TALLINNA TEHNIKAÜLIKOOL
MEHAANIKATEADUSKOND

Mehhatroonikainstituut

Mehhatroonikasüsteemide õppetool

MHK70LT

Kärt Tergem

DOUBLE ROBOTICS PLATVORMI UURIMINE TESTIMISEKS

Autor taotleb
tehnikateaduste magistri
akadeemilist kraadi

Tallinn
2014

AUTORIDEKLARATSIOON

Deklareerin, et käesolev lõputöö on minu iseseisva töö tulemus.

Esitatud materjalide põhjal ei ole varem akadeemilist kraadi taotletud.

Töös kasutatud kõik teiste autorite materjalid on varustatud vastavate viidetega.

Töö valmis ..... juhendamisel

“.....”.....2014. a.

Töö autor

..... allkiri

Töö vastab magistritööle esitatavatele nõuetele.

“.....”.....2014. a.

Juhendaja

..... allkiri

Lubatud kaitsmisele.

..... eriala/õppekava kaitsmiskomisjoni esimees

“.....”.....2014. a.

..... allkiri

TTÜ mehhatroonikinstituut

Mehhatroonikasüsteemide õppetool

MAGISTRITÖÖ ÜLESANNE

2014. aasta kevadsemester

Üliõpilane: Kärt Tergem 121970 MAHM
Õppekava MAHM02/09
Spetsialiseerumine ……
Juhendaja: Maido Hiiemaa, teadur…….. (amet, nimi)
Konsultandid: …….. (nimi, amet, telefon)
……..…….

MAGISTRITÖÖ TEEMA:
(eesti keeles) Double Robotics platvormi uurimine testimiseks
(inglise keeles) Research for testing the platform of Double Robotics

Lõputöös lahendatavad ülesanded ja nende täitmise ajakava:

Nr	**Ülesande kirjeldus**	**Täitmise tähtaeg**
1.	Double Robotics platvormi olemasoleva juhtimistarkvara uurimine	3.04.14
2.	Arendusvahendite leidmine, mis võimaldavad saata robotile fikseeritud parameetritega juhtkäske (riistvara ja tarkvara)	15.04.14
3.	Roboti liikumiste analüüs erinevatel pindadel	28.04.14
4.	Roboti liikumiste analüüs erinevatel pindadel ja analüüsiks vajaliku mudeli koostamine	14.05.14
5.		

Lahendatavad insenertehnilised ja majanduslikud probleemid:
.....
Täiendavad märkused ja nõuded:…….**.....**
Töö keel: eesti keel
Kaitsmistaotlus esitada dekanaati hiljemalt 12.05.14 **Töö esitamise tähtaeg** 22.05.14
Üliõpilane Kärt Tergem /allkiri/ …….. kuupäev 27.03.14

Juhendaja Maido Hiiemaa /allkiri/ ……....…. kuupäev 27.03.14

Konfidentsiaalsusnõuded ja muud ettevõttepoolsed tingimused formuleeritakse pöördel

SISUKORD

SELEDE LOETELU

TABELITE LOETELU

Eessõna

Käesolev lõputöö on valminud Tallinna Tehnikaülikooli mehhatroonikainstituudis juhendaja teadur Maido Hiiemaa ning mehhatroonikainstituudi direktori Mart Tamre initsiatiivil. Töös on kaardistatud instituudile kuuluv robot, töö ülesannete määramiseks vajamineva sisendinfo saamiseks abistasid Maido Hiiemaa ning Mart Tamre mind Tallinna Tehnikaülikooli struktuuriüksustega suhtlemisel. Töö teostamisel kasutasin oma isiklikke ruume ning Mektory mehhatroonikalaborit. Soovin tänada oma juhendajat, kes lõputöö kulgemise osas suureks abiks oli.

1. SISSEJUHATUS

Käesoleva töö eesmärk on Double Robotics platvormi uurimine testimise jaoks. *Double* on mehhatroonikainstituuti vahetult enne lõputöö alustamist ostetud robot, mis on omanäoline toode videokõnede tegemiseks. See on kahel rattal balansseeriv püstpendel, millele kinnitatud iPad tahvelarvutisse on laetud programm seadme juhtimiseks mõne teise arvuti või nutiseadme abil kusagilt eemalt. Seade kannab ühte eesmärki, milleks on telekommunikatsiooni parendamine, andes füüsilise kohalolu tunde selle kasutajana tegelikult hoopis kusagil mujal asudes.

Nagu ka iRobot Roomba tolmuimejal, mis on mõeldud koristamiseks ja mida kasutatakse robootikaentusiastide poolt erinevate projektide elluviimiseks, on *double*'il olemas kõik vajalik riistvara ning tarkvara liikumiseks. Kuna seade on kahel rattal balansseeriv robot, siis pakuks selle arendamine sarnaselt prototüüpimisplatvormidega sellele platvormile ka muid eesmärke peale telekommunikatsiooni parendamise. Näiteks oleks võimalik integreerida seadmega erinevaid andureid, et kasutada robotit joonejälgimisrobotina või takistada selle sõitmist vastu objekte. Selleks on tarvis seade kaardistada, et saada teada, millised võimalused käesoleval hetkel selle platvormi arendamiseks olemas on. Kui muutub seadme eesmärk, võib muutuda ka keskkond, milles seade toimib, milleks on tarvis seadet enne nendes tingimustes testida. Lühidalt on selles lõputöös seadme testimiseks vajaliku programmi loomise näitel uuritud erinevaid meetodeid platvormi liigutamiseks fikseeritud parameetritega käskude saatmise abil. Seda testprogrammi kasutades on läbi viidud mõned primitiivsed näidistestid, samuti on loodud esialgne mudel seadme kirjeldamiseks.

Kuna *double*'i esimesed mudelid jõudsid kasutajateni alles 2013. aasta veebruaris ning see on põhiliselt kasutusele võetud kontorites, ei ole seadme ega selle arendamiskatsete kohta kuigi palju informatsiooni. See koosneb lihtsustatuna kahest erinevast osast, kus liikurplatvormi kohta puudub peale mõõtmete ning kaalu pea igasugune muu info, samas kui iPad tahvelarvuti kohta on võimalik informatsiooni leida.

Selleks, et oleks võimalik liikurplatvormi tahvelarvutita kasutada, on tarvis teada, millisel viisil selle liigutamiseks vajalikud käsud seadmele edastatakse. Kuna robotis on see lahendatud iPad-i ning platvormivahelise sinihamba ühendusega, oleks tarvis välja uurida, milliseid pakette tahvelarvuti liikurile edastab. Selleks on tarvilik saada ligipääs liigutamiseks kasutatava tarkvara lähtekoodile või spetsiaalsete seadmete abil ühenduse pealtkuulamine.

Liikuri kasutamine iPad-i vahenduseta võimaldaks seadme lihtsamat integreerimist ka muu tarkvaraga lisaks praegu ainsana olemasolevale iOS tarkvarale. Näiteks lihtsustaks Android tarkvaraga seadme juhtimine erinevate andurite lisamist ning tarkvara loomist. Kuna praegusel hetkel puuduvad meil sinihamba ühenduse pealtkuulamiseks vajalikud seadmed ning ainus taskukohane lahendus ei ole veel täielikult töökindel, siis hetkeolukorras liikurile saadetavate pakettide kättesaamine võimalik ei ole.

Kuna seadme testimiseks erinevates keskkondades on tarvis programmi, mis saadab seadmele liikumiseks vajalikke fikseeritud parameetritega käske ja seadme põhieesmärgi täitmisel liiguvad käsklused kasutaja arvutist läbi internetiserverite iPad-i ning sellelt edasi liikurile, toimub testprogrammi koostamiseks seadme uurimine kitsamast laiemaks. Et otsene suhtlus vahetult liikurplatvormiga praegu võimalik ei ole, on selles töös uuritud võimalusi iPad-iga suhtlemiseks. Tarkvaraarendajad on seda tahvelarvutit palju uurinud ning selle tarbeks rakendusi koostanud, mistõttu ei tohi unustada, et *double* on siinkohal terviklik seade ning iPad-i aplikatsioon peab arvestama ka sellega ühendatud liikuriga.

Läbi tahvelarvuti kulgeva testprogrammi loomiseks olen uurinud seadme edasist arendamist lihtsustavaid võimalusi seadme ühendamiseks teiste seadmetega sinihamba ühenduse abil. Selleks tuleb luua iPad-i rakendus, mille abil teiste seadmete ühendamine võimalik on. Selleks on kasutatud VMware Workstation tarkvara virtuaalse masina loomiseks, kus OS X Mountain Lion operatsioonisüsteemi abil on simuleeritud Apple arvuti keskkonda. Selles keskkonnas on kasutatud Xcode tarkvaraarenduskeskkonda, milles on võimalik luua aplikatsioone iOS tarkvaral töötavatele tahvelarvutitele. Et tahvelarvuti virtuaalse süsteemiga ühendada, on kasutatud iTunes meediatarkvara, mille protokollid võimaldavad tuvastada füüsilise seadme olemasolu ka virtuaalses keskkonnas. Samuti on eemaldatud tahvelarvuti operatsioonisüsteemi piirangud, kasutades Evasion 7 programmi, et valmis rakendus iPad-i laadida.

Kuna iPad kasutab energiasäästlikku sinihambaühendust, on võimalik sellega ühendada vaid teisi seadmeid, mis sama ühendust kasutavad. Uue rakenduse kirjutamine annab võimaluse pääseda ligi ka teistele sinihamba raamistikele, mis teoreetiliselt peaksid võimaldama ühendust kõikvõimalike sinihamba ühendust kasutavate seadmetega. Selles töös on käsitletud nende raamistike ning alternatiivsete sinihamba pinude kasutamisel esinevaid probleeme.

Kasutades *double*'i arendamise jaoks mõeldud tarkvaraarenduskomplekti, on koostatud testprogramm, milles on võimalik valida lisaks seadme liikumise tüübile liikumise kestus,

soovitav stabiliseerimisperiood enne liikumist ning tehtavate tsüklite arv valitud parameetritega. See programm vajab katse alustamiseks vaid ühekordset lähteandmete sisestamist kasutaja poolt, mistõttu erinevalt koduarvuti klaviatuurilt seadme juhtimisest on võimalik katsete korratavus, valides iga katse jaoks soovi korral samad parameetrid.

Selle testprogrammi ning kaamera abil on läbi viidud näidistestid, millest saadud katsematerjali on hiljem Solid Edge'i abiga analüüsitud. Näidistestide idee on pigem loodud programmi katsetamine kui seadme liikumise analüüs, mistõttu on kogutud andmed ligikaudsed, kuid piisavad selleks, et veenduda testprogrammi kõlblikkuses.

Katsetest saadud infot on võimalik kasutada näiteks tegeliku seadme virtuaalse mudeliga vastavusse viimiseks. Käesoleva töö käigus on tasakaaluliikuri tüüpi liikurplatvormi kirjeldva matemaatilise mudeli abil loodud ka mudel *double*'i platvormi kirjeldamiseks. Kuna liikumise sisend tuleb mootoritelt ning platvormi liikuriosa kohta info puudub, siis ei ole selles töös olnud võimalik seda mudelit realiseerida. Kui need andmed selguvad, on võimalik loodud olekuruumivõrrandites need andmed asendada ning mudelit saab kasutada näiteks selliste testide tegemisel, mida tegeliku seadmega teha ei tihkaks.

Töö erinevates peatükkides on kirjeldatud seadet ning toodud välja selle kohta hetkel olemasolev teave, on uuritud erinevaid tarkvaralisi ning riistvaralisi võimalusi platvormi arendamiseks testprogrammi loomise näitel, kusjuures iga sõlmpunkti juures on selle sobivus uuritava seadmega ära kirjeldatud. On näidistestide abil loodud testprogrammi kasutamine ära näidatud ning seadme jaoks matemaatiline mudel koostatud.

2. DOUBLE ROBOTICS PLATVORM

2.1. Mis on *double*?

Double robot on tasakaaluliikuri tüüpi püstpendelmehhanismiga liikur, millele kinnitatud iPad loob võimaluse veebipõhise videokõne läbiviimiseks. *Startup* ettevõte Double Robotics on arendanud robotplatvormi telekommunikatsiooni parendamiseks eeskätt kontorites. Kui varasemalt on videokõned toimunud kindlal kokkulepitud ajal kindlas kohas, siis see robot võimaldab kasutajal kusagilt mujalt töökeskkonda sulanduda pea sama hästi kui ise kohal olles. Kuna robot võib olla kogu aeg sisse lülitatud, piisab kasutajal vaid oma asukohast läbi Chrome'i veebisirvija vastavale veebilehele sisselogimisest ning videokõne võib alata. Eriliseks teeb selle lahenduse see, et kasutajal on võimalik oma asukohast klaviatuuri või iPhone või iPad-i rakenduse abil robotit ruumis liigutada ning seega jõuda õige inimeseni või temaga kaasa liikuda. Seadet on kasutatud ka õpivahendina keskkonnas, kus lapsed, kes ei ole saanud füüsiliselt õppetöös osaleda, saavad seda tänu sellele robotile teha.

Seadme ülaosas on rakis iPad-i kinnitamiseks, mis tuleb seadmest eraldi soetada. Seadmega ühilduvad iPad-id alates teisest põlvkonnast; ühilduvust esimese põlvkonna seadmetega pole seetõttu, et nendel puudub kaamera. Läbi kasutaja seadme ning *double*'i iPad-i veebikaamerate luuakse kahepoolne kaamerapilt. Selleks, et iPad-i kõrgus oleks olukorrale vastav, on võimalik varda pikkust reguleeries valida iPad-i kõrgust 120–150 cm vahel, mis võimaldab kohandamist nii ruumis istuvate kui ka seisvate inimestega.

2.1.1. *Double*'i teadaolevad spetsifikatsioonid

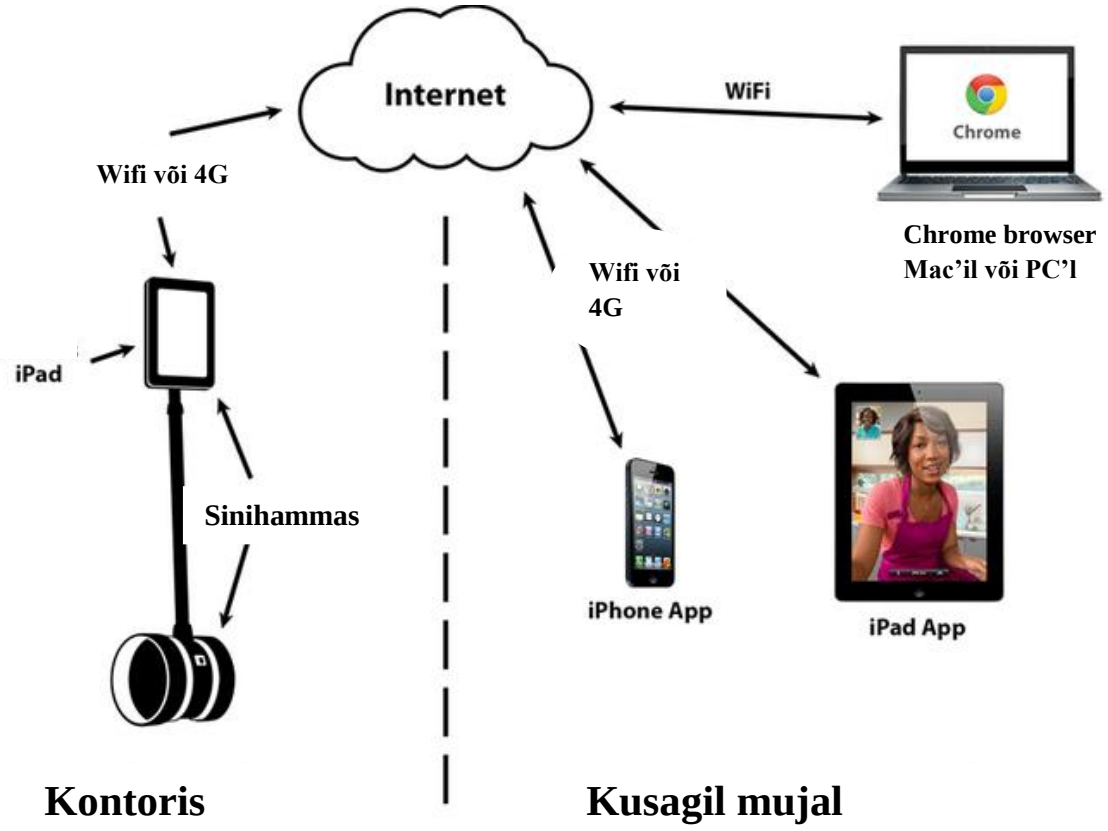
Double'i teadaolevates spetsifikatsioonides on kirjas, et selle kõrgus on kaugjuhitaval teel muudetav vahemikus 47"–59" (1,2–1,5 m) ning selle vertikaalse projektsiooni suurus on 10" x 9" (0,25 x 0,23 m). Seal on täpsustatud, et robot on mõeldud töötamiseks vaid toas ning ta saab hakkama enamiku ruumipindadega, mis on valdavalt tasased, aga suudab üles minna ka kuni viieprotsendilise kaldega pinnast. Toiteallikana on platvormil kasutusel liitiumioon aku, mis kestab ühest laadimisest 8–10 h. Sõidu kaugus lähtepositsioonist on limiteerimata eeldusel, et robot on ühendatud traadita interneti võrku. Küllalt põhjalikku infot on olemas ka iPad-i kohta. LCD ekraan, kõlarid ja videokaamera sõltuvad kõik kasutatavast iPad-ist, kusjuures robot toetab 2., 3. ning 4. seeria iPad-e seetõttu, et erinevalt esimese seeria tahvlitest

on nendel olemas nii esi- kui ka tagakaamera. Võrguühendus on iPad-i ning ruuteri vahel. Videoprotokollina on kasutusel WebRTC standardprotokoll ning video krüptimisel on kasutatud 128-bitist AES krüpteeringut.

Spetsifikatsioonid aga ei maini liikurplatvormi balansseerimist. Puudub igasugune info mootorite ning nende spetsifikatsioonide kohta, samuti ei ole teada kasutatava kontrolleri tüüp. Seadme liikumiskiiruse kohta on üksnes öeldud, et see on aeglane kuni mõõdukas kõndimiskiirus. Seadme massiks on välja toodud 6,8 kg koos iPad-iga, kuid pole teada roboti kaalujaotust, st kui palju kaaluks üksinda platvorm, mis hoiab enda sees akut, mootoreid jms. Ainus, mida platvormi liikuriosa teadaolevalt sisaldab, on aku. Tabel spetsifikatsioonidega on välja toodud lisas 1.

2.1.2. *Double*'i algne tööskeem

Double'i roboti tellimus sisaldab endas baasliikurplatvormi, iPad-i rakist, polt- ning kuuskant mutrivõtit, laadijat ning instruktsioone. [1] Rakisesse asetatakse tagurpidi seadmega ühilduv iPad, kuhu on eelnevalt Apple Store'ist (Apple'i veebipood rakenduste allalaadimiseks) alla tõmmatud tasuta rakendus Double. See rakendus vajab sisselogimist, mis toimub sama kasutaja alt, millega logitakse sisse Chrome'i veebilehel roboti juhtimiseks. Tuleb teha kindlaks, et iPad on ühendatud seadmega sinihamba (ing. k *bluetooth*) kaudu, milleks on tarvilik eelnev paaripanek. Pärast esimest korda tunneb iPad seadme juba ära ning ühenduse loomiseks on tarvilik vaid üks klikk. Samuti tuleb teha kindlaks, et iPad on ühendatud juhtmevaba interneti võrguga, kustkaudu edastatakse käsud, mis kasutaja oma klaviatuurilt sisestab. Senikaua kuni kasutaja oma klaviatuuril nooleklahvi all hoiab, liigub robot vastavalt. Samuti saab kasutaja Chrome'i veebisirvijast reguleerida iPad-i kandva varda kõrgust ning jälgida aku seisu.



Sele 2.1. Seadme algne tööskeem

2.1.3. *Double*'i arendusvõimalused

Kuna Double Robotics jagab oma veebilehel avalikult *double*'i iOS (Apple'i mobiilsetes seadmetes kasutatav tarkvara) tarkvaraarenduskomplekti (ing. k *software development kit*), siis annab see käepärase võimaluse selle liikurplatvormi arendamiseks. Esmalt tuleb aga uurida olemasoleva platvormi võimekust, sest juba lühiajalisel katsetamisel selle funktsionaalses ulatuses on tulnud ette probleeme liikurplatvormi stabiilsusega. Näiteks pole midagi, mis peataks platvormi sõitmist üle juhtmete, mis selle stabiilsust tugevalt häirida võivad. Või jääb üks liikuri pooltest millegi taha kinni, mis on samuti käitumine, mida platvorm endasse programmeeritud algoritmidega lahendada ei suuda. Enne, kui seadeldis anduritega põhjalikult varustada, tuleks analüüsida praegu olemasoleva lahenduse toimetulekut erinevates katsetingimustes, et hiljem oleks võimalik juba konkreetseid olemasolevaid defineeritud probleeme lahendada.

3. ROBOTI MATEMAATILINE MUDEL

Matemaatilise mudeli loomise eesmärgiks on kirjeldada rahuldavalt olemasolevat süsteemi. Virtuaalsete mudelite olemasolul on võimalik katsetada näiteks uusi algoritme, mida tegeliku süsteemi peal katsetada ei tihkaks. Samuti võimaldaks see harjutada roboti juhtimist, sest kirjeldades ära selle käitumise erinevates tingimustes nagu nt liikumise üle lävepakkude või erinevatel kalletel, võimaldab see, juba teades liikumise iseärasusi, juhtida tegelikku platvormi teadlikumalt. Simuleerides mudeli abil liikumist erinevates keskkondades ning analüüsides katsetulemusi, on võimalik tulevase arenduse käigus ka optimeerida praegust kontrollsüsteemilahendust ning miks ka mitte lisada juba olemasolevale uusi elemente.

3.1. Mis on olekumudel

Roboti matemaatiline mudel võimaldab kirjeldada seadme käitumist. Selleks on meil tarvis luua süsteemi olekuruum (ing. k *state space*), mis on abstraktne ruum (vektorruum), mille iga punkt võib vastata teatava süsteemi olekule mingil ajahetkel. Ajaline protsess väljendub siis hetkpunkte siduva joonena (olekutrajektoorina) olekuruumis. Ruumi dimensioon vastab süsteemi järgule, ruumikoordinaatideks on olekumuutujad. Süsteemi olekumuutujate erinevad kogumid vastavad erinevatele koordinaatsüsteemidele olekuruumis. Kusjuures olekumuutuja on süsteemisisene muutuja, mis mingil viisil kajastab süsteemisisest aine, energia vms akumulatsioonivõimet. Süsteemi olekumuutujate kogum on selline minimaalne olekumuutujate hulk, mis täielikult määrab süsteemi akumulatsioonimäära, seega oleku. Süsteemi olekumuutujate piisavat kogumit saab valida erinevalt, kui need muutujad määravad samaväärselt oleku. Tavaliselt eeldatakse, et olekumuutujad või osa neist ei tarvitse olla mõõdetavad või mõõtmiseks kättesaadavad. Olekumudeli abil saab neid aga määrata kaudsete meetoditega. [2]

3.2. Seadme iseärasused matemaatilise mudeli koostamisel

Arvestan mudeli koostamisel seda, et *double*'i rattad puudutavad kogu aeg maad ning libisemist ei toimu. Samuti on külgmised jõud loetud sedavõrd väikseks, et nendega pole vaja arvestada.

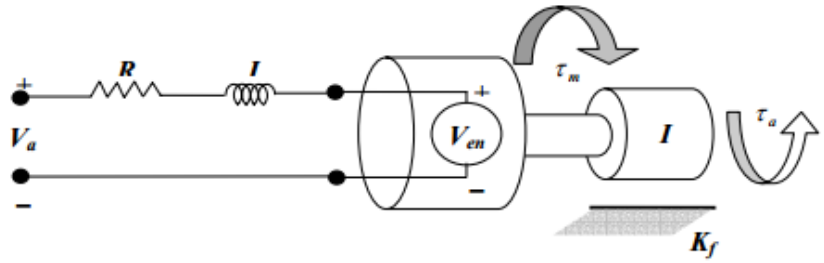
Double balansseerib ennast kasutades kiirendusmõõturit ja güroskoopi, millega mõõdetakse täpne kallutusnurk algasendist. Mõõtetulemuste põhjal liigutab kontroller mootoreid selliselt,

et iPad oleks alati täpselt rataste all. [3] Kuna toote spetsifikatsioonides kasutatava mootori kohta info puudub, siis lähtun mudeli koostamisel sellest, et robot töötab kahe alalisvoolumootoriga, mis saavad oma toite akult, mille puhul on teada, et selle laadimine toimub läbi välise laadija. Kummagi ratta juhtimise jaoks kasutatakse eraldi mootorit.

