TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Morten Lillepruun 143058IAPB

TARKVARA KEEVITUSROBOTI LIIKUMISE PIIRANGUTE ANALÜÜSIMISEKS 2D GEOMEETRILISTE KUJUNDITE PÕHJAL

Bakalaureusetöö

Juhendaja: Ermo Täks
PhD
Alexander Gadalov
MEng

Tallinn 2016

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Morten Lillepruun

14.05.2017

Annotatsioon

Antud lõputöö raames on loodud ASG Robotics keevitusrobotile tarkvara prototüüp, mis analüüsib roboti liikumise piiranguid 2D geomeetriliste kujundite põhjal. Töös annab autor ülevaate tööstusrobotite olemusest ja liigitusest ning tutvustab ka robotit, millele prototüüp luuakse. Loodav tarkvara koosneb robotivälisest klientrakendusest ja robotisisesest programmist, mis suhtlevad omavahel. Töös kirjeldatakse tarkvara ja selle arendust täpsemalt. Prototüüp on aluseks ASG Robotics-i poolt arendatavale tarkvarale, mida hakatakse kasutame tellimustööde piirangute ja võimalikkuse analüüsiks ning potentsiaalselt ka robotite automatiseerimiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 22 leheküljel, 6 peatükki ja 11 joonist.

Abstract

Analysis software for analyzing the limitations of robot movement according to 2D geometrical shapes

The outcome of this thesis is a software prototype for ASG Robotics arc welding robot, which analyzes the limitations of robot movement according to 2D geometrical shapes. The thesis will give an overview of industrial robots and introduce the ASG Robotics robot. The software created consists of a client application and a robot program, which communicate between each other. The software and the creation process is more closely described in the thesis. The prototype is the basis for the software developed by ASG Robotics, which will be used for analysis of arc welding project restrictions and potentially in the future for automating robots.

The thesis is in Estonian and contains 22 pages of text, 6 chapters and 11 figures.

Sisukord

Jooniste loetelu

1 Sissejuhatus

ASG Robotics on Euroopa masinehitus firma, kes soovib oma robotite jaoks välja töötada tarkvara, millega oleks võimalik analüüsida tellimustööde piiranguid ning võimalikkust. Lisaks loodab firma antud tarkvara kasutada ka robotite automatiseerimiseks.

Käesoleva bakalaureusetöö eesmärgiks on luua ASG Robotics keevitusrobotile tarkvara prototüüp, mis analüüsib roboti liikumise piiranguid 2D geomeetriliste kujundite põhjal. Analüüsi alusel on võimalik konkreetses ajahetkes määrata tööks vajalikud parameetrid (tööaluse pöörlemiskiirus, roboti asend ja kiirus), mis võimaldab roboti automatiseerida. Tarkvara koosneb robotivälisest programmist, mis arvutab ja saadab robotile koordinaate, ning robotisisesest programmist, mis võtab koordinaate vastu ning liigutab keevituspea vastavasse punkti. ASG Robotics soovib antud prototüübist välja arendada tarkvara, millega oleks võimalik analüüsida keevitustöö võimalikkust enne keevitusprotsessi alustamist.

Bakalaureusetöö teises peatükis tutvustab autor tööstusrobotite olemust ja liigitust ning kirjeldab täpsemalt töös kasutatavat robotit. Tarkvara ülesehitust ja kasutusjuhendit tutvustatakse töö kolmandas peatükis ning neljas peatükk kirjeldab prototüübi arendusprotsessi ja arenduseks kasutatud vahendeid. Kuues peatükk analüüsib loodud prototüüpi ning toob välja mõned selle puudujäägid.

2 Tööstusrobotite olemus ja liigitus

Robot tuleneb tšehhi keelsest sõnast *robota*, mis tähendab „tööd“. Esmakordselt kasutas seda sõna tšehhi kirjanik Karel Capeki oma 1920. aastal ilmunud näidendis „R.U.R“ (Rossum’s Universal Robots), kus see tähendab inimest [1]. Siit ka arusaam, et robotid on inimesele väga sarnased.

Tänapäeval mõeldakse roboti all automaati, mis asendab inimest tööprotsessis [2]. Peamiselt kasutatakse roboteid tootmisprotsessides, sest paljud tootmisse kuuluvad ülesanded on inimese jaoks liiga rutiinsed.

Robotite kasutamise kolm peamist põhjust on:

1. inimese jaoks ohtlik töökeskkond (nt radioaktiivsus)
2. inimese jaoks liiga rutiinne töö (nt toodete pakkimine)
3. inimese füüsilised võimed ja keskkonnataluvus pole töö tegemiseks piisavad (nt laevaosade keevitamine)

Tööstusrobotil on töö tegemiseks mehaaniline käsi ehk manipulaator. Manipulaatorit juhitakse juhtseadmega, mis põhineb mikroprotsessoritel. Roboteid saab kasutada paljude erinevate tööde jaoks. Robot võib töötada haaratsiga, mis võimaldab näiteks esemete ühest kohast teise tõstmist, või mõne muu tööriistaga, millega tehakse selleks ettenähtud tööd (keevitamine, värvimine jms).

Tööstusroboteid liigitatakse kahte kategooriasse:

- kohtrobotid (stationary robots)
- liikurrobotid (mobile robots)

Kohtrobot on statsionaarne, mis koosneb ühest või mitmest manipulaatorist ja programmjuhtimisseadmest. Vastavalt vajadusele võib robotit kinnitada põrandale, lakke, seinale või liikuvale masinale. Liikurrobot koosneb programmjuhitavast veokist, millel võib paikneda üks või mitu manipulaatorit. [3]

Kohtroboteid on võimalik ehituse järgi liigitada kaheks: jada- ja rööpkinemaatikaga robotid. Jadakinemaatikaga robotitel on kõik lülid üksteise suhtes jadamisi (Joonis 1), mis tõttu on nad väga sarnased inimkäele (käe jadamisi olevateks lülideks on õlavars ja küünarvars). Lülide arv ja asend üksteise suhtes määrab roboti tööruumi ulatuse ja võime tööd teha. Tavaliselt on manipulaatoritel neli kuni kuus lüli. Lülide arv määrab roboti liikuvusastme, mis on üks aspekt, mille järgi jadakinemaatikaga roboteid liigitatakse. Lisaks liigitatakse manipulaatoreid ajamite, juhtimispõhimõtte (kaugjuhitav või autonoomne) ja tehnoloogiliste tegevuste (mille jaoks kasutatakse) järgi.

