TIETO- JA SÄHKÖTEKNIIKAN TIEDEKUNTA

Juho Bruun
Aleksi Tuovinen
Ville Hokkinen

ROBOTIN KÄDEN JA KYYNÄRVARREN TOTEUTUS

Kandidaatintyö
Tietotekniikan tutkinto-ohjelma
Lokakuu 2022

Bruun J., Tuovinen A., Hokkinen V. (2022) Robotin käden ja kyynärvarren toteutus. Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 43 s.

TIIVISTELMÄ

Robotiikka on alana jatkuvassa nousussa, ja se kehittyy huimaa vauhtia. Kärkipään robotiikan tutkimus on kallista ja kehittynyttä tiedettä, mutta myös yksittäisille ihmisille on tullut mahdolliseksi tutkia ja harrastaa robotiikkaa. Varsinkin vapaasti saatavilla olevien ohjelmistojen, sekä avoimen lähdekoodin sovellusten merkitys yksittäisen ihmisen robotiikan harrastamiselle on suuri. Tämän työn tarkoituksena on olla tukena ja ensiaskeleena harrastajille sekä oppilaille, jotka haluavat lähteä toteuttamaan omaa robottitoteutustaan.

Tässä opinnäytetyössä pääpisteenä on InMoov robottimallin käden rakentaminen. Tähän sisältyy fyysisen InMoov-robottikäden kokoaminen, kuuden Dynamixel servomoottorin asentaminen ja niiden toimivaksi saattaminen Robot Operating System 2-kirjaston kanssa. Työssä on kirjoitettu Python-ohjelmointikieltä käyttäen asiakasohjelma, jolla ohjataan Dynamixel-servomoottoreita, jotka ohjaavat käden sormien ja ranteen niveliä tehden käden liikkeestä ihmismäisen. Verrattuna robotiikan teknologian huipputasoon, projektissa rakennettu käsi on yksinkertainen, mutta se kuvastaa hyvin ei-kaupallisten tai tuettuen tekijöiden mahdollisuuksia rakentaa robotiikan sovelluksia.

Työn lopussa pohditaan projektin onnistumista ja sen jatkomahdollisuuksia. Työ onnistui tavoitteessaan saada aikaan toimiva robottikäsikokonaisuus joka kykenee herättämään huomiota ja liikkumaan niin itsenäisesti kuin osana suurempaa kokonaisuutta. Rakenteelliset ja yksinkertaisuudesta johtuvat ongelmat tiedostetaan ja niistä varoitetaan, jotta työn lukijat voivat välttää vastaavat ongelmat. Jatkokehitystä ajatellen pohditaan käden mahdollisuuksia olla osana kokonaista InMoov-robottia ja käden modulaarisuutta. Lisäksi nostetaan esiin, sekä ohjelmistopuolen skaalautuvuuden, että sen helppokäyttöisyyden ongelmat ja onnistumiset.

Avainsanat: Robottikäsi, InMoov, Robot Operating System 2, Sosiaaliset robotit, Ihminen-robotti vuorovaikutus, Dynamixel, kobotti

Bruun J., Tuovinen A., Hokkinen V. (2022) Robot Hand and Forearm Implementation. University of Oulu, Degree Programme in Computer Science and Engineering, 43 p.

ABSTRACT

The robotics sector is on a constant rise and the trajectory of advancements in the area is staggering. The cost of high-end robotics research is way beyond something that a regular person could hope to do, but fortunately, there are also possibilities for low-end robotics work and research. This is done mainly through freeware and open-source software, and by using cheap parts like 3D-printed shells. This thesis is meant to guide and help students and hobbyists who are interested in researching and implementing their own low-end robotics solutions.

The main focus of this thesis is building an InMoov robot arm, installation of six Dynamixel servomotors into the arm, and finally making a working client for the hand using Robot Operating System 2-library. The client has been done using Python-programming language, which controls the Dynamixel servomotors, which in turn control the fingers and the wrist of the arm, trying to simulate a human hand as closely as possible. In comparison to the top of the robotics research, the hand is quite rudimentary, but it showcases the possibilities of low-end robotics research done by hobbyists and students.

The end of the thesis goes through how the project succeeded and what could still be improved upon and how the work can be further developed in the future. The project succeeded in its main goal of building a working robotic hand that can attract attention and showcases the area of low-end robotics and its modularity. The problems caused by the part quality and the simplicity of the hand design are touched upon, to raise awareness of these issues. Future development is considered through the modularity of the project and how it could be expanded into a full-fledged InMoov robot torso. The problems and successes in the software scalability and ease of use are also expanded upon.

Keywords: Robot arm, InMoov, Robot Operating System 2, Social robots, Human-robot interaction, Dynamixel, cobot

SISÄLLYSLUETTELO

ALKULAUSE

Tämä opinnäytetyö tehtiin Oulun yliopiston tieto- ja sähkötekniikan tiedekunnan Sulautettujen ohjelmistojen projekti -kurssilla. Kurssiprojekti toteutettiin ryhmätyönä, johon kaikki ryhmän jäsenet osallistuivat tasapuolisesti. Haluamme kiittää Teemu Tokolaa työn ohjauksesta ja Lauri Haverista työn käytännön asioissa avustamisesta.

Oulussa 10. lokakuuta 2022

Juho Bruun
Aleksi Tuovinen
Ville Hokkinen

LYHENTEIDEN JA MERKKIEN SELITYKSET

HRI	Human Robot Interaction, Ihminen-robotti vuorovaikutus
iHRI	Industrial Human Robot Interaction, Teollinen ihminen-robotti vuorovaikutus
DT	Digital Twin, Digitaalinen kaksonen
CBT	Cognitive Behavioural Therapy, Kognitiivinen käyttäytymisterapia
SAR	Social Assistant Robot, Sosiaalinen avustajarobotti
ROS	Robot Operating System
ROS2	Robot Operating System 2
XML	Extensive Markup Language
xacro	XML Macros
URDF	Unified Robot Description Format
colcon	Collective Construction
YAML	YAML Ain't Markup Language
HUMP	Human-like Upper-limb Motion Planner
STL	Standard Tessellation Language
rclpy	ROS Client Library
CLI	Command line interface, komentoliittymä

1. JOHDANTO

Kirjoissa ja elokuvissa robotit esitetään humanoideina, joiden kanssa ihmiset keskustelevat ja elävät rinnakkain. Tämänhetkisellä teknologialla tällainen toiminta on vielä kaukana. Uusinta teknologiaa edustavat robotit ovat monimutkaisia koneita, jotka suorittavat tarkkaan määriteltyä tehtävää, eivätkä ne kykene sen suurempaan autonomiaan.

Tulevaisuuden robotit, jotka elävät yhdessä ihmisten kanssa ovat vielä haave, mutta jo vuonna 2009 ennustettiin, että robottien käyttöä vanhusten hoidossa erilaisissa pienissä tehtävissä voidaan harkita jo vuonna 2025. [1]

Humanoidirobotti vaatii huomattavan määrän teknologiaa toimiakseen. Nykyteknologialla rakennetut humanoidirobotit ovat hitaita ja niiden liikkeet saavat aikaan outo laakso -ilmiön, eli suurin osa ihmisistä kokee kyseisten robottien toiminnan outoina ja epäilyttävinä[2]. Jotta outo laakso voidaan ylittää, pitää robottien liikkeisiin ja toimintoihin saada enemmän ihmismäistä olemusta ja luontevuutta.

Yksi tärkeä osa-alue luonnollisuuden saavuttamiseksi on robottien käsien liike. Jos robotin kädet liikkuvat epäluonnollisen näköisesti, saa se helposti aikaan edellämainitun outo laakso -efektin. Teollisuusrobotit kykenevät liikuttamaan käsiään nopeasti ja luonnollisen näköisesti, mutta niiden liikeradat ovat tarkalleen laskettuja ja automatisoituja. Humanoidirobotin, joka työskentelee vanhainkodissa tai sairaalassa, pitää olla toiminallisuudeltaan paljon joustavampi liikkeiden suunnittelemisessa ja toteuttamisessa. Tämä vaatii monimutkaisten toimintojen asentamista käteen, jotta robotin toiminta vaikuttaa edes jokseenkin luonnolliselta. [3, 4, 5]

Jotta saadaan täysin itsenäisesti toimiva robottikäsi, tarvitaan toimivat ratkaisut sekä sensoreille, mekaniikalle, että ohjelmistolle. Tässä työssä käydään läpi nykyistä robottiteknologiaa ja uusinta tutkimusta, sekä esitellään teknologioita, joita hyödyntämällä voidaan toteuttaa erilaisia robottikäsialustoja, sekä selitetään miten InMoov robottialusta on rakennettu ja ohjelmoitu käyttäen Robot Operating System 2 -kirjastoa. Tätä alustaa voidaan jatkossa hyödyntää havainnollistamaan robottikäsien toimintaa ja kykyjä.

2. ROBOTIT YHTEISKUNNASSA JA LÄHITULEVAISUUDESSA

Robottien tullessa yhä yleisimmiksi ympäristöissä kuten kodeissa ja kouluissa, aletaan niiltä vaatia sosiaalista älykkyyttä toimia ja auttaa tehokkaasti ihmisiä. Tämä tavoite vaatii tarkkaa monialaista tutkimustyötä. Ihmisten ja robottien välisen vuorovaikutuksen (eng. Human-robot interaction, HRI) tutkimisessa tutkitaan sosiaalisia ja toiminnallisia ihminen-robotti suhteita tekniikan, psykologian, kielitieteen, etologian ja tietojenkäsittelytieteiden välisessä rajapinnassa [6]. Ihmisen ja robotin välinen vuorovaikutus luodaan kommunikaatiolla, niin sanallisella että sanattomalla, ja niiden tehokkaalla käytöllä voidaan luoda toimiva kommunikaatioyhteys robotin ja ihmisen välille [7].

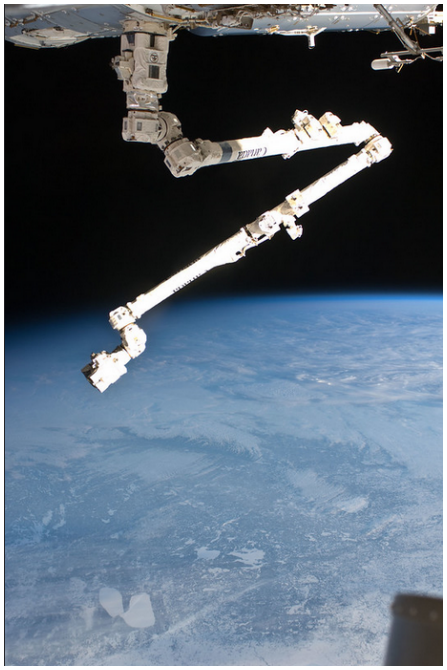
Ihmisiin liittyvien tekijöiden tutkiminen on yksi HRI:n tärkeistä tutkimusalueista. Ihmistekijöiden tärkeimpiin tutkimuskohteisiin kuuluu tehtäväanalyysi, robotin opettaminen ja tahattomien seurauksien välttäminen, robotin ja ihmisen toistensa välisen mallin huomioon ottaminen, robottien käyttö koulutuksessa ja käyttäjäkulttuurin, käyttäjän pelkojen ja arvojen huomioon ottaminen. [8]

2.1. Robotit ihmisten rinnalla

Yhteiskunnan eri aloilla on käytössä mitä erilaisimpia robotiikan ratkaisuja. Käyttökohteet roboteille vaihtelevat avaruustutkimuksesta vanhustenhoitoon, joten myös erilaiset toteutukset voivat erota todella huomattavasti toisistaan.

2.1.1. Avaruustutkimus

Avaruustutkimuksessa on pitkään käytetty onninstuneesti robotteja ihmisten työskentelyn tukemiseksi. Kansainvälisellä avaruusasemalla (ISS) on robottikäsi, joka on nähtävissä kuvasta 1. Sitä on käytetty aseman korjaukseen ja huoltoon onnistuneesti pitkäaikaisesti. ISS:n ja NASA:n avaruussukkuloiden ohjaukseen on myös käytetty robottimanipulaattoreita onnistuneesti. Tulevaisuudessa avaruustehtävissä tullaan luultavasti tukeutumaan koko ajan enemmän robottien ja ihmisten tiimityöskentelyyn[9]. NASA:n tulevassa Artemis-ohjelmassa on tarkoitus tukeutua kuuta kiertävään Gateway-asemaan [10]. On odotettavissa että tämä asema on miehittämätön suuren osan ajasta, minkä avulla voidaan testata ja kehittää robotteja ylläpitämään ja hallitsemaan asemaa ihmisten poissa ollessa. Myös muita tehtäviä on suunnitteilla, joiden yhtenä päätarkoituksena on HRI:n kehittäminen avaruuden tutkinnassa [11, 9]. Nämä tutkimukset ovat merkittäviä, sillä niiden tuottamien tulosten avulla voidaan kehittää robottien työskentelykapasiteettia avaruudessa ja näin vähentää miehistölle aiheutuvaa psykologista ja fyysistä työkuormaa pitkillä avaruusmatkoilla [12]. Suunnitelluilla tulevilla miehitetyillä Mars-tehtävillä odotetaan myös korkean tason itsenäistä automaatiota, yhdistettynä ihmisten kanssa työskenteleviin robotteihin, jota aiemmin esitetyt tutkimukset tulevat suuresti tukemaan [9].



