TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Jordan Jorš 164071IAPB

MOODUL TTÜ100 TUDENGISATELLIIDI ALAMSÜSTEEMIDELE UUE TARKVARA LAADIMISEKS

bakalaureusetöö

Juhendaja: Evelin Halling

PhD

Tallinn 2019

Autorideklaratsioon

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

Autor: Jordan Jorš

21.05.2019

Annotatsioon

Töö eesmärgiks on luua satelliidi alamsüsteemidele tarkvara üleslaadimise moodul ning olemasoleva elektritoitesüsteemi üleslaadimise mooduli kohandamine vastavalt uuele protokollile.

Tarkvara üleslaadimise moodul koosneb kahest osast: kasutajaliidesest ja *back-end*-st.

Töös on kirjeldatud missioonijuhtimisetarkvara üldist arhitektuuri, tarkvara üleslaadimise mooduli töökäiku ning testimist.

Töö tulemusena on võimalik kasutajaliidese kaudu saata soovitud alamsüsteemile uus tarkvara.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 31 leheküljel, 7 peatükki, 14 joonist, 2 tabelit.

Abstract

Module for updating TTÜ100 student satellite's subsystems software

TTU100 student satellite project has been in development since 2014. Creation of Mission Control Software was started by a master student Siim Romanov, who set up the architecture and picked suitable technology for development. Many of the participants in the satellite program are students. The project gives university students the opportunity to be a part of a real space program, having an actual impact and getting to know what it is like to work in a project as a team. The positions available for work in the project are quite diverse, one of them is software updating.

The aim of this thesis is to create a module for updating the software of TalTech student satellite subsystems. As the satellite may orbit the Earth for several years, the software on a subsystem may become outdated and need renewal.

One of the goals is to rework the existing software upload for EPS. It is no longer up to date with the newest protocol. Another objective is to create a new module for updating the other subsystems of the satellite.

The matters assessed in this thesis include a brief overview of the Mission Control Software arhictecture, requirements for the software updating module, differences between satellite subsystems regarding software upload, the software uploading process and testing with unit tests, response imitation tests and prototype testing.

As a result, any of the subsystem’s software can be updated through the MCS user interface.

The thesis is in Estonian and contains 31 pages of text, 7 chapters, 14 figures, 2 tables.

Lühendite ja mõistete sõnastik

EPS	*Electronic power system*, satelliidi elektritoitesüsteem.
OBC	*On-Board Computer*, pardaarvuti, satelliidi aju.
ADCS	*Attitude Determination- and Control System*, satelliidi asendikontrolli ja juhtimise süsteem
COM	*Communication*, kommunikatsiooni alamsüsteem.
MCS	*Mission control software*, missioonijuhtimisetarkvara
JSON	*Javascript Object Notation*, notatsioon objekti andmete talletamiseks.
FRAM	*Ferroelectric Random-Access-Memory*, ferroelektriline muutmälu.
HDLC	*High-level Data Link Control*, *bitipõhise piiratud kaadriga andmeside standardprotokoll* [1]
AX.25	*Amateur X.25*, amatöör raadioside protokoll.
XTEA	*eXtended Tiny Enryption Algorithm*, täiendatud väike krüpteerimise algoritm.
REST	*Representational State Transfer*, tarkvara arhitektuuri laad
API	*Application programming interface*, liides uute komponentide liidestamiseks

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tallinna Tehnikaülikooli tudengisatelliidi projekt sai alguse 2014. aastal, mille käivitas Mektory Kosmosekeskus. Projekti eesmärk on Maa orbiidile saata 1U nanosatelliit, mis hakkab koguma erinevaid andmeid. Missiooni juhtimiseks tarviliku tarkvaraga loomisega alustas Siim Romanov, kes tegi magistritöö kuupsatelliidi missioonijuhtimistarkvara arhitektuurist. Satelliidiprogramm võimaldab üliõpilastel osaleda kosmoseprojektis, kus nad saavad meeskonnatöö kogemusi ning teadmisi erinevatest tehnoloogiatest.

Töö peamine eesmärk on arendada satelliidi alamsüsteemidele uue tarkvara üleslaadimise moodul ning kohandada olemasolevat moodulit, mis oli loodud elektritoitesüsteemi tarkvara uuendamiseks. Olemasolevat moodulit on vaja kohandada, sest sõnumivahetuse protokoll maajaamas on muutunud.

Töö esimeses peatükis kirjeldatakse missioonijuhtimisetarkvara arhitektuuri. Peatükk annab ülevaate üldisest arhitektuurist, haldusrakendusest ning toob välja missioonijuhtimisetarkvara põhimooduli komponendid.

Töö teises peatükis on välja toodud nõuded alamsüsteemide tarkvara uuendamisele. Nõuded on esitatud satelliidi alamsüsteemide uuendamisele, kasutajaliidesele ning tarkvara failide asukohale.

Kolmas peatükk annab ülevaate satelliidi alamsüsteemide uuendamisest. Peatükis on kirjeldatud üldine kommunikatsiooniprotokoll, alamsüsteemide ülesanne satelliidis, alamsüsteemide käsu- ja veakoodid, ning tarkvara andmete krüpteerimine.

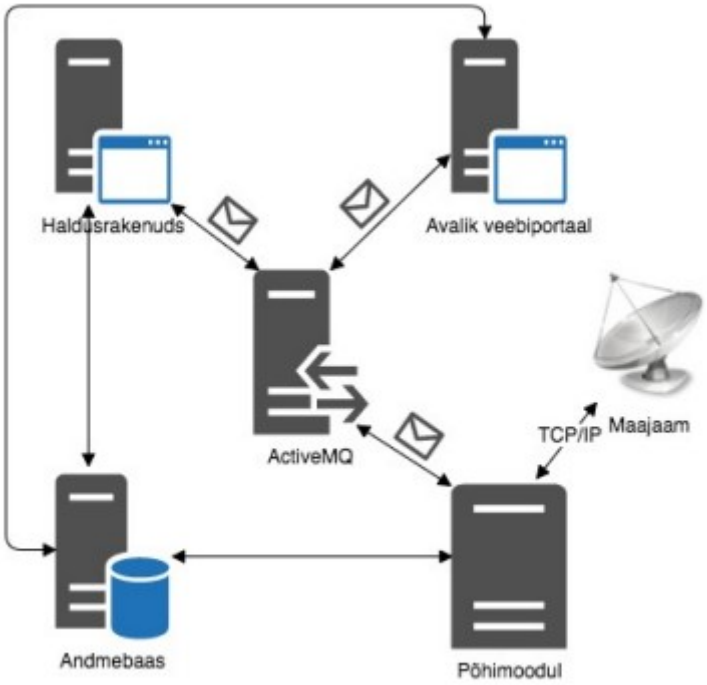
Neljandas peatükis kirjeldatakse töö realisatsiooni. Peatükis sisaldub andmebaasi tabelite kirjeldus, tarkvara failide kättesaamine serverirakendusse, REST teenuse kasutamine ning kogu tarkvara uuendamise töövoog elektritoitesüsteemi näitel.

Viies peatükk toob välja erinevad viisid loodud lahenduse testimiseks. Peatükis on kirjeldatud ühiktestimist, vastuste imiteerimist ning prototüübi testimist.

2 Missioonijuhtimisetarkvara arhitektuur

Missioonijuhtimisetarkvara peamise arhitektuuri moodustavad haldusrakendus, avalik veebiportaal, sõnumivahetuse komponent, andmebaas, põhimoodul ja maajaam. [2]

2.1 Üldine arhitektuur



Joonis 1. Tarkvara üldine struktuur [2, p. 24]

2.1.1 Andmebaas

Missioonijuhtimisetarkvara kasutab PostgreSQL andmebaasi. Siim Romanov valis PostreSQL-i sellepärast, et see on vabavaraline, relatsiooniline, vastab andmebaasile esitatud nõuetele ning tal on sellega varasemalt kogemusi. Teine alternatiiv oleks olnud MySQL, kuid selles ei olnud 2017. aastal nii head ruumiandmete hoiustamise toetust, kui PostgreSQL-s. [2, pp. 24-25]

Andmebaasi muudatuste jälgimiseks ning haldamiseks on projektis kasutusel Liquibase[1] tarkvara. Liquibase ühildub hästi Git-i versioonihaldustarkvaraga, mis võimaldab andmebaasi muudatuste ajalugu jälgida ning rakendada erinevates keskkondades. [2, p. 25]

