

Opinnäytetyö (AMK)

Tietotekniikka

Sulautetut ohjelmistot

2016

Mikko Hollo

ROBOTIN TILATIETOJEN VÄLITYS ULKOISELLE PALVELIMELLE

Mikko Hollo

ROBOTIN TILATIETOJEN VÄLITYS ULKOISELLE PALVELIMELLE

Työn tarkoituksena oli kehittää teollisuusrobotin ja palvelimen välille yhteys, jonka avulla robotilta saadaan välitettyä tilatietoja palvelimelle. Työ koostui robotille ja palvelimelle kehitettävästä ohjelmasta ja niiden välisestä kommunikaatiosta. Laitteistona työssä käytettiin Yaskawa Motoman -teollisuusrobottia ja siihen Ethernet-väylään kytkettyä kannettavaa tietokonetta, joka toimi palvelimena.

Työ aloitettiin opiskelemalla robottivalmistajan tarjoamia MotoPlus-ohjelmointioppaita. Näistä oppaista saadun tiedon perusteella pystyttiin ohjelmoimaan robotille ohjelma, joka kerää robotilta dataa. Ohjelma kuuntelee TCP-porttia, johon palvelimelta tulee komento kerätä data ja lähettää se takaisin palvelimelle.

Robottiohjelman valmistuttua kehitettiin palvelimelle ohjelma, jonka avulla pyydettiin TCP-yhteyden välityksellä robotilta dataa. Sama ohjelma kirjoitti datan XML-tiedostoon jatkokäyttöä varten. Ohjelmaan tehtiin silmukkarakenne, jonka avulla dataa voidaan pyytää robotilta halutun ajan kuluttua uudelleen.

Tällä hetkellä robotilta voidaan pyytää vain servomoottoreiden vääntöä ja nopeutta ja pyydettävä tieto on määritelty molempien ohjelmien koodissa. Palvelimen ohjelmaan voisi jatkokehityksenä tehdä käyttöliittymän, jonka avulla käyttäjä voisi määritellä halutun datan, jota robotilta pyydetään. Tämän lisäksi vaihtoehtoja voisi lisätä. Robottiohjelmaan tehtävät muutokset jatkokehityksen kannalta olisivat lisäykset datan keruuta ja palvelimelta tulevaa pyyntöä varten.

Laajemmin työtä voi kehittää kohti esineiden internetiä, jolloin voidaan kytkeä monia robotteja suoraan pilvipalveluun ja tallentaa datan keskitetysti pilveen. Tällöin datan keruu helpottuu, sillä käyttäjän ei tarvitse mennä fyysisesti robotin välittömään läheisyyteen vaan voi pyytää etäyhteyden kautta tiedot robotilta.

ASIASANAT:

Robotiikka, IoT, MotoPlus, Teollisuusrobotti

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Embedded software

2016 | 30

Jari-Pekka Paalassalo

Mikko Hollo

STATUS INFORMATION TRANSFER FROM A ROBOT TO A SERVER

The objective of this thesis was to develop a communication system for an industrial robot and a server. The communication system is needed for transferring sensor data for further use from the robot to the server.

The work started with setting the requirements for the system. After this, the robot side program was built to listen up a request through a TCP connection from the server. After receiving the request, the robot collected the sensor data and transferred it back to the server. Then the program went back to listen up the next request from the server.

The server program was created to first send the request to the robot and then receive the data. After that, the server program processes the data to a readable form and saves it to XML file.

At this point, the system only retrieves the robot's torque and current speed, but the system could be developed to handle much more data. For example, the robot's position could be saved to an XML file and then transferred to simulating software for monitoring purposes.

KEYWORDS:

Robotics, Industrial robot, IoT, MotoPlus

SISÄLTÖ

KÄYTETYT LYHENTEET	6
1 JOHDANTO	7
2 TAUSTAA	8
2.1 Teollisuusrobotit	8
2.1.1 Robottityypit	8
2.1.2 Yaskawa Motoman	10
2.1.3 MotoPlus	10
2.2 Esineiden internet	11
3 ROBOTTIOHJELMA	12
3.1 Ohjelman toimintaperiaate	12
3.2 Vääntömomentin palauttava funktio	13
3.3 Nopeuden palauttava funktio	14
4 PALVELIMEN OHJELMA	15
4.1 Yhteyden luominen	16
4.2 Datan purku	17
4.3 Datan Kirjoitus XML-tiedostoon	18
5 YHTEENVETO	20
LÄHTEET	21

LIITTEET

- Liite 1. Robottiohjelman lähdekoodi.
- Liite 2. Palvelimen ohjelman lähdekoodi.
- Liite 3. Palvelimen ohjelman luoma XML-tiedosto.

