TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Rain Vink 176843

Juhtimissüsteem TalTech-i satelliidi maajaama Ku riba antennile

Magistritöö

Juhendaja: Evelin Halling
MSc

Tallinn 2019

Autorideklaratsioon

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

Autor: Rain Vink

07.05.2019

Annotatsioon

Käesoleva magistritöö eesmärgiks on luua Tallinna Tehnikaülikooli poolt loodavale parabool antennile osa juhtimissüsteemist, mille abil oleks võimalik tulevikus jälgida orbiidil tiirlevaid satelliite. Tegemist on edasi arendusega Rasmus Tomseni bakalaureuse tööst "Juhtimissüsteemi loomine TTÜ satelliidi maajaam paraboolantennile".

Töö tulemusena valmis Python-is kirjutatud ROS-i sõlmedena osaliselt realiseeritud rakendus, mille abil on võimalik keerata antenni küljes olevat horisontaalset ja vertikaalset mootorit selliselt, et antenn oleks suunaga alati jälgitava satelliidi poole juhul, kui vaadeldav objekt on vaateväljas. Koodi kirjutamise ja testimise kõrval tuli lisaks tegeleda ka analüüsiga. Tähtsal kohal oli komponentide arhitektuuri paika panek, dokumentatsioonide lugemised ja veaparanduse mudeli loomine. Juhtimissüsteemi algne nõue oli, et lõpliku süsteemi juhtimise täpsus peab jääma ±0.5 kraadi sisse.

Lõputöö on kirjutatud keeles ning sisaldab teksti 36 leheküljel, 9 peatükki, 17 joonist, 4 tabelit.

Abstract

Guidance System for the TalTech Ku Band Antenna

The aim of this master's thesis is to create a guidance system for the parabolic antenna created by the Tallinn University of Technology, which would allow to observe orbiting satellites. Current thesis is a further development of Rasmus Tomsen's bachelor's thesis "Writing a steering system for the TTU satellite ground station parabolic antenna".

As a result of this work, a partially realized application in the form of ROS nodes, written in Python, is used to turn the horizontal and vertical motors on the antenna so that the antenna is always in the direction of the satellite in case the satellite is in sight. In addition to writing and testing the application, there was also a need for analysis. For example, creating an architecture for the ROS nodes, understanding devices and protocol documentations and how to do correction calculations during the antenna movement. The initial requirement of the control system was that the accuracy of the final system must be within ± 0.5 degrees.

The thesis is in and contains 36 pages of text, 9 chapters, 17 figures, 4 tables.

Lühendite ja mõistete sõnastik

1U	*One unit* satelliidi standardsuurus
NIR	*Near-infrared* infrapuna spekter
RGB	*Red-Green-Blue* puna-roheline-sinine spekter
AC	*Alternating current* vahelduvvool
SSI	*Synchronous Serial Interface* kommunikatsiooni protokoll
USB	*Universal Serial Bus* universaalne järjestiksiin
PID kontroller	*Proportional–integral–derivative controller* proportsionaal-integraal-diferentsiaalregulaator
RPM	*Revolutions per minute* pöördeid minutis
JSON	*JavaScript Object Notation* andmevahetusvorming
API	*Application Program Interface* rakendusliides
CSV	*Comma-separated values* komadega eraldatud väärtused
MVC	*Model-View-Controller* arhitektuuri muster

Sisukord

Jooniste loetelu

Tabelite loetelu

1. Sissejuhatus

Aastal 2014 alustati koostöös Tallinna Tehnikaülikooli ja Mektory-iga satelliidi programmi rajamisega. Järgnevatel kahel aastal tegeleti missiooni ja tehniliste nõuete paika panemise ning peale selle alustati ka satelliidi ehitamisega. Projekti viimane faas peaks lõppema 2019. aasta suvel, kui Roscosmose rakett Sojuz toimetab antud tehiskaaslase maalähedasele orbiidile [1].

Missiooni peamiseks eesmärgiks on toimetada orbiidile 1U nanosateliit, mille abil oleks võimalik teostada Maa kaugseiret, pidada kõrgsagedusalas olevat andmesidet ning demonstreerida kõige sellega seonduvat tehnoloogiat. Satelliidile on paigaldatud kahte erinevat tüüpi kaamerat: NIR [2] ja RGB [3]. NIR-i abil on võimalik kätte saada infot taimestiku ja kliima kohta ning RGB kaudu värvilisi pilte maast. Kõige selle juures tuleb lisaks tegeleda veel pilditöötlus ja kommunikatsioonitehnoloogiaga [4].

1.1 Probleem

Käesolev töö on edasiarendus Rasmus Tomseni lõputööst “Juhtimissüsteemi loomine TTÜ satelliidi maajaama paraboolantennile” ning kogu probleem saab alguse eelnevas peatükist nimetatud RGB ja NIR kaameratest. Näiteks on RGB kaamera ülesandeks teha kõrge lahutusvõimega pilte, mille ühele pikslile peab vastama maapinnal 50m korda 50m [5]. Foto tegemine iseenesest ei ole raske tegevus, kuid antud probleem esineb selles, mis moodi ja kui kiiresti oleks võimalik satelliidi mälus olevaid pilte maale toimetada. Keerukust lisab omakorda tehiskaaslase pidev liikumine, lühike ülelennu aeg Eesti kohal [6], pildi kvaliteedi varieeruvused ja keskkonnast tulenevad erinevad mõjutus tegurid nagu tuul, vihm ja lumi.

Probleemi lahendamiseks on planeeritud kahte erinevat tüüpi lahendust. Esiteks paigaldatakse satelliidi pardale pilditöötlusprogramm, mis peaks suutma prioritiseerida pildi saatmise järjekorda vastavalt kvaliteedile. Teise lahendusena paigaldatakse Mektory maja katusele kahte erinevat tüüpi antenni: Yagi [7] ja Ku riba [8] paraboolantenn. Yagi antenn töötab 435 MHz sagedusalas ning tema peamisteks eesmärkideks pidada kahepoolset sidet maajaama ja vaadeldava objekti vahel. Näiteks saada andmeid satelliidi põhiandmete ja alamsüsteemide kohta ning edastada maajaamast ees ootava missiooni infot [4]. Yagi antenni eelis seisneb selles, et ta ei nõua väga suurt jälgimise täpsust, kuid selle arvelt on ka andmevahetus kiirus madal. Seoses antud negatiivse omadusega, ei ole mõistlik seda tüüpi antenni kasutada mahukate andmete saatmiseks. Selleks võetakse kasutusel Ku riba parabool tüüpi antenn, millel on kordades parem andmevahetus kiirus. Vastupidiselt Yagi antennile, vajab Ku riba paremat jälgimise täpsust. Täpsuse nõue tuleneb sageduse omadustest [9] [10]. Paraboolantenn töötab 10.5 GHz sagedusalas. Selle taldriku diameeter on 5 meetrit ja kõrgus koos jalgadega 6 meetrit. Kogu kaal jääb umber 8 tonni juurde. Lisaks on antennil kaks liikuvat osa: jalg ja taldrikut ühendav lüli. Jalga on võimalik pöörata 360 kraadi ümber oma telje ning lüli on võimalik liigutada maa suhtes 180 kraadi [11].

Joonis 1. Antenni mõõtmed

1.2 Eesmärgid

Eesmärke võib jagada laiemateks ja kitsamateks. Laiemate eesmärkide all on mõeldud paraboolantenni projekti ja selle juhtimissüsteemi tervikuna ning kitsamate eesmärkide juures on keskendutud lõputöö skoobile.

1.2.1 Laiemad eesmärgid

Eelmises peatükis sai välja toodud probleemi tuum. Selleks, et kätte saada orbiidil olevalt satelliidilt mahukaid pildi andmeid, läheb meil vaja suure andmevahetuse kiirusega kanalit. Kanali loomisel kasutame paraboolantenni Ku sagedusalas.

Alguse sai projekt detailse planeeringuga. Pandi paika asukoht, vajaminevad tegevused ja komponendid ning tänaseks on jõutud faasi, kus algust on tehtud monteerimise ja juhtimissüsteemi loomisega. Lõputöö vältel keskendutakse just selle viimasele. Juhtimissüsteemi peamised eesmärgid on järgnevad [10]:

- Süsteem peab suutma arvutada vaadeldava objekti asukoha antud aja-hetkel
- Süsteem peab suutma käivitada antenni küljes olevaid mootoreid niimoodi, et antenni taldrikud oleks suunatud vaadeldava objekti poole juhul, kui antud objekt on vaateväljas
- Süsteem peab looma võimaluse satelliidilt andmete alla laadimiseks
- Süsteem peab aru saama väliskeskkonna teguritest ning vastavalt sellele tegema korrektuure, et tagada antenni ohutus

1.2.2 Kitsamad eesmärgid

Kuna projektil on väga suur skoop ning lisaks sellele on see seotud väga erinevate teadusvaldkondadega, siis üks inimene ei suuda seda kõike üksi ära teha. Seetõttu, keskendutakse lõputöö raames teemadele, mille tulemusel pannakse alus paraboolantenni juhtimissüsteemile, et oleks võimalik juhtida antenni küljes olevaid mootoreid selliselt, et neid oleks võimalik keerata vastavalt asimuudile [12] ja kõrgusnurgale.

Täpsemalt võetakse vaatluse alla järgnevad teemad:

- Süsteem peab suutma muundurite (Omron MX2) [13] väljunditest lugeda ja registritesse kirjutada informatsiooni, et mõjutada mootorite käitumist
- Panna paika ROS-i [14] sõlmede arhitektuur, leida ühisosa Yagi ja paraboolantenni komponentide vahel
- Panna alus ROS-i sõlmedele ja nende vahelisele vestlusele
- Süsteem peab suutma lugeda sensoritelt [15] mootori seisundi kohta tulenevaid andmeid

1.3 Töö etapid ja valideerimine

Nagu eelnevalt sai mainitud, siis kogu projekti skoop on väga suur ning hõlmab endas erinevaid valdkondi ja inimesi. Selle tõttu on jaotatud lõputöö erinevateks etappideks, et oleks võimalik hinnata täpselt, kui palju on võimalik mingil kindlal ajahetkel tööd ära teha.

Esimeses etapis on planeeritud luua Python-i programm, mida kasutatakse mootori tööle panekuks. Programmi abil peab olema võimalik lugeda muundurilt väljundite ja registrite informatsiooni ning võimaldama neid ka muuta. Väärtuste muutmise korral, peab muundur mootoritesse edastama signaali pöörlemise kohta. Suhtlemiseks on plaanitud kasutada Modbus-i [16] protokolli. Siinkohal tuleb oluliselt rõhku panna muunduri dokumentatsiooni uurimisele ja arusaamisele.

Teises etapis tuleb tegeleda juhtimissüsteemiga terviklikumalt. Panna alus arhitektuurile, leida ühisosa Yagi ja paraboolantenni komponentide vahel ning panna paika nendevaheline suhtlus reeglid. Täpsemini, kuidas toimub andmevahetus ROS-i sõlmede vahel, mismoodi sõlmi registreeritakse ning mis andmeid sõlmed omavahel vahetavad.