Kuva 1. *Kansainvälisen avaruusaseman robottikäsi*[1]

ISS:n kyydissä on myös vuodesta 2017 asti ollut kyydissä Astrobee-robotteja, jotka ovat pieniä kuution muotoisia autonomisia robotteja. Niiden päätarkoitus on tarjota muokattava alusta painottoman tilan vapaasti lentävän robotiikan tutkimukselle. Niiden on tarkoitus olla täysin autonomisia, mutta ne kuitenkin sisältävät manuaalisen ajon ominaisuuden. [13]

Astrobeen kyydissä on myös kamera, jota tehtävänvalvonnan on tarkoitus käyttää pysyäkseen paremmin perillä tilanteesta ISS:n sisällä. Tehtävänvalvonta voi seurata esimerkiksi astronauttien korjausoperaatioita tai tulevan lastin tilaa. Astrobeellä voidaan myös suorittaa erilaisia mittauksia ympäri avaruusasemaa, kuten inventaariota RFID-skannerilla, äänentason mittauksia, ilmanlaadun mittauksia, säteilytasojen mittauksia ja yleisiä tarkastuksia. Tulevaisuudessa tämän tyyppisiä robotteja tullaan käyttämään miehittämään avaruusasemia silloin kun miehistöä ei ole paikalla, korjaten ja huoltaen asemaa tarpeen vaatiessa. [14]

2.1.2. Autonominen ajaminen

Ihminen-robotti vuorovaikutus on tärkeässä osassa myös autonomisessa ajamisessa ja ohjauksessa [15, 16]. Autonomiseen auton ohjaukseen on määritelty 6 autonomian tasoa välillä 0-5. Tässä taso 0 tarkoittaa täysin ihmisen ohjaamaa ajoneuvoa. Taso 1 taas tarkoittaa kuljettajan avustamista teknologioilla, kuten kaistavahti tai mukautuva vakionopeudensäädin. Taso 2 on osittaista autonomista ajamista, jossa kuljettaja valvoo autonomiaa ja tutkii ympäristöä autonomian havaintokyvyn ulkopuolella olevian esineiden varalta ja taso 3 on täysin autonomista ajamista tiettyjen ehtojen täyttessä. Taso 4 on täysin autonominen ajoneuvo kumminkin mahdollisuudella antaa

[1]Kuva NASA: https://www.flickr.com/photos/nasamarshall/5410176673 (Lisenssi CC BY-NC 2.0)

kuskin ohjata ajoneuvoa ja taso 5 tarkoittaa täysin autonomista ajoneuvoa ilman kuskin mahdollisuutta ajaa autoa [17].

Teslan autopilotti (AP) on tason 2 autonominen ominaisuus, jossa auto ohjaa itseään pysymään tiellä ja hallitsemaan auton nopeutta automaattisesti [18]. Sen pääosainen tehtävä on lisätä matkustajan mukavuutta, mutta sen on arvioitu lisäävän turvallisuutta kasvattamalla turvamarginaaleja suuremmiksi kuin mitä ihmiskuskeilla keskimääräisesti [19].

Teslan turvallisuusraportit [20] viittaavat myös siihen, että AP:n käyttö lisää kuljettajan turvallisuutta huomattavasti. Nyholm et al. [16] myös tuo esille kuinka autonomiset ajoneuvot pyrkivät aina turvalliseen ja optimaaliseen ajamiseen, kun taas ihmiskuljettajat ajavat minimiperiaatteen perusteella, juuri niin turvallisesti ja optimaalisesti kuin on tarpeen. Tällä alalla HRI:lla onkin paljon painoarvoa, sillä autonomisten ajoneuvojen täytyy ottaa huomioon ihmisten epäoptimaalinen ajotyyli ja pyrkimys siihen. Ajoneuvoja on jopa ehdotettu ohjelmoitavaksi rikkomaan sääntöä sellaisia tapauksissa missä ihmiset olettavat niitä rikottavan [21], näin käyttäytyen enemmän ihmisten oletusten mukaisesti. Proksemiikka ja kinesiikka ovatkin alalla tärkeässä osassa, jotta autot toimivat ihmisille turvalliselta vaikuttavalla tavalla [22, 23].

2.1.3. Yhteistyörobotit

Teollisuusrobotit on otettu laajasti käyttöön kaikkialla maailmassa. Tämän myötä myös tutkijat ovat alkaneet tutkia teollisuusrobottien ja ihmisten välistä vuorovaikutusta. Tämä on johtanut uuteen HRI:n alamuotoon, teolliseen HRI:hin (eng. Industrial Human Robot Interaction, iHRI).

Työntekijöiden ja teollisuusrobottien välinen vuorovaikutus jää yleensä hyvin vähäiseksi. Teollisuusrobotit työskentelevät hyvin rajatussa tilassa johon ihmisillä ei ole pääsyä, jotta robottia voidaan ajaa mahdollisimman nopeaa täydellä teholla. Tiloissa, joissa robotit eivät ole eristettynä muista työntekijöistä, roboteissa pitää olla paljon sisäisiä turvallisuusominaisuuksia, jotka rajoittavat robottien nopeutta, voimaa sekä muita ominaisuuksia. [24]

Yhteistyörobotit eli kobotit ovat tavallisten teollisuusrobottien versioita, jotka voivat toimia samassa tilassa yhdessä ihmisten kanssa. Kobotit ovat yleensä pieniä ja kevyitä, jotta ne täyttävät standardien mukaiset turvavaatimukset. Kyseiset ominaisuudet tekevät niistä halvempia kuin normaalit teollisuusrobotit, mutta ei kuitenkaan yhtä tuottavia. Kobottien pääsääntönä pidetään turvallisuutta ja kobotit laittavat ihmisturvallisuuden kaiken muun edelle. [25, 26]

Kobotteja ja normaaleja teollisuusrobotteja vertailtaessa havaitaan, että kobotit ovat funktionaalisesti hyvin erilaisia teollisuusrobotteihin verrattuna. Kobotit ovat rakenteeltaan pyöreäreunaisia, jotta niiden kanssa toimivat käyttäjät eivät saa haavoja terävistä reunoista, ja mahdolliset kaapelit on asennettu sisälle kobottiin, jotta käyttäjä ei voi sotkeutua niihin. Perinteisissä teollisuusroboteissa moottorit ja servot on optimoitu voimaa varten, kun taas koboteissa kyseisten osien voimaa on rajattu. Koboteissa on myös suuri määrä erilaisia sensoreita perinteisiin teollisuusrobotteihin verrattuna, jotta ne voivat tunnistaa ja varoa ihmisiä joiden parissa ne työskentelevät. Myös laaja sopeutumiskyky eri tehtäviin on avainsana kobottien kanssa. Kobotit

suunnitellaan siten, että niitä pystytään siirtämään yhdestä tehtävästä toiseen mahdollisimman helposti ja nopeasti. [25, 27]

Tällä hetkellä kobotteja on jo käytössä teollisessa mittakaavassa, muun muuassa General Motorsin Orionin tehtaassa. Tehtaalla kobotit työskentelevät ihmistyöntekijän kanssa niin, että kobotti siirtää renkaita syrjään liukuhihnalta ja laittaa ne kasoihin automallin mukaan. Suurimmaksi ongelmaksi kobotin käytössä muodostui sen liikehdintä ahtaassa paikassa, jossa ihmisiä voi olla puolen metrin etäisyydessä [28]. Ratkaisuna kobottien suoritusongelmiin on ehdotettu digitaalisten kaksosten (eng. digital twin, DT) käyttöä. Digitaalisien kaksosten avulla kobottien liikeradat ja käyttötarkoitukset voidaan optimoida ilman suuria kuluja [26].

2.2. Sosiaaliset robotit

Kun puhutaan sosiaalisista roboteista, tulisi ensin päättää, mitkä tekijät määrittelevät sosiaalisen robotin. Tässä kontekstissa sosiaalisella robotilla tarkoitetaan robottia, jonka ensisijainen tarkoitus on olla jollakin tasolla kanssakäymisessä ihmisten kanssa, ja joka käyttää ihmisille tyypillisiä viestintäkeinoja kuten liikettä, puhetta ja kasvojen ilmeitä kommunikoimiseen. Lisäksi sosiaalisen robotin määritelmään kuuluu usein fyysisen olemuksen käyttö toiminnassa ja viestinnässä, sekä kyky itsenäiseen päätöksentekoon omien havaintojen pohjalta [29, 30, 4, 31, 32, 33]. Sosiaalisille roboteille suosittuja ulkonäköjä ovat realistisen humanoidityylisen rungon lisäksi myös pehmolelua, eläintä tai laitetta muistuttavat ulkomuodot [34].

Tarkastellessa sosiaalisten robottien käyttötarkoituksia, kuten autististen lasten terapiavälineenä toimimista [35], dementiapotilaiden avustamista [36] tai viihde- ja demonstraatiokäyttöä [37], voidaan huomata joidenkin asioiden olevan erityisen tärkeitä positiivisen käyttö- tai kohtaamiskokemuksen kannalta. Robottien ulkonäköä suunnitellessa tulisi välttää ihmisissä negatiivisia tuntemuksia herättävää outo laakso -ilmiötä [4], mikä käytännössä tarkoittaa, että tulisi löytää sopiva tasapaino ihmisen ja robotin piirteiden välillä. Koppenborg et al. [3] tutkimuksen mukaan myös robotin liikkeiden sopivalla nopeudella ja ennustettavuudella voidaan välttyä tuottamasta stressiä sen kanssa tekemisissä oleville ihmisille.

2.2.1. Sosiaaliset robotit hoito- ja terapiavälineenä

Kuva 2. *Softbank roboticsin luoma NAO-humanoidirobotti*[2]

Sosiaalisista roboteista on teknologian edistyessä muodostumassa tehokas terapiaväline autististen lasten oireiden lieventämiseen. Marino et al. [38] suorittivat tutkimuksen, jossa tarkasteltiin robottiavusteista kognitiivista käyttäytymisterapiaa (eng. Robot Assisted Cognitive Behavioural Therapy, RA-CBT) ja sen vaikutuksia autistisiin, neljästä kahdeksaan ikävuotta vanhoihin lapsiin.

Tutkimuksessa käytettiin kuvassa 2 esitettyä terapeutin ohjailtavissa olevaa puoliautonomista NAO-humanoidirobottia, johon oli ohjelmoitu erilaisia tilanteeseen sopivia käyttäytymismalleja. Tutkimuksen tulosten mukaan robottiavusteista terapiaa saaneilla lapsilla tapahtui huomattavasti suurempaa positiivista kehitystä kuin kontrolliryhmän normaalia kognitiivista käyttäytymisterapiaa (eng. Cognitive Behavioural Therapy, CBT) saaneilla lapsilla.

Toinen esimerkki autismin hoitoa varten kehitetystä robotista on vuonna 2005 luotu Kaspar.
Wood et al.[35] kertovat artikkelissaan Kaspar-robotin kehitysprosessista alkuajoista nykyhetkeen. Zaraki et. al [39] kehittivät Kasparin uusimpaan iteraation arkkitehtuurin, jonka avulla se pystyy toimimaan puoliautonomisesti. Robotti pystyy silmissä sijaitsevien kameroiden kautta tunnistamaan esineitä ja ihmisiä. Myös ruumiinosien liikuttaminen onnistuu jäseniin asennettujen servomoottorien avulla. Robotti osaa antaa päähänsä asetetusta kaiuttimesta lapselle palautetta siitä, vastasiko tämä oikein leikkien aikana. Tällä hetkellä palautemekanismissa on toiminto, jonka avulla ihminen vahvistaa robotin havainnon, jotta vältyttäisiin antamasta väärää palautetta lapselle. Kuvantunnistusteknologian kehittyessä toiminnosta voidaan tehdä täysin automaattinen. Robottiin ohjelmoidut leikit perustuvat pitkälti sen näköaistiin ja pään liikkeisiin.

[2]Kuva: https://commons.wikimedia.org/wiki/File:Nao_Robot_%28Robocup_2016%29.jpg (Lisenssi CC0 1.0)

Kuten edellä mainittujen, uusinta teknologiaa sisältävien robottien tilasta voidaan päätellä, ei pystytä vielä valmistamaan sosiaalista terapiarobottia, joka pystyisi olemaan luotettavasti ja autonomisesti vuorovaikutuksessa lapsen kanssa. Teknologia on kuitenkin kehittymässä siihen suuntaan, että joidenkin arvioiden mukaan lähes täysi autonomia voi tulevaisuudessa olla mahdollista [33, 39].

Vanhusten avustaminen on tärkeä, kehittyvä käyttötapa sosiaalisille roboteille. Martinez-Martin, Escalona ja Cazorla [34] kertovat selvityksessään sosiaalisista avustajaroboteista (eng. Social Assistance Robot, SAR). Tällä hetkellä on olemassa SAR:eja, jotka esimerkiksi auttavat muistisairaita muistuttamalla lääkkeiden ottamisesta, tarkistamalla hellan sammuneen, tai soittamalla hätätilanteessa apua. Laajempia, älykoti-tyylisiä ratkaisuja on myös kehitetty ongelmaan robotin rinnalle, mutta ne eivät ole saavuttaneet suurta suosiota työlään asennusprosessin ja kalliin hinnan takia [34].

