TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Rasmus Tomsen 155199IAPB

JUHTIMISSÜSTEEMI LOOMINE TTÜ SATELLIIDI MAAJAAMA PARABOOLANTENNILE

bakalaureusetöö

Juhendaja: Evelin Halling
MSc

Tallinn 2018

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: Rasmus Tomsen

20.05.2018

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua TTU100 tudengisatelliidi projekti käigus valmivale maajaamas asuvale paraboolantennile juhtimissüsteem, mis suudaks jälgida satelliitide ülelende ning sellest sõltuvalt liigutada antenni õiges suunas. Lõplik juhtimissüsteem peaks suutma liigutada antenni täpsusega ±0.5°.

Töö tulemusena valmis ROSi ja Gazebo abil ehitatud simulatsioonimudel, mis liigub analoogselt reaalsele antennile. Samuti sai loodud programm, mis küsib parasjagu saadaval olevat värskeimat informatsiooni huvipakkuva satelliidi trajektoori kohta ning hakkab selle põhjal satelliidi ülelendu jälgima. Satelliidi vaatevälja jõudmisel teeb programmikood satelliidi asukoha ja antenni positsiooni järgi vastavad arvutused ja saadab antenni liigutamiseks vajalikud käsud mudelile. Kood on ehitatud nii, et täpsemate andmete selginemisel oleks sellele võimalik küllaltki lihtsalt juurde lisada sensoreid või muid huvipakkuvaid sisendeid.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 31 leheküljel, 8 peatükki, 7 joonist, 3 tabelit.

Abstract

Writing a steering system for the TTU satellite ground station parabolic antenna

The main aim of the thesis is to create a steering system for the K_u-band parabolic antenna that will be built on top of the Mektory building where the TTU100 student satellite program ground station will be located. The primary goal of the antenna will be to create a fast downlink connection between the TTU100 satellite and ground station to receive as much information as possible during the short times when the satellite is visible. Because of that the antenna must maintain an accuracy of ±0.5° during the whole overflight.

As a result of this thesis a simulation model was built using ROS and Gazebo. The model moves analogically to the real parabolic antenna. The antenna has two joints that allow it to move in every direction from the horizon to the horizon. The main reason the simulation model had to be built was that the real antenna was not ready during the writing of the thesis. In addition a program was written that asks for the currently most recent information about satellite that the ground station wants to communicate with and starts tracking it. When the satellite enters the field of view of the antenna, the program starts doing calculations based on the current coordinates of the satellite and the current position of the antenna to find the necessary data and orders to send to the antenna. The code also has to take into account the possible extra factors that could disturb the work in the future. The software is written while thinking ahead, to make it easy to add extra sensors or some other kind of functionality to the code when they will be needed. As the real antenna is not fully built yet, all the needed functionality can not be added yet either.

The thesis is in Estonian and contains 31 pages of text, 8 chapters, 7 figures, 3 tables.

Lühendite ja mõistete sõnastik

1U	*One Unit*, kuupsatelliitide standardsuurus
API	*Application Programming Interface*, rakendusliides
CAD	*Computer-aided design*, raalprojekteerimine
JSON	*JavaScript Object Notation*, laialdaselt kasutatud andmevahetusvorming
LEO	*Low Earth Orbit*, Maa-lähedane orbiit
PID kontroller	*Proportional-Integral-Derivative controller*, proportsionaal-integraal-diferentsiaalregulaator
ROS	*Robot Operating System*, robotite tarkvara loomiseks mõeldud raamistik
SDF	*Simulation Description Format*, simulatsiooni kirjeldamiseks loodud andmeformaat
TLE	*Two-Line Element*, kaherealine andmeformaat Maa orbiidil asuvate objektide kirjeldamiseks
URDF	*Unified Robot Description Format*, robotite kirjeldamiseks loodud andmeformaat
Xacro	*XML Macros*, lühema ja loetavama XMLi kirjutamiseks loodud formaat
XML	*Extensible Markup Language*, laiendatav märgistuskeel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Kui kosmose- ja satelliittehnoloogia oli veel eelmisel sajandil peamiselt vaid suurte ja rikaste ettevõtete pärusmaa, siis tänasel päeval saab oma käe ühel või teisel moel külge panna praktiliselt igaüks. Üha rohkem ülikoole töötavad selle nimel, et ka ise kosmoseteaduses kaasa lüüa ja enda ehitatud satelliit Maa orbiidile saata. Ka Tallinna Tehnikaülikoolil koostöös Mektoryga on taoline projekt parasjagu käsil ning juba 2019. aasta alguseks loodetakse üles saata kuupsatelliit nimega TTU100.

Tudengisatelliidi TTU100 puhul on tegu 1U nanosatelliidiga, mis tähendab, et satelliidi mõõtmed on vaid 10x10x10 cm. Satelliit hakkab tööle Maa-lähedasel orbiidil (LEO) ning projekti eesmärkideks on teostada Maa kaugseiret, testida uut Maa ja kosmose vahelist kõrgsageduslikku andmesidet ning arendada ja demonstreerida kasutatud tehnoloogiaid [1].

1.1 Probleem

Kuigi satelliit ise hakkab tiirlema Maa orbiidil on vaja sellega ka Maa pealt ühendust saada. Maajaama ja satelliidi omavaheliseks suhtluseks pannakse Mektory hoone katusele vähemalt kaks erinevat antenni, mille abil signaale satelliidile üles saata ning ka vastupidi satelliidilt vastu võtta. Nendeks antennideks on 435MHz töötav ultrakõrgsagedusala Yagi tüüpi antenn, mille jälgimise täpsus ei pea olema kuigi kõrge ning teine 10.5GHz töötav superkõrgsagedusala K_u-riba tüüpi paraboolantenn, mis peab suutma jälgida üle lendavat satelliiti väga suure täpsusega [1], [2].

Lisaks lihtsalt satelliidi jälgimisele, peab antenn pidevalt arvestama ka võimalike sise- ja väliskeskkonna poolt põhjustatud kõrvalnähtudega, mis jälgimist ühel või teisel moel takistavad. Sellisteks teguriteks võivad olla näiteks ilmastiku mõjud, eriti tuul ja sademed, antenni katusele panekul tekkinud ebatasasused ning mootorite ja hammasrataste konarused.

1.2 Eesmärgid

Antud lõputöö peamiseks eesmärgiks on luua juhtimissüsteem teisena mainitud K_u-riba paraboolantennile. Juhtimissüsteemi peamised ülesanded on järgmised:

- Juhtimissüsteem peab suutma reaalajas jälgida soovitud satelliidi trajektoori ning aru saama, millal satelliit antenni nägemisulatusse jõuab.
- Saades antennilt informatsiooni selle positsiooni kohta ning teades satelliidi hetke asukohta, peab süsteem leidma vajalikud parameetrid antenni korrektseks liigutamiseks.
- Antenni liigutamisel peab arvestama võimalike antenni sise- ja väliskeskkonnast tingitud kõrvalmõjudedega, mis võivad tööd häirida.
- Juhtimissüsteemi töö ajal peab olema võimalik vahetada parasjagu jälgitavat satelliiti.

Kokkuvõttes saab töö eesmärgi aga jagada mitmeks alamülesandeks, millega järk-järgult tegeleda tuli.

Esimeseks alameesmärgiks on üles seada simulatsioonikeskkond ja ehitada antennile vajalik simulatsioonimudel kasutades ROSi ja Gazebot.

Teiseks peab uurima erinevaid satelliitide jälgimise ja orbiitide ennustamise süsteeme ja programme ning leidma nende hulgast parim, mille abil vajalikke satelliite jälgima hakata. Seejärel tuleb kirjutada Pythonis kood, mis suudaks reaalajas leida alati satelliidi asukoha.

Kolmandaks peab kirjutama programmi, mis suudab satelliite jälgides liigutada piisavalt täpselt esialgu loodud simulatsiooni ning hiljem ka reaalset antenni.

Neljandaks tuleb ühendada reaalne antenn kirjutatud tarkvaraga ning teha vastavalt muutmisi, et süsteem hakkaks tööle ka päris maailmas.

2 Kasutatud tehnoloogiad

Antud peatükis on lühidalt kirjeldatud projektis kasutatud tehnoloogiaid ja tarkvara, et saada parem ülevaade ülejäänud tehtud tööst.

2.1 ROS

ROS ehk *Robot Operating System* of Linuxil töötav robotite tarkvara kirjutamiseks mõeldud raamistik. Tegu on väga laiaulatusliku ja paindliku raamistikuga, millele on saadaval suures koguses tööriistu ja teeke, mis kõik võimaldavad teha mõne osa roboti loomisest lihtsamaks. Tegu on avatud lähtekoodiga tarkvaraga, mis tähendab, et kõik võivad koodi vabalt muuta ja luua ise vajalikke teeke ning komponente juurde [3].

ROS võimaldab luua vahelüli erinevate suhtlevate osade vahele, mille läbi vastavad sõnumid õige osapooleni jõuavad. Siiski ei saada ROS sõnumeid otseselt ise ühelegi sõlmele, vaid pakub võimalust suhtlevatel pooltel ise keskse vahelüliga ühendust võtta ning sinna uusi sõnumeid saata (*publish*) või vastupidiselt salvestatud sõnumeid kuulama jääda (*subscribe*). Tegu on asünkroonse ning anonüümse süsteemiga, mis loob võimaluse kõikidel sõlmedel mistahes sõnumeid saata ja kuulata ning seeläbi aitab kaasa koodi kapseldamisele ja taaskasutamisele [3], [4].

2.2 Gazebo

Gazebo on robotite simuleerimiseks loodud 3D simulaator, mis pakub kasutajatele võimalikult reaalset päris maailma sarnast simulatsiooni. Gazebo abil on võimalik disainida ja luua roboteid, testida erinevaid algoritme ja õpetada tehisintellekte. Robotite mudelitele on võimalik külge panna erinevaid sensoreid, mis aitavad töö tegemist realistlikumaks muuta. Muu hulgas kasutab Gazebo füüsikamootorit, mille abil on võimalik luua realistlikke keskkondi ja olukordi. Gazebot on lihtne ühendada ka ROSi süsteemiga. Sarnaselt ROSile on ka Gazebo avatud lähtekoodiga [3].

