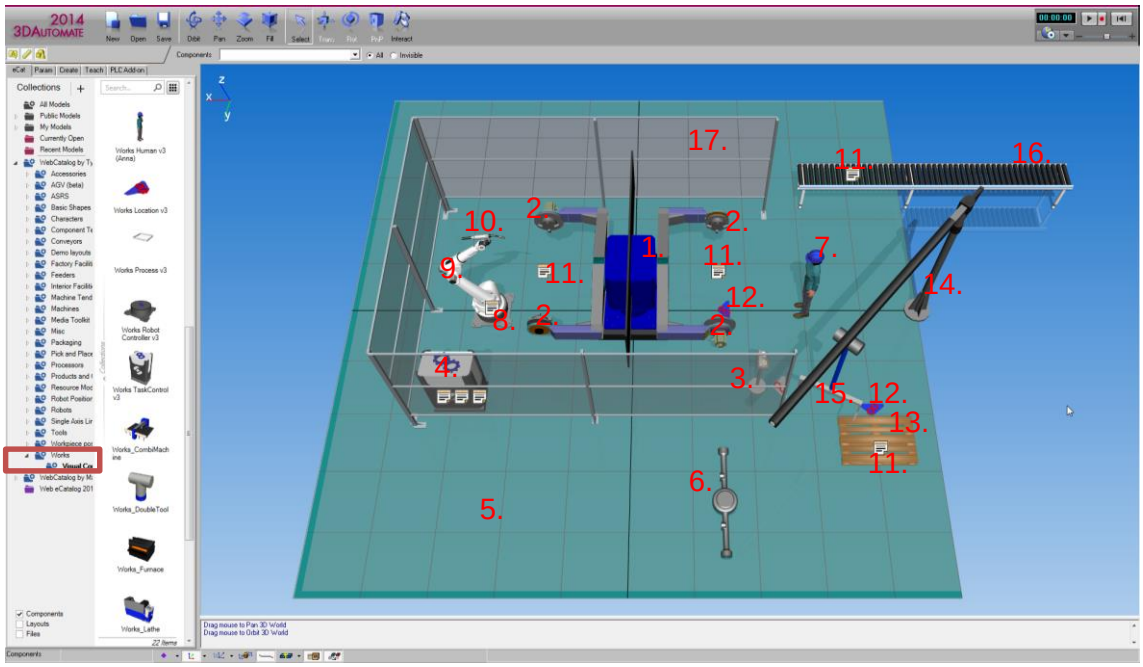


Kuva 34. Hitsauspöydän rajapinnan luominen robottiin.

5 Simulaation luominen

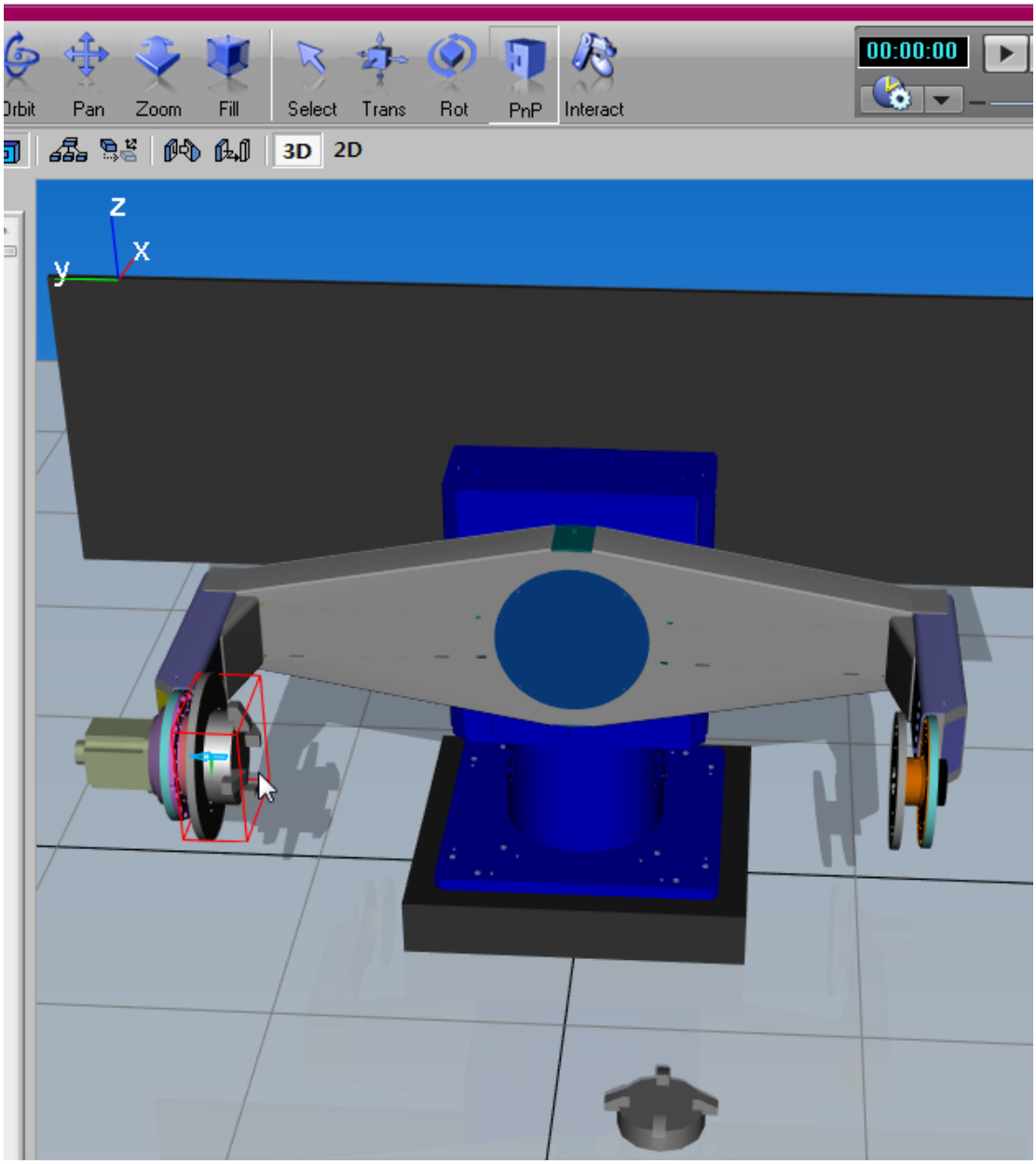
5.1 Komponenttien tuominen simulaatiopohjaan

Komponentteja voidaan tuoda simulaatiopohjaan joko valmiista kirjastoista eCat-välilehdestä tai suoraan tiedostoina. Tässä työssä käytettiin suurimmaksi osaksi Works-kirjastossa olevia komponentteja (kuva 35). Itse tehtyjä osia ovat hitsauspöytä, akseli ja nostokoukku. Simulaatiopohja rakennettiin hitsauspöydän ympärille kuvan 35 mukaisesi. PLC-rajapinnan tekeminen käydään läpi kappaleessa 6.



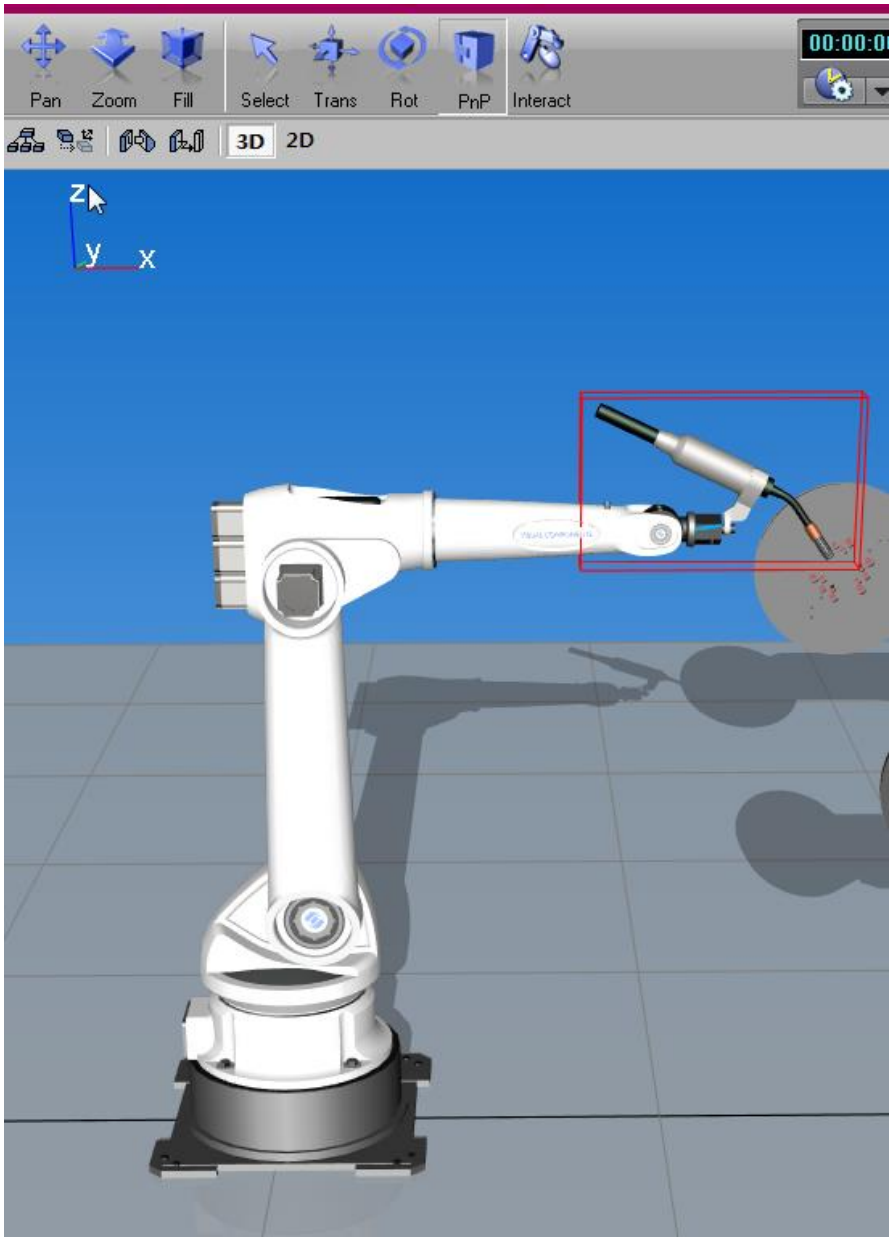
Kuva 35. Works-kirjasto ja simulaation komponentit: Hitasuspöytä - WeldingTable (1), 4 x Istukka - Chuck (2), Painike - Works FloorButton v3 (3), Works TaskControl v3 (4), Työalue - WorksArea v3 (5), Akseli - Axle (6), Työntekijä - WorksHuman v3 (7), Robotin ohjainjalusta - Works Robot Controller v3 (8), Robotti - Generic Articulated Robot v2 (9), Hitsauspilli - Parametric Weld Torch (10), 4 x Prosessiohjelma - WorksProcess (11), 2 x Työsijainti - WorksLocation (12), Kuormalava - Euro Pallet (13), Nostoapuvarsi - Lift Assist Kinematics (telescope boom) (14), Nostokoukku - Lift Assist L (handle) (15), Liukuhihna - Conveyor (16), 6 x Aita - Fence Builder (17).

Osia, joihin on tehty kappaleen 4.6 mukaisesti hierarkiarajapinta, yhdistetään toisiinsa PnP-työkalun avulla. PnP:tä käytetään vetämällä yhdistettävää komponenttia liitoskohdan lähelle, jolloin ohjelma tunnistaa liitosmahdollisuuden ja lukitsee komponentin ylirakenteessa määrätyn koordinaatiston mukaiseen asentoon. Tässä insinöörityössä PnP:tä käytettiin yhdistämään istukat hitsauspöytään kuvan 36 mukaisesti.



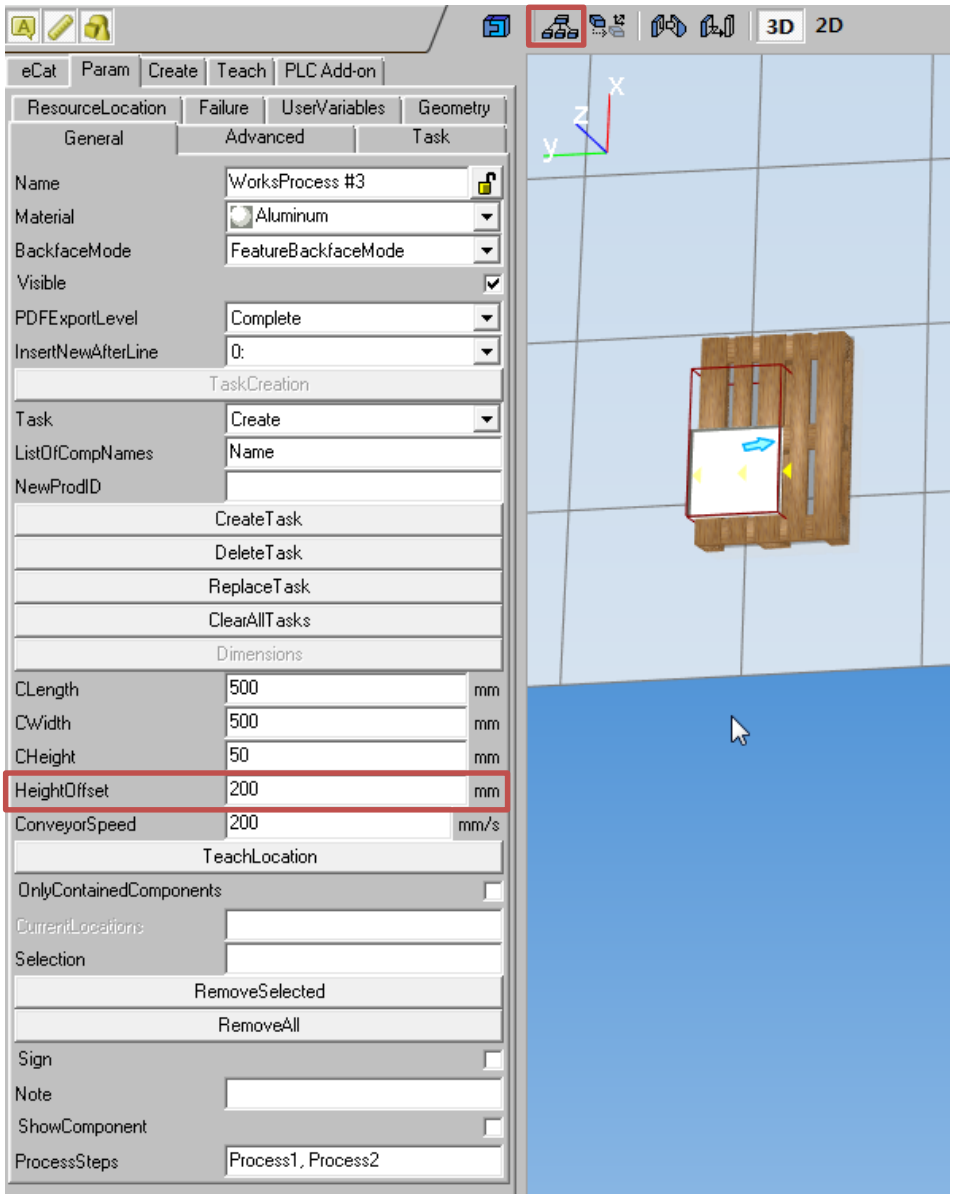
Kuva 36. Istukat asetettiin PNP-työkalun avulla.

