TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Elektroenergeetika ja mehhatroonika instituut

TALLINNA TEHNIKAÜLIKOOL

KUUEJALGSE LIIKURROBOTI PROJEKTEERIMINE JA PROTOTÜÜPIMINE

DESIGNING AND PROTOTYPING OF A HEXAPOD MOBILE ROBOT

BAKALAUREUSETÖÖ

Üliõpilane: Märt Arjus

Üliõpilaskood: 193809EAAB

Juhendaja: Tanel Jalakas, vanemteadur

Tallinn 2024

AUTORIDEKLARATSIOON

Olen koostanud lõputöö iseseisvalt.
Lõputöö alusel ei ole varem kutse- või teaduskraadi või inseneridiplomit taotletud.
Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

"....." ..... 2024.

Autor: .....
/ allkiri /

Töö vastab bakalaureusetöö/magistritööle esitatud nõuetele

"....." ..... 2024.

Juhendaja: .....
/ allkiri /

Kaitsmisele lubatud

"....."..... 2024.

Kaitsmiskomisjoni esimees .....
/ nimi ja allkiri /

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina Märt Arjus

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

„Kuuejalgse liikurroboti projekteerimine ja prototüüpimine“,

mille juhendaja on Tanel Jalakas,

1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

13.05.2024 (kuupäev)

[1] *Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.*

LÕPUTÖÖ LÜHIKOKKUVÕTE

Autor: Märt Arjus *Lõputöö liik:* Bakalaureusetöö

Töö pealkiri: Kuuejalgse liikurroboti projekteerimine ja prototüüpimine

Kuupäev: 13.05.2024 72 lk (Lisad kaasaarvatud)

Ülikool: Tallinna Tehnikaülikool
Teaduskond: Inseneriteaduskond
Instituut: Elektroenergeetika ja mehhatroonika instituut

Töö juhendaja: vanemteadur Tanel Jalakas
Töö konsultant: puudub

Sisu kirjeldus:

Viimaste aastate jooksul on jalgsete liikurrobotite valdkond kasvanud järjest populaarsemaks. Populaarsusega kasvab ka huvi valdkonna vastu nii teadusarenduses kui ka isearendajate seas. Isearendajatele võivad olla teatud arendustööks kasulikud ressursid piiratud kättesaadavusega. Seetõttu võib olla keerukas leida baasressursse, mis lihtsustaksid baasplatvormi loomist. Baasplatvorm on oluline ja vajalik süsteem, et selle peal teha tänapäeva valdkonna arengutrendide põhjal arendustööd.

Antud töö eesmärgiks on lihtsustada kuuejalgse liikurroboti baasplatvormi modelleerimist ja projekteerimist, mille käigus luuakse kuuejalgse robotsüsteemi projekteerimise ja prototüüpimise protsesse selgitav ja lihtsustav baasressurss, mis oleks kõigile vabalt kättesaadav. Töö tulemuseks on baastöövõimekusega kuuejalgse liikurroboti projekteerimiseks vajalikud matemaatilised mudelid ja nendel põhinev prototüüp. Seda prototüüpi kasutati, projekteerimise tulemuste katsetamisel, et kinnitada nende korrektsus.

Märksõnad: jalgsed robotid, kuuejalgne, liikurrobot, kinemaatika, prototüüp, modelleerimine, projekteerimine, bakalaureusetöö

ABSTRACT

Author: Märt Arjus *Type of work:* Bachelor thesis

Title: Designing and prototyping of hexapod a mobile robot

Date: 13.05.2024 72 pages (Appendices included)

University: Tallinn University of Technology
School: School of Engineering
Department: Department of Electrical Power Engineering and Mechatronics

Supervisor of thesis: senior researcher Tanel Jalakas
Consultant: none

Abstract:

The field of legged mobile robots has been gaining popularity in the past few years. With that popularity comes growing interest in the field, both in scientific research and among self-developers. For self-developers, the resources that are helpful for further development may be limited in accessability. Because of that locating the fundamental resources that would make the development of a base platform easier can be difficult. A base platform is an important and necessary system for conducting new development work based on current trends in the field.

The aim of this thesis is to simplify the modeling and designing processes of the base platform of a six-legged mobile robot, during which a base resource that explains and simplifies the processes of designing and prototyping a hexapod mobile robot is created, which would be freely accessible to everyone. The results of the thesis are the mathematical models necessary for designing a hexapod mobile robot that has basic functionality and a prototype based on these models. This prototype was used to test the results of the mathematical models to verify their correctness.

Keywords: legged robots, six-legged, mobile robot, kinematics, prototype, hexapod, modelling, designing, bachelor thesis

Elektroenergeetika ja mehhatroonika instituut

LÕPUTÖÖ ÜLESANNE

Üliõpilane: Märt Arjus
Õppekava, peaeriala: EAAB, mehhatroonika
Juhendaja(d): vanemteadur, Tanel Jalakas, 620 3703
Konsultant: puudub

Lõputöö teema:
(eesti keeles) Kuuejalgse liikurroboti projekteerimine ja prototüüpimine
(inglise keeles) Designing and prototyping of a hexapod mobiile robot

Lõputöö põhieesmärgid:

1. Luua kuuejalgse liikurroboti projekteerimiseks vajalikud vabalt kättesaadavad matemaatilised mudelid
2. Lihtsustada kuuejalgse roboti jõuelektroonika valiku kriteeriumite arvutamist
3. Ehitada kuuejalgse liikurroboti baasplatvormi prototüüp

Lõputöö etapid ja ajakava:

Nr	Ülesande kirjeldus	Tähtaeg
1.	Matemaatiliste mudelite koostamine	Märts III nädal
2.	Prototüübi komponentide valimine	Aprill I nädal
3.	Prototüübi projekteerimine ja ehitamine	Aprill III nädal
4.	Prototüübi katsetamine	Mai I nädal

Töö keel: Eesti keel **Lõputöö esitamise tähtaeg:** "13." Mai 2024 .a

Üliõpilane: …….. ..... "....."..... 20…..a
/allkiri/

Juhendaja: …….. ..... "....."..... 20…..a
/allkiri/

Konsultant: …….. ..... "....."..... 20…..a
/allkiri/

Programmijuht: ..... "....."..... 20…..a
/allkiri/

SISUKORD

EESSÕNA

Käesoleva töö teema lähtub autori isiklikust huvist jalgsete liikurrobotite valdkonna vastu. See huvi jalgsete liikurrobotite vastu on tekkinud viimaste aastate jooksul valdkonna populaarsuse kasvuga. Kuigi liikurrobotite temaatikat käsitleti õpingute käigus pigem põgusalt ja rohkem puudutati statsionaarsete robotitega seonduvat, on liikurrobotite arendus olnud küllaltki pikaajaline ning selle kohta on kättesaadaval mitmeid allikaid. Kuna jalgseid liikurroboteid nimetatakse ka loodust kopeerivateks robotiteks, siis uurides lähemalt looduses eksisteerivate erinevate isendite liikumist on arendustegevuse tulemusena võimalik neid mooduseid üle kanda ka robotitele.

Avaldan siiralt tänu oma juhendajale Tanel Jalakale, kes andis töö käigus konstruktiivset ja julgustavat tagasisidet ning pakkus takistuste puhul välja alternatiivseid lähenemisviise.

Lühendite ja tähiste loetelu

DH – Denavit-Hartenberg
DOF (ingl. keeles Degree of freedom) – liikuvusaste
PWM (ingl. Keeles Pulse Width Modulation) – pulsilaiusmodulatsioon
ROS (ingl. keeles Robotics Operating System) – robootikas kasutatav tarkvara raamistik
NiMH – nikkelmetallhüdriid
Li-ion – Liitiumioon
LiPo - Liitiumpolümeer
NiCd – nikkel-kaadmium
UBEC (ingl. keeles Universal Battery Eliminator Circuit) – aku väljundpinge regulaatori moodul
PLA (ingl. keeles Polylactic Acid) – polülaktiid hape on 3D printimises kasutatav polüester materjal

SISSEJUHATUS

Robootika valdkond jaguneb statsionaar- ja liikurrobotite vahel kaheks. Liikurrobotid on võimelised iseseisvalt keskkonnas ringi liikuma, kas ratastel, roomikutel, jalgadel, lennates, ujudes või hübriidselt. Jalgseid liikurroboteid võib nimetada loodust kopeerivateks robotiteks. Nende eripära teiste liikurrobotite seas seisneb nende liikumisvõimekuses. Võrreldes ratastel või roomikutel põhinevate liikurrobotitega, suudavad jalgsed robotid efektiivsemalt liikuda ebaühtlase tasapinnaga keskkonnas. Teiseks eripäraks on nende väike jalajälg, mis on oluline keskkonnas, kus maapinda ei tohi liigselt deformeerida. Jalgseid liikurroboteid saab klassifitseerida nende jalgade arvu järgi. Nende seast kuuejalgsete robotite erilisus seisneb selles, et nad on võimelised olema, nii liikudes kui ka paigal seistes, staatiliselt stabiilsed ja nad ei pea ennast aktiivselt balansseerima.

Viimaste aastatega on jalgsete liikurrobotite valdkond saanud meedias palju tähelepanu tänu Boston Dynamics neljajalgsele Spot [1] ja kahejalgsele humanoid Atlas [2] robotile. Tähelepanuga kaasneb ka huvi laiem kasv valdkonna ja selle arengu vastu. Valdkonna arendamisega tegelevad nii teadus- ja arendusasutused, ettevõtted kui ka üksikisikud ehk isearendajad. Isearendajad kasutavad olemasolevaid teadmisi, oskusi, uurimistulemusi, praktikaid ja vahendeid, et luua uut teadmist, mida saab kasutada uute toodete valmistamiseks, uute protsesside kasutusele võtmiseks või olemasolevate toodete ja protsesside täiustamiseks. Kuigi isearendajate arendustegevus ei kvalifitseeru reeglina teadus- ja arendustegevuseks, omavad nad valdkonna üldisesse arengusse panustajatena ja teadmiste levitajatena ühiskonnas olulist rolli.

Isearendajatele on arendustööks vabalt kättesaadavate ressurssidena GitHub veebikeskkond, kus jagatakse kuuejalgsete liikurrobotite tarkvaralisi lahendusi, tavamüügist leitavad kallid arendusplatvormide lahendused nagu Phantom AX Hexapod [3] ja ka ligipääs projekteerimise või arendamise protsessi lahkavatele teadusartiklitele. Paraku paljud artiklid ei anna täielikku projekteerimisprotsessi ülevaadet, vaid nendes fokusseeritakse kindlale probleemile või lahendusele nagu näiteks artikkel [4], kus keskendutakse kuuejalgse roboti sammu trajektoori defineerimisele.

Antud töö eesmärgiks on luua kuuejalgse liikurroboti baasplatvormi projekteerimise ja prototüübi konstrueerimise protsesse selgitav ressurss isearendajatele. Selle saavutamiseks on töö jagatud osadeks, mis kajastavad robotsüsteemi projekteerimise ja prototüübi konstrueerimise protsesse etappidena. Esimeses osas tehakse valdkonna ülevaade ajaloost, arengutaset, olemasolevatest lahendustest ning tänapäeva

arendustrendidest. Teises osas määratakse roboti algsed parameetrid, luuakse robotsüsteemi kinemaatiline mudel, mille abil defineeritakse jalgade liikumistrajektoor ja arvutatakse jalgade liigenditele mõjuvad jõumomendid. Kolmandas osas valitakse roboti elektroonilised komponendid, kuhu alla kuuluvad ajamid, juhtseade, toiteallikas, tüürahelad ja vaheühendustena pingemuundurid ning kaitseahelad. Neljandas osas projekteeritakse roboti prototüübi mehaaniline struktuur, toodetakse struktuuri osad 3d printimise abil ja konstrueeritakse füüsiline prototüüp. Viiendas osas viiakse ehitatud prototüübiga läbi kaks katsetust, eesmärgiga kinnitada projekteerimistulemuste korrektsust. Esimesena katsetatakse prototüübi ühe jala juhtimist ning teise katsena testitakse terve roboti otseliikumist defineeritud sammu trajektoori järgi. Kuuendas osas tehakse katsetusete järeldused ning pakutakse välja erinevaid projekteeritud liikurroboti edasiarenduse võimalusi.

Töös koostatud matemaatilised mudelid ja nende visuaalsed kujutused loodi MathWorks Matlab tarkvara abil, millel on Robotics System Toolbox lisa tööriistana. Mehaaniline struktuur projekteeriti Autodesk Inventor CAD tarkvara keskkonnas. Matlab-s loodud programmi koodid kantakse töö lõppu lisade alla ning prototüübi elektriskeem graafilise osa alla.

1. VALDKONNA ÜLEVAADE

Jalgsete liikurrobotite ja mehhanismide arendusvaldkond on eksisteerinud mitmeid aastakümneid ning selle jooksul on toimunud palju läbimurdeid. Tänapäeval on jalgsed robotsüsteemid leidnud rakendust mitmetes erinevates valdkondades ning arendustöös rakendatakse mitmeid uusi lähenemisviise ja lahendusi.

1.1 Valdkonna ajalugu

Esimesed jalgset liikumist jäljendavad mehhanismid pärinevad 15. sajandi lõpust, mil Leonardo Da Vinci disainis ja tõenäoliselt ka ehitas liigendatud soomusrüüs rüütli. Esimesed dokumenteeritud teaduslikud uuringud jalgse liikumise kohta on 19. sajandi teisest poolest, kui Eadweard Muybridge uuris, fotograafia põhjal algselt hobuste ja hiljem ka teiste loomade, sealhulgas ka inimese, kõnnakuid. 1950ndate keskel hakkasid erinevad uurimisrühmad süstemaatiliselt uurima ja arendama kõndimismasinaid ning kümmekond aastat hiljem hakati neid laborites konstrueerima ja ehitama [5].

Esimene kuuejalgne liikurrobot pärineb 1960ndate aastate algusest, mil Space General Corporation töötas välja kaks robotit, et uurida kuukulguri jalgade liikumise kontseptsiooni. Üks neist oli välise jõuallikaga kuue jalaga masin, teine aga iseseisev kaheksajalgne masin. Maastikel kohanemise võime oli neil aga paraku kehv [5]. Esimene erinevaid kõnnakuid omaks võtnud liikur oli General Electric poolt loodud 1,4 tonni kaalunud neljajalgne robot, mille iga jalg oli kolme liikuvusastmega ning liigendeid käitati läbi hüdrosilindri ja 68kW sisepõlemismootoriga. See liikur ei olnud autonoomne, vaid nõudis juhtimiseks oskuslikku operaatorit [5]. Esimene elektrilise käivitamisega ja arvuti juhtimisel toimiv neljajalgne robot töötati Bob McGhee ja Andrew A. Frank poolt välja 1966. aastal. Iga jalg oli kahe liikuvusastmeline ning iga liigendit käivitati välise toitega elektrimootoriga ja kiiruse reduktoriga. Lihtne digiloogika koordineeris kahte erinevat kõnnakut. Roboti peamiseks puuduseks oli asjaolu, et see liikus ainult sirgjooneliselt, suutmata pöörata [5].

1.2 Jalgsete liikurrobotite rakendusvaldkonnad

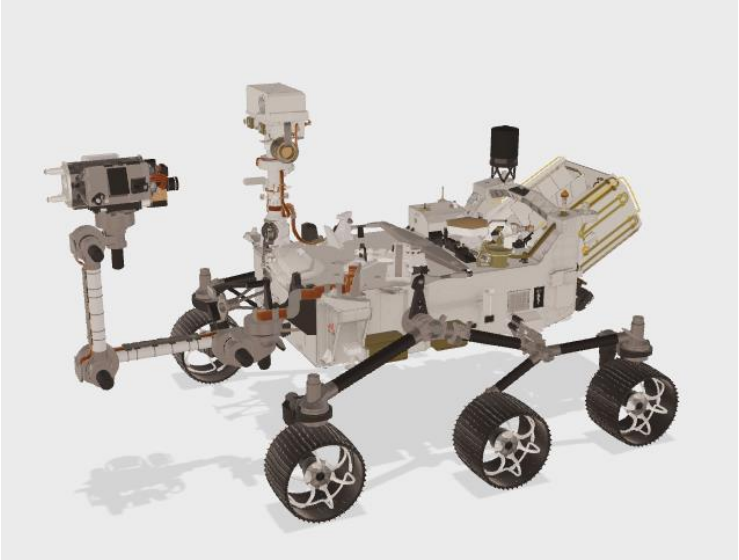
Jalgsed robotid on efektiivsed ebaühtlase pinnaga keskkonnas liikumiseks ning vältimatud abivahendid inimesele ohtlikes oludes tegutsemiseks. Näiteks katastroofipiirkondades on võimalik roboteid kasutada otsingutel ja päästetöödel , nagu seda pakub Boston Dynamics valmislahenduse kujul neljajalgne Spot robot (Joonis 1.1) [1]. Nad on võimelised läbima erinevaid takistusi, mis traditsiooniliste ratas- või roomiksõidukitele on ületamatud. Avalikus ruumis turvalisuse tagamiseks saab jalgseid roboteid kasutada linnakeskkonna jälgimiseks või kahtlaste esemete kahjutuks tegemisel.

Joonis 1.1 Boston Dynamics Spot neljajalgne liikurrobot [1]

Keskkonnaseirel pääsevad jalgsed robotid ligi kaugematele või ökoloogiliselt tundlikumatele aladele näiteks elusloodu jälgimiseks, ökosüsteemide uurimiseks või kliimamuutuste kohta andmete kogumiseks. Jalgseid roboteid saab kasutada ka ebatasastel maastikel põllukultuuride istutamiseks ja jälgimiseks ning saagikoristuseks, minimeerides seeläbi saagi kahjustamist. Ka metsatöödel prototüübiti 1995 aastal kõndivat harvester liikurit (Joonis 1.2) [6].

Joonis 1.2 Plustech poolt prototüübitud jalgadel kõndiv harvester [6]

Ehituse valdkonnas võivad jalgsed robotid abistada materjalide toimetamist raskesti ligipääsetavatesse kohtadesse või infrastruktuuri, nagu näiteks sildade ja torustike kontrollimisel ja hooldamisel, nagu Siemens poolt välja arendatud toru sees roniv robot [5]. Militaarvaldkonnas võivad jalgsed robotid leida rakendust luurel, ohtliku keskkonna uurimisel ja varustuse transpordil juhul, kui traditsioonilised sõidukid osutuvad ebapraktilisteks nagu näiteks Boston Dynamics poolt arendatud BigDog neljajalgne robot [7]. Kosmoseuuringutel on jalgsed robotid kasulikud eelkõige proovide kogumisel ja analüüsimisel nagu 2020. aastal NASA missiooni raames Marsil tegutsema saadetud Perserverance kulgur (Joonis 1.3), mis rakendab hübriidliikumist ehk jalgade poolt abistatud ratastel liikumist [8].



Joonis 1.3 NASA Perseverance kulgur [8]

Hariduses ja teadustöös on nimetatud robotid väärtuslikud vahendid õpetamaks robootikakontseptsioone või toimimaks eksperimentaalsete platvormidena uute algoritmide ja juhtimismeetodite testimisel. Jalgseid roboteid kasutatakse ka meelelahutuslikel eesmärkidel näiteks lõbustusparkides, et pakkuda külastajatele ainulaadseid ja dünaamilisi elamusi. Tulevikuvaates võiksid täiustatud jalgsed robotid olla koduabilisteks nii majapidamistöödes, esemete kandmisel kui raskendatud liikumisega inimestele.

1.3 Jalgsete liikurrobotite arendustrendid

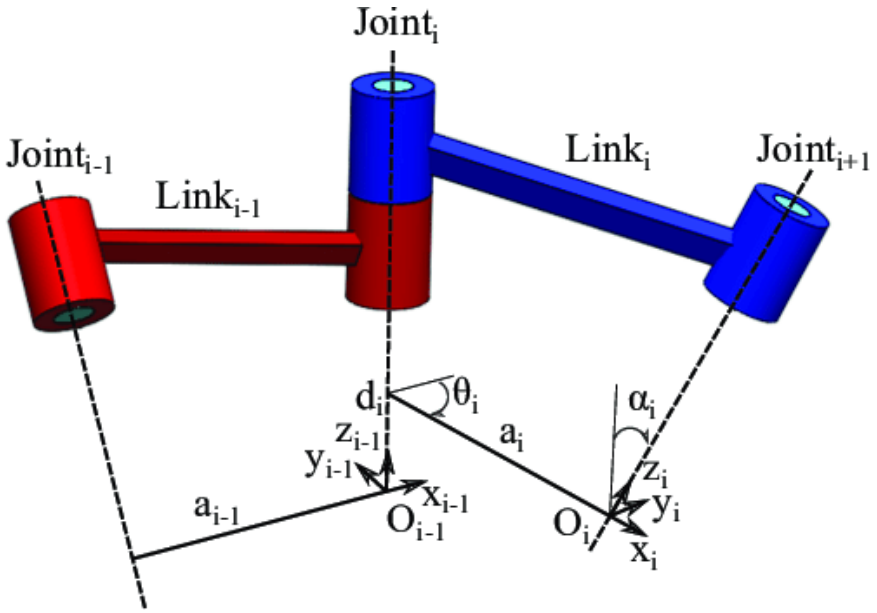
Üheks märkimisväärseks uueks manipuleerimismeetodi lähenemisviisiga uurimis- ja arendusvaldkonnaks robootikas on pehme robootika. Pehmed robotid asendavad tüüpiliste robotite jäigad lülid ja liigendid painduvate ja elastsete membraanidega, mis vähendavad kokkupõrgete korral tekkivat kahju nii robotile kui ka keskkonnale. Pehme robootika lähenemisviisi rakendatakse ka liikurrobotitele, kus uuritakse erinevaid täitursüsteeme ja kõnnakuid pehmete kombitsatega liikumisel [9].