Joonis 1. Jadakinemaatikaga manipulaator (Fanuc.eu)

Liikuvusastmete ehk koordinaatide arv (number of axes) võimaldab järgnevat liikumist:

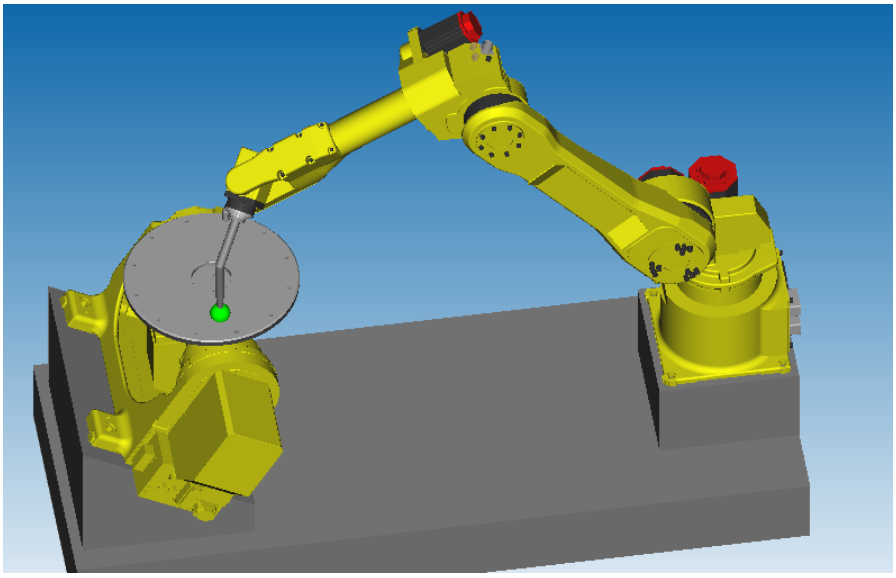
- 1 koordinaat – 1 sirgjooneline või 1 pöördliikumine
- 2 koordinaati – liikuda etteantud punkti tasapinnal
- 3 koordinaati – liikuda etteantud punkti ruumis
- 4 koordinaati – liikuda etteantud punkti ruumis ja pöörata eset ümber ühe telje
- 5 koordinaati – liikuda etteantud punkti ruumis ja pöörata eset ümber kahe telje
- 6 koordinaati – liikuda etteantud punkti ruumis ja pöörata eset suvalisse asendisse (täiuslik liikumine ruumis)

Üha rohkem kasutatakse rööpkinemaatikaga manipulaatoreid (Joonis 2). Sellised robotid on väga suure toimekiirusega, sest nende lülide ajamid asuvad kõik paigalseisval alusel, mis tõttu ei koormata liikuvaid osi. Rööpkinemaatikaga robotid sobivad hästi konveierilt väikese massiga toodete korjamiseks ja pakendamiseks [3].

Joonis 2. Rööpkinemaatikaga manipulaator (Robotshop.com)

2.1 ASG Robotics-i keevitusrobot

Käesolevas töös on loodud tarkavara ASG Robotics-i keevitusrobotile FANUC 7DA7 (Joonis 3), mida kasutatakse kaarkeevitusel (arc welding). FANUC[1] on Jaapanis paiknev automaatide tootja. Joonisel 3 olev pilt on küll tehtud roboti simulatsioonist, kuid päris robot on identne ning seda on võimalik näha Mektory-s.



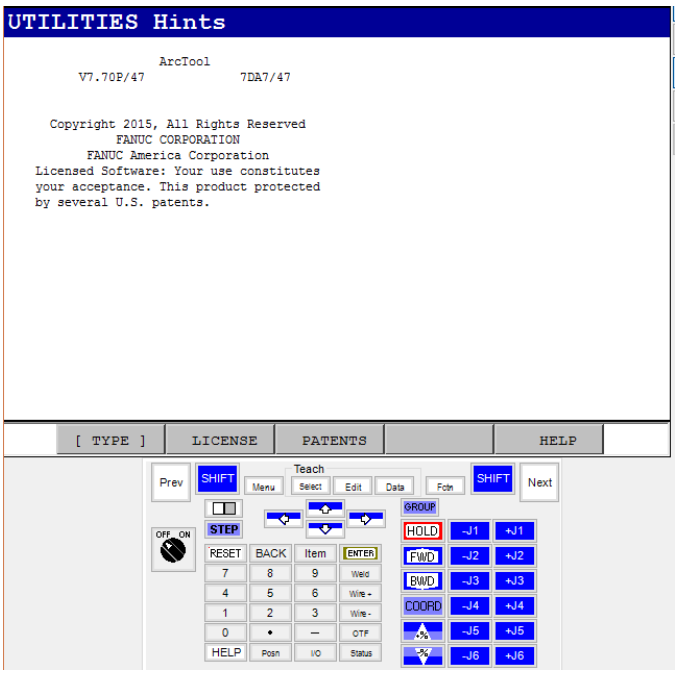
Joonis 3. FANUC 7DA7

Tegemist on jadakinemaatika robotiga, mille liikuvusaste on kuus. See tähendab, et robot on võimeline ruumis täielikult liikuma. Robotit juhitakse käsitsi õpetamispuldist (teach pendant) või juhtimisprogrammi abil. Programm tuleb kirjutada õpetamispuldis ning roboti programmeerimiskeeles. FANUC on loonud enda toodetud robotite tarbeks programeerimiskeele Karel.

