TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Rudolf Tammekivi 154885IALB

TUDENGISATELLIIDI SIDEMOODULI TESTTARKVARA

bakalaureusetöö

Juhendaja: Ivo Müürsepp
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: Rudolf Tammekivi

20.05.2019

Annotatsioon

Tudengisatelliit on TTÜ tudengite ning õppejõudude koostööna loodud nanosatelliit, mille arendamist alustati 2014. aastal.

Käesoleva töö eesmärk oli luua tarkvara võimaldamaks tudengisatelliidi sidemooduliga ühendumist, sellega suhtlemist ning funktsionaalsuse testimist.

Töös käsitletakse satelliidi sees kasutatava sideprotokolli erinevaid parameetreid ning kuidas erinevad moodulid suhtlevad üksteisega. Lisaks kirjeldatakse programmi arendusel kasutatavaid põhimõtteid ning tarkvara tööd sidemooduli testimisel.

Töö praktilise osana valmis PC peal jooksev graafilise kasutajaliidesega tarkvara, milles on integreeritud sideprotokolli erinevad parameetrid ning mis on võimeline lugema informatsiooni ja testima satelliidi sidemoodulit.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 22 leheküljel, 6 peatükki, 18 joonist, 2 tabelit.

Abstract

Student Satellite Communications Module Test Software

Student's satellite is a nanosatellite which was introduced in 2014. It is developed in collaboration between TTÜ students and professors. The satellite's main purpose is Earth observation and demonstration of different technologies. It is also used to perform different scientific experiments, such as computational fault tolerance and optical communication.

The goal of this thesis was to create a software that can connect with the student satellite's communications module, communicate and test its functionality, as well as provide means to run stress tests.

The thesis highlights the many different parameters of the communication protocol embedded inside the satellite as well as how the different submodules of the satellite can communicate with each other. It also demonstrates the methodologies and principles used to create the software.

As part of the practical work of this thesis, a software running on PC with a graphical user interface was developed that can successfully integrate the different parameters of the communication protocol and provide easy access to read the satellite's communication module's configuration and test the functions within the module.

The thesis is in Estonian and contains 22 pages of text, 6 chapters, 18 figures, 2 tables.

Lühendite ja mõistete sõnastik

ADC	*Analog-to-digital converter*
	Analoog-digitaalmuundur
BA	*Base address*
	Põhiaadress
COM	*Communication port*
	Järjestikport, tudengisatelliidi sidemoodul
CRC	*Cyclic redundancy check*
	Tsükkelkoodkontroll
CSV	*Comma-separated values*
	Komaeraldusega väärtused
GUI	*Graphical user interface*
	Graafiline kasutajaliides
HDLC	*High-level data link control*
JSON	*JavaScript Object Notation*
PC	*Personal computer*
	Personaalarvuti
Regex	*Regular expression*
	Regulaaravaldis
UART	*Universal asynchronous receiver-transmitter*
	Universaalne asünkroontransiiver
UHF	*Ultra high frequency*
	Ülikõrgsagedus

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

TTÜ100 tudengisatelliit on TTÜ tudengite ning õppejõudude koostööna loodud nanosatelliit, mille põhiline ülesanne on Maa seire ja sellega seonduva tehnoloogia demonstratsioon. Samuti tehakse satelliidiga erinevaid teaduskatseid, nagu näiteks arvutustehnika tõrkekindluse ning optilise side katseid.

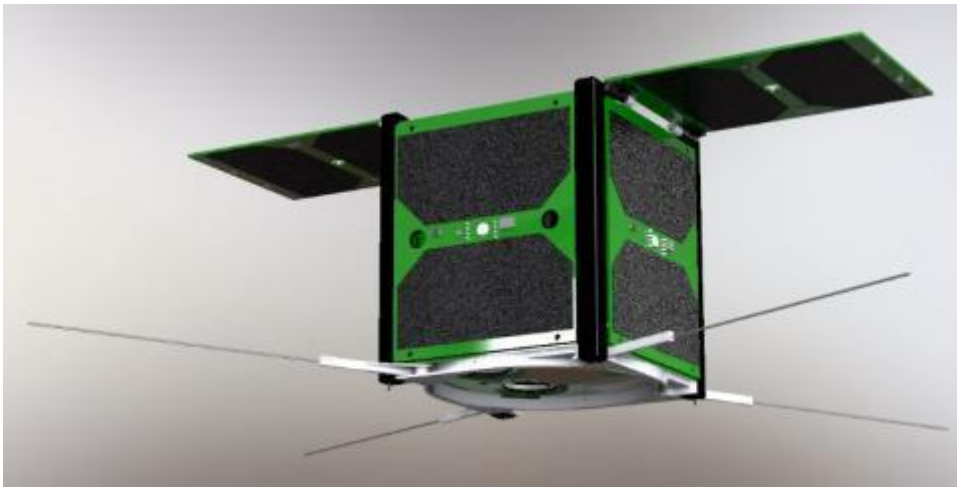
Satelliidi ennustatav orbiidile saatmine on juba sel aastal, mistõttu on vaja põhjalikult kontrollida satelliidi tarkvara ning eemaldada leitud vead. Samuti peab olema võimalik moodulite tarkvara uuendamine läbi raadiolingi. Seetõttu tekkis vajadus luua programm, mis lihtsustaks arendajate tööd, pakkudes võimalust sooritada erinevaid teste, mille põhjal ennetada uute vigade tekkimist.

Käesoleva töö eesmärk on luua tarkvara, mis edukalt suhtleks satelliidi sidemooduliga ning võimaldaks selle funktsionaalsuse testimist.

Ülesandena olid kindlaks määratud järgmised parameetrid: tarkvara peab olema võimeline jooksma Windows PC peal, sel on graafiline kasutajaliides ning lisaks põhilistele käskude saatmisele ning vastuvõtmisele võimaldama ka pikemaajalist testimist koormustestimise näol.

2 Ülevaade satelliidi disainist

Tudengisatelliit koosneb mitmest moodulist, millest igaühel on kindel ülesanne. Tudengisatelliit kuulub 1U kategooriasse, mis tähendab, et tema mass on kuni 1,33kg ning mõõtmed on kuni 10cm x 10cm x 10cm [1].



Joonis 1. TTÜ100 satelliit [2]

2.1 Elektrivarustus

Satelliidi üks olulisemaid osasid on selle toiteallikas EPS, mis peab tagama kõigile teistele alamsüsteemidele stabiilse toitepinge +5V. Energiat ammutatakse kokku üheksalt päikesepaneelilt, mis paiknevad kõigil tahkudel, välja arvatud alumisel, Maa suunas vaataval (Joonis 1). Päikesepaneelidelt saadud energia salvestatakse liitiumioon akudesse [3].

EPS ülesanne on lisaks toitepinge pakkumisele kontrollida teiste süsteemide olekuid, saates neile kindlaks määratud aja tagant päringu, millele oodatakse vastust. Kui vastus jääb saabumata, tehakse vastavale moodulile restart. Samuti on EPS süsteemi kriitilised osad dubleeritud [3].

2.2 Pardakaamera

Maa seiret tehakse kõrgresolutsioonkaamerate abil, millest tehtud pildid saadetakse maajaamale läbi sidemooduli. Kaamerad suudavad maa-lähedasel orbiidil teha pilte, mille ühe piksli resolutsioon on 50m x 50m maa ala [4].

2.3 Sidemoodul

Satelliidi sidemoodul koosneb kahest erisagedustel töötavast moodulist.

435 MHz sagedusel töötav UHF transiiver on kasutusel väiksemate andmemahtude juures, kui on vajalik satelliidile käske edastada ja vastu võtta satelliidi telemeetriainfot [2].

10,5 GHz sagedusel töötav X-riba saatja on kasutusel suurte andmemahtude juures, kui on vajalik edastada kaamerate poolt tehtud pilte maale, sest satelliidi eetriaeg on piiratud [2].

