TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Siim Raudsepp

SATELLIIDI ALAMSÜSTEEMIDELE KÄSKUDE SAATMISE/VASTUVÕTMISE IMPLEMENTEERIMINE MISSIOONIJUHTIMISE TARKVARAS

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: Siim Raudsepp

21.05.2019

Annotatsioon

Käesoleval lõputööl on 4 eesmärki.

Töö tulemusena luuakse TTÜ100 satelliidiprojekti maapealsele missioonijuhtimis tarkvarale funktsionaalsus, mis võimaldab tarkvaral genereerida satelliidile sõnum kasutaja sisestatud informatsiooni või süsteemi enda poolt koostatud informatsiooni alusel. Saadetud ja vastuvõetud sõnumite vaatamiseks luuakse missioonijuhtimis tarkvara kasutajapoolsesse rakenduses vastav funktsionaalsus.

Esimeseks eesmärgiks on luua sideprotokolli alusel HDLC kaadri andmeosa koostamise loogika.

Teine eesmärk on AX.25 kaadri genereerimise loogika sidumine sõnumi genereerimise põhivooga.

Kolmas eesmärk on luua kasutajapoolses rakenduses vaade ja loogika satelliidi alamsüsteemide käskude sisestamiseks.

Viimane eesmärk on luua kasutajapoolses rakenduses vaade ja loogika saadetud ja vastuvõetud sõnumite kuvamiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 29 leheküljel, 10 peatükki, 19 joonist, 3 tabelit.

Abstract

Command Sending/Receiving Implementation for Satellite Subsystems' in Mission Control Software

The given thesis has 4 objectives.

The outcome of the thesis is a functionality that allows The TUT100 satellite project's Mission Control Software to generate satellite's commands from information provided by front-end user or form information provided by Mission Control Software's internal procedures. A functionality will be created in Mission Control Software's front-end application for sent and received satellite's messages.

First objective is to implement a HDLC frame's data field generation functionality according to the given communication protocol.

Second objective of the thesis is to integrate AX.25 frames generation functionality to the mission control software's message generation flow.

Third objective is to create satellite's subsystems' command creation view and logic for the front-end application.

Last objective of the thesis is to create sent and received satellite's messages' catalogue view and logic for the front-end application.

The thesis is in Estonian and contains 29 pages of text, 10 chapters, 19 figures, 3 tables.

Lühendite ja mõistete sõnastik

ADCS	*Attitude Determination and Control System*, asendi määramise ja juhtimise süsteem
API	*Application Programming Interface*, programmiliides
AX.25	Amatöörraadiote kanalikihi protokoll [1, p. 1].
COM	*Communications system*, kommunikatsioonisüsteem
EPS	*Electrical Power Supply*, elektrisüsteem
HDLC	*High-Level Data Link Control*, „bitipõhise piiratud kaadriga andmeside standardprotokoll“ [2].
HTML	*Hypertext Markup Language*, hüperteksti märgistuskeel
ID	Identifikaator
ISO	*International Organization for Standardization*, Rahvusvaheline Standardiorganisatsioon
JPA	*Java Persistence API*, Java liides Java objektiga seotud andmete haldamiseks
JSON	*JavaScript Object Notation*, JavaScripti objekti notatsioon
MCS	*Mission Control Software*, missioonijuhtimis tarkvara
OBC	*On-Board Computer*, pardaarvuti
TLE	*Two-Line Element set*, “kaherealine andmeformaat objekti asukoha kirjeldamiseks kosmoses“ [3, p. 5].
TTÜ	Tallinna Tehnikaülikool
TUT	*Tallinn University of Technology*, Tallinna Tehnikaülikool

Sisukord

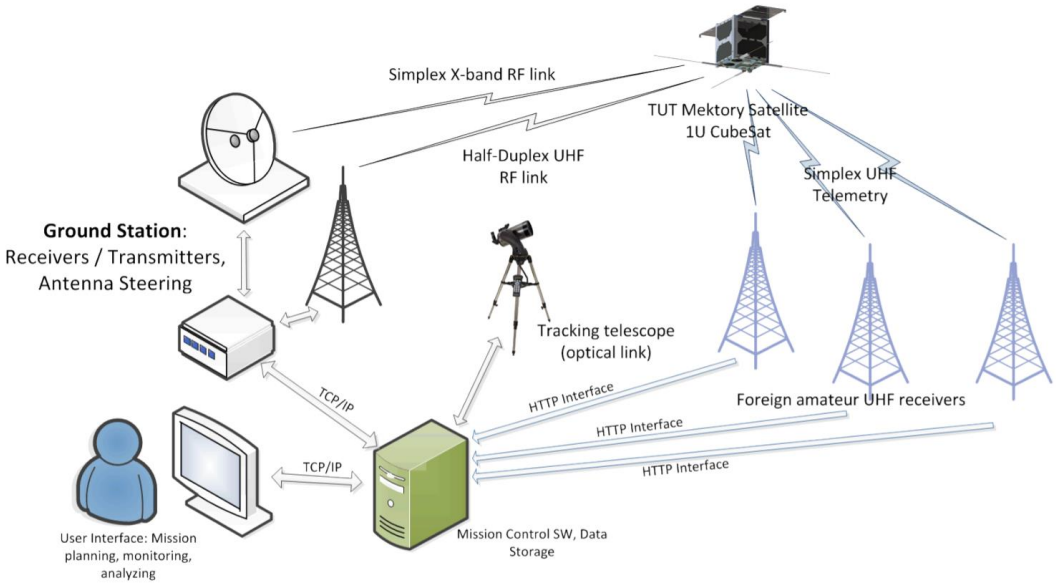
Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

TTÜ100 satelliidiprojekti infosüsteem jaguneb 2 suuremaks alamsüsteemiks – satelliit ja maapealne infosüsteem [4, p. 6]. Maapealne infosüsteem on ülesehitatud nii, et kasutaja ja satelliidi vaheline suhtlus toimuks läbi ühe tarkvara – missioonijuhtimis tarkvara.

Jooniselt 1 on nähe, et missioonijuhtimis tarkvara (edaspidi MCS ehk *Mission Control System*) on tarkvara, mis paikneb kasutaja ja satelliidiga suhtlemiseks vajaminevate süsteemide vahel. MCS tarkvara võimaldab kasutajal saata sõnumeid satelliidile ning saadetud ja vastuvõetuid sõnumeid salvestada ja töödelda [4, p. 7].



Joonis 1. TTÜ satelliidiprojekti üldine arhitektuur [4, p. 6].

Lõputöö probleem seisneb selles, et töö praktilise osa alustamise hetkel oli MCS serveripoolsel tarkvaral puudusi sõnumikaadrite töötlemise funktsionaalsusega ning kliendipoolses rakenduses esines puudusi sõnumite koostamise ning saadetud ja vastuvõetud sõnumite haldamise funktsionaalsusega.

Käesolev lõputöö on seotud MCS tarkvara edasiarendamisega nii kliendipoolses kui ka serveripoolses rakenduses.

Töös luuakse MCS tarkvarale funktsionaalsus, mis võimaldab satelliidi alamasüsteemi käske genereerida kasutajaliidese kaudu või süsteemi poolt saadud informatsiooni alusel. Sisestatud käsu järgi genereerib serveripoolne rakendus sideprotokollis ettenähtud sõnumikaadri – AX.25 kaadri. Loodud AX.25 kaadri sees hoitakse satelliidi sisemises kommunikatsioonis kasutatava kaadri andmeosa ehk HDLC kaadri andmeosa. AX.25 kaader saadetakse läbi maapealsete kommunikatsiooniseadmete satelliidile. Saadetud ja vastuvõetud sõnumid peavad olema nähtavad kasutajapoolses rakenduses ning salvestatud MCS rakenduse andmebaasi.

Antud lõputööl on 4 eesmärki:

1. Sideprotokolli paketi HDLC kaadri andmeosa koostamise loogika loomine.
2. AX.25 kaadri genereerimise loogika sidumine sõnumi genereerimise põhivooga.
3. Luua vaade ja loogika satelliidi alamsüsteemide käskude sisestamiseks kliendipoolses rakenduses.
4. Luua vaade ja loogika kliendipoolses rakenduses saadetud ja vastuvõetud sõnumite kuvamiseks.

HDLC kaadri koostamise algne loogika oli MCS tarkvaras olemas, kuid seoses protokolli muutusega tuli see ümber teha ja täiendada vastavalt uuele protokollile.

AX.25 kaadri loomise loogika oli eelnevalt implementeeritud, kuid puudus selle integreeritus MCS tarkvaraga ning sõnumi koostamise vooga. Töös seoti AX.25 kaadri loomise loogika ülejäänud MCS sõnumite genereerimise vooga.

MCS kasutajapoolses rakenduses oli olemas üksikute käskude sisestamise võimalus, kuid protokolli muutuse ning uute käskude ilmnemisega tuli varasemat loogikat ümber kohandada ja täiustada nii, et see aktsepteeriks uut sideprotokolli ning uusi käske.

Kasutajapoolses rakenduses oli olemas esialgne saadetud ja vastuvõetud sõnumite haldamise vaade. Lõputöö raames tuli see ümber muuta. Juurde tuli lisada täiendavat informatsiooni kaadrite kohta ning siduda see sõnumi genereerimise vooga.

Lõputöö peatükid 2 ja 3 keskenduvad töö teoreetilisele taustale. Kirjeldatakse TTÜ100 satelliidiprojektiga seonduvaid olulisi mõisteid ja elemente.

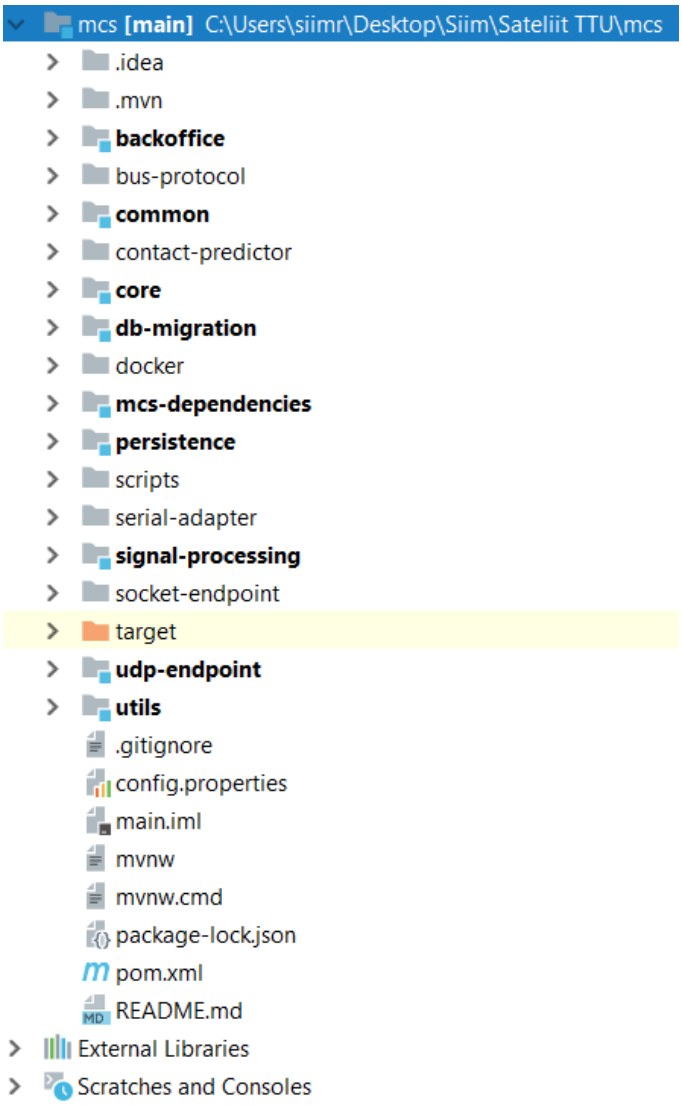
Peatükid 4 kuni 6 keskenduvad satelliidile ja selle alamsüsteemidele. Annan ülevaate kasutatavast kommunikatsiooniprotokollist ja sellest, kuidas alamsüsteemid ja nende käsud kajastuvad MCS tarkvaras.

