Kerttu-Irmeli Pyrhönen

Robotin ohjausjärjestelmän kehitys

Opinnäytetyö
Sähkö- ja automaatiotekniikka

2018

Kaakkois-Suomen ammattikorkeakoulu

Tekijä/Tekijät	Tutkinto	Aika
Kerttu-Irmeli Pyrhönen	Insinööri (AMK)	Toukokuu 2018

Opinnäytetyön nimi

Robotin ohjausjärjestelmän kehitys

42 sivua
2 liitesivua

Toimeksiantaja

Orfer Oy

Ohjaaja

Teemu Manninen, Jesse Luhti

Tiivistelmä

Työ on tehty Orfer Oy:n tarpeisiin kehittää robotin ja PC:n välistä kommunikointia. Tavoitteena oli tutkia mahdollisuuksia luoda nopeampi ja joustavampi kommunikointitapa joko TCP- tai UDP-protokollaa käyttäen. Kapasiteetti oli tarkoitus mitata robotilla, joka piirtäisi PC:n käskyjen mukaan kuvaa mahdollisimman lyhyt viiva kerrallaan.

Työn alkuun selvitettiin UDP- ja TCP-ominaisuuksien toiminta ja nopeus Kawasaki-robotilla. Kokeiden perusteella voitiin todentaa UDP-protokollan olevan liian epävarma toiminnaltaan suurimpaan osaan sovelluksista, joten päädyttiin tekemään varsinainen kommunikointi TCP-protokollaa käyttäen. Työssä muodostettiin robotille työjono piirrettävistä viivoista siten, että oli mahdollista samanaikaisesti sekä täyttää työjonoa että suorittaa töitä. Työtä varten luotiin oma sanomaliikenne sekä mahdollistettiin sekä useamman sanoman lähetys että vastaanotto yhdessä paketissa.

Piirtävää robottisovellusta ei käytettävällä ajalla saatu aikaiseksi, mutta kommunikoinnin tahtiaika saatiin mitattua siten, että robotilla haettiin työ työjonosta, suoritettiin muutama aliohjelma ja kuitattiin työ tehdyksi. Ilman liikekäskyjä varsinaisen työkierron tahti oli erittäin nopea jättäen vähän aikaa kommunikoinnille vastaanottaa töitä sekä kuitata töitä tehdyiksi.

Mitatut tahtiajat olivat nopeampia, mitä lähtökohtaisesti ajateltiin, ja työjonon toiminta sekä palautuminen erilaisista virhe- ja poikkeustilanteista toimivat hyvin. Työssä luotu kommunikointipohja ei sellaisenaan täysin toimi muissa sovelluksissa, mutta on otettavissa käyttöön melko pienillä muutoksilla välittämättä siitä, onko kyseessä lavaus-, purku- vai poimintasovellus.

Asiasanat

Orfer Oy, tietoliikenne, kommunikointi, TCP, UDP, Kawasaki, robotiikka

South-Eastern Finland University of Applied Sciences

Author (authors)	Degree	Time
Kerttu-Irmeli Pyrhönen	Bachelor of Engineering	May 2018

Thesis Title

Development of robot control system

42 pages
2 pages of appendices

Commissioned by

Orfer Oy

Supervisor

Teemu Manninen, Jesse Luhti

Abstract

This thesis has been done for Orfer Oy's needs to develop communication between the robot and the PC. The aim was to explore the possibilities of creating a faster and more flexible communication method using either TCP or UDP protocoll. The capacity was supposed to be measured by a robot that would draw up a short line at a time according to the instructions of the PC.

First the features and the speed of both UDP and TCP protocolls were investigated with a Kawasaki robot. After these investigations it was possible to verify that the UDP protocol was too uncertain for most of companys products, so it was decided to do the actual communication using the TCP protocol.There was a workqueue created for the lines robot should draw. Robot was able to fill the queue with works simultaneously while finishing works. For this thesis was an own communication created with the possibility to send and receive multiple messages in one package.

The robot was not able to really draw anything because of lack of time, but the cycle time of communication was measured so that the robot retrieved a work from the queue, executed a few subroutines and the work was acknowledged to be done. Without any motion commands the total cycle time was very fast leaving only little time for communication to receive jobs, add them to the queue and to acknowledge the work to be done.

Measured cycle times were faster than initially thought, and the working queue was working well and it was able to recover from different error and exceptional situations independently. The communication platform created in this thesis does not work fully in other applications, but can be implemented with small variations, regardless of whether it is used in a palletizing, depalletizing or pickup application.

Keywords

Orfer Oy, communications, TCP, UDP, Kawasaki, robostics

SISÄLLYS

Käsitteiden määrittely

ASCII	= Merkistö jonka jokaiselle merkille on annettu numeerinen arvo välillä 0-255 (American Standard Code for Information Interchange)
Merkkijono	= Sovitun merkistön mukaisia merkkejä sisältä jono
Stringi	= Synonyymi merkkijonolle
Sanoma	= Määritelty merkkijono jolla on alku- ja loppumerkki
Sanomapaketti	= Yhden tai useamman sanoman sisältävä paketti mikä voidaan lähettää tai vastaanottaa
Lähetys	= Sanomapaketin lähetys
Vastaanotto	= Sanomapaketin vastaanotto
Lavauskuvio	= Määrittelee miten tuotteet sijoitetaan lavalle
Purkukuvio	= Määrittelee miten tuotteet puretaan lavalta

1 JOHDANTO

Työ on suunniteltu ja suoritettu työnantajani tiloissa ja resursseilla Orfer Oy:llä. Orfer on vuonna 1970 Orimattilassa perustettu perheyritys, joka suunnittelee, valmistaa ja kehittää robottiteknologiaa hyödyntäviä automaatiojärjestelmiä. Yritys on toiminut vuodesta 1995 Kawasaki ja vuodesta 2001 Toshiba-robottien maahantuojana. Yritys valmistaa asiakaskohtaisesti räätälöityjä automaatiojärjestelmiä avaimet käteen -periaatteella. /8./

Työn päätehtävänä oli tutkia Kawasaki-robotin ja PC:n välisen kommunikoinnin suorituskykyä nykyisellä Orferin tavalla ja parantaa sitä tai luoda kokonaan uusi tapa kommunikoinnille. Nykyisissä järjestelmissä PC:ltä lähetetään melko suuria määriä dataa, ennen kuin robotti lähtee suorittamaan mitään PC:n lähettämistä ”töistä”. Työssä onkin tarkoitus testata, mitkä ovat mahdollisuudet luoda robotille niin sanottu työjono, josta robotti poimii heti työn suoritettavaksi, kun sellainen on saatu purettua PC:n lähettämästä datasta. Työssä luodaan ja testataan työjonon toiminta Kawasaki-robotissa sekä mitataan eri tiedonsiirron vaiheisiin kuluvat ajat. Tämän jälkeen pitäisi olla tiedossa minimitahtiajat ja vaatimukset kommunikoinnille, PC:n ja robotin puolelle sekä toimintavarmuus. Saadut tietoja voitaisiin käyttää uusia järjestelmiä luodessa.

Työn alkuun perehdytään tietojärjestelmien perusteisiin ja niiden kautta UDP- ja TCP-protokolliin, jotka ovat työn, robotti–PC-kommunikoinnin, peruspilarit ja joiden vaikutuksia/eroavaisuuksia suorituskykyyn tutkitaan. Näistä protokollista päädytään toisella luomaan kommunikointi, josta saadaan suorituskyky mitattua. Suorituskyky mitataan robotilla, joka piirtää kuvan PC:n lähettämien käskyjen mukaisesti. Oma osuuteni ohjelmoinnissa on robotin ohjelmointi sisältäen sekä kommunikoinnin suunnittelun ja luomisen että varsinaisen toiminnan ohjelmoinnin. Työnohjaaja työnantajan puolelta luo PC-ohjelman.

Omaan työnkuvaani oli ennen opinnäytetyön tekemistä kuulunut reilun vuoden verran Kawasaki-robottien ohjelmointi. Ohjelmoinnin perusteita ei siis ollut tarve opiskella, vaan ennemminkin tutustua syvemmin sekä Kawasaki-robottien kommunikointiominaisuuksiin että yleisesti tietojärjestelmien toimintaperiaatteisiin. Työssä on paljon tietoja, joita olen saanut kollegoilta urani aikana, kun olen törmännyt erinäisiin ongelmiin tai ihmetellyt, miksi jokin asia on toteutettu

jollakin tietyllä tavalla. Työstä löytyy siten monta kohtaa, johon voisi merkitä lähteeksi keskustelut yrityksen robotiikan suunnittelupäällikön Jukka Rytinimen tai työni ohjaajan teknologiajohtajan Jesse Luhdin kanssa. Molemmilta henkilöiltä olen saanut paljon tukea robottien maailmaan tutustuessani, varsinkin heidän luottamuksensa omiin kykyihini oppia uutta on ollut korvaamaton. Mitä haasteellisempi työ on annettu niin yleensä sitä enemmän se on antanut ammatillisesti, täten on ollut erittäin tärkeää, että on uskallettu antaa haasteita joihin en sillä hetkellä välttämättä olisi itse uskonut pystyväni. Yrityksessä on muutenkin ollut erittäin kannustava ja ammatillista kehitystä tukeva ilmapiiri.

2 HISTORIAA

Vuonna 1995 kun Orferin historian ensimmäinen Kawasaki-robotti myytiin, oli robotissa käytössä AD-ohjain, jossa kommunikointi tapahtui kokonaan sarjaliikenteen kautta. Ensimmäisiä ohjaimia, joissa oli mahdollisuus ethernet-liitäntään, oli C- ja sitä seurannut D-ohjain, mutta se oli erikseen myytävä lisäominaisuus molemmissa malleissa. Tämän lisäksi, että ethernet-yhteys oli lisäominaisuus, olivat TCP-ominaisuudet vielä osittain vajavaiset, eikä siten ollut esimerkiksi mahdollista luoda koko ajan auki olevaa yhteyttä. /5./

TCP- kommunikointia käytettiinkin alkuun siten, että yhteys avattiin, kun robotilla oli jotain lähetettävää, ja suljettiin, kun oli saatu vastaus lähetettyyn viestiin. Yhteys toimi myös siten, että yhteys oli koko ajan auki toiseen osapuoleen, mutta mikäli yhteys katkesi esimerkiksi johdon irrotessa, siitä seurasi vikatilanne, josta robotin ei ollut muutoin mahdollista toipua kuin käynnistämällä se uudestaan. /5./

Vuonna 2011 ensimmäisten E-ohjaimien myötä ethernet-liitäntä kuului jo perusominaisuuksiin, minkä lisäksi TCP-ominaisuuksiin oli lisätty statuskysely. jolla voitiin varmistaa, onko yhteys toiseen osapuoleen kunnossa ennen lähetystä sekä vastaanottoa. Näiden ominaisuuksien myötä luotiin ensimmäinen koko ajan auki oleva sekä itseään ylläpitävä TCP-yhteys. Yhteys tuli käyttöön sovelluksessa, jossa robotti poimi tuotteita useamman kameran avulla. Robotti saattoi vastaanottaa kerralla jopa 30 tuotteen koordinaatit, joten kyseiseen sovellukseen luotiin myös ominaisuus pitkien sanomien vastaanottoon ja pur-

kuun. Kyseinen pohja yhteyden muodostamiselle, ylläpitämiselle, katkaisemisella sekä sanomien purkuun on nykyisenkin järjestelmän pohjana. /5./

3 VAATIMUKSET NYT JA TULEVAISUUDESSA

Orferilla robotin ja PC:n väliseen kommunikointiin on vuosia sitten panostettu ja luotu yksinkertainen ja toimiva kommunikointimalli. Kommunikointi on suoritettu TCP-protokollan mukaan. Robotille on tehty valmiit ohjelmapohjat, joihin käyttöönottajan tarvitsee vain päivittää PC:n IP-osoite. Kommunikoinnissa on hoidettu laajasti virheentunnistukset sekä virhetoipumiset. Virheitä, joita on nykypäivänä vielä havaittu, ovat toisinaan puskuriin jääneet vanhentuneet datat esimerkiksi konenäkösovelluksissa.