Viimases faasis on planeeritud tegeleda mootori sensoritega. Nende eesmärk on anda informatsiooni antenni hetke positsiooni kohta. Vastavalt asukoha andmetele on võimalik teostada veaparandust. Juhul, kui selliseid andmeid ei esine, pole võimalik selgeks teha, kuhu poole peavad mootorid ennast pöörama. Siin etapis tuleb informatsiooni lugemine teha automaatseks ning integreerida see juba olemasolevasse süsteemi.

Kuna tegemist on praktilise lõputööga, siis tulemusi saab valideerida komponentide töötamise järgi. Näiteks, kui esimeses etapis on võimalik mootoreid liikuma panna, siis võib öelda, et on esimese etapi eesmärk on saavutatud.

2. Kasutatud tehnoloogiad ja seadmed

2.1 ROS

ROS ehk Robot Operating System on Python-is kirjutatud raamistik, mis hoiab endas erinevaid robootikaga seotud teeke ja tööriistu. ROS-i arhitektuur sarnaneb väga paljuski mikroteenustele. Mõlemal on olemas loogiliselt eraldatud teenused või ROS-i mõistes sõlmed, mille ühe maha kukkumisel peaksid teised toimima. Mustriliselt on ROS jaotatud kolme erineva elemendi vahel: topikud, teenused ja meetodid. Topik on vahelüli erinevate sõlmede vahel, millesse võib kirjutada ja sealt lugeda informatsiooni igal ajahetkel. Teenuseid kasutatakse andmete kauglugemiseks ja olukordades, kus ei ole vaja pikaajalist käivitamist. Meetodite kaudu peab olema võimalik käivitada juhitava keha tegevusi ja vastavalt sellele saada pidevalt tagasisidet. Seoses sellega, et kogu projekti vältel tegeletakse väga erinevate seadmete liigutamisega ja ROS-i arhitektuur, teegid ja tööriistad lihtsustavad oluliselt töö protsessi, siis ka antud lõputöö käigus kasutatakse samu vahendeid.

2.2 Modbus

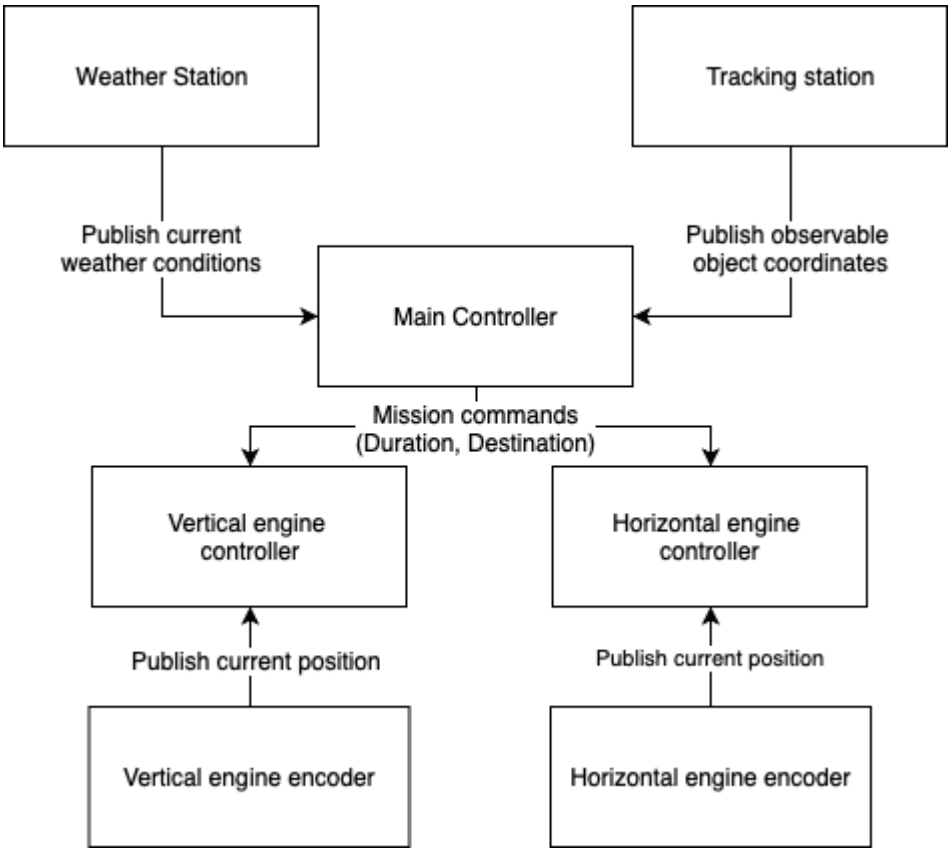
Modbus on jadaside protokoll, mida kasutatakse erinevate tööstusseadmete ühendamiseks. Teabe vahetuseks kasutatakse sõnumeid. Sõnumi struktuuris on olemas alati neli peamist komponenti: seadme aadress, funktsiooni kood, andmed ja veakontroll. Seadme aadressid jagunevad omakorda väljunditeks, sisenditeks ja hoiatavateks registriteks. Nendele kolmele on jagatud seadmes oma mälu aadressi vahemikud. Kõige populaarsemad funktsioonid on näiteks loe väljunditest ja registritest. Nendele vastavad funktsiooni koodid 01 ja 03 [16].

2.3 Omron 3G3MX2-A4007-E

Omron 3G3MX2-A4007-E on sagedusmuundur, mille abil on võimalik muuta antenni mootorite kiirust ja suunda varieerides väljund sagedust. Kõik antenni mootorid kasutavad käivitamiseks vahelduvvoolu (AC). Antud seadme mälus ei hoita funktsionaalsust, mida kasutatakse antenni liigutamiseks. Põhiline informatsioon peab tulema ühelt ROS-i sõlmelt. Arvuti ja muunduri vahel suhtlemiseks kasutatakse Modbus-i protokolli.

3. Arhitektuur

Antud peatükis räägitakse lähemalt paraboolantenni tarkvaralistest komponentidest, mis peavad olema realiseeritud ROS-i sõlmedena. Lisaks sellele tuuakse välja ka ühised komponendid Yagi antenni juhtimissüsteemiga. Järgneval joonisel on välja toodud laiemalt kasutatavad komponendid ning nende omavaheline suhtlus.



Joonis 2. Laiendatud komponentide arhitektuur

3.1 Ilmajaam

Ilmajaama peamine funktsioon on edastada keskkonnaga seotud tegureid. On olemas mitmeid ilmastiku tingimusi, mis võivad kahjustada antenni ennast ning tema kõrval olevaid objekte. Näiteks: lumi, jää, tugev tuul jne. Selleks, et tõsta paraboolantenni enda ja tema ümbruses olevate objektide turvalisust peab põhi kontroller arvestama ilmastiku tingimustega. Tugeva tuule korral tuleb pöörata antenni tagumine külg tuule puhumise suunas, et ei tekiks tuuletaskut. Lumesaju korral tuleb antenn viia vertikaalasendisse, selleks et lumi ei koguneks taldrikule. Vastasel korral võib kogu konstruktsioon kaaluda üle 11-ne tonni, mis omakorda võib lõppeda antenni või tema all oleva Mektory hoone hävinemisega. Kuigi antud lõputöö käigus ei tegeleta ilmajaamast saadetud andmetega, tuleb siiski arhitektuuriliselt ja edasises töös ka sellega arvestada. Seda komponenti kasutab samuti Yagi antenn.

3.2 Jälgimisjaam

Jälgimisjaama peamine eesmärk on edastada jälgitava objekti asukoha informatsiooni. Täpsemalt asimuuti, elevatsiooni ja kellaaega. Asimuut tähendab nurka põhjasuuna ja objekti vahel ja elevatsioon on nurka horisondi ja objekti vahel. Nende andmete saamiseks kasutatakse Python-i moodulit nimega „Pyephem“[1], mis omakorda asub eraldi seisvas ROS-i sõlmes. Antud valiku kohta täpsemalt võib lugeda Rasmus Tomsen-i lõputööd nimega “Juhtimissüsteemi loomine TTÜ satelliidi maajaam antennile”. Valiku peamiseks põhjuseks oli autori sõnade järgi lihtne kasutatavus ja parem arvutamise täpsus. Sarnaselt ilmajaama sõlmele kasutatakse ka seda Yagi puhul.

3.3 Vertikaalse ja horisontaalse mootori sensorid

Antud sõlmede eesmärkideks on edastada antenni mootorite hetke olekud, ehk teisisõnu, mis positsioonis mootori küljes olev võll asub. Väljund väärtus jääb vahemikku 0 kuni 360 kraadi. Selleks, et antud sõlmed oleks võimalik realiseerida tuli kõigepealt uurida, millist sideprotokolli ja seadmeid on mõtet kasutada. Vastavalt sellele informatsioonile sai ära tellida vajalikud komponendid. Valikus oli SSI

[1] https://rhodesmill.org/pyephem/

(Synchronous Serial Interface) ja IO-Link. Otsus valiku kohta on kirjeldatud ära peatükis 5.

3.4 Peamine kontroller

Peamise kontrolleri põhiliseks eesmärgiks on ilmajaamast ja jälgimisjaamast tulenevat informatsiooni, prioritiseerida, tõlgendada need missiooni käskudeks ning õigel hetkel edastada need mootorite sõlmedele. Vahetult enne missiooni informatsiooni esitamist, peab kontroller pöörama antenni missiooni algus punkti. Siinkohal tuleb mainida, et peamine kontroller ei tohi mootoritele edastada ebarealistlike käske. Näiteks asimuut võib arvutusvea tõttu olla 500 kraadi. Sellistest andmetest tulenevad sageduse arvutused võivad lõppeda kõigi seadmete kahjustusega.

3.5 Vertikaalse ja horisontaalse mootori kontroller

Nende sõlmede põhiliseks ülesandeks on kirjutada vastava mootori muunduri registritesse ja väljunditesse väärtusi, mis paneb antenni õigesti liikuma. Lisaks sellele, tuleb pööramise käigus teostada ka veaarvutust. Muundur on ühendatud arvuti külge USB kaudu ning kommunikatsiooni protokolliks on Modbus. Muunduril on mälus kahte tüüpi asukohti, kuhu on võimalik erinevaid väärtusi kirjutada: väljundid (coils) ja registrid (holding registers). Selleks, et aru saada, kuhu midagi kirjutama peab, tuleb lugeda Omron-i dokumentatsiooni, milles on ära kirjeldatud kasutusel olevad mälu aadressid ning mida on nendega võimalik teha.

4. Analüüs

Käesolevas peatükis räägitakse täpsemalt lõputöö teoreetilisest poolest. Peamiseks eesmärgiks on siinkohal vastata, miks lõppkokku võttes sai disainitud vastav antenni juhtimissüsteemi mudel ning mis valikuid tuli selleks teha.

4.1 Eesmärk

Juhtimissüsteemi eesmärgi võib kokkuvõtvalt sõnastada järgmiselt. Süsteem peab suutma muuta antenni suunda selliselt, et antenn oleks suunatud alati välja valitud orbiidil oleva satelliidi poole niimoodi, et oleks võimalik kätte saada tehiskaaslase poolt saadetud signaale. Selleks, et panna kokku esialgne juhtimissüsteemi mudel, tuleb jaotada see kõige pealt loogiliselt eraldatavateks komponentideks. Nendeks on olemas mootori kontrollerid, peamine kontroller ja kooderid. Süsteemi sisendiks on ilmajaama ja missiooni andmed ning väljundiks muundurile edastatav sagedus. Selleks, et suurendada suunamise täpsust, annab meile tagasisidet mootori küljes olevad andurid, mis mõõdavad mootori küljes oleva võlli nurka null punktist. Nullpunkt määratakse mõlemale mootorile kalibreerimise käigus.