2.3 RViz

RViz ehk *ROS visualisation* on ROSi tööriist, mille abil on võimalik visualiseerida kolmemõõtmelises keskkonnas URDFi abil kirjeldatud roboteid ning saada informatsiooni roboti olekute ja sensorite kohta. See aitab kergelt välja selgitada, mida loodud robot igal ajahetkel näeb ning millist informatsiooni sensoritest saab. Antud tarkvara sai kasutatud esialgse mudeli loomiseks, mida hiljem Gazebo abil simuleerida sai [4].

2.4 MoveIt!

MoveIt! on avatud lähtekoodiga tarkvara, mis ühildub väga hästi ROSi ja Gazeboga ning võimaldab lihtsustada erinevate robotkäte ja manipulaatorite liikumisloogika arendamist. Kuigi antud tarkvara sai proovitud ning algselt oli seda plaanis ka reaalselt siin projektis rakendada, siis töö käigus otsest vajadust selle kasutamiseks siiski ei tekkinud. Põhjuseks on antenni küllaltki lihtne üleehitus ning ainult kahe vabadusastme kasutamine [3].

3 Probleemi täpsem püstitus

Nagu juba eelnevalt mainitud on antud bakalaureusetöö eesmärgiks välja töötada süsteem, mis suudab reaalajas jälgida vajaliku satelliidi ülelende ning sellest tingitult juhtida tudengisatelliidi projekti käigus valmivat K_u-band tüüpi paraboolantenni. Kuigi algselt on antenni peamiseks eesmärgiks saada kiire side maajaama ja TTU100 satelliidi vahele, siis tulevikus hakatakse antud antenni ka teistele soovijatele välja rentima. See tähendab, et töötamise käigus peab olema võimalik öelda programmile, millist satelliiti see parasjagu jälgima peab ning programm peab suutma seejärel kõik arvutused uute andmete jaoks kohaldada. Antenn hakkab paiknema Mektory maja katusel.

3.1 Paraboolantenni kirjeldus

Töö lahenduse paremaks mõistmiseks peaks enne teadma ka kavandatava antenni andmeid. Paraboolantenni näol on tegu antud projekti tähenduses oma mõõtmetelt küllaltki suure antenniga, mille taldriku läbimõõt ulatub 5 meetrini.

K_u-riba paraboolantenni peamiseks ülesandeks on kasutada satelliidi ülelennu aega maksimaalselt ära ja luua kiire allalüli ühendus satelliidi ja maajaama vahele. Kuna satelliit on maajaamaga sideühenduses vaid umbes pool tundi ööpäevas, on väga oluline, et selle aja jooksul suudetaks satelliidi käest võimalikult palju infot kätte saada. Kuna TTU100 satelliidi üheks peamiseks ülesandeks on teha pilte Maast, siis maajaamale saadetavad andmed on küllalt mahukad ning sidekiirus peab olema suur [1].

Selleks, et saavutada suurt sidekiirust töötab paraboolantenn 10.5GHz sagedusel. Kuigi selline kõrge sagedus aitab luua suurt kiirust, siis see omakorda tingib väga väikese lainepikkuse ja amplituudi, mis tähendab, et antenni ja satelliidi vaheline eksimisruum peab olema minimaalne. Täpsemalt ei tohi K_u-riba antenn eksida esialgu kogu jälgitava satelliidi ülelennul rohkem kui ±0.5° [2].

Et antenni oleks võimalik liigutada on sellel kaks liigendit, mis võimaldavad suunata antenni igas suunas horisondist horisondini. Üheks liigendiks on antenni maa küljes hoidev jalg, mida saab pöörata 360 kraadi ümber oma telje. Kuigi tehniliselt oleks

võimalik seda lõpmatult ka ainult ühes suunas pöörata, siis sellist käitumist takistavad jala küljes olevad kaablid, mis suure pööramise tagajärjel tekkivat pinget üle ei elaks. Teiseks liigendiks on antenni jalga ja taldrikut ühendav lüli, mida on võimalik liigutada maa suhtes 180 kraadi. Kuigi taldrikul on ees füüsiline takistus, mis ei lase sellel rohkem liikuda, siis ei tohiks algoritm sellest hoolimata üritada taldrikut lõpmatult ühes suunas liigutada ning süsteemi luues nende piiridega arvestama. Antenni täpsemad mõõtmed on näidatud Joonis 1.

Joonis 1. Paraboolantenni mõõtmed.

Antennile pannakse juurde hulk sensoreid, mis hakkavad töötamise käigus andma antenni oleku ja positsiooni ning välisjõudude kohta tagasisidet juhtprogrammile. Kuigi lõputöö

kirjutamise hetkel ei ole veel täpselt teada, mida ning kui täpselt antenni jaoks mõõtma peab, siis arutuste käigus on mõned kindlad osad siiski teada. Antenni liigendite jaoks pannakse antennile külge vähemalt kaks enkoodrit, mis annavad infot liigendi hetke pöördenurga kohta. Seda infot on vaja, et teada täpselt kui palju ja mis suunas peaks antenn edasi liikuma, et soovitud olekusse jõuda. Lisaks peab ühel hetkel arvestama kindlalt ka tuulekiirusega, mistõttu pannakse katusele suure tõenäosusega püsti ka ilmajaam. Kuna antenni puhul on tegu suure tuuletakistusega, siis peab arvestama, et igasuguse ilmaga antenn töötada ei tohiks. Mingi kindla tuulekiiruse saavutamisel peaks antenn satelliitide jälgimise katkestama ja liikuma positsiooni, kus tuuletakistus on väikseim, et minimaliseerida tuulest tingitud kahjude tekkimist.

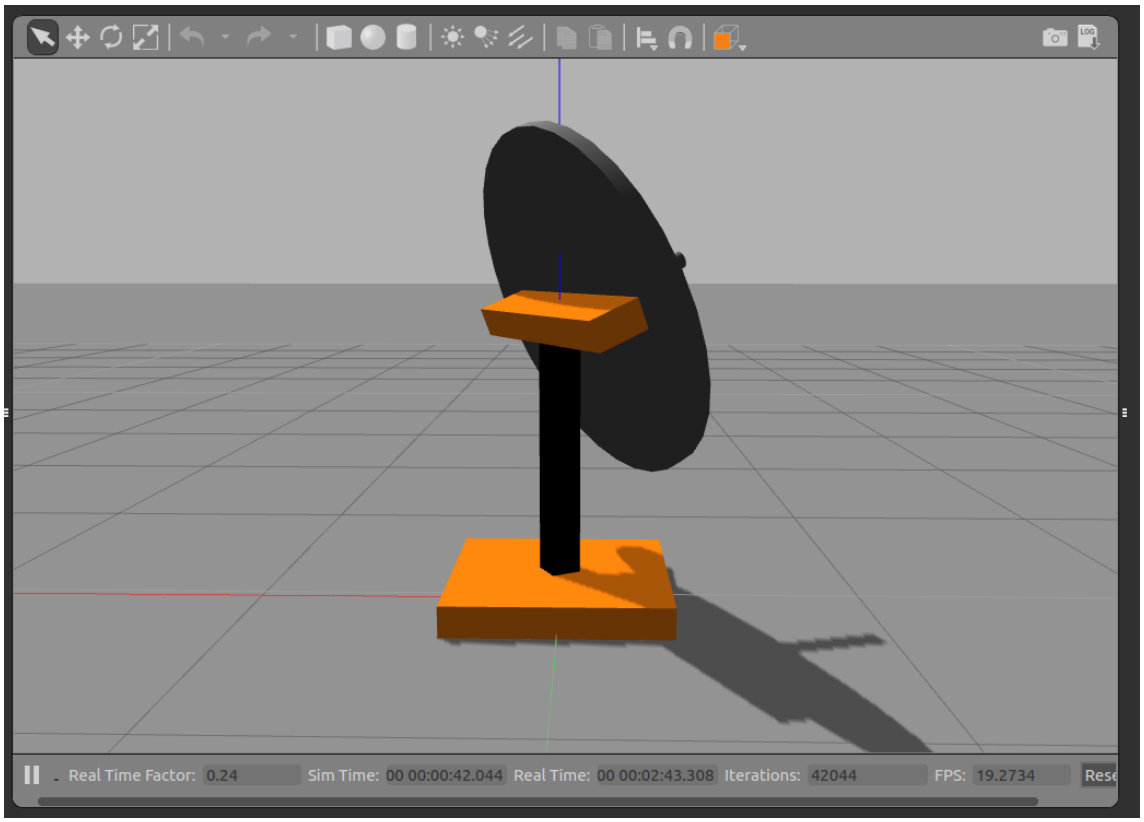
4 Simulatsioonikeskkonna ehitamine

Kuna lõputööd alustades ei olnud paraboolantenn veel kaugeltki valmis, tuli töö esimese osana leida mingi hea võimalus antenni simuleerimiseks. Peale mõneaegset uurimist langes otsus küllaltki kindlalt ROSi ja Gazebo kasuks. Gazebo eeliseks oli võimalikult elulähedaste simulatsioonide tegemise võimaldamine, mis suudavad muu hulgas arvestada ka erinevate loodusjõududega. ROSi arhitektuur aga võimaldab kergelt luua vahekihi reaalse koodiloogika ja simulaatori või füüsilise seadme vahele [3].