Viimaste aastate jooksul on populaarseks muutunud drooni parvede valgusnäitused. Parvrobotid on tavaliselt väiksed üksteisele identsed robotid, mis suudavad suurearvuliselt üksteisega koostööd teha, et täita teatud ülesannet või lahendada kompleksset probleemi. Seda põhimõtet saab rakendada ka maapealsetele liikurrobotitele. Näiteks piiratud võimekusega robotid ühilduvad suurendamaks oma töövõimekust [10].

Veel üheks arendussihiks on energia tarbimise optimeerimine. Jalgsete liikurrobotite peamine nõrkus on nende energia tarbimine. Liikurrobotid peavad töötama iseseisva energiaallika pealt, mis tähendab, et järjestikku töötamise aeg on piiratud. Eriti piiratud on jalgsete liikurrobotite tööaeg, sest nende jalgade ja liigendite ajamite suure arvu tõttu tarbivad nad kordades rohkem energiat kui ratastel või roomikutel liikuvad robotid. Seetõttu on energia tarbimise optimeerimine üks olulisemaid arengusuundi. Optimeerimist uuritakse näiteks simulatsiooni keskkondades, kus rakendatakse kindlaid algoritme, et energia tarbimist vahendada, säilitades optimaalse liikumisvõimekuse [11].

2. KUUEJALGSE ROBOTI KINEMAATIKA

Jalgsete liikurrobotite projekteerimisel on esimeseks etapiks selle kinemaatika mudeli loomine. Kinemaatika on füüsika ja matemaatika alamosa, mis kirjeldab objektide ja kehade liikumist arvestamata nendele mõjuvate jõududega. Robootikas kasutatakse kinemaatikat roboti liikumise ja positsioneerimise uurimiseks ning analüüsimiseks. Kinemaatikat jagatakse kaheks: otseteisendus ja pöördteisendus. Kinemaatika ülesandeid saab lahendada mitmel erineval viisil, millest populaarseim on homogeense teisendusmaatriksi esitlusviis. Teisendusmaatriks on 4x4 maatriks, mis kasutab ära rotatsioonimaatriksit ja asendivektorit, et kirjeldada teljestiku asukohta ja orientatsiooni ruumis. Teisendusmaatriksi meetodile saab lisaks rakendada Denavit-Hartenberg (edaspidi DH) esitusviisi, mis kasutab ainult nelja parameetrit kahe teljestiku vahelise teisenduse kirjeldamiseks. Nendeks parameetriteks on (Joonis 2.1): lüli pikkus a, lüli väändenurk α (alfa), lüli nihe d ja lüli pöördenurk θ (teeta). Lüli pikkus a on leitava teljestiku kaugus eelmisest teljestikust mööda eelmise teljestiku x-telge. Väändenurk α (alfa) määrab leitava teljestiku pöördenurga ümber eelmise teljestiku x-telje, kus pöörlemine toimub vastupäeva. Lüli nihe d on leitava teljestiku kaugus eelmisest teljestikust mööda eelmise teljestiku z-telge. Lüli pöördenurk θ (teeta) määrab leitava teljestiku pöördenurga ümber eelmise teljestiku z-telje, kus pöörlemine toimub vastupäeva [12].



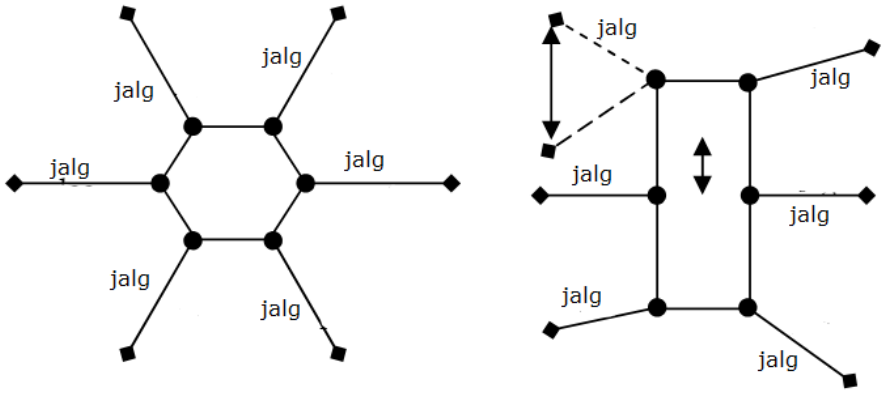
Joonis 2.1 DH parameetrite kujutis [13]

2.1 Geomeetrilised omadused ja parameetrid

Esimese etapina roboti kinemaatiliste mudelite loomisel on vajalik määrata roboti füüsilised ja geomeetrilised parameetrid. Nende alla kuuluvad keha ja jalgade topoloogiad, jalgade konfiguratsioonid, liikuvusastmed ja orientatsioonid, ning kinemaatilisteks arvutusteks vajalikud geomeetrilised dimensioonid. Kõige lihtsam on luua robotit, millel on võimalikult vähe erinevaid elemente, seega on antud töös projekteeritaval kuuejalgsel robotil kõik jalad identsed.

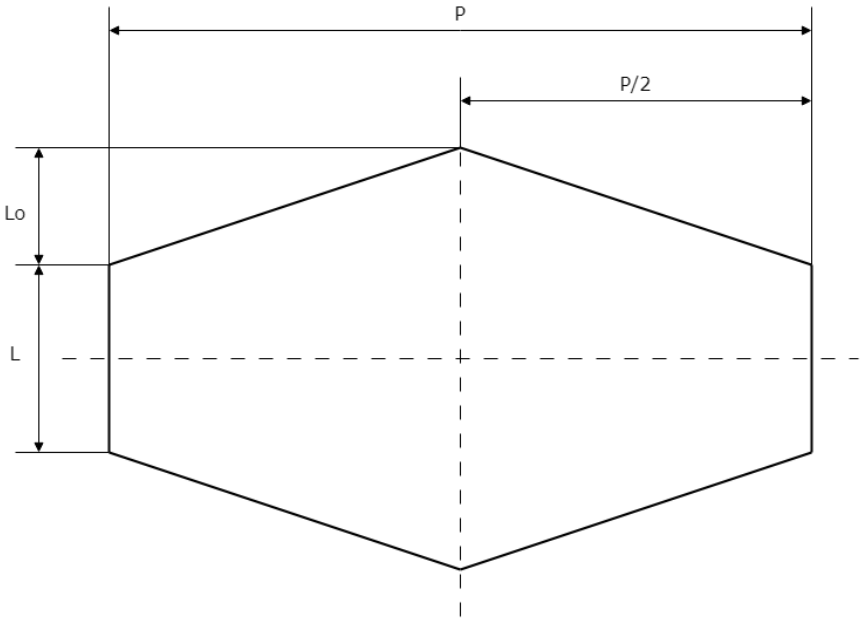
2.1.1 Roboti kere/keha

Kuuejalgseid roboteid saab keha arhitektuuri poolest jagada kahte kategooriasse (Joonis 2.2). Nendeks on radiaalselt- ja bilateraalselt sümmeetrilised kehad. Radiaalselt sümmeetrilised kehad on ringi või korrapärase kuusnurga kujulised, kus kuus jalga on ümber keha võrdselt jaotatud. Bilateraalsed on ristkülikud või kuusnurgad, mille kaks vastastikust külge on ülejäänutest lühemad ning jalad on jaotaud kere kahele poole sümmeetriliselt.



Joonis 2.2 Kuuejalgse roboti keha tüübid [14]

Antud töös kasutatakse bilateraalselt sümmeetriline keha, sest sellel on parem pikisuunaline stabiilsuse tegur, mis sobib paremini ühesuunalisel liikumisel [15] ning võimaldab visuaalselt tuvastada roboti orientatsiooni. Tavapärase ristküliku asemel on roboti keskmised jalad kehast väljapoole toodud, parendades nende liikumisruumi ja suurendades pinnaga kontaktis olevate jalgade tippude poolt tekitatud stabiilsuse hulknurka.

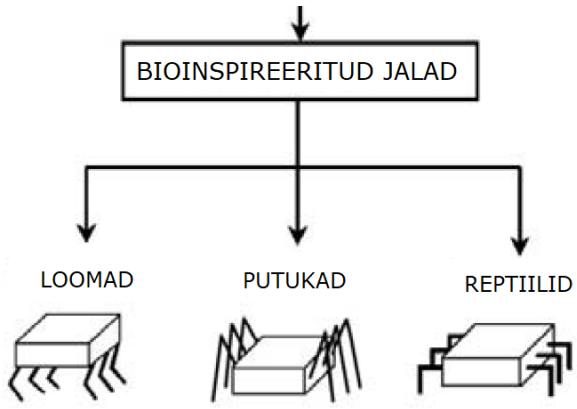


Joonis 2.3 Töös kasutatava roboti keha geomeetriline kujutis

Joonis 2.3 defineeritud kere kuusnurga iga tipp on jala ühenduspunkt kehaga. Keha on sümmeetriline läbi sidestatud joonte ning parameetriteks on keha laius *L*, keskmiste jalgade kaugus teistest *Lo* ja keha pikkus *P*.

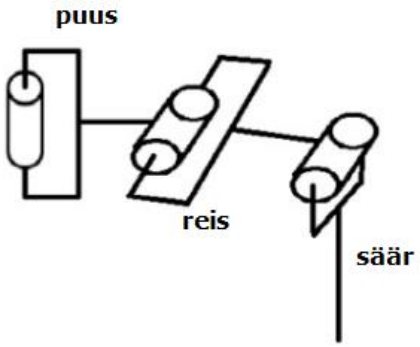
2.1.2 Roboti jalad

Kuuejalgsete robotite jalad on tüüpiliselt inspireeritud looduses leiduvate putukate jalgadest. Jalal on vaja vähemalt kaks liikuvusastet (edaspidi DOF), et saaks robot saaks liikuda, kuid see piirab jala tipu juhtimist kolmemõõtmelises ruumis. Kõige populaarsem on kolm DOF jalad, millel on puus, reis ja säär.



Joonis 2.4 Loodusest inspireeritud jalgade tüübid [15]

Jalad saavad inspiratsiooni loodusest: loomadelt, putukatelt ja reptiilidelt (Joonis 2.4). Selle järgi saab nende jalgade konfiguratsioone klassifitseerida kolmeks. Loomadele omaselt on jalad liikumissuunaga samas suunas orienteeritud ja tüüpasendina on põlved puusadest madalamal. Putukatele omaselt on jalad liikumissuunaga risti ning põlvi hoitakse tüüpiliselt puusadest kõrgemal. Reptiilidele omaselt on jalad liikumissuunaga risti, kuid põlved on puusadega samal kõrgusel. Teiseks tähtsaks elemendiks on jalgade liikuvusastmete DOF arv. Kuigi kolmemõõtmelises ruumis liikumiseks on vaja kahte üksteisega risti orienteeruvat pöörd liigendit, kasutatakse tüüpiliselt kuuejalgsete roboti puhul 3 DOF jalgu. Kolm DOF võimaldab jala positsioneerimist kui ka piiratud orienteerimist keskkonnas ning sellest piisab kuuejalgse roboti efektiivseks manööverdamiseks.



Joonis 2.5 Töös kasutatava jala konfiguratsiooni kujutis [15]

Antud töös kasutatakse putukalt inspireeritud jala konfiguratsiooni, millel on 3 DOF ja rakendab kolme pöörd liigendite (Joonis 2.5). Puusa liigend pöörleb ümber vertikaaltelje ning reie ja sääre liigendid pöörlevad ümber horisontaaltelje.

2.2 Kinemaatika otseülesanne

Kinemaatika otsülesanne kirjeldab kinemaatilise ahela lülide teljestike alguspunkte, positsioone ja orientatsioone kui on defineeritud liigendite pöördenurgad. Otseülesande lahendamise eesmärgiks on välja selgitada kinemaatilise ahela viimase lüli ehk lõpplüli ehk jala tipu teljestiku asukoht baasteljestiku suhtes. Kuuejalgse roboti puhul lahendatakse esimesena ühe jala kinemaatika otseülesanne ja seejärel keha otseülesanne, millele ühendatakse kõigi kuue jala kinemaatilised mudelid saades kogu roboti mudeli.

2.2.1 Ühe jala otseülesanne

Ühe jala otseülesande puhul on kinemaatiliseks ahelaks jala kolm lüli. Ülesande lahendatakse kasutades DH meetodit. Esiteks määratakse iga lüli teljestikule DH parameetrid. DH parameetreid on mugav esitada tabeli kujul, kus iga rida kirjeldab jala ühe lüli teljestikku.

Tabel 2.1 Roboti jala DH parameetrid

Lüli nr	a / mm	α / rad	d / mm	θ / rad
1	50	π/2	0	0
2	70	0	0	-π/2
3	100	0	0	0

Defineeritud DH parameetritega saab tuletada jala iga lüli *i* teisendusmaatriksi jala eelmise lüli *i-1* suhtes. Teisendusmaatrikseid saab esitada kujul [12]:

$$
\begin{aligned}
{}^{i-1}_{i}T &= Trans_z(d_i) \cdot Rot_z(\theta_i) \cdot Trans_x(a_i) \cdot Rot_x(\alpha_i) \\
&= \begin{bmatrix} cos\theta_i & -cos\alpha_i \cdot sin\theta_i & sin\alpha_i \cdot sin\theta_i & a_i \cdot cos\theta_i \\ sin\theta_i & cos\alpha_i \cdot cos\theta_i & -sin\alpha_i \cdot cos\theta_i & a_i \cdot sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{2.1}
$$

$$
Trans_z(d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}
$$

$$
Rot_z(\theta_i) = \begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & 0 \\ sin\theta_i & cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}
$$

$$
Trans_x(a_i) = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}
$$

$$
Rot_x(\alpha_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha_i & -sin\alpha_i & 0 \\ 0 & sin\alpha_i & cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}
$$

Kus $Trans_z(d_i)$ on asendimaatriks z-telje suhtes,

$Rot_z(\theta_i)$ on rotatsioonimaatriks, mis kirjeldab pöörlemist ümber z-telje,

$Trans_x(a_i)$ on asendimaatriks x-telje suhtes,

$Rot_x(\alpha_i)$ on rotatsioonimaatriks, mis kirjeldab pöörlemist ümber x-telje,

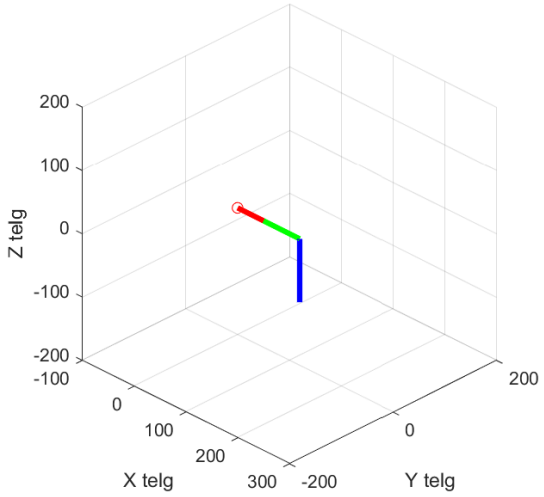
Enne iga lüli teisendusmaatriksi arvutamist peab defineerima jala baasteljestiku. Antud juhul defineeritakse see teisendusmaatriks ühikmaatriksina ehk algkoordinaadid on nullpunktis, sest ei arvestata roboti keha positsiooni ega orientatsiooniga.

$$T_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

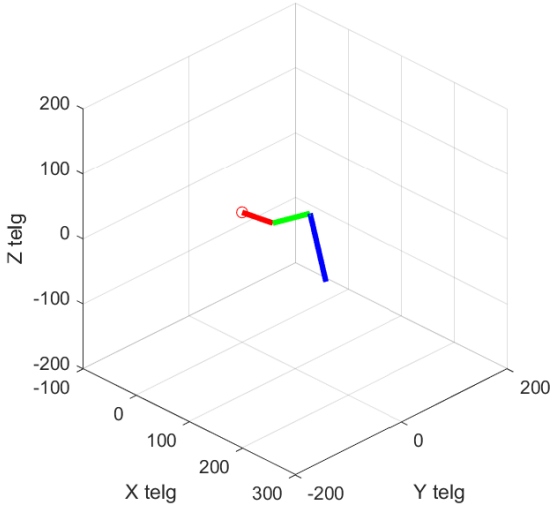
Kuna eesmärgiks on viimase lüli teljestiku asukoht baasteljestiku suhtes, siis saab seda leida korrutades läbi jala kõigi kolme lüli teisendusmaatriksid.

$$ {}^0_3T = {}^0_1T \cdot {}^1_2T \cdot {}^2_3T \tag{2.7}$$

Mudeli korrektsuse kontrolliks ja edasise modelleerimise lihtsustamiseks koostati Matlab tarkvaras programm (Lisa 1) ja visualiseeriti jala algasend (Joonis 2.6) ja asend, mille puhul määrati igale liigendile sisendnurk (Joonis 2.7).



Joonis 2.6 Jalg algasendis



Joonis 2.7 Jalg asendis, kus $\theta_1 = 10^\circ$, $\theta_2 = 30^\circ$, $\theta_3 = -15^\circ$

2.2.2 Keha otseülesanne

Keha otseülesande lahendamine ja mudeli loomine määrab ära keha dimensioonid ja ühendab kõigi kuue jala teljestikud keha baasteljestikuga. Lisaks võimaldab keha otseülesanne juhtida roboti orientatsiooni ümber oma baasteljestiku z-telje ehk roboti liikumissuunda ning vastavalt sellele mõjutada jalgade baasteljestike teisendusmaatrikseid. Jalgade sidumisel kehaga peab defineerima jala baasteljestiku pöörenurga ümber z-telje ehk jala algse orientatsiooni. Keha otseülesannet lahendatakse DH meetodiga ning määratakse DH parameetrite tabel.

Tabel 2.2 Roboti keha DH parameetrid

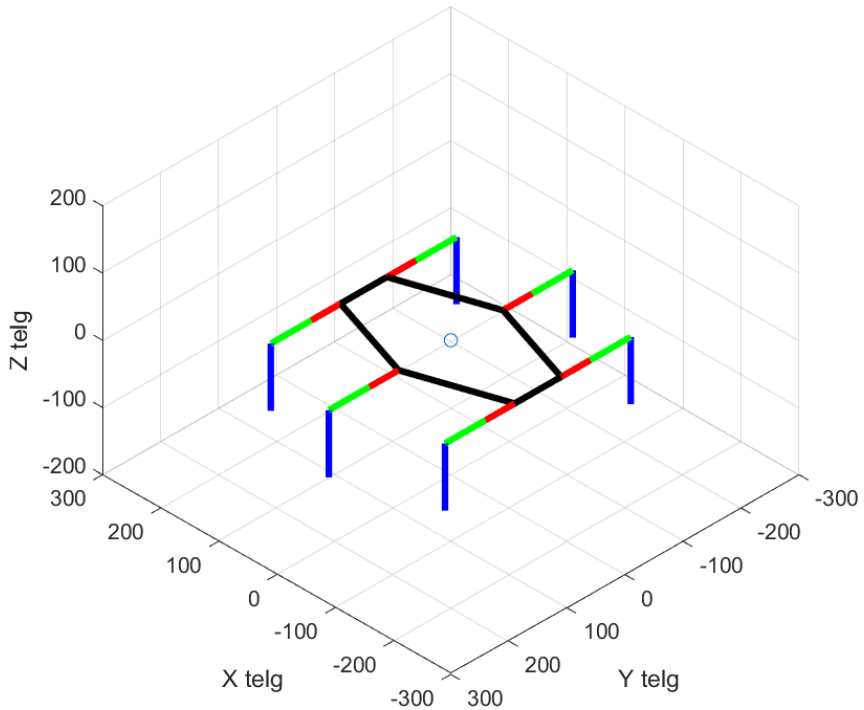
Teljestiku nr	a / mm	α / rad	d / mm	θ / rad
0	0	0	0	θ_0
01	0	π/2	0	0
1	P/2	0	L/2	0
2	0	0	L/2+Lo	0
3	-P/2	0	L/2	0
4	-P/2	0	-L/2	0
5	0	0	-L/2-Lo	0
6	P/2	0	-L/2	0

Kus teljestik 0 on baasteljestik, θ_0 on roboti liikumissuunda mõjutav pöördenurk, suurused *L*, *Lo* ja *P* on (Joonis 2.3) välja toodud parameetrid ning teljestikud 1-6 on jalgade kinnituskohtade teljestikud.