3.2.1. Töö käik mudeli koostamisel

Mudeli loomisel vaatan kolme eraldiseisvat osa, milleks on mootori dünaamiline mudel, liikurplatvormi rataste osa ning pendel, mille hulka on lihtsuse mõttes arvatud ka iPad. Seejärel seon need osad omavahel, et ära kirjeldada süsteemi ühtne olekuruum. Selles osas toimub vaid mudeli tuletamine, sest seadme liikurmehhanismi kohta teadaolev info on mudeli realiseerimiseks ebapiisav. Nimelt on seadme liikumise uurimisel mudeli sisendiks mootori pinge. Teada pole aga kasutatavate mootorite spetsifikatsioone, samuti puudub võimalus saada teada seadme täpne kaalujaotus.

3.3. Alalisvoolumootori lineaarmudel



Sele 3.1. Alalisvoolumootori skeem

Selle mudeli koostamine on vajalik selleks, et luua süsteemi dünaamilises mudelis seos mootorite sisendpinge ning balansseerimiseks vajaliku kontrollpöördemomendi vahel. Selel 3.1. kujutatud efektiivne alalisvoolumootori lineaarne mudel seletab, et terminalidele rakendatava pinge tulemusena genereeritakse mootoris vool i. Mootor tekitab pöördemomendi τ_m, mis on võrdeline vooluga ning seda on võimalik väljendada valemiga 3.1.

$$\tau_m = k_m i, \qquad (3.1)$$

kus τ_m – mootori pöördemoment,

k_m – pöördemomendi konstant,

i – mootoris indutseeritud vool.

3.3.1. Genereeritud elektromotoorjõud

Takisti-induktiivse reaktiivtakisti näivtakistuse paari jadamisi ühendatult pingega V_e on võimalik kasutada mootori elektriskeemi modelleerimisel. See genereeritud elektromotoorjõu (ing. k *counter electromotive force*) pinge V_e tekib, sest mootori poolused liiguvad läbi magnetvälja. Senikaua kui pooluste geomeetria ning magnetväli on pooluste liikumise ajal konstantsed, muutub indutseeritud jõud ja elektromotoorjõud lineaarselt mootori võlli nurkkiirusega. [4] Genereeritud pinge võib üldistada nagu funktsiooni võlli kiirusest ning saab kirjutada valemina 3.2

$$V_e = k_e \omega, \tag{3.2}$$

kus V_e – genereeritud elektromotoorjõud,

k_e – genereeritud elektromotoorjõu konstant,

ω – mootori võlli nurkkiirus.

3.3.2. Lineaarne diferentsiaalvõrrand alalisvoolumootori jaoks

Lineaarne diferentsiaalvõrrand alalisvoolumootori jaoks on võimalik tuletada Kirchoffi II seadusest, mille kohaselt on suletud kontuuris elektromotoorjõudude algebraline summa võrdne pingete algebralise summaga, mistõttu võrdsustades võrrandi nulliga, saame, et suletud kontuuris peab kõikide pingete algebraline summa võrduma nulliga. Alalisvoolumootori jaoks saame kirjutada valemi

$$V_a - Ri - L\frac{di}{dt} = 0, \tag{3.3}$$

kus V_a – terminalidele rakendatud pinge,

R – nominaalne pinge takistil,

L – induktsioon induktiivtakistuselt rootoris,

$\frac{di}{dt}$ – indutseeritud voolu tuletis aja järgi.

3.3.3. Pöördkeha inertsimoment

Mootori liikumisvõrrandi tuletamisel tehakse üldistus, et mootori võlli hõõrdumine k_f on lineaarne funktsioon võlli kiirusest.

Newtoni teine seadus väidab, et kehale mõjuv resultantjõud on võrdne keha massi ja kiirenduse korrutisega.

$$\overrightarrow{F} = m * \overrightarrow{a} \tag{3.4}$$

kus F – kehale mõjuv resultantjõud,

m – keha mass,

a – keha kiirendus.

Seda seadust pöörleva keha puhul kasutades selgub, et iga keha omab vastavat pöördinertsi, mis seob rakendatud pöördemomendi tuleneva nurkkiirendusega

$$\tau = I\alpha = I\dot{\omega}, \tag{3.5}$$

kus τ – keha pöördemoment,

I – keha inertsimoment,

$\dot{\omega}$ – keha kiirendus.

Inertsimoment on pöördkeha puhul vastavalt massi ekvivalent ehk selle põhjal on kõikide võllil rakenduvate pöördemomentide summa M lineaarselt seotud võlli nurkkiirenduse ning armatuuri massi inertsi korrutisega.

$$\sum M = \tau_m - k_f\omega - \tau_a = I_R\omega, \tag{3.6}$$

kus M – võllil rakenduvate pöördemomentide summa,

τ_a – rakendatud pöördemoment,

k_f – hõõrdumise konstant,

I_R – mootori armatuuri massi inerts.

3.3.4. Mootori liikumise juhtimine

Asendades võrrandid 3.1 ja 3.2 võrranditesse 3.3 ja 3.6 ning viies tuletised aja järgi vasakule ning ülejäänu paremale, saame kaks võrrandit, millega juhitakse mootori liikumist

$$\frac{di}{dt} = \frac{R}{L}i + \frac{k_e}{L} + \frac{V_a}{L} \tag{3.7}$$

$$\frac{d\omega}{dt} = \frac{k_m}{I_R}i + \frac{-k_e}{I_R}\omega - \frac{\tau_a}{I_R} \tag{3.8}$$

Mõlemad võrrandid on lineaarsed funktsioonid voolust ning nurkkiirusest ning sisaldavad esimest järku tuletisi. Lihtsustatud alalisvoolumootori mudel on balansseeriva roboti mudeli jaoks piisav. Sellel põhjusel loetakse mootori induktsioon ning hõõrdumine minimaalseks ning siinkohal võrdseks nulliga. Seetõttu saab avaldised 3.7 ja 3.8 esitada lihtsustatud kujul

$$i = -\frac{k_e}{R}\omega + \frac{1}{R}V_a \tag{3.9}$$

$$\frac{d\omega}{dt} = \frac{k_m}{I_R}i - \frac{\tau_a}{I_R} \tag{3.10}$$

Asendades võrrandi 3.9 võrrandisse 3.10 saame üldistuse alalisvoolumootori mudelile, mis on funktsioon hetkelisest mootori pöörlemise kiirusest, rakendatud pingest ning rakendatud pöördemomendist

$$\frac{d\omega}{dt} = -\frac{k_m k_e}{I_R R}\omega + \frac{1}{I_R R}V_a - \frac{\tau_a}{I_R} \tag{3.11}$$

3.3.5. Mootori olekuruumi mudel

Kuna mootori induktsiooni ei arvestata, siis mähist läbivat voolu mootori liikumisvõrrandis ei käsitleta. Erinevalt võlli nurkkiirusest, mis vajab pärast sisendpinge muutmist aega algkiiruselt lõppkiirusele jõudmiseks, jõuaks vool püsivasse seisundisse kohe.

Mootori dünaamikat on võimalik kirjeldada olekuruumi mudeliga, mis on süsteem esimest järku diferentsiaalvõrranditest parameetritega võlli ringjooneline asukoht θ ning nurkkiirus ω, mis on piisavad mudeli kirjeldamiseks. Sisendmuutujateks on rakendatud pinge ning pöördemoment.

$$\begin{bmatrix}\dot{\theta}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & \frac{-k_m k_e}{I_R R}\end{bmatrix}\begin{bmatrix}\theta\\ \omega\end{bmatrix} + \begin{bmatrix}0 & 0\\ \frac{k_m}{I_R R} & \frac{-1}{I_R}\end{bmatrix}\begin{bmatrix}V_a\\ \tau_a\end{bmatrix} \quad (3.12)$$

$$y = [1 \quad 0]\begin{bmatrix}\theta\\ \omega\end{bmatrix} + [0 \quad 0]\begin{bmatrix}V_a\\ \tau_a\end{bmatrix} \quad (3.13)$$

3.4. Dünaamiline mudel kaherattalisele pöördpendlile

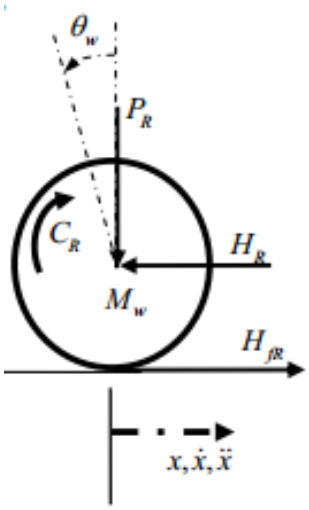
Kuigi kaherattaline pöördpendel on dünaamikas keeruline süsteem, on sellel sarnasusi neljarattalisele kärule toeuva pöördpendliga, mis on süsteemina rohkem levinud. Pendli ning rataste dünaamikat analüüsitakse esialgu eraldi, mille tulemusena saame kaks liikumisvõrrandit, mis kirjeldavad balansseeriva roboti käitumist täielikult.

Kuna roboti käitumist võivad mõjutada nii häiringud kui ka mootori pöördemoment, peab matemaatiline mudel mõlema juhuga arvestama.

3.4.1. Rataste dünaamika

Esmalt leitakse liikumisvõrrandid parema ning vasaku ratta jaoks. Kuna platvorm on sümmeetriline ning mõlema ratta liikumisvõrrand on analoogne, on välja toodud vaid parema ratta võrrand. Selel 3.2. on kujutatud parema ratta vaba keha diagramm, kus C_R on ratastele mootori poolt rakendatav pöördemoment, M_w on ratta mass, θ_w tähistab ratta pöördenurka, H_{fR} on hõõrdejõud ratta ning põranda kokkupuutekohas, H_L, H_R ja P_R on reaktsioonijõud rataste ning pendli vahel.

R



Sele 3.2. Ratta vaba keha diagramm

Rataste liikumisvõrrandid

Newtoni teisest seadusest lähtuvalt on jõudude summa x-telje suunas

$$\sum F_x = Ma$$

$$M_w\ddot{x} = H_{fR} - H_R, \tag{3.14}$$

kus M_w – ratta mass,

H_{fR} – hõõrdejõud ratta ning maapinna vahel,

H_R – reaktsioonijõud rataste ja pendli vahel parema ratta jaoks,

$\ddot{x}$ – ratta keskme kiirendus.

Momentide summa ratta keskme ümber

$$\sum M_o = I\alpha$$

$$I_w\ddot{\theta}_w = C_R - H_{fR}r\ , \tag{3.15}$$

kus I_w – ratta inertsimoment,

$\ddot{\theta}_w$ – ratta nurkkiirendus,

C_R – paremale rattale mootori poolt rakendatav pöördemoment,

R – ratta raadius.

Eespool leitud alalisvoolumootori dünaamikast saab mootori pöördemomenti väljendada järgnevalt

$$\tau_m = I_R\frac{d\omega}{dt} + \tau_a \tag{3.16}$$

Pärast võrrandi liikmete ümberpaigutamist ning alalisvoolumootori tuletamise lõigust parameetrite asendamist saame väljundpöördemomendi ratastele

$$C = I_R\frac{d\omega}{dt} = \frac{-k_m k_e}{R}\dot{\theta}_w + \frac{k_m}{R}V_a, \tag{3.17}$$

kus C – väljundpöördemoment ratastele,

$\dot{\theta}_w$ – ratta nurkkiirus,

mille tagajärjel saab võrrandist 3.15

$$I_w\ddot{\theta}_w = \frac{-k_m k_e}{R}\dot{\theta}_w + \frac{k_m}{R}V_a - H_{fR}r \tag{3.18}$$

ning

$$H_{fR} = \frac{-k_m k_e}{Rr}\dot{\theta}_w + \frac{k_m}{Rr}V_a - \frac{I_w}{r}\ddot{\theta}_w \quad (3.19)$$

Võrrand 3.17 asendatakse võrrandisse 3.14, et saada võrrandid vasaku ja parema ratta jaoks.

Vasaku ratta jaoks

$$M_w\ddot{x} = \frac{-k_m k_e}{Rr}\dot{\theta}_w + \frac{k_m}{Rr}V_a - \frac{I_w}{r}\ddot{\theta}_w - H_L, \quad (3.20)$$

kus H_L – reaktsioonijõud rataste ja pendli vahel vasaku ratta jaoks.

Parema ratta jaoks

$$M_w\ddot{x} = \frac{-k_m k_e}{Rr}\dot{\theta}_w + \frac{k_m}{Rr}V_a - \frac{I_w}{r}\ddot{\theta}_w - H_R \quad (3.21)$$

Pöörlemise teisendamine lineaarliikumiseks

Kuna ratta keskme liikumine on lineaarne piki pinda, millel ratas liigub, siis on võimalik pöörlemine teisendada lineaarliikumiseks kasutades seoseid

$$\ddot{\theta}_w r = \ddot{x} \;\rightarrow\; \ddot{\theta}_w = \frac{\ddot{x}}{r}$$

$$\dot{\theta}_w r = \dot{x} \;\rightarrow\; \dot{\theta}_w = \frac{\dot{x}}{r}$$

kus $\dot{x}$ – ratta keskme liikumise kiirus.

Asendades lineaarliikumise võrranditesse 3.20 ja 3.21, saame lineaarsed võrrandid mõlema ratta jaoks.

Vasaku ratta jaoks

$$M_w\ddot{x} = \frac{-k_m k_e}{Rr^2}\dot{x} + \frac{k_m}{Rr}V_a - \frac{I_w}{r^2}\ddot{x} - H_L \quad (3.22)$$

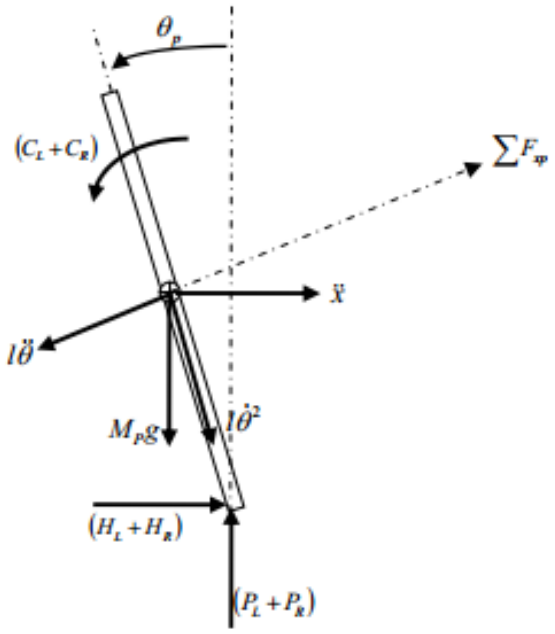
Parema ratta jaoks

$$M_w\ddot{x} = \frac{-k_m k_e}{Rr^2}\dot{x} + \frac{k_m}{Rr}V_a - \frac{I_w}{r^2}\ddot{x} - H_R \quad (3.23)$$

Liites võrrandid 3.22 ja 3.23 kokku, saame

$$2\left(M_w + \frac{I_w}{r^2}\right)\ddot{x} = \frac{-2k_m k_e}{Rr^2}\dot{x} + \frac{2k_m}{Rr}V_a - (H_L + H_R) \quad (3.24)$$

3.4.2. Pendli dünaamika



Sele 3.3. Pendli vaba keha diagramm

Mudeli lihtsustamiseks on pendliks taandatud kogu seadme ülemine pool vardast iPad-ini.

Pendli liikumisvõrrandid

Newtoni teisest seadusest lähtuvalt liidame kokku kõik horisontaalsuunas mõjuvad jõud.

$$\sum F_x = M_p\ddot{x}$$

$$(H_L + H_R) - M_p l\ddot{\theta}_p cos\theta_p + M_p l\dot{\theta}^2{}_p sin\theta_p = M_p\ddot{x}, \quad (3.25)$$

kus M_p – pendli mass,

$\ddot{\theta}_p$ – pendli liikumise nurkkiirendus,

$\dot{\theta}_p$ – pendli nurkkiirus,

θ_p – pendli nurk vertikaalsest tasakaaluasendist,

l – pendli pikkus,

$\ddot{x}$ – pendli raskuskeskme horisontaalsuunalise liikumise kiirendus,

millest

$$(H_L + H_R) = M_p\ddot{x} + M_p l\ddot{\theta}_p cos\theta_p - M_p l\dot{\theta}^2{}_p sin\theta_p \quad (3.26)$$

Pendliga ristiolevate jõudude summa

$$\sum F_{xp} = M_p\ddot{x}cos\theta_p$$

$$(H_L + H_R)cos\theta_p + (P_L + P_R)sin\theta_p - M_p g\ sin\theta_p - M_p l\ddot{\theta}_p = M_p\ddot{x}cos\theta_p, \quad (3.27)$$

kus $P_{L,}\, P_R$ – reaktsioonijõud rataste ning pendli vahel.

Kõikide momentide summa pendli raskuskeskme ümber

$$\sum M_o = I\alpha$$

$$-(H_L + H_R)lcos\theta_p - (P_L + P_R)lsin\theta_p - (C_L + C_R) = I_p\ddot{\theta}_p, \quad (3.28)$$

kus C_L – vasakule rattale mootori poolt rakendatav pöördemoment,

I_p – pendli inertsimoment.

Pärast lineariseerimist mootori poolt avaldatav pöördemoment pendlile võrrandist 3.17

$$C_L + C_R = \frac{-2k_m k_e}{R}\frac{\dot{x}}{r} + \frac{2k_m}{R}V_a$$

ning asendades selle võrrandisse 3.28

$$-(H_L + H_R)lcos\theta_p - (P_L + P_R)lsin\theta_p - \left(\frac{-2k_m k_e}{Rr}\ \dot{x} + \frac{2k_m}{R}V_a\right) = I_p\ddot{\theta}_p$$

millest

$$-(H_L + H_R)lcos\theta_p - (P_L + P_R)lsin\theta_p = I_p\ddot{\theta}_p - \frac{2k_m k_e}{Rr}\ \dot{x} + \frac{2k_m}{R}V_a \quad (3.29)$$

Korrutades võrrandi 3.23 läbi –*l*-iga

$$\left[-(H_L + H_R)lcos\theta_p - (P_L + P_R)lsin\theta_p\right] + M_p glsin\theta_p + M_p l^2\ddot{\theta}_p = -M_p l\ddot{x}cos\theta_p \quad (3.30)$$

Asendades võrrandi 3.29 võrrandisse 3.30

$$I_p\ddot{\theta}_p - \frac{2k_m k_e}{Rr}\ \dot{x} + \frac{2k_m}{R}V_a + M_p glsin\theta_p + M_p l^2\ddot{\theta}_p = -M_p l\ddot{x}cos\theta_p \quad (3.31)$$

Selleks, et eemaldada $(H_L + H_R)$ mootori dünaamikast, asendame võrrandi 3.26 võrrandisse 3.24

$$2\left(M_w + \frac{I_w}{r^2}\right)\ddot{x} = \frac{-2k_m k_e}{Rr^2}\ \dot{x} + \frac{2k_m}{Rr}V_a - M_p\ddot{x} - M_p l\ddot{\theta}_p cos\theta_p + M_p l\dot{\theta}^2{}_p sin\theta_p \quad (3.32)$$

Süsteemi mittelineaarsed võrrandid

Korrastades võrrandid 3.31 ning 3.32, saame süsteemi liikumise mittelineaarsed võrrandid

$$\left(I_p + M_p l^2\right)\ddot{\theta}_p - \frac{2k_m k_e}{Rr}\,\dot{x} + \frac{2k_m}{R}V_a + M_p g l sin\theta_p = -M_p l\ddot{x}cos\theta_p \qquad (3.33)$$

$$\frac{2k_m}{Rr}V_a = \left(2M_w + \frac{2I_w}{r^2} + M_p\right)\ddot{x} + \frac{2k_m k_e}{Rr^2}\,\dot{x} + M_p l\ddot{\theta}_p cos\theta_p - M_p l\dot{\theta}^2{}_p sin\theta_p \qquad (3.34)$$

Võrrandite lineariseerimine

Avaldised 3.33 ning 3.34 on võimalik lineariseerida eeldusel, et $\theta_p = \pi + \varphi$, kus φ on väike nurk pendli vertikaalsest tasakaaluasendist. Siinkohal on lineariseerimine vajalik selleks, et luua lineaarne olekuruum.

Seega

$$cos\theta_p = -1, sin\theta_p = -\varphi \text{ ja } \left(\frac{d\theta_p}{dt}\right)^2 = 0,$$

kus φ – nurk pendli vertikaalsest tasakaaluasendist.

Lineaarne liikumisvõrrand on

$$\left(I_p + M_p l^2\right)\ddot{\varphi} - \frac{2k_m k_e}{Rr}\,\dot{x} + \frac{2k_m}{R}V_a - M_p g l\varphi = M_p l\ddot{x} \qquad (3.35)$$

$$\frac{2k_m}{Rr}V_a = \left(2M_w + \frac{2I_w}{r^2} + M_p\right)\ddot{x} + \frac{2k_m k_e}{Rr^2}\,\dot{x} + M_p l\ddot{\varphi} \qquad (3.36)$$

Olekuruumi maatriksite koostamiseks korrastame võrrandid 3.35 ja 3.36 selliselt, et avalduksid nurga teine tuletis aja järgi ning nihke teine tuletis aja järgi

$$\ddot{\varphi} = \frac{M_p l}{\left(I_p + M_p l^2\right)}\ddot{x} + \frac{2k_m k_e}{Rr\left(I_p + M_p l^2\right)}\dot{x} - \frac{2k_m}{R\left(I_p + M_p l^2\right)}V_a + \frac{M_p g l}{\left(I_p + M_p l^2\right)}\varphi \qquad (3.37)$$

$$\ddot{x} = \frac{2k_m}{Rr\left(2M_w + \frac{2I_w}{r^2} + M_p\right)}V_a - \frac{2k_m k_e}{Rr^2\left(2M_w + \frac{2I_w}{r^2} + M_p\right)}\dot{x} + \frac{M_p l}{\left(2M_w + \frac{2I_w}{r^2} + M_p\right)}\ddot{\varphi} \qquad (3.38)$$

Asendades avaldise 3.37 avaldisse 3.36, avaldise 3.38 avaldisse 3.35 ning korrastades võrrandeid algebraliselt, saame olekuruumi võrrandi

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\varphi}\\ \ddot{\varphi}\end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0\\ 0 & \frac{2k_m k_e\left(M_p lr - I_p - M_p l^2\right)}{Rr^2\alpha} & \frac{{M_p}^2 g l^2}{\alpha} & 0\\ 0 & 0 & 0 & 1\\ 0 & \frac{2k_m k_e\left(r\beta - M_p l\right)}{Rr^2\alpha} & \frac{M_p g l\beta}{\alpha} & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \varphi\\ \dot{\varphi}\end{bmatrix} + \begin{bmatrix} 0\\ \frac{2k_m\left(I_p + M_p l^2 - M_p lr\right)}{Rr\alpha}\\ 0\\ \frac{2k_m\left(M_p l - r\beta\right)}{Rr\alpha}\end{bmatrix}V_a \qquad (3.39)$$

Kus

$$\beta = \left(2M_w + \frac{2I_w}{r^2} + M_p\right) \qquad \alpha = \left[I_p\beta + 2M_p l^2\left(M_w + \frac{I_w}{r^2}\right)\right]$$

[5]

3.5. Mudeli realiseerimine

Mudeli realiseerimiseks olen süsteemi iseärasusi arvestades tuletanud olekuruumi võrrandid. Järgmise sammuna tuleks simuleerida kontrollerit, mille üheks osaks on eelmises punktis tuletatud olekuruumi võrrandid, kuhu on asendatud meie süsteemi tegelikud parameetrid.
Kuna olulisteks parameetriteks olekuruumi kirjeldamisel on meile tundmatud väärtused nagu mootori rakendatav pinge, pöördemomendi konstant, genereeritud elektromotoorjõu konstant ning pole teada terviksüsteemi ligikaudne kaalujaotus, ei ole võimalik olekuruumi piisaval määral kirjeldada. Samuti pole teada balansseerimissüsteemis kasutatava kontrolleri tüüp, mistõttu praegusel hetkel ei ole võimalik terviklikku mudelit luua. Edasises töös uuritakse olemasoleva roboti käitumist katsetuste abil.