4.1 Antenni mudeli loomine

Tööd alustades puudus autoril igasugune varasem kokkupuude nii ROSi kui Gazeboga. Seetõttu pidi enne reaalset antenni simulatsiooni ja juhtimisloogika kirjutamist õppima antud tarkvara üldse kasutama. Õppimise käigus sai ehitatud mitu erinevat roboti mudelit ning sai looma hakatud ka antenni ennast. Kuna mudeli tegemisel ei olnud veel teada reaalse antenni täpseid mõõtmeid ning kuna esialgse liigutamisloogika ehitamiseks ei olnud oluline, et antenn oleks väga täpne, sai loodud küllaltki lihtsa tegumoega antenni mudel. Lihtsa mudeli loomisel olid ka omad eelised. Näiteks nõudis sellise mudeli simuleerimine vähem arvutusvõimsust, mis osutus tööarvutit arvestades väga oluliseks. Nimelt sai töö tehtud veidi vanema ning vähem võimsama arvuti peal, kuna antud hetkel ei olnud autoril paremat Linuxi operatsioonisüsteemiga sülearvutit saadaval. Joonis 2 on kuvatud Gazebos loodud antenni mudel, kus lisaks on ka näha, et ainuüksi nii kerge mudeli simulatsioon töötas reaalajafaktoriga 0.24, ehk reaalajast umbes neli korda aeglasemalt. Lisaks on Joonis 3 kuvatud võrdluseks antenni suhteliselt täpne SolidWorksis[1] loodud CAD mudel.

[1] https://www.solidworks.com/



Joonis 2. Gazebos loodud paraboolantenni simulatsioonimudel

Joonis 3. Paraboolantenni CAD mudel

Mudeli ehitamisel osutusid kõige tähtsamateks osadeks selle liikuvad osad ja nende vahelised liigendid. Kuna oli vähemalt teada, mis moodi antenn tulevikus täpselt liikuma hakkab, siis nende osade loomisele sai pööratud enim tähelepanu. Siiski polnud teada osade vaheliste lülide detailseid andmeid ja seetõttu sai külge pandud katsetuste käigus mõistlikuna tundunud PID kontrollerid, millele sai sisendiks antud täpsed nurgad, kuhu antenn ennast pöörama pidi. Loomulikult ei piisa nii vähestest andmetest hiljem reaalse antenni juhtimiseks, kuid algse simulatsiooni jaoks palju täpsemaid detaile vaja ei läinud [5].

Mudeli loomiseks kasutati XML failiformaadis loodud ROSi standardit, URDFi, ehk *Universal Robotic Description Format*. URDF võimaldab kerge süntaksi abiga kirjeldada kogu soovitava roboti välimust. Kuna URDF on aga mõeldud peamiselt roboti ning selle erinevate osade kujunduse ja mõõtmete kirjeldamiseks, siis puudub sellel suur osa funktsionaalsusest, millega robotite loomisel kindlalt arvestada tuleb. Seetõttu ei saa ka URDFi otse Gazebo abil simuleerida ning esialgu sai loodud mudelit katsetatud Rviz visualiseerimise tööriistaga. Õnneks on aga URDFi võimalik töö käigus ümber teisendada Gazebos kasutavale SDF (*Simulation Description Format*) failiformaadi kujule, mida sai ka antud projektis tehtud [5].

4.2 Mudeli ja simulatsioonikeskkonna kokku panemine

Peale esialgse mudeli ehitamist tuli hakata seda Gazebo abil simuleerima. Kuna algse simulatsiooni puhul ei ole kuigi oluline, et antenni ümbritsev keskkond oleks võimalikult sarnane reaalse maailmaga, siis sai antenn lisatud täiesti tühja Gazebo poolt simuleeritud ruumi. Millegipärast ei nõustunud Gazebo aga esimesena URDFis loodud antenni mudelit korrektselt simuleerima. Peale pikaaegset vigade otsimist ja parandamist otsustas autor ehitada mudeli nullist uuesti, kuid seekord kasutada antenni kirjeldamiseks Xacro (*XML Macros*) failiformaati. Xacro puhul on tegu samuti XML põhjal loodud keelega, mis võimaldab lihtsustada XML faile ning muuta neid lühemaks ja loetavamaks. Eriti kasulikuks muutub antud formaat suurte robotite kirjelduste puhul [6]. Kuna töö käigus loodud antenni mudel oli oma ülesehituselt üsna kerge, siis ei pidanud Xacro faili ümberkirjutamiseks seal kuigi palju muutma. Peale Xacro faili valmimist õnnestus lõpuks mudel ka Gazebos tööle saada [3].

Kuna aga üksi mudeli simuleerimisest, mida kuidagi väliselt liigutada ei saa, pole antud töö jaoks kasu, siis tuli loodud liigenditele külge panna kontrollerid. Kontrollerid on oma loomu poolest lihtsad ROSi teemad ehk *topic*'ud[1], mis kuulavad neile saadetud sõnumeid ning õiges formaadis käskluse korral ütlevad simulatsioonimudelile, mis asendisse see liikuma peaks [7].

4.3 Mudeli valideerimine

Kuna simulatsioonimudel on oma ülesehituselt küllaltki lihtne, siis ei tekitanud ka selle testimine suuri probleeme. Töö kontrolliks sai saadetud antenni positsiooniga seotud teemadele sõnumeid uute koordinaatidega, mille peale neid kuulanud kontrollerid antenni liigutama hakkasid. Selliselt sai kontrollitud antenni liikumise piire, et mudel ei üritaks vastava sõnumi korral hakata näiteks taldrikut rohkem kui 180 kraadi pöörama. Samuti sai katsetuste käigus kontrollitud, et mudel töötaks piisava kiirusega, et jõuda vajaliku aja jooksul sellele saadetud uutele koordinaatidele. Peale antenni pööramist sai selle uut positsiooni võrrelda ka algselt soovitud lõppkoordinaatidega, et kindlustada liigendite korrektne töö.

[1] http://wiki.ros.org/Topics

5 Satelliitide trajektooride jälgimist võimaldavate vahendite uurimine

Peale esialgsete simulatsioonimudelite ehitamist sai alustada üldise juhtimissüsteemi koostamisega. Selle tegemise võib laias laastus jagada kaheks suuremaks osaks. Üheks on tarkvara kirjutamine, mis suudab etteantud juhiste järgi leida üles orbiidil asuva satelliidi ja hakata seda ajas jälgima. Teine pool tööst aga nõuab antennilt saadud ning sateliitide jälgimise teel leitud andmete abil vajalike arvutuste tegemist ning antenni juhtimisloogika kirjutamist. Kuna teine pool nõuab esimese olemasolu ning füüsiline antenn töö tegemise ajal valmis ei olnud, siis sai alustatud just jälgimistarkvara loomisega.

5.1 Tarkvarade esialgne uurimine

Maa orbiidil tiirlevate tehiskaaslaste asukohta ja muid andmeid pidevalt jälgida on väga keeruline töö. Selle asemel kasutatakse enamasti andmefomaati TLE ehk *Two-line element*. TLE on eelmisel sajandil välja töötatud formaat, mis on tänasel päeval kujunenud *de facto* standardiks. See koosneb oma olemuselt kahest 69 tähemärgi pikkusest kindlal kujul kirjutatud andmereast ning selle abil on võimalik arvutada välja Maa orbiidil asuvate objektide mineviku ja tuleviku trajektoorid. Kuigi TLE abil on üldjuhul võimalik küllaltki täpselt ennustada vajaliku satelliidi trajektoori, siis sellegipoolest tekivad pikaaegsel ennustamisel sisse vead, mis võivad töötlemisel anda lõpuks mittesoovitud tulemusi. Nende vigade vältimiseks on soovitatav kasutada alati kõige värskemat infot satelliitide kohta, mida tänasel päeval annab välja NORAD[1] [8].

Ka käesolevas projektis sai satelliitide jälgimiseks kasutatud TLEd. Kuigi ainuüksi TLE abiga oleks võimalik ka ise välja arvutada soovitud satelliidi positsiooni ning trajektoori, leidub palju juba varasemalt loodud programme ja teeke, mis selle keerulise töö ära teevad. Et leida valikute seast parim, mis antud tööga kõige rohkem sobiks, otsis autor

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

erinevaid tööriistu, mis võimaldaksid küsitud satelliitide trajektoore piisavalt täpselt jälgida ja tegi nendega vajalikke mõõtmisi. Kuna programmi tähtsaimaks eesmärgiks on hakata jälgima just TTU100 satelliidi liikumist, siis sai tarkvarade testimisel muu hulgas peamiselt arvestatud sellega, et tarkvara oleks võimeline leidma TTU100 satelliidile hetkel kõige lähedasemat orbiidil asuvat satelliiti, milleks on 2013. aastal üles saadetud ESTCube-1[1].

5.1.1 Orbitron

Orbitron[2] on kahtlemata antud valikutest pikka aega olnud üks enim kasutatud variante ning tegu on väga usaldusväärse tarkvaraga. See on Windowsi operatsioonisüsteemi jaoks juba rohkem kui 10 aastat tagasi loodud programm, mis lisaks satelliitide trajektooride ennustamisele näitab ka visuaalset pilti nende hetke asukohaga. Kuna autor ei leidnud viisi, kuidas ennustamisel saadud andmeid peale käsitsi eksportimise programmist kätte saada ning kuna tegu on ainult Windowsile kirjutatud programmiga, siis selle kasuks mõistagi valik ei langenud. Küll aga sai Orbitron võetud tema usaldusväärsuse tõttu peamiseks mõõdupuuks, mille vastu teisi tarkvarasid testida.

5.1.2 Pyorbital

Pyorbital[3] on Pythoni keele jaoks loodud moodul, mis võimaldab kerge vaevaga enda koodis vajalikke arvutusi läbi viia. Üks positiivne omadus antud mooduli puhul oli ise TLE pärimine internetist, ilma et peaks kasutama mingit muud laiendust. Kuigi tarkvara töötas hästi, siis kahjuks ei suutnud autor saada kätte ESTCube-1 andmeid ning üldjoontes tundus, et kuupsatelliite sellega jälgida ei saa.