7. ja 8. peatükis annan ülevaate töö eesmärkide saavutamiseks loodud MCS tarkvara funktsionaalsusest. Peatükkides seletan lahti nii loodud serveripoolset kui ka kasutajapoolset funktsionaalsust.

Viimane lõputöö osa ehk 9. peatükk keskendub eesmärkide saavutamiseks loodud funktsionaalsuste valideerimisele.

2 Missioonijuhtimis tarkvara

MCS tarkavara on ehitatud Maven nimelises Java projektihaldustarkvaras. Olulisemad missioonijuhtimis tarkvara moodulite nimed, kus lõputöö tarkvaraarendus toimus, on Backoffice, Common ja Core. Kõik MCS tarkvara moodulid on ülesehitatud Spring Boot nimelisel Java raamistikul. MCS moodul on kujutatud joonisel 2.



Joonis 2. Missioonijuhtimisemoodul.

MCS tarkvara moodulite vaheline suhtlus on ülesehitatud ActiveMq nimelisel tarkvaral. ActiveMq on Javal põhinev avatud lähtekoodiga sõnumivahetus tarkvara [5]. ActiveMq

tagab moodulite asünkroonse toimimise ehk kui 2 moodulit suhtlevad teineteisega läbi ActiveMq kanali, siis need moodulid ei sõltu teineteise tööst ega töökorras olemisest [3, pp. 25-26].

Backoffice moodul sisaldab endas veebiraamistikus Angular loodud kasutajapoolset rakendust ja Spring Boot raamistikus loodud serveripoolset rakendust. Backoffice kasutajapoolse rakenduse peamine ülesanne on pakkuda liidest MCS põhimooduli ehk Core mooduli töö juhtimiseks [3, p. 28].

Common moodul hõlmab endas MCS tarkvara moodulites kasutatavaid jagatuid ressursse. Lõputöö raames tegin Common mooduli satelliidi sõnumivahetuses kasutatavad HDLC andmeosa klassid ehk käsu klassid ja käsu klasside poolt kasutatavad abiklassid nagu konstantide klass käsukoodide haldamiseks. Lisaks tegin Common mooduli MCS moodulite vahelise suhtluse hõlbustamiseks kommunikatsiooniklassid.

Core moodul on missioonijuhtimis tarkvara põhimoodul. Selles moodulis paikneb MCS mooduli põhiloogika ning sellesse moodulisse tekkis lõputöö tarkvaraarenduse käigus HDLC paketi andmeosa genereerimise loogika, sõnumite salvestamise loogika ja sõnumi koostamise põhivoo peamine segment.

Kõrvalmoodulid, millega arenduse käigus kokku puutusin, olid Db-migration, Persistence, Signal-processing ja Utlis.

Db-migration moodul sisaldab endas andmebaasiobjektide loomiseks ja muutmiseks vajaminevat loogikat. Mooduli loogika on ülesehitatud Liquibase andmebaasi teegile. Liquibase teegi eesmärk on toetada andmebaasis tehtavate muudatuste haldamist ja toestamist [6].

Persistence moodulis on Java klassid, mis hõlbustavad andmebaasiga suhtlemist. Andmebaasiga suhtlus toimub läbi JPA (*Java Persistence API*) liidese.

Signal-processing on moodul, mille tegin lõputöö arenduse käigus. Mooduli eesmärk on hoida endas AX.25 kaadri genereerimise loogikat.

Utils moodul on moodul, mis on mõeldud hoidma endas MCS moodulite jagatud funktsionaalsusi nagu 16-ndsüsteemi teisendusi ja arvutusi.

3 Satelliit ja satelliidi alamsüsteemid

TTÜ100 satelliidi programmi käigus luuakse satelliit, mille peamisteks eesmärkideks on tegeleda Maa jälgimisega orbiidilt, katsetada uut kommunikatsiooniplatvormi ning demonstreerida satelliidiga seonduvat tehnoloogiat ja selle arendamist [4, p. 6].

Antud lõputöö on seotud satelliidi 4 alamsüsteemiga:

1. Pardaarvuti alamsüsteem ehk OBC (*On-board computer*)
2. Elektrisüsteem ehk EPS (*Electric power supply*)
3. Asendi määramise ja juhtimise alamsüsteem ehk ADCS (*Attitude determination and control system*)
4. Kommunikatsioonisüsteem ehk COM (*Communications*)

Kuna satelliidil puudub päikselt tuleva kiirguse eest kaitse, siis pikema satelliidi tööaja ja parema töökindluse tagamiseks on satelliidile lisatud 2 pardaarvutit.

Kommunikatsioonisüsteeme on satelliidil kaks. Üks neist on mõeldud suhtlemiseks 435 MHz sagedusel ning teine toetab suhtlust 10.5 GHz sagedusel. [7]

Eelmised 2 lõiku kokkuvõttes tähendab see seda, et MCS tarkvara tuli arendada selliselt, et loodav funktsionaalsus teeks vahet mõlemal kommunikatsioonisüsteemil kui ka mõlemal pardaarvutisüsteemil.

4 Kommunikatsiooniprotokoll

TTÜ satelliidi sideprotokolli on kirjeldanud Rain Adelbert oma 2016. aasta dokumendis. Lähtudes dokumendist kirjeldan protokolli kahes segmendis: satelliidi ja maapealsete kommunikatsiooniseadmete segment ning satelliidi segment. [4, pp. 6-7]

Satelliidi ja kommunikatsiooniseadmete vaheline osa tugineb AX.25 amatöörraadioside protokollil ning satelliidi osa tugineb ISO 13239 standardis kirjeldatud HDLC (*High-Level Data Link Control*) protokollil [4, pp. 17, 20].

Järgnevalt kirjeldan kahes eraldi alampeatükis kumbagi sideprotokolli segmenti lähtudes sellest, kuidas vastav protokolli osa kajastub MCS tarkvaras. Lisaks annan ülevaate vastava segmendi kommunikatsioonikaadri struktuurist.

4.1 Satelliidi ja maapealsete kommunikatsiooniseadmete segment

Nagu eelpool mainitud, siis sideprotokollis satelliidi ja maapealsete kommunikatsiooniseadmete segment kasutab andmete vahetamiseks AX.25 kaadrit. Satelliidi projektis tegeleb AX.25 kaadri genereerimisega missioonijuhtimis tarkvara ning satelliidi tarkvara.

AX.25 kaadriga seotud loogika oli suuremosas olemas. Olemas oli nii kaadri genereerimise loogika kui ka kaadri biti toppimise (*bit stuffing*) loogika. Puudu oli ainult sobiv integreeritus MCS tarkvaraga.

Olemasoleva loogika integreerimiseks tegin MCS tarkvarasse Mavenil põhineva Java mooduli nimega Signal-processing ehk signaali töötlemine. Pilt signaali töötlemise moodulist on Lisa 5 peatükis.

Kuna signaali töötlemise moodul tuli ühendada loodava põhivooga, siis mooduli loomise järel tegin moodulile ka sõnumivahetuse funktsionaalsuse. Ühenduse jaoks tegin Java klassi nimega CoreConnection. Loodud Java klass loob ühenduse MCS tarkvara Core mooduliga läbi ActiveMq tarkvara. CoreConnection klassis on 2 sõnumivahetus

otspunkti. Üks neist on nimega „core-to-signal-processing“, kuhu tuleb Core moodulist saadetud sõnumid ja teine on nimega „signal-processing-to-core“ kuhu lähevad sisse tulnud sõnumitele vastussõnumid.

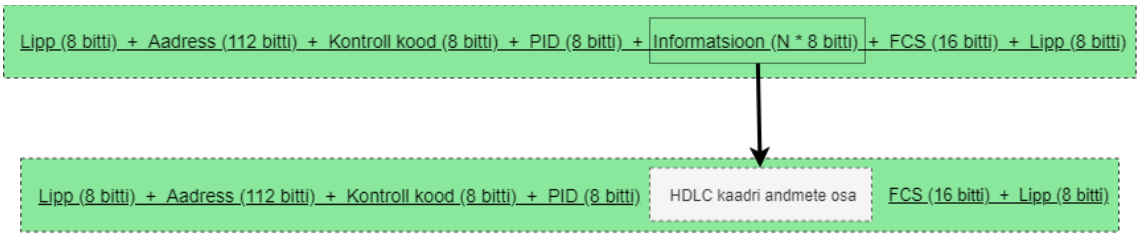
AX.25 kaadri lippe ei tohi sõnumi saatmisel esineda lippudevahelises jaos. Selle teostamiseks tuleb enne kaadri välja saatmist, mis tahes otspunktis, tegeleda kommunikatsiooniprotokollis kirjeldatud biti toppimisega. Satelliidi poolele tegeleb biti toppimisega satelliidi tarkvara. Maapealses osas tuli biti toppimise loogika lisada nii sõnumi saatmisevoole kui ka sõnumi vastuvõtmisevoole. Vastuvõtmisel tuli biti toppimine eemaldada ja saatmisel lisada.

Biti toppimise implementeerimiseks viisin olemasoleva loogika MCS Utils mooduli. Biti toppimist teostan vahetult enne kaadri saatmist lõpp-punkti ning topitudbiti eemaldamist teostab vastuvõetud sõnumile enne sõnumi salvestamist ja enne sõnumi edasi saatmist teistele protsessidele.

4.1.1 AX.25 kaadri struktuur

Kuna AX.25 paketi loomise loogika oli eelnevalt olemas, siis lõputöö raames antud paketi struktuur suurt tähelepanu ei nõudnud. Vaja oli teada ainult seda, et paketi genereerimiseks on vaja ette anda kaadri aadress ja HDLC kaadri andmeosa.

Informatsiooniväli tuleb töö jooksul loodud sõnumi genereerimise funktsionaalsusest.



Joonis 3. AX.25 kaadri struktuur.

Aadresside muutmist projekti raames pole ettenähtud. Teisisõnu tähendab see seda, et aadressid on staatilised. Kuna satelliidi projekti raames luuakse 2 satelliiti ning ühe satelliidi jaoks on 2 võimaliku aadressi kombinatsiooni, siis kahe satelliidi jaoks on projekti raames kasutusel 4 staatilist aadressi.