4. ARENDUSVAHENDITE LEIDMINE

Püstpendelmehhanismi testimiseks erinevatel pindadel on tarvis, et kõigi katsete tingimused oleksid võrdsed, mida olemasoleva lahendusega garanteerida ei saa. Kuna praegu saab pendelmehhanism liikumise sisendi arvuti või Apple'i nutiseadme klaviatuurilt vastavalt impulsi pikkusele, ei saa tagada katsetingimuste korratavust. Platvorm peab olema enne liikumise alustamist stabiilne ning liikumine peab olema kõigis katsetes ühepikkune. Selle tagamiseks tuleb koostada programm. Programmi loomisel tuleb kõigepealt teha kindlaks, millise riistvara osa mõjutamiseks see on mõeldud. Võimalus on mõjutada püstpendelmehhanismi, iPad-i ning ka platvormi esialgses rakenduses kasutatavat arvutit. Seejärel tuleb uurida erinevate lisaseadmete vajalikkust ning ühendamise võimalikkust ning optimaalsust. Lõpuks tuleb valida võimalikest lahendustest optimaalseim.

4.1. Suhtlus püstpendelmehhanismiga

Kuna töö sisuks on Double Robotics platvormi uurimine ning sellel testide läbiviimine, siis on kõige vahetum moodus seadet testida kontrolli võtmine otse püstpendelplatvormi üle. Kuna süsteemi balansseerimine on ära tehtud platvormi sees ning on kinnine, on tarvis juurde pääseda veel vaid sinihamba ühendusele iPad-i ning platvormi vahel, et oleks võimalik platvormi vahetult juhtida mõne teise sinihamba seadmega. Näiteks kui on teada sinihamba protokollid, mis saadetakse praegu iPad-i ning platvormi vahel, on võimalik kirjutada mõnda nutitelefoni rakendus platvormi juhtimiseks kasutades samasugust protokolli. Selliselt ei sõltuks platvormi testimine iPad-i ning platvormi vahelisest ühendusest ning kogu süsteem oleks vaid sinihamba ühendusel kokkuvõttes lihtsam.

4.1.1. Sinihammas

Sinihammas on kahesuunaline lähiala raadiolink, mis toimib sagedusel 2,4–2,485 GHz (mis on üsna sarnane WiFi-võrkude sagedusega) ning võimaldab erinevatel elektrilistel seadmetel luua andmesideühenduse ilma füüsilist ühendust loomata [6]

Sinihammas kasutab *full-duplex* signaali sagedusspektrumil sagedushüpetega nominaaltaktis 1600 hüpet sekundis. See tähendab, et andmeid on võimalik sellistel sagedustel samaaegselt vastu võtta ning saata. Sagedushüppamise võime on disainitud selleks, et vähendada erinevate

seadmete kokkupuudet, mis samuti kasutavad juhtmevaba tehnoloogiat sagedusel 2,4 GHz. Nimelt tuvastab sinihamba tehnoloogia teised seadmed spektril ning väldib sagedusi, mis nende poolt juba hõivatud on. Selline adaptiivne hüppamine 79-l sagedusel 1 MHz intervallidega annab kõrge efektiivsuse andmete saatmisel selle juhtmevaba spektri kasutamisel. [7]

Ülemseadme (ing. k *master*) saab samaaegselt ühendada seitsme alamseadme (ing. k *slave*) külge, kusjuures kõik seadmed jagavad taktsignaali (ing. k *clock*) sünkroonimiseks. Ühenduses olevad seadmed kasutavad ülemseadme informatsiooni, et sünkrooni jääda, ülemseadme ID määrab ära sagedushüpete mustri ning ülemseadme taktsignaal määrab ära nende hüpete faasi. Sinihamba ID on unikaalne 48-bitine number, mida kasutatakse sinihamba protokollis läbivalt – identifitseerimisel, autentimisel, sünkroonimisel. Kõik ühendatud seadmed edastavad infot kasutades ülemseadme ID-d. [8]

4.1.2. Sinihamba ühenduse pealtkuulamine

Selleks, et ellu viia käsud, mis saadetakse iPad-i ning platvormi vahel mõnel teisel seadmel, tuleb esmalt teha kindlaks, milline on infovahetus. Levinud programmid nagu Wireshark, mis võimaldavad võrguprotokollid kinni püüda, sinihamba puhul ei aita. Sinihamba protokolle on võimalik kinni püüda Linuxi operatsioonisüsteemil vastava sinihamba adapteri olemasolul ning ka siis on kinnipüütavad paketid vaid need, mis sisaldavad suhtlust selle sama arvuti ning mõne seadme vahel, mistõttu selline lahendus ka sinihamba adapteri olemasolul passiivse ühenduse pealtkuulamise vajadusel ei aitaks. Nimelt filtreeritakse juba madalal sinihamba pinus (ing. k *stack*) välja paketid, mis pole mõeldud protsessorile saatmiseks. Selleks, et ükskõik millist paketti üle juhtmevaba ühenduse pealt kuulata, peab vastuvõtja opereerima n-ö valimatus režiimis (ing. k *promiscuous mode*), kus saadab kõik paketid olenemata sellest, mis seadmele need mõeldud olid, edasi protsessorile. Aga selle valimatu režiimi rakendamine sinihamba ühenduses on tootja poolt vaadates kallis ning ebavajalik, mistõttu valdav osa seadmetest seda ei toeta. Nendel harvadel juhtudel, kus seadmete poolt on valimatu režiim riistvaras toetatud, on püsivara suletud lähtekoodiga (ing. k *source code*), mistõttu pole ka sellest kasu. [8] Pole välistatud, et tulevikus on võimalik sinihamba pakette ka Wiresharkiga püüda, praegune Wireshark'i Windowsi 1.10.7 versioon seda võimalust aga ei toeta.

Ubertooth One

Kuna meil puudub praegusel hetkel vajalik riistvara sinihamba ühenduse passiivseks pealtkuulamiseks, siis tuleb uurida teisi riistvaralisi lahendusi. Praegu on sinihamba pealtkuulamiseks vajalikud seadmed turul kallid, kuid on üks taskukohane lahendus, milleks on Ubertooth One. See koosneb riistvara toest passiivseks pakettide kuulamiseks, avatud lähtekoodiga püsivarast ning tarkvarast, kuid on veel arendusfaasis, mistõttu ei saa seda veel täielikuks pidada. MIT poolt koostatud „Hacking Bluetooth“ projektis teostatud katsetest Ubertoothiga võib lugeda välja käesoleva töö jaoks vajaliku, et suudeti tuvastada passiivselt sinihamba seadmeid isegi siis, kui need olid avastatavast viisist välja lülitatud, suudeti jälgida ühte sinihamba seadet hüppamas erinevatel sageduskanalitel, passiivselt koguda sinihamba liikluse andmed ühel kanalil ning esitada tulemused Kismeti lisamooduliga Wiresharkis. Ometi ei olnud võimalik püüda sinihamba liiklust läbi kõikide kanalite ning esitada seda lihtsalt analüüsitaval viisil. [8] Katsetuste järeldusest saab välja lugeda, et sellise lahendusega pole võimalik täielikult sinihamba ühendust pealt kuulata ega seda esitada.

Ühenduse krüptimine

Isegi kui Ubertoothi abiga oleks võimalik kuulata pealt kõiki pakette, mis võimaldaksid koguda infot platvormi juhtimiseks sinihamba abil, ei ole mingisugust garantiid, et sealt saadavad andmed krüptitud pole. Krüpteerimine tähendab loetaval kujul oleva informatsiooni muutmist loetamatuks. Pealtnäha kaootiline tulemus allub siiski teatud reeglitele ehk algoritmile, mis võimaldab vastava šifri ehk võtme abil muuta krüpteeritud informatsiooni taas loetavaks ehk selle dekrüpteerida. [9]

Kuna pole teada, millise šifri alusel võib sinihamba ühendus olla krüptitud, siis on tõenäoline, et seda testprogrammi koostamise jaoks loetavaks teha pole võimalik. Näiteks kui on kokku korrutatud neli arvu ja kui teada on vaid korrutis, on väga keeruline ennustada, millised olid need algsed neli arvu, mille kokku korrutamisel selline tulemus saadi.

Järeldus sinihamba ühenduse pealtkuulamise osas käesoleva töö raames

Kuna puudub juurdepääs seadmetele, mille abil on võimalik passiivselt sinihamba ühendust pealt kuulata ning ainus taskukohane variant vajab veel täiustamist ja sellega eksperimenteerimine ei annaks mingisugust garantiid vajaliku info kättesaamiseks, siis käesoleva töö raames sinihamba passiivset kuulamist ei teostata ning vahetult platvormi liigutamist sinihamba seadmega ei kasutata.

4.2. Suhtlus iPad-iga

4.2.1. Suhtluse võimaldamiseks vajalik tarkvara

Double Robotics'i veebilehelt on võimalik juurde pääseda iOSi tarkvaraarenduskomplektile, mis on sedavõrd piisav, et spetsiaalse tarkvara olemasolul on võimalik luua rakendus, mis käsud platvormi liigutamiseks sinihamba abil platvormile edastab. See tähendab, et kui saada ühendus iPad-iga, on võimalik liikumiskäsklused edasi kanda liikurplatvormile. Selleks, et mis tahes iPad-i aplikatsiooni kirjutada, on tarvis Xcode arenduskeskkonda, mis on Apple'i vabavara spetsiaalselt Mac OSX või iOS tarkvara programmeerimiseks. Selle tarkvara kasutamiseks kasutasin VMWare Workstation'it, mis võimaldab minu kasutatavas Windows 7 operatsioonisüsteemis luua virtuaalmasina ning sellega erinevaid operatsioonisüsteeme simuleerida. Virtuaalse masinana kasutasin Mountain Lion OS 10.8 operatsioonisüsteemi, mille pidin uuendama OS 10.8.4 peale, et saaks alla tõmmata iPad-i iOS 7.0.4 tarkvaraarenduskomplekti toetusega Xcode 5. Selleks, et saada ühendus iPad-i ning Xcode'i vahel, mis virtuaalses masinas töötab, kasutasin universaalset järjestiksiin USB (ing. k *universal serial bus*) kaablit, oma Windows 7 operatsioonisüsteemi ning Mountain Lion 10.8.4 installeeritud iTunes'i, mis kasutab samasuguseid protokolle, mis on vajalikud iPad-i sünkroniseerimiseks virtuaalse masinaga.

Kuna Apple'i arendajakonto on tasuline ning eestkätt mõeldud selleks, et valmis tehtud ralendusi ka Apple Store'is jagada saaks, leidsin, et optimaalne on iPad-i operatsioonisüsteemilt piirangud eemaldada (ing. k *jailbreak*). Operatsioonisüsteemi piirangute eemaldamine tähendab seda, et Apple'i krüptitud failisüsteemile saab ka kasutaja ligi ning see võimaldab kasutajal faile muuta. Samuti saab installeerida rakendusi, mis pole Apple'i poolt heaks kiidetud. See pole küll ebaseaduslik, kuid rikub Apple'i seadme garantiitingimusi. Praegusel juhul on tarvis operatsioonisüsteemilt piirangud eemaldada selleks, et oleks võimalik laadida Xcode'is kirjutatud rakendus füüsilise seadme peale, mitte üksnes rakendust Xcode'i sees olevas simulaatoris käitada. Operatsioonisüsteemi piirangute eemaldamise jaoks kasutasin Evasion 7 tarkvara, mis pärast Windows 7-le allalaadimist juhtme otsas olevas iPad-is iseseisvalt protsessi läbi viis. Peale seda kasutasin enda OSX-i, et luua Keychain Access'is (programm Apple'i operatsioonisüsteemis, milles käsitletakse

turvakoode) sertifikaat, mida kasutasin Xcode'i turvaseadetes, kus selle olemasolu on vajalik, et rakendus füüsilisel seadmel käivitada.

4.2.2. Puldiga juhtimine

Puldiga juhtimine on hea variant seetõttu, et on võimalik anda seadmele rida erinevaid käsklusi ilma vahepeal seadet puudutamata ning seda tasakaalust välja viimata. Lisaks rajab seadme puldiga juhtimine teed teistele rakendustele, mis seadme edasisel arendamisel kasuks tuleksid. Selleks, et programm oleks võimalikult universaalne, tuleb kasutada puldiga juhtimisel sinihamba tehnoloogiat, sest näiteks seadme juhtimine läbi interneti vajab juhtmevaba interneti olemasolu, mida erinevalt sinihamba ühendusest pole igal pool võimalik tagada.

4.2.3. Apple'i energiasäästlik tehnoloogia

Uuemad iPad-id kasutavad energiasäästlikku sinihammast, mis põhineb sinihammas 4.0 spetsifikatsioonil, mis lisaks muudele protokollidele defineerib ära ka protokollid, mida on vaja suhtluseks energiasäästlike seadmete vahel. Selleks, et kirjutada rakendus, mis kasutab iPad-i sinihammast, on Xcode 5-s energiasäästliku sinihamba raamistik (CoreBluetooth Framework), mille lisamisel enda kirjutatavasse koodi on võimalik kasutada erinevaid käske sinihamba ühenduse ülesseadmiseks ning ümberkaudsete seadmete leidmiseks. Selles on seadmed jagatud rollidesse, kus tuumik (ing. k *core*), milleks võibki näiteks olla iPad, on võimeline tuvastama äärealade seadmeid (ing. k *pheripheral*), mille infot soovides võib tuumik ääreala seadmega ühenduse luua. Energiasäästlik sinihammas on üsna uus tehnoloogia, mistõttu valdav osa mängukonsoolide kontrollereid ning nutiseadmeid seda ei toeta. Lisaks on võimalik selle iPad-i sinihamba ühenduse olemasolul ümberkaudseid seadmeid skaneerida, mille tulemusel leiab seade vaid teised uuemad Apple'i seadmed. Nimelt lisaks piirangule kasutada Xcode keskkonnas energiasäästliku sinihamba raamistikku, peavad seadmed olema ka iPad-ile tehtud (edaspidi MFI) (ing. k *made for iPad*) sertifikaadiga. [10] Selliseid seadmeid pole kuigi palju ning Apple kiidab heaks vaid seadmed, mis kasutavad energiasäästlikku sinihammast või siis mõnda iOSi poolt toetatud standardset protokolli, milleks selle seadme puhul on HFP 1.6, PBAP, A2DP, AVRCP 1.4, PAN, HID, MAP. Lisaks on iOS seadmete sinihamba ühenduseks vajalik ühendatava seadme

krüpteeringutoe olemasolu, sest seadmete paaripanemisega seatakse üles ka seadmetevaheline krüpteering. [11]

4.2.4. Wii pult

Kuna iOS 7.0.4 toetab HID (Human Interface Device) protokolli, siis oleks üheks võimaluseks kasutada mõnda mängukonsooli pulti. HID profiil defineerib ära protokollid, protseduurid ning käitumisreeglid sinihamba ühenduse abil kasutajaliidese seadmete kasutamiseks nagu nt klaviatuur ja hiir. Enim spetsifikatsioone on Nintendo Wii mängukonsooli pultide kohta, mistõttu kasutasin just seda pulti ühenduse loomise katsetamisel. Sellel konkreetsel puldil puudub liigutusi jälgiv andur (ing. k *motion sensor*), järelikult on testprogrammi toimimise tarbeks vaja luua sinihamba kaudu ühendus puldi ning iPad-i vahel ning seejärel juba puldi spetsifikatsioonides teadaoleva info abil puldi nupud ning iPad-i ekraanil toimuv vastavusse viia. Kuna Wii pult ei kasuta energiasäästlikku sinihammast ega ole ka MFI sertifitseeritud, tuleb leida alternatiiv Xcode energiasäästliku sinihamba raamistikule. Üks alternatiiv on välise aksessuaari raamistik (External Acessory Framework), mida kasutatakse ühenduse loomiseks sinihamba kaudu seadmetel, mis ei kasuta veel energiasäästlikku sinihammast. Kuid ka selle raamistiku kasutamiseks on tarvis, et seade oleks MFI sertifitseeritud.

Wii puldi jaoks on piiramata lähtekoodiga tarkvara leitav DarwiinRemote'i kodulehelt [12], kuid arendajate õnnetuseks uuenevad nii Apple'i tarkvara kui ka selle loomiseks vajalikud vahendid pidevalt. Selles rakenduses kasutatud raamistikud on privaatsed ning Apple on need tehnoloogia arenedes asendanud uutega.

Üks võimalus millegi analoogse loomiseks on kasutada sinihamba haldamise raamistikku (BluetoothManager Framework), mis on privaatne raamistik, mis tuleb tõsta õige programmiasukoha alla ning kuhu tuleb lisada veel mõned internetist vabalt kättesaadavad päisefailid (ing. k *header file*). Nii üritused katsetada seda raamistikku kui ka mõningaid väliseid rakendusi, nagu seda on WIIC [13], on tekitanud olukorra, kus Xcode emulaatoris kood tööle ei hakka. Veakoodi alusel on tegemist veaga, mis tuleneb sellest, et raamistikud on mõeldud kasutamiseks välistel füüsilistel seadmetel ja mitte simuleerimiseks Xcode keskkonnas. Probleem on aga selles, et programm ei hakka tööle ka iPad-is ning Xcode'ist koodi läkitamisel füüsilisse seadmesse pole võimalik koodi enam siluda (ing. k *debug*), mistõttu pole võimalik püüda kinni koodi täitmist ning tuvastada, kus tekib viga.

4.2.5. Btstack

Kuna Apple'i sinihammas on piiratud leidmaks vaid seadmeid, mis on MFI sertifitseeritud või teised uuemad Apple'i seadmed, on väliste seadmete sinihamba ühenduste loomisel võimalik kasutada Btstack'i, mille saab operatsioonisüsteemi piirangute eemaldamisel alla laadida Cydiast. Cydia on operatsioonisüsteemi piirangute eemaldamise tagajärjel tekkinud tarkvara, mis on alternatiiviks Apple Store'ile. [14] Btstack on laiendatud sinihamba pinu, mis on alternatiiviks seadme originaalsele sinihamba seadistusele. Selle kasutamine võimaldab leida üles mis tahes sinihammast kasutava seadme ka iPad-iga. Kuna konfiguratsioonis endas pole skaneerimise võimalust, laadisin alla Wii puldi emulaatori, mis ühendab seadme Wii puldiga ning seejärel kuvab ekraanil puldi pöörded ümber ühe telje. Kuna sellel puldil kiirendusandur puudub, siis seadme pöörlemist programm õigesti ei näidanud, kuid selge oli see, et ühendus seadmete vahel oli olemas. Probleem tekib aga sellega, et kui Btstack on lülitatud sisse, on iPad-i seadetes selle originaalne sinihammas välja lülitatud. Kuna liikurplatvorm on iPad-iga ühendatud selle originaalset sinihamba raamistikku kasutades, peab olema ligipääs sellele originaalsele sinihambale ning seetõttu pole Btstacki võimalik selles lahenduses kasutada. Alternatiive pakub ka Blutrol, mis praegusel hetkel ei ole aga kasutatava operatsioonisüsteemi jaoks veel väljas, mistõttu polnud võimalik seda katsetada. Tõenäoliselt toimib see samamoodi nagu Btstack ning piirab juurdepääsu platvormi liigutamise jaoks vajalikule originaalsele sinihamba raamistikule.

4.2.6. Mänguemulaatorite kasutamine

Pärast operatsioonisüsteemi piirangute eemaldamist on võimalik Cydiast alla laadida mänguemulaatoreid, millest mõned peaksid teoreetiliselt lubama emulaatoril seadme originaalse sinihamba kaudu Wii puldiga ühendust saada. [15], [16] Kuna mängude välise puldiga toimimiseks allalaetavad konfiguratsioonid ei ole legaalsel teel kättesaadavad, siis selle vettpidavust ma kontrollima ei hakanud. Kui ka iPad tõepoolest leiaks sinihamba kaudu ühenduse Wii puldiga, tekiks probleem enda kirjutatud rakenduse kaugjuhtimisega. Nimelt töötab Wii pult emulaatoril just Nintendo mängudega, mille toimimiseks oleks tarvis mängude mälutõmmiseid ehk ROMe. Kuna minu kirjutatav aplikatsioon ei oma seost Nintendo mänguga, oleks selle rakendus läbi emulaatorisse ühendatud Wii pult kasutu.

4.2.7. Android seadmega juhtimine

Uuemad Android seadmed kasutavad samuti energiasäästlikku sinihammast, mistõttu õnnestus iPad-i sinihamba ümberkaudsete seadmete skännimisel üles leida Sony Xperia telefon, mis kasutab Androidi tarkvara. Samuti õnnestus seadmete paari panemine, kuid proovides seadmeid ühendada, kadus ühendus murdosa sekundi jooksul. Konflikt tekib selles, et mõlemad seadmed on konfigureeritud täitma sinihamba ühenduses tuumikurolli, mistõttu ei saa need omavahel ühenduda. Nimelt otsib tuumikrolli täitev seade ootel olevaid andmeid väljareklaamivaid seadmeid, et siis ennast nendega ühendada. Seda ilmselt seetõttu, et autentimise sammus tekib probleem, sest Android seadmetel puudub MFI autentimismooduli riistvara, mistõttu seadmed omavahel ühendust ei saa.

4.2.8. Lähtekoodi muutmine

Üheks üsna lihtsaks viisiks kontrollida platvormi ajaliselt võrdsete pikkustega impulssidega oleks muuta praeguse kasutajasisendina kasutatava Chrome'i veebilehe lähtekoodi. Seda võiks teha näiteks nii, et klikkida Google Chrome'i kohaldamine ja juhtimine ->Tööriistad -> Arendaja tööriistad. See kuvab lehekülje alla serva akna lehekülje lähtekoodiga, mille muutmisel kajastab Chrome'i aken need samaaegselt. Näitlikkuse mõttes on võimalik koheselt näha muutust lehekülje taustavärvi, šrifti jms muutmisel. Kui kõik soovitud muudatused on tehtud, saab kopeerida kogu lähtekoodi hüperteksti märgistuskeele ehk HTMLina ning seejärel selle serverisse üles laadida.

Kuna drive.doublerobotics.com on väga pika lähtekoodiga lehekülg, on kõige lihtsam muuta vaid seda osa lähtekoodist, mis saab kasutajalt sisendit liikumise osas. Näiteks võib kirjutada konkreetselt edasi-tagasi, paremale-vasakule liikumist klaviatuurilt kontrollitud koodijuppide vahele meile vajaliku ajaga taimereid või tõsta liikumiskäsust aktiveeritavad tsüklid loendurega töötab kuni (ing. k *while*) tsükli sisse. Keerulisem oleks kasutaja sisendit liikumise pikkusele arvestava nupu tekitamine lehele selle praeguse lähtekoodi keerukuse tõttu. Testprogrammi lihtsuse mõttes võib vabalt olla tegu konstantse väärtusega, kuid katsetuse käigus selguvate teistsuguste vajaduste korral oleks muudatuste tegemine programmis tülikas ning aeganõudev.

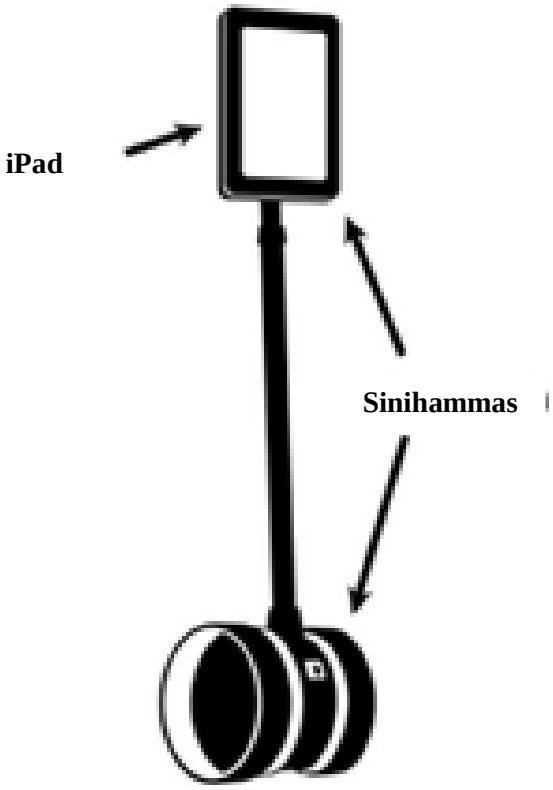
Pealegi oleks selline väike muudatus praeguses terviklikult toimivas süsteemis liialt väike, sest optimaalsem võiks olla kasutada lahendust, mille tarbeks pole vaja luua juurde ühte serverit. Lisaks internetiühenduse vajadusele tuleks sellise lähenemise korral endiselt kindlustada sinihamba ühendus iPad-i ning liikurplatvormi vahel, lisaks iPad-i ühendus internetiga ning seejärel tuleks logida sisse Double Robotics IDga, enne kui üldse on võimalik seadet testima hakata.

