TALLINNA TEHNIKAÜLIKOOL
MEHAANIKATEADUSKOND

Mehhatroonikainstituut
Mehhatroonikasüsteemide õppetool

MHK70LT

Madis Lepiksaar

KAUGHALLATAV TÄNAVAVALGUSTUSE KONTROLLER

Autor taotleb
tehnikateaduse magistri
akadeemilist kraadi

Tallinn
2014

AUTORIDEKLARATSIOON

Deklareerin, et käesolev lõputöö on minu iseseisva töö tulemus.
Esitatud materjalide põhjal ei ole varem akadeemilist kraadi taotletud.
Töös kasutatud kõik teiste autorite materjalid on varustatud vastavate viidetega.

Töö valmis..... juhendamisel

“.....”.....201….a.

Töö autor

..... allkiri

Töö vastab magistritööle esitatavatele nõuetele.

“.....”.....201….a.

Juhendaja

..... allkiri

Lubatud kaitsmisele.

..... eriala/õppekava kaitsmiskomisjoni esimees

“.....”.....201… a.

..... allkiri

TTÜ mehhatroonikainstituut

Mehhatroonikasüsteemide õppetool

MAGISTRITÖÖ ÜLESANNE

2014 aasta kevadsemester

Üliõpilane:	Madis Lepiksaar, 122181	(nimi, üliõpilaskood)
Õppekava	MAHM02	
Eriala	Mehhatroonika	
Juhendaja:	Teadur, Maido Hiiemaa	(amet, nimi)
Konsultandid:	Puudub	(nimi, amet, telefon)

MAGISTRITÖÖ TEEMA:

(eesti keeles) Kaughallatav tänavavalgustuse kontroller

(inglise keeles) Remotely Managed Street Lighting Controller

Lõputöös lahendatavad ülesanded ja nende täitmise ajakava:

Nr	Ülesande kirjeldus	Täitmise tähtaeg
1	**Olemasolevate kaughallatavate tänavavalgustuskontrollerite võrdlemine ning projekteeritava kontrolleri nõuete ja funktsionaalsuse väljaselgitamine**	**1.03.2014**
2.	**Kaughalduse vajadusi rahuldava(te) võrguteenus(t)e leidmine koos vastava riistvaraga**	**1.04.2014**
3.	**Mikrokontrolleri valik, programmeerimine, tänavavalgustuskontrolleri elektroonika**	**15.04.2014**
4.	**Lihtsustatud andmevahetuse demonstratsioon tänavavalgustuskontrolleri ja serveri vahel**	**1.05.2014**
5.	**Kilbi näidiselektriskeem koos väljatöötatava tänavavalgustuskontrolleriga**	**16.05.2014**

Lahendatavad insenertehnilised ja majanduslikud probleemid:

Projekteerida kaughallatav tänavavalgustuskontroller, mis oleks sobilik väiksemate tänavavalgustusvõrkude haldamisel. Projekteerimisel silmas pidada nii valmistamiskulusid, kui hilisemaid halduskulusid.

Täiendavad märkused ja nõuded: Puuduvad

Töö keel: eesti keel

Kaitsmistaotlus esitada hiljemalt 22.05.2014 **Töö esitamise tähtaeg** 22.05.2014

Üliõpilane Madis Lepiksaar */allkirjastatud digitaalselt/* kuupäev 24.05.2014

Juhendaja Maido Hiiemaa */allkirjastatud digitaalselt/* kuupäev 24.05.2014

Konfidentsiaalsusnõuded ja muud ettevõttepoolsed tingimused formuleeritakse pöördel

SISUKORD

SELEDE LOETELU

TABELITE LOETELU

LISAD

GRAAFILINE OSA

EESSÕNA

Käesolev lõputöö on koostatud Madis Lepiksaare poolt. Lõputöö idee on pärit elektriehitusettevõttest ProSystem OÜ[1], kelle peamisteks tegevusvaldkondadeks on välielektrivõrkude ning tänavavalgustuse ehitus. Lõputöö lähteülesannet aitasid formuleerida Asko Kuusalu ning Margus Saar, kes mõlemad on ettevõtte juhtkonnast. Elektroonika projekteerimisel oli suureks abiks elektroonikainsener Mihkel Tedremaa ettevõttest Ericsson Eesti AS[2]. Mobiilivõrkude juures oli suureks abiks mobiilse andmeside insener Marius Siigur ettevõttest Elisa Eesti AS[3].

SISSEJUHATUS

Lõputöö teema valik on peamiselt põhjustatud autori huvist projekteerida elektroonikaskeeme ning luua seade, mis suudab suhelda internetivõrgus.

Käesoleval ajastul soovitakse kõigis valdkondades opereerida nii energiasäästlikult ja keskkonnasõbralikult kui vähegi võimalik. See on tinginud asjaolu, et ka tänavavalgustuse valdkonnas proovitakse raha kokku hoida ning seeläbi keskkonda säästa. Tallinnast pärit inimene sai soovist tänavavalgustuse pealt säästa aru, siis kui tänavate valgustust hakati südaöösiti välja lülitama. Kui varasemalt oldi harjunud, et kodutänaval põles lisavalgustus terve öö läbi, siis peamiselt raha nappuse tõttu hakati tänavaid, mis varasemalt olid mõneti ebaotstarbekalt öösiti valgustatud, välja lülitama.

Tänavate lisavalgustuse piiramine ei ole mitte ainult energiasäästlik, vaid põhjustab ka vähem valgusreostust. Valgusreostuse mõju elavale loodusele on samuti hakatud tähelepanu pöörama. Teadupärast on kõik organismid harjunud tavapärase päiksetõusu ja päikse loojangu tsükliga. Olukord, kus teatud vähemasustatud tänavad, kus esineb rohkesti ka rohelust, on valgustatud ka öö läbi, tekitab keskkonnakahju. Niisamuti kui taimed ja loomad on häiritud valgusreostusest, mõjub valgusreostus ka inimestele. Ka inimene on harjunud bioloogiliselt siiski öö ja päeva tsükliga. Kõike neid aspekte arvestades on hakatud keskenduma intelligentsele tänavate valgustamisele, mis lihtsamas lähenduses tähendab vähemaktiivsete piirkondade öise lisavalgustuse väljalülitamist teatud südaöisetel kellaaegadel.

Viimastel aastatel on kõik suuremad tänavavalgustuse infrastruktuurid välja ehitatud intelligentsetena, nii et neid on võimalik jälgida ja lülitada. Üldiselt ehitatakse tänavavalgustusvõrke pikkadeks perioodideks, seega enamus praeguseid võrke on siiski mitu aastat vanad. Küllaga soovitakse tihti ka neid rekonstrueerida selliselt, et hiljem oleks võimalik energiat ja keskkonda säästa.

Eelnevast ajendatuna on antud lõputöö eesmärgiks projekteerida kaughallatav tänavavalgustuse kontrolleri prototüüp, mis sobiks eelkõige olemasolevate võrkude kaasajastamiseks. Kontrolleriga peab olema võimalik tuvastada tänavavalgustuse liini korrasolekut. Teadupärast on elektriahelad kõik kaitsmestatud, seega piisab kui jälgida väljuvate liinide toitepinge olemasolu. Peamiseks ülesandeks on võimaldada tänavavalgustuse lülitusaegade haldamist ühest keskusest. Lisaks eelneval peab kontrollerit olema võimalik integreerida olemasolevatesse tänavavalgustuse juhtkilpidesse.

Kaughallatavaid tänavavalgustuse kontrollereid on turul palju, kuid peamiselt on kõik süsteemid arendatud välja mingite kindlate ehitusobjektide jaoks ning ehitatakse välja koos

uusehitistega. Lõputöö uurimuslikus osas on põgusalt kirjeldatud Tallinnas asuvat tänavavalgustuse distantsjuhtimisseadet, mis on välja arendatud KH Energia Konsulti[4] poolt ning mida on kasutatud laialdaselt ka olemasolevate lülitusseadmete rekonstrueerimisel.

Käesoleva lõputöö koostamisel on kasutatud erinevaid arvutitarkvarasid. Kõik elektroonikaskeemid on koostatud vabaralise tarkvaraga KiCAD EDA[5]. Elektriskeemid on koostatud joonestustarkvaraga AutoCAD firmalt Autodesk [6]. Erinevate selede koostamiseks on kasutatud veebipõhist skeemiredaktorit Lucidchart [7]. Lõputöö tekstiline osa on koostatud kasutades Microsoft Office [8] kontrotiarkvara paketist tekstiredaktorit Word. Kontrolleri tarkvara on loodud kasutades Arduino[9] programmeerimiskeskkonda. Tarkvara arendamiseks on kasutatud prototüüpelektroonikaplaati Gboard ettevõttelt IteadStudio[10]. Võrgu andmeid on monotooritud kasutades WiresHark [11] tarkvara.

Lõputöö on koostatud projekteeritava kontrolleri esimese prototüübi kohta, milles on lahendatud ära enamus funktsionaalseid küsimusi ning välja selgitatud, kas taolise elektroonika riistvaraga on otstarbekas konstrueerida tänavavalgustuse kontroller.

Käesolev lõputöö on jagatud viide põhiossa. Esimeses põhiosas käsitletakse uurimustööd, mis on tehtud peamiselt olemasoleva olukorra kaardistamiseks ning ühe sarnase lahenduse funktsionaalsuse välja selgitamiseks. Uurimustöö lõpuosas formuleeritakse täpsem lähteülesanne.

Teiseks osaks on kontrolleri riistvara. Antud osas selgitatakse erinevate komponentide valikut ning tööpõhimõtteid. Pikemalt kirjeldatakse mikrokontrolleri tööpõhimõtet, mis on kontrolleri üks peamiseid juhtkiipe. Kontrolleri riistvara osas kommenteeritakse ka lõputöö raames projekteeritud prototüüpelektroonika skeemi. Kõige viimasena kirjeldatakse ka trükkplaadi projekteerimist ning ülesehitust ning esimese valminud tooriku tagasisidet.

Kolmandaks osaks on kontrolleri tarkvara. Kontrolleri tarkvara osas kirjeldatakse kontrolleri tarkvara tööd läbi erinevate algoritmide. Lisaks kirjeldatakse ka programmeerimiskeele eripärasid ning mõningaid lõike programmi koodist.

Neljandaks osaks on serveriga suhtlus. Antud osas kirjeldatakse kekserveriga suhtlust. Milliseid edastusprotokolle kasutatakse ja millise loogika alusel toimub identifitseerimine ning erineva teabe edastus.

Viimases viiendas osas käsitletakse tugevvoolu osa. Tugevvoolu osas keskendutakse elektriskeemile, mis võimaldaks projekteeritava kontroller paigaldada olemasolevasse tänavavalgustuse lülitusjaotusseadmesse.

Lõputöö lisades on ära toodud prototüübile kirjutatud programmi kood, koos teekidega. Lisaks on lisades esimene versioon elektroonikaskeemidest ning elektriskeem.

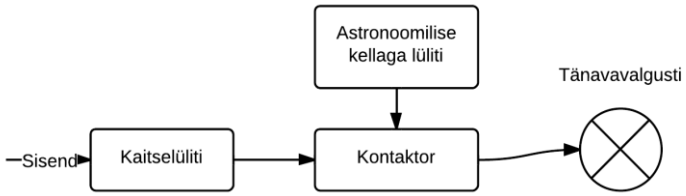
LÜHENDITE LOETELU

RSSI	Vastuvõetud signaali tugevuse indikaator
EEPROM	Elektriliselt kustutatav programmeeritav püsimälu
SPI	Järjestik perifeerne liides
USART	Universaalset sünkroonne/asünkroonne jadaliides
RISC	Kärbitud käsustikuga arvuti
SRAM	Staatilinee juhupöördusega mälu
GPRS	Üldine raadio-pakettandmeside teenus
ETSI	Euroopa Telekommunikatsiooni Standardi Instituud
SIM	Abonendi tuvastusmoodul
DC	Alalisvool
AC	Vahelduvvool
APN	Pakettvõrgu pöörduspunkti nimi
HTTP	Hüperteksti edastusprotkoll
ISP	Süsteemisisene programmeerimine
MISO	Ülema sisend alama väljund
MOSI	Ülema väljund alama sisend
SCK	Jadapordi taktsagedus
OSI	Avatud süsteemide sidumise arhitektuur
URI	Ühtne ressursi-identifikaator
PHP	Hüperteksti preprotsessor
HTML	Hüperteksti märgistuskeel

1 UURIMUSTÖÖ

1.1 Tüüpilisemad tänavavalgustuse lülitusjaotusseadmed

Tänavate lisavalgustust on tarvis lülitada päikese loojangul sisse ja tõusul välja, iga päev eri aegadel. Varasemalt on selleks kasutatud inimest, kes füüsiliselt on võrgu sisse ja välja lülitanud. Tänapäeval kasutatakse selleks lihtsat automaatikat. Järgneval selel on näha plokkskeemi tüüpilisemast tänavavalgustuse lülitamise skeemist.



Sele 1.1 Klassikalise lülitusjaotusseadme plokkskeem

Lülitusjaotusseade koosneb peamiselt:

- Kaitselülitist
- Kontaktorist
- Astronoomilisest kellast

Kaitselüliti rakendub tänavavalgustusliini lühise korral, kaitsteks liinis tekkiva lühisvoolude korral liini põlemisest. Tavaliselt kasutatakse automaatkaitselülitit, mille algolekut on lihtne taastada.

Kontaktoriga lülitatakse tänavavalgustusliini sisse-välja. Kontaktorina kasutatakse lihtsamaid mootorite juhtimise kontaktoreid, peamiselt nende soodsate hindade tõttu. Kontaktori võiks asendada ka releega, kuid majanduslikult ei anna see eeliseid.

Astronoomilise kellaga lülitiga juhitakse kontaktorit. Lüliti lülitatakse sisse päikese loojangul ning välja päikesetõusul. Astronoomiline kell on programmeeritud vastavalt asukoha pikkus ja laiuskraadile, mille alusel arvutatakse iga päev uus sisse ja väljalülitusaeg. Astronoomilise kella asemel võib kasutada ka hämarlülitit. Hämarlülitiga on võimalik lülitada kontaktorit sisse ja välja vastavalt seadistatud valgustaseme saavutamisel. Antud lahenduse puuduseks on hämarlüliti paiknemine kilbist väljaspool ning selline lahendus nõuab majanduslikult kallimat käitu.

1.1.1 Tavalise lülitusjaotusseadme eelised ja puudused

Kõige suuremaks eeliseks tavalise lülitusjaotusseadmel on tema lihtsus, nii paigaldamisel kui hilisemal käidul. Lülitusjaotusseadme paigaldamise ja programmeerimisega saab hakkama tavaline elektrik. Hilisem käit on samuti lihtne, seisneb peamiselt seadmete kontrollis.

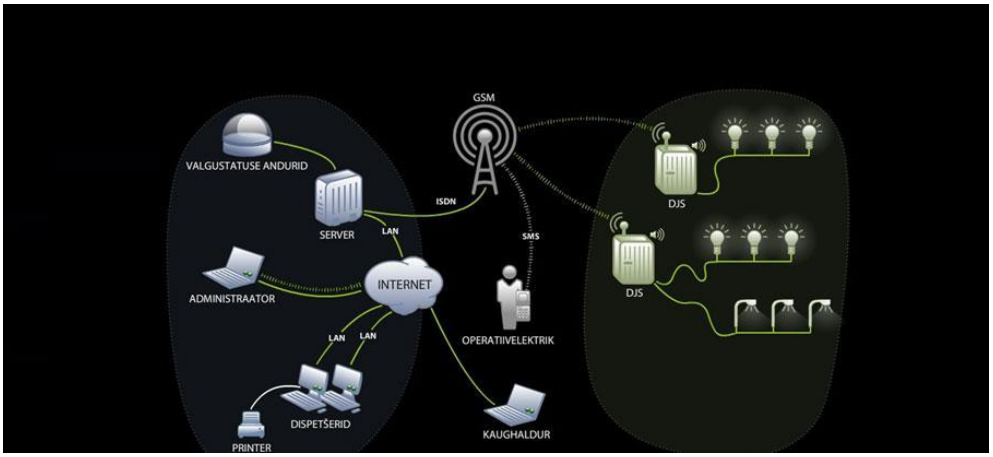
Tavalise lülitusjaotusseadme peamiseks puuduseks on samuti tema autonoomsus. Juhul kui kaitselüliti rakendub, mille rakendumine viitab lühisele liinis, ei saada sellest teada enne, kui märgatakse tänavavalgustuse mittepõlemist. Teiseks suureks puuduseks on energiasäästureziimi rakendamise puudumine. Väiksemate tänavate, mis paiknevad näiteks elamurajoonides, valgustamine teatud aegadel öösiti ei ole otstarbekas. Sellisel juhul oleks tänavavalgustus mõistlik öösel näiteks kella 01:00 ja 05:00 vahel välja lülitada, saavutades energiasäästu ning vähendades valgussaastet.

1.2 Tallinnas kasutatav tänavavalgustusjuhtimissüsteem

Alates 2000. aastas on Tallinnas välisvalgustust hallatud kaugjuhtimisseadmetega. Antud seadmetega on võimalik juhtida ning jälgida välisvalgustuse olekuid. Andmesideühendus toimub kasutades globaalset mobiilsidesüsteemi.

Kogu süsteemi on arendanud ning hallanud ettevõtte AS KH Energia Konsult. Süsteemi soetamisel valmistab ettevõtte juhtimiskilbi ning korraldab hiljem ka käitu.

Kasutajaks on operatiivkorraldaja, kes ööpäevaringselt jälgib välisvõrkude olekut ning seadistab juhtimist. Vea tuvastab operatiivkorraldaja jälgimiskeskusest ning selle ilmnemisel saadetakse välja olukorraga tutvuma operatiivelektrik. Järgneval selel on näha, juhtimissüsteemi protsessi plokkskeemi.



Sele 1.2 KH Energia konsulti distantsjuhtimise skeem

Selelt on näha distantsjuhtimisseaded (DJS), serverit, operaatorjaama ning teisi süsteemi komponente.

1.2.1 KH Energia Konsulti süsteemi eelised ja puudused

Eeliseks on kaughalduse võimalus. Sellele tõttu saab operatiivsemalt vigu avastada ning neid likvideerida. Samuti on neil lahendatud energiasäästusüsteem täna kaughaldusele.

Puuduseks võib lugeda asjaolu, et antud süsteem sobib ainult suurte objektide haldamiseks. Juhtimissüsteemid on loodud pigem mitmevõrguliste juhtimiskilpide tarbeks. Süsteemi ei ole võimalik soetada ilma käiduta. Samuti ei saa olemasolevasse kilpi seadmete paigaldamisega tavaline elektrik hakkama.

1.3 Nõuded elektriseadmetele

Euroopas tarbijatele müüdavad elektriseadmed peavad omama CE märgistust. Märgistuse saamiseks peavad elektriseadmed vastama Euroopa Liidu direktiividele. Peamiseks nõudeks on, et elektriseadmed vastaksid elektromagneetilise ühilduvuse direktiivile, direktiiv 2004/108/EÜ[12].

Elektromagneetilise ühilduvuse direktiiv sätestab seda, et seade ei tohi kiirata elektromagnetlaineid selliselt, et oleks häiritud teda ümbritsevate seadmete töö. Teiseks oluliseks aspektiks on see, et seade ise peab häiringuteta töötama teiste seadmete poolt tekitatud elektromagnetväljas.

Projekteeritavas seadmes kasutatakse 230 V vahelduvpinget, seetõttu peab olema tavakasutaja jaoks isoleeritud ja maandatud kõik elektrijuhtivad kohad, mida kasutaja saab tavaolukorras katsuda. Kuna antud seade on mõeldud siiski elektrikilpi, mis ongi mõeldud juba elektriseadmete isoleerimiseks, siis maandatuse nõude täitmist ei ole tarvilik jälgida. Küllaga on mõistlik kaitsta kõik ahelad ülekoormuskaitsmetega.

1.4 Andmesidevõrgud

Tänavavalgustus asetseb tavaliselt asustatud paikades. Asustatud kohtades on üldiselt hästi välja arendatud infrastruktuur, küllaga kaabelside tänavavalgustuse juhtkilpidesse tavaliselt ei ulatu. Seega tuleb kaaluda ainult juhtmevaba side võimalusi. Sarnased lahendused, mille haldusaladeks on suured piirkonnad, kasutavad tavaliselt mobiilside võrku. Spetsiaalse andmesidevõrgu loomine ainuüksi tänavavalgustuse haldamise pärast ei ole otstarbekas, seega on keskendutud pigem mobiilside võrgule.

1.4.1 Andmevahetus

Tänavavalgustuse jälgimine ja lülitamine ei ole ajakriitiline, seega haldamine ei pea toimuma reaalajas, piisab kui iga teatud aja tagant saadetakse keskseadmesse olek. Tänavavalgustuse kontrolleri ning keskuse vaheline suhtlus võib toimuda isegi mõne minutiliste vahedega.
Mobiilside võrgus kasutatakse andmevahetuseks kolme erinevat viisi.
Helistamisteenus on üks vanimaid. Helistamise puhul toimuks otsekõne serveriga, kus kõne alustajaks võiks olla nii server, kui ka kontroller. Kontroller või keskus tuvastaks kõne allika ning selle alusel toimuks andmevahetus. Kõige suuremaks miinuseks antud lahenduse juures oleks liini hõivatus. Sellisel viisil saaks olla korraga vaid üks kontroller ühendatud keskusega. Süsteem sobiks sellisel juhul, kui ühe keskuse kohta oleks vähe kontrollereid ning kõnesid tehakse suhteliselt harva.
Teine võimalus on kasutada lühisõnumeid. Lühisõnumite saatmine on tarkvaralises mõttes üks lihtsamaid. Osapoolte tuvastus toimuks jällegi adressaatide numbrite järgi. Lühisõnumite puhul pole samuti paralleelühendus võimalik, kuid side järjekorra eest hoolitseb sideoperaator, helistamise puhul oleks järjekorra algoritmi pidanud looma süsteemi sisse. Lühisõnumite saatmise suurimaks miinuseks on kõrge hind, juhul kui saata teavitus iga 30 minuti tagant saadetaks kuus keskmiselt *ca* 3000 lühisõnumit ühe kontrolleri kohta.
Kolmas võimalus on kasutada andmesidet. Andmesideühenduse puhul looks kontroller serveriga ühenduse ning uuendaks olekuandmeid ning samas võtaks vastu seadeväärtuseid. Andmeside puhul saaks mitu kontrollerit ühenduda serveriga sisuliselt samaaegselt ning väikeste andmete puhul järjekorrad oleksid oluliselt lühemad kui lühisõnumite või helistamise puhul. Kontseptsiooni kohaselt, kus keskus suhtleb kontrolleriga näiteks igal pooltunnil on andmevahetus suhteliselt väike, andmepaketi suurus jääks mõne kilobaidi juurde. Näiteks 3 kilobaidise andmete hulga juures saadetaks kuus alla 5 megabaiti andmeid, mis on tänase infrastruktuuri võimekuse juures kaduvväike suurus.

1.4.2 Levi

Asetades kontrolleri tavalisse väliskilpil, mis on tavaliselt valmistatud lehtmetallist, tekib küsimus, kuidas on levi kilpides. Selleks, et välja selgitada, kas on tarvidus tuua antenn kuidagi kilbist välja on tehtud lihtne katsetus. Katsetus tehti iMall IteadStudio poolt välja töötatud mobiilandmeside prototüüpelektroonikaga Gboard, mis on riistvaraliselt sarnane antud lõputöös väljatöötatava kontrolleri riistvaraga.

Katsetamiseks valiti tavaline lehtterasest kilp, ning katse käigus koguti leviandmeid vahetult kilbi kõrvalt ning seejärelt kilbi seest. Andmeid koguti ühe tunni vältel ning valimi suurus oli umbes 3000 lugemit.

Mobiilandmeside seadmed suudavad väljastada leviindikaatorina vastuvõetud signaali tugevuse indikaatori (edaspidi RSSI ingl k *received signal strenght indicator*) väärtuse. RSSI väärtuse seos välja tugevusega ei ole standardiseeritud ning kõik seadmetootjad rakendavad seda omamoodi. Küllaga toovad seadmetootjad välja seose RSSI ja välja tugevuse vahel. Konkreetselt kasutatava SIMCOM SIM900[13] mooduli poolt genereeritava RSSI väärtuse ja väljatugevuse vahelist seost on näha järgnevas tabelis.

Tabel 1.1 RSSI ja väljatugevuse omavaheline seos

RSSI	**Väljatugevus [dBm]**
0	-115 või vähem
1	-111
2...30	-110...-54
31	52 või rohkem
99	pole teada või detekteeritav

Tabelist on näha, et väljatugevust mõõdetakse detsibellides. Seost detsibellide ning millivattide vahelt on näha järgnevast valemist.

$$1\ dBm = 10log_{10}\frac{P}{1mW}, \quad (1.1)$$

millest *P* on vastuvõetud signaali võimsus.

Levi katsetuse tulemusi on näha järgnevast tabelist.

Tabel 1.2 Levi katsetuse tulemused

Kilbist väljas		Kilbis sees	
Välja tugevus [dBm]	**Esinemise sagedus [%]**	**Välja tugevus [dB]**	**Esinemise sagedus [%]**
-64	12	-70	46
-62	40	-68	54
-58	8		
-56	15		
-54	25		

Tabelist on näha, et keskmine levi tugevus küll langes, aga olukorras, kus kilbi piirkonnas on korralik mobiili levi võime eeldada, et see levi on vähemalt rahuldav ka kilbis sees.
Tõrgeteta andmevahetus toimib veel väljatugevusega -100 ... -105 dBm kohta.