4.2 Peamine kontroller

Peamise kontrolleri eesmärgiks on võtta vastu missiooni sõnumeid juhtimise jaamalt, transleerida neid ning edastada käsud õigeaegselt mootorite kontrolleritele. Tegemist on puhtalt sisend-väljund tüüpi mudeliga, kus kasutatakse ainult staatilisi valemeid ning teatud piiranguid ohutuse tagamiseks.

4.2.1 Sisend

Mudeli sisendiks on kollektsioon satelliidi positsioonidest kindlatel ajahetkedel. Vajalikuks informatsiooniks osutub ainult antenni asimuut ja elevatsioon kraadides, ajaline kulu millisekundites ning käsu käivitamise aeg. Näide juhtimise jaamast saadav osaline väljud on kirjeldatud järgnevas tabelis.

Tabel 1. Peamise kontrolleri sisend

Satellite time	Satellite azimuth	Satellite elevation	Antenna azimuth	Antenna elevation	Turning time
2018/5/19 10:08:13	3.24060726166	4.86671478939e-06	3.24060726183	2.94457386394e-05	5
2018/5/19 10:08:18	3.24431681633	0.00434078834951	3.24060726183	2.99935702355e-05	5
2018/5/19 10:08:23	3.24810791016	0.00882132351398	3.2453796458	0.00318863067705	5
2018/5/19 10:08:28	3.25198340416	0.01344090979555	3.24893480696	0.007664322129	5

4.2.2 Väljund

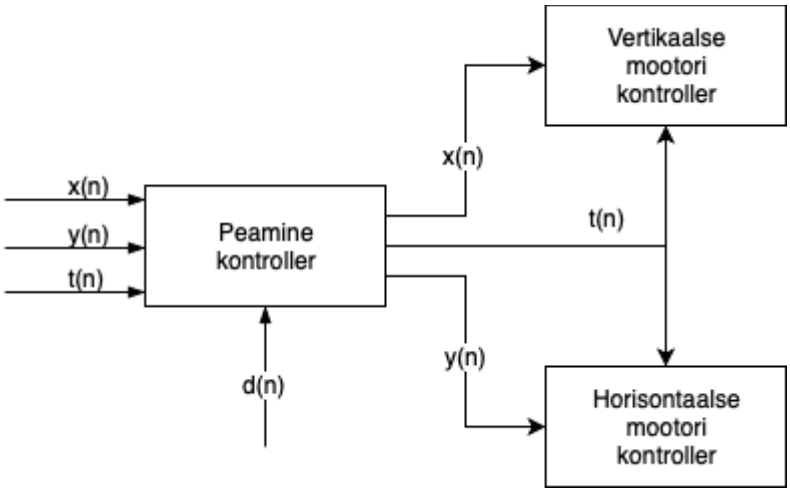
Antud mudelil on väljundeid horisontaalsele ja vertikaalsele mootori kontrollerile. See tingitud asjaolust, et mõlemal kontrolleril on erinevad piirangud asimuudi ja elevatsiooni osas. Mõlema mootori väljundi struktuur on ühesugune ning on kirjeldatud järgnevas tabelis.

Tabel 2. Peamise kontrolleri väljund

Target (kraadides)	Duration (ms)
3.24060726183	5
3.24060726183	5
3.2453796458	5

4.2.3 Valemid ja piirangud

Järgneval joonisel on välja toodud kontrolleri sisendid ja väljundid ning kuhu neid edasi saadetakse.



Joonis 3. Peamise kontrolleri mudel

Nagu joonisel näha on peamisel kontrolleril neli sisendit. x(n) tähistab asimuuti, y(n) elevatsiooni, t(n) kellaaega ja d(n) pööramise aega. Peamine kontroller, ei muuda neid andmeid mitte ühelgi kujul vaid tegeleb erinevate missiooni käskude kokkupanemise ja edastamisega. Siinkohal tuleb lisada, et peamine kontroller peab edastama missiooni käsud juhul kui hetke aeg võrdub d_1-ga ja kui juhtimisjaamalt tulevad andmed on korrektsed. Lisaks sellele ei tohi kontroller edasta käske, kui antenni positsioon ei ole lähtestatud missiooni esimese käsu järgi. Kontrolliks kasutatakse järgnevaid valemeid:

$0 \leq x(n) \leq 360$

$0 \leq y(n) \leq 180$

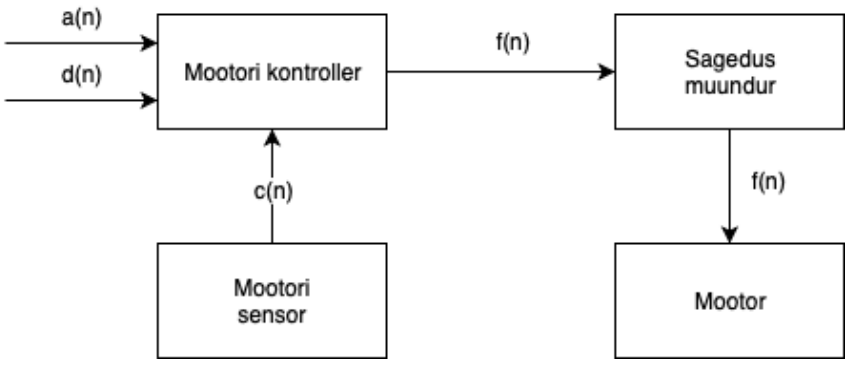
4.2 Mootori kontroller

Mootori kontrolleri ülesandeks on võtta peamiselt kontrollerilt vastu missiooni käske, teisaldada need muundurile sobivaks väärtuseks ning teostada vea parandust. Siinkohal oli võimalik kasutada kahte erinevat tüüpi juhtimise mudeleid: sisend-väljund ja tagasisidestatud.

4.2.1 Juhtimismudel

Sisend-väljund tüüpi mudeli korral on tegemist väga lihtsa juhtimismudeliga. Sisendiks võetakse soovitav pöörde asukoht, hetke positsioon ja kestvus. Nende kolme sisendi pealt saame vastavalt arvutada väljundsageduse, mida muundur mootorile rakendama peab. Kuid antud mudel töötab ainult ideaalsetes tingimustes. Selleks, et sisend-väljund tüüpi mudel töötaks, peab muundurisse saabuvate käskude viivitused olema alati ühesugused, käsu lõppedes hetke positsioon vastama alati soovitud tulemusega, mootorite pöörded saavutatakse alati konstantse asjaga, sageduse arvutamisel ei lähe vaja täiendavat kalibreerimist ning antenn asub keskkonnas, kus talle ei mõju ilmastiku ja mootori käivitamisel tekkivad jõud.

Seoses eelnevalt nimetatud tingimustega, läheb mootori kontrolleril alati vaja sisendit, mis ütleb, kuhu poole antenn parajasti on suunatud. Tänu sellele on ka võimalik garanteerida, et juhtimise täpsus jääb alati pluss miinus 0,5 kraadi sisse. Selleks, et aru saada antenni positsioonist, on iga mootori võlli külge paigaldatud andurid. See tingib omakorda olukorra, kus tegemist on tagasisidestatud juhtimissüsteemiga.



Joonis 4. Mootori kontrolleri mudel

4.2.1 Mootori kontroller

Mootori kontrolleril on kolm sisendit ja üks väljund. Sisenditeks on peamise kontrolleri väljundid: a(n) soovitud positsioon ja d(n) pööramise kestus ning mootori sensorilt tulev pöördenurk c(n). Kontrolleri väljundiks on mootori sagedus f(n) hertsides. Selleks, et mudel annaks kõige parema pööramise täpsust ilma korrektuure tegemata, tuleb sagedus arvutamisel arvestada erinevate teguritega: ülekande parameetritega, kiirenduse ning viivitusega. Ülekande parameetrite puhul on mõeldud antenni ja mootori vahel ülekandest tingitud erinevused, mille tingib antenni ja reduktori hammasratastel olevad hammaste arvud. Need numbrid on horisontaalse ja vertikaalse kontrolleri puhul erinevad. Kiirenduse all tuleb tähelepanu panna kahele erinevale juhtumile. Esiteks, kiirendust nullist kuni ette antud sageduseni. Teiseks, kiirendust ühelt sagedusest teisele. Reaalselt ei tohiks see viimane väga palju probleeme tekitada, kuna sagedus erinevused ühelt olekult teisele ei ole väga suured ning tingituna sellest peaks ajaline kulu olema väike.

4.2.1 Parameetrid

Nagu igal asjal on ka järgneval mudelil teatud tingimused ja piirangud, mida tuleb arvesse võtta. Kõige tähtsam nendest on sagedusmuundurile edastatav väärtus f(n). Kui väärtus on liiga väike, ei suuda mootor vähese jõu tõttu ennast liigutada. Vastupidiselt, kui sagedus on piisavalt suur, võib mootor ennast või antenni inertsist ära lõhkuda. Selleks, on muunduri registrites ära kirjeldatud kiirenduse ja aeglustuse aeg ning maksimum ja miinimum sagedus. Tähistame sagedust f-g (Hz), kiirenduse aega at-ga (s) ja aeglustuse aega st-ga (s). Selle põhjal saame välja kirjutada järgnevad tingimused. Sagedus tingimuse eiramisel mootorit tööle ei panda.

$at = st = 3.5s$
$0Hz \leq f \leq 20Hz, fmin = 0, fmax = 60$

4.2.2 Sagedus arvutused

Sageduse arvutamisel peame lähtuma eelkõige mootorist endast. Nimelt läheb vaja teada, mitu pööret minutis teeb mootor 50 Hz juures. Nimetame seda pöörete arvu muutujaga r ja sagedust f.

$r = 935RPM$
$f = 50Hz$
$t = 60s$
$r = f$ = t

Välja kirjutatud seose põhjal on võimalik küll tuletada pöörete arvu teiste kiiruste juures, aga käesoleva ülesande juures ainult sellega piirduda ei saa. Nimelt sai eelnevalt mainitud, et sageduse arvutamisel tuleb meil arvestada ka ülekande parameetritega, mille seavad antenni ja reduktor küljes olevad hammasrattad. Järgnevad read kirjeldavad ära elevatsiooni komponentide ülekande suhet ehk teisisõnu mitu pööret peab tegema mootor selleks, et teine komponent saaks teha ühe täis pöörde.

$Reduktoriesimeneaste: 14{,}5: 1$
$Reduktoriteineaste: 72: 1$
$Reduktorilõppülekanne: 72 * 14{,}5 = 1044: 1$
$Hammasülekanne: 3: 1$
$Kokku: r = 1044 * 3 = 3132pööret$

Samad arvutused ka rotatsiooni kohta.