Keha otseülesannet lahendatakse samamoodi nagu ühe jala otseülesannet, kuid teljestikud 1-6 ei mõjuta üksteist, vaid on ainult mõjutatud baasteljestikust ehk nende teisendusmaatriksid esinevad kujul.

$$ {}^{0}_{i}T = {}^{0}_{01}T \cdot {}^{01}_{i}T \quad , i = 1 \dots 6 \tag{2.8} $$

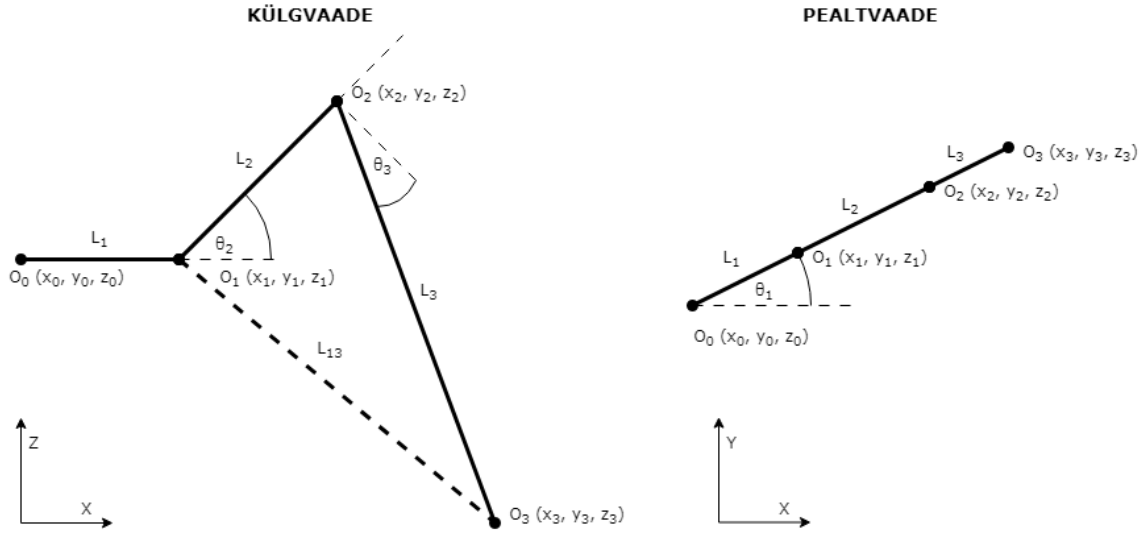
Keha ja selle külge ühendatud jalgade otseülesande lahendamiseks ning visualiseerimiseks sai loodud Matlab programm (Lisa 2), kus keha parameetrid on sisendmuutujatena suurustega $L = 80\,mm$, $Lo = 50\,mm$, $P = 300\,mm$ (Joonis 2.8).



Joonis 2.8 Roboti keha otseülesande kujutis koos jalgadega nullasendis

2.3 Kinemaatika pöördülesanne

Kinemaatika pöördülesanne võimaldab tuletada kinemaatilise ahela iga liigendi pöördenurga asendit kui on teada ahela baasteljestiku ja lõpplüli teljestiku koordinaadid. See võimaldab roboti ajamite juhtimist soovitava asukoha järgi. Pikemate kinemaatiliste ahelate pöördülesande lahendamine võib kujuneda keerukaks või isegi võimatuks, sest soovitava asukohani jõudmiseks võib ahelal olla mitmeid või isegi lõpmatult erinevaid positsiooni ja orientatsioone. Teades, et roboti ühel jalal on ainult kolm ühe liikuvusastmega liigendit on lihtsam pöördülesande lahendamisel kasutada geomeetrilist lähenemist kasutades trigonomeetrilisi valemeid.



Joonis 2.9 Roboti jala joonised külg- ja pealtvaates pöördülesandeks vajalike parameetritega

Eesmärgiks on koostada iga liigendi pöördenurga (θ_1, θ_2, θ_3) seos soovitava lõpplüli asukohaga, kui on teada lülide pikkused, jala algkoordinaadid ja lõpplüli asukoha koordinaadid. Iga liigendi pöördenurgad on leitavad järgevate valemitega:

$$\theta_1 = \tan^{-1}\left(\frac{y_3 - y_0}{x_3 - x_0}\right) \tag{2.9}$$

$$\theta_2 = cos^{-1}\left(\frac{z_0 - z_3}{a_{13}}\right) + cos^{-1}\left(\frac{L_2^2 + L_{13}^2 - L_3^2}{2 \cdot L_2 \cdot L_{13}}\right) - \frac{\pi}{2} \tag{2.10}$$

$$\theta_3 = cos^{-1}\left(\frac{L_2^2 + L_3^2 - L_{13}^2}{2 \cdot L_2 \cdot L_3}\right) - \frac{\pi}{2} \tag{2.11}$$

$$a_{13} = \sqrt{z_3^2 + \left(\sqrt{x_3^2 + y_3^2} - L_1\right)^2} \tag{2.12}$$

Kus L_1, L_2 ja L_3 on jala lülide pikkused, mis on defineeritud [Tabel 2.1] ehk

$L_1 = 50\,mm$, $L_2 = 80\,mm$ ja $L_3 = 100\,mm$,

L_{13} on jala tipu kaugus reie liigendist, mm.

Koostatud võrrandid annavad pöördenurgad radiaanides, mida saab kergelt konverteerida kraadidesse. Valemeid kontrolliti Matlab-ga. Koostati programm, mille sisenditeks olid lõpplüli x, y ja z koordinaadid ning väljundiks jala iga lüli pöördenurgad kraadides (Lisa 3). Saadud nurgad sisestati jala otseülesande Matlab programmi (Lisa 1) ja võrreldi saadud koordinaate algsete sisendkoordinaatidega.

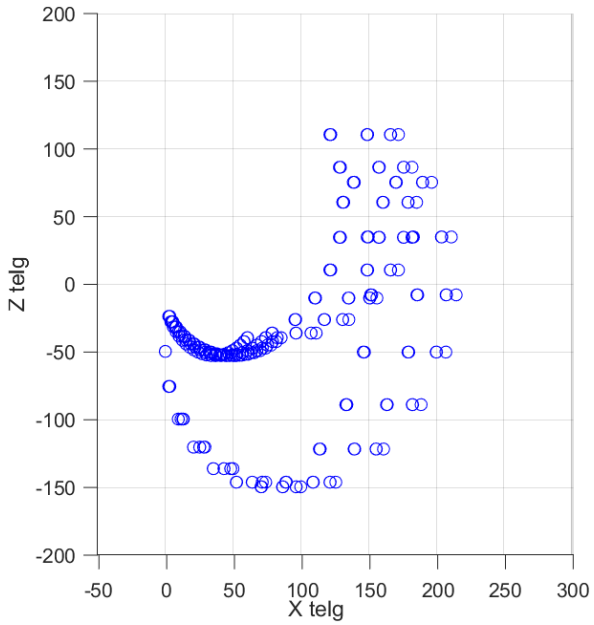
Tabel 2.3 Pöördülesande kontroll otseülesandega

Katse nr	**Pöördülesande sisendi koordinaadid (x, y, z)**	**Pöördülesande väljundi pöördenurgad (θ_1; θ_2; θ_3) / °**	**Otseülesande väljundi koordinaadid (x; y; z)**
1	100, 50, -80	26,5651; 16,4806; -19,5306	99,9999; 50,0001; -80
2	60, 0, -110	0; -22,1376; -11,1196	59.9999; -6,7356e-15; -110
3	70, -20, -60	-15,9454; 27,0694; -50,3546	70; -20; -60

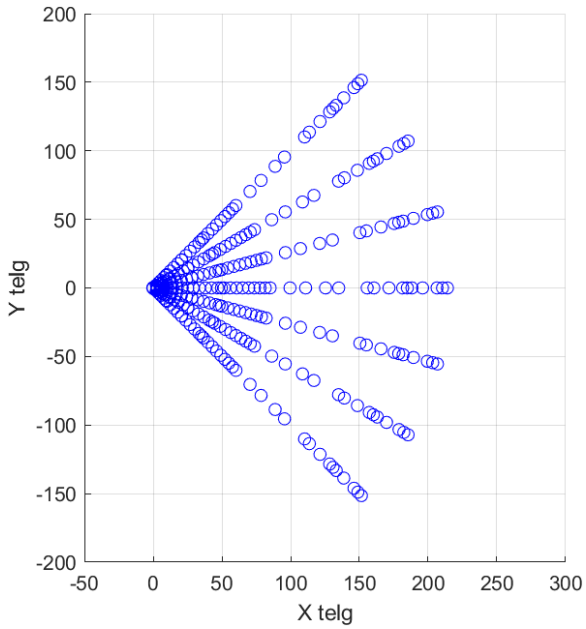
Tabelisse kantud tulemuste järgi saab järeldada, et pöördülesanne on kooskõlas otseülesandega ja valemid on korrektsed.

2.4 Tööruum ja sammu trajektoor

Robot manipulaatori tööala ehk tööruum on ruumala, mille sees asuvad lõpplüli kõikvõimalikud asukohad. Tööala defineerimine aitab vältida jalgade kokkupõrkeid üksteisega ja määrata sammu trajektoori parameetreid. Tööala efektiivsemaks defineerimiseks peab igale liigendile määrama pöördenurga limiidid. Antud töös määrati liigendite pöördenurkade limiitideks θ_1 = {-45, 45}, θ_2 = {-45, 60}, θ_3 = {-60, 60}. Tööala visualiseerimiseks loodi Matlab programm (Lisa 4), mis rakendab jala otseülesannet (Joonis 2.10) ja (Joonis 2.11). Programm arvutab jala lõpplüli koordinaadid kui liigendite pöördenurgad muutuvad miinimumidest maksimumideni ja vastupidi. Koordinaadid väljastatakse iga 15° pöördenurga muutuse tagant.



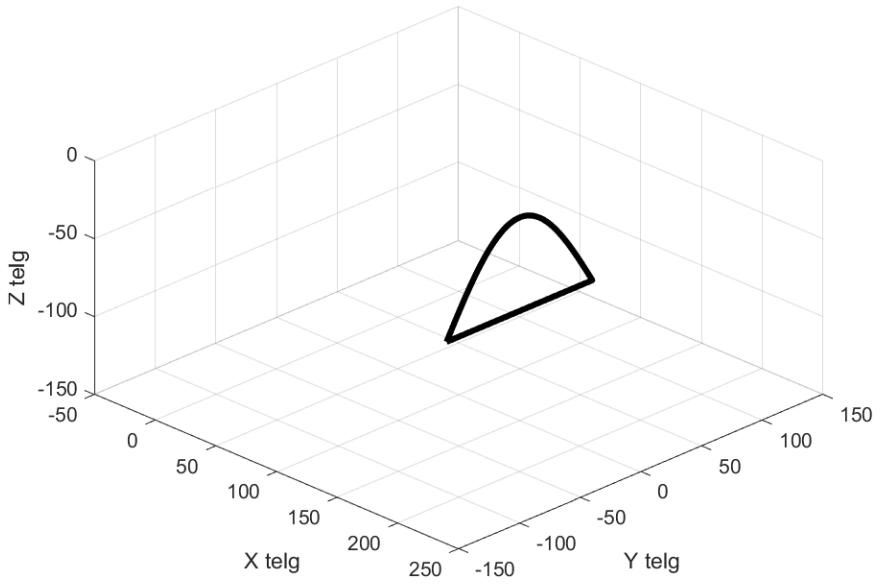
Joonis 2.10 Jala tööala külgvaade



Joonis 2.11 Jala tööala pealtvaade

Sammu trajektoor on jala lõpplüli teekond, mida see lõpplüli läbib ühe sammu tegemisel. Sammu trajektoori defineerimine on oluline roboti juhtimisel. Trajektoor näitab ära kasutatavad koordinaadid, millest saab kinemaatika pöördülesandega tuletada jala ajamite pöördenurgad. Sammu omadused olenevad jala tööalast ja sammu alguspunkti asukohast tööalas. Trajektoor koosneb kahest osast: tõuke faasist kui jala lõpplüli on maapinnaga kontaktis ehk robot tõukab enda keha liikumissuunas edasi ning tõste faasist, kus roboti jalg liigub õhus järgmise sammu algpunkti. Tõuke faasi saab kujutada sirgjoonena maapinnal ja tõste faasi kaarena õhus, mida kõige kergem

kirjeldada siinuse kujul vahemikus {0; π/2}. Sammu trajektoori parameetriteks on sammupikkus, jala tõste maksimaalne kõrgus ehk tõste amplituud ja sammu alguspunkti koordinaadid. Need mõlemad tulenevad tööruumist. Tulenevalt antud roboti tööruumist on sammupikkuseks määratud $L_{sp} = 120\ mm$, kõrguseks $H = 60\ mm$ ja sammu alguspunkti koordinaatideks $x_{s0} = 150$, $y_{s0} = -\frac{L_{sp}}{2}$ ja $z_{s0} = -80$ (Joonis 2.12). Sammu trajektoori arvutamiseks ja visualiseerimiseks koostati Matlab programm (Lisa 5).



Joonis 2.12 Sammu trajektoor jala koordinaatraamistikus

Kui trajektoor on koordinaatraamistikus defineeritud, on võimalik määrata selle läbimise liikumisprofiil. Liikumisprofiil määrab ära jala lõpplüli asukoha muutuse, lineaarkiiruse ja -kiirenduse profiilid (Joonis 2.13), mis on vajalikud dünaamika analüüsis jala liigendite dünaamiliste jõumomentide arvutamisel. Üks tüüpilisemaid liikumise profiile on kolmnurkne profiil. Kolmnurkne profiil jagab trajektoori ühe faasi läbimise kaheks pooleks: esimeses pooles on konstantne kiirendus ja teises pooles konstantne pidurdus ehk kiirendus on negatiivne [16]. See eemaldab ka sammu faasi alguses ja lõpus järsud liikumised. Antud töö raames on olulised ainult liigendite pöördkiirendused sammu tõuke faasis ning sellepärast defineeritakse ainult sellele faasile kolmnurkset profiili.

Kuna sammu trajektoor kajastab roboti otseliikumist, siis asukoht muutub ainult y-teljel ja lõpplüli x ja z koordinaate saab vaadata kui konstante. Rakendades seda sammu trajektoori tõuke faasis, saab jala tipu asukoha muutust ajas kirjeldada järgmiste valemitega.

$$p_k = y_s + \frac{\frac{L_{sp}}{2} \cdot t^2}{\frac{t_t^2}{2}} \tag{2.13}$$

$$p_p = y_s - \left(\frac{\frac{L_{sp}}{2} \cdot (t - t_t)^2}{\frac{t_t}{2}} \right)^2 + L_{sp} \quad (2.14)$$

Kus p_k on asukoht kiirendamise ajal,

p_p on asukoht pidurdamise ajal,

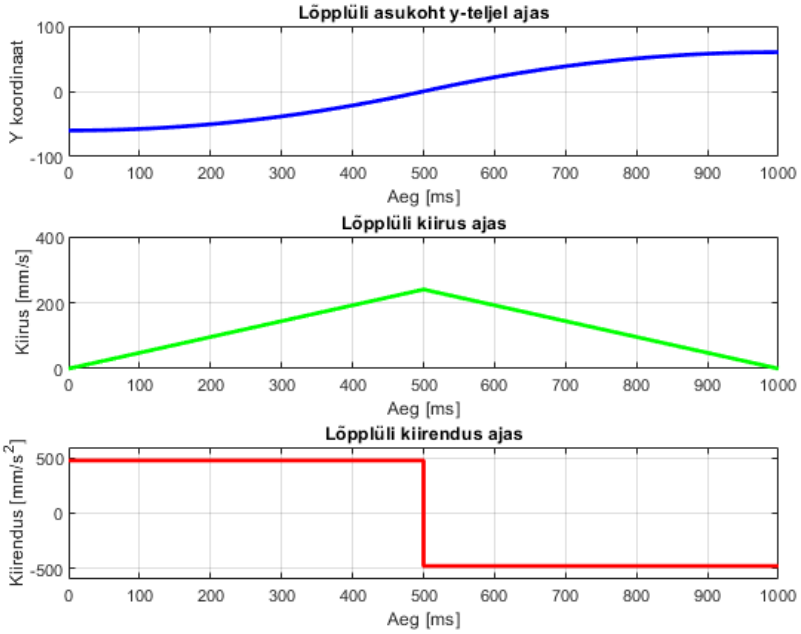
y_s on sammu alguspunkti y koordinaat,

L_{sp} on sammupikkus, mm,

t_t on terve tõuke faasi läbimiseks kuluv aeg, ms,

t on hetke aeg, ms

Lõpplüli kiiruse saab tuletada võttes asukoha ajatuletise $v = \frac{dp}{dt}$ ja lõpplüli kiirenduse saab tuletada võttes kiiruse ajatuletise $a = \frac{dv}{dt}$. Saadud võrrandeid kasutati Matlab (Lisa 6) programmis, kus asukoha muutujad on trajektoori määramisel defineeritud suurused ja tõuke faasi läbimiseks kuluv aeg on üks sekund ehk $t_t = 1\,s = 1000\,ms$.



Joonis 2.13 Lõpplüli liikumisprofiil tõuke faasis

2.5 Liigendite jõumomendid

Enne roboti prototüübi ehitamist on oluline veel välja selgitada, millised jõumomendid avalduvad roboti liigenditele. Jõumomente on vaja teada, et roboti ehitamisel teha õiged otsused ajamite valikul. Kuna ei ole teada kõiki roboti füüsilisi parameetreid nagu

näiteks detailide massid ja massikeskmed, peab tegema jõumomentide arvutamisel eelduseid. Eelduste tõttu ei ole tulemused täpsed. Jõumoment kirjeldab võimet tekitada pöördliikumist ümber telje. Ajami jõumoment koosneb lihtsustatud viisil põhiliselt kahest osast:

$$\tau_m = (\tau_d + \tau_s) \cdot ot \tag{2.15}$$

Kus, τ_d on ajamile avalduv dünaamiline moment, Nm,

τ_s on ajamile avalduv staatiline moment, Nm,

ot on lisategur, et arvestada lihtsustustest tekkiva ebatäpsusega

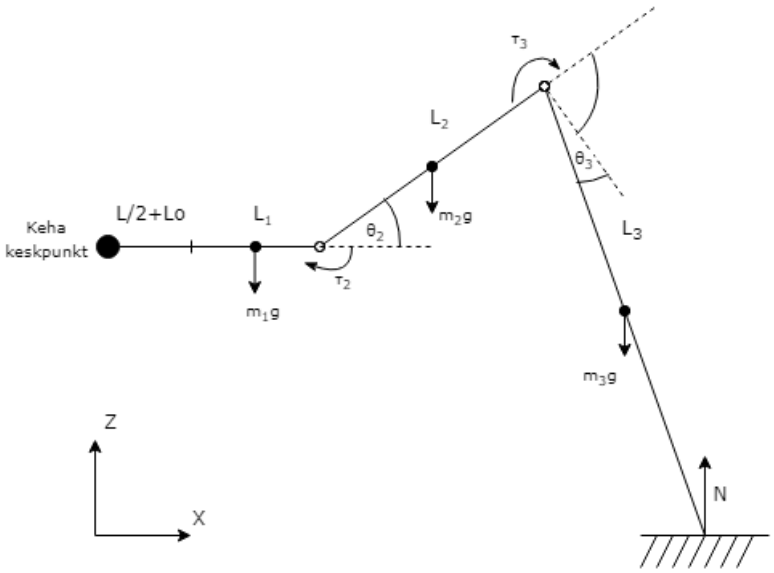
Dünaamiline moment tuleneb inertsmomendist ja ajamite pöördkiirendusest liikumise ajal. Staatiline moment tuleneb aga robotile mõjuvast raskusjõust ning oleneb roboti asendist. Antud alapeatükis analüüsitakse ajamile mõjuvaid momente sammu trajektoori tõuke faasi järgi, sest sel ajal kannavad jalgade ajamid kõige suuremat koormust kogu töötsükli jooksul. Analüüsis rakendatakse mitmeid lihtsustusi ja sellest tulenevalt on tulemused ebatäpsed. Eeldustest ja lihtsustustest tekkivate ebatäpsuste katteks on saadud staatiliste ja dünaamiliste momentide väärtused korrutatud läbi lisateguriga $ot = 1{,}5$.