3 Suhtlus satelliidiga

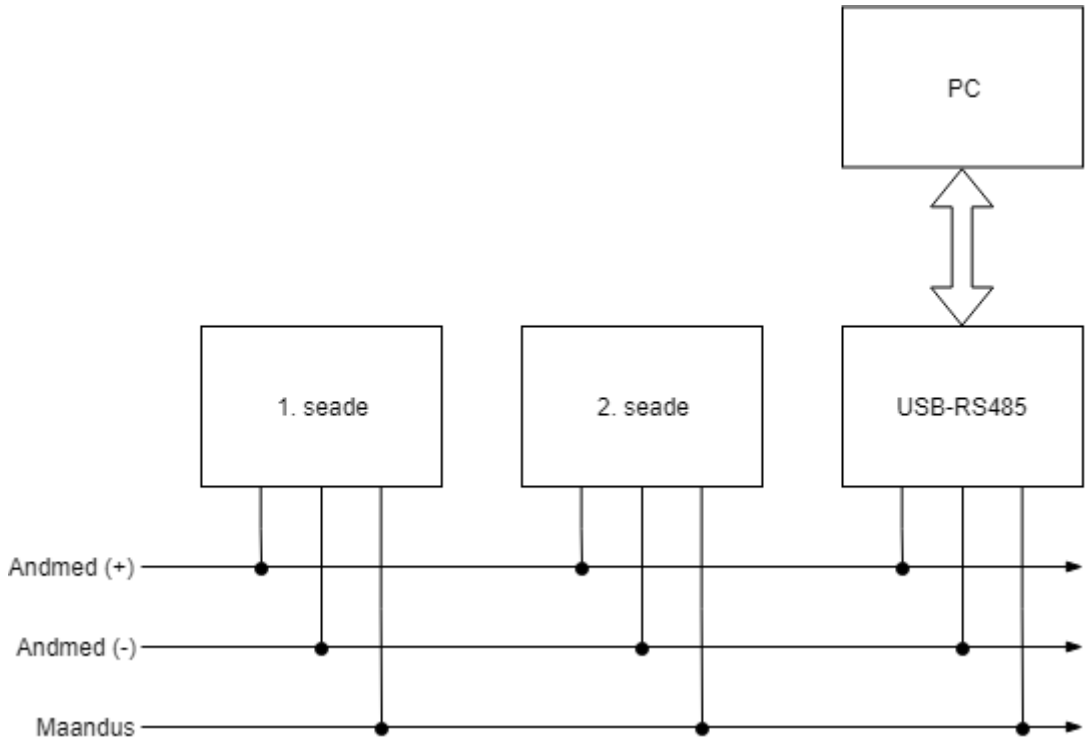
Telekommunikatsioonis on oluline, et informatsioon oleks mõlemale osapoolele üheselt arusaadav. Seetõttu on vajalik kokku leppida kindlates reeglites, mille põhjal mõlemad seadmed informatsiooni saadavad ning vastu võtavad.

Esiteks on vaja informatsioon teistele seadmetele füüsiliselt nähtavaks teha. Tudengisatelliidi puhul on füüsiliseks edastuskanaliks UART liides.

Teiseks on vaja informatsioon õigesti dekodeerida ehk arusaadavaks teha. Selleks on vajalik luua protokoll, mis on võimeline vajaliku informatsiooni üle füüsilise keskkonna üle kandma. Samuti on oluline häirekindlus, sest vastasel juhul võib tekkida olukordasid, kus füüsilise keskkonna tõttu võib informatsiooni valesti tõlgendada.

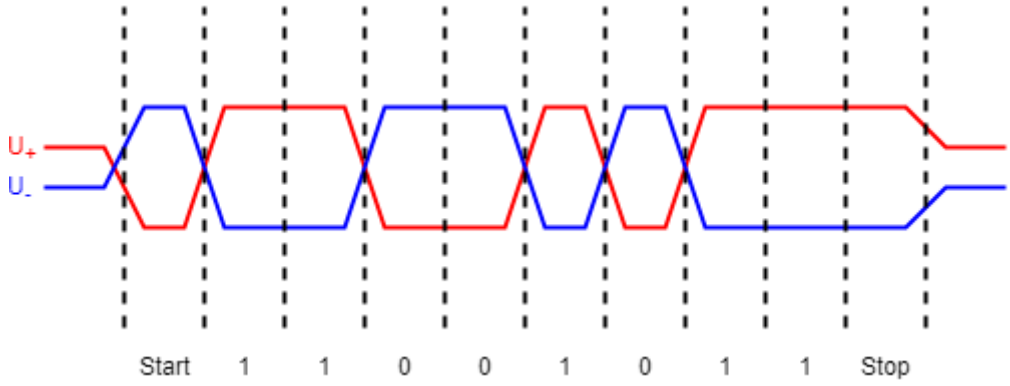
3.1 UART liides

Tudengisatelliit ühendub PC-ga läbi RS-485 standardi UART liidesega. Arvuti poolel on selleks vajalik kasutada USB-RS-485 üleminekut, mis vastavalt standardile tõlgendab füüsilised parameetrid ning loob virtuaalse COM pordi, mis loogiliselt vaadates käitub nagu tavaline COM port (Joonis 2).



Joonis 2. RS-485 ühendus seadmete vahel

Tudengisatelliidi puhul on sisemisel siinil andmeedastuskiiruseks 115200 bitti/s, edastatavad sümbolid on 8 bitti pikad, paarsuskontrolli ei kasutata ja stopp-bitte on üks. [5].



Joonis 3. Andmete saatmine läbi diferentsiaalpaari [6]

Andmete asünkroonne vastuvõtt algab hetkest, kui vastuvõtja näeb siinil ühe muutumist nulliks (langev front). Tudengisatelliidi puhul kasutatakse 115200 bitti/s andmeedastuskiirust, seega võetakse järgnevad bitid iga $\frac{1}{115200} = 8{,}68\ \mu s$ tagant vastu. Näidisel kujutatud baidi 0xd3 saatmine, kõige madalam bitt saadetakse esimesena (Joonis 3).

3.1.1 Põrgete vältimine

Igal seadmel, mis siinil asetseb, on pidevalt vastuvõtja sisse lülitatud. Kui võetakse vastu kaadri alguse lipp 0x7e, märgitakse kanal aktiivseks ning käivitatakse 100ms taimer. Kui selle aja jooksul edastatakse kõik andmed või aegub taimer, märgitakse kanal uuesti vabaks [5].

Andmete edastamisel mooduli saatja edastab ühe sümboli ning samal ajal mooduli vastuvõtja loeb ühe sümboli. Kui vastuvõetud sümbol ei ühti saadetuga, on tegemist põrkega ja saatja on kohustatud saatma nullbaidid tegeliku informatsiooni asemel. See võimaldab teisel seadmel samuti põrke varakult tuvastada [5].

3.2 Sideprotokoll

UART võimaldab läbi füüsilise keskkonna informatsiooni teistele seadmetele edastada, kuid kuna andmeid on võimalik edastada vaid ühe baidi haaval, siis on vajalik andmed omakorda kapsuleerida protokolliga, mis võimaldaks saata suuremas koguses andmeid.

Tudengisatelliidi puhul võeti selleks alguses kasutusele HDLC asünkroonse side protokoll, mis projekti arenedes muudeti satelliidile sobivamaks (Joonis 4).

HDLC					
Lipp	**Aadress**	**Kontroll**	**Andmed**	**Kontrollsumma**	**Lipp**
8 bitti	8 bitti	8 bitti	N baiti	16 bitti	8 bitti
Tudengisatelliidi sisemise siini kaader					
Lipp	**Pikkus**	**Andmed**	**Kontrollsumma**		
8 bitti	16 bitti	Kuni 256 baiti	16 bitti		

Joonis 4. HDLC ja tudengisatelliidi kaadrite võrdlus [5]

3.2.1 Lipp

Iga kaadri alguseks on fikseeritud väärtus 0x7e (binaarkujul 0b01111110). Kuna protokollis on kasutusele võetud kindla tähendusega väärtused, siis on vajalik tagada, et sellised väärtused ei esine mujal andmejadas. Selle teostamiseks lisatakse andmejadasse

spetsiaalne farssbait 0x7d, mis märgib, et temale järgnev bait on inverteeritud viienda bitiga (Joonis 5). Sama protsess toimub 0x7d enda puhul, kui see andmetes esineb [5].

	Algne	**Muudetud**
Heksadetsimaalkood	0x7e	0x7d 0x5e
Binaarkood	0b01**1**11110	0b1111101 0b01**0**11110

Joonis 5. Farssbaidi lisamine andmetesse

3.2.2 Pikkus

Pikkuse väli näitab andmete ja kontrollsumma väljade kogupikkust baitides. Selle põhjal saab vastuvõtja aru, kui pikk on täielik kaader. See on vajalik, sest kaadri lõppedes ei saadeta kaadri lõpu lippu, erinevalt HDLC protokollist [7].