$Reduktoriesimeneaste: 14{,}5: 1$
$Reduktoriteineaste: 72: 1$
$Reduktorilõppülekanne: 72 * 14{,}5 = 1044: 1$
$Hammasülekanne: 3{,}14286: 1$
$Kokku: r = 1044 * 3{,}14286 = 3281{,}15pööret$

Nende suhete põhjal saame välja arvutada, mitu pööret peab mootor tegema selleks, et saavutada õige positsioon. Lahenduse käik seletatakse ära järgneva näidis ülesandega. Oletame, et missiooni käigus tuleb keerata rotatsiooni käigus antenni 180 kraadi 140 sekundi jooksul.

$a = 180\ kraadi$
$t_1 = 140\ sekundit$
$r = 3281{,}15\ pööret$

Teame, et 60 sekundi jooksul suudab mootor 50 Hz juures läbid 935 pööret. Selleks, et kasutada seda võrdlust tuleb meil teada saada mitu pööret peab mootor tegema 60 sekundiga.

$$c = \frac{r}{t_1 : 60}$$
$$c = \frac{3281{,}15}{140 : 60}$$
$$c = 1406{,}21 pööret$$

Lisaks sellele, on ülesandes nõutud, et antenni tuleb liigutada 360 kraadi.

$$d = \frac{c}{360 : a}$$
$$d = \frac{1406{,}21}{360 : 180}$$
$$d = 703{,}11 pööret$$

Nüüd, kui on teada mitu pööret peab mootor läbima 60 sekundi jooksul, saame arvutada ülesandest lähtuvalt muundurile edastatava sageduse kasutades eelnevalt mainitud sõltuvust r = f = t.

$$50Hz = 935RPM$$
$$x = 703{,}11RPM$$

Sageduse arvutamiseks tuleb teha ristkorrutis.

$$x = 703{,}11 * 50 : 935$$
$$x = 37{,}60Hz$$

Näite ülesande põhjal saab välja kirjutada lõpliku sagedus arvutuse valemi mõlema mootori jaoks.

$$x = \left(\frac{\frac{r}{t_1 : 60} * 1}{360 : a} * 50 \right) : 935$$

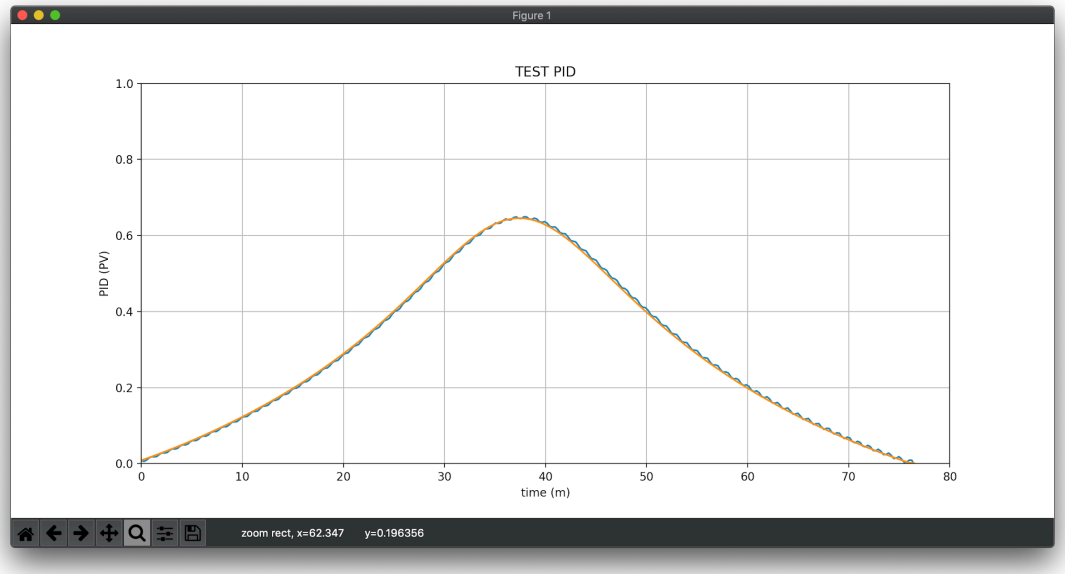
4.2.2 Vea parandus

Pööramise käigus tekkinud ala või üle pööramise viga parandatakse PID-i [17] kontrolleriga ehk proportsionaal-integraal-diferentsiaal regulaatoriga. Viga võib tuleneda käsu edastus viivitusest või mootori kiirendusest. PID-i valiku põhjus seisnes selles, et tegemist on ühe populaarseima ja ennast tõestanud lahendusega. Antud kontroller kasutab korrektuuri väärtuse arvutamiseks järgnevat valemit:

$$u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(t')\,dt' + K_\mathrm{d} \frac{de(t)}{dt}$$

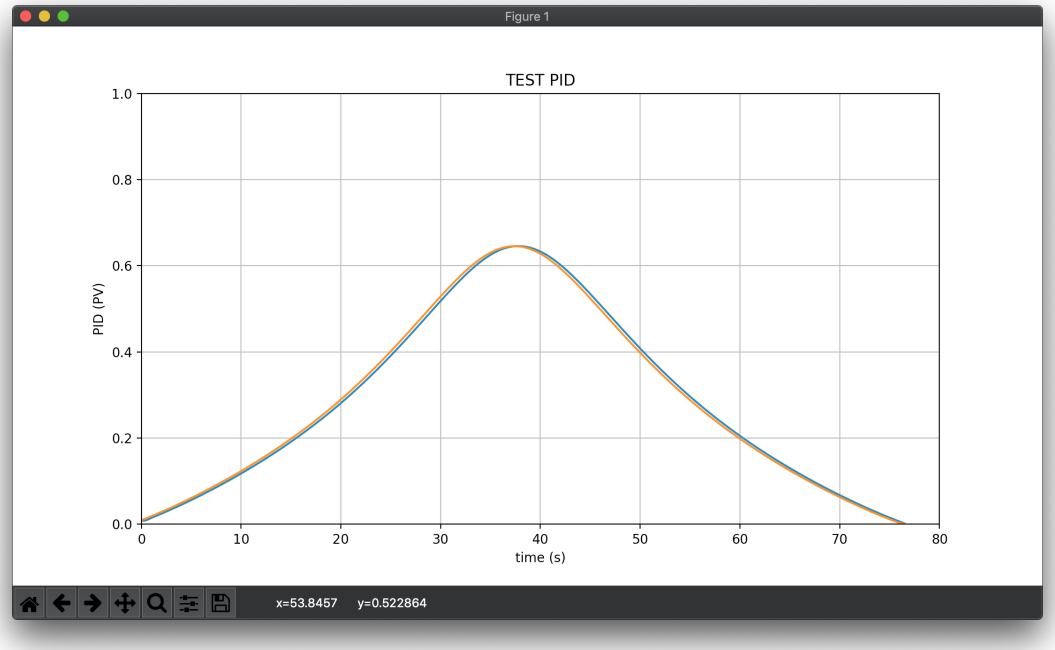
Joonis 5. PID-i algoritm

Missiooni käskude täitmise aeg on määratud piisavalt lühikesteks otsadeks, et tagada 0.5 kraadine täpsus. Sisendiks antakse kontrollerile kaasa asukoht, kuhu antenn ennast pöörama peab. Tagasiside saamiseks kasutatakse mootori küljes olevat sensorit. Selleks, et leida proportsionaalse, integraalse ja diferentsiaalse väärtused, koostati Python-i programmis väikese simulatsiooni antenni liikumise kohta. Esialgselt ei arvestanud võimalikku viga, mis võib keskkonnast tekkida. Järgnevad katsed kirjeldavad elevatsiooni informatsiooni abil mudeli sisendite valikut. Osa sisend andmeid on skaleeritud, selleks väljundite andmed paistaksid graafikul paremini välja.



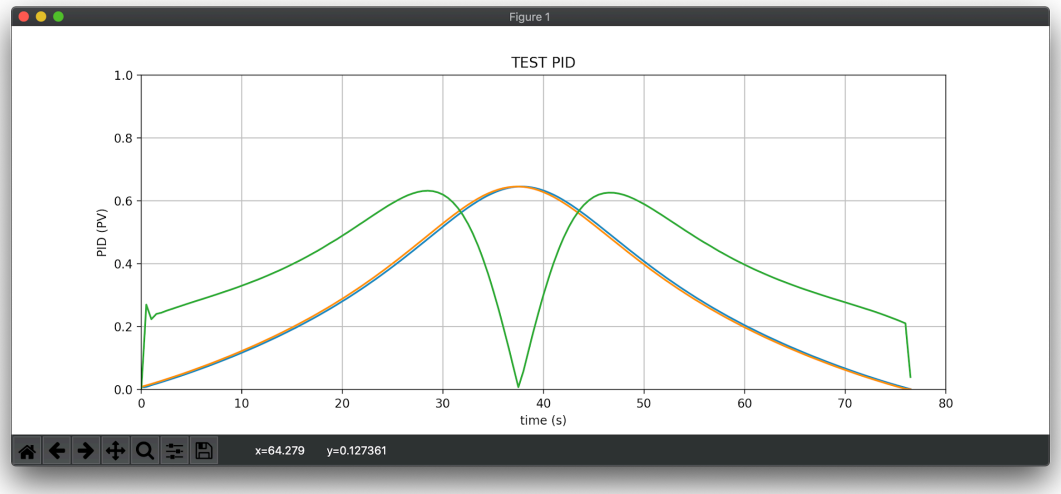
Joonis 6. P = 1, I = 1 ja D = 0

Ülemisel pildil on näha, et kontroller on liiga agressiivne vea parandusel . Selleks, et kontrolleri agressiivsust maandada ja kõrvaldada hilistumisel tekkinud viga tuli korrigeerida P ja I väärtust.



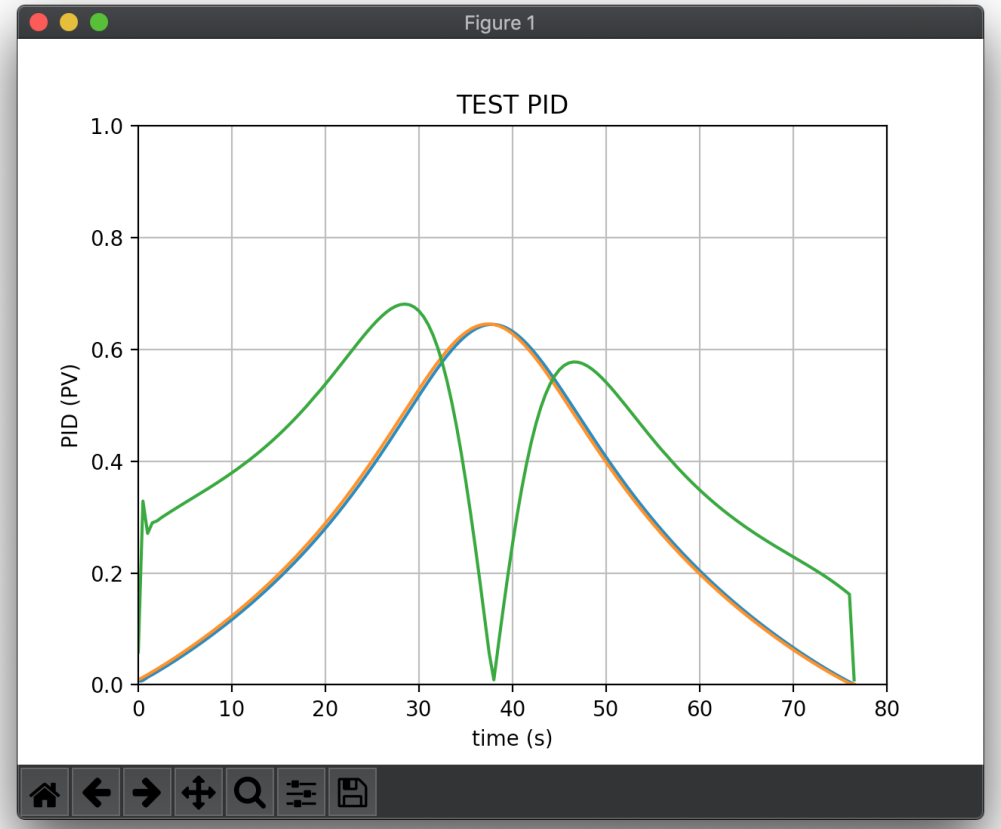
Joonis 7. P = 0.2, I = 0 ja D = 0

Siinkohal on ilusti näha, et kontroller suudab jälgida paremine satelliidi trajektoori kui varasemal katsel. Hilistumine on normi piires. Järgnevalt, lisasin graafiku peale ka väljund sageduse, selleks, et testida sageduse valemi õigsust (roheline joon).