1.5 Lähteülesanne

Antud lõputöö lähteülesandeks on projekteerida tänavavalgustuskontrolleri prototüüp, millel oleksid järgmised omadused:

- 35 mm profiilliistule kinnitusvõimalused
- Võimalus jälgida kahte 230VAC ahelate olekut
- Toide 230 VAC võrgust
- Lülitada kahte väljundit
- Võimaldaks sidet üle võrgu

2 KONTROLLERI RIISTVARA

2.1 Korpus

Korpuse valimisel oli põhikriteeriumiks, et korpust oleks võimalik paigaldada standardsele 35 mm DIN liistule.

Valituks osutus Camdenboss CNMB/5[14] korpus. Korpusesse on võimalik paigaldada kuni kolme tasapinda trükkplaadid ning korpus omab kinnitust, millega on võimalik korpus kinnitada DIN liistule. Lisaks on korpusel eemaldatavad osad, mille eemaldamisel saab luua avasid klemmide jaoks. Järgnevalt selelt on näha valitud korpust.

Sele 2.1 Korpus

Korpuse välismõõtmed on järgmised 90x88x58 mm (pikkus x laius x kõrgus). Tavalise automaatkaitselüliti mõõtmed on 88x17,5x66 mm (pikkus x laius x kõrgus). Võrdluseks on võetud ABB S-200[15] seeria automaatkaitselüliti, mis on laialt levinud Eesti turul. Näha on, et kõrguselt ja pikkuselt on mõõtmed sarnased ning laiuses on mõõtmed umbes 3 korda suuremad. Hea tava on jätta elektrikilbi montaažiliistule 20 % varu. Võttes, et tavalises tänavavalgustuskilbis on 8 automaatkaitselülitit, programmkell, 2 kontaktorit, mis moodustavad laiuses *ca* 300 mm. Arvestades sellest 20 % saame 60 mm laiuse varuks. Selliselt võime eeldada, et antud mõõtmetega korpus mahub olemasolevatesse tänavavalgustuskilpidesse, isegi siis kui ei soovita loobuda programmkellast.

Lõputöö tulemusena valmiva prototüüplahendus, kasutab ära kaks trükkplaaditasandit. Kontrollerit laiendades oleks võimalik kõrgeimale tasandile luua nuppudest ja ekraanist kasutajaliides, millega saaks ära teha esmatasandi seadistuse. Näiteks panna kell õigeks ning valida ära esmased lülitusajad.

2.2 Mikrokontroller

Kontrolleri tööd juhib mikrokontroller. Mikrokontroller on elektroonikakomponent, milles on ühendatud protsessor, mälud sisend-väljundmoodulid, andmesidemoodulid, taktgeneraator jpm. Mikrokontrollereid kasutatakse peamiselt madalat arvutusvõimsust vajavates lahendustes. Mikrokontrolleri valikul peamisteks kriteeriumiteks olid, et neil oleks kõik vajalikud sisend väljundid ning et neile oleks loodud esmatasandi teegid programmeerimiseks, lihtsustamaks prototüüptarkvara loomet.

2.2.1 Mikrokontrolleri valik

Mikrokontrolleri valimisel pidi see vastama vähemalt järgnevatele nõuetele:

1) Omama elektriliselt kustutatavat programmeeritavat püsimälu (edaspidi EEPROM ingl k *Electrically Erasable Programmable Read Only Memory*)
2) Omama järjestik perifeerset liidest (edaspidi SPI ingl k Serial Peripherial Interface)
3) Omama universaalset sünkroonset/asünkroonset jadaliidest (edaspidi USART ingl k *Universal Synchronous/Asynchronous Receiver/Transmitter*)
4) Omama analoog-digitaal muundurit
5) Arduino[16] teekide kasutamise võimalus

Järgnevas tabelis on võrreldud kahte mikrokontrollerit, mis sobisid esialgsete nõudmistega.

Tabel 2.1 Mikrokontrollerite võrdlus

	ATmega328p	**Atmega2561**	**Atmega32u4**
Toitepinge	1,8-5,5 V	4,5-5,5 V	2,7 - 5,5 V
Täätakti sagedus	20 MHz	16 MHz	16 MHz
Arhitektuur	8-bit	8-bit	8-bit
Välkmälu	32 kBaiti	256 kBaiti	32 kBaiti
SRAM	2 kBaiti	8 kBaiti	2,5 kBaiti
EEPROM	1 kBait	4 kBaiti	1 kBait
SPI	1	1	1
USART	1	4	1
Sisendid/Väljundid	26	54	26
Analoog digitaal muunduri sisendeid	8	8	12
Arduino teegid	Ametlikud	Mitte ametlikult	Ametlikud
Viikude arv	32	64	44
Pakendi suurus	9x9 mm TQFP	16x16 mm TQFP	10x10 TQFP
Muud lisad			USB
Hind	€ 2,61	€ 5,89	€ 16,81

Valituks osutus mikrokontroller Atmega 328p[17], peamiselt tema soodsa hinna ning kättesaadavuse tõttu. Samuti on ta ka kõige pisema korpusega, seega võtab trükkplaadil kõige vähem ruumi.

2.2.2 Protsessor ja mälu

AVR Atmega mikrokontrolleri puhul on tegemist kärbitud käsustikuga arvutiga (edaspidi RISC ingl k *Reduced instruction set computer*). Kärbitud käsustik on mikrokontrolleritel laialt levinud. Kärbitud käsustiku puhul on käsud lihtsad ning enamik neist suudetakse täita ühe töötaktiga.

Programmi mäluks on AVR mikrokontrolleril välkmälu. Konkreetsel mudelil on välkmälu 32 kilobaiti. Lisaks kasutatakse vajadusel välkmälu ka buudilaaduri hoiustamiseks. Programmi mälu on võimalik üle kirjutada kuni 10 000 korda. See võimaldab väga palju programmiuuendusi paigaldada või kasutada üht ja sama kontrollerit arenduseks pika aja vältel. Staatilise juhupöördusega mälus (edaspidi SRAM ingl k *Static Random Access Memory*) hoitakse jooksvaid andmeid. Tüüpilisteks jooksvateks andmeteks on registrite andmed ning muutuvate väärtuste andmed. SRAM kustub alati pärast kontrolleri taaskäivitamist. Seega kustuvad sealt kõik andmed, mis on sinna varem talletatud. Nii tuleb alati programmi käivitades seadistada registrid ning muutujate väärtused tuleb uuesti arvutada. SRAM-ist andmete lugemine võtab aega 2 töötakti. SRAM on jagatud erinevateks segmentideks – registrite mälu, sisend/väljundite mälu, laiendatud sisend/väljundite mälu ning sisemine SRAM. Registrite mälus hoitakse erinevate registritena soovitud kontrolleri üldseadistust. Sisend-väljundite mälus hoitakse nende seadistust ning olekut. Laiendatud sisend-väljundite mälus hoitakse andmeid, mis on seotud andmetega, mis läheb tarvis siis, kui sisend väljundit ei kasutata tavaliselt vaid näiteks analoogsisendina vms. Sisemist SRAM-i saab kasutada kasutaja programmis näiteks muutujate väärtuste salvestamisena. Järgmises tabelis on näha SRAM-i jaotuse struktuuri.

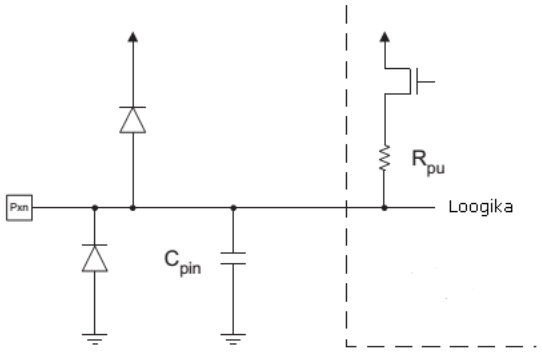
Tabel 2.2 SRAM-i jaotus AVR-is

Andmed	Mälupesad
32 Registrit	0x000 - 0x001F
64 Sisend/Väljund registrit	0x0020 - 0x005F
160 Laiendatud Sisend/Väljund registrit	0x0060 - 0x00FF
Sisemine SRAM 2048x8 baiti	0x0100 - 0x08FF

Andmeid, mida soovitakse säilitada ka pärast kontrolleri taaskäivitamist salvestatakse EEPROM-i. Konkreetsel mikrokontrolleril on 1 kbaiti EEPROM-i. Kaughallatava kontrolleri lahenduses kasutatakse EEPROM-i näiteks väljundite olekute ning järgmiste lülitusaegade hoiustamiseks.

2.2.3 Sisend-väljundid

Mikrokontrolleritel on tavaliselt laialt seadistatavad sisend-väljundid. Kõikide sisendite ning väljundite esmased funktsioonid on digitaalsisendid-väljundid. Digitaalsisend suudab registreerida pingenivoo järgi, kas sisend on loogiliselt tõene või väär. Digitaalne väljund suudab genereerida loogiliselt kõrge või loogiliselt madala pingenivoo.
Järgneval selel on näha AVR mikrokontrolleri sisendi/väljundi skeemi.



Sele 2.2 Mikrokontrolleri sisend-väljundi skeem

Selelt on näha, et kõigil sisenditel on kaitsedioodid ning stabiliseerimiseks kondensaator. R_{pu} tähistab nivootakistit. Nivootakistit saab sisse-välja lülitada.
Kõikide sisend-väljundite seadistamine käib registrite kaudu. Igal sisend-väljundi plokil on kolm registrit DDRx, PORTx ning PINx, x-ide asemel on ploki tähis (A, B, C jne). Iga plokk koosneb maksimaalselt 8-sast viigust. DDRx register on mõeldud sisendi-väljundi tüübi defineerimiseks, kahendväärtuses 1 määrab viigu väljundiks ning 0 määrab viigu sisendiks. Näiteks soovime valida B pordi esimese viigu väljundiks, siis peame seadistama registri nii nagu on näha järgneval selel. X tähistab mitteolulist olekut.

Register	DDRB							
Bit	7	6	5	4	3	2	1	0
Väärtus	x	x	x	x	x	x	1	x

Sele 2.3 Registri seadistus mikrokontrolleris

PORTx register on mõeldud väljundi seadistamiseks. Kirjutades vastava biti kohale kahendväärtuses 1, seadistame väljundi kõrgeks ning kirjutades vastava biti kohale 0, seadistame väljundi madalaks. Riistvaraliselt lülitatakse nivootakistit sisse ja välja, kui soovitakse saada kõrget väljundnivood, siis lülitatakse nivootakisti sisse ning kui soovitakse saada madalat väljundnivood, siis lülitatakse nivootakisti välja.

PINx register on mõeldud sisendite olekute lugemiseks. Lugedes registri teatud biti väärtust saame teada tema loogilise oleku. Juhul kui bit on seatud kõrgeks on väljundi oleks kahendväärtuses 1, kui bit on seatud madalaks, siis 0.

2.2.4 Viikude alternatiivsed funktsioonid

Enamus sisend-väljund viikudel on lisaks tavalisele digitaalsele sisend-väljundi funktsioonile võimalik seadistada ka alternatiivseid funktsioone. Nii nagu ka sisend-väljund funktsiooni seadistamine käib ka alternatiivsete funktsioonide seadistamine vastavate registrite seadete muutmistega.

Kasutataval mikrokontrolleril on võimalik seadistada kuni 8 analoog-digitaal-muunduri sisendit. Muunduri sisend diskrediteerib analoogväärtuse kuni 10 bitise täpsusega. See tähendab, et analoog 0 V nivoo ja analoog referentspinge vaheline ala jagatakse 1024 pingenivooks. Juhul, kui kasutatakse toitepinget ning analoogreferentspinget 3,3 V, siis analoog-digitaalmuunduri jaotis on ~0,03 V. Analoogväärtuste lugemine võtab digitaalelektroonika mõistes väga palju aega. Selleks, et aeglustada analoogväärtuste lugemist on analoog digitaal muundurid eraldi taktsignaali jaotur, millega on võimalik saavutada töötakt vahemikus 50 kuni 200 kHz, mille tulemusena saadakse maksimaalselt täpne analoogväärtus. Üksiku analoogväärtuse lugemine võtab aega 25 töötsüklit ning kui üks takt on 200kHz, saame, et minimaalselt võtab ühe analoogväärtuse lugemine aega 0,125 ms. Võrdluseks digitaalse väärtuse lugemine registrist võtab aega 1 töötakt ilma jagurita, kui kasutada maksimaalset töötakti sagedust, milleks on 20 MHz, siis digaalväärtuse lugemine võtab aega kõigest 0,05 µs. Käesoleva töö raames projekteeritaval kontrolleril on üks lisa analoogsisend, mille lugemisel kasutatakse mikrokontrolleri analoog-digitaal muundurit.

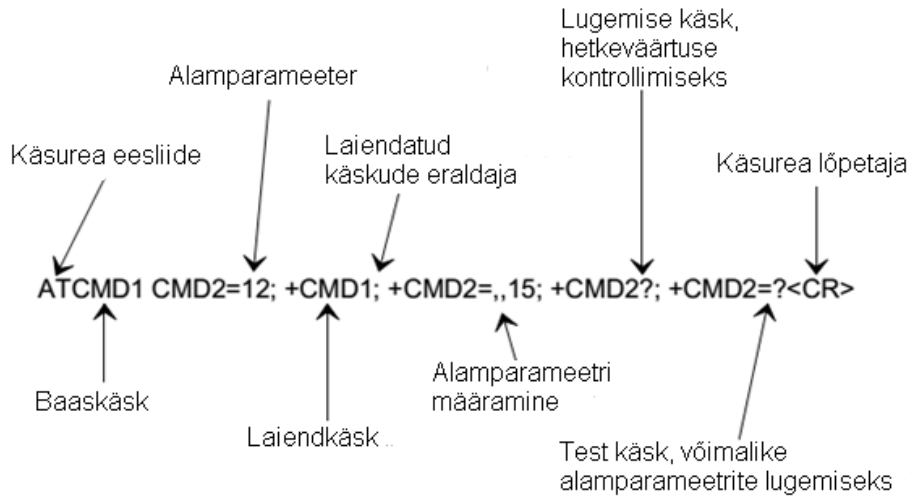
2.3 Mobiilsidevõrgu moodul

Prototüüp kontrolleril kasutatakse SIMCOM SIM900 mobiilside moodulit. Antud side moodul osutus valituks tema odava hinna pärast, kättesaadavusele ning teekide olemasolule.
SIM900 olulisemad omadused:

- Juhitav AT käskudega[18]
- Sagedused: 850/900/1800/1900 MHz
- Üldine raadio-pakettandmeside teenus (edaspidi GPRS ingl k *General Packet Radio Service*)
- Toitepinge 3,2...4,8V
- Voolutarve: kuni 2A
- Jadapordiga suhtlus

2.3.1 AT käsustik

Suuremal osal kasutajapoolsetel mobiilside telekommunikatsiooniseadmetel kasutatakse suhtluse jaoks AT käsustikku. Ajalooliselt on AT käsustik välja kasvanud Hayesi käsustikust. Hayesi käsustik loodi 1981 aastal Hayesi modemi jaoks. Hiljem hakkasid ka teised telekommunikatsiooniseadmete tootjad sarnast käsustikku kasutama. Hiljem standardiseeriti käsustik Euroopa Telekommunikatsiooni Standardi Instituudi poolt (edaspidi ETSI ingl k *European Telecommunications Standards Institute*)[19].
Kasutatav modem kasutab samuti AT käsustikku. Järgneval selel on näha AT käsustiku põhistruktuuri.



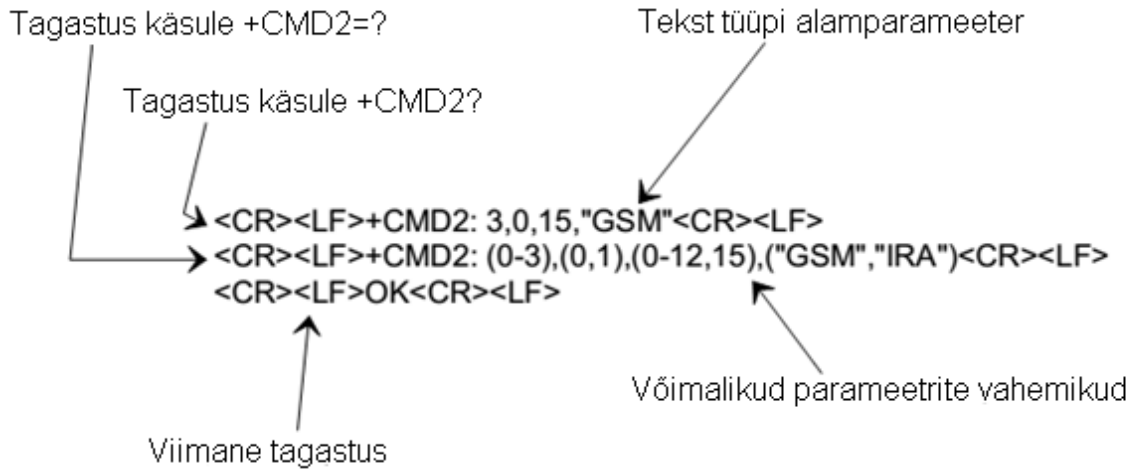
Sele 2.4 AT käsustiku põhistruktuur

AT käskudel on järgnev struktuur:

- Kõik käsud algavad AT-ga
- Baaskäsu puhul järgneb AT-le kohe käsk ilma plussita.
- Juhul kui soovitakse kirjeldada mingit parameetrit baaskäsuga, siis järgneb sellele võrdusmärk ning parameeter.
- Juhul kui soovitakse ühe reana mitu käsku anda, siis erinevad käsud eraldatakse semikooloniga, kuid uuesti AT käsu eesliidet pole.
- Laiendkäsu eesliiteks on pluss märk, mille järel tuleb käsk
- Juhul kui käsule järgneb küsimärk on tegemist lugemise käsuga.
- Juhul kui käsule järgneb võrdusmärk ja küsimärk on tegemist test käsuga
- Käsurea lõpetab keelemärkide koodis <CR>

Näiteks kui soovitakse seadistada abonendi tuvastusmooduli (edaspidi SIM ingl k. *Subscriber Identification Module*) parooli saadetakse käsklus AT+CPIN=0000<CR>. Sellisel juhul AT tähendab käsu eesliidet, plussiga märgitakse, et tegemist on laiendkäsuga, CPIN on käsu identifikaator ning pärast võrdusmärki kirjeldatakse parameeter, antud näite puhul on parameetriks 0000.

Nii nagu käsu saatmisel moodulisse on oma vorming on ka vastuse vastuvõtmisel oma formaat. Järgnevalt selelt on näha vastuse vormingut.



Sele 2.5 AT vastuse vorming

Selel on kujutatud vastuseid hetkeväärtuse lugemisele ning test käsu saatmisele.

AT käskude tagastustel on järgnev struktuur:

- Iga tagastus on järgnevate keelemärkide vahel: <CR><LF> ... <CR><LF>
- Viimasena tagastatake järgnev tekst: <CR><LF>OK<CR><LF>

2.3.2 Pakettandmeside

SIM900 mooduliga on võimalik kasutada üldist raadio-pakettandmeside teenust. Üldine raadio-pakettandmeside teenus on andmesideteenus, mis on arendatud mobiilsidevõrkudele. Vastavalt üldisele arengule on ka raadio-pakettandmeside teenus arenenud ning need on jaotatud vastavalt generatsioonideks. Kõige lihtsam viis erinevaid generatsioone omavahel eristada on vaadata nende andmeside bitikiiruseid, mis on sageli tavakasutajale kõige olulisem. Järgnevas tabelis on välja toodud erinevate põlvkondade erinevused.

Tabel 2.3 Pakettandmeside põlvkonnad

Põlvkond	**Nimetajad**	**Andmeside allalaadimiskiirus**
1G	Analoogside, ainult kõne	Puudub
2G	Digitaalne side, kandis üle ainult kõne ja lühisõnumeid.	Puudub
2,5G	Lisandus pakettandmeside.	kuni ~116 kbit/s
2,75G	Täiustatud mobiilandmeside	kuni ~236 kbit/s
3G	Loodud silmas pidades andmesidevõimekust, suurem läbilaskevõime	kuni ~384 Mbit/s
3,5G	Suurendati allalaadimiskiiruseid	kuni ~14 Mbit/s
3,75G	Suurendati üleslaadimiskiiruseid	kuni ~28 Mbit/s
4G	Ülikiire andmeside	kuni ~100 Mbit/s

Eelnevas tabelis on toodud välja vaid üldised kriteeriumid mille alusel tuvastada erinevaid põlvkondasid. Andmeside allalaadimiskiirused on orienteeruvad ning sõltuvad konkreetsest teenusepakkujast. Lisaks andmeside kiirustele, on erinevate võrkude vahel veel teisi erinevusi, mida tavakasutaja tähele ei pane. Enamasti muutusid erinevate põlvkondade vahel ülekandesagedused, krüpteering, andmepakettide suurused jne. Tehniliselt on kõik võrgud üksteisest piisavalt erinevad, seetõttu ühes seadmes mitut võrku kasutada on keeruline.
Antud töö raames projekteeritaval prototüüpelektroonikal on kasutusel andmesidemoodul mis kasutab 2,5 põlvkonna andmesidevõrku.

2.4 Teised elektroonikakomponendid

Eelnevalt kirjeldati elektroonikakomponente, millel lasub antud riistvaras peamine rõhk. Mikrokontrolleri ja mobiilside mooduli funktsioneerimiseks on tarvilikud ka muud komponendid.

2.4.1 Toitemoodul

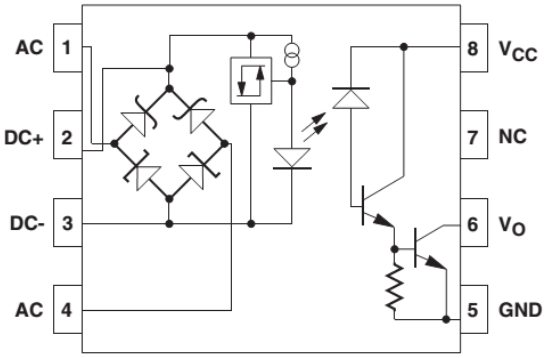
Antud töö raames projekteeritavale prototüübile on valitud toitemoodul[20]. Toitemooduli ülesandeks on muundada 230V vahelduvvool 12V alalisvooluks. Toitemoodul valiti peamiselt võimsuse järgi, antud toitemoodulil on maksimaalseks väljundvõimsuseks 18 W.

Toitemoodul omab CE märgistust järelikult luues seadme, kasutades seda moodulit, võime eeldada, et vähemalt toite osa on vastav peamistele elektrinõuetele. Järgnevalt selelt on näha toitemooduli illustreerivat selet, millelt on näha ka CE märgistust.

Sele 2.6 Toitemoodul

2.4.2 Optiline lahtisidesti

Projekteeritava seadme üheks olulisemaks omaduseks on jälgida väljundite olekut. Enamasti on väljundis 230V vahelduvpinge. Seega on vaja seadet, mis suudab detekteerida vahelduvpinge olemasolu, ilma et võrgus olevad transientpinged üle kanduksid trükkplaadi teistele elektroonika osadele. Selleks on kasutatud spetsiaalselt optilist lahtisidestit HCPL-3700[21]. Järgneval selel on näha optilise lahtisidesti põhimõttelist elektroonikaskeemi.



Sele 2.7 Optilise lahtisidesti põhimõtteline elektroonikaskeem

Skeemilt on näha, et vasakus vahelduvvoolusisendid (viigud 1 ja 4 tähistusega AC) juhitakse läbi dioodsilla, mis alaldab pinge. Seejärel on hüsterees, mis likvideerib väiksema sisendil oleva müra. Viimasena on sisemine valgusdiood, mis põleb vastavalt sisendpinge olemasolul.
Väljundis on detektor, mis tuvastab sisendipoolse valgusdioodi põlemise. Detektor lülitab transistore, mis genereerivad väljundisse Vo vastava pinge, mille lugemisel mikrokontrolleriga näiteks on võimalik tuvastada sisendpinge olemasolu. Järgenvas tabelis on näha tõeväärtustabelit, sisendi sõltuvust väljundist.

Tabel 2.4 Optilise lahtisidesti tõeväärtustabel

SISEND	VÄLJUND
Kõrge nivoo	Madal nivoo
Madal nivoo	Kõrge nivoo

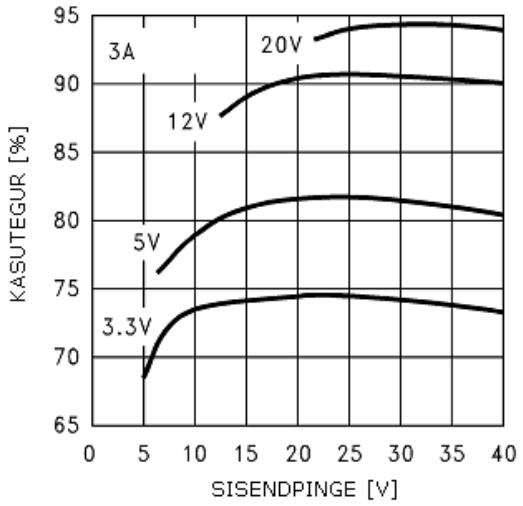
2.4.3 Impulss pingemuundur

Mobiilside mooduli toitepingeks on pingevahemik 3,2 kuni 4,8 volti. Selleks, et muundada toiteplokist tulev 12 volti pinge 4,5 voldiseks pingeks on kasutatud DC/DC muundurit.