3.2.3 Kontrollkood

Kontrollsumma arvutatakse tsükkelkoodkontrolli ehk CRC põhjal. Kaadrite pikkus on maksimaalselt 256 baiti, mistõttu on kasutusel 16 bitine CRC – $2^{16} = 65536$ erinevat kombinatsiooni, seega vea avastamata jäämise tõenäosus on väga madal.

Sidemoodulis kasutatav genereeriv polünoom: $G(x) = 0x8408 = x^{15} + x^{10} + x^3$

CRC arvutamiseks on vaja jagada info bitid läbi genereeriva polünoomiga. Tulemuseks on jääk, mis salvestatakse kaadri lõppu. Kui vastuvõtja on kogu info salvestanud, arvutab ta samamoodi CRC ning võrdleb vastuvõetud infos sisalduva CRCga. Kui numbrid ei ühti, on tekkinud viga ning kaadrit edasi ei töödelda.

Nii sidemoodul kui ka testimistarkvara peavad kasutama sama genereerivat polünoomi.

```
uint16_t HDLC::crc16(uint16_t crc, uint8_t d)
{
	crc ^= d;
	for (uint8_t i = 0; i < 8; i++) {
		if (crc & 0x0001)
			crc = (crc >> 1) ^ FCS_POLYNOMIAL;
		else
			crc >>= 1;
	}
	return crc;
}
```

Joonis 6. CRC arvutuse programmikood [5]

3.3 Sideprotokolli andmed

Sideprotokolli andmete väli sisaldab järgmisi elemente:

Aadress (BA)	**ID**	**Käsk (kood)**	**Lisaandmed**
4+4 bitti	8 bitti	16 bitti	Kuni 252 baiti

Joonis 7. Käsu andmed [5]

3.3.1 Aadress

Aadress koosneb kahest osast – saatja- ning vastuvõtja aadressist. Aadress on 4 biti pikkune, mistõttu on kokku võimalik kasutada 16 erinevat aadressi.

Tabel 1. Satelliidi moodulite aadressid [5]

Aadress	**Sõlm**
0	Maajaam
1	UHF transiiver (COM)
2	Asendikontrolli süsteem (ADCS)
3	Pardaarvuti (OBC)
4	Toitesüsteem (EPS)
5	X-riba saatja (COMX)
14 (0xE)	Sidemooduli testtarkvara

3.3.2 ID

ID ehk identifikaator määratakse kaadri saatmisel saatja poolt. Vastuvõtja seda ei kasuta, tema ülesanne on vaid vastuse saatmisel sama ID kasutada. Selle põhjal saab saatja aru, millisele päringule on vastus määratud.

3.3.3 Käsk

Käsu parameeter sisaldab omas käsu koodi. Käsu koodid erinevad moodulite kaupa ning nendega on võimalik lugeda ja kirjutada registreid ning muuta erinevaid parameetreid (vt Lisa 2, Lisa 3).

3.4 Baitide otsalisus

Baitide otsalisus (*endianness*) määrab baitide järjekorra, kui on vaja edastada andmeid või mällu salvestada arve, mis ei mahu ühe baidi ehk 8 biti sisse. 8 bitise arvu puhul on maksimaalne väärtus $2^8 - 1 = 255$ ehk binaarkujul 0b11111111. Suuremate arvude puhul lisatakse üks bait juurde ning tulemusena on võimalik salvestada juba $2^{16} - 1 = 65535$ maksimaalse väärtusena.

Protsessorite arhitektuurilise erinevuse tõttu on vaja kokku leppida, mis järjekorras baidid läbi kanali saadetakse (Joonis 8). Sidemooduli puhul on mikrokontrollerid *little-endian* tüüpi ning seetõttu saadab sidemoodul 16-bitiseid väärtusi vastavalt madalam bait enne, et vähendada mikrokontrolleri poolset üleliigset tööd.

Mäluaadress	**Little-endian**								**Big-endian**							
#00	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
#01	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Joonis 8. Big- ning little-endian erinevus [8]

4 Tarkvara disain

Tarkvara arendus jaotati kaheks – ees- ning tagasüsteem (*front-* ja *back-end*). Eessüsteemi arendus sisaldas endas kasutajaliidese disaini ning kasutaja programmiga opereerumist. Tagasüsteem tegeles kogu programmi sisemise loogikaga.

4.1 Raamistiku valik

Tarkvaraarendusel on väga oluline juba alguses otsustada, kas kasutada raamistikku ja millist. Raamistik suurendab oluliselt programmeerimise efektiivsust, pakkudes erinevaid teeke, mida arendaja saab kohe kasutada. Kuna testtarkvara eeldab graafilist kasutajaliidest, on raamistiku valik vältimatu.

Tabel 2. Raamistike võrdlus

Nimetus	GUI	COM tugi	Windows	Linux	C#	C++
Microsoft Foundation Classes						
Windows Forms						
GTK						
Qt						

Programmi arendusel otsustati raamistiku valikul Qt kasuks, sest Qt on lisaks GUI raamistikule toetatud ka teistel platvormidel nagu Linux ning OS X. Samuti on Qt-l lai valik lisandmooduleid, nagu näiteks QSerialPort, mistõttu ei ole vaja Windowsile spetsiifilise funktsioone kasutada ning tulevikus saab programmi koodi kompileerida lihtsa vaevaga teistele platvormidele. Qt valikut soodustas ka C++ tugi ja lai kogukond.

4.2 Siseloogika

Kogu tarkvara põhineb sündmus-põhise programmeerimise (*event-driven programming*) põhimõtetel. Sündmuspõhine programmeerimine eeldab, et igas lõimes jookseb kood, mis ootab sündmusi ning reageerib neile. Eelkõige on see vajalik kasutajaliideste puhul, kus ei või kunagi teada, millal nupu vajutus võib toimuda. Sama loogikat kasutati ka sidemooduliga suhtlemisel, sest kui sidemooduliga on tekkinud probleem ning see ei vasta temale saadetud käsule, on vaja sellest kasutajale teada anda.

Sündmuspõhine programmeerimine võimaldas luua modulaarse süsteemi, mida on võimalik hõlpsasti sisse/välja lülitada, ilma et oleks vajadust erioperatsioone omavahel sünkroniseerida, lihtsustades oluliselt programmi koodi.

4.2.1 Sideprotokolli implementeerimine

Programm suhtleb sidemooduliga läbi järjestikpordi, mis on oma olemuselt puhverdamata suhtluskanal. Kõik levinud operatsioonisüsteemid, sh Windows, Linux ning Mac OS on valmis kirjutatud draiverid, mis puhverdavad järjestikporti saabuvat informatsiooni. See on eelkõige vajalik, sest programmi tasemel ei pruugi olla võimalik nii täpselt iga bitti lugeda ning vastasel juhul läheks osa informatsiooni kaduma. Tänu *low-level* draiverile peab programm ainult järjestikpordi draiverit konfigureerima ning talle lugemis-/kirjutamiskäske saatma.

Qt-s pakutav QSerialPort toetab nii sünkroonset andmete vastuvõttu kui ka asünkroonset. Kuna testtarkvara on üles ehitatud sündmuspõhiselt, siis kasutatakse QSerialPorti samuti asünkroonses režiimis.

Programmi käivitamisel avatakse järjestikport ning määratakse funktsioon, mille sündmuste haldaja käivitab, kui järjestikpordis on andmeid.

Saabuvad andmed salvestatakse QByteArray klassi, mis on võimeline dünaamiliselt kasvama, kuid jõudluse huvides on reserveeritud juba alguses 261 baiti, mis on sisemisel siinil saadetava kaadri maksimaalne pikkus.

QByteArray tühjendatakse hetkest, kui saabub uue kaadri alguse lipp. Peale seda loetakse kaadri pikkus ning peale seda võetakse andmed järjest vastu, samal ajal arvutab programm saabuvate baitide põhjal CRC ning kui on vastu võetud terve kaader,

kontrollitakse CRC kaadris olevaga ning kui kõik klapib, saadetakse kaader edasi programmi ülemistele kihtidele.