Joonis 8. P = 0.2, I = 0, D = 0 ja koos sagedusega

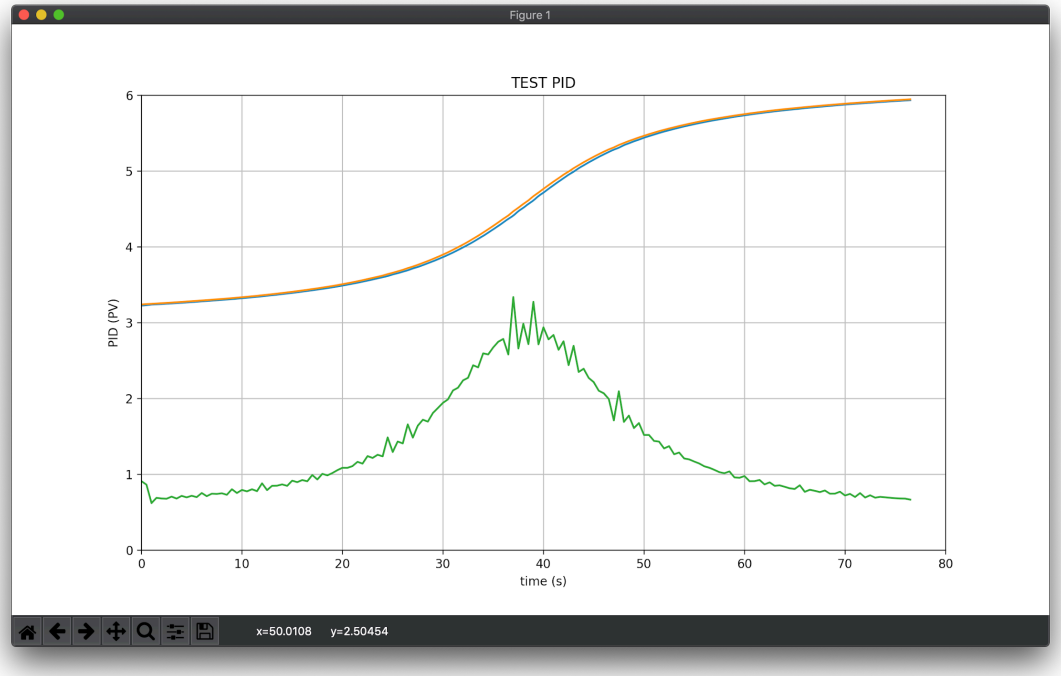
Siinkohal on ilusti näha, et väljund sagedus hakkab tõusma, kui satelliit on tulnud horisondi tagant välja. Seejärel toimub oluline langus, kuna satelliit on jõudnud haripunkti. Kõrgemale ta enam horisondist ei tõuse. Kõige lõpuks peab sagedus langema samamoodi nagu ta esimeses pooles tõusis. Antud kaks katset on tehtud keskkonnas, kus puuduvad igasugused keskkonnas tingitud mõjutus tegurid. Selleks, et testida antenni keeramisel tekkivat viga tuli koodi lisada selleks vastav funktsionaalsus, mis lahutab või liidab iga iteratsiooni käigus tagasisidest saadud arvule juurde vea. Valisin ala keeravuse juhtumi, mis tähendab, et pööramise käigus jääb antenn alati soovitust positsioonist lisaks 0.001 kraadi maha. Tulemus on järgnev:



Joonis 9. P = 0.2, I = 0, D = 0, koos sageduse ja hilistumisega

Antud katsete põhjal võib öelda, et lõpp mudel suudab ka vigade olemasolul ennast piisavalt parandada. Sellele annab kinnituse ülemine pilt, kus on näha, et satelliidi tõusmise faasis peab kontroller antenni mootorile rakendama rohkem jõudu kui langedes. Kõik tagasiside väärtused jäävad pluss miinus 0,5 kraadi sisse, mis tähendab omakorda, et eelnevalt valitud PID väärtused on sobilikud. Kui arvestades natukene tulevikku, siis reaalselt antenni peal võib tekkida olukord, kus neid sisendeid tuleb korrigeerida jällegi tingituna keskkonnast tulevatest teguritest.

Kuna projektis kasutatakse lisaks elevatsiooni mootorile ka rotatsiooni mootorit, siis tingituna sellest, tulid eelnevalt nimetatud katsed uuesti läbi teha uute andmetega. Lõputöösse on lisatud ainult viimase katse pilt:



Joonis 10. P = 0.2, I = 0, D = 0 rotatsioon koos sageduse ja hilistumisega

Sarnaselt elevatsiooni puhul, kasutati rotatsiooni puhul samu PID-i väärtusi. Esialgselt võib tunduda sageduse arvutusel võib midagi valesti olla, kuid sageduse hüppamine esineb jälgimisjaamalt tulevate andmete pärast. Reaalselt, ei tohiks see probleeme tekitada, sest kui vaadata suurima hüppekoha minimaalset ja maksimaalset väärtus, siis vahe mõlema punkti vahe on ainult 0.5Hz.

5. Tehniline analüüs

Siin peatükis räägitakse lähemalt lõputöö käigus tehtud tehnilisest analüüsist, täpsemalt, mis komponente tuli valida selleks, et oleks võimalik implementeerida juhtimissüsteem. Siia hulka kuulub koodrite protokolli ja erinevate teekide valik.

5.1 Kooderi protokoll

Koodrite protokolli valikuks oli ette antud kaks varianti: SSI ja IO-Link. Valikus lähtuti põhiliselt dokumentatsiooni kvaliteedist, seadistamise lihtsusest ja projekti nõuetest.

5.1.1 SSI

SSI ehk sünkroonne sensori liides [18] on jadaliides, mida kasutatakse üle kogu maailma tehaste automatiseerimiseks. See arendati aasta 1908 Max Stegmann GmbH ettevõtte poolt, selleks, et kätte saada pöörkooderite informatsiooni.

Protokolli uurimist alustati dokumentatsiooni lugemisest. Üldiselt oli ilusti ära kirjeldatud mismoodi protokoll töötab, kuidas on sõnumi struktuur ja mismoodi on seda võimalik kasutada. Kuid kohe alguses tekkisid sellega teatud mure kohad. Näiteks, enamus ühenduse näited oli kirjeldatud, kuidas oleks võimalik pöörkooderit ühendada Raspberry Pi või Arduino plaadiga ja selle kaudu vajalik informatsioon välja lugeda. Selline variant ei ole kõige parem, kuna kõik see nõuab liigset lisatööd ja piisavalt palju vahekihte. Näiteks tuleks ühenda omavahel ära kooder ja plaat ning pärast seda kirjutada lisaks programm, mille kaudu saaksid välised rakendused vajaminevaid andmeid ära kasutada.

Siinjuures üritati lisaks leida rohkem universaalsemaid lahendusi. Näiteks kasutatakse sagedusmuunduriga suhtlemisel Modbus-i protokolli. Antud teadmiste põhjal tuli leida seade, mis toetab nii Modbus-i ja ka SSI-i protokolli. Seade ise oleks tõlgendanud SSI protokolli ning salvestanud registritesse sensori informatsiooni. Seejärel, oleks Modbus-i abil võimalik see sama informatsioon kätte saada arvutis. Pika otsingu käigus leiti

vastavad seadmed kuid ka siin kohal tekkis probleeme dokumentatsiooni kvaliteedi osas.

5.1.2 IO-Link

Võrreldes IO-Link-i [19] SSI-iga on tegemist palju uuema lahendusega. Seda kasutatakse täpselt samamoodi tehaste automatiseerimisel kuid lisaks koodrite ja sensorite andmete lugemisele, suudab ka IO-Link-i protokoll edastada käske täituritele ehk aktuaatoritele. See omakorda annab väga suure eelise SSI ees.

Dokumentatsiooni koha peal on IO-Link kirjeldatud ära hästi. Lisaks sellele on ka seadmete ülesseadmine lihtne. Põhimõtteliselt läheb vaja kahte seadet. IO-Link master ja IO-Link andurit. Need ühendatakse omavahel spetsiaalse kaabliga. Seejärel tuleb ühendada, kas master seade võrku või otse arvuti külge, eesmärgiga sensoreid seadistada või nendelt andmeid lugeda. Olenevalt masteri seadmest, saab ühe seadme külge lisada mitu erinevat sensorit, mis osutub kasulikuks ka projekti juures. Lisaks sellele toetab masteri seade ka Modbus TCP protokolli. Uurides lähedalt erinevaid master seadmeid, oli dokumentatsioonist ilusti välja toodud koht, millistesse mälu pesadesse kogu vajalik informatsioon kirjutatakse.

5.1.3 Kokkuvõte

Uuringu käigus valiti välja IO-Link-i peamiseks koodrite protokolliks. Põhiliseks otsustus punktis sai seadistamise lihtsus. Kuna antennil kasutatakse kahte mootorit, mis tähendab omakorda kahte kooderit, siis IO-Link-i puhul on võimalik ühendada need ühe masteri seadme külge lihtsalt, kasutades selleks spetsiaalset kaablit. Selleks, et arvutis sensori informatsiooni kätte saada tuleb esiteks ühendada arvuti ja master seade omavahel ning seejärel kasutada suhtluseks Modbus-i protokolli.

5.2 Flask ja React

Antud tehnoloogiad tuli päeva korrale olukorras, kus oli vaja tegeleda sagedus muunduri ja arvuti vahelise suhtlusega. Probleemi põhjustas muunduri keeruline olekute lugemine. Näiteks kiirus, pöörde suund ja sisse lülitus. Selleks, et probleemi parandada, tehti lõputöö käigus lihtne veebirakendus, mille abil oleks võimalik visuaalselt näha kõiki väärtusi, mida on võimalik muundurist küsida ja kirjutada.