2.5.1 Staatiline jõumoment

Staatiline moment ehk koormusmoment on moment, mida ajam peab rakendama, et hoida robotit ettenähtud asendis. Lihtsustamise eesmärgil arvestatakse antud töös ainult gravitatsioonist tuleneva raskusjõuga. Raskusjõudu avaldavad massikeskmed on keha geomeetrilises keskpunktis ja jalgade lülide keskpunktides. Seega koormusmoment avaldub valemiga:

$$\tau_s = F \times r = mg \times r \tag{2.16}$$

Analüüsitakse ainult keskmist parempoolset jalga, sest kõik jalad on identsed. Veel on tehtud lihtsustus, mis oletab, et ühe jala mass jaguneb võrdselt selle iga lüli vahel. Arvestades, et robot seisab kolme jala peal, jaguneb kogu roboti raskusjõud nende vahel võrdselt. Nende eeldustega saab arvutada jala liigenditele mõjuvad koormusmomendid lihtsalt kui vaadata pinnakontaktis olevat jalga kui tavalist statsionaarset robot manipulaatorit. Koostati vaba keha diagramm (Joonis 2.14), et visualiseerida olukorda ja arvutusteks vajalikke parameetreid.



Joonis 2.14 Roboti jala staatika diagramm

Kus N on pinnakontaktist tulenev reaktsioonijõud, mis on võrdeline kolmandiku kogu roboti raskusjõust. Diagrammist tulenevalt on võimalik reie ja sääre liigenditele mõjuvad koormusmomente arvutada võrranditega [17]:

$$\tau_{s2} = N[L_2 cos\theta_2 + L_3 cos(\theta_2 + \pi - \theta_3)] - \frac{L_2}{2} m_2 g \cdot cos\theta_2 - m_3 g \left[L_2 cos\theta_2 + \frac{L_3}{2} cos(\theta_2 + \pi - \theta_3)\right] \quad (2.17)$$

$$\tau_{s3} = N \cdot L_3 cos(\theta_2 + \pi - \theta_3) - \frac{L_3}{2} m_3 g \cdot cos(\theta_2 + \pi - \theta_3) \quad (2.18)$$

Kus *N* on kogu roboti raskusjõust kolmandik $N = \frac{1}{3}(m_{keha} + 6m_{jalg})g$,

m_2, m_3 on jala lülide massid, mis on kõik võrdselt kolmandik jala massist, kg,

L_2, L_3 on lülide pikkused, m,

g on gravitatsioonikiirendus, m/s^2

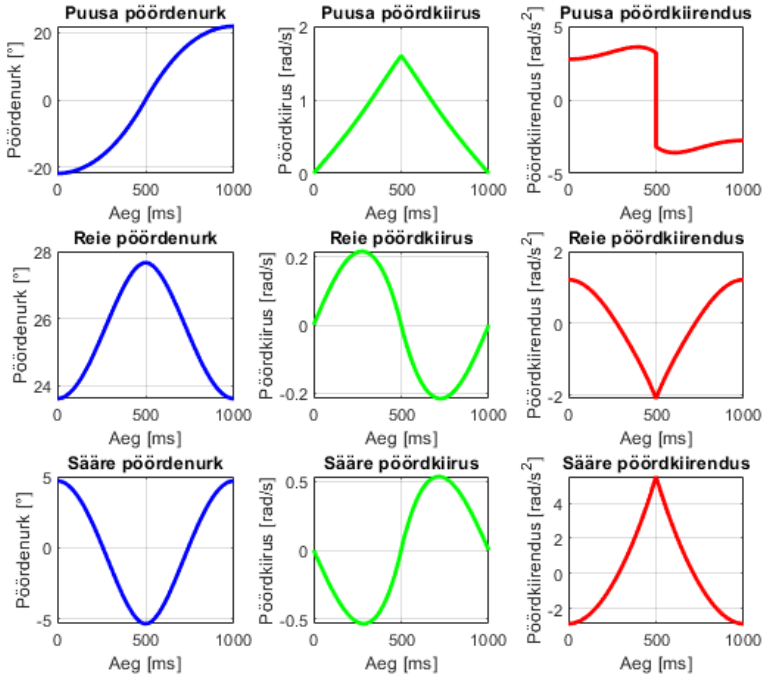
2.5.2 Dünaamiline jõumoment

Nagu varasemalt mainitud sõltub dünaamiline moment inertsmomendist ja pöördkiirendusest. Inertsmomendi arvutamisel on rakendatud lihtsustus, mis muudab kõik objektid kangi otsas olevateks punktmassideks. Seega dünaamilise momendi valem avaldub kujul.

$$\tau_d = I \cdot \alpha = m \cdot R^2 \cdot \alpha \quad (2.19)$$

Pöördkiirenduse määramiseks saab ära kasutada jala lõpplüli lineaarse liikumise profiili (Joonis 2.13), et tuletada jala iga liigendi pöördliikumine, -kiirus ja kiirendus (Joonis

2.15). Seda saab teha läbi kinemaatika pöördülesande asendades liigendite pöördenurkade valemites lõpplüli asukoha y koordinaadi muutuja lineaarliikumise võrranditega. Arvutamiseks koostati Matlab programm (Lisa 7), mis kalkuleerib välja jala iga liigendi pöördenurga, pöördkiiruse ja -kiirenduse tõuke faasi ajal.



Joonis 2.15 Liigendite pöördenurk, pöördkiirus ja -kiirendus tõuke faasis

Teades liigendite pöördkiirendusi tõuke faasi ajal on veel vaja arvutada liigenditele mõjuvad inertsmomendid. Nende inertside leidmisel rakendati lihtsustusi, mis paigutavad massi punktmassina keha keskpunkti. Lihtsustusest tulenevalt on massi kaugust liigendist sammu tõuke faasi ajal kergem arvutada. Liigendite dünaamilist momenti saab arvutada valemitega:

$$\tau_{d1} = \frac{1}{3}\left(m_{keha} + 3m_{jalg}\right) \cdot {L_{k1}}^2 \cdot \alpha_1 \quad (2.20)$$

$$\tau_{d2} = \frac{1}{3}\left(m_{keha} + 3m_{jalg} + \frac{m_{jalg}}{3}\right) \cdot {L_{k2}}^2 \cdot \alpha_2 \quad (2.21)$$

$$\tau_{d3} = \frac{1}{3}\left(m_{keha} + 3m_{jalg} + \frac{2m_{jalg}}{3}\right) \cdot {L_{k3}}^2 \cdot \alpha_3 \quad (2.22)$$

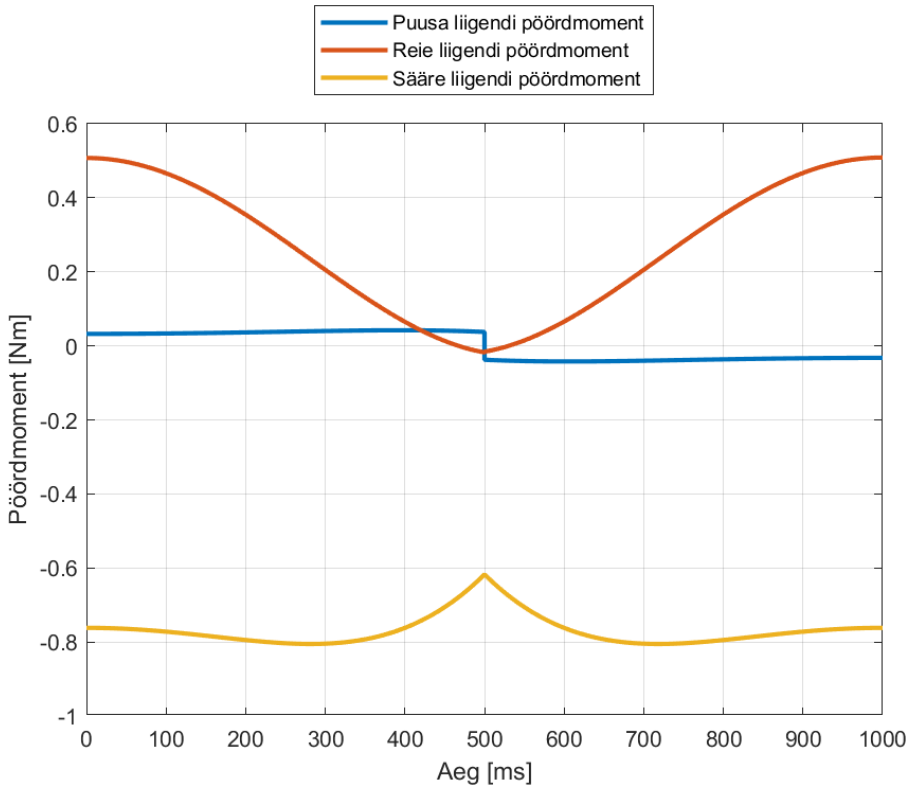
Kus,

$$L_{k1} = \frac{L}{2} + Lo \quad (2.23)$$

$$L_{k2} = \sqrt{{L_{k1}}^2 + {L_1}^2 - 2 \cdot L_{k1} \cdot L_1 \cdot cos(\pi - \theta_1)} \quad (2.24)$$

$$L_{k2} = \sqrt{\left(\sqrt{{L_{k1}}^2 + (L_1 + L_2 \cdot cos\theta_2)^2 - 2 \cdot L_{k1} \cdot (L_1 + L_2 \cdot cos\theta_2) \cdot cos(\pi - \theta_1)}\right)^2 + (L_2 \cdot sin\theta_2)^2} \quad (2.25)$$

Kasutades saadud võrrandeid arvutati iga liigendile mõjuvad jõumomendid sammu tõuke faasi ajal pöördliikumise profiili Matlab programmi täiendades (Lisa 7) ning visualiseeriti alapeatüki alguses defineeritud valemi (2.15) põhjal arvutatud tulemused graafikuna (Joonis 2.16). Arvutamisel kasutati eeldatavaid masside väärtuseid $m_{keha} = 2\ kg$ ja $m_{jalg} = 0{,}3\ kg$.



Joonis 2.16 Liigendite staatilise ja dünaamilise jõumomendi summad, mis on ohutusteguriga läbi korrutatud

Saadud tulemustes peab arvestama, et jõumomendi positiivsus ja negatiivsus näitab selle suunda. Arvutuste tulemustest saab teada, et suurim ajamilt nõutav jõumoment sammu tõuke faasi jooksul määratud eeldatavate parameetritega on $|\tau_3| = 0{,}8054\ Nm$.

Käesolevas peatükis defineeriti modelleerimise käigus mitmeid prototüübi ehitamiseks olulisi parameetreid. Defineeritud ja välja arvutatud parameetrid ning nende väärtused on koondatud kokku tabelisse (Tabel 2.4).

Tabel 2.4 Modelleerimisel defineeritud ja välja arvutatud olulised parameetrid

Parameetrite tähised	**Väärtused**	**Selgitus**
L, Lo, P	80, 50, 300 [mm]	Roboti kere dimensioonid: otsade laius, keskmiste jalgade lisalaius, pikkus
a_1, a_2, a_3	50, 70, 100 [mm]	Jala lülide pikkused: puus, reis, säär
θ_1 min ja max, θ_2 min ja max, θ_3 min ja max	{-45°, 45°}; {-45°, 60°}; {-60°, 60°}	Liigendite pöördenurkade limiidid
L_{sp}, H	120, 60 [mm]	Sammu parameetrid: sammupikkus, -kõrgus
x_{s0}, y_{s0}, z_{s0}	150, -60, -80	Sammu trajektoori algkoordinaadid jala koordinaatteljestikus
T_3	0,8054 [Nm]	Sammu tõuke faasis suurim liigenditele mõjuv jõumoment

Nendeks parameetriteks on roboti keha dimensioonid, jala dimensioonid, jala liigendite pöördenurkade limiidid, sammu parameetrid ja selle trajektoori koordinaadid jala baasteljestiku suhtes ning sammu tõuke faasi ajal liigenditele mõjuv maksimaalne jõumoment. Lisaks parameetritele tuletati ka roboti juhtimiseks vajalikud liigendite pöördenurkade leidmise valemid sammu trajektoori järgimisel.

3. ROBOTI KOMPONENDID

Füüsilise prototüübi ehitamisel esimeseks etapiks on selle elektrooniliste komponentide valimine. Kuuejalgsed robotid saavad autonoomsuse võimaluse tõttu koosneda paljudest erinevatest elektroonika komponentidest. Siin töös on eesmärgiks luua robotsüsteemi baasplatvorm ehk valitakse ainult baastöövõimekust tagavad komponendid. Nendeks on roboti liigendites asuvad ajamid, robotit juhtiv juhtseade, toiteallikas, ajamite kontroller ja vaheühendustena toite lüliti, pingemuundurid ning kaitseahel.

3.1 Ajamid

Roboti jalgade liigendite ajamitena on kõige kergem rakendada elektrimootoreid. Kuna roboti liikumist juhitakse asukoha järgi, siis on oluline, et roboti juhtseade teaks mootorite pöördenurki, kas ennustades või mootoritelt tagasisidet saades. Ajami pöördenurga ennustamist võimaldavad sammmootorid või servomootorid. Sammmootorid on energia ahned, mille tõttu need ei ole aku toitel põhinevatel süsteemidel efektiivsed. Seega kasutatakse antud töö raames roboti ajamitena hobi servomootoreid (edaspidi ka servo). Servomootorid on alalisvoolu mootorid, millele on sisseehitatud positsiooni tagasisideahel ja hammasrataste ülekande mehhanism. Servomootorite olulised parameetrid on maksimaalne jõumoment, pöörlemiskiirus, toitepinge, juhtsignaali tüüp ja servo suurus. Antud töös võrreldakse kolme potentsiaalset robotile sobivat servomootorit ning valitakse üks prototüübi ehitamiseks. Võrdlus viiakse läbi tabelis.

Tabel 3.1 Servomootorite võrdlus

Servo nimetus	**Pinge / V**	**Max jõumoment / kgf*cm**	**Pöörlemiskiirus / s/60°**	**Juhtsignaal**	**Dimensioonid / mm**
MG996R [18]	4,8-7,2	9,4-11	0,17-0,14	PWM	40,7*19,7*42,9
TD-8120MG [19]	4,8-7,2	20,5-22,8	0,18-0,14	PWM	40*20,5*40,5
LD-27MG [20]	6-7,4	20 @7,4 V	0,2 @7,4 V	PWM	54,5*20*47,5

Prototüübi jaoks valiti TD-8120MG servomootorid (Joonis 3.1), sest see on piisavalt võimekas, valikust kõige väiksemate dimensioonidega ning oli kergesti kättesaadav.

Joonis 3.1 TD-8120MG servo [19]

3.2 Juhtseade

Robotsüsteemi juhtseade on arvuti või kontroller, mis täidab kõiki roboti juhtimisfunktsioone. Lihtsamatele robotsüsteemidele sobivad juhtseadmeks mikrokontrollerid. Mikrokontrollerid, nagu Arduino poolt kasutatav ATMega328P, suudavad lahendada lihtsat trajektoori planeerimist ja läbimist, kuid jäävad võimekuse poolest puudulikuks autonoomse roboti juhtimisel. Autonoomsed jalgsed liikurrobotid vajavad mikrokontrolleritest võimsamaid juhtseadmeid. Nendeks võimsamateks juhtseadmeteks sobivad monoplaatarvutid nagu, neist populaarseim, Raspberry Pi (Joonis 3.2).

Joonis 3.2 Raspberry Pi 3 Mudel B+ [21]

Raspberry Pi võimaldab edasiarendamisel rakendada tarkvaraplatvormi raamistikku ROS [22]. Raspberry Pi-d on läbiaastate järjest aretatud ja sellest on mitmeid erinevaid versioone. Antud robotil kasutatakse Raspberry Pi 3 Mudel B+, sest see on piisavalt võimekas liikumisprotsessi lahendamiseks jättes selle kõrvalt ka ruumi edasiarendamise joaks ning sellel on sisseehitatud Bluetooth ja wi-fi moodulid kaugjuhtimise kergeks integreerimiseks [21].

3.3 Toiteallikas

Kuna tegemist on liikurrobotiga ei saa kasutada statsionaarset juhtmeühendusega toiteplokki, vaid peab kasutama akul põhinevat toidet. Akude karakteristikud määravad nende koostiselemendid. On arendatud palju erinevate koostiselementidega akusid, kuid nendest populaarsemad on NiMH (nikkelmetallhüdriid), Li-ion (liitiumioon), LiPo (liitiumpolümeer), pliihappe ja NiCd (nikkel-kaadmium). Antud töö kontekstis on tähtsateks aku omadusteks energia tihedus, nominaalpinge, mahalaadimisvool, energiamahutavus ning kaal. Energiatiheduse ja kaalu karakteristikute tõttu võrreldakse siin ainult nikkelmetallhüdriid, liitiumioon ja liitiumpolümeer akusid. Nominaalpinged ühel aku elemendil: Li-ion ja LiPo – 3,7V ja NiMH – 1,2V [23]. Kõrgema pingega akud koosnevad mitmetest elementidest, mis on ühendatud üksteisega jadamisi. Akude elementide arvu tähistatakse xS-ga ehk 2 elemendiga aku on 2S jne. Arvestades, et Raspberry Pi töötab 5V peal [21] ning üks Servo on võimeline töötada 7,2V peal [19] on mõistlikuks valikuks 2S Lipo aku, mille nominaalpinge on 7,4V. Järgmiseks parameetriks on energia mahutavus. Seda tähistatakse akudel tüüpiliselt mAh (milliampertund). Näiteks 4000mAh-st akut saab kasutada tund aega kui kasutatakse konstantselt 4A voolu ehk see määrab ära kui kaua saab akut ühe laadimiskorraga kasutada. Mahutavuse kõrvalt peab jälgima ka aku mahalaadimisvoolu, mida tähistatakse C-teguriga. C-tegur näitab ära, mis on maksimaalne voolutugevus, mida aku välja suudab anda. Maksimaalse voolutugevuse saab teada korrutades aku mahutavuse C-teguriga. Näiteks kui on 1000mAh ja 2C aku, siis selle aku maksimaalne voolutugevus on 2A.

Kuna robot kasutab 18 servomootorit, on vaja suure C-teguriga akut, sest tüüpiliselt 2S akud 6000mAh mahutavusest üle ei lähe. Tavaliselt NiMH akude C-tegur piirdub 2C juures ning seega peab vaatama LiPo-de poole.

Joonis 3.3 Gens ace 4000mAh 2S 7,4V 60C LiPo aku [24]

Antud töös sai prototüübi jaoks valitud Gens ace 4000mAh 2S 7,4V 60C kõva kestaga LiPo aku (Joonis 3.3) [24], sest sellel on suur C-tegur ning nominaalpinge on 7,4V ja mahutavus 4000mAh, mis on prototüübi jaoks piisav. Sellel akul on lisaks koormuse ühendusele ka balansseerimisühendus, mille kaudu on võimalik jälgida iga aku elemendi pinget eraldi.

3.4 Tüürahelad ja vaheühendused

Kuuejalgse roboti iga jalg kasutab kolme servomootorit ehk roboti peale kokku on neid 18. Raspberry Pi 3-l on 26 tarkvaraliselt genereeritud PWM signaali võimaldavat GPIO ühendust, seega oleks võimalik kõiki servosid otse Raspberry Pi-ga juhtida, kuid kui arvestada edasiarendusega, siis võib neid GPIO ühendusi erinevatel eesmärkidel vaja minna. Seega on mõistlik servode juhtimiseks kasutada vahemoodulit. Tavamüügis selliseid mooduleid, mis suudaksid 18 servot korraga juhtida, palju ei leidu. Üheks selliseks mooduliks on Pimoroni servo 2040 mikrokontroller (Joonis 3.4) [25]. See mikrokontroller põhineb RP2040 kiibil, mis on disainitud Raspberry Pi Foundation poolt, kes on Raspberry Pi monoplaatarvutite autorid. Mikrokontrolleri kasutamiseks on olemas põhjalik dokumentatsioon ja teekide kogu. Lisaks on võimalik panna Pimoroni 2040 Raspberry Pi-ga suhtlema läbi USB porti.

Joonis 3.4 Pimoroni Servo-2040 mikrokontroller [25]

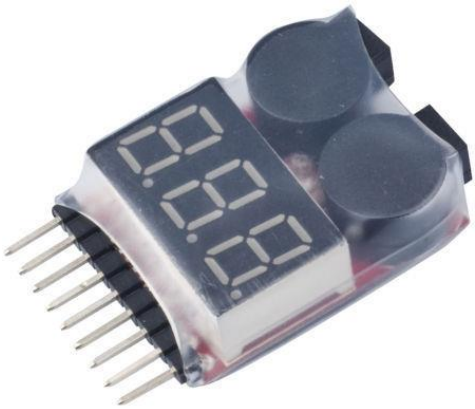
Teiseks vaheahelaks on roboti prototüübil pingemuundurid. Kuna valitud aku nominaal väljundpinge on 7,4V, aga Raspberry Pi töötab 5V peal ning servomootorid taluvad maksimaalselt 7,2V, on vaja aku pinget erinevatele tasemetele all tuua. Selleks on vaja kasutada pingemuundureid. Raspberry Pi-l on kaks erinevat toiteallika ühendamise võimalust. Esimene on läbi mikro-USB pordi ning teine on GPIO reas määratud ühenduste kaudu. Kuna ainult läbi mikro-USB pordi on rakendatud kaitseahel, mis piirab voolu, siis kasutatakse töös seda. Raspberry Pi toitepinge saamiseks ja voolutugevuse piiramiseks kasutatakse LM2596 põhinevat step-down konverteri moodulit (Joonis 3.5), mis võimaldab tuua 6-40V pinge alla 5V peale ja piirab maksimaalse voolutugevuse 3A peale [26]. Sellel moodulil on väljundiks 2 USB porti, mis teeb Raspberry Pi-ga ühendamise mugavaks ning jääb ka ruumi edasiarenduses lisa seadmete ühendamiseks.