4.2.9. Lokaalne rakendus iPad-is

Olles uurinud erinevaid võimalusi, tundub, et optimaalseks variandiks oleks kirjutada iPad-i jaoks lokaalne rakendus. Kui juhtimine toimub otse iPad-ist, puudub vajadus lisaks liikurplatvormile veel kolmanda seadme kasutamiseks. Arvestades, et Double Robotics'il on olemas iPad-ide ning *double*'i platvormi arendamiseks avalik tarkvaraarenduskomplekt, on võimalus luua testaplikatsioon, mis peale sinihamba ühenduse ühegi teise ühenduse olemasolu ei vaja (sele 4.1).

Double'i tarkvaraarenduskomplekt on arendajale suunatud juhtprogrammi raamistike ja iOS rakenduste raamide kogum, kus on ära defineeritud erinevad meetodid platvormi liigutamiseks. Nende kasutamine on aga iga arendaja enda teha. Selleks, et koodi kasutada mõne teise platvormiga, mis pole iOS, on koodi võtmekohad privaatsed, mistõttu mõne teise seadme abil otsene liikurplatvormi juhtimine selle tarkvaraarenduskomplekti abil võimalik ei ole.

Rakendust luues tuleb pidada silmas, et oluline on katsete korratavus, mis tähendab seda, et erinevad liikumised peavad toimuma ajaliselt ühepikkuste perioodide vältel. Samuti on oluline, et kui programm käivitub otse iPad-i ekraanilt, jõuaks liikurplatvorm pärast kasutajasisendi saamist iPad-i ekraanil ennast taas stabiliseerida.



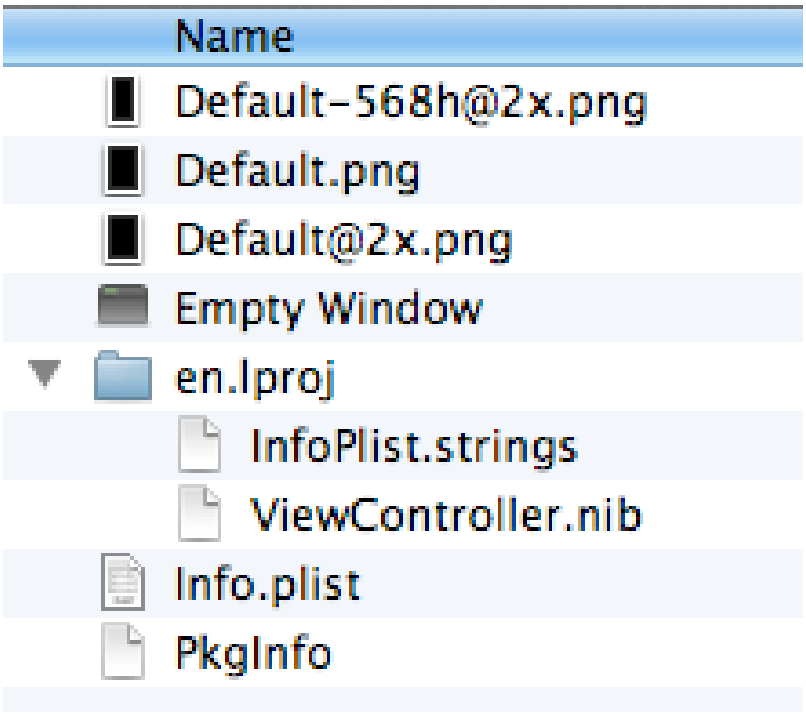
Sele 4.1. Andmeside skeem liikurplatvormi juhtimiseks lokaalse rakenduse teel

Xcode

Xcode on integreeritud arenduskeskkond, mis sisaldab Apple'i poolt loodud OSX ja iOS tarkvara arendamise vahendeid. Apple'i versioonide Mac OS X Lion ja OS X Mountain Lion kasutajate jaoks on see riistvara tasuta. Programmi versioon Xcode 5 sisaldab ka iOS 7.0.x takrvaraarenduskomplekti, mis tähendab seda, et uue tühja projekti loomisel laetakse Xcode rakenduse alamkaustadest vastava konfiguratsiooniga šabloon. Seda sisaldav kaust kopeeritakse ning vastavalt projekti loomisel tehtud valikutele väärtustatakse mõned muutujad.

Uue projekti loomisel Xcode keskkonnas loodavad failitüübid

Uue tühja projekti (nimega „Empty Window") andmepuus olevad failid on näidatud selel 4.2.



Sele 4.2. Uue projekti loomisel andmepuus olevad failid

Default.png ning kaks sarnase nimega .png faili – kolm käivitatavat pildifaili on kopeeritud andmepuu ülemisele tasemele, kust nad on käivitamisel kuvamiseks leitavad.

Empty Window – tühja rakenduse kompileeritud kood (binaarkood). Kui aplikatsioon käivitatakse, seotakse binaarkood mitme raamistikuga ning kood käivitub sisenedes enda alamfailides main peafunktsiooni (ing. k *main*).

Info.plist – konfiguratsioonifail rangelt teksti formaadis. See on tuletatud projektifaili Empty Window Info.plist failist. See sisaldab instruktsioone süsteemile selle kohta, kuidas rakendust käitada ning käsitleda. Näiteks kui aplikatsioonil on ikoon, siis Info.plist ütleb süsteemile selle nime, et süsteem leiaks ning kuvaks selle andmepuust. Samuti sisaldab see andmeid binaarkoodi kohta, et süsteem õigesti rakenduse leiaks ning avaks.

PkgInfo – tekstifail, mis loeb rakenduse tüüpi ning looja koodi. See fail pole programmi toimimiseks hädavajalik ning genereeritakse automaatselt. Kasutajana pole tarvis seda faili kunagi muuta.

InfoPlist.strings – tekstifail, mis on vajalik Info.plist failis teksti ilmumiseks, mida on vaja teistesse keeltesse tõlkida. See on otse InfoPlist.strings failist kopeeritud. Seda faili pole vaja muuta, kui projekti jaoks sobib vaikimisi inglise keel.

ViewController.nib – rakenduse ainus .nib fail, mis sisaldab instruktsioone aplikatsiooni peaakna esialgse sisu genereerimiseks. See on kompileeritud ViewController.xib failist projektis, kusjuures .xib ja .nib laiendiga failid on erinevad vormid samast asjast. Need failid sisaldavad osa kasutajaliidesest.

Reaalsuses on projektipuus rohkem faile, kuid need erinevad pigem arvult kui tüübilt. [17] Samuti luuakse uue projekti käivitamisel projekti navigaatori asukohta kolme tüüpi failid, mis kannavad .xib, .m ja .h laiendeid. Kui klikata .xib laiendiga faili peale, siis kuvab Xcode automaatselt kasutajaliidese ehitamise aknale, kus on võimalik menüüst objekte graafiliselt kasutajaliidesele lohistada. Nendes failides salvestatakse rakenduse kasutajaliidese seaded. .h ja .m failid on objektorienteeritud C programmeerimiskeeles kasutatavad failid, kus .h fail viitab päisefailile, kus toimub koodi liidestamine ning .m fail on fail, kus täidetakse kood ning liidesed. Näiteks deklareeritakse päisefailis ära klass ning selle meetod, teadmata, kuidas see töötab, pärast päisefaili .m faili kaasamist on võimalik vastavas päisefailis loodud liides .m failis ellu viia. [18]

Objektorienteeritud C

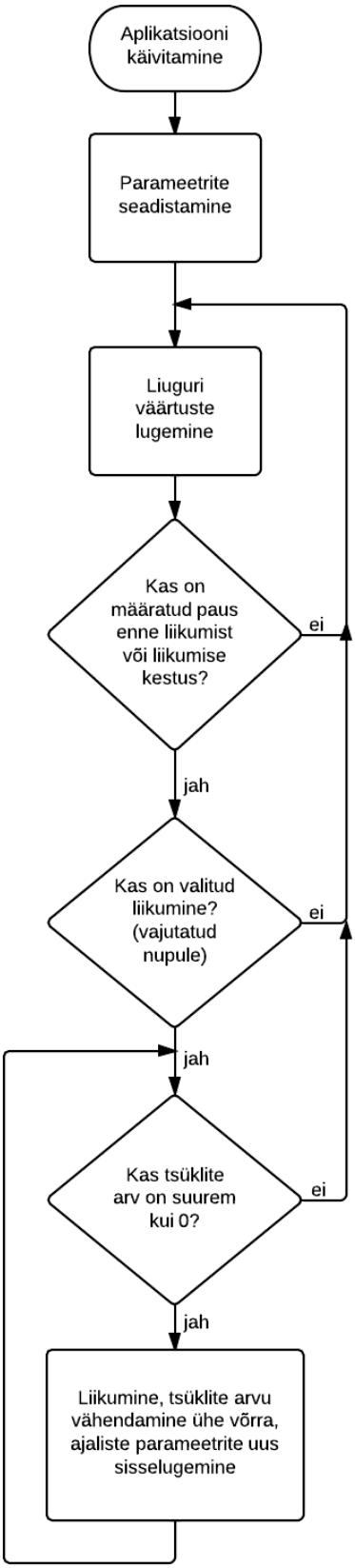
Objektorienteeritud C on põhiline programmeerimiskeel OSX ja iOS tarkvara kirjutamisel. Objektorienteeritud C programmeerimiskeel põhineb C keelel, mis on võrreldes C++ keelega riistvajalähedasem. Sellises protseduurilises keeles on fookuses probleemilahendus samm-sammult meetmetega, mida on võimalik taaskasutada ning mida kutsutakse funktsioonideks. Objektorienteeritud C keel säilitab kõik C keele aspektid, lisaks on lisatud objektorienteeritud C keelde süntaks ning semantika, mis on tarvilik objektorienteerituseks. See erineb protseduurilisest programmeerimisest seetõttu, et rõhk on andmete ning klassides olevate andmete manipuleerimiseks kasutatavate meetodite grupeerimisel. Objektorienteeritud disainil on eeliseid suurte probleemide lahendamisel, sest võimaldab jagada probleemi lihtsasti väiksemateks alamprobleemideks ning mooduliteks, mida on võimalik teistest sõltumatult testida. Teisalt on ka lihtsam enda või teiste mooduleid taaskasutada. [19]

4.3. Testprogrammi lähtekoodi kirjeldus

Kasutades *double*'i avalikku tarkvaraarenduskomplekti HelloWorld projekti näitel ning tundes Xcode'is uue projekti loomisel tekkivaid faile ning nende tüüpe, oskan navigeerida erinevates failides ning enda soovitud rakendusekoodi õigesse kohta kirjutada.

Testprogrammi põhi on leitav lisas 3, kus esialgne päisefaili kood on L3.1 ning esialgne implementeerimisfaili kood L3.2.

Kasutajaliidese .xib failis jätsin alles kõik nupud ning sildid, muutsin vaid nende asukohta, et kuvatav info oleks testprogrammile loogiliselt grupeeritud. Lisasin vaatele vaid kolm liugurit ning nendele vastavad sildid (lisa 2). Koodis on minu poolt lisatud muudatused kuvatud rasvases kirjas. Rakenduse lihtsustatud algoritm on toodud selel 4.3. Lisaks sellele põhimõttelisele skeemile kuvatakse reaalajas ka infot nt akukasutuse ja ühendusstaatuse kohta, samuti on võimalik muuta seadme kõrgust ning eemaldada või tagastada seisujalg otse aplikatsioonist. Oluline on teada, et liuguritel valitud tegevuste kestused ei pruugi olla vastavuses kuvatava ajaga millisekundi täpsusega, sest kogu programmi koodi tsükli pikkus on sisse kirjutatud Double Robotics'i lähtekoodi, millest minul on rakenduse loomiseks kasutada vaid osa. See aga ei oma selle testprogrammi loomisel suurt tähtsust, sest oluline osa, milleks on tegevuste korratavus tänu tsüklite võrdsetele pikkustele, on tagatud.



Sele 4.3. Rakenduse lihtsustatud algoritm

4.3.1. Päisefaili kood

```
1  #import <UIKit/UIKit.h>
2  #include <QuartzCore/QuartzCore.h>
3  @interface DRViewController : UIViewController
4  {
5        IBOutlet UILabel *statusLabel;
6        IBOutlet UILabel *poleHeightPercentLabel;
7        IBOutlet UILabel *kickstandStateLabel;
8        IBOutlet UILabel *batteryPercentLabel;
9        IBOutlet UILabel *batteryIsFullyChargedLabel;
10       IBOutlet UILabel *firmwareVersionLabel;
11       IBOutlet UIButton *driveForwardButton;
12       IBOutlet UIButton *driveBackwardButton;
13       IBOutlet UIButton *driveLeftButton;
14       IBOutlet UIButton *driveRightButton;
15       IBOutlet UILabel *ActionLabel;
16       IBOutlet UISlider *TimeUntilStart;
17       IBOutlet UISlider *TimeForAction;
18       IBOutlet UITextField *Starttext;
19       IBOutlet UITextField *Actiontext;
20       IBOutlet UITextField *cyclenumber;
21       IBOutlet UISlider *cycleslider;
22 }
```

Tegu on ainsa päisefailiga, mida ma muutsin selleks, et muuta testprogrammi, sest seda faili on HelloWorld projektis kasutatud kasutajaliidese objektide defineerimiseks. Xcode 5 vajab liugurite töötamiseks QuartzCore raamistikku, mis on sinna ka reaga 2 lisatud. Reas 5 luuakse tüübile UILabel viidaga statusLabel mälu asukoht, analoogselt toimub see kõikides ridades 5–21. UId kasutatakse kasutajaliideses kuvatavate elementide defineerimiseks, IBOutlet seostab omadused rakenduses komponentidega. UILabel on kasutajaliideses kasutatav silt, UISlider liugur, UIButton nupp ning UITextField tekstiväli.

```
23 @property(nonatomic, assign)bool IsValid;
24 @property(nonatomic,assign) NSString* ButtonPushed;
25 @property(nonatomic,assign) int counter;
26 @property (nonatomic, assign) double EndTime;
27 @property(nonatomic,assign) double sleepTime;
28 @property(nonatomic,retain) IBOutlet UISlider *cycleslider;
29 @property (nonatomic, retain)IBOutlet UITextField
   *cyclenumber;
30 @property (nonatomic, retain) IBOutlet UISlider
   *TimeUntilStart;
31 @property(nonatomic, retain) IBOutlet UISlider *
   TimeForAction;
```

```
32  @property (nonatomic, retain)IBOutlet UITextField
   *Starttext;
33  @property (nonatomic, retain) IBOutlet UITextField
   *Actiontext;
34 -(IBAction)UntilStartValueChanged:(id)sender;
35 -(IBAction)ActionSliderValueChanged:(id)sender;
36 -(IBAction)cyclechanged:(id)sender;
37  @end
```

Ridades 23–33 tähendavad *property*'d muutujaid. Seades muutuja deklareerimisel *nonatomic* atribuudi, on võimalik uutel loodavatel lõimedel seda muutujat kasutada. See loob eelduse kasutada mitut paralleelselt jooksvat lõime. Ridades 23–27 tähendab *assign*, et selle muutujaga ei tehta mälutoiminguid. Ridades 28–33 tähendab *retain* seda, et lubatakse muutujale sisse lugeda uus väärtus, vana väärtus kustutatakse. Need muutujad on sellised, mille väärtused on selle programmi puhul kasutajaliideses muudetavad.

Reas 23 on *bool* tüüp, mille väärtus saab olla seatud kas tõeseks või vääraks. Reas 24 kasutatav NSString on tüüp, mille sees saab hoida tähemärkide jada ehk on võimalik määrata tüübile vaste sõna kujul. Reas 25 on *int* täisarvulise väärtusega tüüp. Muutujat *counter* kasutame selleks, et rakendada programmis täisarvulise tsüklite liuguri väärtust. Ridades 26 ja 27 kasutan *double*'it, mis on komakohaga arvu väärtusega tüüp. Seda kasutame muutujates EndTime ning sleepTime, sest teostame aritmeetilisi tehteid nende muutujate ja CACurrentMediaTime meetodiga, mis on samuti vaikimisi millisekundites ning kuvatav reaalarvuna.

Ridades 34–36 seostatakse IBAction meetoditega, mis ei tagasta mitte midagi ja mis seotakse kasutajaliidese toimingutega. „-„ tähistab seda, et tegu on meetodi, mitte muutujaga. Nende meetodite sisend on id tüüpi, mis kannab endaga kaasas kogu id-klassi informatsiooni, *sender* on seal selle sisendmuutuja nimi.

Terviklikult kujul on loodud päisefaili kood välja toodud lisas 3 L3.3.

4.3.2. Implementeerimisfaili kood

```
1  #import "DRViewController.h"
2  #import <DoubleControlSDK/DoubleControlSDK.h>
3  @interface DRViewController () <DRDoubleDelegate>
4  @end
```

Reas 3 luuakse klassi DRDoubleDelegate laiendus DRViewControlleri loomiseks.

```
5  @implementation DRViewController
6  @synthesize cyclenumber, cycleslider, Actiontext, Starttext,
   TimeForAction, TimeUntilStart;
```

Rida 5 evitab loodud kontrolleri, reas 6 toimub objektorienteeritud C keele kompaileri omaduste autosüntees.

```
7  -(void)viewDidLoad
   {
8          [super viewDidLoad];
9          [DRDouble sharedDouble].delegate = self;
10         _IsValid = NO;
    }
```

See lõik käivitab programmi vaate, reas 8 kutsutakse välja klass, mis sisaldab viewDidLoad konfiguratsiooni, rida 9 saadab teateid *double*'i kohta programmile. Reas 10 taasväärtustatakse liikumise kestuse määrav muutuja.

```
11 -(BOOL)shouldAutorotateToInterfaceOrientation:
   (UIInterfaceOrientation)toInterfaceOrientation
12 {
          Return
          UIInterfaceOrientationIsPortrait
          (toInterfaceOrientation);
   }
```

See lõik määrab ära, millises iPad-i asendis programm töötab. Rida 12 määrab ära selle, et programm töötab vaid iPad-i püstises asendis.

```
13 #pragma mark - Actions
14 -(IBAction)poleUp:(id)sender
   {
          [[DRDouble sharedDouble] poleUp];
   }
15 -(IBAction)poleStop:(id)sender
   {
          [[DRDouble sharedDouble] poleStop];
   }
16 -(IBAction)poleDown:(id)sender
   {
          [[DRDouble sharedDouble] poleDown];
   }
17 -(IBAction)kickstandsRetract:(id)sender
```

```
   {
         [[DRDouble sharedDouble] retractKickstands];
   }
18 -(IBAction)kickstandsDeploy:(id)sender
   {
         [[DRDouble sharedDouble] deployKickstands];
   }
```

Selles lõigus toimub kasutajaliidese nuppudele käskude määramine, mis on id tüüpi meetodite deklaratsioon, mis vastavalt sisendmuutuja väärtusele meetodile edastatakse. Siin tähendab meetod poleUp väärtuse saamist ning sellele reageerimist *double*'i kõrguse muutmist suuremaks, poleStop kõrguse muutmise seiskamist, poleDown kõrguse vähendamist, retractKickstands seisujala eemaldamist ning deployKickstands seisujala rakendamist.

```
19 #pragma mark - DRDoubleDelegate
20 -(void)doubleDidConnect:(DRDouble *)theDouble
   {
21       statusLabel.text = @"Connected";
   }
22 -(void)doubleDidDisconnect:(DRDouble *)theDouble
   {
23       statusLabel.text = @"Not Connected";
   }
24 -(void)doubleStatusDidUpdate:(DRDouble *)theDouble
   {
25       poleHeightPercentLabel.text = [NSString
         stringWithFormat:@"%f",[DRDouble
         sharedDouble].poleHeightPercent];
26       kickstandStateLabel.text = [NSString
         stringWithFormat:@"%d",
         [DRDouble sharedDouble].kickstandState];
27       batteryPercentLabel.text = [NSString
         stringWithFormat:@"%f",
         [DRDouble sharedDouble].batteryPercent];
28       batteryIsFullyChargedLabel.text = [NSString
         stringWithFormat:@"%d",
         [DRDouble sharedDouble].batteryIsFullyCharged];
29       firmwareVersionLabel.text = [DRDouble
         sharedDouble].firmwareVersion;
   }
```

Selles lõigus toimub kasutajaliideses parameetrite oleku kuvamine, iga meetodi jaoks luuakse viidaga uus mälu asukoht.

```
30 -(void)doubleDriveShouldUpdate:(DRDouble *)theDouble
   {
```

```
31      float drive;
32      float turn;
33      if (driveForwardButton.highlighted ||
        driveBackwardButton.highlighted||
        driveRightButton.highlighted
        ||driveLeftButton.highlighted)
        {
34            _counter = cycleslider.value;
35            if (driveForwardButton.highlighted)
              {
36                  _ButtonPushed = @"Forward";
              }
37            else if (driveBackwardButton.highlighted)
              {
38                   _ButtonPushed = @"backward";
              }
39            else if (driveRightButton.highlighted)
              {
40                  _ButtonPushed = @"Turn Right";
              }
41            else if (driveLeftButton.highlighted)
              {
42                  _ButtonPushed = @"Turn Left";
              }
43            _EndTime = (CACurrentMediaTime()+
              TimeUntilStart.value) +
              TimeForAction.value;
44             _sleepTime = CACurrentMediaTime() +
               TimeUntilStart.value;
        }
```

Selles lõigus deklareeritakse ja väärtustatakse liikumise alustamiseks muutujad. Real 33 algav „if“ lause sisaldab endas tingimust, et kui mõnele nupule on vajutatud, siis muutujasse *counter* loetakse real 34 sisse tsüklitearvu liuguri väärtus ning vastavalt sellele, kas on vajutatud nuppu *forward, backward, turn right* või *turn left,* loetakse muutujasse ButtonPushed ridadel 35–42 sisse tähemärgiline muutuja. Samuti toimub tsükli pikkuse ja ooteaja pikkuse määramine, kus tsükli lõppemise aja EndTime saamiseks liidetakse reas 43 meetodile, mis tagastab hetke kellaaja arvutist, väärtused liuguritelt. sleepTime on ooteaja pikkuse väärtustamine, kus liidetakse hetkekellaajale arvutist juurde liuguri väärtus, milles on määratud ooteaeg enne liikumise alustamist.

Tsüklilisus on tekitatud kasutades aritmeetikat ajaliste parameetrite vahel seetõttu, et selleks üldkasutatav „for“-tsükkel siinkohal ei toimi soovikohaselt. Seda seetõttu, et tervikprogrammis, millele see tarkvaraarenduskomplekt ligipääsu ei taga, on ära määratud

programmi tsüklite kestus ning uue tsükli alustamise kord, olenemata pausidest selles koodiosas.

```
45      double currentTime = CACurrentMediaTime();
46      if (currentTime >= _EndTime)
        {
47            if (_counter > 1)
              {
48                  _counter = _counter -1;
49                  _EndTime = (CACurrentMediaTime()+
                    TimeUntilStart.value) +
                     TimeForAction.value;
50                  _sleepTime = CACurrentMediaTime() +
                    TimeUntilStart.value;
              }
        }
```

Reas 45 luuakse muutuja *currenttime*, mis pannakse võrduma arvuti hetkekellaajaga. „Kui"-lausest lähtuvalt ridades 46–50 vaadatakse, et kui hetkekellaaeg on väiksem või võrdne tsükli lõppemise ajaga ning kui tehtavate tsüklite arv on suurem kui üks, siis vähendatakse veel tehtavate tsüklite arvu ühe võrra ning taasväärtustatakse tsükli lõppemise ning ooteaja lõppemise väärtused samamoodi, nagu seda tehti programmi esimeses tsüklis.