Käesoleva töö otstarbeks on tehtud Mektory-s olevast keevitusrobotist Roboguide-i simulatsioon. Roboguide on FANUC-i robotisüsteemidele mõeldud arenduskeskkond, kus on võimalik animeerida virtuaalses 3D-keskkonnas roboti manipulaatori liikumist. Manipulaatorit saab arenduskeskkonnas liigutada nii hiire, virtuaal õpetamispuldi (Joonis 4) kui ka programmi abil. Virtuaalne õpetamispult töötab samamoodi kui reaalne kontroller, samuti tuleb juhtimisprogrammid samamoodi puldis kirjutada. Kõik,

[1] http://www.fanuc.eu/uk/en/who-we-are

mida on võimalik teha Roboguide-i simulatsioonis, on võimalik teha ka päris roboti peal.



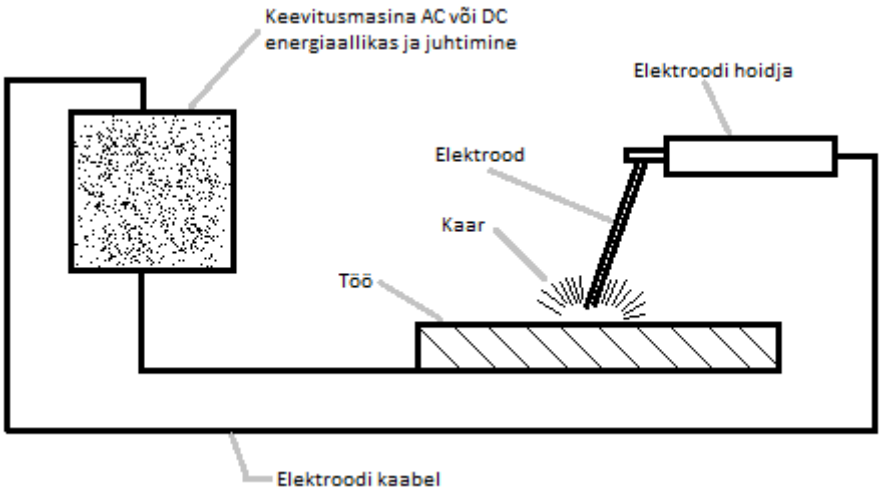
Joonis 4. Virtuaalne õpetamispult

3 Probleemi kirjeldus

Antud töö eesmärgiks on luua ASG Robotics keevitus robotile tarkvara, mis analüüsib roboti liikumise piiranguid ja erinevate 2D geomeetriliste kujundite põhjal. Analüüsi alusel on võimalik konkreetses ajahetkes määrata tööks vajalikud parameetrid, mille abil on võimalik robot iseseisvalt keevitama panna.

Praegu juhitakse keevitusrobotit manuaalselt, mis toob kaasa inimfaktorist tingitud probleeme. Tihti on keevitustööd ajaliselt väga pikad ja inimesele liialt väsitavad. Keevitamine peab olema sujuv ning järjepidev. Väsimus võib siinkohal liialt mõjutada töö kvaliteeti. Vahepeal lõpetada ei ole võimalik keevitusprotsessiga kaasnevate piirangute tõttu.

Kaarkeevitusel tekitatakse kõrgel kuumusel baasmetalli pinnale „kaar“, mille sees on elektroodi ning baasmetalli sulam, mis kivistub protsessi käigus. Metallid kõrgetel temperatuuridel reageerivad väga hästi õhus olevate elementidega (lämmastik ja hapnik). Tekkivate oksiididega kaovad ühenduse tugevus ja vastupidavus. Oksüdeerumise vältimiseks kasutatakse keevitusel kaitsvat gaasi.



Joonis 5. Kaarkeevitusprotsess

Keevitustöö peab toimuma järjepidevalt ning õigel kiirusel. Kui keevitamine vahepeal peatada, siis vedel sulam oksüdeerub ning ühenduse valmides ei ole see nii tugev.

Keevitatakse ühe korraga, ilma pausideta, nii on lõppprodukt kõige tugevam ja vastupidavam.

Väikesel kiirusel keevitamise tagajärjeks on lai ja kõrge keevisliide. Keevise kuju jääb ebaühtlaseks, sest sulam on kogunenud ning siis vajunud tekkinud „kraavi“. Selline keevituspraktika võib kaasa tuua nõrga liite ning räbu sisalduse keevises. [4]

Suurel kiirusel jääb ühendus õhukeseks ja nõrgaks. Keevituse hari on väljavenitatud ja kolmnurkse kujuga. Suurt kiirust saab kompenseerida suurema sulami vooga, kuid sellisel juhul jääb hari jätkuvalt väljavenitatuks. Mõislikum on ikkagi sobiv kiirus valida. [4]

Joonis 6. Keevitamine erinevatel kiirustel (Mig-welding.co.uk)

Robotil on piiratud tööala. Tööala on ruumiosa, kuhu manipulaatori otsas olev keevituspea reaalselt ulatub ning on võimeline tööd tegema. Tööala suurus oleneb robotkäe pikkusest, mida pikem on käsi, seda suurem on roboti haardeulatus. Lisaks tuleb arvestada tööruumis olevate esemetega, mis võivad piirata manipulaatori tegutsemist. Takistavaks objektiks võib olla ka keevitatav detail ise.

Robotkäel on ka ehitusest tulenevaid piiranguid. Nii nagu inimene ei saa oma kätt igatepidi liigutada ja painutada, ei ole seda võimeline tegema ka robotkäsi. See tuleneb käe lüliliselt ehitusest. Omavahel ühendatud lülid takistavad teineteise liikumist, mille tulemusel ei ole võimalik kätt igasse asendisse panna.

