TALLINNA TEHNIKAÜLIKOOL
MEHAANIKATEADUSKOND

Mehhatroonikainstituut

Kvaliteeditehnika ja metroloogia õppetool

MHT40LT

**Raul Kippar**

# ELEKTRILISE PÕRANDAKÜTTE ELEKTRIBÖRSI PÕHINE JUHTIMISSEADE

Bakalaureusetöö

Autor taotleb

Tehnikateaduste bakalaureuse

akadeemilist kraadi

**Tallinn 2016**

# Autorideklaratsioon

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

Töö valmis Edi Kulderknupi ning Madis Uuemaa juhendamisel
"…."…….2016. a.

Töö autor

…… allkiri

Töö vastab bakalaureusetööle esitatavatele nõuetele.

"…."…….2016. a.

Juhendajad

Edi Kulderknup …… allkiri "…."…….2016. a.

Madis Uuemaa …… allkiri "…."…….2016. a.

Lubatud kaitsmisele.

MAHB õppekava kaitsmiskomisjoni esimees

"…."…….2016. a.

…… allkiri

# BAKALAUREUSETÖÖ ÜLESANNE

2016 aasta kevadsemester

Üliõpilane: Raul Kippar 135106MAHB61

Õppekava: MAHB02/13

Eriala: Mehhatroonika

Juhendajad: dotsent Edi Kulderknup

doktorant Madis Uuemaa

Konsultandid: Taaniel Uleksin, Martin Valvur

**Bakalaureusetöö teema:**

(Eesti keeles) **Elektrilise põrandakütte elektribörsi põhine juhtimisseade**

(Inglise keeles) **Electricity spot market based control device for electric floor heater**

| **Nr** | **Ülesande kirjeldus** | **Täitmise tähtaeg** |
|---|---|---|
| 1. | Ülesande püstitus. Ülevaade olemasolevatest lahendustest. Ülevaade testobjektist, elektriturgudest ja Smart Load Solutions (SLS) börsihinnapõhisest optimeerimistarkvarast. | 01.02.2016 |
| 2. | Seadme komponentide valik ja seadme elektroonika koostamine ja testimine. | 21.03.2016 |
| 3. | Seadme juhtimisalgoritmi ja tarkvara välja töötamine. Seadme sidumine SLS optimeerimistarkvaraga. | 04.04.2016 |
| 4. | Seadme korpuse projekteerimine ja valmistamine. | 11.04.2016 |
| 5. | Seadme testimine ning tulemuste analüüs. | 25.04.2016 |
| 6. | Töö lõppvormistus, trükkimine ja köitmine. | 19.05.2016 |

**Lahendatavad inseneritehnilised ja majanduslikud probleemid:** Töö eesmärgiks on luua juhtseade, mis optimeerib elektrilise põrandakütte tööd kasutades SLS optimeerimistarkvara. Selleks tuleb luua juhtseade, mis suudab koguda infot sensoritelt, juhtida põrandakütte tööd ning kasutada pilvepõhist SLS tarkvara. Lõpptulemusena hoiab kütteseade toas soovitud

temperatuuri, kuid töötab majanduslikult optimaalsetel tundidel ja vähendab seeläbi elektriarvet. Töö käigus koostatakse nii elektroonikaskeem, juhtimistarkvara kui ka korpuse joonised.

Töö keel: Eesti

Kaitsmistaotlus esitada dekanaati hiljemalt 16.05.2016. Töö esitamise tähtaeg 20.05.2016

Üliõpilane Raul Kippar /allkiri/ kuupäev 21.03.2016

Juhendaja Edi Kulderknup /allkiri/ kuupäev 21.03.2016

Juhendaja Madis Uuemaa /allkiri/ kuupäev 21.03.2016

# Sisukord

## Eessõna

Antud bakalaureusetöö teema on välja pakutud ettevõtte Smart Load Solutions poolt. Katsete tegemine toimus Tallinnas Põllu tee 135 kortermaja korteris, mille omanik on Mart Mere. Andmete kogumisega ja konsultatsiooniga aitasid bakalaureusetööl valmida Smart Load Solutionsi tegevjuht Madis Uuemaa ja arendaja Taaniel Uleksin.

# Sissejuhatus

Aastatel 1999 kuni 2014 oli elektrienergia tarbimine Eestis majapidamistes pidevas kasvamises ning näitas esimesi langemise märke alles 2011. aastal ning langes siis veidi 2012. kuni 2014. aastatel. Sellest hoolimata oli elektrienergia tarbimine majapidamistes aastal 2014 kõrgem kui mõned aastakümned tagasi ning üldine elektrienergia tarbimine jätkas kasvamist [1]. Seepärast on oluline leida uusi võimalusi kulude kokku hoidmiseks elektri tarbimise pealt. Eesti üleminek avatud elektriturule 1. jaanuaril 2013. tekitas neid võimalusi.

Eesti kuulub Nord Pool Spot elektripiirkonda, kuhu kuuluvad veel Soome, Rootsi, Norra, Taani, Suurbritannia, Saksamaa, Läti ja Leedu. Nord Pool Spotis määratakse elektrihind igaks tunniks eraldi. Nord Pool Spot kodulehel avaldatakse iga järgneva 24 tunni elektrihinnad eelneva päeva kella 12.42-ks või hiljem (Kesk-Euroopa aja järgi) [2].

Nord Pool Spoti elektriturul võib elektrihind ühe päeva lõikes märkimisväärselt varieeruda. Tundidel, mil tarbitakse elektrienergiat rohkem, on elektri hind märgatavalt kõrgem kui öisel ajal, mil elektrienergia tarbimine on väike. Kuigi hinna kujunemisel tekib teatud öö-päeva muster, tekib siiski iga päev unikaalne hinnagraafik. Tekkinud hinnagraafikut on võimalik kasutada elektrienergia kulude vähendamiseks, kui tarbida elektrienergiat tundidel, mil üldine tarbimine on väike ja hind madal.

Samast ideest on lähtunud ettevõte Smart Load Solutions (edaspidi SLS). SLS on Eestis tegutsev ettevõte, mis pakub oma klientidele võimalust optimeerida erinevate seadmete tööd vastavalt tunnist tundi kõikuvatele hindadele. Sellised seadmed võivad olla küttesüsteemid, pumbad, kuid ka näiteks hüdroelektrijaamad (müües elektrit kõrge hinnaga ajal, muul ajal veevarusid täiendades).

SLSi optimeerimisteenus baseerub pilvetarkvaral, mille ettevõte on välja töötanud. See pilvetarkvara võtab Nord Pool Spoti kodulehel avalikustatava elektrihindade graafiku ning pakub kliendi kodus või ettevõttes asuvale elektrienergial töötavale seadmele selle seadme spetsiifilise optimeerimisgraafiku. SLSi kodulehel on juhendid selle kohta, kuidas valmistada elektroonilist seadet, mis suhtleb SLSi pilveteenusega ning juhib elektrienergiat tarbiva seadme tööd. Nüüdseks pakub SLS ka ettevõtte enda disainitud termostaati, mis on võimeline juhtima koduküttesüsteemi tööd elektribörsipõhiselt optimeerituna.

Käesoleva bakalaureusetöö käigus valmistatakse sellise seadme üks esimene prototüüp. Töö eesmärgiks on elektribörsipõhiselt optimeerida kahe põrandakütte tööd läbi ühe seadme ning

saavutada seeläbi kulude sääst. Eesmärgi täitmiseks on tarvis valida välja sobivad elektroonilised seadmed põrandakütte töö juhtimiseks, planeerida ning teostada välja valitud elektroonikaseadmete juhtimine ning analüüsida põrandakütte töö optimeerimist. Lisaks disainitakse töö käigus seadmele korpus.

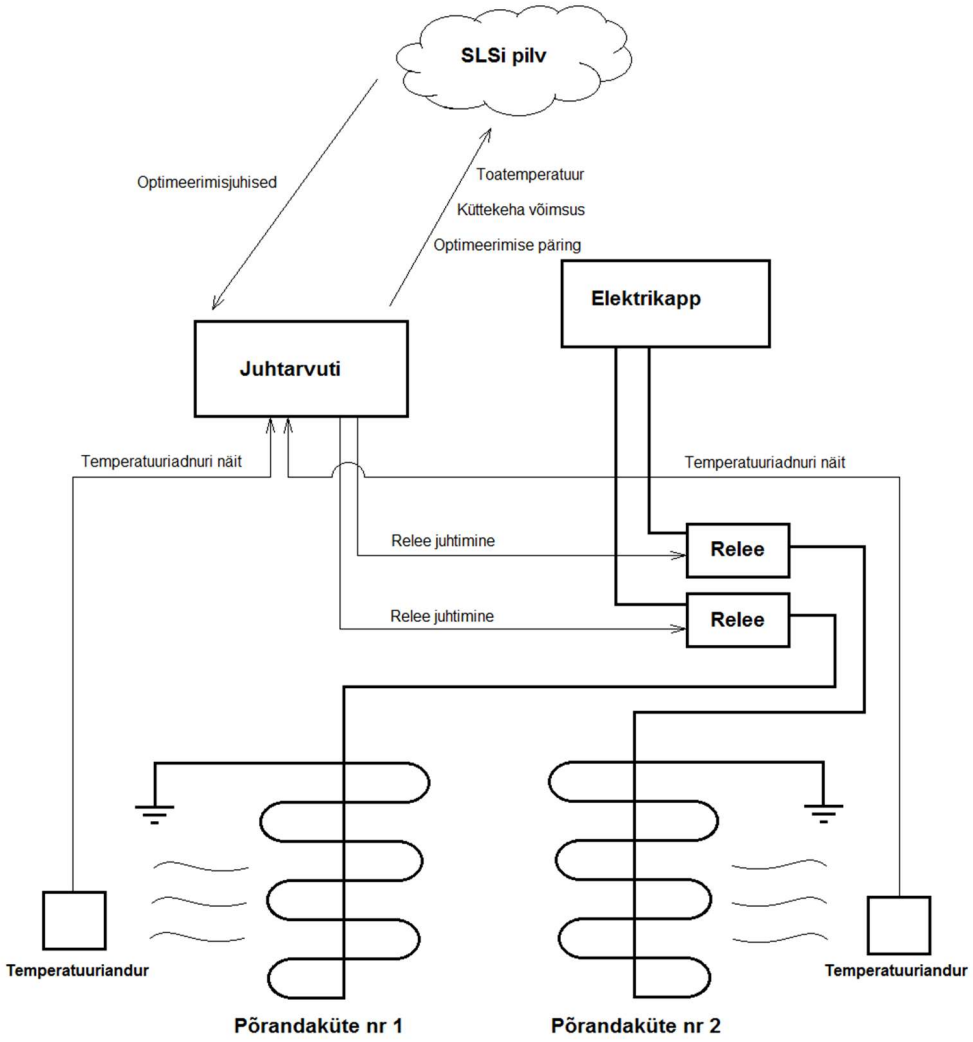
Seade paigaldatakse 173-ruutmeetrise pindalaga korterisse juhtima kahe põrandakütte tööd. Põrandakütete köetavad toad on 50-ruutmeetrine elutuba (edaspidi ülemine tuba) ning kokku 52-ruutmeetrine esik ning köök (edaspidi alumine tuba). Mõlemad põrandakütted on nimivõimsustega 1,85 kW (kilovatti), maksimaalse sisendvoolutugevusega 16 A. Põrandakütte termostaadid jäetakse alles. Sellega tagatakse, et optimeerimisseadme võimalike rikete korral takistavad termostaadid põrandakütteid ülekoormuse eest. Termostaadid seatakse maksimaalse võimsuse peale, et mitte optimeerimise tulemusi segada.

Töö on koostatud neljas osas. Esimeses osas kirjeldatakse optimeerimisseadme koostamiseks vajaminevaid elektroonilisi komponente. Teises osas püstitatakse juhtimisülesanded ning kirjeldatakse nende lahendamist. Kolmandas osa kirjeldatakse korpuse disaini. Viimases osas analüüsitakse optimeerimisseadme töö tõhusust.

# 1. ELEKTROONIKA

## 1.1. Komponentide valik

Põrandakütete SLSi optimeerimisteenuse järgi juhtimiseks on vaja koostada elektrooniline seade, mis oleks võimeline kumbagi põrandakütet sisse ja välja lülitama, mõõtma mõlemas toas temperatuuri ning olema võimeline SLSi pilveteenusega infot vahetama (sele 1.1.). Elektrooniliste komponentide valikul on oluline jälgida komponendi sobivust nõutud ülesande täitmiseks, nende hinda ning sensorite puhul – vastavust etteantud parameetritele ja sobivat täpsust. Järgnevalt on tehtud ülevaade vajalikest komponentidest.