Aadressi 112-bitiline ehk 14-baidiline väli koosneb peamiselt 2 osast – saatja ja vastuvõtja aadress. Aadressi esimesed 7 baiti on vastuvõtjaga seotud informatsioon ja

viimased 7 baiti on saatja informatsioon. 7-baidilisest väljast esimesed 6 baiti on mõeldud amatöörraadiokutsungi väljadeks. [8, pp. 7-8]

Kui raadiokutsung on vähem kui 6 baiti, siis tuleb aadressi paremale poolele lisada juurde tühiku karaktereid kuueteistkümnendsüsteemis. Algselt oli igast 7-baidilisest väljast viimane bait mõeldud alamsüsteemi aadressiks, kuid protokolli muutuse järel otsustati, et alamsüsteemide aadresse AX.25 kaadri ei lisata ning seega viimane bait muutus staatiliseks nii vastuvõtja kui ka saatja 7-baidilises väljas.

Tabel 1. AX.25 kaadri aadressiväljad.

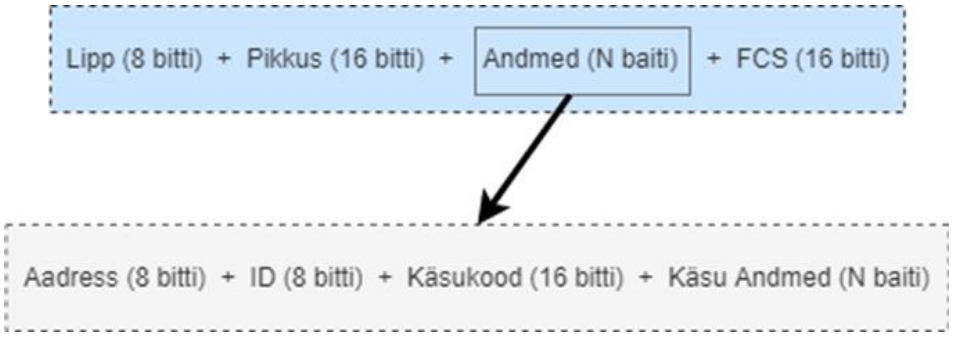
Saatja nimetus	**Vastuvõtja nimetus**	**AX.25 aadressväli 16-ndsüsteemis**
ES1WS satelliit	ES1ZW maajaam	8AA662B4AE40608AA662AEA64061
ES2WS satelliit	ES1ZW maajaam	8AA662B4AE40608AA662AEA64061
ES1ZW maajaam	ES1WS satelliit	8AA662AEA640608AA662B4AE4061
ES1ZW maajaam	ES2WS satelliit	8AA664AEA640608AA662B4AE4061

4.2 Satelliidi segment

Andmevahetuse efektiivsuse tõstmiseks saadetakse AX.25 kaadri informatsiooniosas HDLC paketi andmeosa ja mitte kogu HDLC kaader. Sellepärast luuakse antud lõputöö raames MCS tarkvaras ainult HDLC paketi andmeosa genereerimise loogika. Terviklik HDLC pakett pannakse kokku ainult satelliidis, sest terviklikul HDLC paketil tugineb satelliidi alamsüsteemide kommunikatsioon.

4.2.1 HDLC struktuur

Joonisel 4 on sinises kastis kujutatud terviklik HDLC kaader. Kuid nagu eelnevalt öeldud, siis MCS tarkvaras tuleb luua ainult kaadri andmeosa struktuur, mis on kujutatud joonisel 4 alumises hallis kastis.



Joonis 4. HDLC kaadri ja HDLC kaadri andmeosa struktuur.

HDLC andmeosa koosneb 4-st osast. Aadressivälja esimesed 4 bitti on saatja aadress ja viimased 4 bitti on vastuvõtja aadress. Kokku on satelliidiprojektis satelliidi alamsüsteemidel 8 aadressi ning maajaamal on kokku 3 aadressi ehk kokku on olulisi aadresse 11.

ID-väli omab numbrilist väärtust 0-st kuni 255-ni. Kui ID-väli saab väärtuseks 255, siis järgmine väärtus algab uuesti 0-st. Seda välja kasutatakse sõnumite järjestuseks ja paremaks käsk-vastus grupeerimiseks. ID-välja väärtuse saamiseks tegin MCS tarkvara PostgreSQL andmebaasi vastava funktsionaalsusega numbrijada generaatori.

Käsukood on 16-bitine väärtus, mis markeerib satelliidile, mis käsuga on tegu ning millise struktuuriga andmed sõnumis asuvad. Enamus käsukoode ei kattu alamsüsteemide vahel ehk üks käsukood ühes alamsüsteemis ei pruugi tähendada sama käsku teises alamsüsteemis. Käsukoodide haldamiseks tegin MCS tarkvara Common mooduli Java konstantide klassi nimega CommandCodes. See klass hoiab endas nii sõnumi saatmise käsukoodi kuueteistkümnendkuju kui ka sõnumi vastuse käsukoodi kuueteistkümnendkuju.

Käsu andmed on erinevates alamsüsteemides ja erinevates käskudes erinevad. Näiteks on nii OBC alamsüsteemil kui ka EPS alamsüsteemil mitmesse registrisse kirjutamise käsk, mis erineb selle poolest, milliseid andmeid alamsüsteemid oma HDLC pakti andmeosas vajavad.

5 Alamsüsteemide omapärad lähtudes MCS arendusest

Järgnevalt toon välja EPS ja OBC alamsüsteemide erinevused. Välja toodud erinevused mõjutavad peamiselt lõputöö tarkvaraarenduse aspekti.

Lõputöö tarkvaraarendamise osas ADCS ja COM alamsüsteemidel olulist omapära ei esinenud ning seega ma ei kirjuta nendest plaatidest pikemalt.

5.1 EPS plaadi omapärad

EPS-plaat erineb teistest plaatidest oma tarkvara poolest. EPS-i tarkvara on arendatud selliselt, et see konverteerib vastuvõetud sõnumi mitmebaidilised väljad päriesitlus (*big-endian*) stiili.

Arenduse aspektis tähendas see seda, et kui tuli genereerida lõplik EPS-i HDCL kaadri andmeosa, siis mitmebaidilised väärtused tagastasin päriesitlus stiilis.

5.2 OBC plaatide omapärad

Satelliidil on kaks OBC süsteemi. OBC-d erinevad satelliidi teistest alamsüsteemidest peamiselt selle tõttu, et võrreldes teiste alamsüsteemidega kasutavad mõlemad plaadid kahe peale kokku kolme siiniaadressi. Mõlemale OBC plaadile on mõeldud üks kindel aadress ning üks lisa aadress, mida OBC plaadid jagavad. Jagatav aadress viitab hetkel aktiivses seisundis olevale OBC plaadile.

Seega tuli MCS tarkvara arendada selliselt, et selle kaudu oleks võimalik saata kõigile kolmele aadressile käske ning vastuvõetuid OBC sõnumeid pidi oskama aadressi järgi eristada.

6 Alamsüsteemide käsud

Käsk antud lõputöö kontekstis on sõnum, mis saadetakse satelliidi alamsüsteemile. Lõputöö praktilise osa alguses olid MCS tarkvaras valmis mõned üksikud käsud. Esmalt oli vaja olemas olevaid käske muuta vastavalt uuele protokollile ning seejärel tuli missioonijuhtimis tarkvaras käske juurde luua. Lõputöö käigus implementeerisin kokku 28 käsku.

Tabelis 2 on kujutatud näidis HDLC kaadri andmeosa kuueteistkümnendsüsteemis, kus iga kaadri osa on ära grupeeritud. Näidis kaadris on käsukoodiks OBC mitmest registrist lugemisekäsk. Käsukoodist paremal on käsu andmeosa. Käsu andmete struktuur algab 8-bitisest väärtusest, mis tähistab ära mitmest registrist satelliit lugema peab. Sellele järgneb 16-bitiste aadresside jada, mis näitab satelliidile millistelt aadressidelt lugeda.

Tabel 2. HDLC andmeosa näidis kaader.

Aadress	**ID**	**Käsukood**	**8-bitine loetavate registrite arv**	**16-bitiste aadresside jada**		
03	01	0001	03	0001	0002	0003

Tabelis 3 olen toonud välja kõik OBC süsteemide käsud. Tabeli kolmandas veerus on välja toodud käsu andmete struktuur, kus erinevad struktuuri osad on eraldatud plussmärkidega. Kolmandat veergu lugedes tuleb meeles pidada, et veeru andmemahu maksimaalne suurus on 252 baiti.

Kolmandas tabelis on välja toodud käske, millel puuduvad käsu andmed. Käsu andmete puudumist on 3. tabeli kolmandas veerus tähistatud miinusmärgiga.

Ülejäänud alamsüsteemide käsud olen lisanud lõputöö lisade sektsiooni tabelitena, mis kasutavad samalaadset loogikat.

Tabel 3. OBC süsteemide käsud.

Käsu nimetus	**Käsukood 16-ndsüsteemis**	**Käsu andmete struktuur**
Ping käsk	0000	Ping sõnum
Mitmest registrist lugemise käsk	0001	8-bitine loetavate registrite arv + 16-bitiste aadresside jada
Mitmesse registrisse kirjutamise käsk	0002	8-bitine aadress-väärtus paaride arv + aadress-väärtus paaride jada
Kellaaja seadistamise käsk	0003	4-baidine unix-aeg
Kellaaja pärimise käsk	0004	-
TLE seadistamise käsk	0005	1. rida teksti + 8-bitine uue rea tähistuse sümbol + 2. rida teksti + 8-bitine uue rea tähistuse sümbol
TLE pärimise käsk	0006	-
Versiooniinfo pärimise käsk	002A	-
Süsteemikujutise kustutamise käsk	0800	-
Süsteemikujutise uuendamise käsk	0801	Krüpteeritud andmed
Süsteemikujutise koostamise käsk	0803	-

6.1 Käskude implementatsioon MCS tarkvaras

Kõik satelliidi sõnumid, mis liiguvad MCS tarkvarast edasi satelliidi poole, peavad olema konverteeritud kuueteistkümnendsüsteemi. Ehk kui MCS kasutaja poolses rakenduses muutsin kasutaja sisendi JSON kujule, siis põhimoodulis tuli see viia kuueteistkümnendkujule.

Käsu kuueteistkümnendkujule konverteerimisel lähtusin olemasolevast loogikast ehk kasutajapoolne sisend tuli esmalt siduda vastava MCS põhimoodulis oleva Command ehk käsuklassiga.

MCS tarkvaras on ettenähtud, et kõik satelliidi käskude jaoks loodud klassid peavad Command klassi laiendama. Klassi kuueteistkümnendkujule konverteerimiseks lisasin Command klassile juurde vastava meetodi.

Command klassile sai juurde lisatud meetod, mis tegeleb klassi parameetrite valideerimisega. Ehk kui luua MCS tarkvarasse uus satelliidi alamsüsteemi käsule vastav klass, siis on ettenähtud, et klassi looja peab ära defineerima parameetrite konverteerimise ja valideerimise loogika.