5.1.3 Pyephem

Nagu eelnevalt mainitud Pyorbital, on ka Pyephemi[4] puhul tegu Pythoni mooduliga. Siin aga tuleb kahjuks enne trajektoori ennustamist anda TLE moodulile ise ette. See aga töö tegemisel suuri probleeme ei põhjusta, kuna TLE andmeid ei ole keeruline ka mujalt

[1] https://www.estcube.eu/estcube-1-0

[2] http://www.stoff.pl/

[3] http://pyorbital.readthedocs.io/en/latest/

[4] http://rhodesmill.org/pyephem/

pärida. Pyephem töötab ka kõikide Maa tehiskaaslastega, ehk tegu on väga hea võimaliku variandiga.

5.1.4 Gpredict

Gpredict[1] on Linuxi operatsioonisüsteemile loodud rakendus, mille abil on võimalik satelliite jälgida ka visuaalse liidese teel. Sarnaneb Orbitronile, kuid Gpredictil on rohkem funktsionaalsust. Satelliitide jälgimisel on ligipääs rohkematele võimalikele andmetele, kui eelnevalt mainitud tarkvarade puhul. Üldiselt on tegu väga hea tööriistaga, eriti kui eesmärgiks on kuvada tehiskaaslasi ka visuaalselt kaardi peal.

5.1.5 Satellites.calum.org

Satellites.calum.org[2] on API (*Application programming interface*), mille abil on võimalik saada kätte küsitud satelliidi järgmise ülelennu algus- ja lõpuajad ning koordinaadid JSON (*JavaScript Object Notation*) formaadis. Kuna aga nii vähesest infost jälgimiseks ei piisa, siis antud lehekülg siin projektis kasutusele ei tule.

5.1.6 Orbit-Predictor

Orbit-Predictori[3] puhul on samuti tegu Pythoni jaoks ehitatud mooduliga, kuid millegipärast ei õnnestunud autoril seda oma arvutis üldse tööle saada.

5.1.7 Predict

Predict[4] on Linuxi Terminalis töötav rakendus. Algselt tööle pannes on nimekirja juba eelnevalt salvestatud hulk satelliite, mida on võimalik läbi rakenduse ka välja vahetada. Satelliidi lisamisel või muutmisel aga tahetakse saada peale TLE veel käsitsi suurel hulgal erinevaid andmeid, mille peaks ka TLEd lugedes kätte saama. Peale nimekirja ESTCube-1 andmete lisamist ja ennustaja tööle panemist jooksis aga programm kokku ning andis tagasisides mitmeid veateateid. Proovides ennustada aga juba eelnevalt

[1] http://gpredict.oz9aec.net/

[2] https://satellites.calum.org/

[3] https://pypi.org/project/orbit-predictor/

[4] http://www.qsl.net/kd2bd/predict.html

salvestatud satelliitide trajektoore, sai vastuseks andmed, mis erinesid mujalt pärit vastustest märkimisväärselt. Seetõttu sai otsustatud, et seda projektis ei kasuta.

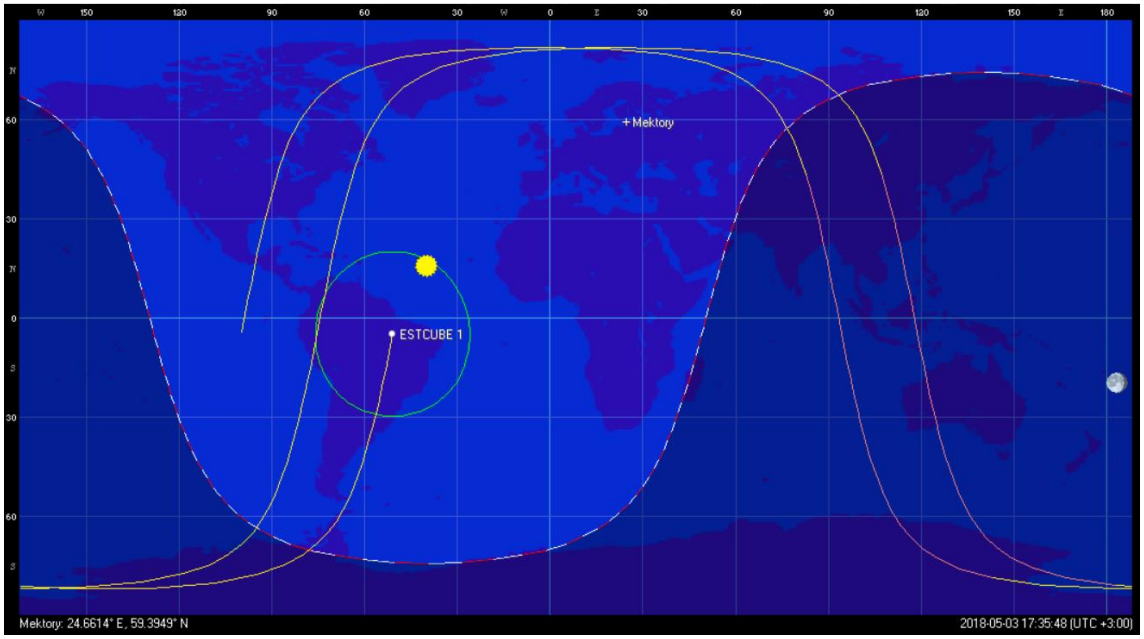
5.1.8 Xephem

Xephem[1] on Linuxi ja Mac OS-i jaoks loodud visuaalse liidesega programm, mille abil on võimalik jälgida erinevaid taevakehi, teiste hulgas ka satelliite. Kuna aga varasemalt mainitud Pyephem on ehitatud Xephemi põhjal, siis seda pikemalt uurima ei pidanud.

5.2 Mõõtmiste läbiviimine

Uurimisest selgus, et võimalikke variante, millega tehiskaaslaste liikumist jälgida on mitmeid. Leitud kaheksast variandist sai viis juba enne mõõtmisi välja jätta, kuna kõigil oli üks või teine puudujääk, mille tõttu nad projektis kasutamiseks ei sobinud. Allesjäänud Orbitroni, Pyephemi ja Gpredictiga viisin läbi mõõtmised, et nende täpsust omavahel võrrelda.

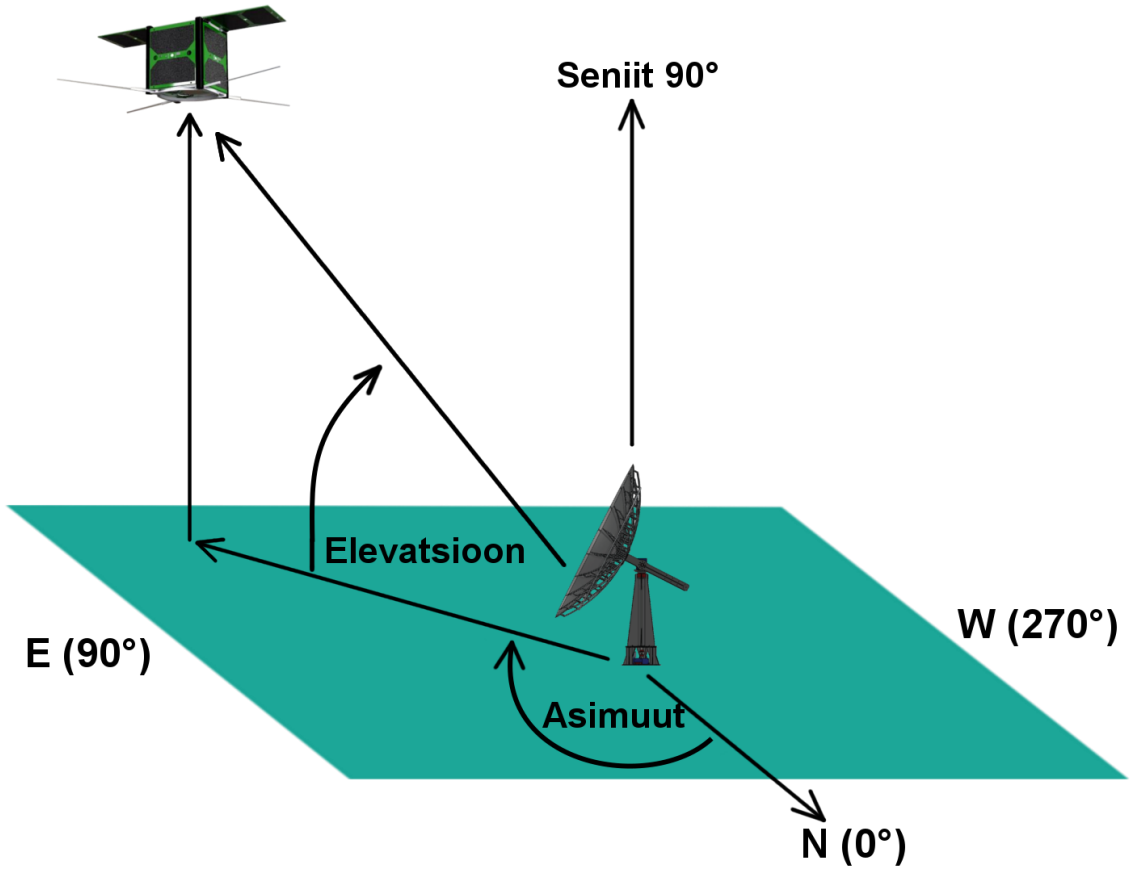
Parema ülevaate saamiseks, mis kujul satelliitide jälgimine toimub on Joonis 4 kujutatud Orbitroni visuaalse liidesega ESTCube-1 asukoht, selle edasine trajektoor ning piirkond, kus satelliit parasjagu nägemisulatuses on.



Joonis 4. ESTCube-1 Orbitronis

[1] http://www.clearskyinstitute.com/xephem/

Kuna satelliitide vaatlemine sõltub jälgimise asukohast Maal, siis tavalistest geograafilistest koordinaatidest siinkohal ei piisa. Jälgimiseks on kasutusel horisondiline koordinaatsüsteem, mis koosneb kahest koordinaadist: asimuudist ning kõrgusest ehk elevatsioonist. Asimuudiks nimetatakse nurka põhjasuuna ja vaadeldava objekti vahel ning elevatsiooniks nurka horisondi ja vaadeldava objekti vahel [9]. Joonis 5 on kujutatud horisondiline koordinaatsüsteem antud töö mõistes.