QByteArray klassi on laiendatud, et luua funktsioonid saabuvate/saadetavate andmete mugavaks töötlemiseks (Joonis 9). Sarnaselt on loodud lihtsustavad funktsioonid farssbaitide lisamiseks.

```
uint16_t SByteArray::readEndian(int index) const
{
    uint8_t lsb = static_cast<uint8_t>(at(index));
    uint8_t msb = static_cast<uint8_t>(at(index+1));

    int value = msb << 8 | lsb;

    return static_cast<uint16_t>(value);
}
```

Joonis 9. 16-bitise väärtuse lugemine andmejadast

Andmete saatmise protseduur:

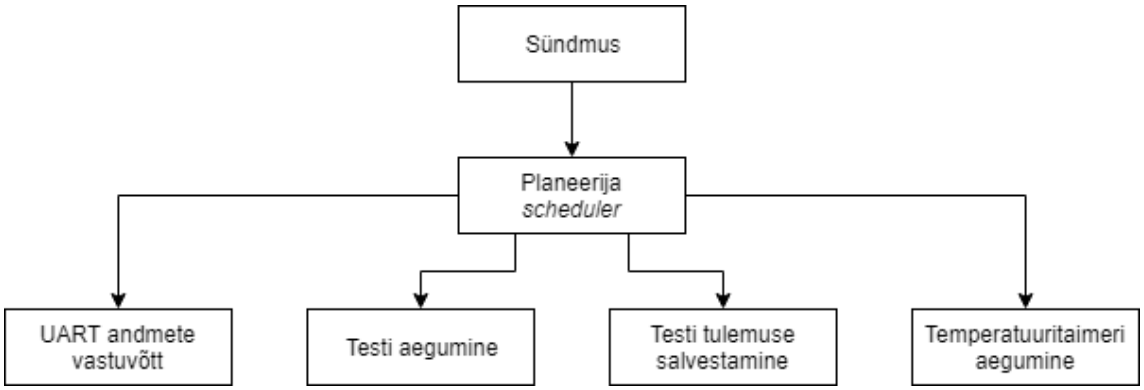
1. Ülemiselt kihilt andmete vastuvõtt (testid, temperatuurijälgija)
2. Käsu saaja, ID, käsu koodi ning lisaandmete salvestamine kaadrisse (vt Lisa 4)
3. Sideprotokolli kaadri loomine (vt Lisa 4)
4. Järjestikpordile andmete saatmine

Andmete vastuvõtu protseduur:

1. Järjestikpordi sündmus uute andmete kohta
2. Andmete lugemine puhvrisse (vt Lisa 5)
3. Kaadri töötlus ning vigade tuvastus (vt Lisa 5)
4. Ülemistele kihtidele töödeldud kaadri edastus (testid, temperatuurijälgija, logimine)

4.2.2 Automaattestimine

Automaattestimine on realiseeritud eraldi lõimes. Eraldi lõime kasutamine võimaldab kogu testimisega seotud loogika hoida eraldi kasutajaliidesest. Kõik sündmused on eraldiseisvad ning näiteks testitulemuse salvestamine käivitab uue sündmuse, milleks on järgmise testi täitmine või testimise lõpetamine (Joonis 10). Samuti võimaldab see signaalide vahetamist mitme lõime vahel. Näitena saab automaat-testija oma staatust edastada signaali abil kasutajaliidesele, mis uuendab graafiliselt programmi ning kirjutab logi.



Joonis 10. Automaat-testija lõime sündmuste planeerija

4.2.3 Testide töötlemine

Kõik testid, mida automaat-testija täidab, on esitatud tekstikujul, sest see võimaldab kõige lihtsamini toetada erinevat tüüpi teste, millel võivad olla erinevat tüüpi parameetrid. Kõik testid töötlevad neile määratud parameetreid eraldiseisvalt ning automaat-testija peab ainult kindlaks määrama, millise testiga on tegu (Joonis 11).

```
Test *TestFactory::makeTest(const QString &name, COM &com, QObject *parent)
{
    static const QHash<QString, TestMaker*> hashMap({
        { "CMD", make<Command> },
        { "PING", make<Ping> },
        { "RREG", make<RWReg> },
        { "WREG", make<RWReg> },
        { "SLEEP", make<Sleep> },
        { "XBANDOFF", make<XBandOnOff> },
        { "XBANDON", make<XBandOnOff> },
        });

    if (hashMap.contains(name))
        return hashMap.value(name)(com, parent);
    return nullptr;
}
```

Joonis 11. Automaat-testija teksti töötlemine testiks

4.2.4 Sätete salvestamine

Testtarkvaral on mitmed parameetrid, mida igakordsel käivitusel oleks ebamugav uuesti sisestada. Seetõttu salvestatakse programmi sätted automaatselt muutmisel JSON failina (Joonis 12). Programmeerimisel on selleks kasutatud Qt-s pakutav QJsonObject ning QJsonDocument.

```
{
    "comPortBaud": 9600,
    "comPortName": "COM8",
    "debugIOCommands": true,
    "debugIORaw": false,
    "dumpMemory": false,
    "logTemp": false,
    "stopOnFailure": true
}
```

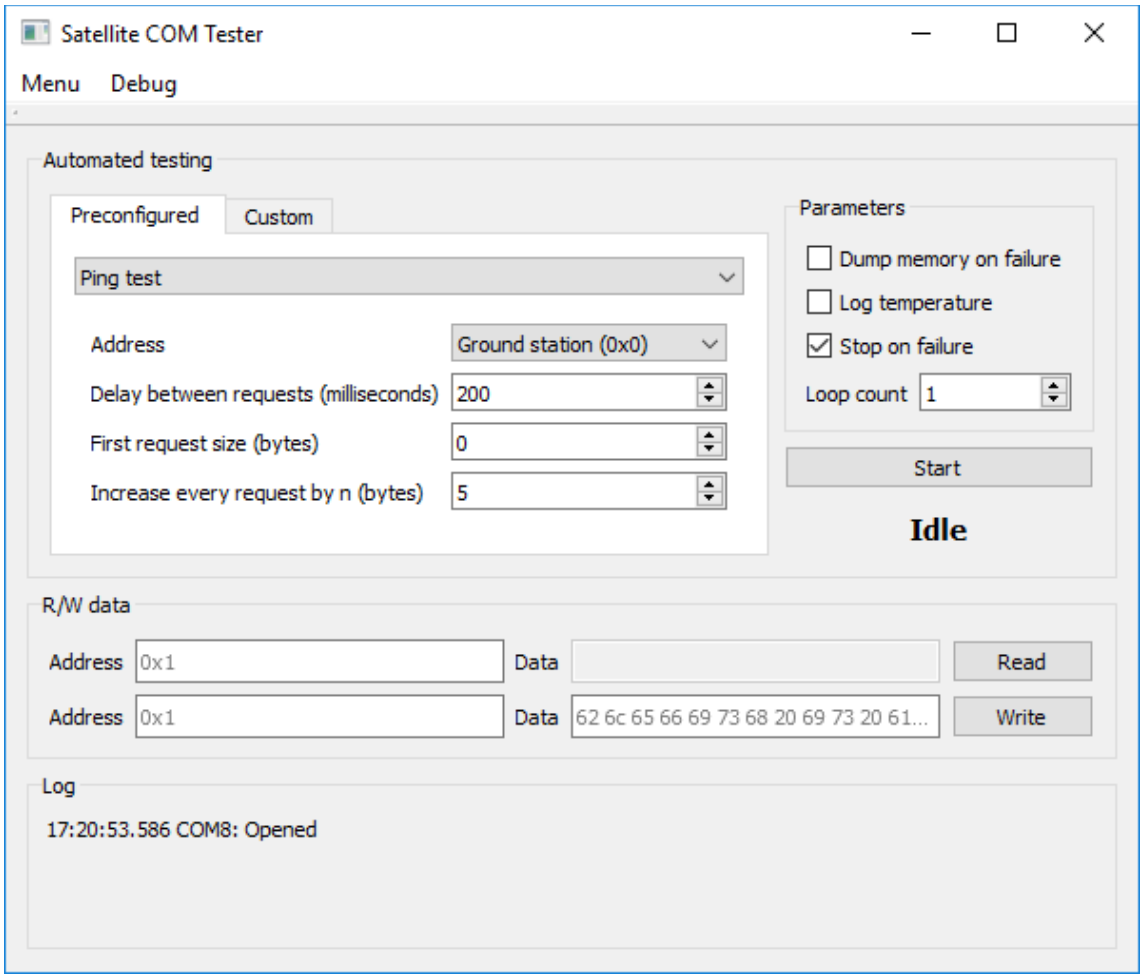
Joonis 12. Programmi sätted JSON kujul

4.3 Kasutajaliides

Tarkvarale kasutajaliidese loomisel on oluline kasutajasõbralikkus. Mida keerulisemaks muutub programm, seda rohkem tekib erinevaid nuppe ja valikuid. Seetõttu otsustasin, et põhiline võimekus oleks esiaknas esindatud ning lisavõimalused on menüüdes kategoriseerituna (vt Lisa 6).