DC/DC muundur valiti vastavalt side mooduli riistvara projekteerimisjuhistele Texas Instrumentsi LM2596[22].

DC/DC muunduri näol on tegemist elektroonikakomponendiga, mis muundab sisendpinge vastavalt soovitud väljundpingeks, kusjuures sisendpinge suurus ei sõltu väljundpingest. Antud prototüüplahendusel kasutatav pingemuundur on lülitatavat tüüpi. See tähendab, et pinge alaldamiseks lülitatakse väljundit sisse välja teatud kõrge sagedusega impulsiga nii, et pärast kondensaatoritega pinge silumist saadakse küllaltki ühtlane väljundpinge. Impulss pingemuundurite suurimaks eeliseks lineaarsete pingeregulaatorite ees on nende kõrgem efektiivsus. Lineaarsed pingeregulaatorid muundavad pinget selliselt, et kasutavad ära kogu energia, energia mis jääb pingemuundamisest üle läheb soojuseks. Näiteks olukorras, kus meil on vaja muundada 12 V pinge 4,5 V ning voolutarbimine on 1 A, siis 4,5 W läheb tarbijale ja kogu ülejäänud pinge läheb soojuseks, seega 7,5 W läheb soojuseks. Sellise muundamise korral on kasutegur kõigest 37,5%.

Küllaga kasutades impulss pingemuundurit ei ole energiakadu nii suur. Järgnevalt selelt on näha LM2596 kasutegurit.



Sele 2.8 Impulss pingemuunduri kasutegur

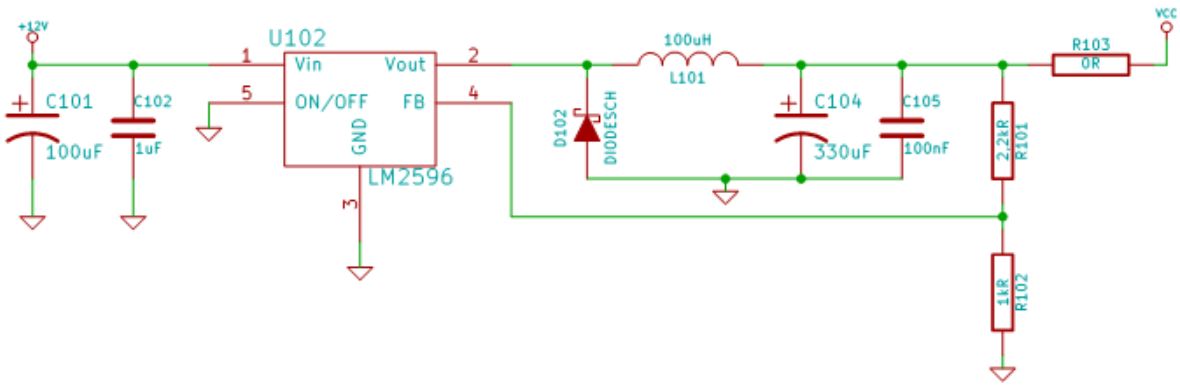
Selelt on näha, et 12 V sisenpinge juures on pinge 5 V muundamiseks kasutegur ~80 %, mis on oluliselt kõrgem kui lineaarsel pingeregulaatoril.

2.5 Elektroonika skeem

Lõputööna projekteeriti tänavavalgustuskontrolleri, elektroonika skeem. Elektroonika skeemi loomiseks kasutati vabavaralist tarkvara KiCAD, millega on võimalik luua nii elektroonika skeemi, trükkplaadi disaini, kui ka genereerida tööprogramme ning materjalide nimistut.
Antud peatükis on kirjeldatud elektroonika skeemi keerulisemaid sõlmi, kogu skeemi on näha lisast.

2.5.1 Skeemi toited

Varem kirjeldatuna kasutatakse mooduli toiteks 230 V toitepinget otse võrgust. 230 V vahelduvpinge muundab 12 V alaldispingeks spetsiaalne toitemoodul. Toitemoodul on terviklik moodul, ning selle kasutamine lisakomponente ei nõua.
Järgmise pingenivoo loomiseks kasutatakse alalispingemuundurit. Alalispingemuunduriga kaasneb küllaltki keerukas elektroonika skeem, sest konkreetne komponent nõuab palju väliseid elektroonikaelemente. Pingemuunduriga kaasnev elektroonikaskeem on loodud vastavalt mobiilside mooduli riistvara projekteerimise juhendile. Järgneval selel on näha alalispingemuunduri elektroonikaskeemi.



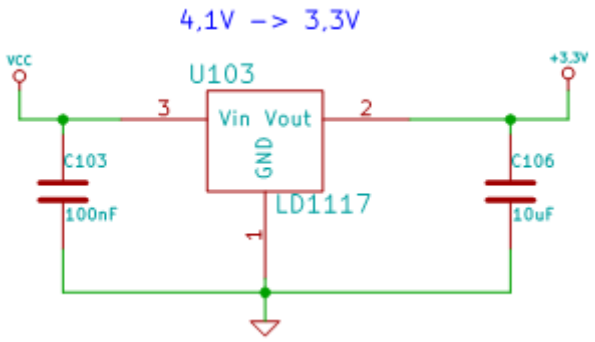
Sele 2.9 Alalispingemuunduri elektroonikaskeem

Komponent tähisega U102 on alalispingemuundur. Alalispingemuunduri toite silumiseks kasutatakse kahte kondensaatorit, üks on elektrolüüt C101 ning teine keraamiline kondensaator C102. Elektrolüütkondensaatorit kasutatakse ära tema suure mahtuvuse tõttu ning selle pärast on tal head omadused toitepinge ühtlustamiseks. Küllaga on elektrolüütkondensaatoritel suur parasiittakistus, seetõttu on lisatud sisendisse ka keraamiline kondensaator, mille koosmõjul kahe kondensaatori summaarne mahtuvus suureneb ning summaarne parasiittakistus väheneb.
Väljundis kasutatakse induktiivpooli, kondensaatorit ning kiiretoimelist dioodi väljundpinge stabiliseerimiseks. Induktiivpool piirab kondensaatori laadimise voolu ning sellega piirataksegi

kondensaatoril olevat pinget. Impulssmoodul reguleerib väljundi sisse ja välja lülitamisega kondensaatoril olevat pinget ning mõõdab seda pingejaguriga. Juhul kui väljund lülitatakse välja siis pinge induktiivpoolidel vahetab polaarsust, kiire diood avaneb ning induktiivpoolil olev energia juhitakse eemale.

Takistid R101 ning R102 on pingejagurid, millega genereeritakse sobiv sisendpinge alalispingemuunduri juhtsisendisse.

Lisaks 4,1 V pingele on kasutusel ka 3,3 V pinge. 3,3 V pingenivoo loomiseks kasutatakse tavalist lineaarset pingeregulaatorit. Lineaarse pingeregulaatori eeliseks võrreldes impulssregulaatoriga on väikene väliste komponentide maht. Olukorras kus võimsused on väiksed on seetõttu otstarbekas kasutada lineaarseid pingeregulaatoreid. Antud rakenduses kasutatakse lineaarset pingeregulaatorit vaid väikese võimsusega digitaalelektroonika toiteks ning seetõttu jäävad ka pingeregulaatoril eralduvad soojuskaod väikeseks. Järgnevalt selelt on näha lineaarse pingeregulaatori elektroonikaskeemi.

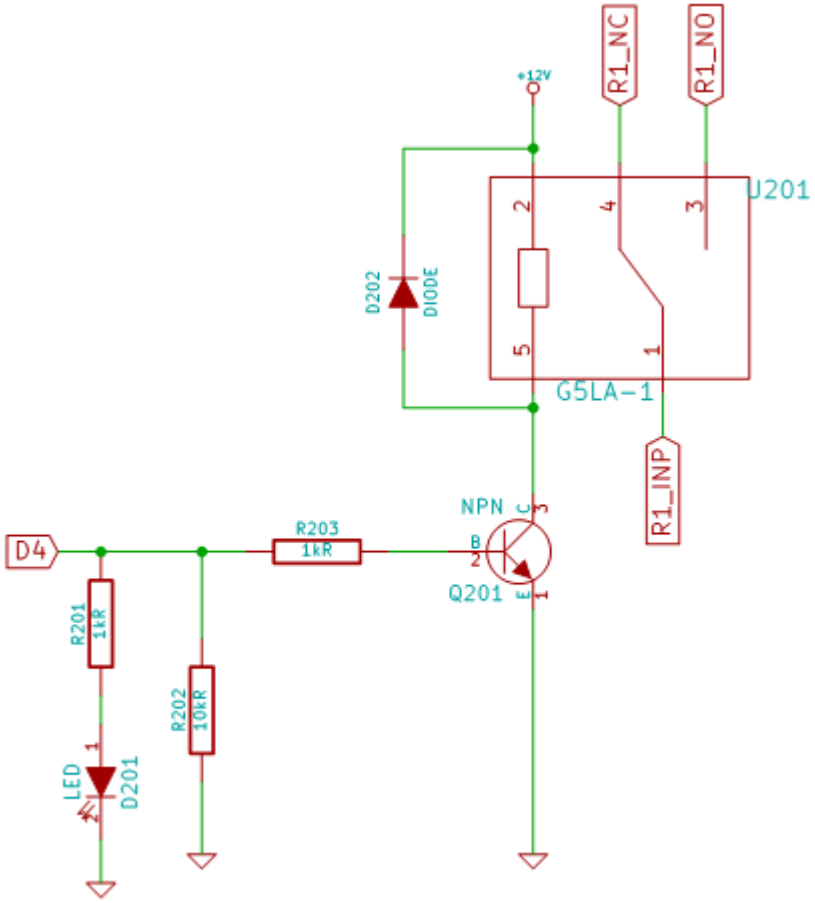


Sele 2.10 Pingeregulaatori skeem

Skeemilt on näha, et kui kasutada lineaarset pingeregulaatorit on vaja vaid kahte välist komponenti. Kondensaatoreid C103 ja C106 kasutatakse sisend ja väljundpinge stabiliseerimiseks. Võrreldes impulssregulaatoriga saab lineaarse pingeregulaatori juures kasutada väiksema mahtuvusega kondensaatoreid, sest pinget muundatakse lineaarselt.

2.5.2 Väljundi lülitamine

Väljundi lülitamiseks kasutatakse esimeses versioonis releed, seetõttu, et releed on töökindad ja lihtsad. Küllaga on relee lülitamine tülikas, sekundaarmähise suure lülitusvoolu ning induktiivsuse pärast. Järgnevalt selelt on näha relee lülitamise ahelat.

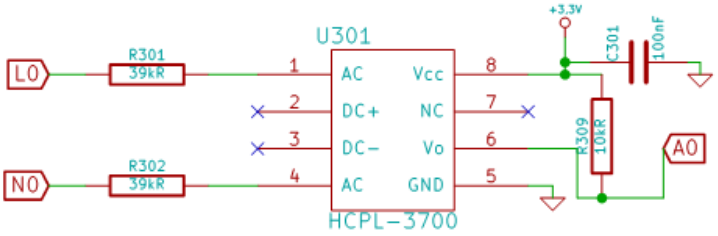


Sele 2.11 Relee lülitamise ahel

Skeemil on märgitud tähisega D4 on ühendatud mikrokontrolleri väljundiga. R1_INP, R1_NC ning R1_NO on primaarahela ühendused. Selleks, et mikrokontrolleriga releed lülitada on vaja kasutada transistori. Lisaks on skeemile lisatud valgusdiood, mis on relee oleku indikaatoriks. Kasutades NPN transistori on võimalik lülitada relee mähist 0 nivoole. Diood relee mähiste vahel on juhuks kui toide katkestatakse ning mähis on vaja laadida tühjaks.
Tavalist transistori juhitakse vooluga, seetõttu on asetatud transistori ette 1 kΩ takisti.

2.5.3 Optiline lahtisidesti

Nii nagu varem kirjeldatud kasutatakse 230 VAC tuvastamiseks optilist lahtisidestit. Optiline lahtisidesti ei ole kõik ühes elektroonikakomponent ja vajab siiski väliseid elemente. Järgnevalt selelt on näha optilise lahtisidesti elektroonikaskeemi.



Sele 2.12 Optilise lahtisidesti elektroonikaskeem

Skeemil on märgitud tähistega L0 ja N0 230 VAC sisendid mida tuvastatakse ning tähisega A0 on märgitud ühendus mikrokontrolleri sisendi viiguga. Optilise lahtisidesti sees paikneb valgusdiood, mis saab toite otse tuvastamise sisendist. Selleks, et seada valgusdioodile sobiv toitevool on tarvis kasutada väliseid takisteid, skeemil märgitud tähisega R301 ning R302. R301 ning R302 tuleb valida vastavalt valgusdioodi võimsusele. Vastavate takistitite takistused on valitud selliselt, et optilise lahtisidesti sees oleva valgusdioodil oleks toitepingeks 5 V. Takisti saab arvutada kasutades järgmist valemit.

$$R_x = \frac{V_+ - V_{TH+}}{I_{TH+}} \quad (2.1)$$

Millest V_+on toitepinge maksimaalsest pingest 60 %. 230 V vahelduvvoolu nimipinge puhul, mille maksimum on *ca* 325 V on V_+ 195 V. V_{TH+} tähistab valgusdioodi jalgadel olevat pinget. I_{TH+} on valgusdioodi poolt tarbitav vool, 2,5 mA.

Arvutame

$$R_x = \frac{195-5}{2{,}5*10^{-3}} = 38\ k\Omega \quad (2.2)$$

Teiseks tuleb silmas pidada võimsust, mida peab takisti pinge alaldamisel eraldama, selle arvutamiseks kasutame järgnevat valemit.

$$P = I^2 * R \quad (2.3)$$

Millest *I* on vool mis läbib takistit ning *R* on takisti takistus. Arvutame

$$P = 0{,}0025^2 * 38000 = 0{,}24\ W \quad (2.4)$$

Saame võimsuseks 0,24 W. Kasutatavatel pindmontaaži takistitel on maksimaalne võimsus 0,25 W, vältimaks takisti ülekuumenemist ongi kasutatud kahte takistit poole võrra väiksema takistusega. Lisaks väljundis olevale takistile on tarvilik ka Vo viigule lisada nivootakisti, skeemil tähisega R309.

2.6 Trükkplaadi disain

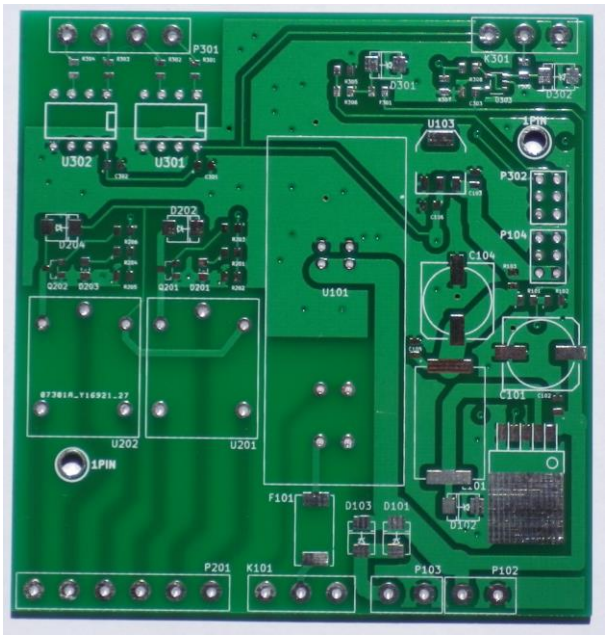
Vastavalt eelnevalt kirjeldatud elektroonikaskeemidele loodi antud lõputöö raames ka trükkplaadi riistvaraline disain. Selleks, et kogu trükkplaat oleks väiksem loodi kaks erinevat trükkplaati, selliselt on võimalik need üksteise peale paigutada ning seeläbi ruumi kokku hoida. Trükkplaadid jaotati vastavalt toidemoodul ning protsessormoodul.

2.6.1 Toitemoodul

Toitemoodul on trükkplaat kuhu on koondatud toitega seotud komponendid. Toitemoodulil paiknevad AC/DC muundur DC/DC muundur, 3,3 V pingeregulaator, optilised lahtisidestid, releed ning analoogmõõteahel. Lisaks erinevatele ahelatele paikneb toitemoodulil ka kruvidega väljundkontaktid sisenditele-väljunditele. Toitemooduli füüsilise disaini loomisel tuleb silmas pidada erinevate moodulite suurte soojuseraldustega, selleks oli tarvis luua näiteks DC/DC muunduri alla piisavalt suur vaseala, et jahutus oleks hea.

Toitemoodulil kasutatakse suuremaid voole kui signaaliahelates, sellega on vaja arvestada erinevate rajalaiuste puhul. Lisaks on toitemoodulil ka piirkonnad kus levib 230VAC vahelduvpinge, tegemist on pingega mis on inimesele ohtlik. 230 VAC alade juures tuleb arvestada suuremate radade vahekaugustega.

Järgneval selel on näha toitemooduli trükkplaati valmiskujul.



Sele 2.13 Toitemooduli trükkplaat

Nii nagu eelnevalt selelt näha on toitemooduli esimene versioon valmis toodetud. Antud moodulil katsetati erinevaid ahelaid.

2.6.2 Protsessormoodul

Protsessormoodul on trükkplaat kuhu on koondatud signaali ja protsessorahelad. Protsessormoodulil paiknevad mikrokontroller ning andmesidemoodul koos kõige juurde kuuluvaga. Võrreldes toitemooduli trükkplaadi disainiga on protsessormooduli füüsilise disain lihtsam, kuivõrd ei ole vaja arvestada erinevate radade vaheliste kaugustega ning radade jämedustega. Piisab tootjapoolt sätestatud miinimumväärtustest.

2.6.3 Trükkplaadi projekteerimine

Trükkplaadi projekteerimisel kasutati sama tarkvara, mida elektroonikaskeemi projekteerimisel. Trükkplaadi projekteerimisel, lisaks vasealade suurusele tuleb silmas pidada tootja poolt etteantud nõudeid. Esimene prototüüp valmistati Imall Iteadstudio prototüüpimis ettevõtte poolt. Antud ettevõttel rakenduvad trükkplaatidele järgmised nõuded:

1) Kihte: 1-4
2) Maksimaalsed trükkplaadi mõõtmed: 500x1100 mm
3) Minimaalsed trükkplaadi mõõtmed: 10x10 mm
4) Trükkplaadi paksus: 0,4-2,0 mm (vaikimisi 1,6 mm)
5) Minimaalne raja laius: 0,15 mm
6) Minimaalne radade vaheline kaugus: 0,15 mm
7) Raja paksus: 17-100 µm
8) Vasetatud ava läbimõõt : 0,3-6,3 mm

Järgnevalt on üles loetud projekteeritud trükkplaatide omadused:

1) 2 kihiline
2) Lakitud
3) Komponentide jalad tinatatud
4) Siiditrükitud

2.6.4 Tagasiside toitemooduli esimesest versioonist

Toitemooduli esimene prototüüp on valmis toodetud, nii nagu varem kirjeldatud. Prototüübi keerukamad funktsioonid töötasid, kuid prototüübil olid ka mõningad vead.

Prototüübil oli osade komponentide viikude asukohad valed. Teiseks siidikirjal oleks võinud olla rohkem informatsiooni, nagu viikude tähiseid jne. Üldiselt jämedaid vigu ei esinenud. Järgmisel versioonil tuleks ära parandada eelpool kirjeldatud vead ja muuta veidi komponentide asukohta, siis saaks juba tootelaadse prototüübi, millega testida toote täit funktsionaalsust.

2.7 Riistvara hind

Võib eeldada, et seadme hinnast valdav enamus on materjali hind. Seega saab kogu kontrolleri omahinnast teada kogu komplekti hinna suurusjärgu. Järgnevas tabelis on ära toodud erinevate positsioonide hinnad ning kogu riistvara hind.

Tabel 2.5 Riistvara hinna kalkulatsioon

Nr	**Positsioon**	**Hind**
1	Mikrokontroller ATmega328p	3
2	Sidemoodul SIMCOM SIM900	24
3	Toitemoodul	17
4	Alalispingemuundur	6
5	Antennipesa + antenn	10
6	Optiline lahtisidesti	4
7	Korpus	15
8	Klemmid	10
9	Trükkplaat x2	10
10	Muud elektroonikakomponendid	50
11	Komplekteerimine (töö 3h)	75
12	Testimine (töö 1h)	30
	Kokku	254

Tabelis on antud üksikkomponentide hinnad. Ostes hulgi on hinnad oluliselt väiksemad. Paljalt riistvara ehituseks kulub *ca* 250€. Koos kõige muuga võib eeldada kontrolleri hinna suurusjärku 350-400€ mis on antud valdkonnas tavaline hind.

2.8 Riistvara võimalik arengusuunad

2.8.1 Toitemooduli edasiarendused

Toitemooduli esimeseks edasiarenduseks oleks kindlasti tarvis ära parandada projekteerimisvead.

Esimene versioon trükkplaadist tehti võimalikult kindel, kasutati releesid, toited projekteeriti suured jne. Toitemoodulilt järgnevates versioonides oleks mõistlik releed vahetada väiksemate transistoride vastu, nii oleks võimalik oluliselt ruumi kokku hoida.

Teiseks probleemiks võib osutuda lülitus- ja monitoorimisahelate mittekaitsmestatus, ka nendele ahelatele oleks kindlasti tulevikus vaja lisada sisendi kaitsed.

Toitemoodulil hetkel võtab kõige enam ruumi 12 V toiteplokk, antud plokk sai valitud arvestades kõige suuremate võimsustega, mida tarbiksid peamiselt releed ja mobiilside moodul. Hetkel on selgunud tõsiasi, et projekteeritud komponendid tarbivad oluliselt vähem energiat. Mobiilside moodul võtab kõige enam voolu võrku registreerides, seega toiteprobleemid tuleks lahendada pigem suuremate kondensaatoritega ning toite osas võib komponendid asendada mõnevõrra väiksematega, kuid seda peaks kindlasti enne veel testima.

2.8.2 Protsessormooduli edasiarendused

Protsessormoodulil on esimeses versioonis kasutatud võrdlemisi lihtsa ehituse ning madalate näitajatega kontrollerit, nii nagu selgub töö tarkvara osas osutub see probleemiks, mitte küll vältimatuks, aga lihtsam oleks protsessor asendada, millegi kiiremaga.

Teiseks on hetkel kasutatud protsessormoodulil võrdlemisi vana mobiilside moodulit, jälgides edasisi andmeside arenguid jõuab kätte kindlasti aeg millal lülitatakse vanad võrgud välja ning sellisel juhul on kontroller kasutuskõlbmatu. Antud moodul tuleks asendada viimastes etappides kindlasti uuemaga, mis on kallim ning andmesidemahtude mõttes ebavajalik aga üldise arengu suhtes kindlasti kindlam valik. Tarkvaraliselt see palju ei muuda, sest nii nagu selgub töö kontrolleri tarkvara osas, siis mooduliga suhtluseks kasutatakse sarnast keelt.

3 KONTROLLERI TARKVARA

3.1 Arduino teegid

Mikrokontrolleriks kasutati Atmeli AVR seeriast Atmega328 protsessorit. Samasugust protsessorit kasutatakse ka Arduino vabavaralisel arendusmoodulil. See võimaldab kasutada ka Arduino teeke, kus on tehtud ära madalama taseme programmeerimine.

Arduino on avatud lähtekoodiga mikrokontrollerite programmeerimise keskkond, millel on oma teegid. Arduino programmeerimistarkvaraga on võimalik programmeerida Arduino nõuetele vastavaid elektroonikaseadmeid. Arduino nõuded on lihtsad, tuleb kasutada vastavaid mikrokontrollereid. Nii nagu varem kirjeldatud kasutatakse antud lõputöö raames sarnast mikrokontrollerit, mida kasutatakse Arduino Duemilanove arendusplaadil.

Arduino teegid on loodud sellisena, et ära on lahendatud enamus madalama taseme programmeerimine, nagu näiteks sisend väljund viikude deklareerimine, analoogväärtuste lugemine jne. Antud lõputöö raames kasutatakse ära järgmiseid funktsioone Arduino teekidest:

1) Viikude defineerimine
2) Analoogväärtuste lugemine
3) Väljundviikude lülitamine
4) Jadaport
5) String klass
6) EEPROM funktsoonid
7) Tarkvaraline jadaport
8) Aja funktsioonid[23]

Arduino programmeerimiskeel põhineb C++ keelel, seetõttu on seal võimalik kasutada ka kõiki C++ keele funktsioone ning üldine struktuur sarnaneb samuti C++ programmeerimiskeelele.

Arduino teekide kasutamise põhjuseks ongi peamiselt asjaolu, et sellisel juhul ei pea enam alamtaseme programmeerimist teostama ning kogu programmeerimine võtab kokku vähem aega.

Lõputöö lisades toodud programmikoodid on loodud Arduino programmeerimiskeskkonnas versiooniga 1.0.5-r2.