Käsuklass hoidis endas algselt ainult HDLC käsu andmeid. Kuid arenduse käigus selgus, et selline realisatsioon tähendaks, et igale alamsüsteemi käsule tuleb luua 2 klassi. Dubleerimise probleem tulenes sellest, et käsuklass hoidis endas algselt ainult käsu andmeid. See tähendas, et käsukood ja sihtkoha aadress lisati loodavale käsule juurde läbi HDLC andmeosaklassi. Seega kui tuli luua klass, mis kaudselt või otseselt laiendas Command klassi tuli ka luua klass, mis sammuti laiendas, kas kaudselt või otseselt, BusCommand klassi.

Dubleerimis probleemi lahendasin sellega, et Command klass hakkas endas sisaldama käsukoodi ja sihtkoha aadressi. Selline lahendus aitas elimineerida MCS tarkvarast umbes 65 klassi.

Käsuklassi konverteerimise meetod tagastab käsu klassi sisu kuueteistkümnendsüsteemis. Tagastatud sisu antakse ette HDLC andmeosa klassile, mille alusel luuakse HDLC andmeosale vastav 16-ndsüsteemikood.

HDLC klassi andmeosa ja käsuklassi klassidiagrammi joonised olen lisanud lõputöö lisadesse.

7 Loodud sõnumite töötlemise põhivoog

Sõnumi töötlemise põhivoog MCS tarkvaras arendasin selliselt, et põhivoog võib alguse saada kas kasutaja soovist saata satelliidile sõnum või mingi MCS mooduli automaatsest protsessist.

Kuna sõnumid liiguvad erinevate MCS tarkvara moodulite vahel, siis suhtluse hõlbustamiseks tegin 4 Java klassi. Nendele klassidele vastavasid JSON sõnumeid saadavad moodulid üksteise vahel andmete vahetamiseks.

7.1 Sõnumite saatmine satelliidile

Sõnumi genereerimise tegevusdiagrammil (vt. joonis 6) on kujutatud sõnumi genereerimise põhivoog, mis on saanud alguse kasutaja soovist saata satelliidile sõnum. Joonisel 6 on lisaks olulistele sõnumi genereerimise sammudele rõhutatud ka momente voos, mille alusel hiljem koostatakse joonisel 5 välja toodud AX.25 kaadri segmentidele reaalne väärtus.

Joonis 5. AX.25 kaader koos HDLC andmeosaga.

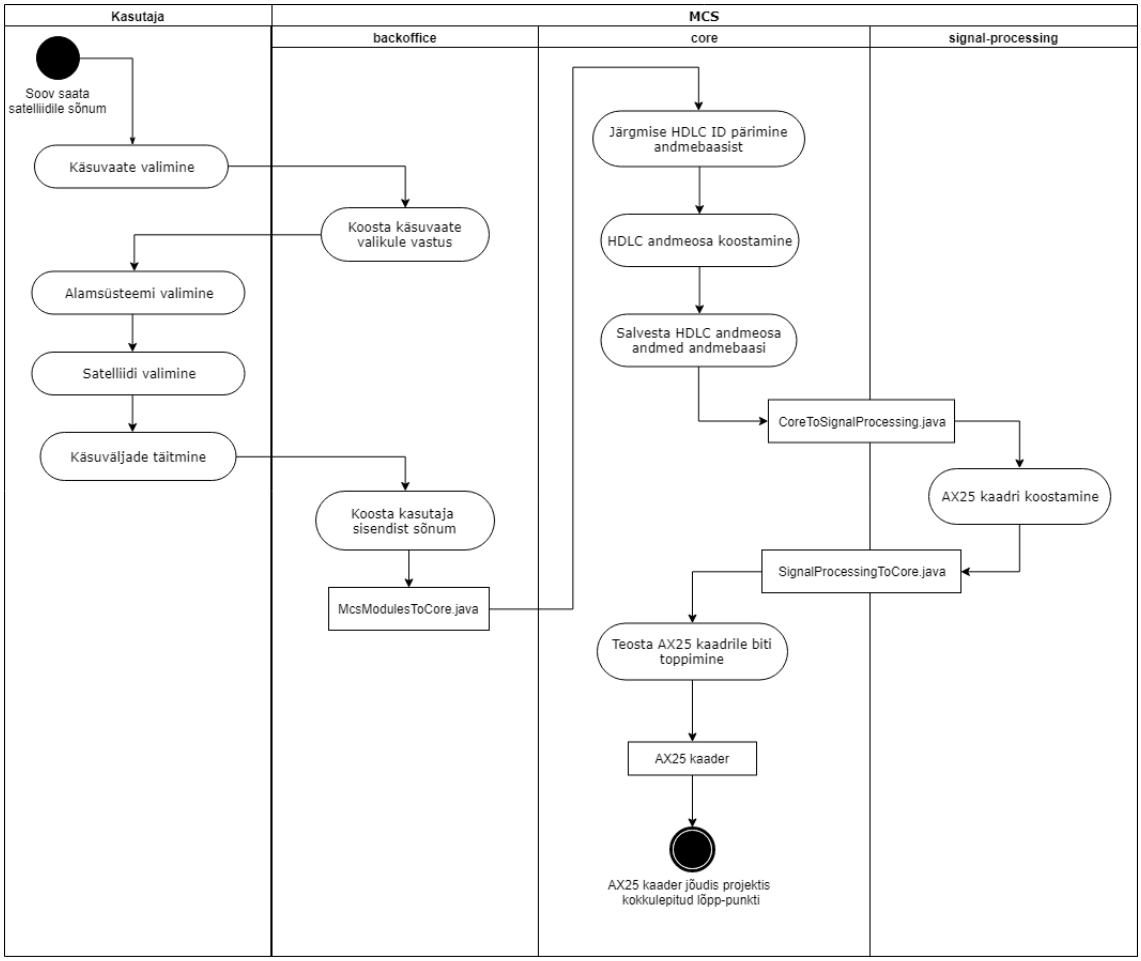
Tegevusdiagrammi joonise olen ära jaganud neljaks alampeatükiks. Esimene alampeatükk ehk sõnumi genereerimise kasutajapoolne segment saab alguse soovist saata satelliidile sõnum ja lõpeb McsModulesToCore sõnumi klassi koostamisega.

Peatüki teine osa ehk sõnumi genereerimise HDLC andmeosa segment algab sellega, et McsModulesToCore sõnum jõudis MCS tarkvara põhimooduli mooduli ning lõpeb CoreToSignalProcessing sõnumi genereerimisega.

Viiendalt jooniselt vaadatuna on kolmas alampeatükk ehk sõnumi genereerimise AX.25 kaadri segment seotud Signal-processing mooduliga.

Viimane alampeatükk ehk sõnumi genereerimise sõnumi saatmise segment algab SignalProcessingToCore sõnumist ja lõppeb tegevusdiagrammil sellega, et AX.25 kaader jõudis projektis kokkulepitud lõpp-punkti.

Igas alampeatükis annan ülevaate süsteemi või kasutaja poolt tehtavatest tegevustest. Lisaks toon välja antud sõnumi genereerimisvoo segmendiga seotud detailsemate tegevustega infovoogudega tegevusdiagrammi ja sõnumi struktuuri, mis saadetakse antud segmendi lõpus järgmisele moodulile.



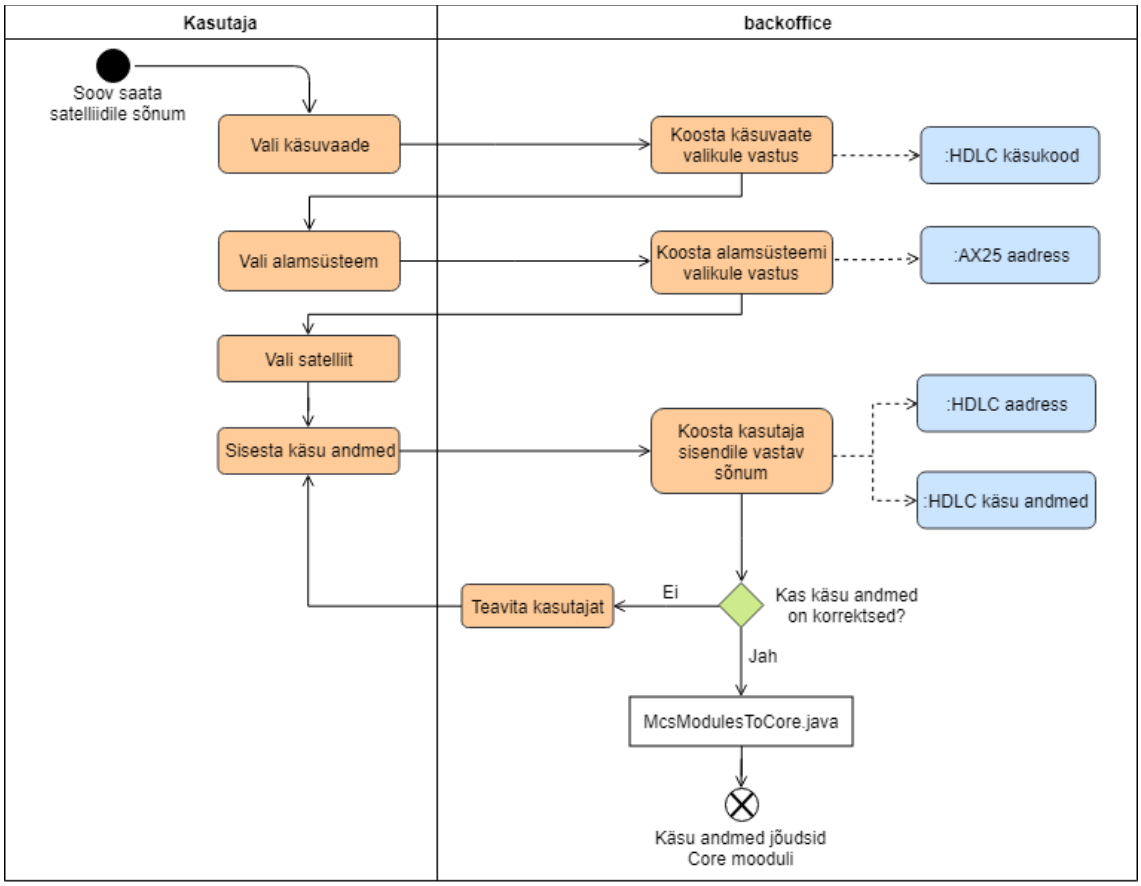
Joonis 6. Sõnumi genereerimise tegevusdiagramm.

7.1.1 Sõnumi genereerimise kasutajapoolne segment

Sõnumi genereerimine algab kasutajapoolsest rakendusest, kus kasutaja peab ära täitma vajalikud väljad selleks, et põhimoodul oskaks HDLC andmeosa käsuvälja genereerida.