Joonis 5. Horisondiline koordinaatsüsteem

Selline koordinaatsüsteem on rangelt seotud vaatleja hetke asukohaga Maa peal ning seetõttu kasutatakse seda sageli just taevakehade leidmiseks [9]. Seetõttu osutus satelliitide jälgimise ning mõõtmiste puhul eriti oluliseks, et programmilt oleks võimalik kerge vaevaga kätte saada jälgitava satelliidi ülelendude ajad, asimuudid ja elevatsioonid soovitud ajatäpsusega.

Mõõtmiseks sai iga valitud tarkvara puhul ennustatud ESTCube-1 kümmet järgmist ülelendu koordinaatidel 59° 23' 41.532" N, 24° 39' 41.0364" E kõrgusel 0 meetrit merepinnast. Ülelendude puhul arvestasin kõiki kordi, kui satelliit jõudis horisondi tagant

välja, olenemata selle maksimaalsest elevatsioonist. Iga ülelennu puhul mõõtsin ülelennu alguse aega (*rise_t*); alguse asimuuti (*rise_az*); aega, millal elevatsioon oli kõrgeim (*max_t*); asimuuti, millal elevatsioon oli kõrgeim (*max_az*); kõrgeimat elevatsiooni (*max_el*); ülelennu lõpu aega (*set_t*) ning lõpu asimuuti (*set_az*). Mõõtmised said läbi viidud 28. märtsil 2018. aastal kell 18:00. Kuna ühegi mõõtmise juures ülelennu kuupäev ei erinenud, siis ruumi kokkuhoiu huvides sai aegade lahtritesse alles jäetud ainult kellaaeg. Saadud mõõtmiste tulemused on näha Tabel 1, Tabel 2 ja Tabel 3.

Tabel 1. Orbitroniga saadud mõõtmiste tulemused

Ülelennu number		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Orbitron	**rise_time**	18:07:55	19:41:53	21:15:11	22:49:52	00:26:46	02:06:46	10:33:15	12:09:05	13:45:55	15:22:56
	rise_az, °	23.6	53.0	95.4	139.9	185.9	239.4	34.4	20.4	15.3	13.5
	max_time	18:11:19	19:45:48	21:20:44	22:56:33	00:33:28	02:11:37	10:37:10	12:15:39	13:52:50	15:28:54
	max_az, °	353.5	17.7	40.9	62.7	263.7	285.1	69.5	91.7	292.0	313.7
	max_el, °	3.1	4.5	13.0	44.0	37.8	7.3	3.9	26.3	63.9	17.1
	set_time	18:14:43	19:49:43	21:26:20	23:03:19	00:40:15	02:16:30	10:41:05	12:22:10	13:59:42	15:34:51
	set_az, °	323.4	342.5	346.5	345.5	341.2	330.8	104.8	162.6	209.3	253.8

Tabel 2. Pyephemiga saadud mõõtmiste tulemused

Ülelennu number		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Pyephem	**rise_time**	18:07:54	19:41:52	21:15:10	22:49:51	00:26:45	02:06:46	10:33:14	12:09:04	13:45:54	15:22:55
	rise_az, °	23.61	52.987	95.448	139.935	185.839	239.382	34.373	20.401	15.269	13.462
	max_time	18:11:18	19:45:47	21:20:44	22:56:33	00:33:27	02:11:36	10:37:10	12:15:38	13:52:49	15:28:53
	max_az, °	353.518	17.711	40.815	62.591	263.525	284.924	69.679	91.498	293.277	313.678
	max_el, °	3.115	4.46	13.041	43.992	37.821	7.274	3.903	26.275	63.851	17.101
	set_time	18:14:43	19:49:43	21:26:19	23:03:19	00:40:14	02:16:30	10:41:04	12:22:09	13:59:41	15:34:50
	set_az, °	323.403	342.465	346.488	345.437	341.225	330.793	104.689	162.591	209.275	253.774

Tabel 3. Gpredictiga saadud mõõtmiste tulemused

Ülelennu number		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Gpredict	**rise_time**	18:08:13	19:42:09	21:15:21	22:50:00	00:26:54	02:06:59	10:33:33	12:09:13	13:46:03	15:23:06
	rise_az, °	21.17	50.98	94.47	139.67	186.23	240.97	36.70	20.98	15.16	12.69
	max_time	18:11:18	19:45:47	21:20:44	22:56:34	00:33:29	02:11:37	10:37:09	12:15:36	13:52:47	15:28:53
	max_az, °	353.52	17.63	40.64	61.1	264.92	285.31	69.5	91	294	313.78
	max_el, °	2.89	4.28	12.97	43.96	37.79	7.16	3.71	26.24	63.83	17.05
	set_time	18:14:23	19:49:26	21:26:07	23:03:09	00:40:04	02:16:15	10:40:45	12:21:59	13:59:32	15:34:39
	set_az, °	325.88	344.43	347.44	345.69	340.83	329.2	102.41	162.02	209.39	254.54

Nagu mõõtmistest näha, siis ei erinenud ühegi tarkvara poolt saadud tulemused üksteisest kuigi palju. Eriti väike oli vahe Orbitroniga ja Pyephemiga saadud väärtuste vahel. Nende asimuudi ja elevatsiooni maksimaalsed erinevused jäid vaid alla 0.1° ning ka siis oli vigade peamiseks põhjustajaks Orbitroni väiksem täpsus. Gpredictil tulid mõnel pool veidi suuremad vahed sisse, kuid üldjoontes ei erinenud ka selle tulemused väga. Huvitaval kombel erinesid Gpredicti andmed eriti satelliidi tõusu ja languse ajal. Tundub, et Gpredict alustas mõõtmisi veidi muul ajal, kui teised tarkvarad, kuigi üle kontrollides ei erinenud selle seaded teistest. Seetõttu sai otsustatud Pyephemi kasutamise kasuks, kuna peale täpsete tulemuste on antud moodulit ka käesolevas projektis väga mugav kasutada.

6 Pythonis juhtimistarkvara arendamine

Kui vajalik tehiskaaslaste ennustamistarkvara sai välja valitud, sai hakata kirjutama satelliitide jälgimis- ning antenni juhtimisloogikat. Selle loomiseks on kasutusel mitu Pythonis kirjutatud ROSi *node*'i ehk sõlme, mis töötavad pidevalt paralleelselt ning täidavad erinevaid sateliidi jälgimiseks ning antenni juhtimiseks vajalikke ülesandeid. Sõlmedeks nimetatakse ROSis töötavaid protsesse, mis suudavad üksteisega suhelda, kasutades teisi ROSi poolt pakutavaid formaate: *topic'*uid ehk teemasid, *service'*d[1] ehk teenuseid ning parameetri serverit[2]. Sõlmed on üldiselt loodud ühe kindla ülesande täitmiseks, mis peaksid suurelt osalt olema võimelised töötama teistest sõltumatult. Lisaks aitab sõlmede kasutus muuta koodi struktuuri lihtsamaks ja arusaadavamaks [10].

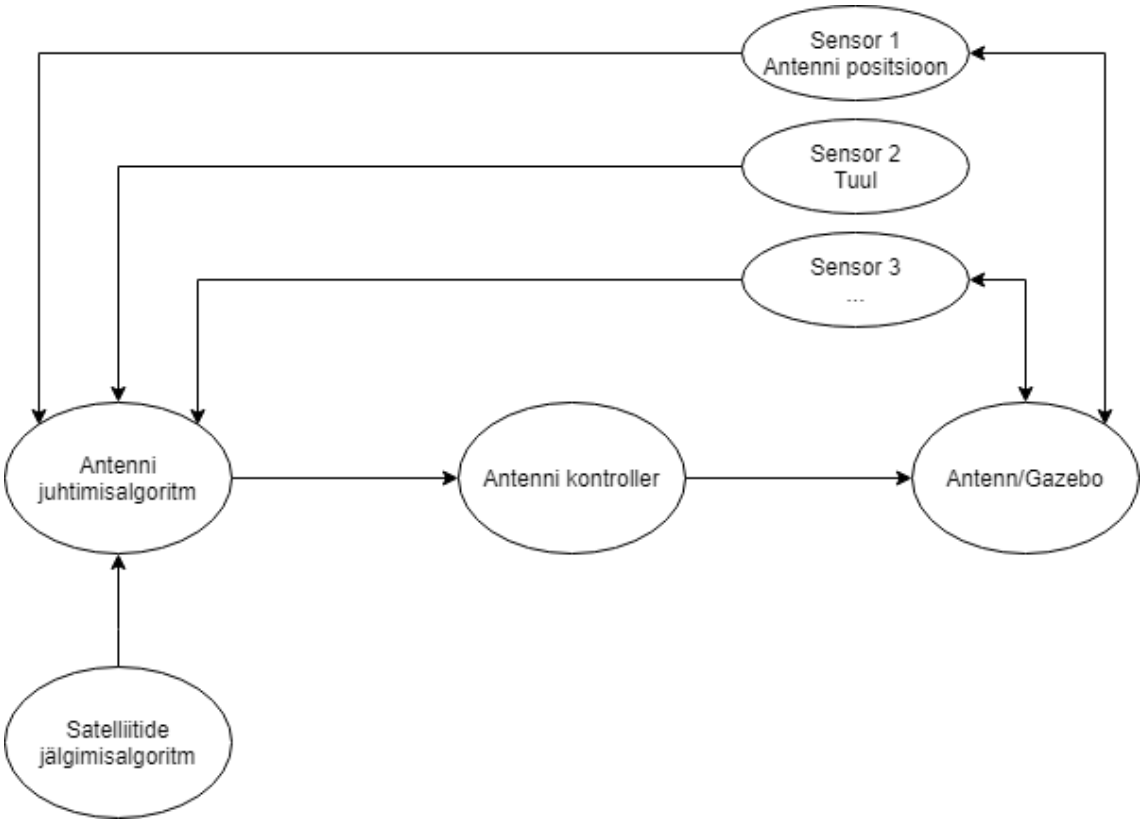
Loodud sõlmed moodustavad ühe tervikliku ROS *package*'i ehk paketi. Paketid on grupeeritud kogumid muudest ROSi osadest, mis koos töötades täidavad mingit kindlat funktsionaalsust. Pakettide kasutamise peamiseks eesmärgiks on luua kergelt kasutatavaid komponente, mida oleks võimalik enda projektis hiljem kasutada. Sellisteks pakettideks võivad olla näiteks erinevate sensorite loogika või mudelite kirjeldus [11]. Ka käesolevas töös on kasutatud struktuuri lihtsustamiseks mitut erinevat paketti, kuid antud punktis kirjeldan just juhtimispoolega tegeleva paketi tööd.