KUVAT

Kuva 1. Yleisimmät robottirakenteet (ISO 8373).	9
Kuva 2. Ohjelman osa, jossa vastaanotetaan pyyntö, haetaan data ja lähetetään data palvelimelle.	12
Kuva 3. Servomoottorien väännön tallentamiseen tarkoitettu funktio.	13
Kuva 4. Servomoottorien nopeuden seurantaan tarkoitettu funktio.	14
Kuva 5. Palvelimen ohjelman määritelmiä.	15
Kuva 6. Kommunikaatio robotin kanssa.	16
Kuva 7. Datan purku	17
Kuva 8. XML-tiedostoon kirjoitus.	18
Kuva 9. Robotilta pyydetty data XML muodossa.	19

KÄYTETYT LYHENTEET

API	Application Programming Interface. Ohjelmointirajapinta. Määritelmä, jonka mukaan ohjelmat keskustelevat keskenään. Yleensä osa SDK:ta.
IoT	Internet of Things. Esineiden internet
MotoPlus	Motomanin kehittämä ohjelmointiympäristö
Operaattori	Robotin käyttäjä
SDK	Software Development Kit. Ohjelmisto, jonka avulla voidaan luoda tietylle alustalle sopivia ohjelmia.
TCP	Transmission Control Protocol. Tietoliikenneprotokolla, jolla luodaan yhteyksiä verkossa olevien koneiden välille.
Telnet	Yhteysprotokolla, jota voidaan käyttää verkossa olevien koneiden väliseen kommunikaatioon.
XML	Extensible Markup Language. Merkintäkieli, jota käytetään tiedonvälitykseen järjestelmien välillä.

1 JOHDANTO

Teollisuusrobotit yleistyvät maailmalla huimaa vauhtia. Teknologian kehityksen ansios-
ta robotit voidaan jo kytkeä Internetiin, ja tätä kautta hyödyntää robotin tiedonkeruu ja
pilvipalveluiden tallennuskapasiteetti.

Tämän työn tarkoituksena on kehittää järjestelmä, jonka avulla teollisuusrobotilta voi-
daan kerätä dataa ja välittää se ulkoiselle palvelimelle. Lisäksi halutaan selvittää, mitä
jatkokehitysmahdollisuuksia järjestelmällä voi olla.

Ennen ohjelmoinnin aloittamista suunniteltiin, mitä tietoa robotilta halutaan ja miten
tieto välitetään palvelimelle. Robottivalmistajan dokumentaatiosta selvitettiin, mitä da-
taa robotilta on mahdollista kerätä. Päädyttiin keräämään robotilta servomoottorien
nopeutta ja vääntömomenttia ja välittämään nämä tiedot TCP-yhteydellä palvelimelle,
jossa se tallennetaan XML-tiedostoon.

MotoPlus IDE -ohjelmalla ohjelmoidaan C-kielinen ohjelma, joka ladataan robotille.
Tämän ohjelman tehtävä on kuunnella palvelimelta tulevaa pyyntöä ennalta määritel-
tyyn TCP-porttiin. Kun pyyntö vastaanotetaan, se prosessoidaan ja palautetaan robotil-
ta saadut arvot. Tämän jälkeen ohjelma palautuu lepotilaan ja odottaa uutta pyyntöä
palvelimelta.

Palvelimelle ohjelmoidaan Python-ohjelmointikielellä ohjelma, jota käytetään Window-
sin komentokehoteen avulla. Ohjelma ottaa yhteyden robottiin ja pyytää robotilta da-
taa. Kun data on vastaanotettu, palvelimen ohjelma käsittelee sen ja tallentaa datan
XML-tiedostoon. Tämän jälkeen palvelimen ohjelma käynnistää ajastimen ja odottaa
uuden pyynnön lähettämistä.

2 TAUSTAA

Opinnäytetyö tehtiin Loimaalaiselle Pemamek Oy:lle, joka on erikoistunut hitsaus- ja tuotantoautomaatioon. Yritys suunnittelee ja valmistaa automatisoituja hitsaus- ja tuotantojärjestelmiä, sekä työkappaleiden käsittelylaitteita.


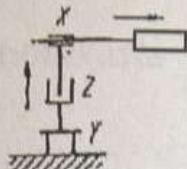
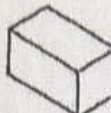

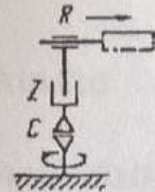


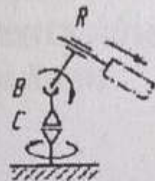

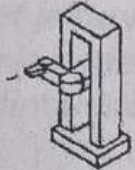
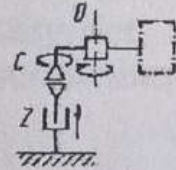
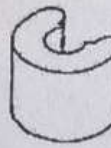

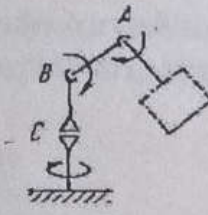


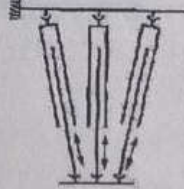
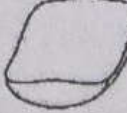
2.1 Teollisuusrobotit

ISO 8373 –standardin mukaan teollisuusrobotti on automaattisesti ohjattu, uudelleenohjelmoitavissa oleva, vähintään kolmenivelinen mekaaninen laite, joka on suunniteltu monenlaisten tehtävien suorittamiseen teollisuuden sovelluksissa. (Suomen Automaatioseura 2016)