**Sele 1.1. Optimeerimisseadme lihtsustatud skeem**

**Juhtarvuti**

Seadme töö juhtimiseks on vaja valida juhtarvuti, mis oleks võimeline põrandakütte tööd juhtima, võtma vastu ning salvestama infot anduritelt ja samaaegselt juhtima sidepidamist SLSi pilvega. Arvutile peab saama peale laadida programmid, mis nende ülesannetega tegelevad. Seade saab olema paigaldatud inimese koju, seega võiks see olla võimalikult kompaktne ning mõõtmetelt väike. Samal põhjusel valitakse juhtarvutiks pigem mõni mikroarvuti.

**Voolurelee**

Katsealuste küttekehade nominaalvoolutugevus on 16 amprit. Sellise koguse vooluga töötamiseks on tarvis valida kaks voolureleed kuna juhitakse kahe põrandakütte tööd. Voolureleed tuleb ühendada elektrikapist põrandakütete termostaatidesse minevate juhtmete külge.

**Temperatuuriandurid**

Mõlemasse katsealusesse tuppa on tarvis paigaldada temperatuuriandurid. Andurid peavad ühenduses olema juhtarvutiga ning andma edasi mõõtmistulemusi.

**Vooluadapter**

Juhtarvutile toite andmiseks on tarvis vooluadapterit vastavalt juhtarvuti vajadustele.

**Ühenduvus internetiga**

Seade peab olema võimeline infovahetuseks SLSi pilvega, seepärast peab seade olema ühendatud internetti. Seda on võimalik saavutada Ethernet kaabliga või wifi adapteriga.

**Seadme komponentide ühendamise tarvikud**

Komponentide omavahelise töö tagamiseks on komponendid omavahel tarvis ühendada. Selleks valitakse erinevaid kaableid. Milliseid kaableid tarvis on, selgub teiste komponentide valikust.

### 1.1.1 Voolurelee

Alustuseks on vaja välja valida põrandakütete sisse- ja väljalülitamiseks voolureleed. Katsealuste kütteelementide nominaalvoolutugevus on 16 amprit. Seega on kütteelementide lülitamiseks vaja kasutada kahte vähemalt 16 amprist voolureleed. Antud nõudmisele vastavalt leiti relee Songle 30A 5V C-Type Opto-Isolated Relay Module.

**Tabel 1.1. Songle 30A 5V C-Type Opto-Isolated relee tehnilised parameetrid** [3]

| Parameeter | Väärtus | Ühik |
|---|---|---|
| Sisendpinge | 5 | V |
| Tööpinge | 30 DC / 250 AC | V |
| Töövoolutugevus | 30 | A |
| Mõõtmed | 39 x 34 x 24 | mm |
| Hind | 4,68 | € |

### 1.1.2 Juhtarvuti

Seadme erinevate komponentide töö ning SLSi pilvega suhtlemise juhtimiseks on tarvis kasutada kontrollerit, mida oleks võimalik programmeerida neid ülesandeid täitma. Kontroller võiks olla väike, sellel peaks olema hulk sisend- ja väljundkontakte ning see peaks olema suhteliselt odav. Antud nõuetele vastavaid mikrokontrollereid on olemas mitmeid. Järgnevalt on tehtud lühike ülevaade ja analüüs kolmest kõige populaarsemast avatud lähtekoodiga mikrokontrollerist.

**Raspberry Pi**

Raspberry Pi 2 model B (sele 1.2.) on mikroarvuti, mis on võimeline enamikke lauaarvuti ülesandeid täitma, olles ise seejuures mõõtmetelt väga väike. Raspberry Pi on võimeline jooksutama Linux operatsioonisüsteeme ning selle soovitatav programmeerimiskeel on Python. Raspberry Pi 2 Model B suurteks eelisteks on tema suhteline odavus, teda on küllaltki lihtne programmeerida, teda on võimalik ühendada internetti läbi Ethernet kaabli, või lisakomponendiga läbi wifi. Raspberry Pi 2 Model B tehnilised andmed on välja toodud tabelis 1.2.

**Sele 1.2. Raspberry Pi 2 model B [5]**

**Tabel 1.2. Raspberry Pi 2 model B tehniline spetsifikatsioon [4] [5]**

| Komponent | Täpsustus |
|---|---|
| Protsessor | 900 MHz quad-core ARM Cortex-A7 |
| Mälu | 1 GB LPDDR2 |
| Voolupesa | Micro USB 5V, 2 A |
| Ethernet | 10/100 BaseT Ethernet pesa |
| USB | 4 x USB 2.0 pesa |
| GPIO (General purpose input/output) | 40-pin 2,54 mm |
| Mälukaardi pesa | Micro SDIO |
| Mõõtmed | 85 x 56 x 17 mm |
| Hind | $35 |

## Arduino Uno


**Sele 1.3 Arduino Uno [6]**

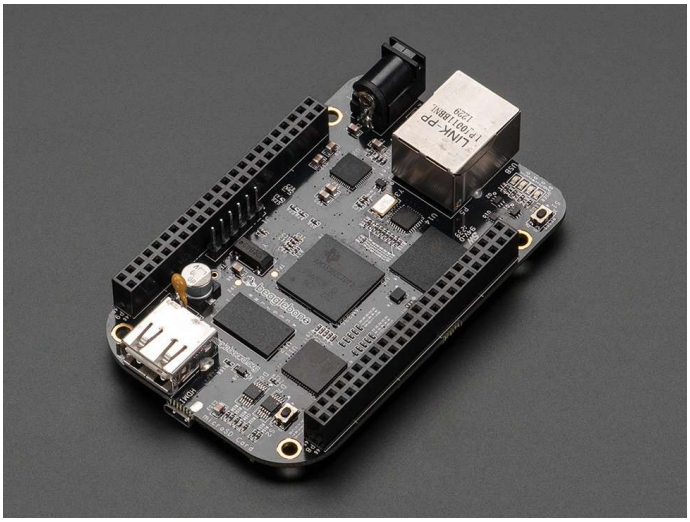
Arduino on avatud lähtekoodiga elektroonika-projektide koostamise platvorm. Arduinot kasutatakse palju nii algajate asjaarmastajate poolt kui ka erinevate ettevõtete elektroonilistes seadmetes. Arduino Uno (sele 1.3.) on mõistliku hinnaga mikrokontroller, mis utiliseerib programmeerimiskeele C++ lihtsamat versiooni. Arduino Uno sobib kõige paremini kasutamiseks lihtsate tsükliliste ülesannete täitmiseks. Kuna voolureleede töö juhtimine on just selline ülesanne, oleks Arduino Uno valimine igati põhjendatud. Arduino Uno tehnilised andmed on välja toodud tabelis 1.3.

**Tabel 1.3. Arduino Uno tehniline spetsifikatsioon [6] [7]**

| Komponent | Täpsustus |
|---|---|
| Mikrokontroller | ATmega328P |
| Soovitatav tööpinge vahemik | 7-12 Volti |
| Digitaalseid sisend/väljund kontakte | 14 |
| Analoog sisend kontakte | 6 |
| Alalisvoolutugevus kontakti kohta | 20 mA |
| Mälu | 32 KB Flash, 2KB SRAM, 1KB EEPROM |
| USB | 1 B tüüpi USB 2.0 pesa |
| Mõõtmed | 68,6 x 53,4 mm (pikkus x laius) |
| Hind | 25 € |

## Beaglebone Black

Beaglebone Black (sele 1.4.) on pigem sarnane Raspberry Pi-ga kui Arduino Unoga. See on mikroarvuti, mis on võimeline asendama lauaarvutit teatud ülesannete täitmisel. See on võimeline jooksutama Linuxi operatsiooni-süsteeme ning seda on äärmiselt lihtne töökorda seada ja kasutada. Võrreldes Pi ja Unoga on Beaglebone Black veidi kallim. Beaglebone Black tehnilised andmed on välja toodud tabelis 1.4.



**Sele 1.4. Beaglebone Black [8]**

**Tabel 1.4. Beaglebone Black tehniline spetsifikatsioon [8] [9]**

| Komponent | Täpsustus |
|---|---|
| Protsessor | Sitara AM3358BZCZ100 1GHz |
| Mälu | 512MB DDR3L 800MHz |
| Voolupesa | USB pesa või DC jack (alalisvoolu pesa) |
| USB | 1 USB 2.0 pesa ja miniUSB pesa |
| Ethernet | 10/100 Ethernet RJ45 |
| Sisend/väljund kontakte | 2x46 |
| Mälukaardi pesa | microSD |
| Hind | £35 |

Kolmest võrreldavast mikro-kontrollerist sobib kõige paremini juhtarvuti rolli Raspberry Pi. Kuigi Beaglebone Black peaks olema jooksutamise ja kasutamise seisukohalt lihtsam, siis tema suureks puuduseks on USB-portide vähesus (seadme kompaktsuse saavutamiseks on parem kasutada vooluadapterit USB väljundiga kui alalisvooluadapterit DC jack väljundiga, samuti kirjeldatakse hiljem, et internetiühenduse saavutamiseks kasutatakse USB pesa kasutavat wifi adapterit). Lisaks on Raspberry Pi-l rohkem suvapöördus-mälu ning väiksem hind. Raspberry Pi valikul mängis rolli ka SLSi töötajate eelnevad kogemused selle mikroarvuti kasutamisega.

Raspberry Pi-st üksi juhtarvutiks ei piisa. Probleem tuleneb sellest, et voolureleede töö juhtimiseks on tarvis 5-voldist kontakti, kuid nagu muude komponentide valikul selgub, on Raspberry Pi mõlemad 5-voldised kontaktid juba kasutuses. Seega tuleb hankida vahekomponent, mis oleks võimeline juhtima releede tööd ja varustama releedele 5-voldise sisendsignaali. Neid rolle sobib suurepäraselt täitma Arduino Uno.

Juhtarvutiks valiti kaks mikrokontrollerit: andmete kogumist teostama ning SLSi pilvega suhtlema Rasbperry Pi 2 model B ning releede lülitamist juhtima Arduino Uno.

### 1.1.3 Temperatuuriandurid

Temperatuuri mõõtmine peab toimuma mõlemas katsealuses toas eraldi, seega peab kasutama kahte temperatuuriandurit. Elektrikapp, mis põrandakütte termostaatidele voolu annab, asub ühes põrandaküttega tubadest. Elektrikapi kõrvale asetatakse ka koostatud seade, sest voolureleed ühendatakse otse elektrikapist tulevate põrandakütete voolujuhtmetega. Järelikult vähemalt üks anduritest saab asuda ülejäänud seadme komponentide juures ilma, et peaks lisakaablitega juhtarvuti ja anduri vahel ühenduse looma.

Teine andur peab aga asetsema teises põrandaküttega toas ning tuleb kasutada kas lisakaableid anduri ja juhtarvuti ühendamiseks või kasutada juhtmevaba andurit. Kummagi lahendusega on teatud probleemid.

Lisakaablite paigaldus tuleb teha selline, et korteris oleks võimalik vabalt liigelda ning paigaldus peaks ka esinduslik välja nägema. Seega peab kaabliliin olema korrektselt kinnitatud seintele ning kaablid peavad olema paigaldatud nii, et nad võimalikult vähe silma torkaks, näiteks lae ja seinte ühendusservadest.

Teine variant on kasutada juhtmevaba temperatuuri andurit. Raspberry Pi model B-l ega Arduino Unol pole sisseehitatud juhtmevaba signaali vastuvõtjaid, seega on teise variandi puhul tarvis muretseda juhtmevaba signaali vastu võttev moodul.

Valiku lahendus tuli korteri omanikult, kellele ei sobinud kaablite paigaldamise variant. Seega valiti kaugemas toas temperatuuri mõõtma juhtmevaba temperatuuriandur.

**Juhtmega temperatuuriandur**

Juhtmega temperatuurianduriks valiti DS18B20 veekindel andur (sele 1.5.). Sellel anduril on sobiv töövahemik piisavalt suure varuga, hea täpsus, digitaalne väljund, see on odav ning seda on kontrolleriga infovahetuseks võimalik ühendada ühe kontaktiga. Antud temperatuuriandurit võetakse kasutusele kaks, mõlemad mõõtma temperatuuri samas kohas. Kahe anduri näitudest võetakse keskmine väärtus täpsuse suurendamiseks.


**Sele 1.5. Temperatuuriandur DS18B20 [11]**

**Juhtmevaba temperatuuriandur**

Juhtmevaba temperatuurianduriks valiti Wireless-Things'i Wireless Ambient Temperature Sensor SB-CA (sele 1.6.). Juhtmevabadest temperatuurianduritest on see üks odavamaid, võimeline turvaliselt suhtlema WirelessThings raadiosignaali vastuvõtjatega ning seda on lihtne üles seada. Anduri vooluallikas on 200 milliampertunnine CR2032 patarei [10].

Andurite tehnilised andmed on välja toodud tabelites 1.5 ja 1.6.