Real 51 toimub liikumise kestust määrava muutuja määramine tõeseks vahemikus ooteaja võrdseks saamisest hetkekellaajaga kuni tsükli lõpuaja võrdseks saamiseni hetkekellaajaga.

```
51      _IsValid = currentTime <= _EndTime && currentTime
        >=_sleepTime;
52      if (_IsValid && [_ButtonPushed  isEqual: @"Forward"])
        {
53            drive = kDRDriveDirectionForward;
54            ActionLabel.text =  @"forward";
        }
55      else if ((_IsValid && [_ButtonPushed  isEqual:
        @"backward"]))
        {
56            drive = kDRDriveDirectionBackward;
57            ActionLabel.text = @"backward";
        }
58      else
        {
59             drive =kDRDriveDirectionStop;
60             ActionLabel.text = @"Stop";
```

```
61              turn= 0.0;
        }
62      if ( _IsValid && [_ButtonPushed  isEqual: @"Turn
        Right"])
        {
63              turn= 1.0;
64              ActionLabel.text =  @"Turn Right";
        }
65      else if ( _IsValid && [_ButtonPushed  isEqual: @"Turn
        Left"])
        {
66              turn= -1.0;
67              ActionLabel.text =  @"Turn Left";
        }
68      [theDouble drive:drive turn:turn];
  }
```

Ridades 52–67 toimub liikumine ning liikumise viisi kuvamine liidese sildil juhul, kui liikumise kestuse määrav muutuja on tõene ning kui muutujas *buttonpushed* sisaldub mõni eelpool defineeritud tähemärgiline muutuja. Reas 68 võetakse põhiraamistikus muutujad vastu ning käivitatakse liikumine.

```
69 -(IBAction)UntilStartValueChanged:(UISlider*)sender
  {
70      Starttext.text = [NSString
        stringWithFormat:@"%.0f",[sender value]];
71      NSString * Starttextvalue =[ Starttext text];
72      float value= [Starttextvalue floatValue];
73      if (value < 0)value=0;
74      if (value > 20)value=20;
75      TimeUntilStart.value=value;
76      Starttext.text= [NSString
        stringWithFormat:@"%.0f",value];
77      if ([Starttext canResignFirstResponder])[Starttext
        resignFirstResponder];
  }
```

Ridades 69–95 toimub liugurite initsialiseerimine. Reas 70 määratakse täisarvulise väärtusega number, mis kuvatakse tekstina sildil ning reas 72 omistatakse see väärtus andmetüübile *float,* mis võib olla nii *int* kui ka *double* selleks, et seda oleks võimalik võrrelda liugurile määratud piirväärtustega, et kuvatav number jääks etteantud vahemikku kasutajaliideses. Selle liuguri valimisel tehakse rea 77 abil see liugur aktiivseks, kuni mõne teise elemendi aktiveerimiseni, mille puhul jääb selle liuguri väärtuseks viimane seadistatud väärtus.

```
78 -(IBAction)ActionSliderValueChanged:(UISlider *)sender
```

```
  {
79       Actiontext.text = [NSString
         stringWithFormat:@"%.1f",[sender value]];
80       NSString * Actiontextvalue =[ Actiontext text];
81       float value= [Actiontextvalue floatValue];
82       if (value < 0)value=0;
83       if (value > 5)value=5;
84       TimeForAction.value=value;
85       Actiontext.text= [NSString
         stringWithFormat:@"%.1f",value];
86       if ([Actiontext canResignFirstResponder])[Actiontext
         resignFirstResponder];
  }
87 -(IBAction)cyclechanged:(UISlider *)sender
  {
88       cyclenumber.text = [NSString
         stringWithFormat:@"%.0f",[sender value]];
89       NSString * cyclenumbervalue =[ cyclenumber text];
90       float value= [cyclenumbervalue floatValue];
91       if (value < 1)value=1;
92       if (value > 25)value=25;
93       cycleslider.value=value;
94       cyclenumber.text= [NSString
         stringWithFormat:@"%.0f",value];
95       if ([cyclenumber canResignFirstResponder])[cyclenumber
         resignFirstResponder];
  }
```

Read 78–95 täidavad sama koodi teise ning kolmanda liuguri jaoks esimese näitel.

```
96   -(void) touchesBegan:(NSSet *)touches withEvent:(UIEvent
       *)event
     {
97         if (Starttext)
           {
98               if ([Starttext canResignFirstResponder])
                 [Starttext resignFirstResponder];
           }
99         [super touchesBegan:touches withEvent:event];
100       if (Actiontext)
          {
101           if ([Actiontext canResignFirstResponder])
              [Actiontext resignFirstResponder];
           }
102      [super touchesBegan:touches withEvent:event];
103      if (cyclenumber)
           {
104            if ([cyclenumber canResignFirstResponder])
               [cyclenumber resignFirstResponder];
           }
```

```
105     [super touchesBegan:touches withEvent:event];
106 }
107 @end
```

Ridades 96–105 toimub liugurite väärtuste muutmiseks puudutuste tuvastamine. Terviklikult kujul on loodud implementeerimisfaili kood välja toodud lisas 3 L3.4.

5. ROBOTI LIIKUMISTE ANALÜÜS ERINEVATEL PINDADEL

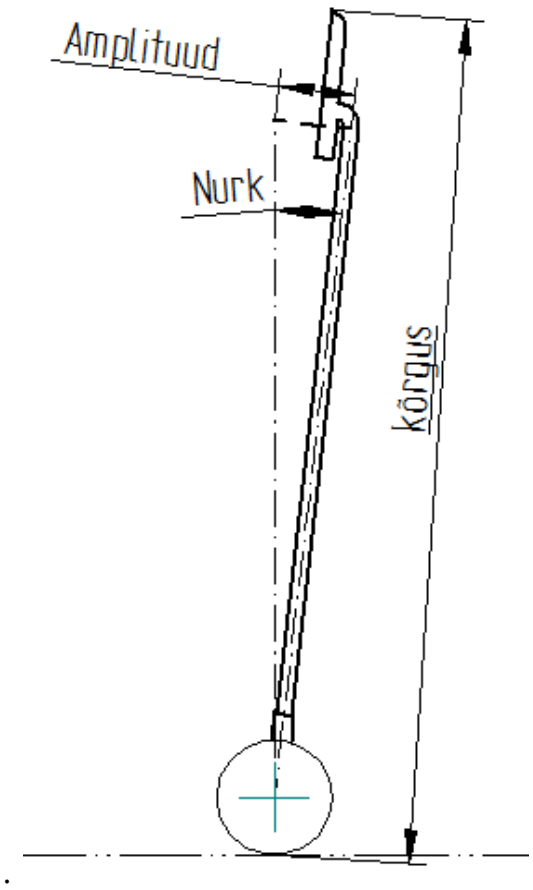
Näidistestidena viisin läbi seitse erinevat testi. Testide jaoks horisontaalpinnal kasutasin tsükliaega kaks sekundit, kaldpinnal kolm sekundit ning korduste arvuna kolme tsüklit, et ühtlaste tulemuste jaoks samades tingimustes ühtlast materjali saada. Kuna käesoleva töö raames ei toimu platvormi liikumise käitumise põhjalikku testimist, siis valisin kõikidest erinevatest testidest välja ühe, mida selles peatükis kirjeldan.

Katseandmete ülesvõtmiseks kasutasin kaamerat, millest renderdasin materjali pildikujule kaadrisagedusega 25 kaadrit sekundis mõõtkavaga 1:6 tegelikkuse suhtes. Kasutatud kaameraks on Sony handycam HDR-XR500, mille sensori, läätse ja säristuse parameetrid on välja toodud lisas 5. Katseteks kasutatud seaded on näidatud tabelis 5.1.

Tabel 5.1. Katses kasutatud kaamera seadistus

Kaadrisagedus	25 kaadrit sekundis
Fookus	automaatne
Automaatne säritus	väljas
Valge tasakaalu nihe	0
Resolutsioon	6,6 MP

Esmalt veendusin selles, et saadud piltidel kujutatul oleks maapind pildi suhtes täielikult horisontaalne. Seejärel otsisin välja kaadri, kus robot on ennast stabiliseerinud, mis on vahetult enne kaadreid, kus toimub juba roboti liikumine vastavalt etteantud käsklustele. Lisaks sellele kaadrile analüüsisin ka iga kuuendat kaadrit, millest tuleneb katseandmete võtmise sagedus, milleks on 0,25 sekundit nelja sekundi jooksul horisontaalpinnal ning viie sekundi jooksul kaldpinnal. Kokku analüüsisin iga katse juures horisontaalpinnal seitsetteistkümmet kaadrit ning kaldpinnal kahtekümmend üht kaadrit. Kasutades Solid Edge joonestusosa, kandsin igale kaadrile piki roboti telge sirge ning samast punktist pendli ühenduskohas ratastega vertikaalse sirge. Selliselt leidsin nende sirgete vahelise nurga, mis on pendli nurk vertikaalsest tasakaaluasendist ning sirgetevahelise kauguse, mis on nihke amplituud tasakaaluasendist (sele 5.1). Tegeliku amplituudi saamiseks korrutasin jooniselt leitud väärtuse kuuega, mis annab meile väärtuse tegelikus mõõtkavas.



Sele 5.1. Skeem nurga ning amplituudi leidmiseks

Liikumise jooksul kogutud nurga ning amplituudi väärtused koondasin tabelitesse, mille põhjal koostasin MS Excelis graafikud, mis kirjeldavad roboti nurga ning amplituudi muutumist kogu liikumise jooksul. Liikumise alguseks on võetud hetk enne liikumise algust, kus robot on stabiilses asendis. Graafikul on kirjeldatud nurgad ning amplituud kogu liikumise ulatuses, kus viimasel sekundil toimub taas roboti asendi stabiliseerimine, kui liikumine on juba lõppenud.

Näidistestidena tehtud seitse erinevat katset on roboti liikumine selle madalas asendis, roboti liikumine kõrges asendis põrandal edasi, põrandal tagasi, vaibal ning kaldpindadel kalletega 2°, 5° ning 7°.

5.1. Analüüsimeetodi eripärad

Sellisel meetodil kaamerapildilt nurga ning amplituudi mõõtmisel tuleb arvesse võtta kaamera läätsest tingitud võimalikke moonutusi, perspektiivist tingitud moonutusi ning samuti pildi

eraldusvõimet. Kui pilt on madala resolutsiooniga, siis on sellel vähem piksleid ning sellel kujutatu olulised osad, nagu seadme telg, hajuvad taustaga ühte. Mida halvem on eraldusvõime, seda suurem viga võib tekkida joone positsioneerimisel pildile.

Kaamera läätsest tingitud suurim tõenäoline optiline moonutus on tünnmoonutus. Tünnmoonutus on eelkõige seotud lainurk läätsega, mis muudab pildi sfääriliseks, nõnda et pildi nurgad tunduvad inimsilmale kõverad. [20]

Perspektiivist tingitud moonutus tähendab seda, et objektid, mis on kaamerale lähemal, tunduvad suuremad kui objektid, mis on kaamerast kaugemal. Näiteks kui liikurplatvorm eemaldub kaamerast katse ajal, muutub ka mõõtkava ning tegelik hälve vertikaalsest asendist on suurem, kui pildi pealt tundub. [21]

Kuna näidistestide eesmärk on eelkõige testprogrammi illustreerimine ning töö ülesanne on platvormi üldine kirjeldamine ning kaardistamine, siis ei ole siinkohal andmete täpsus sedavõrd oluline. Tünnmoonutuse vähetähtsuse kontrollimiseks tegin paralleelsete markerite vaatlemise põhjal kaadri erinevates kohtades kindlaks, et moonutus võrreldes üksnes sirgete joonestamisest tulenenud määramatusega on väike. Samuti kontrollisin, et seade liiguks katse ulatuses kaameraga risti sellest eemaldumata või sellele lähenemata.

Testide tegemisel täpsete katsetulemuste saamiseks tuleb silmas pidada seda, et kaamera eraldusvõime oleks piisavalt hea, et pildi kvaliteet võimaldaks sirgeid piltidele kanda väiksema mõõtemääramatusega. Tünnmoonutuste likvideerimiseks on moodsates fototöötlusprogrammides olemas spetsiaalsed tööriistad. Näiteks Photoshop'is on olemas parandusfilter nimega läätse moonutused (ing. k *lens distortion*). Samuti on alternatiivsed tööriistad vabavarana saadaolevates fototöötlusprogrammides. Perspektiivmoonutus ei ole antud testide puhul väga suureks probleemiks, kuna vaatluse all on vaid üks objekt, mille omadused ei sõltu sellest, millises perspektiivis see taustaga asub. Kui mõõtkava referentsiks on mõni teine objekt, tuleks see asetada seadmega samasse tasapinda selle lähedale, et mõõtkava võimalikult täpselt üle kanda. Samuti tuleks veenduda, et kaamera on vaadeldava seadmega ning selle liikumisega risti, et tagada mõõtkava säilimine katse ulatuses.

5.2. Roboti liikumine edasi

Roboti liikumisel edasisuunas tegin katsed selle kõrges ning madalas asendis. Madalas asendis tehtud katse sai tehtud selleks, et kinnitada ka katseliselt tõsiasja, et madala raskuskeskmega on pendelmehhanism stabiilsem kui kõrgema raskuskeskmega mehhanism. Seetõttu on ka kõik ülejäänud katsed tehtud robotiga selle kõige kõrgemas asendis.

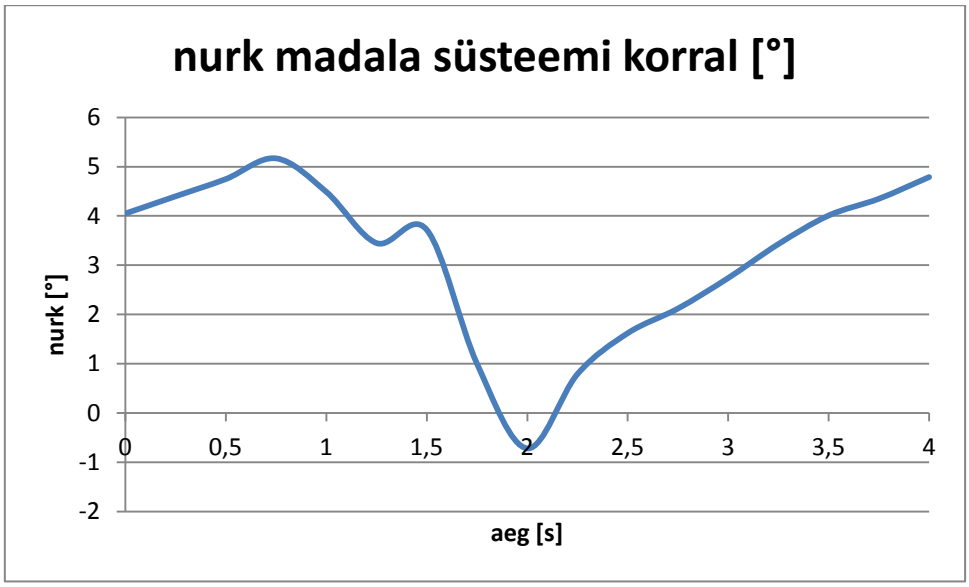
5.2.1. Roboti liikumine edasi madalas asendis

Selel 5.2. on kujutatud pendli nurga muutumist kogu liikumise vältel. Sellelt on näha, et süsteemi stabiilne asend enne liikumise algust ning ka asend selle stabiliseerumisel pärast liikumise lõppemist umbes kolmandal sekundil on umbes neli kraadi vertikaalsest asendist liikumise suunas. See nurk roboti stabiilses asendis on kõikide läbiviidud katsete suurim hälve vertikaalasendist. Kuna robot reguleerib oma asendit ka lihtsalt püüdes paigal püsida, siis on täpset tasakaaluasendit nende katsete põhjal keeruline määrata. Roboti madalas asendis toimub kogu liikumise vältel nurga muutumine umbes kuue kraadi ulatuses.

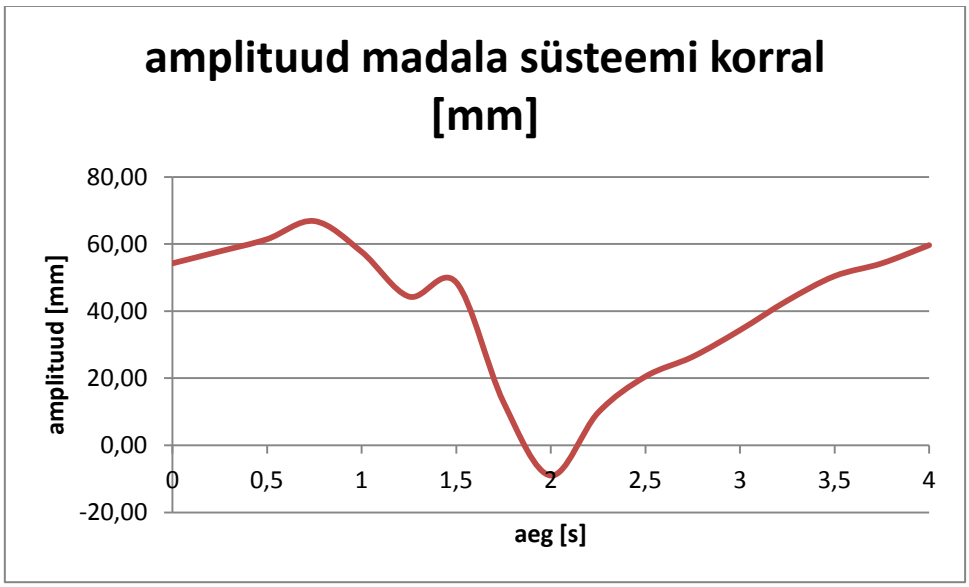
Graafikult on näha ka seda, et enne liikumise alustamist suureneb nurk liikumise suunas, mis on vajalik selleks, et liikumise inertsi toimel saaks pendel vaid vertikaalasendile läheneda, mitte liikumisele vastupidises suunas viltu vajuda. Liikumahakkamisel suurendatud kalle on piisav selleks, et liikumisel saaks pendli nurk läheneda tasakaaluasendile ning 1,5 sekundi juures on näha asendi reguleerimist taas ette-suunas, et tagada piisav lähenemismaa tasakaaluasendile liikumise vältel. Enne peatumist reguleerib platvorm end selliselt, et nurk tekib pendelmehhanismi ning vertikaalasendi vahel liikumisele vastupidises suunas. Peatumisel paiskub pendliosa taas liikumise suunas, kus kaldumine enne peatumist on olnud taas piisav selleks, et pendel saaks pärast peatumist taas üsna ühtlaselt läheneda oma esialgsele tasakaaluasendile.

Graafikul on aga näha, et pärast liikumise lõppemist on pendli reguleerimine tasakaaluasendi suunas üsna aeglane ning ka sekund pärast liikumise lõppemist jätkub pendli ühtlane liikumine oletatava tasakaaluasendi ümber. Reguleerimisaeg tundub sellelt graafikult olevat üsna pikk ning kuna ei ole võimalik määrata täpset tasakaaluasendit, pole võimalik järeldada, kas nurga lähenemine viiele kraadile neljandal sekundil on ülereguleerimine või mitte.

Selel 5.3. kujutatud amplituud pendli vertikaalsest asendist kirjeldab põhimõtteliselt seda sama, mida näeme ka nurgagraafikul. Sellelt on näha, et pendli tasakaaluasend on umbes 60 mm vertikaalsest asendist liikumise suunas ning liikumise vältel muutub amplituud samamoodi nagu nurk vertikaalasendist, sest nende vahel on otsene seos. Maksimaalne amplituudierinevus vertikaalasendist on umbes 10 mm enne liikumise lõppemist liikumisele vastupidises suunas. Kogu liikumise amplituud on umbes 80 mm. Tabel katse jooksul kogutud väärtustest on lisas 4.



Sele 5.2. Nurk madala süsteemi korral

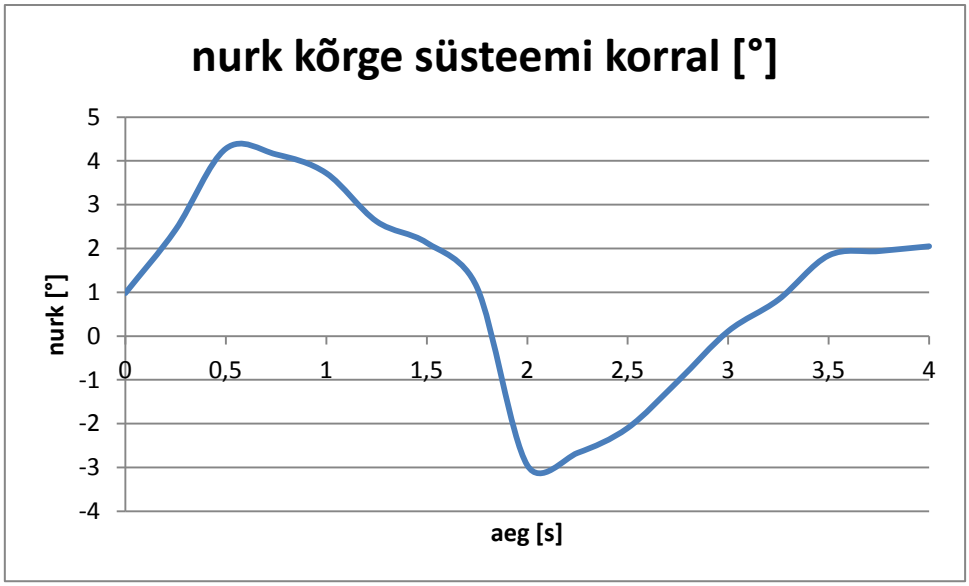


Sele 5.3. Amplituud madala süsteemi korral

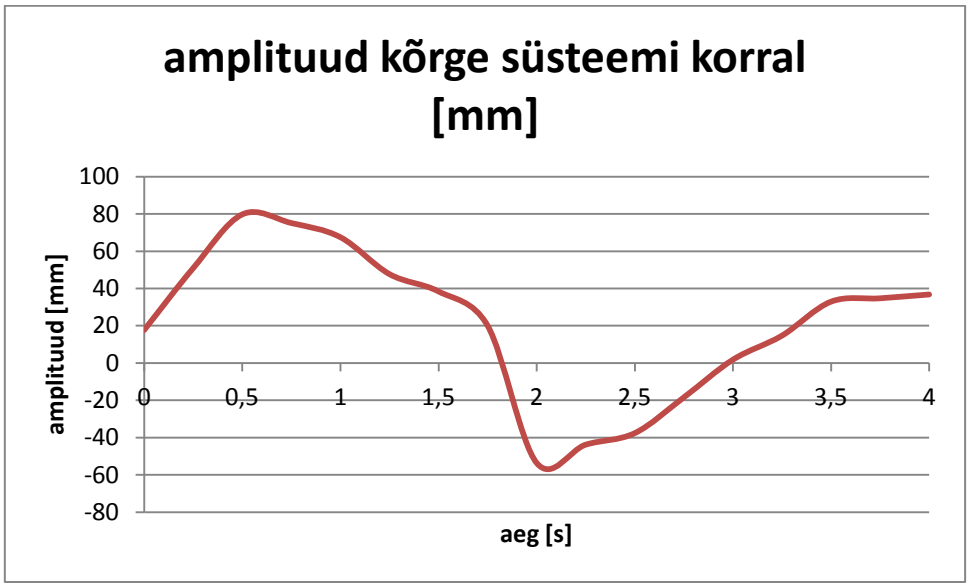
5.2.2. Roboti liikumine edasi kõrges asendis

Selel 5.3. kujutatud nurga muutumise graafikul on võimalik täheldada seda, et süsteem on võrreldes madala asendiga ebastabiilsem, sest kogu nurga muutumise ulatus liikumise vältel on pisut enam kui 7 kraadi, mis on rohkem, kui sama katse läbiviimisel selle madalas asendis. Süsteemi tasakaaluasend näib olevat umbkaudu kaks kraadi vertikaalasendist liikumisega samas suunas. Enne liikumise alustamist reguleerib süsteem ennast liikumise suunas, et takistada pendli tagasivajumist liikumise vältel sellele vastupidises suunas. Kuna asend on kõrgem eelmises alapunktis analüüsitud asendist, siis on näha, et liikumiseks ettevalmistumine nõuab kalde suurendamist lausa kolme kraadi ulatuses, kusjuures madala asendi puhul oli vajalik vaid üks kraad. Liikumise vältel väheneb pendli nurk vertikaalasendist üsna ühtlaselt ning toimub vaid kaldumise kiiruse vähenemine vahemikus 0,5–1 s. Enne liikumise lõppemist toimub nurga reguleerimine liikumisele vastupidisesse suunda üsna järsult, jõudes kolme kraadini vertikaalasendist pendli esialgsele kaldele vastupidises suunas. See nurk on piisav, et pärast peatumist üsna aeglaselt reguleerida nurk tasakaaluasendini. Kuna 3,5 s juures on näha, et saavutatud reguleerimisnurk on tasakaaluasendi ümbruses ning läheneb kahele kraadile üsna aeglaselt, võib selle katse põhjal oletada, et peatumisel ülereguleerimist ei toimunud.

Selel 5.5. on välja toodud pendli amplituudi muutumine liikumise vältel. Tasakaaluasendiks võib lugeda umbes 30 mm vertikaalsest asendist liikumise suunas. Enne liikuma hakkamist suurendatakse amplituudi 80 mm-ni vertikaalsest asendist, kust see väheneb enne liikumise peatumist 60 mm-ni vertikaalasendist liikumisele vastupidises suunas. Kogu liikumise amplituud on umbes 140 mm. Tabel katse jooksul kogutud väärtustest on lisas 4.