Iga kasutajapoolt alguse saav sõnumi genereerimine algab käsuvaate valimisest. Tehtud valiku alusel kuvatakse kasutajale vastav käsuvaade. Satelliidi valimine ja käsu andmete

täitmine ei pea toimuma välja toodud järjekorras. Küll aga on oluline, et käsu andmete lisamisele eelneks alamsüsteemi valik. Oluline on see sellepärast, et sama käsu struktuur võib erinevates alamsüsteemides erineda ja seega erineb kasutajapoolt sisestatavad andmed.



Joonis 7. Sõnumi genereerimise kasutajapoolse segmendi infovoogudega tegevusdiagramm.

Pärast andmete sisestamist toimub kasutajapoolses rakenduses andmete valideerimine. Kui sisestatud andmed on korrektsed koostatakse McsModulesToCore klassile vastav JSON sõnum. Koostatud sõnum saadetakse edasi MCS põhimooduli ehk Core mooduli.

McsModulesToCore sõnum hoiab enda „commandParametersJson“ väljal valitud käsku JSON formaadis. „User“ väli hoiab käsu saatnud kasutajanime ning „satelliteId“ väli sisaldab vastava satelliidi numbrit, mis on sellele satelliidile määratud andmebaasi satelliiditabelis.

Sõnumi andmed salvestatakse andmebaasi erinevatesse tabelitesse sõnumi genereerimisvoo erinevates etappides. Sõnumi erinevate andmete seostamiseks

andmebaasitabelite vahel lisatakse sõnumi „messageUuid“ väljale unikaalne 128-bitine räsikood.

McsModulesToCore sõnum saadetakse esmalt Backoffice mooduli kasutajapoolsest rakendusest Backoffice serveripoolsele rakendusele. Backoffice serveripoolne rakendus saadab läbi ActiveMq kanali temani jõudnud sõnumi edasi Core mooduli. ActiveMq kanal kuhu Backoffice moodul sõnumi saadab on nimega „command-data“.

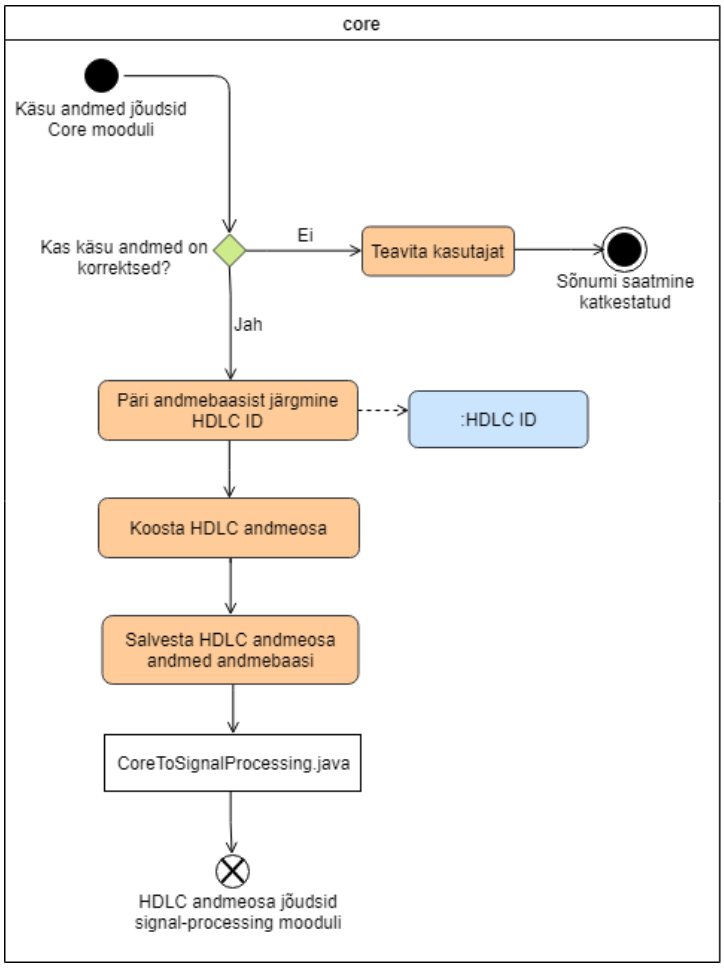
```
{
  "commandParametersJson" : "",
  "messageUuid" : "",
  "user" : "",
  "satelliteId" : ""
}
```

Joonis 8. McsModulesToCore klassi JSON kuju.

7.1.2 Sõnumi genereerimise HDLC andmeosa segment

HDLC andmeosa segment algab McsMoculesToCore klassi valideerimisega. Kui andmed on korrektsed, siis andmebaasist päritakse järgmine HDLC ID väärtus.

Seejärel koostab programm HDLC kuueteistkümnendsüsteemis koodi. Saadud HDLC kood salvestatakse andmebaasi tabelisse, kus hoitakse sõnumi kohta käivaid detailandmeid.



Joonis 9. Sõnumi genereerimise HDLC andmeosa segmendi infovoogudega tegevusdiagramm.

Pärast andmebaasis salvestamist koostatakse CoreToSignalProcessing klassile vastav JSON sõnum.

Eelnevalt määratud satelliidi ID alusel päritakse andmebaasist vastav AX.25 kaadri aadressivälja kuueteistkümnendkood ning lisatakse sõnumi klassi aadressiväljale. Seejärel lisatakse kasutajaliidesest alguse saanud sõnumi unikaalne räsikood ning genereeritud HDLC kuueteistkümnendkood lisatakse sõnumi „busCommandHex“ väljale.

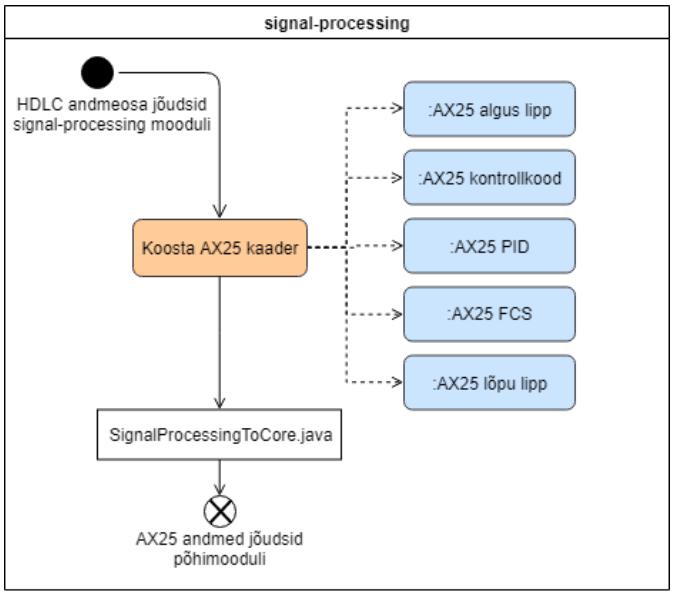
Pärast sõnumi saatmist saadetakse CoreToSignalProcessing klassile vastav JSON kuju läbi ActiveMq „core-to-signal-processing“ kanali Singal-processing mooduli.

```
{
  "messageUuid" : "",
  "satelliteAddress" : "",
  "busCommandHex" : ""
}
```

Joonis 10. CoreToSignalProcessing klassi JSON kuju.

7.1.3 Sõnumi genereerimise AX.25 kaadri segment

Lõpliku kaadri loomiseks on vaja Signal-processing moodulis olevale loogikale ette anda HDLC andmeosa ning AX.25 kaadri aadressväli.



Joonis 11. Sõnumi genereerimise AX.25 kaadri segmendi infovoogudega tegevusdiagramm.

Pärast AX.25 kaadri loomist luuakse SignalProcessingToCore klassile vastav JSON sõnum. Loodud AX.25 kaadri kuueteistkümnend kuju lisatakse „ax25Hex“ väljale ning „messageUuid“ saadakse vastuvõetud CoreToSignalProcessing sõnumist.

Loodud sõnum saadetakse läbi ActiveMq „signal-processing-to-core“ kanali tagasi MCS põhimooduli.

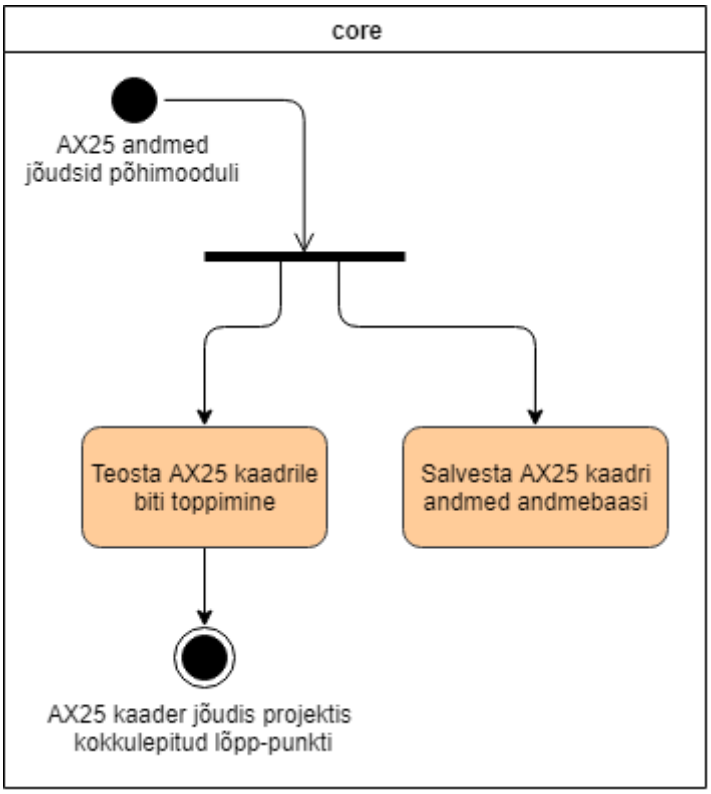
```
{
   "messageUuid" : "",
   "ax25Hex" : ""
}
```

Joonis 12. SignalProcessingToCore klassi JSON kuju.

7.1.4 Sõnumi genereerimise sõnumi saatmise segment

Pärast seda, kui AX.25 kaader on lisatud ActiveMq „signal-processing-to-core“ kanali, loetakse sõnum paralleelselt kahe protsessi poolt välja. Põhimooduli osa, mis tegeleb andmete salvestamisega, salvestab saadud kaadri andmebaasi sõnumi tabelisse.

Teine põhimooduli osa, mis tegeleb sõnumi edasise töötlemisega, teostab kaadrile eelnevalt olemas olnud loogika alusel biti toppimist. Pärast biti toppimist saadab AX.25 kaadri kuueteistkümnend kujul ActiveMq „uplink“ kanali. See kanal on kokkulepitud lõpp-punkt kuhu kaader pidi jõudma.



