TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Thomas Johann Seebecki elektroonikainstituut

IEE40LT

Karl Schasmin 104215

HAJUTATUD ANDMEHÕIVESÜSTEEMI JUHTKONTROLLER RASPBERRY PI BAASIL

Bakalaureusetöö

Juhendaja: Eero Haldre
vaneminsener

Tallinn 2016

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: Karl Schasmin

26.05.2016

Annotatsioon

Bakalaureuse lõputööks on hajutatud andmehõivesüsteemi juhtmoodul, mis on arendatud Raspberry Pi platvormile. Juhtkontrolleri ülesandeks on mõõtemoodulite töö juhtimine eksperimendi jooksul ning mõõtetulemuste kogumine. Andmehõivesüsteem koosneb juhtmoodulist ning kuni 31 mõõtemoodulist. Kontrollerite kommunikatsioon toimub RS485 interfeisi kaudu.

Mõõtmiste sünkroniseerimist tehakse sidekanali kaudu spetsiaalset sidepaketti kasutades. Andmed salvestatakse ringpuhvrisse ning tehakse kättesaadavaks veebiserveris. Tehtav töö kooskõlastatakse teiste süsteemi osade loojatega.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 39 leheküljel, 7 peatükki, 6 joonist, 9 tabelit.

Abstract

Distributed Data Acquisition System's Control Module Based on Raspberry Pi

The object of this thesis is a control module for a distributed data acquisition system. The control module is based on Raspberry Pi computer platform. Control module will be responsible for synchronization of up to 31 measurement module's work. Communication between network nodes will be based on serial RS485 standard.

A communication protocol must be developed to ensure synchronized measurements and completeness of data. Data will be saved to a circular buffer and made available on a web server. This web server will also be used as a user interface for the control module.

The development of the control module will be accordance to the work from other authors working on this data acquisition system, mainly with the author of measurement modules.

The thesis is in Estonian and contains 39 pages of text, 7 chapters, 6 figures, 9 tables.

Lühendite ja mõistete sõnastik

ADC	*Analog Digital Converter*, analoog-digitaalmuundur
ARM	*Advanced RISC Machine,* arvutiarhitektuur
Etherner	Juhtmega kohtvõrk
GPIO	*General Purpose Input/Output*, üldiseks kasutamiseks mõeldud kontaktid
Linux	Vabavaraline Unixi laadne operatsioonisüsteem
Micro-USB	USB konnektor
RAM	*Random Access Memory* ehk suvapöördusmälu
SD	*Secure Digital,* mälukaardi formaat
SoC	*System on a Chip*, ühte kiipi mahutatud terviklik süsteem
UART	*Universal Asynchronous Receiver/Transmitter* ehk universaalne asünkroontransiiver
USB	*Universal Serial Bus*, universaalne järjestik andmesiin

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Hajutatud andmehõivesüsteemi jaoks on tarvis süsteemi juhtivat kontrollerit, mis suhtleb mõõtemoodulitega. Juhtmooduli ülesanne on eksperimendis vajalike teadete ja sünkrosignaalide saatmine mõõtekontrolleritele, andmete kogumine ja salvestamine ringpuhvrisse ning eksperimendi andmete paketi kättesaadavaks tegemine eksperdimendi tegijale.

Eesmärgiks on kuni 31 mõõtemoodulit juhtiva kontrolleri tarkvara ning juhtkontolleri sideahela väljatöötamine. Mõõtemoodulite sünkroniseerimist silmas pidades on välja töötatud sideprotokoll. Juhtmoodul on ehitatud populaarse Raspberry Pi 2 Model B platvormi baasil. Valmis tarkvara on hõlpsasti muudetav vastavalt eksperimendi parameetritele ning töötab ka varasematel Raspberry Pi mudelitel. Juhtmoodul kasutab sideks mõõtemoodulitega RS485 standardit. Töö käigus loodud süsteem on testitud ning dokumenteeritud süsteemi kasutajat silmas pidades.

Käesoleva töö esimeses peatükis antakse ülevaade andmehõivest ning teises peatükis tutvutatakse täpsemalt juhtmooduli raudvara ning tarkvara teeke. Kolmandas peatükis käsitletakse kommunikatsiooni liidest. Neljas peatükk on pühendatud juhtseadme tarkvara ja käitumise kirjeldamiseks ning viiendas peatükis käsitletakse veebiliidest ja andmete kättesaadavaks tegemist eksperimendi tegijale.

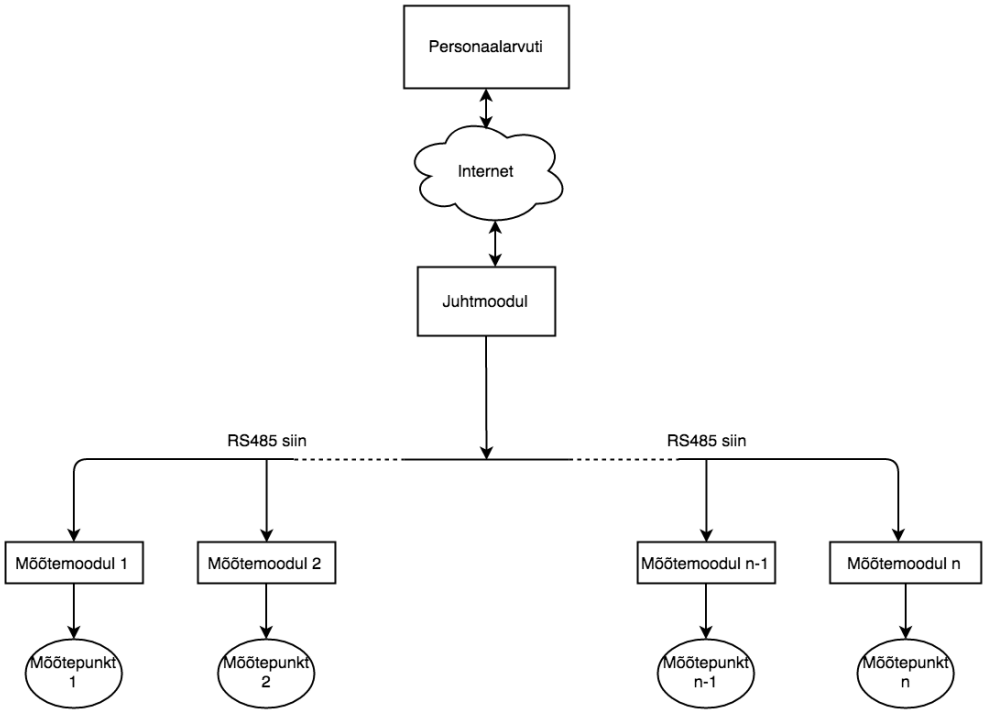
2 Andmehõive

2.1 Süsteemi tutvustus

Andmehõive on protsess, mille käigus füüsiliste suuruste mõõtmistest saadud tulemused teisendatakse digitaalsele kujule, et neid saaks arvutiga käidelda. Andmehõive algab seega füüsikalisest nähtusest või -omadusest, mida mõõdetakse [1]. Üldjuhul muudetakse mõõtmisest saadud analoog lainekuju digitaalseteks suurusteks. Füüsikalise suuruse mõõtmine ja teisendamine digitaalsele kujule on sensorite töö. Olenevalt sensori tüübist võib see signaal olla takistuse, pinge, voolu või mõne muu elektrilise omadusena. Analoog-digitaalmuundur on seade, mis väljastab analoogsignaali digitaalset kuju ühel ajahetkel.