6.1 Paketi esialgne struktuur

Töö lihtsustamiseks ning soovitud lõpptulemuse paremaks ülevaateks sai algselt välja mõeldud, millise struktuuriga juhtimisalgoritm tööle võiks hakata. See esiagne struktuur sai ka visuaalse pildi ette saamiseks üles joonistatud ning seda on näha Joonis 6. Loomulikult aga ei jäänud struktuur siiski päris selliseks ning muudatusi, mida algselt ei osanud ette näha tuli töö käigus ette.

[1] http://wiki.ros.org/Services

[2] http://wiki.ros.org/Parameter%20Server



Joonis 6. Paketi esialgne plaanitav struktuur

6.2 TLE getter node

Esimese loodud sõlme peamiseks ülesandeks on varustada süsteemi võimalikult värske tehiskaaslaste informatsiooniga. Selleks peab programm suutma küsida iga etteantud aja tagant internetist kõige uuemat TLEd. Hetkel on selliseks ajavahemikuks määratud neli tundi, kuid seda saab vajadusel ka lihtsalt muuta. Kuna TLEd ei loeta pidevalt reaalajas vaid selle andmeid uuendatakse sõltuvalt satelliidist alates mõnest korrast päevas kuni ühe-kahe korrani nädalas, siis ei ole väga oluline, et ka projektis loodud programm infot tihedamini küsiks [12].

Värske TLE küsimiseks on kasutatud Space-Track.org-i[1] poolt pakutavat APIt. API abil on võimalik saada korraga andmeid nii ühe kui mitme Maa tehiskaaslase kohta küsitud formaadis. Selleks peab teadma vaid vajaliku satelliidi NORADi kataloginumbrit. Kuigi vaikimisi annab Space Track andmeid TLE kujul ilma esimese reata, kus on kirjas vaid

[1] https://www.space-track.org/

satelliidi nimi, siis pyephem vajab arvutuste läbiviimiseks ka satelliidi nime. Seetõttu küsitakse Space Trackilt hoopis 3LE formaadis andmeid.

6.3 Satellite chooser node

Kuna antenni hakatakse hiljem kasutama peale TTU100 ka teiste satelliitide jälgimiseks, siis on vaja, et oleks võimalik programmi töö ajal ka hetkel jälgitavat satelliiti vahetada. See tähendab, et vajadusel peab olema programm võimeline mujalt tuleva päringu peale enda hetke töö pooleli jätma ning asuda jälgima uut huvipakkuvat satelliiti. Selle päringu vastu võtmiseks ja töötlemiseks on loodud sõlm *satellite_chooser_node*. Peale sõlme käivitamist ei tee see algselt mitte midagi rohkemat, kui loob teema nimega */antenna/satellite_chooser* ning jääb teistest sõlmedest eraldiseisvalt kuulama sellele teemale saadetud sõnumeid. Seni kuni juhtimisalgoritm töötab ja satelliidi vahetamiseks vajadust ei ole, pole ka antud sõlmel kuigi palju teha. Kui ühel hetkel aga saadetakse loodud teemale sõnum uue satelliidi NORADi kataloogi numbriga, saab sõlm tööle asuda.

Sõlme enda töö on lihtne. Esiteks peab sõlm kontrollima sõnumiga saadud numbri korrektsust, ehk seda, kas sellise identifikaatoriga satelliiti üldse eksisteerib ning kas see satelliit ilmub mingil ajal piisavalt kõrgele horisondist, et seda oleks mõtet jälgima hakata. Kui id on korrektne, siis salvestab sõlm selle faili, kust saab muutusest teada ka satelliite jälgiv sõlm. Vastasel juhul antakse veast kasutajale teada ning hetkel jälgitavat satelliiti ei vahetata. Samuti teavitatakse kasutajat ka õnnestunud satelliidi vahetuse korral ning kontrolliks kuvatakse ka uue satelliidi andmed.

6.4 Satellite tracker node

Suurim osa satelliitide ülelendude jälgimiseks loodud programmi loogikast asub sõlmes nimega *satellite_tracker_node*. Kuigi sõlme peamiseks ülesandeks oli jälgida küsitud TLE põhjal satelliitide järgmisi ülelende ning nende põhjal leida, kuhu antenn pöörama peab, siis tegelikkuses osutus ülesanne palju keerulisemaks ning erinevaid kontrolle, millega pidi pidevalt arvestama, tuli juurde mitmeid.

Programmi tööle pannes jääb see algselt uut TLEd ootama. Kui TLE on teada, saab selle põhjal ennustada satelliidi trajektoori ja aega, millal see antenni vaatevälja jõuab. Antenni koordinaadid on täpselt ette antud ning kuna antenn jääb statsionaarselt Mektory maja

katusele, siis ei ole ka vajadust seda programmi töö ajal muuta. Hetkel, kui antenn veel enda lõppasupunktis ei ole, on nendeks koordinaatideks võetud Google Mapsist[1] saadud Mektory hoone pikkus- ja laiuskraadid. Need koordinaadid koos jälgitava satelliidi TLEga antakse ette Pyephemile, mis arvutab nende põhjal satelliidi trajektoori. Täpsemalt küsitakse satelliidi järgmise ülelennu, mille puhul tõuseb satelliit horisondist kõrgemale kui 10 kraadi, asimuuti, elevatsiooni ja aega hetkel 5 sekundi täpsusega. Peale seda saab kood hakata aja järgi kontrollima, millal satelliit vaatevälja jõuab.

Olles saanud kätte ülelennu andmed saab asuda seda jälgima. Esimese ülesandena peab aga antenni koheselt pöörama ülelennu alguspunkti, et vältida ülelennu ajal infokadu. Selleks annab programm testimiseks simulatsioonile ette asimuudi ja elevatsiooni, kuhupoole peab antenn pöörama. Päris antenni liigutamiseks aga arvutatakse esialgu vajaminev kiirus ja kiirendus ning need saadetakse *antenna_controller_node* sõlmele. Võimalik, et tulevikus nendest andmetest ei piisa ning antenni liigutamiseks on vajalik ka mõni muu parameeter. Peale pööramist hakkab programm aja järgi kontrollima satelliidi asukohta ning sellele vastavalt andma käsklusi antenni pööramiseks. Pidevalt enne käsu andmist küsitakse antennilt tema hetke asukohta, kuna reaalses maailmas ei toimu liigutamine kindlasti perfektselt ning antenn võib liikuda soovitud asukohast mööda. Küsides hetke asukohta, saab vajalikud arvutused sooritada täpsemalt ja ei pea muretsema, et positsioon aja möödudes järjest rohkem eksima hakkaks.

Nagu juba mainitud, siis antenni liigutamiseks on vaja teada vähemalt lõppkiirust ω ja kiirendust α, mida juhitavatele mootoritele rakendada. Et vajalikke parameetreid leida on praeguses programmi versioonis kasutatud arvutamiseks kinemaatika valemeid. See tähendab, et jälgimise ajal on igal ajavahemikul kiirendus enda ajavahemiku ulatuses konstantne [13]. Sellise kiirenduse leidmiseks on kasutatud valemit (1).

$$\theta = \omega_0 t + \frac{1}{2}\alpha t^2 \qquad (1)$$

Kus θ on nurknihe radiaanides, ehk nurk, mille antenn läbib, ω_0 antenni algkiirus, α antenni kiirendus, t aeg, mis liikumisele kulub.

[1] https://www.google.com/maps/@59.3946328,24.6608732,17z

Kuna muud parameetrid on varasemalt juba olemas, siis on valemist (1) võimalik kerge teisendusega leida α väärtus. Antenni lõppkiiruse arvutamiseks kasutatakse aga valemit (2).

$$\omega^2 = \omega_0^2 + 2\alpha\theta \Rightarrow \omega = \sqrt{\omega_0^2 + 2\alpha\theta} \quad (2)$$

Kuna valemi (2) abil arvutatakse uus lõppkiirus iga soovitud ajahetke tagant, siis saab leitud väärtuse võtta järgmise ajahetke algkiiruseks.

