TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mart Hütt 155133IAPB

TALTECH SATELLIIDI PROGRAMMI PARABOOLANTENNI JAOKS JUHTIMISSÜSTEEM LOOMINE ROS'I BAASIL

Bakalaureusetöö

Juhendaja: Evelin Halling
PhD

Tallinn 2020

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: Mart Hütt

22.06.2020

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on teha valmis TTU100 tudengisatelliidi programmi jaoks paraboolantenni juhtimissüsteem. Juhtimissüsteem peab olema võimeline iseseisvalt koguma infot etteantud satelliitide orbiitide koha, arvutama orbiidi põhjal ülelennu toimumise aja ning suunama paraboolantenni pidevalt satelliidile ülelennu ajal ±0.2° täpsusega. Juhtimissüsteemi poolt jälgitavaid satelliite peab olema võimalik lisa ja eemaldada kasutades missioonijuhtimise tarkvara.

Töö käigus valmis paraboolantenni juhtimissüsteem, mis koosneb 9-st ROS'i sõlmest. Süsteem on võimeline iseseisvalt küsima etteantud satelliitide kataloogi numbrite aluses vastava TLE. TLE põhjas arvutab süsteem satelliitide ülelennu ajad ning arvutatud aegade põhjal teeb otsuse, millist jälgima asuda. Süsteemis poolt jälgitavaid satelliite on võimalik lisada ja eemaldada MCS'ist. Töö juures oli oluline luua juhtimissüsteemi jaoks arhitektuur, mida lihtne tulevikus laienda erinevate funktsionaalsustega ning testida.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 28 leheküljel, 9 peatükki, 13 joonist ja 2 tabelit.

Abstract

ROS based control system for TalTech satellite program’s parabolic antenna

The aim of the thesis is to create a control system for TTU100 student satellite program’s parabolic antenna which is capable of pointing the antenna towards the satellite with an accuracy of ±0.2°. To achieve this the system must collect TLE information for each tracked satellite, calculate satellite flyovers and control the horizontal and vertical motor.

The result of this thesis is a ROS based system composed of 9 different nodes that collects TLE information for each satellite defined in list, which is maintained by the MCS through the system. Based of the gathered TLE the system calculates flyover for each TLE and chooses the next one which to track. On an active flyover system continuously points the parabolic antenna towards the satellite. Special consideration was put on the architecture to achieve easy extendability and testability.

The thesis is in Estonian and contains 28 pages of text, 9 chapters, 13 figures, 2 tables.

Lühendite ja mõistete sõnastik

1U	*One unit,* 10x10x10 cm kuupsatelliitide standardsuurus
MCS	*Mission control system,* missioonijuhtimise tarkvara
NIR	*Near-infrared,* infrapuna spekter
RGB	*Red-Green-Blue*, puna-roheline-sinine spekter
TCP/IP	*Transmission control protocol/Internet Protocol, arvutivõrkudes enim kasutusel olev internetiprotokollistik*
JSON	*JavaScript Object Notation, a*ndmevahetusvorming
TLE	*Two-line element,* andmeformaat orbiidil asuvate objektide kirjeldamiseks
LEO	*Low Earth Orbit*, Maa-lähedane orbiit
API	*Application Programming Interface*, programmiliides mille abil saab liidestada tarkvaraga uusi komponente
GPS	*Global Positioning System*, üleilmne kordinaatide süsteem asukoha määramiseks

Sisukord

1 Sissejuhatus....10
1.1 Probleem....10
1.2 Eesmärgid....12
2 Kasutatud tehnoloogiad....14
2.1 ROS....14
2.2 ActiveMQ....15
2.3 STOMP....15
3 Satelliidi orbiidi jälgimine ja ülelennud arvutamine....16
3.1 Satelliidi orbiit....16
3.2 Ülelennu leidmine....18
4 Paraboolantenni suunamine....20
5 Liidestus MCS'iga....23
5.1 REST teenus põhine ühendus....24
5.2 Socket'i põhine ühendus....25
5.3 ActiveMQ põhine sõnumivahetus....25
5.4 Sõnumivahetuse teostus....25
6 Paraboolantenni juhtimissüsteemi arhitektuur....28
6.1 ActiveMQ listner....30
6.2 ActiveMQ writer....30
6.3 Satelite manager....30
6.4 TLE Getter....31
6.5 File Service....31
6.6 Flyover tracker....31
6.7 Antenna mover....31
6.8 Main....32
6.9 Vertical motor....32
6.10 Horizontal motor....33
7 Valideerimine....34

8 Edasinetöö....36
9 Kokkuvõte....37
Kasutatud kirjandus....38
Lisa 1 – ActiveMQ teema „ros_out” teema kuulaja MCS’is....39
Lisa 2 – Paraboolantenni juhitmissüsteemi veakoodid....40
Lisa 3 – Jälgitava satelliidi lisamise kulg....41

Jooniste loetelu

Joonis 1. Paraboolantenni mõõtmed [4, p. 12].....12
Joonis 2. TLE.....17
Joonis 3. Kordinaatidesüsteem [5, p. 25].....18
Joonis 4. Peamise kontrolleri mudel [4, p. 22].....20
Joonis 5. MCS’i arhitektuur.....23
Joonis 6. MCS’ist saadetud sõnum JSON kujul.....26
Joonis 7. MCS’i saadetud päringu vastus JSON kujul.....26
Joonis 8. MCS’i saadetud päringu vastus JSON kujul.....27
Joonis 9. ActiveMQ sõnum MCS’i.....27
Joonis 10. Paraboolantenni juhtimissüsteemi arhitektuur.....29
Joonis 11. *PowerFlex 40* sagedusmuundur koos mootoriga.....35
Joonis 12. See on programmikoodi lisamise näide.....39
Joonis 13. Jälgitava satelliidi lisamise kulg.....41

Tabelite loetelu

Tabel 1. Satelliidi ülelennu informatsioon.....21
Tabel 2. ROS’i veakoodid.....40

1 Sissejuhatus

TTÜ100 tudengisatelliidi programmiga tehti algust 2014. aasta sügisel. Programmi peamiseks eesmärgiks on valmistada ning edukalt LEO ehk Maa-lähedasel orbiidile saata kaks 1U suurust nanosatelliit nimedega „Koit" ja „Hämar", mis sooritaks seal Maa kaugseiret. „Koit" saateti orbiidiline 2019. aasta juunis ning „Hämara" on planeeritud orbiidile saata 2020. aasta teisel poolel. Lisaks satelliitidele on programmi raames tarvis luua tarkvara missiooni juhtimiseks MCS ja ülessead infrastruktuur satelliidiga kommunikatsiooniks. [1]

Maa kaugseire sooritamiseks omab satelliit kahte kaamerat: NIR ja RGB. NIR kaamera võimaldab koguda informatsiooni taimestiku ja kliima kohta. 5 Mpx'ine RGB kaamera on mõeldud Maast värviliste piltide tegemiseks. [2]

Kommunikatsiooni eesmärgil on paigaldatud Mektori maja katusele kaks antenni: K_u-riba[1] tüüpi paraboolantenn ja Yagi[2] tüüpi antenn. Yagi antenn, mis võimaldab poolduplekskommunikatsiooni 435MHz ultrakõrgsagedusel on mõeldud missiooni käskude ja tarkavauuenduste üleslaadimiseks ning väiksemahuliste andmete allalaadimiseks. 10.5GHz superkõrgsagedusala töötava parapoolantenn on suuremahuliste andmete, nagu näiteks Maa kaugseire käigus tehtud piltide, allalaadimiseks [3].

1.1 Probleem

Antud töö on jätk Rain Vingi magistritööle „Juhtimissüsteem TalTech-i satelliidi maajaama Ku riba antennile", mis on omakorda jätk Rasmus Tomseni lõputööle "Juhtimissüsteemi loomine TTÜ satelliidi maajaama paraboolantennile" ning lahendus probleemile, mis tuleneb vajadusest allalaadida Maa kaugseire teostamisel tehtud pilte. Nagu eelmises peatükis sai mainitud, siis on sellel otstarbel paigaldati Mektori maja

1 https://en.wikipedia.org/wiki/Ku_band

2 https://en.wikipedia.org/wiki/Yagi%E2%80%93Uda_antenna

katusele paraboolantenn, mida on tarvis suunata satelliidile täpsusega ±0.2° satelliidi ülelennu ajal. Selleks, et oleks võimalik suunata paraboolantenni piisava täpsusega on esmalt tarvis teada, milline on satelliidi orbiit ning selle põhjal välja arvutada, millal on ilmub satelliit paraboolantenni vaateväljas.