Sele 5.4. Nurk kõrge süsteemi korral



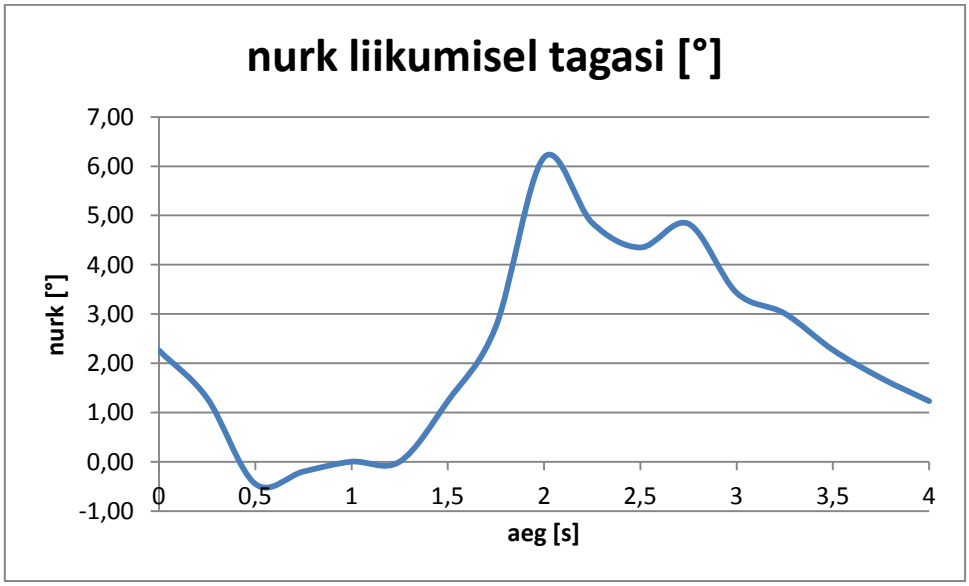
Sele 5.5. Amplituud kõrge süsteemi korral

5.3. Roboti liikumine tagasi

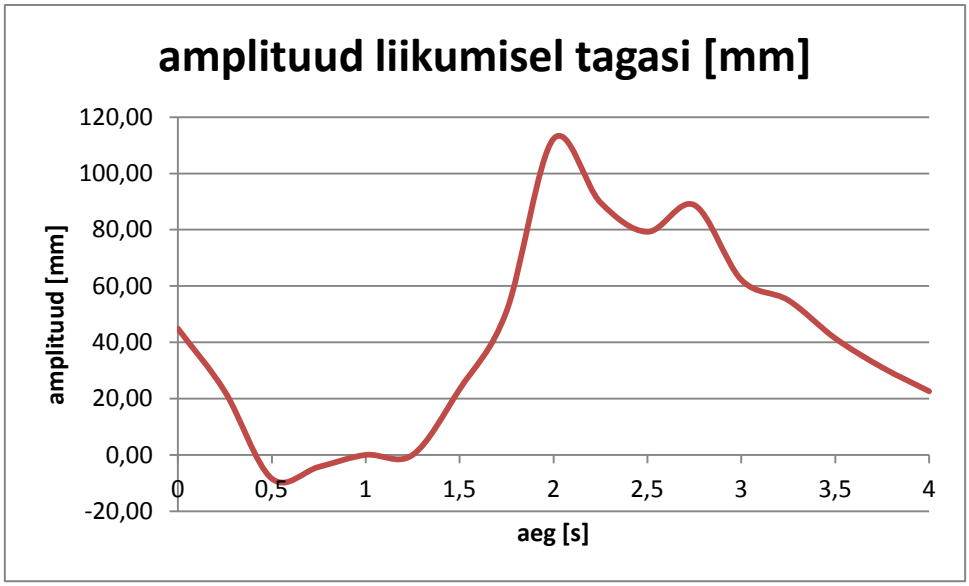
Selel 5.6. on kujutatud pendelmehhanismi nurga muutumist liikumisel tagurpidi. Nagu ka edasiliikumisel, on süsteemi stabiilne asend umbes 2° vertikaalse asendi suhtes edasiliikumise suunas, kust see enne liikumise alustamist reguleeritakse umbes 3° võrra liikumisele vastupidisesse suunda. Liikumise alguses lähendatakse nurka vertikaalse asendi juurde üsna aeglaselt, kus 1,25 s juures on näha korrigeerimist, misjärel valmistudes peatuseks suurendatakse nurka liikumisele vastupidises suunas üsna järsult umbes 4°-ni süsteemi tasakaaluasendist. Pärast peatumist läheneb nurk üsna aeglaselt tasakaaluasendini, kuid

peatumine pole sama stabiilne kui liikumisel edaspidi, sest 2,75 s juures on näha süsteemi reguleerimist tasakaaluasendile lähenemisele vastupidises suunas. Kuna tasakaaluasend pole täpselt määratud, siis võib oletada, et tasakaaluasendile lähenemise jätkumine pärast 3,5 s ei pruugi olla ülereguleerimine. Kogu liikumise vältel muutub nurk umbes 7° ulatuses, mis on sama palju kui eelmises alapunktis käsitletud edasiliikumisel.

Selel 5.7. on kujutatud amplituudi muutumist kogu liikumise vältel. Tasakaalupunkt on umbes 30 mm vertikaalsest asendist, kust see liikumise alustamiseks viiakse 10 mm teisele poole vertikaalasendit ning enne peatumist umbes 110 mm kaugusele vertikaalasendist algasendi suunas. Kokku muutub amplituud 120 mm võrra kogu liikumise vältel. Tabel katse jooksul kogutud väärtustest on lisas 4.



Sele 5.6. Nurk liikumisel tagasi



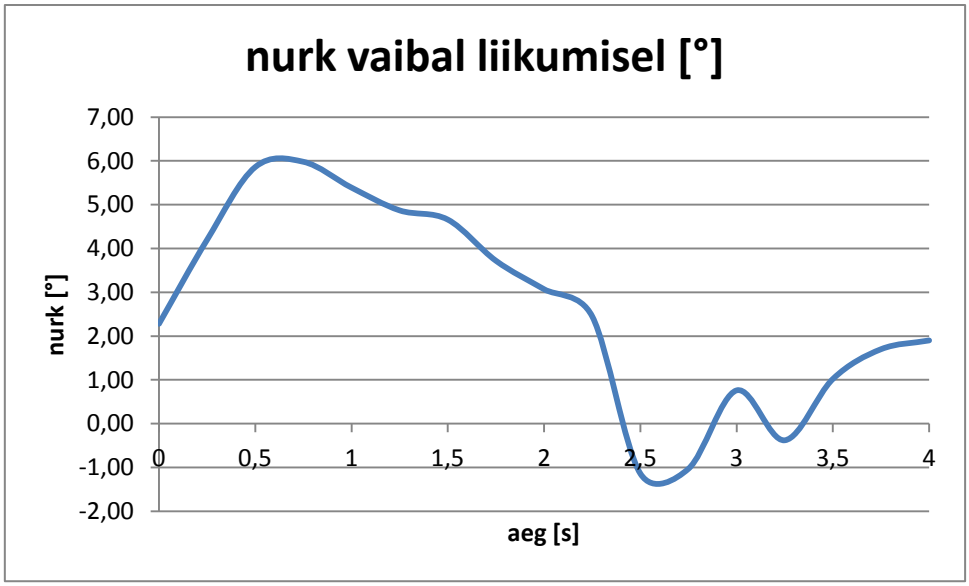
Sele 5.7. Amplituud liikumisel tagasi

Analüüsitud kolme katse põhjal on näha, et olenemata vahepealsetest reguleerimisprotsessidest, on suurima nurga- ning amplituudierinevusega katse kõrge pendli liikumine edasisuunas. Kuna pildi pealt andmete saamiseks on ülevaatlikuma graafiku saamiseks parem suurem erinevus amplituudis ning nurgas, siis on kõik järgnevad katsed tehtud roboti kõrges asendis liikumisel edasi.

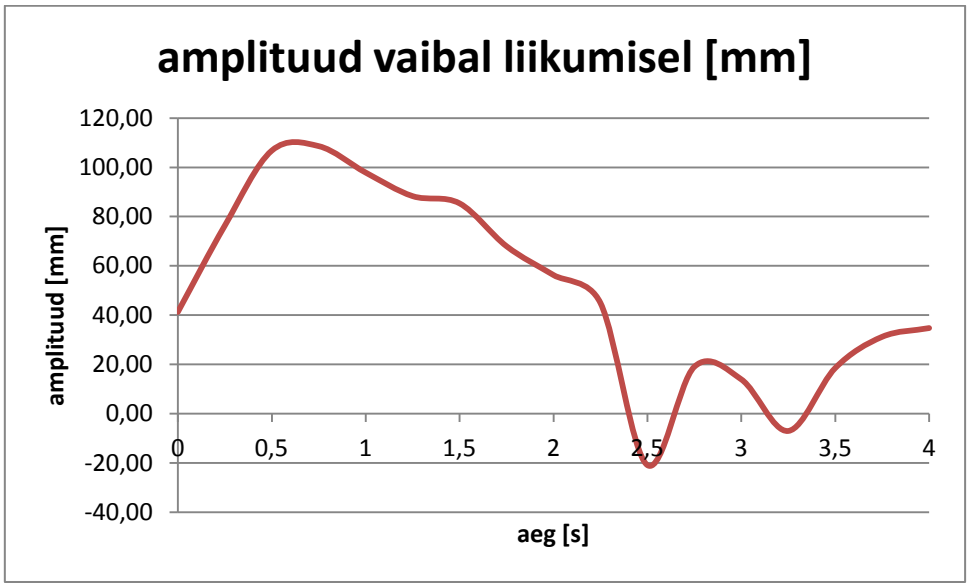
5.4. Liikumine vaibal

Sele 5.8. kujutab pendli nurga muutumist liikumisel vaibal. Sarnaselt teistele liikumistele horisontaalsel pinnal kõrge süsteemi korral on tasakaaluasend umbes 2° vertikaalasendist. Enne liikumise alustamist reguleeritakse nurka liikumise suunas 4° võrra liikumise suunda, mida on rohkem, kui enne liikumise alustamist põrandal. Liikumise vältel läheneb nurk tasakaaluasendile küllalt ühtlaselt ainsa aeglustumisega 1,25 s ja 1,5 s vahel. Enne peatumist reguleeritakse pendli nurk liikumisele vastupidises suunas veidi üle 1° vertikaalasendist. Pärast peatumist on 3 s juures näha, et lähenemine tasakaaluasendile pärast liikumise lõppemist on olnud liiga kiire ning nurka korrigeeritakse taas vastupidises suunas, misjärel läheneb nurk tasakaaluasendile. Jooniselt on näha, et ülereguleerimist ei toimu.

Selel 5.9. toodud graafikul on toodud pendli amplituudi muutumine vaibal liikumisel. Tasakaaluasend on umbes 40 mm vertikaalasendist liikumise suunas. Enne liikumise alustamist reguleeritakse see samas suunas vertikaalasendist umbes 110 mm-ni ning enne peatumist 20 mm-ni vertikaalasendist vastupidises suunas. Kogu liikumise kestel muutub amplituud umbes 130 mm võrra. Tabel katse jooksul kogutud väärtustest on lisas 4.



Sele 5.8. Nurk liikumisel vaibal



Sele 5.9. Amplituud liikumisel vaibal

5.5. Liikumine kaldpinnal

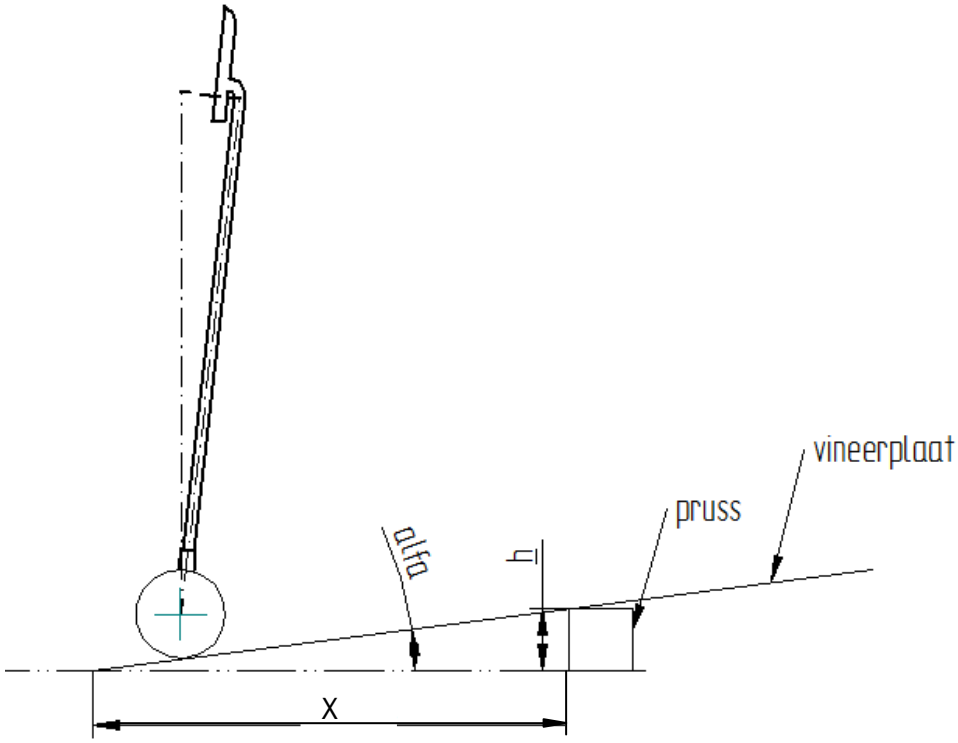
Roboti teadaolevate spetsifikatsioonide põhjal on liikur suuteline üles minema kuni viie protsendilise kallakuga pindadest, mis on võrdne umbes 2,86 kraadiga (valem 5.1). Kuna katseliselt selgus, et ülesminekul kaldpinnast ei ole seismajäämiseni läbitav vahemaa roboti tagasivajumise tõttu sama suur kui horisontaalpinnal, siis kasutasin kõikide kaldpinnal tehtud katsete juures tsüklit pikkusega 3 s. Katsetasin liikumist 2-, 5- ning 7-kraadisel kaldpinnal, mille jaoks mul oli kasutada vineerplaat ning 40 mm ja 90 mm laiused puitprussi jupid. Kombineerides prussi tükke ning teades soovitavaid nurkasid, arvutasin tangensi (valem 5.2)

abil välja kauguse punktist, kus vineerplaadi alumine pool puudutab põrandat punktini, kuhu toeks asetada prussi jupid (sele 5.10.).

$$\alpha = arctan\left(\frac{\%\alpha}{100}\right) \quad (5.1)$$

kus α- kaldpinna nurk kraadides,

%α- kaldpinna nurk protsentides.



Sele 5.10. Tangensi abil kaldpinna konstrueerimine

$$x = \frac{h}{tan\alpha} \quad (5.2)$$

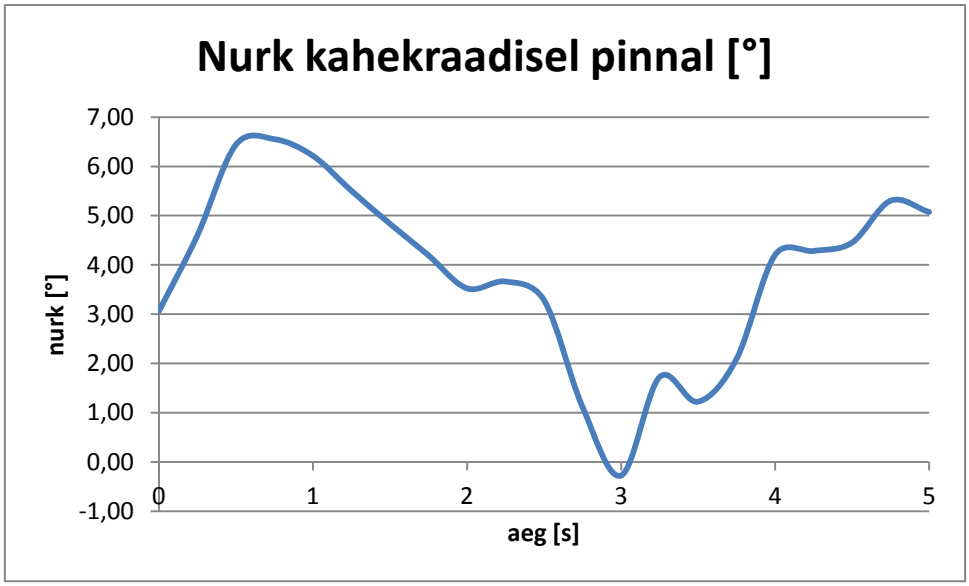
kus *x* - kaugus vineerplaadi ja põranda kokkupuutepinna ja tugiprussi vahel,

h -tugiprussi kõrgus.

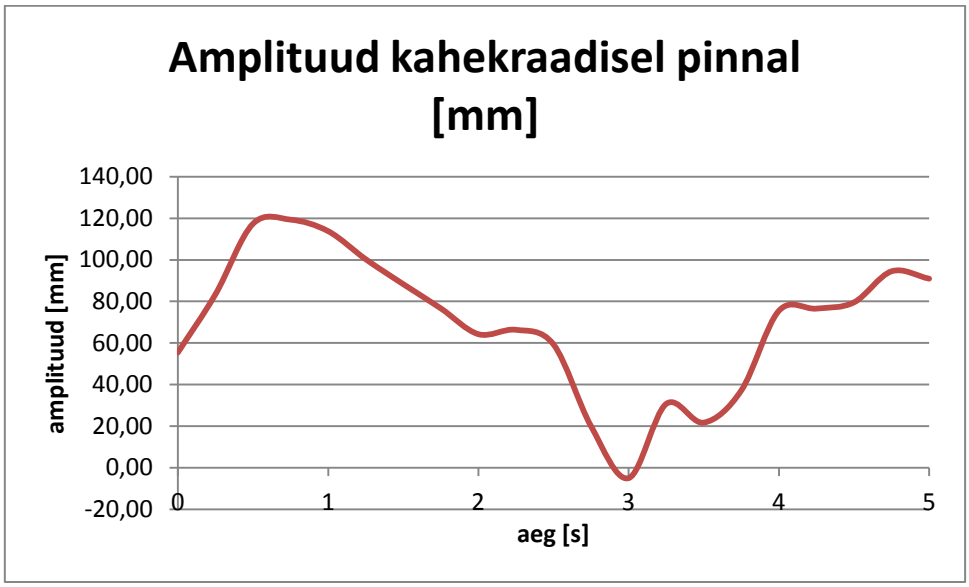
5.5.1. Liikumine kahekraadisel kaldpinnal

Selel 5.11. on kujutatud pendli nurga muutumine liikumise vältel kahekraadisel pinnal. Graafikult on näha, et kahekraadisel pinnal on tasakaaluasend 3° ümbruses vertikaalasendist liikumise suunas, mis on ühe kraadi võrra rohkem kui horisontaalsel pinnal. Sellise väikese kaldega pinna juures on nurga muutumise graafik üsna sarnane graafikutega, kus on kujutatud liikumist horisontaalsel pinnal. Enne liikumise alustamist reguleeritakse nurk liikumise suunas ligi 7°-ni vertikaalasendist ning 2 s juures reguleeritakse nurka liikumise ajal, 2,5 s paiku hakatakse valmistuma peatumiseks, kus enne peatumist viiakse nurk veidi liikumisele vastupidisesse suunda üle vertikaalasendi. Pärast peatumist tehakse nurga reguleerimisel tasakaaluasendini palju korrektiive ning seda ilmselt seepärast, et kaldpinna tõttu toimub tagasivajumine, mistõttu peab robot arvestama teistsuguste tasakaalutingimustega kui horisontaalsel pinnal. Ilmselt ka seetõttu toimub nurga ülereguleerimine tasakaaluasendist enam kui 2° võrra, sest on tarvis hoida nurka kalde tõttu allaliikumisele vastupidises suunas.

Selel 5.12. on kujutatud pendli amplituudi muutumine liikumise vältel kahekraadisel pinnal. Tasakaaluasend tundub olevat 60 mm ümbruses vertikaalsest asendist liikumise suunas. Enne liikumist suureneb amplituud 120 mm-ni vertikaalasendist liikumise suunas ning enne peatumist väheneb see veidi liikumisele vastupidisesse suunda vertikaalasendi läheduses. Pendli amplituud kogu liikumise vältel on umbes 120 mm. Tabel katse jooksul kogutud väärtustest on lisas 4.



Sele 5.11. Nurk kahekraadisel pinnal



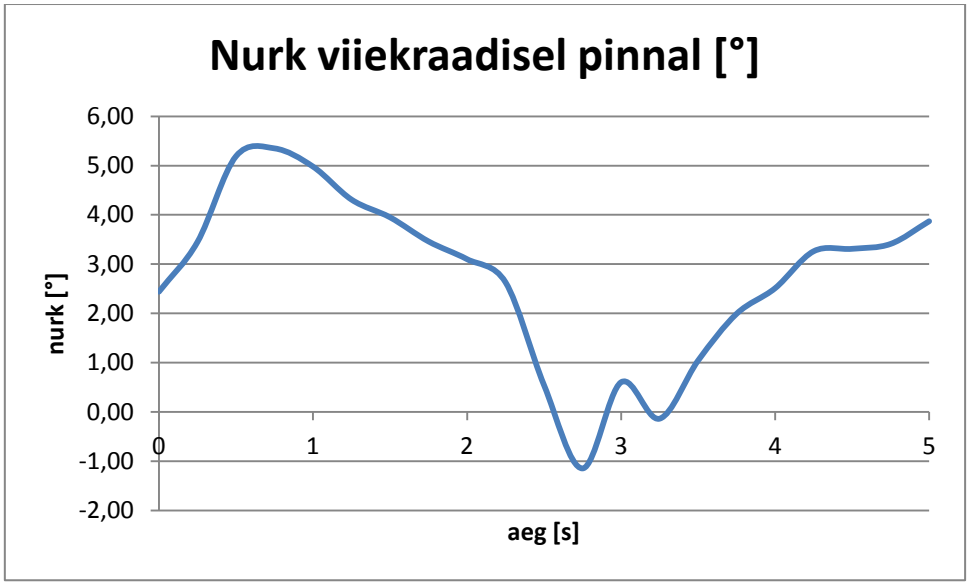
Sele 5.12. Amplituud kahekraadisel pinnal

5.5.2. Liikumine viiekraadisel kaldpinnal

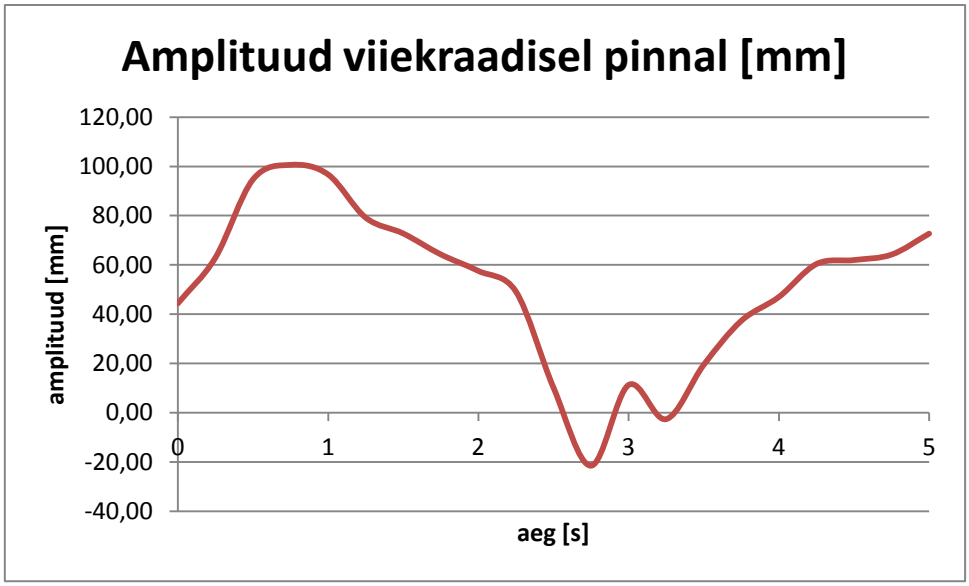
Selel 5.13. on kujutatud nurga muutumine viiekraadisel kaldpinnal kogu liikumise vältel. Pendli tasakaalupunkt näib olevat 3° ümbruses vertikaalasendist liikumise suunas. Enne liikumise alustamist suurendatakse seda nurka 6°-ni vertikaalasendi suhtes ning liikumise ajal on pendli liikumine tasakaaluasendi suunas küllalt ühtlane. Enne peatumist vähendatakse nurka 1°-ni vertikaalasendi suhtes liikumisele vastupidises suunas, kust see hakkab üsna

kiiresti tasakaalupunkti poole tõusma, mistõttu 3. sekundi juures toimub reguleerimine vastassuunda. Lähenemine tasakaaluasendile on küllalt sujuv, kuid pärast liikumise lõppemist on näha ülereguleerimist. Seda taaskord seetõttu, et tegelikult pole liikumine lõppenud, vaid gravitatsiooni tõttu liigub robot kallakut mööda alla tagasi, mistõttu püüab robot hoida nurka liikumisele vastu. Kogu liikumise vältel muutub pendli nurk veidi enam kui 6°.