**Sele 1.6. WirelessThings juhtmevaba temperatuuriandur [12]**

**Tabel 1.5 Anduri DS18B20 andmed [11]**

| Parameeter | Väärtus | Ühik |
|---|---|---|
| Tööpinge | 3,2 – 5,25 | V |
| Töövool | 2 | mA |
| Töötamisvahemik | -55 kuni 110 | ºC |
| Mõõtmistäpsus | +/- 0,5 | ºC |
| Hind | 4,59 | € |

**Tabel 1.6 WirelessThings juhtmevaba anduri SB-CA andmed [12]**

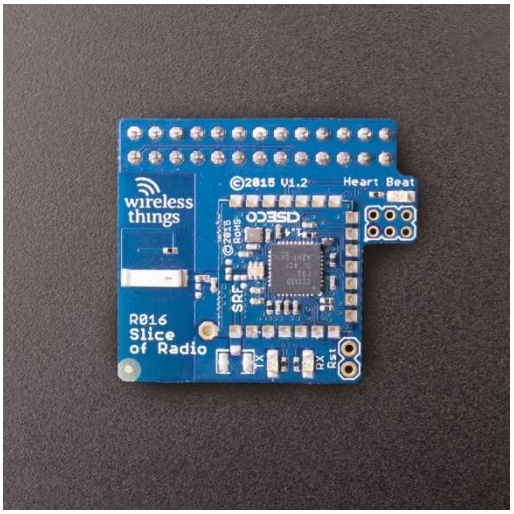
| Parameeter | Väärtus | Ühik |
|---|---|---|
| Tööpinge | 2 – 3,6 | V |
| Töövool puhkeolekus | 0,5 | µA |
| Töötamisvahemik | -10 kuni 40 | ºC |
| Mõõtmistäpsus | +/- 1 | ºC |
| Hind | 10 | £ |

### 1.1.4 Muud komponendid

Lisaks eelnevates alapeatükkides loetletud komponentidele on toimiva optimeerimissüsteemi jaoks tarvis elektroonikat, mis teeb olemasolevate komponentide vahelise ühendamise võimalikuks. Käesolevas alapeatükis on loetletud need komponendid.

**Juhtmevaba raadiosignaali vastuvõtja**

Juhtmevaba raadiosignaali vastuvõtjat on tarvis juhtmevaba termomeetri ning juhtarvuti ühendamiseks. Valiti WirelessThings Slice of Radio – Wireless RF Tranceiver (sele 1.7.). Valitud vastuvõtja ühildub varem valitud WirelessThings juhtmevaba temperatuurianduriga ning on disainitud ühilduma Raspberry Pi GPIO kontaktidega (26 kontakti 40-st). Juhtmevaba infovahetus on 128 bitise AES (Advanced Encryption Standard) krüpteeringuga. Selle kiibi hind on £10 [13].




**Sele 1.7. WirelessThings radiosignaali vastuvõtja [13]**

**Internetiga ühenduvus**

Seadme internetiga ühendamiseks on kaks variant. Üks variant on kasutada Ethernet kaablit, Raspberry Pi-l on vastav pesa olemas. Modem asub optimeerimisseadme paigaldusasukohast eemal teises ruumis, seega oleks tarvis mööda seinasid ja lage teostada kaablipaigaldus. Teine variant oleks kasutada wifi adapterit, mille saab ühendada ühte Raspberry Pi USB pessa. Sellisel juhul peab korteris olemas olema wifi ruuter.

Nagu ühe temperatuuri anduri puhul, valiti ka internetiga ühendamiseks korteriomaniku soovide kohaselt juhtmevaba lahendus. Valiti Digitus juhtmevaba 150N USB adapter. Adapteri hind on 13,85 €, hind oligi adapteri valikul peamine arvesse võetav parameeter [14].

**Vooluadapter**

Vooluadapteri valikul peab arvestama Raspberry Pi voolu nõuetega. Arduino Uno saab tööks vajaliku voolu läbi ühenduse Raspberry Pi-ga. Raspberry Pi 2 Model B saab voolu 5-voldisest micro-USB pesast ning tema soovitatav vooluallika voolutugevus on 1,8 amprit. Kasutusele võeti Lenovo 5,2-voldine ning 2-amprine vooluadapter.

**Mälukaart**

Juhtimisprogrammide seadmele peale laadimiseks on vaja mälukaarti, millele need programmid salvestatakse. Raspberry Pi 2 model B-l on microSD kaardi pesa, seega otsiti sobiv microSD mälukaart. Valiti SanDisk M2 2 GB MS Micro mälukaart, hinnaga 6,80 € [15].

**Juhtmed ja kaablid**

Komponentide omavaheliseks ühendamiseks on tarvis juhtmeid. Arduino Uno ja Raspberry Pi omavahelise side loomiseks USB 2.0 A/B tüüpi kaabel ning juhtmeid voolureleede Arduino Uno külge ühendamiseks. Uno kõik sisend- ja väljund kontaktid on „emased“, releede kontaktid on aga „isased“, seega on vaja ühest otsast „isaseid“, teisest „emaseid“ juhtmeid.

## 1.2. Seadme majanduslik analüüs

Kogu seadme hind on kõigi komponentide ning kaablite hindade summa. Osa komponente telliti välismaalt, seega nende hinnad on kirjas USA dollarites ($) või Suurbritannia naelades (£). Naelad ja dollarid tuleb ümber konverteerida eurodesse. Arvesse ei ole võetud transpordikulusid.

Naela kurss euro suhtes seisuga 14.05.2016. : £1 = 1,27€

Dollari kurss euro suhtes seisuga 14.04.2016. : $1 = 0,88€

Raspberry Pi 2 model B hind: $35

hind eurodes: $35 * 0,88 = 30,8€

Arduino Uno hind: 25€

Songle 30A 5V C-Type Opto-Isolated Relay Module

hind: 4,68*2 = 9,36€

DS18B20 veekindle temperatuuriandur

hind: 4,6*2 = 9,2€

Wireless-Things Wireless Ambient Temperature Sensor SB-CA

hind: £10

hind eurodes: £10 * 1,27 = 12,7€

CR2032 patarei hind: 1€ [10]

WirelessThings Slice of Radio – Wireless RF Tranceiver

hind: £10

hind eurodes: £10 * 1,27 = 12,7€

Digitus juhtmevaba 150N USB adapter

hind: 13,85€

Lenovo vooluadapter. Antud vooluadapteri hinda ei teata, seega siin võeti hinnaks samasuguste näitajatega adapteri Telemark TC-USB2A hind [16].

hind: 11€

SanDisk M2 2 GB MS Micro

hind: 6,8€

USB 2.0 A/B tüüpi kaabel

hind: 2,2€ [17]

Ühenduskaablid hind: 2,7€ [18]

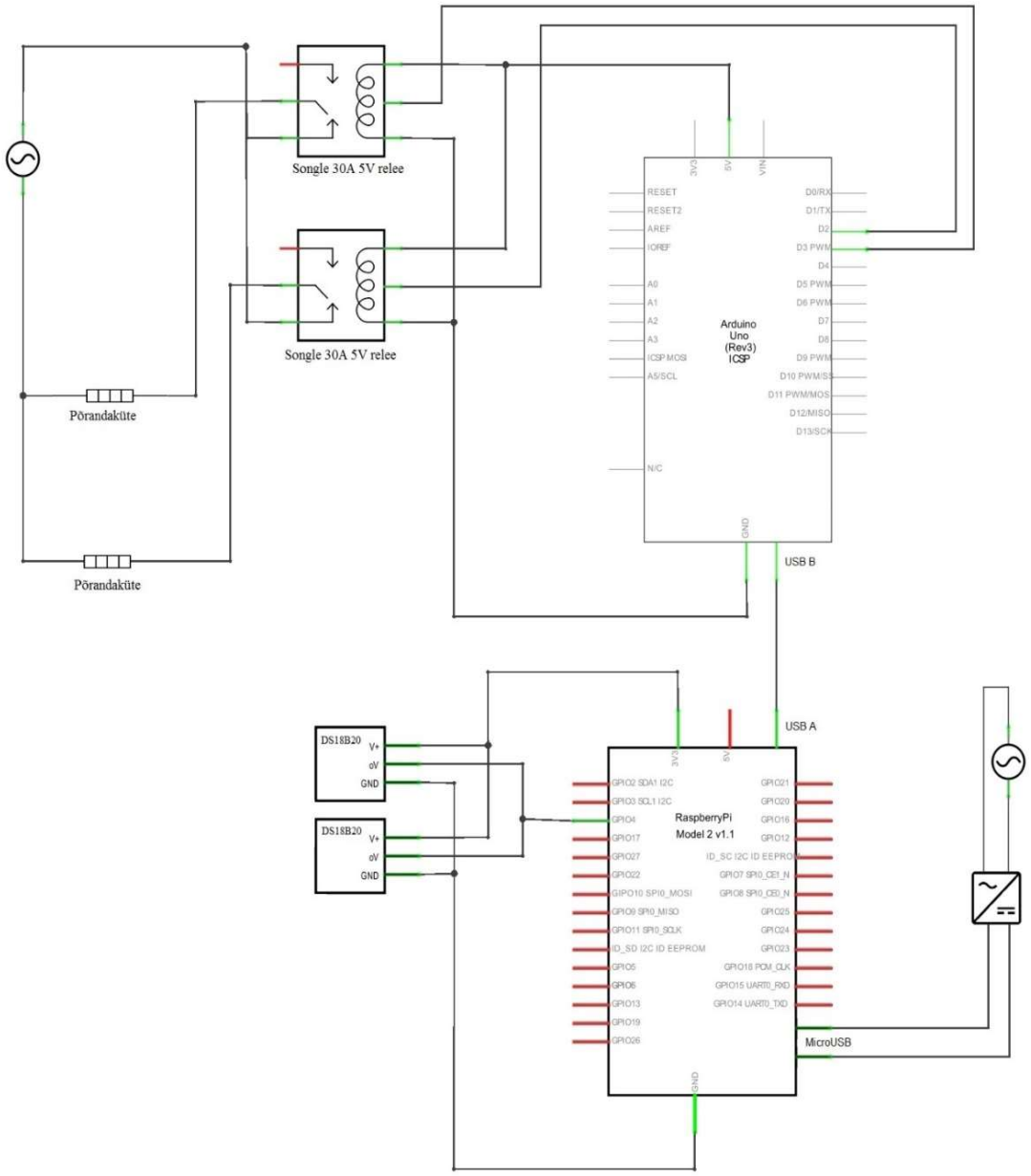
**Hinnaarvutus:**

30,8 + 25 + 9,36 + 9,2 + 12,7 + 1 + 12,7 + 13,85 + 11 + 6,8 + 2,2 + 1,9 = 137,31 (€)

Seadme kogu elektroonika maksumuseks saadi 137 eurot ja 31 senti.

## 1.3. Seadme koostamine

Optimeerimisseadme komponentide omavaheline ühendamine toimus vastavalt elektroonika-skeemile selel 1.8. Skeemilt on puudu WirelessThings raadiosignaali vastuvõtja ning juhtmevaba temperatuuriandur. Vastuvõtja on paigutatakse otse 26-le Raspberry Pi 40-st sisend-väljund kontaktist. Releed ühendati põrandakütete külge, mis on varasemalt koostatud ja paigaldatud professionaalsete elektrikute poolt. Skeemi osad, mis on üle 24 V vahelduvvoolu on projekteeritud vastava kategooria elektriinseneri poolt ja ei kuulu antud töö koosseisu.




**Sele 1.8. Seadme koosteskeem.**

# 2. JUHTIMINE

## 2.1. Täidetavad ülesanded

SLSi pilveteenus on ülesehitatud nii, et enamik keerulisemaid ülesandeid lahendab pilvetarkvara. Selline ülesehitus tagab, et objektil asuv seade saab olla programmeeritavuse seisukohalt võimalikult lihtne.

Et SLSi pilveteenuse abil põrandakütete tööd juhtida, peab koostatud seade olema võimeline täitma teatud ülesandeid. Järgnevalt kirjeldatakse neid ülesandeid ning nende rolli optimeerimisprotsessis.

**Masinõppe tarkavarale informatsiooni saatmine**

Masinõpe ise toimub SLSi pilves, seega peab seade olema võimeline saatma iga minuti järel pilve sensoritelt kogutud infot, milleks on termomeetrite näidud ning kütteelementide sisendvõimsused. Saadud info põhjal ning järgneva 24 tunni elektribörsi hindade põhjal saab masinõpe koostada antud seadme jaoks optimeerimisgraafiku.

**Andmete kogumine**

Seade peab olema võimeline koguma andmeid temperatuuri anduritelt. Kuna releed on seadistatud nii, et nende asendit ei saa sujuvalt reguleerida vaid need saavad vaid olla kas sisse või välja lülitatud, saavad sisendvõimsused olla vastavalt kas 0 või maksimaalne väärtus. Seega saab küttekeha võimsust sisse lülitatud olekus võtta kui küttekeha maksimaalset võimsust 1,85 kilovatti.