Lisaks TTÜ100 tudengisatelliidi programmi jaoks kiire andmeside kiirusega allalaadimise funktsionaalsuse pakkumisele on plaanis paraboolantenni aega välja rentida kolmandatele osapooltele ressursi optimaalseks kasutamiseks. Paraboolantenni aja väljarentimiseks peab eksisteerima võimalus jälgida mitme satelliidi orbiite ning iga orbiidi jaoks arvutada ülelennu aeg. Lisaks peab eksisteerima võimalus lisada ning eemaldada jälgitavaid satelliite.

Konkreetse paraboolantenni puhul on tegu antenniga, mille parabooli diameeter on 5 meetrit, kõrgus koos jalgadega on 6 meetrit ja kogu kaal on ligikaudu 8 tonni. 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. Liigutamiseks on kaks *Omron 3G3MX2-A4007-E*[1] sagedusmuundurit koos mootoritega. Üks parabooli horisontaalseks liigutamiseks ja teine vertikaalseks [4].

1 http://www.ia.omron.com/products/family/3164/dimension.html

Joonis 1. Paraboolantenni mõõtmed [4, p. 12]

1.2 Eesmärgid

Käesoleva töö eesmärgiks on luua süsteem paraboolantenni juhtimiseks, mis on võimeline täitma järgnevaid ülesandeid:

- Süsteem peab koguma infot etteantud satelliitide orbiitide kohta
- Süsteem peab arvutama iga satelliidi jaoks ülelennu aja, mis jääb paraboolantenni vaatevälja

- Süsteem peab liigutama paraboolantenni sellisel viisil, et see oleks pidevalt suunatud satelliidile ülelennu ajal
- Süsteemi poolt jälgitavaid satelliite peab olema võimeline lisada ning eemaldada läbi MCS'i

Töö on jagatud viite etappi.

Esimeseks on luua funktsionaalsus, mis kogub informatsiooni etteantud satelliitide orbiitide kohta ning salvestab selle edasisteks arvutusteks. Informatsiooni hoidmine orbiitide kohta peaks toimuma kujul, mis võimaldaks edasistel sammudel toimid ka selle funktsionaalsuse kadumisel.

Teiseks on salvestatud orbiidi põhjal välja arvutada, millal toimub järgmine vaadeldav ülelend. See peaks toimuma sõltumatu protsessina ning pidama meeles arvutatud tulemusi kuni arvutatud ülelennu lõpuni.

Kolmandaks on satelliidi ülelennu ajal paraboolantenni liigutamine kasutades horisontaalset ning vertikaalset mootorit selliselt, et parabool olema pidevalt suunatud satelliidile. Protsess peaks toimima viisil, et seda oleks võimalik peatada ohutuse eesmärkidel.

Neljandaks on tarvis luua sõnumivahetus MCS'iga. MCS'il peab olema võimalik lisada ning eemdalda jälgitavaid satelliite ning süsteemil peab olema moodus teavitada MCS'i probleemidest, mis tekivad. Kuna uus süsteem saab olema eraldiseisev MCS'ist siis on tarvis missiooni edukaks tööks teavitada koheselt teavitada tekkinud probleemidest.

Viiendaks on tarvis veenduda, et juhtimissüsteem tervikuna töötab. Kuna Mektori maja katusel oleva paraboolantenni kaabeldus on poolik, siis ei ole võimalik süsteemi testida sellega vaid on kaasaskantav sagedusmuundur koos mootoriga.

2 Kasutatud tehnoloogiad

Eesmärkide saavutamiseks on kasutatud mitmeid tehnoloogiaid ja tarkvarasid, millest antud peatükk annab pealiskaudse ülevaate.

2.1 ROS

ROS ehk *Robot Operating System* on vaba lähtekoodiga raamistik, mis pakub erinevaid teenuseid, mille hulka kuuluvad: madalama tasemelise riistvaraga suhtlemine, tarkvarapaketide haldamine ja sõnumivahetus erinevate protsesside vahel. Kuigi raamistik on eelkõige mõeldud juhtimise tarkvara loomiseks robotite jaoks on seda võimalik kasutada laialdasemalt eri probleemide lahendamiseks. Seda tänu raamistiku paindlikkusele ning avatud lähetekoodile, mis võimaldab lihtsalt lisada funktsionaalsust. [6], [7]

ROS võimaldab lihtsalt siduda erinevaid ülesandeid täitvaid koodi rutiine ning organiseerida nendevahelist sõnumite vahetust. Koodi rutiinidest moodustatakse ROS-is *node*[1] ehk sõlm, mis koondab need kokku üheks ROS-i poolt hallatavaks rakenduseks ning annab igale sõlmele ligipääsu ROS-i sisemisele sõnumite vahetuse teenusele. Sõnumite vahetus teenus on ROS-is ülesehitatud *topics*[2] ehk teemadena. Teema kujutab endast fikseeritud nime, millel on *subscriber*[3] ehk kuulaja ning *publisher*[4] ehk postitaja. Kui postitaja postitab sõnumi teemas siis saadetakse see igale kuulajale. Sõnumite vahetus toimub asünkroonselt. [7]

1 https://wiki.ros.org/Nodes
2 https://wiki.ros.org/Topics
3 https://wiki.ros.org/roscpp/Overview/Publishers%20and%20Subscribers
4 https://wiki.ros.org/roscpp/Overview/Publishers%20and%20Subscribers

2.2 ActiveMQ

ActiveMQ on Apache poolt arendatud Java põhine raamistik sõnumite vahetamiseks. ActiveMQ töötab teemade põhiselt. Teema on kindla nimetusega järjestus sõnumeid, millest on erinevatel rakendustel võimalik sõnumeid lugeda ning neid sinna lisada. Sõnumi vahetus toimub asünkroonselt. [8]

2.3 STOMP

Stopm ehk *Streaming Text Oriented Message Protocol* on protokoll, mis põhineb TCP/IP protokollil ning on mõeldud teksti edastus. Protokollis on implementeritud järgnevad käsud: *CONNECT, SEND, SUBSCRIBE, UNSUBSCRIBE, BEGIN, COMMIT, ABORT, ACK, NACK, DISCONNECT*. Kuna Stomp on implementeertitud sõnumi vahetus raamistike nagu ActiveMQ, *RabbitMQ*[1], *OpenMQ*[2] ja teiste sarnaste puhul siis üks olulisemaid protokolli otstarbeid on sõnumite lugemine ning sõnumite saatmine vastavatesse raamistikesse. [9]

1 https://www.rabbitmq.com/

2 https://javaee.github.io/openmq/

3 Satelliidi orbiidi jälgimine ja ülelennud arvutamine

Paraboolantenni juhtimissüsteemi kõige olulisemaks osaks on võimekus liigutada parabool, et see oleks ülelennu ajal pidavalt suunatud satelliidile. Selleks, et seda sooritada on tarvis teada, milline on satelliidi asukoht antenni suhtes igal ajahetkel. Teades satelliidi asukohta on võimalik selle põhjal välja arvutada, millal jõuab satelliit paraboolantenni vaatevälja, kaua ta seal püsib ning mööda millist trajektoori toimub ülelend.

3.1 Satelliidi orbiit

Erinevate taevakehade, nagu näiteks satelliitide, orbiitide leidmine ei ole lihtne tegevus, see nõuab spetsiaalset riistvara, mis võimaldab kaardistada objektide liikumist ümber Maa ehk nende orbiiti. Abiks on planeeritud satelliidi orbiidi informatsioon, mis antakse satelliidi orbiidile toimetaja poolt, aga tegu on planeeritud orbiidiga, mis võib muutuda vastavalt sellele kuidas satelliidi kosmosesse paiskamine kulgeb. Selleks, et paraboolantenni suunamiseks piisava täpsusega ei pruugi sellest piisata. TTÜ100 tudengisatelliidi orbiidi ei hakka olema ka konstantsed, sest satelliidid on planeeritud Maa-lähedasele orbiidile. Lähedus Maale tähendab seda, et orbiiti mõjutab Maa atmosfääri hõõrdetegur. [10]

Siin kohas tuleb appi asjaolu, et ümber Maa tiirleb umbes 500000 objekti, mis omavad läbimõõtu 1cm või rohkem [14]. Selleks, et tööstuslikud satelliidid kõikide nende erinevate objektidega orbiidil kokku ei põrkuks on NORAD[1] poolt ülesseatud radarite ja satelliitide võrgustik orbiitide kaardistamiseks. NORAD'i poolt kogutav info, mida avalikustatakse kord päevas, on piisavalt täpne, et selle põhjal on juhtimissüsteemil võimalik välja arvutada ülelennud algusaeg koos ülelennu trajektoor. Orbiidi kirjeldamiseks kasutab NORAD TLE formaati. TLE formaat on 1960's väljatöötatud ning hoiab info kahel 69 sümboliga real [11].