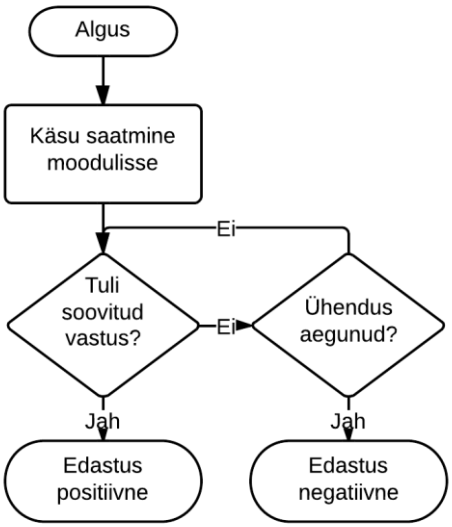
3.2 Andmesidemooduliga suhtlus

Varem kirjeldatuna käib andmesidemooduliga suhtlus kasutades AT käsustikku. Füüsilisel tasandil on kasutusel jadaport, küllaga mitte riistvaraline jadaport vaid tarkvaraline, riistvaralist jadaporti kasutatakse katsetamisel.

Andmeside mooduliga suhtlusel on ära kasutatud selle jaoks loodud teeki.

Teegis on lahendatud andmesideprotokolliga seotud probleemid. Küllaga andmesidevõrguga ühenduse loomine tuli luua.

Järgneval skeemil on näha andmesidemooduli suhtlusprotokolli plokkskeemi.



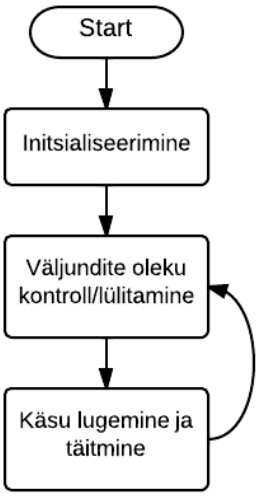
Sele 3.1 Andmesidemooduli suhtlusprotokolli plokkskeem

Selelt on näha, et kõigepealt saadetakse kogu käsk mobiilside moodulisse, misjärel jäädakse standardset vastust ootama, kui tuli oodatud vastus, siis edastus oli positiivne. Juhul kui vastust ei olnud oodatav ning ühendus on aegunud, siis edastus oli negatiivne.

3.3 Algoritmid

3.3.1 Üldine tarkvara algoritm

Kontrolleri üldist tööd võib kirjeldada lihtsa algoritmiga. Järgneval selel on näha kontrolleri üldist tarkvara algoritmi.



Sele 3.2 Kontrolleri üldine tarkvara algoritm

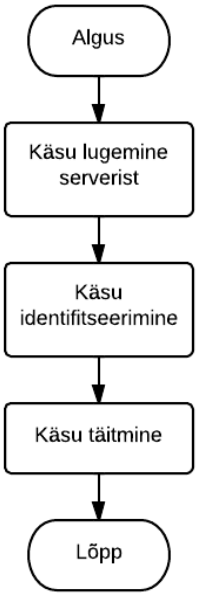
Initsialiseerimise faasis seadistatakse mikrokontroller ning mobiilside moodul. Mikrokontrolleris seadistatakse sisend-väljund viigud, jadapordid, loetakse EEPROM-ist eelmiste olekute väärtused ning seadistatakse väljundid vastavalt eelmisele tuntud seade olekule. Andmesidemooduli initsialiseerides lülitatakse moodul sisse, seadistatakse suhtlus, vajadusel avatakse SIM kaardi lukk. Lisaks seadistatakse ka pakettvõrgu pöörduspunkti nimi (edaspidi APN ingl k Access Point Name) ning määratakse andmesideühenduseks GPRS.

Järgnevalt algab pidev tsükkel. Esmalt kontrollitakse, kas väljundid on õiges olekus, nii nagu kellaaeg ütleb. Juhul kui väljund on tarvis sisse lülitada lülitatakse väljund sisse, kirjutatakse väljundi oleku info püsimällu, et katkestuse korral oleks võimalik olek taastada. Viimasena kirjutatakse relee lülitamise aeg ümber järgmisele päevale samale ajale, juhuks kui andmesideühendus ei peaks toimima serveri ja kontrolleri vahel, siis lülitub väljund ka järgneval päeval sarnaselt.

Teisena kontrollitakse, kas serveris on antud kontrolleri jaoks ootel mõni käsk, kui käsk on ootel, siis täidetakse käsk. Lihtsamaks ja enamkasutatavamaks käsuks võiks olla relee lülitamise aja muutmine.

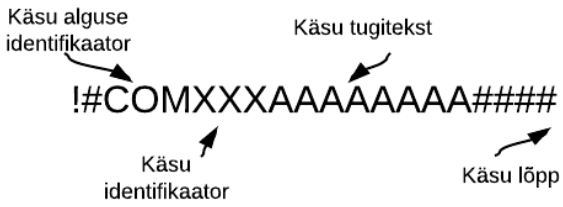
3.3.2 Käsu täitmise algoritm

Kontrolleri peamiseks ülesandeks on serverist lugeda käskusid ning vastavalt nendele täita käske. Järgneval selel on näha käsu täitmise algoritmi.



Sele 3.3 Käsu tätmise algoritm

Kõigepealt loetakse serverist käsk sisse. Seejärel toimub käsu identifitseerimine. Käsu struktuuri on näha järgneval selel.

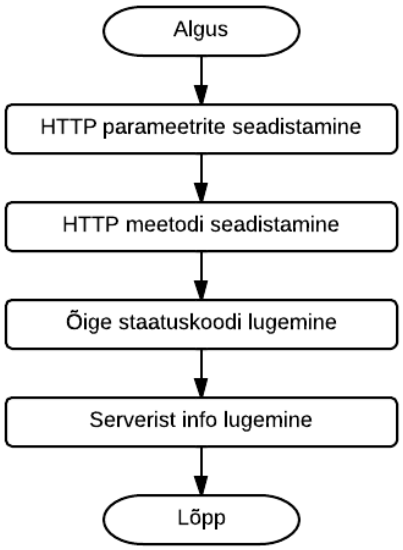


Sele 3.4 Käsu struktuur

Käsu alguse identifikaatoriks on !#COM. Seejärel järgneb kolme tähemärgiline käsu identifikaator. Peale käsu identifikaatorit on vabas pikkuses käsu tugitekst, kuhu vajadusel paigutatakse käsuga kaasnev vajalik tekst. Käsu lõpetab #### identifikaator. Antud töö käigus valminud prototüübi erinevad käsud on koos lühikeste seletusetega on toodud lisas.
Vastavalt identifitseeritud käsule täidetakse käsk.

3.3.3 Hüperteksti edastusprotkolliga suhtluse algoritm

Mobiilside moodul suhtleb serveriga kasutades hüperteksti edastusprotokolli[24] (edaspidi HTTP ingl k *Hypertext Transfer Protocol*). Suhtluse loomiseks on vajalik järgida teatud protokolli. Järgneval selel on näha algoritmi, mille alusel toimus HTTP suhtlus.



Sele 3.5 HTTP suhtluse algoritm

Esmalt tuleb seadistada erinevad HTTP parameetrid. Seejärel tuleb seadistada meetod, antud juhul GET meetod. Seejärel jäädakse ootama õiget staatuskoodi. Viimasena loetakse serverist tulev info

3.4 Programmeerimine

3.4.1 Programmi koodi ülesehitus

Programmi kood loodi Arduino programmeerimiskeskkonnas ning kood kirjutati C++ keeles. C++ keele eripäraks on klasside kasutamine. Antud töö raames valminud prototüübi programmi koodi juures kasutati samuti klasse.

Programmi kood jagati kolmeks:

- EEPROMist lugemine ning väljundite seaded
- HTTP ühendus
- Käskude täitmine ning identifitseerimine

Iga osa kohta moodustati teek, mis koosnes ühest klassist.

3.4.2 Seadete teek

Seadete teegiga on ära kirjeldatud mooduli põhiparameetrite talletamine püsimällu ning väljundite asendite lülitamise algoritmid.

Kõik moodulile vajalik informatsioon salvestatakse püsimälus, kus info säilib ka pärast mikrokontrolleri taaskäivitamist. EEPROM-i info talletamise programmi koostamisel kasutati ära Arduino teeke. Järgneval teksti väljavõttel on näha programmi osa, mis tegeleb EEPROM-i kirjutamisega.

```
/*
--------------------------------------------------------------------------------

      EEPROM SALVESTAMINE

      Funktsioon kirjutab EEPROMi
      Pikkus on sõna pikkus
      algus on sõna algus EEPROMis

--------------------------------------------------------------------------------
*/
void EEPROM_w(char *str, int pikkus, int algus)
{
      for(int i = 0; i<pikkus; i++)
      {
            EEPROM.write(i+algus,str[i]);
            // Väike delay kirjutamisel
            delay(100);
      }
}
```

Väljavõttest on näha, et madalamataseme programmeerimine Arduino teekide kasutamise tõttu on juba lahendatud. EEPROM-i kirjutamise funktsioon on kõigest 6 rida programmikoodi.

3.4.3 HTTP teek

HTTP ühenduse parameetrite seadmisteks, mooduliga suhtlemiseks ja võrgust andmete alla laadimiseks loodi samuti enda teek. Antud teegi loomise juures kasutati ära peamiselt mobiilside teeki[25]. Järgnevalt teksti väljavõttest on näha programmi koodi, mis seadistab mooduli PIN koodi.

```
/*
---------------------------------------------------------------------------

      PINi SEADMINE

      Võtab EEPROMist PINi ning seadistab selle
      Tagastab:
                        Tõene - PINi seadistamine õnnestus
                        Väär  - Pini seadistamine ebaõnnestus

---------------------------------------------------------------------------
*/
bool HTTP::setPIN()
{
      // Setpini stringi tegemine
      char cmnd[13];
      strcpy(cmnd,"AT+CPIN=");
      strcpy(cmnd+8,simPIN);
      // Laeme moodulisse
      if(gsm.SendATCmdWaitResp(cmnd,0, 2000, "OK",5)) return true;
      else return false;
}
```

Väljavõttest on näha, et mobiilside mooduliga suhtlemisel kasutatakse järgmist funktsiooni:

Gsm.SendATCmdWaitResp(cmnd,0,2000,”OK”,5)

Funktsioon saadab käsu kujul „AT+CPIN=XXXX” side moodulisse ning jääb vastust ootama, vastust oodatakse kuni 2 sekundit, vastuseks peab olema „OK”.

Analoogselt on üles ehitatud terve mooduliga suhtlus ning võrgust info alla laadimine. Kogu programmi kood on toodud töö lisas.

3.4.4 Käskude teek

Käskude identifitseerimiseks ning nende täitmiseks on käskude teek. Esimeses prototüübi versioonis on käskude teek tehtud puhtalt programmi koodi sisse kirjutatuna ning ei ole dünaamiline.

Järgneval programmi väljavõttel on näha käsu teegi juures keerulisimat algoritmi.

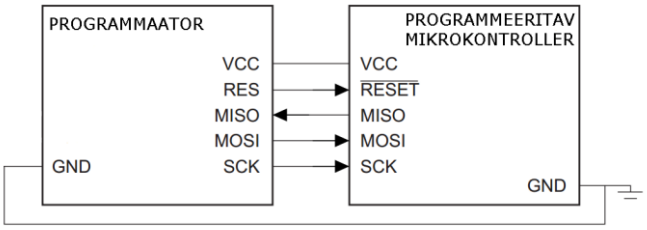
```
/*
-------------------------------------------------------------------------------

      KÄSU OTSIMINE STRINGIST

      Otsib Stringist esimese commandi ja tagastab selle.

-------------------------------------------------------------------------------
*/
void Command::CmndSearch(char *input)
{
      // Otsime käsu alguse
      uint8_t algus = strfind(input,"!#COM");
      // Juhul kui algust pole siis lihtsalt lõpetame
      if(algus == 0)
      {
            input[0]='\0';
            return;
      }
      // Eemaldame esimese otsa käsu viite
      strcpy(input,input+algus+4);
      // Otsime käsu lõpu
      uint8_t lopp = strfind(input,"####");
      // Eemaldame lõpu, kui on, kui ei ole siis teeb rea tühjaks
      if(lopp) input[lopp-1]='\0';
      else input[0]='\0';
}
```

Antud funktsioon otsib etteantud tähtede massiivist käsu ning kirjutab üles leitud käsu olemasolevasse andmemassiivi, kus edasi saab juba käsu identifitseerida ning selle täita.

Käsu identifitseerimise teegi juures on kasutatud võimalikult palju viitamisi, et kasutataks vähem muutmälu, sest käsu koodid on pikad ning muutmälu võib liiaste muutujate kasutamisel täis saada.

3.4.5 Mikrokontrolleri programmeerimine

Atmel AVR tüüpi mikrokontrolleri programmeerimine käib kasutades süsteemisisest programmeerimist (edaspidi ISP ingl k *In-System Programming*)[26]. Programmeerimiseks kasutatatakse tavalist SPI-d. Olemasoleva SPI kasutamine programmeerimise jaoks võimaldab mikrokontrollerit programmeerida ilma selle elektroonika trükkplaadilt eemaldamata. Järgneval selel on näha ühendust programmaatori ja programmeeritava mikrokontrolleri vahel.



Sele 3.6 Programmaatori ja mikrokontrolleri vaheline ühendus

Programmaatori ja programmeeritava vahel on järgnevad ühendused:

- Vcc – Toitepinge
- RESET – Mikrokontrolleri lähtestamine
- MISO – Ülema sisend alama väljund (ingl k *Master Input Slave Output*)
- MOSI – Ülema väljund alama sisend (ingl k *Master Output Slave Input*)
- SCK – Jadapordi taktsagedus (ingl k *Serial Clock*)
- GND – 0 nivoo

Programmeerimisel on programmaator ülema rollis ning programmeeritav mikrokontroller alama rollis. Normaalses tööolekus peab AVR mikrokontrolleri lähtestamise viik olema ühendatud kõrge pingenivooga.

Programmeerimise esimeseks sammuks on lähtestamise viigu nivoo viimine 0 V pingenivooni. Kohe pärast lähtestamise viigu lülitamist on programmeeritav mikrokontroller valmis minema programmeerimise režiimi. Nüüdsest on võimalik programmaatoril saata programmeerimise režiimi mineku käsk programmeeritavale.

Seejärel programmeeritakse mikrokontroller vastavalt ettenähtud protokollile. Mikrokontroller kasutab programmi mälu hoidmiseks välkmälu, mistõttu programmeerimisel eriti suuri võimsuseid ei tarbita ning pinge kõikumise vms pärast muret ei pea tundma.

Kõik see on loodud selleks, et kontrollerit oleks võimalik võimalikult lihtsate vahenditega ja kiireresti programmeerida.

Antud lõputöö raames kasutatud prototüüpelektroonika programmeerimisel kasutati võrdlemisi odavat ning ehituselt lihtsalt firma Sparkfun Electronics programmaatorit[27].

3.4.6 Programmeerimise tagasiside

Prototüüpelektroonika peal katsetati esmast programmikoodi, selleks, et mitte toota koheselt elektroonika ning alles pärast tootmist avastada, et projekteerimises on tehtud vigu.

Mikrokontrollerid on loodud võrdlemisi lihtsate ülesannete täitmise jaoks ning võrgusuhtlus on vanemate mikrokontrollerite võimekuse piiril. Prototüübi loomiseks valiti mikrokontroller, millel oleks suur välkmälu programmi jaoks. Peamiseks valiku põhjuseks oli prototüüp programmikoodi vähene optimeeritus ning suurt programmimälu kasutades on võimalik mikrokontrollerisse talletada ka optimeerimata programmikoode. Küllaga katsetamise käigus tekkis väga palju algselt seletamatuid anomaaliaid. Selgus, et optimeerimata programmikood, mis moodustab *ca* 50% välkmälu mahust ei ole ainukene argument, mida peab programmeerimise poole pealt arvestama. Varem kirjeldatuna kasutatakse mikrokontrolleritel muutujate salvestamiseks SRAM-i, mida konkreetsel mikrokontrolleril on 2 kbaiti.

Programmeerides kasutati algselt laialdaselt Arduino objekti String, mis sisuliselt on dünaamiline tähtmuutujate massiiv. Antud andmetüübi kasutamine võimaldas programmikoodi oluliselt lihtsustada, küllaga suuremahulise programmikoodi puhul tekkis probleem SRAM-i ületäitumisega ning seetõttu läks kontroller ebastabiilseks, ega täitnud käske õigesti. Hilisemas programmikoodis kasutati võimalikult palju viitamisi ning dünaamiline andmemassiiv asendati fikseeritud massiiviga, et ei tekiks ületäitumist.

Probleem tekib peamiselt, sellest, et mobiilside mooduliga suhtlusel kasutatakse palju erinevaid käskusid, mis kõik on salvestatud SRAM-i staatiliste muutujatena.

Järgneval programmi väljavõttel on programmi funktsioon, millest on näha SRAM-is talletavat käsku.

```
/*
-------------------------------------------------------------------------------

      GPRS SEADISTAMINE

      Seadistab GPRS ühenduse
      Tagastab:
                        Tõene - GPRSi seadistamine õnnestus
                        Väär  - GPRSi seadistamine ebaõnnestus

-------------------------------------------------------------------------------
*/
bool HTTP::setGPRS()
{
      if(gsm.SendATCmdWaitResp("AT+SAPBR=3,1,\"Contype\",\"GPRS\"",0,  2000,
"OK",5)) return true;
      else return false;
}
```

Eelnevalt väljavõttelt on näha, et staatilised muutujad, mis on programmi sisse kirjutatud teksti kujul nagu selel „AT+SAPBR=3,1\"Contype\",\"GPRS"" kirjutatakse SRAM-i staatilise muutujana. SRAM-i ületäitumise probleemi on võimalik vähendada, kui kirjutada osad muutujad SRAM-i asemel programmi mällu, mida on rohkem.

3.5 Tarkvara edasine arendus

Käesoleva töö raames lahendati tarkvaraliselt ära peamiselt töö raskuspunktiks olnud andmesideühenduse osa. Kontrolleri üheks eeliseks teiste ees peaks olema tema paigalduse ja esmase tööle rakendamise lihtsus. Kontrolleri esmase tööle rakendamise lihtsus on antud hetkel veel lahendamata. Eelduseks on, et kui kontroller paigaldada olemasolevasse kilpi ning ühendades see hämarlülitiga peab kontroller hakkama tööle nii nagu tavaline seade koos hämarlülitiga.

Lisaks eelnevale on vaja siiski pärast installeerimist võimaldada mõningane seadistamine elektriku poolt. Riistvaraliselt on selleks lisatud seadmele viigud kuhu on võimalik paigaldada teine andmesidemoodul, mis suhtleks näiteks lähivõrgu abil elektriku nutiseadmega, mille abil oleks võimalik kirjutada kontrollerisse esmased seadistused.

Kõik uuemad arendused soovivad juba lahendust, kus igat tänavavalgustit oleks võimalik üks haaval kontrollida ja jälgida. Sellisel juhul oleks tarvilik kasutada ära vaba andmesidemoodul loomaks andmesideühendust mobiilside mooduli ja üksikute tänavavalgustite vahel. Selline lähenemine nõuaks tõenäoliselt ka mikrokontrolleri välja vahetamist, sest juba antud prototüübil hakkas muutujate mälu täis saama.

4 ANDMEVAHETUS SERVERIGA

4.1 Avatud süsteemide sidumise arhitektuur

Antud peatükis kirjeldatakse kontrolleri ja serveri vahelist andmevahetust ning serveri algoritme.

Andmevahetusprotokolle klassifitseeritakse avatud süsteemide sidumise arhitektuuri (edaspidi OSI ingl k. *Open System Interconnection*)[28] konseptuaalse mudeli järgi. Antud mudeli järgi on sõnumi edastamiseks vajalikud funktsioonid jagatud ära seitsme kihi vahel. Antud mudeli järgi suhtlevad kihid otseselt vaid oma naaberkihtidega. Tänapäeval kasutatakse OSI mudelit peamiselt andmesidevõrkude kirjeldamise lihtsustamiseks.

Järgnevas tabelis on näha tabeli kujul erinevate kihtide tõlgendusi.

Tabel 4.1 OSI mudel

	Andmeühik	**Nr**	**Kiht**	**Funtsioon**	**Näide**
Võrguseadme kihid	Andmed	7	Rakenduskiht	Võrgu suhtlus rakendustega	HTTP
		6	Edastuskiht	Andmeedastus ja krüpteerimine	SSL
		5	Seansikiht	Osapoolte vahelise ühenduse kontrollimine	TCP
	Segment	4	Transpordikiht	Määrab ära andmeedastuse sokli (pordi)	TCP
Edastus-keskkonna kihid	Pakett	3	Võrgukiht	Marsruutimine ja loogiline adresseerimine	IP
	Kaader	2	Andmelüli kiht	Andmepakettide muundamine binaarkoodiimpulssideks, veatuvastus	MAC
	Bitt	1	Füüsiline kiht	Võrgumeediumi töö. Bitide edastamine	RS-232

Tabelis on väga paljude lihtsustustega kirjeldatud OSI mudel. OSI mudeliga on võimalik ära kirjeldada, millised võrgukihid on võrgupakkuja poolt lahendatud ning millised on tarvilik ise lahendada.

Kasutades tavalisi kommertsvõrke on OSI mudeli järgi kihid 1-5 on lahendatud võrguteenuse pakkuja poolt. Sellisel juhul tuleb lihtsalt jälgida, kas valitud võrguseade vastab võrguteenuse pakkuja poolt kasutatavatele protokollidele.

Kasutades hüperteksti edastusprotokolli on lahendatud ära nii kuues kui seitsmes kiht.

4.2 Hüperteksti edastusprotokoll

Andmesidevahetusel kasutatakse ära hüperteksti edastusprotokolli. Avatud süsteemide sidumise arhitektuuri mudeli järgi paikneb HTTP rakenduskihis.

HTTP on mõeldud tekstiliste andmete ülekandmiseks üle võrgu. HTTP-d on kasutatud veebis juba alates 1990. aastast. HTTP-sid on erinevaid versioone antud töös kasutatakse versiooni 1.1.

HTTP töötab kui päring-vastus süsteem. Kus klient, milleks antud juhul on mobiilside moodul, edastab serverile päringu ning server pärast päringu töötlemist saadab moodulile vastuse.

HTTP ressursid identifitseeritakse kasutades ühtset ressursi-identifikaatorit (edaspidi URI ingl k *Uniform Resource Identificator*). URI-d viitavad pöördutava serveri aadressile ning esitatakse kujul http://XXXX.

HTTP-l on erinevad päringu meetodid. Järgnevalt on kirjeldatud kolme põhilist meetodit:

- GET – Klient saadab kogu info URI väljal ning server tagastab vastavalt sellele kliendile info
- POST – meetod, kus klient saadab info vormi siseselt, antud meetod on turvalisem kui GET meetod.
- HEAD – sarnaneb GET meetodile, kuid tagasi soovitakse lugeda vaid HTTP päiseid

Antud töös kasutatakse GET meetodit. See tähendab et kogu vajaminev info edastatakse URI väljal. URI väli võiks välja näha järgmine:

http://www.mingiserver.ee?id=aaa5b

Antud juhul edastatakse lisaandmed serverile pärast küsimärki. Ning nende lisaandmete töötlemisel koostab server päringule vastuse.

Server saadab päringu vastuseks alati olekukoodi. Tüüpilisemad olekukoodid koos seletusega on toodud järgnevas tabelis.

Tabel 4.2 HTTP olekukoodid

Kood	Tekst	Tähendus
200	OK	Kõik on korras, päring õnnestus
202	Accepted	Päring on vastu võetud, töödeldakse
301	Moved	Leht on teise kohta viidud
302	Found	Lehte sellest kohast ei leitud, kuid leiti mujalt
400	Bad Request	Halb päringu süntaks
401	Unauthorized	Antud lehele minekuks ei ole luba
403	Forbidden	Serveril puuduvad antud lehele õigused
404	Not Found	Lehte ei leitud
500	Server Error	Serveril on tõrge
502	Service Overloaded	Server on ülekoormatud

Eelnevas tabelis on kirjeldatud põhilisemad olekusõnumid, neid on rohkem.

Antud töös kasutatava HTTP ühendust on monitooritud programmiga Wireshark. Järgneval selel on näha kliendi päringut serverile.

```
GET / HTTP/1.1\r\n
Host: 83.180.8.208:8080\r\n
```

Sele 4.1 Päring serverisse

Selelt on näha, et serverile edastatakse GET meetodiga päring, lisaks märgitakse ära millist protokolli versiooni kasutatakse. Viimasena antakse URI info, selel Host: taga.

Järgneval selel on serveri vastus päringule.