Vuosien myötä suorituskyky on myös tullut vastaan sovelluksissa, joissa robotti varsinaisen liikeohjelman taustalla keskustelee useamman laitteen kanssa sekä suorittaa samaan aikaan datan käsittelyä ja laskentaa. Tänä päivänä yrityksen luomissa sovelluksissa robotti suorittaa suuren osan sovelluksen älystä. Robotille lähetetään hyvin karkeaa dataa, minkä pohjalta robotti laskee ja päättelee suoritettavat tehtävät. Nämä toiminnot kuormittavat robotin suorituskykyä ja näkyvät suoraan myös liikeohjelmassa, kun taustaohjelmat käsittelevät suurenevan määrän dataa eivätkä enää ehdi liikkeen mukaan. Tämän pohjalta onkin pohdittu mahdollisuutta robotin tarkempaan ohjaukseen ulkoiselta PC:ltä, jonka tehtävänä olisi vain suorittaa datan käsittelyt ja laskennat ja lähettää robotille valmiimpaa dataa, valmiita töitä ja mahdollista olisi työjonon muodostaminen robotille. Tällöin robotin tehtävä olisi vain hakea tarkasti määritelty työ työjonosta, suorittaa se, kuitata PC:lle ja noutaa seuraava työ. Tämän myötä robotin suorituskykyä voitaisiin parantaa.

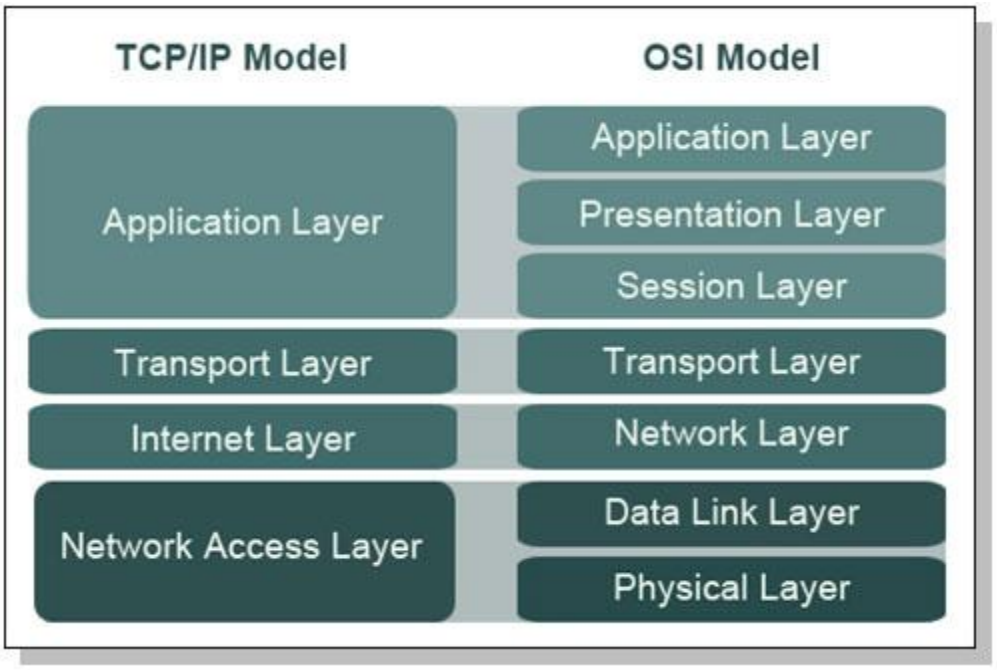
Tulevaisuudessa mitä enemmän tulee mahdollisuuksia käyttää konenäkösovelluksia ja muita PC -pohjaisia ohjelmia ohjaamaan robotin toimintaa, varsinkin nopeutta vaativat sovellukset, sitä tärkeämpää on, että yrityksellä on käytössä tai ainakin selvitettynä PC:n ja robotin välisen kommunikoinnin nopeus. Nopeus täytyy olla selvitettynä, jotta tahtiaikojen määrittely on luotettavaa tai edes mahdollista myyntivaiheessa.

4 JOHDANTO TIETOJÄRJESTELMIIN

Tässä luvussa käydään läpi tietojärjestelmän rakenne sekä eri protokollat kuljetuskerroksen toteuttamiseen. Luvun avulla selviää, mikä on kuljetuskerros ja mitä eri vaihtoehtoja siihen on valittavana työn kannalta.

4.1 OSI

Tietojärjestelmien rakenne voidaan useimmiten kuvata joko OSI- (Open Systems Interconnection Reference Model) tai TCP/IP- (Transmission Control Protocol / Internet Protocol) mallin mukaan. OSI-malli on ISO:n (International Standardization Organization) määrittelemä malli tietojärjestelmien rakentamiseen. Mallissa tietojärjestelmät jaetaan seitsemään eri kerrokseen. TCP/IP on yksinkertaistettu malli OSI-mallista sisältäen neljä tai viisi kerrosta. TCP/IP on myös käytetympi malli näistä kahdesta pitkälti sen johdosta, että useat TCP/IP:n standardit oli vahvistettu aiemmin kuin OS-mallia edes määriteltiin /1, s.181/. Molemmat mallit helpottavat ymmärtämään monimutkaisia verkkoja, kun eri osat voidaan jakaa omiin kerroksiinsa. Kerroksien hahmottamisella esitetään tietojärjestelmän eri osien tapaa palvella toisiaan. Periaate on, että alempi taso palvelee aina ylempää tasoa. /1, s.184./



Kuva 1. OSI-malli verrattuna TCP/IP-malliin

Kuvassa 1 on OSI-malli verrattuna TCP/IP:n nelikerroksiseen malliin. OSI:n seitsemän eri kerrosta alimmasta ylimpään ovat fyysinen- (physical), siirtoyh-

teys- (data link), verkko- (network), kuljetus- (transport), istunto- (session), esitystapa- (presentation) ja sovelluskerros (application layer. Nämä samat kerrokset on TCP/IP-mallissa useimmiten sisällytetty neljään kerrokseen. Ensimmäinen kerros, peruskerros, sisältää fyysisen- sekä siirtoyhteyskerroksen. Kerros määrittää fyysiset ominaisuudet kaikille käytössä oleville fyysisille laitteille ja huolehtii siirtovarmuudesta sekä -nopeudesta. /1, s.139./

Toinen kerros on yhteinen OSI-mallin kanssa eli verkkokerros. Verkkokerros määrittelee siirrettävän datan suuruuden sekä huolehtii datan reitittämisestä osoitteiden mukaisille laitteille. Kolmas kerros on myös vastaava OSI-mallin mukaan eli kuljetuskerros. Kuljetuskerroksen toiminnan määrää käytettävä kuljetusprotokolla. TCP/IP-mallissa on käytettävissä kaksi eri kuljetusprotokollaa, TCP (Transmission Control Protocol) ja UDP (User Datagram Protocol). Kuljetuskerros huolehtii yhteyksien muodostamisesta sekä siirrettävän datan pilkkomisesta protokollan mukaisiin osiin, joita kutsutaan yleisesti paketeiksi (packet) tai segmenteiksi (segment). Datan perille pääsyn varmistus, datavirran hallinta sekä kuittausmenetelmät ovat myös kerroksen tehtäviä yhteydellisissä protokollia käytettäessä. Yhteydettömiin ja yhteydellisiin protokolliin perehdytään myöhemmin, kun käydään läpi eri kuljetuskerrosprotokollat. /1, s.139-140./

Neljäs kerros on sovelluskerros ja sisältää OSI-mallin istunto-, esitystapa- ja sovelluskerroksen. Sovelluskerros määrittelee, kuinka lähetettävästä datasta muodostetaan sanomia ja pakkaustavan sekä saapuneiden pakettien purkutavan. /1, s.139-141./

Tässä työssä perehdytään syvemmin TCP/IP-mallin kuljetuskerrokseen (TCP ja UDP) sekä sovelluskerrokseen Kawasaki-robotin osalta.

4.2 Kuljetuskerros

Tässä luvussa käydään läpi TCP/IP:n molemmat kuljetuskerroksen protokollat. Protokolla yleiskielessä tarkoittaa sovittua käytäntöä sääntöineen, kuinka käyttäydytään ja toimitaan eri tilanteissa. Arkielämän protokollia ovat esimerkiksi liikenne, pallopelit, kaupassakäynti sekä erilaiset juhlat. Tietojärjestelmissä protokolla tarkoittaa samalla tavalla sovittuja käytäntöjä verkkolaitteiden

välillä. Eri protokollille on määritelty tavat, kuinka yhteys muodostetaan ja katkaistaan. Kuinka tietoa lähetetään ja vastaanotetaan, miten tieto on pakattu sekä kuitataanko pakettien lähetyksiä ja vastaanottoja? Verkkolaitteen toimiessa vastoin protokollaa, kommunikointi toisten laitteiden välillä on mahdotonta. /2./

4.2.1 TCP

Kuljetuskerroksena TCP on yhteydellinen eli vaatii lähettääkseen ja vastaanottaakseen paketteja avoinna olevan yhteyden vastaanottajaan. TCP-protokolla on hyvin luotettava tiedonsiirrossa, koska pakettien perille meno ja vuon ohjaus sekä pakettien oikea saapumisjärjestys on varmistettu protokollan taholta. Saapumisjärjestyksen seurantaa ja pakettien oikeaa numerointia varten lähettäjän ja vastaanottajan täytyy olla tahdissa. Tämä tahdistus tehdään kolmivaiheisella yhteyden avaamisella. /3, s.94./

Tapauksessa, jossa lähetetystä paketista lähettäjä ei saa kuittausta määrätyssä ajassa, paketti lähetetään automaattisesti uudelleen. Tämä voi jossain tapauksissa aiheuttaa sen, että samalla pakettinumerolla saapuu useampi paketti. TCP- vastaanotto tunnistaa samalla pakettinumerolla saapuvat ja tuhoaa niistä automaattisesti ylimääräiset paketit. /1, s.303./

Vuon ohjauksella tarkoitetaan lähetettävän datamäärän ja nopeuden ohjaamista ja muuttamista vastaanottajan vastaanottokyvyn mukaan. TCP-paketeissa menee lähettäjälle tieto siitä, miten paljon dataa voidaan vielä vastaanottaa, jolloin lähettäjä voi hidastaa lähetysnopeutta tai pienentää lähetettävää datamäärää. /3, s.105./

TCP-protokollan mukaista kommunikointia käytetään yleisesti silloin, kun on tarve olla varma, että vastaanottaja vastaanottaa kaiken datan oikeassa järjestyksessä. Varmistuksien ja monivaiheisten kuittauksien myötä protokolla on hyvin raskas ja siten soveltuvuus nopeutta vaativiin järjestelmiin on huono. /3, s.93./

4.2.2 UDP

UDP-protokolla on yhteydetön. Yhteydettömällä protokollalla tarkoitetaan sitä, että yhteys avataan vasta lähetystilanteessa. Ennen lähetystä ei tarkastella yhteyden olemassa oloa tai sitä, kuunteleeko vastaanottaja. UDP-protokollassa verkkolaite ei varoita vastaanottajaa lähetystilanteessa eikä lähetä kuittauksia vastaanotetuista paketeista. /3, s.87./

Puuttuvat kuittaukset ja varmennukset tekevät UDP-protokollasta hyvin kevyen ja nopean. Näiden ominaisuuksien ansiosta protokollaa voidaan käyttää sovelluksissa, jotka vaativat nopeutta ja joiden toiminnan kannalta jokin puuttuva data tai väärä järjestys ei ole haitaksi. Mikäli UDP-protokollaa käytetään ja halutaan jotain samoja ominaisuuksia kuin TCP-protokollassa esimerkiksi vuon hallinta, tulee ne tehdä sovellustasolle. /3, s.93./

4.3 Sovelluskerros

Sovelluskerros on rajapinta, jonka kautta varsinaisesta sovelluksesta välitettävä tai sinne välitettävä data kirjoitetaan sanomiin tai puretaan sanomista. Kerros määrittelee, koska ja miten dataa lähetetään ja vastaanotetaan. Riippuen käytettävästä kuljetuskerroksen protokollasta kerroksen toimintoja ja tehtäviä voidaan muokata. Esimerkiksi UDP-protokollaa käytettäessä sovelluskerrokseen voidaan luoda vuon ohjaus tai pakettien kuittausominaisuus.

5 KAWASAKIOHJELMOINTI

Ohjelmointityökaluna oli käytössä tekstinkäsittelyohjelma Notepad++ ja ohjelmien lataukseen ja tulostuksien seuraamiseen oli käytössä Kawasakin terminaaliohjelma KRterm. Kawasaki-robottiin voitaisiin ohjelmia ladata myös usb-muistista ja ohjelmia voitaisiin muokata robotin käsiohjaimesta.

Kawasaki-robottien käyttämä koodikieli on Kawasakin oma AS-kieli. Kieli on melko yksinkertaista ja siten helposti ymmärrettävissä suppeammallakin tietämyksellä ja kokemuksella. Jotta ohjelmaa voisi kirjoittaa, täytyy tietyt perusasiat opetella sekä tutustua muun muussa ohjelmien säikeistykseen.