Ensimmäiset teollisuusrobotit otettiin käyttöön 1960-luvun alussa, kun George Devolin ja Joseph Engelbergerin kehittämä Unimate-robotti toimitettiin General Motorsin tehtaalte Yhdysvalloissa. Robotti painoi 200 kg, ja sen liikekäskyt oli tallennettu rumpumuistille. Eurooppaan saapunut ensimmäinen teollisuusrobotti oli myös Unimate ja se otettiin käyttöön Ruotsissa. 1969 General Motors otti käyttöön ensimmäiset hitsausrobotit, joiden avulla se pystyi automatisoimaan yli 90 % auton korin hitsauksesta. 80-luvun alussa teollisuusrobotteja oli käytössä yli 60 000 ja vuonna 2003 niitä oli käytössä jo 800 000. Vuoteen 2018 mennessä arvellaan teollisuusrobotteja olevan käytössä 1,3 miljoonaa. (IFR – International Federation of Robotics 2016)

2.1.1 Robottityypit

Teollisuusrobotteja on valmistettu lukemattomien yritysten toimesta ja yleensä näillä on ollut valikoimissaan useita robottimalleja. Patenttien vuoksi robottien rakenteita on jouduttu erilaistamaan joten tästä syystä erilaisia teollisuusrobotteja on suunniteltu useita tuhansia. Osa näistä malleista on määritelty ISO 8373 –standardissa. Siinä yleisimmät robottimallit erotellaan niiden mekaanisen rakenteen mukaan. Kuvassa 1 on esitetty yleisimpien robottityyppien rakenteet, kinemaattiset kaaviot ja työalueet.

Nimitys pääakseleiden mukaan	Rakenne	Kinemaattinen kaavio	Työalue
Suorakulmainen robotti			
Sylinterirobotti			
Napa-koordinaatistirobotti			
Scara-robotti			
Kiertyvänivelinen robotti			
Rinnakkaisrakenteinen robotti			

Kuva 1. Yleisimmät robottirakenteet (ISO 8373).

Yleisesti nämä robotit koostuvat tukivarsista ja niiden välissä olevista nivelistä. Nivel voi olla joko tietyn suoran suuntainen tai suoran ympäri. Näitä niveliä kutsutaan vapausasteiksi. Yleensä yhtä vapausastetta kohti on yksi moottori tai sylinteri, joka mahdollistaa tukivarren liikkeen. Monikäyttöisimpiä teollisuusrobotteja ovat sellaiset, joiden työkalun saa mihin tahansa asentoon ja paikkaan sen työalueella. Tähän vaaditaan vähintään kuusi vapausastetta, joista ainakin kolmen tulee olla kiertyvänivelisiä.

Teollisuusrobottien liikkeet perustuvat suurimmaksi osaksi koordinaatistoihin ja annettuihin pisteisiin robotin koordinaatistoissa. Pisteiden määrittäminen koordinaatistoihin voidaan hoitaa monella tavalla. Maailmakoordinaatisto on robotin työskentely-ympäristöön sidottu ulkopuolinen koordinaatisto. Robotin peruskoordinaatisto on sen jalustaan sidottu ja sen avulla pisteet voidaan määrittää robotin työalueella. Työkalukoordinaatisto sidotaan työkalumäärittämisellä haluttuun kohtaan robotin työkalulaipasta. Tämän avulla esimerkiksi hitsauksessa voidaan työkalua siirtää lineaarisesti haluttuun suuntaan annetulla nopeudella.

2.1.2 Yaskawa Motoman

Motoman on Yaskawa Electric Corporationin robotiikkayksikkö, joka valmistaa samanlaisia teollisuusrobotteja eri teollisuuden aloille. Valikoimasta löytyy esimerkiksi hitsaukseen, lavaukseen, kappaleenkäsittelyyn ja puhdistila tarkoitukseen robottijärjestelmiä. Motoman on yksi maailman johtavia robotiikkavalmistajia, ja sen tuotteita on ympäri maailmaa yli 300 000. Yhtiöllä on toimipiste myös Suomessa, jossa työskentelee 35 henkilöä (Yaskawa Finland Oy 2016)

2.1.3 MotoPlus

MotoPlus on Motomanin kehittämä ohjelmointityökalu, jolla pystytään ohjelmoimaan MotoPlus sovelluksia teollisuusrobotille. Ohjelmointityökalu sisältää kattavan valikoiman APEja, joiden avulla robotin toiminnallisuutta voidaan lisätä.

Järjestelmän seurantaan tarkoitettuja APEja löytyy monia. Näiden API:n avulla pystytään esimerkiksi pyytämään robotilta sen hetkinen paikka eri koordinaatistoissa. Näiden pisteiden kerääminen mahdollistaa esimerkiksi robotin liikeratojen virtuaalisesti

reaaliajassa. Tästä on hyötyä tilanteessa jossa robotti työskentelee suljetussa tilassa johon ei ole suoraa näköyhteyttä.

APIen avulla voidaan myös tehdä ohjelmia, jotka kontrolloivat robottia. Ohjelmat voidaan esimerkiksi määrittää vastaanottamaan komentoa palvelimelta, jonka jälkeen se antaa robotille signaalin esimerkiksi pysähtyä tai käynnistämään hitsausohjelman. Käytännössä robottia pystytään hallitsemaan etäyhteydellä. Robotin läheisyydessä tulisi kuitenkin olla operaattori mahdollisten törmäysten tai hätätilanteiden varalta.