4.3.1 Põhiaken

Programmi käivitades avaneb kasutajale põhikasutajaliides, kus on võimalik valida automaatsete sisseehitatud- ning käsitsi loodud testide vahel. Programmi allosas on jooksev logi, millega on võimalik jälgida moodulite vahel saadetavaid sõnumeid.



Joonis 13. Programmi põhiaken

4.3.2 Uute testide loomine

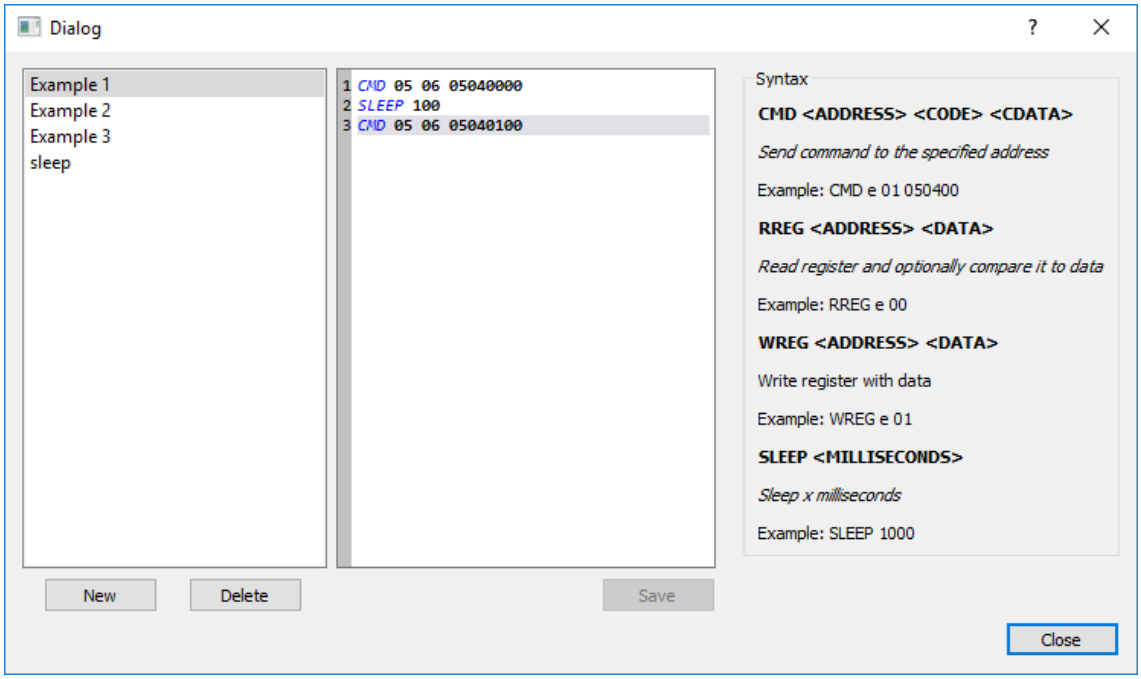
Kuna satelliidi sidemoodulit arendatakse ajas edasi ning võib tekkida vajadus muuta mõnda parameetrit jooksupealt, oli vaja luua programmile võimalus lisada uusi teste. Selle jaoks on programmi ehitatud lihtsustatud tekstitöötleja, mis on võimeline aru saama kindlatest ülesannetest (vt Lisa 1; Joonis 15).

Tekstitöötlejale on lisatud teksti esile toomine, kõik käsud ning nende parameetrid, samuti kommentaarid, on tavatekstist välja toodud. Selle jaoks on kasutatud

regulaaravaldist (*regex*). Teksti värvimine ja parameetrite eraldi välja toomine vähendab kasutaja poolt tehtud vigasid ning on visuaalselt lihtsam lugeda (Joonis 14).

```
numberFormat.setFontWeight(QFont::Bold);
rule.pattern = QRegularExpression("\\b[0-9]+\\b|\\b0[Xx][0-9A-Fa-f]+\\b");
rule.format = numberFormat;
highlightingRules.append(rule);
```

Joonis 14. Regulaaravaldis numbrite esile toomiseks



Joonis 15. Uute testide loomine

5 Sidemooduli testimine

Testtarkvaral on mitmed parameetrid, millega on võimalik tulemusi jälgida:

- *Dump memory on failure* – loe kõik registri väärtused ning kuva need logisse ebaõnnestunud testi puhul.
- *Log temperature* – salvesta UHF ja X-riba moodulite võimendi temperatuur CSV formaadis faili 5s intervalliga.
- *Stop on failure* – peata järgnevad käsud pärast ebaõnnestumist. On oluline, et isegi kui see on valimata, jätab programm meelde, et käsk on ebaõnnestunud ning see kajastub tulemustes.
- *Loop count* – korduste arv.

5.1 Temperatuuri mõõtmine

Testtarkvarale on sisse ehitatud automaatne temperatuuri mõõtmine. Kui valik on kasutajaliideses sisse lülitatud, salvestatakse iga 5s intervalliga sidemooduli temperatuur CSV formaadis faili. CSV formaati oskavad töödelda tabeliks kõik enimlevinud tabelitöötlustarkvarad, sh Microsoft Excel ning LibreOffice Calc.

Sidemoodulis kasutatakse temperatuuri mõõtmiseks Texas Instruments poolt loodud LM94021BIMG kiipi, mille väljundiks on temperatuuriga võrdeline alalispinge. Andmelehel on välja toodud pinge sõltuvus temperatuurist (1), mille põhjal on avaldatud temperatuur (2) [9].

$$V_{TEMP} = 2230{,}8 - 13{,}582(T - 30) - 0{,}00433(T - 30)^2\ [mV] \quad (1)$$

$$T = \frac{10}{433}\left(10\sqrt{55777045 - 4330 \cdot V_{TEMP}} - 66611\right)\ [C°] \quad (2)$$

Pinge lugemiseks saadetakse sidemoodulile käsk võimendi temperatuuri registri lugemiseks. Register sisaldab ADC poolt mõõdetud väärtust, mis on vaja teisendada pingeks (3).

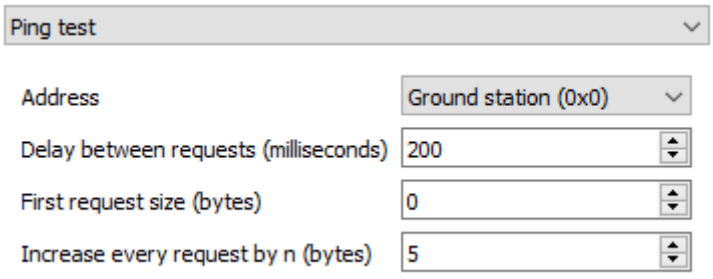
$$V_{TEMP} = \frac{3330 \cdot V_{meas}}{1024} \; [mV] \tag{3}$$

5.2 Koormustestimine

Testtarkvara koormustestimine on lahendatud võimalusega näiteks ühte testi mitmeid kordi saata. Näiteks on kasutajal võimalus saata X-riba saatjale käsk võimendi sisse-välja lülitamiseks ning muutes korduste arvu, on võimalik vajadusel korrata sama testi kuni 9999 korda. Sama võimalus on realiseeritud ka automaatsel testimisel. Lisaks saab sisse lülitada temperatuuri logimise, mille põhjal testimise lõppemist on võimalik tabelitöötlustarkvara abil jälgida, kas temperatuur on normide piires terve testimise vältel.