5.1 Muuttujien määrittely ja arvo

Ainoa rajoite muuttujien määrittelyssä on se, että muuttuja täytyy määritellä joko lukuarvoksi, merkkijonoksi tai pisteeksi. Lukuja ei määritellessä tarvitse kuitenkaan luokitella totuusarvomuuttujiksi, reaali- tai kokonaisluvuiksi vaan kaikki luvut ovat lähtökohtaisesti reaalilukuja ja tarvittaessa niitä voidaan käyttää halutulla tavalla. Oli muuttuja kokonaisluku, totuusarvo tai ASCII-merkki, voidaan sen arvo esittää reaalilukuna.

```
a[1] = 0
a[2] = 15
a[3] = 14.5
a[4] = INT(a[3])
a[5] = -1
a[6] = TRUE

FOR .i = 1 TO 6
    IF a[.i] == TRUE THEN
        PRINT "Arvo on totta!! "
    END
END
```

Kuva 2. Lukuarvomuuttujien esittely

Kuvassa 2 on esitelty kuusipaikkainen taulukkomuuttuja ”a”. Jokaiselle taulukon paikalle on annettu alkuarvo. mikä voi olla kokonaisluku, reaaliluku tai totuusarvo totta/tarua. Taulukon 3 ensimmäistä muuttujaa on reaalilukuja ja paikan 4 muuttujaan on kirjoitettu paikan 3 arvo kokonaislukuna INT käskyllä. Muuttujan arvon ollessa -1 saa muuttuja totuusarvona arvon totta. Siten kuvan mukaisessa FOR-lausekkeesta tulostuu teksti ”Arvo on totta!! ” ainoastaan taulukon paikkojen 5 ja 6 kohdalla.

Muuttujien nimien pituus on rajoitettu 15 merkkiin. Rajoitus koskee myös ohjelmien nimiä. Muuttujan nimen pituuden ylittäessä 15 merkkiä katkeaa se vasemmalta oikealle 15 merkin jälkeen.

Muuttujien ensimmäisen merkin täytyy olla kirjain. Muuttujasta riippuen alkumerkin eteen täytyy lisätä tunnus. Merkkijonon tunnus on dollari ”$” ja akselikulmin määritelty pisteen tunnus on risuaita ”#”. /6./

5.2 Muuttujien luokittelu

Muuttujat luokitellaan joko globaaleihin tai paikallisiin muuttujiin. Kawasaki-robotilla paikallismuuttujan tunnistaa siitä, että muuttujan ensimmäinen merkki on piste ”.”.

Globaalit muuttujat tallennetaan robotin muistiin, kun ne määritellään, ja ne ovat käytössä myös muissa ohjelmissa. Paikalliset muuttujat kirjoitetaan robotin muistiin vasta, kun ne ensi kerran määritellään ohjelmassa ja muuttujat ovat käytössä vain kyseisen ohjelman sisällä. Paikallisia muuttujia voidaan lisäksi käyttää ohjelmin kutsujen sisällä, jolloin aliohjelma palauttaa paikallismuuttujan arvon kutsuneelle ohjelmalle. Paikallisista muuttujista on varmuus, että mikään muu ohjelma ei kirjoita samanaikaisesti samaan muuttujaan. /6./

5.3 Kiertoaika sekä taustaohjelmien toiminta

Kommunikointi suoritetaan liikeohjelman taustalla robotin yhdessä taustaohjelmassa. Kyseinen taustaohjelma on varattuna sekä robotin kommunikoinnille että sen sovelluskerrokselle. Liikeohjelmassa kommunikointia ei suoriteta, koska tällöin liikeohjelma pysähtyisi aina tiedonsiirron ajaksi.

Robotin CPU:n kiertoaika on 2 ms, josta liikeohjelma vie tarvitsemansa ajan ja tuosta ylijäävä aika jää robotin taustaohjelmille käytettäväksi. Nykyisellä - ohjaimella Kawasaki-robotissa on käytettävissä 5 taustaohjelmaa. Taustaohjelmille ei voida määrittää mitään minimiaikaa tai prosenttiosuutta kiertoajasta, vaan aika on täysin riippuvainen liikeohjelmalle varatusta ajasta. Käytettävissä olevat taustaohjelmat ovat tasavertaisia eikä niille jaettavaa aikaa voida priorisoida, näin ollen kommunikoinnille varatun taustaohjelman nopeus on paljon siitä kiinni, kuinka liikeohjelma ja toiset taustaohjelmat ovat luotuja. Liikeohjelma, jossa on paljon esimerkiksi toistorakenteita (jäädään odottamaan jonkun ehdon toteutumista) ilman odotuskäskyä, käyttää suuren osan, ellei koko kiertoaikaa, jolloin taustaohjelmille jäävä aika vähenee hidastaen niiden toimintaa. /6./

5.4 Ohjelmarakenne

Taustaohjelmien ja liikeohjelmien toiminta ja rakenne ovat vapaita ohjelmoitaviksi. Työssä olenkin käyttänyt pohjana yrityksen hyviksi kokemia rakenteita, periaatteita sekä säilyttänyt mahdollisuuden yleisimmille ominaisuuksille.

Robotin taustaohjelmien ollessa tasavertaisia ei olisi väliä, minkä taustaohjelmista valitsee kommunikoinnille. Työssä olenkin valinnut ensimmäisen taustaohjelman perinnesyistä, koska niin on ollut yrityksellä aina tapana. Ensimmäisen taustaohjelman lisäksi kolmas ja neljäs taustaohjelma ovat yleensä käytössä yrityksen toimittamissa sovelluksissa. Robotin hälytykset ja tilatiedot päivittyvät taustaohjelma 4:n kautta ja 3:n kautta päivittyy turva-asioita, kuten robotin pysäytys törmäyssuojasta ja automaatilla ajon esto, jos robotti on ajettu käsin muualle kuin kotiasemaan. Toinen ja viides taustaohjelma ovat yleensä käyttämättöminä. Nämä ”ylimääräiset” taustaohjelmat ovat käytössä, kun on tarve suorittaa esimerkiksi laskentaa tai kameran triggauksia robotin liikkuessa.

Kun robotti käynnistetään kotiasemasta, käynnistetään myös käytössä olevat taustaohjelmat. Tämän lisäksi liikeohjelmassa on kohtia, joissa odotetaan tietoa taustaohjelmilta. Näissä kohdissa useimmiten varmistetaan, että taustaohjelma on käynnissä ja käynnistetään taustaohjelma tarvittaessa uudelleen.

Liikeohjelma tai toinen taustaohjelma keskustelee taustaohjelmien kanssa sisäisillä signaaleilla sekä tavallisilla muuttujilla. Sisäisille signaaleille on sovittu jokin funktio, mitä sanomia esimerkiksi lähetetään. Kun on käytössä sovellus, jossa robotti tarvitsee toimintaansa tietoja PC:ltä, tapahtuu liikeohjelman muuttujiin kirjoitus samassa taustaohjelmassa, missä sanomat vastaanotetaan ja puretaan. Liikeohjelmassa tarkistetaan kyseisten muuttujien tiedot ja tallennetaan toisiin muuttujiin, joita käytetään pelkästään liikeohjelmassa. Tällä muuttujien arvon tallentamisella toisiin arvoihin voidaan varmistua siitä, että liikeohjelman aikana taustaohjelma ei ylikirjoita alkuperäisiä arvoja. Tämän lisäksi taustaohjelman muuttujat voidaan vapauttaa aiemmin käytettäväksi taustaohjelmalle, kun muuttujien arvo on jo tallennettu toisiin muuttujiin.

6 VAATIMUKSET KOMMUNIKOINNILLE

6.1 Käytettävyys

Kommunikoinnin tulisi olla toiminnaltaan sellainen, että henkilön, joka ottaa robotin käyttöön, ei tarvitsisi syvällisesti tutustua ja muuttaa ohjelmia, vaan ohjelmien käyttöönotto olisi vaivatonta. Orferilla on pitkään ollut käytössä TCP-kommunikointi robotin ja PC:n/käyttöliittymän välillä. Kyseinen kommunikointi on vuosien ajan kehittynyt ja muodostunut hyvin helppokäyttöiseksi. Henkilön, joka ottaa robotin käyttöön, ei varsinaisesti tarvitse tietää kommunikoinnista muuta kuin käyttöliittymän IP-osoite ja osata ladata oikeat ohjelmat robottiin. Tämän jälkeen robotin taustaohjelman käynnistyttyä robotti muodostaa automaattisesti yhteyden käyttöliittymään, mikäli se on mahdollista. Yhteyden muodostumisen lisäksi sanomaliikenne on siten rakennettu, että toimii sellaisenaan lähes kaikissa lavausroboteissa välittämättä, kuinka monesta paikasta robotti poimii tuotteita, moneenko paikkaan jättää tuotteita tai mikä on käsiteltävä tuote.

Mikäli työn pohjalta saadaan aikaiseksi toimiva kommunikointimalli, jää se helposti käyttämättä, mikäli helppokäyttöisyys jää toteutumatta. Helppokäyttöisyys oli yksi tavoitteista työssä, mutta siitä voidaan joutua joustamaa, sillä päätehtävänä on kuitenkin selvittää kommunikoinnin suorituskyvyn mahdollisuudet.

6.2 Joustavuus

Joustavuudella tässä yhteydessä tarkoitetaan sitä, että kommunikointimallin täytyisi sopia mahdollisimman erilaisiin sovelluksiin mahdollisimman pienillä muutoksilla. Näin ollen on otettava huomioon myös työn sovelluksesta poikkeavat sovellukset ja luoda näiden pohjalta työn sovelluksen kannalta ehkä turhia, mutta välttämättömiä ominaisuuksia muiden sovelluksien kannalta. Sovelluksia, joihin mallin olisi käytävä, ovat muun muussa lavaus-, purku- ja poimintasovellukset.

6.3 Virhetoipuminen

Virhetoipuminen tarkoittaa, kuinka ilmenneistä virhetilanteista palaudutaan takaisin normaalitilaan. Mahdollisimman pitkälle on pyrittävä siihen, että robotin ohjelmaan on lisätty yleisimpien virhetilanteiden tunnistus ja niistä itsenäinen toipuminen. Lisäksi on tunnistettava virhetilanteet, joista robotti ei itsenäisesti pysty toipumaan, näissä tilanteissa on luotava robotilta hälytyksiä, jotta operaattori osaisi korjata tilanteen. Virhetoipumisella estetään tilanteiden syntymiset, joissa robotti ei lähde suorittamaan seuraavaa työtä tai pysähtyy kesken työkierron ilman ilmoitusta siitä, mikä on vialla.

Opinnäytetyön sovelluksessa odotettavia virhetilanteita on muun muussa saapuneiden töiden väärä järjestys, työn puutteelliset tai virheelliset parametrit sekä työjonon ruuhkatilanteet. Kommunikointi ja työjonon hallinta suoritetaan robotin taustaohjelmassa, jossa suoritetaan hyvin paljon muitakin prosesseja. Siten ei ole sopivaa pysäyttää taustaohjelmaa virhetilanteissa, vaan ensin pyrittävä esimerkiksi uudelleen lähetys, lähetyksen tauotus pyynnöllä tai työjonon tyhjentämisellä selviytymään virhetilanteesta. Mikäli virhetilanteesta ei selviydytä taustaohjelmassa, on annettava siitä tieto robotin liikeohjelmaan, jossa toiminnan pysäyttäminen sekä hälytyksen antaminen on turvallisempaa.

7 VUON HALLINTA

Vuon hallinnalla tässä työssä tarkoitetaan sanomaliikenteen tahdistamista sanomien purun sekä töiden suorittamisen tahtiin. Hallittavia tilanteita ovat lähetyksen pysäyttäminen, uudelleen lähetyksen aloitus sekä työjonossa peruuttaminen.

TCP-protokollassa vuon hallinta on helpompaa, koska protokolla itsessään tunnistaa vastaanottajan vastaanottopuskurin täyttymisen ja osaa pysäyttää lähetyksen ajoissa. Sitä vastoin UDP-protokollaa käytettäessä vuon ohjausta ei ole olemassa protokollan osalta. Käytettäessä UDP-protokollaa vastaanoton täyttyessä tai ollessa pois päältä voi useita jopa satoja paketteja mennä ohi huomaamatta. Lisäksi Kawasakissa UDP-vastaanottopuskuri on pienempi kuin TCP: n vastaava. TCP-vastaanottopuskuri on robotissa aina auki, kun yhteys on kunnossa, siten vaikka ohjelmassa TCP-vastaanotto käsky ei olisi päällä, paketit saapuvat puskuriin. UDP-vastaanotossa puskuri on avoinna

vain silloin, kun UDP-vastaanotto on auki. Tästä seurauksena lähetystä täytyy tauottaa robotin sovellustasolla.

7.1 Virhetilanteet

TCP-kommunikoinnilla pakettien perillemeno ja oikea järjestys on hyvin pitkälle hoidettu protokollan puolesta. UDP-kommunikointia käytettäessä täytyy molemmat toiminnot tarvittaessa suorittaa ohjelmatasolla.