Eelpool loetletud piirangute põhjal saab sõnastada töö uurimisküsimuse: kuidas analüütiliselt tuvastada vabalt valitud kahemõõtmelise objekti piirjoonte parameetrid roboti optimaalse liikumistee määramiseks ja võimalike liikumispiirangute avastamiseks

4 Roboti liikumise arvutamiseks kasutatav metoodika

Keevituspea asukohta ruumis saab määrata koordinaatidega x, y, z. Lisaks neile parameetritele on veel oluline, mis asendis on tööriist, sest detailidele ei saa iga nurga alt ligi. Keevituspea asendimääramiseks on antud töö raames kasutatud kvaternioone.

Kvaternioonid on arvusüsteem, mis laiendab kompleksarve. Esimesena kirjeldas neid iiri matemaatik William Rowan Hamilton 1843. aastal [7]. Kvaternioonide üldkuju esitatakse valemiga

$$q = a + bi + cj + dk,$$

kus a, b, c, d on reaalarvud ning i, j, k on imaginaarühikud.

Kvaternioonidest tuletatavate valemitega 0 saab kirjeldada objekti pöörlemist kolmemõõtmelises ruumis ning seda on käesolevas töös kasutatud roboti asendi kalkuleerimiseks (Lisa 1).

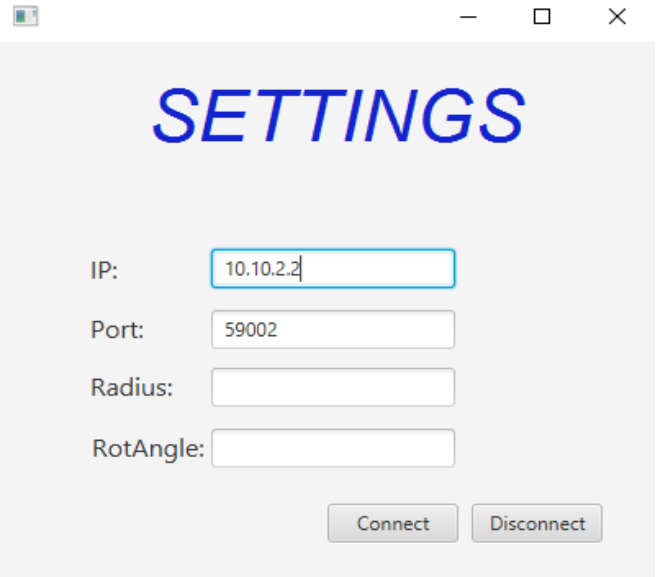
Kvaternioonid on inimesele raskemad kasutada, kui seda on näiteks Euleri nurgad, küll aga on neid mõistlik kasutada masinarvutamise korral, sest kvaternioonid on kompaktsemad ning efektiivsemad.

5 Keevitusroboti tarkvara tutvustus

Antud töö raames on loodud keevitusrobotile tarkvara, millega saab analüüsida roboti liikumise piiranguid 2D kujundite põhjal. Tarkvara koosneb robotivälisest rakendusest, mis saadab masinale koordinaate, ja robotisisesest programmist, mis võtab koordinaate vastu ning liigutab keevituspea õigesse punkti. Välise rakenduse ja roboti programmi omavaheline suhtlus toimub *socket messaging*-i abil.

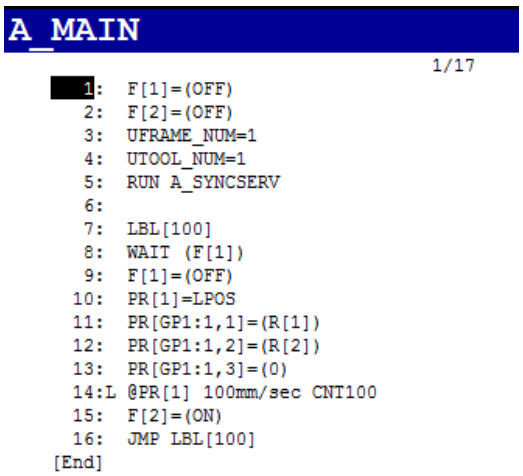
Socket messaging baseerub TCP/IP protokollil. Andmeedastuses on kaks osapoolt: klient (client) ja server. Käesolevas töös on serveriks robot ja kliendiks arvutis olev rakendus. Klient saadab serveri IP aadressile soovi ühenduse loomiseks koos pordi numbriga, mille läbi andmesidet soovitakse luua. Serveri „nõustumisel“ luuakse ühendus kliendiga. Kogu andmeside toimub läbi määratud pordi numbri, s.t kõik sõnumid saadetakse sinna porti ning neid ka „kuulatakse“ sealt.

Tarkvara käivitamisel tuleb luua vastav ühendus arvuti ja roboti vahel. Selle jaoks on rakendusele loodud lihtne kasutajaliides (Joonis 5), kus on vaja sisestada ühenduse jaoks vajalikud parameetrid. „IP“ väljale tuleb sisestada roboti IP-aadress ning pordiks tuleb määrata andmesideks kasutatav pordi number. Ülejäänud väljad on kujundi parameetrite jaoks. Loodud prototüüp on esialgu peamiselt ringi joonistamiseks, sellest tulenevalt on vaja määrata ringi raadius (Radius) ja pöördenurk (RotAngle). On võimalik ka teisi 2D kujundeid joonistada, kuid need on alati sümmeetrilised (nt ruut ja võrdkülgne kolmnurk). Näiteks ristkülikut ei ole praeguse prototüübiga võimalik teha. Enne „Connect“ nupule vajutamist tuleb kindlasti robotisisene programm käivatada, et rakendusel oleks võimalik robotiga ühendus luua.