Robotti yhdistettiin PnP:llä robotin ohjausalustaan ja hitsipuikko yhdistettiin robottiin kuvan 37 mukaisesti.



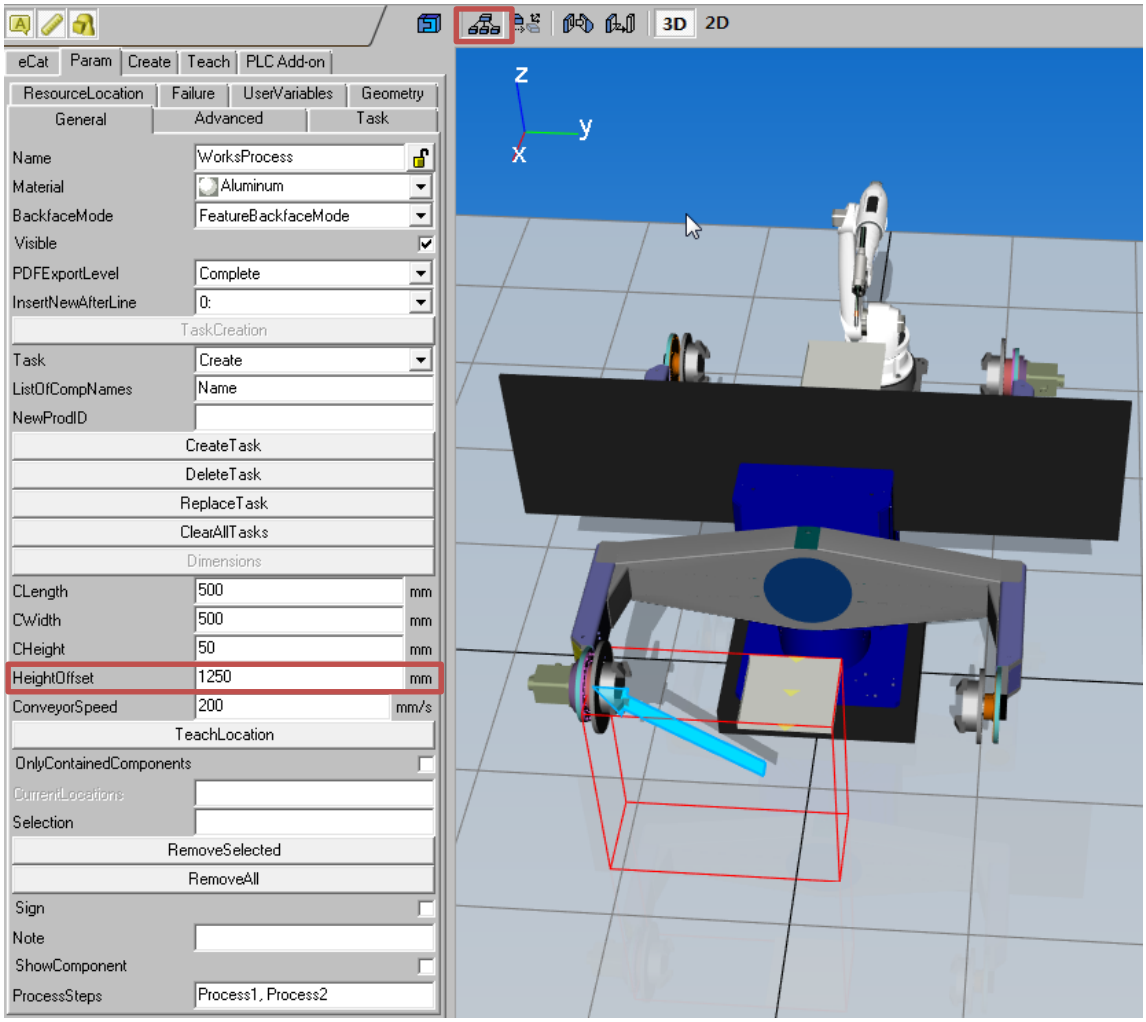
Kuva 37. Robotin ohjausjalusta ja hitsauspilli yhdistettiin PnP:llä.

Jos kummallakaan komponentilla ei ole hierarkiarajapintaa, komponentti voidaan yhdistää toiseen komponenttiin PnP:n kuvassa 38 näkyvää Set Parent Node for Selected Component -työkalua käyttäen. Yksi WorksProcess yhdistettiin kuormalavalle ja sen korkeutta säädettiin kuvan 38 mukaisesti Param-välilehden General-välilehdessä. Tämän WorksProcessin tarkoitus on luoda akseleita, joten sen tarkka paikka riippuu akselipinon muodosta.



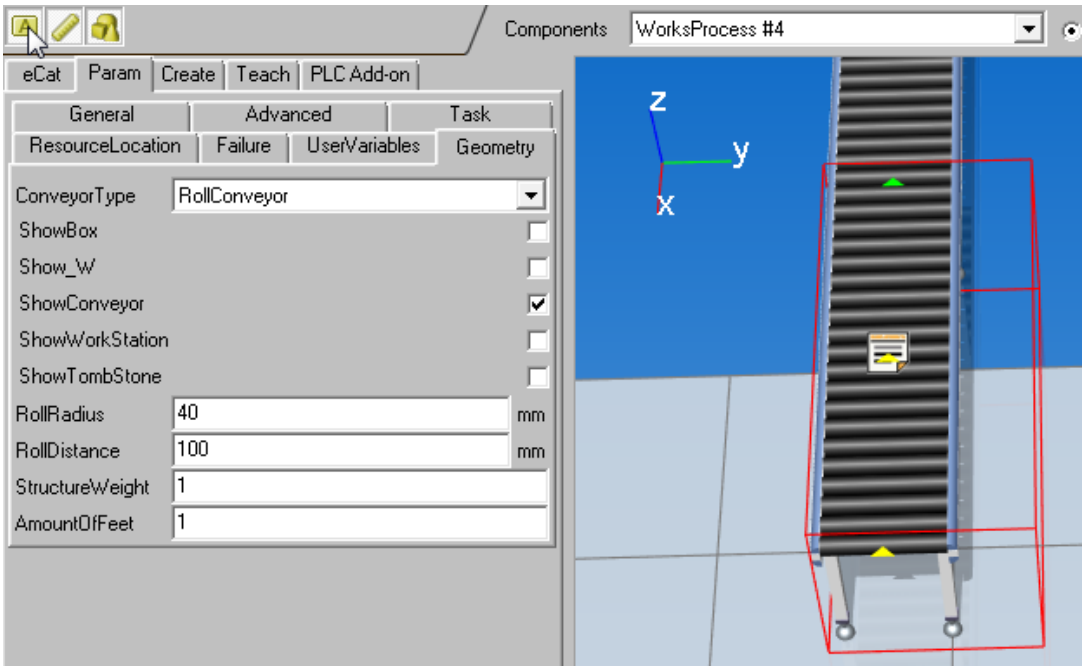
Kuva 38. WorksProcess yhdistettiin kuormalavalle ja sen korkeutta säädettiin Param-välilehdellä.

Kaksi WorksProcess:a yhdistettiin hitsauspöytään PnP:llä ja ne korotettiin kuvan 39 mukaisesti 1250 mm General-välilehdessä.



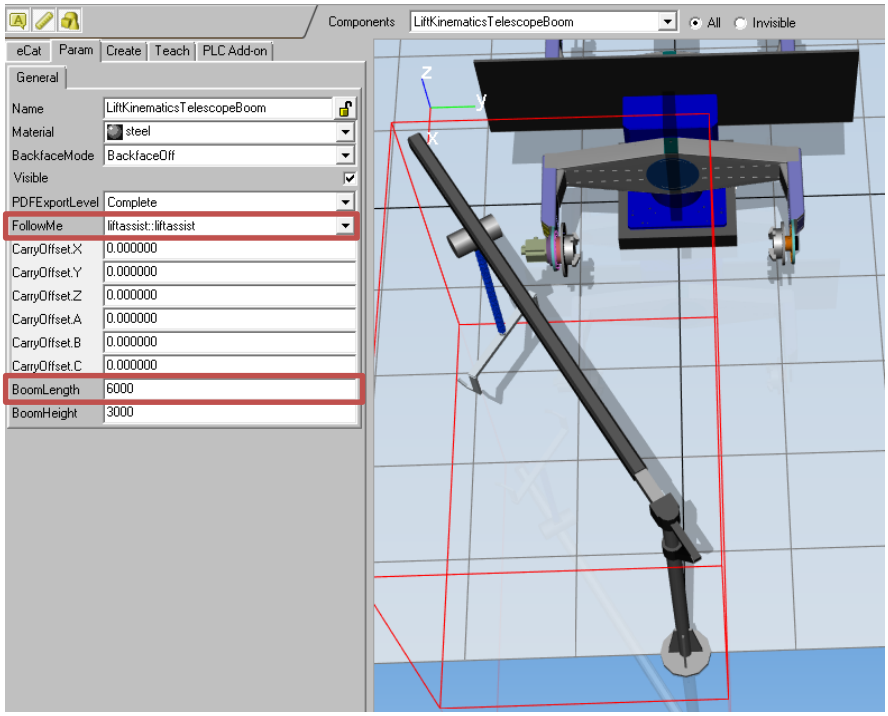
Kuva 39. Kaksi WorksProcessia yhdistettiin hitsauspöydän molempiin käsiin.

WorksProcesseihin on luotu valmiiksi erilaisia toimintoja, joita pääsee muokkaamaan Param-välilehden Geometry-välilehdessä. Toiminnoista kuvassa 40 näkyvä oletusasetuksena oleva ShowBox otettiin pois kaikista WorksProcessista, mutta neljäs muutettiin liukuhihnaksi kuvan mukaisesti.



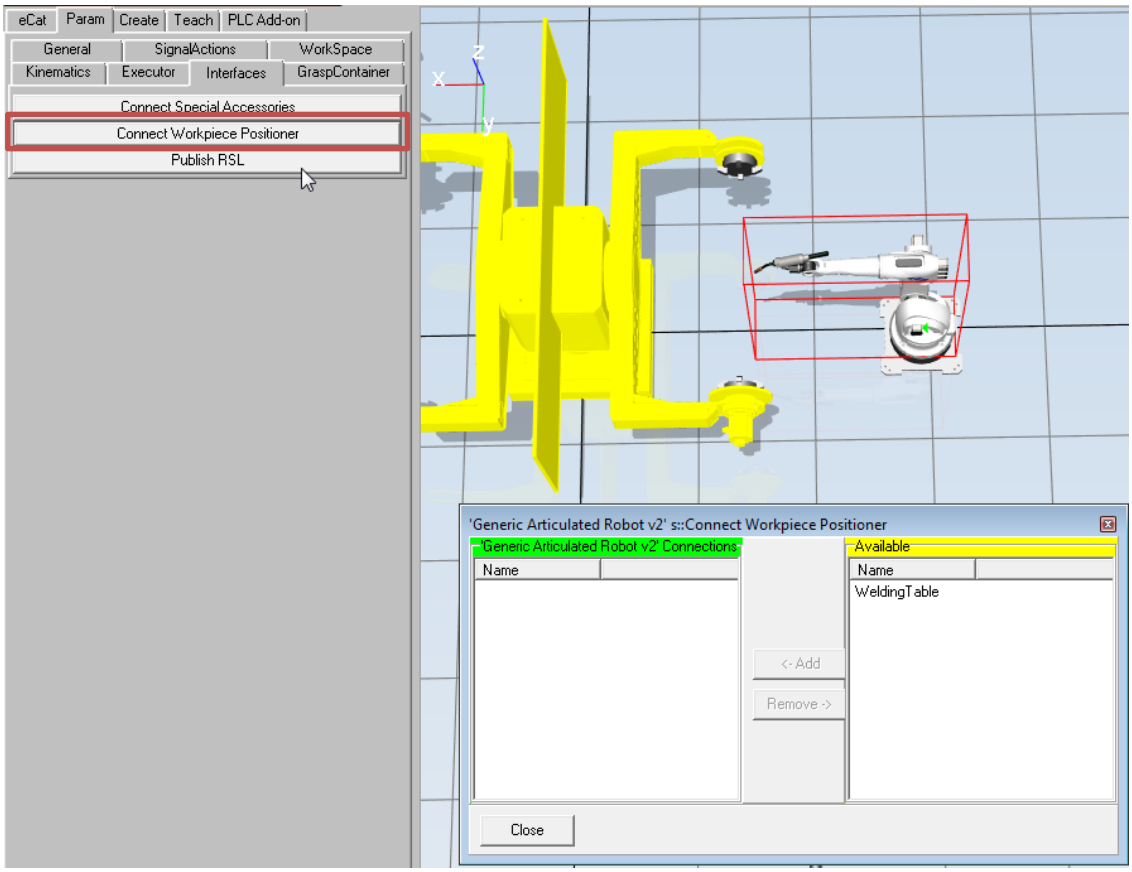
Kuva 40. WorksProcessista poistettiin ShowBox ja se muutettiin liukuhihnaksi.

Nostoapuvarsi ja nostokoukku yhdistettiin nostoapuvarren Param-välilehdessä valitsemalla nostokoukku FollowMe-valikosta kuvan 41 mukaisesti. Myös varren pituutta muutettiin.