Sekä vuonohjaus että töiden lisäys työjonoon voidaan suorittaa samalla tavalla molemmilla kommunikointitavoilla. TCP-kommunikoinnilla työnumeroinnin tarkastuksen voisi ajatella turhaksi, kun tiedetään, että paketit eivät pääse muuttamaan saapumisjärjestystään. Tarkistuksella voidaankin havaita mahdolliset virheet, jotka on muodostunut PC:llä jo lähetysvaiheessa. Sen lisäksi, että ominaisuudella voidaan tunnistaa PC:llä syntyneet virheet, voidaan ominaisuutta pitää myös tietoturvallisena seikkana. Mikäli vieras verkossa nauhoittaisi liikennettä ja myöhemmin yrittäisi toistaa sitä robotin suuntaan, tulisi siitä virhe vanhojen tilausnumeroiden sekä työnumeroiden takia. Nykyinen järjestelmä on tästä poiketen puutteellinen, koska sanomilla ei ole mitään tunnuksia, joista voitaisiin päätellä oikea-aikaisuus.

UDP-kommunikoinnilla työnumeroiden tarkastus on pakollista, koska varmuutta siitä, onko kaikki PC:n lähettämät paketit vastaanotettu, ei ole. Vastaavasti työnumerotarkastuksella huomataan myös lähetysvaiheessa syntyneet virheet. PC:n vastausnopeutta on myös rajoitettava UDP:llä, sillä on mahdollisuus, että robotin lähetyksen jälkeen robotti ei ole ehtinyt vastaanottovaiheeseen, kun PC on jo vastauksensa lähettänyt. Minimiviive vastaanottoon on 2 ms, mikä on Kawasaki-robotin kiertoaika.

Mahdollisia virhetilanteita ovat myös kesken katkenneet sanomat. Näitä tilanteita varten verrataan aina saapuneen sanoman todellinen pituus ja annettu pituus sekä se, että sanomasta löytyy määritellyt alku- ja loppumerkit.

7.2 Pakettien vastaanotto

Ihanteellinen tilanne olisi, kun saapuneet paketit saataisiin purettua töiksi työjonoon nopeammin kuin mitä paketteja vastaanotetaan. Tällöin voitaisiin pa-

kettien vastaanotto ja purku suorittaa asynkronisesti eli eriaikaisesti. UDP-protokollan nopeus näkyisi merkittävimmin TCP-protokollaa vasten työjonoa täytettäessä tällä tavalla, PC ei odottaisi lupaa tai kuittausta lähettää lisää dataa, vaan lähettäisi dataa siihen asti, kunnes työjono olisi täynnä. Työjonon ollessa täynnä tauotettaisiin lähetystä, kunnes sovittu määrä töitä olisi purettu jonosta.

Lähetyksen tauottamisen jälkeen voitaisiin joko jatkaa purkua siitä, mihin jäätiinkin, tai pyytää uudelleen lähetystä seuraavasta työstä alkaen. Jälkimmäisessä tavassa mahdollisuus, että jokin paketti on pysäytyksen aikana vuotanut ohi vastaanoton, pienenisi. Eli jos paketissa saapuu 5 sanomaa, joista 3. puretun työn jälkeen työjono on täynnä, tuhotaan loppuosa paketin sanomista ja pyydetään työjonon tyhjentymisen jälkeen uudelleen lähetystä viimeksi työjonoon lisätystä työnumerosta lähtien.

Sanomien purun ollessa hitaampaa kuin sanomien vastaanotto on vaihtoehtoja, kuinka toimia kaksi kappaletta. Joko muodostetaan työjonon lisäksi toinen puskuri purkamattomille saapuneille paketeille tai sovitaan, ettei PC lähetä lisää paketteja, ennen kuin on saanut kuittauksen edellisistä paketeista. Jälkimmäinen hidastuttaa kommunikointia varsinkin UDP:llä, kun siihen lisätään ominaisuus sovellustasolle, minkä puuttuminen on suurimmaksi osaksi kommunikoinnin nopeuden perusta. Toisaalta vastaava kuittaus tulee myös TCP-kommunikoinnin sovellustasolle, joten kuittaus tulee hidastuttamaan molemmilla protokollilla.

Vaihtoehto muodostaa toinen puskuri saapuneille sanomille antaisi enemmän aikaa purulle ja tekisi pakettien ohivuodosta epätodennäköistä. Tässä kohtaa täytyisi pohtia, mikä on hyöty ajallisesti, kun järjestelmän nopeus on juuri niin nopea kuin järjestelmän hitain kohta. Lisäksi kahden puskurin käyttö lisäisi virheiden mahdollisuuksia ja lisäisi monimutkaisuutta. Puskurit, joiden dataa sekä luetaan että kirjoitetaan, vaativat perusteellisen ohjelmoinnin, jotta ylikirjoittamiselta vältyttäisiin ja puskurin virhetoipuminen olisi nopeaa ja yksinkertaista. Kirjoittaminen ja lukeminen täytyy tahdistaa siten, että nämä eivät tapahtuisi samaan aikaan. Samanaikainen datan lukeminen ja kirjoittaminen luo riskin datan korruptoitumiselle. Näiden syiden takia kahden puskurin luominen nopeutta ja tarkkuutta vaativaan järjestelmään ei ole toivottua.

7.3 Sanomien purku

Työssä on käytössä Orferin määrittelemä sanomarakenne ja siten on mahdollista käyttää myös valmista sanomien purkuohjelmaa. Vastaanotossa ja purussa on useamman sanoman sisältävien pakettien käsittely, mutta sitä ei ole paljoa käytetty. Eli aiemmin paketti on yleensä sisältänyt vain yhden sanoman, kun nyt työssä pyritään sisällyttämään pakettiin niin monta sanomaa kuin mahdollista.

Sanoma voi sisältää yhden työn, joten mitä useampi sanoma on yhdessä paketissa, sitä lyhyempi on aika, mikä menee PC:ltä lähetykseen robotilta vastaanottoon, purkuun ja työjonoon lisäämiseen. Eli niin sanottu kommunikoinnin tahtiaika työtä kohden nopeutuu.

Purku erottelee työt toisistaan, tunnistaa työn luonteen ja sen perusteella purkaa varsinaisen dataosuuden muuttujiin, joita robotti käyttää myöhemmin valitessaan kyseisen työn työjonosta. Työnluonteita on sekä sovelluskohtaisia että yleisiä kaikissa sovelluksissa käytössä olevia. Purussa tarkistetaan myös työnumerointi sekä dataosuuden oikeanmukaisuus esimerkiksi koordinaattien arvon on oltava minimi- ja maksimirajojen sisäpuolella.

7.4 Sanomapakettien lähetys

Kawasaki robotilla stringin maksimipituus on 255 tavua. Sama rajoitus on siten myös pakettien lähetyksessä ja vastaanotossa. Rajoituksen myötä lähetettävää stringiä/stringejä luodessa on otettava huomioon, että 255 tavun pituus ei ylity. /7./

Aiemmin on lähetetty robotilta PC:lle vain yksi sanoma per lähetys. Työn sovelluksessa lähetetään niin monta sanomaa kuin yhteen pakettiin mahtuu, eli ohjelmakierron aikana kerätään kaikki lähtevät sanomat yhteen nippuun ja lähetetään sellaisenaan PC:lle. Ohjelmallisesti se tarkoittaa sitä, että kun lähetettävä sanoma luodaan, täytyy tehdä tarkastelu sille, mahtuuko sanoma lähetys stringiin vai joudutaanko jakamaan useammalle stringille tai mahtuuko ylipäätään enää sen kierroksen lähetyspakettiin.

```
IF LEN($x.send_data[x.sDataAmount]) + LEN(.$sanoma_lah) < 255 THEN
    $x.send_data[x.sDataAmount] = $x.send_data[x.sDataAmount] + .$sanoma_lah
ELSE
    .pit = 0
    FOR .i = 1 TO x.sDataAmount
        .pit = .pit + LEN($x.send_data[.i])
    END
    .pituus = .pit + LEN(.$sanoma_lah )
    IF .pituus > pt.maxsanomapit THEN
        .retval = -100
    ELSE
        .jama = 255 - LEN($x.send_data[x.sDataAmount])
        .$mahtuu = $LEFT(.$sanoma_lah,.jama)
        .$sanoma = $RIGHT(.$sanoma_lah,LEN(.$sanoma_lah)-LEN(.$mahtuu))
        $x.send_data[x.sDataAmount] = $x.send_data[x.sDataAmount] + .$mahtuu
        x.sDataAmount = x.sDataAmount + 1
        $x.send_data[x.sDataAmount] = ""
        $x.send_data[x.sDataAmount] = .$sanoma
    END
END
```

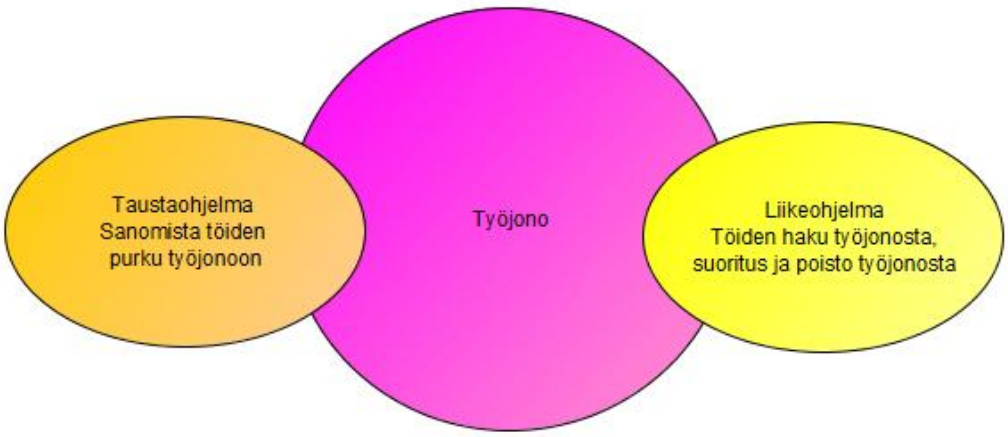
Kuva 3. Lähetettävän sanoman vaatiman tilan tarkistus

Kuvassa 3 on esitetty tarkastelu, mikä luodulle sanomalle tehdään. Tarkastelussa katsotaan, ettei lähetettävän stringin pituuden ja luodun sanoman pituuden summa ylitä sallittua 255 tavua. Yhteispituuden ollessa alle 255 tavua voidaan luotu sanoma liittää suoraan lähetettävään stringiin. Tavumäärän ylittyessä tarkistetaan vielä, mahtuuko luotu sanoma lähetettävään pakettiin. Paketin maksimitavumäärä on käytettävästä protokollasta riippuvainen, joten arvo on parametrina muuttujassa pt.maxsanomapit. Mikäli sanoma mahtuu pakettiin, jaetaan se paketin peräkkäisiin stringeihin. Vastaavasti, jos sanoma ei mahdu pakettiin, palautetaan tieto siitä, että sanoma ei ole mahtunut lähtevään pakettiin, jotta sen lähetystä yritettäisiin uudelleen seuraavalla kierroksella.

8 TYÖJONON TOIMINTA

Työjono toimii rajapintana sanomien purun ja robotin liikeohjelman välissä. Purussa työjonoon lisätään eli kirjoitetaan dataa ja liikeohjelmassa luetaan sekä työn suorittamisen päätteeksi poistetaan dataa. Ennen kuin työjonoon lisätään työ, on varmistuttava sen oikeamukaisuudesta, ja vastaavasti työ voidaan poistaa työjonosta vasta, kun työ on saatu suoritettua loppuun saakka tai tilaus keskeytetään esimerkiksi operaattorin toimesta.

Puskurityyppi määrittelee sen, mihin kohtaan työjonoa seuraava työ lisätään, sekä sen, mistä kohtaan seuraavaksi suoritettava työ luetaan. Puskurityypeistä käydään läpi jono- sekä rengaspuskuri, jotka olivat työn kannalta vaihtoehdot, joista valita.



Kuva 4. Töiden kirjoittamisen ja luennan rajapinta

Kuvassa 4 on kuvattu sanomien purun, työjonon ja varsinaisen työn suorituksen suhdetta. Sanomien purku ja työn suoritus ovat toisistaan muuten täysin erillisiä lukuun ottamatta työjonoa, johon molempien on välttämätöntä päästä käsiksi. Puskurityypistä riippuen voidaan kirjoittaa ja lukea töitä työjonosta samanaikaisesti tai joudutaan lukitsemaan työjono aina jommankumman toiminnan ajaksi.