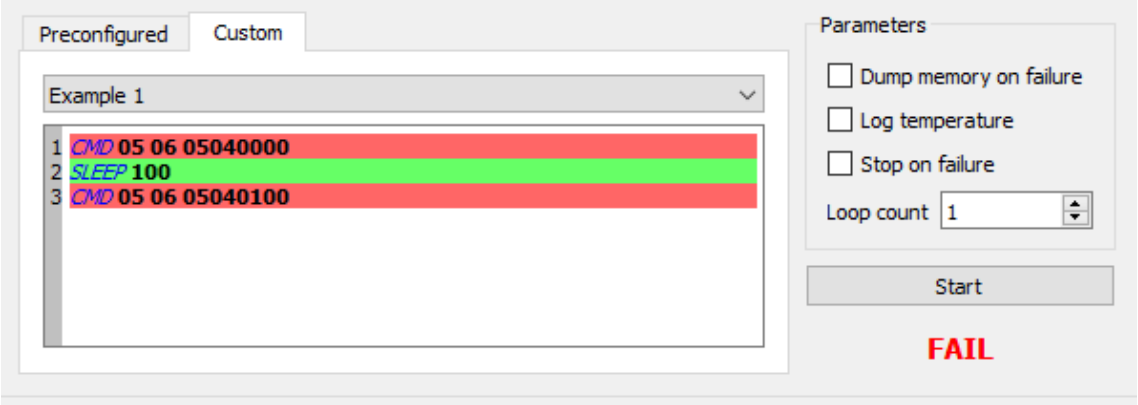
5.3 *Ping* test

Ping test sisaldab omas saatja aadressi, vastuvõtja aadressi ning suvalist andmejada, mis käsule juurde lisatakse (Joonis 16). Testi põhimõte on saata moodulile *PING* käsk koos kindla pikkuse andmetega vastu võtta täpselt samasugune andmejada ning kui vastuvõetud andmed ei ühti või on jäänud vastus tulemata, on testi tulemuseks *FAIL*.



Joonis 16. Ping testi valikud

5.4 Käsitsi loodud testid



Joonis 17. Käsitsi loodud testi tulemus

Käsitsi loodud testidel on lisaks logile võimalus näha iga testi tulemus programmiaknas visualiseerituna (Joonis 17).

5.5 Testimistulemused

Testimise lõppedes salvestatakse sellest logifail, mida on hiljem võimalik avada. Tulemused salvestatakse faili kujul TestResults/PP_KK_AA/TT_MM_SS.log.txt. Logis on kuvatud kogu satelliidi sisemisel siinil toimunud suhtlus ning testide tulemused (Joonis 18).

```
17:16:20.143 Automated testing STARTED
17:16:20.143 COM11: e->5 (req): cmd 000 #014
17:16:20.144 79 db a8 92 50 54 7d 5d 96 03 40 ea fa 0d bd e6 f3 a4 b7 f8 f3 3d 5f
      e5 c2 52
17:16:20.152 COM11: 5->e (rep): cmd 000 #014
17:16:20.153 79 db a8 92 50 54 7d 96 03 40 ea fa 0d bd e6 f3 a4 b7 f8 f3 3d 5f e5
      c2 52
17:16:22.149 Slept for 2000 ms
17:16:22.150 COM11: e->5 (req): cmd 000 #015
17:16:22.151 d9 8b d0 63 35 0d ca 3b a1 30 6b 74 dd 05 95 3d ef c9 a0 5b 17 09 f2
      6e 05 09 cf 76 4a 67 73 95 0d fc 5e 17 ff 03 f0 af 91 7d 5d 35 03 64 d4 02
      6b 00 be 41 f6 ca 64 89 af 30 c4 f2 1e
17:16:22.171 COM11: 5->e (rep): cmd 000 #015
17:16:22.172 d9 8b d0 63 35 0d ca 3b a1 30 6b 74 dd 05 95 3d ef c9 a0 5b 17 09 f2
      6e 05 09 cf 76 4a 67 73 95 0d fc 5e 17 ff 03 f0 af 91 7d 35 03 64 d4 02 6b
      00 be 41 f6 ca 64 89 af 30 c4 f2 1e
17:16:24.176 Slept for 2000 ms
17:16:26.218 COM11: e->5 (req): cmd 000 #017
17:16:26.218 01 79 63 e1 25 d0 5b 43 62 2d 42 d5 18 08 00 ef 12 8b 67 29 50 46 17
      b9 09 24 9e fc f0 9e e4 52 b7 2e c7 2f d1 07 02 6f 7d 5d 03 53 ea 00 dd dd
      49 31 a0 cb 18 3b 5f 36 1c 9f 27 48 e6 78 32 a2 a8 03 5d fc 48 5e dc 0b b3
      90 2d a8 74 ca 4a 2e 85 ed 23 24 64 4b 4b 1c 6a b2 f2 da 59 a7 13 b9 34 ef
      ee 4b 53 54 b9 40 b6 a5 93 89 9a ff b9 bd 4a 4b fc bb 38 08 73 91 4c 4b 6d
      9c 7c 03 a9 f1 9d 82 ca
17:16:47.218 Slept for 2000 ms
17:16:47.220 Automated testing FAILED
```

Joonis 18. Testilogi

6 Kokkuvõte

Töö peamiseks eesmärgiks oli luua tarkvara satelliidi tudengisatelliidi sidemooduliga suhtlemiseks ja selle funktsionaalsuse testimiseks.

Töö esimeses peatükis tudengisatelliidi üldist modulaarset disaini, et saada üldpilti moodulist, millega tarkvara peab suhtlema. Töö eesmärgi saavutamiseks oli vajalik kindlaks teha, kuidas sidemooduliga suhtlemine käib, mistõttu on töö kolmandas peatükis põhjalikult kirjeldatud satelliidiga ühendumise liidest ning satelliidi sisemisel siinil jooksva protokolli omadused ning parameetrid. Neljandas peatükis on kirjeldatud tarkvara arendusel järgitavaid põhimõtteid ning miks on otsustatud kasutada just Qt raamistikku. Kirjeldatud on programmi osasid, millel on otsene seos sidemooduliga. Samuti on välja toodud programmi graafiline kasutajaliides. Viimases peatükis on kirjeldatud sidemooduli testimisel pakutavad valikud ning mis on testimise tulemus, kuidas salvestatakse logi testide kohta ning mis infot need sisaldavad.

Töö tulemina valmis Windowsil jooksev graafilise kasutajaliidesega tarkvara, mis läbi COM pordi suhtleb sidemooduliga, võimaldab koormustestimist pikemal perioodil ning jälgib ka sidemooduli temperatuuri. Kuigi põhifunktsionaalsus sisseehitatud testide ning käsitsi uute testide loomise võimaluse näol täidab püstitatud eesmärke, on selge, et tulevikus võib tekkida vajadus luua uut tüüpi teste, mille jaoks on vajalik tarkvara edasi arendada.

Kasutatud kirjandus

[1] „CubeSat Overview,“ [Võrgumaterjal]. Available: https://www.nasa.gov/mission_pages/cubesats/overview. [Kasutatud 11 05 2019].

[2] „TTÜ100 Satelliidi tuvustus,“ [Võrgumaterjal]. Available: https://www.ttu.ee/projektid/mektory-est/satelliidiprogramm-4/satelliidiprogramm/. [Kasutatud 11 05 2019].

[3] „Elektrivarustus < TTÜ100 Satelliidi tuvustus,“ [Võrgumaterjal]. Available: https://www.ttu.ee/projektid/mektory-est/satelliidiprogramm-4/satelliidiprogramm/wp2-e-elektrivarustus/. [Kasutatud 05 11 2019].

[4] „Pardakaamera < TTÜ100 Satelliidi tutvustus,“ [Võrgumaterjal]. Available: https://www.ttu.ee/projektid/mektory-est/satelliidiprogramm-4/satelliidiprogramm/wp1-b-pardakaamera/. [Kasutatud 11 05 2019].

[5] M. Kaal, „Bus protocol,“ [Võrgumaterjal]. [Kasutatud 12 05 2019].

[6] „RS-485,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/RS-485. [Kasutatud 09 05 2019].

[7] „ISO/IEC 13239, High-level data link control (HDLC) procedures,“ 2002.

[8] „Endianness,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Endianness. [Kasutatud 04 05 2019].

[9] T. I. Incorporated, „LM94021/LM94021Q Multi-Gain Analog Temperature Sensor,“ 2013. [Võrgumaterjal]. Available: https://www.ti.com/lit/ds/symlink/lm94021-q1.pdf. [Kasutatud 12 05 2019].