Üks olukord, mis esineb küll suhteliselt harva, kuid millega peab siiski arvestama tekib siis, kui uue satelliidi jälgima asumise hetkel on satelliit juba nägemisulatuses. Sellisel juhul peaks antenn koheselt pöörama juba pea kohal asuva satelliidi poole, et ei jääks ilma olulisest infovoost. Küll aga ei ole alati mõistlik sellist satelliidi ülelendu siiski jälgima hakata. Kuna antennil on kindel maksimumkiirus, siis ei pruugi antenn jõuda enne satelliidi horisondi taha kadumist selleni üldse ennast pöörata. Lisaks ei toimu ka vajaliku kiiruseni kiirendamine hetkega. Seetõttu tuleks arvutada aeg, mille jooksul antenn suudaks ennast piisavalt pöörata ning vaadata, kas peale saadud aja möödumist on satelliit veel nägemisulatuses. Kui on, siis asuda seda jälgima, kui mitte, siis liikuda järgmise ülelennu peale. Kuna hetkel ei ole antenni maksimaalne liikumiskiirus veel täpselt teada, siis peab selle tuletama. Arutelu käigus sai arvestatud, et antenn peaks liikuma küllaltki mõistliku ajaga näiteks suutma keerata ennast 360 kraadi võrra 5 minutiga. Siit saab kerge arvutuse teel leida antenni oletusliku maksimaalse liikumiskiiruse (3).

$$\omega = 360 \text{ deg} / 5 \text{ min} / 60 \text{ s} = 1.2 \text{ deg/s} \approx 0.0209 \text{ rad/s} \quad (3)$$

Teades antenni hetke positsiooni ning positsiooni, kuhu see peab jõudma, saab leida nende positsioonide vahe kraadides θ ehk nurga, mille võrra peab antenn suutma ennast keerata, et jõuaks satelliiti veel jälgida. Valemitest (1) ja (2) saab tekitada võrrandisüsteemi (4).

$$\begin{cases} \theta = \omega_0 t + \frac{1}{2}\alpha t^2 \\ \omega^2 = \omega_0^2 + 2\alpha\theta \end{cases} \quad (4)$$

Võrrandisüsteemi (4) lahendades saab kätte ruutvõrrandi (5).

$$(\omega^2 - \omega_0^2)t^2 + 4\theta\omega_0 t - 4\theta^2 = 0 \quad (5)$$

Rakendades leitud ruutvõrrandile valemit (6), saab lõpuks lihtsustamise teel kätte avaldised (7).

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{6}$$

$$\begin{cases} t_1 = \frac{2\theta}{\omega + \omega_0} \\ t_2 = \frac{-2\theta}{\omega - \omega_0} \end{cases} \tag{7}$$

Eeldusel, et kõik eelnevalt leitud andmed on korrektsed, on leitud lahenduste seas üks positiivne ning teine negatiivne arv. Kuna aga pööramise aeg ei saa olla kunagi negatiivne, siis antud teise vastuse saab välistada.

6.5 Antenna position publisher node

Kuna töö käigus peab pidevalt arvestama antenni hetke täpse asendiga, siis oli vaja luua sõlm, mis edastaks antennilt saadud informatsiooni õigel kujul programmile. Kuigi ainult simulatsiooniga töötades ei ole selleks otsest vajadust ning töö saaks ka tehtud ilma vahepealse sõlmeta, siis hilisemat tööd arvestades muudab see andmete töötlemise ja pärimise lihtsamaks ja mugavamaks. Antud sõlm hetkel jälgib pidevalt Gazebo simulatsioonist saadetud mõlema liigendi positsiooni ning paneb need kokku üheks uueks sõnumiks, mis seepeale väljastatakse. Seda sõnumit saab juba küsida põhiprogramm.

6.6 Weather tracker node

Oma tavapärase töö kõrvalt peab antenni juhtimisel arvestama ka sellega, et igasuguse ilma juures ei tohi antenn satelliitide jälgimist jätkata. Et vältida võimalike tuulest tingitud kahjude tekkimist, peab juhtimissüsteem suutma vajadusel keerata antenni asendisse, kus tuul sellele kõige vähem mõju avaldab. Hetkel läbi viidud arvutuste põhjal on selliseks antenni asendiks risti tuule suunaga ning maapinna suhtes paarikraadise nurga all olek.

Et oleks võimalik aga ilmastikuga arvestada on programmile loodud juurde sõlm *weather_tracker_node*. Antud sõlm saab andmeid tuule kiiruse, tuule suuna ning muude võimalikke mõju avaldavate ilmategurite kohta ning viib nende andmete põhjal läbi vajalikud arvutused, et leida, kas antenni töö peab ettevaatuse huvides peatama või mitte. Samuti peab see vajadusel leidma ka optimaalsed nurgad, kuhu antenni pöörama peab. Hiljem, kui antenni juurde ka ilmajaam püsti on rajatud, hakkab sõlm saama andmeid just

sealt. Kuna aga hetkel ei ole veel detailseid andmeid rajatava ilmajaama kohta, siis on testimiseks kasutatud praegu OpenWeatherMapi[1] APIt, mille käest küsitakse järgmise paari päeva ilmaennustuse andmed 3 tunnise intervalliga. Antud andmete põhjal tehtud arvutused kirjutatakse ROSi parameetri serverisse, kust saab need kätte juba põhiprogramm.

6.7 Antenna controller node

Peale antenni uue positsiooni, kiiruste ning kiirenduste arvutamist, peab selle informatsiooniga saama ka antenni reaalselt juhtida. Selle tööga tegeleb antud sõlm. Sõlm kuulab pidevalt põhiprogrammi poolt saadetud infoliini ning uue sõnumi saabumisel annab antennile käskluse pööramiseks. Kuna hetkel on antenni asemel vaid Gazebo simulatsioon, mis võtab sisendiks vaid uued radiaanid, siis muu info lihtsalt trükitakse kasutaja jaoks välja. Hiljem leiavad ka teised andmeosad antenni juhtimise juures kasutust.

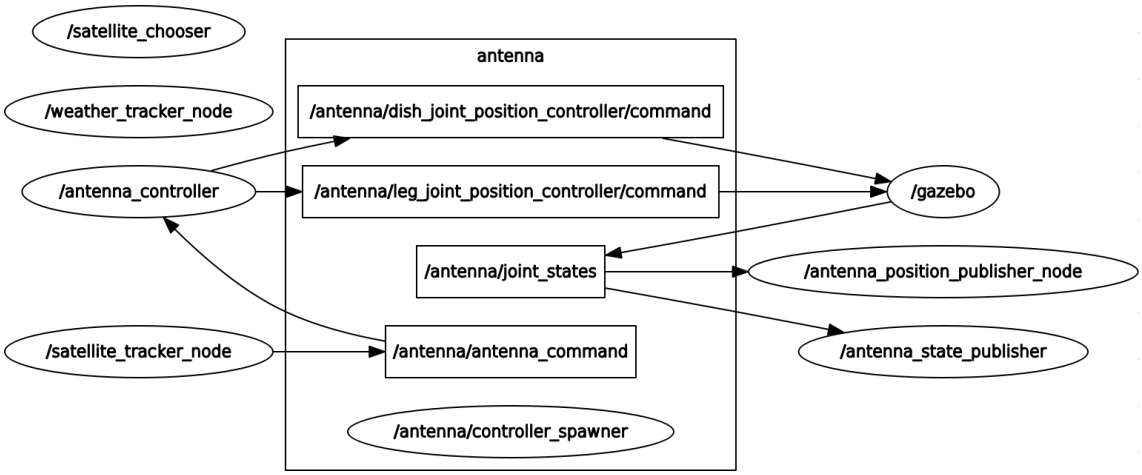
Sama põhimõttega on võimalik tulevikus lisada ka muudest sensoritest saadavat infot algoritmi täpsemaks töötamiseks. Peale antenni valmimist ja üles seadmist saab programmile lisada küllaltki kerge vaevaga muid osasid, millega tuleks juhtimissüsteemi puhul arvestada. Sellisteks teguriteks võivad olla näiteks muud keskkonna mõjud, antenni valmistamisel ja ülesseadmisel tekkinud ebatasasused, mootori sisetemperatuur jne. Kuna hetkel aga ei ole veel täpselt teada, mis ning kui palju süsteemi mõjutama hakkab, siis neid veel koodis realiseeritud ei ole.

6.8 Juhtprogrammi ülesehitus

Töö käigus kiirelt erinevate osade käitumise kontrollimiseks sai tihti kasutatud ROSi tööriista nimega rqt. Rqt on tarkvara, mis võimaldab graafilise kujunduse abil luua robotiga suhtlemiseks vaja minevaid liideseid. Rqt-l on hulk pluginaid, mis lasevad lihtsal moel muuta erinevate osade väärtusi, anda ette tööks vaja minevaid käsklusi või kujutada mõne osa tööd ette graafil. Nii sai näiteks katsetatud PID kontrolleritele erinevate väärtuse määramist, ilma et oleks vahepeal pidanud antenni töö katkestama. Et visualiseerida

[1] https://openweathermap.org/api

loodud juhtimissüsteemi struktuuri korrektsust ning saada sellest parem ülevaade on kasutatud ühte suurimat rqt pluginat rqt_graph. Rqt_graph loob hetkel töös olevast ROSi süsteemist visuaalse pildi, kuvades sellel kõiki parasjagu töösolevaid sõlmi ja ühendusi nende vahel [4]. Antenni juhtimissüsteemi visuaalne mudel on kujutatud Joonis 7.



Joonis 7. rqt_graph antenni juhtimissüsteemi mudel

6.9 Juhtprogrammi testimine

Et olla kindel tehtud töö korrektsuses, pidi loodud tarkvara ka pidevalt testima. Kuna programm jälgib pidevalt reaalajas ülelendavaid satelliite ning teeb nendelt saadud andmete põhjal arvutusi, siis tavapäraste testide otsene kirjutamine oli raskendatud. Kõige kasulikumaks osutus programmi töö otsene visuaalne testimine, kas siis rqt abil või mõnda muud loogikat kasutades, peamiselt integratsioonitestimise tasemel. Testimist sai pidevalt tööle paralleelselt praktiseeritud.

Üks olulisemaid osasid valminud programmi juures oli satelliidi ülelennu detaile ennustav ning satelliidi liikumist ajas jälgiv kood. Peale Pyephemilt ülelennu andmete kätte saamist sai saadud tulemusi võrreldud ka Orbitronist ning Gpredictist saadud andmetega, et olla kindel kirjutatud programmi töös. Olles veendunud andmete korrektsuses pidi kindlaks tegema, et antud andmete põhjal suudab kood hakata õigel ajal antenni veatult liigutama. Reaalajas jälgimiseks kuvatakse kasutajale pidevalt hetke aega ning aega, millal satelliit vaatevälja jõuab ja programm antenni liigutamisega pihta saab hakata.