Kuva 3. *Hyljettä muistuttava PARO-terapiarobotti*[3]

Toinen esimerkki avustajarobotista on kuvassa 3 esitetty, tekoälyn avulla käyttäjäänsä mukautuva PARO. Robotti reagoi älykkäästi kosketukseen ja puheeseen. Tämän paljon tutkitun terapiarobotin avulla on hoidettu vanhusten yksinäisyyttä ja kroonista kipua suurimmaksi osaksi erinomaisin tuloksin [40].

SAR:eihin ja niiden tulevaisuuteen liittyy erilaisia haasteita. Tällä hetkellä suurimpia ongelmia robottien käyttämisessa vuorovaikutuksessa vanhustenhoidossa ja terapiakäytössä on niiden autonomisuuden puute. SAR, joka tarvitsee ihmisen, kuten hoitajan tai terapeutin valvomaan tai jopa osittain ohjaamaan sitä, vaikuttaa potilaalle enemmän työkalulta kuin hoitajan korvikkeelta. SAR:ien laajempaa käyttöönottoa helpottaisi, jos käsitys roboteista mahdollisena avustajana tai kumppanina yleistyisi. Uusinta teknologiaa olevissa roboteissa on ilmennyt ongelmaksi myös vanhempien sukupolvien vähäinen kokemus nykyteknologioiden, kuten kosketusnäyttöjen käytöstä. Muita kehitysalueita ovat robottien ulkonäön ja rakenteen parantaminen, jonkin tasoisen keinotekoisen tunneälyn kehittäminen, sekä lisätutkimuksen tekeminen SAR:ien pitkäaikaisesta käytöstä. [33]

[3]Kuvan omistaja Theron Trowbridge: https://www.flickr.com/photos/therontrowbridge/4261112915 (Lisenssi CC BY-NC 2.0)

3. ROBOTTIKÄSIEN TEKNOLOGIA

Kun roboteilta halutaan vaivatonta ja luonnollista vuorovaikutusta, tulisi niiden käsillä tekemistä eleistä ja liikkeistä tehdä sen mukaisia. Tyydyttävän tasoista liikettä voitaisiin lähteä toteuttamaan käyttämällä erilaisia teknologioita, kuten erilaisia sensoreita, ohjelmistoja ja algoritmisovelluksia.

3.1. Sanaton vuorovaikutus

Sanatonta kommunikaatiota pidetään tärkeämpänä kuin sanallista ja sen arvioidaan kattavan enemmän kuin 60% ihmisten kommunikaatiosta. Sanaton kommunikaatio yleensä luokitellaan muutamaan erilliseen, mutta toisiinsa liittyviin luokkiin. Näitä ovat proksemiikka, haptiikka, kronemiikka, vokaliikka ja presentaatio. Näistä pääasiassa pelkästään kinesiikka, proksemiikka, haptiikka ja kronemiikka voidaan suoraan hyödyntää robotin liikekommunikaation kehittämisessä. Kinesiikka tarkoittaa robotin käsien ja kasvojen liikettä, proksemiikka robotin ja ihmisen etäisyyden vaikutusta vuorovaikutukseen, haptiikka kosketuksen vaikutusta vuorovaikutuksessa ja kronemiikka kommunikaation tempon vaikutusta. [6]

Varsinkin teollisuuden alalla työskentelevien kobottien kanssa sanaton vuorovaikutus ja sen ymmärtäminen on äärimmäisen tärkeää. Tehdastyöntekijöiden tulee pystyä käskyttämään robottia sanattomasti, ja tällöin robotin pitää ymmärtää sanatonta kommunikaatiota ja aikaisempien tehtäviensä muistamista ymmärtääkseen ympäristöllisen kontekstin ja päättää siitä mitä siltä halutaan. Virheet sanattoman kommunikaation ymmärtämisessä voivat johtaa vaaratilanteisiin. [28]

3.1.1. Haptiikka

Ihmiset käsittelevät vaivattomasti samanaikaisesti useita erilaisia haptisia ärsykkeitä. Samaa ei voi sanoa tämänhetkisistä roboteista. Robotin ja ympäristön haptinen vuorovaikutus on esimerkiksi konenäköön verrattuna toistaiseksi melko tutkimaton robotiikan ja koneaistimisen aihealue, eikä järjestelmien suunnittelun kannalta tärkeisiin kysymyksiin, kuten "Kuinka esittää datana esineen haptisia ominaisuuksia, kuten esimerkiksi lämpötilaa, jäykkyyttä ja materiaalin laatua?"ole vielä löydetty selkeitä vastauksia. [41]

Robotteihin sovellettavissa olevat tuntoaistiin liittyvät ratkaisut ovat siis vielä toistaiseksi alkutekijöissään, mutta erilaisia lähestymistapoja on esitetty. Yksi tapa kerätä esineen muotoon liittyvää haptista dataa on anturilla, jossa on useita painetta tunnistavia yksiköitä, joilta saadaan yhdessä muotoa kuvastava matriisi ja usean tällaisen matriisin avulla voidaan luoda kolmiulotteinen kuva esineen tai ympäristönpiirteen muodosta [42, 41]. Myös HRI:in sovellettavia, pehmeyttä tunnistavia ratkaisuja on ilmentynyt viime vuosina. Esimerkkinä tästä Qiu et al. [43] kehittämä ihmisihoa jäljittelevä jäjestelmä, joka kykenee aistimaan esineen pehmeyttä. Edellä esitetyt tämänhetkiset haptiikan ratkaisut eivät sellaisenaan mahdollista ihmisen ja autonomisen robotin erityisen turvallista ja tehokasta keinotekoiseen tuntoaistiin perustuvaa fyysistä vuorovaikutusta.

3.1.2. Liikkeen antropomorfisuus

Robotin liikkeiden antropomorfisuudella eli ihmisenkaltaisuudella voidaan tehdä vuorovaikutuksesta ja yhteistyöstä ihmisen kanssa luontevampaa. Ihmisen on helpompi ymmärtää ja ennakoida robotin toimintaa sen liikkeiden muistuttaessa ihmisen liikkeitä, mikä puolestaan sujuvoittaa vuorovaikutusta huomattavasti [44]. Robottikäden antropomorfisen liikkeen suunnittelussa käytetään usein erilaisia käänteiskinematiikan sovelluksia [45]. Käänteiskinematiikalla tarkoitetaan lyhyesti sanottuna kinemaattisten yhtälöiden käyttöä haluttuun pisteeseen johtavan robottikäden liikkeen laskennassa.

Yksi esimerkki robottikäsien ihmisenkaltaisten liikkeiden suunnittelun ratkaisusta on Gulletta et al. [5] kehittämä Human-like Upper-limb Motion Planner (HUMP), joka on antropomorfisia kädenliikkeitä tuottava algoritmi. HUMP-algoritmi ottaa huomioon myös törmäyksien välttämisen antropomorfisen liikkeen aikana, mikä on ongelmana saanut melko vähän huomiota osakseen aikaisemmin. Gulletta et al. [46] ovat jälkeenpäin parantaneet HUMP-algoritmia ihmisiltä kerätystä liikedatasta tehtävällä jatkuvan oppimisen periaatetta käyttävällä koneoppimisalgoritmilla. Robottien yläraajojen antropomorfisten liikkeiden ratkaisut ovat tällä hetkellä pääsääntöisesti hyvin laskennallisesti vaativia, mikä tekee niiden käytännön sovelluksista kalliita ja usein vaikeita implementoida.

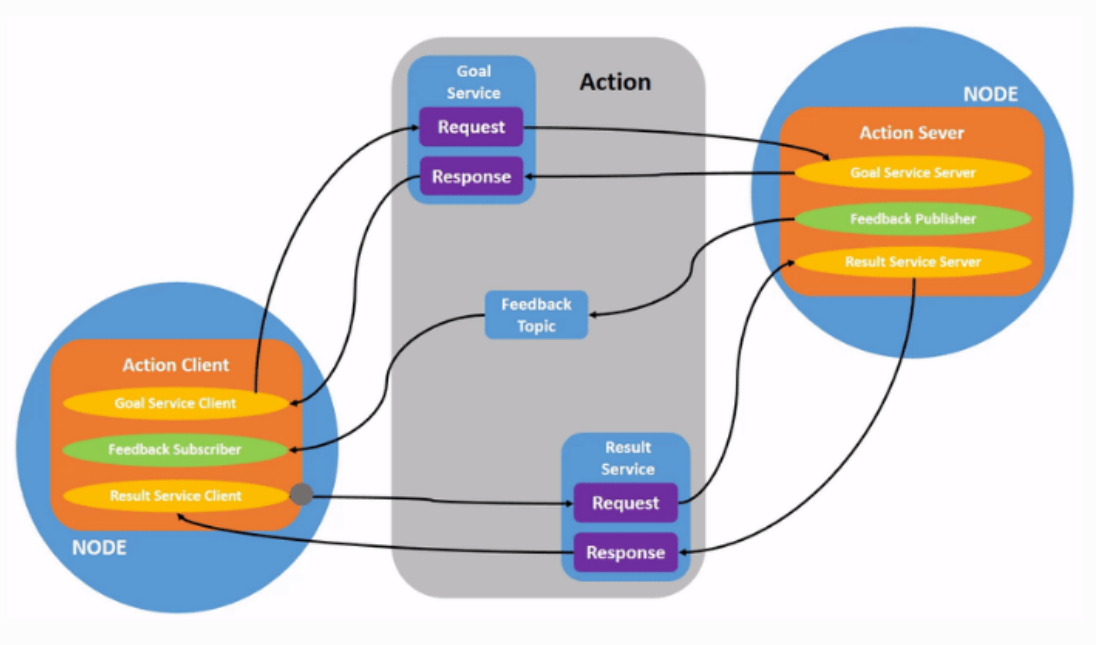
Kun lähdetään toteuttamaan robotin kättä ja kyynärvartta, tarvitaan fyysinen runko, sekä ohjelmisto, joka ohjaa runkoa. Nämä molemmat osat heijastavat toisiaan kompleksisuudellaan. Jos fyysinen runko sisältää useita niveliä ja moottoreita, pitää sitä ohjaavan ohjelmiston olla tarpeeksi monimutkainen, jotta se kykenee ohjaamaan runkoa ilman, että osat osuvat toisiinsa ei-halutulla tavalla tai liikkuvat mahdottomiin asentoihin.

3.2. Robot Operating System 2

Vuonna 2007 Stanfordin yliopiston tekoälylaboratorion projektina alkunsa saanut Robot Operating System (ROS) on kokoelma ohjelmistoja, joka sisältää muun muuassa ajureita ja työkaluja robottisovellusten kehittäjille [47]. Robot Operating System 2 (ROS2) on ROS-ohjelmiston myöhemmin julkaistu versio, johon on tehty muun muuassa teollisuuskäytölle tarpeellisia parannuksia, kuten reaaliaikaisen ohjelmoinnin (eng. real-time programming) mahdollistaminen [48].

3.2.1. Solmut

Yksinkertaisesti esitettynä, ROS2 järjestelmä koostuu solmuista (eng. node), jotka kommunikoivat keskenään yhteisen alustan kautta, muodostaen robottikokonaisuuden, kuten kuvassa 4. Yksittäisessä solmussa voi olla esimerkiksi sensori, servomoottori tai useita moottoreita ja sensoreita. [48, 49]



Kuva 4. *ROS2 solmujen välinen toimintamalli. (Kuvaa muokattu)*[4]

Solmuja ohjaavat joko Python tai C++ kielellä kirjoitetut koodipalvelimet. Palvelimet on tehty helposti muokkautuviksi niin, että yhtä palvelinta voi kutsua useampi toinen palvelin, jolloin palvelimia yhdisteltäessä voidaan havainnoida helposti mahdolliset virhetekijät, sekä rakentaa suurempi kokonaisuus

3.2.2. Colcon

ROS2 koostuu useasta suuremmasta ja pienemmästä palvelinkokonaisuudesta. Usein näitä palvelimia on niin paljon, että yksilöllinen hallinta pakettien sisäisistä ja ulkoisista vaatimuksista on hyvin monimutkainen. Tämän hallinnan helpottamiseksi ROS2 kirjasto käyttää Collective Construction (colcon) hallintaohjelmaa. Colcon automatisoi ohjelmien kompilaation, järjestelee ne oikeaan järjestykseen toimivuuden takaamiseksi, sekä valmistelee ympäristön missä ohjelmia voi ajaa.