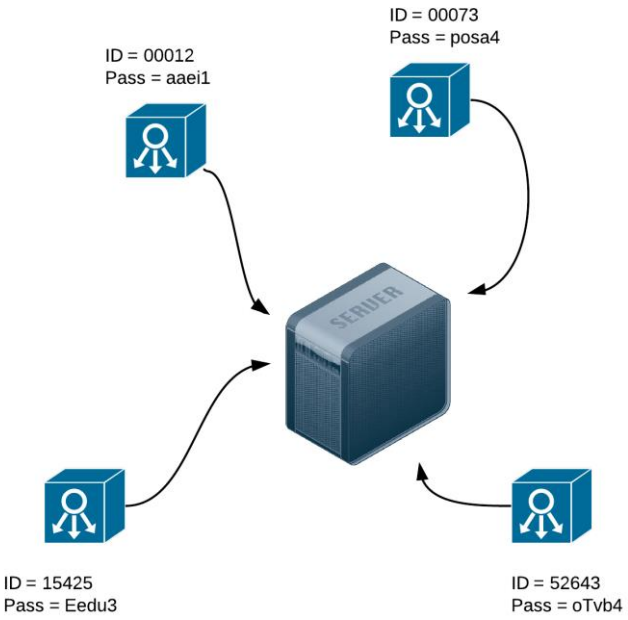
```
⊞ HTTP/1.1 200 OK\r\n
  Content-type: text/html; charset=utf-8\r\n
  Transfer-Encoding: chunked\r\n
  Date: Wed, 14 May 2014 19:45:10 GMT\r\n
  Server: lighttpd/1.4.32 - Palapa Web Server (http://alfanla.com)\r\n
  \r\n
  [HTTP response 1/1]
  [Time since request: 0.175509000 seconds]
  [Request in frame: 137]
⊞ HTTP chunked response
Line-based text data: text/html
  !#COM00613998287484###
```

Sele 4.2 Serveri vastus päringule

Selelt on näha, et server tagastab olekukoodina 200 OK. Lisaks tagastab server päringu kellaaja, serveri tarkvara, päringu aja, lisaks võib veel päises olla teksti formaat teksti pikkus jne. Viimasel real on näha ka serveri tagastatud käsklust moodulile.

4.3 Serveriga suhtlus

Varem kirjeldatuna kasutatakse serveri ja kontrolleri omavaheliseks suhtluseks hüperteksti edastusprotokolli. HTTP on klient-server süsteem ning vajalik oleks et iga klient näeks serverist sellist infot, mis on talle tarvilik sealt näha. Selleks on otstarbekas kasutada klientide identifitseerimist. Serveriga identifitseerimiseks on igal seadmel oma identifikaator ning parool. Järgneval selel on näha skemaatiliselt identifikaatorite ja paroolide jaotust.



Sele 4.3 Identifikaatorite ja paroolide jaotus

Nii nagu varem kirjeldatud kasutatakse edastuse puhul GET meetodit. See tähendab et kogu info, mis klient saadab serverile peab kajastuma aadressiribal. Käesoleva töö jaoks serveripoolse näidiskoodi jooksutamiseks on kasutatud tavalist nutitelefoni. Nutitelefoni kasutamise miinuseks on asjaolu, et osad pordid on suletud, nende hulgas ka tavapärane HTTP port 80. Seega tuleb kasutada erinevat porti ning ka pordi info peab olema kajastatud aadressiribal. Seega aadressiriba peab välja nägema sarnaselt nagu on näha järgneval selel.

Sele 4.4 Aadressiriba

Selelt on näha et esimesena kirjutatakse serveri interneti protokolli aadress, seejärel port ning kõige lõpus serverisse saadetav info.

4.3.1 Dünaamiline veebilehekülg

Hüperteksti preprotsessor (edaspidi PHP ingl k *Hypertext Preprocessor*)[29] on süntaks, mida kasutatakse dünaamiliste veebilehtede loomisel. PHP on serveripoolne protokoll. Kui tavaline, lihtsam veebilehtede süntaks hüperteksti märgistuskeel (edaspidi HTML ingl k *Hypertext Markup Language*) kuvatakse kogu pikkuses ka kasutajale ning kasutajapoolne veebilehitseja loob lähtekoodist ise veebilehekülje kujunduse, siis PHP koodi kasutaja ei näe. PHP programmeerimisskript on loodud selleks, et peita kasutaja eest serveripoolset rakendust. Seda kasutataksegi peamiselt olukorras, kus erinevatele kasutajatele on tarvis näidata erineva sisuga veebilehti.

Antud lõputöö raames loodi lihtne serveripoolne veebileht, mis suudab tuvastada ühendunud seadme võttes aadressiribalt vajalikud parameetrid.

Lisaks on võimalik kasutada PHP skripti andmebaaside ja muu vajaliku rakendustega suhtlemisel.

Järgneval programmi väljavõttel on näha programmikoodi PHP skriptist, mis võimaldab serveril identifitseerida klient ning tagastab kliendile vajaliku info.

```
// Võtame seadme id ja passwordi parameetrid URList
$id=$_REQUEST["id"];
$pass=$_REQUEST["pass"];
// Kontrollime ID ja passwordi
// Aadressi näide => http://ipaadress?id=00001&pass=aaa11
if (($id == 00001)&&($pass == "aaa11"))
{
      // Siia vastus mida kuvatakse
      echo "!#COM00613998287 48####";
}
```

Väljavõttelt on näha et vajalikud parameetrid loetakse sisse $_REQUEST käskudega. Antud käsuga loetakse sisse aadressiribal olev info. Järgnevalt on tingimuslause, mille süntaks sarnaneb paljuski C keelega. Echo funktsiooniga tagastatakse veebilehitsas kuvatav info.

4.4 Serveripoolne algoritm

Selleks, et testida esimest prototüüpi loodi lihtne programmikood, kuid reaalses kasutuses oleks vaja siduda serveri tarkvara omavahel nii andmebaasiga kui ka kasutajaliidesega.

Serveri poolel peaks paiknema kõigepealt veebileht, millest saavad info kontrollerid analoogselt eelpool kirjeldatuna. Laiemas kasutuses on tarvis, et ka klient saadaks serverisse infot. Selleks oleks kaks võimalust:

1) Serverisse saadetakse infot läbi URI välja nii nagu ka identifikaator ja parool
2) Vahetatakse ära GET meetod POST meetodi vastu ning siis on võimalik kliendile ka muid kanaleid viisi infot saata

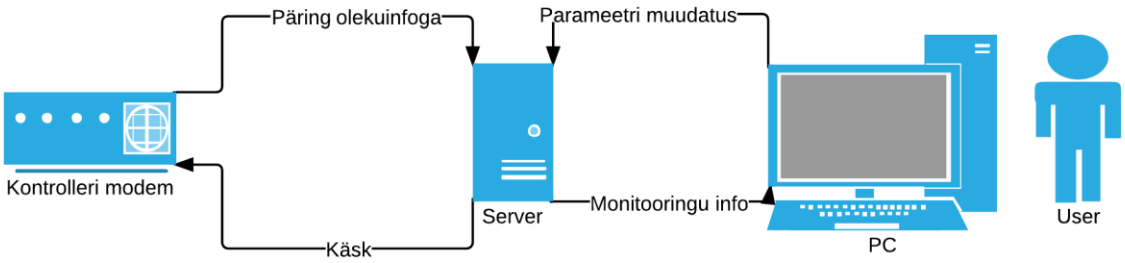
Andmete hoiustamiseks tuleks serveri poole peal kasutada kindlasti mingisugust andmebaasi. Laialt on levinud relatsiooniliste andmebaaside kasutamine, millest enamikega suhtlemiseks on võimalik kasutada PHP skripte. Serveripoolses andmebaasis peaks talletama enamiku süsteemi info, sest kontrolleri poolel on mälu vähem. Süsteemi loomulikuks tööks oleks tarvis et serveripoolses andmebaasis talletataks vähemalt järgnev info:

1) Erinevate seadmete identifikaatorid, mis võivad ühenduda serveriga
2) Erinevatele kontrolleritele vastavad paroolid
3) Kontrollerite sisse ja väljalülitamise kellaajad
4) Kontrolleri väljundite olekud
5) Erinevad häired
6) Ootel olevad käsud

Käskusid tuleks saata kontrollerisse üks haaval, sest kontrolleri mälumaht, milles talletatakse käsk töötlemise käigus on väike. See tingib asjaolu, et on tarvilik kasutada ootenimekirja.

Lisaks kontrollerile peab oma info saama serverist kätte ka kasutaja. Selleks on tarvis serveripoolne kasutajaliides, kus kasutaja saab seadistada erinevaid kontrollereid ning jälgida võrke.

Järgneval selel on näha skemaatiliselt, kuidas käib kasutajal suhtlus kontrolleriga.



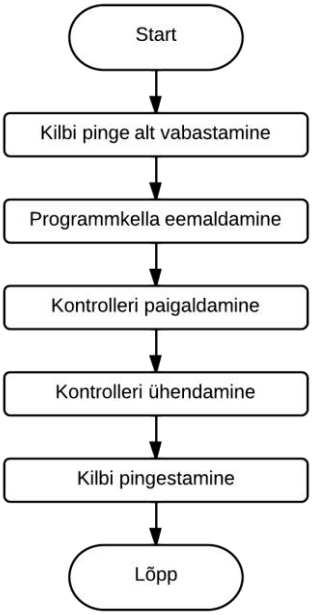
Sele 4.5 Serveri suhtlus kontrolleri ja kasutajaga

Selline keeruline suhtlemise skeem on loodud turvalisuse tõttu. Oluliselt turvalisem on kui klient pöördub ise serveri poole mitte vastupidi. Server on enamasti turvatud erinevate turvaalgoritmide ja programmidega, kuid kontrolleris oma lihtsuse tõttu neid pole. Seega tuleb eos juba vältida turvariske põhjustavaid olukordasid. Kasutaja ei saa otse kontrolleriga ühenduda, seetõttu et kui kasutada sarnast algoritmi otse kasutajaliideses oleks tarvis, et kasutaja ootaks kuni kontroller võtaks temaga ise ühendust ning sooviks saada seadistuse parameetreid. Mugavamaks tööks ongi tarvis asetada kasutaja ja kontrolleri vahele server, mis vahendab kasutaja infot kontrollerisse.

5 TUGEVVOOL

5.1 Kilpi ühendamise põhimõte

Käesolevas peatükkis on kirjeldatud lõputöö tugevvoolulist osa. Antud kontrolleri prototüüp on projekteeritud lülitama tänavavalgusteid. Antud peatükis kirjeldatakse Kontrolleri ühendamist olemasolevasse või uude tänavavalgustuse lülitusjaotusseadmesse.
Kontrolleri ühendamisega olemasolevasse võrku peab saama hakkama tavaline elektrik, seega ei tohi erineda ühendamise skeem tavalistest elektriskeemidest. Ühendamise protsessi üldist kontseptsiooni on näha järgneval selel.



Sele 5.1 Kilpi ühendamise protsess

Selelt on näha elektriku tööprotsesse. Pärast pingestamist hakkab kontroller tööle valgusanduriga(kui on installeeritud), hiljem lisades seadme enda andmebaasi on võimalik seade ka ära seadistada.
Prototüüpversioonis on lahendamata eelseadistus ja releede lülitamine hämarlülitiga. Samuti on planeeritud, seadmele lisa andmesidemoodul, selleks et kohapeal saaks näiteks tahvelarvutiga seadistada võrgu, SIM-i ning serveri parameetrid. Viimasena kirjeldatud funktsioonid lahendatakse järgnevates prototüübi versioonides.
Lisas on näha kaughallatava tänavavalgustuse kontrolleriga kilbiskeemi.

6 KOKKUVÕTE

Käesoleva lõputöö eesmärgiks oli projekteerida kaughallatava tänavavalgustuskontrolleri prototüüp.

Lõputöö uurimustöö osas kirjeldati tavalist tänavalgustuse juhtimist, analüüsiti ühte sarnast süsteemi, anti väike ülevaade elektrinõuetest, analüüsiti erinevaid andmete ülekandmisvõimalusi mobiilside võrgus, tehti levi test ning formuleeriti lõplik lähteülesanne.

Tavalist tänavavalgustussüsteemi juhitakse enamasti programmkella või hämarlüliti alusel ühes kilbis paiknevate kontaktoritega. Antud süsteemi peamiseks eeliseks on tema lihtsus ja töökindlus, küllaga võib puudusena välja tuua energiasäästureziimi lülituse lisamise keerukuse ning liinide oleku jälgitavuse puudumise.

Tallinnas kasutatakse tänavavalgustuse distantsjuhtimist, mille on välja arendanud ettevõte KH Energia Konsult. Tegemist on sellise süsteemiga, mida on võimalik integreerida ka olemasolevatesse mitte kaugjuhitavatesse võrkudesse. Süsteemi suurimaks eeliseks võib lugeda tema kaughaldamise võimekust. Süsteemi suurimaks puuduseks võib lugeda tema liigset keerukust, tähendades, et süsteemi oskavad ja kasutavad ainult eelmainitud ettevõte ning ilma käiduta seda soetada ei ole võimalik.

Euroopa liidus toodetavad elektriseadmed peavad kandma CE märgistust, mille saamiseks peavad nad põhiliselt vastama elektromagneetilise ühilduvuse direktiivile. Direktiivi kohaselt peab seade eraldama vaid sellisel määral elektromagnetlaineid, et see ei häiriks teiste seadmete tööd ning samas ise olema töökindel teiste seadmete poolt põhjustatud elektromagnetväljas.

Andmete ülekandmisel serveri ja kontrolleri vahel kasutatakse olemasolevat mobiilsidevõrku, sest kontrollerid paiknevad hajusalt väga suure maa-ala peal ning eraldi andmesidevõrku ainult tänavavalgustuse jaoks välja ehitada ei ole otstarbekas. Andmete ülekandmisel mobiilsidevõrgus on peamiselt kolm võimalust: helistamisteenus, lühisõnumite abil ning mobiilandmeside. Helistamisteenus on võrdlemisi turvaline, sest autenditakse üksteis telefoninumbrite abil, mida on raske jäljendada. Helistamisteenuse suurimaks puuduseks on, et kõnekeskusega saab korraga ühenduses olla vaid üks kontroller. Lühisõnumite saatmine ja autentimine on lihtne. Puudusena võib esile tuua sõnumiside kalliduse, mis sõltub peamiselt andmevahetuse sagedusest. Mobiilandmeside on ühendustest üks keerulisemaid, kuid odavam võrreldes lühisõnumite teenusega, tänu väiksele andmemahule. Mobiilandmeside on jällegi ebaturvaline, seetõttu on otstarbekas kasutada teenust, kus kontroller on klient ja keskseade server.

Lõputöö kontrolleri riistvara osas on ära kirjeldatud riistvara projekteerimisega seotud teemad. Ära on kirjeldatud kasutatav korpus, mikrokontrolleri valik, mikrokontrolleri ehitus, mis on vajalik antud töö seisukohast, mobiilside mooduli tööpõhimõte, teised olulised elektroonikakomponendid, elektroonika skeemi olulisemad kohad ning trükkplaadi projekteerimine.

Korpus on valitud ostutoode, mis on võimalik paigaldada standardsele 35 mm profiiliga liistule. Mikrokontrolleri valiku olulisimaks kriteeriumiks oli, et seadmel oleks olemas internetist kättesaadavad teegid, mis lihtsustavad prototüüptarkvara loomist. Mikrokontrolleri valik tehti peamiselt kolme Atmel AVR seeria mikrokontrolleri vahel: ATmega328p, ATmega32u4 ning ATmega2561. Valituks osutus ATmega328p oma soodsa hinna, väikeste välismõõtmete ning esmapilgul rahuldavate tehniliste näitajate poolest.

Mikrokontroller on elektroonikakomponent, milles on ühendatud protsessor, mälud, sisend-väljundid ning andmesidemoodulid. AVR tüüpi mikrokontrolleril on RISC protsessor. RISC tüüpi protsessor on oma lihtsuse tõttu mikrokontrolleritel laialt levinud. Programmi mäluna kasutatakse mikrokontrolleril välkmälu. SRAM-is talletatakse mikrokontrolleri viikude info jaoks registrid ning sellest kahte kilobaiti kasutatakse programmi muutujate hoidmiseks. Lisaks on mikrokontrolleril ka EEPROM tüüpi mälu, kuhu on võimalik salvestada andmeid, mida on tarvis säilitada ka pärast kontrolleri taaskäivitust.

Mikrokontrolleril on 20 viiku mida on võimalik kasutada sisend-väljunditena. Lisaks tavalisele digitaalse sisend-väljundi funktsioonile on viikudele võimalik määrata ka alternatiivseid funktsioone nagu analoog-digital muundur jne. Viikude defineerimine ning nende oleku lugemine ja määramine käib läbi registrite.

Mobiilside mooduliks valiti SIMCOM SIM900 moodul. Tegemist on mobiilside mooduliga, millega on võimalik kasutada mobiilandmeside. Mobiilside mooduliga on võimalik suhelda kasutades laialt levinud AT käsustikku. Seda tüüpi käsustikku kasutatakse paljude kasutajapoolsete telekommunikatsiooniseadmetel.

Mobiilside moodul kasutab andmesideks üldist pakettandmesidet, mis on võrdlemisi vana tehnoloogia. Pakettandmeside on kommertslikult jagatud erinevatesse põlvkondadesse, neist viimane 4 põlvkond on kõige viimasena kasutusele võetud. Põlvkonnad erinevad kasutaja jaoks peamiselt andmeside kiiruste poolest, kuid ka tehniliselt on nad väga erinevad.

Skeemi toiteks on 230 V vahelduvpinge, mis alaldatakse 12 V peale kasutades spetsiaalset toitemoodulit.

230 V vahelduvpinge tuvastamiseks on kasutatud optilist lahtisidestit. Optiline lahtisidesti on elektroonikakomponent, mis koosneb valgusdioodist ning valgustundlikust elemendist.

Valgusdioodi toiteks kasutatakse tuvastatavat pinget. Pingestatuna lülitab valgustundlik element transistori ning signaali väljund lülitatakse madalale pingenivoole.

Mobiilside mooduli toiteks vajalik 4,1 V pingenivoo luuakse kasutatakse impulss pingemuundurit, tema kõrge kasuteguri pärast.

Elektroonika skeem on toodud lisas.

Trükkplaadi projekteerimisel jaotati elektroonika kaheks: protsessormoodul ja toitemoodul. Toitemoodulil paiknevad sisendite väljundite klemmid, toiteskeemid, optiline lahtisidestus, väljundite lülitus ning analoogsisendi muundamine. Protsessormoodulil paiknevad mikrokontroller, mobiilside moodul ning pesa lisa andmesidemooduli jaoks. Trükkplaadi projekteerimisel kasutati tootja poolt etteantud nõudeid ning komponentide andmelehtedel toodud infot. Toitemooduli prototüüp on ühtlasi valmis toodetud ning enamus funktsioonid testitud. Probleemideks olid mõningased projekteerimisvead ning siiditrükil olnud vähene info. Edasise arenduse käigus tuleks asendada toitemoodulil olevad releed pooljuhtidega ning parandada ära projekteerimisvead. Protsessormoodulil tuleks mikrokontroller asendada võimekamaga ning mobiilsidemoodul uuemaga.

Kontrolleri tarkvara kirjeldavas osas on kirjeldatud Arduino teeke, erinevaid programmi algoritme, loodud programmi teeke, mikrokontrolleri programmeerimisest ning tarkvara edasisest arengusuundadest.

Kontrolleri tarkvara loomiseks kasutati Arduino programmeerimiskeskkonda, koos kaasasolnud teekidega, mis lihtsustas programmeerimist.

Andmesidemooduliga suhtlus käib käsu saatmise ja soovitud vastuse ootamise põhimõttel. Üldiselt on programmil algoritmiliselt kolm etappi: initsialiseerimine, väljundite kontroll ja lülitamine ning käsu lugemine ja täitmine. Käsu lugemisel kõigepealt käsk identifitseeritakse, selleks on loodud eraldi käsu struktuur, mida on näha lisas. Seejärel täidetakse käsk. Käsk võetakse serverist kasutades hüperteksti edastusprotokolli.

Programmeerimisel loodi kolm teeki: seadete teek, HTTP teek ning käskude teek. Seadete teegis on kirjeldatud ära väljundite lülitamine ning EEPROM-iga suhtlus. Käskude teegis on ära kirjeldatud käsu identifitseerimine ning täitmine. HTTP teegis on ära kirjeldatud mobiilside mooduliga suhtlus, selleks et serverist laadida käsu infot.

Mikrokontrolleri programmeerimiseks kasutatakse SPI andmesiini. Mikrokontrolleri käivitamisel on SPI reserveeritud programmeerimiseks, pärast programmeerimise käsu saatmist on võimalik kontroller programmeerida.

Programmeerimisel kasutati elektroonika arendusplaati, millel oli kasutusel sama mikrokontroller ning mobiilsidemoodul, mis projekteeritaval kontrolleril. Programmeerimise

käigus selgus, et kontrolleril jääb väheseks SRAM-i, peale seda hakati programmeerimisel kasutama rohkem viitamisi.

Tarkvara edasises arengufaasis tuleks lisada väljundi oleku tuvastus ning esmasel käivitusel peaks kontroller tööle hakkama nagu varasemalt kasutatud tavaline hämarlülitiga lahendus.

Serveriga andmevahetuse peatükis on põgusalt kirjeldatud andmeside ülesehitust, hüperteksti edastusprotokolli, serveriga suhtlust ning serveri algoritmi.

Andmevahetusprotokolle klassifitseeritakse OSI mudeli järgi, mille alusel jaotatakse andmesidevõrk seitsmeks kihiks, alustades andmeside füüsilisest tasandist ja lõpetades rakendusega.

Lõputöö käigus on kasutatud hüperteksti edastusprotokolli mis paigutub OSI mudeli järgi kahele kõrgemale kihile. Alumised kihid on lahendatud võrguteenuse pakkuja poolt.

Serveriga suhtluse testimiseks loodi dünaamiline veebilehekülg, kasutades PHP skripti. Server võtab lehekülje aadressiväljalt info ning vastavalt sellele tuvastab kontrolleri ning kuvab identifitseeritud kontrollerile vajaliku info.

Reaalses kasutuseks oleks vaja serveripoolne tarkvara ühendada nii andmebaasiga kui kasutajaliidesega. Andmebaasis hoitaks kontrolleritega seotud informatsiooni, erinevaid lülitusaegasid, olekuid, kontrollerite identifikaatoreid jne. Kasutajaliidese kaudu peaks lõppkasutaja saama seadistada kontrollereid.

Tugevvoolu osas on põgusalt ära kirjeldatud kontrolleri ühendamine olemasolevasse tänavavalgustuse juhtimiskilpi. Juhtimiskilpi ühendamine peab olema nii lihtne, et sellega saaks hakkama ka tavaline elektrik. Kilbi näidisskeem on toodud töö lisas.

7 SUMMARY

The aim of the master thesis was to design a prototype of remotely managed street lighting controller. It had to be able to switch street lighting and monitor their state from central unit with the controller.

Usual street lighting management, analysis of one similar system, brief overview of electrical requirements, analysis of different communication opportunities in mobile network, signal strength test and the initial task were described in the research part of the work.

Street lighting is commonly switched with contactors based on daytime or dawn switch. The main advantage of this common system is its simplicity and reliability. The main disadvantage is the lack of energy save mode and inability to monitor the state.

Street lighting distance management is used in Tallinn, it has been developed by KH Energia Konsult. This system can be integrated into existing networks. The biggest advantage of this system is distance management. The biggest disadvantages are its excessive complexity, it means that the system can only be managed or created in the company and without handling it can't be purchased.

All electrical products which are made in European Union have to have CE marking. To get the marking all products must meet the electromagnetic compatibility directive. According to the directive the product can only radiate such amount of electromagnetic waves that it won't interfere other devices and the device itself have to work without errors disturbed by electromagnetic waves emitted from other devices.

General mobile network is been used to exchange data between controller and server, it's not rational to build separate communication network just for street lighting management, because controllers are wide apart in big area. There is three main opportunities to exchange data in mobile network: call service, short message service and data service. Call service is quite secure, because the authentication is based on caller numbers, which are hard to copy. The biggest disadvantage of call service is that there can be only one caller at time. Sending short messages and authentication of the sender is simple. On the other hand short messaging tends to be expensive. Mobile data service is one of the hardest to implement, but it is cheaper than short messaging service, because of the small amount of data. On the other hand mobile data service is more unsecure, therefore it is rational to use service, where controller is client and central unit is server.

Hardware design of controller is described in the hardware design part. Casing, microcontroller selection, brief overview of microcontroller architecture, mobile data service module, other

important electronic components, important electrical schema parts and layout design is described.

Case is chosen on the basis of mounting on 35 mm profile rail. The most important aspects choosing microcontroller were libraries available for programming. The selection of microcontroller was made between three AVR microcontrollers: ATmega328p, ATmega32u4 and ATmega2561. ATmega328p were chosen because of its cheap price, small size and satisfying technical specification.

Microcontroller is electronic component that consists of processor, memory, input-output module and data communication modules. AVR microcontroller has RISC processor. RISC processors are wide spread in microcontrollers. Flash memory is used for program memory. 2 kilobytes for program variables and pin configuration registers are stored in SRAM. Microcontroller has also EEPROM, where is stored data, that needs to be stored after controller power down.

Microcontroller has 20 pins, that can be used as input-outputs. In addition to usual input-output alternative functions like analogue-digital converter etc. can be also addressed for pins. Defining pins and reading their state is done through registers.

SIMCOM SIM900 is chosen for mobile service communication module. The module can be used for mobile data communication. The communication between mobile service module and microcontroller is held by using AT commands. This type of commands are widely used in user sided telecommunication devices.

Mobile communication module is using general radio packet service for data communication, which is relatively old technology. Radio packet service is divided into different generations the last 4^{th} generation is lastly used. Different generations distinguish for users mainly from different data speeds, but technically they are also different.

Electronics is powered from general 230 V AC network, which is converted into 12 V DC using special power module.

Optical isolators are used to detect 230 VAC. Optical isolator is device that consists of light emitting diode and light sensitive element. Light emitting diode is powered directly from detective line. When the diode is powered the light sensitive element will switch the transistor and the signal output is switched to ground.