1 https://en.wikipedia.org/wiki/North_American_Aerospace_Defense_Command/

Joonis 2. TLE

Paraboolantenni juhtimissüsteem ei küsi jälgitavate satelliitide TLE'sid otse NORDA'i käest vaid sooritab päringuid Space-Track.org[1] portaali, kasutades selleks nendepoolt pakutavat API'i teenust. Space-Track.org tagastab satelliidi kataloogi numbri[2] ehk satelliidi ID aluse tema TLE. Päringu poolt saadud tulem salvestatakse eraldi faili. Otsus hoida TLE'd süsteemis eraldi failis võrreldes lokaalses mälus tuleneb sellest, et see tagab TLE'd kasutavatele protsessidele ligipääsu infole ka siis kui TLE pärimise protsess peaks seiskuma või sellele tehti taaskäivitus. Jälgitavate satelliitide kataloogi numbreid hoitakse sammuti eraldi failis samadel põhjustel nagu ka TLE informatsiooni. TLE enda pärimine toimub ühel kolmest juhust:

1. Süsteemi käivitamiel
2. Perioodiliselt iga 4 tunni järel
3. Siis kui jälgitavate satelliitide ID'd muutuvad

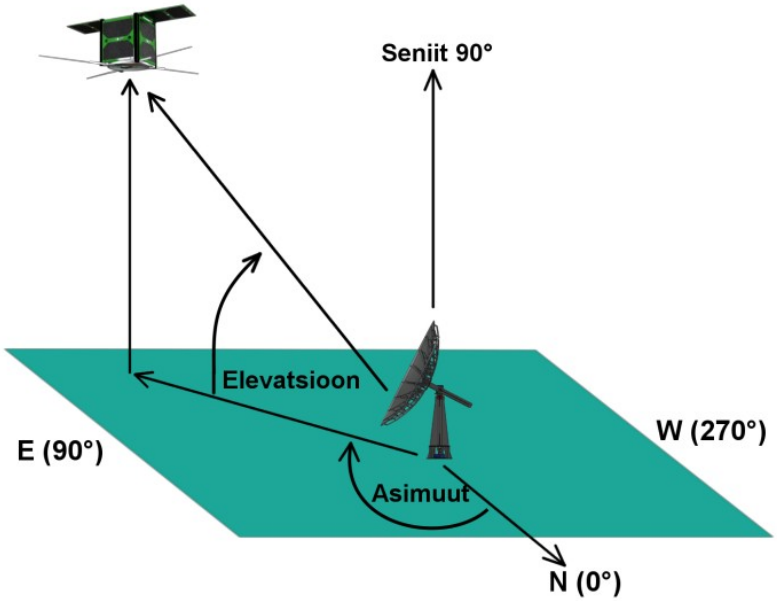
Süsteem käivitusel on tarvis veenduda, et satelliitide kataloogi numbrid ei ole muutunud ning failis olevad TLE'd oleks vigadeta ja aktuaalne. Selle tagamiseks päritakse värske TLE süsteemi käivitamisel ning märgitakse maha TLE uuendamise aega pluss neli tundi. Mahamärgitud aeg on see, millal on tarvis sooritada järgmine TLE uuendus. Nagu eelnevalt sai mainitud siis NORAD avalikustab uuendatud TLE info kord päevas ning nelja tunnine tsükkel tagab selle, et vananenud TLE'l tegutsemine on minimaalne. TLE uuendus jälgitavate satelliitide loetelu uuendamisel toimub süsteemis TLE pärimise aja hetkeajaks muutmisega, et lisanduks uute satelliitide TLE või eemalduks üleliigne TLE. Peale TLE'de uuendamist teavitab protsess kõiki teisi protsess, mis kasutavad TLE'd.

1 https://www.space-track.org/
2 https://en.wikipedia.org/wiki/Satellite_Catalog_Number

3.2 Ülelennu leidmine

Omades satelliidi TLE'd on süsteemil võimalik välja arvutada periood, millal satelliit ilmub paraboolantenni vaatevälja ning milline on satelliidi ülelennu trajektoor. Selle arvutuse sooritamiseks võrdles Rasmus Tomsen enda lõputöös kolme raamistiku: Orbitron[1], Pyephem[2] ja Gpredict[3]. Pyephemi kasuks langes otsus kuna võrreldes teise sarnaste teekidega osutusid arvutused väga täpseks ning seda on mugav kasutada [5].

Pyephem arvutab satelliidi TLE ja paraboolantenni GPS kordinaatide alusel aja ning asimuudi, millal satelliit ilmub paraboolantenni vaatevälja, aja ning asimuudi, millal satelliit lahkub vaateväljast ja aja millal satelliit saavutab maksimaalse elevatsiooni.



Joonis 3. Kordinaatidesüsteem [5, p. 25]

Süsteem sooritab ülelennu arvutusi kolmel juhul:

1. Protsessi käivitamisel kui on TLE on olemas
2. Siis kui saabub teavitus TLE uuenduse kohta

1 http://www.stoff.pl/
2 https://rhodesmill.org/pyephem/
3 http://gpredict.oz9aec.net/

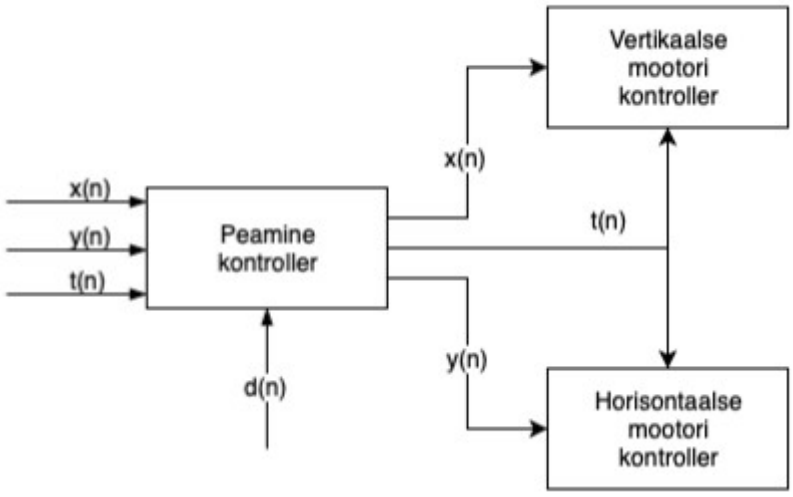
3. Siis kui aktiivne ülelend on lõppenud

Ülelennu arvutuse puhul on ainukeseks muutuvaks faktoriks TLE, kuna paraboolantenni asukoht ei muuta. Tänu sellele faktile piisab sellest kui leida järgmine uue TLE saabumisel. TLE saabudes arvutab süsteem ülelennu algusajal iga satelliidi TLE jaoks, leiab nende hulgast esimese, mis vastab tingimustele. Kui aktiivne ülelend on läbi siis arvutatakse kõik uuesti. Käivitamisel arvutuse sooritamine on vajalik selleks, et kui peaks olema tarvis protsessile sooritada taaskäivitus.

Käivitusel kontrollitakse, kas TLE failis on TLE informatsioon. Kui ei siis jääb protsess ootama teavitust uue TLE saabumisest ja kui on olemas siis sooritatakse arvutused. Ülelennu puhul on oluline, et selle käigus saavutaks satelliit minimaalse elevatsiooni, milleks on 5°. Madala ülelennu puhul ei paista satelliit puude tagant välja või teeb seda nii lühikest aega, et andmevahetust ei ole võimalik algatada.

4 Paraboolantenni suunamine

Rain Vink, enda magistritöö raames, kirjutas tarkvara kasutades ROS'i, mis on võimeline liigutama paraboolantenni vertikaalselt ja horisontaalselt kasutades vastavaid sagedusmuundureid ning mootoreid. Loodud tarkvara koosneb kolmest ROS'i sõlmest: „Main", „Vertical motor" ja „Horizontal motor". „Main" on ROS'i sõlm, mis ootab käske paraboolantenni liigutamiseks, seda soovitud horisontaalse ja vertikaalse nurgana kraadides ning ajas sekundites, mille jooksul tuleks nurk saavutada. Peale uue käsu saamist sooritakse selle valideerimine, et olla kindel võimekusest seda sooritada ning seejärel eraldatakse horisontaalne ja vertikaalne nurga muut ajas „Vertical motor" ja „Horizontal motor" sõlmele. „Vertical motor" ja „Horizontal motor" tegelevad vastavate mootorite juhtimisega kasutades selleks modbus[1] protokolli [4].



Joonis 4. Peamise kontrolleri mudel [4, p. 22]

1 https://en.wikipedia.org/wiki/Modbus

Paraboolantanni liigutamiseks sai loodavasse juhtimissüsteemi lisatud Rain Vingi kolm põhilist ROS'i sõlme: „Main", „Vertical motor" ja „Horizontal motor", lisaks sõlmedela ka „coils.csv" ja „registers.csv" failid, kus on vajalik info registritesse kirjutamiseks ning *python*'i klassid „PID", „RegisterElement" ja „RegisterMapping", mille puhu on tegu abistavate klassidega. Käskude lihtsamaks edastamiseks muutsin „Main" sõlme kuulamaks minupoolt loodud ROS'i sõlmest „Antenna Mover" saadetud sõnumeid. Eelnevalt tulid sõnumid *flask*[1]'is loodud veebirakendusest [4]. Lisasin „Main" ka veahalduse.