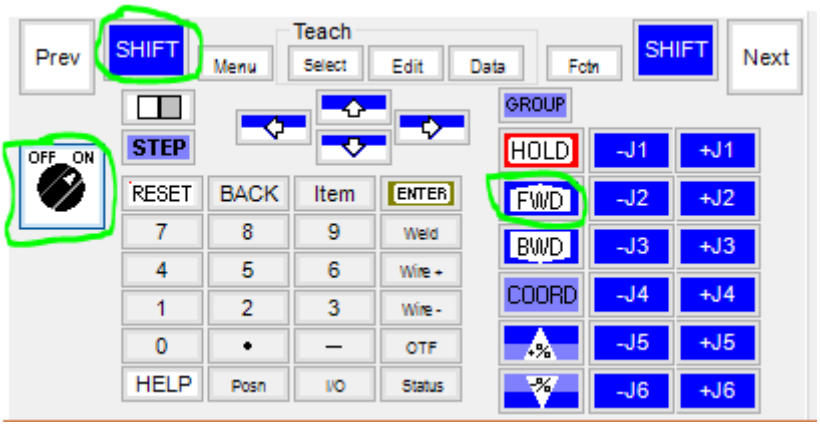


Joonis 7. Robotivälise rakenduse kasutajaliides

Joonisel 6 on kujutatud robotipoolne programm õpetamispuldis. Programmi käivitamiseks peab alati olema ära märgistatud esimene rida (Joonisel 6 mustaga märgitud rida 1). Kui see tingimus on täidetud, siis töö alustamiseks tuleb kontrolleris lüliti keerata ON režiimi, all hoida nuppu SHIFT (simulaatoris piisab ühest hiireklikkist) ning vajutada nupule FWD. Vajalikud nupud klaviatuuril on märgitud Joonisel 7 rohelisega. Samamoodi käib programmide käivitamine ka pärisroboti puhul.



Joonis 8. Robotisisene programm õpetamispuldis



Joonis 9. Vajalikud nupud programmi käivitamiseks

Peale ühenduse loomist, saadab rakendus robotile esimesed koordinaadid ning ootab roboti poolset vastust. Kui manipulaatori pea on õigesse kohta liigutatud, siis saadab robot rakendusele sõnumi, et on valmis uue koordinaadi vastu võtmiseks. Selline protsess toimub nii kaua kuni robot on läbinud täisringi (360 kraadi) või saadetud koordinaat asub roboti tööruumist väljas, mis tähendab, et robot ei ole võimeline antud punkti liikuma. Viimane variant tähendab ka seda, et etteantud keevitustööd ei ole võimalik antud robotiga realiseerida. Sellise olukorra tekkimisel jääb simulatsioon seisma ning klientrakenduses kuvatakse veateadet.

6 Keevitusroboti tarkvara prototüübi arendus

Prototüüp koosneb kahest osast:

- Robotivälisest rakendusest, kus läbi kasutajaliidese seadistatakse tööks vastavad parameetrid (roboti ip aadress, port, ringi raadius ja pöördenurk). Rakendus arvutab ja saadab koordinaate robotile.
- Robotisisesest programmist, mis võtab vastu rakenduse poolt saadetud koordinaadid, liigutab roboti vastavasse punkti ning saadab rakendusele kinnituse, et uued andmed võib teele saata.

Prototüübi mõlema osa arendamine toimus paralleelselt.

6.1 Robotivälise rakenduse arendus

Rakendus on kirjutatud Java programmeerimiskeeles.

Rakenduse kõige olulisem funktsionaalsus on andmevahetus robotiga. Ühendust on Javas väga mugav luua Socket-i abil (Joonis 8). Socketi loomisel kasutatakse parameetrina kasutaja poolt sisestatud roboti IP-aadressi ja porti. Ainult Socketi loomisest ei piisa. Selleks et programm saaks robotile andmeid saata ning sissetulevaid andmeid lugeda on vaja luua andmevood (DataOutputStream ja DataInputStream). DataOutputStream on andmevoog, kuhu Java rakendus kirjutab robotile saadetavad koordinaadid. DataInputStream on andmevoog, kust programm loeb serveri poolt saadetud andmeid. Roboti poolt saadetavad sõnumid annavad rakendusele teada, et võib edastada uued koordinaadid.

```
robot = new Socket(addr, port);
os = new DataOutputStream(robot.getOutputStream());
is = new DataInputStream(robot.getInputStream());
```

Joonis 10. Ühenduse loomine robotiga läbi rakenduse

Robotile on vaja andmed saata baitide kujul. DataOutputStream-i baitide kirjutamiseks on Javas funktsioon write() ja DataInputStream-ist lugemiseks on kasutusel funktsioon readFully().

Koordinaate arvutatakse reaalarvudena (float), mis on 4 baidi suurused, ning saata on vaja kaks koordinaati. See tähendab, et sõnumi pikkus on 8 baiti. Robot kasutab baitide hoidmiseks andmestruktuuri *stack*, s.t et sõnumisse esimesena lisatud baidi loeb robot viimasena. Selle probleemi lahendamiseks on rakenduses loodud funktsioon reverseBytes(), mis muudab baitide järjekorra vastupidiseks. Järjekorra muutmiseks peavad arvud aga olema *byte array* kujul. Seda tööd teeb programmis meetod toByteArray(), mis sisemiselt kasutab Javasse sisse ehitatud ByteBufferi meetodit wrap().

Lisaks on veel rakendusele loodud lihtne kasutajaliides, mis küsib kasutajalt vajalikke parameetreid roboti töö käivitamiseks. Liidese tegemiseks on kasutatud JavaFX-i ja SceneBuilder-it. JavaFX on Java programmide kasutajaliideste arendamiseks mõeldud platvorm. SceneBuilder[1] on JavaFX liideste kujundamiseks ja disainimiseks loodud programm.

Java rakenduse arenduseks on kasutatud Eclipse IDE-t. Eclipse[2] valiti arenduskeskkonnaks sellepärast, et autoril on eelmainitud programmiga varasem kokkupuude.