Selleks kasutati veebirakenduse loomisel Javascript-i raamistikku React[2]. Samuti tähendas see omakorda olukorda, et lisaks kontrollerite implementeerimisele tuli külge luua API kiht, mille kaudu oleks vajaminevat informatsiooni kätte saada. Siinkohal võeti kasutusele Python-i raamistikku Flask[3]. Mõlemad tehnoloogiad valiti nende ülesseadmise lihtsuse ning autori kogemuste põhjal.

5.3 ivPID

"ivPid"[4] on Python-is kirjutatud lihtne PID-i kontrolleri implementatsioon, mida kasutatakse vea paranduse arvutamisel antenni pööramise käigus. Töö lihtsustamise mõttes ei olnud mõtet keskenduda PID implementeerimisele. Pigem oli tähtsal kohal leida mõistlik lahenduse pööramisel tekkinud vigade korrigeerimisele. Siinkohal oli valikus mitmeid samalaadseid mooduleid, kuid otsustati kasutada "ivPID"-i, kuna see tundus proovimise käigus kõige paremini integreeritavam olemas olevasse rakendusse.

2 https://reactjs.org
3 http://flask.pocoo.org
4 https://github.com/ivmech/ivPID

6. Lahenduskäik

Käesolevas peatükis räägitakse lähemalt lõputöö käigus tehtud praktilisest tööst. Tuuakse välja töö käigus tekkinud erinevad etapid ning nendega seonduvad probleemid ja raskused. Järgnevad etapid on laiendatud esimeses peatükis nimetatutega.

6.1 Side sagedusmuunduriga

Praktiline osa algas sagedusmuunduriga suhtluse pidamisest. Selleks tuli seada eesmärgiks kirjutada muunduri registrisse maksimum sagedus ning seejärel sama väärtus seadme mälust küsida. Kuna sellel alal puudus autoril varasem kogemus, tuli algust teha teooria õppimisega. Täpsemalt keskenduti siinkohal Modbus- i protokolli ja muunduri dokumentatsiooni lugemisega.

Lõputöö alguses sai mainitud, et Modbus on jadaside protokoll, mida kasutatakse erinevates tööstusseadmetes. Protokolliga on ära määratud, missuguseid funktsioone on võimalik kasutada ning mis mälu vahemikes väljundid, sisendid ja registrid asuvad. Mälu pesa tegevus ja eesmärk pannakse paika seadme tootja poolt.

Probleemseks kohaks sai kohe alguses ROS-i käivitamine, sest antud tehnoloogiat toetatakse põhiliselt ainult Debian-i masinates ning autoril endal ei olnud sellise võimekusega seadet kohe võtta. Virtuaalmasinas ROS-i tööle panek oli liiga aegavõttev ja OS X Mojave teegid vananenud. Selle põhjal langetati otsus töö seisaku vältimiseks panna rakenduse arhitektuur paika hilisemates etappides. Järgnevalt tuli tegeleda Modbus-i protokolli realiseerimisega. Töö mahu vähendamiseks, võeti kasutusele moodul nimega “pymodbus”[5]. Kuigi sarnaseid mooduleid leiti teisigi, siis põhiliseks otsustus punktiks jäi lihtne kasutus viis ja hästi kirjeldatud dokumentatsioon.

Pärast teegi välja valimist sai käsile võetud muunduri dokumentatsioon. Siinkohal avastati suurel hulgal tabeleid, mis jagunesid tulenevalt Modbus-i protokollist

[5] https://pymodbus.readthedocs.io/en/latest/

väljunditeks ja hoidvateks registrikeks. Võttes kokku kõik tabelid oli kõigi nende peale umbes 500 rida informatsiooni, mida muunduriga võimalik teha on. Järgnevad tabelid on üks osa väljunditest ja registritest.

Tabel 3. Omron-i muunduri väljundid

Coil No.	Item	R/W	Setting
0000h	unused	–	(Inaccessible)
0001h	Operation command	R/W	1: Run, 0: Stop (valid when A002 = 03)
0002h	Rotation direction command	R/W	1: Reverse rotation, 0: Forward rotation (valid when A002 = 03)
0003h	External trip (EXT)	R/W	1: Trip
0004h	Trip reset (RS)	R/W	1: Reset

Tabel 4. Omron-i muunduri registrid

Register No.	Function name	Function code	R/W	Monitoring and setting items	Data resolution
0000h	unused	–	–	Inaccessible	
0001h	Frequency source	F001 (high)	R/W	0 to 40000 (valid when A001 = 03)	0.01 [Hz]
0002h		F001 (low)	R/W		
0003h	Inverter status A	–	R	0: Initial status 2: Stopping 3: Running 4: Free-run stop 5: Jogging 6: DC braking 7: Retrying 8: Tripping 9: Undervoltage (UV),	–
0004h	Inverter status B	–	R	0: Stopping, 1: Running, 2: Tripping	–
0005h	Inverter status C	–	R	0: – 1: Stopping 2: Decelerating 3: Constant-speed operation 4: Accelerating 5: Forward rotation 6: Reverse rotation 7: Switching from fwd. to rev. rotation, 8: Switching from rev. to fwd. rotation, 9: Starting fwd. 10: Starting rev.	–
0006h	PID feedback	–	R/W	0 to 10000	0.01 [%]

Mõlema tabeli pealt on võimalik näha ühisosana väljundi aadressi, milleks seda aadressi kasutatakse, kas aadressilt on võimalik lugeda ja kirjutada väärtusi ning kõige lõpuks, milliseid väärtusi võib mälupesale saata. Registrite tabelis lisandub veerg nimega funktsiooni kood. Näiteks, kui ma soovin sagedusmuundurit tööle panna, tuleb kirjutada väljundi mälus aadressile 0001h väärtus 1.

Kõige keerulisem osa siin etapis oli tegelikult õigete mälupesade kätte saamine. Seda illustreerib kõige paremini väljundite tabel. Sealt on võimalik välja lugeda, et kuueteistkümnendik süsteemis kirjeldatud 0000h aadress on ligipääsmatu ja kõik käsud hakkavad pihta alates 0001h. Kuna “pymodbus” ise nõuab aadressi sisendiks *integer* tüüpi väärtusi tuli kuueteistkümnendik süsteemi väärtus muuta kümnendsüsteemiks. Ehk 0001h võrdub 1-ga. Proovides kirjutada aadressile 1 mingisugust väärtust, ei reageerinud muundur sellele. Selleks, et teha kindlaks, et probleem pole ühenduses,

võeti kasutusele rakendus nimega Modbus Probe, mille abil sai küll väärtusi kirjutada ja lugeda, aga teatud aja pärast jooksis rakendus lihtsalt kokku. Probleemi tuum seisnes selles, et kõik seadmete registrid olid nihkes seoses “pymodbus”-i teegi tõttu, mis tähendas, et aadressile kümmnendsüsteemis 1 vastas tegelikult tabelis 0002h. Selle probleemi kõrvaldamiseks tuli lihtsalt konverteerimise käigus lahutada maha saadud tulemusest -1.

6.2 Mootori pööramine

Järgnevas etapis oli olulisel kohal mootori pööramine läbi arvuti. Kuid kõigepealt oli vaja panna tööle mootor manuaalses režiimis, mis tähendas, et tuli kasutada muunduri enda sõrmistikku. Pildil 10 on näha, et sõrmistikul on olemas 6 nuppu. RUN ja STOP/RESET nupp käivitab ja peatab muunduri manuaalses režiimis, nende vahele jäävad ülesse ja alla nupud, mida kasutatakse funktsioonide vahel liikumiseks ja nende väärtuste muutmiseks. Kollast nuppu kasutatakse funktsiooni väärtuse ja koodi vahel liikumiseks. Sinist nuppu registrite väärtuste salvestamiseks ja funktsiooni gruppide vahel liikumiseks.

Funktsioonid on loogiliselt jaotatud gruppidesse ning on tähistatud tähtedega d, F, A, b, C, H, P, ja U. Igale funktsioonile käib ette number nagu näiteks 001 või 002. Kombineerides numbreid ja tähti, saame lõpliku funktsiooni nimetuse (näiteks 002U).

Joonis 11. Omron-i sagedusmuunduri sõrmistik

Manuaalse režiimi seadistamisel tuli lähtuda mootori andmetest. Sõrmistikul tuli seadistada järgmised read:

1. A001 = 02 - Seadistada mootori kiiruse sisendiks muunduri mälu
2. A002 = 02 - Seadistada mootori käivitamise sisendiks muunduri mälu
3. A003 = 50 - Seadistada mootori põhi sagedus 50Hz peale
4. A082 = 230 - Seadistada mootori pinge
5. b012 = 70 - Seadistada mootori maksimum temperatuur
6. H004 = 6 - Seadistada mootori poolide arv

Käivitamiseks tuli järgida järgmisi samme:

1. Teha kindlaks, et PWR indikaator töötab
2. Teha kindlaks, et RUN nupu kohal indikaator töötab
3. Teha kindlaks, et PRG ja ALM indikaator ei töötaks

4. Vajutada RUN nuppu
5. Hoida peal üles nuppu, mis hakkab muutma väljund sagedust

Pärast mitmeid proovimisi ja taas käivitusi, hakkas mootor juhendis olnud käskude peale tööle. Esialgne probleemi põhjus ei ole täpselt teada, kuid võib arvata, et põhjuseks oli muunduri ja mootori vahel olevad ühendused.

Selleks, et oleks võimalik käivitada mootor läbi arvuti, tuli muunduri seadetes teha kaks olulist muudatust.

1. A001 = 03 - Seadistada mootori kiiruse sisendiks Modbus
2. A002 = 03 - Seadistada mootori käivitamise sisendiks Modbus

Seejärel, kui kirjutada muunduri väljund mälu aadressile 0001h väärtus 1 ja hoidva registri aadressile 0002h väärtus 1000, peaks tulemus olema manuaalrežiimiga sama.

Joonis 12. Kasutatav mootor

6.3 Veebirakendus

Veebirakenduse eesmärk oli võimaldada luua lihtsat ülevaadet muunduriga seotud seadetest ning testida manuaalselt mootori käivitamist missiooni korras. Töö etapp algas väljundite ja registrite tabelite konverteerimisega CSV faili formaati. Selleks, tuli luua Excelis kaks uut tabelit mõlema tüübi jaoks eraldi ning käsitsi kopeerida dokumentatsioonis olevad funktsiooni read uutesse tabelitesse. Peale seda tuli käia kõik uutes tabelites olevad read üle ning vajadusel neid korrigeerida või kustutada.