2.2 Esineiden internet

Internet of Things (IoT, esineiden internet) on terminä yleistynyt nopeasti. Tällä esineiden internetillä tarkoitetaan nykypäivän laitteita, jotka ovat yhteydessä Internet-verkkoon ja tätä kautta niitä voidaan mitata, ohjata ja sensoroida verkon yli. Esineiden internet on muodostunut digitalisaation ja Internet-verkon kasvamisen seurauksena. Tavalliselle käyttäjälle konkreettisimmin IoT tulee esille esimerkiksi älypuhelimissa ja aktiivisuusrannekeissa. Näiden kuluttajille suunnattujen älylaitteiden tarkoituksena on parantaa käyttäjäkokemusta ja tehdä siitä entistä yksilöllisempää.

Esineiden Internetiin liittyy vahvasti myös teollisuuden versio samasta käsitteestä, teollinen internet. Kansainvälisen ICT-alan tutkimus- ja konsultointiyritys Gartnerin määritelmän mukaan Teollisessa Internetissä on kyse fyysisistä laitteista, jotka aistivat ympäristöään ja pystyvät viestimään tai toimimaan aistimansa perusteella. Jotta tämä olisi mahdollista, pitää laitteissa tietysti olla aistimiseen tarkoitettuja antureita ja ohjelmistojia. Lisäksi laitteessa on oltava tietoliikenneyhteys, jotta tiedonsiirto datan varastoinniseksi on mahdollista. Tätä dataa jalostamalla voidaan ennakoida ja automatisoida työvaiheita ja tarkkailla esimerkiksi laitteiden kuntoa. Teollisen internetin tuottama lisäarvo perustuu kerättyyn tietoon ja sen hyödyntämiseen. Mahdollisuudet toiminnan kehittämiseen muodostuu siitä, miten hyvin kerätty tieto saadaan jalostettua hyödynnettävään muotoon ja yrityksen toiminnan tueksi. (Elisa 2015)

3 ROBOTIOHJELMA

3.1 Ohjelman toimintaperiaate

Robottiohjelmaa lähdettiin suunnittelemaan mahdollisimman yksinkertaiseksi ja helposti laajennettavaksi. Ohjelma päädyttiin rakentamaan niin, että ohjelma käynnistettäessä avaa TCP-portin ja jää kuuntelemaan porttiin tulevaa pyyntöä kerätä dataa ja välittää se eteenpäin. Ohjelman huomatessa porttiin tulevan pyynnön se vastaanottaa pyynnön ja tarkistaa mitä dataa pyydetään. Tarkistettuaan datan ohjelma kutsuu oikeita funktioita, jotka keräävät datan ja tallentavat sen muistiin lähetystä varten. Jokaiselle kerättävälle datalle on ohjelmaan luotu oma funktionsa ja niitä on helppo lisätä myöhemmin. Robotilta kerätyt tiedot ohjelma lähettää serverille, jossa tieto käsitellään. Tämän jälkeen robotin ohjelma palaa kuuntelemaan seuraavaa pyyntöä kerätä dataa.

```
bytesRecv = mpRecv(acceptHandle, rBuff, BUFF_MAX, 0);

printf("Data recieved %d\n", bytesRecv);

if (bytesRecv < 0)
    break;

msg = (sensor_msg*)rBuff;

//haetaan sensoridata
if (msg->cmd & GET_TORQUE)
    bytes += GetTorque(sBuff + bytes, msg->ctrl_grp);

if (msg->cmd & GET_FB_SPEED)
    bytes += GetFBSpeed(sBuff + bytes, msg->ctrl_grp);

//lähetetään tieto
bytesSend = mpSend(acceptHandle, sBuff, bytes, 0);

printf("Data sent %d\n", bytesSend);
```

Kuva 2. Ohjelman osa, jossa vastaanotetaan pyyntö, haetaan data ja lähetetään data palvelimelle.

Robottiohjelma on kirjoitettu C-kielellä ja sen kääntämiseen on käytetty Motomanin omaa MotoPlus IDE-sovellusta, jolla ohjelmointi on myös toteutettu. Käännetty ohjelma on ladattu robotille, jonka jälkeen se on käyttövalmis. Ohjelman etenemistä voi seurata avaamalla robotille Telnet-yhteyden, johon ohjelmakoodiin sijoitetut tulostuslausekkeet tulostavat ohjelman eri vaiheissa seuraamista helpottavia tekstejä.

3.2 Vääntömomentin palauttava funktio

GetTorque-funktio on robotin servomoottorien vääntömomentin seuraamiseen tarkoitettu funktio. Funktiolle annetaan parametrina palautettavan akseliryhmän arvo, jonka jälkeen funktio tallentaa tiedon ja palauttaa sen pääohjelmaan. Mikäli väännön haussa on tapahtunut ongelma, funktio palauttaa arvon -1. Funktiolla on mahdollista palauttaa Robotin, ulkoisten akselien ja esimerkiksi käsittelypöytien servomoottorien vääntö. palautettava arvo ilmoitetaan muodossa 0,01 % maksimiväännöstä.

```
int GetTorque(char* ret, char grp)
{
    LONG nRet;
    MP_CTRL_GRP_SEND_DATA torque_send_data;
    MP_TORQUE_RSP_DATA torque_recv_data;
    torque_send_data.sCtrlGrp = grp;

    nRet = mpGetTorque(&torque_send_data, &torque_recv_data);

    printf("get torque %d\n",nRet);

    if (nRet == OK)
    {
        *(int*)ret = TYPE_TORQUE;
        memcpy(ret+4, &torque_recv_data, sizeof(MP_TORQUE_RSP_DATA));
        return sizeof(MP_TORQUE_RSP_DATA)+4;
    }
    else
    {
        memset(ret, 0xff ,sizeof(MP_TORQUE_RSP_DATA)+4);
        return 0;
    }
}
```

Kuva 3. Servomoottorien väännön tallentamiseen tarkoitettu funktio.