6.2 Robotisisese programmi arendus

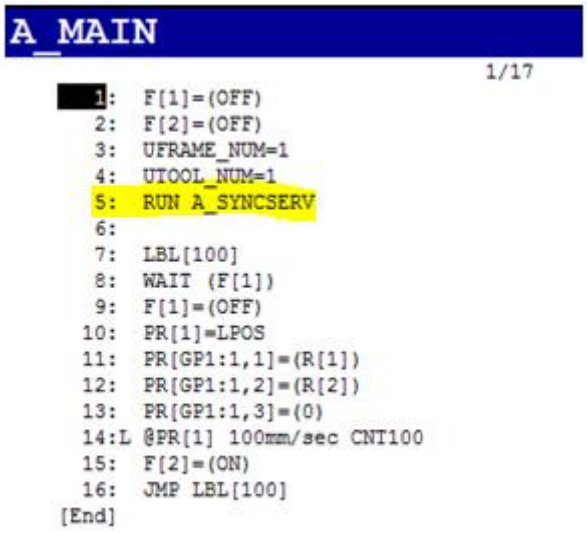
Prototüübi arenduse kõige töömahukam osa oli robotiprogrammide kirjutamine. Roboti programmeerimiskeeleks on Karel, millega autor pole varem kokku puutunud.

Robotipoolne programm võtab vastu Java rakenduse saadetud koordinaate, liigutab keevituspea vastavasse punkti ning saadab rakendusele tagasi kinnituse, et robot on soovitud punkti jõudnud.

[1] http://gluonhq.com/products/scene-builder/

[2] https://eclipse.org/

Joonisel 9 on näidatud roboti programm Roboguide-i õpetamispuldis. Programmi käivitamiseks peab olema märgistatud esimene rida mustaga (nagu Joonisel 9). Esimesed kaks rida on väga sarnased Boole'i muutujatele, millel on kaks väärtust (true või false). Kolmas ja neljas rida on vajalikud robotile sobiva koordinaatteljestiku paika panemiseks.



Joonis 11. Roboti programm õpetamispuldis

Real viis jooksutatakse Karelis kirjutatud programmi nimega A_SyncServ, kus toimub suurem osa tööst. Programmis luuakse ühendus ja suheldakse klientrakendusega (Joonis 10 ja 11). Ridadel 7-16 toimub roboti liigutamine, täpsemalt read 11-13 on koordinaadid, kuhu robotit liigutama hakatakse. Viimane koordinaat on alati null, sest programm tegeleb antud juhul ainult tasapinnaliste kujunditega. Teoreetiliselt on 3D kujundite lisamine programmi väga lihtne, andes väärtuseid ka kolmandale koordinaadile. Rida 14 määrab roboti liikumiskiiruse. Read 7 ja 16 moodustavad omavahel for-tsüklile sarnaselt töötava koodi. Kui programmi lugemisel jõuatakse viimasele reale, siis hüppab (JMP) progamm tagasi reale 7, kus asub LBL[100]. Seda tsüklit läbitakse nii kaua, kuni klientrakendus saadab robotile koordinaate.

```
WHILE(TRUE) DO

    IF (FLG[2]) THEN
        WRITE m_file_var (1)
        FLG[2]=FALSE
    ENDIF

    BYTES_AHEAD(m_file_var, n_bytes, m_Stat)
    IF (n_bytes>0) THEN
        READ m_file_var (m_X, m_Y)
        WRITE('X:=',m_X,', Y:=', m_Y, CR)
        SET_REAL_REG(1, m_X, m_Stat)
        SET_REAL_REG(2, m_Y, m_Stat)
        FLG[1] = TRUE
    ENDIF

    DELAY(10)
ENDWHILE
```

Joonis 12. Ühenduse kuulamine ja sõnumite saatmine programmis A_SyncServ

```
BEGIN
    SET_VAR(m_entry, '*SYSTEM*','$HOSTS_CFG[3].$SERVER_PORT',59002,stat)
    -- Connect the tag
    WRITE('Connecting..',CR)
    MSG_CONNECT(c_server,stat)
    WRITE(' CONNECT stat = ',stat,CR)
    IF (stat = 0) THEN -- Open S3:
            WRITE ('Connected',CR)
            WRITE ('Opening file',CR)
                SET_FILE_ATR(m_file_var,ATR_UF,1000)
                OPEN FILE m_file_var ('rw',c_server)
                stat = IO_STATUS(m_file_var)
            IF (NOT (stat =0)) THEN
                WRITE('open file failed',CR)
            ELSE
                WRITE('file open',CR)
            ENDIF

    Else
            WRITE('could not connect, stat = ',stat,CR)

    ENDIF

END connServ
```

Joonis 13. Rutiin ühenduse loomiseks klientrakendusega

Põhilised probleemid tekkisid sõnumite saatmisel robotist klientrakendusele. Karel programmide kirjutamisel lähtus autor peamiselt robotiga kaasa antud Kareli programmeerimiskeele manuaalist ja juhendaja nõuannetest.

7 Keevitusroboti tarkvara prototüübi analüüs

Valminud prototüüp võimaldab ringi raadiuse ja pöördenurga alusel analüüsida keevitustöö võimalikkust konkreetse robotiga. Antud töö raames sellest piisab, kuid reaalselt peaks tarkvara suutma analüüsida kõikvõimalikke 2D kujundeid ning lisaks veel ka 3D kujundeid, sest enamus keevitustöid on siiski kolme mõõtmelised. Loodud prototüüp on heaks aluseks sellise tarkvara loomisele, pakkudes juba praegu andmeühendust robotiga ja lihtsamate kujundite analüüsimist. ASG Robotics soovib antud prototüübi alusel enda jaoks välja töötada tarkvara, mis suudab analüüsida kõikvõimalikke kujundeid, nii kahe- kui kolmemõõtmelisi.