Tabel 1. Satelliidi ülelennu informatsioon

Välja nimi	**Väärtus**	**Kireldus**
rise_time	18:05:53	*UTC*[2] aeg, millal satelliit ilmub parabooli vaatevälja
rise_azimuth	53.0	Asimuut kraadides, kust satelliit ilmub parabooli vaatevälja
set_time	18:17:43	*UTC* aeg, millal satelliit lahkub parabooli vaateväljast
set_azimuth	323.4	Asimuut kraadides, kust satelliit lahkub parabooli vaateväljast
max_time	18:11:19	*UTC* aeg, millal satelliit saavutab maksimaalse elevatsiooni
max_elevation	13.1	Maksimaalne elevatsioon kraadides

Kasutades TLE põhjal arvutatud satelliidi ülelennu informatsiooni on võimalik välja arvutada horisontaalne ja vertikaalne nurgamuut ajas paraboolantenni liigutamiseks. Horisontaalne liikumine on koosneb asimuudist, mis on tarvis saavutada kindla aja jooksul sekundites. Ajaks on aeg, millal satelliit lahkub parabooli vaateväljast miinus aeg, millal satelliit ilmub parabooli vaatevälja. Vertikaalne liikumine koosneb kahest faasist. Tõusva faasi jooksul on tarvis saavutada maksimaalne elevatsioon aja jookusul, mis kulub ülelennu algusest maksimaalse ülelennu saavutamise ajahetkeni sekundites. Langeva faasi jooksul on tarvis saavutada 0

1 https://www.fullstackpython.com/flask.html

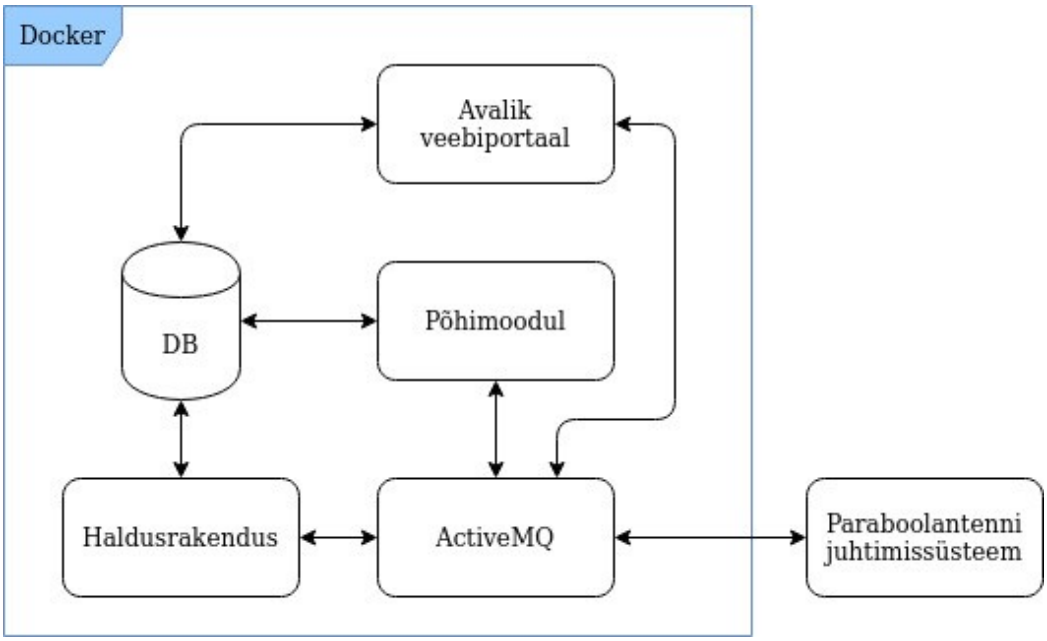
2 https://en.wikipedia.org/wiki/Coordinated_Universal_Time

elevatsioon aja jookusul, maksimaalse elevatsiooni ajast ajani, millal satelliit lahkub parabooli vaateväljast. Paremaks kontrolliks, ei edastata terve ülelennu trajektoori korraga vaid antakse 5 sekundiliste etappidena. Ehk siis nurk, mis on tarvis saavutada viie sekundi jooksul.

Kõikide nende arvutuste jaoks on eraldi protsess, mis pidavalt kontrollib saabunud ülelennu informatsiooni. Lähtudes ajast, millal satelliit ilmub parabooli vaatevälja, kuni viis minutit enne seda suunatakse paraboolantenn asimuudi suunas, kust satelliit ilmub vaatevälja. Suunamise ajaks on ettenähtud 30 sekundit lähtudes Rain Vingi tööst [4]. Kui ülelennu info saabub vähem kui 30 sekundit enne ülelennu algust siis suunatakse paraboolantenn asendisse, milles peaks viibima satelliit 30 sekundi pärast. Ülelennu algus momendil arvutatakse välja nurgamuudud koos ajaga, mille jooksul see on tarvis sooritada ning edastatakse need käsuna „Main” sõlme. Maksimaalse elevatsiooni saavutamise ajal edastatakse languse nurk koos ajaga, mille jooksul see on tarvis saavutada.

5 Liidestus MCS’iga

Nagu nimi juba ütleb siis on missioonijuhtimise tarkvara tsentraalse tarkvaraga, mida kasutatakse kogu missiooni haldamiseks. MCS tegeleb satelliidi poolt saadetud andmete töötlemise ning haldamisega, prioritiseerib kommunikatsiooniside sõnumeid, haldab kommunikatsiooniside sessioone üle mitme ülelennu, visualiseerima satelliidi asukoha ja veel palju muud. MCS ise koosneb kolmest *SpringBoot*[1] rakendusest, postresql andmebaasist ja ActiveMQ’st sõnumite vahetamiseks. Kõik *SpringBoot* rakendused, ActiveMq ja andmebaas jooksevad *docker*[2] konteinerites. Kolm rakendust, millest MCS koosneb on: avalik veebiportaal, mis on avalikkusele mõeldud informeeriv veebileht TTÜ100 programmi kohta, põhimoodul, mis hoiab endas tuumik funktsionaalsust ja haldusrakendus, mis on kasutajaliides missiooni juhtimiseks. [12]



Joonis 5. MCS’i arhitektuur

1 https://spring.io/projects/spring-boot/
2 https://www.docker.com/

Kuna kogu missiooni juhtimine toimub läbi MCS'i siis on oluline, et paraboolantenni juhtimissüsteem oleks võimeline suhtlema MCS'iga. Seda kahel põhjusel:

1. Nagu kogu ülejäänud missiooni juhtimine peaks satelliitide lisamine ja eemaldamine, mida paraboolantenni juhtimisesüsteem jälgib, käima sammuti läbi MCS'i

2. Paraboolantenni juhtimisesüsteemil peab olema võimalus tekkinud vigadest teavitada võimalikult kiirelt missiooni juhtimise meeskonda

Lähtudes MCS'i arhitektuurist oleks võimalik sõnumivahetust paraboolantenni juhtimisesüsteemiga ülesehitada kolm erineval viisil.

1. Kasutades REST[1] teenuseid

2. Läbi socket[2] ühenduse

3. Läbi ActiveMQ

5.1 REST teenus põhine ühendus

REST teenuse peale ülesehitamine sõnumivahetus kujutaks endast nii MCS' kui ka paraboolantenni juhtimissüsteemi poolseid veebiteenuseid kuhu on võimalik sooritada erinevaid HTTP[3] päringud. Näiteks oleks paraboolantenni süsteemi poolel *URL*[4] kuhu oleks võimalik teha POST[5] päring satelliidi lisamiseks või DELETE[6] päring satelliidi eemaldamiseks ja MCS'i poolel oleks *URL* parabooli süsteemi vigadest teavitamiseks.

MCS'i poolel on *SpringBoot* raamistiku kasutamisest tulenevalt funktsionaalsus olemas, mis võimaldab lihtsalt REST teenuse otspunktide ülesseadmist. Paraboolantenni süsteemis tuleks kasutusele võtta lisa raamistik nagu näiteks *flask*. Isegi võimekuse olemasolul on tarvis mõlemas süsteemis suhteliselt palju arendus tegevust

1 https://en.wikipedia.org/wiki/Representational_state_transfer
2 https://en.wikipedia.org/wiki/Network_socket
3 https://en.wikipedia.org/wiki/Http
4 https://en.wikipedia.org/wiki/Uniform_Resource_Locator
5 https://en.wikipedia.org/wiki/POST_(HTTP)
6 https://en.wikipedia.org/wiki/DELETE_(HTTP)

sõnumivahetuse ülesseadmiseks ning haldamiseks. Iga uus funtsionaalusl tähedaks uut URL'ide loomist.