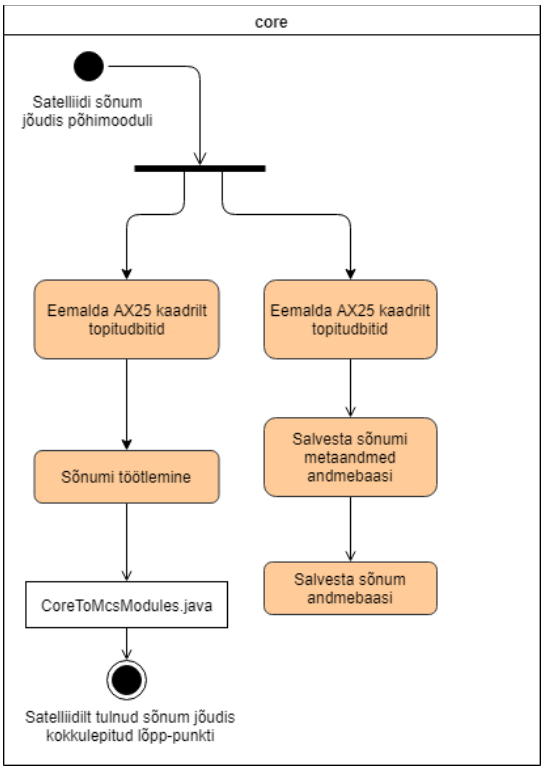
Joonis 13. Sõnumi genereerimise sõnumi saatmise segmendi tegevusdiagramm.

7.2 Satelliidi poolt saadetud sõnumid

Satelliidi poolt saadetud sõnumid saadetakse MCS väliste süsteemide poolt ActiveMq „downlink“ kanali. Kanalist loetakse sõnum paralleelselt nii sõnumi salvestamise loogika poolt kui selle töötlemise loogika poolt.

Salvestamisel eemaldatakse võimalikud topitudbitid ning seejärel salvestatakse andmebaasi nii sõnumi metaandmed kui ka sõnum ise.

Töötlemise loogika algab samamoodi topitudbittide eemaldamisest. Pärast bittide eemaldamist seotakse vastuvõetud kaader lõputöö käigus loodud vastussõnumi klassiga. Kui vastu võetakse kaader, millel puudub vastussõnumi klass, siis seotakse kaadri andmed üldise vastussõnumi klassiga.



Joonis 14. Vastuvõetud sõnumite töötlemise tegevusdiagramm.

Pärast seda, kui vastuvõetud sõnum on välja loetud, luuakse CoreToMcsModules klass, mis on mõeldud sisaldama satelliidi poolset vastust ja vastusega seotud üldisi andmeid.

Joonisel 15 kujutatud JSON sõnumi „responseMessagaJson“ väli sisaldab endas vastuvõetud sõnumi klassi JSON kuju. „CommandCode“ on mõeldud hoidma vastuvõetud sõnumi käsukoodi, „satelliteName“ sisaldab endas satelliidi nime ning „subsuystemId“ hoiab alamsüsteemi identifikaatorit, millega alamsüsteem on markeeritud andmebaasis.

```
{
   "responseMessageJson" : "",
   "commandCode" : "",
   "satelliteName" : "",
   "subsystemId" : "",
}
```

Joonis 15. CoreToMcsModules klassi JSON kuju.

Kui CoreToMcsModules klass on genereeritud, siis saadetakse sõnumi klassi JSON kuju ActiveMq kanalisse. ActiveMq kanalid on jaotatud nii, et iga alamsüsteemi vastus läheb vastava alamsüsteemi jaoks loodud kanalisse.

8 Kasutajapoolne rakendus

MCS kasutajapoolne rakendus asub Backoffice moodulis ning on loodud kasutades Angular veebiraamistikku. Angular veebiraamistik kasutab *single page application* disaini, mis tähendab, et kogu rakenduse peale on üks HTML fail. Faili HTML sisu muutub kui kasutaja navigeerib rakenduse erinevatele otspunktidele.

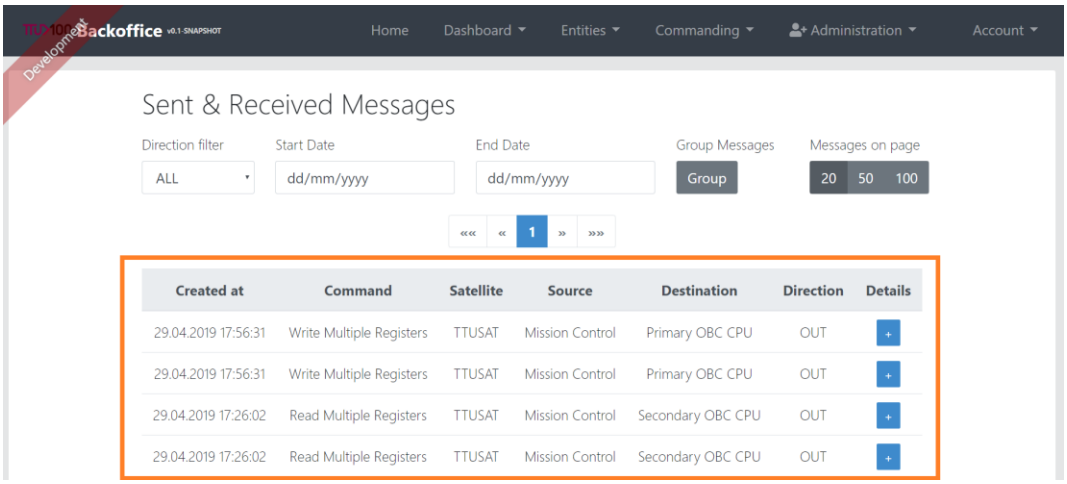
Angular raamistiku rakendused järgivad arhitektuurilist disaini, kus seotud äriloogikaga programmikood grupeeritakse mooduliteks. Moodulites jaotatakse programmikood veelkord koostöötavateks loogilisteks osadeks, mida Angular raamistikus kutsutakse komponentideks. [9]

Lõputöö raames tuli MCS tarkvara Angular veebirakenduses edasi arendada olemasolevaid funktsionaalsusi saadetud ja vastuvõetud sõnumite kuvamise moodulis ning käskude sisestamise moodulis.

8.1 Saadetud ja vastuvõetud sõnumite kuvamine

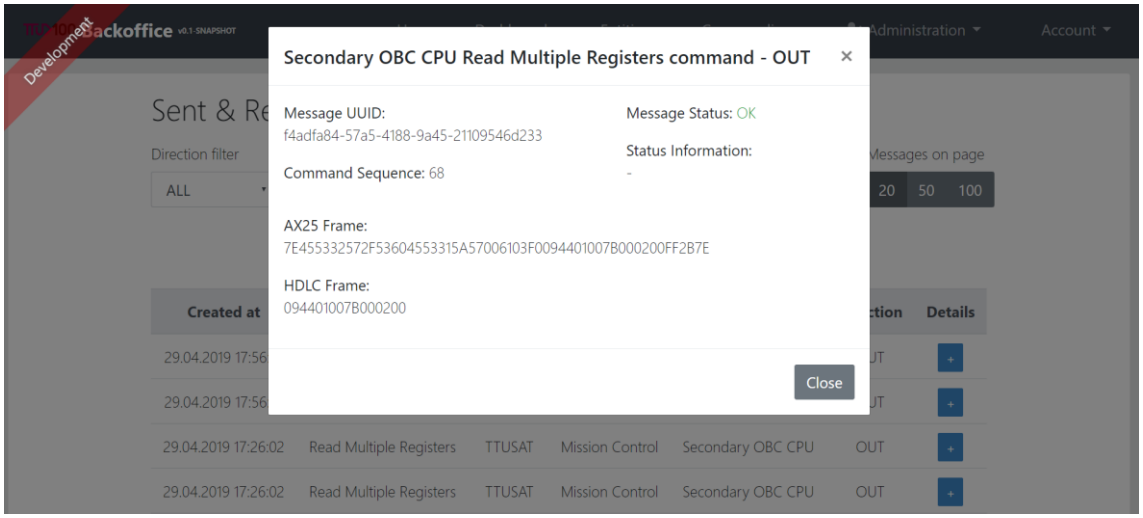
Saadetud ja vastuvõetud sõnumite kuvamise jaoks oli eelnevalt Backoffice veebirakenduse osas olemas vastav Angular moodul.

Antud moodulis tehtud muudatused on seotud joonisel 16 kujutatud oranžis kastis oleva HTML osaga. Juurde lisasin andmete kuvamiseks tabeli ning tabelile lisasin erinevaid sõnumiga seotuid üldisi andmeid.



Joonis 16. Vastuvõetud ja saadetud sõnumite vaade.

Lisaks lisasin tabeli juurde „details“ veeru, mille veerus olevale nupule vajutades kuvatakse kasutajale sõnumi detailsemad andmed. Detailsemate andmete vaade on kujutatud joonisel 17.



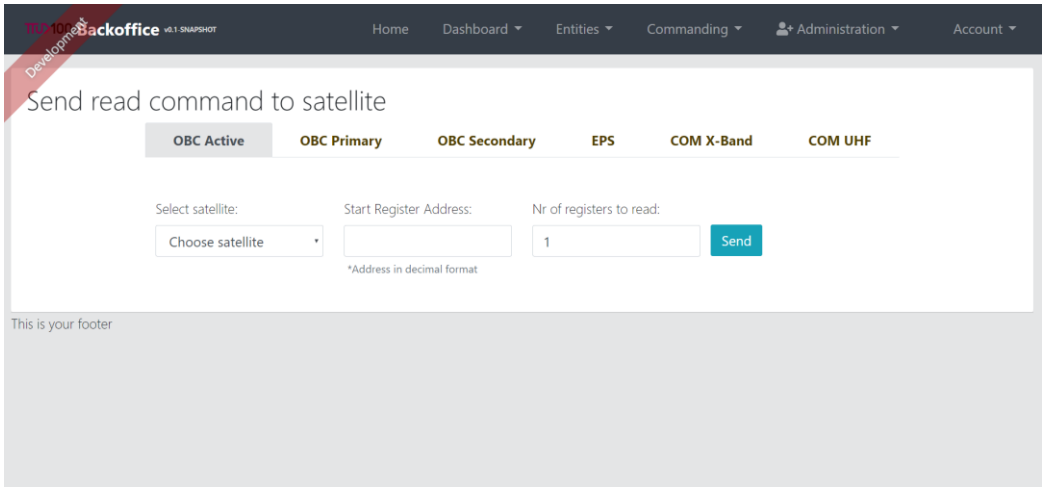
Joonis 17. Sõnumi detailandmete vaade.

8.2 Käskude sisestamine

Käskude sisestamiseks oli Backoffice veebirakenduses olemas Read-register moodul ning muude erinevate sõnumite saatmiseks mõeldud moodul.

Read-register mooduli ehk lugemismooduli HTML vaade on kujutatud joonisel 18. Lugemismooduli komponendile lisasin juurde funktsionaalsuse, mis võimaldab saata

käske erinevatele alamsüsteemidele ning erinevatele satelliitidele. HTML vaate osas tuli juurde lisada alamsüsteemide ja satelliitide valiku võimalus.



Joonis 18. Lugemissõnumi vaade.