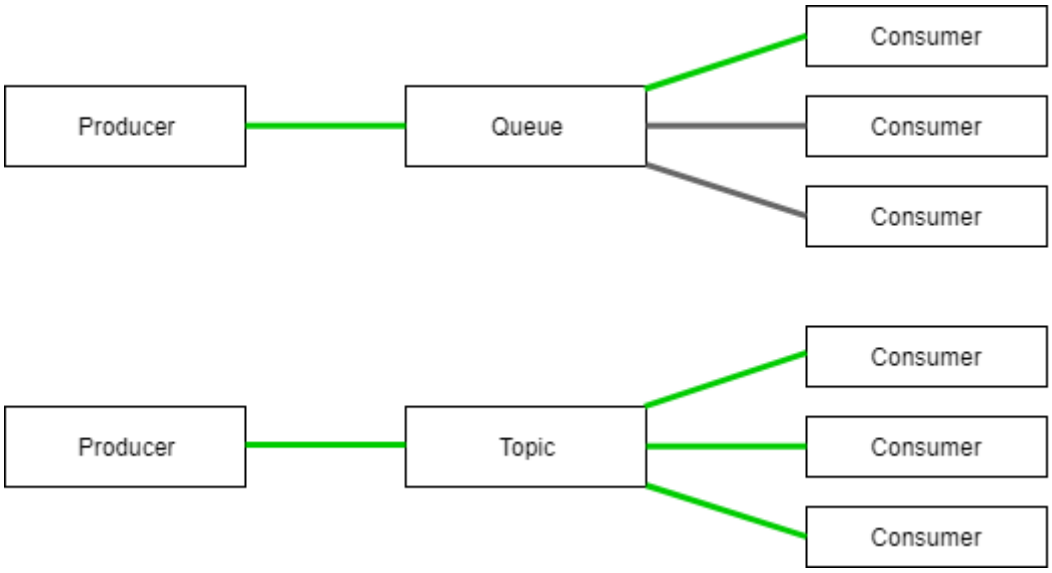
2.1.2 Sõnumivahetuse komponent

Missioonijuhtimisetarkvara koosneb mitmetest erinevatest moodulitest. Nende moodulite liidestamiseks on kasutusel sõnumivahetuskomponent ActiveMQ[2], mis põhineb sõnumipõhisel arhitektuuril. [2, p. 25] *Sõnumipõhise arhitektuuri korral on süsteemis keskne sõnumivahetuskomponent, mis vahendab ja koordineerib sõnumite liiklust saatvate ja tarbivate osapoolte vahel* [2, p. 26].

Arhitektuuri looja otsustas sõnumivahetusel põhineva arhitektuuri kasuks, sest sellel on mitmeid eeliseid. Nendeks eelisteks on järgnevad funktsionaalsused: *andmeside asünkroonsus, sõnumite suunamine erinevate tarbijate vahel ja sõnumite puhverdamine* [2, p. 26].

[1] http://www.liquibase.org/

[2] http://activemq.apache.org/



Joonis 2. Sõnumivahetuse mudel [3]

Sõnumivahetuse mudel kirjeldab kaht tüüpi sihtpunkti: *queue* ehk järjekord ning *topic* ehk teema. Järjekorrast suunatakse sõnum ühele tarbijale, kes on selle konkreetse järjekorra tarbija. Kui tarbijaid ei ole, siis sõnum hoitakse alles seni, kuni tarbija on saadaval. Teema puhul tehakse sõnumist koopia igale tarbijale, kes on selle konkreetse teema tarbijad. [4]

2.2 Haldusrakendus

Haldusrakendus on veebiportaal, mille kaudu kasutaja saab teostada missioonijuhtimist. Kasutajaliidese kaudu on võimalik:

- *Vaadata viimaseid ja järgmiseid ülelende;*
- *Vaadata ajas satelliidi asukohta kaardil;*
- *Vaadata satelliidi telemeetria andmeid;*
- *Planeerida järgimisi missioone ja saata käsklusi satelliidile;*
- *Vaadata alla laetud andmefaile;*
- *Hallata rakenduse kasutajaid*. [2, pp. 28-29]

Kasutajaliides on realiseeritud SPA (*single page application*) veebirakendusena. SPA tüüpi veebileht koosneb ainult ühest HTML (*hypertext markup language*) leheküljest,

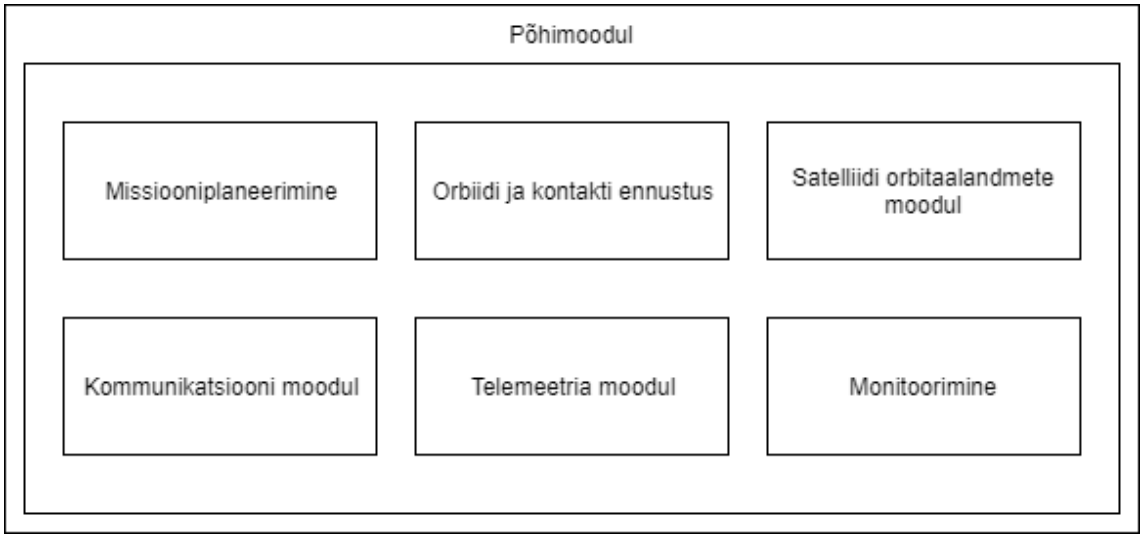
mille sisu vastavalt muudetakse dünaamiliselt Javascripti abil, kui kasutaja veebilehte kasutab. [5]

Kasutajaliidesepoolne veebirakendus on realiseeritud Angular 6[1] raamistiku abil, mis toetabki SPA veebilehtede realiseerimist. Serverirakendus on ehitatud Spring Bootis[2], mis on Java põhine teenuste loomist abistav raamistik. See suhtleb andmebaasidega ja vastab päringutele kasutajaliideselt.

Mikroteenuste ja veebirakenduse arendust toetab JHipster[3].

2.3 Missioonijuhtimistarkvara põhimoodul

Missioonijuhtimistarkvara põhimoodul jaotub 6-ks alammooduliks. Igal moodulil on kindel ülesanne ning need suhtlevad omavahel sõnumivahetuskomponendi kaudu.



Joonis 3. Põhimooduli alammoodulid [2, p. 31]

[1] https://angular.io/

[2] https://spring.io/

[3] https://www.jhipster.tech/

2.3.1 Missiooniplaneerimine

Missiooniplaneerimine on üks missioonijuhtimisetarkvara põhimoodulitest. See moodul aitab planeerida käske järgmisteks ülelendudeks. Magistritöös on näitena toodud missiooniplaneerimise mõned ülesanded:

- Ülelennu kontakti aja leidmine läbi telemeetria mooduli;
- Kontroll, kas kõik planeeritud tegevused on võimalikud ülelennu ajal;
- Käskude planeerimisel aku laetuse arvestamine [2, pp. 31-32];
- Käskude prioriteetsustasemete arvestamine [6].

2.3.2 Satelliidi orbiidi- ja kontaktiennustuse alammoodul

S. Romanovi magistritöös [2, p. 32] on satelliidi orbiidi- ja kontaktiennustuse alammooduli ülesanne kirjeldatud järgnevalt: *Satelliidi orbiidiennustus- (satellite orbit propagation) mooduli peamiseks ülesandeks on arvutada välja järgmise kontakti täpne aeg ja orbiit. Saadud tulemus salvestatakse andmebaasi ning edastatakse sõnumivahetuse komponenti. Saadud arvutuse peamiseks tarbijateks on maajaam, et muuta enda asend vastavaks järgmise kontakti alguseks ning missiooniplaneerimise moodul, teadmaks planeerida järgmist sideseanssi. Kolmandate osapooltega jagatakse arvutusi avaliku veebiportaali kasutajaliidese ja API kaudu*.