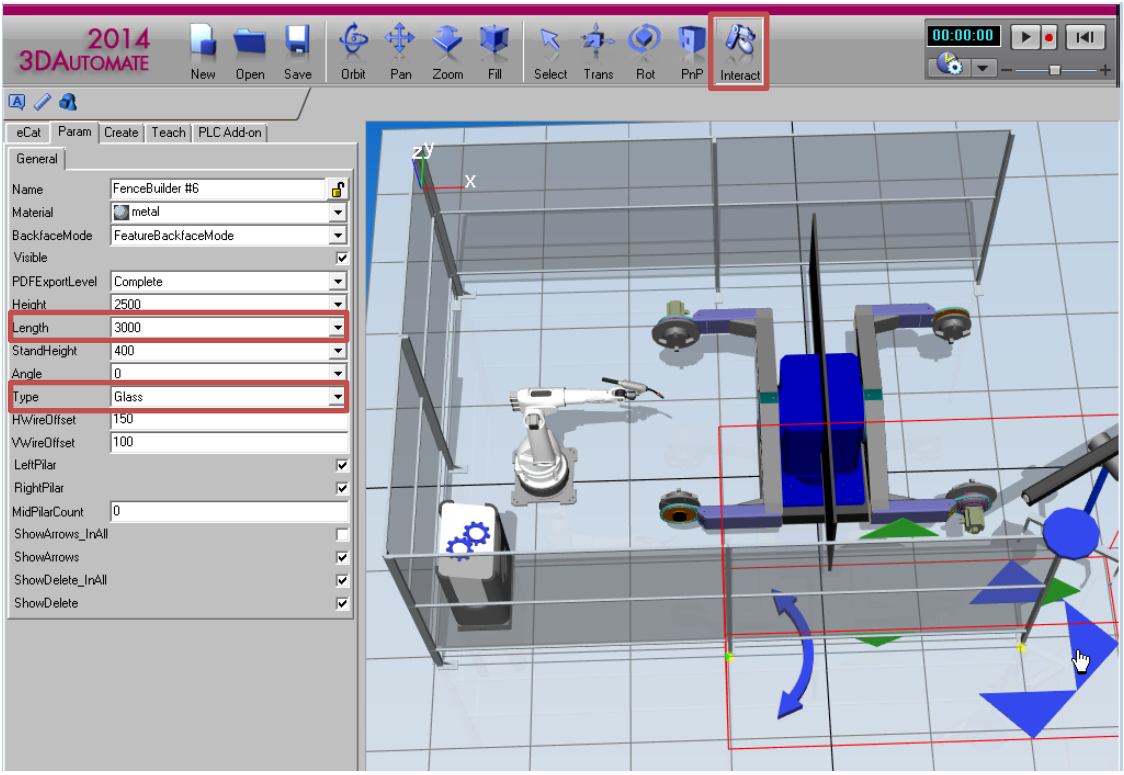
Kuva 41. Nostoapuvartta pidennettiin ja se laitettiin seuraamaan nostokoukkua.

Aikaisemmin tehdyn robottirajapinnan avulla hitsauspöydästä tehtiin työkappaleen paikoittaja robotille sen Param-välilehden Interface-välilehden Connect Workpice Positioner -valikossa kuvan 42 mukaisesti. Valikossa kaikki yhdistettävät vaihtoehdot näkyvät keltaisina ja valitut muuttuvat vihreiksi.



Kuva 42. Hitsauspöydästä tehtiin töykappaleen paikoittaja robotin Param-välilehden Interface-välilehdessä.

Aitaa muokataan kuvan 43 mukaisesti aidan Param-välilehdessä. Uusi aita luodaan vanhan perään painamalla Interact-työkalu valittuna aidassa näkyviä sinisiä nuolia. Tässä työssä aita rakennettiin pöydän ja robotin ympärille. Aidan tyypiksi valittiin lasi ja aidan pituuksia muunneltiin sivujen ollessa 3 000 mm ja päädyn 2 500 mm per aita.



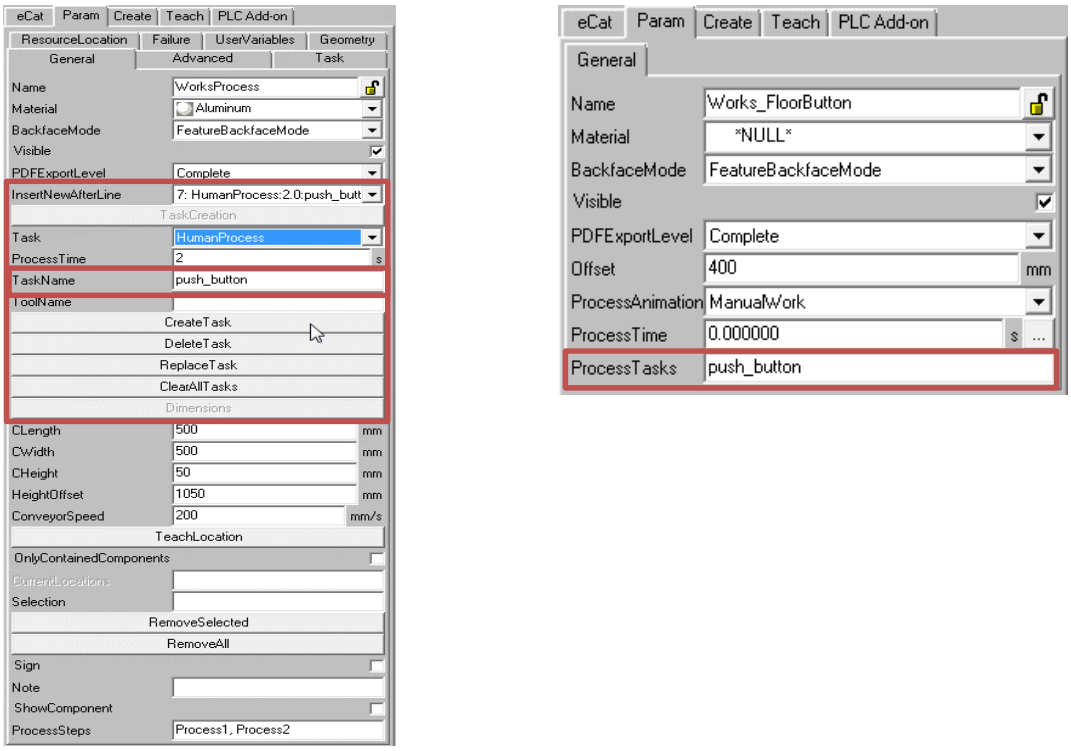
Kuva 43. Aidan pituutta ja tyyppiä säädettiin Param-välilehdessä. Uusi aita luotiin painamalla Interact-työkalun ollessa päällä sinistä nuolta.

5.2 WorksProcess

5.2.1 WorksProcessin käyttö

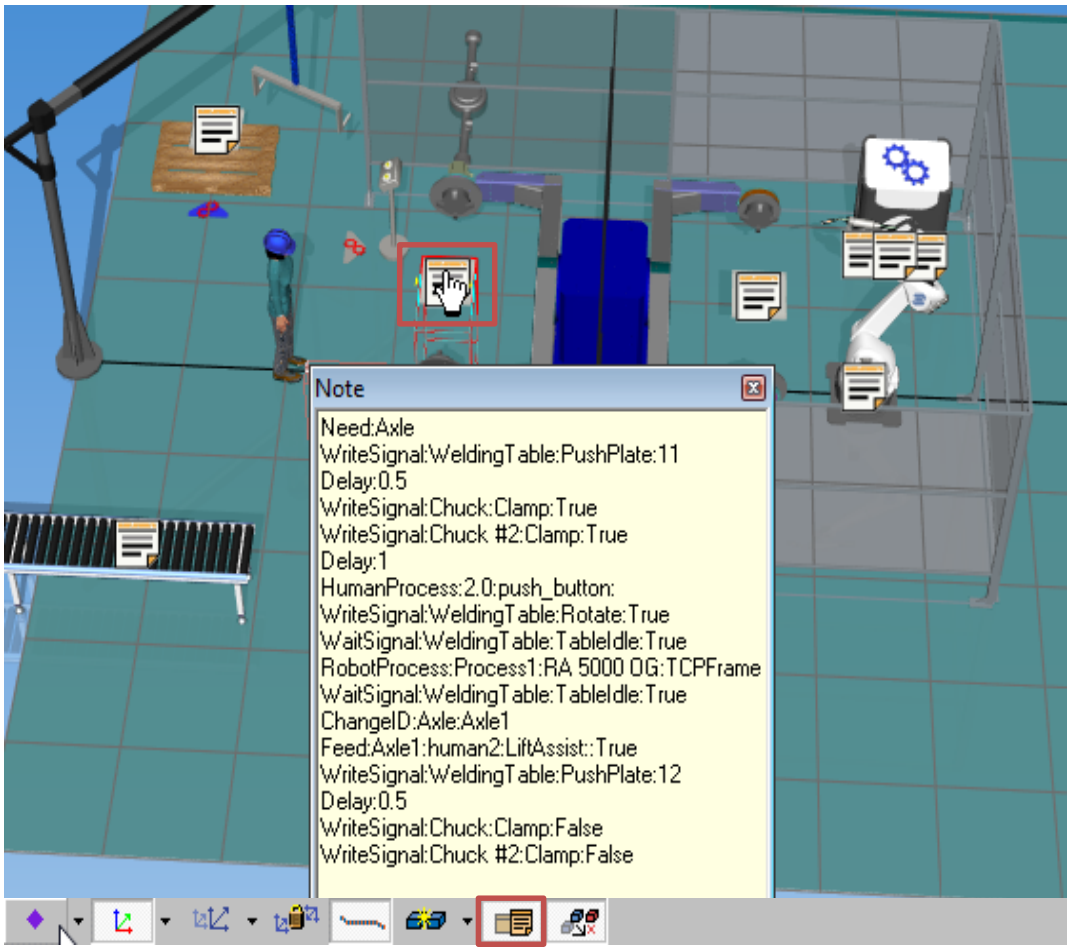
WorksProcess on simulaation sisäinen ohjelmointi, jolla ohjataan simulaation työnkulkua luomalla tehtäviä komponenteille. Näitä tehtäviä ovat muun muassa *objektien luonti, objektien syöttö työntekijälle, signaalien lähettäminen komponenteille, robottien tai muiden työkoneiden käskyttäminen ja objektin tunnuksen muuttaminen toista prosessia varten.* Tehtävät suoritetaan aina loppuun ennen kuin mennään uuteen tehtävään, mutta samaan aikaan voi toimia monia WorksProcesseja.

Tehtävä luodaan WorksProcessin Param-välilehdessä kuvan 44 mukaisesti valitsemalla tehtävä Task-valikosta ja syöttämällä halutut arvot. Kuvan 44 tapauksessa luodaan työtekijälle tehtävä napin painamisesta. Tehtävä kohdennetaan haluttuun nappiin sijoittamalla tehtävälle annettu nimi halutun napin Param-välilehdessä PocessTaskiin. Lisäksi napin painamisen ajaksi asetettiin kaksi sekuntia. Luotu tehtävä ilmestyy InsertNewAfterLine-valikkoon aikaisemmin valitun tehtävän taakse.



Kuva 44. WorksProcess-tehtävä luodaan Param-välilehdessä. Työntekijän tehtäville määritetään TaskNameen sen objektin ProcessTask, johon työntekijä halutaan tekemään tehtävää.

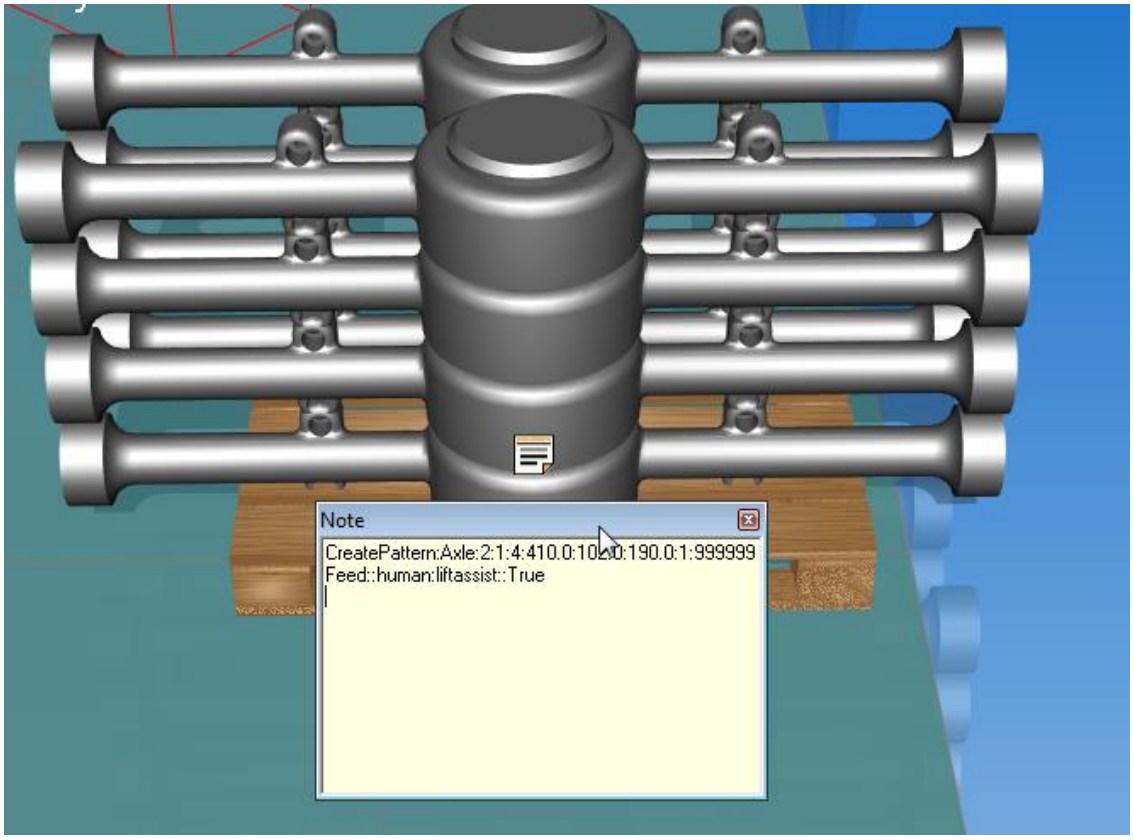
Tehtäviä voi myös kirjoittaa suoraan simulaatiopohjassa oleviin WorkProcessin Note-muistilappuihin, jonka saa näkyviin painamalla alareunassa olevaa Toggle Notes Visibility -nappia ja muistilapun kuvaa kuvan 45 mukaisesti.



Kuva 45. Muistilappu saadaan näkyviin alareunassa valitsemalla Toggle Notes Visibilityn.