Joonis 3.5 LM2596 step-down moodul [26]

Servode toite muundamisel ei saa samasugust moodulit kasutada, sest kõik 18 servot võivad kokku tarbida üle 3A voolu. Selleks on efektiivsem kasutada kaugjuhitavates hobimudelites leiduvat UBEC (universal battery eliminator circuit) moodulit. UBEC on spetsiifiliselt 2-6S akude kasutamiseks mõeldud pinge regulaator, mis stabiliseerib toitepinge ja toob selle aku nominaalpingelt alla määratud pinge peale ning tavaliselt lubavad UBEC-d võrdlemisi kõrgeid voolutugevusi. Kuna valitud servosid juhtival moodulil on servo toite maksimaalne lubatud voolutugevus 10A [25], siis antud töös rakendatakse Henge 8A UBEC moodulit (Joonis 3.6), mis väljastab 6V pinge ja piirab maksimaalse pideva voolu 8A-ni [27].

Joonis 3.6 Henge 8A UBEC pinge regulaator [27]

Viimaseks oluliseks osaks elektriahelas on aku madalpinge tuvastusahel. Kuna kasutatakse liitiumpolümeerakut, siis ilma seda jälgimata võib aku ennast kasutamise käigus liiga tühjaks laadida, peale mida enam seda sama akut kasutada ei saa. Sellise olukorra vältimiseks rakendatakse antud töös prototüübil madalpinge alarmi moodulit.



Joonis 3.7 Madalpinge alarmi moodul [28]

See moodul (Joonis 3.7) ühendatakse aku balansseerimisühenduse külge, läbi mille jälgib see aku iga elemendi pinget ja kuvad elementide pingeid seitsme segmendiga LED näidiku peal. Kui ühegi elemendi pinge langeb alla 3,3V teeb moodul häält, et kasutaja saaks aku toite ühenduse katkestada. Aku toite manuaalseks lahti lülitamiseks on aku ja pinge regulaatorite vahele ühendatud kahte asendit võimaldab on/off lüliti. Kui moodul on alarmi andnud ja aku on elektriahelast lahti ühendatud peab ka selle mooduli aku küljest lahti ühendama, muidu tarbib see moodul aku energiat edasi.

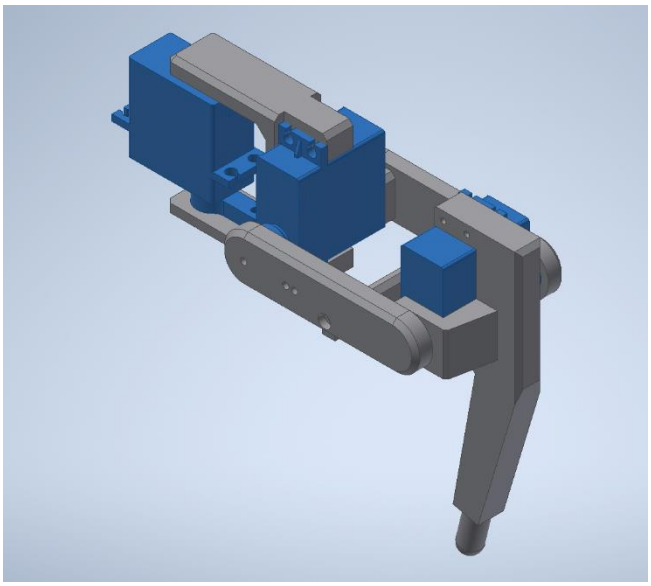
4. PROTOTÜÜBI KONSTRUKTSIOON

Selles peatükis kirjeldatakse kuuejalgse liikurroboti füüsilise prototüübi loomist. Esimesena projekteeritakse roboti mehhaanilisest struktuurist CAD tarkvara abil 3D mudel. Mudel koosneb roboti keha ning jalgade struktuuri detailidest, mida kasutatakse nende osade tootmisel 3D printimise tehnoloogiaga. Lisaks selgitatakse prototüübi kokkupaneku protsessi ning roboti elektriskeemi olemust.

4.1 CAD disain

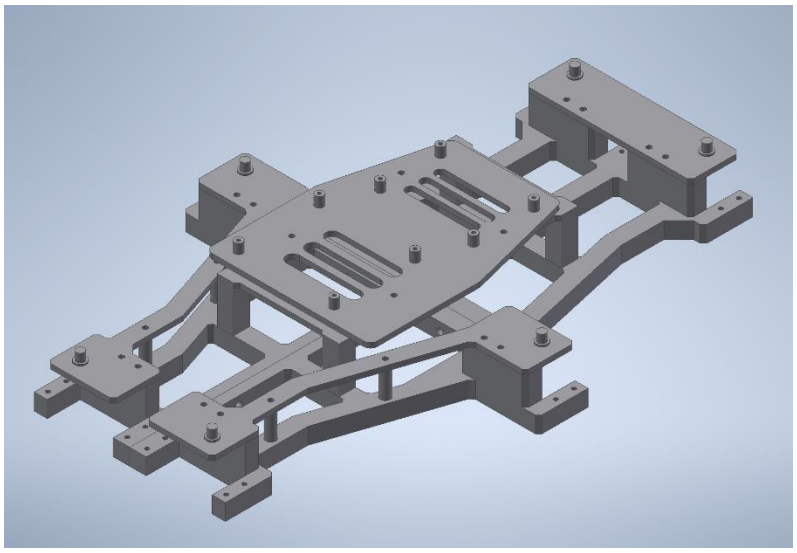
Roboti prototüübi mehaanilise struktuuri projekteerimiseks kasutati Autodesk Inventor tarkvara. Eesmärgiks oli projekteerida jalgade ja keha struktuuri detailid, et saada ülevaade prototüübi ilmest ja kasutada loodud 3D mudeleid struktuuri detailide tootmisel 3D printimisega. 3D mudelitel on projekteerimisel arvestatud poltide ja kruvide kinnituskohtadega, kuid polte ega kruve mudelisse ei lisatud. Sama kehtib ka elektroonika komponentide kohta, väljaarvatud servod, mis on vajalikud roboti struktuuri sidumiseks.

Nagu kinemaatika mudeli koostamisel sai defineeritud, on kõik kuus jalga identsed ja koonevad kolmest liigendist ning lülist. Seega peab projekteerima ainult ühe jala, mida saab kasutada kuus korda. Jala iga lüli on liigendi külge kinnitatud kahelt poolt, ühelt poolt otse ajami väljundvõlli külge ja teiselt poolt laagri peale (Joonis 4.1). Liigendites olevate laagrite eesmärgiks on ajami võlli mehaanilise koormuse jagamine. Jala struktuuri disainimisel jälgiti kinemaatika mudelis defineeritud jala lülide pikkuseid (Tabel 2.1) ning prooviti ka läbi tööruumi defineerivas alapeatükis [2.4] määratud pöördenurkade limiidid, et kontrollida ja vältida potentsiaalseid struktuurilisi kokkupõrkehetki.



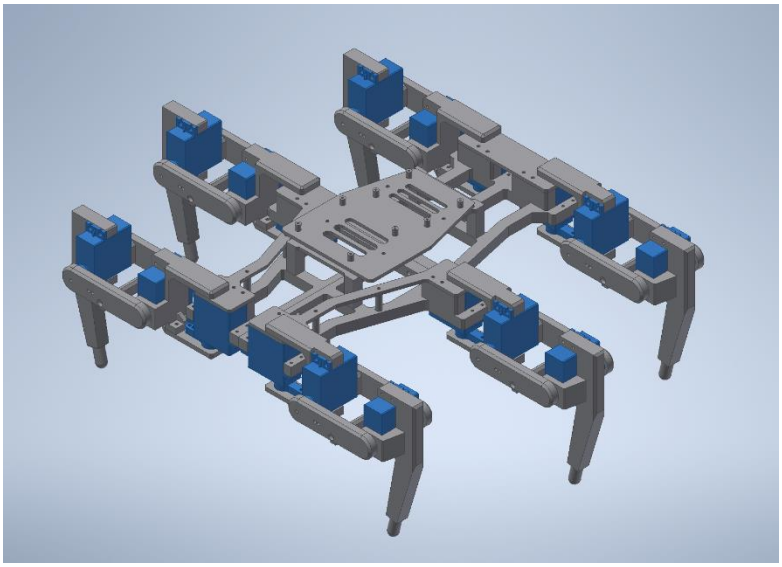
Joonis 4.1 Prototüübi ühe jala 3D mudel

Roboti keha koosneb alumisest raamist, millele ühenduvad iga jala puusa liigendi ajamid, keskmisest tõstetud platvormist, kuhu kinnituvad toiteahel ja juhtseade ning ajamite peal lisa osad, mille külge kinnituvad puusa liigendi laagrid (Joonis 4.2). Kõige Alumine raamistik koosneb kolmest osast, sest kogu raamistik on 3D printimise jaoks liiga suur, need kolm osa kinnituvad üksteise külge poltide ja mutritega. Kõik puusa ajamid on roboti otseliikumis suunaga füüsiliselt risti kinnitatud, kuid nende külge jalgade ühendamisel saab nullasendi orientatsiooni muuta. Kõrgendatud platvorm hoiab üleval pool roboti aju, milleks on Raspberry Pi ja ajameid juhtiv Pimoroni mikrokontroller ning allpool toiteahela pingeregulaatoreid.



Joonis 4.2 Prototüübi keha 3D mudel ilma elektroonikata

Roboti prototüübi mudel on ka tervikuna esitatud (Joonis 4.3), kus kere külge on ühendatud kõik kuus jalga. Tervik mudelis ei kujutata toiteahela komponente ega juhtseadet.



Joonis 4.3 Prototüübi tervik 3D mudel

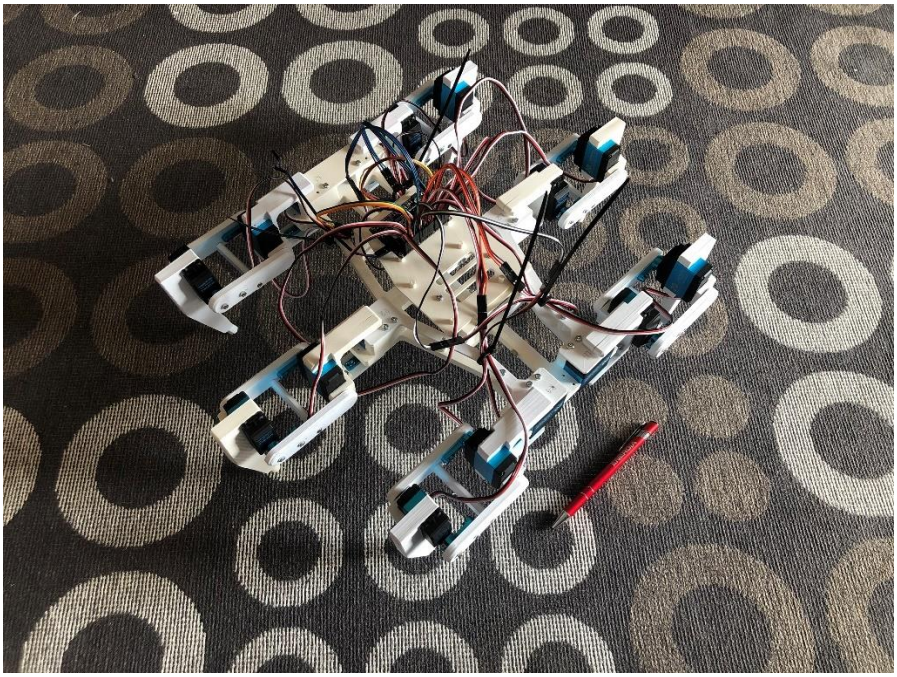
Mudeli kujutistest kõik halli värvi detailid on struktuuri osad, mis on vaja iseseisvalt toota. Nende tootmiseks kasutatakse 3D printimise tehnoloogiat.

4.2 3D printimine ja prototüübi kokkupanek

3D printimine on viimase aastakümnega väga kiirelt arenenud ja seetõttu ka eraisikutele kasutajasõbralikuks, kättesaadavaks ning ohutuks muutunud. 3D printimine võimaldab arendustöös kiiret prototüüpimist, mis võimaldab kiirelt katse-eksitus meetodiga projekteerimisse parandusi ja uuendusi rakendada. Antud töös kasutati 3D printimist 3D mudeliks projekteeritud struktuuri osade tootmisel. Osade printimiseks kasutati PLA materjali ja Creality Ender 3 Pro printerit (Joonis 4.4).

Joonis 4.4 Ender 3 Pro 3D printer [29]

Kuna selle printeri printimisala on 220*220mm [29], peab keha raamistiku mitme tükina printima. Printida nõuab 14 erinevat detaili, millest kuute jala detaili on vaja printida kuus korda ehk kokku on vaja välja printida 44 struktuuri osa.



Joonis 4.5 Kokkupandud prototüüp, millel puudub Raspberry Pi ja toiteahel. Pastakas roboti ees on suuruse referentsiks

Füüsiline prototüüp (Joonis 4.5) pandi kokku samm haaval. Esimesena konstrueeriti keha selle detailidest. Keha külge ühendati kõik kuus puusa servot. Jalgade puusa lüli ühendamisel juhiti puusa servod mikrokontrolleri abiga tarkvaraliselt nende nullasendisse, et saavutada Joonis 4.3 kujutatud asend. Sama protsessi korrati kõigi jalgade reie ja sääre lülide ühendamisel. Viimasena kinnitati Raspberry Pi monoplaatarvuti ja Servo 2040 mikrokontroller keha ülemise plaadi külge ning ühendati

sellele kõigi 18-ne servo kaablid. Esi- ja tagajalgade kaablid tõmmati kaablisidemetega kere detailide külge kinni, et need ei jääks liikumisel jalgade vahele kinni.

4.3 Elektriskeem

Prototüübi elektriskeemi kujutis, kus ei ole näidatud 18-ne servo ühendusi, on kantud töö lõppu lisadesse (Lisa 8). Ettenähtud elektriskeemis on Raspberry Pi ühendatud Servo 2040 mikrokontrolleriga läbi USB kaabli, mille mikrokontrolleri ots on USB-C port. Selle kaabli kaudu käib nii mikrokontrolleri ja monoplaatarvuti vaheline andmevahetus, kui ka mikrokontrolleri loogika ahela toide. Raspberry Pi toide tuleb, läbi USB kaabli, LM2596 step-down pingeregulaatori mooduli väljundist. Mooduli sisend on ühendatud roboti aku väljundisse. Aku väljund on jagatud LM2596 mooduli ja UBEC pingeregulaatori vahel. UBEC regulaatori väljund on ühendatud Servo 2040 mikrokontrolleri plaadil olevatesse servode toite klemmidesse. Aku ja pingeregulaatorid on eraldatud kahepooluselise lihtlülitiga. Aku balansseerimisühendusse on kinnitatud aku elementide pinge näidiku ja madalpinge alarmi moodul.

5. KATSETUSED

Prototüübi plaanipärase töötamise katsetamiseks koostatakse Raspberry Pi peal lihtsad juhtprogrammid, mis testivad servode korrektset tööd. Lisaks proovitakse implementeerida projekteerimisel defineeritud sammu trajektoori läbimist ja selle põhjal ka roboti otseliikumist kolmejalgsel kõnnakul.

Joonis 5.1 Katsetustel kasutatav reguleeritav toiteplokk

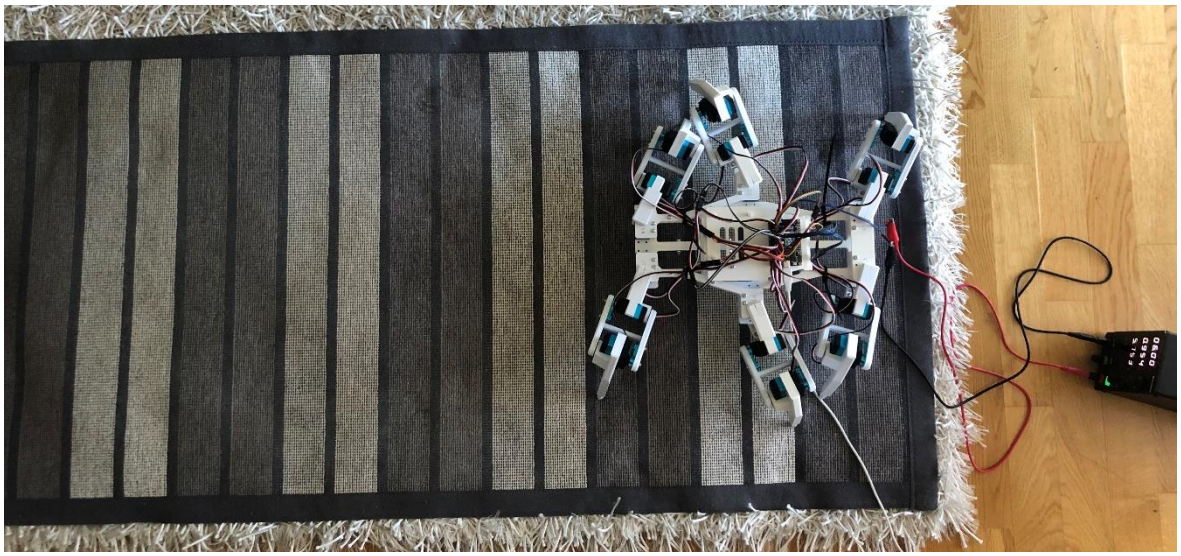
Katsetused on läbi viidud ilma akuta, selle asemel kasutatakse servode toiteallikana statsionaarset toiteplokki (Joonis 5.1) ja juhtseadme toiteallikana seinas olevat elektripistikut ehk välja valitud komponentidest on kasutusel ainult servod, Raspberry Pi monoplaatarvuti ja Servo 2040 mikrokontroller. Statsionaarne toiteplokk on seadistatud nii, et selle väljund oleks toiteahela UBEC pingeregulaatori (Joonis 3.6) väljundiga võrdne ehk pinge on 6V ja maksimaalne voolutugevus on 8A.

5.1 Prototüübi jala juhtimine

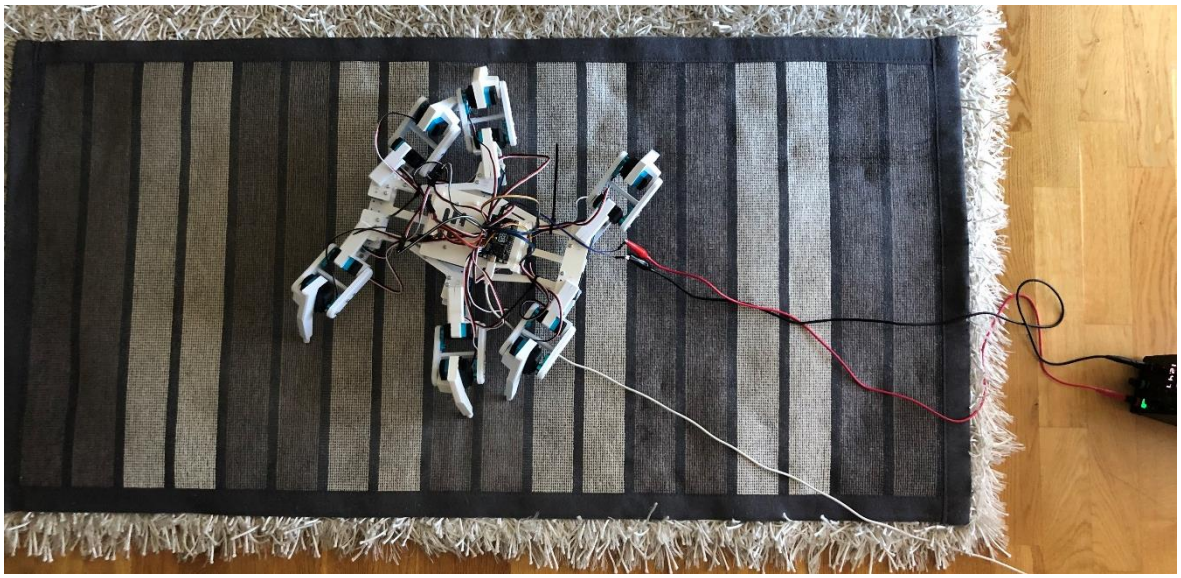
Esimese katse eesmärgiks oli välja selgitada servode pöörlemissuunad, et nendega arvestada sammu trajektoori läbimisel ja ka otseliikumisel, uurida tööruumile määratud limiite ja nendest tulenevaid jalgade vahelisi kokkupõrkeohtusid ning rakendada ühe jala juhtimiseks alapeatükis 2.4 defineeritud sammu trajektoor. Nende eesmärkida saavutamiseks koostati Python programmeerimiskeeles programmikood (Lisa 9), mida jooksutatakse mikrokontrolleril. Programmi kood kasutab ära Servo 2040 mikrokontrolleri loojate poolt koostatud teekide ja näidisprogrammide kogu [30], et lihtsustada servode juhtimist nii, et vaja läheb ainult servo pöördenurka soovitava positsioonina. Katse programmi alguses arvutatakse kinemaatika pöördülesande ja sammu trajektoori valemite järgi välja liigendite nurgad kindla diskreetse aja tagant sammu jooksul ja salvestatakse need väärtused massiivi. Neid väärtuseid kasutatakse hiljem sammu trajektoori läbimisel. Esimese juhtimisena viiakse ühe jala servod nende nullasendisse, peale mida liigutatakse neid soovitavasse sammu algasendisse, mille pealt tuvastatakse servode pöörlemissuunad. Viimasena kasutatakse algselt salvestatud pöördenurkade väärtuseid, et läbida sammu trajektoor. Esimesena läbib jalg tõste faasi ning teisena tõuke faasi, jõudes samasse algasendisse tagasi. Katsetamise käigus muudeti sammu läbimise kiirust ja ka sammupikkust, et eemaldada kokkupõrked teiste jalgadega.