2.3.3 Telemeetria alammoodul

Telemeetria alammoodul on S. Romanovi magistritöös kirjeldatud järgmiselt: *Telemeetria mooduli eesmärgiks on kirjeldada telemeetria sõnumite sisu ning dekodeerida see vastavalt iga satelliidi alamsüsteemi spetsifikatsioonile. Igal satelliidi alamsüsteemil on kindlaks määratud registrid, mis hoiavad infot konkreetse süsteemi toimimise kohta. Saades satelliidilt telemeetria paketi, liigub see kõigepealt sõnumivahetuse komponenti ning sealt saadetakse see edasi telemeetria moodulisse, kus tuvastatakse paketi päise järgi, mis liiki telemeetriga on tegemist. Pärast paketi tuvastamist loetakse andmebaasist paketi kirjeldus, mille abil pakett dekodeeritakse ning saadud tulemused salvestatakse andmebaasi. Täiendavalt edastatakse saadud tulemused sõnumivahetusse, mis võimaldab reaalajas kõigil huvitatud komponentidel jälgida*

sissetulevaid telemeetria andmeid ning rakendada nende põhjal ettenähtud tegevusi. [2, pp. 33-34]

2.3.4 Kommunikatsiooni alammoodul

Missioonijuhtimisetarkvara arhitekt kirjeldab moodulit järgmiselt: *Kommunikatsiooni alammoodul omab ühte kõige suuremat tähtsust missioonijuhtimistarkvara põhimoodulis, kuna see realiseerib süsteemi kommunikatsiooniprotokolli dokumendis [30] kirjeldatud L3/L4 taseme protokolli. Kui haldusrakenduses on võimalik kõrgetasemeliselt ära kirjeldada satelliidile saadetavad käsud, siis kommunikatsioonimoodul kodeerib käsud vastavalt protokollile, edastab vastavalt kodeeritud paketid maajaamale ning juhib andmeside seanssi satelliidi ja maajaama vahel*. [2, p. 35]

2.3.5 Satelliidi orbitaalandmete alammoodul

Satelliidi orbitaalandmete moodul laeb alla andmeid satelliidi asukoha ja asendi kohta. Andmete küsimine toimub kindla ajaintervalli tagant. Saadud andmed dekodeeritakse, salvestatakse andmebaasi ning edastatakse teistele moodulitele sõnumivahetuskomponendi kaudu. [2, p. 37]

3 Nõuded alamsüsteemide tarkvara uuendamisele

Käesolevas peatükis on kirjeldatud nõuded, mis peavad olema täidetud, et satelliidi tarkvara uuendada. Nõuded saab jagada kaheks vastavalt esitlus- ja andmekihi põhjal. Esitluskihti esitab kasutajaliides, mida kasutades kasutaja saab teostada tarkvara uuendamist. Andmekihiks nimetame *back-end*-s paiknevat serverirakendust, mis tegeleb andmete salvestamise, muutmise ning vahendamisega. Siinkohal on ka oluline, et jääksid kehtima nõuded, mille püstitas Taavi Sutt elektritoitesüsteemile tarkvara uuendamise mooduli loomisel.

Satelliidil on kokku 4 alamsüsteemi. Nendeks on elektritoitesüsteem (EPS), kommunikatsioonimoodul (COM), asendikontrolli ja juhtimise süsteem (ADCS) ja pardaarvuti (OBC). Tarkvara uuendamine on ühesugune nendest kolmel alamsüsteemil – COM, ADCS ja OBC. Elektritoitesüsteemile uue tarkvara laadimine erineb nii käskude, tarkvara andmete paketeerimise kui ka töövoo poolest. Sellest tulenevalt on ka nõuded alamsüsteemidel erinevad.

3.1 Nõuded elektritoitesüsteemi uuendamisele

Elektritoitesüsteemi tarkvara uuendamisele on püstitatud järgnevad nõuded:

- Tarkvara uuendust saadetakse ühe rea kaupa tarkvara failist.
- Enne järgmise rea saatmist tuleb oodata eelmise rea vastust.
- Vigane või saatmata jäänud rida tuleb kohe uuesti saata enne järgmist rida.
- Saatmata ja saadetud tarkvara read peavad olema eristatavad süsteemi jaoks.
- Tarkvara uuendamine peab olema jätkatav järgmisel ülelennul.
- Tarkvara uuendamist peab saama tühistada.
- Kui ühe tarkvara rea saatmine ebaõnnestub kasutaja poolt kindlaks määratud arv korda järjest, tuleb automaatselt üleslaadimine tühistada.

3.2 Nõuded teiste alamsüsteemide uuendamisele

Alamsüsteemidele COM, ADCS ja OBC on püstitatud järgnevad nõuded:

- Tarkvara uuendust saadetakse 192 baidi suuruse andmepaketina.
- Tarkvara andmepakett peab olema krüpteeritud.
- Enne järgmise rea saatmist tuleb oodata eelmise rea vastust.
- Vigane või saatmata jäänud pakett tuleb kohe uuesti saata enne järgmist paketti.
- Saatmata ja saadetud tarkvara read peavad olema eristatavad süsteemi jaoks.
- Tarkvara uuendamine peab olema jätkatav järgmisel ülelennul.

3.3 Nõuded kasutajaliidesele

Kasutajaliidesele esitatud nõuded on järgmised:

- *Kasutaja näeb eelmise edukalt üleslaetud tarkvaraga seotud andmeid: tarkvara versioon, käsu andmise kuupäev ja kellaaeg, üleslaadimise lõpu kuupäev ja kellaaeg*
- *Kasutaja saab valida rippmenüüst üleslaetava tarkvara faili.*
- *Kasutaja ei saa pooleli oleva üleslaadimise protsessi ajal uut üleslaadimist alustada.*
- *Kasutaja saab katkestada üleslaadimise protsessi.*
- *Kasutajale visualiseeritakse üleslaadimise protsessi laadimisriba abil*. [7, p. 20]
- Kasutaja saab valida satelliidi, millele tarkvara uuendust saata.
- Kasutaja saab valida alamsüsteemi, mida uuendada.
- Kasutajale näidatakse tarkvara faile vastavalt valitud alamsüsteemile.
- Kasutaja saab üleslaadimise ajutiselt peatada.

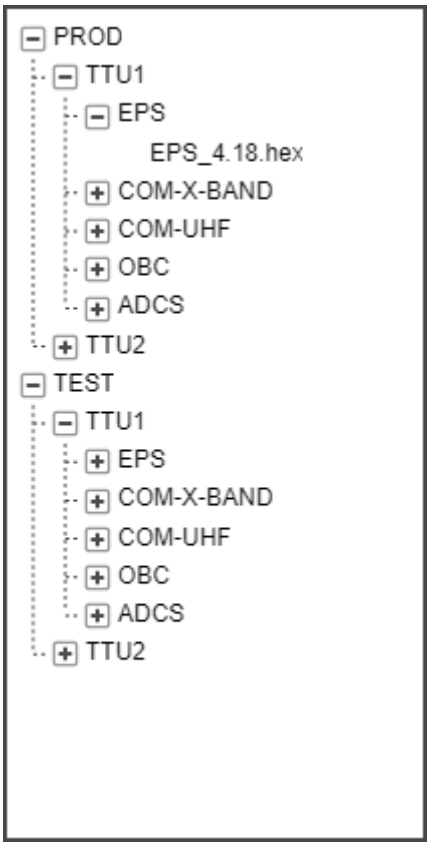
- Kasutaja saab suvalist peatatud üleslaadimist jätkata.
- Kasutaja ei saa mõnda poolikut üleslaadimist jätkata, kui on käimas aktiivne tarkvara üleslaadimise protsess.
- Kasutaja saab kustutada tarkvara uuenduste infotabelist ridu.
- Kasutaja ei saa kustutada aktiivse üleslaadimisega seotud infotabeli rida.
- Kasutaja näeb üleslaadimisega seotud tarkvara andmetes ka satelliidi nime, alamsüsteemi ning üleslaadimise staatust.

Kasutajaliidesele on lisandud juurde mitmeid nõudeid, sest Taavi Suti arendatud lahendus toetas ainult elektritoitesüsteemi uuendamist ning tarkvara infotabelis ühe rea andmete kuvamist. Uued esitatud nõuded võimaldavad tarkvara uuendamist teostada kõikidele alamsüsteemidele. Samuti kuvatakse nüüd tarkvara infotabelis informatsiooni igas seisundis üleslaadimiste, mitte ainult edukalt peale laetud tarkvara kohta. Infotabel on näidatud lisas nr 1.

3.4 Uue tarkvara asukoht

Tarkvara failid asuvad TTÜ gitlabi *satellite-software-updates* repositooriumis. Failid on paigutatud kindla struktuuriga kataloogidesse, et serverirakendus neid õigest kohast otsida oskaks. Faili nimes sisaldub tarkvara versiooni number. Uue tarkvara sisu on alati kuueteistkümnendsüsteemis ning fail on laiendiga .hex.

Järgneval joonisel on näidatud tarkvara failide kataloogipuu:



Joonis 4. Tarkvara failipuu

PROD kausta kasutatakse siis, kui rakendus on produktsioonis ehk päriselt toodangus ja kasutusel. *TEST* kataloog on kasutamiseks arenduskeskkonnas. Kataloog jaguneb edasi omakorda TTU1 ja TTU2 kaustadeks, mis on mõeldud erinevatele satelliitide tarkvara failide haldamiseks. Edasi toimub jagunemine satelliidi alamsüsteemide kataloogideks, milles sisalduvad tarkvara failid.

4 Satelliidi alamsüsteemide uuendamine

4.1 Üldine kommunikatsiooniprotokoll

Maajaama ja satelliidivaheline suhtlus toimub AX.25[1] amatöör raadioside protokolli kaudu. Andmeid saadetakse satelliidile AX.25 kaadritena, mis sisaldavad kindlaks määratud struktuuri järgi vajalikku informatsiooni.