Andmehõivesüsteem koosneb nii riist- kui ka tarkvara osadest. Teiste mõõtmissüsteemidega võrreldes kasutab andmehõivesüsteem ära arvutite töötlusvõimsust, ühendusvõimekust, produktiivsust ja andmete kuvamise võimalust [1]. Kõnealuses süsteemis on mõõtemooduliteks ARM põhised kontrollerid EK-TM4C123GXL [13]. Diagramm hajutatud andmehõivesüsteemi kohta on välja toodud Joonises 1.

Andmehõive süsteeme kasutatakse näiteks teadusuuringutes ja -analüüsides, kontrollsüsteemi osana otsuste tegemises, süsteemide monitooringus, diganostikas, ning muus säärases.



Joonis 1: Hajutatud andmehõivesüsteem

2.2 Juhtmoodul andmehõivesüsteemi osana

Juhtmooduli ülesanne on mõõtmisprotsessi juhtimine, mõõtemoodulite töö sünkroniseerimine, andmete kogumine ning mõõtemoodulitelt saadud andmete kättesaadavaks tegemine eksperimendi jooksutajale. Tarkvara on eksperimendi tüübi suhtes kohandatav ning universaalne. Juhtkontroller suudab suhelda vähemalt 31 mõõtemooduliga, kasutades selleks järjestiksidet. Juhtmooduli tööd detailsemalt kirjeldame viiendas peaktükis.

3 Juhtkontroller

3.1 Juhtkontrolleri riistvara

3.1.1 Raspberry Pi

Raspberry Pi on umbes pangakaardi suurune arvuti, mida luues on silmas peetud nii noori kui vanemaid arvutitehnika huvilisi. Kui tänapäevaseid arvuteid on üha raskem lahti võtta (seda nii füüsiliselt, legaalselt kui ka materiaalselt) ning uurida kuidas nad töötavad, siis Raspberry arvutid on seda silmas pidades arendatud. Tootjalt tellides ei panda isegi plastikust ümbrist kaasa, klient saab vaid trükkplaadi.

Kasutada saab seda kõigeks, milleks kasutatakse personaalarvutit – interneti kasutamine, videote mängimine, tekstitöötlus, programmeerimine, jne. Raspberry arvutil on võimalus suhelda muu digitaalse maailmaga, kasutades selleks sisend-väljund kontakte. Lisada saab väliseid liideseid nagu näiteks kaamerad, ekraanid (ka puutetundlik), klaviatuurid, ja muu säärane.

Raspberry Pi sai juhtmooduliks valitud oma kättesaadavuse ning ulatusliku funktsionaalsuse tõttu. Raspberryle leidub väga palju vabavaralisi riist- ning tarkvara komponente ning tänu oma populaarsusele ning ühtsele kommuunile on saadaval suures koguses abimaterjale. Oma mõõtutelt on ta väike ning võib vajadusel paikneda kitsastes tingimustes mõõtemoodulite läheduses. Alternatiivid on enamasti kallimad või pole neil kogu Linuxi süsteemi jooksutamise võimekust.

3.1.2 Tehnilised andmed

Seni on erinevaid variante Raspberry Pi'st 8

- Pi 1 Model A
- Pi 1 Model A+
- Pi 1 Model B
- Pi 1 Model B+
- Pi 2 Model B
- Pi 3 Model B
- Pi Zero
- Compute Module

Käesoleva töö autor kasutab arendamiseks Pi 2 Model B varianti [2]. Pi 3 avaldati alles 2016 aasta alguses. Pi 2 keskseks elemendiks on Broadcomi SoC (System on Chip) nimega BCM2836 [4], kus on ühes kiibis koos nii protsessor kui ka videoprotsessor. Raspberry Pi 2 Model B andmed on järgnevad:

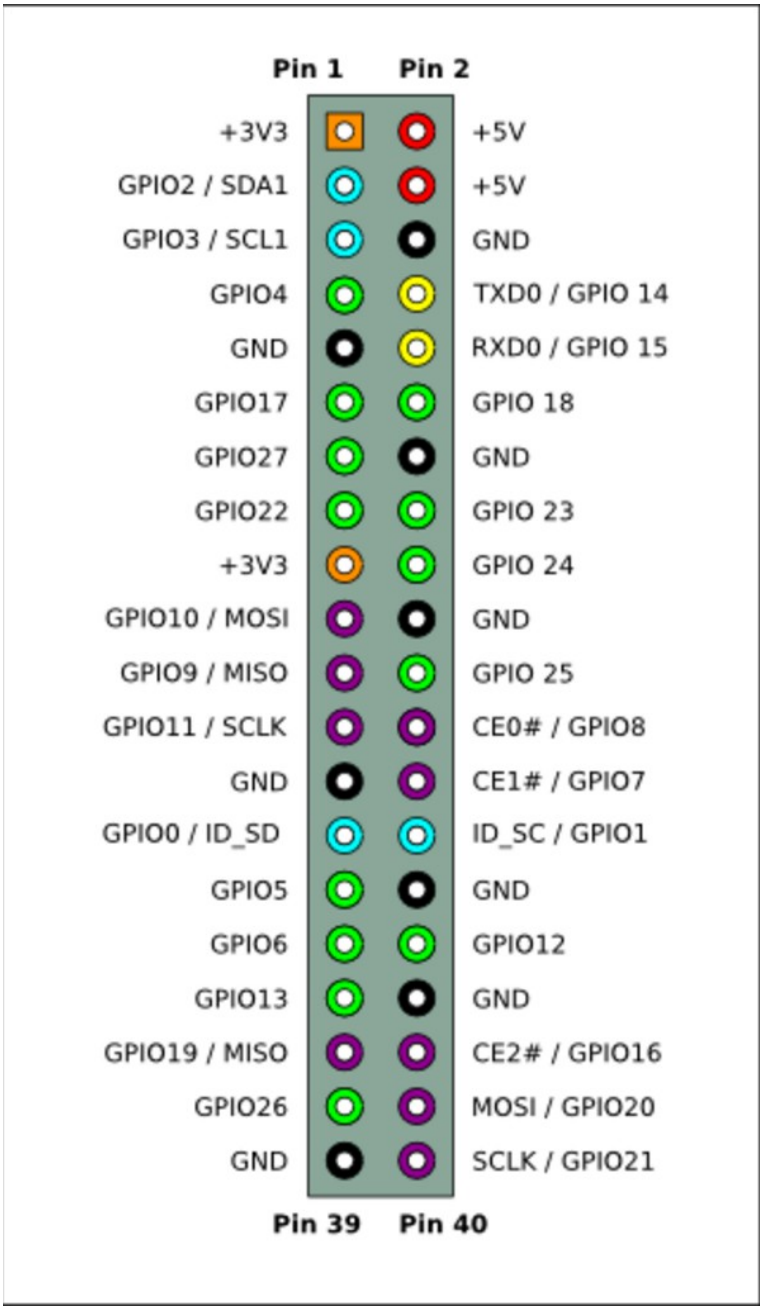
- 900 Mhz, neljatuumaline ARM Cortex-A7 arhitektuuril protsessor
- 1 gigabait muutmälu
- 4 USB porti
- 40 üldotstarbelist sisend-väljund viiku
- Ethernet port
- Mõõdud 85.60mm x 56mm x 21mm
- Kaal 45 grammi

3.1.3 GPIO

GPIO päis annab võimalused, et Rasperry saaks juurde lisafunktsioone ning suhtleks muu maailmaga. Raspberryl on olenevalt versioonist 26-40 sisend-väljund kontakti [8].