5.2 Socket'i põhine ühendus

Socket'i põhist sõnumivahetus oleks lihtsam üles seada ja hallata kui REST'i teenuse põhist. Sõnumivahetuseks oleks tarvis luua ainult kaks TCP/IP protokolli põhist ühendust, üks MCS'ist paraboolantenni süsteemi ja teine vastassuunaline ning kogu vajalik funktsionaalsus on mõlemas süsteemis olemas. Süsteemida vahel liikuvad sõnumid omaksid tunnuseid, mille alusel oleks võimalik tuvastada soovitud tegevust süsteemis lisaks infole.

5.3 ActiveMQ põhine sõnumivahetus

MCS'i sisemine sõnumite vahetus toimub kasutades ActiveMQ'd ja ActiveMQ's on tugi olemas STOMP protokolli joaks, mis võimaldab välisest süsteemist sõnumeid lugeda ning postitada erinevatesse teemadesse. Kasutades seda lähenemist oleks MCS'i poolel sõnumivahetus parabooli süsteemiga sama ülesehitusega nagu MCS'i sisemine sõnumivahtus. Parabooli poolel oleks tarvis kasutausele võtta STOMP protokolli jaoks teenus ning ActiveMQ'sse oleks tarvis lisada uued teemad.

5.4 Sõnumivahetuse teostus

Kuna sõnumite vahetus ActiveMQ'ga langeb kõige paremini kokku üldise MCS'i arhitektuuriga ning kasutades *stomp.py*[1] teeki parabooli süsteemis ei ole tarvis sooritada mahukamat arendust kui teiste lahenduste puhul. Lähtudes nendest kahest asjaolust sai sõnumivahetuseks loodud kaks ActiveMQ teemat „ros_is" ja „ros_out". „ros_in" on teema mida kuulab parabooli süsteem kasutades stomp.py'd ning „ros_out" on teema mida kuulab MCS'i teenus (vt. Lisa 1). Kõik sõnumid liiguvad läbi nende kahe teema ning omad tunnust mille alusel sooritatakse vastavas süsteemis tegavust. Kuna STOMP protokoll toetab teksti edastust siis on sõnumite struktuuriks JSON teksti kujul. JSON kuna seda väga mugav mõlemas süsteemis parsida ning on ka kenasti inimloetav.

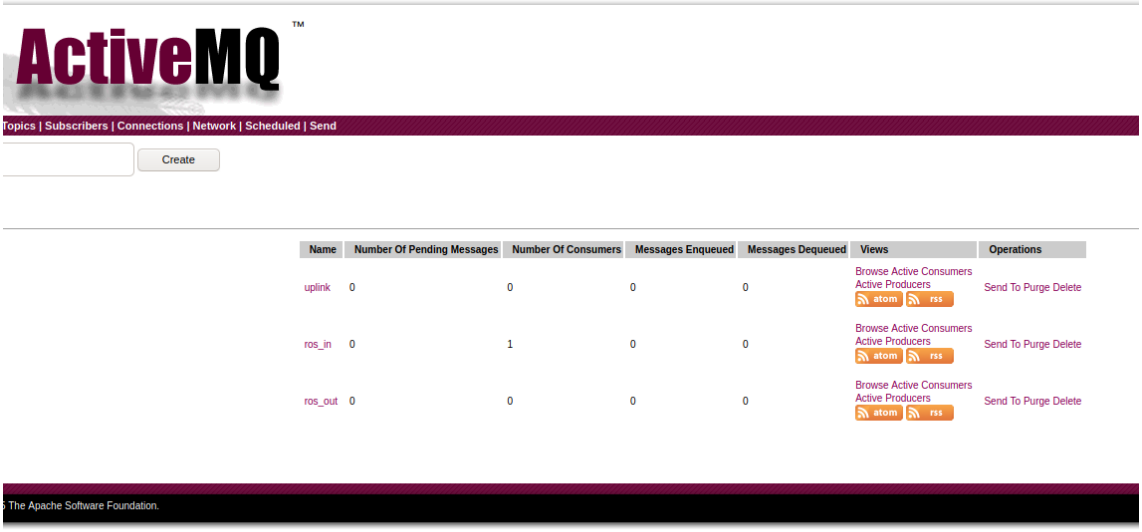
1 https://jasonrbriggs.github.io/stomp.py/index.html.

MCS’i poolt saadetava sõnumi struktuur omab paraboolantenni juhtimissüsteemi alamsüsteemi viidet, mis tegevuse peaks sooritama („node”), tegevuse kood („action”), andmed („satellite_ids”) ja sõnumi ID („request_id”) mille järgi on võimalik siduda hiljem vastusega. Kuna *python* ei ole tugevalt tüübitu keel siis andmete välja asendada erinevatega, see tähendab välja („satellite_ids”) asemel võib olla teise nime ja andmestruktuuriga väli.

Joonis 6. MCS’ist saadetud sõnum JSON kujul.

MCS’i saadetud sõnumis on sõnumi ID („request_id”), mis viitab MCS’ist saadetud sõnumile või on veakood (vt. Lisa 2), mis kindlale parabooli alamsüsteemile, andmete („data”), mis on teksti massiiv ja tunnus („ok”), mis ütleb, kas tegu on veaga. Andmevälja massiivi või olla erinev informatsioon nagu näiteks vigade kirjelduse, satelliidi tunnused või muu informatsioon.

Joonis 7. MCS’i saadetud päringu vastus JSON kujul.



Joonis 8. MCS'i saadetud päringu vastus JSON kujul.

Joonis 9. ActiveMQ sõnum MCS'i.

6 Paraboolantenni juhtimissüsteemi arhitektuur

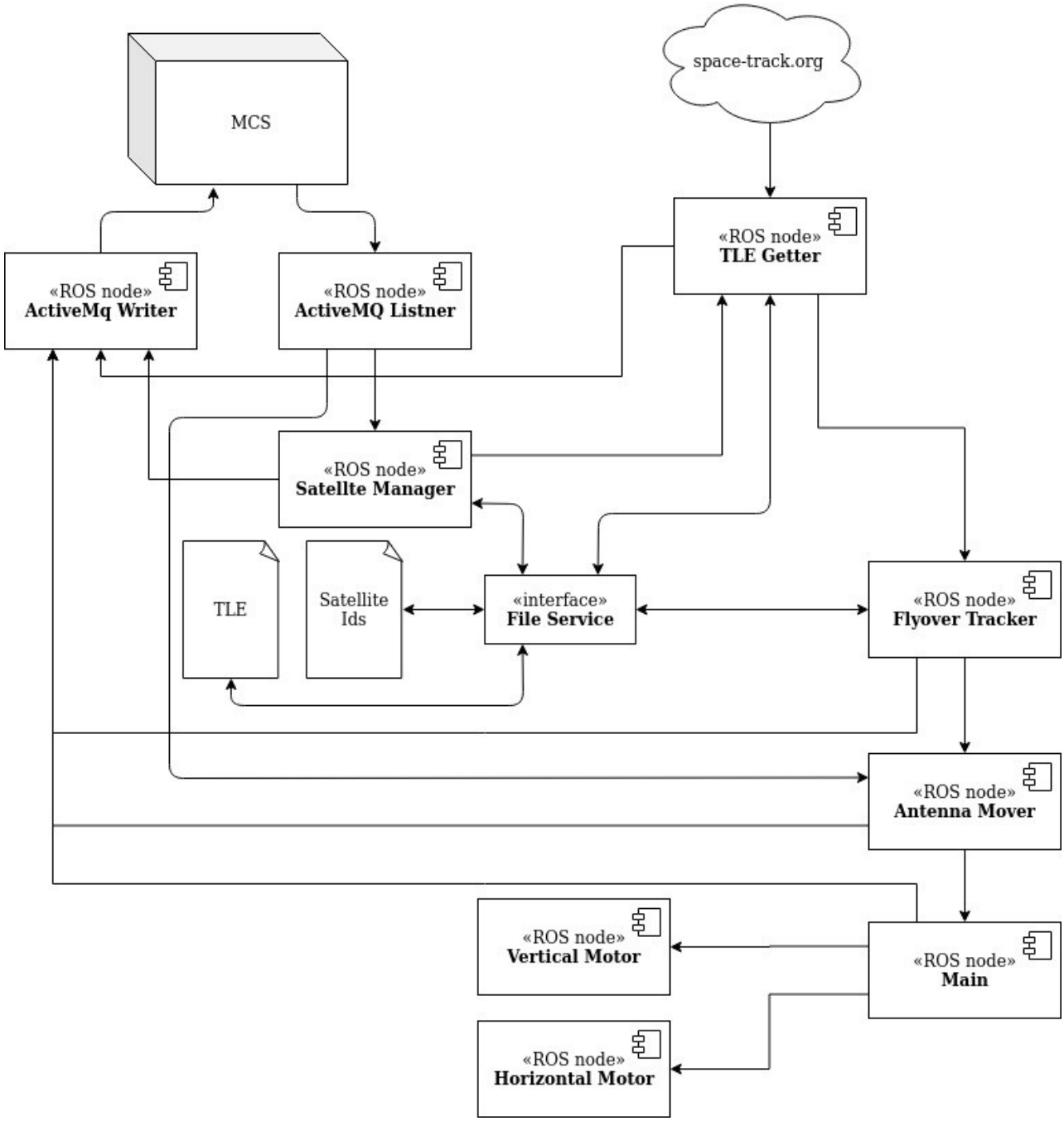
Eelnevates etappides sai loodudud erinev funktsionaalsusi, mis tarvis, et paraboolantenni pidavalt suunatud satelliidile siis kui satelliit on paraboolantenni vaateväljas ning funktsionaalsus suhtluseks MCS'iga. See peatükk annab ülevaate kuidas eelnevates etappides loodud funktsionaalsus sai koondatud ühtseks juhtimissüsteemiks kasutades ROS'i raamistikku. ROS'i kasutamise idee sai alguse Rasmus Tomseni lõputööst ning Rain Vink jätkas enda magistritöös selle kasutamist. ROS ise on hea raamistik juhtimissüsteemi loomiseks mehaanilistele süsteemidele ning kuna Rain Vink implementeeris sagedusmuundurite registritesse kirjutamise kasutades ROS'i siis ei ole mõistlik implementeerida terviklik juhtimissüsteem mingis teise raamistikus ning hakata seda siis ROS'iga siduma. Tulemuseks on süsteem, mis koosneb 9-st ROS'i sõlmest ning ühest abi teenusest.