**Optimeerimise küsimine**

Seade peab läbi interneti saama ühendust SLSi pilvega, täpsemalt igal tunni tagant küsima SLSi pilvest optimeerimist.

**Optimeerimise salvestamine**

Seade peab pärast optimeerimise päringu saatmist ning optimeerimisgraafiku kättesaamist oskama graafikut salvestada.

**Optimeerimise teostamine**

Seade peab salvestatud optimeerimiskäskudele vastavalt põrandakütte elementide tööd juhtima.

**Lokaalne kontroll**

On võimalik määrata temperatuuri ülemine ja alumine piir, mille vahel peab korteritemperatuur asuma. Seade peab olema võimeline kontrollima ega need piirid pole ületatud.

**Internetiühenduse kontroll**

Infovahetus SLSi pilvega käib läbi interneti. Seega on tarvis, et seade kontrolliks interneti-ühenduse olemasolu.

## 2.2. Juhtimise põhimõtted

Eelmises punktis kirjeldatud ülesannete täitmine peab toimuma teatud loogilises järjekorras. Sel põhjusel on ülesanded omavahel rühmitatud ning koostatud on algoritmid, mis tagavad seadme rikketu töö.

**Seadme käivitamine**

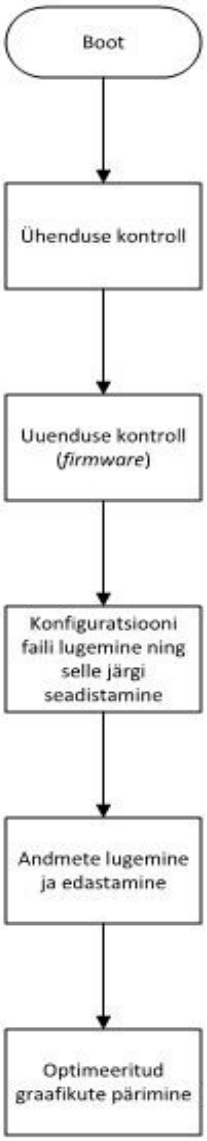
Seadme käivitamise korral teeb seade kõigepealt interneti-ühenduse kontrolli. Seejärel kontrollib seade firmware'i uuenduste olemasolu. Järgmiseks loeb seade konfiguratsiooni faili ning seadistab end vastavalt (sele 2.1.). Konfiguratsiooni fail on kogum parameetreid, mille järgi seade teab, kuidas see töötama peab.

**Ühenduvuse kontroll**

Ühenduvuse kontrolli teostatakse enne iga tegevust, mis nõuab ühenduse olemasolu SLSi pilvega. Kontrolli käigus testitakse erinevaid võimalusi, miks ühendus puudu võib olla. Ühenduse puudumise korral väljastatakse vastav signaal (sele 2.2.).

**Andmete lugemine ja edastamine**

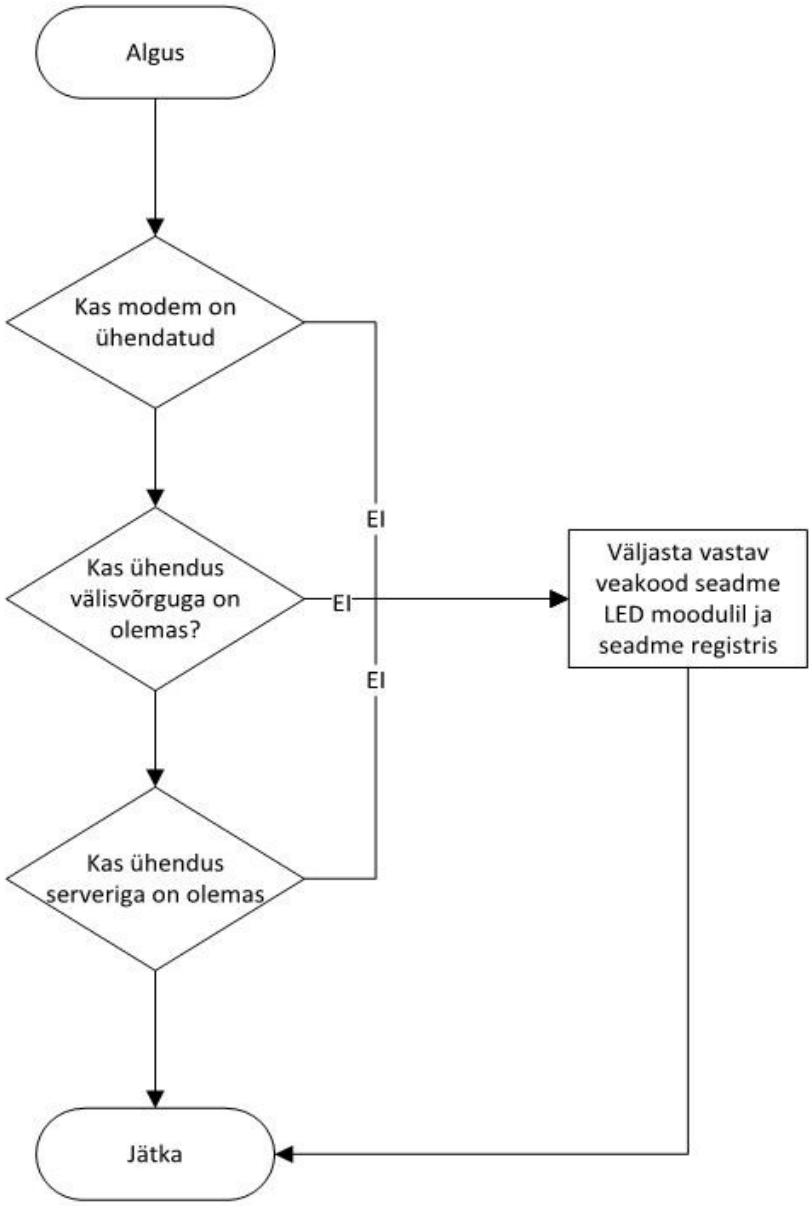
Temperatuuri anduritelt andmete kogumist teostatakse pidevalt. Seade salvestab andurite näite ning arvutab 10 viimase salvestatud näidu keskmise väärtuse. See eemaldab temperatuurinäitude liigse kõikumise, mis muidu temperatuuriandurite täpsustest tuleneb. Põrandakütte võimsus arvutatakse voolureleede oleku järgi,



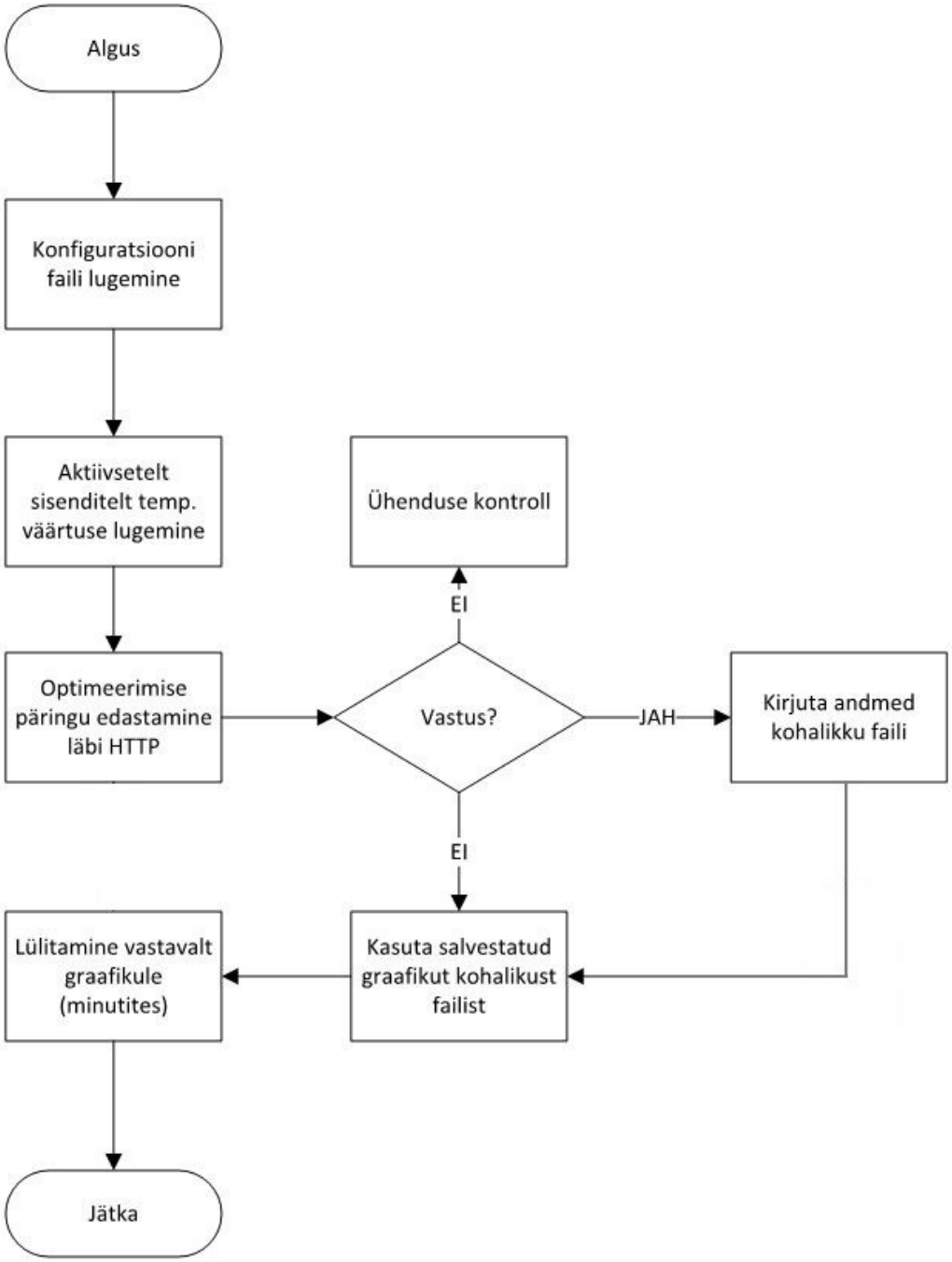

**Sele 2.1. Tegevuste põhimõte seadme käivitamisel**

sisselülitatud voolurelee tähendab, et küttekeha töötab täisvõimsusel, väljalülitatud olek tähendab, et küttekeha võimsus on 0.

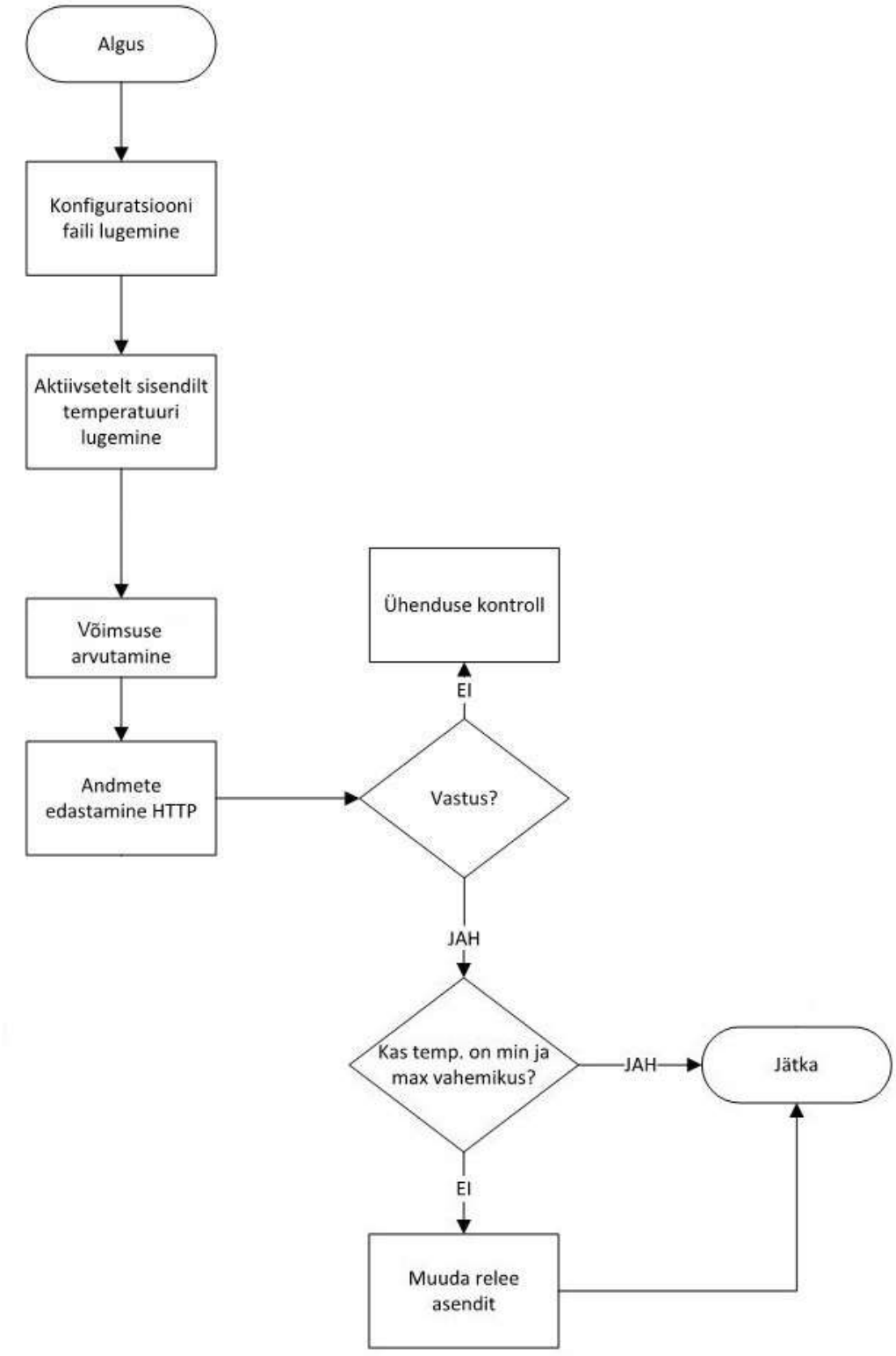
Kogutud andmed edastatakse SLSi pilve antud seadmele määratud ID all. ID on igale optimeerimisseadmele SLSi pilvest määratud identifitseerimiskood. Salvestatud andmete põhjal luuakse pilves optimeerimisgraafik. Ühtlasi kontrollib seade, kas temperatuur toas on ettemääratud vahemikus ning muudab releede asendeid, kui peaks vaja olema. Neid tegevusi korratakse iga minuti järel (sele 2.3.).