Meie kasutame kontakte 4 ja 6 sidekanali toite ja maandusena, kontakte 8 ja 10 sisendviikudena jadaühenduse jaoks ning kontakti 18 väljundviiguna, et lülitada juhtkontroller saatmisolekust kuulamisolekusse ning vastupidi.

Kontaktide päis on jaotatud kolmeks blokiks. Iga blokk on varustatud 3.3V pingega. Kõrgema pingega ühendus hävitaks suure tõenäosusega keskse süsteemikiibi GPIO bloki. Joonisel 2 näidatud GPIO kontaktide asetus ja funktsioonid.



Joonis 2: GPIO kontaktid [8]

3.1.4 Toide

Juhtmoodul vajab toiteks 5V ja 700mA. Toiteallikas peab olema võimalikult kvaliteetne, kuna vähese voolu korral hakkab juhtmoodul ennast taaskäivitama. Juhtmooduli toide toimub USB seinaadapterit kasutades, ühendades selle Raspberry micro-USB pessa.

Raspberryt on võimalik toita ka läbi GPIO kontaktide, kuid toitallikas tuleks olla veel kindlam. Nimelt, kui micro-USB kaudu toites on süsteemi disainitud ülepinge kaitsed, siis GPIO kontaktidel on otseühendus protsessoriga. Seega ebakvaliteetne sisendpinge võiks kogu mooduli kasutuks muuta.

3.2 Kasutatav programmvara

3.2.1 Operatsioonisüsteem

Raspberry jooksutab operatsioonisüsteemina enamusi Linuxi distributsioone, millest kõige populaarsem on spetsiaalselt Raspberryle kohandatud ja optimeeritud Raspbian [11]. Hiljuti tekkis ka võimalus kasutada Windows 10 IoT Core'i, mis on minimaalne Windows 10 versioon, mõeldud Asjade Interneti seadmete arendamiseks [12]. Operatsiooni süsteemi jooksutatakse SD kaardilt.

Juhtmooduli jaoks kasutame Raspberry Pi jaoks mõeldud Raspbiani [11].

Populaarseim ning ametlikum programmeerimiskeel Raspberryl, mille kohta kõige rohkem materjale leiab, on Python [9]. Vaikimisi on installitud ka C, C++, Java, Scratch, ja Ruby. Kasutada saab kõiki keeli, mis kompileeruvad ARMv6 (Pi 1) või ARMv7 (Pi 2) arhitektuurile.

3.2.2 Python

Juhtmooduli programm on implementeeritud Pythoni programmeerimiskeeles. Python on Raspberry kommuuni poolt enim kasutatav programmeerimiskeel, mis tähendab, et selle kohta on palju abimaterjale [9].

Python on üsna algajasõbralik keel, kuna ta süntaks on väga verboosne ja lihtsasti mõistetav. Tegemist on interpreteeritava kõrgtaseme keelega, mis on dünaamiliste andmetüüpidega. See teeb Pythoni aeglasemaks kui madalama taseme keeled nagu C või C++. Kuna tahame, et juhtmoodul oleks võimalikult universaalne ning seadistatav, siis kasutame prototüübi faasis Pythonit, kuna see on lihtsasti mõistetav ja eksperimendi jooksutaja poolt hoomatavam. Kuna tegemist on esialgse prototüübiga, võime madala taseme mäluhalduse ja kiiruse ohverdada, et hoida kokku arendusaega ja -skoopi.

3.2.3 PySerial

Jadaühenduse implementeerimiseks on kasutuses Pythoni tarkvarateek nimega PySerial [3]. Seda kasutades tuleb enda projektile vastavalt seadistada Pythoni objekt, mis siis initsialiseeritakse PySeriali konstruktori poolt.

Peamised parameetrid mida kasutatakse on seletatud järgnevas loetelus. Mõnede parameetrite puhul on oodatav tüüp Pythoni andmetüüp, teisel juhul PySerial teegi deklareeritud enumite hulgast, nagu näiteks Serial.PARITY_NONE [3].

- Port – seadme kasutatav suhtlusport, vaikimisi '/dev/ttyAMA0'
- Baudrate – baudrate ehk modulatsioonikiirus, vaikimisi 115200.
- Bytesize – andmebitide arv
- Parity – samaväärsuse kontrollimine
- Stopbits – stop bitide arv
- Timeout - kui kaua oodatakse andmeid lugemise puhul
- write_timeout – kui kaua oodatakse kirjutamise puhul

4 Kommunikatsioon mõõtemoodulitega

Juhtmoodili ja mõõtemoodulite vahel kasutatakse kommunikatsiooniks jadasidet. Juhtkontrollerill on selleks olemas UART liides. Jadaside puhul saadetakse andmeid ühe biti kaupa järjest üle sidekanali. Järjestikside puhul ei pea enne lugemist või saatmist sõnumiga tegema lisatehteid, nagu seda on tarvis teha paralleelside puhul. Järjestikside vajab vähem kaableid ning esineb vähem müra, mis paralleelside puhul esineb näiteks crosstalki näol [5].

4.1 Kommunikatsiooni seadistus

RS-485 standardit saab edukalt kasutada elektriliselt mürarikastes kohtades ning üle pika distantsi. Võrku võib olla ühendatud kuni 32 liiget, lineaarses multi-drop seadistuses. Sellises seadistuses võime võrgu kiiruseks saavutada 35 Mbit/s 10 meetri puhul ning 100 kbit/s 1200 meetri puhul. Ühenduse kuni 1200 meetrini tagab diferentsiaalse signaali kasutamine ning keerdpaarjuhtmete kasutamine müra vähendamiseks [5].

Seadmed ühendatakse kahe juhtmega punktist-punkti ühendusega, ehk elemendid on ühendatud kommunikatsioonisiini külge. Soovitatav oleks algus- ja lõppseadme juures paigaldada sidejuhtmete vahele nn. lõpetamise takisti. Vastasel juhul võib sideliini sattuda peegeldumistest tingitud müra. Keerdpaarjuhtme puhul sobib tavaliselt 120 oomine takisti.

Suhtlusel kasutatakse ülem-alam (master-slave) suhtluse põhimõtteid. See tähendab, et ülema rolli täitev seade initsialiseerib suhtluse ja koordineerib seda. Antud süsteemi puhul on ülem kõnealune juhtseade. Et saavutada maksimaalset sidesignaali tugevust ning kiirust, tasuks võrgutopoloogia mõttes panna ülemseade sideliini keskele, mitte liini lõppu. RS485 draiverina kasutatakse MAX485 mikrokiipi [6], mis tuleb ühendada vastavalt Tabel 1.