Töö käigus selgus, et Java programmeerimiskeel ei ole kõige sobilikum tööstusrobotite klientrakenduste programmeerimiseks. Varem on firma siseselt kasutatud taolise rakenduse tegemiseks C#-i. Eriti hästi tuli ebasobilikkus välja kasutajaliidese arendamisel. Plaanis oli teha liides, mis näitab reaalajas kujundi joonistamist, kuid selle tegemine Javas osutus liiga keeruliseks ülesandeks. Kuna autor on kõige rohkem tuttav Java programmeerimiskeelega, siis selle põhjal tehtigi valik Java kasuks. ASG Robotics plaanib ka kasutajaliidest edasi arendada, muutes seda interaktiivsemaks ja kasutajasõbralikumaks.

8 Kokkuvõte

Käesoleva töö raames loodi ASG Robotics-ile keevitusroboti tarkvara prototüüp, mis analüüsib roboti liikumist 2D kujundite põhjal. Töö esimeses pooles tutvustati tööstusroboteid üldiselt ning samuti töös kasutatavat konkreetset robotit. Lisaks kirjeldatakse käsitletavat probleemi ning metoodikat selle lahendamiseks. Töö teises pooles kirjeldatakse loodud tarkvara ning selle loomisprotsessi.

Valminud tarkvara võimaldab kasutajal sisestada kujundi parameetrid läbi robotivälise rakenduse, mis hakkab saatma koordinaate loodud andmeühenduse kaudu robotile. Robotiprogramm liigutab keevituspea vastavasse koordinaati ning saadab rakendusele tagasi teate roboti valmisolekust uue koordinaadi vastu võtmiseks. Kui saadetud asukohta ei ole võimalik keevituspead liigutada, siis robot jääb seisma ning kuvatakse veateadet.

ASG Robotics plaanib loodud prototüübi baasil välja töötada tarkvara, millega saaks analüüsida kõikvõimalikke kujundeid ning mida oleks ka võimalik kasutada roboti töö automatiseerimisel.

Antud tööle püstitatud eesmärgid said suuremas osas täidetud. Loodi tarkvara, mis suudab analüüsida lihtsamaid kahemõõtmelisi kujundeid, kasutades roboti liikumise arvutamiseks kvaternioone. Siiski ei suuda programm analüüsida kõikvõimalikke 2D kujundeid, mida ei jõutud töö käigus valmis teha. Lisaks sellele tuleks edasiarendada kasutajaliidest ning anda programmile võimekus ka 3D kujundeid analüüsida.

Kasutatud kirjandus

[1] Who did actually invent the word "robot" and what does it mean? [WWW] https://web.archive.org/web/20120204135259/http://capek.misto.cz/english/robot.html (29.04.2018)

[2] Lehtla, Tõnu. Robotitehnika. Tallinna Tehnikaülikool, Elektriajamite ja jõuelektroonika instituut, Tallinn, 2008. 201 lk.

[3] Lehla, Tõnu; Müür, Margus; Rätsep, Tiit. Robotitehnika kutsekoolidele [Võrguteavik], Innove, Tallinn, 2014, 201 lk.

[4] Arc Welding Faults - Examples of Speed, Arc Length, and Current Problems. [WWW] http://www.mig-welding.co.uk/arc-welding-faults.htm (29.04.2018)

[5] Handbook - Welding Techniques. [WWW] http://www.esabna.com/euweb/mig_handbook/592mig7_7.htm (29.04.2018)

[6] Arc Welding Fundamentals | Lincoln Electric. [WWW] http://www.lincolnelectric.com/en-us/support/process-and-theory/Pages/arc-welding-detail.aspx (29.04.2018)

[7] Rozenfeld, Boris A. The history of non-euclidean geometry: evolution of the concept of a geometric space, Springer, 385 lk.

[8] Ntrs.nasa.gov [WWW] https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19770024290.pdf (01.05.2018)

[9] Maxwelli teooria lähtekohad (Füüsika ajalugu) [WWW] http://opik.fyysika.ee/index.php/book/section/4592 (01.05.2018)