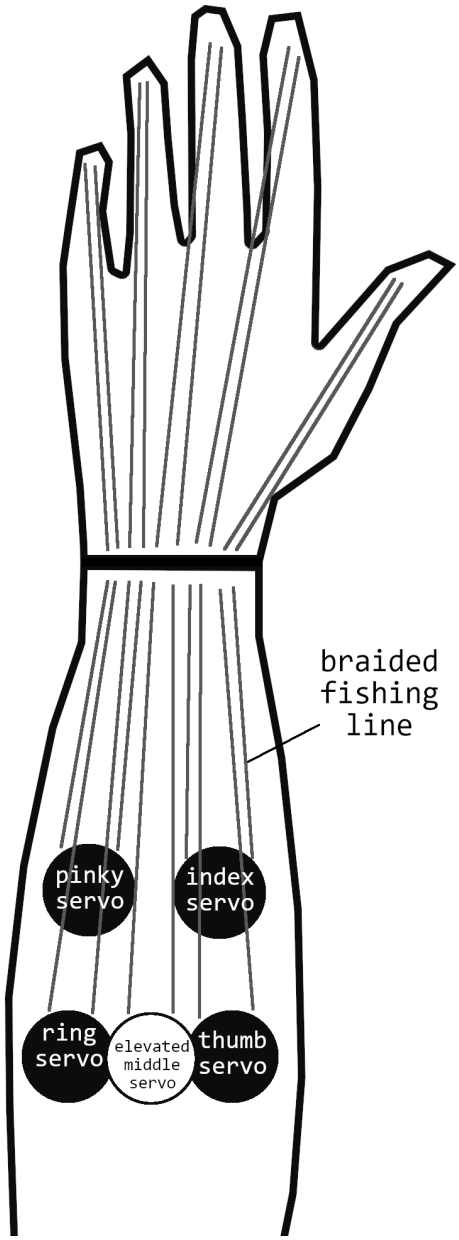
3.3. InMoov

InMoov on alun perin taiteilija Gaël Langevinin vuonna 2012 luoma avoimen lähdekoodin omaava robotiikan kehitys- ja oppimisalusta. InMoov perustuu helposti 3D-tulostettaviin, korkeintaan 12 kuutiosenttimetrin kokoisiin rungon osiin [50]. Tämän ansiosta InMoov on suosittu harrastajien keskuudessa sen saatavuuden ja halpuuden vuoksi.

Kuvassa 5 esitetty InMoovin ratkaisu käden ja sormien ohjaamiseen on yksinkertainen, joten sen käyttö robottikäsiratkaisussa on suhteellisen helppoa. Jokaisen sormen liikettä ohjaa vain yksi moottori, joten sormien liikkuminen rajoittuu

[4]Kuva ROS2 Project: https://docs.ros.org/en/foxy/_images/Action-SingleActionClient.gif (Lisenssi Creative Commons Attribution 4.0 International)

niiden koukistamiseen ja suoristamiseen. Robotin osat löytyvät sekä 3D-tulostettuna, että digitaalisena kopiona toimivana kirjastona. Fyysisestä robotista tulostettuna on työn teon hetkellä vain pää ja käsivarsi, mutta digitaalinen kopio sisältää robotin koko yläkehon. Robotin DT:tä voidaan simuloida käyttämällä ROS2-kirjaston Rviz simulaattoria.



Kuva 5. *Illustraatio InMoovin sormienohjausratkaisusta*[5]

3.4. Rviz

Rviz on alunperin ROS:ille tehty simulaattori, jonka avulla voidaan luoda digitaalisia versioita roboteista ja testata koodipakettien toimivuutta. Rviz simuloi 3D-mallin

[5]Kuvan omistaja Juho Bruun (Lisenssi CC BY 4.0)

robotista ja kaikista sen osista. Robottia pystyy liikuttamaan haluttuihin asentoihin ja se voidaan laittaa tekemään kaikki mitä rakennettu versio voisi tehdä. Rviz käyttää Unified Robot Description Format (URDF) tiedostoja muodostaaksen robotista toimivan mallin. Toteutuksessa URDF-tiedostot on muodostettu käyttäen Extensive Markup Language Macro-kieltä (xacro).[51]

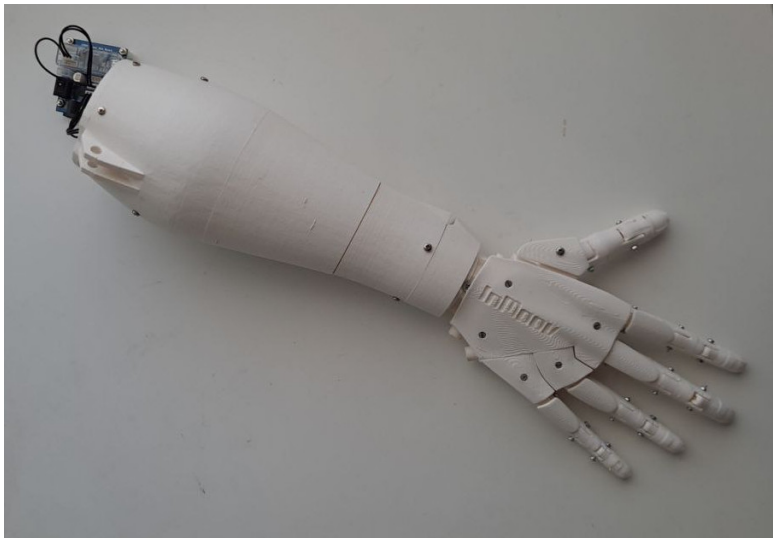
3.4.1. Extensive Markup Language Macros

URDF-robotti mallit koostuvat suuresta määrästä Extensive Markup Language (XML) kielellä kirjoitetuista robotin osien kuvauksista. Kuvausten lukeminen ja -korjaaminen on raskasta niiden vaikealukuisuuden ja suuren määrän vuoksi, minkä takia URDF-tiedostot kasataan helposti luettaviksi Extensive Markup Language Macro (xacro) tiedostoiksi. Näiden muokkaaminen ja lukeminen on paljon käyttäjäystävällisempää, kuin puhtaiden URDF-tiedostojen lukeminen.[52]

Xacro tiedostot sisältävät tietoa robotin osien koosta ja -muodosta. Xacro tiedostossa palanen XML kieltä kasataan yhdeksi isoksi makroksi, johon voidaan viitata pelkällä makron nimellä muissa tiedostoissa. Ne voivat myös sisältää ehtolauseita esimerkiksi siitä, että käytetäänkö robotissa simuloituja osia vai todellisia osia. Myös osittainen rakenne on mahdollinen niin, että robotista on fyysisesti kasattu vain osa, mutta ROS2 rakentaa tiedostojensa avulla virtuaalisen robotin muistiinsa kokonaisena. Tällöin robotin muitakin osia voi ohjata, vaikka niitä ei fyysisesti olisi olemassa. [52]

4. TOTEUTUS

Projektin fyysinen osuus perustuu robotiikan kehitysalustaksi suunniteltuun 3D-tulostettavaan avoimen lähdekoodin InMoov-robottiin. Täysin koottu käsi esitetty kuvassa 6. Ero alkuperäiseen InMooviin on se, että tässä implementaatiossa käytettiin TowerPro-MG996 ja -MG946r servojen sijaan Dynamixel XL-320 -servomoottoreita. Ohjelmistopuolella käytetään InMoovin MyRobotLabin sijaan ROS2:ta.



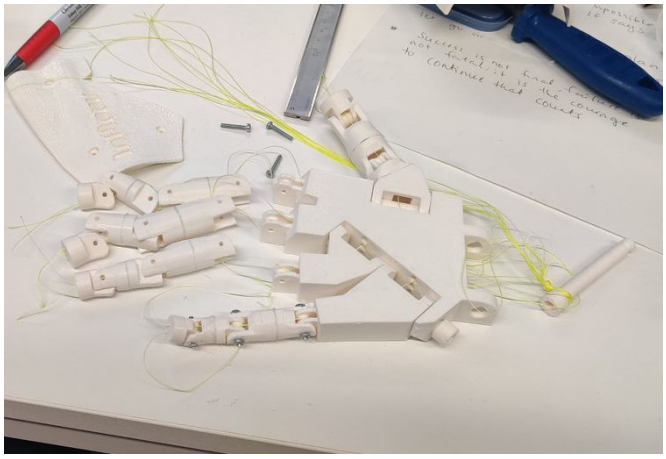
Kuva 6. *Projektissa koottu käsi kokonaisuudessaan*[6]

4.1. Kokoaminen

Käden rungon osat tulostettiin 3D-tulostimella. Suurin osa osien 3D-malleista otettiin valmiina InMoovin STL-galleriasta [53] osioista ’Right-Hand’, ’Forearm-and-Servo-Bed’ ja ’Rotation-Wrist’. Dynamixel XL-320 servomoottorit ovat kooltaan ja kiinnityksiltään erilaisia kuin InMoovin tarjoamiin 3D-malleihin valmiiksi sopivat TowerPro MG946R servomoottorit. Osia piti siis muuttaa XL-320 servojen kiinnityksiin sopiviksi. Osien muokkaaminen toteutettiin Autodesk Fusion 360 3D-mallinnusohjelmalla [54].

Kokoaminen aloitettiin sormista ja kämmenestä. Jokainen robotin sormista on kiinnitettynä servomoottooriin kahdella paksulla siimalla 7. Kaikki kymmenen siimaa kulkevat kämmenen ja ranteen sisältä kyynärvarren sisälle, missä servomoottorit sijaitsevat. Siimojen päät ovat solmittu yhteen ja kiinnitetty servomoottoreissa oleviin taljoihin. Näin saadaan aikaan rakenne, jossa servomoottoria liikuteltaessa sitä vastaava sormi koukistuu ja suoristuu.

[6]Kuvan omistaja Juho Bruun (Lisenssi CC BY 4.0)



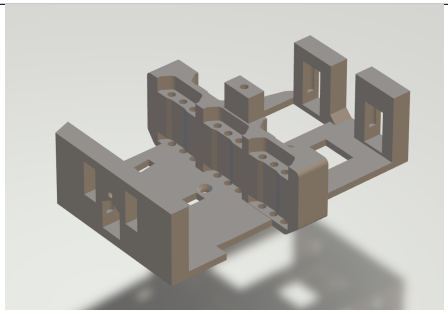
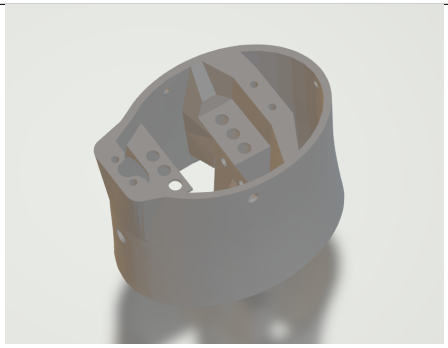
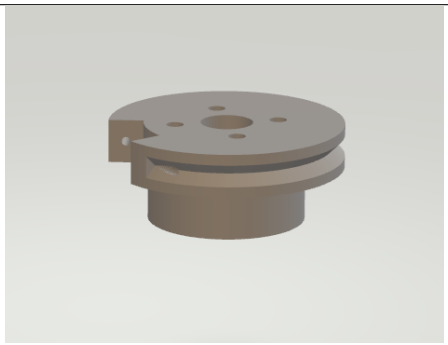
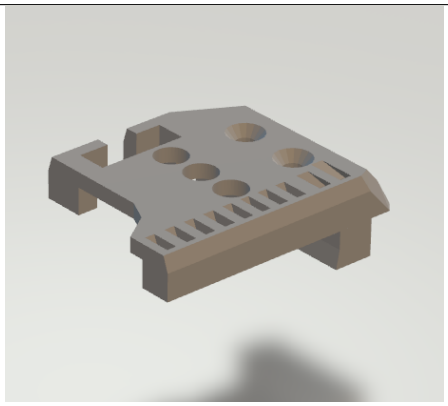
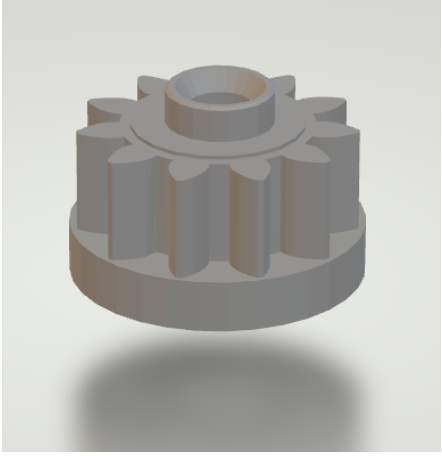
Kuva 7. *Kämmen kokoamisvaiheessa*[7]

Robotin ranteessa on hammasratas, jota liikutellaan yhdellä Dynamixel XL-320 servomoottorilla. Täten robotin ranne voi liikkua vain yhdellä akselilla. Ranteen servomoottori ylikuormittuu hyvin herkästi, jolloin servomoottorin ohjelma menee vikatilaan. Ongelmaa voisi lieventää voitelemalla hammasrattaat ja rannenivelen liukastavalla aineella tai viilaamalla hammasrattaita.

Käden eri osat liitettiin toisiinsa käyttäen ruuveja ja epoksiliimaa. Näin käden rakenteesta saatiin luja ja pysyvä, mutta myös helposti purettava, jotta rikkinäisten osien vaihto olisi helppoa. Kokoamisen aikana jotkin osat paljastuivat yhteensopimattomiksi uusien servojen kanssa, jolloin niitä jouduttiin muokkaamaan tai tulostamaan uudelleen. Muokatut osat ovat näkyvillä taulukossa 1.