Tabel 1: MAX485 kontaktid [6]

Nimi	Kirjeldus
R0	Vastuvõtja väljund
RE	Vastuvõtja väljundi võimaldaja
DE	Ajaja väljundi võimaldaja
DI	Ajaja sisend
A	Mitteinverteeriv vastuvõtja
B	Inverteeriv vastuvõtja
Vcc	Toide
GND	Maa, Ground

4.2 Protokoll

Suhtluseks lepime kokku sideprotokolli, et seadmed teaksid, mis kujul sõnumeid oodata.

4.2.1 Nõuded sideprotokollile

- Juhtkontroller saab suhelda ühe mõõtemooduliga korraga. Seadmeid võib võrgus olla kuni 32.
- Juhtkontroller saab saata sõnumit kõigile seadmetele korraga.
- Sõnumite kontrolliks kasutame CRC algoritmi
- Paketi pikkus 4 baiti
- Suhtlus juhtkontrollerist mõõtemooduli suunal on kindla sisuga
- Suhtlus mõõtemoodulist juhtkontrolleri suunal on muutuva sisuga

4.3 Protokolli disain

4.3.1 Ühe paketi sisu

Tabel 2: Ühe paketi sisu

Bait	Kirjeldus
1	Saaja aadress, unikaalne adressaadi ID
2	Sõnumi andmed
3	Sõnumi andmed
4	CRC kood

4.3.2 Esimese baidi sisu

Tabel 3: Esimese baidi sisu

Biti järk	Kirjeldus
0-4	Unikaalne ID
5-7	Ei kasutata

Võimalikud ID numbrid

Tabel 4: ID number

ID	Kirjeldus
0	Null bait, paketi sisu mõeldud kõigile seadmetele
1-31	Unikaalne ID, mis on omane ühele kindlale võrgus olevale seadmele

4.3.3 Juhtkontrolleri saadetavad käsud

Tabel 5: Juhtkontrolleri saadetavad käsud

2. bait	Kirjeldus	3.bait	Kirjeldus
0	Ei kasutata.	0-255	Ei kasutata.
1	Alustada eksperimendiga: initsialiseeri mõõtemoodulid	0-255	Ei kasutata.
2	Alustada mõõtmisega	0-255	Ei kasutata.
3	Ergutussignaali genereerimise algus	0-255	Ei kasutata.
4	Ergutussignaali genereerimise lõpp	0-255	Ei kasutata.
5	Saata mõõtetulemused juhtkontrollerile.	0-255	Ei kasutata.
6	Kustuta mõõtetulemused.	0-255	Ei kasutata.
7-254	Ei kasutata.	0-255	Ei kasutata.
255	Staatuse päring.	0-255	Ei kasutata.

4.3.4 Mõõtemooduli saadetavad andmed

Tabel 6: Mõõtemooduli saadetavad andmed

2. bait	3. bait	Kirjeldus
0-255	0-255	Mõõtetulemused

4.3.5 Tulemuste formaat

Kuna mõõtetöö tulemused on 12-bitises formaadis, siis mõõtetulemused tagastatakse kahes osas, mis on jagatud 2. ja 3. baidi vahel. Kaks suuremat järku biti baidis annavad andmete kohta metaandmed, mis on täpsustatud järgnevas tabelis.

Tabel 7: Mõõteandmete metaandmed

Bitid	Kirjeldus
00** ****	Andmed on lõplikud, kogu info on selles samas paketis
01** ****	Andmete algus, esimene osa saadetavast infost
10** ****	Andmed jätkuvad, vahepealne osa saadetavast infost
11** ****	Andmete lõpp, viimane osa saadetavast infost

4.3.6 Staatuse vastus

Juhul kui juhtmoodul küsib mõõtemoodulilt staatust, on oodatav vastus järnevas formaadis.

Tabel 8: Staatuse formaat

1. bait	2.bait	3.bait	4.bait
Mõõtemooduli ID	Staatuse päringu bait (255)	Staatuse kood	CRC

Tabel 9: Staatuse küsimuse vastus

Staatuse bit	Kirjeldus
0000 0001	Valmis mõõtetööga alustama: ergutussignaal, mõõtmine või mõlemad
0000 0010	Mõõtetulemused olemas
0000 0100	Mõõtetulemused saadetud
1000 0000	Viimane päring oli vigane

4.4 CRC

CRC ehk cyclic redundancy check kasutatakse paketi terviklikkuse tuvastamiseks. Andmete peal kasutatakse mõnda kokkulepitud matemaatilist funktsiooni, ning tulemus pannakse paketile kaasa. Paketi vastu võttev moodul saab teha sama tehte, ning kontrollida tulemust CRC baidiga võrreldes. Antud süsteemis arvutatakse esimese kolme baidi põhjal ühebaidine CRC.

Iga baidi arvutamisel, antakse arvutusele kaasa varasemalt arvutatud CRC ning vastav bait. Esimese baidi arvutamisel on kaasa antav CRC null. CRC arvutamisel nihutatakse baidi sisu vasakule tsüklis kaheksa korda, ning juhul kui kontroll biti väärtus enne nihutamist oli 1, siis pärast nihutamist inverteeritakse vastavad bitid.

5 Juhtprogrammi töö

5.1 Juhtseadme töö

Andmehõivesüsteemi juhtseadme töö on süsteemi protsesside juhtimine ajas. Suures plaanis saadab juhtseade ainult käske ning protsessi lõpus kogub kokku andmed. Järgnevalt räägime täpsemalt protsessi juhtivatest algoritmidest, vajalikest parameetritest ja muust taolisest.

Tarkvara jooksutamiseks on vaja teada esiteks seda, kas tahetakse teha üks või mitu mõõtetsüklit. Kontrollitakse kas mõõtmiseks kasutatakse kõiki mõõtemooduleid, ainult osa nendest või ainult ühte moodulit. Vaikimisi kasutatakse kõiki ühendatud mõõtemooduleid. Sisendina saab seadistada, mitu viimast mõõtetsüklit andmeid hoitakse korraga failisüsteemis salvestatuna.