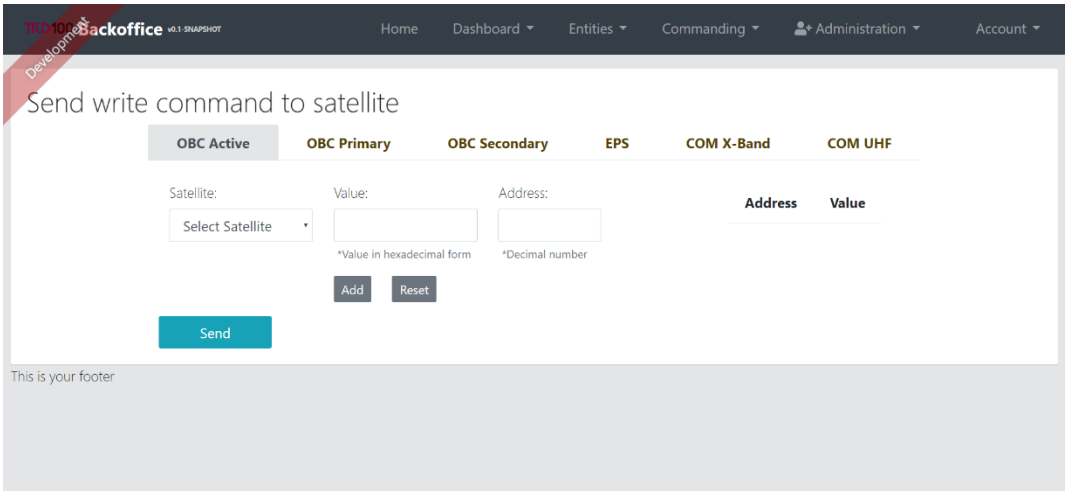
Erinevate käskude saatmise mooduli muutsin ümber nii, et antud moodul hoiaks endas lihtsamate käskude saatmise funktsionaalsust. Lihtsamad käsud on käsud, mis ei vaja keerulist valideerimist kasutajapoolses rakenduses või käsud, mis ei vaja kasutajapoolset sisendit. Näiteks võib tuua ADCS alamsüsteemi versiooniinfo pärimise käsu, mis ei vaja kasutajapoolset sisendit. Antud mooduli nimetasin ümber General-messages mooduliks ehk üldiste käskude moodul. General-messages mooduli vaade on kujutatud Lisa 8 peatükis.

Kasutajapoolses rakenduses lisasin juurde registrisse kirjutamisekäsu mooduli ehk Write-register mooduli.

Kuna satelliidi OBC alamsüsteemid kasutavad mitmesse registrisse kirjutamisekäsus erinevat käsu struktuuri, siis tuli seda mooduli loomisel arvestada. Selleks tegin mooduli 2 alamkomponenti: Write-obc ja Write-regular. Write-obc komponent ehk OBC kirjutamise komponent on mõeldud OBC plaatidele registrisse kirjutamise sõnumi saatmiseks ning Write-regular ehk üldine kirjutamise komponent on mõeldud ülejäänud alamsüsteemidele sama tüüpi sõnumi saatmiseks. Üldine kirjutamisekäsu mooduli vaade on väljatoodud Lisa 9 peatükis.

OBC plaatide kirjutamisekäsu vaate loomisel tuli arvestada sellega, et antud OBC käsus saadetakse plaadile aadress-väärtus paarid. Paremaks kasutajakogemuseks lisasin OBC

kirjutamisekäsu vaatele juurde tabeli, kuhu lisatakse kasutaja poolt lisatud aadress-väärtus paar, kui kasutaja vajutab joonisel 19 välja toodud vaates „Add“ nuppu.



Joonis 19. OBC registrisse kirjutamisekäsu vaade.

9 Tulemuste valideerimine

Lõputöö eesmärkide tulemuste valideerimiseks teostasin nii manuaalset kui ka automaatset testimist. Kasutajaliidesega seotud eesmärkide tulemisi kontrollisin manuaalsetestimisega ning HDLC kaadri genereerimisega seotud eesmärgi tulemust valideerisin nii manuaalse- kui ka automaatsetestimisega. AX.25 kaadriga seotud tulemust testisin manuaalsetetestidega.

Automaatsel testimisel kirjutasin kokku 59 ühiktesti. Teste kirjutasin nii HDLC andmeosa genereerimisele kui ka vastuvõetud sõnumitest informatsiooni lugemise loogikale.

Kasutajaliidese ja sõnumi genereerimise manuaalset testimist võib pidada ühtseks protsessiks, sest kasutajaliidesest sisestatud sõnumid liikusid sõnumi genereerimisvoose ning sealt edasi sai genereerimise tulemusi näha kasutajaliidese saadetud ja vastuvõetud sõnumite vaadest.

Lisaks sai manuaalsel testimisel kasutada satelliidi alamsüsteemide prototüüpe. Prototüüpide testimisel sai testitud nii EPS kui ka COM alamsüsteemi. Pilt EPS-i ja COM-i alamsüsteemidest on peatükis Lisa 4.

COM ja EPS testimise üks eelis, võrreldes COM ja OBC testimisega, on see, et EPS on ainus alamsüsteem, mis vajab HDLC andmeosas mitmebaidilisi väärtusi päriesitlus stiilis. COM ja EPS prototüüpide testimine võimaldas kontrollida nii päriesitlus kui ka pöördesitus (*little-endian*) stiilis sõnumite genereerimist.

10 Kokkuvõte

Lõputöö eesmärkideks oli luua loogika HDLC paketi andmeosa koostamiseks, AX.25 kaadri genereerimis loogika sidumine sõnumi genereerimise põhivooga ning kasutajapoolses rakenduses luua vaade ja loogika satelliidi alamsüsteemide käskude sisestamiseks ning saadetud ja vastusõetud sõnumite kuvamiseks.

Lõputöö kõik 4 eesmärki said täidetud.

Esimese eesmärgi jaoks implementeerisin satelliidiprojekti sideprotokolli alusel MCS tarkvara põhimooduli HDLC andmeosa koostamise loogika.

Teine eesmärk sai täidetud kui AX.25 kaadri genereerimise loogika jaoks tegin MCS tarkvarasse eraldi mooduli, mille integreerisin sõnumi genereerimise põhivooga.

Kolmanda eesmärgi jaoks loodi vastuvõetud ja saadetud sõnumite kuvamiseks kasutajapoolses rakenduses üldiste ja detailsemate andmete vaated.

Viimase eesmärgi saavutamiseks koostasin erinevate alamsüsteemi käskude jaoks kasutajapoolses rakenduses erinevaid veebilehe vaateid koos vastava käsu kasutajapoolse valideerimisega.

Lõputöö raames sai TTÜ100 satelliidiprojekti missioonijuhtimis tarkvaras valmis funktsionaalsus, mis võimaldab kasutajal kasutajaliideses saata käske satelliidile ning näha saadetud ja vastuvõetud sõnumeid. Veel sai loodud funktsionaalsus MCS tarkvarale, mis võimaldab sellel kas kasutajaliidesest või süsteemi poolt sisestatud andmete alusel genereerida sideprotokollis ettenähtud kaader. Genereeritud kaader saadetakse edasi süsteemidele, mis tegelevad kaadri toimetamisega satelliidile.

Kasutatud kirjandus

[1] W. A. Beech, D. E. Nielsen ja J. Taylor, „AX.25 Amateur Packet-Radio Link-Layer Protocol,“ Tucson Amateur Packet Radio Corp, Juuli 1998. [Võrgumaterjal]. Available: https://www.tapr.org/pub_ax25.html. [Kasutatud 17 Mai 2019].

[2] „HDLC,“ Cybernetica AS, [Võrgumaterjal]. Available: https://akit.cyber.ee/term/13166-hdlc. [Kasutatud 17 Mai 2019].

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

[4] R. Adelbert, „TUT-MEKTORY NANOSATELLITE System Requirements Specification,“ Tallinn, 2016.

[5] „ActiveMq Homepage,“ The Apache Software Foundation, [Võrgumaterjal]. Available: https://activemq.apache.org/. [Kasutatud 2 Mai 2019].

[6] „Liquibase home page,“ Datical, [Võrgumaterjal]. Available: http://www.liquibase.org/index.html. [Kasutatud 18 Mai 2019].

[7] „TTÜ100 Satelliidi tutvustus,“ Tallinna Tehnikaülikool, [Võrgumaterjal]. Available: https://www.ttu.ee/projektid/mektory-est/satelliidiprogramm-4/. [Kasutatud 7 Mai 2019].

[8] F. George, S. Billeter ja M. Richard, „AX.25 Telemetry and Telecommand Transfer Frames Format,“ Lausanne, 2013.

[9] „Angular: Architecture overview,“ Google Inc., [Võrgumaterjal]. Available: https://angular.io/guide/architecture. [Kasutatud 3 Mai 2019].

Lisa 1 – EPS alamsüsteemi käsud

Nimetus	**Käsukood 16-ndsüsteemis**	**Käsu andmete struktuur**
Ping käsk	0000	Ping sõnum
Mitmest registrist lugemise käsk	0003	16-bitine lugemise algus registri aadress + 16-bitine loetavate registrite arv
Ühte registrisse kirjutamise käsk	0006	16-bitine registri aadress + 16-bitine registri väärtus
Mitmesse registrisse kirjutamise käsk	0010	16-bitine kirjutamise algus registri aadress + 16-bitine kirjutatavate registrite arv + 16-bitiste registri väärtuste jada