Kuna liigutamisprotsessi ajal oli uute mõõdetud ja arvutatud andmete hulk küllaltki suur, siis ei olnud kuigi kerge nende korrektsust reaalajas kontrollida. Et saada kogu protsessist

parem ülevaade ning veenduda et antenn jälgib terve ülelennu ajal satelliiti ühtlaselt, salvestab programm erinevad töö jaoks olulised andmed faili. Antud andmete põhjal oli võimalikke jälgimise ja liigutamisega seotud vigu palju mugavam otsida. Lisaks sai andmete põhjal joonistada Exceli[1] ning Pythoni pyploti[2] raamistikuga graafikuid, mille abil oli võimalikke vigade või murekohtade leidmine juba palju kergem. Paar sellist graafikut on lisatud ka lisadesse 1 ja 2.

Koodi katsetamisel ja testimisel osutus suurimaks tüliks pidevalt uute satelliitide ülelendude ootamine. Kuna programm jälgib satelliite reaalajas, siis pidi pidevalt enne töö korrektsuse kontrollimist ootama, et parasjagu huviorbiidis olev satelliit vaatevälja jõuaks. Kui esimestel päevadel sai koodi testitud ainult EstCube-1 abil, siis üsna kiiresti sai selgeks, et nii pole mõtet jätkata ja peab leidma viisi lähemal asuva satelliidi leidmiseks. Suure osa ajast aitas siinkohal hädast välja Gpredict, mille abiga oli salvestatud satelliitidest võimalik leida järgmisena vaatevälja jõudev. Kuna aga ka see vajas manuaalselt satelliitide kontrollimist ja järgmise kirja panekut, siis sai lõpuks kirjutatud lisasõlm, mis selle töö ise ära tegi. Täpsemalt küsib sõlm tööle panekul andmeid kõigi kuupsatelliitide kohta ning leiab nende seast lähima, et seda siis jälgima asuda. See kiirendas töö testimist märgatavalt.

[1] https://products.office.com/en-us/excel

[2] https://matplotlib.org/api/pyplot_summary.html

7 Edasine töö

Siiamaani tehtud töö on olnud suures osas teoreetiline. Kuigi enamik kirjutatud koodist leiab praegusel kujul ka reaalse antenni liigutamise juures kasutust, siis leidub ka punkte, mis suure tõenäosusega uute täpsemate andmete selginemisel ära muutma peab. Näiteks on üks sellistest osadest kindlasti süsteemi antenniga ühendamine, mis simulatsiooni puhul töötab reaalse maailmaga erinevalt.

Lisaks mingite osade muutmisele peab tulevikus peale antenni valmimist uurima sellele mõjuvaid võimalikke keskkonnategureid ning ehitamise käigus tekkinud ebatäpsuseid ja otsustama, mis neist ning millisel moel võivad segada või muudmoodi mõjutada juhtimisalgoritmi tööd. Probleemide tekkimisel peab ka vajalikud arvutused nende lahendustega kooskõlla viima. Üldiselt on praegust tarkvara luues peetud silmas ka seda, et töö oleks laiendatav ning et uute sensorite juurde panek ROSi sõlmede kujul ei oleks kuigi keeruline. Kindlasti kavatseb autor projekti juures edasi töötada ning tegeleda juhtimissüsteemiga ka peale antenni lõplikku valmimist.

8 Kokkuvõte

2019. aasta esimeses pooles saab Tallinna Tehnikaülikoolist teine Eesti ülikool, kes on enda ehitatud satelliidi Maa orbiidile saatnud. Et ka peale orbiidile saatmist oleks võimalik TTU100 satelliidiga ühendust pidada pannakse maajaama püsti paraboolantenn. Töö põhieesmärgiks oli luua sellele antennile juhtimissüsteem, mis suudaks jälgida parasjagu huvi pakkuvat satelliiti ning sellele vastavalt antenni pöörata.

Bakalaureusetöö käigus sai ROSi ja Gazebot kasutades loodud simulatsioonimudel paraboolantennile, mis liigub sarnaselt ka valmivale antennile. Lisaks sai loodud Pythonis tarkvara, mis suudab TLE abil ajas jälgida satelliidi ülelende ning sealt ja antennilt saadud informatsiooni järgi liigutada simulatsioonimudelit. Programm jälgib ka hetke ilma ning vajadusel pargib antenni ära positsiooni, kus sellele tuul kõige vähem kahju võiks tekitada. Tarkvara arvutab ka hilisemaks reaalse antenni liigutamiseks vaja minevad teadaolevad parameetrid. Süsteem töötab iseseisvalt ja ei vaja kasutaja pidevat juuresolekut.

Kuigi töö algne eesmärk oli luua paraboolantennile juhtimissüsteem, siis kogu funktsionaalsust veel võimalik täide viia ei olnud. Põhjuseks on reaalse antenni alles pooleli olek ning seetõttu ei ole veel täpselt teada, milliste parameetritega hilisem juhtimine välja hakkab nägema. Antenni enda valmimine aga ei sõltu autorist. Sellegipoolest on süsteem loodud selliselt, et uute vajaminevate osade juurde lisamine ei oleks kuigi keeruline ning et programmi oleks võimalik vajaminevate muutustega täiendada.

Kasutatud kirjandus

[1] R. Gordon, „TTÜ100 satelliidi tutvustus,“ [Võrgumaterjal]. Available: https://ttu.ee/projektid/mektory-est/satelliidiprogramm-4/satelliidiprogramm/. [Kasutatud 04 2018].

[2] R. Adelbert, „TUT-MEKTORY NANOSATELLITE System Requirements Specification TMSS-SYS-RS-01,“ 2016.

[3] 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.

[4] „Core Components,“ Open Source Robotics Foundation, [Võrgumaterjal]. Available: http://www.ros.org/core-components/. [Kasutatud 04 2018].

[5] „URDF in Gazebo,“ Open Source Robotics Foundation, 2014. [Võrgumaterjal]. Available: http://gazebosim.org/tutorials/?tut=ros_urdf.

[6] „xacro - ROS Wiki,“ Open Source Robotics Foundation, 2018. [Võrgumaterjal]. Available: http://wiki.ros.org/xacro.

[7] „ros_control - ROS Wiki,“ Open Source Robotics Foundation, 2017. [Võrgumaterjal]. Available: http://wiki.ros.org/ros_control.

[8] T. Kelso, „Frequently Asked Questions: Two-Line Element Set Format,“ Jaanuar 1998. [Võrgumaterjal]. Available: http://celestrak.com/columns/v04n03/.

[9] „Altitude & Azimuth: The Horizontal Coordinate System,“ Time and Date AS, [Võrgumaterjal]. Available: https://www.timeanddate.com/astronomy/horizontal-coordinate-system.html. [Kasutatud 05 2018].

[10] „Nodes - ROS Wiki,“ Open Source Robotics Foundation, 2012. [Võrgumaterjal]. Available: http://wiki.ros.org/Nodes. [Kasutatud 04 2018].

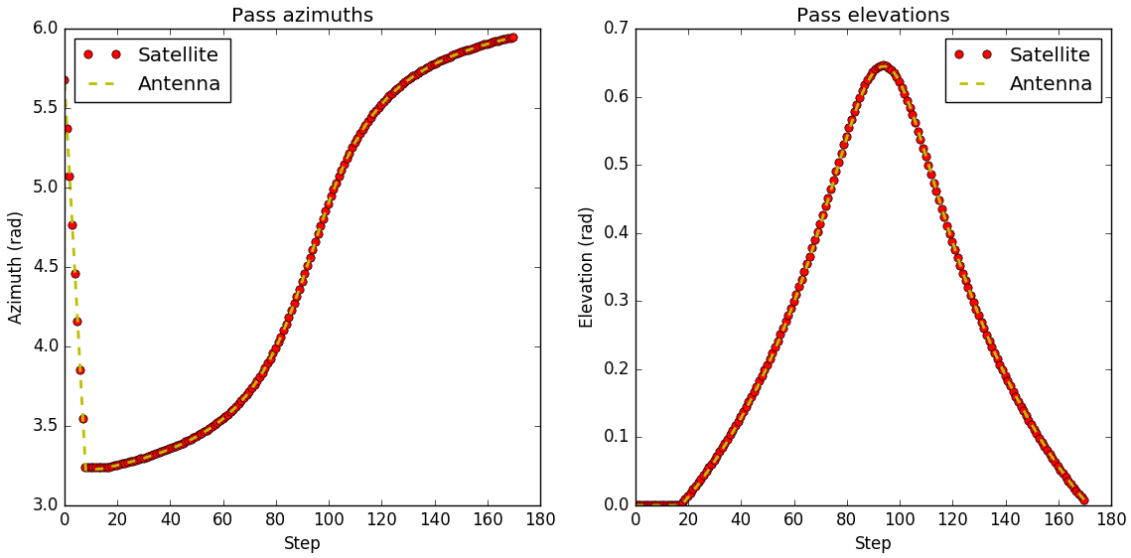
[11] „Packages - ROS Wiki,“ Open Source Robotics Foundation, 2015. [Võrgumaterjal]. Available: http://wiki.ros.org/Packages. [Kasutatud 04 2018].

[12] T. Kelso, „More Frequently Asked Questions,“ Märts 1998. [Võrgumaterjal]. Available: https://celestrak.com/columns/v04n05/.

[13] „What are the kinematic formulas?,“ Khan Academy, [Võrgumaterjal]. Available: https://www.khanacademy.org/science/physics/one-dimensional-motion/kinematic-formulas/a/what-are-the-kinematic-formulas. [Kasutatud 05 2018].

Lisa 1 – Antenni ja satelliidi koordinaatide võrdlus

Joonisel on kujutatud kuupsatelliidi NanoSatC-Br1 ühe ülelennu asimuudid ja elevatsioonid matplotlib.pyplot graafikul (1 step = 5 sekundit).



Lisa 2 – Jala kiirus

Joonisel on kujutatud kuupsatelliidi Diamond Blue ühe ülelennu antenni jala kiiruste arvutused Exceli graafikul (1 step = 5 sekundit).