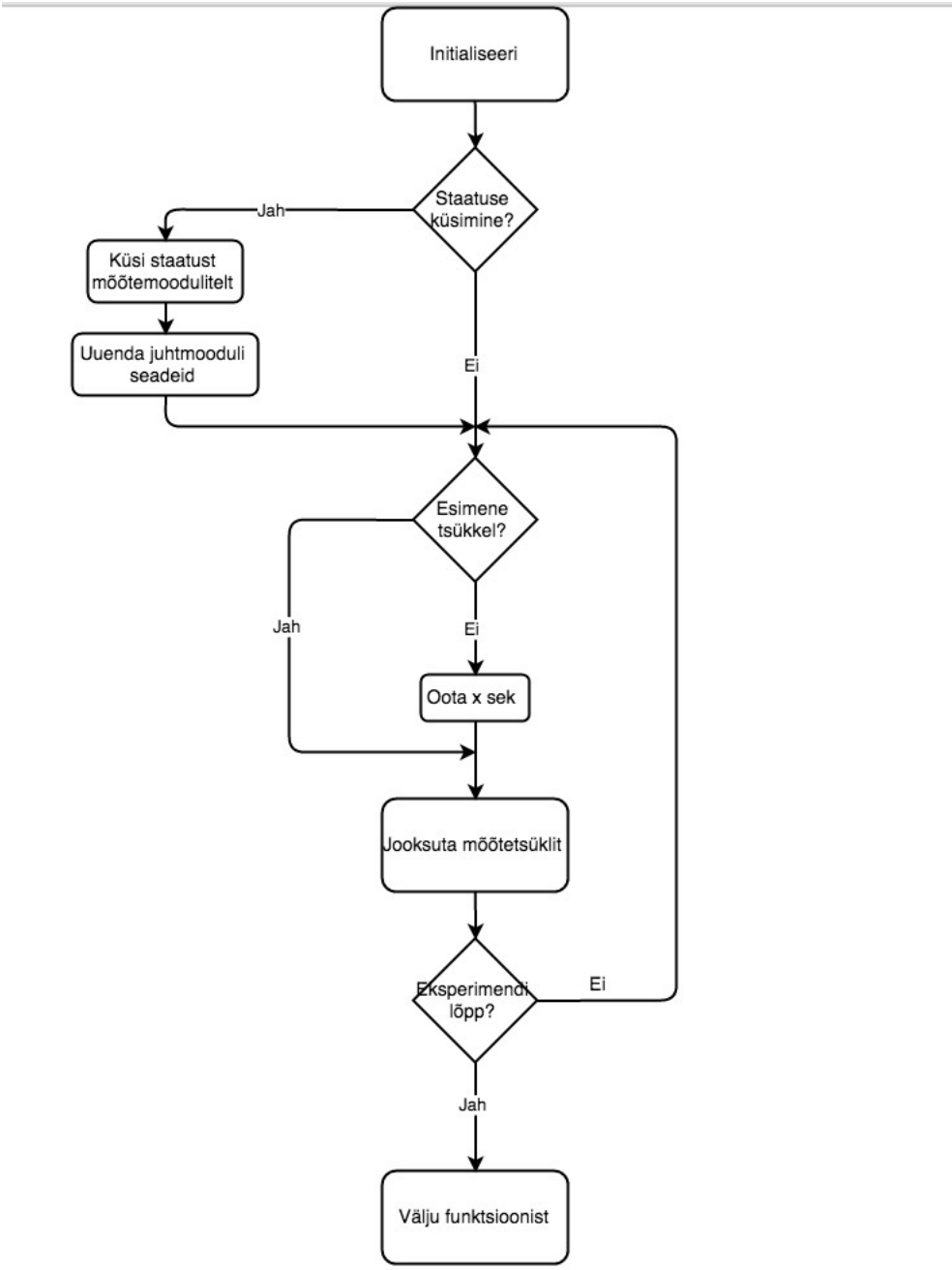
5.2 Eksperimendi algoritm

Eksperimendi juhtimise loogika on järgmine. Enne eksperimenti tehakse staatuse küsimine liinilt vastava järjekorraga.

1. Initsialiseeritakse seaded ning ühendused
2. Juhul kui seadmel pole manuaalselt kontrollerite aadresseid seadistatud, toimub pärast initsialiseerimist liinilt staatuse küsimine
3. Juhtseade lülitatakse saaterežiimi, saadetakse kõigile staatuse küsimise käsk ning lülitutakse tagasi kuulamisrežhiimi
4. Tehakse läbi maksimaalse seadmete arvuga tsükkel, kus iga tsükli sammu korral oodatakse ühe mõõtemooduli vastust oma staatusega
5. Kui viimased 3 sammu on olnud vastuseta, eeldatakse, et rohkem seadmeid liinil ei ole ning lõpetatakse tsükkel.

Pärast seda ollakse eksperimendiks valmis. Enne mõõtetsükli jookustamist kontrollitakse veel, kas tegu on esimese mõõtetsükliga. Juhul, kui tegu on esimese tsükliga, minnakse otse mõõtetsükli väljakutsumise juurde. Vastasel juhul käivitub timeout funktsioon, ehk oodatakse selle aja jagu, mis eksperimendi jooksutaja on määranud mõõtetsüklite vahele. Mõõtetsükli algoritmist täpsemalt järgmises paragrahvis ja Joonisel 4.

Pärast mõõtetsüklit kontrollitakse, kas see oli viimane mõõtetsükkel või tuleb neid veel. Kui tuleb veel, siis minnakse algusesse tagasi, punkti kus kontrollitakse, kas tegu on esimese tükliga ning minnakse timeout funktsiooni. Kui tegu oli viimase mõõtetsükliga, kutsutakse välja dekonstruktor ning väljutakse eksperimendi funktsioonist.



Joonis 3: Eksperimendi algoritm

5.3 Mõõtetsükli algoritm

Mõõtetsükli ajal kästakse mõõtemoodulitel kas ühe, mitme või kõigi kaupa alustada mõõtmisi ning oodatakse siis tulemusi. Kuna initsialiseerimised on tehtud siis võib kohe minna saatmisrežhiimi ning mõõtmiskäsu välja saata. Pärast käsu saatmist võime kohe lülituda kuulamisrežiimi, kuna rohkem käske pole ning peame tulemusi kuulama.

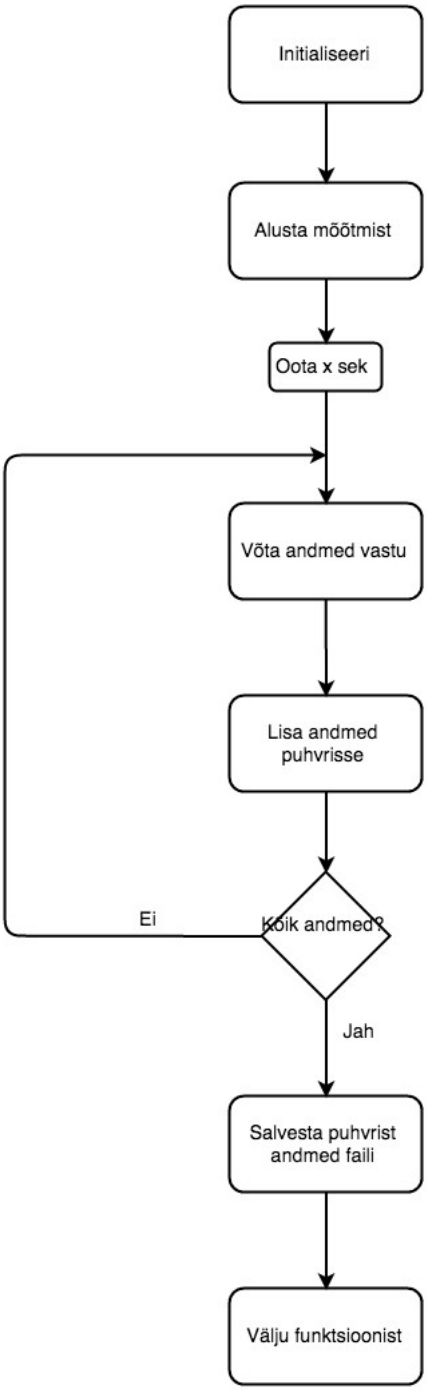
1. Pärast käsu saatmist minnakse timeouti, selle pikkus on eksperimendi jooksutaja poolt seatud, teades kaua mõõtemoodulitel mõõtmiseks ning andmete saatmiseks aega läheb.