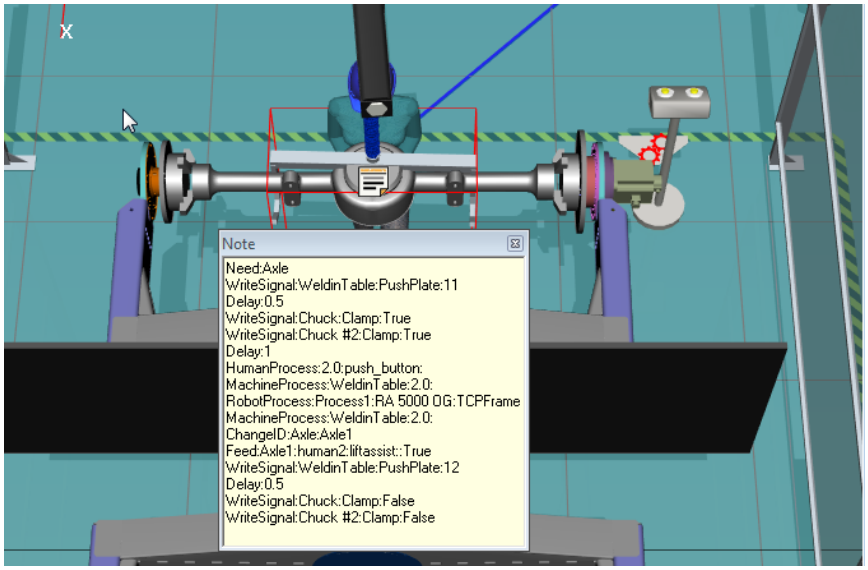
5.2.2 Hitsausprosessin WorksProcess

Tässä työssä käytettiin neljää WorkProcessia yksi luomaan akseleita kuormalavalle, kaksi ohjaamaan hitsauspöytää ja yksi akselin pois kuljettamiseen. Kuormalavan WorksProcess luo kuvan 46 mukaisesti lavan päälle kahdeksan akselia ja syöttää akseleita työntekijälle. Akseleiden lopputtua - WorksProcess luo uuden sarjan akseleita. Prosessiin on myös määritelty, että työntekijän on haettavat akselit nostoapuvälinettä käyttäen.



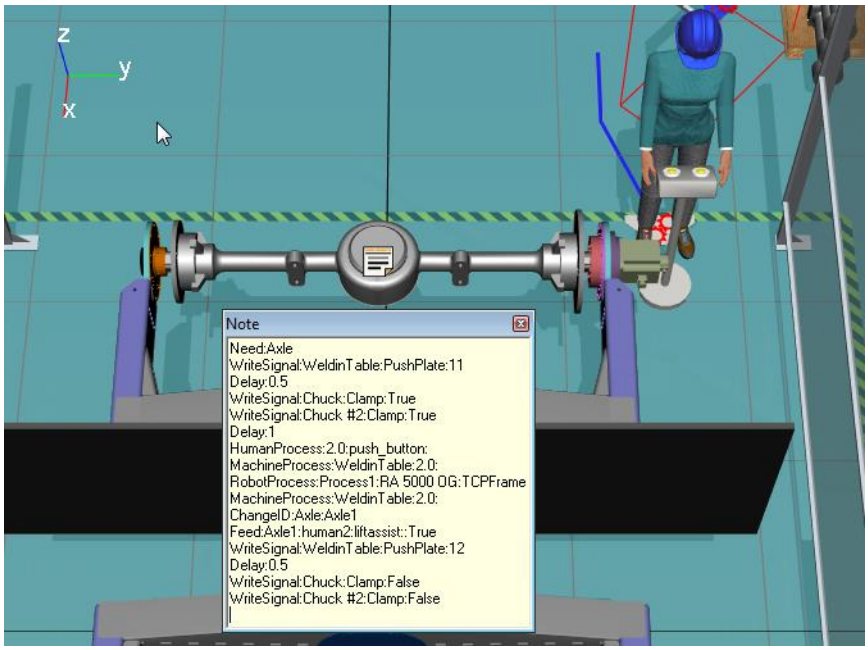
Kuva 46. Kuormalavan WorksProcess.

Ensimmäinen hitsauspöydän WorksProcess pyytää akselia, jonka työntekijä tuo kuormalavalta kuvan 47 mukaisesti.



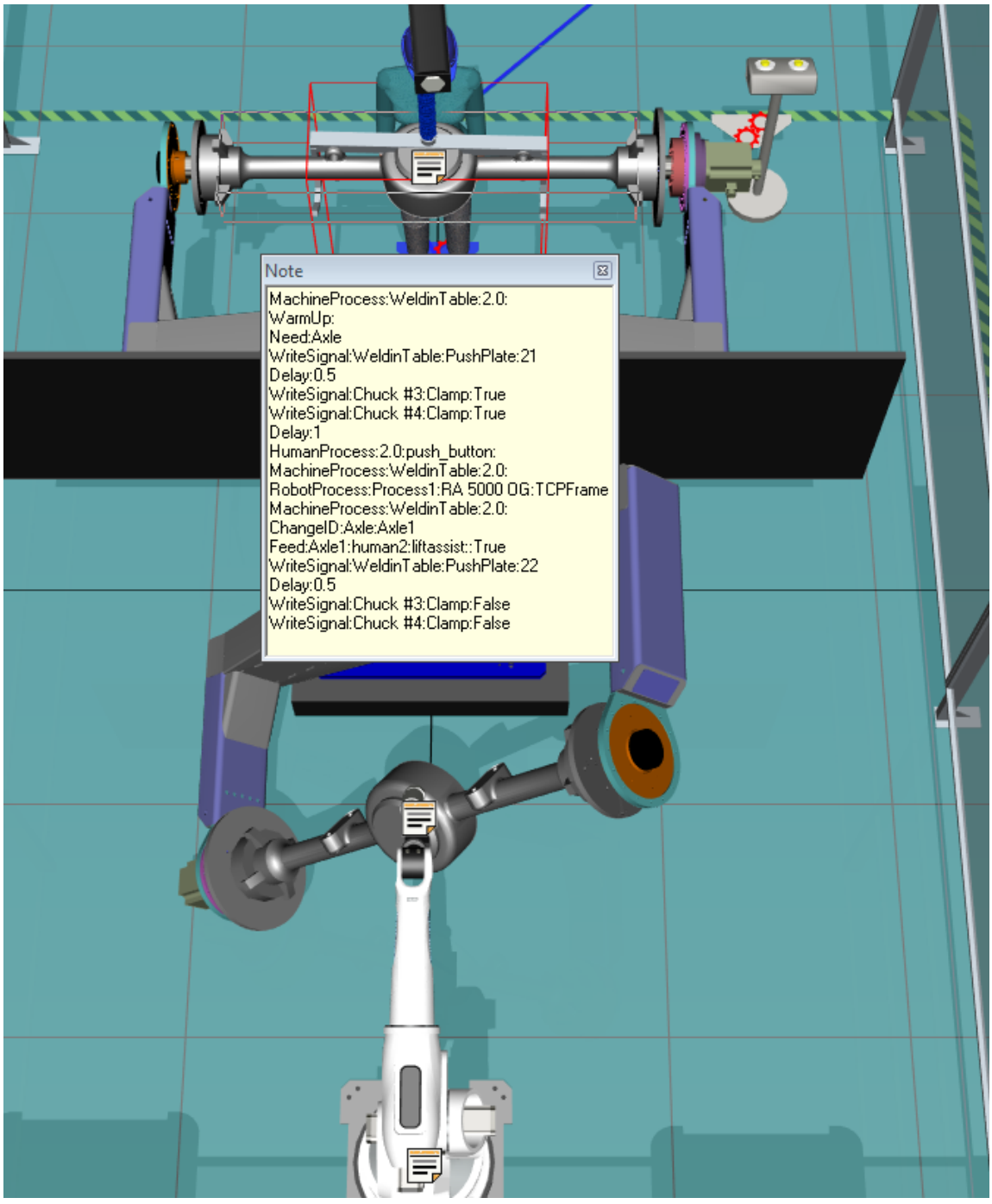
Kuva 47. Akselin tuominen hitsauspöytään.

Tämän jälkeen WorksProcess lähettää hitsauspöydälle signaalin levyjen työntämiseen, odottaa levyjen työntyneen, lähettää signaalit molemmille istukoille sulkemaan sormet, odottaa sormien sulkeutuneen ja käskee työtekijää painamaan nappia kuvan 48 mukaisesti.



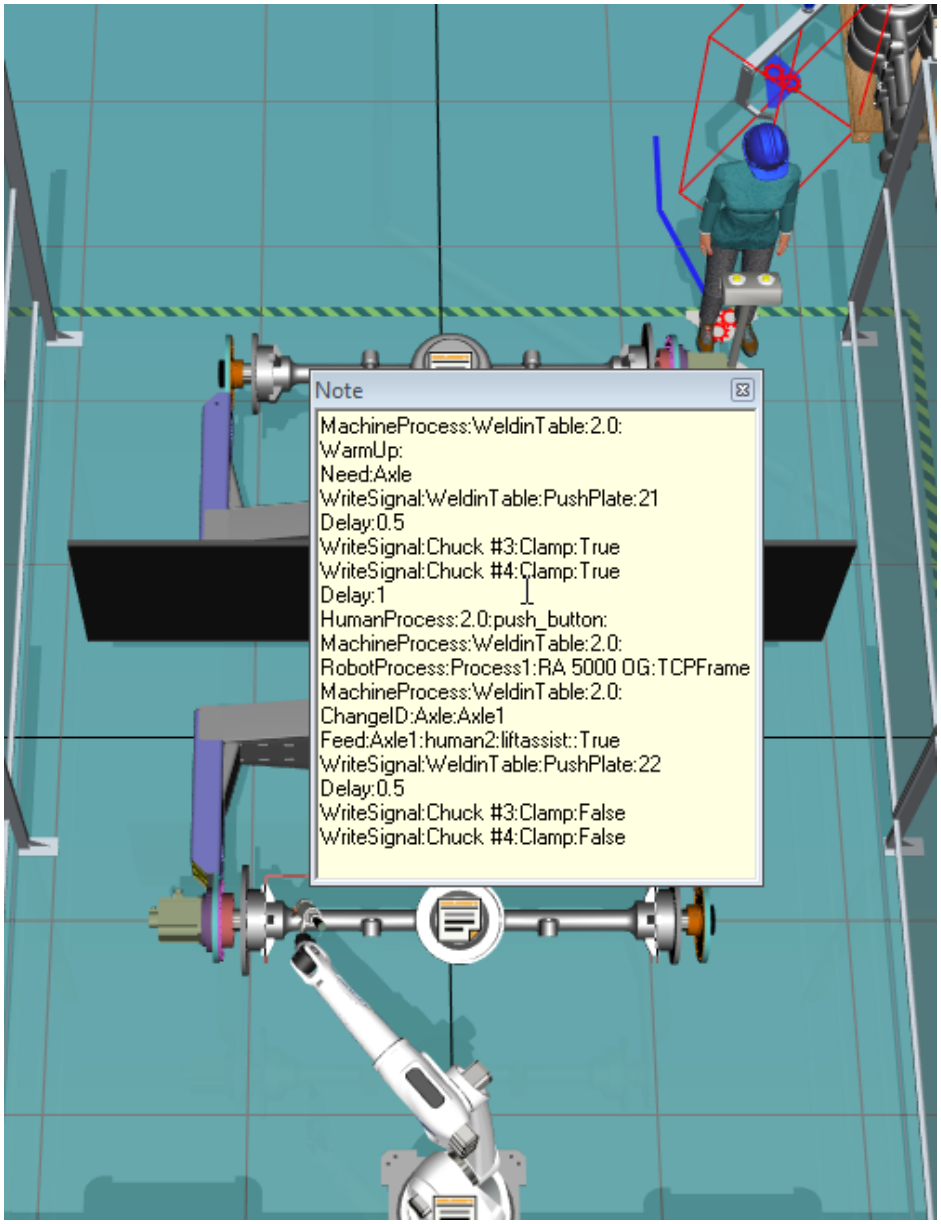
Kuva 48. Akselin kiinnittyminen hitsauspöytään ja napin painaminen.

Kun nappia on painettu, saa hitsauspöytä signaalin kääntyä. Kappaleessa 4.5 määriteltiin hitsauspöydän Python-ohjelmassa tarvitsevan kaksi signaalia ennen kuin pöytä voi kääntyä. Toinen käsky on jo annettu simulaation alussa hitsauspöydän toisessa WorksProcessissa kuvan 49 mukaisesti.



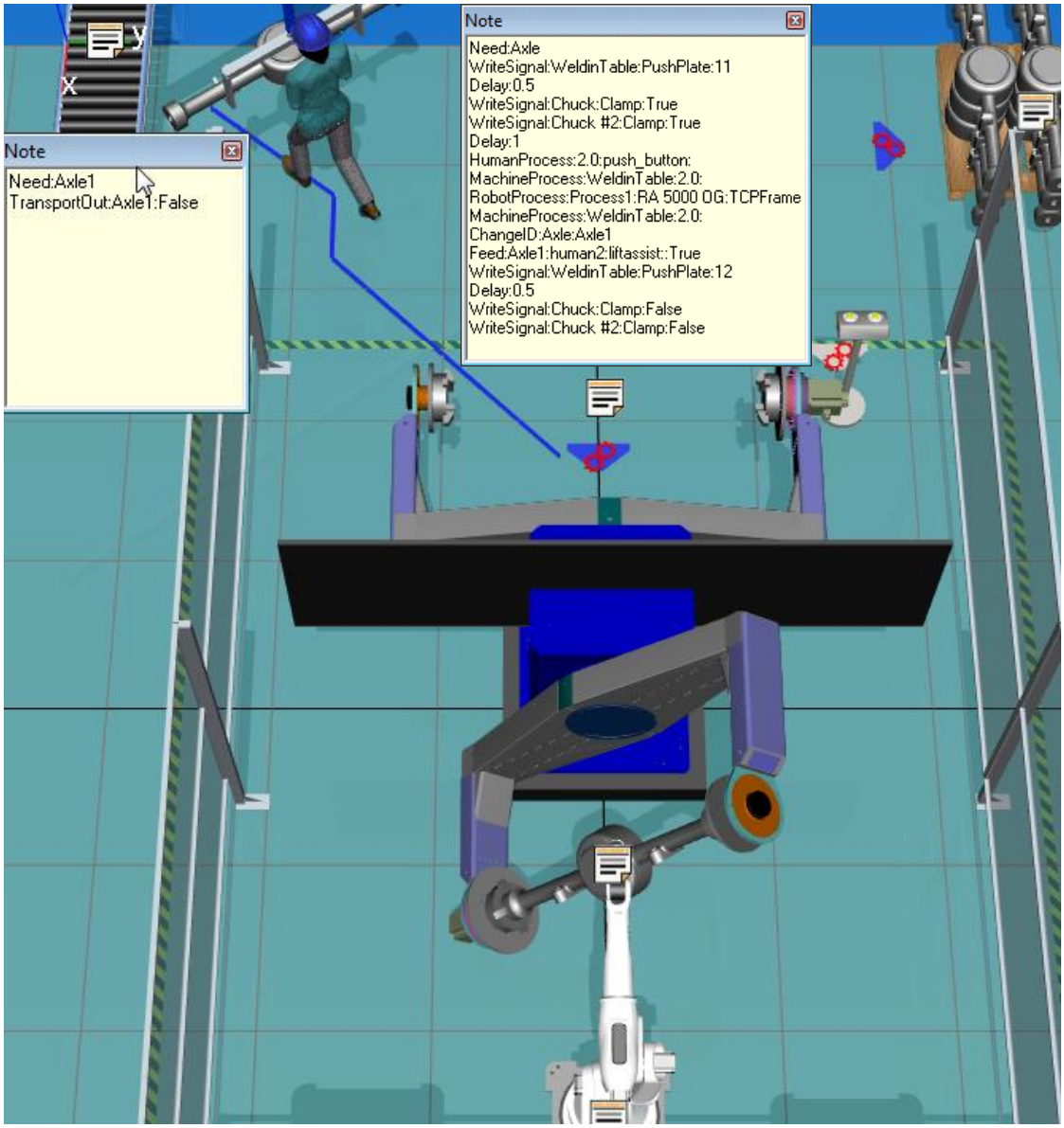
Kuva 49. Toinen hitsauspöydän WorksProcess antaa signaalin kääntyä.