Tabel 1. AX.25 kaadri struktuur

Flag	Address	Control	PID	Info	FCS	Flag
01111110	112 bitti	8 bitti	8 bitti	N*8 bitti	16 bitti	01111110

Flag ehk lipp tähistab vastavalt kas kaadri algust või lõppu. *Address* ehk aadress väli sisaldab endast nii lähte- kui ka sihtkoha aadressi. *Control* ehk kontrollbitid määravad kaadri tüübi. *PID* ehk protokolli identifikaator ütleb seda, kas ja milline võrgukihi protokoll on kasutusel. *Info* ehk informatsioon sisaldab saadetavaid andmeid. *FCS* ehk kaadri kontrollsumma on summa, mille põhjal saab kindlaks teha, kas kaadri sisu on vahepeal muutunud. [8]

AX.25 kaadrit ei moodustata tarkvara üleslaadimise moodulis, vaid kommunikatsiooni moodulis, sest kogu kaadri moodustamine ise igas moodulis ei oleks eriti otstarbekas. Projektis on spetsiaalsed geneerilised klassid, kuhu saab vajalikud andmed kirjutada ning millest sõnumiehitusmoodulid oskavad lõpuks AX.25 kaadri kokku panna.

Sõnumid moodulite vahel liiguvad läbi sõnumivahetuskomponendi sõnedeks muudetud Java klasside kujul. Klassid teisendatakse enne saatmist andmete vahetamise formaadi JSON[2] kujule.

[1] https://www.tapr.org/pub_ax25.html

[2] https://www.json.org/

Tarkvara üleslaadimise andmete edastamiseks kasutatakse *McsModulesToCore* klassi.

```
public class McsModulesToCore {
	private String commandParametersJson;
	private String messageUuid;
	private String user;
	private String satelliteId;
}
```

Joonis 5. Põhimoodulisse andmete saatmise klassi üldstruktuur.

Väli *commandParametersJson* kirjeldab alamsüsteemile saadetava käsu parameetreid. Väljale kirjutatakse JSON kujule teisendatud Java klass. Sõnumit vastuvõttev klass oskab JSON kuju taas Java klassiks tagasi teisendada. Väli *messageUuid* on sõnumi identifikaator, mis on vajalik sõnumi andmebaasi salvestamiseks. Väli *user* on kasutaja, kes käsku saadab. Väli *satelliteId* on satelliidi identifikaator, mida kasutatakse käsu ja satelliidivahelise seose tekitamiseks.

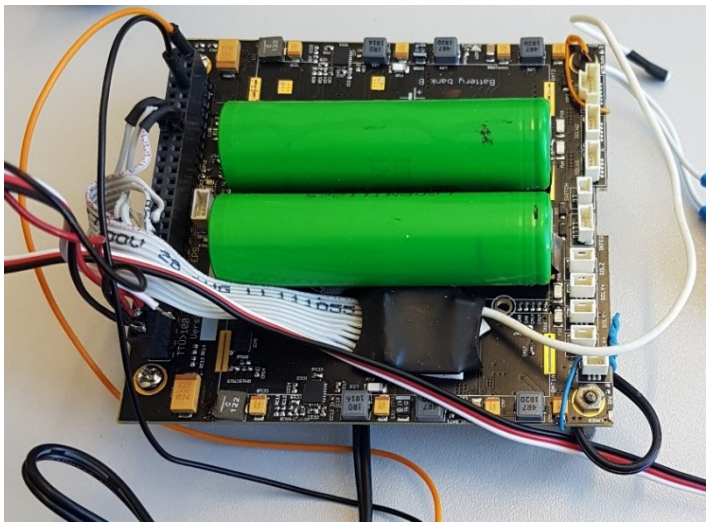
Järgneval joonisel on toodud elektritoitesüsteemi tarkvara uuendamise protsessi alustamise käsk JSON kujul, mis saadetakse edasi põhimoodulile.

```
{
	„commandParametersJson“: „{
		„type“: „CommandEpsWriteMulti“,
		„startRegisterAddress“: 997,
		„numberOfRegistersToWrite“: 255,
		„registerValues“: „“
	}“,
	„messageUuid“:“ 293cac31-8d15-4c52-b31d-6b4711a9a7a1“,
	„user“:“admin“,
	„satelliteId“:“1“
}
```

Joonis 6. Elektritoitesüsteemi tarkvara uuendamise alustamise käsk põhimoodulile.

4.2 Elektritoitesüsteem

Elektritoitesüsteem on satelliidi toiteallikas, mille ülesandeks on kõikidele alamsüsteemidele stabiilse toitepinge tagamine. Süsteem saab ise energiat Li-Ion akudest, mida toidavad päikesepaneelidega kogutud energia. Paneelid on paigutatud kuupsatelliidi igale tahule, välja arvatud sellele, millel kaamerad paiknevad. Tarkvara uuendamine toimub raadioside teel.



Joonis 7. Elektritoitesüsteemi plaat.

4.2.1 Tarkvara saatmine

Elektritoitesüsteemile tuleb saata tarkvara rida-rea kaupa selliselt, nagu on read tarkvara failis. Kui tarkvara failis on 3000 rida, siis tuleb alamsüsteemile saata vähemalt 3000 sõnumit. Mõne rea saatmise ebaõnnestumisel tuleb saata veel rohkem ridu.

EPS vastab igale sõnumile, mille ta on kätte saanud. Vastuse põhjal saab otsustada, kuidas edasi käituda. Tänu sellele, et elektritoitesüsteem vastab igale sõnumile, oli võimalik moodustada töökindel üleslaadimise töövoog, mis põhineb dupleksside ühendusel. Dupleksside on kahesuunaline samaaegne kommunikatsioon mitme seadme vahel, antud juhul maajaama ja satelliidi vahel.

4.2.2 Käsukoodid

Kõik tarkvara üleslaadimisega seotud käsud saadetakse elektritoitesüsteemile *write multiple registers* käskudena registrisse 997. Saadetavad tarkvara paketid kirjutatakse kõigepealt EPS-i *FRAM* ehk muutmällu ning kui kõik read on saadetud, siis loetakse mälust kogu tarkvara sisse.

Käesoleva lõputöö skoobis arendatud moodulis kasutusel olevad tarkvara uuendamisega seotud käsud on järgnevad:

- *0x0000 – üleslaadimise tühistamine.*
- *0x0001 – tarkvara faili rea saatmine.*
- *0x0002 – kogu tarkvara faili kontrollsumma saatmine.*
- *0x00FF – bootloader A käivitamine ning FRAM-i tühjendamine.*
- *0x0055 – tarkvara paigaldamine alamsüsteemile* [9].

Käsukood ja saadetavad andmed edastatakse põhimoodulile kasutades *McsModulesToCore* klassi (Vt Joonis 6, lk 22), kus need edasi pakitakse HDLC paketiks ja see omakorda AX.25 kaadriks.

Kui elektritoitesüsteem on käsu suutnud edukalt täita, siis vastab ta selle sama käsuga, kuid ilma andmeosata. Käsule vastamine andmetega oleks ainult lisa ajakulu. Tarvis on ainult teada, et saadetud käsu täitmine õnnestus. Järgmisel joonisel on kujutatud EPS-le saadetud tarkvara rida ning vastus AX.25 kaadri kujul (Vt Tabel 1, lk 21).

AX25 Frame:
7E455331572F53604553315A57006103F004C9100003E50001020000040000FA84057E
AX25 Frame:
7E8AA662B4AE40608AA662AE5EA66103F040C9108003E50001C6847E

Joonis 8. Tarkvara faili rea saatmine ja EPS-i vastus AX.25 kaadri kujul.

Punasega joonitud osa tähistab käsukoodi ning sinine andmeosa. Jooniselt on näha, et elektritoitesüsteem vastab sama käsuga, aga ilma andmeosata.

Käsu täitmise ebaõnnestumise korral vastab alamsüsteem veakoodiga. Sellest on täpsemalt kirjutatud järgmises peatükis.

4.2.3 Veakoodid

Elektritoitesüsteem võib vastata veakoodiga, kui proovitakse täita tegevust, mis läheks vastuollu tarkvaralise loogikaga. Näiteks võib tuua tarkvara faili rea saatmise. Iga tarkvara faili rea lõpus sisaldub selle rea kontrollsumma. EPS arvutab iga saadud rea põhjal kontrollsumma ning võrdleb seda rea lõpus oleva kontrollsummaga. Kui kontrollsummad erinevad, siis järelikult midagi on valesti. Veakood koosneb 2 baidist. Tarkvara uuendamisega seotud põhilised veakoodid on järgnevad:

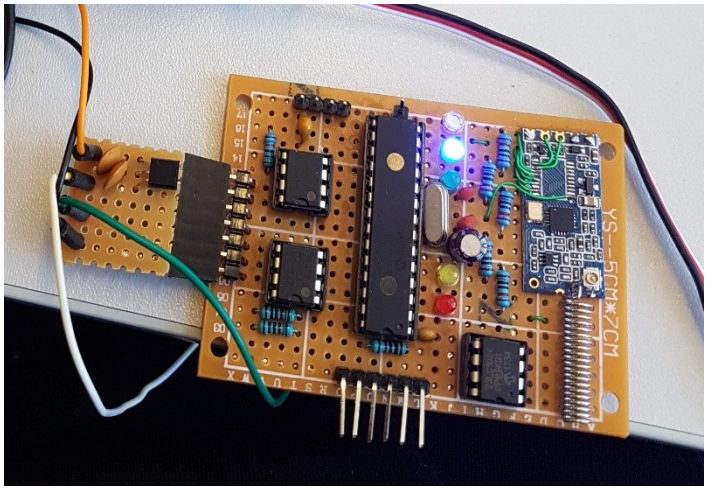
- *0x0d – kontrollsumma viga.*
- *0x0e – bootloader pole käivitatud.*
- *0x0f – bootloader on juba käivitatud.*
- *0x10 – tarkvara paigaldamine õnnestus (tegemist on erandiga, see pole viga)* [9]

Sõltuvalt veakoodist, tuleb toimida vastavalt. Kontrollsumma vea korral tuleb sama tarkvara faili rida uuesti saata. Kui *bootloader* pole käivitatud, siis tuleb saata käsk selle käivitamiseks. Juhul, kui tuleb viga, et *bootloader* on juba käivitatud, siis tuleb saata alamsüsteemile tühistamise käsk ning tarkvara uuendamist uuesti alustada. Tarkvara paigaldamise õnnestumise korral saab tarkvara üleslaadimise lugeda lõppenuks.