2. Pärast timeouti sisenetakse tsüklisse, kus iga tsükli ajal saame ühelt mõõtemoodulilt andmed nind kirjutame need ringpuhvrisse.

3. Kontrollime, kas tegu oli viimase mõõtemooduliga või ei. Kui ei, siis lähme uuele tsükli sammule. Kui oli tegu viimase sammuga, siis kirjutame puhvrist andmed faili ja salvestame failisüsteemi.

4. Kutsume välja dekonstruktori ja väljume funktsioonist.

Täpsemalt mõõtetsükli algoritmi kohta Joonisel 4.



Joonis 4: Mõõtetsükli algoritm

5.4 Juhtseadme mälumooduli optimeerimine

Juhtseadme operatsioonisüsteemi jooksutamiseks ning failide salvestuseks kasutatakse välkmälu, täpsemalt microSD kaarti. Välkmälu miinuseks on piiratud mällu kirjutamiste arv ja mälu eluiga. See on olenevalt mudelist alates 10000 kuni 200000 kirjutamis-kustutamis tsüklit. Olenevalt eksperimendi iseloomust võime me mõnel juhul teha suure hulga kirjutamisi üsna pika aja jooksul, seega peame me arvestama mälukaardi elueaga [7].

5.4.1 Mälukaardi valik

Enne Raspberry töökorda seadmist, peab sellele valima mälukaardi. Suuremad kaardid peavad ajaliselt kauem vastu. Mida rohkem on vaba ruumi kaardil, seda vähem see kaarti koormab, kuna mälu kontrolleri tasemel jaotatakse kirjutamise koormus mööda kaarti erinevatesse mäluelementidesse. Kasutades 16 GB kaarti 8GB asemel pikendab mälu eluiga potensiaalselt kahekordselt, kuna valikus olevaid mäluelemente on rohkem. Võimalusel tasub kasutada tuntud tootja kaarti, kuna koormuse jaotamine ei ole SD kaartide standard. Enamus tuntumaid firmasid kasutab seda praktikat, kuid mõni väiketootja seda teha ei pruugi, millest tulenebki kaardi hind.

5.4.2 Linuxi seadistusest tulenev optimeerimine

Linuxi seadistusega võib samuti saavutatada mälu eluea optimeerimise. Maksimaalselt tasub kasutada ära süsteemi muutmälu. Kuna RAM'il tema mälu olemuselt puuduvad kirjutamise eluea piirangud, tasub mõned süsteemi seaded seda arvestades ringi teha. Soovitatav on muuhulgas

- operatsioonisüsteemi tasemel tehtav logimine (/var/log, /var/run, ja teised taolised) suunata SD kaardilt RAM'i
- swapi e. vahetuse kasutamine seadetest kinni panna
- kogu failisüsteemile anda vaikimisi vaid lugemise õigused
- SD kaarti regulaarselt varundada

Kuna süsteem logib ise üsna palju infot, siis selle informatsiooni kirjutamise suunamine RAM'i vähendaks kõvasti kirjutustsüklite arvu.

Võimalus on swapi e. vahetuse kasutamine seadetest kinni panna [7]. Swap allokeerib mingi osa failisüsteemist selleks puhuks kui RAM saab täis. Sellel juhul liigutatakse mitte aktiivselt kasutuses olevad mälu andmed mõneks ajaks swap'i. Selle optimeerimise kasutamisel tuleb arvestada, et kui RAM saab tihti täis ning siis ei saa osa koormust RAM'ilt ära anda failisüsteemile, võib süsteem muutuda väga aeglaseks.

Kõige ekstreemsem variant oleks kogu failisüsteemile anda vaikimisi vaid lugemise õigused. See aga teeb kasutamise väga ebamugavaks ning ei sobi enamuste projektidega. Sellisel juhul peaks alati väga täpselt teadma kunas kirjutamisfunkstsioone kasutatakse ning et see oleks õigel ajal ka lubatud.

Muutmälu kasutades tuleb arvestada, et voolu kadudes kaovad ka seal olevad andmed. Seega peab garanteerima, et süsteemile vajalikud andmed salvestatakse õigel ajal püsivamasse sisemällu ning süsteemi toidet ei lõpetataks enne nende salvestamiste lõppu. See tähendaks alati korrektset süsteemi kinni panemist ning kvaliteetse toiteahela olemasolu, eelistatavalt UPS'i või akuga.

Kindlasti tuleks aegajalt SD kaart varundada, et juhul kui midagi juhtub, saab kiiresti süsteemi uuesti tööle ning eksperimenti ei tuleks suurt pausi.

5.4.3 Tarkvara tasemel optimeerimine

Programmi tasemel peame samuti minimeerima failisüsteemi kirjutamiste arvu. Seega ei tasu iga mõõtemooduli ühe mõõtetsükli andmeid eraldi failiks kirjutada. Kuigi andmete hulk on küll sama, kirjutame me igakord uuesti mällu uue faili asukoha.

Kasutame andmete lugemisel sidekanalist ringpuhvrit, kuhu kogume mõõtemoodulitelt saadud andmed. Kui eksperiment on läbi või küsitakse andmeid, siis kirjutame kogu andmetehulga failisüsteemi. Ringpuhvri eeliseks on asjaolu, et pole vaja allokeerida lõpmatu hulk mälu, vaid saab defineerida endale vajaliku suurusega buffri ning seal kirjutatakse mälu täis saades ühe kaupa vanad kirjed üle uutega.

Väga suure säästmise tooks kaasa voogedastuse kasutamine, kuid see eeldaks spetsiaalsemat klient-serveri tarkvara ehitamist ning süsteemi teistsugust ülesehitust. Seljuhul peaksime süsteemi lisama ühe klienttarkvara jooksutava arvuti, mis kuulaks voogedastust ning salvestaks mõõteandmed kõvakettale või näiteks andmebaasi. Selleks võiks kasutada nii lokaalvõrgus asuvat arvutit või siis digitaalset serveri instantsi mõnelt teenusepakkujalt. See tähendaks, et voogedastusrezhiimis juhtmoodul ei peaks mälukaardile eksperimendi jooksul peaaegu midagi salvestama ning mälukaardi eluiga pikeneks oluliselt. Mõõtmoodulilt saadud tulemused loetaks mällu ning voogedastataks kliendile. Püsimällu ei salvestataks midagi.

5.4.4 Kokkuvõte

Eelnevad soovitused vajaks täiendavat uurimist, testimist ning andmete võrdlemist. Kõik optimeerimised on valikulised, kuid tasub kasutusele võtta vastavalt eksperimendi tüübile, selle skoobile ning eelarvele. Selleks võiks proovida erinevaid andmete ja logide salvestamise seadeid ning kirjutada testkood, millega seda lühema aja jooksul saab kontrollida. Kuna esialgu pole meil huvi SD kaardi limitatsioone koormata kuni selle hävinemiseni siis jääb mälu eluea testimine esimese iteratsiooni tegemisel skoobist välja.