4,1 V is needed for powering mobile service module. That voltage is generated using step-down voltage regulator, because of its high efficiency.

Electronics were divided into two while designing printed circuit board layout: processor module and power module. On the power module there is: input-output terminals, power

electronics, optical isolation, output switching and analogue converter. On the processor module there are: mobile service module, microcontroller and socket for extra data module. Producer requirements were used for designing. Power module is fabricated and almost tested. There were some design faults and there could have been more info on silk screen. For further development the design faults should be fixed and relays should be replaced with transistors. On the processor module the microcontroller and mobile service module should be replaced.

On the software part there are described the following: Arduino libraries, different algorithms, created libraries, programming microcontroller and further development.

Arduino programming environment was used to develop software.

The communication between mobile service module and microcontroller is based on the sending instruction and receiving the right answer.

General program has three stages: initialisation, output check and switching, instruction reading and execution. Instruction is identified after reading, instruction set is in the appendix. After that the instruction is executed. Instructions are taken from server using HTTP.

Three different libraries were made: HTTP library, setup library and commands library. Switching outputs and communication with EEPROM is described in setup library. Instruction identification and execution is described in commands library. Communication with mobile service module is described in HTTP library.

SPI is used for programming microcontroller. SPI is reserved for programming on the power up, after receiving programming instruction the microcontroller enters programming mode.

Electronic prototype board with similar hardware were used for programming. Different pointers are used through program because of the lack of SRAM in microcontroller.

On the further software development, detection of 230 VAC is needed to be added and on the initial start-up the controller should work with dawn switch like previously.

Server's algorithm, hypertext transfer protocol and data communication are described in server part. Data communication protocols are classified using OSI model. OSI model divides data communication network into seven different layers starting from physical layer and ending with application layer.

HTTP, which is on the 6-7th layer on the OSI model, is used through the work. Lower levels are specified by network service provider.

Simple web page, using PHP script, was made for testing the communication. Server takes information from URI and according to that identifies the controller and responses with the information needed.

In the real situation servers software should be connected with database and user interface. Controller information should be stored in the database. The user should be able to setup controllers using user interface.

Connection of controller to existing street lighting cabinet is described on the high current part. The installation should simple enough for usual electrician. The example of cabinet schematics is in the appendixes.

KASUTATUD KIRJANDUS

[1] ProSystem OÜ kodulehekülg [WWW] http://www.prosystem.ee (17.05.2014)

[2] Ericsson Eesti AS kodulehekülg [WWW] http://www.ericsson.com/ee (17.05.2014)

[3] Elisa Eesti AS kodulehekülg [WWW] http://www.elisa.ee (18.05.2014)

[4] KH Energia Konsulti kodulehekülg [WWW] http://www.energia-konsult.ee (17.05.2014)

[5] KiCAD EDA kodulehekülg [WWW] http://www.kicad-pcb.org (17.05.2014)

[6] Autodeski kodulehülg [WWW] http://www.autodesk.com (17.05.2014)

[7] Lucidcharti kodulehekülg [WWW] https://www.lucidchart.com (17.05.2014)

[8] Microsoft Office kodulehekülg [WWW] http://office.microsoft.com/et-ee (17.05.2014)

[9] Arduino kodulehekülg [WWW] http://www.arduino.cc (17.05.2014)

[10] IteadStudio kodulehekülg [WWW] http://imall.iteadstudio.com (17.05.2014)

[11] WireShark kodulehekülg [WWW] http://www.wireshark.org (17.05.2014)

[12] Euroopa parlamendi ja nõukogu direktiiv 2004/108/EÜ 15. detsember 2004, Euroopa parlament ja Euroopa Liidu nõukogu

[13] SIMCOM SIM900 kodulehekülg [WWW] http://wm.sim.com/producten.aspx?id=1019 (19.05.2014)

[14] Camdenboss CNMB kodulehekülg [WWW] http://www.camdenboss.com/enclosures/din-rail-enclosures/cnmb-standard-kit (19.05.2014)

[15] ABB S-200 andmeleht [WWW] http://www05.abb.com/global/scot/scot209.nsf/veritydisplay/44c5b5ac208f3f25c1257ad7004ec86a/$file/2CDC002157D0202_view.pdf (19.05.2014)

[16] Arduino kodulehekülg [WWW] http://arduino.cc/ (19.05.2014)

[17] AVR ATmega 328P andmeleht [WWW] http://www.atmel.com/Images/doc8161.pdf (19.05.2014)

[18] AT command set for User Equipment (UE). Digital cellular telecommunications system (Phase 2+). Universal Mobile Telecommunications System (UMTS). TS 27.007 versioon 8.3.0 väljaanne 8. 3GPP

[19] Euroopa Telekommunikatsiooni Standardi Instituudi kodulehekülg [WWW] http://www.etsi.org (19.05.2014)

[20] Wellemann PSS1215M toitemooduli koduleheküg [WWW] http://www.velleman.eu/products/view/?country=be&lang=en&id=346181 (19.05.2014)

[21] Avaga HCPL-3700 optilise lahtisidesti koduleheküg [WWW] http://www.avagotech.com/pages/en/optocouplers_plastic/isolated_voltage_current_detector/hcpl-3700 (19.05.2014)

[22] Texas Instrumentsi LM2596 koduleheküg [WWW] http://www.ti.com/product/lm2596 (19.05.2014)

[23] Aja teegi koduleheküg [WWW] http://www.pjrc.com/teensy/td_libs_Time.html (19.05.2014)

[24] Hüperteksti edastusprotokolli andmeleht [WWW] http://www.w3.org/Protocols/rfc2616/rfc2616.html (19.05.2014)

[25] Mobiilside teegi koduleheküg [WWW] https://github.com/jgarland79/GSM_Shield (19.05.2014)

[26] Süsteemisisese programmeerimise andmeleht [WWW] http://www.atmel.in/images/doc0943.pdf (19.05.2014)

[27] Sparkfun Pocket programmaatori andmeleht [WWW] https://www.sparkfun.com/products/9825 (19.05.2014)

[28] Open Systems Interconnection – Basic Reference Model: The Basic Model. Information Technology. ISO/IEC 7498-1. Rahvusvaheline standardiseerimise organisatsioon.

[29] PHP arenduse koduleheküg [WWW] http://www.php.net (19.05.2014)

LISAD

Lisa 1: Seaded teegi pealdis

```
/*
      Versioon
      GSM mooduli seadete teek
      Madis Lepiksaar, Mai 2014
*/
#ifndef Seaded_H
#define Seaded_H

#include <Arduino.h>
#include <EEPROM.h>
#include <Time.h>

// Defineerime releede arvu
#define RELEE_ARV 2

class Seaded
{
      private:
      // Relee PINide massiiv
      static int relee_pin[RELEE_ARV];
      // Releede aegade massiiv esimene on nr ja teine on positsioon
      uint32_t relee_aeg[RELEE_ARV][4];
      public:
      Seaded();
      void write_seadmeID(char *id);            //   Seadme   ID   kirjutamine
EEPROMi
      void read_seadmeID(char *id);       // Seadme ID lugemine EEPROMist
      void write_seadmePWD(char *pwd);    // Seadme pwd kirjutamine EEPROMi
      void read_seadmePWD(char *pwd);           //    Seadme    pwd    lugemine
EEPROMist
      void write_simPIN(char *pin);       // SIMi PINi kirjutamine EEPROMi
      void read_simPIN(char *pin);        // SIMi PINi lugemine EEPROMist
      void write_APN(char *APN);                // võrguop.  APN  kirjutamine
EEPROMi
      void read_APN(char *apn);                               // APNi lugemine
EEPROMist
      void write_server(char *server);    //   Serveri   aadressi   kirjutamine
EEPROMi
      void read_server(char *server);                         //         Serveri
aadress lugemine EEPROMist
      void relee_olek_w (bool aktiivsus, uint8_t nr);       //   relee   oleku
määramine
      bool relee_olek_r (uint8_t nr);                                 //    relee
oleku lugemine
      void relee_setup();                                             //  releede
setup algseadistus enne käivitust
      void relee_sw(uint8_t nr, bool asend);          // Releede lülitamine
      void relee_aeg_w(uint8_t nr, uint8_t pos, uint32_t aeg);   //  Releede
lülitamise aeg
      uint32_t relee_aeg_r(uint8_t nr, uint8_t pos);                    //
Releede lülitamise aja lugemine
      void relee_kontroll();                                          //
Kontrollib aegasid ja lülitab relee

};
#endif
```

Lisa 2: Seaded teek

```
/*
      Versioon 3
      GSM mooduli seadete teek
      Madis Lepiksaar, Mai 2014
*/

#include "Seaded.h"
#include <Arduino.h>
#include <EEPROM.h>
#include <Time.h>

/*
-------------------------------------------------------------------------------

      RELEEDE PINide MASSIIV
      Defineerime ära relee pinid
      NB! analoog on 14-19

-------------------------------------------------------------------------------
*/
      int Seaded::relee_pin[RELEE_ARV]={4,5};
/*
-------------------------------------------------------------------------------

      EEPROM SALVESTAMINE

      Funktsioon kirjutab EEPROMi
      Pikkus on sõna pikkus
      algus on sõna algus EEPROMis

-------------------------------------------------------------------------------
*/
void EEPROM_w(char *str, int pikkus, int algus)
{
      for(int i = 0; i<pikkus; i++)
      {
            EEPROM.write(i+algus,str[i]);
            // Väike delay kirjutamisel
            delay(100);
      }
}
/*
-------------------------------------------------------------------------------

      EEPROM LUGEMINE

      Funktsioon loeb EEPROMist

-------------------------------------------------------------------------------
*/
void EEPROM_r(char *str,int pikkus, int algus)
{
      // Loeme mälupesast 5 tähemärki
      for(int i = 0; i<pikkus; i++)
      {
            str[i]=(char)EEPROM.read(i+algus);
      }
}
/*
```

```
------------------------------------------------------------------------------

      EEPROM LUGEMINE

      Funktsioon loeb EEPROMist

------------------------------------------------------------------------------
*/
String EEPROM_r(int pikkus, int algus)
{
      String str = "";
      // Loeme mälupesast 5 tähemärki
      for(int i = 0; i<pikkus; i++)
      {
            str+=(char)EEPROM.read(i+algus);
      }
      return str;
}
/*
------------------------------------------------------------------------------

      SEADED KONSTRUKTOR

------------------------------------------------------------------------------
*/
Seaded::Seaded()
{
      // Releede olekute lugemine EEPROMist
      // Kontrollime niipalju releesid kui meil on
      for(int i = 1; i<(RELEE_ARV+1); i++)
      {
            for(int j = 1; j<5; j++)
            {
                  // Loeme eepromist aja ja kirjutame massiivi
                  relee_aeg[i-1][j-1]=relee_aeg_r(i,j);
            }
      }
}

/*
------------------------------------------------------------------------------

      SEADME ID LUGEMINE

      Seadme ID pikkus 5 tähemärki
      Seadme ID EEPROMi pesad 0-4
      Tagastab ID

------------------------------------------------------------------------------
*/
void Seaded::read_seadmeID(char *id)
{
      EEPROM_r(id,5,0);
}
/*
------------------------------------------------------------------------------

      SEADME ID KIRJUTAMINE

      Seadme ID pikkus 5 tähemärki
      Seadme ID EEPROMi pesad 0-4
```

```
-------------------------------------------------------------------------
*/
void Seaded::write_seadmeID(char *id)
{
      EEPROM_w(id, 5,0);
}
/*
-------------------------------------------------------------------------

      SEADME PASSWORDi KIRJUTAMINE

      Seadme salasõna kirjutamine EEPROMi
      Seadme salasõna pikkus 5 tähemärki
      Seadme ID EEPROMi pesad 5-9

-------------------------------------------------------------------------
*/
void Seaded::write_seadmePWD(char *pwd)
{
      EEPROM_w(pwd, 5,5);
}
/*
-------------------------------------------------------------------------

      SEADME PASSWORDi LUGEMINE

      Seadme salasõna lugemine EEPROMi
      Seadme salasõna pikkus 5 tähemärki
      Seadme ID EEPROMi pesad 5-9
      Tagastab parooli

-------------------------------------------------------------------------
*/
void Seaded::read_seadmePWD(char *pwd)
{
      EEPROM_r(pwd,5,5);
}
/*
-------------------------------------------------------------------------

      SIMi PIN KIRJUTAMINE

      Seadme PIN kirjutamine EEPROMi
      SIMi PINi pikkus 4 tähemärki
      SIMi PIN EEPROMi pesad 10-14

-------------------------------------------------------------------------
*/
void Seaded::write_simPIN(char *pin)
{
      EEPROM_w(pin, 4,10);
}
/*
-------------------------------------------------------------------------

      SIMi PIN LUGEMINE

      Seadme PIN lugemine EEPROMist
      SIMi PINi pikkus 4 tähemärki
      SIMi PIN EEPROMi pesad 10-14
```

```
      Tagastab PINi

-------------------------------------------------------------------------
*/
void Seaded::read_simPIN(char *pin)
{
      EEPROM_r(pin,4,10);
}
/*
-------------------------------------------------------------------------

      APN KIRJUTAMINE

      Seadme APNi kirjutamine EEPROMi

      APNi pikkus 25 tähemärki
      APNi pikkus kirjutatakse 14 mälupesale pikkus 1 byte
      APNi EEPROMi pesad 15-40

-------------------------------------------------------------------------
*/
void Seaded::write_APN(char *APN)
{
      // Leiame pikkuse
      int pikkus = strlen(APN);
      // Kirjutame pikkuse
      EEPROM.write(14,pikkus);

      // Kirjutame APNi
      EEPROM_w(APN,pikkus,15);
}
/*
-------------------------------------------------------------------------

      APN LUGEMINE

      Seadme APNi lugemine EEPROMist

      APNi pikkus 25 tähemärki
      APNi pikkus kirjutatakse 14 mälupesale pikkus 1 byte
      APNi EEPROMi pesad 15-40

-------------------------------------------------------------------------
*/
void Seaded::read_APN(char *apn)
{
      // Leiame pikkuse
      uint8_t pikkus = EEPROM.read(14);

      // Loeme APNi
      EEPROM_r(apn,pikkus,15);
}
/*
-------------------------------------------------------------------------

      SERVERI AADRESSI KIRJUTAMINE

      Serveri aadressi kirjutamine EEPROMi

      Aadressi pikkus 39 tähemärki
      Aadressi pikkus kirjutatakse 80 mälupesale pikkus 1 byte
```

```
      Aadressi EEPROMi pesad 41-79

-------------------------------------------------------------------------------
*/
void Seaded::write_server (char *server)
{
      // Leiame pikkuse
      int pikkus = strlen(server);
      // Kirjutame pikkuse
      EEPROM.write(80,pikkus);

      // Kirjutame serveri adre
      EEPROM_w(server,pikkus,41);
}
/*
-------------------------------------------------------------------------------

      SERVERI AADRESSI LUGEMINE

      Serveri aadressi lugemine EEPROMist

      Aadressi pikkus 39 tähemärki
      Aadressi pikkus kirjutatakse 80 mälupesale pikkus 1 byte
      Aadressi EEPROMi pesad 41-79

-------------------------------------------------------------------------------
*/
void Seaded::read_server (char *server)
{
      // Leiame pikkuse
      uint8_t pikkus = EEPROM.read(80);

      // Loeme serveri adre
      EEPROM_r(server,pikkus,41);
}
/*
-------------------------------------------------------------------------------

      RELEE OLEKU KIRJUTAMINE

      Relee oleku määramine

      Aktiivsus
            1. TÕENE - relee on sees
            2. VÄÄR - relee on väljas
      NR - Relee nr 1-8

      Jagama 8 bit ära 8 relee vahel

      EEPROM pesa 81

-------------------------------------------------------------------------------
*/
void Seaded::relee_olek_w (bool aktiivsus, uint8_t nr)
{
      // Loene eepromist hetkeoleku
      int olek = EEPROM.read(81);
      // Kontrollime kas nr on normaalses vahemikus
      if((nr>0)&&(nr<9))
      {
            // Lahutame nr-ist 1e sest bitid on 0-7
```

```
            nr-=1;
            // Juhul kui soovitakse tõeseks panna
            if(aktiivsus)
            {
                  olek = olek | (1 << nr);
            }
            else olek = olek & ~(1 << nr);
      }
      else Serial.println("ERROR");
      // Kirjutame EEPROMi
      EEPROM.write(81,olek);
}
/*
-------------------------------------------------------------------------------

      RELEE OLEKU LUGEMINE

      Relee oleku lugemine

      Tagastab
            1. TÕENE - relee on sees
            2. VÄÄR - relee on väljas
      NR - Relee nr 1-8

      Jagama 8 bit ära 8 relee vahel

      EEPROM pesa 81

-------------------------------------------------------------------------------
*/
bool Seaded::relee_olek_r (uint8_t nr)
{
      // Loome puhvri
      int puhver;
      // Kontrollime kas relee on ikka õigesti sisestatud
      if((nr>0)&&(nr<9))
      {
            // Lahutame nr-ist ühe
            nr-=1;
            // Loeme EEPROMist puhvrisse
            puhver = EEPROM.read(81);
            // Loogiline korrutus ,et järele jätta ainult biti väärtus
            puhver = puhver & (1<<nr);
            return puhver>>nr;
      }
      else
      {
            Serial.println("ERROR");
            return false;
      }
}
/*
-------------------------------------------------------------------------------

      RELEE SETUP

      Releede algseadistamine

      Siin määratakse koodis ära millised PINid on kasutused releedel
      Loetakse sisse releede endised olekud ning need taastatakse
```

```
------------------------------------------------------------------------------
*/
void Seaded::relee_setup ()
{
      // Defineerime väljundid ja määrame oleku
      for(int i=0; i<RELEE_ARV; i++)
      {
            // Määrame väljundiks
            pinMode(relee_pin[i], OUTPUT);
            // Juhul kui olek on sees siis lülitame sisse
            if(relee_olek_r(i+1)) digitalWrite(relee_pin[i], HIGH);
            else digitalWrite(relee_pin[i], LOW);
      }

}
/*
------------------------------------------------------------------------------

      RELEE LÜLITAMINE

      Releede lülitamine
      NR - relee nr
      ASEND - true sees, false välja

      Lülitab vastava relee õigesse asendisse kirjutab ka tulemuse EEPROMi

------------------------------------------------------------------------------
*/
void Seaded::relee_sw(uint8_t nr, bool asend)
{
      // Kontrollime relee nri
      if((nr>0)&&(nr<RELEE_ARV+1))
      {
            // Kirjutame EEPROMi
            relee_olek_w(asend,nr);
            // Muudame väljundi
            if(asend) digitalWrite(relee_pin[nr-1],HIGH);
            else digitalWrite(relee_pin[nr-1], LOW);
      }
}
/*
------------------------------------------------------------------------------

      RELEEDE LÜLITAMISE AJAD

      Releede lülitamise aegade kirjutamine EEPROMi
      NR - relee nr
      POS - lülituspositsioon
            1 - astronoomile kellaga sisselülitus
            2 - astronoomilise kellaga väljalülitus
            3 - lisa sisselülitus
            4 - lisa väljalülitus
      aeg - Unix aeg sekundites alates 1 jaanuar 1970.
      igal ajal on 4byte ning iga relee võtab 4x4 byte ehk 16 byte
      alustame 82 EEPROM pesalt
      Märgib vastava soovitud lülitusaja EEPROMi

------------------------------------------------------------------------------
*/
void Seaded::relee_aeg_w(uint8_t nr, uint8_t pos, uint32_t aeg)
{
```

```
	// Kontrollime kas relee nr ja positsioon on õige
	if((nr>0)&&(nr<9)&&(pos>0)&&(pos<5))
	{
		// Kirjutame massiivi
		relee_aeg[nr-1][pos-1]=aeg;
		// Teisendame relee nri mälupesa nriks
		nr = 82+(nr-1)*16+(pos-1)*4;
		// Kirjutame EEPROMi
		EEPROM.write(nr,aeg);
		EEPROM.write(nr+1,aeg>>8);
		EEPROM.write(nr+2,aeg>>16);
		EEPROM.write(nr+3,aeg>>24);
	}
	else Serial.println("ERROR");
}
/*
--------------------------------------------------------------------------

	RELEEDE LÜLITAMISE AEGADE LUGEMINE

	Releede lülitamise aegade lugemine EEPROMist
	NR - relee nr
	POS - lülituspositsioon
		1 - astronoomile kellaga sisselülitus
		2 - astronoomilise kellaga väljalülitus
		3 - tavaline sisselülitus
		4 - tavaline väljalülitus
	aeg - Unix aeg sekundites alates 1 jaanuar 1970.
	igal ajal on 4byte ning iga relee võtab 4x4 byte ehk 16 byte
	alustame 82 EEPROM pesalt

--------------------------------------------------------------------------
*/
uint32_t Seaded::relee_aeg_r(uint8_t nr, uint8_t pos)
{
	// Kontrollime kas relee nr ja positsioon on õige
	if((nr>0)&&(nr<9)&&(pos>0)&&(pos<5))
	{
		uint32_t tagastus=0;
		// Teisendame relee nri mälupesa nriks
		nr = 82+(nr-1)*16+(pos-1)*4;
		for(int i = 0; i<4; i++)
		{
			tagastus = tagastus <<8;
			tagastus = tagastus + EEPROM.read(nr+3-i);
		}
		return tagastus;
	}
	else Serial.println("ERROR");
}
/*
--------------------------------------------------------------------------

	RELEEDE LÜLITAMISE KONTROLL

	Kontrollitakse releede lülitamise vajadust
	Juhul kui releed lülitatakse, siis muudetakse ka järgmine lülitus
järgmisele päevale

--------------------------------------------------------------------------
*/
```

```
void Seaded::relee_kontroll()
{
	// Loeme sisse praeguse aja
	uint32_t time = now();
	// Kontrollime niipalju releesid kui meil on
	for(int i = 1; i<(RELEE_ARV+1); i++)
	{
		// Kõigepealt kontrollime astronoomilise kellaga sisselülitust
		if(time>relee_aeg[i-1][0])
		{
			// Lülitame sisse
			relee_sw(i, true);
			// Liigutame aja järgmisele päevale järgides ast aega
			// TODO
			// Liigutame lihtsalt järgmisele päevale
			relee_aeg[i-1][0]+=86400;
			// Kirjutame EEPROMi sama aja
			relee_aeg_w(i,1,relee_aeg[i-1][0]+86400);
			// Kirjutame Serialisse
			Serial.print("R");
			Serial.print(i);
			Serial.println("ON");
		}
		// Kontrollime astronoomilise kellaga väljalülitust
		if(time>relee_aeg[i-1][1])
		{
			// Lülitame välja
			relee_sw(i, false);
			// Liigutame aja järgmisele päevale järgides ast aega
			// TODO
			// Liigutame lihtsalt järgmisele päevale
			relee_aeg[i-1][1]+=86400;
			Serial.println(relee_aeg[i-1][1]);
			// Kirjutame EEPROMi sama aja
			relee_aeg_w(i,2,relee_aeg[i-1][1]+86400);
			// Kirjutame Serialisse
			Serial.print("R");
			Serial.print(i);
			Serial.println("OFF");
		}
		// Kontrollime tavalist sisselülitust
		if(time>relee_aeg[i-1][2])
		{
			// Lülitame sisse
			relee_sw(i, true);
			//  Liigutame  aja  järgmisele  päevale  ilma  astr  aja
arvamiseta
			relee_aeg_w(i,3,relee_aeg[i-1][2]+86400);
			// Kirjutame Serialisse
			Serial.print("R");
			Serial.print(i);
			Serial.println("ON");
		}
		// Kontrollime tavalist väljalülitust
		if(time>relee_aeg[i-1][3])
		{
			// Lülitame välja
			relee_sw(i,false);
			// Liigutame järgmisele päevale ilma astr aja arvamiseta
			relee_aeg_w(i,4,relee_aeg[i-1][3]+86400);
			// Kirjutame Serialisse
```

```
                    Serial.print("R");
                    Serial.print(i);
                    Serial.println("OFF");
                }
            }
        }
```

Lisa 3: HTTP teegi pealdisfail

```
/*
      Versioon 2
      GSM mooduli HTTP suhtluse teek
      Madis Lepiksaar, Mai 2014
*/
#ifndef HTTP_H
#define HTTP_H

#include <Arduino.h>
#include <EEPROM.h>
#include "Seaded.h"
#include "GSM.h"
#include <SoftwareSerial.h>
#include <Time.h>
// Käsu pikkuse
#define KASU_PIKKUS 3
// Defineerime GSM mooduli reset ja power pinid
#define GSMRESET 7
#define GSMPWR 6
// Parameetrite id-d
#define PARAM_SERVER 1
#define PARAM_APN 2
#define PARAM_PIN 3
#define PARAM_SEADMEID 4
#define PARAM_SEADMEPWD 5

class HTTP
{
      private:
      // GSM classi defineerimine
      GSM gsm;
      // EEPROM seadete defineerimine
      char seadmeID[6];
      char seadmePWD[6];
      char APN[25];
      char server[39];
      char simPIN[5];
      char error[5];;
      public:
      // Seadete klassi defineerimine
      Seaded GSMmoodul;

      HTTP();                              // Võtab seadme ID-d ja muud
parameetrid EEPROMist
      void initGSM();         // GSMmooduli initsialiseerimine
      void setParam(char *txt, uint8_t param); // Seadistab uuesti valitud
parameetri
      bool setPIN();                // PIN koodi seadistamine
      bool setGPRS();               // GPRSi seadistamine
      bool setAPN();                // APNi seadistamine
      bool startHTTP();       // LOOB HTTP ühenduse
      bool stopHTTP();        // Lõpetab HTTP ühenduse
      void HTTPget(char *data, uint8_t pikkus);      // Loob HTTP ühenduse
ja tagastab stringi
      void fullHTTPget(char *data,uint8_t pikkus);    // Käivitab GSM mooduli
loob HTTP ühenduse ja tagastab stringi
};
#endif
```