LISA 1 – Karelis kirjutatud programm roboti liigutamiseks

```
PROGRAM A_ROT

VAR
	mAxis:XYZWPR
	mCurPos:XYZWPR
	mRotAngle:REAL
	mRegIsReal:BOOLEAN
	mInt, mStatus:INTEGER
	mQ1, mQ2, mQ3 : ARRAY[4] OF REAL


BEGIN

	--Get rotation axis from PR[1]
	mAxis	= GET_POS_REG(1,mStatus,1);  IF mStatus<>0 THEN
POST_ERR(mStatus,",0,1); GOTO ex; ENDIF

	--Get current position stored in PR[2]
	mCurPos = GET_POS_REG(2,mStatus,1);  IF mStatus<>0 THEN
POST_ERR(mStatus,",0,1); GOTO ex; ENDIF

	--Get rotation angle from R[1]. This is another type of registers. Press 'Data' to
find them. These registers can contain integer or real numbers.
	GET_REG(1,mRegIsReal, mInt, mRotAngle, mStatus);  IF mStatus<>0 THEN
POST_ERR(mStatus,",0,1); GOTO ex; ENDIF
	IF mRegIsReal=FALSE THEN mRotAngle = mInt; ENDIF --Convert integer
number to real one

	-- Create quaternion to be rotated
	mQ2[1] = 0
	mQ2[2] = mCurPos.x - mAxis.x
	mQ2[3] = mCurPos.y - mAxis.y
	mQ2[4] = mCurPos.z - mAxis.z

	--FORCE_SPMENU( TP_PANEL, SPI_TPUSER, 1)

	--WRITE('Angle: ', mRotAngle, CR)
	--WRITE('Q1.1: ', mQ1[1], CR)
	--WRITE('Q1.2: ', mQ1[2], CR)
	--WRITE('Q1.3: ', mQ1[3], CR)
	--WRITE('Q1.4: ', mQ1[4], CR)

	-- Create quaternion for rotation
	mQ1[1] = COS(mRotAngle/2)
	mQ1[2] = mAxis.w * SIN(mRotAngle/2)
	mQ1[3] = mAxis.p * SIN(mRotAngle/2)
	mQ1[4] = mAxis.r * SIN(mRotAngle/2)
```

```
	-- mQ1 * mQ2
	mQ3[1] = mQ1[1]*mQ2[1]-mQ1[2]*mQ2[2]-mQ1[3]*mQ2[3]-
mQ1[4]*mQ2[4]
	mQ3[2] = mQ1[1]*mQ2[2]+mQ1[2]*mQ2[1]+mQ1[3]*mQ2[4]-
mQ1[4]*mQ2[3]
	mQ3[3] = mQ1[1]*mQ2[3]+mQ1[3]*mQ2[1]-
mQ1[2]*mQ2[4]+mQ1[4]*mQ2[2]
	mQ3[4] = mQ1[1]*mQ2[4]+mQ1[4]*mQ2[1]+mQ1[2]*mQ2[3]-
mQ1[3]*mQ2[2]

	-- Inverse quaternion for rotation
	mQ2[1] = mQ1[1]
	mQ2[2] = -1*mQ1[2]
	mQ2[3] = -1*mQ1[3]
	mQ2[4] = -1*mQ1[4]

	mQ1[1] = mQ3[1]
	mQ1[2] = mQ3[2]
	mQ1[3] = mQ3[3]
	mQ1[4] = mQ3[4]


	mQ3[1] = mQ1[1]*mQ2[1]-mQ1[2]*mQ2[2]-mQ1[3]*mQ2[3]-
mQ1[4]*mQ2[4]
	mQ3[2] = mQ1[1]*mQ2[2]+mQ1[2]*mQ2[1]+mQ1[3]*mQ2[4]-
mQ1[4]*mQ2[3]
	mQ3[3] = mQ1[1]*mQ2[3]+mQ1[3]*mQ2[1]-
mQ1[2]*mQ2[4]+mQ1[4]*mQ2[2]
	mQ3[4] = mQ1[1]*mQ2[4]+mQ1[4]*mQ2[1]+mQ1[2]*mQ2[3]-
mQ1[3]*mQ2[2]

	--
	mCurPos.x = mQ3[2] + mAxis.x
	mCurPos.y = mQ3[3] + mAxis.y
	mCurPos.z = mQ3[4] + mAxis.z


	-- Create quaternion of the current tool orientation
	mQ1[1] =
SIN(mCurPos.r/2)*SIN(mCurPos.p/2)*SIN(mCurPos.w/2)+COS(mCurPos.r/2)*COS(
mCurPos.p/2)*COS(mCurPos.w/2)
	mQ1[2] = -
1*SIN(mCurPos.r/2)*SIN(mCurPos.p/2)*COS(mCurPos.w/2)+SIN(mCurPos.w/2)*CO
S(mCurPos.r/2)*COS(mCurPos.p/2)
	mQ1[3] =
SIN(mCurPos.r/2)*SIN(mCurPos.w/2)*COS(mCurPos.p/2)+SIN(mCurPos.p/2)*COS(
mCurPos.r/2)*COS(mCurPos.w/2)
	mQ1[4] = SIN(mCurPos.r/2)*COS(mCurPos.p/2)*COS(mCurPos.w/2)-
SIN(mCurPos.p/2)*SIN(mCurPos.w/2)*COS(mCurPos.r/2)
```

```
        mQ2[1] = COS(mRotAngle)
        mQ2[2] = mAxis.w * SIN(mRotAngle)
        mQ2[3] = mAxis.p * SIN(mRotAngle)
        mQ2[4] = mAxis.r * SIN(mRotAngle)

        mQ3[1] = mQ1[1]*mQ2[1]-mQ1[2]*mQ2[2]-mQ1[3]*mQ2[3]-
mQ1[4]*mQ2[4]
        mQ3[2] = mQ1[1]*mQ2[2]+mQ1[2]*mQ2[1]+mQ1[3]*mQ2[4]-
mQ1[4]*mQ2[3]
        mQ3[3] = mQ1[1]*mQ2[3]+mQ1[3]*mQ2[1]-
mQ1[2]*mQ2[4]+mQ1[4]*mQ2[2]
        mQ3[4] = mQ1[1]*mQ2[4]+mQ1[4]*mQ2[1]+mQ1[2]*mQ2[3]-
mQ1[3]*mQ2[2]

        mCurPos.r = ATAN2((1-2*mQ3[3]*mQ3[3]-
2*mQ3[4]*mQ3[4]),(2*mQ3[2]*mQ3[3]+2*mQ3[4]*mQ3[1]))
        mCurPos.p = ATAN2(SQRT(1-(2*mQ3[2]*mQ3[4]-
2*mQ3[3]*mQ3[1])*(2*mQ3[2]*mQ3[4]-2*mQ3[3]*mQ3[1])),(-
1*(2*mQ3[2]*mQ3[4]-2*mQ3[3]*mQ3[1])))
        mCurPos.w = ATAN2((1-2*mQ3[2]*mQ3[2]-
2*mQ3[3]*mQ3[3]),(2*mQ3[3]*mQ3[4]+2*mQ3[2]*mQ3[1]))


        SET_POS_REG(2, mCurPos, mStatus, 1);  IF mStatus<>0 THEN
POST_ERR(mStatus,",0,1); GOTO ex; ENDIF

        --FORCE_SPMENU( TP_PANEL, SPI_TPUSER, 1)

        --WRITE('Q3.1: ', mQ3[1], CR)
        --WRITE('Q3.2: ', mQ3[2], CR)
        --WRITE('Q3.3: ', mQ3[3], CR)
        --WRITE('Q3.4: ', mQ3[4], CR)

ex::
END A_ROT
```