8.1 Jonopuskuri

Jonopuskuri perustuu siihen, että nimensä mukaisesti töitä lisätään jonoon ja aina jonon ensimmäinen työ luetaan ensin. Luentapaikka pysyy siis aina samana, mutta kirjoituspaikka vaihtelee sen mukaan, luetaanko töitä samaan aikaan. Kun työ suorituksen päätteeksi poistetaan työjonosta, siirtyy jokainen työ jonossa yhden paikan eteenpäin. Voidaan ajatella kirjapinoa, jossa aina poistetaan alin kirja, jolloin ylemmät kirjat siirtyvät alaspäin.

10	
9	
8	
7	
6	
5	5
4	4
3	3
2	2
1	1

10	
9	
8	
7	
6	7
5	6
4	5
3	4
2	3
1	2

10	
9	
8	
7	
6	
5	7
4	6
3	5
2	4
1	3

Kuva 5. Jonopuskurin täyttäminen ja purku

Kuvassa 5 on kuvattu jonopuskurin täyttymistä ja purkamista. Oranssilla pohjalla olevat numerot ovat työnumeroita, joiden siirtymistä kohti jonon alkua voidaan kuvasta seurata. Jonoa täytetään aina jonon viimeisimpään vapaaseen paikkaan ja luenta tapahtuu aina samasta kohtaa jonon ensimmäisestä paikasta.

Hyvänä puolena kyseisessä puskurissa voidaan pitää sitä, että luentapaikka pysyy aina samana. Mahdollisuus lukea tietoa väärästä kohdasta jonoa on siis hyvin pieni. Huonona puolena sitä vastoin on tietojen siirtely. Aina työn poistamisen hetkellä täytyy kaikkia töitä jonossa siirtää paikan verran eteenpäin, ja tällöin myös jonoon töiden lisääminen on pysäytettävä tietojen siirron ajaksi. 10 paikan jonossa siirtoon kuluva aika on melko mitätön, mutta jos jonon kokoa kasvatetaan satakertaiseksi, siirtoon kuluva aika kasvaa samassa suhteessa.

Käytännössä puskurin hallinta robotissa tapahtuu robotin taustaohjelmassa. Mikäli taustaohjelmassa tapahtuu jokin virhe kesken tietojen siirron, on mahdollisuus, että vain osa töistä on siirtynyt seuraavaan paikkaan.

8.2 Rengaspuskuri

Rengaspuskuri poikkeaa jonopuskurista siten, että luentakohta vaihtelee. Puskurin koko on määritelty 1-n, kirjoitus aloitetaan 1 kohdasta samoin kuin luenta. Rengaspuskurissa luetaan aina seuraavan paikan tietoja aina puskurin loppuun saakka, minkä jälkeen aloitetaan luenta nimensä mukaisesti uudelleen ensimmäisestä paikasta.

10			10			10	10		10		
9			9	9		9	9		9		
8			8	8		8	8		8		
7			7	7		7	7		7		
6	6		6	6		6	6		6		
5	5		5	5		5	5		5		
4	4		4	4		4			4	14	
3	3		3	3		3			3	13	
2	2		2	2		2			2	12	
1	1		1			1			1	11	

Kuva 6. Rengaspuskurin täyttäminen ja purku

Kuvassa 6 on hahmoteltu 10 paikan rengaspuskuri. Vasemman reunan numerointi kertoo sarakkeiden paikkanumerot työjonossa. Keltaisella maalatut solut paikat ovat tietoa sisältäviä paikkoja ja niiden numerot ovat työnumeroita. Ensimmäisessä sarakkeessa jonoon on lisätty 6 työtä (1-6). Seuraavassa sarakkeessa ensimmäinen työ on suoritettu ja poistettu puskurista ja puskuriin on lisätty 3 työtä (7-9). Kolmannessa sarakkeessa on suoritettu ja poistettu työt 2-4 sekä lisätty yksi työ (10). Viimeisessä sarakkeessa työt 5-10 on suoritettu ja poistettu puskurista ja seuraavat 3 työtä (11 -14) on lisätty jonoon alkaen taas paikkanumerosta 1.

Rengaspuskurin hyvä puoli on se, että kun työ ja siihen liittyvät parametrit kirjoitetaan yhteen paikkaan, niitä ei siirrellä mihinkään ennen kuin suorituksen lopuksi, jolloin työ poistetaan listasta. Huonoina puolina voidaan pitää kirjoitus- ja luentapaikkojen päivitystä.

8.3 Puskurin valinta

Näistä edellä mainitusta kahdesta puskurityypistä valitsin työtä varten rengaspuskurin. Rengaspuskurissa ei tarvitse jo kirjoitettuja tietoja siirrellä ”turhaan”, minkä koin ohjelmointimielessä turvallisempana. Turhien siirtojen lisäksi rengaspuskuri mahdollistaa samanaikaisen kirjoittamisen ja lukemisen jonosta, koska kirjoitus ja luenta paikka ovat eri. Jonopuskurissa puskuri täytyy lukita kirjoittamisen ja luennan ajaksi vain kyseiselle toiminnolle ylikirjoittamisen, väärän tai vanhan tiedon lukemisen riskin takia. Toisin sanoen töiden siirto jonopuskurissa ei saa olla käynnissä samaan aikaan, kun puskurista luetaan tai sinne kirjoitetaan tietoja.

Puskurin koon kasvaessa rengaspuskurin toiminta nopeutuu tai pysyy samana, kun kirjoitus- ja luentapaikkojen välissä on enemmän paikkoja. Jonopuskurin toiminta sitä vastoin hidastuu puskurin koon kasvaessa, sillä siirtojen määrä kasvaa samassa tahdissa lukiten puskurin pidemmäksi ja pidemmäksi aikaa.

9 KAWASAKIN UDP-OMINAISUUDET

Kiinnostavin osio protokollien vertailussa oli niiden vastaanoton toiminta Kawasaki-robotilla. Työn alkuun oli tarkoitus valita jompikumpi TCP- tai UDP-protokolla työn suorittamiseen. UDP oli vieraampi ja sen todellisen nopeuden ja varmuuden selvitys olisi kiinnostavaa. TCP oli tutumpi ja sen varmuus ja nopeus olivat jo hyvin tiedossa, mutta mahdollisuus kehittää nykyistä tapaa työn tarkoitukseen olisi mielenkiintoista. Täten päädyttiin ensin testaamaan UDP-protokollan ominaisuudet yksinkertaisilla pakettien vastaanottokokeilla. Kokeiden jälkeen olisi tiedossa UDP-protokollan soveltuvuus työn tarpeisiin sekä tulevaisuuden sovelluksiin.

UDP-protokolla oli testattava ja sen ominaisuudet selvitettävä, jotta tiedettäisiin, voidaanko kyseistä protokollaa käyttää työn kaltaisessa järjestelmässä. Kokemusta UDP-protokollan käytöstä oli nopeista poimintasoluista, joissa robotti vastaanottaa aina pyynnöstä yhden työn kerrallaan.

9.1 Koejärjestelyt

Kawasakin UDP-ominaisuuksia testattiin yrityksen laboratoriossa, jossa robotti ja PC yhdistettiin samaan verkkoon. Tehtiin nopeuskokeita erilaisilla variaatioilla sekä koe, missä seurattiin pakettien saapumisjärjestystä robotille. Nopeuskokeita oli 5 kappaletta. Nopeus mitattiin PC:n puolelta, jossa seurattiin saapuneiden sanomien tahtia. Pakettien vastaanottokokeessa PC:ltä lähettiin 255-1472 tavua pitkiä numeroituja sanomapaketteja robotille. Numerointia seurattiin robotin päässä sekä suoritettiin sanomapakettien purkua.

9.2 Nopeuskoe

Kuvassa 7 on kuvattu kaikki viisi tapausta, mitä robotilta lähetettiin, oliko vastaanotto käytössä, printit päällä vai ei sekä mitkä taustaohjelmat olivat käytös-

sä. Mielenkiintoista oli tietää, onko sanomapaketin pituudella vaikutusta sekä mikä on muiden taustaohjelmien vaikutus kommunikoinnin tahtiin.

Case 1:	"Testiviesti" sanoman lähetys ja vastaanotto ilman printtejä sekä taustaohjelmia
Case 2:	"Testiviesti" sanoman pelkkä lähetys ilman printtejä sekä taustaohjelmia
Case 3:	"Testiviesti" sanoman pelkkä lähetys PC3 ja PC4 päällä
Case 4:	250 merkkisen sanoman lähetys ja "Terveisiä" sanoman vastaanotto, ei printtejä PC3 ja PC4 päällä
Case 5:	250merkkinen sanoma*5 lähetys ja "Terveisiä" sanoman vastaanotto, ei printtejä PC3 ja PC4 päällä

Kuva 7. Nopeuskokeen eri tapaukset

9.3 Pakettien vastaanotto

Suoritettiin koe UDP-protokollalla, jossa seurattiin pakettien saapumisjärjestystä robotille. Robotin puolella asetettiin vastaanotto auki ja vastaanotetut paketit tallennettiin maksimissaan 255 merkkiä pitkiksi stringeiksi. Sanomien purussa stringistä etsitään alku-, loppu- ja välimerkkejä, tämä etsintä tapahtuu käymällä stringistä jokainen merkki läpi, kunnes merkki vastaa etsittyä merkkiä. Näin ollen simuloitiin tallennetuille stringeille sanomien purku käymällä jokaisesta stringistä jokainen merkki läpi.

9.4 Koetulokset

Nopeuskokeesta saatiin lähetykselle ja vastaanotolle keskimääräiseksi nopeudeksi 31 pakettia sekunnissa. Liitteessä 1 on taulukoitu kaikkien eri tapauksien nopeudet ja laskettu keskiarvot. Sanomapaketin pituudella eikä muiden taustaohjelmien toiminnalla ollut vaikutusta nopeuteen. Tästä voidaan päätellä, että kommunikoinnin tahti nousee sanoma per lähetys sitä mukaa, mitä enemmän sanomia voidaan liittää yhteen lähetykseen.

Pelkän lähetyksen nopeudeksi taustaohjelmien kanssa saatiin keskimäärin 2270 pakettia sekunnissa. Tästä tuloksesta päädyttiin siihen, että osa robotin

lähetettävistä tilatiedoista voitaisiin siirtää UDP-lähetykseen nopeuden ollessa niin suuri.

Pakettien vastaanottokokeen tulokset eivät olleet yhtä hyvät. Lopputuloksena sanomapakettien numerointia seuratessa huomattiin, että jopa 100 pakettia jäi robotin päässä vastaanottamatta. Ajatus asynkronisesta vastaanotosta ja purusta kariutui tässä vaiheessa.

Testin perusteella pystyttiin todentamaan, että Kawasakin UDP-vastaanottokäsky toimii siten, että kun se esitetään, antaa se ulos määritellyn ajan sisällä saapuneet paketit. Aika on määriteltävissä parametreina käskyn yhteydessä, oletusaika on sekunti. Minkäänlaista puskurointia paketeille, jotka ovat saapuneet ennen vastaanottokäskyä, ei siis ole olemassa. Kun paketti on vastaanotettu, vastaanotto sulkeutuu ja ohjelmassa siirrytään suorittamaan seuraavaa käskyä esimerkiksi purkamaan sanomaa tai tulostamaan tietoja. Testiohjelmassa suoritettiin stringien tallennus ja sanomien purku tai tulostus, ennen kuin palattiin takaisin vastaanottoon. Noissa toiminnoissa meni kaikissa enemmän aikaa kuin PC:llä pakettien lähetykseen, jolloin ohjelman palatessa vastaanottoon oli monia paketteja jo tuhoutunut. Edes ilman stringien tallennusta ja sanomien purkua ei pystytty vastaanottamaan kaikkia paketteja.

Kokeen perusteella päädyttiin siihen, että PC voi lähettää seuraavan paketin vasta siinä vaiheessa, kun on saanut kuittauksen edellisestä paketista. Kuittaus annetaan siinä vaiheessa, kun saapuneiden sanomapakettien työt on purettu työjonoon. Tämä hidastaa kommunikointia, mutta antaa varmuuden siitä, että paketit vastaanotetaan oikeassa järjestyksessä.

Vaikka TCP-kommunikoinnilla tätä ei tapahtuisikaan, ei ole syytä käyttää asynkronista vastaanottoa ja purkua sillä kommunikoinnilla, koska suurin hyöty eriaikaisuudesta saadaan vain UDP-protokollaa käytettäessä, kun voidaan kuittaukset jättää tekemättä niin sovellus- kuin kuljetuskerrostasolla.