Lisa 4: HTTP teek

```
/*
      Versioon 2
      GSM mooduli GSM mooduli HTTP suhtluse teek
      Madis Lepiksaar, Mai 2014
*/

#include <Arduino.h>
#include <EEPROM.h>
#include "Seaded.h"
#include "GSM.h"
#include <SoftwareSerial.h>
#include "HTTP.h"
#include <Time.h>

/*
-------------------------------------------------------------------------------
      HTTP KONSTRUKTOR

      Loeb EEPROMist parameetrid

-------------------------------------------------------------------------------
*/
HTTP::HTTP()
{
      // Errori string
      strcpy (error, "ERROR");
      //    EEPROM ei tööta siis kirjutame käsitsi need read
      /*
      GSMmoodul.read_seadmeID(seadmeID);
      GSMmoodul.read_seadmePWD(seadmePWD);
      GSMmoodul.read_simPIN(simPIN);
      GSMmoodul.read_APN(APN);
      GSMmoodul.read_server(server);
      */
      //    EEPROM ei tööta siis kirjutame käsitsi need read
      strcpy (seadmeID, "00001");
      strcpy (seadmePWD, "aaa11");
      strcpy(simPIN,"5417");
      strcpy (APN, "internet.tele2.ee");
      strcpy (server, "83.180.25.66:8080/");
}

/*
-------------------------------------------------------------------------------

      GSM MOODULIGA ÜHENDUSE INITSIALISEERIMINE

      Loob GSM mooduliga ühenduse
      Seadistab vajalikud parameetrid

-------------------------------------------------------------------------------
*/
void HTTP::initGSM()
{
      // Prindime serialisse et initsialiseerime SIM900-nt
      Serial.println("GSM");
      // Lülitame sisse GSM mooduli
      gsm.begin(2400);
      // Seadistame Pini
```

```
	setPIN();
	// GPRSi seadistamine
	setGPRS();
	// APNi seadistamine
	setAPN();
}
/*
-------------------------------------------------------------------------------

	GSM MOODULIGA ÜHENDUSE INITSIALISEERIMINE

	Loob GSM mooduliga ühenduse
	Seadistab vajalikud parameetrid

-------------------------------------------------------------------------------
*/
void HTTP::setParam(char *txt, uint8_t param)
{

	// Seadistame serveri aadressi
	if (param == PARAM_SERVER)
	{
		// Kirjutame programmi mällu, kui suurus klapid
		if(strlen(txt)<39)
		{
			strcpy(server,txt);
			// Kirjutame eepromi
			GSMmoodul.write_server(server);
			Serial.println(server);
		}
		else Serial.println(error);
	}
	// Seadistame APNi
	if (param == PARAM_APN)
	{
		if(strlen(txt)<25)
		{
			// Kirjutame programmi mällu
			strcpy(APN,txt);
			// Kirjutame eepromi
			GSMmoodul.write_APN(APN);
			// Seadistame ka mooduli
			gsm.begin(2400);
			setAPN();
			gsm.stop();
			Serial.println(APN);
		}
		else Serial.println(error);
	}
	// Seadistame SIMPINi
	if (param == PARAM_PIN)
	{
		if(strlen(txt)<4)
		{
			// Kirjutame programmi mällu
			strcpy(simPIN,txt);
			// Kirjutame eepromi
			GSMmoodul.write_simPIN(simPIN);
			// Seadistame ka mooduli
			gsm.begin(2400);
			setPIN();
```

```
                gsm.stop();
                Serial.println(simPIN);
            }
            else Serial.println(error);
      }
      // Seadistame SeadmeID
      if (param == PARAM_SEADMEID)
      {
            if(strlen(txt)<5)
            {
                // Kirjutame programmi mällu
                strcpy(seadmeID,txt);
                // Kirjutame eepromi
                GSMmoodul.write_seadmeID(seadmeID);
            }
            else Serial.println(error);
      }
      // Seadistame SeadmePWD
      if (param == PARAM_SEADMEPWD)
      {
            if(strlen(txt)<5)
            {
                // Kirjutame programmi mällu
                strcpy(seadmePWD,txt);
                // Kirjutame eepromi
                GSMmoodul.write_seadmePWD(seadmePWD);
            }
            else Serial.println(error);
      }
}
/*
-------------------------------------------------------------------------

      PINi SEADMINE

      Võtab EEPROMist PINi ning seadistab selle
      Tagastab:
                        Tõene - PINi seadistamine õnnestus
                        Väär  - Pini seadistamine ebaõnnestus

-------------------------------------------------------------------------
*/
bool HTTP::setPIN()
{
      // Setpini stringi tegemine
      char cmnd[13];
      strcpy(cmnd,"AT+CPIN=");
      strcpy(cmnd+8,simPIN);
      // Laeme moodulisse
      if(gsm.SendATCmdWaitResp(cmnd,0, 2000, "OK",5)) return true;
      else return false;
}
/*
-------------------------------------------------------------------------

      GPRS SEADISTAMINE

      Seadistab GPRS ühenduse
      Tagastab:
                        Tõene - GPRSi seadistamine õnnestus
                        Väär  - GPRSi seadistamine ebaõnnestus
```

```
------------------------------------------------------------------------------
*/
bool HTTP::setGPRS()
{
      if(gsm.SendATCmdWaitResp("AT+SAPBR=3,1,\"Contype\",\"GPRS\"",0,  2000,
"OK",5)) return true;
      else return false;
}
/*
------------------------------------------------------------------------------

      APNi SEADISTAMINE

      Seadistab APNi
      Tagastab:
                        Tõene - APNi seadistamine õnnestus
                        Väär  - APNi seadistamine ebaõnnestus

------------------------------------------------------------------------------
*/
bool HTTP::setAPN()
{
      // Teeme stringi
      char cmnd[47];
      strcpy(cmnd,"AT+SAPBR=3,1,\"APN\",\"");
      strcpy(cmnd+20,APN);
      strcpy(cmnd+strlen(cmnd),"\"");
      // Saadame moodulisse
      if(gsm.SendATCmdWaitResp(cmnd,0, 2000, "OK",5)) return true;
      else return false;
}
/*
------------------------------------------------------------------------------

      SEADISTAB HTTP ÜHENDUSE
      Seadistab HTTP ühenduse
      Tagastab:
                        Tõene - HTTPi seadistamine õnnestus
                        Väär  - HTTPi seadistamine ebaõnnestus

------------------------------------------------------------------------------
*/
bool HTTP::startHTTP()
{
      gsm.SendATCmdWaitResp("AT+SAPBR=1,1",0, 5000, "OK",5);
      if(gsm.SendATCmdWaitResp("AT+HTTPINIT",0, 10000, "OK",5)) return true;
      else return false;
}
/*
------------------------------------------------------------------------------

      Lõpetab HTTP ÜHENDUSE
      Lõpetab HTTP ühenduse
      Tagastab:
                        Tõene - HTTPi lõpetamine õnnestus
                        Väär  - HTTPi lõpetamine ebaõnnestus

------------------------------------------------------------------------------
*/
bool HTTP::stopHTTP()
```

```
{
      if(gsm.SendATCmdWaitResp("AT+HTTPTERM",0, 10000, "OK",5)) return true;
      else return false;
}
/*
-------------------------------------------------------------------------

      HTTP ÜHENDUS

      Loob HTTP ühenduse serveriga ning tõmbab sealt vastava info
      Tagastab allalaetud stringi

-------------------------------------------------------------------------
*/
void HTTP::HTTPget(char *data, uint8_t pikkus)
{
      // Paneme kokku aadressi
      char aadress[strlen(server)+50];
      strcpy(aadress,"AT+HTTPPARA=\"URL\",\"");
      strcpy(aadress+19,server);
      strcpy(aadress+19+strlen(server),"?id=");
      strcpy(aadress+23+strlen(server),seadmeID);
      strcpy(aadress+23+strlen(server)+strlen(seadmeID),"&pass=");
      strcpy(aadress+29+strlen(server)+strlen(seadmeID),seadmePWD);
      strcpy(aadress+strlen(aadress),"\"");

      // HTTP parameetrite paika sättimine CID ja URL
      gsm.SendATCmdWaitResp("AT+HTTPPARA=\"CID\",1",0, 5000, "OK",5);
      gsm.SendATCmdWaitResp(aadress,0, 5000, "OK",5);
      // Viide 3 seki
      delay(3000);
      // GET funktsioon
      gsm.SendATCmdWaitResp("AT+HTTPACTION=0",0,                        5000,
"+HTTPACTION:0,200",5);
      // Viide 20 seki
      delay(20000);

      // Loeme siiamaani tulnud käsud
      gsm.WhileSimpleRead(data, pikkus);
      // Saadame READ käsu
      gsm.SimpleWriteln("AT+HTTPREAD");
      // puhastame data stringi
      data[0]='\0';
      // Viide 30 seki
      delay(30000);
      gsm.WhileSimpleRead(data,pikkus);
}
/*
-------------------------------------------------------------------------

      TÄIS HTTP GET
      Lülitab sisse GSM mooduli
      Loob HTTP ühenduse
      Tagastab HTTP ühendusest saadud tagastuse.
      Lõpetab HTTP ühenduse
      Tagastab:
                        Tõene - HTTPi lõpetamine õnnestus
                        Väär  - HTTPi lõpetamine ebaõnnestus

-------------------------------------------------------------------------
*/
```

```
void HTTP::fullHTTPget(char *data, uint8_t pikkus)
{
	// Loome tagastuse muutuja
	//String tagastus = "";
	// Lülitame sisse mooduli
	Serial.println("GSM_ON");
	gsm.begin(2400);
	Serial.println("HTTP_ON");
	// HTTP seadistamine
	startHTTP();
	// HTTP GET
	Serial.println("GET");
	HTTPget(data, pikkus);
	// HTTP lõpetamine
	Serial.println("HTTP_OFF");
	stopHTTP();
	// Lülitame GSM mooduli välja
	Serial.println("GSM_OFF");
	gsm.stop();
}
```

Lisa 5: Käskude teegi pealdisfail

```
/*
      Versioon 1
      GSM mooduli käskude teek
      Madis Lepiksaar, Mai 2014
*/
#ifndef Command_H
#define Command_H

#include <Arduino.h>
#include <EEPROM.h>
#include <Time.h>
#include <SoftwareSerial.h>
#include "Arduino.h"
#include "Seaded.h"
#include "GSM.h"
#include "HTTP.h"

class Command
{
      private:
      // HTTP deklareerimine
      HTTP http;
      public:
      Command();                                                  //          Käsu
konstruktor
      void Init();                                                //
Initsialiseerib süsteemi
      void UARTRead(char *read,uint8_t pikkus);
      // Jadapordist lugemine
      void CmndSearch(char *input);                    // Käsu otsimine
      void CmndIdent(char *input);                     //                Käsu
identifitseerimine ja selle täitmine
      void CmndExec(char *id, char *txt);              // Käsu täitmine
      void CmndHTTP(char *data, uint8_t pikkus);       // Loob HTTP ühenduse
võtab sealt andmed
      uint8_t strfind(char *tekst,char *otsitav);      // Otsib stringi char
massiivist
      void CmndCheck();                                       //     Kontrollib
mooduli parameetreid
};
#endif
```

Lisa 6: Käskude teek

```
/*
      Versioon 1
      GSM mooduli käskude teek
      Madis Lepiksaar, Mai 2014
*/

#include <Arduino.h>
#include <EEPROM.h>
#include <Time.h>
#include <SoftwareSerial.h>
#include "Arduino.h"
#include "Seaded.h"
#include "GSM.h"
#include "HTTP.h"
#include "Command.h"

/*
------------------------------------------------------------------------------

      KÄSU KONSTRUKTOR

------------------------------------------------------------------------------
*/
Command::Command()
{
      // Siin pole midagi
}
/*
------------------------------------------------------------------------------

      SÜSTEEMI INITSIALISEERIMINE

------------------------------------------------------------------------------
*/
void Command::Init()
{
      // Seriali alustamine
      Serial.begin(9600);
      Serial.println("Start");
      // GSM mooduli initsialiseerimine
      http.initGSM();
      // Releede seadistamine
      http.GSMmoodul.relee_setup();
}
/*
------------------------------------------------------------------------------

      JADAPORDIST LUGEMINE

      Loeb jadapordist info ja tagastab selle

------------------------------------------------------------------------------
*/
void Command::UARTRead(char *read, uint8_t pikkus)
{
      // Loeme niipalju kui on sisendmassiivi suurus
      for(int i = 0; i<pikkus-1; i++)
      {
            // Kontrollime kas serial üldse on, kui ei ole siis katkestame
```

```
lugeminse
            if(Serial.available()>0) read[i]=(char)Serial.read();
            else break;
      }
}
/*
-------------------------------------------------------------------------------

      KÄSU OTSIMINE STRINGIST

      Otsib Stringist esimese commandi ja tagastab selle.

-------------------------------------------------------------------------------
*/
void Command::CmndSearch(char *input)
{
      // Otsime käsu alguse
      uint8_t algus = strfind(input,"!#COM");
      // Juhul kui algust pole siis lihtsalt lõpetame
      if(algus == 0)
      {
            input[0]='\0';
            return;
      }
      // Eemaldame esimese otsa käsu viite
      strcpy(input,input+algus+4);
      // Otsime käsu lõpu
      uint8_t lopp = strfind(input,"####");
      // Eemaldame lõpu, kui on, kui ei ole siis teeb rea tühjaks
      if(lopp) input[lopp-1]='\0';
      else input[0]='\0';
}
/*
-------------------------------------------------------------------------------

      KÄSU IDENTIFITSEERIMINE

      Otsib stringist käsud ja täidab need

-------------------------------------------------------------------------------
*/
void Command::CmndIdent(char *input)
{
      // Otsime käsu
      CmndSearch(input);
      // Eraldame käsu nri
      char cmnd_id[4];
      // algväärtustame muutuja
      cmnd_id[0]='\0';
      strncpy(cmnd_id,input,3);
      // Lisame lõpu tunnuse
      cmnd_id[3]='\0';
      //Eraldame käsu
      strcpy(input,input+3);
      // Täidame käsu
      CmndExec(cmnd_id,input);
}
/*
-------------------------------------------------------------------------------

      KÄSU TÄITMINE
```

```
      Täidab soovitud käsu

------------------------------------------------------------------------
*/
void Command::CmndExec(char *id, char *txt)
{
      // Loome aja muutuja
      uint32_t aeg = atol(txt);
      // Käsk 001 seadistab serveri aadressi
      if(!strcmp(id,"001"))
      {
            Serial.print("Server:");
            // Kirjutame serveri aadressi
            http.setParam(txt,PARAM_SERVER);
      }
      // Käsk 002 seadistab uue APN aadressi
      else if(!strcmp(id,"002"))
      {
            Serial.print("APN:");
            // Kirjutame APNi
            http.setParam(txt,PARAM_APN);
      }
      // Käsk 003 seadistab uue SIMi PINi
      else if(!strcmp(id,"003"))
      {
            Serial.print("PIN:");
            // Kirjutame APNi
            http.setParam(txt,PARAM_PIN);
      }
      // Käsk 004 seadistab uue SeadmeID
      else if(!strcmp(id,"004"))
      {
            // Kirjutame SeadmeID
            http.setParam(txt,PARAM_SEADMEID);
            Serial.print("SeadmeID:");
            Serial.println(txt);
      }
      // Käsk 005 seadistab uue SeadmeID
      else if(!strcmp(id,"005"))
      {
            // Kirjutame SeadmeID
            http.setParam(txt,PARAM_SEADMEPWD);
            Serial.print("SeadmePWD");
            Serial.println(txt);
      }

      // Käsk 006 seadistab uue kellaaja
      else if(!strcmp(id,"006"))
      {
            // Kirjutame kellaaja
            setTime(aeg);
            Serial.print("Aeg:");
            Serial.println(now());
      }
      // Käsk 011 Muudab 1 relee astr kellaga sisselülitusaja
      else if(!strcmp(id,"011"))
      {
            // Kirjutame kellaaja
            http.GSMmoodul.relee_aeg_w(1, 1, aeg);
            Serial.print("R1-A-ON:");
```

```
      Serial.println(aeg);
}

// Käsk 012 Muudab 2 relee astr kellaga sisselülitusaja
else if(!strcmp(id,"012"))
{
      // Kirjutame kellaaja
      http.GSMmoodul.relee_aeg_w(2, 1, aeg);
      Serial.print("R2-A-ON:");
      Serial.println(aeg);
}
// Käsk 021 Muudab 1 relee astr kellaga väljalülitusaja
else if(!strcmp(id,"021"))
{
      // Kirjutame kellaaja
      http.GSMmoodul.relee_aeg_w(1, 2, aeg);
      Serial.print("R1-A-OFF:");
      Serial.println(aeg);
}
// Käsk 022 Muudab 2 relee astr kellaga valjalylitusaja
else if(!strcmp(id,"022"))
{
      // Kirjutame kellaaja
      http.GSMmoodul.relee_aeg_w(2, 2, aeg);
      Serial.print("R2-A-OFF:");
      Serial.println(aeg);
}
// Käsk 031 Muudab 1 relee lisa sisselylitusaja
else if(!strcmp(id,"031"))
{
      // Kirjutame kellaaja
      http.GSMmoodul.relee_aeg_w(1, 3, aeg);
      Serial.print("R1-L-ON:");
      Serial.println(aeg);
}
// Käsk 032 Muudab 2 relee lisa sisselylitusaja
else if(!strcmp(id,"032"))
{
      // Kirjutame kellaaja
      http.GSMmoodul.relee_aeg_w(2, 3, aeg);
      Serial.print("R2-L-ON:");
      Serial.println(aeg);
}
// Käsk 041 Muudab 1 relee lisa valjalylitusaja
else if(!strcmp(id,"041"))
{
      // Kirjutame kellaaja
      http.GSMmoodul.relee_aeg_w(1, 4, aeg);
      Serial.print("R1-L-OFF:");
      Serial.println(aeg);
}
// Käsk 042 Muudab 2 relee lisa valjalylitusaja
else if(!strcmp(id,"042"))
{
      // Kirjutame kellaaja
      http.GSMmoodul.relee_aeg_w(2, 4, aeg);
      Serial.print("R2-L-OFF:");
      Serial.println(aeg);
}
// Käsk 051 Lülitab sisse 1 relee
else if(!strcmp(id,"051"))
```

```
	{
		// Kirjutame kellaaja
		http.GSMmoodul.relee_sw(1, true);
		Serial.print("R1-ON");
	}
	// Käsk 052 Lülitab sisse 2 relee
	else if(!strcmp(id,"052"))
	{
		// Kirjutame kellaaja
		http.GSMmoodul.relee_sw(2, true);
		Serial.print("R2-ON");
	}
	// Käsk 061 Lülitab välja 1 relee
	else if(!strcmp(id,"061"))
	{
		// Kirjutame kellaaja
		http.GSMmoodul.relee_sw(1, false);
		Serial.print("R1-OFF");
	}
	// Käsk 062 Lülitab välja 2 relee
	else if(!strcmp(id,"061"))
	{
		// Kirjutame kellaaja
		http.GSMmoodul.relee_sw(1, false);
		Serial.print("R1-OFF");
	}
}
/*
-------------------------------------------------------------------------

	HTTP KÄSU TÄITMINE

	Laeb HTTP serverist käsu
	Täidab soovitud käsu

-------------------------------------------------------------------------
*/
void Command::CmndHTTP(char *data, uint8_t pikkus)
{
	http.fullHTTPget(data,pikkus);
}
/*
-------------------------------------------------------------------------

	STRINGI OTSIMINE CHAR MASSIIVIST

	Otsib stringi char massiivist
	Tagastab esimese tähe asukoha
	Kui ei leidnud tagastab 0-i

-------------------------------------------------------------------------
*/
uint8_t Command::strfind(char *tekst,char *otsitav)
{
	// Loome pointer muutuja
	char * pt;
	// Otsime vaste
	pt=strstr(tekst,otsitav);
	// Kui ei leidnud tagastame 0-i
	if(pt == NULL) return 0;
	else return (int)pt-(int)tekst+1;
```

```
}
/*
-------------------------------------------------------------------------------

      PARAMEETRITE KONTROLL

      Kontrollib releede olekut

-------------------------------------------------------------------------------
*/
void Command::CmndCheck()
{
      http.GSMmoodul.relee_kontroll();
}
```

Lisa 7: Põhiprogramm

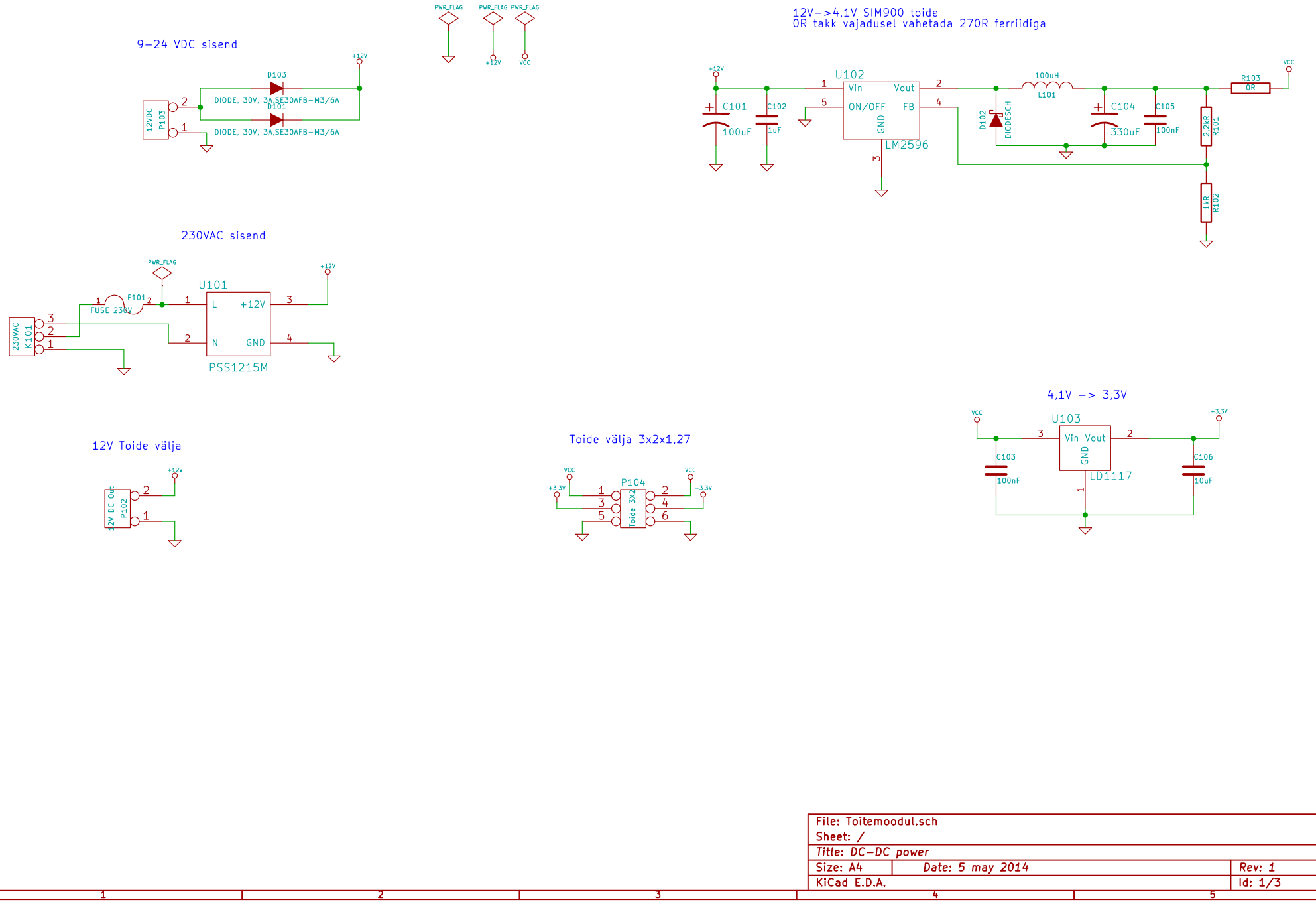
```
/*
Kaughallatav tänavavalgustuse kontroller
Versioon 1

Loonud Madis Lepiksaar
Mai 2014
*/
#include <Arduino.h>
#include <EEPROM.h>
#include <Time.h>
#include <SoftwareSerial.h>
#include "Arduino.h"
#include "Seaded.h"
#include "GSM.h"
#include "HTTP.h"
#include "Command.h"

// Seadistame commandi
Command cmnd;
int jarjekord = 0;
void setup() {
  cmnd.Init();
}

void loop() {
  char sisend[120];
  sisend[0]='\0';
  // Kontrollime jadapordi käsku
  cmnd.UARTRead(sisend,sizeof(sisend));
  // Iga 10 minuti tagant kontrollime serverit
  if(jarjekord>120)
  {
    // Algväärtustame sisendi
    sisend[0]='\0';
    cmnd.CmndHTTP(sisend,sizeof(sisend));
    jarjekord = 0;
  }
  delay(5000);
  jarjekord++;

  // Käsu identifitseerimine
  cmnd.CmndIdent(sisend);
}
```