4.3 Teised alamsüsteemid

4.3.1 Kommunikatsiooni alamsüsteem

Kommunikatsiooni alamsüsteem tegeleb maajaamaga suhtlemisega. Alamsüsteem võtab vastu sõnumid maajaamalt ning edastab need õigetele adressaatidele ehk teistele alamsüsteemidele. Samuti vahendab kommunikatsiooni alamsüsteem sõnumeid teistelt alamsüsteemidelt maajaamale.



Joonis 9. Kommunikatsioonimooduli X-riba plaat.

4.3.2 Pardaarvuti alamsüsteem

Pardaarvuti alamsüsteemi ülesanne on monitoorida ja juhtida kogu satelliidi ja selle alamsüsteemide tööd. OBC kogub informatsiooni teistelt alamsüsteemidelt, analüüsib seda ning teeb vastavad otsused nende andmete põhjal. Kui satelliit pole maajaama vaateväljas, siis vastutab pardaarvuti kogu satelliidil toimuvate protseduuride eest. [10]

4.3.3 Asendikontrolli ja juhtimise alamsüsteem

Asendikontrolli ja juhtimise alamsüsteem tegeleb satelliidi asendi reguleerimisega. ADCS mõjutab satelliidi asendit selliselt, et päikesepaneelide kaudu oleks võimalik maksimaalselt energiat koguda. Samuti suunab ADCS satelliiti maajaama antenni suunas, kui satelliit on maajaama vaateväljas. Süsteem aitab ka kõrgema kvaliteedi piltide tegemisele kaasa, reguleerides asendit selliselt, et oleks võimalik jäädvustada parem pilt.

4.3.4 Tarkvara faili ettevalmistamine saatmiseks

COM-le, ADCS-le ja OBC-le ei saadeta tarkvara rida-rea kaupa failist nagu elektritoitesüsteemi puhul. Tarkvara saadetakse hoopis fikseeritud suurusega pakettidena, kuhu võib mahtuda mitu rida tarkvara failist.

4.3.4.1 Kaadri moodustamine

Kogu tarkvara faili sisu teisendatakse kuueteistkümnendsüsteemist kahendsüsteemi rida-rea kaupa ning lisatakse baitmassiivi. See on vajalik selleks, et andmeid saaks krüpteerida järgmises peatükis kirjeldatud algoritmiga.

4.3.4.2 Andmete krüpteerimine

Andmevahetus maajaama ja satelliidi vahel peab olema krüpteeritud. Krüpteerimine on vajalik sellepärast, et kolmandad isikud ei saaks andmeid lugeda ega neid modifitseerida. Kõik alamsüsteemid peale elektritoitesüsteemi on võimelised krüpteeritud andmeid dekrüpteerima. Elektritoitesüsteemile ei ole veel dekrüpteerimise tuge implementeeritud.

Pakettide krüpteerimiseks on kasutusel XTEA[1] sümmeetriline algoritm, mis on täiendus algsele TEA algoritmile. XTEA publitseeriti David Wheeleri ja Roger Needhami poolt 1997 aastal. Algoritmi arendamisel on pandud suurt rõhku selle kiirusele ning minimaalsele mälu kasutusele. [11]

Projektis oli krüpteerimise algoritm juba realiseeritud programmeerimiskeeles Python[2], seega oli tarvis ainult see Javasse ümber kirjutada. Tarkvara faili ettevalmistamise protsess hõlmab nelja etappi – *hash, encrypt, hash digest ja stream cryption*.

Räsimiseks kasutatakse Davies–Meyeri räsi funktsiooni. Funktsioon võtab argumendiks baitmassiivi ning tagastab kaks elementi, milles sisalduvad räsiväärtused. Räsifunktsioon

[1] https://pypi.org/project/xtea/

[2] https://www.python.org/

kasutab ka XTEA algoritmi, et räsi omakorda veel krüpteerida. Pythonist Javasse kirjutatud krüpteerimise funktsioon on välja toodud järgmisel joonisel.

```
private static Pair<BigInteger, BigInteger> encrypt(int num_rounds, BigInteger[] v, int[]
keyInt) {
    BigInteger v0 = v[0];
    BigInteger v1 = v[1];
    BigInteger[] key = new BigInteger[]{
        BigInteger.valueOf(keyInt[0]),
        BigInteger.valueOf(keyInt[1]),
        BigInteger.valueOf(keyInt[2]),
        BigInteger.valueOf(keyInt[3])};
    BigInteger sum = new BigInteger("0");
    BigInteger delta = new BigInteger("2654435769");
    while (num_rounds != 0) {
      v0 = (v0
          .add(((v1.shiftLeft(4)
              .xor(v1.shiftRight(5)))
              .add(v1))
              .xor(sum
                  .add(key[sum
                      .and(new BigInteger("3")).intValue()])))
          .and(new BigInteger("4294967295"));

      sum = (sum
          .add(delta))
          .and(new BigInteger("4294967295"));

      v1 = (v1
          .add((((v0.shiftLeft(4)
              .xor(v0.shiftRight(5))
              .add(v0)))
              .xor(sum
                  .add(key[sum.shiftRight(11)
                      .and(new BigInteger("3")).intValue()])))))
          .and(new BigInteger("4294967295"));

      num_rounds--;
    }
    return new Pair<>(v0, v1);
```

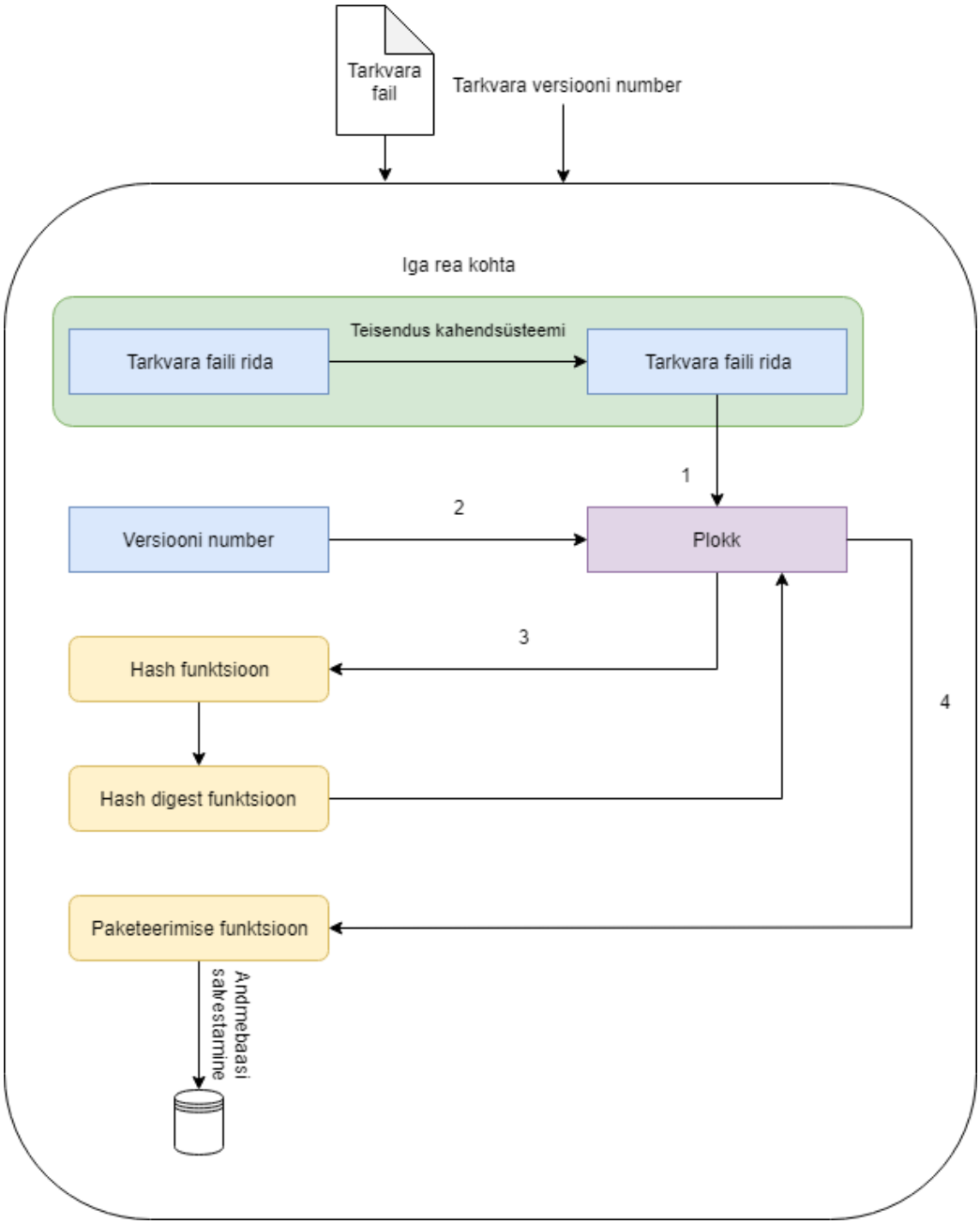
Joonis 10. XTEA krüpteerimise funktsioon

Hash digest funktsiooni ülesanne on räsi funktsioonist saadud 2 elementi konverteerida üheks 8 baidiliseks baitmassiiviks. See baitmassiivist räsiplokk, mis on arvutatud kogu tarkvara faili sisu pealt, lisatakse otsa esialgsele kogu tarkvara faili sisu sisaldavale baitmassiivile. Räsi lisamine plokile tagab kogu ploki tervikluse.