3.3 Nopeuden palauttava funktio

GetFBSpeed-funktio palauttaa servomoottorien todellisen nopeuden sillä hetkellä. Funktiolle annetaan parametrina akseliryhmän arvo, jonka nopeudet halutaan palauttaa. Mikäli funktio ei saa noudettua servomoottorin nopeutta, se palauttaa arvon -1. Funktiolla voidaan palauttaa robotin, ulkoisten akselien ja esimerkiksi käsittelypöytien servomoottorien nopeuksia. Palautettava arvo ilmoitetaan pulsseina sekunnissa.

```
int GetFBSpeed(char* ret, char grp)
{
    LONG nRet;
    MP_CTRL_GRP_SEND_DATA speed_send_data;
    MP_FB_SPEED_RSP_DATA speed_recv_data;
    speed_send_data.sCtrlGrp = grp;

    nRet = mpGetFBSpeed(&speed_send_data, &speed_recv_data);

    printf("get speed %d\n",nRet);

    if (nRet == OK)
    {
        *(int*)ret = TYPE_FB_SPEED;
        memcpy(ret+4, &speed_recv_data, sizeof(MP_FB_SPEED_RSP_DATA));
        return sizeof(MP_FB_SPEED_RSP_DATA)+4;
    }
    else
    {
        memset(ret, 0xff ,sizeof(MP_FB_SPEED_RSP_DATA)+4);
        return 0;
    }
}
```

Kuva 4. Servomoottorien nopeuden seurantaan tarkoitettu funktio.

4 PALVELIMEN OHJELMA

Palvelimen ohjelmaa lähdettiin suunnittelemaan pohtimalla mitä tietoja haluamme pyytää robotilta. Ohjelman alkuun luotiin tarvittavat määritelmät vastaamaan robottiohjelmassa vastaavia määrittelyjä esimerkiksi IP-osoitteen ja TCP-portin osalta. Tämän jälkeen ohjelmassa luotiin viestin rakentamiseen tarvittava osa. Viesti sisältää tiedon minkälaista tietoa halutaan ja miltä akseliryhmiltä kyseiset tiedot halutaan. Tällä hetkellä muutokset pyydettäviin tietoihin tehdään suoraan ohjelmakoodiin. Ohjelman toteuttamiseen valittiin kieleksi python sen helppokäyttöisyyden vuoksi. Ohjelman toteuttamiseen valittiin kieleksi python sen helppokäyttöisyyden vuoksi. Ohjelma käynnistetään Windowsin komentokehotteen kautta ja ohjelmakoodiin on laitettu tulostuslausekkeita ohjelman etenemisen seuraamisen helpottamiseksi.

```
from socket import *
import time
from struct import *
import csv
import xml.etree.ElementTree as ET

ROBOT_IP = '192.168.2.120'
ROBOT_PORT = 11000
BUFF_SIZE = 1024
RSP_DATA_SIZE = 36

TYPE_TORQUE = 1
TYPE_FB_SPEED = 2

#command constants
GET_TORQUE = 1
GET_SPEED = 2

CMD = GET_TORQUE + GET_SPEED
#robot 1-8=0-7, base 1-8=8-15, station 1-24=16-39
CTRL_GRP = 0
MSG = pack('BB',CMD,CTRL_GRP)
```

Kuva 5. Palvelimen ohjelman määritelmiä.

Tämän jälkeen ohjelma lähettää viestin robotille ja odottaa tältä vastausta. Mikäli ohjelma saa robotilta vastauksen ja vastaanottaa viestin, alkaa ohjelma käsitellä vastaanotettua viestiä. Ohjelma purkaa datan ja kirjoittaa datan XML-tiedostoon. Data jaotellaan XML-tiedostossa helposti luettavaan muotoon. Viimeiseksi ohjelma siirtyy lepotiilaan ja odottaa uudelleenkäynnistymistä. Ajastimeen voidaan määrittää kuinka nopeasti robotilta halutaan uudelleen pyytää dataa.

4.1 Yhteyden luominen

Ohjelman alussa on määritelty tarvittavat osoite ja TCP-portti johon ohjelma ottaa yhteyttä. Robotilla on määritelty vastaava TCP-portti, jota kuunnellaan, jotta kommunikaatio voi onnistua. Kun yhteys on avattu, lähetetään robotille viesti. Viesti sisältää tiedon, mitä tietoa robotilta halutaan ja miltä akseliryhmältä se halutaan. Vastaanotettu data tallennetaan ja yhteys suljetaan.

```
s = socket(AF_INET, SOCK_STREAM)
print 'connecting... ',ROBOT_IP,ROBOT_PORT
s.connect((ROBOT_IP,ROBOT_PORT))
print 'Sending message...'
s.send(MSG)
print 'Receiving data...'
data = s.recv(BUFF_SIZE)
print 'Closing socket...',len(data)
s.close()
```

Kuva 6. Kommunikaatio robotin kanssa.