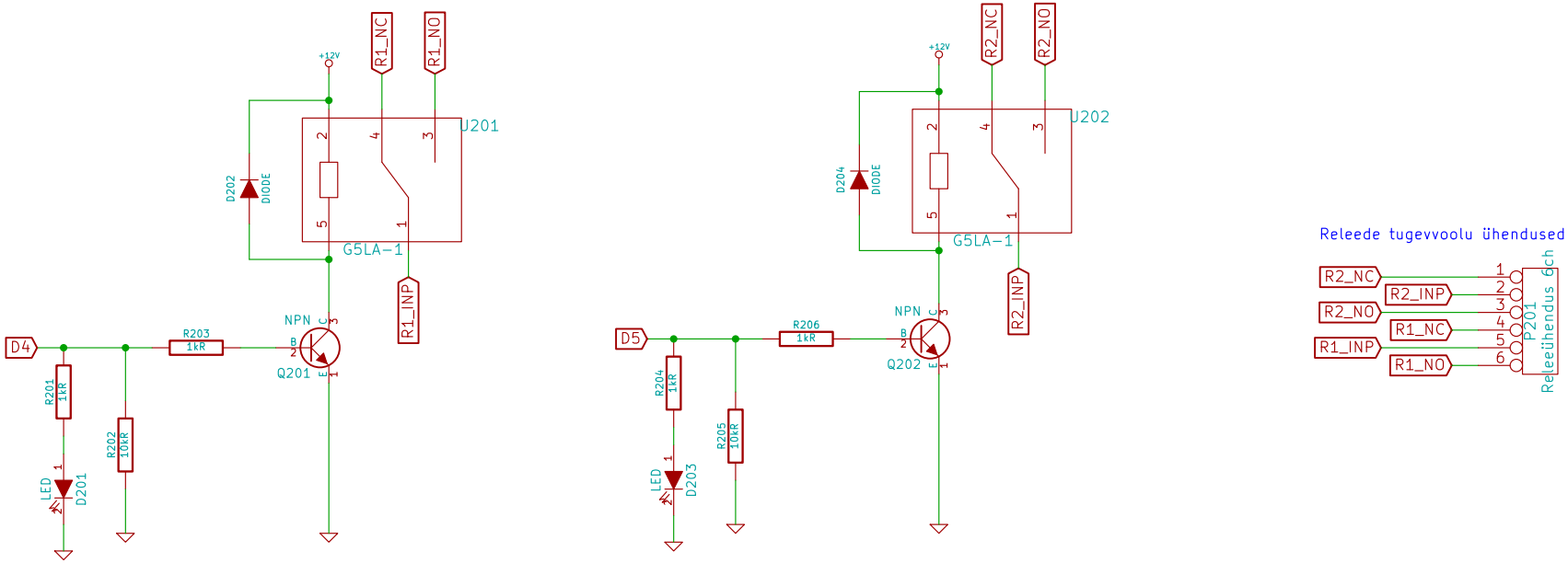
Lisa 8: Käskude register

Nr	Vajalik tekst	Tegevus	Piirangud	Näide
001	Serveri aadress	Muudab serveri aadressi	Pikkus kuni 39 tähemärki	!#COM00183.187.26.3:8080####
002	APN aadress	Muudab APN aadressi	Pikkus kuni 25 tähemärki	!#COM002internet.tele2.ee####
003	SIMipin	Muudab SIMiPINi	Pikkus 4 tähemärki	
004	SeadmeID	Muudab seadmeID	Pikkus 5 tähemärki	!#COM00454444####
005	SeadmePWD	Muudab seadmePWD	Pikkus 5 tähemärki	!#COM00564444####
006	Hetkeaeg	Seadistab hetkeaja	UTS aeg	!#COM0061397044300####
011	Sisselülitusaeg	Muudab 1 relee astr kellaga sisselülitusaja	UTS aeg	!#COM0111397044300####
012	Sisselülitusaeg	Muudab 2 relee astr kellaga sisselülitusaja	UTS aeg	!#COM0121397044300####
021	Väljalülitusaeg	Muudab 1 relee astr kellaga väljalülitusaja	UTS aeg	!#COM0211397044300####
022	Väljalülitusaeg	Muudab 2 relee astr kellaga väljalülitusaja	UTS aeg	!#COM0221397044300####
031	Sisselülitusaeg	Muudab 1 relee lisa sisselülitusaja	UTS aeg	!#COM0311397044300####
032	Sisselülitusaeg	Muudab 1 relee lisa sisselülitusaja	UTS aeg	!#COM0321397044300####
041	Väljalülitusaeg	Muudab 1 relee lisa väljalülitusaja	UTS aeg	!#COM0411397044300####
042	Väljalülitusaeg	Muudab 1 relee lisa väljalülitusaja	UTS aeg	!#COM0421397044300####
051	-	Lülitab sisse 1 relee		!#COM051####
052	-	Lülitab sisse 2 relee		!#COM052####
061	-	Lülitab välja 1 relee		!#COM061####
062	-	Lülitab välja 2 relee		!#COM062####

GRAAFILINE OSA

1. Toitemooduli elektroonikaskeem
2. Protsessormooduli elektroonikaskeem
3. Elektroonika materjalide loend
4. Näidis kilbiskeem

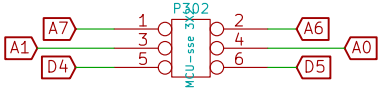

9–24 VDC sisend
12VDC
P103
D103
DIODE, 30V, 3A,SE30AFB–M3/6A
D101
DIODE, 30V, 3A,SE30AFB–M3/6A
+12V
PWR_FLAG
PWR_FLAG
PWR_FLAG
+12V
VCC
12V–>4,1V SIM900 toide
0R takk vajadusel vahetada 270R ferriidiga
U102
Vin
Vout
ON/OFF
FB
GND
LM2596
C101
100uF
C102
1uF
100uH
L101
D102
DIODESCH
C104
330uF
C105
100nF
R103
0R
2,2kR
R101
1kR
R102
230VAC sisend
230VAC
K101
F101
FUSE 230V
PWR_FLAG
U101
L
N
+12V
GND
PSS1215M
12V Toide välja
12V DC Out
P102
Toide välja 3x2x1,27
P104
Toide 3X2
4,1V –> 3,3V
U103
Vin
Vout
GND
LD1117
C103
100nF
C106
10uF
+3,3V
File: Toitemoodul.sch
Sheet: /
Title: DC–DC power
Size: A4
Date: 5 may 2014
Rev: 1
KiCad E.D.A.
Id: 1/3



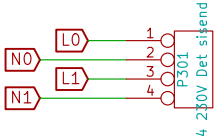
Releede tugevvoolu ühendused

File: Relee.sch		
Sheet: /Releelülitus/		
Title:		
Size: A4	*Date: 5 may 2014*	*Rev:*
KiCad E.D.A.		Id: 2/3

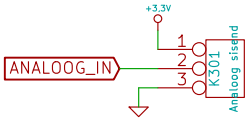
Mikrokontrollerisse 3x2x1,27



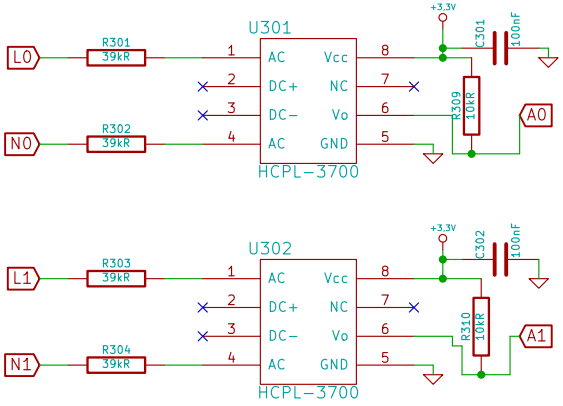
230 V AC detekteerimise sisendid



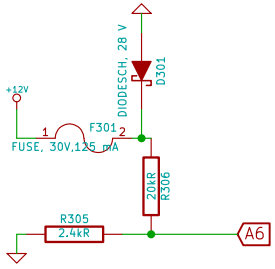
Analoog sisendid



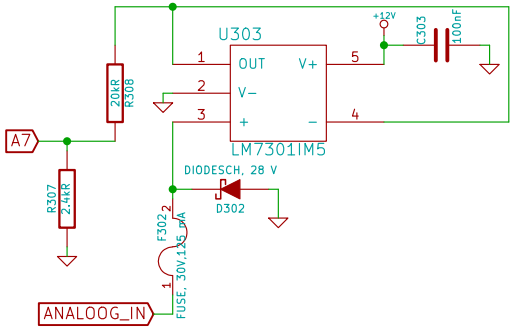
230 V AC detekteerimine spets IC-ga



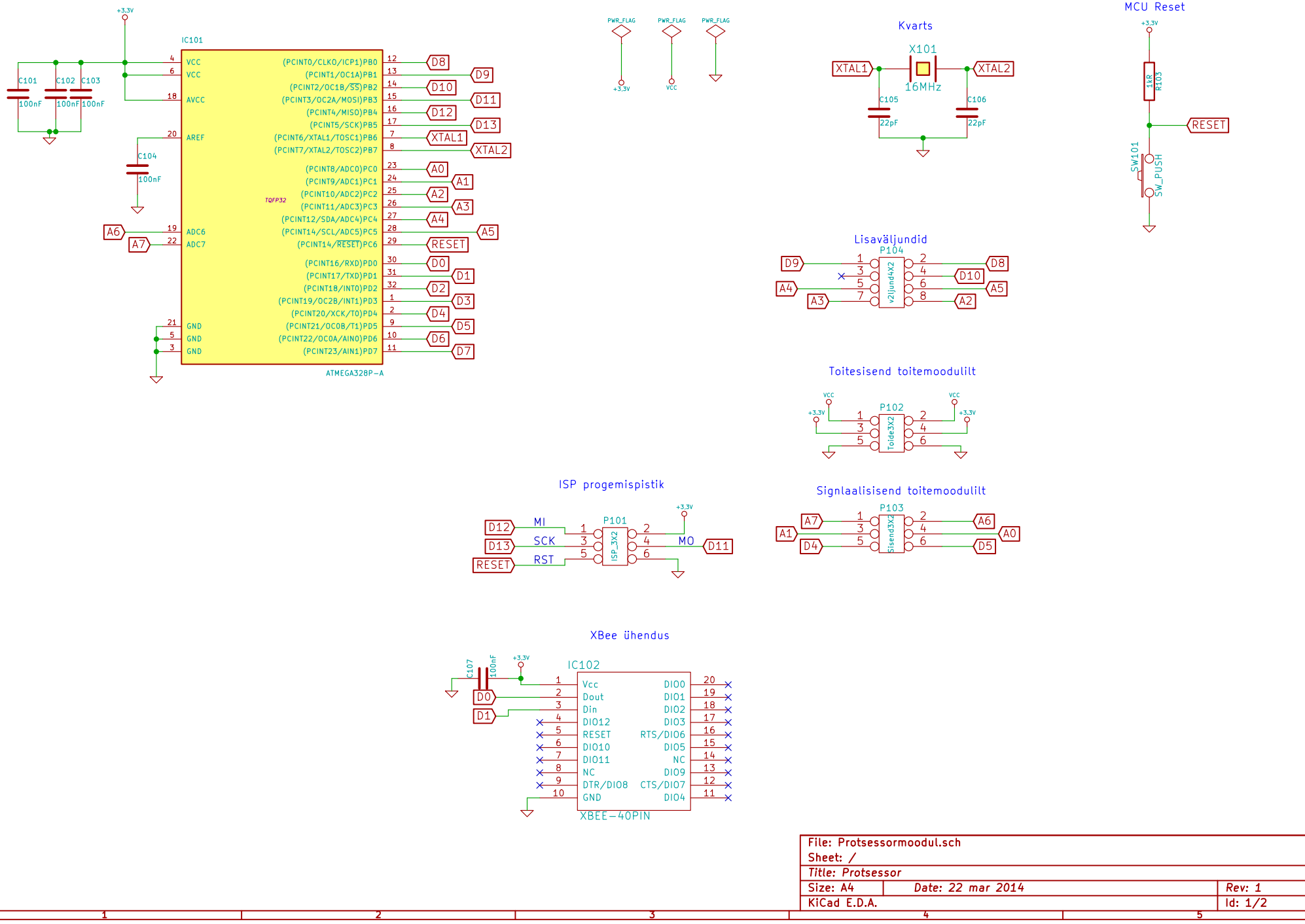
Sisendpinge mõõtmine



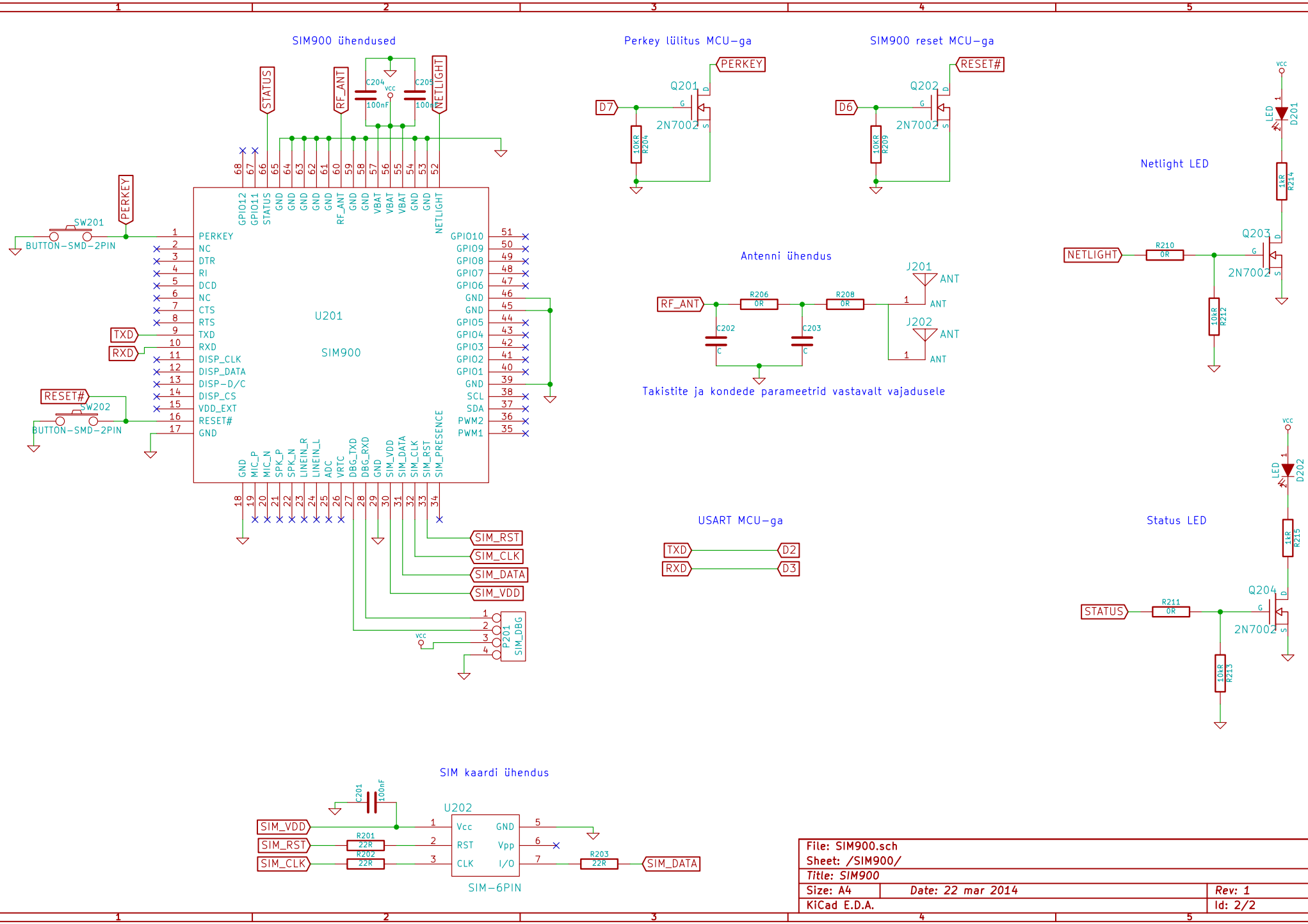
Analoog sisendi opamp ja pingejagur maks 30 V



File: sisendid.sch		
Sheet: /sisendid/		
Title:		
Size: A4	Date: 5 may 2014	Rev:
KiCad E.D.A.		Id: 3/3



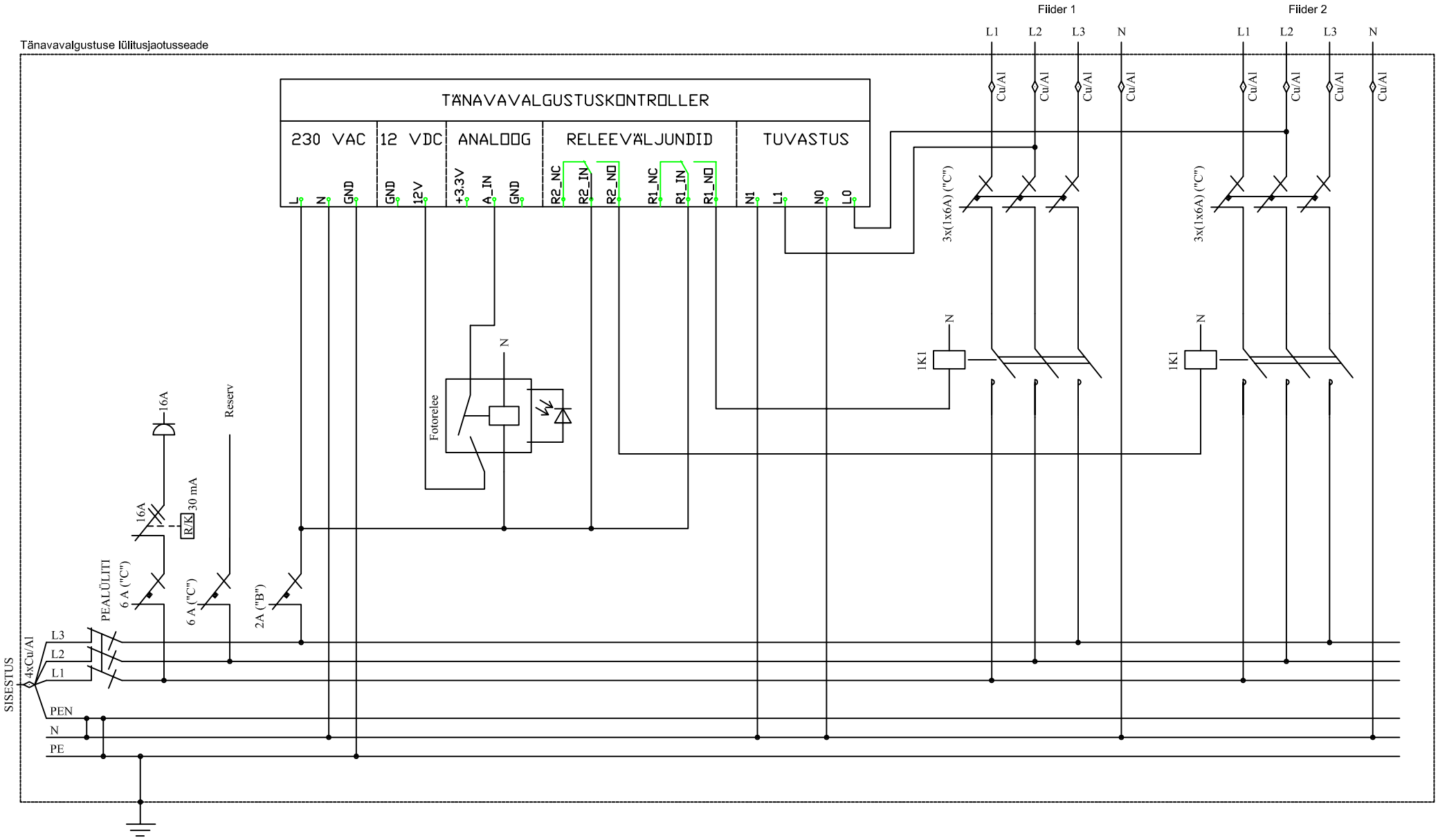
File: Protsessormoodul.sch		
Sheet: /		
Title: Protsessor		
Size: A4	*Date: 22 mar 2014*	*Rev: 1*
KiCad E.D.A.		Id: 1/2


SIM900 ühendused
Perkey lülitus MCU-ga
SIM900 reset MCU-ga
Netlight LED
Antenni ühendus
Takistite ja kondede parameetrid vastavalt vajadusele
USART MCU-ga
Status LED
SIM kaardi ühendus
U201
SIM900
U202
SIM-6PIN
SW201
SW202
BUTTON-SMD-2PIN
Q201
Q202
Q203
Q204
2N7002
J201
J202
ANT
D201
D202
P201
SIM_DBG
PERKEY
RESET#
STATUS
RF_ANT
NETLIGHT
TXD
RXD
SIM_RST
SIM_CLK
SIM_DATA
SIM_VDD
D2
D3
D6
D7
C201
C202
C203
C204
C205
100nF
R201
R202
R203
R204
R206
R208
R209
R210
R211
R212
R213
R214
R215
22R
10kR
1kR
0R
VCC
LED
File: SIM900.sch
Sheet: /SIM900/
Title: SIM900
Size: A4
Date: 22 mar 2014
Rev: 1
KiCad E.D.A.
Id: 2/2

ELEKTROONIKA MATERJALIDE NIMEKIRI

PROTSESSORMOODUL		
Kirjeldus	**Hulk**	**Tähis skeemil**
Kondensaator, 22pF	2	C105 C106
Kondensaator, 100nF	8	C101 C102 C103 C104 C107 C201 C204 C205
Kondensaator, C	2	C202 C203
Valgusdiood, LED	2	D201 D202
Protsessor, ATMEGA328P-A	1	IC101
Klemmid, XBEE-40PIN	1	IC102
Antenni pesa, ANT	2	J201 J202
Klemmid, ISP_3X2	1	P101
Klemmid, Toide3X2	1	P102
Klemmid, Sisend3X2	1	P103
Klemmid, väljund4X2	1	P104
Klemmid, SIM_DBG	1	P201
Transistor, 2N7002	4	Q201 Q202 Q203 Q204
Takisti, 0R	4	R206 R208 R210 R211
Takisti, 22R	3	R201 R202 R203
Takisti, 1kR	3	R103 R214 R215
Takisti, 10KR	2	R204 R209
Takisti, 10kR	2	R212 R213
Lüliti, SW_PUSH	1	SW101
Lüliti, BUTTON-SMD-2PIN	2	SW201 SW202
Andmesidemoodul, SIM900	1	U201
Kaardipesa, SIM-6PIN	1	U202
Kvarts, 16MHz	1	X101

TOITEMOODUL		
Kirjeldus	**Hulk**	**Tähis skeemil**
Kondensaator, 100nF	5	C103 C105 C301 C302 C303
Kondensaator, 1uF	1	C102
Kondensaator, 10uF	1	C106
Kondensaator, 100uF	1	C101
Kondensaator, 330uF	1	C104
Diood, 30V, 3A,SE30AFB-M3/6A	2	D101 D103
Schottky diood	1	D102
LED	2	D201 D203
Diood	2	D202 D204
Schottky diood	2	D301 D302
Kaitse, 230V	1	F101
Kaitse, 30V,125 mA	2	F301 F302
Klemmid, 230VAC	1	K101
Klemmid, Analoog sisend	1	K301
Induktiivpool, 100uH	1	L101
Klemmid, 12V DC Out	1	P102
Klemmid, 12VDC	1	P103
Klemmid, Toide 3X2	1	P104
Klemmid, Releeühendus 6ch	1	P201
Klemmid, 4 230V Det sisend	1	P301
Klemmid, MCU-sse 3X2	1	P302
Transistor, NPN	2	Q201 Q202
Takisti, 0R	1	R103
Takisti, 1kR	5	R102 R201 R203 R204 R206
Takisti, 2,2kR	1	R101
Takisti, 2.4kR	2	R305 R307
Takisti, 10kR	4	R202 R205 R309 R310
Takisti, 20kR	2	R306 R308
Takisti, 39kR	4	R301 R302 R303 R304
Toitemoodul, PSS1215M	1	U101
Pingemuundur, LM2596	1	U102
Pingemuundur, LD1117	1	U103
Relee, G5LA-1	2	U201 U202
Optiline lahtisidesti, HCPL-3700	2	U301 U302
Operatsiooni võimendi, LM7301IM5	1	U303



ProSystem

Tel.+372 671 4183, E-mail info@prosystem.ee
Tegevuslitsents EL10859120-0001 13.11.2002

Projekteeris:	Madis Lepiksaar
Kinnitas:	Asko Kuusalu
Kuupäev	17.05.2014
Fail	LJS Skeem.dwg

Objekt	Lõputöö		Projekt/Staadium: Näidis
Tellija	-		
Objekti asukoht	-		Töö nr.: MHK70LT
Joonise nimi	Tänavavalgustuse lülitusjaotusseadme skeem		Joonise nr.: EV-1
Lehekülg	1/1	Lehe mõõt: A4	Mõõtkava