9.5 Päätelmät UDP-protokollan käytöstä

Vastaanottokokeen jälkeen pystyttiin todentamaan, että UDP-kommunikoinnin suorituskyky laskee TCP-kommunikoinnin tasolle. TCP-kommunikoinnissa on

se suuri etu tässä tilanteessa, että TCP-puskuri on suurempi kuin UDP:llä, jolloin voidaan lähettää pidempiä merkkijonoja. TCP-puskuri on 4096 tavua pitkä ja UDP-puskuri 1472 tavua pitkä /7/. Tämä johtaa siihen, että yhdellä lähetyksellä PC saa lähetettyä enemmän sanomia robotille. Kokeen perusteella päädyttiin jatkamaan työtä pelkän TCP-protokollan kanssa, sillä UDP-protokollan suorituskyvyn laskiessa TCPprotokollan tasolle ei koettu, että olisi tarpeen jatkaa molemmilla tavoilla.

10 SANOMIEN MÄÄRITTELY

Kuten aiemmin on kerrottu, Orferin kehittämän Kawasaki–PC-kommunikoinnin helppokäyttöisyys ja nopea käyttöönotto ovat kommunikoinnin suurimmat edut. Nämä pohjautuvat hyvin pitkälle laajaan, yksinkertaiseen ja helppolukuiseen sanomaliikenteeseen. Työssä olisi tarkoitus pyrkiä noudattamaan melko samanlaista sanomarakennetta, siten että tulevaisuuden tarpeisiin olisi helppo lisätä ja tarpeen mukaan luoda enemmän sovelluskohtaisia sanomia.

10.1 Sanomarakenne

Sanoma muodostuu karkeasti kahdesta osuudesta, otsikko- ja dataosuudesta. Otsikko-osuudella tunnistetaan, mitä dataa sanoma sisältää. Dataosuuteen on sisällytetty otsikko-osuuden määräämä tieto. Otsikko-osuuden perusteella pystytään päättelemään dataosuuden rakenne, mitä se sisältää, sekä tehdään päättelyt, kuinka sanomaan vastataan. Osuudet pystytään erottamaan toisistaan sovituilla merkeillä. Työhön luodussa sanomaliikenteessä käytetään pienemmyys-, suuremmuus-, dollarimerkkiä sekä risuaitaa. Yksittäinen sanoma on merkitty alkavaksi pienemmyys- ja päättyväksi suuremmuusmerkillä. Dollarimerkki on niin sanottu dataerotinmerkki jolla erotetaan kaikki tiedot toisistaan. Risuaita toimii erotinmerkkinä, jolla erotetaan dataosuus otsikko-osuudesta. Viimeinen numero ennen dataosuutta on tarkistussumma, joka kertoo varsinaisen dataosuuden pituuden välimerkkeineen.

<TilausAlkaa$4#50$1#>

Tilausnumero = 50

Työnumero = 1

<TilausKuittaus$2#50#>

Aktiivinen tilausnumero = 50

Kuva 8. Sanomarakenteen esittely

Kuvassa 8 on kuvattua tilauksen aloitus ja sen kuittaussanomat. Otsikko-osuus on merkitty punaisella, ja dataosuus on risuaitojen välinen osa. ”TilausAlkaa”-otsikkoisen sanoman ensimmäinen datapaikka sisältää tilausnumeron, seuraava paikka ensimmäisen työnumeron ja sen, että siihen vastataan ”TilausKuittaus”-sanomalla, jonka ensimmäinen datapaikka on tilausnumerolle. Jos tilaus on mennyt läpi robotille, niin PC palauttaa ”TilausAlkaa”-sanoman sisältäneen tilausnumeron, ja vastaavasti, jos tilaus ei ole mennyt läpi, niin palautetaan joko 0 tai sen hetkinen aktiivinen tilausnumero.

10.2 Hyötykuorma

Hyötykuorma tarkoittaa yleisesti kuljetusvälineen maksimikuorman määrää pois lukien kuljetusvälineen ominaiskuorman. Sanomaliikenteessä vastaavasti sanoman dataosuus ilman välimerkkejä on sanoman hyötykuorma. Näin ollen mitä lyhyemmin otsikko-osuus voidaan ilmaista ja mitä vähemmän joudutaan käyttämään välimerkkejä tai lähettämään dataa, mikä olisi pääteltävissä jostain toisesta arvosta, niin sitä suurempi on sanomien hyötykuorma. Mitä suurempi hyötykuorma, sitä parempi on tietoliikenteen hyötysuhde. Hyötykuorman merkitys sanomissa kasvaa, kun sanomapaketeissa lähetetään useampia sanomia. 1000 merkkiä pitkä sanomapaketti, jonka hyötykuormaa on ainoastaan 100 merkkiä, on 90 prosenttisesti turhaa tietoa. Mikäli hyötykuorman määrää saadaan kasvatettua, saadaan vastaanotettua enemmän tietoa yhdessä sanomapaketissa.

Esimerkiksi ”TilausAlkaa”-otsikko varaa jo 11 tavua sanomasta ja on siten pidempi kuin varsinainen dataosuus, jos sanoma sisältää pelkästään tilausnu-

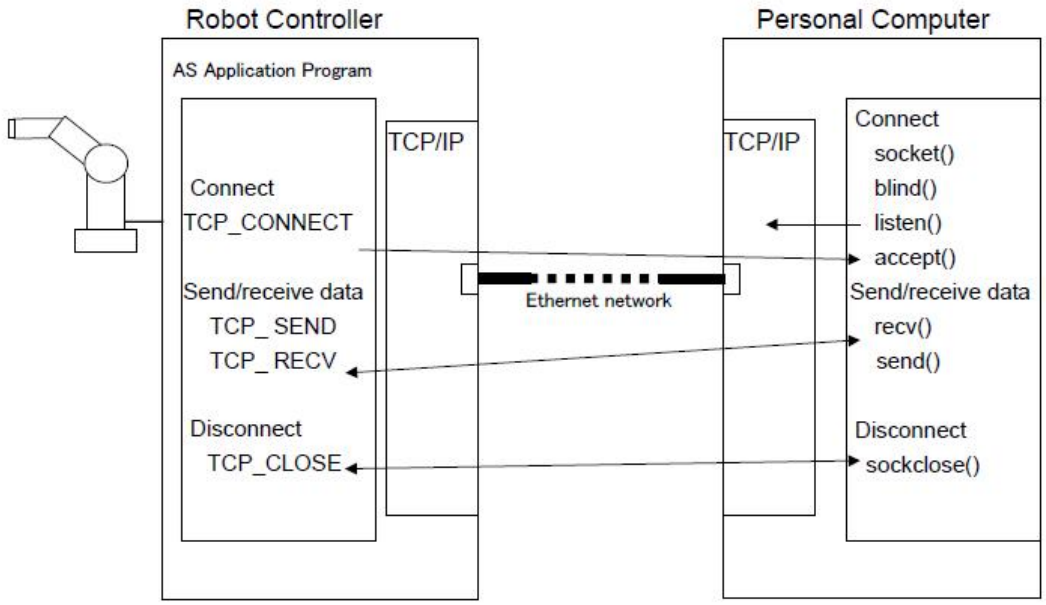
meron ja työnumeron. Otsikko voitaisiin lyhentää ”TA”, jolloin se veisi pelkästään sen 2 tavua, mutta sen ymmärrettävyys ihmiselle huononisi. Hyötykuorman osuuden pienenemisen kustannuksella saadaankin edellä mainitun tapauksen kohdalla helpommin ymmärrettävissä oleva sanoma.

10.3 Tarkistussumma

Tarkistussummaa käytetään sanoman purun jälkeen, kun verrataan puretun dataosuuden pituutta annettuun pituuteen. Kuvassa 8 tarkistussumma on mustalla fontilla oleva arvo. Arvojen poiketessa toisistaan ei dataa saa käyttää. Arvojen poikkeaminen kertoo, että dataosuus on jossain vaiheessa katkennut tai kaikkia sen osia ei ole vastaanotettu. Tarkistussummakin vähentää hyötykuormaa, mutta on pieni harmi siitä, että voidaan havaita virheellisiä sanomia.

10.4 Sanomaliikenteen perusperiaate

TCP-protokollaa käytettäessä on määriteltävä osapuolten kesken, kumpi on palvelin (server) ja kumpi asiakas (client). Yhteyden avaus tapahtuu asiakkaan toimesta, palvelin on valmiina odottaen yhteyden muodostuspyyntöä asiakkaalta.



Kuva 9. PC serverinä ja robotti asiakkaana

Orferin nykyisessä mallissa robotti on asiakas ja PC palvelin, johon robotti yhdistää. Malli juontaa juurensa Kawasakin ohjelmapohjista itseään ylläpitäviin TCP-yhteyksiin. Molemmista tavoista, robotti on asiakas tai palvelin, oli olemassa ohjelmapohjat, mutta robotti asiakkaana oli yksinkertaisempi. Samoin tässä työssä robotti on asiakas, joka yhdistää palvelimeen eli PC:hen. Kuvassa 9 robotti asiakkaana ja PC palvelimena nähdään Kawasaki-robotin käskyjä, joilla yhdistetään PC:hen eli palvelimeen, lähetetään ja vastaanotetaan dataa sekä suljetaan yhteys palvelimeen.

Ristiriitana normaaleihin asiakas–palvelin-yhteyksiin on, että robotti on aktiivinen osapuoli sanomaliikenteessä. PC ei saa lähettää dataa, ellei robotti ole ensin lähettänyt jotain. PC:n lähetystä on tahdistettu siihen, että ennen lähetystä on robotilta ensin vastaanotettava jotain. Tahdistus johtuu siitä, että robotilla ei ole käskyä, millä tyhjentäisi vastaanottopuskuria, siten turvallisinta on antaa PC:n lähettää dataa vain silloin, kun robotti sitä odottaa.

Onnistuneen yhteyden muodostuksen jälkeen robotti lähettää jokaisella ohjelmakierroksella jonkun sanoman PC:lle ja odottaa sen jälkeen vastausta PC:ltä. Tämä tarkoittaa sitä, että jos robotilla tai PC:llä ei ole mitään järkevää tietoa lähettää, on sen lähetettävä niin sanottua huuhaa-tietoa yhteyden ylläpitämiseksi. Mikäli huuhaa-sanomia ei lähetetä, voi sanomien väli kasvaa liian pitkäksi, jolloin TCP:n aikakatkaisu katkaisee yhteyden. Näin ollen täytyy PC:llä vastaanotetut sanomat määritellä joko sanomiksi, joihin on määritelty, ja sanomiksi, joihin ei ole määritelty vastaussanomaa. Robotti odottaa tietyn ajan jakson PC:ltä vastausta, mikäli sitä ei tule ajan sisällä, palaa robotti ohjelmakiertonsa alkuun.

11 TYÖN KUVAUS

PC:llä olevaan ohjelmaan voidaan ladata haluttu kuva, mikä lähetetään myöhemmin robotille piirrettäväksi. Kuva puretaan mahdollisimman yhtenäisiksi viivoiksi, joista lähetetään aina suoran viivan alku- ja loppukoordinaatit robotille. Robotti siirtyy koordinaattien välin joko piirtäen tai irti piirtoalustasta riippuen siitä, onko kyseessä piirto- vai siirtokäsky.

11.1 Määritelmät

Tilaus tarkoittaa työssä yhtä kuvaa, ja työt ovat joko piirto tai alustalla siirtymiskäskyjä. Robotilla ei voi olla aktiivisena kuin yksi tilaus, ja töiden suoritusjärjestys on aina numerojärjestyksessä alkaen 1:sta. PC:ltä voidaan lähettää myös työjonon ohi meneviä käskyjä, joiden ei tarvitse sisältää työnumeroita, vaan ne voidaan ottaa heti käyttöön, kun käsky on purettu ja robotti on suorittanut mahdollisesti kesken olevan työn loppuun. Työn sovelluksessa on käytössä vain yksi työjonon ohi menevä käsky, ja se on tilauksen keskeytys, jolla voidaan keskeyttää kuvan piirtäminen. Keskeytyksen jälkeen ei voida jatkaa kuvan piirtämistä, vaan ainoa mahdollisuus on aloittaa saman tai eri kuvan piirtoalusta. Myöhemmin on mahdollista lisätä enemmänkin työjonon ohi meneviä sanomia, kuten erilaisia toimintaviiveitä tai robotin nopeusasetuksia.

11.1.1 Työjonon luominen

Haasteellisin osuus ohjelmoinnissa oli työjonon luominen ja siihen ominaisuuksien luominen, joilla olisi mahdollisuus selviytyä virhetilanteista sekä muista poikkeustilanteista. Ohjelmassa tarvittaisiin jokaiselle työjonon paikalle muuttuja, joka kertoisi, onko paikka varattuna vai vapaana. Kyseiset muuttujat tulisivat olemaan globaalimuuttujia indekseillä 1-n (n= työjonon koko). Työjonon ehdoton maksimikoko on 9999, sillä 5 numeron indeksejä ei voida Kawasaki-robotilla käyttää.