Lisa 1 – Testide loomise süntaks

CMD <ADDRESS> <CODE> <CDATA>

Send command to the specified address

Example: CMD e 01 050400

PING <ADDRESS> <DATA>

Ping an address and wait for the result

Example: PING 5 0a1b2c3d4e5f

RREG <ADDRESS> <DATA>

Read register and optionally compare it to data

Example: RREG e 00

Example: RREG e 00 ff

WREG <ADDRESS> <DATA>

Write register with data

Example: WREG e 01

SLEEP <MILLISECONDS>

Sleep x milliseconds

Example: SLEEP 1000

XBANDON / XBANDOFF

Shorthand for turning X-band transmitter on/off

Lisa 2 – Sidemooduli käsud

```
UHF
COM_PING = 0x0000, // ping
COM_READMULTI = 0x0003, // loe mitu registrit
COM_WRITEREG = 0x0006, // kirjuta register
COM_WRITEMULTI = 0x0010, // kirjuta mitu registrit
COM_ERASE_IMAGE = 0x800, // kustuta uuendamise ruum
COM_UPDATE_IMAGE_ENCRYPTED = 0x801, // kirjuta krüptitud andmed ruumi
COM_PROGRAM_IMAGE=0x803, // kopeeri uuendamise ruumilt kood töötavaks koodiks

X-riba
COMX_PING = 0x0000, // ping
COMX_READMULTI = 0x0003, // loe mitu registrit
COMX_WRITEREG = 0x0006, // kirjuta register
COMX_WRITEMULTI = 0x0010, // kirjuta mitu registrit
COMX_ERASE_IMAGE = 0x800, // kustuta uuendamise ruum
COMX_UPDATE_IMAGE_ENCRYPTED = 0x801, // kirjuta krüptitud andmed ruumi
COMX_PROGRAM_IMAGE=0x803, // kopeeri uuendamise ruumilt kood töötavaks
koodiks
```

Lisa 3 – Sidemooduli registrid

```
UHF
0x0000 staatus
       bit0 saatja sisselülitatud
       bit1 raadio hõivatud
       bit2 madal pinge
       bit3 võimendi ülekuumenemine
       bit4 väljundvõimsus madal
       bit5
       bit6
       bit7
0x0001 väljundvõimsus
0x0002 võimendi temperatuur
0x0003
0x0004
0x0005 mikrokontrolleri pinge
0x0006
0x0007
0x0008 raadiokaadreid vastuvõetud
0x0009 kehtetuid kaadreid vastuvõetud
0x000a katkestatud kaadreid (vastuvõtmine katkestatud vastuvõtmise ajal)
0x000b raadiokaadreid saadetud

X-riba
0x400 staatus
       bit3 võimendi ülekuumenemine
0x401 väljundvõimsus
0x402 võimendi temperatuur
0x403 kasutamata
0x404 pinge
0x405 saatja sisse/väljalülitamine (1=väljas, 0=sees)
0x406 PLL register 4, kõrge sõna
0x407 PLL register 4, madal sõna
0x408 PLL register 3, kõrge sõna
0x409 PLL register 3, madal sõna
0x40a PLL register 2, kõrge sõna
0x40b PLL register 2, madal sõna
0x40c PLL register 1, kõrge sõna
0x40d PLL register 1, madal sõna
0x40e PLL register 0, kõrge sõna
0x40f PLL register 0, madal sõna
0x410 kasutamata
0x411 kasutamata
```

Lisa 4 – Kaadrite loomine tarkvaras

```
QByteArray CMD::constructCommandFrame()
{
    SByteArray cdata;
    // Address (BA) 8 bits
    uint8_t ba_address = (ADDR_SCOMTESTER << 4) | (receiverAddress & 0xf);
    cdata.append(ba_address);
    // ID 8 bits
    cmdId = globalCmdIdCounter;
    if (globalCmdIdCounter++ == 0xff)
        globalCmdIdCounter = 0;
    cdata.append(cmdId);
    // Command (Code) 16 bits LE
    cdata.appendEndian(cmdCode);
    // Command-specific data N bytes
    cdata.append(cmdData);
    return cdata;
}
```

```
SByteArray HDLC::constructFrame(const QByteArray &data)
{
    uint16_t crc = 0xffff;
    for (auto c : data) {
        uint8_t &byte = reinterpret_cast<uint8_t&>(c);
        crc = crc16(crc, byte);
    }
    crc ^= 0xffff;
    SByteArray frame;
    frame.append(FRAME_START);
    frame.appendStuffed(static_cast<uint16_t>(data.length() + FCS_SIZE));
    frame.appendStuffed(data);
    frame.appendStuffed(crc);
    return frame;
}
```

Lisa 5 – Kaadrite vastuvõtt tarkvaras

```
void HDLC::receiveData(const QByteArray &data)
{
    for (auto c : data) {
        uint8_t &byte = reinterpret_cast<uint8_t&>(c);

        if (escapeChar) {
            byte ^= FRAME_ESCAPE_BITS;
            escapeChar = false;
        } else if (byte == FRAME_ESCAPE) {
            escapeChar = true;
            continue;
        } else if (byte == FRAME_START) {
            reset();
        }

        recvBuffer.append(c);

        int bufferLength = recvBuffer.length();
        if (bufferLength < HEADER_LENGTH) {
            /* EMPTY */
        } else if (bufferLength == HEADER_LENGTH) {
            /* LE */
            bufferLen = static_cast<uint16_t>(recvBuffer.at(2) << 8 |
recvBuffer.at(1));
        } else if (bufferLength <= HEADER_LENGTH + bufferLen - FCS_SIZE) {
            /* Calculate CRC on the fly. */
            dataCrc = crc16(dataCrc, byte);
        } else if (bufferLength == HEADER_LENGTH + bufferLen) {
            verifyFrame();
        }
    }
}
```

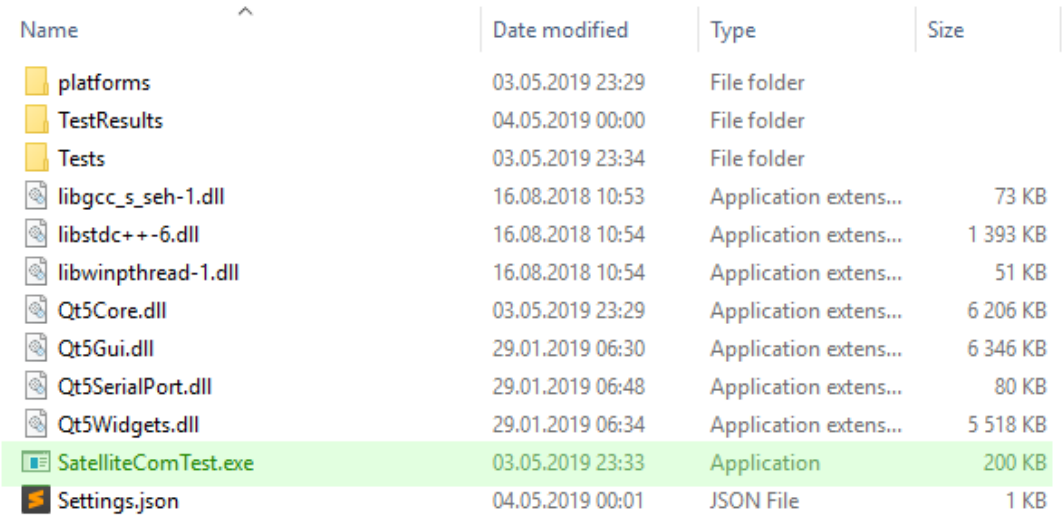
```
CMD::CMD(const SByteArray &cdata)
{
    // Address (BA) 8 bits
    char cAddress = cdata.at(0);

    uint8_t ba_address = reinterpret_cast<uint8_t&>(cAddress);
    receiverAddress = ba_address & 0xF;
    senderAddress = ba_address >> 4;

    // ID 8 bits
    char id = cdata.at(1);
    cmdId = reinterpret_cast<uint8_t&>(id);

    // Command (Code) 16 bits LE
    uint16_t cmd = cdata.readEndian(2);
    isReply = cmd & (1 << 15);
    isError = cmd & (1 << 14);
    cmdCode = cmd & 0xFFF;

    for (int i = 4; i < cdata.length(); i++) {
        cmdData.append(static_cast<uint8_t>(cdata.at(i)));
    }
}
```

Lisa 6 – Kasutajamanuaal

1. Installeerimine

Programm on pakitud ZIP arhiivina. ZIP arhiivi lahti pakkimisel tekib kaust nimega SComTest.