5.2 Prototüübi otseliikumine

Teise katse eesmärgiks oli saavutada kolmejalgse kõnnakuga otseliikumine, millega testitakse, kas modelleerimisel väljaarvutatud parameetrid on korrektsed ja piisavad. Kolmejalgne kõnnak tähendab, et kõndimise vältel on kolm jalga korraga sammu tõuke faasis ja kolm jalga korraga tõste faasis. Lisaks jälgitakse statsionaarse toiteploki kuvarit, millelt näeb voolutugevust kõigi 18 servo töö ajal, et välja selgitada, kas UBEC pingeregulaatori maksimaalsest väljundist piisab roboti korrektse töö sooritamiseks. Katse sooritamiseks koostati Python programmeerimiskeeles programmikood, mida jooksutatakse mikrokontrolleril (Lisa 10). Katse programm kasutab ära mikrokontrolleri jaoks koostatud teeke kõigi 18 servo juhtimise lihtsustamiseks. Kolmejalgsel kõnnakul määrati ära, mis jalad sünkroonis liiguvad. Tõste faasiga alustavad parempoolsed esi- ja tagajalg ning vasakpoolne keskmine jalg, tõuke faasiga alustavad vasakpoolsed esi- ja tagajalg ja parempoolne keskmine jalg. Programmis rakendatakse eelmises katses defineeritud sammu trajektoori läbimisel liigendite pöördenurkade väärtuseid.



Joonis 5.2 Otse kõndimise katse algasukoht ja -asend



Joonis 5.3 Otse kõndimise katse lõppasukoht ja -asend

Katse ajal kõnnib robot otse edasi kuni kõik jalad on läbinud kaks sammu tsüklit (Joonis 5.2 ja Joonis 5.3), sest statsionaarse toiteploki ja Raspberry Pi ning mikrokontrolleri vahelise andmeside juhtmete pikkused hetkel rohkem ei võimalda. Katse ajal toiteploki kuvarilt loetud kõige suurem voolutugevus, mida ajamid nõudsid, oli ~2,56A.

6. JÄRELDUSED JA EDASIARENDUSE VÕIMALUSED

Antud töös sai modelleeritud ja projekteeritud kuuejalgse liikurroboti baasplatvorm ning sellest ka prototüüp ehitatud. Modelleerimise ning projekteerimise tulemused võimaldavad ainult robotsüsteemi baastöövõimekust ja ei ole autonoomse kuuejalgse liikurroboti terviklahendus. Baastöövõimekus on antud juhul roboti ühe-suunaline ette defineeritud sammu järgi kõndimine.

Katsete tulemustest sai järeldada, et algselt defineeritud puusa liigendi pöördenurga limiidid on liiga suured ja põhjustaksid jalgade omavahelisi kokkupõrkeid, sest kinemaatika mudel ei arvesta mehaanilise struktuuri mõõtmetega. Prototüübi ohutuks puusa liigendi limiidiks peab olema nullasendist 25° mõlemas suunas. Sellest tulenevalt lühendati katsetes ka sammupikkust varasemast 120 mm pealt 90 mm peale. Lisaks pidi sammu alguspunkti z-koordinaati muutma, et robot seisaks maapinnast kõrgemal ja ei hõõruks puusa liigendi alumist osa vastu liikumispinda. Teise katse tulemustest saab järeldada, et valitud ajamid on adekvaatselt võimsad, et tagada roboti baastöövõimekus. Otsekõndimisel robot pööras natuke paremale (Joonis 5.3), sest kõigi jalgade nullasendid ei olnud täpselt kalibreeritud. See tulenes füüsilisest servo võlli ja sarve konstruktsioonist, kuid seda on võimalik tarkvaraliselt parandada. Lisaks kinnitab toiteplokilt loetud voolutugevus, et toiteahela UBEC pingeregulaator on piisav ajamite nominaalse töö tagamiseks.

Baasplatvorm on disainitud arvestades selle edasiarendusvõimalusi. Nendeks edasiarendusvõimaluseteks võivad olla:

- Kinemaatika mudeli täiendamine nii, et see arvestab ja võimaldab juhtida roboti kere kaldenurki globaalses koordinaatteljestikus.
- Sammu trajektoori arvutamise täiendamine nii, et see võimaldaks roboti pööramist või igasuunalist liikumist.
- Juhtsüsteemi arendus, integreerides selle ROS tarkvara raamistikku.
- Arendada kaugjuhitavuse võimalus, kasutades näiteks pulti läbi Bluetooth kommunikatsiooni.
- Lisada ja integreerida tagasiside andureid ja ahelaid. Näiteks jala tippude kontaktandurid, millega oleks võimalik tuvastada maapinna kontakti ebaühtlasel maastikul, Lidar- või kaamerasüsteemi lisamine keskkonna tuvastamiseks ja sellele rakendada masinnägemist, mis võimaldaks robotil autonoomselt keskkonnas liikuda.

KOKKUVÕTE

Käesoleva töö eesmärgiks oli luua kuuejalgse liikurroboti baasplatvormi projekteerimise ja prototüübi konstrueerimise protsesse selgitav ja vabalt kasutatav ressurss isearendajatele. Robotsüsteemi projekteerimise ja prototüübi konstrueerimise protsess jaguneb etappideks. Esmalt määratakse roboti algsed parameetrid ja luuakse robotsüsteemi kinemaatiline mudel, mille abil defineeritakse jalgade liikumistrajektoor ning arvutatakse jalgade liigenditele mõjuvad jõumomendid. Matemaatilised mudelid ja nende visuaalsed kujutused on loodud MathWorks Matlab tarkvara abil, millel on Robotics System Toolbox lisa tööriistana. Mehaaniline struktuur on projekteeritud Autodesk Inventor CAD tarkvara keskkonnas. Seejärel valitakse roboti elektroonilised komponendid, milleks on ajamid, juhtseade, toiteallikas, tüürahelad ja vaheühendustena pingemuundurid ning kaitseahelad. Järgmisena on vaja projekteerida roboti prototüübi mehaaniline struktuur, toota struktuuri osad 3D printimise abil ja konstrueerida füüsiline prototüüp.

Prototüübi testimiseks on vaja läbi viia katsetused, et kinnitada projekteerimistulemuste korrektsust ja tuvastada võimalikud puudujäägid. Antud töös katsetati esmalt prototüübi ühe jala juhtimist ning teise katsena testiti terve roboti otseliikumist defineeritud sammu trajektoori järgi. Esimese katse käigus osutus vajalikuks muuta sammu läbimise kiirust ja ka sammu pikkust, et eemaldada kokkupõrked teiste jalgadega. Otseliikumise katsel pööras robot pisut paremale, sest kõigi jalgade nullasendid ei olnud täpselt kalibreeritud. Katsete tulemustest järeldub, et algselt defineeritud puusa liigendi pöördenurga limiidid olid liiga suured ja põhjustaksid jalgade omavahelisi kokkupõrkeid, sest kinemaatika mudel ei arvesta mehaanilise struktuuri mõõtmetega. Katsed tõestasid, et valitud ajamid on adekvaatselt võimsad tagamaks roboti baastöövõimekus.

Modelleerimise ning projekteerimise tulemused võimaldavad ainult robotsüsteemi baastöövõimekust ja ei ole autonoomse kuuejalgse liikurroboti terviklahendus. Baastöövõimekus on antud juhul roboti ühesuunaline ette defineeritud sammu järgi kõndimine. Töövõimekuse täiendamiseks pakutakse välja erinevaid edasiarendamise suunad.

KASUTATUD KIRJANDUSE LOETELU

[1] „Spot® - The Agile Mobile Robot," Boston Dynamics, [Võrgumaterjal]. Available: https://bostondynamics.com/products/spot/. [Kasutatud 22 aprill 2024].

[2] „Atlas® and beyond: the world's most dynamic robots," Boston Dynamics, [Võrgumaterjal]. Available: https://bostondynamics.com/atlas/. [Kasutatud 22 aprill 2024].

[3] „PhantomX Hexapod MK-IV," Tribotix, [Võrgumaterjal]. Available: https://tribotix.com/product/phantomx-hexapod-mk-4/. [Kasutatud 22 aprill 2024].

[4] M. García-López, E. Gorrostieta-Hurtado, E. Vargas-Soto, J. Ramos-Arreguín, A. Sotomayor-Olmedo ja J. M. Morales, „Kinematic analysis for trajectory generation in one leg of a hexapod robot," *Procedia Technology,* kd. 3, pp. 342-350, 2012.

[5] J. A. T. Machado ja M. F. Silva, „An Overview of Legged Robots," in *MME 2006 - International Symposium on Mathematical Methods in Engineering*, Cankaya, Ankara, Turkey, 2006.

[6] J. Billingsley, A. Visala ja M. Dunn, „Robotics in Agriculture and Forestry," in *Springer Handbook of Robotics*, Springer, 2008, pp. 1065-1077.

[7] M. Raibert, K. Blankespoor, G. Nelson ja R. Playter, „BigDog, the Rough-Terrain Quadruped Robot," *IFAC Proceedings Volumes,* kd. 41, nr 2, pp. 10822-10825, 2008.

[8] „Mars 2020: Perseverance Rover," NASA, [Võrgumaterjal]. Available: https://science.nasa.gov/mission/mars-2020-perseverance/. [Kasutatud 24 aprill 2024].

[9] D. Sun, J. Zhang, Q. Fang, P. Xiang, Y. Xue, Y. Wang, R. Xiong ja H. Lu, „Analysis and control for a bioinspired multi-legged soft robot," *Biomimetic Intelligence and Robotics,* kd. 2, nr 1, 2022.

[10] Y. Ozkan-Aydin ja D. I. Goldman, „Self-reconfigurable multilegged robot swarms collectively accomplish challenging terradynamic tasks," *Science Robotics,* kd. 6, nr 56, 2021.

[11] M. Y. Harper, J. V. Nicholson, E. G. Collins, J. Pusey ja J. E. Clark, „Energy Efficient Navigation for Running Legged Robots," in *International Conference on Robotics and Automation*, Montreal, QC, Canada, 2019.

[12] M. R. Elhami ja I. Dashti, „A New Approach to the Solution of Robot Kinematics Based on Relative Transformation Matrices," *IAES International Journal of Robotics and Automation,* kd. 5, nr 3, pp. 213-222, 2016.

[13] F. Cursi, W. Bai, E. M. Yeatman ja P. Kormushev, „GlobDesOpt: A Global Optimization Framework for Optimal Robot Manipulator Design," *IEEE Access,* kd. 10, pp. 5012-5023, 2022.

[14] X. Ding, Z. Wang, A. Rovetta ja J. Zhu, „Locomotion Analysis of Hexapod Robot," in *Climbing and Walking Robots*, InTech, 2010.

[15] F. Tedeschi ja G. Carbone, „Design Issues for Hexapod Walking Robots,“ *Robotics,* kd. 3, nr 2, pp. 181-206, 2014.

[16] D. Collins, „How to calculate velocity from triangular and trapezoidal move profiles,“ Linear Motion Tips, [Võrgumaterjal]. Available: https://www.linearmotiontips.com/how-to-calculate-velocity/. [Kasutatud 06 aprill 2024].

[17] L. Zhang, F. Yang, D. Li, J. Guo ja C. Liu, „Analysis of Low Energy Static Posture for Hexapod Bionic Robot,“ in *Proceedings of the 2017 2nd International Conference on Electrical, Automation and Mechanical Engineering*, 2017.

[18] Terasic Inc, „MG996R Datasheet,“ DigiKey, [Võrgumaterjal]. Available: https://www.digikey.ee/en/htmldatasheets/production/5014637/0/0/1/mg996r. [Kasutatud 19 aprill 2024].

[19] „TD-8120MG Digital Servo,“ SunFounder, [Võrgumaterjal]. Available: http://wiki.sunfounder.cc/images/9/9a/TD-8120MG_Digital_Servo.pdf. [Kasutatud 19 aprill 2024].

[20] „LD-27MG Full Metal Gear Digital Servo with 270 Control Angle,“ Hiwonder, [Võrgumaterjal]. Available: https://www.hiwonder.com/products/ld-27mg. [Kasutatud 19 aprill 2024].

[21] „Raspberry Pi 3 Model B+,“ Raspberry Pi, [Võrgumaterjal]. Available: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/. [Kasutatud 19 aprill 2024].

[22] „ROS 2 on Raspberry Pi,“ [Võrgumaterjal]. Available: https://docs.ros.org/en/foxy/How-To-Guides/Installing-on-Raspberry-Pi.html. [Kasutatud 19 aprill 2024].

[23] „Types of Battery,“ Chip One Stop, [Võrgumaterjal]. Available: https://www.chip1stop.com/sp/knowledge/084_types-of-battery_en. [Kasutatud 21 aprill 2024].

[24] „4000mAh 2S1P 7.4V 60C HardCase 8# car Lipo Battery pack with T-plug,“ Gens ace, [Võrgumaterjal]. Available: https://www.gensace.de/gens-ace-bashing-series-4000mah-2s1p-7-4v-60c-hardcase-8-car-lipo-battery-pack-with-t-plug.html. [Kasutatud 21 aprill 2024].

[25] „Servo 2040 - 18 Channel Servo Controller,“ Pimoroni, [Võrgumaterjal]. Available: https://shop.pimoroni.com/products/servo-2040?variant=39800591679571. [Kasutatud 19 aprill 2024].

[26] „LM2596 DC to DC Double USB 3A Step Down Module Buck Converter Power Supply Module Voltage Regulator DC 6V-40V to DC 5V for Car Solar Charger,“ IC Station, [Võrgumaterjal]. Available: https://www.icstation.com/lm2596-double-step-down-module-buck-converter-power-supply-module-voltage-regulator-solar-charger-p-8663.html. [Kasutatud 21 aprill 2024].

[27] „User Manual of Henge 8A UBEC,“ [Võrgumaterjal]. Available: http://www.henge-rc.com/UploadFiles/201751721654.pdf. [Kasutatud 23 aprill 2024].

[28] „LiPo Alarm/Tester 1-8S,“ RC-Est, [Võrgumaterjal]. Available: https://rc-est.eu/pood/elektroonika/elektroonika-akud-ja-laadimine/akud/lipo-alarm-tester-1-8s/. [Kasutatud 05 mai 2024].

[29] „Ender-3 Pro,“ Creality, [Võrgumaterjal]. Available: https://www.creality.com/products/ender-3-pro-3d-printer. [Kasutatud 24 aprill 2024].

[30] „GitHub - Pimoroni micropython Servo 2040 library,“ [Võrgumaterjal]. Available: https://github.com/pimoroni/pimoroni-pico/tree/main/micropython/modules/servo. [Kasutatud 09 mai 2024].

LISAD

Lisa 1

Ühe jala kinemaatika otseülesande Matlab programmi kood

```
function [x3, y3, z3] = HEX_LEG_FK(x0, y0, z0, rz, ang1, ang2, ang3)
%Jala lülide pikkused, mm
L1 = 50;
L2 = 70;
L3 = 100;

%DH parameetrid, tulbad: 1-a, 2-alpha, 3-d, 4-theta
dhp = [L1  pi/2  0  0;
       L2  0     0  -pi/2;
       L3  0     0  0];

%Vajalikud rotatsiooni maatriksid
Rz0 = [cos(rz*(pi/180)) -sin(rz*(pi/180)) 0 0; sin(rz*(pi/180))
cos(rz*(pi/180)) 0 0; 0 0 1 0; 0 0 0 1];
Rx1 = [1 0 0 0; 0 cos(dhp(1,2)) -sin(dhp(1,2)) 0; 0 sin(dhp(1,2))
cos(dhp(1,2)) 0; 0 0 0 1];
Rx2 = [1 0 0 0; 0 cos(dhp(2,2)) -sin(dhp(2,2)) 0; 0 sin(dhp(2,2))
cos(dhp(2,2)) 0; 0 0 0 1];
Rz2 = [cos(dhp(2,4)) -sin(dhp(2,4)) 0 0; sin(dhp(2,4)) cos(dhp(2,4)) 0 0;
0 0 1 0; 0 0 0 1];

%Baas X,Y,Z koordinaadid, mis määratakse funktsiooni parameetritest
X0 = x0;
Y0 = y0;
Z0 = z0;

%Lülide sisendnurgad, mis määratakse funktsiooni parameetritest
J1 = ang1;
J2 = ang2;
J3 = ang3;

%Lüli 1 ehk puus teisendusmaatriksisse DH parameetrite arvestamine
trvec1 = [dhp(1,1) 0 dhp(1,3)];
T1 = trvec2tform(trvec1);
TDH1 = T1*Rx1;

%Lüli 2 ehk reis teisendusmaatriksisse DH parameetrite arvestamine
trvec2 = [dhp(2,1) 0 dhp(2,3)];
T2 = trvec2tform(trvec2);
TDH2 = T2*Rz2*Rx2;

%Lüli 3 ehk säär teisendusmaatriksisse DH parameetrite arvestamine
trvec3 = [dhp(3,1) 0 dhp(3,3)];
TDH3 = trvec2tform(trvec3);

% Pöördenurka (ümber z-telje) kirjeldav maatriks lüli koordinaatide
arvutamiseks
% 1. lüli pöördenurk
g1 = [0 0 1 J1*(pi/180)];
R1 = axang2tform(g1);
% 2. lüli pöördenurk
g2 = [0 0 1 J2*(pi/180)];
```

```
R2 = axang2tform(g2);
% 3. lüli pöördenurk
g3 = [0 0 1 J3*(pi/180)];
R3 = axang2tform(g3);

%Jala nullpunkti teisendusmaatriksi arvutamine
P0 = [1 0 0 X0; 0 1 0 Y0; 0 0 1 Z0; 0 0 0 1];
P0 = P0*Rz0;

%Lülide teisendusmaatriksite arvutamine
P1 = P0*R1*TDH1;
P2 = P1*R2*TDH2;
P3 = P2*R3*TDH3;

%Lõpplüli koordinaatide määramine
x3 = P3(1,4);
y3 = P3(2,4);
z3 = P3(3,4);

%Jala lülide plottimine
%if funktsioon, et plottida ainult kui baasteljestikku on muudetud
if x0 ~= 0 || y0 ~= 0
    plot3([P0(1,4) P1(1,4)], [P0(2,4) P1(2,4)], [P0(3,4)
P1(3,4)],'Color','r','LineWidth',3);
    plot3([P1(1,4) P2(1,4)], [P1(2,4) P2(2,4)], [P1(3,4)
P2(3,4)],'Color','g','LineWidth',3);
    plot3([P2(1,4) P3(1,4)], [P2(2,4) P3(2,4)], [P2(3,4)
P3(3,4)],'Color','b','LineWidth',3);
end
end
```