Molemmat signaalit saatuaan pöytä kääntyy ja ensimmäisessä WorksProcessissa robotille annetaan tehtävä aloittaa hitausprosessinsa. Hitausprosessi käydään tarkemmin läpi kappaleessa 5.3. Toisessa WorksProcessissa tapahtuu WarmUp-tehtävä, joka tarkoittaa, että sitä ennen tapahtuneita tehtäviä ei enää toisteta simulaation aikana. Hitausprosessin aikana toinen WorksProcess on jo antanut tehtävät hakea toiselle puolelle uuden akselin, kiinnittää sen pöytään ja painaa nappia sekä antanut ensimmäisen signaalin kääntää pöytää. Näin työntekijä jää odottamaan kuvan 50 mukaisesti, että robotin hitausprosessi päättyy ja pöytä saa toisen signaalin kääntymiseen.



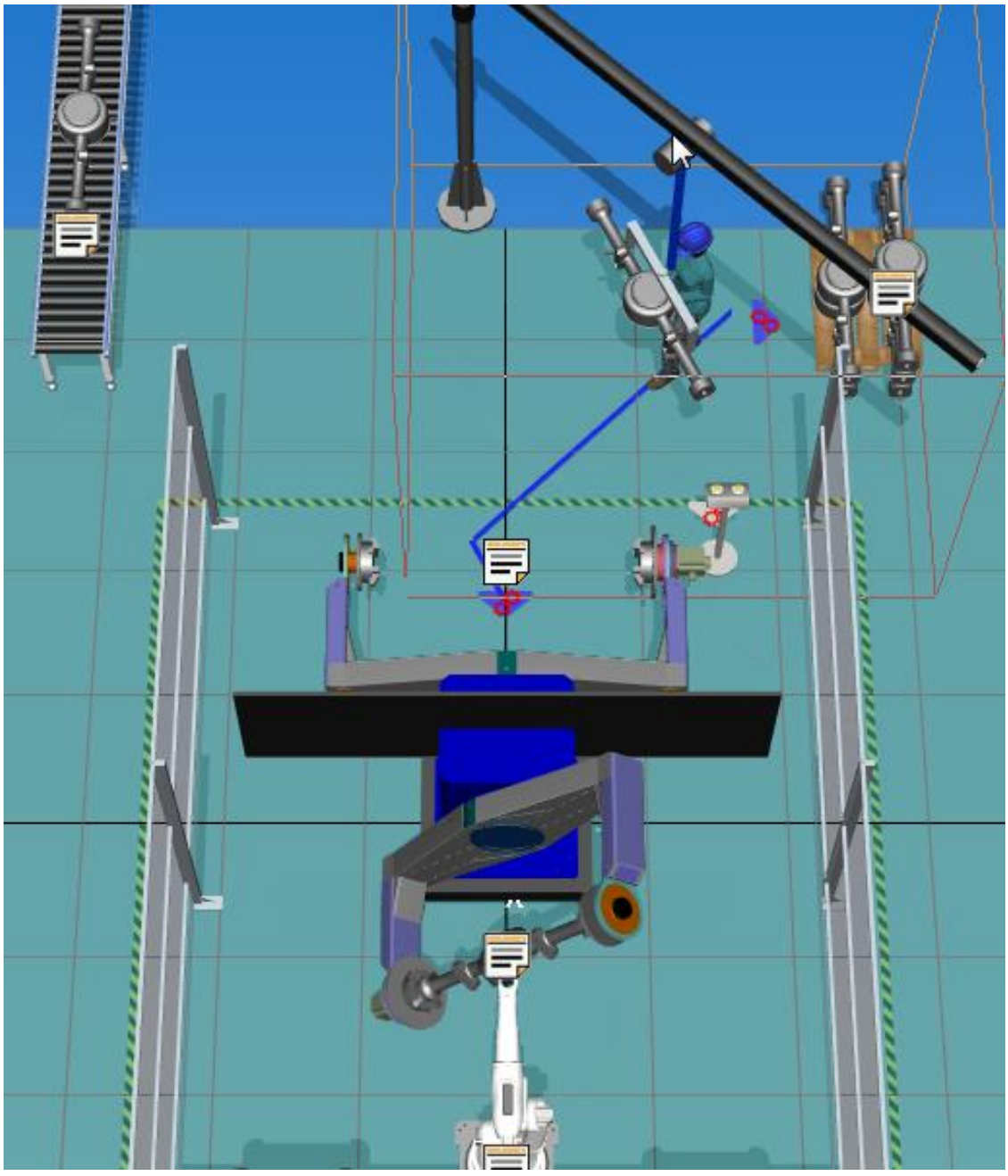
Kuva 50. Työntekijä odottaa hitausprosessin loppumista.

Pöydän käännyttyä toinen WorksProcess antaa robotille tehtävän aloittaa prosessinsa ja ensimmäinen muuttaa akselin tunnuksen Axle1:ksi, jotta työntekijä osaa erottaa hitsatun ja hitsaamattoman akselin. Tämän jälkeen ensimmäinen WorksProcess antaa työntekijälle tehtävän hakea nostoapuvälineellä akselia, jota liukuhihnana toimiva WorksProcess on pyytänyt simulaation alusta alkaen. Heti kun työntekijä tarraa nostoapuvälineellä akseliin, antaa hitsauspöydän ensimmäinen WorksProcess tehtävän avata istukoiden sormet ja hitsauspöydän levyt. Tämän jälkeen työntekijä vie akselin kuvan 51 mukaisesti liukuhihnalle ja hitsauspöydän ensimmäinen WorkProcess aloittaa tehtävänsä uudestaan.



Kuva 51. Työntekijä vie hitsatun akselin liukuhihnalle.

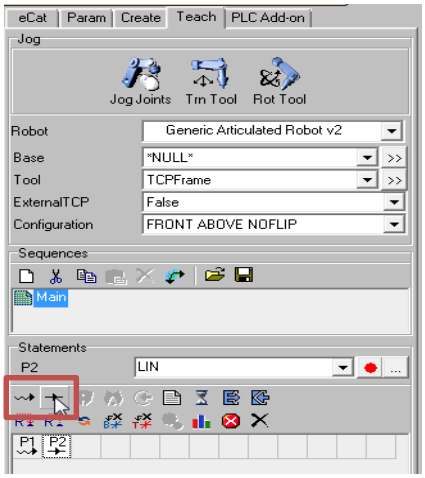
Lopulta liukuhihnan WorksProcess antaa, kuvan 52 mukaisesti, liukuhihnalle tehtävän kuljettaa akseli pois. Samalla työntekijä lähtee hakemaan uutta akselia ja simulaatio tekee samaa silmukkaa loputtomiin.



Kuva 52. Prosessi alkaa alusta.

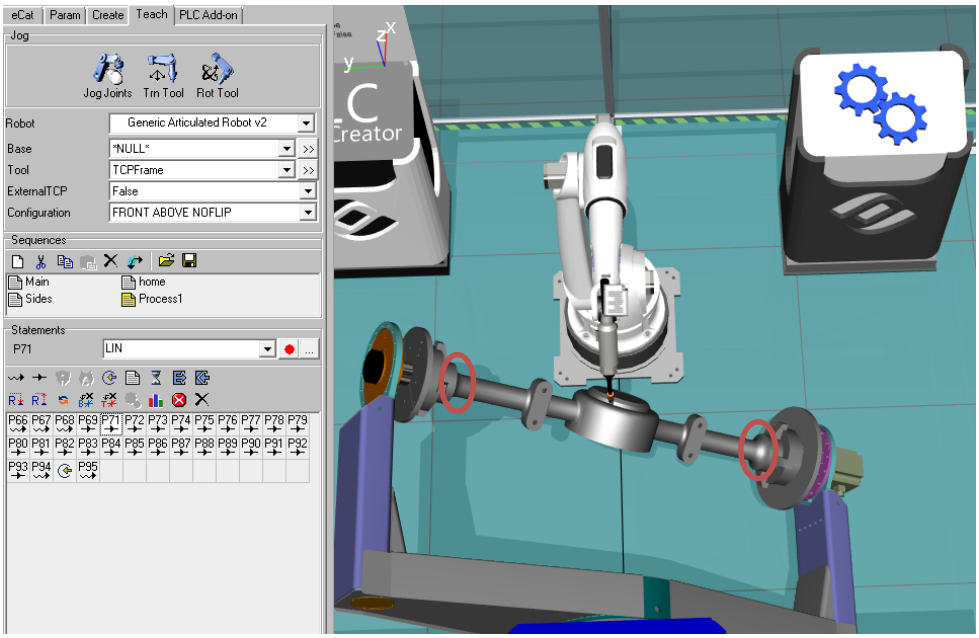
5.3 Robotin prosessin määrittäminen

Teach-välilehdessä voidaan roboteille opettaa liikeratoja. Tämä tapahtuu viemällä robotti haluttuun asentoon ja valitsemalla joko pisteestä pisteeseen liike tai suoraviivainen liike kuvan 53 mukaisesti.



Kuva 53. Robotin liikkeiden määrittäminen.

Tässä insinöörityössä simuloidaan taka-akselin päälisuojan ja sivukiinnikkeiden pysyvän kiinnityksen hitsausta kuvan 54 mukaisesti.



Kuva 54. Taka-akselin hitsaus.

Hitsauspillin haluttiin pysyvän paikoillaan hitsauksen ajan ja hitsausasennon olevan alaspäin jalkohitsauksena. Näin vain hitsauspöydän kädet ja levyt liikkuvat hitsauksen aikana. Tällöin tarvittiin kulma, jonka verran akselin on käännyttävä, jotta hitsauspilli osuu aina suoraan taka-akselin kulmaan. Tämä saatiin mittaamalla kuvan 55 mukaisesti measurement-työkalulla kulman korkeus (82 mm) ja leveys (144 mm) akselin keskustasta ja laskemalla kulma kaavalla 1:

$$\alpha = tan^{-1}\left(\frac{144}{82}\right) = 60.34° \approx 60° \quad (1)$$

Päälisuojan hitsauksen kaartaessa ympyrärataa on siis käden ja levyjen kääntäjän arvo joko 0 tai +/- 60 astetta aina 90 asteen välein. Simulaation näyttämiseksi sulavalta määritettiin myös 90 asteen väliset arvot jakamalla 90 astetta 15 asteen välisiin kulmiin ja kääntäjien arvot laskettiin kaavoilla 2 ja 3:

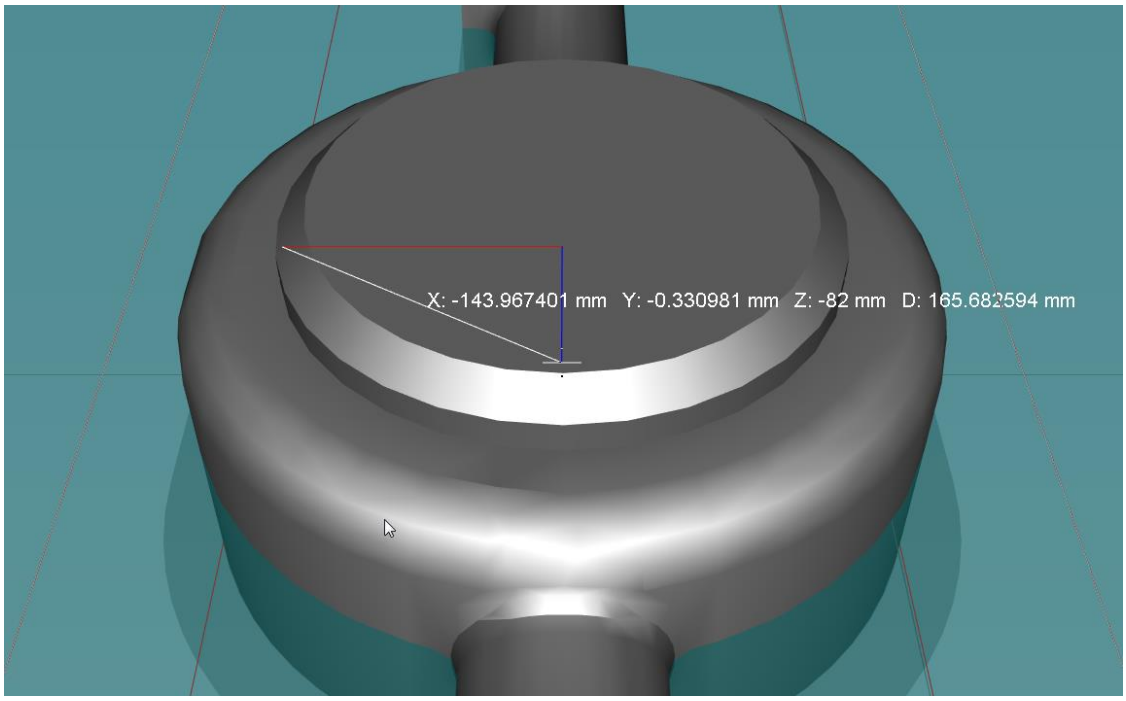
$$x = 60° * \cos(\alpha) \quad (2)$$

$$y = 60° * \sin(\alpha) \quad (3)$$

Tuloksiksi saatiin taulukon 1 mukaiset arvot, joissa x kuvaa levyjen kääntäjää ja y käden kääntäjää.

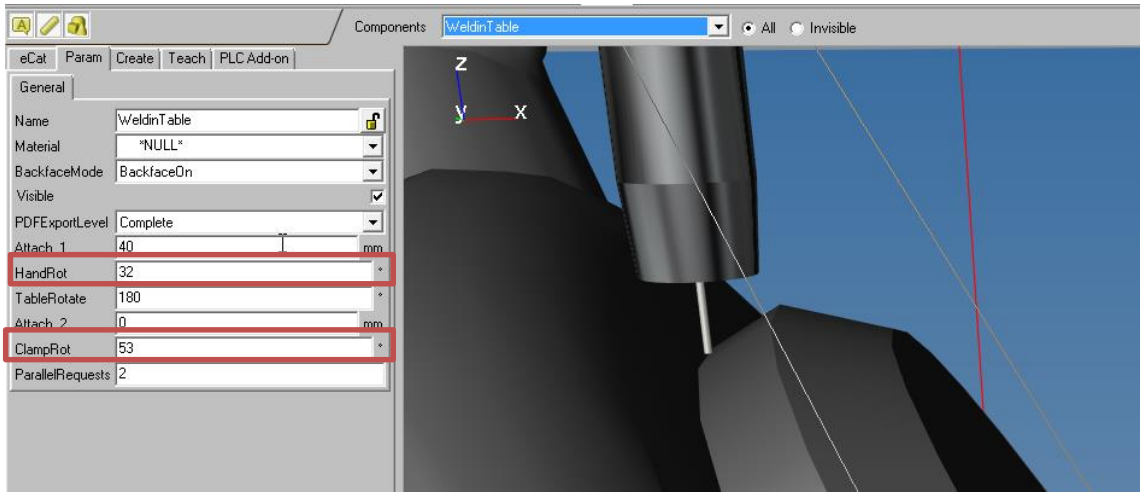
Taulukko 1. Kääntäjien arvot

α(°)	**x(°)**	**y(°)**
0	60	0
15	58	15
30	52	29
45	43	43
60	29	52
75	15	58
90	0	60



Kuva 55. Akselin ja päälisuoja liitoskohdan etäisyyden mittaus akselin keskipisteestä.