4.2 Datan purku

Data vastaanotetaan robotilta bittijonona ja se pitää purkaa ennen kuin sitä voidaan hyödyntää. Bittijonosta tallennetaan ensin datan tyyppi ja sen jälkeen jokaisen servomoottorin arvo erikseen omaan muuttujaansa.

```
while i < len(data)/RSP_DATA_SIZE:
    print i, len(data)
    type,data0,data1,data2,data3,data4,data5,data6,data7 = unpack(
        'LLLLLLLLL',data[i*RSP_DATA_SIZE : ((i+1) * (RSP_DATA_SIZE))])
    print type,data0,data1,data2,data3,data4,data5,data6,data7
```

Kuva 7. Datan purku

4.3 Datan Kirjoitus XML-tiedostoon

Kun data on purettu, ohjelma luo XML-tiedoston, johon se kirjoittaa jokaisen servomoottorin arvon sille varatulle paikalle. Kirjoitus tiedostoon tapahtuu samassa silmukassa kuin datan purku, jotta XML-tiedostoon saadaan kirjoitettua datalle oikea tyyppi oikeille arvoille. Vastaanotettu data on kokonaislukuna, joten data pitää vielä muuttaa String- eli merkkijono muotoon ennen kirjoittamista.

```
#XML Printing
if type == 1:
    TYPE = ET.SubElement(document, 'Servo_Torque')
elif type == 2:
    TYPE = ET.SubElement(document, 'Servo_Speed')
    DATA0 = ET.SubElement(TYPE, 'S_Axis')
    dataString0 = str(data0)
    DATA0.text = dataString0
    DATA1 = ET.SubElement(TYPE, 'L_Axis')
    dataString1 = str(data1)
    DATA1.text = dataString1
    DATA2 = ET.SubElement(TYPE, 'U_Axis')
    dataString2 = str(data2)
    DATA2.text = dataString2
    DATA3 = ET.SubElement(TYPE, 'R_Axis')
    dataString3 = str(data3)
    DATA3.text = dataString3
    DATA4 = ET.SubElement(TYPE, 'B_Axis')
    dataString4 = str(data4)
    DATA4.text = dataString4
    DATA5 = ET.SubElement(TYPE, 'T_Axis')
    dataString5 = str(data5)
    DATA5.text = dataString5
    DATA6 = ET.SubElement(TYPE, 'E_Axis')
    dataString6 = str(data6)
    DATA6.text = dataString6
    DATA7 = ET.SubElement(TYPE, "Eight_Axis")
    dataString7 = str(data7)
    DATA7.text = dataString7
    tree = ET.ElementTree(document)
    tree.write("data.xml")
```

Kuva 8. XML-tiedostoon kirjoitus.

Lopputuloksena ohjelma luo seuraavanlaisen XML-tiedoston.

```
<?xml version="1.0"?>
- <Sensor_data>
  - <Servo_Torque>
    <S_Axis>4294967288</S_Axis>
    <L_Axis>56</L_Axis>
    <U_Axis>4294967217</U_Axis>
    <R_Axis>3</R_Axis>
    <B_Axis>7</B_Axis>
    <T_Axis>4</T_Axis>
    <E_Axis>0</E_Axis>
    <Eight_Axis>0</Eight_Axis>
  </Servo_Torque>
  - <Servo_Speed>
    <S_Axis>2995</S_Axis>
    <L_Axis>4294961556</L_Axis>
    <U_Axis>249</U_Axis>
    <R_Axis>6364</R_Axis>
    <B_Axis>4294960557</B_Axis>
    <T_Axis>4294963428</T_Axis>
    <E_Axis>0</E_Axis>
    <Eight_Axis>0</Eight_Axis>
  </Servo_Speed>
</Sensor_data>
```

Kuva 9. Robotilta pyydetty data XML muodossa.

5 YHTEENVETO

Työn tarkoituksena oli kehittää järjestelmä, jonka avulla teollisuusrobotilta voidaan kerätä dataa ja välittää se ulkoiselle palvelimelle. Työn testausvaiheessa ohjelmat toimivat oletetulla tavalla ja robotilta saatiin tallennettua dataa XML-tiedostoon.

Jotta järjestelmästä saataisiin paras mahdollinen hyöty, tulisi sitä kehittää vielä hieman. Tällä hetkellä eri kerroilla kerätty data ei ole keskenään vertailukelpoista, sillä ne saatavat olla kerätty eri kohdassa robotin liikettä. Jotta kerättyä dataa voidaan verrata tehokkaasti, tulee sen olla kerätty täsmälleen samassa kohdassa samaa liikerataa, josta edellinen data on kerätty. Tämän avulla mahdolliset muutokset servomootoreissa on helppo huomata. Esimerkiksi jos vääntömomentti on yhden moottorin kohdalla kasvanut mittausten välillä, voi se olla merkki mahdollisesta rikkoutumisesta.

Järjestelmää on mahdollista jatkokehittää monella tavalla. Jotta dataa voisi verrata keskenään, tulisi järjestelmä integroida osaksi jotain sellaista robottityötä, jonka avulla data saataisiin kerättyä aina samasta kohdasta robotin liikettä. Lisäksi MotoPlus-ohjelmointirajapintojen avulla työtä on mahdollista laajentaa moneen eri tarkoitukseen.