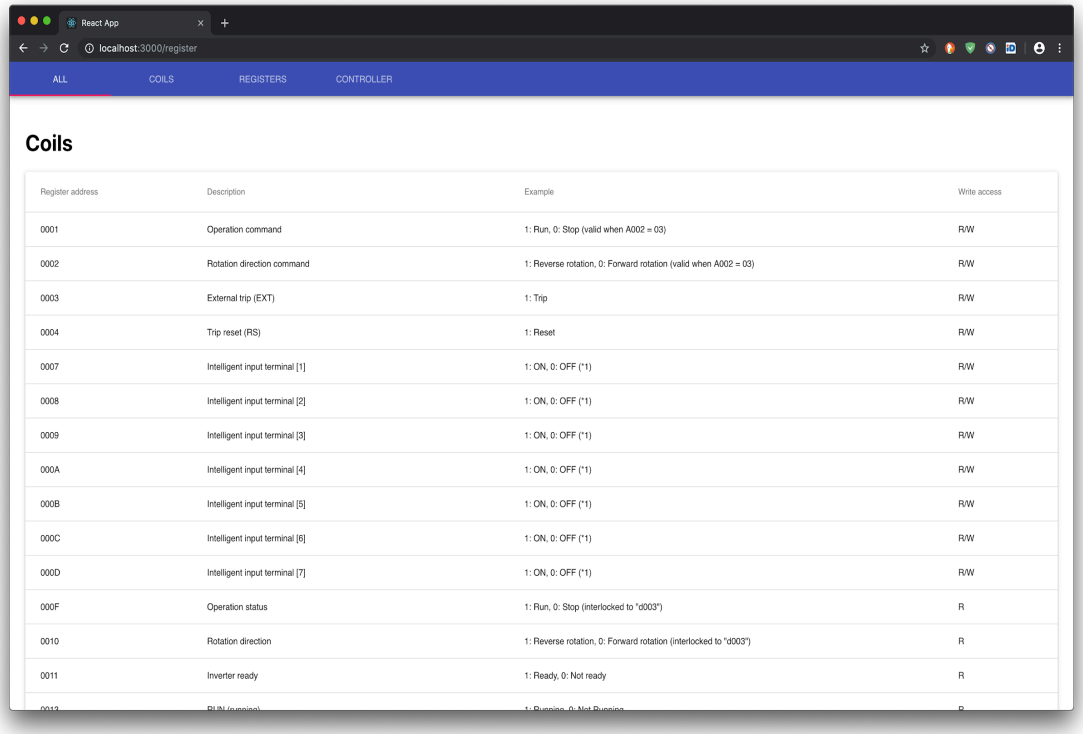
Koodi tasemel tuli luua algselt objekt nimega “RegisterElement”, mis hoiab endas ühe CSV rea andmeid. Selleks, et saada kõiki või ühte kindlat elementi tuleb pöördud “RegisterMapping”-u klassi poole. Klassi initsialiseerimise loetakse CSV failidest kõik read ning vastavalt sellele konverteeritakse need õigele kujule. Sisemiselt on funktsiooni read jaotatud kahte erinevasse sõnastikku. Vastavalt väljundid ja registrid. Sõnastiku objekti võtmeks pannaks muunduri mälu aadress kuuekümnendsüsteemis ning väärtuseks “RegisterElement”.

Selleks, et järgida MVC mustrit rohkem, loodi nende kahe klassi jaoks eraldi teenuse kihi nimetusega “RegisterService”. Läbi selle klassi on võimalik kontrolleril saada kätte registrite informatsiooni, nende hetke väärtusi või neid muuta. “RegisterService” initsialiseerimisel tuleb anda kaasa USB pordi nimetus ja seadme number võrgus. Näiteks on nendeks mõlemaks vastavalt “/dev/tty.usbserial-A906O081” ja 1.

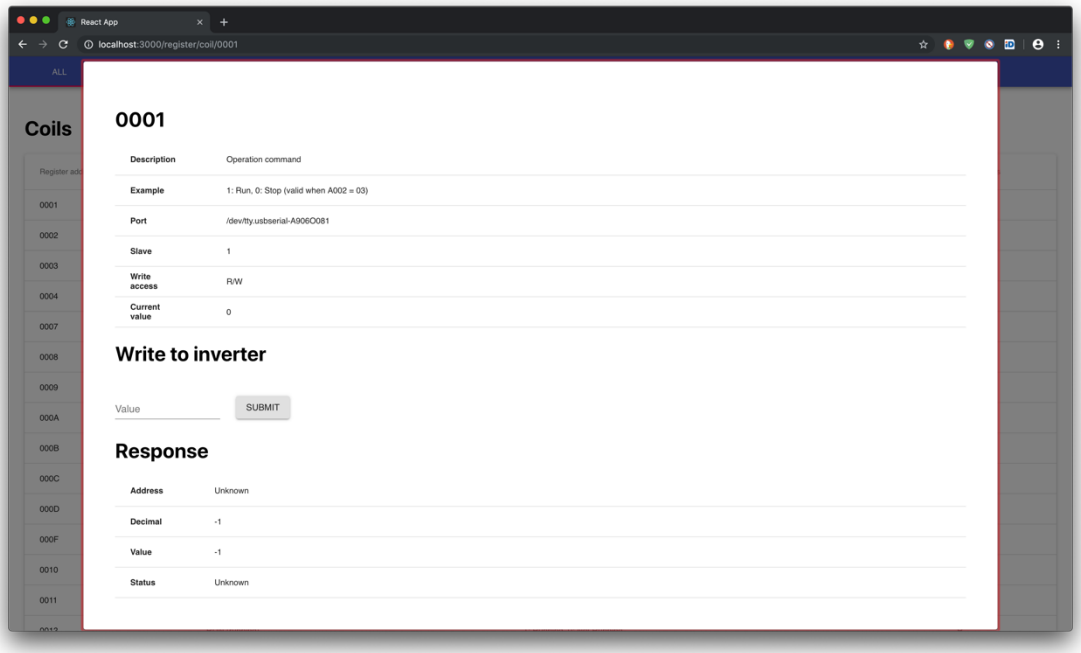
Kontrolleri kihi loomiseks kasutati Python-i teeki Flask-i, sest autoril on olemas sellega varasem kogemus ning tegemist on lihtsasti kasutatava tehnoloogiaga. Sõnumite vahetuseks kasutatakse JSON-i formaati. Samuti suhtleb see sama kiht “RegisterService” teenuse kihiga. Otse pöördumist registritesse siinkohal ei lubata.

Pärast seda, kui backend-i poolel oli loodud vajalikud funktsionaalsused, tuli rohkem keskendutud veebirakenduse loomisele. Selleks võeti kasutusele JavaScript teek nimega React, mis lihtsustas oluliselt rakenduse püsti paneku ja arendamise aega. Kuid kõige pealt enne arendamist, tuli seada paika mõned nõuded, mida rakendus peab toetama.

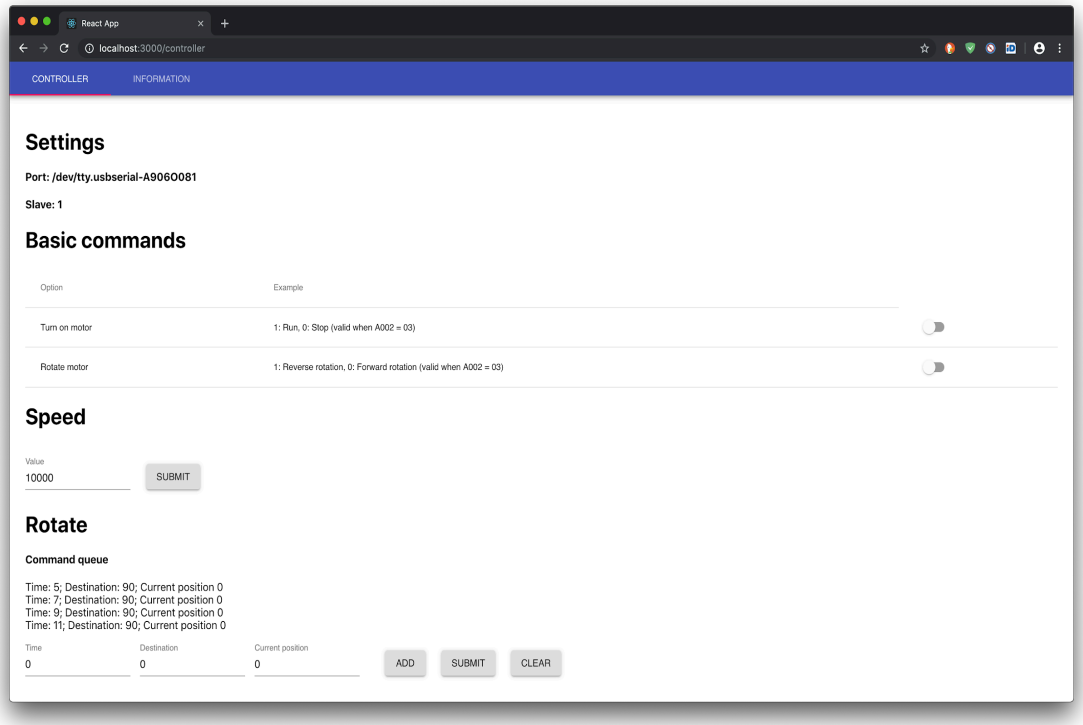
1. Kuvama kõigi väljundite seonduvat informatsiooni
2. Kuvama kõigi registritega seonduvat informatsiooni
3. Võimaldama lugeda väärtusi ette antud mälu aadressil
4. Võimaldama kirjutada väärtusi ette antud mälu aadressile
5. Anda kiiret ülevaadet mootori käimise koha pealt
6. Sisestada manuaalselt missioon testimise eesmärgil



Joonis 13. Väljundite ja registrite informatsiooni vaade



Joonis 14. Väljundi väärtuse muutmise vaade



Joonis 15. Muunduri testimise vaade

6.4 Põhifunktsionaalsus

Olles valmistanud vajamineva põhja sai edasi liikuda põhifunktsionaalsuse arendamise. Nendeks on varasemalt mainitud peamine ja mootori kontrolleri funktsionaalsus. Peamise kontrolleri põhiliseks eesmärgiks on vastu võtta missioni informatsiooni ning töödelda ning õigeaegselt edastada need mootori kontrollerile. Enne seda, kui kõik käsud edastatakse, saadetakse eelkäsk, et antenn keeraks ennast õigesse suunda.

Mootori kontroller seevastu võtab käsud vastu ning hakkab neid ükshaaval läbi käima. Ühes käsus on ära määratud pööramise aeg ning pööramise asukoht. Selle põhjal arvutatakse välja muundurile edastatav sagedus. Lisaks tuleb kontrollida ka pööramise suunda. Selleks, võetakse kaks järjestikku olevat käsku ja lahutatakse viimase käsu asukohast esimese käsu asukoht. Kui tulemus on negatiivne peab mootor töötama vastu päeva, teisel juhul päripäeva.

Selleks, et antenn ennast üle ei keeraks või maha ei jääks, kasutatakse vea parandamiseks PID kontrollerit, kus soovitud väärtuseks on pööramise asukoht ja tagasiside tuleneb mootori külge paigaldatud enkoodritest.

Antud etapi käigus sai rakendusse loodud kaks teenuse kihti “RotationService” ja “ControllerService”. Kontrolleri teenuse puhul sai funktsionaalsusest realiseeritud satelliidi positsiooni lugemine CSV-failist, nende ridade transformeerimine, antenni suuna initsialiseerimine ja missioni käskude õigeaegne edastamine. Õigeaegne siin kontekstis tähendab, et mootorid pannakse käima alles siis kui satelliit on horisondi tagant nähtavale tulnud.