**Sele 2.2. Ühenduse kontrolli tööpõhimõte**




**Sele 2.3. Andmete kogumise ja edastamise tööpõhimõte**




**Sele 2.4. Optimeerimise tööpõhimõte**

**Optimeerimine**

1-tunnise intervalliga teostab seade optimeerimise päringu. Päring saadetakse SLSi pilve koos seadme ID-ga. Päringule vastuseks saab seade optimeerimisgraafiku, mis salvestatakse seadme mällu. Optimeerimisgraafik on graafik, milles on optimeerimisseadmele juhised. Juhiste järgi teab seade, mitu minutit peab põrandaküte soodsatel tundidel täisvõimsusel töötama, et toatemperatuur püsiks soovitud vahemikus. Vastavalt saadud graafikule toimub põrandakütte sisse- ja väljalülitamine (sele 2.4.).

## 2.3. Programmeerimiskeele valik

Raspberry Pi-le ja Arduino Unole on tarvis laadida tarkvara, mis üle-eelmises punktis seatud ülesandeid täita suudab eelmises punktis toodud põhimõtetel. Valmistatud algoritmide realiseerimiseks koodis on tarvis valida programmeerimiskeeled, millega Raspberry Pi-le ning Arduino Unole programmid valmistatakse.

**Raspberry Pi**

Kogu seadme töö juhtimine käib läbi juhtarvuti, Raspberry Pi. Raspberry Pi programmeerimise puhul on keele valikuga mitu varianti. Iga keel, mis suudab kompileerida protsessoril ARMv7 arhitektuuriga, sobib Raspberry Pi 2 programmeerimiseks. Tootjate soovitatud keeled on Python ja Scratch, kuid sobivad ka C, C++, Java ja Ruby [19]. Töö koostajal on varasemat kogemust programmeerimiskeeles Python, seega uute keelte õppimisega tuleneva lisatöö vältimiseks see keel ka valiti.

**Arduino Uno**

Arduino programme saab kirjutada iga programmeerimiskeelega, mille kompileerija on võimeline tootma binaarkoodi. Arduino Uno programmeerimiseks saab kasutada Arduino IDE-d (integrated development environment), mis programmeerimiskeeles Java valmistatud aplikatsioon ning baseerub programmeerimiskeelel Processing. On võimalik programmeerida keeltes C ja C++ kasutades teatud reegleid koodi organiseerimiseks [20]. Varasema kogemuse põhjal keelega, valiti Arduino Uno programmi koostamiseks keel C.

## 2.4. Juhtimise teostamine

Koostati programmid eelnevalt valitud keeltes ning eelnevalt kirjeldatud põhimõtete põhjal. Programmide koostamisel jälgiti, et kõik püstitatud juhtimisülesanded saaksid täidetud. Koostatud programmid laeti kontrollerite mällu. C-keeles valmistatud programm laetakse Arduino Unole, Pyhtonis kirjutatud programm Raspberry Pi-le. Raspberry Pi-le laetavad programmid paigutatakse microSD mälukaardile, mis omakorda asetatakse Raspberry Pi-sse vastavasse pessa. Raspberry Pi juthimise programmikoodid on toodud lisades 1 kuni 3.

Lisaks programmile laetakse Rasbperry Pi-le ka konfiguratsioonifail. Konfiguratsioonifail sisaldab endast hulga parameetreid. Nende parameetrite hulka kuuluvad näiteks optimeerimis-seadme ID ning temperatuuride miinimumid ja maksimumid. Konfiguratsioonifailis olevad miinimum ja maksimumväärtused ei ole samad väärtused, mis vahemikus peab seade toatemperatuuri hoidma. Konfiguratsioonifaili miinimum ja maksimum on absoluutsed väärtused, mis tagavad, et temperatuur korteris ei tõuseks üle ega langeks alla ohtliku piiri.

# 3. MEHAANIKA

Olemasolevale elektroonikale disainiti korpus, mis kogu seadme füüsilise ülesehituse kompaktsemaks teeb. Elektroonika peatükkis kirjeldatud nõute tõttu on korpuse asukoht korteris elektrikapi juures mõne aluse peal.

Korpus on disainitud teatud põhimõtetest lähtuvalt:

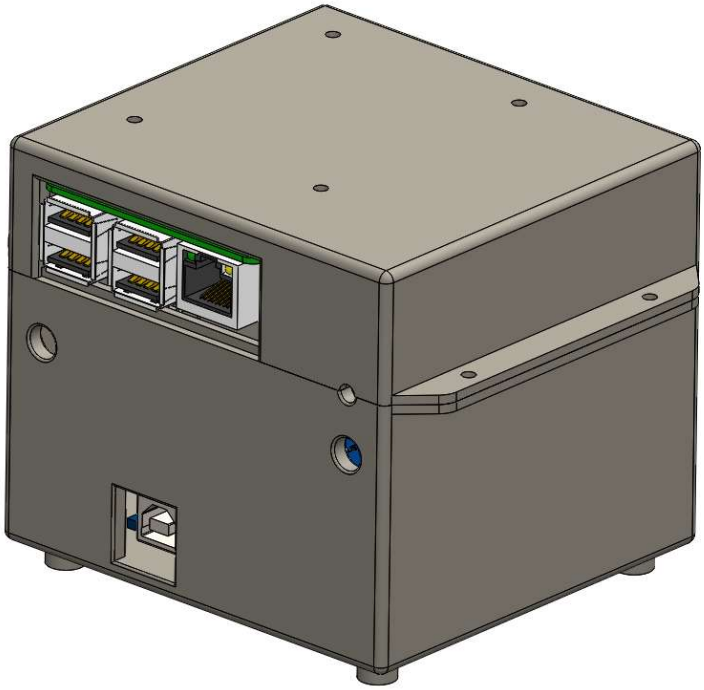
1) Korpus peab tagama seadme kompaktsuse.
2) Korpus peab elektroonikakomponendid kindlalt enda külge siduma.
3) Korpuses peab olema ruumi komponentide vahel õhu liikumiseks. Samuti peab olema ruumi komponente ühendavate kaablite jaoks.
4) Korpuses peavad olema avad elektroonikakomponentide ühendusportidele (USB-pesad) ligipääsuks ning avad kaablitele (küttekehade voolujuhtmed releede tarvis).
5) Korpus võiks olla 3D-prinditav, materjaliks PLA (Polylactic acid).

Korpus disainiti CAD (computer-aided design) tarkvara Solidworks abil.

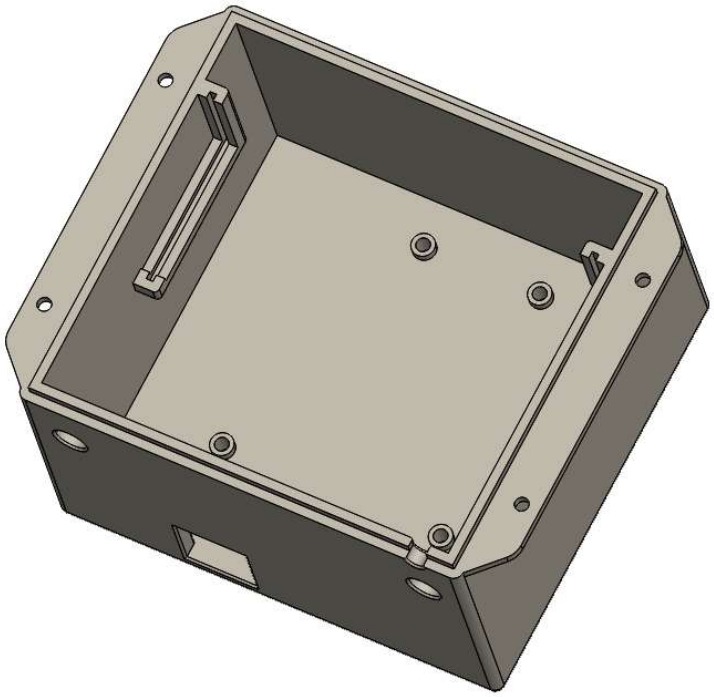
## 3.1. Korpuse disain

Korpus koosneb kahest osast: alumine osa (sele 3.2), mille põrandale kinnitub Arduino Uno ja seintele kinnituvad voolureleed, ning ülemine osa (sele 3.4), mille lae külge kinnitub Raspberry Pi 2 model B. Kahe osa ühendamiseks on osade külgedel avadega tiivad, kinnitamine käib läbi avade M3 poltide ja mutritega. Lisaks on korpuse esiküljel kaks 7-millimeetrise läbimõõduga ava releesid küttekehadega ühendavate voolujuhtmete jaoks ning üks 4-millimeetrise läbimõõduga ava seadmele toidet pakkuva microUSB-kaabli jaoks. Korpuse all on neli 5-millimeetri kõrgust jalga, mille peal korpus seisab (sele 3.3). Korpuse osade joonised on toodud lisas 1.

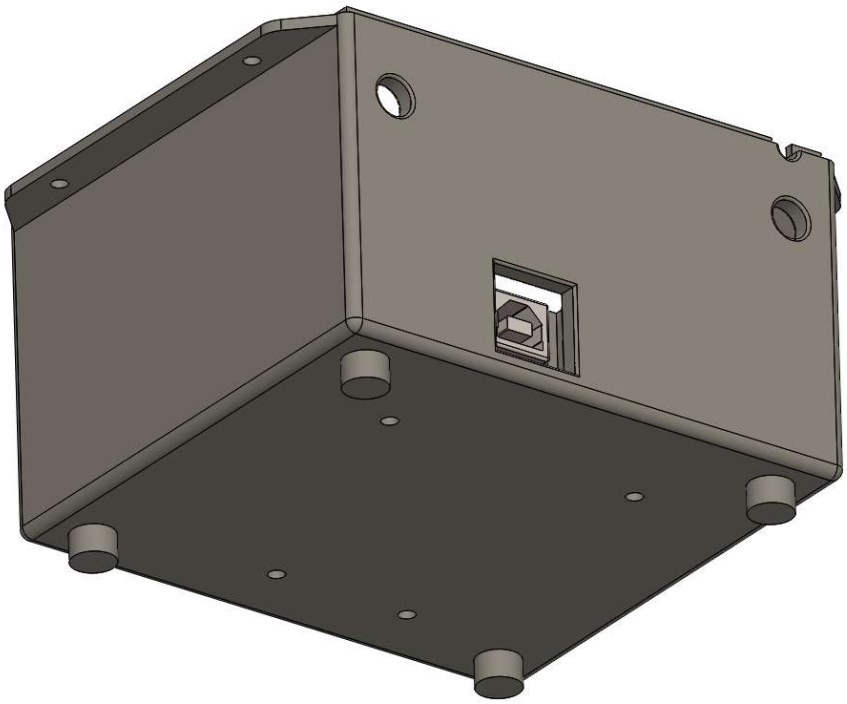
M3 poltide ja mutritega kinnituvad vastavate korpuste osade külge ka Raspberry Pi ja Ardunio Uno. Disainimise käigus laeti saidilt https://grabcad.com/ alla Arduino Uno ning Rasbperry Pi 2 CAD-mudelid [21] [22]. Need komponendid paiknevad 3-millimeetriste jalgade peal, mis asuvad komponentide kinnitusavadega kohakuti. Songle voolureleedel kinnitusavad puuduvad, need sobituvad korpuse seintele disainitud pesadesse.