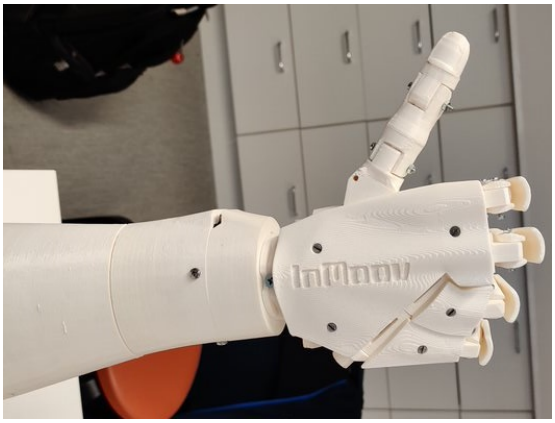
[7]Kuvan omistaja Aleksi Tuovinen (Lisenssi CC BY 4.0)

Taulukko 1. *Taulukko kaikista implementaatiossa käytetyistä InMoovin osien muokatuista versioista*[9]

Osan nimi	**Kuva**	**Kuvaus muutoksista**
ServoBed		InMoovin suosittelemille servoille tehdyt kiinnitysreiät on poistettu. Keskiosaa on levennetty kolmen XL-320 servon mahduttamiseksi. Keskiosaan muokattu sopivat kiinnityspaikat, joihin servot voidaan kiinnittää ruuveilla.
RotaWrist1		Lisätty sopivat tukialustat XL-320 servoa varten. Alustaan tehdyt reiät mahdollistavat servon kiinnittämisen ruuveilla ja epoksiliimalla.
ServoPulley		Taljan keskiosaa on kohotettu, jotta kolme servoa taljoineen mahtuvat vierekkäin.
CableHolderWrist		Lisätty XL-320 servojen kiinnittämiseen soveltuvat reiät.
WristGear		Tehty hammasrattaan servoon kiinnitettävästä pohjasta korkeampi, jotta saadaan kompensoitua servomallien korkeusero.

4.2. Käden ajaminen

Käden ajaminen tapahtuu ROS2 ohjelmiston päälle rakennetulla ohjelmalla, joka kirjoitettiin Python-ohjelmointikielellä. Jotta kättä pystyttiin ohjaamaan, piti valmiiksi saatuun koodipakettiin määrittää itse käden nivelet, niveliä käyttävä solmu, kyseistä solmua ajavava toimintapalvelin, sekä toimintapalvelinta käskyttävä asiakaspalvelin. Asiakaspalvelinta pystyy ajamaan itsekseen tai sen voi alistaa toiselle ohjelmalle, jolloin sitä voi käyttää osana suurempaa kokonaisuutta. Itsenäisesti ajettuna käyttäjä voi syöttää kädelle komentoja. Käsi kykenee yli kymmeneen erilaiseen toimintoon, kuten etusormella osoittamiseen, peukalon näyttämiseen (kuvassa 8), tarttumiseen ja ’kivi, paperi, sakset’ -pelin pelaamiseen.



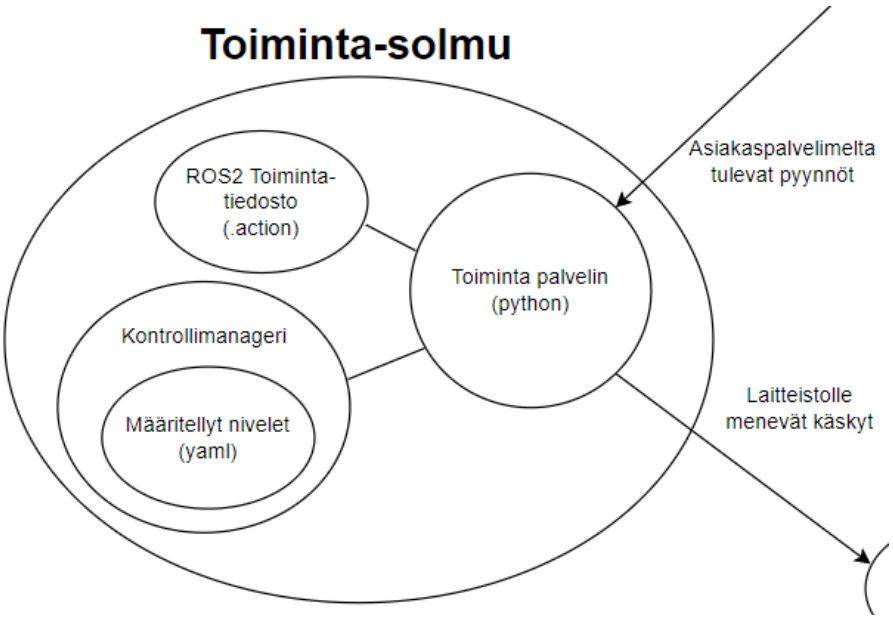
Kuva 8. *Yksi käteen ohjelmoiduista eleistä*[10]

Robotin ohjaamiseen käytettävä ROS2-ohjelmisto saatiin valmiina pakettina kurssiohjaajien toimesta. Paketti pystytettiin Vagrant-ohjelmalla, joka loi paketista virtuaalikoneen, missä oli kaikki tarvittava robotin kehitykseen ja käyttämiseen. Tämä mahdollisti nopean ja helpon kehityksen usealla eri tietokoneella.

Toteutuksessa robotille annettavat komennot kulkevat kahden ROS2-solmun ja niiden palvelimien lävitse. Komentoja ajaa rajapinnassa oleva toimintapalvelin, joka vastaanottaa käskyjä, tarkistaa niiden oikeanlaisuuden vaadittaviin toimintakriteereihin ja välittää käskyn eteenpäin robotin laitteistolle. Kuvassa 9 on havainnollistettu toimintasolmun rakennetta. Laitteistosta palautuvan datan toimintapalvelin välittää takaisin toiminnan antaneelle asiakasohjelmalle.

Toimintatiedostoon (eng. action-file) määritellään komennon vaatimat kriteerit. Ensimmäinen osa on maali, johon komennolla pyritään, sekä tiedot siitä, mitä toimintapalvelin palauttaa asiakasohjelmalle toimintoa suorittaessa ja kun se on päässyt maaliin. Asiakasohjelma kykenee sitten käyttämään näitä tietoja hyödykseen jatkokomennoille tai keskeytyksiin, jos data poikkeaa halutusta. Käden toimintaa varten muuttujia on runsaasti.

[9]Taulukon kuvien omistaja Juho Bruun (Lisenssi CC BY 4.0)
[10]Kuvan omistaja Ville Hokkinen (Lisenssi CC BY 4.0)



Kuva 9. *Toimintasolmun rakenne*[11]

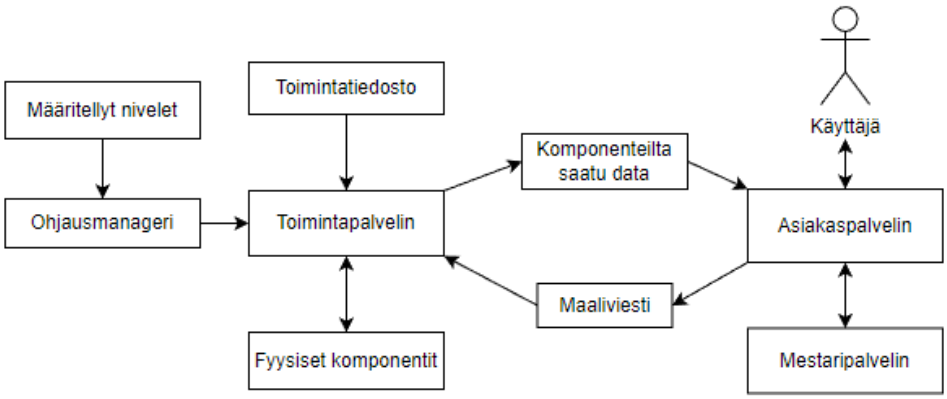
```
Käden toimintatiedosto:
    # Maali
    trajectory_msgs/JointTrajectory trajectory
    JointTolerance[] path_tolerance
    JointTolerance[] goal_tolerance
    builtin_interfaces/Duration goal_time_tolerance
    ---
    # Loppu
    int32 error_code
    int32 SUCCESSFUL = 0
    int32 INVALID_GOAL = -1
    int32 INVALID_JOINTS = -2
    int32 OLD_HEADER_TIMESTAMP = -3
    int32 PATH_TOLERANCE_VIOLATED = -4
    int32 GOAL_TOLERANCE_VIOLATED = -5
    string error_string
    ---
    # Palaute
    std_msgs/Header header
    string[] joint_names
    trajectory_msgs/JointTrajectoryPoint desired
    trajectory_msgs/JointTrajectoryPoint actual
    trajectory_msgs/JointTrajectoryPoint error
```

[11] Kuvan omistaja Aleksi Tuovinen (Lisenssi CC BY 4.0)

Toinen tärkeä määritelmä on käden käyttämät nivelet. Nivelet määritellään erilliseen YAML (YAML Ain't Markup Language) tiedostoon. Kyseisessä tiedostossa on nimettynä nivelet ja niiden nimet, sekä nivelten pääohjaaja (eng. controller). Valmiiksi saadussa paketissa vain pään nivelet olivat valmiiksi määriteltyinä ja nämä määriteltiin itse erillisen käden ohjaajan alaisuuteen. Toimintapalvelimen saadessa käskyn, se tarkistaa mitä niveliä komennossa on tarkoitus käyttää, varmistaa, että nivelet ovat oikean ohjaajan alaiset ja sitten liikuttaa niveliä ohjaajan kautta.

Kun käyttäjä lähettää asiakaspalvelimen kautta komennon robotille, palvelin luo ensin toimintapalvelimelle lähtevän maaliviestin. Maaliviestiin määritellään käytettävät nivelet, sekä pisteet, joihin kyseisten nivelten tulee liikkua. Kun viesti on koottu, lähetetään se toimintapalvelimelle. Toimintapalvelin lähettää vastauksen takaisin jossa komento hyväksytään tai hylätään. Mikäli komento on hylätty, ei ohjelma tee mitään. Jos komento on hyväksytty, jää asiakaspalvelin kuuntelemaan toimintapalvelinta. Kun toiminto on suoritettu, asiakaspalvelin ilmoittaa siitä käyttäjälle. Tätä toimintamallia esittää kuva 10.

Oikeanlaisen maaliviestin saadessaan toimintapalvelin lähettää viestin eteenpäin Dynamixel-workbench palvelimelle. Dynamixel-workbench palvelin kääntää komennot servomoottoreille sopivaksi konekieleksi, ja raportoi takaisin toimintapalvelimelle, miten servomoottoreiden liike etenee. Riippuen Dynamixel-solmulle asetetusta taajuudesta, sormet liikkuvat joko sulavasti tai kankeasti. Sormien nopeutta ei ole mahdollista säätää. Kontrollimanagerille on spesifioitu viisi eri niveltä, joita kaikkia ajetaan yhtä aikaa yhdellä komennolla.



Kuva 10. *Toimintapalvelimen ja asiakaspalvelimen rakenne.*[12]

Yllä mainitut tiedostot ja paketit kasataan ajamalla colconin kompilaatio-ohjelma, joka kasaa kaikki paketit yhdeksi isoksi toimivaksi pakkaukseksi. Colconin ansiosta ympäristö pystyy käyttämään useita paketteja yhtä aikaa, ja ne kasataan niin, että tärkeimmät paketit kompiloidaan ensimmäiseksi, jolloin myöhemmät paketit voivat poimia vaatimansa paketit suoraan aiemmin kompiloiduista. Esimerkiksi kurssipaketissa on dynamixel-paketti, joka hoitaa kommunikoinnin Dynamixel-servomoottoreiden kanssa, face_tracker-paketti, joka ajaa robotin katsetta,

[12]Kuvan omistaja Aleksi Tuovinen (Lisenssi CC BY 4.0)

head_gestures-paketti, joka hoitaa pään liikkeet, inmoov_description paketti, joka sisältää tiedot robotin ulkomuodosta, sekä tärkeimpänä robot-paketti, joka sisältää robotin pääohjauskirjaston.

Asiakaspalvelimen toimivuus tarkistettiin ROS2-kirjaston mukana tulevalla RViz-simulaattorilla. Kun toimintapalvelimelle annetaan komento, Rviz-ohjelma poimii komennon ja näyttää simulaattorissa, mitä kyseinen komento tekee. Simulaattorilla pystyy kuitenkin myös mahdottomiin liikkeisiin, kuten kääntämään nivelen akselinsa ympäri. Tarkkoja toimivuuksia simulaattorilla ei voitu täten tarkistaa.

Simulaattorin ongelmien vuoksi servomoottoreiden kääntymisasteet kirjattiin ylös ilman, että niitä ajettiin vielä ROS2-ohjelmistolla, jotta käden servomoottorit tai siimat eivät vahingoittuisi. Tähän mittaamiseen käytettiin Dynamixelin omaa DYNAMIXEL Wizard-ohjelmaa, joka näyttää missä kulmassa servomoottorit ovat. Tärkeimmät kulmat olivat ääripäät, eli milloin robotin sormi oli tiukasti suljettuna tai täysin avoinna. Nämä pisteet ovat käytössä asiakaspalvelimella ääripisteinä. Jos sormi taittuu näitä pisteitä pitemmälle, lopettaa palvelin komennon suorittamisen.