LÄHTEET

Elisa 2015. Yritysjohdon opas IoT:n ja teollisen internetin hyödyntämiseen. Viitattu 30.4.2016 http://quva.fi/site/attachments/yritysjohdon_opas_iot_ja_teollisen_internetin_hyodyntamiseen.pdf

International Federation of Robotics 2016. Robottiikan kansainvälinen kattojärjestö. Viitattu 30.4.2016 http://www.ifr.org/fileadmin/user_upload/downloads/forms___info/History_of_Industrial_Robots_online_brochure_by_IFR_2012.pdf

Motoman Finland Oy 2016. robotisoinnin ja automaattisten tuotantojärjestelmien toimittaja. Viitattu 30.4.2016 www.motoman.fi > yritys

Suomen Automaatioseura ry 2016. Yleishyödyllinen yhdistys. Viitattu 30.4.2016 <https://www.automaatioseura.fi/index/tiedostot/Robotit.doc>

Vento V. 2015. IoT kansankielellä. Viitattu 30.4.2016 <http://blogit.sonera.fi/2015/10/iot-kansankielella-ville-vento/>

Robottihjelman lähdekoodi

```
#include "motoPlus.h"

// for API & FUNCTIONS
void moto_plus0_task(void);
void ap_TCP_Sserver(ULONG portNo);
#define PORT      11000
#define BUFF_MAX   1023

#define TYPE_TORQUE    1
#define TYPE_FB_SPEED  2

//sensor message komennot
#define GET_TORQUE    1
#define GET_FB_SPEED  (1<<1)

extern int GetTorque(char* ret, char grp);
extern int GetFBSpeed(char* ret, char grp);

typedef struct _sensor_msg {
    char cmd;
    char ctrl_grp;
}
    sensor_msg;

void moto_plus0_task(void)
{
    puts("Activate moto_plus0_task!");

    ap_TCP_Sserver(PORT);

    mpSuspendSelf;
}

void ap_TCP_Sserver(ULONG portNo)
{
    int      sockHandle;
    struct   sockaddr_in      serverSockAddr;
    int      rc;

    printf("Simple TCP server\n");

    sockHandle = mpSocket(AF_INET, SOCK_STREAM, 0);
    if (sockHandle < 0)
        return;

    memset(&serverSockAddr, 0, sizeof(serverSockAddr));
    serverSockAddr.sin_family = AF_INET;
    serverSockAddr.sin_addr.s_addr = INADDR_ANY;
    serverSockAddr.sin_port = mpHtons(portNo);

    rc = mpBind(sockHandle, (struct sockaddr *)&serverSockAddr,
sizeof(serverSockAddr));
    if (rc < 0)
        goto closeSockHandle;
```

```

rc = mpListen(sockHandle, SOMAXCONN);
if (rc < 0)
    goto closeSockHandle;

while (1)
{
    int      acceptHandle;
    struct   sockaddr_in  clientSockAddr;
    int      sizeofSockAddr;

    memset(&clientSockAddr, 0, sizeof(clientSockAddr));
    sizeofSockAddr = sizeof(clientSockAddr);

    acceptHandle = mpAccept(sockHandle, (struct sockaddr
*)&clientSockAddr, &sizeofSockAddr);

    if (acceptHandle < 0)
        break;

        printf("client connected\n");

//    while( 1 )
    {
        int      bytesRecv = 0;
        int      bytesSend = 0;
        char      sBuff[BUFF_MAX + 1];
                                char      rBuff[BUFF_MAX + 1];

        sensor_msg* msg;
        int      bytes = 0;

        memset(rBuff, 0, sizeof(rBuff));
        memset(sBuff, 0, sizeof(sBuff));

        bytesRecv = mpRecv(acceptHandle, rBuff, BUFF_MAX, 0);

        printf("Data recieved %d\n",
bytesRecv);

        if (bytesRecv < 0)
            break;

        msg = (sensor_msg*)rBuff;

        //haetaan sensoridata
        if (msg->cmd & GET_TORQUE)
            bytes += GetTorque(sBuff +
bytes, msg->ctrl_grp);

        if (msg->cmd & GET_FB_SPEED)
            bytes += GetFBSpeed(sBuff + bytes, msg->ctrl_grp);

        //lähetetään tieto
        bytesSend = mpSend(acceptHandle, sBuff, bytes, 0);

        printf("Data sent %d\n", bytesSend);

    }
    mpClose(acceptHandle);
}

```

```

        printf("Socket closed\n");
    }
closeSockHandle:
    mpClose(sockHandle);

    return;
}

int GetFBSpeed(char* ret, char grp)
{
    LONG nRet;
    MP_CTRL_GRP_SEND_DATA speed_send_data;
    MP_FB_SPEED_RSP_DATA speed_recv_data;
    speed_send_data.sCtrlGrp = grp;

    nRet = mpGetFBSpeed(&speed_send_data, &speed_recv_data);

    printf("get speed %d\n", nRet);

    if (nRet == OK)
    {
        *(int*)ret = TYPE_FB_SPEED;
        memcpy(ret+4, &speed_recv_data,
sizeof(MP_FB_SPEED_RSP_DATA));
        return sizeof(MP_FB_SPEED_RSP_DATA)+4;
    }
    else
    {
        memset(ret, 0xff
, sizeof(MP_FB_SPEED_RSP_DATA)+4);
        return 0;
    }
}

int GetTorque(char* ret, char grp)
{
    LONG nRet;
    MP_CTRL_GRP_SEND_DATA torque_send_data;
    MP_TORQUE_RSP_DATA torque_recv_data;
    torque_send_data.sCtrlGrp = grp;

    nRet = mpGetTorque(&torque_send_data, &torque_recv_data);

    printf("get torque %d\n", nRet);

    if (nRet == OK)
    {
        *(int*)ret = TYPE_TORQUE;
        memcpy(ret+4, &torque_recv_data,
sizeof(MP_TORQUE_RSP_DATA));
        return sizeof(MP_TORQUE_RSP_DATA)+4;
    }
}