Selel 5.14. on kujutatud amplituudi muutumist liikumisel viiekraadisel kaldpinnal. Tasakaaluasend on umbes 40 mm kaugusel vertikaalasendist liikumisega samas suunas. Liikumise alustamiseks suurendatakse seda kaugust 100 mm-ni ning enne peatumist vähendatakse kaugus umbes 20 mm-ni vertikaalasendi suhtes liikumisele vastupidises suunas. Kogu liikumise ulatuses on amplituud umbes 120 mm. Tabel katse jooksul kogutud väärtustest on lisas 4.



Sele 5.13. Nurk viiekraadisel pinnal

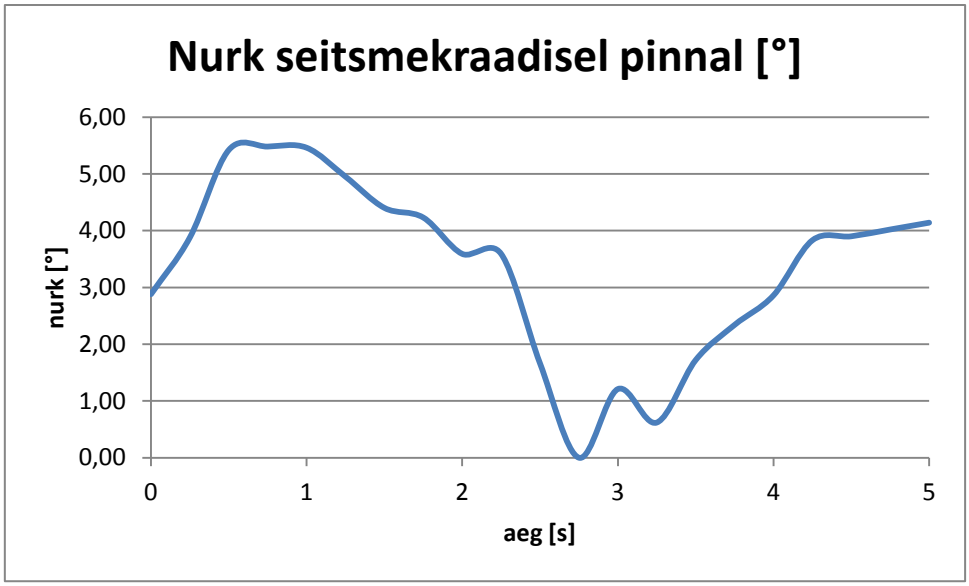


Sele 5.14. Amplituud viiekraadisel pinnal

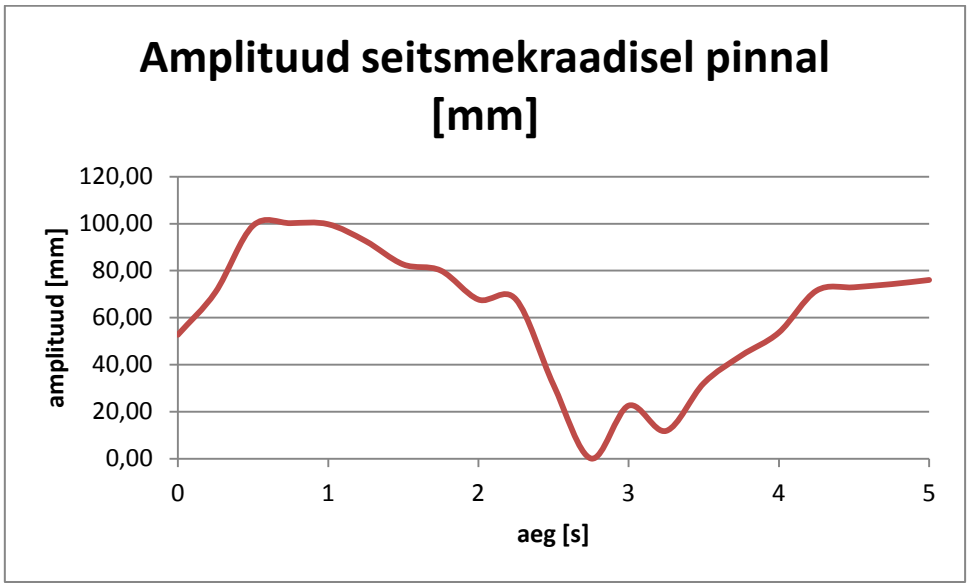
5.5.3. Liikumine seitsmekraadisel kaldpinnal

Sele 5.15. kirjeldab nurga muutumist liikumisel seitsmekraadisel kaldpinnal. Tasakaaluasend asub 3° ümbruses liikumise suunas vertikaalasendi suhtes. Liikumise alustamiseks suurendatakse nurka ligi 6°-ni vertikaalasendi ning liikumise kestel reguleeritakse pendli asendi liikumist tasakaaluasendi poole üsna ühtlaselt. Enne peatumist viiakse pendli asend vertikaalseks ning pärast liikumise lõppemist hakatakse tasakaaluasendile kiiresti lähenema, mistõttu toimub 3 s juures asendi reguleerimine vastupidises suunas. Lähenemine tasakaaluasendile on üsna sujuv ning ülereguleerimist on üllataval kombel vähem kui väiksemate kallete korral. Siiski toimub gravitatsiooni mõjul pendli tagasivajumine, kuid katse järgi suudab pendel paremini oma asendit tasakaaluasendi ümbruses säilitada kui väiksemate kallete korral. Kogu liikumise vältel muutub nurk vähem kui 6°.

Selel 5.16. on kirjeldatud amplituudi muutumist liikumise jooksul. Tasakaaluasend asub umbes 60 mm kaugusel vertikaalasendist liikumise suunas. Liikumise alustamiseks suurendatakse seda kaugust 100 mm-ni ning enne peatumist liigub pendel vertikaalsesse asendisse. Kogu liikumise vältel muutub amplituud 100 mm. Tabel katse jooksul kogutud väärtustest on lisas 4.



Sele 5.15. Nurk seitsmekraadisel pinnal



Sele 5.16. Amplituud seitsmekraadisel pinnal

Olenemata sellest, et roboti spetsifikatsioonides on välja toodud maksimaalse kaldena vähem kui kolme kraadine kaldpind, on liikur suuteline stabiilselt üles veerema ka järsematest tõusudest. Katsetest oli näha, et roboti seismajäämine keset kaldpinda ei ole eriti võimalik, sest ta hakkab tagasi veerema. Kui aga eesmärgiks on kaldpinnast üles veeremine, mitte sellel peatumine, siis saab liikurplatvorm hakkama ka järsemate kallakutega kui seda on viis protsenti.

5.6. Tähelepanekuid katsetest

Olgugi, et näidiskatsete eesmärk on pigem testprogrammi illustreerimine kui platvormi liikumiskäitumise süvitsi uurimine, on tehtud testidest siiski võimalik näha teatavat mustrit. Näiteks oskab platvorm arvestada liikumisega kaasneva inertsiga, sest kõikide katsete juures reguleeris robot oma nurka liikumise suunda veel enne liikumise alustamist. Samuti toimus nurga reguleerimine liikumisele vastupidises suunas enne peatumist. Sellisest käitumisest võib eeldada, et kontroller roboti liikumist üle ei reguleeri ning nii liikumise ajal kui ka pärast peatumist toimub seadme sujuv reguleerimine tasakaaluasendi poole. Samuti on nüüd teada, et seade suudab liikuda üles ka enam kui viieprotsendilise kaldega pinnast. Ometi oli kõikidel katsetatud kallakutel seisma jäämisega probleeme. Olgugi, et robot ennast tasakaalus hoidis, toimus raskusjõu tõttu tagasivajumine.

Samuti võib täheldada, et tegelikult on kõikide katsete nurga ning amplituudi muutumise graafikud sarnased selles mõttes, et ei toimu ettearvamatut käitumist. Graafiku väärtuste ümardamise korral võiks täheldada sujuvat suuruse muutumist ühele poole ning seejärel teisele poole. Häiringutest või liigkiirest reguleerimisest tingitud kontrolleri poolt tehtud korrektsioonid on nendes katsetingimustes läbi viidud testide puhul tühised.

KOKKUVÕTE

Double Robotics platvormi uurimise käigus on kogutud kokku roboti kohta teadaolev informatsioon ning kasutatud seda olemasoleva süsteemi kaardistamiseks. Esmalt on koostatud matemaatiline mudel, mille loomise eesmärgiks on olemasolevat süsteemi rahuldavalt kirjeldada. Virtuaalsete mudelite olemasolul on võimalik katsetada näiteks uusi algoritme, mida tegeliku süsteemi peal katsetada ei tihkaks. Samuti võimaldab see harjutada roboti juhtimist, sest kirjeldades ära selle käitumise erinevates tingimustes, nagu nt liikumise üle lävepakkude või erinevatel kalletel, võimaldab see, juba teades liikumise iseärasusi, juhtida tegelikku platvormi teadlikumalt. Simuleerides mudeli abil liikumist erinevates keskkondades ning analüüsides katsetulemusi, on võimalik tulevase arenduse käigus optimeerida praegust kontrollsüsteemilahendust ning miks ka mitte lisada uusi elemente juba olemasolevale.

Selleks on olemasolev süsteem lihtsustatud kolmeks osaks, milles on vaadeldud loodava olekumudeli sisendiks oleva mootori dünaamikat, rataste ning pendli dünaamikat eraldi. Nende kolme osa kombineerimisel on koostatud olekuruumi võrrand, millesse platvormi tegelike väärtuste asendamisel on võimalik luua virtuaalne mudel. Süsteemi lihtsuse mõttes on tervet pendliosa koos iPad-iga vaadeldud ühtse tervikuna. Tuletatud olekuruumi võrrand sisaldab endas lähteandmeid, milleks on pendli mass, pikkus ning inertsimoment, raskuskiirendus, ratastega osa mass, inertsimoment, rataste raadius. Samuti mootori väljundpinge, elektrisüsteemi nominaalne takistus ning pöördemomendi konstant ja genereeritud elektromotoorjõu konstant.

Kuna aga olulisteks parameetriteks olekuruumi kirjeldamisel on meile tundmatud väärtused nagu mootori rakendatav pinge, elektriline takistus, pöördemomendi konstant, genereeritud elektromotoorjõu konstant ning ei ole teada terviksüsteemi ligikaudne kaalujaotus, ei ole võimalik olekuruumi piisaval määral kirjeldada. Samuti pole teada balansseerimissüsteemis kasutatava kontrolleri tüüp, mistõttu selle lõputöö raames mudelit realiseeritud ei ole.

Samuti on testprogrammi loomisega uuritud erinevaid arendusvõimalusi. Suurimaks abiks platvormi edasiarendusel oleks võimalus sinihamba ühenduse passiivseks pealtkuulamiseks, kuid selleks meil hetkel riistvaralised võimalused puuduvad ning ainus taskukohane turulahendus on veel katsetusjärgus, seega ei ole garantiid kõikide sinihamba pakettide leidmiseks. Teine takistav tegur võib olla sinihamba ühenduse krüptimine, mille

lahtimuukimine on väga mahukas töö. Sinihamba ühenduse abil mängukonsooli pultide ning teiste sinihammast kasutatavate seadmete ühendamine seadme juhtimiseks on praeguses etapis samuti veel ülesanne vilunud Apple'i arendajatele. Seadmete ühendamiseks liikurplatvormiga oleks tarvis teada protokolle, mis vastavad hetkel iPad-ist saadetud liikumiskäskudele. Apple'i seadmete iseärasuse tõttu on nende ühendamine iPad-iga keerukas, kuid tehtav, selle süsteemi puhul aga, kus sinihamba ühendus on lisaks juba iPad-i ning platvormi vaheliseks ühenduseks, ei ole ka tervikliku süsteemi kontrollimiseks hetkel võimalusi. Kuna iPad-i on võimalik ühendada vaid teiste Apple'i toodetega või Apple'i poolt sertifitseeritud ühilduvate toodetega, siis on tarvis teiste seadmete sinihambaga ühendamise jaoks kasutada iPad-i sisseehitatud sinihamba pinule alternatiivset laiendatud võimalustega pinu. Selle pinu kasutamine aga keelab sisseehitatud pinu kasutamise, mille abil on iPad ühendatud liikurplatvormiga. Kuna *double*'i tarkvaraarenduskomplekt, mis on vabalt saadaval, ei sisalda muid faile peale nende, kus on võimalik seadme liikumist mõjutada, siis pole võimalik ka sinihamba pinu alternatiivsele laiendatud pinule ümber kirjutada. Alternatiividena lähteprogrammi loomisel jäid sõelale lokaalse rakenduse loomine ning juhtimiseks kasutatava veebisirvija lehe lähtekoodi muutmine. Nende kahe võrdlemisel on optimaalsem kasutada lokaalset aplikatsiooni juba seetõttu, et sellega saab välistada juhtmevaba interneti olemasolu vajaduse, samuti ei pea arvestama viivitustega, mida info liikumine läbi serverite ning internetiühenduse põhjustada võib.

Testprogrammi loomisel lokaalse rakendusena palju arendustööd ei toimu, sest mis tahes süsteemi muudatuse juures, nagu teisest seadmest sinihamba abil juhtimine, iPad-i asendamine mõne teise seadmega jms, taandub kõik sellele, et on tarvis teada saada sel hetkel iPad-i ning platvormi vahel vahetatavaid sinihamba protokolle. Seda kas ühenduse passiivse pealtkuulamise abil või seadme originaalaplikatsiooni tervikkoodile ligi pääsedes.

Kuna programmi koostamisel sai kasutatud arvutit, mis kasutab Windows 7 operatsioonisüsteemi, siis tuli lokaalse rakenduse loomiseks WMware Workstationit kasutada, milles sai virtuaalne masin loodud, milles käivitub Apple'i operatsioonisüsteem OS X Mountain Lion. Selles keskkonnas laeti alla Xcode 5, mis on iOS arendamiseks vajaminev arenduskeskkond, mis sisaldab ka meie süsteemi iPad-i tarkvaraversiooni tarkvaraarenduskomplekti.

Testprogrammi idee seisneb selles, et oleks võimalik saata liikurplatvormile fikseeritud kestusega käske, et katsed oleksid korratavad erinevates tingimustes. iPad-ile sai kirjutatud

lokaalne rakendus, kus kasutajal on võimalik määrata ära tsükli kestus, stabiliseerimisaeg enne katse algust ning katse korduste arv. Seejärel on võimalik valida soovitud liikumine kas edasi, tagasi, pööramine vasakule või paremale.

Kasutades loodud testprogrammi, on selle töö käigus läbi viidud seitse näidistesti pendli liikumisel edasi madalas asendis, edasi ning tagasi pendli kõrges asendis, liikumisel vaibal ning kolme erineva kaldenurgaga kaldpinnal. Katsed on üles võetud kaameraga, millest saadud videomaterjal sai kaadrisagedusega 25 kaadrit sekundis piltideks renderdatud. Kasutades iga kuuendat kaadrit, saadi katsetest pendli nurga ning amplituudi muutumise graafikud 0,25 s sagedusega vastavalt nelja ning viie sekundi jooksul sõltuvalt katsest. Andmete saamiseks sai Solid Edge'i abil kõikidele kasutatud kaadritele sirged joonestatud, mille abil saadi kätte nurga muutumine ning peale piltide mõõtkavasse teisendamist ka tegelik amplituud. Saadud andmed on koondatud tabelitesse, mille abil on koostatud graafikud mõlema muutuja jaoks. Olgugi, et tegu on selle töö raames vaid näidistestidega, on nendes katsetes juba võimalik leida mõningaid seaduspärasusi. Näiteks reguleeritakse seadme nurka juba enne liikumise alustamist, samuti enne liikumise lõppemist, et korvata inertsist põhjustatud kõikumist.

Kirjutatud töö on oma eesmärgi täitnud, sest kõik magistritöös püstitatud ülesanded on saadaoleva info piires täidetud. On koostatud süsteemi kirjeldav mudel ning uuritud erinevaid võimalusi testprogrammi loomise näitel seadme edasiseks arenduseks, loodud lokaalne rakendus platvormi liigutamiseks, mis ei vaja peale käivitamist enam kasutajapoolset abi ning viidud seda kasutades läbi seitse näidiskatset. Töö käigus platvormi kaardistamine võiks olla abiks roboti edasisel uurimisel ning katsetamisel.

SUMMARY

The current work is a research of the Double Robotics platform. Available information about the robot has been gathered and used to map the current system. Firstly, a mathematical model of the system is composed to describe the system. Such tests like new algorithms that one would not carry out on a real platform can be made on a virtual model. The model would also allow the practice of controlling the robot, because while being able to describe its behaviour in different conditions, such as moving over thresholds or on slopes, it enables a more conscious control of the real system. Simulating the movement in different environments and analyzing the test results allows optimizing the current control system and adding new elements to the ones already a part of the system.

For that purpose the current system has been simplified into three segments in which the dynamics of the motor, which is an input of the state space model, the dynamics of the wheels and the dynamics of the pendulum, are observed separately. By combining the segments a state space model for the system is obtained. By substituting the real parameters of the system into the obtained model a virtual model can be created. To simplify the system the pendulum part including the iPad on top has been viewed as a whole. The state space model includes parameters of the system that are the mass, length and the moment of inertia and the gravitational acceleration of the pendulum, the mass and moment of inertia of the moving system of wheels and the radius of the wheels. It also includes the output voltage of the motor, the nominal resistance of the electrical system and the constants of torque and counter electromotive force.

As the crucial parameters for describing the state space model, such as the output voltage, electrical resistance, constants of torque and counter electromotive force, are unknown likewise to the exact weight distribution of the system, the state space model for this system cannot be described sufficiently to be executed. Since also the controller used to balance the robot is unknown, the state space model has not been put into practice in this thesis.

Different options for further development for this platform have been researched as well on the example of the creation of a test program for the robot. A big contribution for further development would be the possibility to passively gather information about the Bluetooth connection. However, we do not have any possibilities for that at this moment. The only affordable appliance for that has still not been fully tested and does not give any guarantee for

finding all the necessary packets. Another hindering factor is that it is likely that the Bluetooth connection has been crypted and the data can be difficult to extract. Also the tasks to connect other appliances like game console remotes in order to control the platform are still appropriate for the experienced Apple developers to be solved. To connect the appliances straight to the moving platform one would need to know the protocols that are relevant to the moving of the platform sent from iPad. Because of the features of Apple products the connection between external accessories and the iPad is complex but manageable. As in this system the iPad needs to be connected to the platform via Bluetooth as well, there are currently no possibilities for controlling the whole system. Since iPad can be connected only to other Apple products or Made For iPad certified products, the only way for connecting external accessories which are neither of the two, an alternative stack for Bluetooth has to be used. The problem is that using an alternative extended stack disables the original iPad Bluetooth stack that is needed for the connection between the iPad and the platform itself. Since the open-source software development kit for Double does not include any other files besides the ones that are needed to control the movement, it is also not possible to change the Bluetooth stack the written application will be using. Two remaining alternatives for creating the test program are a local application in the iPad and changing the source code of the web browser on the page that is used to control the robot. From these two it is more optimal to choose the local application in the iPad, because it discards the need for having a wifi connection. Also delays caused by information moving through servers or low internet speed can be left aside.

The creation of the test program as a local application in the iPad will not be contributing much to the further development of the platform. That is because in the case of any changes of the system like controlling it from another device via Bluetooth connection, replacing the iPad with some other device etc., we would anyhow need of know the protocols exchanged between the moving platform and the iPad. We can find it out either by passively eavesdropping the Bluetooth connection or by getting access to the whole code used to control Double.

As the computer used for the code runs on Windows 7, VMware Workstation was used in order to create the local application. In the workstation a virtual machine where OS X Mountain Lion simulated the Apple operation system environment was created. Then Xcode 5 was downloaded, which is a program for developing applications for devices running on iOS.

This program includes the software development kit needed for the current software version of the iPad.

The idea of the test program is that commands with fixed duration can be sent to the platform so the tests would be repeatable in different conditions and environments. The application written for this enables the user to set the duration of the test cycle, time for the device to stabilize itself before each test and the number of cycles to be made. Then the user can choose the mode of the movement which is either forward, backward or turning right or left.

In this thesis the developed program has been used to perform seven demo tests where the movement of the system is described while moving forward in its low position, forward and backward in its high position, moving on a carpet and on slopes with three different angles. The tests have been recorded with a video camera from which the material was rendered into frames with the frame rate of 25 fps. Using every sixth frame the graphs of the changing angles and amplitude of the pendulum with the accuracy of 0.25 s and duration of 4 s and 5 s with respect to the tests are obtained. For extracting the data from the frames Solid Edge draft was used to draw straight lines on the picture, which helped to measure the angle from the vertical position and also the amplitude after converting the obtained value into the real scale. The collected data has been arranged to the tables that are used to get the graphs for both variables. Even though in this thesis the tests are demos demonstrating rather the program than describing the platform, these few tests can describe some of its behaviour. For example, it can be seen that the angle of the pendulum is adjusted already prior to the movement as well as the stopping to make up for the inertial forces affecting it.

This thesis has filled its purpose, because all the set assignments have been completed within the existing information about the platform. A state space model for the system has been created, the further possibilities for development by creating a testing program for the device have been researched, a local iPad application for moving the device that does not need any user input after actuating has been created and seven demo tests using the created program have been performed. The research made within this thesis could be of help for further development and testing of this platform.

KASUTATUD KIRJANDUS

[1] „Apple e-pood,“ [WWW] http://store.apple.com/us/product/HE494LL/A/double-telepresence-robot. [Kasutatud 02.04.2014].

[2] „Automaatikainstituut, süsteemiteooria põhimõistete seletusi,“ [WWW] http://www.dcc.ttu.ee/las/SILTERM.html#o. [Kasutatud 22.04.2014].

[3] „Double Robotics veebitugi,“ [WWW] http://support.doublerobotics.com/customer/portal/articles/1366957-how-does-double-balance- . [Kasutatud 20.04.2014].

[4] „Rotary Electrodynamics of a DC Motor: Motor as Mechanical Capacitor,“ [WWW] http://www2.ece.ohio-state.edu/~passino/lab2_rotary_dynamics.pdf . [Kasutatud 21.04.2014].

[5] R. C. Ooi, „Balancing a Two-Wheeled Autonomus Robot,“ The University of Western Australia , Crawley, 2003.

[6] „IT college wiki,“ [WWW] https://wiki.itcollege.ee/index.php/Bluetooth. [Kasutatud 27.02.2014].

[7] „Bluetooth basics,“ [WWW] http://www.bluetooth.com/Pages/Basics.aspx. [Kasutatud 27.02.2014].

[8] E. Chai, B. Deardorff ja C. Wu, „Hacking Bluetooth,“ Massachusetts Institute of Technology, Cambridge, MA, 2012.

[9] „Arvutikaitse koduleht,“ [WWW] http://www.arvutikaitse.ee/arvutikaitse-algtoed/krupteerimine/. [Kasutatud 27.02.2014].

[10] „iOS developer library,“ [WWW] https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/AboutCoreBluetooth/Introduction.html . [Kasutatud 10.03.2014].

[11] „Apple veebitugi,“ [WWW] http://support.apple.com/kb/HT3647. [Kasutatud 10.03.2014].

[12] „DarWiin-remote project,“ [WWW] http://sourceforge.net/projects/darwiin-remote/. [Kasutatud 11.03.2014].

[13] „WiiC project,“ [WWW] http://wiic.sourceforge.net/. [Kasutatud 13.03.2014].

[14] „Btstack,“ [WWW] https://code.google.com/p/btstack/ . [Kasutatud 15.03.2014].

[15] T. Klosowski, „How to Use a Gamepad for Any iOS Game (Not Just Emulators),“ [WWW] http://lifehacker.com/5991266/how-to-use-a-gamepad-for-any-ios-game-not-just-

emulators/all. [Kasutatud 11.03.2014].

[16] S. Spencer, „How to Connect a Wii Remote to an iPod Touch," [WWW] http://science.opposingviews.com/connect-wii-remote-ipod-touch-13929.html . [Kasutatud 13.03.2014].

[17] M. Neuburg, „Anatomy of an Xcode Project," *Programming iOS 6, 3rd Edition*, O'Reilly Media, 2013, peatükk 6.

[18] „iOS Programming Basic: How Does the Hello World App Work?," [WWW] http://www.appcoda.com/ios-programming-basic-how-does-the-hello-world-app-work/. [Kasutatud 20.04.2014].

[19] „Mac Developer Library," [WWW] https://developer.apple.com/library/mac/documentation/cocoa/conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html. [Kasutatud 20.04.2014].