Lisa 2

Terve roboti kinemaatika otseülesande Matlab programmi kood

```
%Roboti keha orientatsioon ümber oma z-telje (kraadides)
Rz0 = 0;

%Jalgade baasteljestike orientatsioonid ümber z-telje (kraadides)
Rz0l = [-90 -90 -90 90 90 90];

%Keha dimensioonid
l = 80;
p = 300;
lo = 50;
%DH parameetrid keha keskkohale ja igale jala kinnituskohale
%Tulbad: 1-a, 2-alpha, 3-d, 4-theta
dhp = [0     pi/2  0        0;
       p/2   0     l/2      0;
       0     0     l/2+lo   0;
       -p/2  0     l/2      0;
       -p/2  0     -l/2     0;
       0     0     -l/2-lo  0;
       p/2   0     -l/2     0];

%Keha baasteljestiku defineerimine
T0 = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
g0 = [0 0 1 Rz0*pi/180];
R0 = axang2tform(g0);
P0 = R0*T0;
g1x = [1 0 0 dhp(1,2)];
R1x = axang2tform(g1x);
P1 = P0*R1x;

%Esimese jala kinnituskoha teisendusmaatriks
trvec11 = [dhp(2,1) 0 dhp(2,3)];
T11 = trvec2tform(trvec11);
P11 = P1*T11;

%Teise jala kinnituskoha teisendusmaatriks
trvec12 = [dhp(3,1) 0 dhp(3,3)];
T12 = trvec2tform(trvec12);
P12 = P1*T12;

%Kolmanda jala kinnituskoha teisendusmaatriks
trvec13 = [dhp(4,1) 0 dhp(4,3)];
T13 = trvec2tform(trvec13);
P13 = P1*T13;

%Neljanda jala kinnituskoha teisendusmaatriks
trvec14 = [dhp(5,1) 0 dhp(5,3)];
T14 = trvec2tform(trvec14);
P14 = P1*T14;

%Viienda jala kinnituskoha teisendusmaatriks
trvec15 = [dhp(6,1) 0 dhp(6,3)];
T15 = trvec2tform(trvec15);
P15 = P1*T15;
```

```
%Kuuenda jala kinnituskoha teisendusmaatriks
trvec16 = [dhp(7,1) 0 dhp(7,3)];
T16 = trvec2tform(trvec16);
P16 = P1*T16;

%Roboti keha plottimine
figure;
hold on; grid on; axis equal;
view(-135,35);
plot3(0,0,0,'o')        %Keha keskpunkt

b1 = plot3([P11(1,4) P12(1,4)], [P11(2,4) P12(2,4)], [P11(3,4)
P12(3,4)],'Color','#000000','LineWidth',3);
b2 = plot3([P12(1,4) P13(1,4)], [P12(2,4) P13(2,4)], [P12(3,4)
P13(3,4)],'Color','#000000','LineWidth',3);
b3 = plot3([P13(1,4) P14(1,4)], [P13(2,4) P14(2,4)], [P13(3,4)
P14(3,4)],'Color','#000000','LineWidth',3);
b4 = plot3([P14(1,4) P15(1,4)], [P14(2,4) P15(2,4)], [P14(3,4)
P15(3,4)],'Color','#000000','LineWidth',3);
b5 = plot3([P15(1,4) P16(1,4)], [P15(2,4) P16(2,4)], [P15(3,4)
P16(3,4)],'Color','#000000','LineWidth',3);
b6 = plot3([P16(1,4) P11(1,4)], [P16(2,4) P11(2,4)], [P16(3,4)
P11(3,4)],'Color','#000000','LineWidth',3);

xlabel('X telg'), ylabel('Y telg'), zlabel('Z telg');
xlim([-300 300]);
ylim([-300 300]);
zlim([-200 200]);

%Kõigi kuue jala keha külge ühendamine
for i = 1:6
    if i == 1
        HEX_LEG_FK(P11(1,4), P11(2,4), P11(3,4), Rz0l(1,1), 0, 0, 0);
    elseif i == 2
        HEX_LEG_FK(P12(1,4), P12(2,4), P12(3,4), Rz0l(1,2), 0, 0, 0);
    elseif i == 3
        HEX_LEG_FK(P13(1,4), P13(2,4), P13(3,4), Rz0l(1,3), 0, 0, 0);
    elseif i == 4
        HEX_LEG_FK(P14(1,4), P14(2,4), P14(3,4), Rz0l(1,4), 0, 0, 0);
    elseif i == 5
        HEX_LEG_FK(P15(1,4), P15(2,4), P15(3,4), Rz0l(1,5), 0, 0, 0);
    elseif i == 6
        HEX_LEG_FK(P16(1,4), P16(2,4), P16(3,4), Rz0l(1,6), 0, 0, 0);
    end
end
```

Lisa 3

Ühe jala kinemaatika pöördülesande Matlab programmi kood

```
function [ang1, ang2, ang3] = HEX_LEG_IK(x3, y3, z3)
    %Jala DH parameetrid, tulbad: 1-a, 2-alpha, 3-d, 4-theta
    dhp = [50   pi/2  0  0;
           70   0     0  -pi/2;
           100  0     0  0];

    %Nullkoha defineerimine
    P0 = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];

    q0 = atan((y3-P0(2,4))/((x3-P0(1,4))));

    %Reie liigendi koordinaatide arvutamine
    x1 = cos(q0)*dhp(1,1);
    y1 = sin(q0)*dhp(1,1);
    z1 = P0(3,4);

    %Sääre liigendi kauguse jala tipust arvutamine
    P13vec = [x3-x1, y3-y1, z3-z1];
    Lcc = norm(P13vec);
    %Lcc = sqrt(z3^2+(sqrt(x3^2+y3^2)-dhp(1,1))^2);     %Lcc valem

    q1 = acos((z1-z3)/Lcc);
    q2 = acos((dhp(2,1)^2+Lcc^2-dhp(3,1)^2)/(2*dhp(2,1)*Lcc));
    q3 = acos((dhp(2,1)^2+dhp(3,1)^2-Lcc^2)/(2*dhp(2,1)*dhp(3,1)));

    %Lülide sisendnurgad
    ang1 = q0/pi*180;
    ang2 =(q1 + q2)/pi*180 -90;
    ang3 = q3/pi*180 -90;
end
```

Lisa 4

Jala tööruumi visulaiseerimise Matlab programmi kood

```
%Jala liigendite pöördenurkade limiidid
ang_lim = [-45, 45; -45, 60; -60, 60];
%Jala tipu koordinaadite defineerimine
endpos = zeros(1,3);
%Mitme kraadi tagant jala tipu asukoht arvutatakse
step = 15;

figure;
hold on; grid on; axis equal;
view(45,90);
xlabel('X telg'), ylabel('Y telg'), zlabel('Z telg');
xlim([-50 300]);
ylim([-200 200]);
zlim([-200 200]);

%teeta1 muutub min väärtusest max väärtuseni
for j = ang_lim(1,1):step:ang_lim(1,2)
    %teeta3 on min väärtus ja teeta2 muutub min väärtusest max väärtuseni
    for i = ang_lim(2,1):5:ang_lim(2,2)
        [endpos(1,1),endpos(1,2),endpos(1,3)] = HEX_LEG_FK(0, 0, 0, 0, j,
i, ang_lim(3,1));
        plot3(endpos(1,1), endpos(1,2), endpos(1,3),'o','Color','b')
    end
    %teeta2 on max väärtus ja teeta3 muutub min väärtusest max väärtuseni
    for t = ang_lim(3,1):step:ang_lim(3,2)
        [endpos(1,1),endpos(1,2),endpos(1,3)] = HEX_LEG_FK(0, 0, 0, 0, j,
ang_lim(2,2), t);
        plot3(endpos(1,1), endpos(1,2), endpos(1,3),'o','Color','b')
    end
    %teeta3 on max väärtus ja teeta2 muutub min väärtusest max väärtuseni
    for k = ang_lim(2,2):-step:ang_lim(2,1)
        [endpos(1,1),endpos(1,2),endpos(1,3)] = HEX_LEG_FK(0, 0, 0, 0, j,
k, ang_lim(3,2));
        plot3(endpos(1,1), endpos(1,2), endpos(1,3),'o','Color','b')
    end
    %teeta2 on min väärtus ja teeta3 muutub min väärtusest max väärtuseni
    for l = ang_lim(3,2):-step:ang_lim(3,1)
        [endpos(1,1),endpos(1,2),endpos(1,3)] = HEX_LEG_FK(0, 0, 0, 0, j,
ang_lim(2,1), l);
        plot3(endpos(1,1), endpos(1,2), endpos(1,3),'o','Color','b')
    end
end
```

Lisa 5

Sammu trajektoori arvutamise (liikumisprofiilita) Matlab programmi kood

```
%Algkoordinaadid on x=150, y=0, z=-80
xs0 = 150;
ys0 = 0;
zs0 = -80;
%Sammmupikkus, mm
sp = 120;
%Sammu tõste kõrgus, mm
h = 60;

%Liigendite pöördenurgad sammu ajal
leg_ang_toste = zeros(sp+1,3);
leg_ang_touge = zeros(sp+1,3);

f1 = figure('Name','Liigendite nurgad tõste faasis','NumberTitle','off');
f2 = figure('Name','Liigendite nurgad tõuke faasis','NumberTitle','off');
f3 = figure('Name','Trajektoor','NumberTitle','off');

tr = animatedline('Color','black');

grid on; axis equal;
view(45,25);
tr.LineWidth = 3;
xlabel('X telg'), ylabel('Y telg'), zlabel('Z telg');
xlim([-50 250]);
ylim([-150 150]);
zlim([-150 0]);

%Sammu tõste faas
for i = 0:1:sp
    x = xs0;
    y = ys0 + (-sp/2+i);
    z = h*sin(((-sp/2+i)*pi)/(sp) + (pi/2))+zs0;

    %Liigendite pöördenurkade salvestamine
    [leg_ang_toste(i+1,1), leg_ang_toste(i+1,2), leg_ang_toste(i+1,3)] =
HEX_LEG_IK(x, y, z);
    addpoints(tr, x, y, z);
    drawnow;
    pause(0.01);
end

%Sammu tõuke faas
for j = sp:-1:0
    x = xs0;
    y = ys0 + (-sp/2+j);

    %Liigendite pöördenurkade salvestamine
    [leg_ang_touge(j+1,1), leg_ang_touge(j+1,2), leg_ang_touge(j+1,3)] =
HEX_LEG_IK(x, y, zs0);
    addpoints(tr, x, y, zs0);
    drawnow;
    pause(0.01);
end
```

```
%Liigendite nurkade visualiseerimine tõuke faasi ajal
figure(f2);
plot(leg_ang_touge(:,1),'LineWidth',2);
hold on;
grid on;
plot(leg_ang_touge(:,2),'LineWidth',2);
plot(leg_ang_touge(:,3),'LineWidth',2);
hold off;
legend({'Puusa pöördenurk','Reie pööednurk','Sääre
pöördenurk'},'Location','southeast');
xlabel('Aeg'), ylabel('Pöördenurk');
xlim([0 sp]);

%Liigendite nurkade visualiseerimine tõste faasi ajal
figure(f1);
plot(leg_ang_toste(:,1),'LineWidth',2);
hold on;
grid on;
plot(leg_ang_toste(:,2),'LineWidth',2);
plot(leg_ang_toste(:,3),'LineWidth',2);
hold off;
legend({'Puusa pöördenurk','Reie pööednurk','Sääre
pöördenurk'},'Location','southeast');
xlabel('Aeg'), ylabel('Pöördenurk');
xlim([0 sp]);
```

Lisa 6

Tõuke faasis jala tipu lineaarse liikumisprofiili Matlab programmi kood

```
%Sammu y-koordinaadi algpunkt
y0 = -60;
%Sammupikkus, mm
sp = 120;
%Tõuke faasi kestus, ms
tt = 1000;

T1 = linspace(0,tt/2);
T2 = linspace(tt/2,tt);
syms t;
%Kiirendamise osa valemite tuletamine
p1 = y0 + (sp/2)*t^2/(tt/2)^2;
v1 = diff(p1,t);
a1 = diff(v1,t);

%Pidurdamise osa valemite tuletamine
p2 = y0 - ((sp/2)*(t-tt)^2)/(tt/2)^2+sp;
v2 = diff(p2,t);
a2 = diff(v2,t);

%Tõuke faasi esimese osa ehk kiirendamise osa välja arvutamine
P1 = double(subs(p1, t, T1));
V1 = double(subs(v1, t, T1))*1000;
A1 = double(subs(a1, t, T1))*1000^2;

%Tõuke faasi teise osa ehk pidurdamise osa välja arvutamine
P2 = double(subs(p2, t, T2));
V2 = double(subs(v2, t, T2))*1000;
A2 = double(subs(a2, t, T2))*1000^2;

%Kiirendamise ja pidurdamise kokku toomine
TT = cat(2,T1,T2);
P = cat(2,P1,P2);
V = cat(2,V1,V2);
A = cat(2,A1,A2);

figure('Name','Lõpplüli tõuke faasis');
%Asukoha visualiseerimine
subplot(3,1,1);
plot(TT, P,'LineWidth',2,'color','b');
hold on;
ylim([-100 100]);
xlabel('Aeg [ms]'), ylabel('Y koordinaat');
title('Lõpplüli asukoht y-teljel ajas');
grid on;
%Kiiruse visualiseerimine
subplot(3,1,2);
plot(TT, V,'LineWidth',2,'color','g');
hold on;
ylim([0 400]);
xlabel('Aeg [ms]'), ylabel('Kiirus [mm/s]');
title('Lõpplüli kiirus ajas');
grid on;
%Kiirenduse visualiseerimine
```

```
subplot(3,1,3);
plot(TT, A,'LineWidth',2,'color','r');
hold on;
ylim([-600 600]);
xlabel('Aeg [ms]'), ylabel('Kiirendus [mm/s^2]');
title('Lõpplüli kiirendus ajas');
grid on;
```

Lisa 7

Tõuke faasis liigendite pöördliikumisprofiili ja liigenditele mõjuvate jõumomentide arvutamise Matlab programmi kood

```
%Jala DH parameetrid
dhp = [50   pi/2  0  0;
       70   0     0  -pi/2;
       100  0     0  0];
%Sammu alguspunkti koordinaadid
x0 = 150;
y0 = -60;
z0 = -60;
%Sammupikkus ja kõrgus, mm
sp = 120;
sk = 60;
%Sammu kestvus, ms (millisekundites)
tt = 1000;

f1 = figure('Name','Tõuke faas');
f3 = figure('Name','Pöördmomendid');

T = linspace(0,sp);
TT1 = linspace(0,tt/2);
TT2 = linspace(tt/2,tt);
syms t;


%Liigendite pöördenurkade, pöördkiiruste ja pöördkiirenduste valemite
tuletamine sammu tõuke faasi esimeses pooles
%Puusa liigend
tt1 = atan((y0 + (sp/2)*t^2/(tt/2)^2)/x0);

ot1 = diff(tt1,t);      %Asukoha valemist ajatuletise võtmine

at1 = diff(ot1,t);      %Kiiruse valemist ajatuletise võtmine

%Reie liigend
tt2 = acos((0-z0)/(sqrt(z0^2+(sqrt(x0^2+(y0 + (sp/2)*t^2/(tt/2)^2)^2)-
dhp(1,1))^2))) + acos((dhp(2,1)^2+(sqrt(z0^2+(sqrt(x0^2+(y0 +
(sp/2)*t^2/(tt/2)^2)^2)-dhp(1,1))^2))^2-
dhp(3,1)^2)/(2*dhp(2,1)*(sqrt(z0^2+(sqrt(x0^2+(y0 +
(sp/2)*t^2/(tt/2)^2)^2)-dhp(1,1))^2)))) - pi/2;

ot2 = diff(tt2,t);      %Asukoha valemist ajatuletise võtmine

at2 = diff(ot2,t);      %Kiiruse valemist ajatuletise võtmine

%Sääre liigend
tt3 = acos((dhp(2,1)^2+dhp(3,1)^2-(sqrt(z0^2+(sqrt(x0^2+(y0 +
(sp/2)*t^2/(tt/2)^2)^2)-dhp(1,1))^2))^2)/(2*dhp(2,1)*dhp(3,1))) - pi/2;

ot3 = diff(tt3,t);      %Asukoha valemist ajatuletise võtmine

at3 = diff(ot3,t);      %Kiiruse valemist ajatuletise võtmine
```

```
%Liigendite pöördenurkade, pöördkiiruste ja pöördkiirenduste valemite
tuletamine sammu tõuke faasi teises pooles
%Puusa liigend
tl1 = atan((y0 - ((sp/2)*(t-tt)^2)/(tt/2)^2+sp)/x0);

ol1 = diff(tl1,t);      %Asukoha valemist ajatuletise võtmine

al1 = diff(ol1,t);      %Kiiruse valemist ajatuletise võtmine

%Reie liigend
tl2 = acos((0-z0)/(sqrt(z0^2+(sqrt(x0^2+(y0 - ((sp/2)*(t-
tt)^2)/(tt/2)^2+sp)^2)-dhp(1,1))^2))) +
acos((dhp(2,1)^2+(sqrt(z0^2+(sqrt(x0^2+(y0 - ((sp/2)*(t-
tt)^2)/(tt/2)^2+sp)^2)-dhp(1,1))^2))^2-
dhp(3,1)^2)/(2*dhp(2,1)*(sqrt(z0^2+(sqrt(x0^2+(y0 - ((sp/2)*(t-
tt)^2)/(tt/2)^2+sp)^2)-dhp(1,1))^2)))) - pi/2;

ol2 = diff(tl2,t);      %Asukoha valemist ajatuletise võtmine

al2 = diff(ol2,t);      %Kiiruse valemist ajatuletise võtmine

%Sääre liigend
tl3 = acos((dhp(2,1)^2+dhp(3,1)^2-(sqrt(z0^2+(sqrt(x0^2+(y0 - ((sp/2)*(t-
tt)^2)/(tt/2)^2+sp)^2)-dhp(1,1))^2))^2)/(2*dhp(2,1)*dhp(3,1))) - pi/2;

ol3 = diff(tl3,t);      %Asukoha valemist ajatuletise võtmine

al3 = diff(ol3,t);      %Kiiruse valemist ajatuletise võtmine

%Väärtuste välja arvutamine tuletatud valemite järgi
Tt1 = double(subs(tt1, t, TT1))/pi*180;
Ot1 = double(subs(ot1, t, TT1))*tt;
At1 = double(subs(at1, t, TT1))*tt^2;
Tt2 = double(subs(tt2, t, TT1))/pi*180;
Ot2 = double(subs(ot2, t, TT1))*tt;
At2 = double(subs(at2, t, TT1))*tt^2;
Tt3 = double(subs(tt3, t, TT1))/pi*180;
Ot3 = double(subs(ot3, t, TT1))*tt;
At3 = double(subs(at3, t, TT1))*tt^2;
Tl1 = double(subs(tl1, t, TT2))/pi*180;
Ol1 = double(subs(ol1, t, TT2))*tt;
Al1 = double(subs(al1, t, TT2))*tt^2;
Tl2 = double(subs(tl2, t, TT2))/pi*180;
Ol2 = double(subs(ol2, t, TT2))*tt;
Al2 = double(subs(al2, t, TT2))*tt^2;
Tl3 = double(subs(tl3, t, TT2))/pi*180;
Ol3 = double(subs(ol3, t, TT2))*tt;
Al3 = double(subs(al3, t, TT2))*tt^2;

%Kahe tõuke faasi osa kokku üheks viimised
TTT = cat(2,TT1,TT2);
%Asukohad
T1 = cat(2,Tt1,Tl1);
T2 = cat(2,Tt2,Tl2);
T3 = cat(2,Tt3,Tl3);
%Kiirused
O1 = cat(2,Ot1,Ol1);
O2 = cat(2,Ot2,Ol2);
O3 = cat(2,Ot3,Ol3);
%Kiirendused
A1 = cat(2,At1,Al1);
```