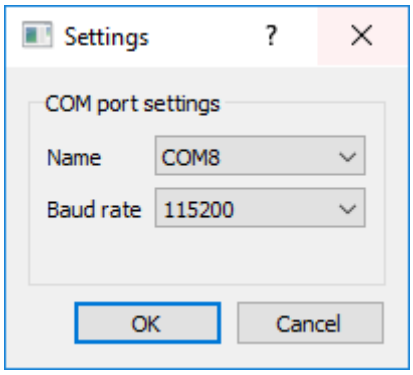
Programmi käivitamiseks tuleb avada SComTest kaustas programm nimega SatelliteComTest.exe



Name	Date modified	Type	Size
platforms	03.05.2019 23:29	File folder	
TestResults	04.05.2019 00:00	File folder	
Tests	03.05.2019 23:34	File folder	
libgcc_s_seh-1.dll	16.08.2018 10:53	Application extens...	73 KB
libstdc++-6.dll	16.08.2018 10:54	Application extens...	1 393 KB
libwinpthread-1.dll	16.08.2018 10:54	Application extens...	51 KB
Qt5Core.dll	03.05.2019 23:29	Application extens...	6 206 KB
Qt5Gui.dll	29.01.2019 06:30	Application extens...	6 346 KB
Qt5SerialPort.dll	29.01.2019 06:48	Application extens...	80 KB
Qt5Widgets.dll	29.01.2019 06:34	Application extens...	5 518 KB
SatelliteComTest.exe	03.05.2019 23:33	Application	200 KB
Settings.json	04.05.2019 00:01	JSON File	1 KB

2. COM pordi ülesseadmine

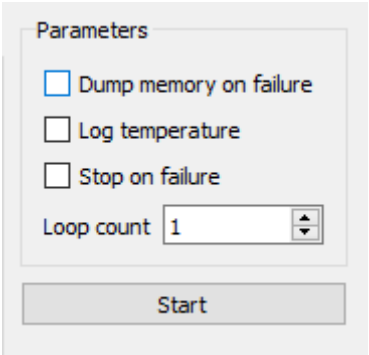
Esimesel käivitusel küsitakse COM pordi valikut, mis tuleb käsitsi seada. Kui COM porti ei ole valikus saadaval, tuleb kontrollida, kas Windowsil on vajalikud draiverid installeeritud ning kas COM port tekib Windowsi Device Manageri. Seejärel on vajalik seadete aken uuesti avada.



Edukast COM pordi avamisest teatab logiaken. Kui COM port ei ole avatud, ei ole võimalik ühtegi testi sooritada.

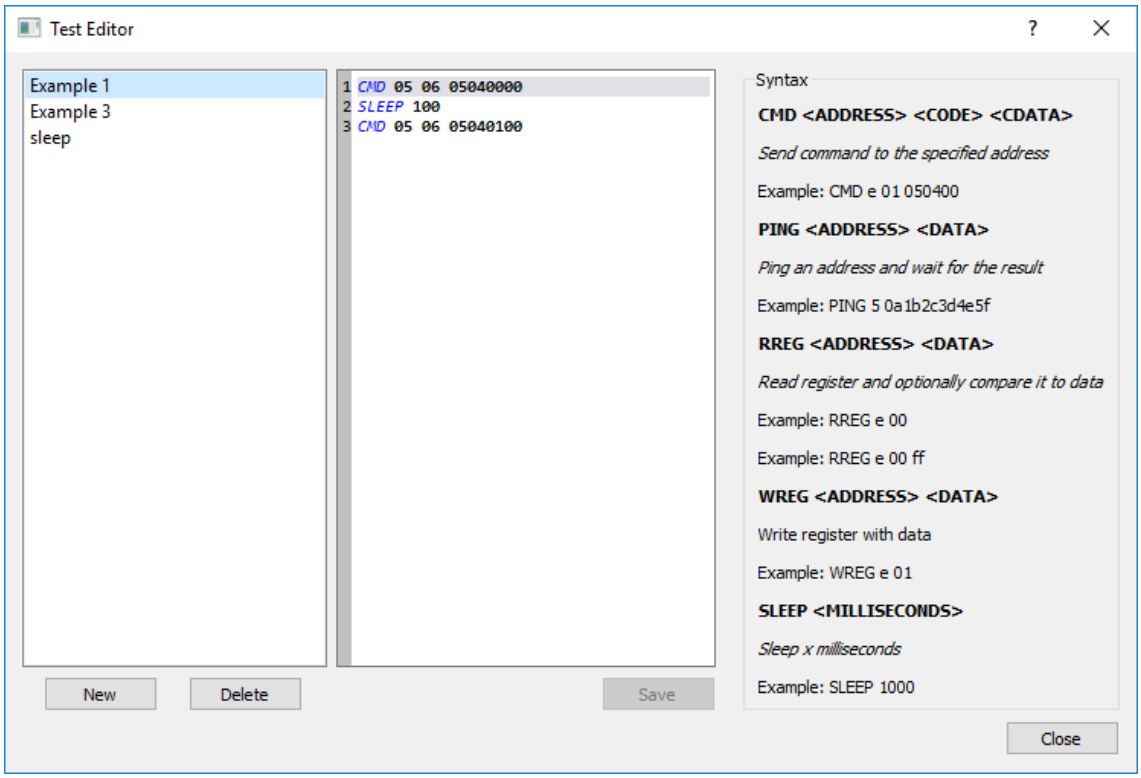
3. Sidemooduli testimine

Programmi põhiaknas on võimalik valida eelseatud ning käsitsi tehtud testide vahel. Kui test on valitud, tuleb seada testimisparameetrid:



- *Dump memory on failure* – loe kõik registri väärtused ning kuva need logisse ebaõnnestunud testi puhul
- *Log temperature* – salvesta UHF ja X-riba moodulite võimendi temperatuur CSV formaadis faili 5s intervalliga
- *Stop on failure* – peata järgnevad käsud pärast ebaõnnestumist. On oluline, et isegi kui see on valimata, jätab programm meelde, et käsk on ebaõnnestunud ning see kajastub tulemustes.
- *Loop count* – korduste arv

4. Uute testide loomine



Uusi teste on võimalik luua, vajutades nupul New. Kõik testid, mis hetkel on tarkvara poolt toetatud, on parempoolses tulbas esile toodud.

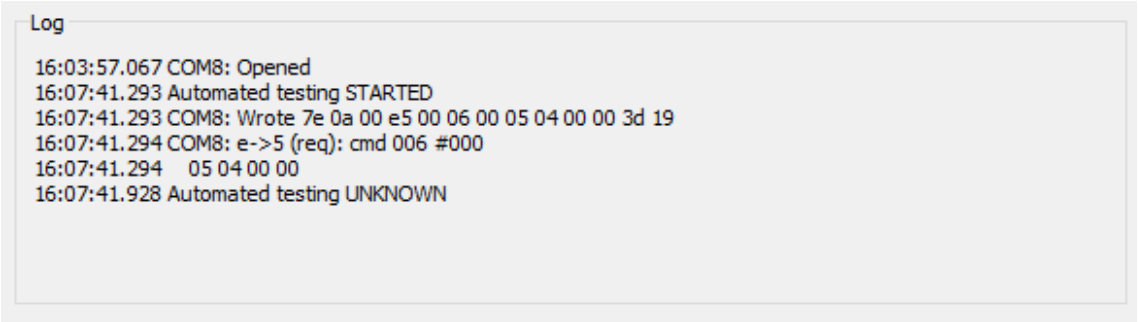
Programm loeb reahaaval käsusisendi ning parameetrid vastavalt süntaksile. Kui programm ei tunne käsku, jäetakse rida vahele. Kõik käsud, mis programm ära tunneb, värvitakse siniseks.

5. Testimistulemused

Testitulemused on esindatud programmi enda logis, kui ka failina salvestatult.

Logifail salvestatakse järgmiselt: TestResults/PP_KK_AA/TT_MM_SS.log.txt

Juhul kui on sisse lülitatud temperatuurilogimine, on salvestatud ka CSV formaadis temperatuur: TestResults/PP_KK_AA/TT_MM_SS.temp.csv.


Log
16:03:57.067 COM8: Opened
16:07:41.293 Automated testing STARTED
16:07:41.293 COM8: Wrote 7e 0a 00 e5 00 06 00 05 04 00 00 3d 19
16:07:41.294 COM8: e->5 (req): cmd 006 #000
16:07:41.294 05 04 00 00
16:07:41.928 Automated testing UNKNOWN