[20] „What is barrel lens distortion," [WWW] http://cameras.about.com/od/technologies/a/What-Is-Barrel-Lens-Distortion.htm. [Kasutatud 07.05.2014].

[21] „What is distortion," [WWW]. Available: http://photographylife.com/what-is-distortion. [Kasutatud 07.05.2014].

[22] „Double Robotics veebitugi," [WWW] http://support.doublerobotics.com/customer/portal/articles/1379820-double-specifications. [Kasutatud 01.03.2014].

[23] „Double Robotics SDK," [WWW] https://github.com/doublerobotics/Basic-Control-SDK-iOS. [Kasutatud 07.03.2014].

[24] „Sony Handycam HDR-XR500V 120 GB Camcorder," [WWW] http://www.ebay.com/ctg/Sony-Handycam-HDR-XR500V-120-GB-Camcorder-Black-/99971179. [Kasutatud 07.05.2014].

LISAD

Lisa 1 Roboti spetsifikatsioonid

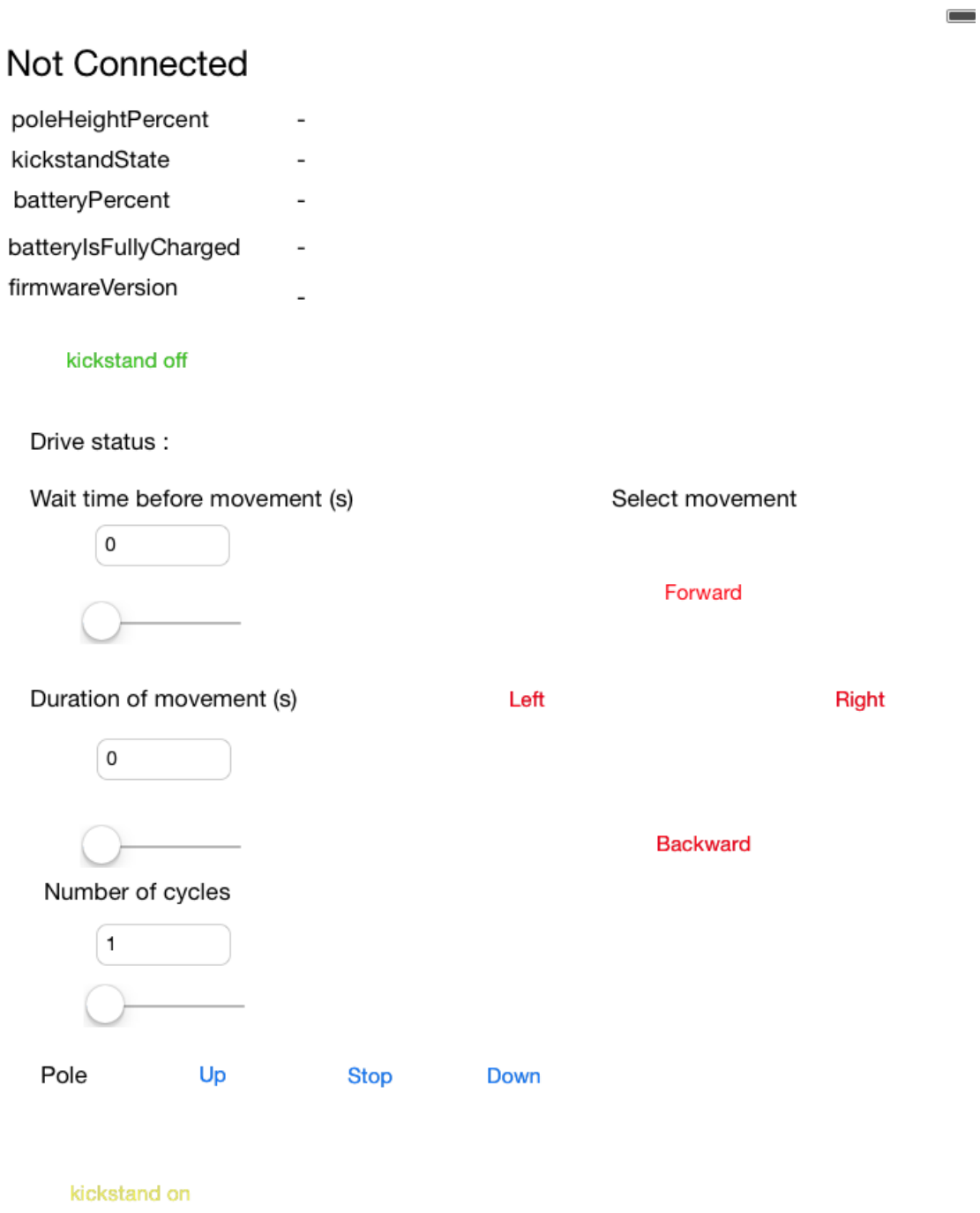
Tabel L1.1 *Double*'i teadaolevad spetsifikatsioonid

Kõrgus	47"–59" (1,2 m – 1,5 m) (kaugjuhitav)
Jalajälje suurus	10" x 9" (0,25 m x 0,23 m)
Mass	15 naela (6,8 kg)
Kiirus	Aeglane kuni mõõdukas kõndimise kiirus
Tööpind	Enamik ruumipindu, valdavalt tasane pind. Näiteks: vaip, betoon, plaadid, puitpõrandad
Kaldpind	*double* läheb üles kuni 5% kallakust
Kasutamine õues	Ei
Aku	Liitiumioon aku. Kestab 8–10 tundi ühest laadimisest. Laadimisaeg on umbes 2 tundi
Mikrofon ja kõlarid	Sõltub kasutatavast iPad-ist. Toetab iPad-i versioone 2, 3 ja 4
Kaamera, LCD displei	Sõltub kasutatavast iPad-ist. Toetab iPad-i versioone 2, 3 ja 4
Esikaamera	iPad-i esikaamera
Tahapoole vaatav kaamera	iPad-i peakaamera
Liikumise kaugus	Limiteerimata, toas on võimalik sellega liikuda igale poole senikaua, kui *double* on ühendatud juhtmevaba internetiga
Kasutajaliides	*Double*'t on võimalik liigutada iPhone'i, iPad-i või mis tahes arvuti abil doublerobotics veebisirvijast, mis töötab Mac OS X või Windowsi peal
Autonoomne käitumine	Puudub, kasutaja peab *double*'t juhtima soovikohaselt
Võrguühendus	WiFi või 4G/LTE. *Double*'il endal pole WiFid ega 4G/LTE-d. Internetiühendus on

	WiFi ruuteri ning iPad-i vahel
Videoprotokoll	Standardne WebRTC
Video krüpteering	Jah, 128-bitine AES. Video on algusest lõpuni krüptitud, mitte salvestatud ega varundatud.
iOS SDK iPad'i rakendustega suhtlemiseks	Jah
Igakuine lepingutasu	Ei
iPad	Toetab versioone 2, 3 ja 4. Tuleb soetada eraldi
Garantii	Jah, vaid tehasedefektide ilmnemisel tingimuslepingu rikkumatuse puhul

[22]

Lisa 2 Rakenduse kasutajaliides



Sele L1.1 Kasutajaliides

Lisa 3 Koodid

L3.1 Esialgne päisefaili kood

```
#import <UIKit/UIKit.h>

@interface DRViewController : UIViewController {
    IBOutlet UILabel *statusLabel;
    IBOutlet UILabel *poleHeightPercentLabel;
    IBOutlet UILabel *kickstandStateLabel;
    IBOutlet UILabel *batteryPercentLabel;
    IBOutlet UILabel *batteryIsFullyChargedLabel;
    IBOutlet UILabel *firmwareVersionLabel;
    IBOutlet UIButton *driveForwardButton;
    IBOutlet UIButton *driveBackwardButton;
    IBOutlet UIButton *driveLeftButton;
    IBOutlet UIButton *driveRightButton;
}

@end
```

L3.2 Esialgne implementeerimisfaili kood

```
//
//  DRViewController.m
//  DoubleBasicHelloWorld
//
//  Created by David Cann on 8/3/13.
//  Copyright (c) 2013 Double Robotics, Inc. All rights reserved.
//

#import "DRViewController.h"
#import <DoubleControlSDK/DoubleControlSDK.h>

@interface DRViewController () <DRDoubleDelegate>
@end

@implementation DRViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    [DRDouble sharedDouble].delegate = self;
    NSLog(@"SDK Version: %@", kDoubleBasicSDKVersion);
}

-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)toInte
rfaceOrientation
{
    return UIInterfaceOrientationIsPortrait(toInterfaceOrientation);
}

#pragma mark - Actions

- (IBAction)poleUp:(id)sender
{
```

```
    [[DRDouble sharedDouble] poleUp];
}

- (IBAction)poleStop:(id)sender
{
    [[DRDouble sharedDouble] poleStop];
}

- (IBAction)poleDown:(id)sender
{
    [[DRDouble sharedDouble] poleDown];
}

- (IBAction)kickstandsRetract:(id)sender
{
    [[DRDouble sharedDouble] retractKickstands];
}

- (IBAction)kickstandsDeploy:(id)sender
{
    [[DRDouble sharedDouble] deployKickstands];
}

#pragma mark - DRDoubleDelegate

- (void)doubleDidConnect:(DRDouble *)theDouble
{
    statusLabel.text = @"Connected";
}

- (void)doubleDidDisconnect:(DRDouble *)theDouble
{
    statusLabel.text = @"Not Connected";
}

- (void)doubleStatusDidUpdate:(DRDouble *)theDouble
{
    poleHeightPercentLabel.text = [NSString stringWithFormat:@"%f",
   [DRDouble sharedDouble].poleHeightPercent];
    kickstandStateLabel.text = [NSString stringWithFormat:@"%d", [DRDouble
    sharedDouble].kickstandState];
    batteryPercentLabel.text = [NSString stringWithFormat:@"%f", [DRDouble
    sharedDouble].batteryPercent];
    batteryIsFullyChargedLabel.text = [NSString stringWithFormat:@"%d",
    [DRDouble sharedDouble].batteryIsFullyCharged];
    firmwareVersionLabel.text = [DRDouble sharedDouble].firmwareVersion;
}

- (void)doubleDriveShouldUpdate:(DRDouble *)theDouble
{
    float drive = (driveForwardButton.highlighted) ?
    kDRDriveDirectionForward : ((driveBackwardButton.highlighted) ?
    kDRDriveDirectionBackward : kDRDriveDirectionStop);
    float turn = (driveRightButton.highlighted) ? 1.0 :
    ((driveLeftButton.highlighted) ? -1.0 : 0.0);
    [theDouble drive:drive turn:turn];
}

@end
```
[23]

L3.3 Tervikprogrammi päisefaili kood

```
#import <UIKit/UIKit.h>
#include <QuartzCore/QuartzCore.h>

@interface DRViewController : UIViewController {
    IBOutlet UILabel *statusLabel;
    IBOutlet UILabel *poleHeightPercentLabel;
    IBOutlet UILabel *kickstandStateLabel;
    IBOutlet UILabel *batteryPercentLabel;
    IBOutlet UILabel *batteryIsFullyChargedLabel;
    IBOutlet UILabel *firmwareVersionLabel;
    IBOutlet UIButton *driveForwardButton;
    IBOutlet UIButton *driveBackwardButton;
    IBOutlet UIButton *driveLeftButton;
    IBOutlet UIButton *driveRightButton;
    IBOutlet UILabel *ActionLabel;
    IBOutlet UISlider *TimeUntilStart;
    IBOutlet UISlider *TimeForAction;
    IBOutlet UITextField *Starttext;
    IBOutlet UITextField *Actiontext;
    IBOutlet UITextField *cyclenumber;
    IBOutlet UISlider *cycleslider;
}

@property(nonatomic, assign)bool IsValid;
@property(nonatomic,assign) NSString* ButtonPushed;
@property(nonatomic,assign) int counter;
@property (nonatomic, assign) double EndTime;
@property(nonatomic,assign) double sleepTime;
@property(nonatomic,retain) IBOutlet UISlider *cycleslider;
@property (nonatomic, retain)IBOutlet UITextField *cyclenumber;
@property (nonatomic, retain) IBOutlet UISlider *TimeUntilStart;
@property(nonatomic, retain) IBOutlet UISlider * TimeForAction;
@property (nonatomic, retain)IBOutlet UITextField *Starttext;
@property (nonatomic, retain) IBOutlet UITextField *Actiontext;

- (IBAction)UntilStartValueChanged:(id)sender;
- (IBAction)ActionSliderValueChanged:(id)sender;
- (IBAction)cyclechanged:(id)sender;

@end
```

L3.4 Tervikprogrammi implementeerimisfaili kood

```
#import "DRViewController.h"
#import <DoubleControlSDK/DoubleControlSDK.h>

@interface DRViewController () <DRDoubleDelegate>
@end
@implementation DRViewController
@synthesize cyclenumber, cycleslider, Actiontext, Starttext, TimeForAction,
TimeUntilStart;

- (void)viewDidLoad {
    [super viewDidLoad];
    [DRDouble sharedDouble].delegate = self;
    _IsValid = NO;
```

```
}
-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)toInte
rfaceOrientation
{
    return UIInterfaceOrientationIsPortrait(toInterfaceOrientation);
}
#pragma mark - Actions
- (IBAction)poleUp:(id)sender
{
    [[DRDouble sharedDouble] poleUp];
}
- (IBAction)poleStop:(id)sender
{
    [[DRDouble sharedDouble] poleStop];
}
- (IBAction)poleDown:(id)sender
{
    [[DRDouble sharedDouble] poleDown];
}
- (IBAction)kickstandsRetract:(id)sender
{
    [[DRDouble sharedDouble] retractKickstands];
}
- (IBAction)kickstandsDeploy:(id)sender
{
    [[DRDouble sharedDouble] deployKickstands];
}

#pragma mark - DRDoubleDelegate
- (void)doubleDidConnect:(DRDouble *)theDouble
{
    statusLabel.text = @"Connected";
}
- (void)doubleDidDisconnect:(DRDouble *)theDouble
{
    statusLabel.text = @"Not Connected";
}
- (void)doubleStatusDidUpdate:(DRDouble *)theDouble
{
    poleHeightPercentLabel.text = [NSString stringWithFormat:@"%f",
    [DRDouble sharedDouble].poleHeightPercent];
    kickstandStateLabel.text = [NSString stringWithFormat:@"%d", [DRDouble
    sharedDouble].kickstandState];
    batteryPercentLabel.text = [NSString stringWithFormat:@"%f", [DRDouble
    sharedDouble].batteryPercent];
    batteryIsFullyChargedLabel.text = [NSString stringWithFormat:@"%d",
    [DRDouble sharedDouble].batteryIsFullyCharged];
    firmwareVersionLabel.text = [DRDouble sharedDouble].firmwareVersion;
}
- (void)doubleDriveShouldUpdate:(DRDouble *)theDouble
{
    float drive;
    float turn;
    if (driveForwardButton.highlighted ||
    driveBackwardButton.highlighted||driveRightButton.highlighted
    ||driveLeftButton.highlighted)
    {
        _counter = cycleslider.value;

        if (driveForwardButton.highlighted)
```

```
        {
            _ButtonPushed = @"Forward";
        }
        else if (driveBackwardButton.highlighted)
        {
            _ButtonPushed = @"backward";
        }
        else if (driveRightButton.highlighted)
        {
            _ButtonPushed = @"Turn Right";
        }
        else if (driveLeftButton.highlighted)
        {
            _ButtonPushed = @"Turn Left";
        }
         _EndTime = (CACurrentMediaTime()+ TimeUntilStart.value) +
         TimeForAction.value;
         _sleepTime = CACurrentMediaTime() + TimeUntilStart.value;
    }
    double currentTime = CACurrentMediaTime();

    if (currentTime >= _EndTime)
    {
        if (_counter > 1)
        {
            _counter = _counter -1;
            _EndTime = (CACurrentMediaTime()+ TimeUntilStart.value) +
            TimeForAction.value;
            _sleepTime = CACurrentMediaTime() + TimeUntilStart.value;
        }
    }
    _IsValid = currentTime <= _EndTime && currentTime >=_sleepTime;
    if (_IsValid && [_ButtonPushed  isEqual: @"Forward"])
    {
        drive = kDRDriveDirectionForward;
        ActionLabel.text =  @"forward";
    }
    else if ((_IsValid && [_ButtonPushed  isEqual: @"backward"]))
    {
        drive = kDRDriveDirectionBackward;
        ActionLabel.text = @"backward";
    }
    else
    {
        drive =kDRDriveDirectionStop;
        ActionLabel.text = @"Stop";
        turn= 0.0;
    }
    if ( _IsValid && [_ButtonPushed  isEqual: @"Turn Right"])
     {
        turn= 1.0;
        ActionLabel.text =  @"Turn Right";
     }
    else if ( _IsValid && [_ButtonPushed  isEqual: @"Turn Left"])
    {
        turn= -1.0;
        ActionLabel.text =  @"Turn Left";
    }
        [theDouble drive:drive turn:turn];
}
```

```
- (IBAction)UntilStartValueChanged:(UISlider*)sender
{
    Starttext.text = [NSString stringWithFormat:@"%.0f",[sender value]];
    NSString * Starttextvalue =[ Starttext text];
    float value= [Starttextvalue floatValue];
    if (value < 0)value=0;
    if (value > 20)value=20;
    TimeUntilStart.value=value;
    Starttext.text= [NSString stringWithFormat:@"%.0f",value];
    if ([Starttext canResignFirstResponder])[Starttext
    resignFirstResponder];
}

- (IBAction)ActionSliderValueChanged:(UISlider *)sender
{
    Actiontext.text = [NSString stringWithFormat:@"%.1f",[sender value]];
    NSString * Actiontextvalue =[ Actiontext text];
    float value= [Actiontextvalue floatValue];
    if (value < 0)value=0;
    if (value > 5)value=5;
    TimeForAction.value=value;
    Actiontext.text= [NSString stringWithFormat:@"%.1f",value];
    if ([Actiontext canResignFirstResponder])[Actiontext
    resignFirstResponder];
}

- (IBAction)cyclechanged:(UISlider *)sender
{
    cyclenumber.text = [NSString stringWithFormat:@"%.0f",[sender value]];
    NSString * cyclenumbervalue =[ cyclenumber text];
    float value= [cyclenumbervalue floatValue];
    if (value < 1)value=1;
    if (value > 25)value=25;
    cycleslider.value=value;
    cyclenumber.text= [NSString stringWithFormat:@"%.0f",value];
    if ([cyclenumber canResignFirstResponder])[cyclenumber
    resignFirstResponder];
}
-(void) touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    if (Starttext)
    {
        if ([Starttext canResignFirstResponder]) [Starttext
        resignFirstResponder];
    }
    [super touchesBegan:touches withEvent:event];
    if (Actiontext)
    {
        if ([Actiontext canResignFirstResponder]) [Actiontext
        resignFirstResponder];
    }
    [super touchesBegan:touches withEvent:event];
    if (cyclenumber)
    {
        if ([cyclenumber canResignFirstResponder]) [cyclenumber
        resignFirstResponder];
    }
    [super touchesBegan:touches withEvent:event];
}
@end
```

Lisa 4 Näidistestide katseandmed

Tabel L4.1 Edasiliikumine madalas asendis

Edasiliikumine madalas asendis		
aeg [s]	nurk [°]	amplituud [mm]
0	4,05	54,24
0,25	4,40	57,81
0,5	4,75	61,44
0,75	5,17	66,85
1	4,49	57,69
1,25	3,45	44,34
1,5	3,72	48,59
1,75	1,00	12,85
2	-0,72	-9,04
2,25	0,79	9,72
2,5	1,62	20,48
2,75	2,12	26,38
3	2,74	34,32
3,25	3,43	43,11
3,5	4,01	50,43
3,75	4,35	54,24
4	4,79	59,66

Tabel L4.2 Edasiliikumine kõrges asendis

Edasiliikumine kõrges asendis		
aeg [s]	nurk [°]	amplituud [mm]
0	0,98	17,76
0,25	2,43	51,00
0,5	4,28	79,80
0,75	4,15	75,12
1	3,72	67,50
1,25	2,62	47,70
1,5	2,13	38,40
1,75	1,11	20,04
2	-2,95	-53,64
2,25	-2,67	-43,80
2,5	-2,10	-37,62
2,75	-1,03	-18,42
3	0,11	1,80

3,25	0,83	14,70
3,5	1,84	32,94
3,75	1,94	34,74
4	2,05	36,78

Tabel L4.3 Tagasiliikumine

Tagasiliikumine		
aeg [s]	nurk [°]	amplituud [mm]
0	2,26	44,88
0,25	1,29	22,53
0,5	-0,45	-8,40
0,75	-0,20	-4,26
1	0,00	0,00
1,25	0,00	0,00
1,5	1,23	23,32
1,75	2,75	51,03
2	6,18	112,24
2,25	4,85	89,64
2,5	4,35	79,23
2,75	4,83	88,73
3	3,43	62,18
3,25	3,01	54,93
3,5	2,27	41,41
3,75	1,70	31,06
4	1,23	22,59

Tabel L4.4 Liikumine vaibal

Liikumine vaibal		
aeg [s]	nurk [°]	amplituud [mm]
0	2,28	41,30
0,25	4,20	76,35
0,5	5,86	106,79
0,75	5,98	108,62
1	5,39	97,88
1,25	4,87	88,32
1,5	4,66	85,37
1,75	3,72	67,97
2	3,07	56,29
2,25	2,45	44,96
2,5	-1,14	-20,89

2,75	-1,03	19,06
3	0,76	13,92
3,25	-0,38	-7,02
3,5	1,02	18,53
3,75	1,70	31,21
4	1,90	34,75

Tabel L4.5 Liikumine kahekraadisel kaldpinnal

Liikumine kahekraadisel pinnal		
aeg [s]	nurk [°]	amplituud [mm]
0	3,06	55,50
0,25	4,60	83,46
0,5	6,44	117,42
0,75	6,55	119,40
1	6,21	113,88
1,25	5,50	100,38
1,5	4,83	88,38
1,75	4,19	76,74
2	3,52	64,32
2,25	3,66	66,42
2,5	3,28	59,40
2,75	1,11	20,04
3	-0,28	-4,92
3,25	1,72	30,78
3,5	1,22	21,78
3,75	2,09	37,26
4	4,20	75,54
4,25	4,28	76,62
4,5	4,45	79,62
4,75	5,30	94,56
5	5,07	91,02

Tabel L4.6 Liikumine viiekraadisel kaldpinnal

Liikumine viiekraadisel pinnal		
aeg [s]	nurk [°]	amplituud [mm]
0	2,44	44,39
0,25	3,45	62,95
0,5	5,19	94,69
0,75	5,35	100,60

1	4,98	96,70
1,25	4,31	79,18
1,5	3,95	72,75
1,75	3,46	64,25
2	3,10	57,57
2,25	2,63	49,19
2,5	0,54	10,19
2,75	-1,15	-21,48
3	0,60	11,29
3,25	-0,14	-2,60
3,5	1,04	19,47
3,75	1,99	37,32
4	2,51	46,99
4,25	3,26	60,36
4,5	3,31	61,98
4,75	3,41	64,19
5	3,87	72,69

Tabel L4.7 Liikumine seitsmekraadisel kaldpinnal

Liikumine seitsmekraadisel pinnal		
aeg [s]	nurk [°]	amplituud [mm]
0	2,88	52,66
0,25	3,88	70,79
0,5	5,42	99,09
0,75	5,48	100,17
1	5,46	99,72
1,25	4,95	92,54
1,5	4,40	82,62
1,75	4,23	80,01
2	3,59	67,67
2,25	3,59	67,73
2,5	1,66	31,35
2,75	0,00	0,00
3	1,21	22,51
3,25	0,62	11,77
3,5	1,72	32,05
3,75	2,34	43,82
4	2,86	53,61
4,25	3,83	71,42
4,5	3,90	72,89
4,75	4,02	74,22
5	4,14	76,00

Lisa 5 Näidistestide jaoks kasutatud kaamera parameetrid

Tabel L5.1 Näidistestide jaoks kasutatud kaamera parameetrid

Optiline sensor	Exmor R CMOS 1/5,08 cm
Sensori suurus	1/2,88'' (1/5,08 cm)
Lääts	12x f1.8-3.4 43 - 516mm (16:9)
Fokaalkaugus	5,5 mm-66 mm
Läätse filtri suurus	37 mm
Fookuse kohaldamine	Automaatne ja manuaalne
Resolutsioon	6,6 MP
Optiline zoom	12x
Digitaalne zoom	150x
Maksimaalne katikukiirus	1/800 s
Minimaalne katikukiirus	1/8 s
Minimaalne valgustatus	3 lux
Säriviis	Automaatne, programmile vastav
Välgu tüüp	Sisseehitatud välk

[24]