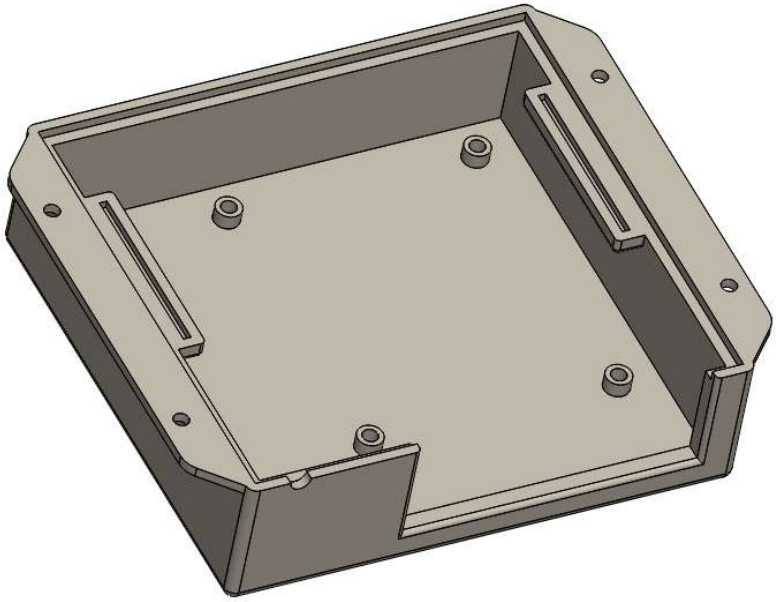
**Sele 3.1. Kokkupandud korpus Arduino Uno ja Raspberry Pi 2 mudelitega.**



**Sele 3.2. Korpuse tühi alumine osa ülevalt vaadates**



**Sele 3.3. Korpuse alumine osa alt vaadates**



**Sele 3.4. Korpuse tühi pealmine osa alt vaadates**

# 4. TULEMUSTE ANALÜÜS

Tulemuste analüüsi tarbeks pandi seade katseperioodiks põrandakütete tööd juhtima. Katseperioodi algus oli 1. mai õhtul kell 10 aastal 2016 ning katseperioodi lõpp oli 4. mai õhtul kell 10 aastal 2016. Selle aja vältel jälgiti põrandakütete töö optimeerimise tulemusi SLSi klienditele mõeldud veebileheküljelt, kus on klientidel võimalik seadmele määrata soovitud temperatuurivahemikku ning vaadata seadme töögraafikuid. Graafikuid on võimalik määrata näitama elektrihinda börsil, toaõhu temperatuuri, välistemperatuuri, kütteseadme võimsust, ning seadmele etteantud temperatuurivahemikku. Välistemperatuuri väärtused on saadud SLSi pilvest, kuhu need on salvestatud muust välisest allikast.

Katseperioodi vältel korteris inimesi ei viibinud, kui välja arvata 2. mai päeval kell 12, kui käidi katseseadet ning põrandakütte termostaate üle vaatamas. Katsealuste ruumide kõik uksed olid kinni ning kogu korteri keskkütteradiaatorid olid välja lülitatud. Katseperioodil olid mõlema katsealuse toa põrandakütte termostaadid keeratud maksimaalse võimsuse asendisse. Katseperioodi vältel kasvas välistemperatuuri maksimum 18,7 kraadini celsiuse järgi (2.05.2016. päevane maksimum 15,9°C ja 05.05.2016. pävane maksimum 18,7°C) ja päike paistis enamuse osa valgest ajast (vahemikus 14,8 tundi kuni 15,2 tundi, päeva pikkus kasvas 16 tunnilt ja 38 sekundilt 02.05.2016. 16 tunni 15 minuti ja 34 sekundini 05.05.2016.) [23] [24].

## 4.1. Graafikute analüüs

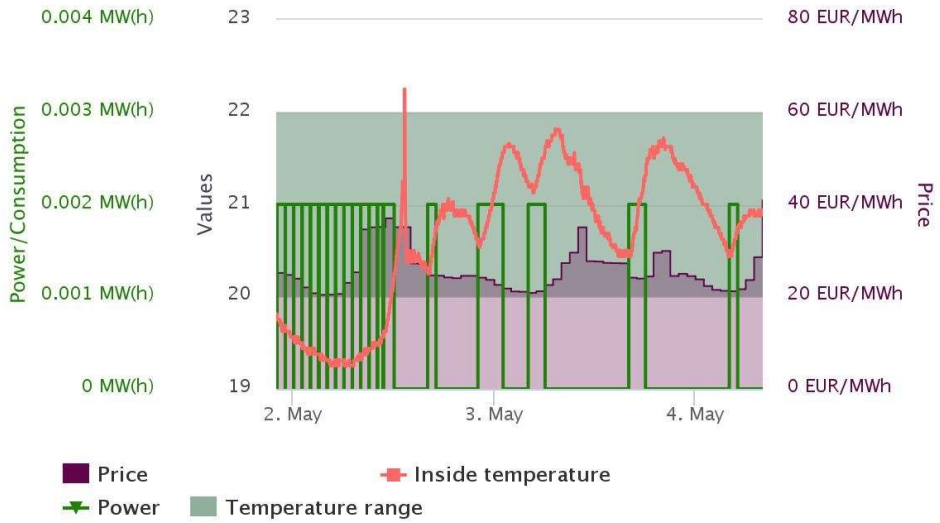
Järgnevalt on analüüsitud katseperioodi töögraafikuid katseperioodil.

### 4.1.1 Alumise toa töögraafiku analüüs

Optimeerimisseadmele määrati temperatuurivahemik 20 kuni 22 ºC. Kuna katse alguses oli temperatuur toas alla vahemiku miinimumi, lülitas seade küttekeha sisse. 2. mai ööl ja hommikul on näha toatemperatuuri languse aeglustumist ning ka temperatuuri kasvu. Sellel hetkel hoidis seade põrandakütteid sees vaatamata elektrienergia hetkehinnale.

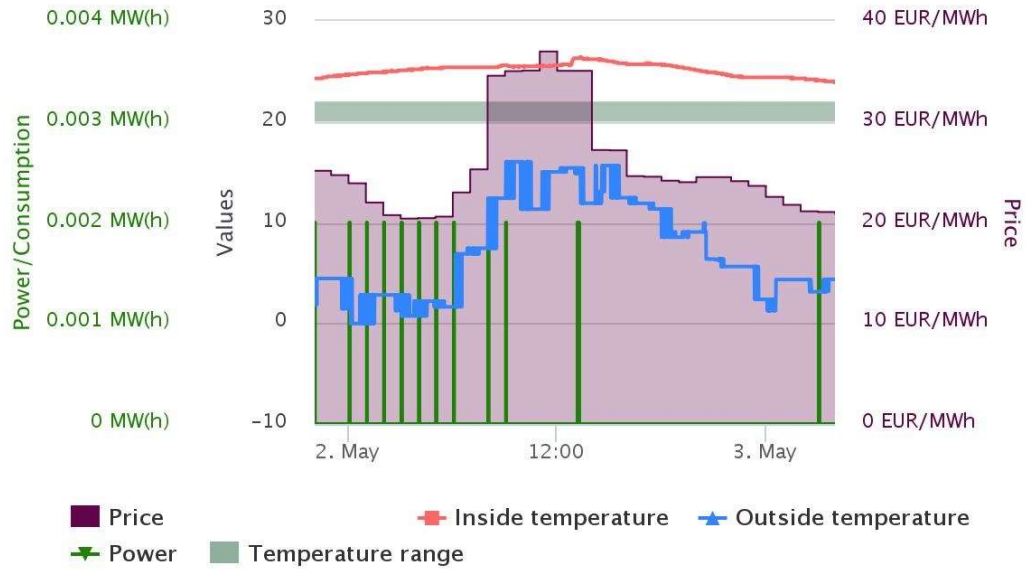
Kui temperatuur ületas miinimumi piiri ning jätkas soojusliku inertsi tõttu kasvu, lülitas seade põrandakütte välja. Järsk kasv üle vahemiku maksimumi on tingitud sellest, et katseobjektil käidi seadet üle vaatamas. Objektil käimise käigus tõsteti ümber temperatuuriandureid ning graafikul selel 4.1 on näha selle mõju temperatuurianduri näitudele temperatuuri järsu kasvu, ning hiljem languse, näol. Selline hüpe tuleneb inimese kehasoojuse ülekandumisest andurile.

Edasi on 2. mai keskpäevast 4. mai õhtuni näha seadme ootuspärast tööd. Temperatuuri langemise korral lülitatakse odava elektrienergiaga tundidel põrandaküte sisse, piisava kasvu korral jällegi välja. Tuleb tähele panna, et väiksema elektrienergia hinnaga tundidel soojendati tuba maksimumpiiri lähedale, et hiljem kallimate tundide ajal poleks tarvidust kütta (sele 4.1.).