Stream crypto funktsioon koostöös *encrypt* funktsiooniga krüpteerib kogu sisseantava baitmassiivi. Sisendiks on suvaline 32 bitti nonss, võti ning plokk baitmassiivi kujul. Väljundiks on krüpteeritud baitmassiiv. Funktsiooni kasutatakse paketeerimisel.

4.3.4.3 Tarkvara ridade paketeerimine

Tarkvara read on tarvis paketeerida, sest kogu tarkvara ei saa ühes tükis üles saata. Paketid moodustatakse 192 baidi suurused, mis sisaldavad krüpteeritud tarkvara faili andmeid. Paketid teisendatakse kuueteistkümnendsüsteemi, et põhimoodulis need HDLC kaadrile lisada. Järgneval joonisel on näidatud ettevalmistamise protsess tarkvara faili saatmiseks.



Joonis 11. Tarkvara ettevalmistus saatmiseks.

4.3.5 Käsukoodid

COM, ADCS ja OBC uuendamiseks on 3 käsukoodi. Need kolm alamsüsteemi kasutavad samu koode. Koodid on järgnevad:

- 0x800 – mälu vabastamise käsk pakettide jaoks.
- 0x801 – krüpteeritud paketi saatmine.
- 0x803 – tarkvara paigaldamise alustamise käsk.

Käsu täitmise õnnestumise korral vastab alamsüsteem sama käsu koodiga, kuid ilma andmeteta. Erand on viimase käsuga, mille õnnestumise puhul alamsüsteem ei vasta midagi.

4.3.6 Veakoodid

Käsu täitmine võib ebaõnnestuda, kui alamsüsteem peab selle käsu täitmist võimatuks. Sellisel juhul vastab alamsüsteem veakoodiga, mille põhjal saab otsustada, mida edasi teha. Veakoodid on järgnevad:

- 0x22 – ploki räsi on väär, uuendamine on tühistatud.
- 0x23 – tarkvara paigaldamine ebaõnnestus.
- 0x24 – uue tarkvara versioon on madalam või samaväärne hetke versiooniga.
- 0x25 – mälu vabastamine tarkvara ridade jaoks ebaõnnestus.

5 Realisatsioon

5.1 Andmebaasi tabelid

Tarkvara üleslaadimisega seonduvate andmete hoidmiseks on kasutusel 2 andmebaasi tabelit. Nendeks on *software_upload* ja *software_upload_packet*.

Tabel *software_upload* oli juba varasemalt tehtud Taavi Suti poolt elektritoitesüsteemi tarkvara info hoidmiseks. Bakalaureusetöös on tabelit kirjeldatud järgnevalt: *Tabelis olev rida iseloomustab ühte üleslaadimise protsessi. Andmebaasi salvestatakse üleslaadimise protsessi unikaalne id, tarkvarafaili versiooni nimi, algus ja lõpp kuupäev koos kellaajaga, üleslaadimise staatus, viimasena saadetud rea number, tarkvarafailis olevate ridade arv ja tarkvara faili sisu* [7]. Tabel kirjeldab nüüd mitmeid üleslaadimise protsesse. Tabelile lisandusid väljad *satellite_id* ja *satellide_subsystem_id*, mis vastavalt viitavad satelliidile ja selle alamsüsteemile.

Tabelis *software_upload_packet* on üleslaadimiseks ettevalmistatud tarkvara faili read. Tabelis on rea identifikaator, rea number, rea uuesti saatmise järele jäänud katsete arv, tarkvara üleslaadimise identifikaator, andmerida, rea saadetuse staatus tõeväärtusena. Järgneval joonisel on näide tarkvara pakettide tabelis sisut.

	id [PK] integer	row integer	retries_left integer	software_upload_id uuid	data text	is_sent boolean
1	105595	0	3	1ded4825-be9a-4fff-9bc0-...	020000040000FA	true
2	105597	2	3	1ded4825-be9a-4fff-9bc0-...	020004000000FA	true
3	105598	3	3	1ded4825-be9a-4fff-9bc0-...	040008004AEF00F0CB	true
4	105599	4	3	1ded4825-be9a-4fff-9bc0-...	08001800176ED8381...	true
5	105600	5	3	1ded4825-be9a-4fff-9bc0-...	100020001FF00001E9...	true

Joonis 12. Tarkvara ridade metaandmete tabel.

5.2 Tarkvara failide saamine

Tarkvara failid asuvad TTÜ GitLab-i repositooriumis. Failide kättesaamiseks on kasutusel GitLab4J-API, mis võimaldab repositoorimist faile küsida. Selleks, et API toimiks, on tarvis see seadistada. API-l on vaja teada GitLabi aadressi, privaatset tõendit ning projekti identifikaatorit. [7, pp. 21-22]

Konkreetse faili kättesaamiseks tuleb GitLab API-le anda sisendiks alamsüsteemi ja faili nimi. Nende andmete põhjal pannakse kogu faili kataloogi aadress kokku ning küsitakse vastav fail.

5.3 REST teenus

REST teenuseks nimetatakse teenust, mis vahendab kliendirakendusele serverirakendusest küsitud ressursse. Näiteks, kui kasutaja soovib näha maajaama ja satelliidivahelisi saadetud sõnumeid, siis kliendirakendus teeb selleks pöördumise kindlale aadressile serverirakenduses. Serverirakendus pärib andmed andmebaasist, vormistab need õigele kujule ja saadab kliendirakendusele.

REST teenuses kirjeldatakse igat otspunkti aadressi annotatsiooniga. Kui otspunkt peab andmeid andmebaasi salvestama, siis lisatakse annotatsioon @PostMapping. Lihtsalt andmete küsimisel korral tähistatakse otspunkt @GetMapping annotatsiooniga. Kustutamise korral @DeleteMapping tähistusega.

5.3.1 Päringud

Järgnevalt on esitatud REST teenuse päringud, mida kasutatakse tarkvara üleslaadimisega seotud protseduuride täitmiseks.

Tabel 2. REST teenuse päringud.

Päringu aadress	Meetod	Tegevus
api/software-files	GET	Tagastatakse tarkvara failide loend.
api/software-upload/completed	GET	Tagastastatakse tõeväärtus tõene, kui üleslaadimine on edukalt lõpetatud.
api/software-upload/versions	GET	Tagastatakse käimasolevad, pooleliolevad, tühistatud ja edukalt lõppenud üleslaadimised.
api/software-upload/progress	GET	Tagastatakse aktiivse üleslaadimise saadetud ja kogu ridade arvu.
api/software-upload/active-upload	GET	Tagastatakse aktiivse üleslaadimise.
api/software-upload/start	POST	Alustatakse tarkvara üleslaadimise protsessi.
api/software-upload/pause	POST	Peatatakse üleslaadimine.
api/software-upload/resume	POST	Jätkatakse kasutaja valitud üleslaadimist.
api/software-upload/cancel	POST	Tühistatakse üleslaadimise.
api/software-upload/delete	DELETE	Kustutatakse üks rida üleslaadimiste infotabelist.

5.4 Tarkvara uuendamise töövoog

5.4.1 Üleslaadimise alustamine

Üleslaadimine algab tarkvara üleslaadimise lehelt kasutajaliideses. Kasutaja peab valima kolmest rippmenüüst satelliidi, alamsüsteemi ning tarkvara faili. Kõik valikud peavad olema valitud, sest vastasel korral ei ole võimalik üleslaadimist alustada. *Start uploading* nupust algab üleslaadimine. Lisades nr 2-5 on näidatud valikute lahtrid ning nende sisu. Nupu vajutamisel saadab kliendirakendus üleslaadimise alustamise aadressile päringu REST teenusse.