Aiemmin oli päätetty puskurityypiksi rengaspuskuri, täytyy pitää huoli luenta- ja kirjoituspaikkojen numeroinnista. Tätä varten loin oman listausohjelman, jolla tarkistetaan työjonon tila (tilaa/täynnä), päivitetään alustavasti kirjoituspaikan numero sekä seuraavaksi saapuvan työn numero. Ohjelman avulla pystytään myös tyhjentämään työjono esimerkiksi tilauksen keskeyttämisen johdosta. Koska edellä mainitut tarkastelut tehdään aina ennen sanoman purkua, tietojen todenperäisyyden tarkastelua sekä työn jonoon lisäämistä, tulee lopullinen paikan numeron sekä seuraavan työnumeron päivitys tehdä vasta sitten, kun kaikki tiedot puretusta sanomasta on tarkastettu ja kirjoitettu varsinaisiin muuttujiin, joita robotti käyttää liikeohjelmassa.

```
;Paikan indeksin päivitys
.seurwriteInd = x.writeInd + 1
.writeInd = x.writeInd
IF .seurwriteInd > pt.maxWorkInd THEN
    .writeInd = 0
END

;Paikan tilan tarkastus
IF x.dataOk[.writeInd+1] == TRUE THEN
    ;Varattu
    .retval = -200
    RETURN
END

;Työnumeron päivitys
.seurTyo = x.aLastWork + 1
.viimTyo = x.aLastWork
IF .seurTyo > pt.maxWork THEN
    .viimTyo = 0
END


;Kaikki ok
x.writeInd = .writeInd
x.aLastWork = .viimTyo
.retval = 0
```

Kuva 10. Työjonon paikan sekä seuraavan työnumeron päivitys

Kuvassa 10 on ohjelmasta osa, jossa tarkistetaan seuraavan työn kirjoituspaikan ja numeron oikeellisuus. Kumpikaan arvoista ei saa ylittää annettua maksimiarvoa, vaan silloin täytyy arvo palauttaa taas alkuun. Ohjelmassa tarkistetaan myös työjonon seuraavan paikan tila (vapaa/varattu) ja estetään kirjoitus, mikäli paikka on vielä varattu (ruuhka). Kaikki muuttujat pidetään paikallismuuttujina (. – alkuiset), kunnes voidaan todeta kaikkien ehtojen olevan ok, jolloin varsinaisten muuttujien arvot päivitetään. Globaalimuuttuja pt.maxWorkInd kertoo työjonon paikkojen maksimimäärän ja vastaavasti pt.maxWork ilmaisee työnumeron maksimiarvon. Työjonon paikan tilan ilmaisee x.dataOk[] taulukkomuuttuja. Muuttujan arvo asetetaan todeksi, kun paikalle on kirjoitettu työn parametrit ja muuttuja nollataan robotin liikeohjelmassa, kun paikassa oleva työ on suoritettu.

Kyseinen listausohjelma toimii myös rajapintana sanomien purun, töiden kirjoittamisen ja töiden lukemisen sekä suorittamisen välillä. Ohjelmakutsu suoritetaan aina robotin taustaohjelmassa, normaalitilanteessa vain töiden lisäyksen yhteydessä. Erikoistilanteissa, joissa joudutaan esimerkiksi resetoimaan tai tyhjentämään, työjonopyyntö suoritetaan liikeohjelmasta sisäisillä signaaleilla, joiden perusteella taustaohjelma suorittaa pyynnön. Liikeohjelmasta voidaan pyytää kahdenlaista resetointia, pelkkä listan tyhjennys tai sekä tyhjen-

nys että aktiivisen työnumeron ja tilausnumeron nollaus. Listan tyhjennys suoritetaan, jos esimerkiksi työjonossa seuraavana olevan työn numero on väärin. Tyhjennys ja työnumeron sekä tilausnumeron nollaus suoritetaan keskeytyksen ja lopetuksen yhteydessä.

Työjonon paikkamäärän lisäksi tuli määritellä maksimityönumero. Sovelluskohtaisesti maksimityönumero tulisi määritellä siten, että se kattaa kaikki tilauksen työt. Kun maksimityönumero kattaa koko tilauksen työt, vähenee virheen mahdollisuus tilanteessa, jossa maksimityönumero on saavutettu ja numerointi on aloitettava alusta. Maksimityönumero täytyisi siis olla sovittuna yhteisesti robotin ja PC:n puolella, jotta :n puolella aloitettaisiin työnumerointi uudestaan alusta maksimityönumeron jälkeen ja vastaavasti robotin päässä osattaisiin odottaa seuraavaksi työnumeroksi niin sanottua minimiarvoa maksimin jälkeen. Työnumerointi toimii siis vastaavasti kuin varsinaisen työjonon paikkojen päivitys ja aloittaa numeroinnin uudelleen alusta numeroinnin tullessa maksimiin. Työnumeroita ei käytetä indekseinä, joten maksimiarvoksi voidaan asettaa jokin huomattavasti suurempi luku, joka kattaisi normaalitilauksen kaikki työt.

11.2 Kuittaukset

Jotta tilauksen etenemistä voitaisiin seurata PC:ltä, täytyy robotin kuittailla tehtyjä töitä vähintään silloin, kun sillä ei ole mitään muuta lähetettävää. Sovelluksessa suoritetaan työt aina numerojärjestyksessä, eli kun robotti kuittaa työn tehdyksi, voidaan olettaa PC:n puolella kaikki kuitattavan työn numeroa arvoltaan pienemmät työt tehdyiksi. Täten kyseisessä sovelluksessa ei olisi välttämätöntä kuitata kaikkia suoritettuja töitä, vaan voitaisiin sopia, että kuitataan vain joka toinen tai kymmenes työ.

Toisenlaisissa sovelluksissa, joissa töiden suoritusjärjestyksellä ei ole väliä, vaan kaikki työjonossa olevat työt ovat niin sanotusti tasa-arvoisia, on välttämätöntä kuitata kaikki suoritetut työt. Ilman kuittausta ei voida muuten PC:n puolella varmistua, mitkä työt ovat suoritettuja ja mitkä vielä mahdollisesti kesken. Kyseisissä sovelluksissa voidaan hypätä työjonossa töiden yli, mikäli jokin ehto estää yli hypättävien töiden suorittamista tai on tehokkaampaa suorittaa jokin muu työ ensin. Esimerkiksi lineaarisesti liikkuva robotti suorittaa työn

aivan liikeratansa loppupäässä, tällöin seuraavaa työn ollessa liikeradan alkupäässä olisi ajallisesti tehokkainta ensin tarkistaa, onko työjonossa robotin sen hetkiselle alueelle töitä, jotka suoritettaisiin ennemmin.

Vaikka tämän työn sovelluksen kannalta kaikkien suoritettujen töiden kuittaus ei ole välttämätöntä, päädyttiin kuittaamaan kaikki työt, jotta saataisiin tämä ominaisuus mukaan laskettuun tahtiaikaan. Kuittaus suoritetaan liikeohjelman lopuksi samaan aikaan, kun kyseisen työn paikka työjonossa vapautetaan.

Töiden lisäksi sovittiin robotin kuittaavan tilauksen aloituksen sekä päättymisen. On mahdollista, että robotilla ja PC:llä on eri tilaukset käynnissä tai robotilla on tilaus kesken, kun PC:ltä haluttaisiin lähettää jo seuraavaa tilausta. PC sekä robotti lähettävät toisilleen tilasanomia, joiden perusteella tiedetään, mikä tilaus ja missä vaiheessa tilaus sillä hetkellä on. Tilasanomasta saadaan tarvittava tieto PC:lle lopettaa robotilla kesken oleva tilaus tai robotilla voitaisiin tilaus lopettaa automaattisesti, jos PC:n tilatiedoissa tilausnumero on 0 tai poikkeaa robotilla sillä hetkellä kesken olevasta tilauksesta. Edellä mainitusta turvallisempaa on, että PC:ltä lähetetään tilauksen aloitus, ja mikäli robotilla on vielä aktiivisena tilaus, robotti kuittaa tilauksen aloituksen epäonnistuneeksi aktiivisen tilausnumeron takia. Kuittauksen jälkeen PC:ltä voidaan lähettää ensin robotilla aktiivisena olevan tilauksen lopetus tai keskeytys ja odottaa tähän kuittaus, että robotin tilausnumero on taas 0. Kun robotilla ei ole enää tilausnumeroa aktiivisena ja sille lähetetään tilauksen aloitus kuittaa sen hyväksytyksi, jolloin PC aloittaa tilaukseen kuuluvien töiden lähetyksen.

11.3 Suurin muutos aikaisempaan

Kuten kappaleessa 8 käytiin läpi, robotilta lähetettävät sanomat ovat kahdenlaisia: sanomia, joihin on määritelty vastaus, sekä sanomia, joihin ei ole määritelty vastauksia. Kun aiemmin on lähetetty robotilta vain yksi sanoma per paketti, nyt yksi paketti saattaakin sisältää sekä kuittauksen suoritetusta työstä, mihin ei ole määritelty vastausta, sekä töiden lähetyksen, johon on määritelty vastaus. Mikäli PC:n puolelta vastataan paketin ensimmäiseen sanomaan, johon ei ole määritelty vastausta tutkimatta, onko paketissa sanomia, joihin on määritelty vastaus, lähtee ensin niin sanottu huuhaa-sanoma ja myöhemmin käsitellyn sanoman, johon määritelty vastaus menee PC:n lähetysjonoon.

Näin ollen lähettää robotti uudelleen lähetyspyynnön uudelleen samalle työnumerolle, koska se ei saanut määriteltyä vastausta ensimmäiseen pyyntöön. Jos PC:n lähetyslistaa ei nollata tässä vaiheessa, niin PC lähettää kahteen kertaan samat sanomat.

12 SUORITUSKYKYKOE

Varsinainen työ suoritettiin Orferin laboratoriossa. Robotti ja PC kytkettiin samaan verkkoon minkä jälkeen robotti ajettiin kotiasemaan ja käynnistettiin automaatille. Robotin käynnistyessä se avasi yhteyden PC:lle ja sanomien välityksellä saatiin tilaus robotille. PC lähetti robotin kuittauksesta lisää töitä kunnes sai työjononsa täyteen töitä. Työjonon tyhjentyessä määritellyn verran pyysi robotti PC:tä jatkamaan lähetystä.

Robotti haki työn työjonosta, tarkisti sen oikeellisuuden, ”suoritti” työn ja lähetti aina kuittauksen valmiiksi saadusta työstä PC:lle. Robottia ei pysäytetty missään välissä, vaan annettiin suorittaa töitä ja samanaikaisesti täyttää työjonoaan. Useammilla ajastimilla ohjelman sisällä tallennettiin viimeisimpiä tahtiaikoja sekä 20 viimeisen kierroksen keskimääräiset tahtiajat.

12.1 Mitatut tahtiajat

Tahtiajat mitattiin, sanomapakettien purkamisesta sisältäen työn työjonoon lisäämisen, sanomapaketin lähetyksestä paketin vastaanottoon ja purkuun kokonaisuudessa kulunut aika, työjonosta työn hakuun, työnkuittaukseen kuluva aika sekä työn alusta seuraavan työn alkuun, josta saatiin koko työkierrolle tahtiaika. Ajat mitattiin siten, että robotti ei liikkunut mihinkään, eli robotin työkierto ei sisältänyt yhtään liikekäskyä. Mittaukset siten, että robotti liikkuu ja piirtää kuvaa, suoritetaan myöhemmin, kun on enemmän aikaa käytettävissä. Robotin liikkuessa sovelluksen tahtiaika pitenee ja siten antaa enemmän aikaa kommunikoinnille.

Yllättävintä mittaustuloksissa oli, että pakettien purku ja töiden lisäys työjonoon olikin töiden kuittauksen jälkeen kaikista nopein osuus. Purkuun ja töiden lisäämiseen jonoon meni alle 50 ms. Tästä pystyttiin jo päättelemään, että muutoksena vanhaan tapaan olisi tehokkainta lähettää PC:ltä niin monta sanomaa paketissa kuin pakettiin vain mahtuu. Sanomapaketin koolla ei ollut

juurikaan merkitystä itse purkunopeuteen. Robotin koko työkierron tahti oli keskimäärin alle 110 ms, siinä ajassa työjonosta on haettu työ, suoritettu se sekä kuitattu työ PC:lle. Valmiin työn kuittaus tapahtuu robotin liikeohjelmasta sisäisillä signaaleilla, joiden mukaan robotin taustaohjelmassa lähetetään kuittaus. Sisäisien signaalien tarkastelu tapahtuu taustaohjelman alussa, joten tästä voidaan päätellä, että koko taustaohjelman kiertoajan on oltava vähintään yhtä nopea tai nopeampi kuin liikeohjelman, jotta liikeohjelmassa ei jouduttaisi odottamaan kuittauksen läpimenemistä. Liitteessä 2 on laskettu mainitut keskiarvot.