4.3. Asiakaspalvelin

Asiakaspalvelin on rakennettu Python-ohjelmointikielellä. Sen tarkoituksena on toimia käyttäjän tai muun robottiverkoston linkkinä toimintapalvelimelle ja sen ohjaamille fyysisille komponenteille. Asiakaspalvelin voi ottaa komentoja suoraan ihmiskäyttäjältä tai toiselta ROS2-solmulta, sekä sen voi liittää osaksi isompaa solmupalvelin kokonaisuutta. Asiakaspalvelin vastaanottaa dataa toimintapalvelimelta, kuten servojen kääntymisasteet, ja se pystyy käyttämään dataa arvioimaan ohjelman suoritusta, sekä esittämään sen käyttäjälle.

Asiakaspalvelin on muodoltaan Python-luokka (eng. class), joka käyttää pohjanaan ROS Client Library (rclpy) ohjelmistopakettia. Ohjelmistopakettia tarvitaan, jotta ROS2 kykenee ymmärtämään asiakaspalvelimen yhtenä osana sen kokonaisuutta, ja että muut solmut voivat helposti hallita sitä, sekä kommunikoida sen kanssa. Luokka tarvitsee toimiakseen myös ROS2-kirjastosta kaksi viestiluokkaa, joita se käyttää rakentaessaan toimintapalvelimelle lähtevän komennon, sekä toimintapalvelimen hyväksymän toiminnon (eng. action), jotta se kykenee viestimään kyseisen palvelimen kanssa. Luokka tarvitsee myös Pythonin sisäisiä random- ja time-kirjastoja, joita se käyttää pelatessaan 'kivi, paperi, sakset' -peliä . Asiakaspalvelimeen on tallennettu muistiin sen yhteys toimintapalvelimen kanssa, sen käyttämät komennot ja niiden asennot, sekä datan liikenteen lukevan lokiohjelman. Toimintapalvelin on nimetty käden ohjaimen ja tehtävän toiminnon mukaan.

Kun asiakaspalvelinohjelma käynnistetään itsenäisesti, se aloittaa toimintansa yhdistämällä toimintapalvelimelle ja aloittamalla lokiohjelman. Sen jälkeen asiakaspalvelin siirtyy odotustilaan. Odotustilassa palvelin odottaa käskyjä käyttäjältä. Käyttäjä voi syöttää käskyjä komentoliittymän (eng. command line interface, CLI) kautta kädelle kirjoittamalla komennon sille varattuun paikkaan[13]. Jos komento on väärä tai viallinen, tulostuu tästä käyttäjälle kuvan 11 mukainen ilmoitus. Samalla

palvelin listaa ohjelman käyttäjälle kaikki saatavilla olevat komennot. Ohjelman suoritus loppuu käyttämällä **quit** komentoa.

```
vagrant@vagrant-ros:/workspace$ python3 client/hand_action_client.py
Input command: wrong
Command not recognized, available commands are
['open', 'fist', 'scissors', 'point', 'thumbs_up', 'grasp', 'pen_grasp', 'hard_r
ock', 'rps', 'trial']
and quit.
```

Kuva 11. *Komentoliittymä kun käyttäjä syöttää väärän komennon.*

Hyväksyttävän komennon saadessaan asiakaspalvelin lähettää komennon eteenpäin toimintapalvelimelle **send_goal** funktion kautta. Funktio luo ensin toimintatiedoston mukaisen maalin lähteväksi viestikseen. Viestiin lisätään käden ohjaimen muistissa olevien nivelten nimet, sekä pisteet joihin kyseiset nivelet halutaan. Pisteet lisätään **JointTrajectoryPoint** nimiseen luokkaan rclpyn kirjastosta. Luokkaan pisteet menevät listana, jossa on jokaisen nivelen haluttu piste. Jos pistettä ei ole ilmoitettu kaikille nivelille, ei toimintapalvelin hyväksy viestiä. Pisteluokkaan lisätään myös aika, jonka toiminnon oletetaan kestävän. Aika ilmoitetaan rclpyn omalla **Duration** luokalla.

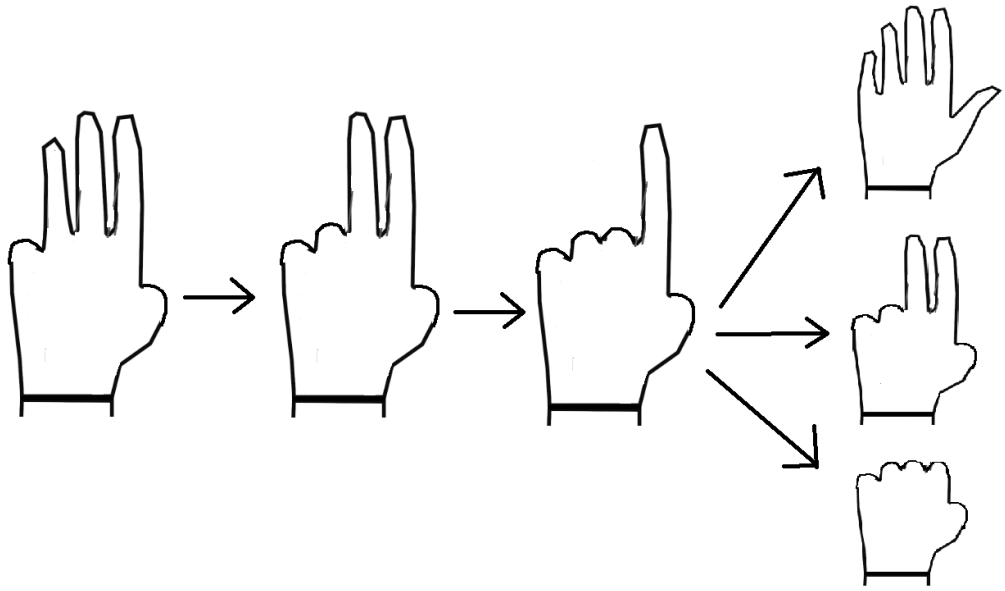
```
#Send_goal funktio
    def send_goal(self, action):
        goal_msg = FollowJointTrajectory.Goal()
        goal_msg.trajectory.joint_names = ["r_thumb_joint
            ", "r_index1_joint", "r_middle1_joint", "
            r_ring_joint", "r_pinky_joint"]
        goal_msg.trajectory.points = []
        point = JointTrajectoryPoint()
        point.positions = action
        dur = Duration()
        dur.sec = 1
        point.time_from_start = dur
        goal_msg.trajectory.points.append(point)

        self.get_logger().info(f"Sending request")
        self._action_client.wait_for_server()
        self._send_goal_future = self._action_client.
            send_goal_async(goal_msg)
        self._send_goal_future.add_done_callback(self.
            goal_response_callback)
```

Komennon lähettämisen jälkeen asiakaspalvelin jää odottamaan vastausta toimintapalvelimelta. Toimintapalvelin ilmoittaa asiakaspalvelimelle hyväksyttiinkö vai hylättiinkö komento. Jos komento hylättiin, asiakasohjelma ilmoittaa tiedon käyttäjälle, ja jää odottamaan seuraavaa komentoa. Toimintapalvelimen hyväksyessä komennon jää asiakaspalvelin seuraamaan toimintapalvelimelta tulevaa dataa. Toimintapalvelimen ilmoittaessa komennon päättymisestä asiakaspalvelin tiedottaa käyttäjää komennon päättymisestä ja jää odottamaan seuraavaa komentoa. Jos

[13] Komentojen syöttämisen tarkat ohjeet ja kaikki saatavilla olevat komennot löytyvät liitteestä 1.

toimintapalvelin ei palautakkaan mitään tietoa, mutta komento on hyväksytty, olettaa asiakaspalvelin komennon suoritetuksi ja jää odottamaan seuraavaa komentoa.

Peruskomennoista poiketen komennot **rps** ja **trial** eivät suoraan syötä yhtä tiettyä ajettavaa komentoa toimintapalvelimelle. Kun käyttäjä syöttää asiakaspalvelimelle käskyn **rps**, käsi aloittaa automaattisesti kivi-paperi-sakset pelin pelaamisen. Pelin pelaamiseen sisältyy neljän eri komennon lähettäminen vuoron perään toimintapalvelimelle. Koska kyseessä on pelkkä käsi, eikä kyynärvartta tai suuta, joka laskisi kolmesta alaspäin ole, käsi pelaa peliä niin että se ensin näyttää käyttäjälle kolme sormea pystyssä, sitten kaksi, ja lopuksi yksi. Kuva 12 ilmentää tätä toimintaa. Tämän jälkeen se arpoo pythonin random kirjaston avulla yhden kolmesta liikkeestä, avoimesta kädestä, joka edustaa paperia, nyrkistä, joka edustaa kiveä, ja voittomerkistä, joka toimii saksina. Kun tämä on toteutettu, siirtyy asiakaspalvelin jälleen odottamaan käskyä käyttäjältä.



Kuva 12. *Kivi-paperi-sakset pelin toimintajärjestys*[14]

Trial-komento antaa käyttäjän määrittää kulmat, joihin haluaa jokaisen sormen siirtyvän. Funktio pyytää yksi kerrallaan jokaiselle sormelle halutun asteluvun. Käyttäjän kirjoittama asteluku lisätään asiakaspalvelimelle lähtevään komentoon pisteenä, ja kun kaikkien viiden sormen kulma on määritetty, lähettää asiakaspalvelin komennon eteenpäin toimintapalvelimelle. Komennon käytössä on huomioitavaa se, että kulmat jotka asiakaspalvelimelle tulevat, voivat olla välillä -0.5 ja 2.0, sillä korkeammat luvut kääntävät servomoottoreita liikaa. Jos servoa käännetään liian pitkälle, voi se hajoittaa sormen tai katkaista sormessa olevan siiman.

[14]Kuvan omistaja Aleksi Tuovinen (Lisenssi CC BY 4.0)

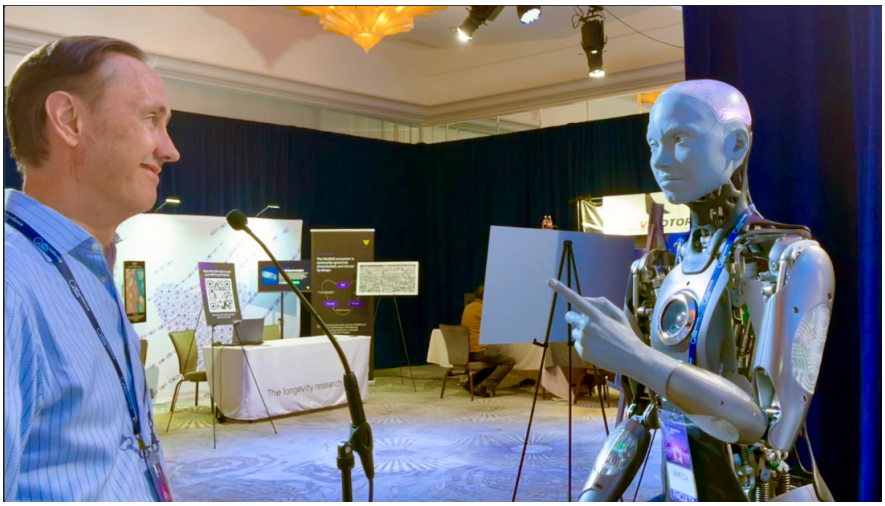
```
#Trial komento:
    def trial(self):
        command = []
        i = 0
        fingers = ["thumb", "index", "middle", "ring","
            pinky"]
        while len(command) < 5:
            angle = float(input(f"Angle␣for␣{fingers[i]}␣
                finger:␣"))
            if isinstance(angle, float):
                command.append(angle)
                i += 1
            else:
                print("Angle␣must␣be␣float.␣Safe␣values␣
                    are␣-0.5-2.0.")
        self.logger.info(f"Sending␣positions␣{command}")
        self.send_goal(command)
```

5. POHDINTA

Työn lopullinen tulos on puutteista huolimatta toimiva robottikäsi, joka kykenee imitoimaan ihmismäistä sormien liikettä. Toteutus esittelee, minkälaisen robottikäden voi matalallakin budjetilla saada rakennettua. Ennen kaikkea tämä toteutus toimii pohjana jatkokehitykselle, joka voi suuntautua esimerkiksi nykyisen ohjelmiston toiminnallisuuden laajentamiseen tai uusien osien lisäämiseen.

5.1. Toteutuksen arviointi

Käden yksinkertaisuuden takia se kalpenee toiminnallisuutensa ja joltain osaa myös ulkonäkönsä puolesta humanoidirobotiikan alan teknillisen huipun, kuten esimerkiksi kuvassa 13 esitetyn Amecan [55], rinnalla.



Kuva 13. *Tämänhetkisten humanoidirobotiikan kärkipäätä edustava Ameca-robotti (kuvassa oikealla)*[16]

Käsi on kuitenkin hyvä kiinnittämään huomiota, ja sitä voidaan käyttää mm. korkeakoulupäivien ohessa esittämään, mitä tietotekniikan alalla voi tehdä. Robotin useat komennot ja niiden esitteleminen voivat herättää mahdollisissa tulevissa opiskelijoissa kiinnostusta alan suuntaan, sekä saada heidät pohtimaan robottikäsien kehittämistä ja mahdollisuuksia.