6 Veebiliides

6.1 Tööriistad

Tulemuste kättesaadavaks tegemiseks jooksutatakse juhtmooduli peal veebiserverit. Veebiserveri loogika on implementeerinud Node.js keelega [10]. Node.js inerpreteerib Javascripti süntaksit, kasutades selleks Google poolt loodud Javascripti V8 nimelist Javascripti kompilaatorit. Server serveerib HTTP päringute peale klienttarkvara, mis on kirjutatud HTML'i, CSS'i ja Javascripti kasutades.

6.2 Kasutajaliides

Eelnimetatud tehnoloogiaid kasutades on loodud kasutajaliides, mis pakub loendi failisüsteemis olevatest eksperimendi andmetest, ning järjestab neid ajaliselt. Veebimoodulis jookseb ka tarkvara eksperimendi juhtimiseks, mõõtemoodulitele käskude saatmiseks ning parameetrite seadistamiseks. Selle kaudu saab saata mõõtemoodulitele eksperimendi alustamise käske või küsida viimaseid mõõtetulemusi, juhul kui need veel mõõtemooduli mälus alles on.

Veebiliides jaotatud neljaks osaks. Igaühe jaoks on olemas ka vastav alaleht.

- Staatuse leht
- Seadistuste leht
- Andmete leht
- Eksperimendi juhtimise leht

6.3 Juhtloogika

Kuna andmete kättesaadavaks tegemiseks jooksutame juhtmoodulis serverit, siis serveerime sealt ühtlasi ka kasutajaliidest. Kuna juhtmooduli tarkvara on kirjutatud Pythonis ja oma olemuselt on nad skriptid, siis kasutame ära Node.js võimalust luua alamprotsesse [10]. Kui saadame juhtpaneeli lehelt mõne käsu, siis luuakse Pythoni alamprotsess, kuhu antakse vajalikud parameetrid kaasa ning jäädakse vastust ootama.

6.3.1 Staatuse leht

Staatuse lehel on näha infot süsteemi kohta. Kas hetkel jookseb mõni eksperiment, kaugel ollakse eksperimendiga, mitu mõõteseadet on süsteemis, jne. Joonisena toodud Joonis 6.

6.3.2 Seadistuste leht

Seadistuse lehel saab eksperimendi ja seadme parameetreid muuta. Juhul kui eksperiment käib, siis kõik seaded ei rakendu enne, kui eksperiment on lõppenud.

6.3.3 Andmete leht

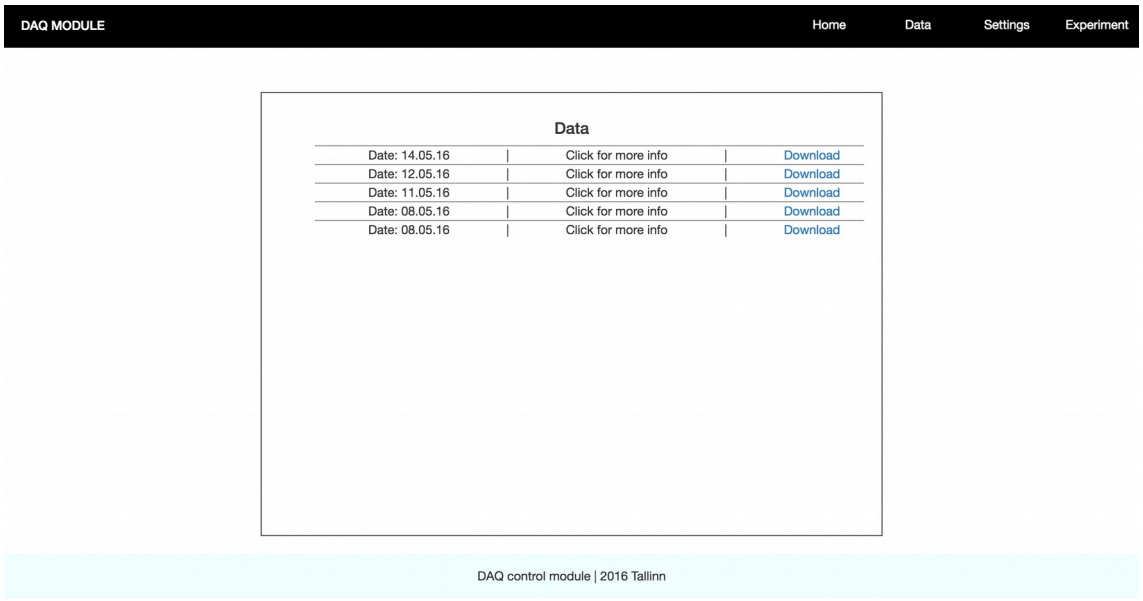
Andmete lehel on näha nimekirja viimastest andmetest ning ajaline info nende kohta. Saab muuta parameetrit, mitme viimase eksperimendi andmed salvestatakse. Näitena toodud Joonis 5.

6.3.4 Eksperimendi juhtimise leht

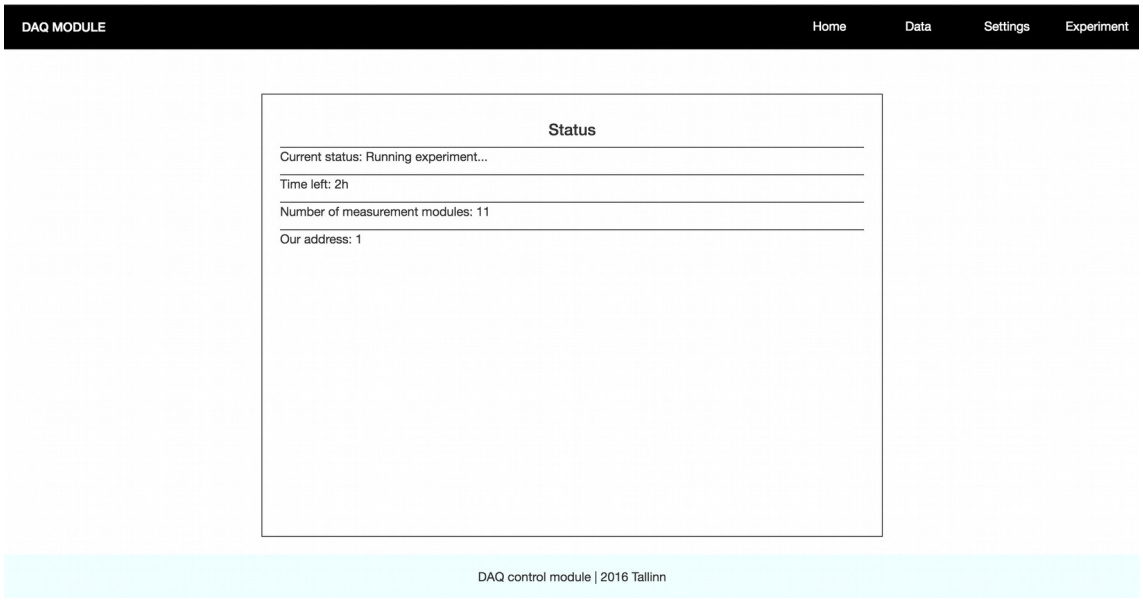
Eksperimendi lehel saab saata käske mõõtemoodulitele. Saab alustada mõõtmisi, küsida viimaseid aindmeid uuesti, küsida seadete staatust, jne. Kasutajaliides eksperimendi juhtimiseks.