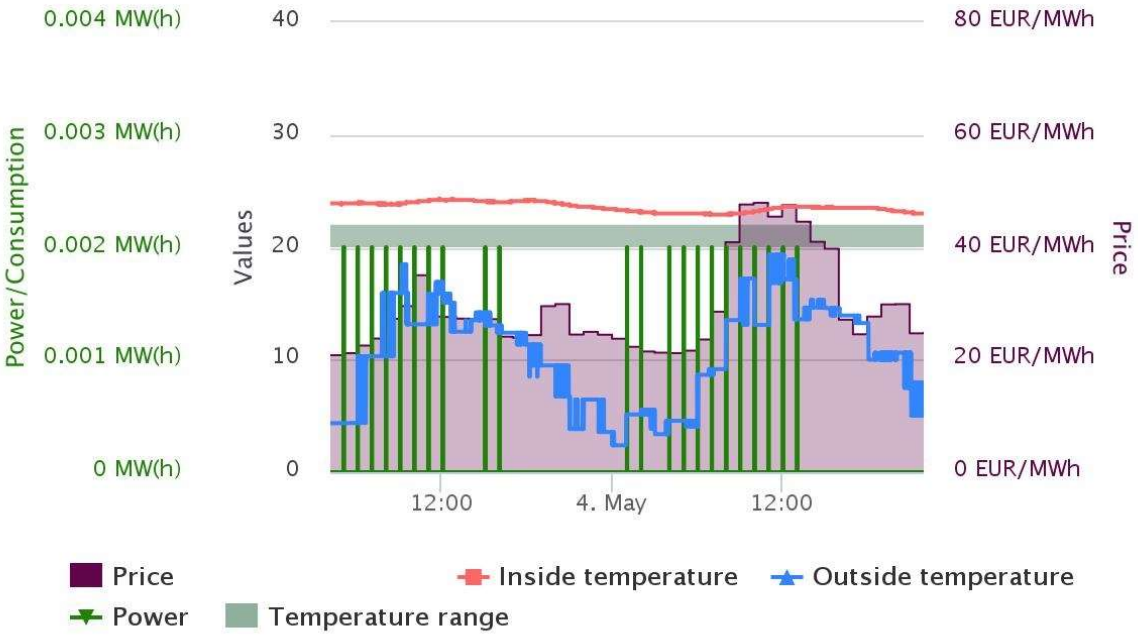



**Sele 4.1. Alumise toa katseperioodi töögraafik**

### 4.1.2 Ülemise toa töögraafiku analüüs




**Sele 4.2. Ülemise toa katseperioodi esimese poole (1. mai kell 22:00 kuni 3. mai kell 04:00) töögraafik**




**Sele 4.3. Ülemise toa katseperioodi teise poole (3. mai kell 04:00 kuni 4. mai kell 22:00) töögraafik**

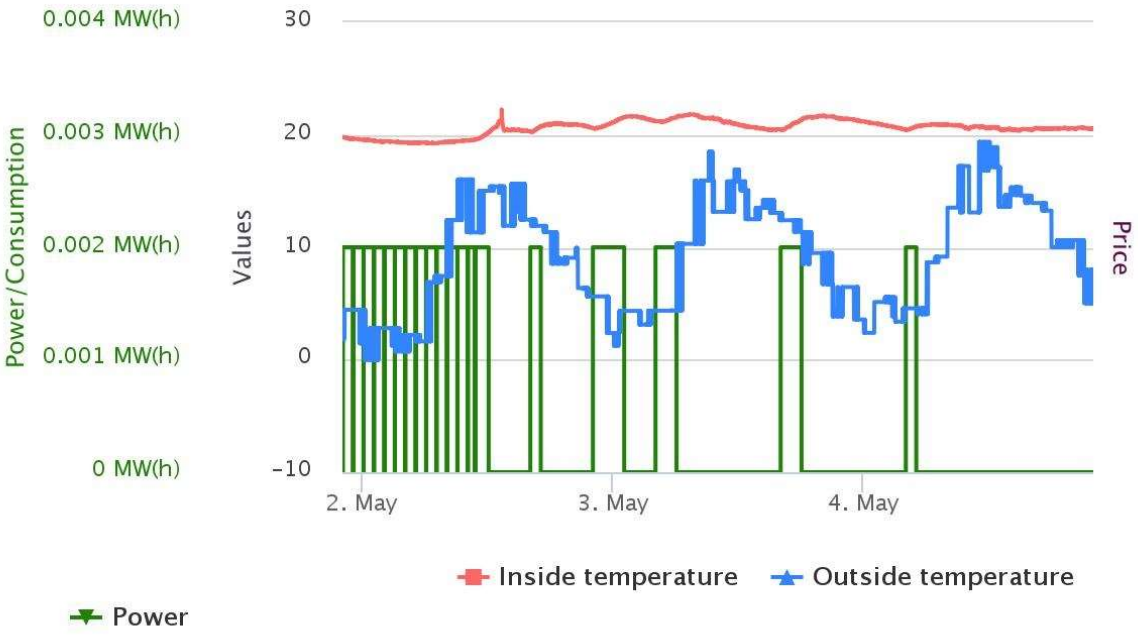
Optimeerimisseadmele määrati temperatuurivahemik 20 kuni 22 ºC, nagu alumises toaski. Katseperioodi alguses oli temperatuur toas 24,6 ºC, seade hoidis seega põrandakütet väljalülitatud olekus. Kuid graafikul selel 4.2 on näha, et toatemperatuur langema ei hakka. Samuti on näha ka graafikul selel 4.3, et temperatuur toas ei langegi ettemääratud temperatuurivahemikku.

Ülemises toas seade põrandakütte tööd katseperioodil küll optimeeris, kuid optimeerimine nägi ette, et põrandaküte töötama ei pea. Sellel on üks võimalik põhjus.

Mõlema toa põrandakütete töö käis identse juhtimissüsteemi järgi. Alumise toa töögraafiku analüüsis leiti, et selle toa põrandakütte tööd juhiti vastavalt püstitatud eesmärkidele. Seega mõlemas toas põrandakütete töö optimeerimine toimus.

Põhjuseks, miks temperatuur ülemises toas oli üle maksimumi piiri, on ilm. Ülemine tuba asub kortermaja lõunapoolses otsas ning selle kaks akent avanevad lõuna suunda. Katseperioodi vältel paistis päevasel ajal päike enamuse osa päevast akendest sisse ning soojendas tuba. Ühtlasi oli katseperioodi vältel päevane temperatuur õues 15 kuni 19 ºC. Külmema välitemperatuuri korral oleks toatemperatuur ilmselt oluliselt kiiremini langenud.

Alumises toas seda probleemi ei tekkinud, kuna tuba on ümbritsetud teiste tubadega ja selle seintest polnud nii suur osa kortermaja välisseinad. Samuti paistab akendest tuppa päikesevalgust vaid õhtupoole, päikesekiirgus ei soojenda tuba nii palju üles. Graafikul selel 4.4 on näha, et välistemperatuur alumise toa sisetemperatuuri ei mõjutanud.




**Sele 4.4. Alumise toa temperatuuri ning välistemperatuuri suhte graafik koos põrandakütte töötsüklitega katseperioodil**

Ülemise toa töögraafikutel seledel 4.2 ja 4.3 on näha põrandakütte hetkelisi sisselülitamisi (võimsuse graafik). Ilmselt on sisselülitused toimunud mingisuguse tarkvaralise vea tõttu.

## 4.2. Põrandakütte töö majanduslik analüüs

Käesolevas alapeatükkis arvutatakse elektribörsipõhise optimeerimisega ning optimeerimiseta põrandakütte töö kulude vahet. Arvutused teostatakse ainult alumise toa kohta, kus toimus eesmärgipõhiseid põrandakütte ümberlülitamisi. Ülemise toa elektrienergia tarbimine katseperioodil oli 0,5 kWh (kilovatt-tundi) (tarbimine toimus hetkeliste sisselülituste ajal) ning arvutamisel pole eesmärki.

SLSi klientidele mõeldud veebileht kuvab lisaks töögraafikutele ka keskmist elektrihinda valitud perioodil. Samuti on võimalik kuvada nö kaalutud keskmist hinda, mis arvestab

keskmise hinna arvutamisel ainult nende tundide hindu, mil põrandaküte oli sisselülitatud ning vastavate kaaludega lähtuvalt tunnis tarbitud võimsusest. Veel kuvatakse valitud perioodi elektritarbimist. Nende parameetrite väärtuse arvutatakse välja SLSi pilves.

Eelnimetatud katseperioodil oli kogu aja keskmine elektrienergia hind $Hind_{kesk}$ = 27,75 €/MWh (eurot megavatt-tunni kohta), elektrienergia kaalutud keskmine hind oli $Hind_{kaal\ kesk}$ = 24,94 €/MWh. Elektrienergia tarbimine katseperioodil oli *Tarbimine* = 0,0456 MWh. Lihtsuse mõttes ei arvestata ei arvestata arvutustes võrgutariife.

Saab arvutada elektrienergia maksumuse katseperioodil:

$$Maksumus_{optim} = Hind_{kaal\ \ kesk} \cdot Tarbimine = 24{,}94 \cdot 0{,}0456 = 1{,}138\ \ € \qquad (4.1)$$

Saab ka arvutada elektrienergia maksumuse oletades, et optimeerimisseadmeta oleks katseperioodi vältel tarbitud sama kogus elektrienergiat. Täpselt ei ole teada, kuidas seade ilma optimeerimiseta oleks elektrit tarbinud, kuid lihtsuse huvides oletatakse, et seade oleks tarbinud igas tunnis võrdse koguse elektrienergiat:

$$Maksumus_{tava} = Hind_{kesk} \cdot Tarbimine = 27{,}75 \cdot 0{,}0456 = 1{,}265\ \ €$$

(4.2)

Saab arvutada kahe maksumuse erinevuse protsentides:

$$Sääst = 100 - \frac{Maksumus_{optim}}{Maksumus_{tava}} \cdot 100 = 100 - \frac{1{,}138}{1{,}265} \cdot 100 = 10{,}04\ \ \%$$

(4.3)

Leiti, et optimeerimisega saavutati 10,04 %-ne sääst.

Sääst oleks võinud kindlasti olla ka suurem. Kui arvestada välja katseperioodi algus, kui toatemperatuuri viimiseks nõutud vahemikku põrandaküte töötas järjest hoolimata elektrienergia hetkehinnast.

# KOKKUVÕTE

Käesoleva töö käigus disainiti seade, mis optimeerib kahe põrandakütte tööd ettevõtte Smart Load Solutions-i pilvetarkvarast saadud juhendite põhjal. Seade valmistati ning optimeerimist analüüsiti.

Töö esimeses osas tegeldi optimeerimisseadme elektroonikaga. Seadme tööd valiti juhtima Rasbperry Pi model B ning voolureleede lülitamist valiti juhtima Arduino Uno. Ühe katsealuste toa temperatuuri valiti mõõtma nii juhtmega temperatuuri andur DS20B18, teise tuppa juhtmevaba andur firmalt WirelessThings. Valiti välja ka seadme tööd toetavad komponendid nagu wifi-adapter, vooluadapter ning ühenduskaablid. Leiti seadme elektroonika kogumaksumus.

Töö teises osas kirjeldati optimeerimisseadme juhtimist. Püstitati seitse juhtimisega seotud ülesannet. Esitati seadme käivitamise, ühendevuse kontrolli, andmete lugemise ning edastamise ja optimeerimisgraafiku päringu esitamise ning optimeerimise teostamise põhimõtted. Seadme juhtimisprogrammide keelteks valiti Raspberry Pi-le Python ning Arduino Unole keel C.

Kolmandas osas disainiti seadmele korpus. Korpus disainiti nii, et seadme elektroonika oleks korpuse seintele kinnitatud ja kaitstud välismõjude, nagu lapsed ja koduloomad, eest. Korpusega tagati seadme kompaktsus. Esitati pilte disainitud korpuse CAD-mudelitest.

Töö viimases osas uuriti põrandakütte optimeeritud töö tulemusi. Optimeerimist teostati nelja päevasel katseperioodil. Selgus, et koostatud seade töötas nagu oli ette nähtud. Arvutati põrandakütte optimeeritud töö maksumus ning optimeerimata töö maksumus ning neid võrreldi. Tulemuseks saadi, et optimeeritud tööga hoiti elektrienergia maksumuses kokku 10,04 protsenti.

Koostatud optimeerimisseadme edasiseks uurimiseks tuleks pikendada katseperioodi ning analüüsida saadavaid tulemusi. Samuti tasuks järgnevates uurimistes leida, kuidas optimeerimata küttekeha tegelikult töötab.

Kõik töö alguses püstitatud eesmärgid said täidetud. Seade hoidis ühes toas ettemääratud temperatuuri, seejuures töötas vaid majanduslikult soodsatel tundidel. Temperatuur teises toas oli välistingimuste tõttu üle nõutud piiri ning seade tegi seda milleks oli võimeline: hoidis küttekehasid väljas. Seega saab öelda, et koostati nõuetepäraselt töötav optimeerimisseade.

# CONCLUSION

During this bachelor's thesis a device was designed, that could optimize the work of two electric floor heaters based on the directions received from the cloud software developed by the company Smart Load Solutions. The device was put together and the optimization was analyzed.

In the first part of the thesis, the electronic part of the optimization device was worked on. Raspberry Pi 2 model B was chosen as the controller for the device and Arduino Uno was chosen for the switching of the relays. In addition, components that support the work of the device, like a Wi-Fi adapter, a power converter and connection cables, were chosen. The price of the electronic part of the device was calculated.

In the second part of the thesis the control of the optimization device was explained. Seven tasks regarding control were put forth. The principals of booting the device, of connection check, of the gathering and the transmission of data, and of requesting optimization and optimizing were presented. The programming languages chosen for the control programs were Python for Raspberry Pi and C for Arduino.

In the third part the case was designed for the device. The case was designed so that the electronics of the device would be fixed on the walls of the case and also protected from external influences, such as children and pets. Pictures of the CAD-model of the designed case were presented.

In the last part of the thesis the results of the work of the floor heaters were studied. Optimization was conducted in a four-day test period. It was made clear that device worked as was intended. The price of optimized work and non-optimized work was calculated and compared. As a result, it was found that with the optimization of work there was 10,04 per cent drop in the price of electrical energy.

For further study of the optimization device the test period should be lengthened and the results of this longer test period should be analyzed. In addition, the actual work of a non-optimized heater should be studied in further research.

All of the goals put forth in the beginning of the thesis were accomplished. The device regulated the temperature of one room according to pre-determined values, doing so on economically desirable hours. The temperature in the other room was above the pre-determined value due to external weather conditions and the device acted accordingly: it kept the heating elements

switched off. Thus, it is possible to conclude that an optimization device was assembled that fulfilled all the requirements.

# KASUTATUD KIRJANDUS


1 Elektri tarbimine http://www.energiatalgud.ee/index.php?title=Elektri_tarbimine&menu-26 (14.05.2016)

2 Nord Pool Spot koduleheküIg http://nordpoolspot.com/ (14.05.2016.)

3 Songle 30A 5V C-Type Opto-Isolated relee tehnilised parameetrid ja hind http://eud.dx.com/product/5v-30a-c-type-opto-isolated-relay-module-blue-844214657#.Vzc_RuTYZBj (16.02.2016.)

4 Raspberry Pi 2 Model B tehnilised andmed https://www.adafruit.com/pdfs/raspberrypi2modelb.pdf (19.02.2016.)

5 Raspberry Pi 2 model B hind ja pilt https://www.element14.com/community/community/raspberry-pi/raspberrypi2 (19.02.2016.)

6 Arduino Uno hind ja pilt http://store.mansteri.com/index.php/fi/arduino/genuino-arduino-boards/genuino-uno.html (19.02.2016.)

7 Arduino Uno tehnilised andmed https://www.arduino.cc/en/main/arduinoBoardUno (19.02.2016.)

8 Beaglebone Black hind ja pilt https://www.adafruit.com/product/1278 (19.02.2016.)

9 Beaglebone Black tehnilise andmed http://www.farnell.com/datasheets/1818802.pdf (19.02.2016.)

10 CR2032 patarei http://www.oomipood.ee/en/product/hq_cr2032_cr2032_liitium_patarei_3v_200mah_20mm_3_2mm_1tk?q=CR2032 (13.05.2016.)