Süsteemi arhitektuuri loomisel proovisin saavutada järgnevaid tulemusi:

1. Süsteemi peab olema lihtne laiendada uute sõlemedega.
2. Süsteemi peab olema lihtne testida.
3. Süsteemi peab olema lihtne hallata ja seadistada

Lihtsa laiendatavuse saavutamiseks lähtusin ma põhimõttest, et üks funktsionaalsus sõlme kohta ning kattuv äriloogika koondada staatiliste funktsioonidesse eraldi faili. Hoides äriloogikat staatilistes funktsioonides on uutel sõlmedel lihtne ligipääs funktsionaalsusele. Suhtlus erinevate sõlmede vahe toimub täielikult kasutades ROS'i enda sõnumivahetuse teenust. See tagab uute sõlmedelisandumisel on neil mugav ligipääsu informatsioonile, mida edastavad erinevad sõlmed. Üks funktsionaalsus sõlme kohta teeb ka testimise lihtsamaks. On võimalik süsteemi käivitada sõlmede kaupa ning anda üksikule sõlmele ette sisend ning kontrollida väljundit.

Süsteemi lihtsaks haldamiseks on kõik konstandi konfiguratsiooni parameetrid koondatud eraldi failidesse, millele on ligipääs kõikidel sõlmedel. Konfiguratsiooni parameetrid on võimalik kaasa anda ROS'i käivitus skriptiga. See võimaldab ka luua erineva konfiguratsiooniga skripte eri keskkondade jaoks.



Joonis 10. Paraboolantenni juhtimissüsteemi arhitektuur

6.1 ActiveMQ listner

„ActiveMQ Listner” on ROS’i sõlm, mis tegeleb ActiveMQ „ros_in” teema kuulamisega. Kui tuleb uus postitus teemasse loeb sõlm sealt sõnumi sisse, teisaldab JSON’i tekstikujulise esinduse *python dictionary*[1] tüüpi objektiks ning valideerib selle. Valideerimise käigus kontrollitakse, et kogu informatsioon oleks olemas. Õnnestunud valideerimise korral toimub omakorda teisaldus ROS’i sõnumivahetus objektiks ning seejärel postitakse sõnum ROS’i sõlme mille jaoks see sõnum mõeldud on. Veakorral koostatakse vastav veasõnum ning saadetakse see MCS’i.

6.2 ActiveMQ writer

„ActiveMQ writer” tegeleb sõnumite postitamisega ActiveMQ teemasse „ros_out”. Sõnumid, mis ROS’i sõlmede poolt välja saadetakse jagunevad kaheks, süsteemi sisemised vead, mille puhul on tarvis teavitada MCS’i ja vastus MCS’ist tulnud sõnumile. Kõik ROS’i poolt väljasaadetavad sõnumitest tehakse esmalt JSON, milles on päringu ID, tõeväärtus, mis ütleb, kas tegu on veaga ning andmete massiiv ja siis postitatakse see JSON tekstina ActiveMQ teemasse.

6.3 Satelite manager

Süsteemis peab olema võimalik muuta jälgitavaid satelliite, sellega tegeleb sõlm „satelite manager”. Sõlm lisab, kustutab, asendab ja tagastab satelliitide tunnustega faili sisu, mille jaoks TLE päritakse. Iga MCS’ist tulnud päring peab sisaldama „ADD_ID”, „REMOVE_ID”, „REPLACE_IDS” või „READ_IDS” käsku ning kõik peale „READ_IDS” käsu peavad sisaldama ka loetelu satelliidi tunnustega. Peale satelliidi tunnuste edukat uuendamist saadab sõlm vastuse MCS’i tunnuste hetkeseisuga ja teavitab TLE pärimise sõlme uutest tunnustest (vt. Lisa 3). Ebaõnnestumisel saadab vastava veateatega sõnumi MCS’i.

1 https://www.tutorialspoint.com/python/python_dictionary.htm

6.4 TLE Getter

„TLE getter” sõlmes koosneb kahes osast. Üks on sõlm ise ja teine on eraldiseisev protsess, mis käivitatakse sõlme käivitamisega. Eraldiseisev protsess tegeleb TLE pärimisega Space-Track.org API’lt. Sõlmes endas on loogika, mis kuulab sõnumeid satelliitide uuendamisest ning loogika, mis teavitab ülelennu arvutusega tegelevat sõlme TLE uuendamist.

6.5 File Service

„File Service” ei ole ise ROS’i sõlm vaid teenus, mida sõlmed kasutavad selleks, et failist lugeda, faili kirjutada või failist kustutada. Kõik faili muudatused, olgu selleks lisamine või kustutamine, toimuvad samamood. Esmalt loetakse faili sisu mällu, seejärel sooritatakse muudatused ning muudetud andmed kirjutatakse uute faili. Kui see protsess õnnestus siis asendatakse vana fail uuega. Selline lähenemine vähendab vigase faili tekkimise tõenäosuse.

6.6 Flyover tracker

Nagu „TLE getter” koosneb „Flyover tracker” sammuti kahes osast. Üks on sõlm ise ja teine on eraldiseisev protsess, mis käivitatakse sõlme käivitamisega. Eraldiseisev protsess kontrollib iga sekund, kas tarvis uuesti ülelende arvutada. Sõlm ise loeb TLE sisse kui saabub sõnum TLE uuenemisest, sõles toimub pyephem abi ülelennu arvutus. Sammuti hoitakse sõlmes arvutatud ülelendude infot ja sõlm edastab järgmise ülelennu antenni juhtivale sõlmele. Välja arvutaud järgmise ülelennu informatsioon saadetakse „Antenna mover” sõlme.

6.7 Antenna mover

„Antenna mover” on sõlm milles toimub paraboolantenni liigutamiseks vajaliku horisontaalse ja vertikaalse nurga muudu arvutus ajas. Esmalt paraboolantenni stardi asendisse liigutamiseks ning siis kui ülelennu algusaeg kätte jõuab, satelliidi suunamiseks. Arvutuste sooritamiseks on siis sõlmes sammuti eraldiseisev protsess,

mis arvutab ning edastab paraboolantenni nurga muudud aktiivse ülelennu informatsiooni põhjal. Kui „Flyover tracker” sõlmest tuleb järgmise ülelennu informatsioon siis selle algusaja põhjal sooritab sõlm erinevaid tegevusi. Kuni 5 minutit enne ülelennu algust toimub paraboolantenni suunamine ülelennu alguspunkti. Kui ülelennu alguseni on vähem kui 30 sekundit siis suunatakse parabool mitte ülelennu alguspunkti vaid juba arvutatud nurga alla. MCS’il on võimalik siis sõlme saata ka „ABORT” käsk, mis peatab paraboolantenni liigutamise.