6.4 Turvalisus

Serverit jooksutame vaikimisi kohalikus võrgus. Serveri kättesaadavaks tegemine välisest võrgust ning turvalisuse tagamine jääb eksperimendi tegija hoolde. Selline seadistamine ei erine tavalise Linuxi serveri seadistamisest.



Joonis 5: Andmete leht



Joonis 6: Staatuse leht

7 Kokkuvõte

Lõputöö tulemusena valmis hajutatud andmehõivesüteemi juhtmooduli programmvara. Töötati välja kommunikatsiooni protokoll ning juhtseadme veebiliides. Juhtkontrolleri eluea pikendamiseks analüüsiti võimalikke lahendusi, kuidas mälukaardi eluiga pikendada ning viidi mälu säästvad muudatused sisse ka juhtmooduli loogikasse. Juhtseadme riistvara jäi prototüübi tasemele.

Sidekanali töö ja mõõtetöö sünkroniseerimine testiti koos süsteemi mõõtemooduli autoriga, kasutades kahte EK-TM4C123GXL kontrollerit. Tehtud töö ja tarkvara dokumenteeriti.

Edasiarenduse puhul tasuks järgmises iteratsioonis tuua mõõtetöö täpsemaks sünkroniseerimiseks süsteemi juurde spetsiaalne sünkrojuhe. Toiteahelasse oleks tarvis akut, et voolu kadumisel lõpetaks süsteem mõõtetöö. Analüüsida tuleks mälukaardi eluiga koormustestidega ning vajadusel viia sisse parandavad seadistused.

Kasutatud kirjandus

[1] National Instruments, What Is Data Acquisition? [WWW] http://www.ni.com/data-acquisition/what-is/ (01.05.2016)

[2] Raspberry Pi Foundation, Raspberry Pi Model 2 [WWW], https://www.raspberrypi.org/products/raspberry-pi-2-model-b/ (01.05.2016)

[3] Chris Liechti, PySerial documentation [WWW] https://pythonhosted.org/pyserial (01.05.2016)

[4] Raspberry Pi Foudation, BCM2836 [WWW] https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2836/README.md (01.05.2016)

[5] Maxim Integrated, Guidelines for Proper Wiring of an RS-485 (TIA/EIA-485-A) Network [WWW] https://www.maximintegrated.com/en/app-notes/index.mvp/id/763 (19.05.2001)

[6] Maxim Integrate, MAX485 documentation [WWW] http://datasheets.maximintegrated.com/en/ds/MAX1487-MAX491.pdf (01.09.2014)

[7] Stack Exchange Inc, How can I extend the life of my SD card? [WWW] http://raspberrypi.stackexchange.com/questions/169/how-can-i-extend-the-life-of-my-sd-card (27.09.2014)

[8] Embedded Linux Wiki, RPi Low-level peripherals [WWW] http://elinux.org/RPi_Low-level_peripherals (03.07.2015)

[9] Python Software Foundation, Python Documentation [WWW] https://docs.python.org (26.05.2016)

[10] Node.js Foundation, Node.js documentation [WWW] https://nodejs.org/en/docs/ (26.05.2016)

[11] Raspberry Pi Foundation, Raspbian [WWW], https://www.raspbian.org (01.05.2016)

[12] Microsoft, Windows IoT Core [WWW] https://developer.microsoft.com/en-us/windows/iot (01.05.2016)

[13] Texas Instruments, ARM Based MCU TM4C123G [WWW] http://www.ti.com/tool/ek-tm4c123gxl (25.09.2015)

Lisa 1 – Juhend

Tõmba versioonihaldusest alla viimane juhtmooduli tarkvara. See asub aadressil https://bitbucket.org/karlsch/raspdaq. Eelnevalt peaks kontrollima, et olemas on järgnevad programmid:

- Python
- PySerial
- Node.js

Mine käsurealt projekti kausta ning sealt omakorda ’src/web/’ alla. Esimesel korral jooksutades sisesta enne käsk ’npm install’, mis installeerib serverile vajalikud teegid. Seejärel jooksuta serverit käsuga ’node index.js’.

Seadistamiseks muuda ’/src’ kaustas olevat ’config.py’ faili. Enne esimese eksperimendi jooksutamist tasub see kindlasti üle vaadata ning eksperimendile vastavaks seada. Seadistatavad parameetrid on järgnevad:

1) Ekperimendi tsüklite arv

2) Aeg mõõtmiste vahel

3) Kirjuta-Loe režhiimi valiku kontakti number Raspberry peal (vaikimisi PIN24)

4) Juhtmooduli aadress

5) Mõõtemoodulite arv

6) Alles hoitavate mõõtmiste arv, n viimast

Lisa 2 – Mõõtetsükli tarkvara

Loodud tarkvara on täielikult üleval lingil: https://bitbucket.org/karlsch/raspdaq. Lisana on osa tarkvarast, täpsemalt mõõtetsükli loogika.

```
#!/usr/bin/env python

import time
import serial
import array
import RPi.GPIO as GPIO
import StringIO

maxReadCount = 20
readBuffer = array.array('c')
# Setup serial connection options
ser = serial.Serial(
  port='/dev/ttyAMA0',
  baudrate=115200,
  parity=serial.PARITY_NONE,
  stopbits=serial.STOPBITS_ONE,
  bytesize=serial.EIGHTBITS,
  timeout=1
)

ID = config.id
BEGIN_TESTS_ALL = config.begintestsall
BEGIN_TESTS = config.begintests
ASK_STATUS_ALL = config.askstatusall
ASK_STATUS = config.askstatus
GET_RESULTS = cofig.getresults

def setup():
  # ----Set PIN24 to LOW to read in Raspberry(AMA0), set PIN24 to HIGH to write in
Raspberry(AMA0)
  GPIO.setmode(GPIO.BCM)
  GPIO.setup(24, GPIO.OUT)
  GPIO.output(24, GPIO.LOW)

def teardown():
  print 'now cleaning up'
  GPIO.cleanup()
```

```
  ser.close()
  return

def set_to_write():
  print 'setting to write mode'
  GPIO.output(24, GPIO.HIGH)
  return

def set_to_read():
  print 'setting to read mode'
  GPIO.output(24, GPIO.LOW)
  return

def send_code(code):
  print 'trying to send code'
  set_to_write()
  ser.write(code)
  time.sleep(0.5)
  set_to_read()
  print 'code should be sent, rasp set to READ mode'
  return

def build_message(data, address):
  message = data + address
  return message

def get_answer():
  i = 0
  k = 0
  output = StringIO.StringIO()
  while (i < 16050):
    if k == 5:
      print 'getting zero data, breaking'
      break
    x = ser.read(4)
    if x == '':
      print 'incrementing k, x is empty'
      k = k+1
    output.write(' ' + str(i) + ' ' + x)
    i = i+1

  print 'buffer: ' + output.getvalue()
```

```
if __name__ == "__main__":
  setup()
  time.sleep(2)
  send_code(BEGIN_TESTS)
  print 'sleep over, get results'
  time.sleep(5)
  send_code(GET_RESULTS)
  get_answer()

  print 'answer loop ended, going to teardown'
  time.sleep(1)
  teardown()
```