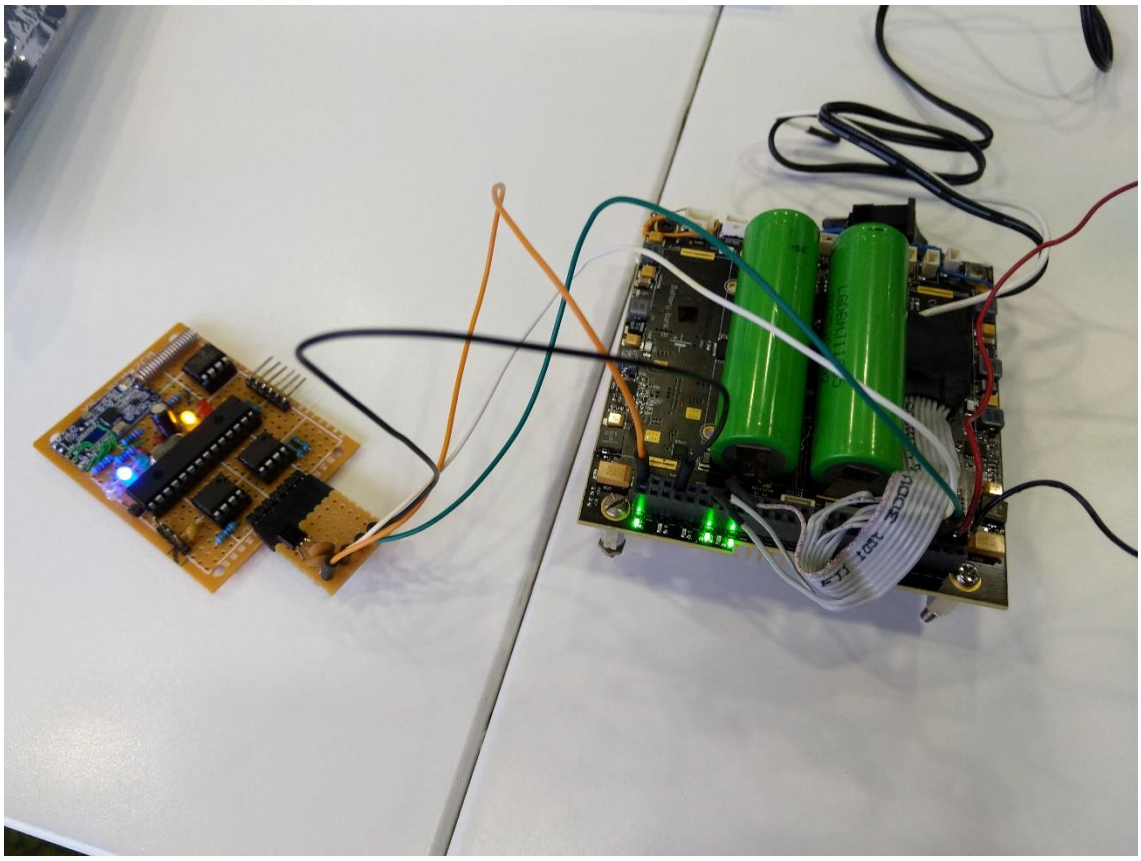
Lisa 2 – COM alamsüsteemide käsud

Nimetus	**Käsukood 16-ndsüsteemis**	**Käsu andmete struktuur**
Ping käsk	0000	Ping sõnum
Mitmest registrist lugemise käsk	0003	16-bitine lugemise algus registri aadress + 16-bitine loetavate registrite arv
Ühte registrisse kirjutamise käsk	0006	16-bitine registri aadress + 16-bitine registri väärtus
Mitmesse registrisse kirjutamise käsk	0010	16-bitine kirjutamise algus registri aadress + 16-bitine kirjutatavate registrite arv + 16-bitiste registri väärtuste jada
Süsteemikujutise kustutamise käsk	0800	-
Süsteemikujutise uuendamise käsk	0801	Krüpteeritud andmed
Süsteemikujutise koostamise käsk	0803	-

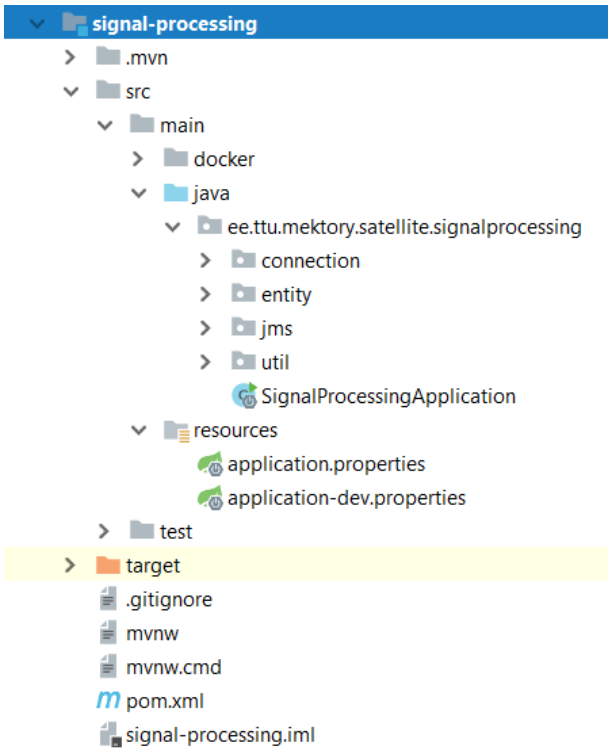
Lisa 3 – ADCS alamsüsteemi käsud

Nimetus	**Käsukood 16-ndsüsteemis**	**Käsu andmete struktuur**
Ping käsk	0000	Ping sõnum
Versiooniinfo pärimise käsk	002A	-
Püsivarainfo pärimise käsk	002B	-
Süsteemikujutise kustutamise käsk	0800	-
Süsteemikujutise uuendamise käsk	0801	Krüpteeritud andmed
Süsteemikujutise koostamise käsk	0803	-

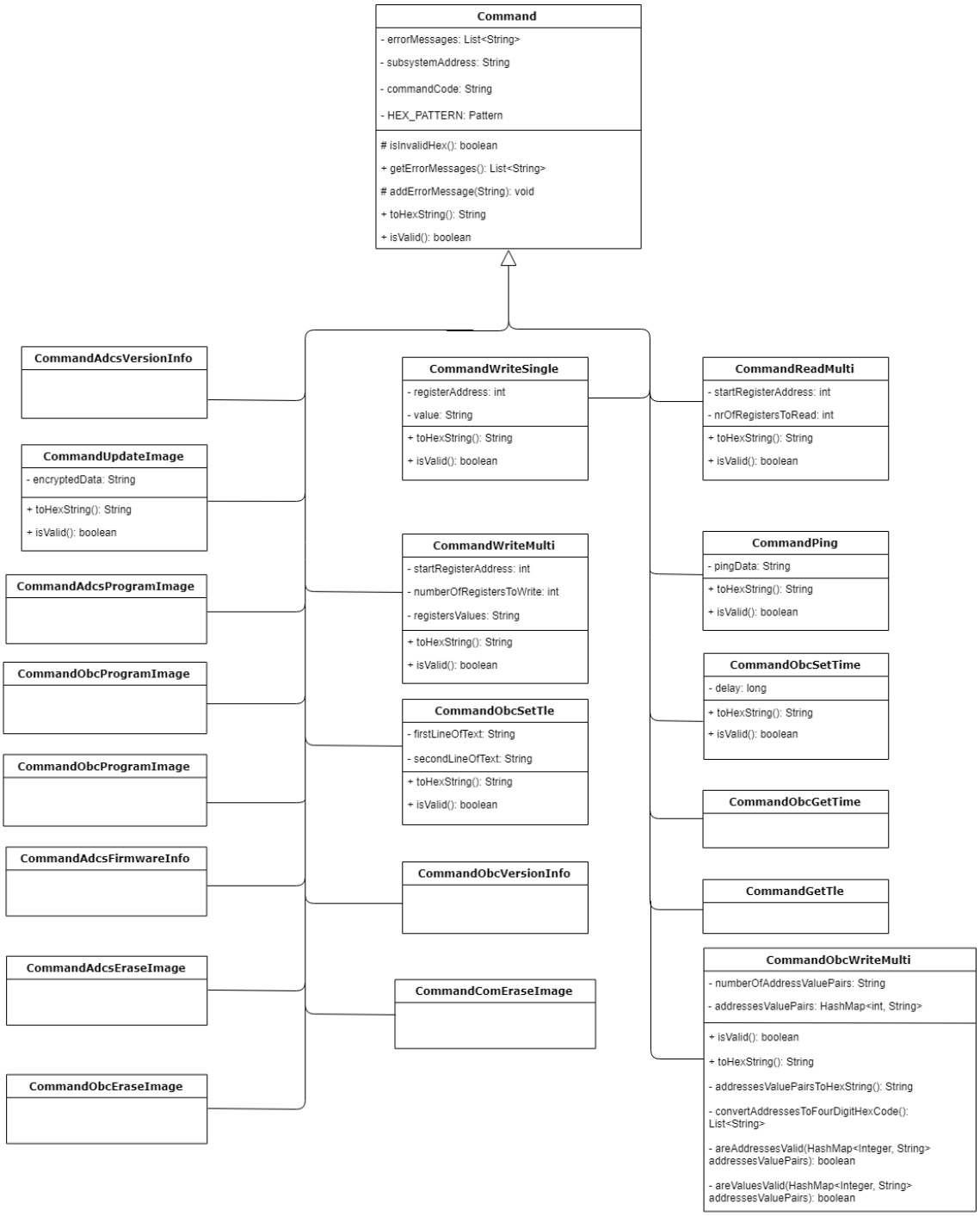
Lisa 4 – COM ja EPS alamsüsteemid



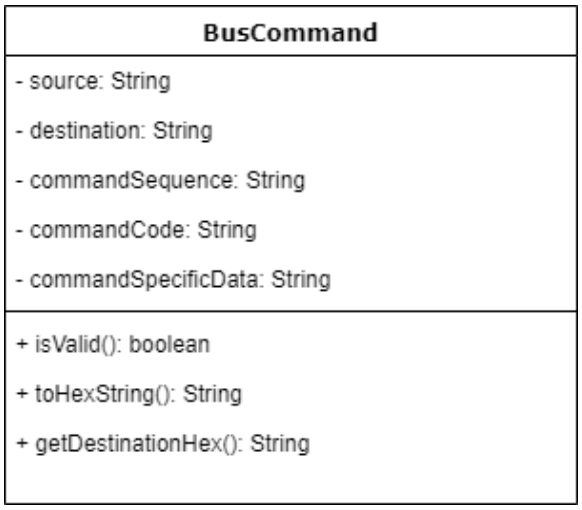
Lisa 5 – Signaali töötlemise moodul



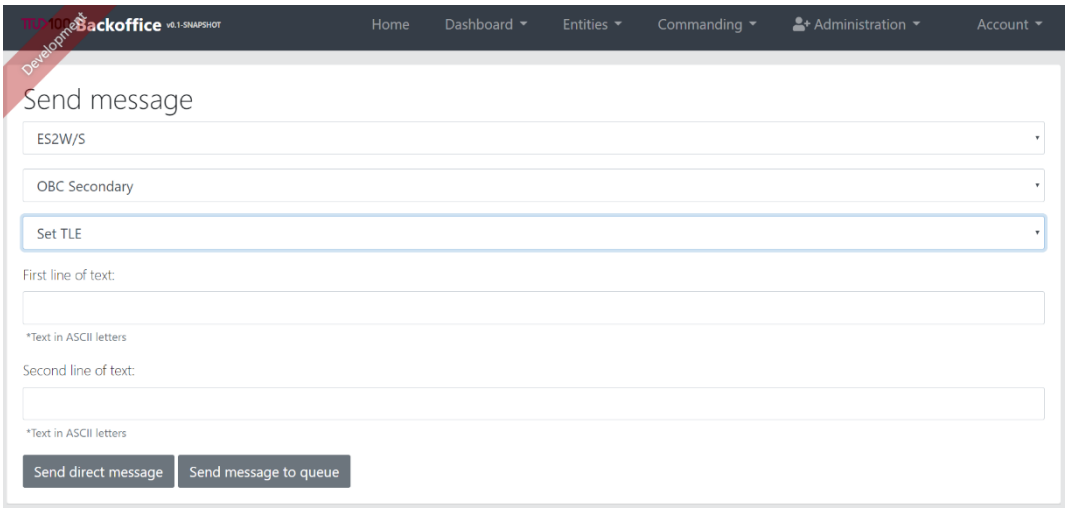
Lisa 6 – Käsuklassi klassidiagramm



Lisa 7 – HDLC andeosaklassi klassidiagramm



Lisa 8 – Üldiste käskude saatmise vaade



Lisa 9 – Üldise kirjutamiskäsu mooduli vaade