Teenuses tehakse nüüd kaks operatsiooni. Esiteks vormistatakse üleslaadimise käsk, mis *McsModulesToCore* klassina edastatakse põhimoodulile. Teiseks küsitakse GitLab4J API kaudu kasutajaliideses valitud tarkvara fail GitLab repositooriumist. Tarkvara read valmistatakse üleslaadimiseks ette ning salvestatakse andmebaasi. Järgneval joonisel on näide elektritoitesüsteemi tarkvara faili sisust.

```
:020000040000FA
:0400000069EF00F0B4
:020004000000FA
:040008004AEF00F0CB
:08001800176ED838186EE0CF16
:100020001FF00001E9CF19F0EACF1AF0E1CF1BF081
:10003000E2CF1CF0D9CF1DF0DACF1EF0F8CFDCF004
:10004000F7CFDDF0F6CFDEF02650030B296EA1B21C
:1000500083EC05F019C0E9FF1AC0EAFF1BC0E1FFFD
:100060001CC0E2FF1DC0D9FF1EC0DAFFDCC0F8FFD4
:10007000DDC0F7FFDEC0F6FF2650FC0B266E2950D0
:10008000030B26261838E8CFD8FF173A17381FC0B9
:10009000E0FF1000206ED838216EE0CF22F0000182
```

Joonis 13. Osa tarkvara faili sisust.

Kasutajaliideses üleslaadimise lehel ilmub üleslaadimisriba, mis kuvab saadetud tarkvara ridade ja kogu tarkvara ridade arvu protsentuaalset suhet. Üleslaadimisriba oli juba varsemalt realiseeritud Taavi Suti poolt. Lisatud on arvuline näit, mis kuvab mitu rida

mitmest on saadetud. Selline progress on kasutajale kiireim tagasiside, et üleslaadimisel ridasid ikkagi saadetakse. Infotabelisse tekib rida alustatud üleslaadimise kohta. Lehele tekivad ka nupud üleslaadimise tühistamiseks ning peatamiseks.

5.4.2 Üleslaadimise protsess

Üleslaadimise ajal kasutaja ise enam midagi tegema ei pea. Kasutajaliidesest saab jälgida üleslaadimise edenemist ning vajadusel üleslaadimine tühistada või peatada. Tarkvara uuendamise äriloogika paikneb serverirakenduses. Üleslaadimise protsess on kirjeldatud elektritoitesüsteemi uuendamise näitel.

Peale käsukoodi 0x00FF (Vt Käsukoodid, lk 24) saatmist jääb serverirakendus ootama kinnitust. Alamsüsteemi vastus saabub tarkvara üleslaadimise moodulise põhimoodulist ActiveMQ kaudu. Vastuste kuulamiseks ActiveMQ-st on klass *VirutalTopicListener*, milles kuulatakse erinevaid teemasid (Vt Sõnumivahetuse komponent, lk 16). Elektritoitesüsteemi sõnumid saabuvad *VirtualTopic.eps-received* nimelisse teemasse. Sealt suunatakse sõnum edasi *EpsResponseProcessor* klassi, milles toimub sõnumite töötlemine. Järgneval joonisel on EPS-ilt saabuvate sõnumite kuulamise meetod.

```
@JmsListener(destination="Consumer.messageReceiver.VirtualTopic.epsreceived",
       containerFactory = "topicListenerFactory")
public void receiveEpsMessage(Object rawMsg) {
       epsResponseProcessor.processIncomingEpsMessage(rawMsg);
}
```

Joonis 14. Elektritoitesüsteemi sõnumite kuulamine ActiveMQ-st.

Kui elektritoitesüsteem vastab käsuga 0x00FF, siis võib üleslaadimisega jätkata. Järgmine samm on tarkvara ridade saatmine. Tarkvara read on juba ettevalmistatud andmebaasis, seega tuleb võtta õige rida saatmiseks. Taavi Suti bakalaureusetöös oli tehtud andmebaasi tabel, milles hoitakse tarkvara üleslaadimise infot. Esialgselt on viimase saadetud rea number -1. Bakalaureusetöös oli see põhjendatud järgmiselt: *Massiivide indeksid algavad 0ga ja seetõttu on viimasena saadetud numbri veerus väärtus -1. Võttes viimasena saadetud rea numbri, suurendades seda ühe võrra, saadakse number 0 ja selle abil küsitakse massiivist esimene tarkvarafaili rida* [7, p. 29]. Teades viimati saadetud rea numbrit, on võimalik *software_upload_packet* tabelist küsida õige rida, mis saadetakse käsukoodiga 0x0001 põhimoodulisse.

Alamsüsteemi rea edukalt kättesaamise korral vastatakse sama käsukoodiga ning sellisel juhul saadetakse järgmine tarkvara rida. Võib juhtuda, et tekib kontrollsumma viga. Kontrollsumma vea korral saadetakse tarkvara rida uuesti ning vähendatakse *software_upload_packet* tabelis selle konkreetse rea saatmise katsete arvu. Kui katsete arv taandub nulliks, siis edastab moodul automaatselt käsu üleslaadimise tühistamiseks.

5.4.3 Üleslaadimise lõpetamine

Üleslaadimine lõpeb kogu tarkvara faili kontrollsumma saatmisega ning tarkvara paigaldamisega alamsüsteemile.

Juhul, kui kõik tarkvara read on edukalt saadetud alamsüsteemile, siis arvutatakse kontrollsumma kogu tarkvara faili sisust ning saadetakse satelliidile käsukoodiga 0x0002. Alamsüsteem arvutab saadetud ridade põhjal samuti kontrollsumma ning võrdleb maajaamalt saadetud kontrollsummaga. Kui need on võrdsed, siis vastatakse maajaamale sama käsuga, vastasel korral saadetakse veakood 0x0f (Vt Veakoodid, lk 25). Vea korral saadetakse üleslaadimise tühistamise käsk alamsüsteemile.

Satelliidi kinnitus õigest kontrollsummast tähendab, et võib anda käsu tarkvara paigaldamiseks. Tarkvara paigaldamise käsk edastatakse koodiga 0x0055. Paigaldamine võtab satelliidil aega umbes 5-7 sekundit. Edukas paigaldamine võrdub sama käsukoodiga vastusega. Serverirakendusepoolne protsess on nüüd lõppenud.

Üleslaadimise õnnestumise korral ilmub kasutajaliideses teade, et tarkvara uuendamine on olnud edukas. Tarkvara üleslaadimiste infotabelit täiendatakse lõpetamise ajaga ning uuendatakse üleslaadimise staatus. Üleslaadimise leheküljel taastub algolek ning võimalik on alustada uut üleslaadimise protsessi.

5.4.4 Muud operatsioonid

5.4.4.1 Üleslaadimise tühistamine

Üleslaadimist on võimalik tühistada igal ajahetkel üleslaadimise ajal. See tähendab, et üleslaadimine ei tohi olla seisundis tühistatud, peatatud ning lõpetatud. Üleslaadimise tühistamise nupp saadab käsu üleslaadimise lõpetamiseks. Tühistatud üleslaadimisi enam jätkata ei ole võimalik.

5.4.4.2 Üleslaadimise ajutine peatamine ja jätkamine

Aktiivset tarkvara uuendamise protsessi on võimalik peatada. Uuendamise protsessi ajutine peatamine ei edasta satelliidile käsku, vaid peatab nii kliendi- kui ka serverirakenduses üleslaadimise töövoo. Ajutiselt peatatud üleslaadimist on võimalik mistahes ajal jätkata. Jätkamine käivitab töövoo täpselt sellest kohast, kus see pooleli jäi.

5.4.4.3 Üleslaadimise kustutamine infotabelist

Üleslaadimiste infotabelist on võimalik ka ridu kustutada. Ridade kustutamine võib olla vajalik, kui tabelisse on kogunenud näiteks mitmeid ebaõnnestunud üleslaadimiste katseid. Kustutamise soovi korral avaneb hüpikaken, mis küsib tegevuse kinnitust. Kustutamise kinnitamise korral kontrollitakse ka seda, et tegemist ei oleks aktiivse üleslaadimsie protsessiga. Eduka kustutamise tagajärjel eemaldatakse andmebaasist nii *software_upload* kui ka *software_upload_packet* tabelitest vastavad andmed.

6 Testimine

6.1 Ühiktestimine

Ühiktestidega on kaetud tarkvara andmete krüpteerimise äriloogika. Suur osa tarkvara üleslaadimise moodulist tegeleb vaid andmete vahendamisega ning nende edastamisega. Üleslaadimise töövoo meetodid ei tagastata väärtuseid, seega ühiktestidega selliseid meetodeid katta ei saa.

6.2 Vastuste imiteerimine

Üks väga hea meetod töövoo testimiseks oli imiteerida vastuseid põhimoodulilt. Kuna alati ei olnud võimalik prototüüpi kasutada, siis tuli põhimoodulile teha mõned modifikatsioonid. Põhimoodulilt vastuste imiteerimine oli lihtne, sest päris vastused on väga sarnaselt välja saadetud sõnumile. Reaalses tarkvara üleslaadimise protsessis satelliit vastab sõnumile sama käsukoodiga, mis alamsüsteemile saadeti. See tähendab, et vastuse imiteerimiseks tuli põhimoodulis sama sõnum üleslaadimise moodulisse tagasi saata, aga ilma andmeosata. Vastuste imiteerimise testimine võimaldab üldise töövoo vead kergesti ja efektiivselt avastada, et prototüüpi saaks testida juba ilma suuremate vigadeta.