```



```
else
{
    memset(ret, 0xff ,sizeof(MP_TORQUE_RSP_DATA)+4);
    return 0;
}
}
```

Palvelimen ohjelman lähdekoodi

```
from socket import *

import time

from struct import *

import csv

import xml.etree.ElementTree as ET


ROBOT_IP = '192.168.2.120'

ROBOT_PORT = 11000

BUFF_SIZE = 1024

RSP_DATA_SIZE = 36


TYPE_TORQUE = 1

TYPE_FB_SPEED = 2


#command constants

GET_TORQUE = 1

GET_SPEED = 2


CMD = GET_TORQUE + GET_SPEED


#robot 1-8=0-7, base 1-8=8-15, station 1-24=16-39

CTRL_GRP = 0
```

```
MSG = pack('BB',CMD,CTRL_GRP)
```

```
while True:
```

```
    s = socket(AF_INET, SOCK_STREAM)
```

```
    print 'connecting... ',ROBOT_IP,ROBOT_PORT
```

```
    s.connect((ROBOT_IP,ROBOT_PORT))
```

```
    print 'Sending message...'
```

```
    s.send(MSG)
```

```
    print 'Recieving data...'
```

```
    data = s.recv(BUFF_SIZE)
```

```
    print 'Closing socket...',len(data)
```

```
    s.close()
```

```
    i = 0
```

```
    document = ET.Element("Sensor_data")
```

```
    with open('data.csv', 'wb') as csvfile:
```

```
        writer = csv.writer(csvfile, delimiter=',')
```

```
        while i < len(data)/RSP_DATA_SIZE:
```

```
            print i, len(data)
```

```
            type,data0,data1,data2,data3,data4,data5,data6,data7 = un-
```

```
pack('LLLLLLLLL',data[i*RSP_DATA_SIZE : ((i+1) * (RSP_DATA_SIZE))])
```

```
            print type,data0,data1,data2,data3,data4,data5,data6,data7
```

```
            #CSV Printing
```

```
            if type == 1:
```

```
                writer.writerow(['Servo Torque'])
```

```
elif type == 2:

    writer.writerow(['Servo Speed'])

writer.writerow([data0,data1,data2,data3,data4,data5,data6,data7])

#XML Printing

if type == 1:

    TYPE = ET.SubElement(document, 'Servo_Torque')

elif type == 2:

    TYPE = ET.SubElement(document, 'Servo_Speed')

    DATA0 = ET.SubElement(TYPE, 'S_Axis')

    dataString0 = str(data0)

    DATA0.text = dataString0

    DATA1 = ET.SubElement(TYPE, 'L_Axis')

    dataString1 = str(data1)

    DATA1.text = dataString1

    DATA2 = ET.SubElement(TYPE, 'U_Axis')

    dataString2 = str(data2)

    DATA2.text = dataString2

    DATA3 = ET.SubElement(TYPE, 'R_Axis')

    dataString3 = str(data3)

    DATA3.text = dataString3

    DATA4 = ET.SubElement(TYPE, 'B_Axis')

    dataString4 = str(data4)

    DATA4.text =dataString4
```

```
DATA5 = ET.SubElement(TYPE, 'T_Axis')

dataString5 = str(data5)

DATA5.text = dataString5

DATA6 = ET.SubElement(TYPE, 'E_Axis')

dataString6 = str(data6)

DATA6.text = dataString6

DATA7 = ET.SubElement(TYPE, "Eight_Axis")

dataString7 = str(data7)

DATA7.text = dataString7

tree = ET.ElementTree(document)

tree.write("data.xml")

i+=1


print 'Sleeping... 10 sec'

time.sleep(10)
```

Palvelimen ohjelman luoma XML-tiedosto

```
<?xml version="1.0"?>
- <Sensor_data>
  - <Servo_Torque>
    <S_Axis>4294967288</S_Axis>
    <L_Axis>56</L_Axis>
    <U_Axis>4294967217</U_Axis>
    <R_Axis>3</R_Axis>
    <B_Axis>7</B_Axis>
    <T_Axis>4</T_Axis>
    <E_Axis>0</E_Axis>
    <Eight_Axis>0</Eight_Axis>
  </Servo_Torque>
  - <Servo_Speed>
    <S_Axis>2995</S_Axis>
    <L_Axis>4294961556</L_Axis>
    <U_Axis>249</U_Axis>
    <R_Axis>6364</R_Axis>
    <B_Axis>4294960557</B_Axis>
    <T_Axis>4294963428</T_Axis>
    <E_Axis>0</E_Axis>
    <Eight_Axis>0</Eight_Axis>
  </Servo_Speed>
</Sensor_data>
```