Koko ympyrän pisteet saatiin muodostettua asettamalla aina 90 asteen välein joko levyjen tai käsien tai molempien arvot miinusmerkkisiksi. Nämä arvot toimivat suuntaa antavina lopullisten arvojen ollessa kuitenkin hyvin lähellä laskettuja (kuva 56), ja koska liike on jaettu tasaisiin osiin, on myös simulaatio sulavamman näköinen.



Kuva 56. Käden ja levyjen kääntäjien arvot 30 asteen kulmassa.

6 PLC-rajapinta Beckhoff TwinCATillä ja tutoriaali

6.1 PLC-rajapinta

PLC yhdistetään simulaation PLC Add-on -lisäosassa suoraan komponenttien osiin tai yhdistämällä signaalit PLC-rajapintojen kautta. Varsinainen yhdistäminen käydään läpi liitteenä olevassa tutoriaalissa.

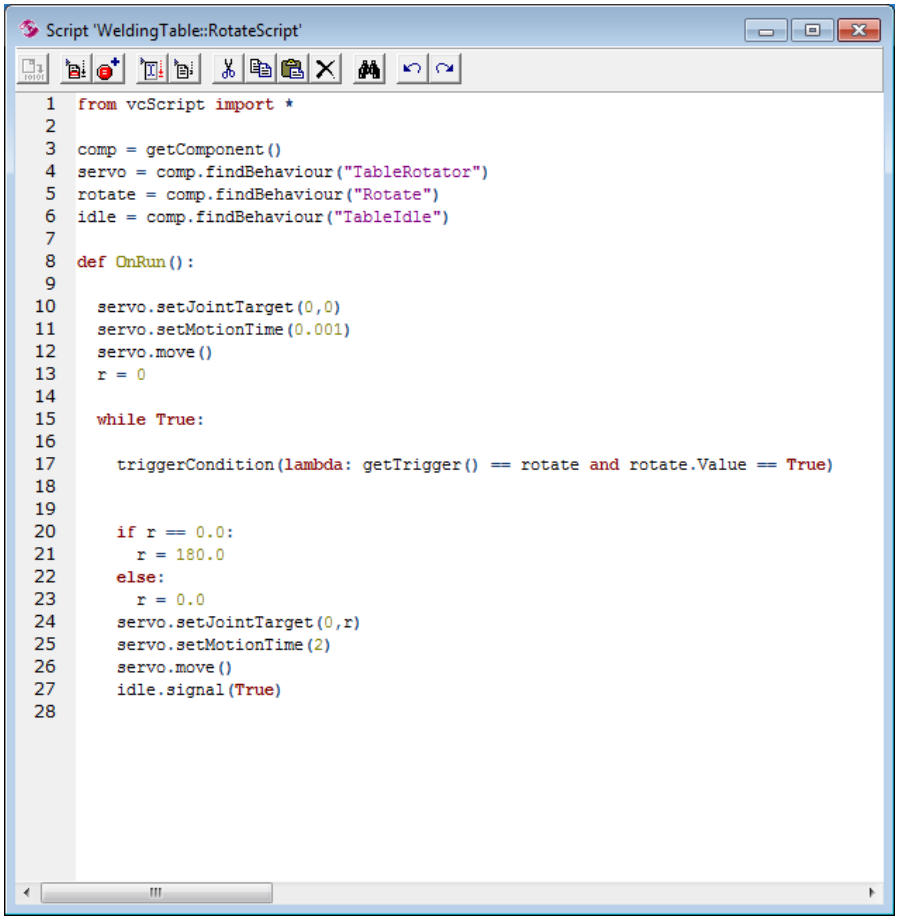
PLC:n ja simulaation välisen keskustelun saamikseksi mahdollisimman hyvin toimivaksi tarvitsee huomioida, että TwinCat lukee aina tietyllä aikavälillä, tässä työssä 50 ms:n välein, signaalit. Tämän seurauksena jos simulaation ajonopeus on liian suuri, PLC ei pysy perässä ja menettää signaalit simulaation kanssa. Lisäksi on huomioitava, että kun simulaatiota ja PLC:tä ajetaan kahdella erillisellä koneella etäyhteytenä, on tarpeellista, että molemmissa koneissa on TwinCat asennettuna. Muuten yhteydestä tulee liian hidas yhdistämään signaaleja.

Tässä työssä PLC:nä käytettiin TwinCat 3:n PLC:tä, eli PLC tehtiin ohjelman kautta ja tietokone toimii prosessorina PLC:lle. PLC:n rooli työssä on toimia logiikkana pöydän kääntymiselle ja pöydän kiinnikkeille.

6.2 Tutoriaali

Tutoriaalin ideana on opettaa, miten käytetään Works-kirjaston PLC Signal Creatoria, miten tehdään komponenteissa olevista signaaleista yhteenliitettäviä PLC:hen ja miten yhdistetään simulaation ja TwinCATin signaalit PLC Add-on -välilehdellä. Tutoriaali on suunnattu henkilöille, joilla on tietämys TwinCat:n käytöstä, mutta jotka tarvitsevat opastusta 3DAutomaten käytössä. Tutoriaali on tämän työn liitteenä (Liite 1).

Versio hitsausprosessista, josta tutoriaalia lähdetään tekemään, poikkeaa hieman aikaisemmissa kappaleissa tehdystä versioista. Suurimpana erona on hitsauspöydän kääntymistä ohjaava Python-koodi, joka on tehty eri periaatteella (kuva 57), jotta pöydän yhdistäminen PLC:hen olisi helpompaa.



Kuva 57. Tutoriaalin pyödän kääntymistä ohjaava Python-koodi.

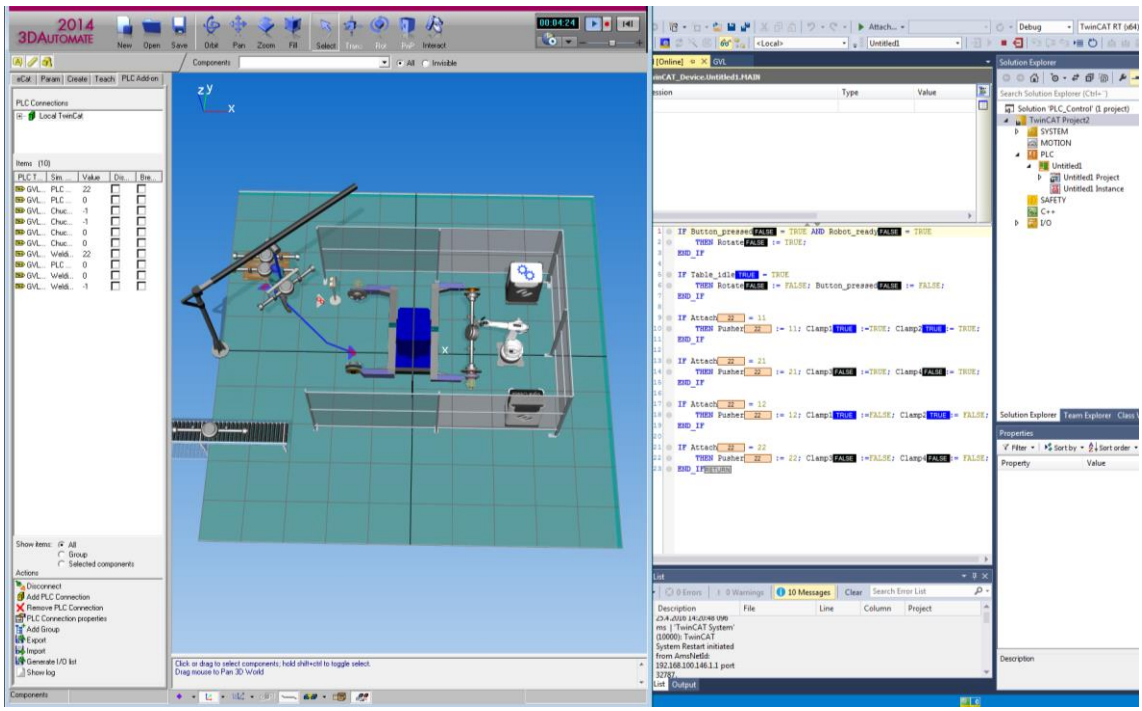
Tutoriaalissa pöydän kääntymisen ohjaus on siirretty PLC:n puolelle, josta pöytä saa käskyn kääntyä vasta, kun simulaatiosta on saatu tiedot, että nappia on painettu ja robotti on valmis. Koska tutoriaali on suunnattu TwinCATia jo osaaville, annetaan tutoriaalissa vain yleiskuva, mitä TwinCATiin on tehty. Mikäli tutoriaalia tekee henkilö, jolle TwinCat ei ole tuttu, tutoriaalin liitteenä on valmis ohjelma ja tutoriaalissa näytään, miten se laitetaan käyttövalmiiksi.

7 Yhteenveto

Työn tarkoituksena oli simuloida hitausprosessi, jota ohjataan PLC:llä, ja tehdä tutoriaali PLC:n liittämisestä simulaatioon.

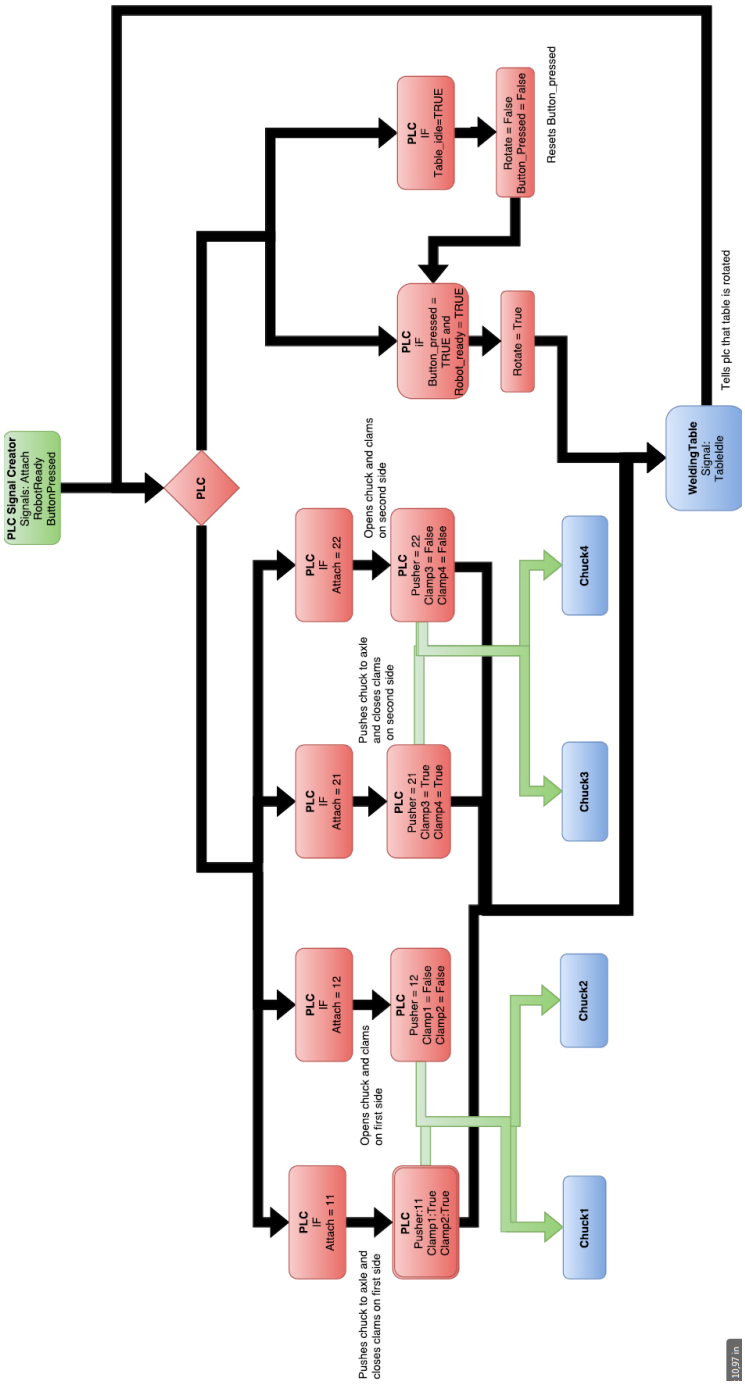
7.1 Lopputulos

Lopputuloksena saadussa simulaatiossa työntekijä hakee kuormalavalta nostoapuvälinettä käyttäen taka-akselin ja tuo sen hitauspöytään. Hitauspöydässä olevat istukat työntyvät akselia vasten, minkä jälkeen istukan sormet lukitsevat akselin paikoilleen. Tämän jälkeen työntekijä menee painamaan nappia, joka antaa pöydälle käskyn kääntyä. Pöydän käännyttyä toisella puolella odottava robotti aloittaa hitausprosessinsa, jossa se hitsaa väliaikaisella kiinnityksellä olleet akselin päälisuojuksen ja sivukiinnikkeet. Samalla työntekijä hakee uuden akselin ja painaa nappia. Pöytä kääntyy vasta, kun se on saanut tiedon sekä napilta että robotilta, että se voi kääntyä. Hitauksen valmistuttua pöytä kääntyy ja työntekijä vie hitsatun akselin liukuhihnalle, joka kuljettaa akselin eteenpäin. Työntekijä hakee taas uuden akselin ja prosessi jatkuu silmukkana loputtomiin. Työnkulku näkyy kuvassa 58 sekä liitteenä olevan PDF-tiedoston videossa (Liite 2).