6.3 Prototüübi testimine

Prototüübi peal testimine on väga tähtis, sest see on kõige reaalsem simulatsioon, mida on võimalik saavutada. Prototüübi simulatsiooni teostamiseks oli vaja raadioside emulaatorit. Prototüüp tuli füüsiliselt otseühendada arvutiga USB kaudu. Prototüübil testimise käigus sai uut tarkvara reaalselt alamsüstemile saata ning kogu üleslaadimise voogu algusest lõpuni läbi proovida. Alamsüsteemi peal testimine võimaldas väga hästi ka veahalduse toimimist testida.

7 Kokkuvõte

Töö eesmärgiks oli arendada uue tarkvara üleslaadimise moodul satelliidi alamsüsteemide tarkvara uuendamiseks ning kohandada olemasolevat elektritoitesüsteemi moodulit uuele protokollile vastavalt.

Töös kirjeldati missioonijuhtimisetarkvara üldist arhitektuuri, mis oli aluseks loodava mooduli arendamisele. Mooduli toimimisele oli esitatud kindlad nõuded, mis peavad olema täidetud. Nõuded olid esitatud elektritoitesüsteemi uuendamisele, teiste alamsüsteemide uuendamisele ja kasutajaliidesele. Töös on välja toodud tarkvara uuendamisega seotud alamsüsteemide vahelised erinevused, sealhulgas erinevate käsukoodide ja veakoodide selgitused. Realisatsioonis on kirjeldatud andmebaaside tabeleid, REST teenuse olemust, tarkvara failide kättesaamist ning üldist tarkvara uuendamise töövoogu. Lahenduse üldine äriloogika on kaetud ühiktestitega. Imiteerimistestimine aitas üleslaadimise sõnumivahetuse voo toimimist kontrollida ning prototüübi testimine kinnitas kogu lahenduse töötamist.

Töö tulemusena valmis moodul, mis võimaldab kõiki satelliidi alamsüsteeme uue tarkvaraga uuendada. Tarkvara uuendamise protsessi on võimalik jälgida kasutajaliidesest. Kasutajaliidese kaudu saab kasutaja üleslaadimisele rakendada ka erinevaid toiminguid. Tarkvara saatmise töövoog hõlmab pakettide saatmist väikeste osade kaupa ning satelliidi poolt saadetud veakoodidega tegelemist.

Kasutatud kirjandus

[1] Cybernetica AS, „Andmekaitse ja infoturbe leksikon,“ [Võrgumaterjal]. Available: https://akit.cyber.ee/term/13166-hdlc. [Kasutatud 7 5 2019].

[2] S. Romanov, „Kuupsateliidi missioonijuhtimistarkvara arhitektuur,“ TTÜ raamatukogu digikogu, Tallinn, 2017.

[3] C. Webster, N. Shi ja I. S. Smith, „IEEE Xplore Digital Library,“ 19 4 2012. [Võrgumaterjal]. Available: https://ieeexplore.ieee.org/document/6187329. [Kasutatud 1 5 2019].

[4] Apache, „Apache ActiveMQ,“ Apache, [Võrgumaterjal]. Available: http://activemq.apache.org/how-does-a-queue-compare-to-a-topic.html. [Kasutatud 1 5 2019].

[5] M. Wasson, „Microsoft Developer Network,“ November 2013. [Võrgumaterjal]. Available: https://msdn.microsoft.com/en-us/magazine/dn463786.aspx. [Kasutatud 1 5 2019].

[6] V. Pustynski, Communication window prediction, Tallinn, 2017.

[7] T. Sutt, „Moodul TTÜ100 tudengisatelliidi elektritoitesüsteemile uue tarkvara laadimiseks,“ Tallinna Tehnikaülikooli raamatukogu digikogu, Tallinn, 2018.

[8] W. A. Beech, D. E. Nielsen ja J. Taylor, „AX.25 Link Access Protocol,“ Juuli 1998. [Võrgumaterjal]. Available: https://www.tapr.org/pdf/AX25.2.2.pdf. [Kasutatud 06 05 2019].

[9] V. Sinivee, „EPS communication protocol and register map,“ Tallinn, 2019.

[10] M. N. Ayyaz, M. R. Suddle ja S. Zahid, „Systems Design of an Economical and General-Purpose On-Board Computer for Low-Earth-Orbit Micro-Satellites,“ IEEE Xplore, Pakistan, 2008.

[11] D. J. Wheeler ja R. M. Needham, „TEA, a Tiny Encryption Algorithm,“ Google Scholar, Berlin, 1994.

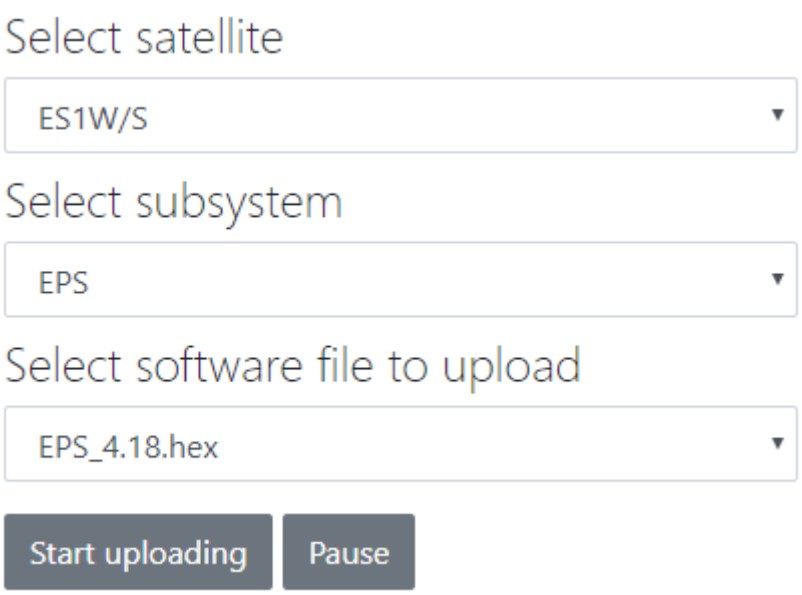
[12] D. W. A. A. Al-Saud, „Satellite Sub-systems,“ King Fahd University of Petroleum and Minerals, Dhahran, 2006.

Lisa 1 – Tarkvara uuendamise infotabel

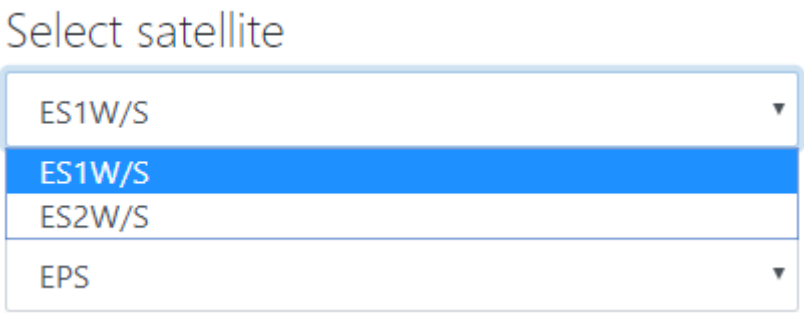
Software info

Satellite	Subsystem	Version	Started at	Finished at	Status	Action
ES1W/S	EPS	EPS_4.14.hex	19.05.2019 14:50:40		Updating	Show actions

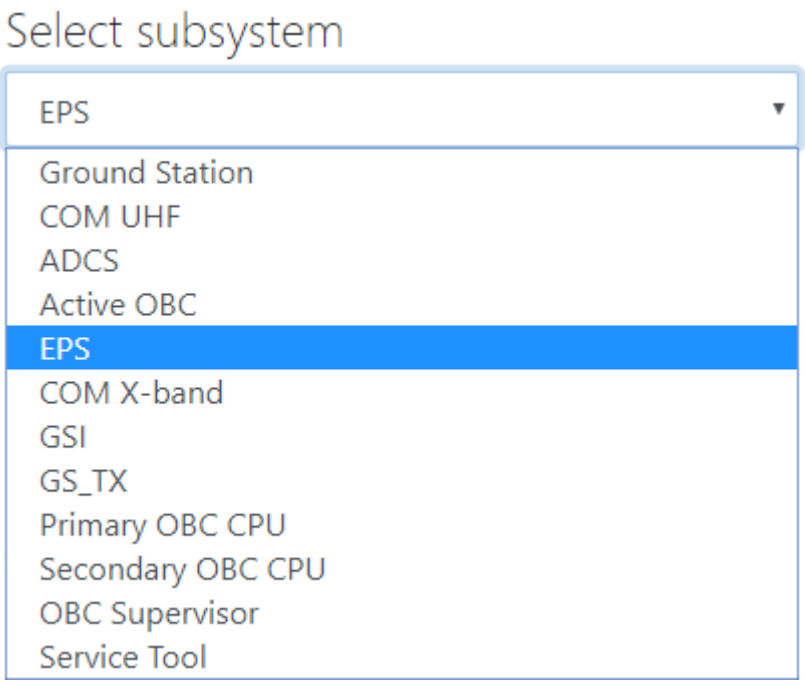
Lisa 2 – Tarkvara üleslaadimise valikute lahtrid



Lisa 3 – Satelliitide valikud



Lisa 4 – Alamsüsteemide valikud



Lisa 5 – Tarkvara failide valikud