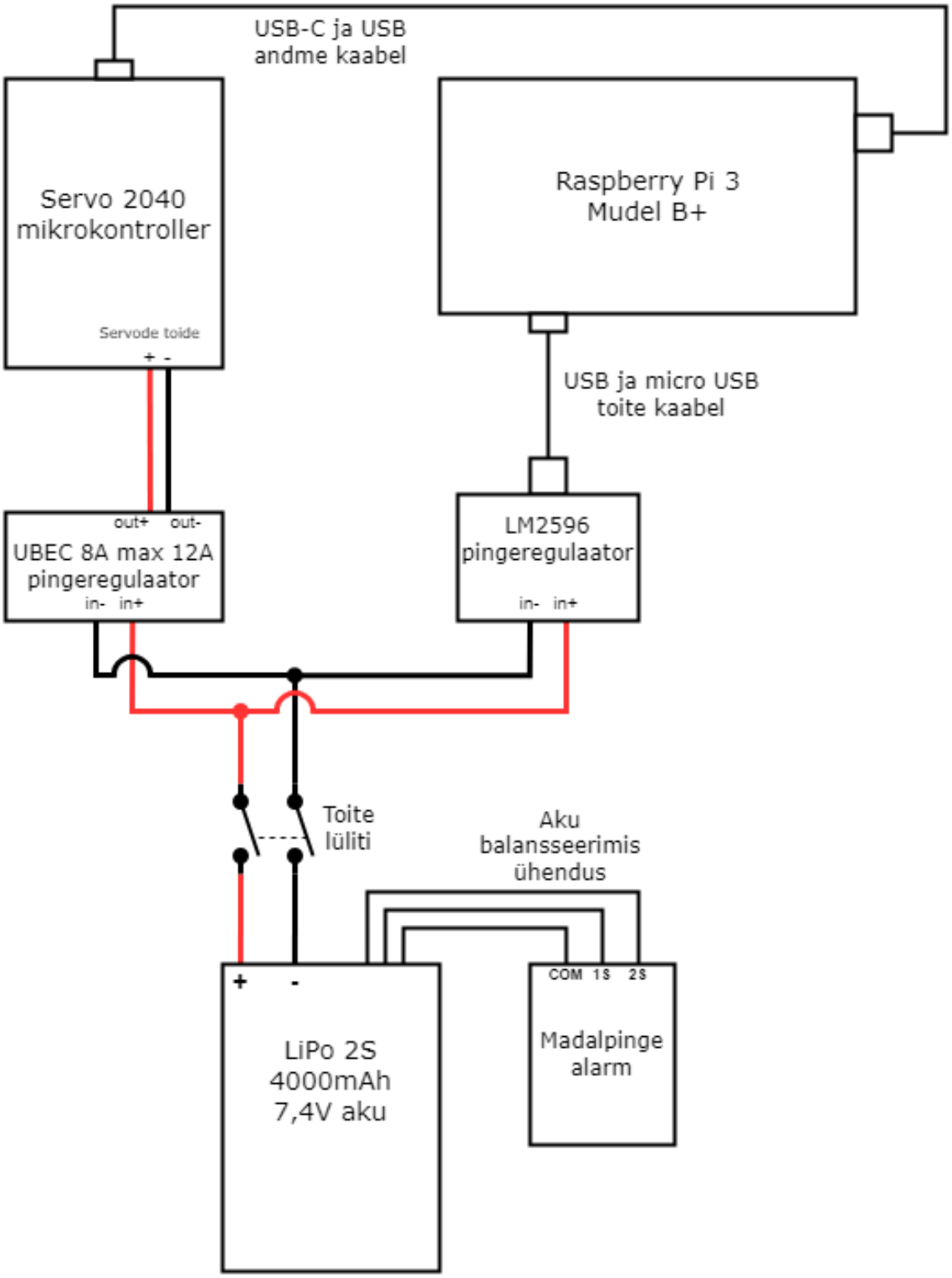
```
A2 = cat(2,At2,Al2);
A3 = cat(2,At3,Al3);

%Puusa pöördenurga visualiseerimine tõuke faasis
figure(f1);
subplot(3,3,1);
plot(TTT, T1 ,'LineWidth',2,'color','b');
hold on;
xlabel('Aeg [ms]'), ylabel('Pöördenurk [°]');
title('Puusa pöördenurk');
grid on;
%Puusa pöördkiiruse visualiseerimine tõuke faasis
subplot(3,3,2);
plot(TTT, O1 ,'LineWidth',2,'color','g');
hold on;
xlabel('Aeg [ms]'), ylabel('Pöördkiirus [rad/s]');
title('Puusa pöördkiirus');
grid on;
%Puusa pöördkiirenduse visualiseerimine tõuke faasis
subplot(3,3,3);
plot(TTT, A1 ,'LineWidth',2,'color','r');
hold on;
xlabel('Aeg [ms]'), ylabel('Pöördkiirendus [rad/s^2]');
title('Puusa pöördkiirendus');
grid on;

%Reie pöördenurga visualiseerimine tõuke faasis
subplot(3,3,4);
plot(TTT, T2 ,'LineWidth',2,'color','b');
hold on;
xlabel('Aeg [ms]'), ylabel('Pöördenurk [°]');
title('Reie pöördenurk');
grid on;
%Reie pöördkiiruse visualiseerimine tõuke faasis
subplot(3,3,5);
plot(TTT, O2 ,'LineWidth',2,'color','g');
hold on;
xlabel('Aeg [ms]'), ylabel('Pöördkiirus [rad/s]');
title('Reie pöördkiirus');
grid on;
%Reie pöördkiirenduse visualiseerimine tõuke faasis
subplot(3,3,6);
plot(TTT, A2 ,'LineWidth',2,'color','r');
hold on;
xlabel('Aeg [ms]'), ylabel('Pöördkiirendus [rad/s^2]');
title('Reie pöördkiirendus');
grid on;
%Sääre pöördenurga visualiseerimine tõuke faasis
subplot(3,3,7);
plot(TTT, T3 ,'LineWidth',2,'color','b');
hold on;
xlabel('Aeg [ms]'), ylabel('Pöördenurk [°]');
title('Sääre pöördenurk');
grid on;
%Sääre pöördkiiruse visualiseerimine tõuke faasis
subplot(3,3,8);
plot(TTT, O3 ,'LineWidth',2,'color','g');
hold on;
xlabel('Aeg [ms]'), ylabel('Pöördkiirus [rad/s]');
title('Sääre pöördkiirus');
grid on;
```

```
%Sääre pöördkiirenduse visualiseerimine tõuke faasis
subplot(3,3,9);
plot(TTT, A3 ,'LineWidth',2,'color','r');
hold on;
xlabel('Aeg [ms]'), ylabel('Pöördkiirendus [rad/s^2]');
title('Sääre pöördkiirendus');
grid on;


%JÕUMOMENTIDE ARVUTAMINE SAMMU TÕUKE FAASI AJAL
Mkeha = 2;      %Keha eeldatav mass, kg
Mjalg = 0.3;    %Ühe jala eeldatav mass, kg
g = 9.81;       %Gravitatsiooni kiirendus, m/^2
lk1 = 90;       %Puusa liigendi kaugus keha massikeskmest, mm
%Ühe jala vastujõud pinnakontaktist
N = 1/3*(Mkeha+6*Mjalg)*g;      %Kui kolm jalga vastu maad, siis
vastujõud ühel jalal on 1/3 terve roboti raskusjõust
%Lisategur, et arvutatud pöördmomendid ei jääks lihtsustuste pärast liiga
väikseks
ot = 1.5;

%Dünaamiline jõumoment(tulenevalt pöördkiirendustest)
%Puusa liigend
Tau1da = 1/3*(Mkeha+3*Mjalg)*(lk1/1000)^2*At1;

Tau1db = 1/3*(Mkeha+3*Mjalg)*(lk1/1000)^2*Al1;

Tau1d = cat(2,Tau1da,Tau1db);
%Reie liigend
Tau2da = 1/3*(Mkeha + 3*Mjalg +
Mjalg/3)*(sqrt((lk1/1000)^2+(dhp(1,1)/1000)^2 -
2*(lk1/1000)*(dhp(1,1)/1000)*cos(pi-(Tt1*pi/180)))).^2 .* At2;

Tau2db = 1/3*(Mkeha + 3*Mjalg +
Mjalg/3)*(sqrt((lk1/1000)^2+(dhp(1,1)/1000)^2 -
2*(lk1/1000)*(dhp(1,1)/1000)*cos(pi-(Tt1*pi/180)))).^2 .* Al2;

Tau2d = cat(2,Tau2da,Tau2db);
%Sääre liigend
Tau3da = 1/3*(Mkeha + 3*Mjalg +
2*Mjalg/3)*(sqrt((sqrt((lk1/1000)^2+(dhp(1,1)/1000+dhp(2,1)/1000.*cos(Tt2
*pi/180)).^2 -
2*(lk1/1000)*(dhp(1,1)/1000+dhp(2,1)/1000.*cos(Tt2*pi/180)).*cos(pi-
(Tt1*pi/180)))).^2+(dhp(2,1)/1000.*sin(Tt2*pi/180)).^2)).^2 .* At3;

Tau3db = 1/3*(Mkeha + 3*Mjalg +
2*Mjalg/3)*(sqrt((sqrt((lk1/1000)^2+(dhp(1,1)/1000+dhp(2,1)/1000.*cos(Tl2
*pi/180)).^2 -
2*(lk1/1000)*(dhp(1,1)/1000+dhp(2,1)/1000.*cos(Tl2*pi/180)).*cos(pi-
(Tl1*pi/180)))).^2+(dhp(2,1)/1000.*sin(Tl2*pi/180)).^2)).^2 .* Al3;

Tau3d = cat(2,Tau3da,Tau3db);

%Staatiline pöördmoment (tulenevalt raskusjõust)
%Reie liigend
Tau2sa =
N*((dhp(2,1)/1000)*cos(Tt2*pi/180)+(dhp(3,1)/1000)*cos((Tt2*pi/180)+(pi/2
-(Tt3*pi/180))))-(dhp(2,1)/1000)/2*Mjalg/3*g*cos(Tt2*pi/180)-
Mjalg/3*g*((dhp(2,1)/1000)*cos(Tt2*pi/180)+(dhp(3,1)/1000)/2*cos((Tt2*pi/
180)+(pi/2-(Tt3*pi/180))));
```

```
Tau2sb =
N*((dhp(2,1)/1000)*cos(Tl2*pi/180)+(dhp(3,1)/1000)*cos((Tl2*pi/180)+(pi/2
-(Tl3*pi/180))))-(dhp(2,1)/1000)/2*Mjalg/3*g*cos(Tl2*pi/180)-
Mjalg/3*g*((dhp(2,1)/1000)*cos(Tl2*pi/180)+(dhp(3,1)/1000)/2*cos((Tl2*pi/
180)+(pi/2-(Tl3*pi/180))));

Tau2s = cat(2,Tau2sa,Tau2sb);
%Sääre liigend
Tau3sa = N*(dhp(3,1)/1000)*cos((Tt2*pi/180)+(pi/2-(Tt3*pi/180)))-
(dhp(3,1)/1000)/2*Mjalg/3*g*cos((Tt2*pi/180)+(pi/2-(Tt3*pi/180)));

Tau3sb = N*(dhp(3,1)/1000)*cos((Tl2*pi/180)+(pi/2-(Tl3*pi/180)))-
(dhp(3,1)/1000)/2*Mjalg/3*g*cos((Tl2*pi/180)+(pi/2-(Tl3*pi/180)));

Tau3s = cat(2,Tau3sa,Tau3sb);

%Liigenditele mõjuvate jõumomentide arvutamine(koos lisateguriga)
Tau1 = Tau1d*ot;
Tau2 = (Tau2s + Tau2d)*ot;
Tau3 = (Tau3s + Tau3d)*ot;
%Jõumomentide plottimine
figure(f3);
plot(TTT,Tau1,'LineWidth',2);
hold on;
plot(TTT,Tau2,'LineWidth',2);
plot(TTT,Tau3,'LineWidth',2);
legend('Puusa liigendi pöördmoment','Reie liigendi pöördmoment','Sääre
liigendi pöördmoment','Location','northoutside');
xlabel('Aeg [ms]'), ylabel('Pöördmoment [Nm]');
grid on;
```

Lisa 8

Prototüübi lihtsustatud elektriskeem ilma servodeta



Lisa 9

Ühe jala juhtimise katsetamise programmi kood Python keeles

```
import time
import math
from servo import Servo, servo2040
#Sammu parameetrid
sp = 90                 #Sammupikkus, mm
sk = 40                 #Sammukõrgus, mm
tt = 3000               #Sammu faasi kestus, ms
yp = 40                 #Servo positsiooni uuenduse delay, ms
#Sammu algkoordinaadid
x0 = 150
y0g = sp/2
y0s = -sp/2
z0 = -80
#Pöördenurkade listide defineerimine (tõuge = g, tõste = s), (p =
puus, r = reis, s = säär)
p_ang_s = []
r_ang_s = []
s_ang_s = []
p_ang_g = []
r_ang_g = []
s_ang_g = []
#Sammu tegemisel liigendite pöördenurkade ühekordne välja arvutamine
#Tõste faasis liigendite nurkade arvutamine
for i in range(0, tt+yp, yp):
    x = x0
    y = y0s + (sp*(i/tt))
    z = sk * math.sin((math.pi/tt)*i) + z0
    #Järjest listi lõppudesse nurkade lisamine kinemaatika
pöördülesande valemite järgi
    p_ang_s.append(math.atan(y/x) / math.pi*180)
    r_ang_s.append(((math.acos((0-
z)/(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2)))) +
(math.acos((70**2+(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2))**2-
100**2)/(2*70*(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2)))))) /
math.pi*180 -90)
    s_ang_s.append((math.acos((70**2+100**2-
(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2))**2)/(2*70*100))) /
math.pi*180 -90)
#Tõuke faasis kiirenduse osas liigendite nurkade arvutamine
for i in range(0,1520, yp):
    x = x0
    y = y0g + (-sp/2)*(i**2)/((tt/2)**2)
    z = z0
    #Järjest listi lõppudesse nurkade lisamine kinemaatika
pöördülesande valemite järgi
    p_ang_g.append(math.atan(y/x) / math.pi*180)
    r_ang_g.append(((math.acos((0-
z)/(math.sqrt(z**2+(math.sqrt(x**2+(y**2))-50)**2)))) +
(math.acos((70**2+(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2))**2-
100**2)/(2*70*(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2)))))) /
math.pi*180 -90)
    s_ang_g.append((math.acos((70**2+100**2-
(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2))**2)/(2*70*100))) /
math.pi*180 -90)
#Tõuke faasis pidurdamise osas liigendite nurkade arvutamine
```

```
for i in range(1520, tt+yp, yp):
    x = x0
    y = y0g - ((-sp/2)*(i-tt)**2)/(tt/2)**2 - sp
    z = z0
    #Järjest listi lõppudesse nurkade lisamine kinemaatika
pöördülesande valemite järgi
    p_ang_g.append(math.atan(y/x) / math.pi*180)
    r_ang_g.append(((math.acos((0-
z)/(math.sqrt(z**2+(math.sqrt(x**2+(y**2))-50)**2)))) +
(math.acos((70**2+(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2))**2-
100**2)/(2*70*(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2)))))) /
math.pi*180 -90)
    s_ang_g.append((math.acos((70**2+100**2-
(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2))**2)/(2*70*100))) /
math.pi*180 -90)
time.sleep(3)
# Servode defineerimine
l2p = Servo(servo2040.SERVO_3)
l2r = Servo(servo2040.SERVO_2)
l2s = Servo(servo2040.SERVO_1)
#Servode käivitamine
l2p.enable()
l2r.enable()
l2s.enable()
time.sleep(5)
#Testitakse, mis suunas servod pöörlevad/pöörlema peavad
l2p.value(p_ang_s[0])
time.sleep(2)
l2r.value(r_ang_s[0])
time.sleep(2)
l2s.value(s_ang_s[0])
time.sleep(2)
#Sammu trajektoori läbimine (tõste ja tõuge)
for j in range(len(p_ang_s)):
    l2p.value(p_ang_s[j])
    l2r.value(r_ang_s[j])
    l2s.value(s_ang_s[j])
    time.sleep(0.04)              #0,04s = yp/1000
time.sleep(1)
for j in range(len(p_ang_g)):
    l2p.value(p_ang_g[j])
    l2r.value(r_ang_g[j])
    l2s.value(s_ang_g[j])
    time.sleep(0.04)              #0,04s = yp/1000
#Lülitatakse kasutatavad servod välja
l2p.disable()
l2r.disable()
l2s.disable()
```

Lisa 10

Roboti otseliikumise katsetamise programmi kood Python keeles

```
import gc
import time
import math
from servo import Servo, servo2040
from servo import ServoCluster, servo2040
#Sammu parameetrid
sp = 90                 #Sammupikkus, mm
sk = 40                 #Sammukõrgus, mm
tt = 3000               #Sammu faasi kestus, ms
yp = 40                 #Servo positsiooni uuenduse delay, ms
x0 = 150
y0g = sp/2
y0s = -sp/2
z0 = -80
#Pöördenurkade listide defineerimine (tõuge = g, tõste = s), (p =
puus, r = reis, s = säär)
p_ang_s = []
r_ang_s = []
s_ang_s = []
p_ang_g = []
r_ang_g = []
s_ang_g = []
#Sammu tegemisel liigendite pöördenurkade ühekordne välja arvutamine
#Tõste faasis liigendite nurkade arvutamine
for i in range(0, tt+yp, yp):
    x = x0
    y = y0s + (sp*(i/tt))
    z = sk * math.sin((math.pi/tt)*i) + z0
    #Järjest listi lõppudesse nurkade lisamine kinemaatika
pöördülesande valemite järgi
    p_ang_s.append(math.atan(y/x) / math.pi*180)
    r_ang_s.append(((math.acos((0-
z)/(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2)))) +
(math.acos((70**2+(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2))**2-
100**2)/(2*70*(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2)))))) /
math.pi*180 -90)
    s_ang_s.append((math.acos((70**2+100**2-
(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2))**2)/(2*70*100))) /
math.pi*180 -90)
#Tõuke faasis kiirenduse osas liigendite nurkade arvutamine
for i in range(0,1520, yp):
    x = x0
    y = y0g + (-sp/2)*(i**2)/((tt/2)**2)
    z = z0
    #Järjest listi lõppudesse nurkade lisamine kinemaatika
pöördülesande valemite järgi
    p_ang_g.append(math.atan(y/x) / math.pi*180)
    r_ang_g.append(((math.acos((0-
z)/(math.sqrt(z**2+(math.sqrt(x**2+(y**2))-50)**2)))) +
(math.acos((70**2+(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2))**2-
100**2)/(2*70*(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2)))))) /
math.pi*180 -90)
    s_ang_g.append((math.acos((70**2+100**2-
(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2))**2)/(2*70*100))) /
math.pi*180 -90)
```

```
#Tõuke faasis pidurdamise osas liigendite nurkade arvutamine
for i in range(1520, tt+yp, yp):
    x = x0
    y = y0g - ((-sp/2)*(i-tt)**2)/(tt/2)**2 - sp
    z = z0
    #Järjest listi lõppudesse nurkade lisamine kinemaatika
pöördülesande valemite järgi
    p_ang_g.append(math.atan(y/x) / math.pi*180)
    r_ang_g.append(((math.acos((0-
z)/(math.sqrt(z**2+(math.sqrt(x**2+(y**2))-50)**2)))) +
(math.acos((70**2+(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2))**2-
100**2)/(2*70*(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2)))))) /
math.pi*180 -90)
    s_ang_g.append((math.acos((70**2+100**2-
(math.sqrt(z**2+(math.sqrt(x**2+y**2)-50)**2))**2)/(2*70*100))) /
math.pi*180 -90)
#Servo cluster-i defineerimine
START_PIN = servo2040.SERVO_1
END_PIN = servo2040.SERVO_18
servod = ServoCluster(0, 0, pins=list(range(START_PIN, END_PIN + 1)))
NOS = 2                      #Num of Steps
servod.enable_all()
time.sleep(10)
#Servod viiakse sammu algpositsiooni
servod.value(0, s_ang_s[0])
servod.value(1, r_ang_s[0])
servod.value(2, p_ang_s[0])
servod.value(3, s_ang_g[0])
servod.value(4, r_ang_g[0])
servod.value(5, p_ang_g[0])
servod.value(6, s_ang_g[0])
servod.value(7, r_ang_g[0])
servod.value(8, p_ang_g[0])
servod.value(9, -p_ang_s[0])
servod.value(10, r_ang_s[0])
servod.value(11, s_ang_s[0])
servod.value(12, -p_ang_s[0])
servod.value(13, r_ang_s[0])
servod.value(14, s_ang_s[0])
servod.value(15, -p_ang_g[0])
servod.value(16, r_ang_g[0])
servod.value(17, s_ang_g[0])
time.sleep(5)
for j in range(NOS):   #Mitu sammu järjest tehakse
    #Terve sammu esimene pool (kolm jalga teevad tõste, kolm tõuke)
    for i in range(len(p_ang_g)):
        #Tõstet sooritavad liigendid
        servod.value(0, s_ang_s[i])
        servod.value(1, r_ang_s[i])
        servod.value(2, p_ang_s[i])
        servod.value(9, -p_ang_s[i])
        servod.value(10, r_ang_s[i])
        servod.value(11, s_ang_s[i])
        servod.value(12, -p_ang_s[i])
        servod.value(13, r_ang_s[i])
        servod.value(14, s_ang_s[i])
        #Tõuget sooritavad liigendid
        servod.value(3, s_ang_g[i])
```

```
        servod.value(4, r_ang_g[i])
        servod.value(5, p_ang_g[i])
        servod.value(6, s_ang_g[i])
        servod.value(7, r_ang_g[i])
        servod.value(8, p_ang_g[i])
        servod.value(15, -p_ang_g[i])
        servod.value(16, r_ang_g[i])
        servod.value(17, s_ang_g[i])
        time.sleep(0.04)

    time.sleep(1)
    #Terve sammu teine pool (nüüd need kolm jalga, mis tegid tõste
teevad tõuke ja vastupidi)
    for i in range(len(p_ang_g)):
        #Tõstet sooritavad liigendid
        servod.value(3, s_ang_s[i])
        servod.value(4, r_ang_s[i])
        servod.value(5, p_ang_s[i])
        servod.value(6, s_ang_s[i])
        servod.value(7, r_ang_s[i])
        servod.value(8, p_ang_s[i])
        servod.value(15, -p_ang_s[i])
        servod.value(16, r_ang_s[i])
        servod.value(17, s_ang_s[i])
        #Tõuget sooritavad liigendid
        servod.value(0, s_ang_g[i])
        servod.value(1, r_ang_g[i])
        servod.value(2, p_ang_g[i])
        servod.value(9, -p_ang_g[i])
        servod.value(10, r_ang_g[i])
        servod.value(11, s_ang_g[i])
        servod.value(12, -p_ang_g[i])
        servod.value(13, r_ang_g[i])
        servod.value(14, s_ang_g[i])
        time.sleep(0.04)
    time.sleep(1)
#Kõik servod lülitatakse välja
servod.disable_all()
```