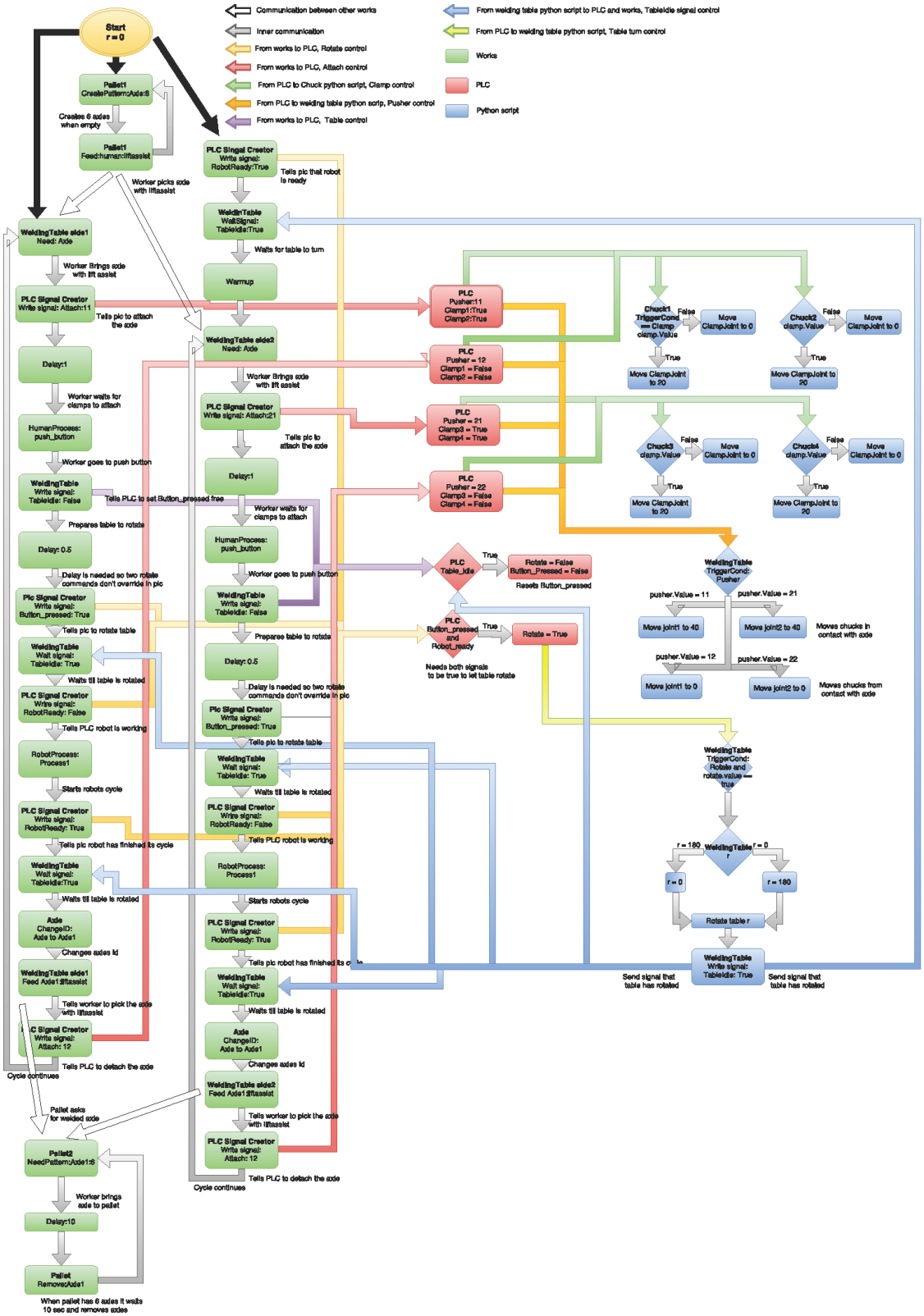


Kuva 58. Työnkulku ja simulaatio.

Koko prosessia ohjaa WorksProcess, joka lähettää PLC:lle tiedot napin painamisesta, robotin valmiudesta ja istukoiden kiinnittämisestä. Kun sekä nappia on painettu että robotti on valmis, pöytä kääntyy. Pöydän käännyttyä se lähettää tiedon PLC:lle, että pöytä on kääntynyt ja nollaa samalla napin. Riippuen käskystä, PLC lähettää hitsaupöydän työntäjille ja istukoille käskyn avata tai sulkea akselin kiinnitys. PLC:n tehtävää prosessissa havainnollistaa kuva 59 sekä koko prosessia kuva 60.



Kuva 59. PLC:n prosessikaavio.



Kuva 60. Koko hitsausprosessin prosessikaavio.

7.2 Päätelmä

Insinöörityö oli vain pintaraapaisu tehdassimulointiin ja PLC:n käyttämiseen siihen käytetyn lyhyen ajan takia. Työtä tehdessä huomattiin, että jos prosessista tehtävästä simulaatiosta halutaan myös todenmukaisia tietoja, on simulaation oltava mahdollisimman lähellä oikeaa. Todenmukaisen simulaatioon pääseminen vaatii mallintajalta lajaa tuntemusta vastaavista prosesseista, jotta hän pystyy erottamaan, miten tilanne käytännössä oikeasti tehdään ja saa sen myös mallinnettua simulaatioon. Hyvällä mallintajalla, jolla on paljon kokemusta automaatioprosesseista, vaikuttaisi olevan lähes rajattomat mahdollisuudet luoda simulaatioita.

Työhön asetetut tavoitteet kuitenkin saavutettiin ja työ oli onnistunut. Simulaatio näyttää melko todenmukaiselta ja sulavalta, sekä PLC ohjaa hitsauspöydän ja robotin toimintaa. Lisäksi tutoriaalissa käydään selkeästi ja yksityiskohtaisesti läpi PLC:n yhdistäminen simulaatioon.

Insinöörityön simulaatiota voi vielä kuitenkin kehittää paljon. Esimerkiksi PLC:tä voisi käyttää enemmän. Nyt pöydän ohjauksesta, josta ison osan voisi siirtää PLC:lle, sisältyy simulaation koodiin. Hitsausprosessia voi myös muuttaa luomalla sille liikeradat ohjelman työkaluilla, sillä nyt trigonometrian avulla tehty prosessi toimii vain, jos hitsausrata menee ympyränmuodossa ja akseli osuu sopivasti keskelle. Näitä ei lähdetty kuitenkaan tekemään, kun nykyinen hitsausprosessi osoittautui toimivaksi ja projektiin käytettävä aika oli päättymässä.

Lähteet

1 Visual Components Oy. Verkkosivu. http://www.visualcomponents.com/. Luettu 13.4.2016

2 Visual Components Oy. Verkkoartikkeli. http://www.visualcomponents.com/insights/articles/factory-simulation-vs-visualization-difference/. Luettu 19.4.2016

3 Virtual Commissioning of Automated Systems. Verkkodokumentti. http://cdn.intechopen.com/pdfs/37992/InTech-Virtual_commissioning_of_automated_systems.pdf. Luettu 14.4.2016.

4 Delfoi Oy. Verkkosivu. http://www.delfoi.com/web/solutions/robotiikka/en_GB/offline/. Luettu 19.4.2016.

5 AMCI. Advanced Micro Controls Inc. Verkkosivu. http://www.amci.com/tutorials/tutorials-what-is-programmable-logic-controller.asp. Luettu 18.4.2016.

6 Motoman Oy. Verkkosivu. http://www.motoman.co.uk/en/products/positioners/product-view/?tx_catalogpositoner_pi1%5Buid%5D=117&cHash=fe9687b9319618207f8b751b4f6adf33. Luettu 3.3.2016.

Liite 1. Tutoriaali: Connecting simulation with TwinCAT 3

Connecting simulation with TwinCAT 3

Related Files:

Visual Components layout and TwinCAT 3 program: TwinCat3_Tutorial.zip

Requirements

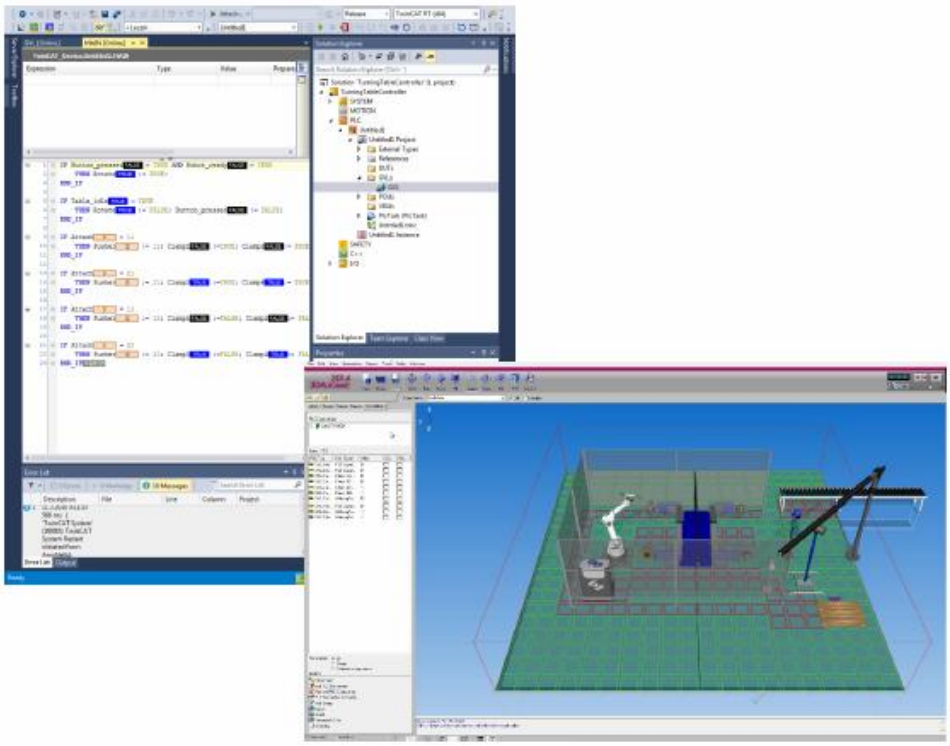
3DSimulate or higher
Beckhoff TwinCAT 3 PLC Add-On

OBJECTIVE

This example demonstrates the connectivity between Visual Components PLC Add-On and Beckhoff TwinCAT 3. The PLC Add-on connects tags defined in the connected Beckhoff PLC to signal input- outputs (I/Os) defined in the simulation model. The communication is bidirectional.

The premade layout consist of a robot, welding table, conveyor, worker, axle feeder pallet and lift assist. The worker brings an axle using a lift assist to the table and the table rotates to robot so it can weld the axle. At same time, the worker brings a new axle. After welding, the table rotates again and the worker moves the axle to the conveyor.

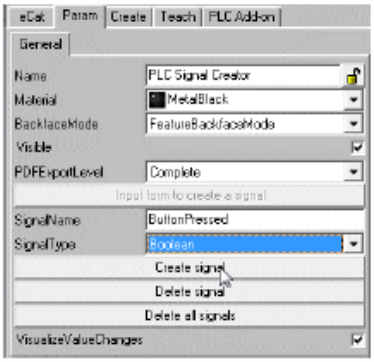
The simulation needs a command to block table from rotating while robot is working. That will be created in PLC. Also signals to attach the axle and rotate the table are changed to the PLC controlled.



Changes in the simulation

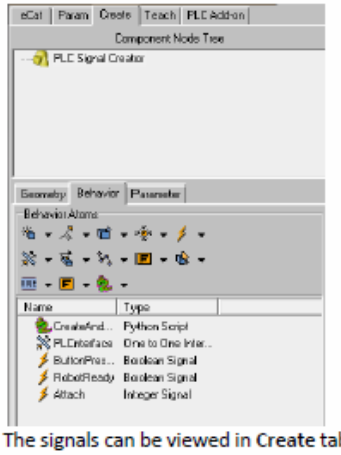
In this section, you will learn to create PLC connectable signals with **PLC Signal Creator** and within an object.

1. Start 3DCreate© and locate the downloaded layout **WeldingTablePLCTutorial.vcm** and load it into the 3D world.
2. Bring **PLCSignalCreator.vcm** to the layout, move it in the corner inside the fence and select it.
3. In the **Param** tab in the **SignalType** box type **ButtonPressed**, change the **SignalType** to **Boolean** and click the **Create signal** button.



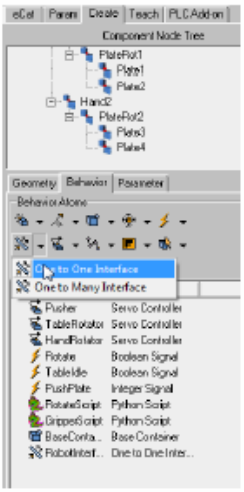
This creates a signal and connects it automatically to a PLC Interface.

4. Create two more signals, one **Boolean** named **RobotReady** and one **Integer** named **Attach**.

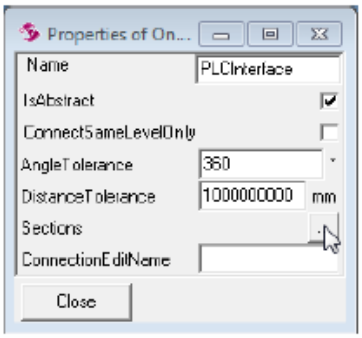


The signals can be viewed in **Create** tab in **PLC Signal Creator**.

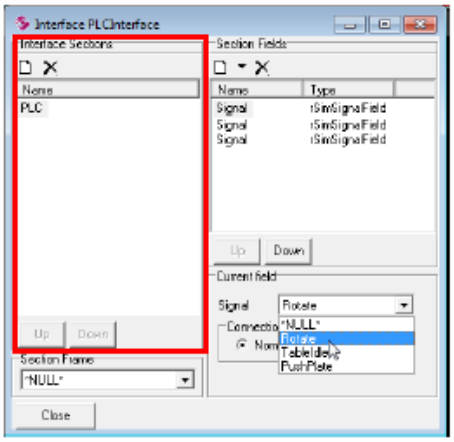
5. Select **WeldingTable**, in the **Create** tab click the **Behavior** tab and create **One to One Interface**.



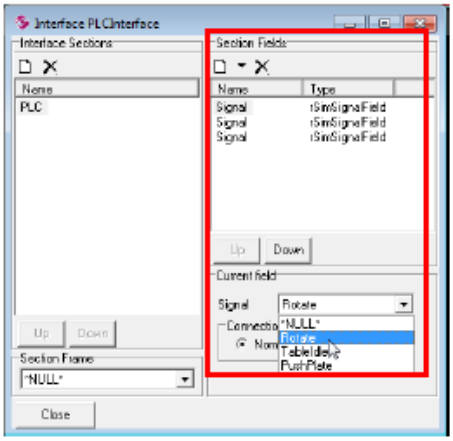
6. Name it **PLCInterface**, check **IsAbstract** and click the **Sections** button.