Käden liike ei ole saavuttanut täydellistä ihmismäistä tasoa, mikä ei toteutuksessa käytettyjen servojen ja siimojen takia olekaan täysin mahdollista. Ohjelman ensimmäisissä versioissa käsi liikkui todella hitaasti ja kankeasti. Nykyisessä versiossa käsi simuloi tarkemmin oikean käden nopeutta, eikä sen liikehdintä enää näytä epätasaiselta, vaan kaikki nivelet liikkuvat yhdessä samaan tahtiin.

Kivi-paperi-sakset peli kärsii kaikista eniten inMoov-robottikäden rakenteesta. Koska rakenteessa ei ollut olkavartta mukana, eikä päätä ole tässä yhteydessä käytössä, pelin kulku oli hyvin vaikea raportoida käyttäjälle ilman, että joku ohjeisti pelin toiminnan käyttäjälle. Jos robotin käsi olisi ollut kokonainen, ja sisältänyt olkavarren,

[16]Kuvan omistaja Steve Jurvetson: https://www.flickr.com/photos/jurvetson/52023220709 (Lisenssi CC BY 2.0)

olisi pelaajalle voitu esittää pelin kulku esimerkiksi kättä heiluttamalla, tai jos robotin pää olisi ollut mukana, olisi pää voinut laskea alaspäin pelaajalle, milloin robotti aikoo näyttää valintansa. Testauksessa projektin ulkopuolisten kanssa vain osa käyttäjistä tajusi miten peliä tulisi pelata pelkän käden kanssa.

Tietyillä toiminnoilla käden sormet voivat juuttua tiettyihin asentoihin ja ne pitää siirtää pois jumiutumiskohdasta manuaalisesti. Tämä johtuu 3D-tulostimen virhemarginaalista, jonka takia robotin osien pinnassa on rosoisuutta, sekä robotin liikuttamiseen käytettävien siimojen joustavuudesta. 3D-tulostettuja osia viilattiin ja hiottiin tämän ongelman välttämiseksi, mutta tietyt osat jäivät silti liian rosoisiksi. InMoov-käden sormien nivelet on liitetty toisiinsa käyttämällä ruuveja 14, jotka nekin tekevät nivelten liikkumisesta hankalampaa. Ruuvit voitaisiin korvata esimerkiksi rautalangalla, joka ei puristaisi sormien osia yhtä lujaa kuin ruuvit. Toinen vaihtoehto käden liikkeen helpottamiseksi olisi rasvata kaikki käden nivelet, mikä olisi työlästä, sillä se vaatisi käden purkamista alkutekijöihinsä.

Kuva 14. *Sormien ruuvit*[17]

Muutamat seikat vaikeuttivat työn toteutusta. Työssä käytetty ROS2 on tehty tutkijoilta tutkijoille, mikä on johtanut siihen, että sen dokumentaatio on kovin hajanaista ja vaikeaselkoista. ROS2:n yhteisö on yrittänyt korjata tätä ongelmaa uudella turtlesim aloittelijakurssillaan [56], mutta tässä työssä siitä ei ollut apua, sillä kurssin ohjelmistopaketti sisälsi laajan kirjon sisäisiä ohjelmia, joista ei ollut tietoa ROS2 dokumentaatiossa. Toteutuksessa täytyi yhdistää laaja kirjo ROS-lähteitä, niin alkuperäisestä ROS:ta, kuin sen jatkajasta ROS2:ta, kommentteja GitHubista liittyen molempiin kirjastoihin ja erilaisten tekniikka-blogien hajanaisia toteutuksia, jotka kertoivat mitä kirjastoja asiakaspalvelimessa tulisi käyttää, jotta asiakaspalvelin kykenisi rakentamaan toimivan maaliviestin ja vastaanottamaan viestejä.

Ottaen huomioon, että käsi rakennettiin osaksi suurempaa kokonaisuutta, jossa on myös mukana pää ja myös puheen-, että liikkeenseuranta, onnistui käden toteutus hyvin. Käsi kykenee itsenäiseen liikkeeseen ja sen ohjelmistopuoli on tehty helposti ymmärrettäväksi ja jatkettavaksi.

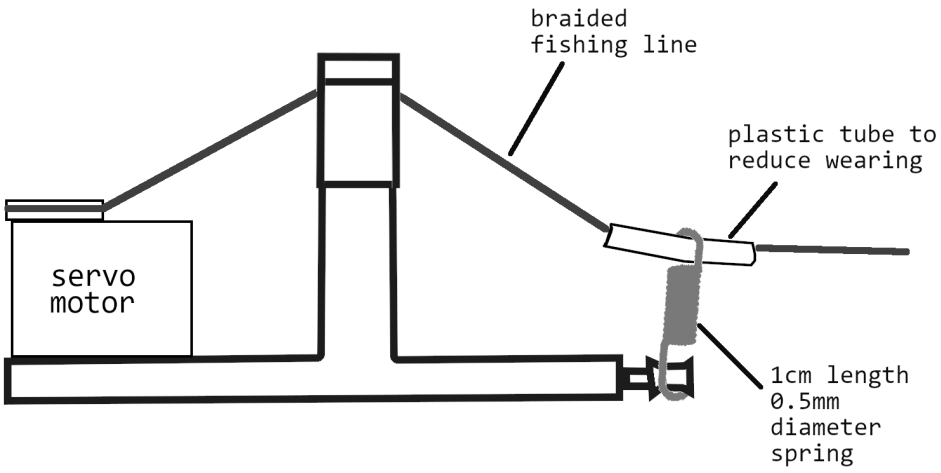
[17] Kuvan omistaja Aleksi Tuovinen (Lisenssi CC BY 4.0)

5.2. Jatkokehitys

Robotin pystyttämiseen tarvitaan useita komentoja, ja virhetapauksessa, kuten servomoottorin ylikuormittumisessa tai virtakatkossa, joutuu käyttäjä syöttämään uudelleen kaikki ylösajo-komennot. Ylösajamiseen tarvitaan colconin pakettien rakentaminen, pakettien liittäminen osaksi CLI-ympäristöä, sekä tarvittavien servo-ohjainten aktivoiminen. Tämä vaihe olisi tärkeä optimoida toimimaan esimerksiksi yhdellä komennolla. Muussa tapauksessa robottikäden virhetilanteet voivat helposti johtaa ongelmiin esittelytarkoituksessa, kun robotin kättä joudutaan laittamaan toistuvasti ajokelpoiseksi.

Tulevaisuudessa käsi olisi mahdollista liittää osaksi suurempaa robottikokonaisuutta, kuten osaksi kokonaista inMoov-robottia. Myös käden asiakasohjelma on helppo ottaa käyttöön kokonaisen robotin ajo-ohjelmaa varten, sillä ROS2:n solmurakenne mahdollistaa solmujen uudelleen käytön helposti ja nopeasti, jos niiden rakenne on selvillä. Käden asiakasohjelma on rakennettu niin, että uusien asentojen tekemiseen tarvitsee vain määrittää asennon nimi ja kohta mihin sormien nivelien tulee liikkua, joten jos tuleva käyttäjä haluaa muokata niitä, tarvitsee hänen muuttaa vain yksittäisiä rivejä asiakasohjelman koodista.

Kun käden fyysistä osuutta lähdetään jatkokehittämään, voitaisiin kyynärvarren sisälle asentaa InMoovin kokoamisohjeissa mainittu kuvan 15 mukainen "Tensioner", eli sormia ohjaavien siimojen jännitystä servojen kaikissa asennoissa ylläpitävä jousiratkaisu [57]. Lisäksi jatkokehityksessä voitaisiin tutkia tapoja estää keskisormen ja ranteen niveliä jumittumasta tai ylikuormittumasta tiettyjen liikkeiden aikana.



Kuva 15. *Illustraatio InMoovin siimojen jännitystä ylläpitävän ratkaisun toiminnasta sivulta päin kuvattuna*[19]

Jos haluttaisiin saada robottikäsi, jonka sormilla voitaisiin tehdä tarkempia, hienosäätöä vaativia liikkeitä, kuten pinsettiotteella tarttumista, tulisi nykyisestä siimoihin perustuvasta ratkaisusta luopua. Kun jokaisen sormen jokainen nivel

[19]Kuvan omistaja Juho Bruun (Lisenssi CC BY 4.0)

olisi servomoottoriin pohjautuva solmu, voisi liikkeistä tehdä tarkempia esimerkiksi soveltamalla käänteiskinematiikkaa. Tällöin käsi tarvitsisi lisäksi kämmenen ja sormien tarttumapintoihin jonkinlaisen keinoihon, sillä tämänhetkisellä ratkaisulla esineet liukuvat todella herkästi pois käden otteesta. Jos käden toteutus haluttaisiin viedä vielä pitemmälle, voitaisiin käteen asentaa tarttumapinnan lisäksi haptiset sensorit, jotka osaisivat tulkita esimerkiksi käden koskettamien esineiden muotoa ja jäykkyyttä.

6. YHTEENVETO

Tässä työssä esitettiin toimiva robottikäsitoteutus, joka soveltuu robottikäsien kykyjen havainnollistamiseen. Työhön liittyvät ongelmat ja siinä käytetyt ratkaisut tuotiin myös esille. Alustaa kyettiin ajamaan onnistuneesti siihen kehitetyllä ohjelmistolla, mutta muihin robottikäsialustoihin verrattuina ovat robotin liikkeet melko alkukantaisia.

Ohjelmisto kehitettiin myös havainnollistamaan, miten alustaa ajetaan tarjoten mahdolliselle tulevalle kehitykselle pohjaa. Tätä kehitystä voi olla esimerkiksi käyttäjän liikkeisiin vastaaminen tai viittomakielelen käyttäminen. Myös nivelten ja osien lisääminen käteen ovat kehityskohteita, joihin työ antaa mahdollisuuksia. Tämän kehityksen avulla voidaan käden esittelykapasiteettia lisätä demonstroimaan monimutkaisempia käden liikkeitä ja kommunikointitapoja.

7. VIITTEET

[1] Kajita S. & Sugihara T. (2009) Humanoid robots in the future. Advanced Robotics 23, ss. 1527–1531.

[2] Kim S.Y., Schmitt B.H. & Thalmann N.M. (2019) Eliza in the uncanny valley: anthropomorphizing consumer robots increases their perceived warmth but decreases liking. Marketing Letters 30, ss. 1–12. URL: https://doi.org/10.1007/s11002-019-09485-9.

[3] Koppenborg M., Nickel P., Naber B., Lungfiel A. & Huelke M. (2017) Effects of movement speed and predictability in human–robot collaboration. Human Factors and Ergonomics in Manufacturing & Service Industries 27, ss. 197–209. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/hfm.20703.

[4] Duffy B.R. (2003) Anthropomorphism and the social robot. Robotics and Autonomous Systems 42, ss. 177–190. URL: https://www.sciencedirect.com/science/article/pii/S0921889002003743, socially Interactive Robots.

[5] Gulletta G., Silva E.C.e., Erlhagen W., Meulenbroek R., Costa M.F.P. & Bicho E. (2021) A human-like upper-limb motion planner: Generating naturalistic movements for humanoid robots. International Journal of Advanced Robotic Systems 18, s. 1729881421998585.

[6] Saunderson S. & Nejat G. (2019) How robots influence humans: A survey of nonverbal communication in social human–robot interaction. International journal of social robotics 11, ss. 575–608. URL: https://www.researchgate.net/publication/331552270_How_Robots_Influence_Humans_A_Survey_of_Nonverbal_Communication_in_Social_Human-Robot_Interaction.

[7] Sidner C.L., Lee C., Kidd C.D., Lesh N. & Rich C. (2005) Explorations in engagement for humans and robots. Artificial Intelligence 166, ss. 140–164. URL: https://www.sciencedirect.com/science/article/pii/S0004370205000512.

[8] Sheridan T. (2016) Human-robot interaction: Status and challenges. Human factors 58.

[9] Hambuchen K., Marquez J. & Fong T. (2021) A review of nasa human-robot interaction in space. Current Robotics Reports 2, s. 265–272.

[10] Crusan J., Bleacher J., Caram J., Craig D., Goodliff K., Herrmann N., Mahoney E. & Smith M. (2019) Nasa's gateway: An update on progress and plans for extending human presence to cislunar space. Teoksessa: 2019 IEEE Aerospace Conference, ss. 1–19.

[11] Hady G.G., Abigail C.D., Sebastian H., Nicola D.Q., Andrea A. & Damian B. (2018) Alcides: A novel lunar mission concept study for the demonstration of enabling technologies in deep-space exploration and human-robots interaction. Acta Astronautica 151, ss. 270–283. URL: https://www.sciencedirect.com/science/article/pii/S0094576517301248.

[12] Kanas N., Sandal G., Boyd J., Gushin V., Manzey D., North R., Leon G., Suedfeld P., Bishop S., Fiedler E., Inoue N., Johannes B., Kealey D., Kraft N., Matsuzaki I., Musson D., Palinkas L., Salnitskiy V., Sipes W., Stuster J. & Wang J. (2009) Psychology and culture during long-duration space missions. Acta Astronautica 64, ss. 659–677. URL: https://www.sciencedirect.com/science/article/pii/S0094576508003883.

[13] Smith T., Barlow J., Bualat M., Fong T., Provencher C., Sanchez H. & Smith E. (2016), Astrobee: A new platform for free-flying robotics on the international space station. URL: https://ntrs.nasa.gov/citations/20160007769.