12.2 Havaitut virheet

Havaittuja virheitä kokeen aikana olivat muutamat virheet sanomien purussa. Virheet eivät kuitenkaan aiheuttaneet sovelluksen pysäyttämistä vaan ainoastaan sen, että PC:ltä pyydettiin lähettämään uudelleen samoja töitä. Virheen syytä en toistaiseksi ole löytänyt, mutta virhe liittyy useampia sanomia sisältäviin stringeihin, jotka purkamisen aikana osittain rikkoutuvat.

Toinen useammin toistuva virhe oli se, kun PC lähetti useampaan kertaan peräkkäin samaa dataa. Kyseinen virhe on täysin lähtöisin PC:n lähetyspuskurin toiminnasta, jota on kappaleessa 9.4 avattu enemmän. Virheestä päästiin lähes täysin eroon, kun PC:n puolella tyhjennettiin lähetyslistaa ennen lähetystä.

13 LOPPUPÄÄTELMÄ

Vaikka työssä ei päästy tavoitteeseen luoda piirtävää robottia, saatiin sekä omasta että työnohjaajan mielestä hyödyllisiä tuloksia niin kommunikoinnin tahdista kuin työjonon toiminnasta robotilla. 110 ms tahtiaika oli huomattavasti parempi, mitä ennakkoon osattiin odottaa. Useamman sanoman lähetys ja vastaanotto yhdessä paketissa ilman muutoksia tahtiajassa vahvistivat sitä käsitystä, että tulevaisuudessa tullaan pyrkimään sanomien paketoimiseen.

13.1 Käytettävyys muissa järjestelmissä

Työhön luotua työjonoa sekä kommunikointipohjaa voitaisiin käyttää useimmissa yleisimmistä sovelluksista. Lavaussovelluksissa voitaisiin lähettää

lavauksen aikana muuttuvaa lavauskuviota. Lavauskuviota voitaisiin muuttaa tai laskea osittain uudelleen kesken ajon esimerkiksi tuotteen mittojen muuttuessa tai jonkun tuotteen puuttuessa. Robotille lähetettäisiin tällöin työjonoon niiden vientien tiedot, jotka ovat jo varmoja ja loput sitä mukaa, kun niiden paikat lavalla on laskettu tai tuotteen mitat varmistettu.

Lasketulla 110 ms tahtiajalla työn sovellusta voitaisiin käyttää myös poimintasovelluksiin, joissa robotin tahtiaika on alle sekunnin luokkaa. Mikäli käytetään kameroita tuotteiden tai jättöpaikan paikantamiseen, voidaan lähettää järjestyksessä seuraavien tuotteiden poiminta- tai jättöpaikat. Liikkuvalta kuljettimelta poimittaessa tai sellaiselle tuotteita jätettäessä on ollut käytössä kuljetinseuranta, jolla paikannetaan kuljettimen suunnassa tuotteen sijainti. Kuljetinseurantaan voidaan liittää työn sovelluksessa luotuja ominaisuuksia siten, että tuotteen sijaintitiedot niin poikittaissuunnassa kuin kiertymäsuunnassa voidaan havaita kameroilla ja lähettää robotille kuljettimen ensimmäisestä tuotteesta lähtien. Mitä aiemmin kameroilla tunnistetaan tuotteita, sitä enemmän ennakkoon voidaan täyttää robotin työjonoa näillä sijaintiedoilla.

Purkusovelluksissa, joissa robotti purkaa tuotteita pinosta, voidaan kuten lavauksessakin muuttaa purkukuviota purun edetessä. Erona lavauskuvion muutokseen purkukuvion muutos voi tulla kameroilta, joilla voidaan tunnistaa esimerkiksi korkeuden tai muun sijaintitiedon poikkeama verrattuna kuvioon. Varsinaista työjonoa purkusovelluksissa ei välttämättä olisi tarpeen luoda, ellei pystytä kuvaamaan kerrosta kokonaisuutena ja lähettämään kyseisen kerroksen kaikkien tuotteiden sijaintitietoja kerralla.

Lavaus-, poiminta- ja purkusovellukset ovat kaikki toisistaan poikkeavia sovelluksia. Kaikissa edellä mainituissa sovelluksissa olisi kuitenkin mahdollista käyttää työssä luotua kommunikointia ja työjonoa, mistä voidaan päätellä työssä luotuja ominaisuuksia joustavina muihinkin sovelluksiin kuin vain työssä mainittuun piirtämiseen.

13.2 Jatkokehitys

Seuraava askel olisi kehittää kommunikointipohjaa sellaiseksi, että se olisi käytettävissä jossakin todellisessa projektissa. Tämä vaatisi tarkemman sa-

nomaliikenteen suunnittelun siten, että olisi mahdollista säilyttää osa vanhoista sanomista ja että uusia sanomia voitaisiin käyttää eri sovelluksissa. Myös muuttujat tulisi nimetä yrityksen mallin mukaan. Työjonoa voitaisiin pitää optiona siten, että se voitaisiin ottaa käyttöön tarvittaessa. Kaikki muutokset sanomaliikenteeseen tulisi tehdä myös PC:n puolelle.

Yrityksellä on ollut tapana, että on määritelty, mistä projektista lähtien on mikäkin kommunikoinnin versio käytössä. Työn pohjalta tulevat muutokset olisivat sen verran isoja, että mikäli ne otettaisiin käyttöön, vaadittaisiin sisäistä koulutusta ominaisuuksien käyttöön.

LÄHTEET

(1) Hakala, M. & Vainio, M. 2005. Tietoverkon rakentaminen. Jyväskylä: Docendo

(2) Mitchell, B. 2018. Network Protocols. WWW-dokumentti. Päivitetty 6.2.2018. Saatavissa: https://www.lifewire.com/definition-of-protocol-network-817949 [viitattu 9.11.2017]

(3) Kaario, K. 2002. TCP/IP-verkot. Jyväskylä: Docendo

(4) Learn-Networking. 2008. The TCP/IP Stack and the OSI Model. WWW-dokumentti. Saatavissa: http://learn-networking.com/tcp-ip/the-tcpip-stack-and-the-osi-model [viitattu 10.5.2018]

(5) Rytiniemi, J., Robotiikan suunnittelupäällikkö. Haastattelu 4.5.2018, Orfer Oy

(6) Kawasaki. 2015. Kawasaki Robot Controller E series AS Language Reference Manual. PDF-dokumentti. Orferin kirjasto. Viitattu[8.5.2018] 90209-1022DEC_E-Series-AS-Language_Reference_Manual.pdf

(7) Kawasaki. 2009. Kawasaki Robot Controller E series TCP/IP Communication Manual. PDF-dokumentti. Orferin kirjasto. Viitattu[8.5.2018] 90201-1249DEA_TCPIP Communication (Eseries).pdf

(8) Orfer Oy. 2017. Orfer historia. WWW-dokumentti. Saatavissa: https://www.orfer.fi/suomeksi/WITHORFER/HISTORIA/2010LUKU/tabid/13092/language/en-US/Default.aspx [viitattu 10.5.2018]

KUVA- TAI TAULUKKOLUETTELO

Case 1: "Testiviesti" sanoman lähetys ja vastaanotto ilman printtejä sekä taustaohjelmia
Case 2: "Testiviesti" sanoman pelkkä lähetys ilman printtejä sekä taustaohjelmia
Case 3: "Testiviesti" sanoman pelkkä lähetys PC3 ja PC4 päällä
Case 4: 250 merkkisen sanoman lähetys ja "Terveisiä" sanoman vastaanotto, ei printtejä PC3 ja PC4 päällä
Case 5: 250merkkinen sanoma*5 lähetys ja "Terveisiä" sanoman vastaanotto, ei printtejä PC3 ja PC4 päällä

	Case 1 Paketin saapumisaika PC:lle		Case 2 Paketin saapumisaika PC:lle		Case 3 Paketin saapumisaika PC:lle		Case 4 Paketin saapumisaika PC:lle		Case 5 Paketin saapumisaika PC:lle	
1	0		0		0		0		0	
2	0,03213893	0,03213893	0,00023825	0,000238	0,0000438	0,0000438	0,03199281	0,031993	0,02709407	0,027094
3	0,06447685	0,03233792	0,00059167	0,000353	0,00246334	0,0024195	0,06355819	0,031565	0,05918317	0,032089
4	0,09666637	0,03218952	0,00093942	0,000348	0,00293947	0,0004761	0,09732526	0,033767	0,09098605	0,031803
5	0,12820986	0,03154349	0,00120977	0,00027	0,00336349	0,000424	0,12928635	0,031961	0,12342931	0,032443
6	0,16027328	0,03206342	0,00149635	0,000287	0,00363044	0,000267	0,16227974	0,032993	0,15502074	0,031591
7	0,1926633	0,03239002	0,0017735	0,000277	0,00386756	0,0002371	0,19564165	0,033362	0,18945161	0,034431
8	0,224877	0,0322137	0,00205517	0,000282	0,00410241	0,0002349	0,227622	0,03198	0,22144969	0,031998
9	0,25689057	0,03201357	0,00236101	0,000306	0,00441958	0,0003172	0,25972998	0,032108	0,28754598	0,066096
10	0,29052624	0,03363567	0,00269253	0,000332	0,00482926	0,0004097	0,29130858	0,031579	0,30978361	0,022238
11	0,32325721	0,03273097	0,0030067	0,000314	0,00528311	0,0004539	0,32377073	0,032462	0,31725404	0,00747
12	0,35472819	0,03147098	0,00333328	0,000327	0,00554439	0,0002613	0,35595384	0,032183	0,34912679	0,031873
13	0,38667533	0,03194714	0,00369651	0,000363	0,00578076	0,0002364	0,38939542	0,033442	0,3815051	0,032378
14	0,41860583	0,0319305	0,00398801	0,000292	0,00602468	0,0002439	0,42177525	0,03238	0,41297382	0,031469
15	0,45070964	0,03210381	0,00427497	0,000287	0,00624669	0,000222	0,45379448	0,032019	0,44497043	0,031997
16	0,48285839	0,03214875	0,00468238	0,000407	0,00663749	0,0003908	0,48573557	0,031941	0,47750279	0,032532
17	0,51692981	0,03407142	0,00512075	0,000438	0,00696561	0,0003281	0,51774764	0,032012	0,50899112	0,031488
18	0,5486092	0,03167939	0,00558895	0,000468	0,00725219	0,0002866	0,55922222	0,041475	0,54106516	0,032074
19	0,58012211	0,03151291	0,00620063	0,000612	0,00766942	0,0004172	0,58160824	0,022386	0,57300204	0,031937
20	0,61209148	0,03196937	0,00662881	0,000428	0,00807532	0,0004059	0,61981678	0,038209	0,60495561	0,031954
21	0,64464575	0,03255427	0,00703735	0,000409	0,00855673	0,0004814	0,65178585	0,031969	0,63741696	0,032461
22	0,67662007	0,03197432	0,00738321	0,000346	0,0090272	0,0004705	0,68379939	0,032014	0,66900688	0,03159
23	0,70865142	0,03203135	0,00768528	0,000302	0,00951352	0,0004863	0,71580356	0,032004	0,7014215	0,032415
24	0,74086887	0,03221745	0,00797979	0,000295	0,00990469	0,0003912	0,74752563	0,031722	0,73348564	0,032064
25	0,77258527	0,0317164	0,00833169	0,000352	0,01028114	0,0003764	0,779239	0,031713	0,8162846	0,082799
26	0,80461359	0,03202832	0,0088267	0,000495	0,0109219	0,0006408	0,81132394	0,032085	0,84643972	0,030155
27	0,83657658	0,03196299	0,00918956	0,000363	0,01137575	0,0004539	0,84379363	0,03247	0,84931314	0,002873
28	0,86833906	0,03176248	0,00944442	0,000255	0,01187113	0,0004954	0,875927882	0,032134	0,86141461	0,012101
29	0,90061355	0,03227449			0,01231743	0,0004463	0,90797597	0,032048	0,89338058	0,031966
30	0,93235636	0,03174281					0,93987137	0,031895	0,92849086	0,03511
Keskiarvo s/paketti		0,032150219		0,00035		0,0004399		0,032409		0,032017
Pakettia sekunnissa		31,10398689		2858,831		2273,2015		30,85529		31,23348

Purku ja töiden lisäys työjonoon	Robotin koko työkierto
0,0361328	0,107983
0,0117188	0,108105
0,0200195	0,108008
0,0219727	0,108105
0,0117188	0,108203
0,0117188	0,108081
0,0117188	0,108008
0,0180664	0,108301
0,0175781	0,107983
0,0576172	0,108105
0,0543634	0,108008
0,0195259	0,108081
0,0598612	0,109717
0,027077877	0,108206769

Tahdin ka sekunneissa