6.8 Main

„Main” on Rain Vingu pool loodud sõlm, mis võtab vastu käske parabool antenni liigutamiseks. Sissetulnud käsud valideeritakse veendumaks, et sooritamine on võimalik ning seejärel. Eduka valideerimise korral loetakse käsust välja horisontaalne nurk ning aeg sekundites, mille jooksul on see tarvis saavutada ning vertikaalne nurk ja aeg sekundites, mille jooksul on see vaja saavutada. Vertikaalne ja horisontaalne nurk koos ajaga edastatakse vastavatele sõlmedele. Mina, selle töö raames, kohendasin sõlme sisendi formaati, et sõlm oleks võimeline sõnumeid vastu võtma „Antenna mover” sõlmelt ning lisa funktsionaalsuse MCS’ile vigadest teatamiseks.

6.9 Vertical motor

„Vertical motor” on sammuti Rain Vingu pool loodud sõlm, mis arvutab etteantud vertikaalse nuurga ja aja põhjal moortori pöörete kiiruse, mis on vajalik, et saavutada soovitud tulemus. Tulemus põhjal moodustatakse käsk sagedusmuunduri jaoks, mis juhib vertikaalset mootorit, ning see kirjutatakse selle registritesse. Sagedusmuunduri registritesse kirjutamiseks kasutatakse modbus’i.

6.10 Horizontal motor

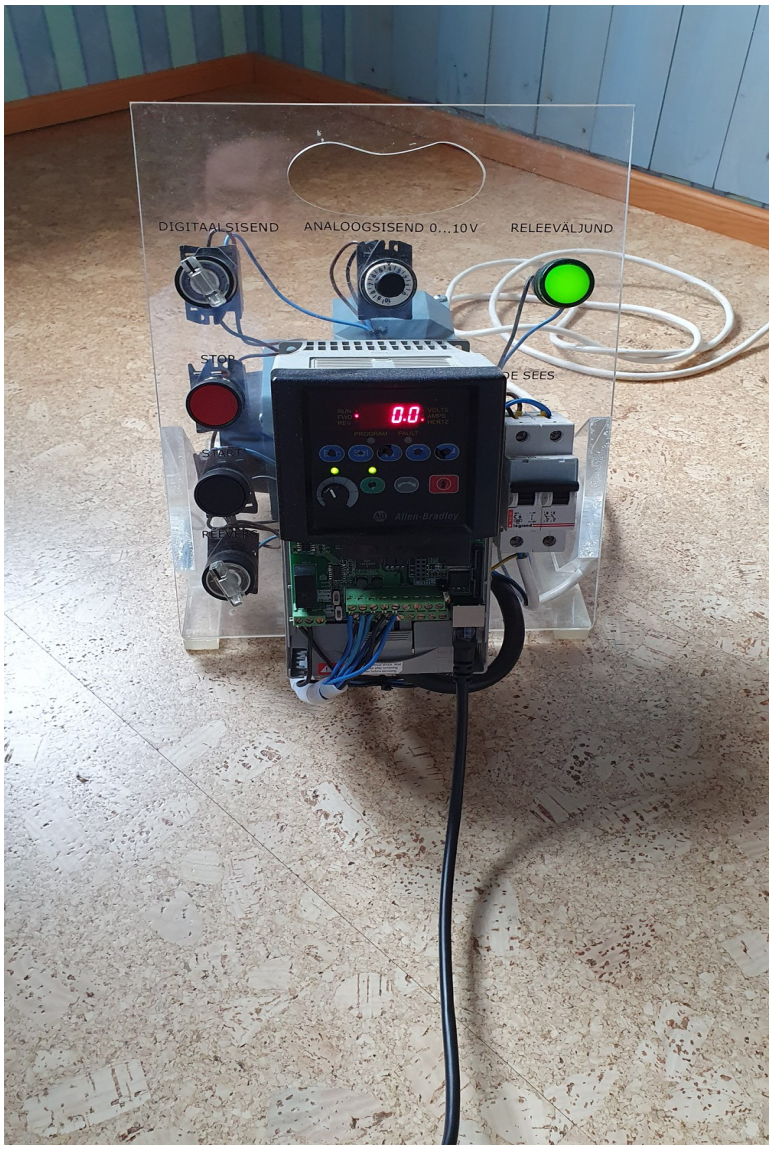
„Horizontal motor” on sammuti Rain Vingu pool loodud sõlm, mis arvutab etteantud horisontaalse nurga ja aja põhjal moortori pöörete kiiruse, mis on vajalik, et saavutada soovitud tulemus. Tulemus põhjal moodustatakse käsk sagedusmuunduri jaoks, mis juhib vertikaalset mootorit, ning see kirjutatakse selle registritesse. Sagedusmuunduri registritesse kirjutamiseks kasutatakse modbus’i.

7 Valideerimine

Nüüd kus erinevad komponendid said kokku seotud ühtseks süsteemiks on tarvis veenduda, et süsteem teeb seda, mida ta tegema peab ning seda ilma vigadeta. Juhtimissüsteemi valideerimiseks sooritasin manuaalset testimist.

Manuaalseks testimiseks kasutasin kaasaskantavat *PowerFlex 40*[1] sagedusmuundurit koos mootoriga. Selle sagedusmuunduri puhul on tegemist funktsionaalselt ekvivalentse sagedusmuunduriga, mida kasutatakse paraboolantenni liigutamiseks. Kuna Mektori maja katusel olev paraboolantenn ei ole täielikult kaabeldatud siis olnud võimalik juhtimissüsteemi testida reaalse paraboolantenniga ning tuli leppida kaasaskantavat *PowerFlex 40*[2] sagedusmuunduri ja mootoriga.

1 https://www.pesquality.com/manufacturers/allen-bradley-powerflex-40-repair
2 https://www.pesquality.com/manufacturers/allen-bradley-powerflex-40-repair



Joonis 11. *PowerFlex 40* sagedusmuundur koos mootoriga

Manuaalseks testimiseks ühendasin *PowerFlex 40* sagedusmuunduri arvutikülge kasutades *RJ45*[1] kaablit ja saatsin test käske sagedusmuundurile, mille tabloolt oli võimalik vaadata, kas käsk jõudis kohale. Kuna sagedusmuundureid testimiseks oli ainult üks siis tulu horisontaalseid ja vertikaalseid liigutamisi eraldi testida. Asi, mida testida ei õnnestunud oli parabooli suunamise täpsus. Selle sooritamiseks peaks olema reaalse paraboolantenni kaabeldus tehtud ning oleks tarvis leida satelliit orbiidil, mille signaali tugevust saaks mõõta.

1 https://en.wikipedia.org/wiki/RJ45

8 Edasinetöö

Kuna Mektori maja katusel olev paraboolantenn ei ole veel järgi ühendatud siis ei olnud võimalik ka sellega testida. Esimene samm edasises töös oleks testida juhtimissüsteemi reaalse paraboolantenniga uuesti veendumaks, et juhtimissüsteem tervikuna töötab ja vajadusel sooritada kohendusi süsteemile. Peale seda oleks tarvis leida sobiv satelliit orbiidil, mille peal saaks veendud paraboolantenni suunamise täpsuses. Mõistlik oleks lisada paraboolantenni liikumise täpsuse ning sujuvuse suurendamiseks kaks lisa mootorit üks mõlemale sagedusmuundurile, mis töötaksid töötaksid vastu olemasolevatele mootoritele. Loomuliult tuleks liigutamise algoritmi ka vastavalt kohendada. Lisa mootorid vähendaksid tunduvalt liigutamisest tulevat vibratsiooni ja tänu sellele suureneks liikumise täpsus ning sujuvus.

Tulevikus oleks tarvis ka täiendada juhtimissüsteemi funktsionaalsust. Lisada võiks manuaalse paraboolantenni juhtimise läbi MCS'i, et vajadus saaks paraboolantenni parkida ootasendisse ilma kohapeale minemata. Loomulikult tuleks lisada signaali töötlemise funktsionaalsul. Ilma signaali töötluseta ei võimalik paraboolantenniga satelliidilt andmeid alla laadida ning see funktsionaalsus peaks toetama sammuti satelliidi põhised sagedusi.

9 Kokkuvõte

Käesoleva töö eesmärk on luua TTÜ100 tudengisatelliidi programmi jaoks paraboolantenni juhtimissüsteem, mis oleks võimalik jälgida orbiidil lendavaid satelliite, mis on defineeritud MCS'ist hallatavas loetelus ning suunama paraboolantenni satelliidile ülelennu ajal. Suunatavaks täpsuseks peaks olema ±0.2°.

Töö käigus sai loodud arhitektuur koos implementatsiooniga juhtimissüsteemi jaoks kasutades ROS raamistikku. Valminud süsteem päris TLE informatsiooni jälgitavate satelliitide kohta, leidis selle põhjal järgmise ülelennu, mida jälgida ning arvutas paraboolantenni liigutamiseks horisontaalse ja vertikaalse nurga muudu ajas. Leitud nurgad paraboolantenni suunamiseks edastatakse ROS'i sõnumi näol paraboolantenni mootorid juhtivale süsteemile. Kõike seda teeb süsteem iseseisvalt ning kui probleem tekkib edastatakse see MCS'i inimesele lahendamiseks.