[14] Bualat M.G., Smith T., Smith E.E., Fong T. & Wheeler D. (2018) Astrobee: A new tool for iss operations. 2018 SpaceOpsConference .

[15] Biondi F., Alvarez I. & Jeong K.A. (2019) Human–vehicle cooperation in automated driving: A multidisciplinary review and appraisal. International Journal of Human–Computer Interaction 35, s. 932–946.

[16] Nyholm S. & Smids J. (2018) Automated cars meet human drivers: Responsible human-robot coordination and the ethics of mixed traffic. Ethics and Information Technology 22, s. 335–344.

[17] committee O.R.A.D.O. (2021) Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. URL: https://doi.org/10.4271/J3016_202104.

[18] Morando A., Gershon P., Mehler B. & Reimer B. (2021) A model for naturalistic glance behavior around tesla autopilot disengagements. Accident Analysis & Prevention 161, s. 106348. URL: https://www.sciencedirect.com/science/article/pii/S0001457521003791.

[19] Marcano M., Diaz S., Perez J. & Irigoyen E. (2020) A review of shared control for automated vehicles: Theory and applications. IEEE Transactions on Human-Machine Systems 50, s. 475–491.

[20] (2022), Tesla vehicle safety report. URL: https://www.tesla.com/VehicleSafetyReport.

[21] Reed N., Leiman T., Palade P., Marieke M. & Kester L. (2021) Ethics of automated vehicles: breaking traffic rules for road safety. Ethics and Information Technology 23, ss. 777–789. URL: https://www.proquest.com/scholarly-journals/ethics-automated-vehicles-

breaking-traffic-rules/docview/2607083432/se-2?accountid=13031, copyright - © The Author(s), under exclusive licence to Springer Nature B.V. 2021; Last updated - 2022-01-13.

[22] Camara F. & Fox C. (2020) Space invaders: Pedestrian proxemic utility functions and trust zones for autonomous vehicle interactions. International Journal of Social Robotics 13, s. 1929–1949.

[23] Gupta S., Vasardani M. & Winter S. (2016) Conventionalized gestures for the interaction of people in traffic with autonomous vehicles. Teoksessa: Proceedings of the 9th ACM SIGSPATIAL International Workshop on Computational Transportation Science, IWCTS '16, Association for Computing Machinery, New York, NY, USA, s. 55–60. URL: https://doi-org.pc124152.oulu.fi:9443/10.1145/3003965.3003967.

[24] Robotics I.., Robots and robotic devices—collaborative robots(iso/ts 15066:2016). URL: https://www.iso.org/standard/62996.html.

[25] Ranz F., Komenda T., Reisinger G., Hold P., Hummel V. & Sihn W. (2018) A morphology of human robot collaboration systems for industrial assembly. Procedia CIRP 72, ss. 99–104. URL: https://www.sciencedirect.com/science/article/pii/S2212827118301094, 51st CIRP Conference on Manufacturing Systems.

[26] Malik A.A. & Brem A. (2021) Digital twins for collaborative robots: A case study in human-robot interaction. Robotics and Computer-Integrated Manufacturing 68, s. 102092. URL: https://www.sciencedirect.com/science/article/pii/S0736584520303021.

[27] Kopp T., Baumgartner M. & Kinkel S. (2021) Success factors for introducing industrial human-robot interaction in practice: an empirically driven framework. The International Journal of Advanced Manufacturing Technology 112. URL: https://doi.org/10.1007/s00170-020-06398-0.

[28] Djuric A., Urbanic R.J. & Rickli J. (2016) A framework for collaborative robot (cobot) integration in advanced manufacturing systems. SAE International Journal of Materials and Manufacturing 9, ss. 457–465.

[29] Fong T., Nourbakhsh I. & Dautenhahn K. (2003) A survey of socially interactive robots. Robotics and Autonomous Systems 42, ss. 143–166. URL: https://www.sciencedirect.com/science/article/pii/S092188900200372X, socially Interactive Robots.

[30] Breazeal C. (2003) Toward sociable robots. Robotics and Autonomous Systems 42, ss. 167–175. URL: https://www.sciencedirect.com/science/article/pii/S0921889002003731, socially Interactive Robots.

[31] Bartneck C. & Forlizzi J. (2004) A design-centred framework for social human-robot interaction. Teoksessa: RO-MAN 2004. 13th IEEE international workshop on robot and human interactive communication (IEEE Catalog No. 04TH8759), IEEE, ss. 591–594.

[32] Kimr N., Bodunkov N. & Sinyavskaya J. (2021) Hardware and software structure for a social robot capable of situation analysis. nide 2096. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85121514784&doi=10.1088%2f1742-6596%2f2096%2f1%2f012070&partnerID=40&md5=a8f21d15b5f8632024f76eca211a416e, cited By 0.

[33] Ghafurian M., Hoey J. & Dautenhahn K. (2021) Social robots for the care of persons with dementia: A systematic review. ACM Transactions on Human-Robot Interaction 10. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85115155319&doi=10.1145%2f3469653&partnerID=40&md5=655bab8904e4a48c54fa3d47c781a75d, cited By 0.

[34] Martinez-Martin E., Escalona F. & Cazorla M. (2020) Socially assistive robots for older adults and people with autism: An overview. Electronics (Switzerland) 9. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85079881689&doi=10.3390%2felectronics9020367&partnerID=40&md5=398d4d529bb1eb6e71c0113370179e3e, cited By 16.

[35] Wood L.J., Zaraki A. & Dautenhahn K. (2021) Developing Kaspar: A Humanoid Robot for Children with Autism. International Journal of Social Robotics .

[36] Zuschnegg J., Schüssler S., Paletta L., Voithofer C., Lodron G., Orgel T., Russegger S., Schneeberger M., Fellner M., Steiner J., Pansy-Resch S., Holter M., Prodromou D., Brunsch S., Carnevale L., Lammer L. & Koini M. (2021) Psychosocial effects of the humanoid socially assistive robot coach pepper on informal caregivers of people with dementia: A mixed-methods study. Alzheimer's & dementia : the journal of the Alzheimer's Association 17, s. e052150. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85123036265&doi=10.1002%2falz.052150&partnerID=40&md5=b9a1492bdb6c290f7bf5e232649b9c1e, cited By 0.

[37] Vilk J. & Fitter N. (2020) Comedy by jon the robot. s. 80. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85083202944&doi=10.1145%2f3371382.3378201&partnerID=40&md5=e85fd3f0242ea6b5fccdf8704316d45a, cited By 1.

[38] Marino F., Chilà P., Sfrazzetto S., Carrozza C., Crimi I., Failla C., Busà M., Bernava G., Tartarisco G., Vagni D., Ruta L. & Pioggia G. (2020) Outcomes of a robot-assisted social-emotional understanding intervention for young children with autism spectrum disorders. Journal of Autism and Developmental Disorders 50, ss. 1973–1987. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85062835394&doi=10.1007%2fs10803-019-03953-x&partnerID=40&md5=3807c7d26bc18f29959b1ea2c2a08040, cited By 20.

[39] Zaraki A., Wood L., Robins B. & Dautenhahn K. (2018) Development of a semi-autonomous robotic system to assist children with autism in developing visual perspective taking skills. Teoksessa: 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), ss. 969–976.

[40] Pu L., Moyle W. & Jones C. (2020) How people with dementia perceive a therapeutic robot called paro in relation to their pain and mood: A qualitative study. Journal of Clinical Nursing 29, ss. 437–446. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85076294312&doi=10.1111%2fjocn.15104&partnerID=40&md5=7a51c37d24a9e2ed8cc49eb25237fb80, cited By 14.

[41] Moringen A., Fleer S., Walck G. & Ritter H. (2020) Attention-based robot learning of haptic interaction. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 12272 LNCS, s. 462 – 470. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85091287375&doi=10.1007%2f978-3-030-58147-3_51&partnerID=40&md5=565f6db072d97be3f36b70e9c090716b, cited by: 1.

[42] Kappassov Z., Corrales J.A. & Perdereau V. (2020) Touch driven controller and tactile features for physical interactions. Robotics and Autonomous Systems 123. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85073989660&doi=10.1016%2fj.robot.2019.103332&partnerID=40&md5=144c503c3342c8a81d24e2b37abc23a2, cited by: 7.

[43] Qiu Y., Sun S., Wang X., Shi K., Wang Z., Ma X., Zhang W., Bao G., Tian Y., Zhang Z., Ding H., Chai H., Liu A. & Wu H. (2022) Nondestructive identification of softness via bioinspired multisensory electronic skins integrated on a robotic hand. npj Flexible Electronics 6. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85132292322&doi=10.1038%2fs41528-022-00181-9&partnerID=40&md5=09125eaa3d63d059add92e6e7661097f, cited by: 0.

[44] Zacharias F., Schlette C., Schmidt F., Borst C., Rossmann J. & Hirzinger G. (2011) Making planned paths look more human-like in humanoid robot manipulation planning. Teoksessa: 2011 IEEE International Conference on Robotics and Automation, ss. 1192–1198.

[45] He C., Xu X.W., Zheng X.F., Xiong C.H., Li Q.L., Chen W.B. & Sun B.Y. (2021) Anthropomorphic reaching movement generating method for human-like upper limb robot. IEEE Transactions on Cybernetics , ss. 1–12.

[46] Gulletta G., Erlhagen W. & Bicho E. (2021) Continual learning of human-like arm postures. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85114555933&doi=10.

1109%2fICDL49984.2021.9515565&partnerID=40&md5=9b9d8dc79f344e950941615808545aa1, cited By 0.

[47] Ros website. URL: https://www.ros.org/.

[48] Ros 2 explained: Overview and features | electronic design. URL: https://www.electronicdesign.com/industrial-automation/article/21214053/electronic-design-ros-2-explained-overview-and-features.

[49] Ros 2 documentation. URL: http://docs.ros.org/en/foxy/.

[50] Inmoov - open-source 3d printed robot. URL: http://inmoov.fr/.

[51] Rviz simulator wiki. URL: http://wiki.ros.org/rviz.

[52] xacro wiki. URL: http://wiki.ros.org/xacro.

[53] Inmoov body parts library. URL: http://inmoov.fr/inmoov-stl-parts-viewer/.

[54] Fusion 360 | autodesk. URL: https://www.autodesk.com/products/fusion-360/overview?term=1-YEAR&tab=subscription.

[55] Engineered arts ameca brochure. URL: https://cloud.engineeredarts.co.uk/s/GgOLM7v6ME7xG0D/download.

[56] Ros 2 tutorials. URL: https://docs.ros.org/en/foxy/Tutorials.html.

[57] Inmoov hand and foreram assembly tutorial. URL: http://inmoov.fr/hand-and-forarm/.

Liite 1	Käden kontrolli-ohjeet
Liite 2	Lyhyt video käden toiminnasta
Liite 3	Työtuntijakauma

A. README.MD

1.1. How to run

Ensure that you have set up the robot as instructed in Robot bring-up[20] and that r_hand_controller is up and running. If not, refer back to the bring-up[21].

Once the robot is running properly, open a new command window, and start the client with the following command.

```
python3 client/hand\_action\_client.py
```

1.2. How to use

Once the client is up and running you will be greeted by a command line interface asking for your next command.

```
Input command:
```

Type in the command you want and watch as the hand completes the action.

1.3. The commands

Currently, the following actions are available.

Action What it does
open - Extends all fingers
fist - Forms a fist
scissors - Extends index and middle finger
point - Extends index finger
thumbs_up - Gives a thumbs up!
grasp - Grasps an object
hard_rock - Heavy metal...
pen_grasp - Grasps with index finger and thumb
rps - Plays a round of Rock-Paper-Scissors
trial - Allows you to input a custom position
quit - Ends the program

1.3.1. Trial

As told above the trial command allows you to set a custom position for the fingers. This is done by asking a position for each finger separately. Each finger must be given a position as ROS2 does not allow the hand to move without specifying positions for all the joints in the r_hand_controller.

IMPORTANT THE CUSTOM STATES MUST BE BETWEEN -0.5 AND 2 YOU WILL BREAK THE HAND IF YOU GO BEYOND THESE

[20] <../docs/BRINGUP.md>
[21] <../docs/BRINGUP.md>

B. DEMOVIDEO

Kuva 16. Linkki videoon https://youtu.be/fAL0m9YUirw QR-koodina

C. TYÖTUNTIJAKAUMA

Vaihe 1: Taustatutkimus ja aiheen valitseminen ryhmänä	
Tekijä	Työtunnit
Juho Bruun	51
Aleksi Tuovinen	48
Ville Hokkinen	50

Vaihe 2: Fyysinen toteutus ja 3D-mallinnus	
Tekijä	Työtunnit
Juho Bruun	66
Aleksi Tuovinen	47
Ville Hokkinen	70

Vaihe 3: Kehitysympäristön pystyttäminen ja ohjelmiston toteutus	
Tekijä	Työtunnit
Juho Bruun	44
Aleksi Tuovinen	70
Ville Hokkinen	40

Vaihe 4: Dokumentointi, kirjoittaminen ja tekstin ulkoasun muokkaus	
Tekijä	Työtunnit
Juho Bruun	48
Aleksi Tuovinen	45
Ville Hokkinen	40

Työtunnit yhteensä	
Juho Bruun	209
Aleksi Tuovinen	210
Ville Hokkinen	200