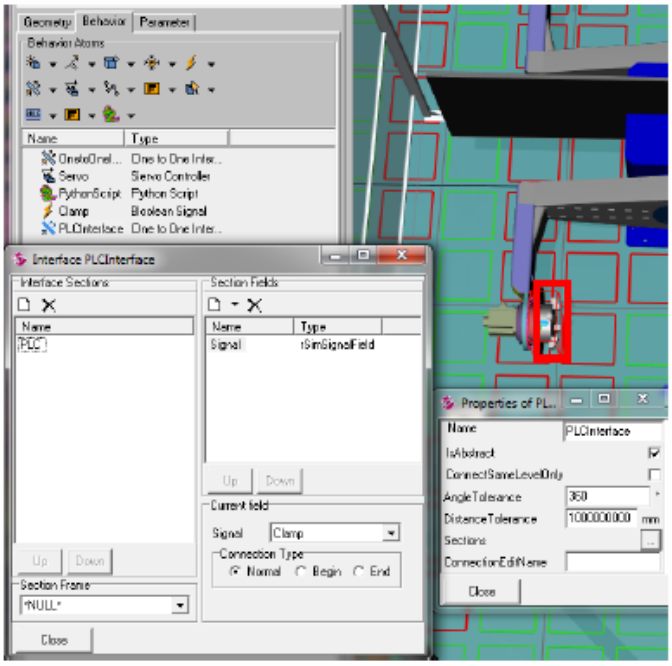
7. In the **Interface Sections** create a **New Section** and name it **PLC**.



8. In the **Sections Fields** box create three **Signal Fields**. Select one **Signal Field**, in **Current field** set Signal to **Rotate**. For second **Signal Field** set **Signal** to **TableIdle** and for third **Signal Field** set **Signal** to **PushPlate**.



9. Select one of the **Chuck** components in layout, in **Create** tab click **Behavior**, create **One to One Interface** and do same things as was done in **WeldingTable** but just create one **Signal Field** and put **Clamp** as signal. Do this to **Chuck #2**, **Chuck #3** and **Chuck #4**.



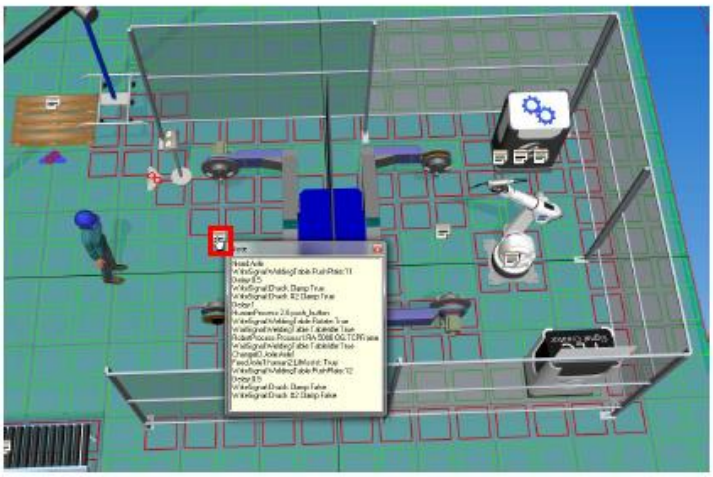
Signals in table and chucks are now connectable to PLC with **PLC Add-On**.

Works process changes

Now that the signals are connectable with PLCs, we need change tasks in **WorkProcesses** to send the signals to PLC.

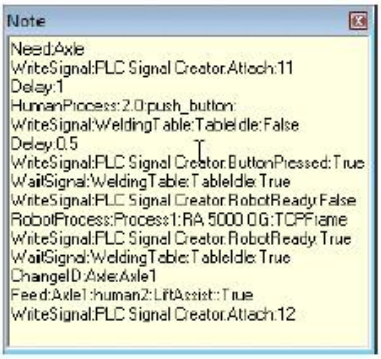
1. In **Status** bar click **Toggle Notes Visibility** to see the notes that include the works tasks. Tasks form a command chain that controls the simulation.

2. Click the note on the human side.



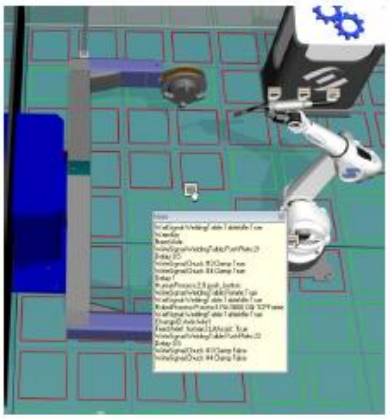
3. Change the text to:

Need:Axle
WriteSignal:PLC Signal Creator:Attach:11
Delay:1
HumanProcess:2.0:push_button:
WriteSignal:WeldingTable:TableIdle:False
Delay:0.5
WriteSignal:PLC Signal Creator:ButtonPressed:True
WaitSignal:WeldingTable:TableIdle:True
WriteSignal:PLC Signal Creator:RobotReady:False
RobotProcess:Process1:RA 5000 OG:TCPFrame
WriteSignal:PLC Signal Creator:RobotReady:True
WaitSignal:WeldingTable:TableIdle:True
ChangeID:Axle:Axle1
Feed:Axle1:human2:LiftAssist::True
WriteSignal:PLC Signal Creator:Attach:12



4. On the other side change the text to:

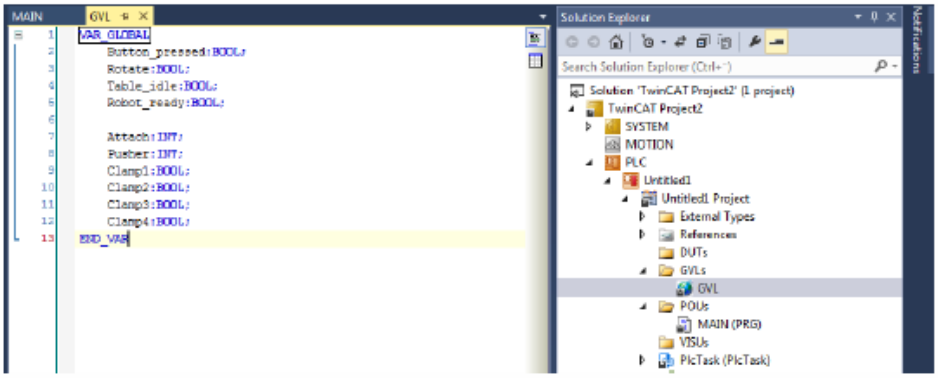
WriteSignal:PLC Signal Creator:RobotReady:True
WaitSignal:WeldingTable:TableIdle:True
WarmUp:
Need:Axle
WriteSignal:PLC Signal Creator:Attach:21
Delay:1
HumanProcess:2.0:push_button:
WriteSignal:WeldingTable:TableIdle:False
Delay:0.5
WriteSignal:PLC Signal Creator:ButtonPressed:True
WaitSignal:WeldingTable:TableIdle:True
WriteSignal:PLC Signal Creator:RobotReady:False
RobotProcess:Process1:RA 5000 OG:TCPFrame
WriteSignal:PLC Signal Creator:RobotReady:True
WaitSignal:WeldingTable:TableIdle:True
ChangeID:Axle:Axle1
Feed:Axle1:human2:LiftAssist::True
WriteSignal:PLC Signal Creator:Attach:22



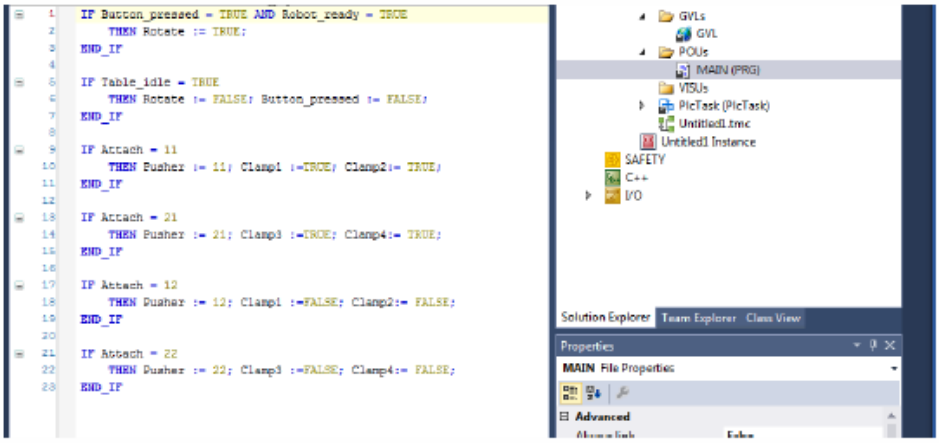
TwinCAT setup

1. The Beckhoff TwinCAT© needs to be installed and activated. Make sure the following requirements are met:
   - TwinCAT© needs to be run in **Run mode.** Locate the files that came with this tutorial and open the **PLC_Control** folder and open the **PLC_Control.sln** file.
   - In the top bar, choose **Activate configuration** and **Login** to build the program and make it connectable with the simulation.
   - Agree when asked to download the program to the PLC

2. Alternatively, the PLC program can be created from scratch by creating new project and PLC Project. Also create a new global variable list and type in the following variables



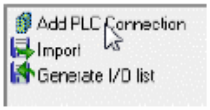
3. In the Main POU, type in the following commands:



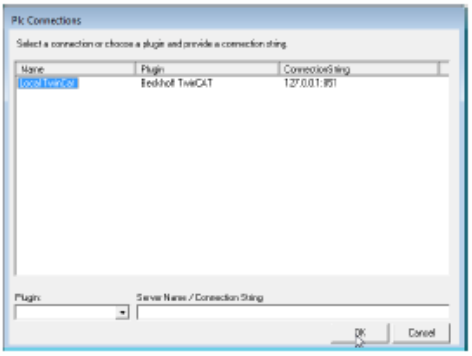
4. Click Activate Configuration, OK to all messages and login

Link the signals

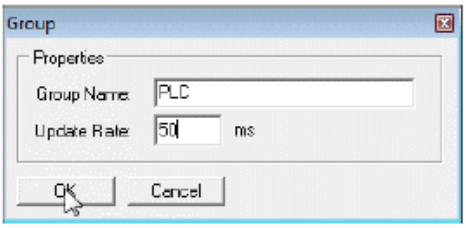
1. Go back to the simulation software
2. In **PLC Add-on** tab click Add PLC Connection



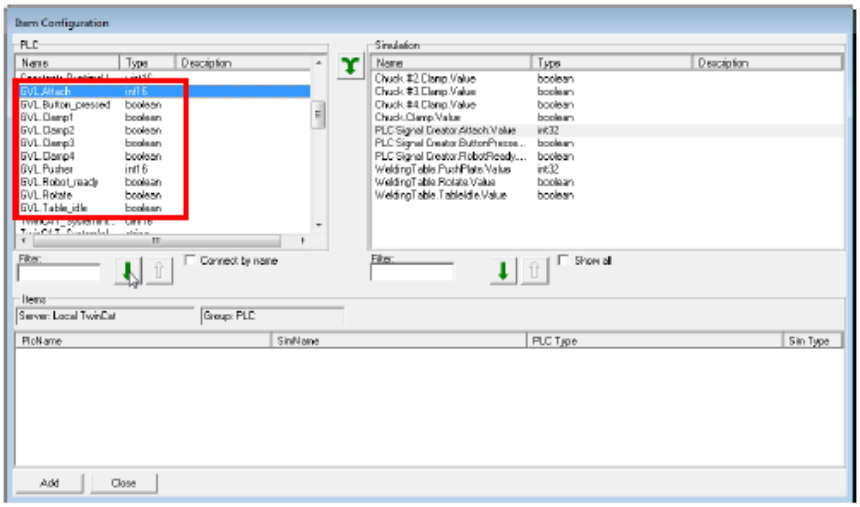
3. Select **Local TwinCat** and click OK



4. Change **Group Name** to **PLC** and **Update Rate** to **50 ms**.



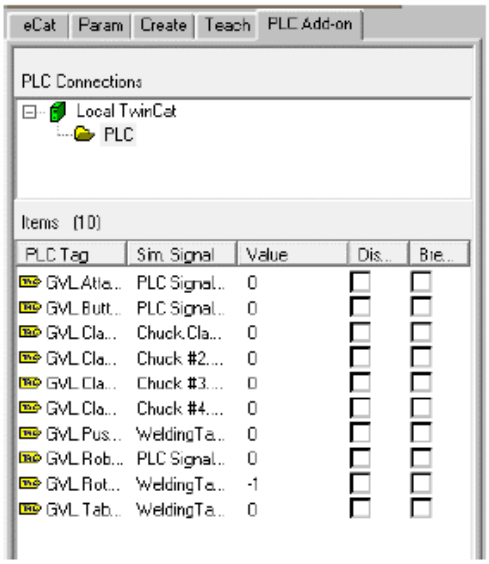
5. In **PLC** find all variables starting GVL and connect them by selecting a variable and the corresponding signal, then clicking the green down arrow. If there are missing signals in simulation box, check if the **PLCInterfaces** we created earlier have any errors and try again.



Connect PLC variables with matching Simulation signals. The signal pairs are:

PLC		**Simulation**
GVL Attach	-	PLC Signal Creator.Attach.Value
GVL Button_pressed	-	PLC Signal Creator.ButtonPressed.Value
GVL Clamp1	-	Chuck.Clamp.Value
GVL Clamp2	-	Chuck #2.Clamp.Value
GVL Clamp3	-	Chuck #3.Clamp.Value
GVL Clamp4	-	Chuck #4.Clamp.Value
GVL Pusher	-	WeldingTable.PushPlate.Value
GVL Robot_ready	-	PLC Signal Creator.RobotReady.Value
GVL Rotate	-	WeldingTable.Rotate.Value
GVL Table_idle	-	WeldingTable.TableIdle.Value

When all are connected click **Add**. The signals should appear in **Items** bar.



6. Run the simulation

Attention: The real time mode should be used to run the simulation, a too fast simulation speed will result in lost signals. This may result to that table doesn't turn even in normal speed. This can be fixed by forcing **Button_pressed** signal value to **TRUE** in TwinCAT or reconnecting with TwinCat in **PLC Add-on** tab.

- Login to TwinCAT. 

- Check that **Local TwinCat** is connected in PLC Add-on
- Run the simulation in 3DCreate by pressing **Play**.

In the simulation the worker brings the axle to the welding table and sends a signal to PLC to attach the axle to table, PLC sends signals for table to push the plates and close the clamps. Then worker goes to push the button, which sends signal to PLC that button is pressed. PLC then checks robots status and if robot is ready, PLC sends a signal to table to rotate.

After robot knows that table has rotated it starts its cycle. Table won't rotate until both the button is pressed and robot has done its cycle. Finally, when axle has been welded and table has finished the rotation, the worker sends a signal to PLC to detach the axle and then brings it to a conveyor.

Liite 2. Simulaation työnkulku

Video saadaan näkyviin kaksoisnapauttamalla alla olevaa nuppineulan kuvaketta, jolloin PDF, jossa video on, avautuu. Lopuksi, dokumenttiin on luotettava sen asetuksista ja napauttaa PDF:ää videon käynnistämiseksi.