Töö esialgseks nõudeks oli paraboolantenni suunamine täpsusega ±0.2°, aga paraku selles veenduda ei olnud võimalik. Põhjuseks, miks seda ei olnud võimalik testida tuleneb sellest, et Mektori maja katusel asuv paraboolantenn ei ole täielikult kaabeldatud. Süsteemi sai testitud erinevatel viisidel, kuid nende põhjal ei ole kahjuks võimalik garanteerida, et reaalne paraboolantenn saavutab soovitud täpsuse.

Kasutatud kirjandus

[1] R. Gordon, „TTU100 Satellite,“ [Võrgumaterjal]. Available: https://ttu.ee/projektid/mektory-est/satelliidiprogramm-4/satelliidiprogramm/ttu-mektory-satelliidiprogramm/. [Kasutatud 07.05.2020].

[2] R. Gordon, „Pardakaamera,“ [Võrgumaterjal]. Available: https://ttu.ee/projektid/mektory-est/satelliidiprogramm-4/satelliidiprogramm/wp1-b-pardakaamera/. [Kasutatud 07.05.2020].

[3] R. Gordon, „TTÜ100 Satelliidi tutvustus,“ [Võrgumaterjal]. Available: https://www.ttu.ee/projektid/mektory-est/satelliidiprogramm-4/satelliidiprogramm/. [Kasutatud 07.05.2020].

[4] R. Vink, „Juhtimissüsteem TalTech-i satelliidi maajaama Ku riba antennile,“ [Võrgumaterjal]. Available: https://digikogu.taltech.ee/et/Item/9093d30e-163f-48a5-b210-640c46ac123a.

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

[6] „Core Components,“ Open Source Robotics Foundation, [Võrgumaterjal]. Available: http://www.ros.org/core-components/. [Kasutatud 08.05.2020].

[7] Y. Pyo, H. Cho, R. Jung ja T. Lim, „ROS Robot Programming: From the basic concept to practical programming and robot application,“ ROBOTIS Co., Ltd., Seoul, 2017.

[8] „Apache ActiveMQ,” [Võrgumaterjal]. Available: https://activemq.apache.org/. [Kasutatud 08.05.2020].

[9] „Streaming Text Oriented Messaging Protocol,” [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Streaming_Text_Oriented_Messaging_Protocol/. [Kasutatud 08.05.2020].

[10] „Low Earth orbit,” [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Low_Earth_orbit. [Kasutatud 08.05.2020].

[11] „Two-line element set,” [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Two-line_element_set/. [Kasutatud 08.05.2020].

[12] S. Romanov, „Kuupsateliidi missioonijuhtimistarkvara arhitektuur,“ Tallinn, 2017.

[13] „Pyephem,” [Võrgumaterjal]. Available: https://rhodesmill.org/pyephem. [Kasutatud 09.05.2020].

[14] J.-C. Lio, PhD, „The Near-Earth Orbital Debris Problemand the Challenges for Environment Remediation,” [Võrgumaterjal]. Available: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20120012893.pdf. [Kasutatud 10.05.2020].

Lisa 1 – ActiveMQ teema „ros_out” teema kuulaja MCS’is

```
@JmsListener(destination = "ros_out", containerFactory = "queueListenerFactory")
public void receiveResponse(Object rawMessage) {
   try {
      ActiveMQBytesMessage message = (ActiveMQBytesMessage) rawMessage;
      byte[] byteData = new byte[(int) message.getBodyLength()];
      message.readBytes(byteData);
      message.reset();

      JSONObject object = new JSONObject(new String(byteData));
      System.out.println((String) object.get("request_id"));
      System.out.println((boolean) object.get("ok"));
      System.out.println((String) object.get("data"));
   } catch (JMSException e) {
      e.printStackTrace();
   }
}
```

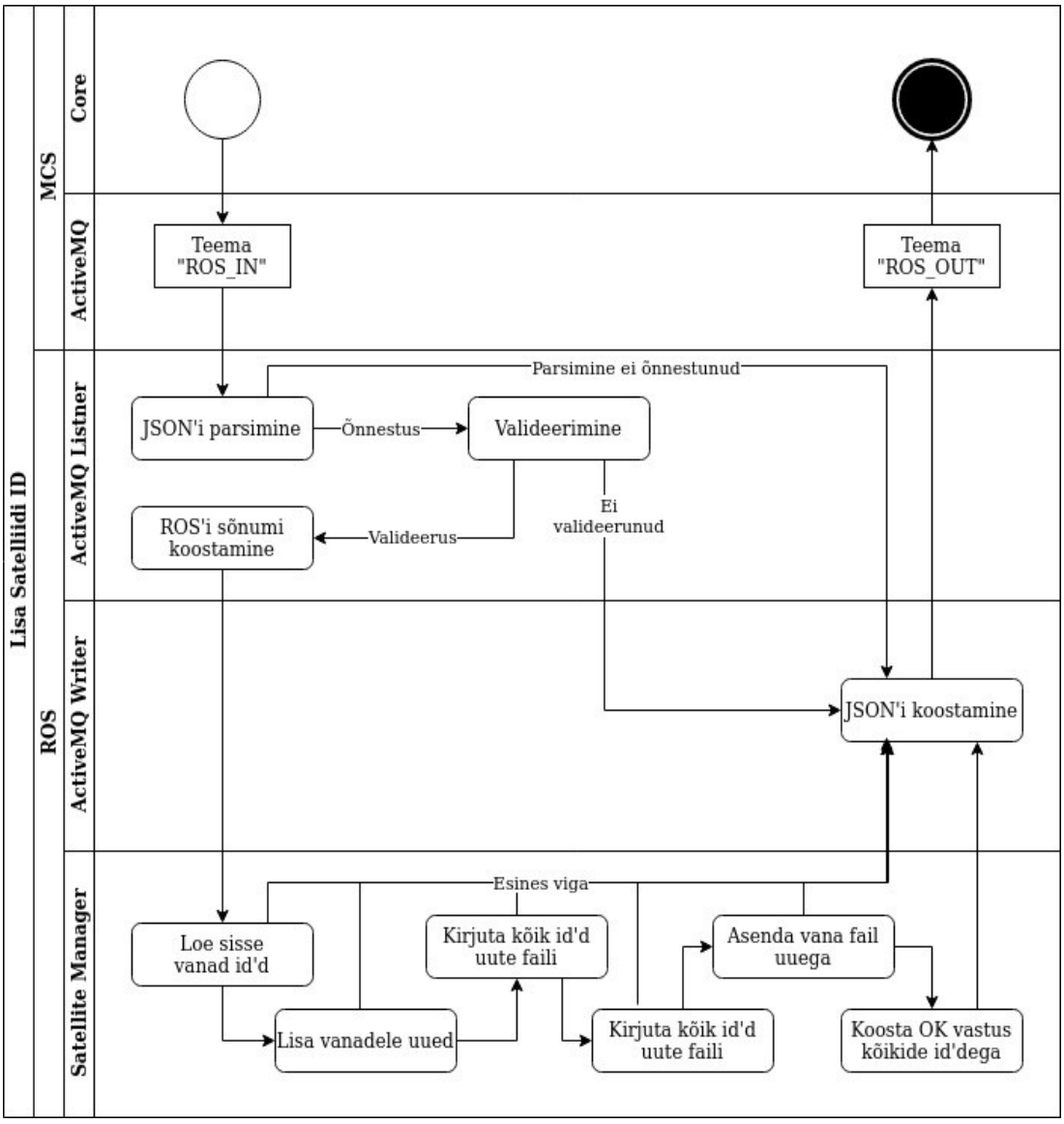
Joonis 12. See on programmikoodi lisamise näide.

Lisa 2 – Paraboolantenni juhitmissüsteemi veakoodid

Tabel 2. ROS'i veakoodid

Kood	**Sõlm**	**Kireldus**
-1	ActiveMQ listner	JSON'i parsmini ebõnnestus
-2	ActiveMQ listner	MCS'i sõnumist on päringu ID puudu
-3	Flyover tracker	Sõlme sisemine viga, täpsustav info vea kirjeduses
-4	Satelite manager	Sõlme sisemine viga, täpsustav info vea kirjeduses
-5	TLE getter	Sõlme sisemine viga, täpsustav info vea kirjeduses
-6	Antenna manager	Sõlme sisemine viga, täpsustav info vea kirjeduses
-7	Flyover tracker	Sõlme sisemine viga, täpsustav info vea kirjeduses

Lisa 3 – Jälgitava satelliidi lisamise kulg



Joonis 13. Jälgitava satelliidi lisamise kulg