Pööramise teenuse funktsionaalsuseks oli sageduse arvutamine, pöörde suuna kindlaks tegemine ja veaparandus. Siinkohal oli suureks abiks analüüsi käigus tehtud töö. Kui kaks eelnevalt nimetatud eesmärkidest sai ilusti täidetud, siis viimase implementeerimisel tekkis probleem. Kuigi PID-i algoritm oli koodi sisse migreeritud, puudus võimalus seda testida, põhjusel, et vajalikud komponendid ei jõudnud

õigeaegselt kohale. Nimelt tuli lisaks tellida juurde mõned IO-Link-i kaablid USA-st, mille tarne aeg ületas lõputöö esitamise tähtaega.

Töö viimases etapis tuli lõpuks kogu loodud backend-i kood viia ROS-i sõlmede peale. MVC arhitektuuri muster asendus ROS-i mustriga. Selleks, tuli lahutada oma vahel teenuse kihid ja luua arhitektuuri joonise põhjal eraldi sõlmed. Tekkisid järgnevad sõlmed: peamine kontroller, horisontaalse mootori kontroller, horisontaalse mootori sensor, vertikaalse mootori kontroller ja vertikaalse mootori sensor. Igale mootori kontrollerile tuli eraldi külge panna „RegisterService“, sest tulevikus ei pruugi kasutatavad seadmed sarnased olla. “ControllerService” teenuse funktsionaalsus kandus peamise kontrolleri sõlme ja „RotationService“ mootorite kontrolleritesse.

7. Edasine töö

Lõputöö käigus ei suudetud täita ühte põhilist eesmärk, milleks oli, süsteemi valmidus suuta lugeda mootori sensoritel hetke pöörde asukohta, mille põhjustas komponentide pikk tarneaeg. Kuid, kui vaadata kogu projekti laiemalt, siis see ei ole ainukene ülesanne, mis vajas täitmist.

Selleks, et alustada mootori täieliku testimisega, koos vea parandusega, tuleb mootori külge monteerida enkoodrid. Seejärel luua programmi uus teenus, mis tegeleb kooderitelt informatsiooni lugemisega ning kõige lõpuks migreerida see olemas olevatesse mootorite kontrolleritesse. Testimiseks saab kasutada sama laadset meetodit, mida kasutati käesoleva lõputöö valideerimiseks. Esiteks tuli mootori peale ära märgistada 90, 180, 270 ja 360 kraadi. Seejärel anda ette missioni käsud, näiteks pööra antenni 90 kraadi positsioonilt 0 kraadi 10 sekundi jooksul. Testimine osutub õnnestuks alles siis, kui mootor on suutnud ennast pöörata täpselt 90 kraadi.

Juhul, kui testimine õnnestus ja rakenduse taristu on pandud paika, saab edasi liikuda reaalsete antenni katsetega eeldusel, et kogu antenn on kokku monteeritud. Siinkohal tuleb meeles pidada, et eelnevad testimised ei pruugi olla piisavad, põhjusel, et ei ole arvestatud antenni enda omaduste ja keskkonna tingimustega. Kuna juhitav objekt on piisavalt suur ja raske, on suur tõenäosus, et pööramise alguses või üleminekult ühelt sageduselt teisele, võib tekkida libistamine, mis tähendab, et mootori võlli pööramisel kaob osa jõudu ära, mida reaalselt tuleks rakendada antenni pööramisele. Täpselt samasugune nähtus võib tekkida ka tuule korral, mis võib põhjustada antenni võnkumist. Selle probleemi peaks ideaalis ära kõrvaldama PID-i algoritm. Testimist siinkohal saab loetud õnnestunuks, kui juhtimissüsteem suudab reaalselt jälgida üle lendavat satelliiti saades sisendiks tugeva signaali.

Olukorras, kus testimine ei pruugi nii edukalt minna, tuleb koostada põhjalik analüüs. Üheks võimalikest nõrkustest võib olla PID-i kontrolleri etteantud sisendid. Võib osutuda, et reaalsele olukorrale reageerib algoritm liiga aeglaselt või liiga kiiresti.

Selliseid asju ilma antenni olemasoluta on väga keeruline ette ennustada, kuna mõjutus tegureid on piisavalt palju.

Teiseks nõrkuseks võib osutuda tagasisideks kasutatava mootori kooderite informatsioon võib olla ebapiisav, kuna need ei suuda ära kirjeldada piisavalt hästi antenni hetke positsiooni. Selle lahenduseks on pakutud kiirendus andurite lisamist antenni külge. See omakorda tähendab, et antud koodibaasile tuleb lisada juurde lisa funktsionaalsus, mille abil veaparandus kvaliteeti parandatakse.

8. Valideerimine

Välja arvatud juhtumile, mille käigus ei õnnestunud testida veaparandust, sai siiski valideerida kõige tähtsamat funktsionaalsust - pööramist. Selleks tuli mootori peal ära märkida null punkt, kus pööramist alustatakse ja vähendada oluliselt kiirenduse aega. Kiirenduse aeg seadistati 1 sekundi peale, mis tähendab, et mootor peab saavutama maksimum sageduse etteantud aja jooksul. Vähendamise eesmärgiks oli suurendada pööramise täpsust ja kiirust.

Katse näeb välja järgnev. Andes mootorile ette mitu erinevat käsku, sai vaadata, kas võll jõudis missiooni lõppedes kindlasse punkti. Etteantud käsud on järgnevad:

1. Pööra võlli 90 kraadi 5 sekundi jooksul
2. Pööra võlli 90 kraadi 7 sekundi jooksul
3. Pööra võlli 90 kraadi 9 sekundi jooksul
4. Pööra võlli 90 kraadi 11 sekundi jooksul

Nagu käskudest näha peab mootor töötama kokku 23 sekundit ja pöörama selle aja jooksul 360 kraadi. Ainukene väärtus, mis käskude vahetusel pidi muutuma oli väljund sagedus, mis omakorda mõjutas mootori pöörlemis kiirust. Antud katse osutus väga kasulikuks ning tõi välja mitu sageduse arvutusel tekkinud viga. Lõpp kokku võttes katse õnnestus.

Selleks, et testida suuna muutust edastasin pööramise teenusele järgnevad käsud:

1. Pööra võlli 45 kraadi 5 sekundi jooksul
2. Pööra võlli 90 kraadi 5 sekundi jooksul
3. Pööra võlli 45 kraadi 5 sekundi jooksul
4. Pööra võlli 90 kraadi 5 sekundi jooksul

Siinkohal on käskudest näha, et mootori kontroller peab suunda muutma kahel korral. Esimesel korral käsu 2 ja 3 juures ning pärast seda käsu 3 ja 4 juures. Sarnaselt esimese katse tulemusele, õnnestus ka järgnev katse.

Joonis 16. Näide testkeskkonnast

Joonis 17. Näide testimisest

9. Kokkuvõte

Käesoleva magistritöö eesmärk on luua Tallinna Tehnikaülikooli poolt loodavale parabool antennile osaline juhtimissüsteem, mille abil oleks võimalik tulevikus jälgida orbiidil lendavaid satelliite. Osaline juhtimissüsteem all mõeldakse, et ajaliste piirangute ja projekti mahukuse tõttu ei olnud võimalik kõike valmis saada.

Lõputöö vältel tegeleti järgnevate valdkondadega: pandi alus juhtimissüsteemis osalevate komponentide arhitektuurile, analüüsiti, mis moodi oleks võimalik liigutada mootoreid ja vastavalt sellele teostada ka vea arvutust. Lõpp faasis lisandus juurde rakenduse implementeerimisega eelnevalt kogutud teadmiste põhjal. Rakenduse funktsionaalsus on realiseeritud Python-is ning on jagatud ära erinevateks ROS-i sõlmedeks.

Töö tulemusena valmis piisava funktsionaalsusega juhtimissüsteem, mis võimaldas liigutada antenni mootoreid vastavalt vajadusele. Ainuke puudus süsteemis on reaalse tagasiside saamine kooderitelt, kuna teatud IO-Link-i komponendi ei saabunud õigeks ajaks. Seoses sellega jäi ka üks töö alguses püstitatud eesmärk täitmata.

Vaatamata sellele, et kogu projekt valmis ei saanud, tuleb sellega edasi minna. Esiteks, lõpetada ära kooderite andmete integreerimine PID-i kontrollerisse ning tegeleda selle testimisega. Teiseks, tuleb kokku monteerida kogu antenn ning paigaldada see Mektory maja katusele. Alles siis saab hakata tegelema lõpliku testimisega. Võib osutuda, et olemas olevat juhtimissüsteemi mudelit tuleb edasi arendada tingituna keskkonnast olevatest teguritest.

Kasutatud kirjandus

[1] R. Gordon, „Satelliidiprogrammi ajajoon,“ [Võrgumaterjal]. Available: https://ttu.ee/projektid/mektory-est/satelliidiprogramm-4/satelliidiprogramm/satelliidiprogrammi-ajajoon/.

[2] „Near-infrared spectroscopy,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Near-infrared_spectroscopy.

[3] „RGB color model,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/RGB_color_model.

[4] R. Gordon, „TTÜ100 Satelliidi tutvustus,“ [Võrgumaterjal]. Available: https://ttu.ee/projektid/mektory-est/satelliidiprogramm-4/satelliidiprogramm/.

[5] R. Gordon, „Pardakaamera,“ [Võrgumaterjal]. Available: https://ttu.ee/projektid/mektory-est/satelliidiprogramm-4/satelliidiprogramm/wp1-b-pardakaamera/.

[6] I. Ali, N. Al-Dhahir ja J. E. Hershey, „Predicting the Visibility of LEO Satellites,“ [Võrgumaterjal]. Available: http://www.utdallas.edu/~aldhahir/visibility.ps.

[7] „Yagi antenn,“ [Võrgumaterjal]. Available: https://et.wikipedia.org/wiki/Yagi_antenn.

[8] „Ku band,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Ku_band.

[9] D. Stone, „The Advantages of Higher Bandwidth Frequencies,“ [Võrgumaterjal]. Available: https://www.techwalla.com/articles/the-advantages-of-higher-bandwidth-frequencies.

[10] C. Hudson, „Ka-band or Ku-band – Which is Better for You?,“ [Võrgumaterjal]. Available: https://www.intelsatgeneral.com/blog/ka-band-or-ku-band-which-is-better-for-you/.

[11] R. Tomsen, „Juhtimissüsteemi loomine TTÜ satelliidi maajaama paraboolantennile,“ [Võrgumaterjal]. Available: https://digi.lib.ttu.ee/i/?11182.

[12] „Asimuut,“ [Võrgumaterjal]. Available: https://et.wikipedia.org/wiki/Asimuut.

[13] „MX2 Users Manual,“ [Võrgumaterjal]. Available: https://www.miel.si/wp-content/VsebinaPDF/I570-E2-01-X+MX2+UsersManual.pdf.

[14] „ROS,“ [Võrgumaterjal]. Available: http://www.ros.org.

[15] „Rotary encoder,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Rotary_encoder.

[16] „Modbus,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Modbus.

[17] „PID controller,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/PID_controller.

[18] „SSI - Synchronous Serial Interface,“ [Võrgumaterjal]. Available: https://www.kunbus.com/ssi.html.

[19] „IO-Link System Description,“ [Võrgumaterjal]. Available: https://io-link.com/share/Downloads/At-a-glance/IO-Link_System_Description_eng_2018.pdf.