11 Temperatuurianduri DS18B20 andmed http://eud.dx.com/product/waterproof-ds18b20-temperature-sensor-100cm-ds18b20-adapter-module-for-arduino-844414361#.Vzc_d-TYZBj (03.03.2016.)

12 WirelessThings juhtmevaba anduri SB-CA andmed https://www.wirelessthings.net/wireless-temperature-sensor (03.03.2016.)

13 WirelessThings Slice of Radio – Wireless RF Tranceiver andmed https://www.wirelessthings.net/slice-of-radio-wireless-rf-transciever-for-the-raspberry-pi (03.03.2016.)

14 Digitus juhtmevaba 150N USB adapter http://arvutitark.ee/est/tootekataloog/Vorguseadmed-USB-WiFi445/DIGITUS-juhtmevaba-150N-USB-adapter-35359 (05.03.2016.)

15 SanDisk M2 2 GB MS Micro

http://www.onoff.ee/foto--ja-videokaamerad/malukaardid/sandisk-m2-2-gb-ms-micro/ (05.03.2016.)

16 Telemark TC-USB2A vooluadapter

http://www.onoff.ee/telefonid-ja-gps/mobiilitarvikud/telemark-tcusb2a/ (05.03.2016.)

17 USB 2.0 A/B tüüpi kaabel http://progear.ee/accu-cable-ac-usb-ab-1-kaabel (16.02.2016.)

18 Arduino Uno ühenduskaablid http://www.eu.diigiit.com/jumper-wire-male-female-40pcs-20cm (13.05.2016.)

19 Raspberry Pi korduma kippuvad küsimused https://www.raspberrypi.org/help/faqs/#softwareOS (19.02.2016.)

20 Arduino https://en.wikipedia.org/wiki/Arduino (19.02.2016.)

21 Arduino Uno CAD-mudel https://grabcad.com/library/arduino-uno-r3-3 (14.05.2016)

22 Raspberry Pi CAD-mudel https://grabcad.com/library/raspberry-pi-2-1 (14.05.2016)

23 Ilmavaatlused perioodil 02.05.2016. kuni 05.05.2016. http://www.ilmateenistus.ee/ilm/ilmavaatlused/vaatlusandmed/oopaevaandmed/ (19.05.2016.)

24 Päeva pikkused perioodil 02.05.2016. kuni 05.05.2016. http://www.timeanddate.com/sun/estonia/tallinn (19.05.2016.)

# LISAD

## Lisa 1 – Raspberry Pi 2 model B juhtimisprogrammi kood

```
import json
import requests
import datetime
import io
import time
import commands
import RPi.GPIO as GPIO
import serial
import numpy
import os
import smbus

def InitGPIO(pinNr):
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    #Relee
    GPIO.setup(pinNr,GPIO.OUT)

#FUNCTION DEFINITIONS
def GetJSON(filename):
    with io.open(filename, encoding="UTF-8-sig") as json_file:
        data = json.load(json_file)
        return data

#Logger
def logger(logMessage):
    date_now = datetime.datetime.now().time()
    print(date_now, logMessage)
    return

#Functions for I2C communication
def readI2CMessage():
    bus = smbus.SMBus(1)
    vals = bus.read_i2c_block_data(address,0)
    mess = ""
    for i in vals:
        if i < 255:
            mess += chr(i)
    return mess

def sendI2CMessage(message):
    vals = []
    for i in message:
```

```
            vals.append(ord(i))
        bus.write_i2c_block_data(address,vals[0],vals[1:])

#Serial communication
def ConnectArduino():
        ArduinoSerial = serial.Serial('/dev/ttyUSB0', 9600)
#       ArduinoSerial.flush()
#       ArduinoSerial.flushInput()
#       ArduinoSerial.flushOutput()
        time.sleep(2)
        return ArduinoSerial

def CommunicateArduino(message, response, i2c):
        responseMessage = ""
        if(i2c):
                sendI2CMessage(message)
                if(response):
                        responseMessage = readI2CMessage()
        else:
                ArduinoSerial = ConnectArduino()
                ArduinoSerial.write(msg)
                if(response):
                        responseMessage = ArduinoSerial.readline()
        return responseMessage

#Controller methods
#Power acquisition
def GetTemperatureFromArduino(pin, i2c = True):
        msg = "rT" + str(pin)
        TempReading = float(CommunicateArduino(msg, True, i2c))
        #ArduinoSerial.close()
        return TempReading

def GetPowerFromArduino(pin, i2c = True):
        msg = str.format("rA{0}", pin)
        PowerReading = float(CommunicateArduino(message, True, i2c))
        #ArduinoSerial.close()
        return PowerReading

def SwitchFromArduino(pin, switchPosition, i2c = True):
        msg = str.format("rR{0}S{1}", pin, switchPosition)
        a = CommunicateArduino(message, False, i2c)
        #ArduinoSerial.close()
        return

def DefinePriceColors(pin):
        #TODO:
        #Price acquisition
        priceRequest                                                                    =
requests.get('http://slsoptimizationservice.cloudapp.net/json/reply/PriceRequest')
```

```
        prices = []
        for i in json.loads(priceRequest.text):
            prices.append(i['value'])

        #Calculate percentiles
        splitPoints = [0,0]
        splitPoints[0] = numpy.percentile(prices, 100/3)
        splitPoints[1] = numpy.percentile(prices, 200/3)

        #Assign colour levels
        priceLevelMessage = "s"
        for i in range(1,3):
            if(prices[i] <= splitPoints[0]):
                priceLevelMessage += "1"
            elif(prices[i] <= splitPoints[1]):
                priceLevelMessage += "2"
            else:
                priceLevelMessage += "3"

        #Send
        ArduinoSerial = ConnectArduino()
        ArduinoSerial.write(priceLevelMessage)
        return

#Temperature acquisition
def GetSensorTemperature(SensorAddress):
    temp = open ('/sys/bus/w1/devices/%s/w1_slave' % SensorAddress)
    temp_text = temp.read()
    temp.close()
    tempdata = temp_text.split("\n")[1].split(" ")[9]
    temperature = float(tempdata[2:])
    temperature = temperature/1000
    return temperature

#Wireless temperature acquisition
def GetWirelessSensorTemperatures(sensors):
    DEVICE = '/dev/ttyAMA0'
    BAUD = 9600
    ser = serial.Serial(DEVICE, BAUD)

    insideTemperatures = []
    outsideTemperatures = []

    #MASSIVE STRING METHOD! BAUD! Made by Hackermen!!!
    #General message format: a + devID(2) + Message(9) i.e. "aITTEMP029.5"
    #TODO: Foolproofing if ID doesn't exist/send messages!
    while len(insideTemperatures) + len(outsideTemperatures) != len(sensors):
        startByte = ''
        try:
            startByte = ser.read(size = 1)
```

```
            except Exception as inst:
                print(inst)

            if startByte == 'a':
                msg = ser.read(size = 11)
                if msg[2:6] == 'TEMP':
                    devID = msg[0:2]
                    if any(d['Address'] == devID for d in sensors):
                        for d in sensors:
                            if d['Address'] == devID:
                                if d['Inside']:
                                    insideTemperatures.append(float(msg[7:]))
                                else:
                                    outsideTemperatures.append(float(msg[7:]))
        ser.close()
        return insideTemperatures, outsideTemperatures

    #Temperature retrieval
    def GetTemperatures(sensorArray):
        insideTemperatures = []
        outsideTemperatures = []

        insideTemperatures, outsideTemperatures =
    GetWirelessSensorTemperatures(sensorArray['Wireless'])
        if 'Wired' in sensorArray:
            for sensor in sensorArray['Wired']:
                if(sensor['Inside']):
                    insideTemperatures.append(GetTemperatureFromArduino(sensor['Address']))
                #OLD: insideTemperatures.append(GetSensorTemperature(sensor['Address']))
                else:
                    outsideTemperatures.append(GetSensorTemperature(sensor['Address']))

        #Calculate average!
        InsideTemperature = numpy.mean(insideTemperatures)
        OutsideTemperature = numpy.mean(outsideTemperatures)

        return InsideTemperature, OutsideTemperature

    #Switching
    def Switch(switchOn, pinNr):
        InitGPIO(pinNr)

        GPIO.output(pinNr,not switchOn)
        return

    def SaveSchedule(data, deviceID):
        PATH = '/SLS/SLSPrototype/'+deviceID+'/'
        if not os.path.exists(PATH):
            os.makedirs(PATH)
        with open(PATH+'schedule.json', 'w+') as outfile:
```

```
        json.dump(data, outfile)
    return

#Methods to communicate to SLS cloud
def SLSOptimizationRequest(StartValue, MinValue, MaxValue, MinPower, MaxPower, deviceID, OutsideTemperature = [0], StartUpCost = 0, ShutDownCost = 0, heatConductance = 0, heaterCoefficient = 0):
    url = 'http://slsoptimizationservice.cloudapp.net/json/reply/OptimizeHeaterCooler'

    #json message
    data = {}
    data['startTemperature'] = StartValue
    data['MinTemperature'] = MinValue
    data['MaxTemperature'] = MaxValue
    data['maxHeaterCoolerPower'] = MaxPower
    data['minHeaterCoolerPower'] = MinPower
    data['OutsideTemperature'] = OutsideTemperature
    #NOTE: Shut-down and Start-up costs are omitted.
    data['heatConductanceCoefficient'] = heatConductance
    data['heaterCoefficient'] = heaterCoefficient
    data['deviceID'] = deviceID
    #POST request
    r_optimization = requests.post(url, json = data)
    logger(data)
    if(r_optimization.status_code == 200):
        PowerSchedule = json.loads(r_optimization.text)
        SaveSchedule(PowerSchedule,deviceID)
    else:
        logger('Optimization schedule request SLS server failed!')
        #PowerSchedule = GetSavedSchedule(deviceID)
    return PowerSchedule

#Get power from saved schedule for upcoming hours
def GetSavedSchedule(deviceID):
    response = {}
    response["power"] = []
    SavedSchedule = GetJSON("/SLS/SLSPrototype/"+deviceID+"/schedule.json")
    now = datetime.datetime.now()
    logger('Optimizing from saved schedule')
    for (i,JsonTimeStamp) in enumerate(SavedSchedule["timeStamps"]):
        #Convert JSON string to python datetime object
        MILLISECONDS = JsonTimeStamp.split('(')[1][:-2]
        pythonDateObj = datetime.datetime.utcfromtimestamp(int(MILLISECONDS) / 1000)
        if(pythonDateObj > (now - datetime.timedelta(seconds = 3600))):
            response["power"].append(SavedSchedule["power"][i])
    return response
```

```
def SLSSendData(InsideTemperature, OutsideTemperature, Power, deviceID, authorizationHeader):
    data = {}
    data['ID'] = deviceID
    data['IT'] = InsideTemperature
    data['OT'] = OutsideTemperature
    data['P'] = Power

    authorization_header = {'Authorization' : authorizationHeader}
    r_data = requests.post('https://slsdataservice.servicebus.windows.net/slssensordata/messages', json = data, headers = authorization_header)
    if(r_data.status_code != 201):
        raise NameError('Data sending to SLS server failed!')
    return

#Local control
def LimitationStatus(Temperature, MinTemperature, MaxTemperature):
    #Note: for cooling this limitation should be switched!
    flipSwitch = False
    #off
    switchOn = True
    if(Temperature < MinTemperature):
        switchOn = True
        flipSwitch = True
    elif(Temperature > MaxTemperature):
        switchOn = False
        flipSwitch = True
    return flipSwitch, switchOn

def calculateMinutes(power, MaxPower):
    if(MaxPower > 0):
        return round(power/MaxPower)*60
    else:
        return 0
'''
def InsertDatabase():
    settings = GetJSON("settings.json")
    try :
        conn = MySQLdb.connect(host= "localhost",
                               user="root",
                               passwd="elpelp",
                               db="sls")
        cursor = conn.cursor()
        timestamp = int(time.time())
        add_row = ("INSERT INTO sls_data "
                   "(dev,temp1,temp2,temp3,amp,date,price) "
                   "VALUES (%s,%s,%s,%s,%s,%s,%s)")
        data = ("1",sensor2,sensor3,sensor1,ard_data,timestamp,price)
        cursor.execute(add_row, data)
```

```
            conn.commit()
            conn.close()
        except :
            print("MySQL insert failed")

#Send internet configuration for remote connection
def SendInformation(deviceID):
    url = 'http://enoptremoteservice.azurewebsites.net/api/device/connection'

    data = {}
    data["DeviceID"] = deviceID
    data["NetworkConfiguration"] = commands.getoutput("/sbin/ifconfig")

    #POST request
    print(str.format("Sending data {0}",data))
    r_internet = requests.post(url, json = data)
    print(r_internet.status_code)
    if(r_internet.status_code != 200):
        logger("Internet information post failed")
'''
```

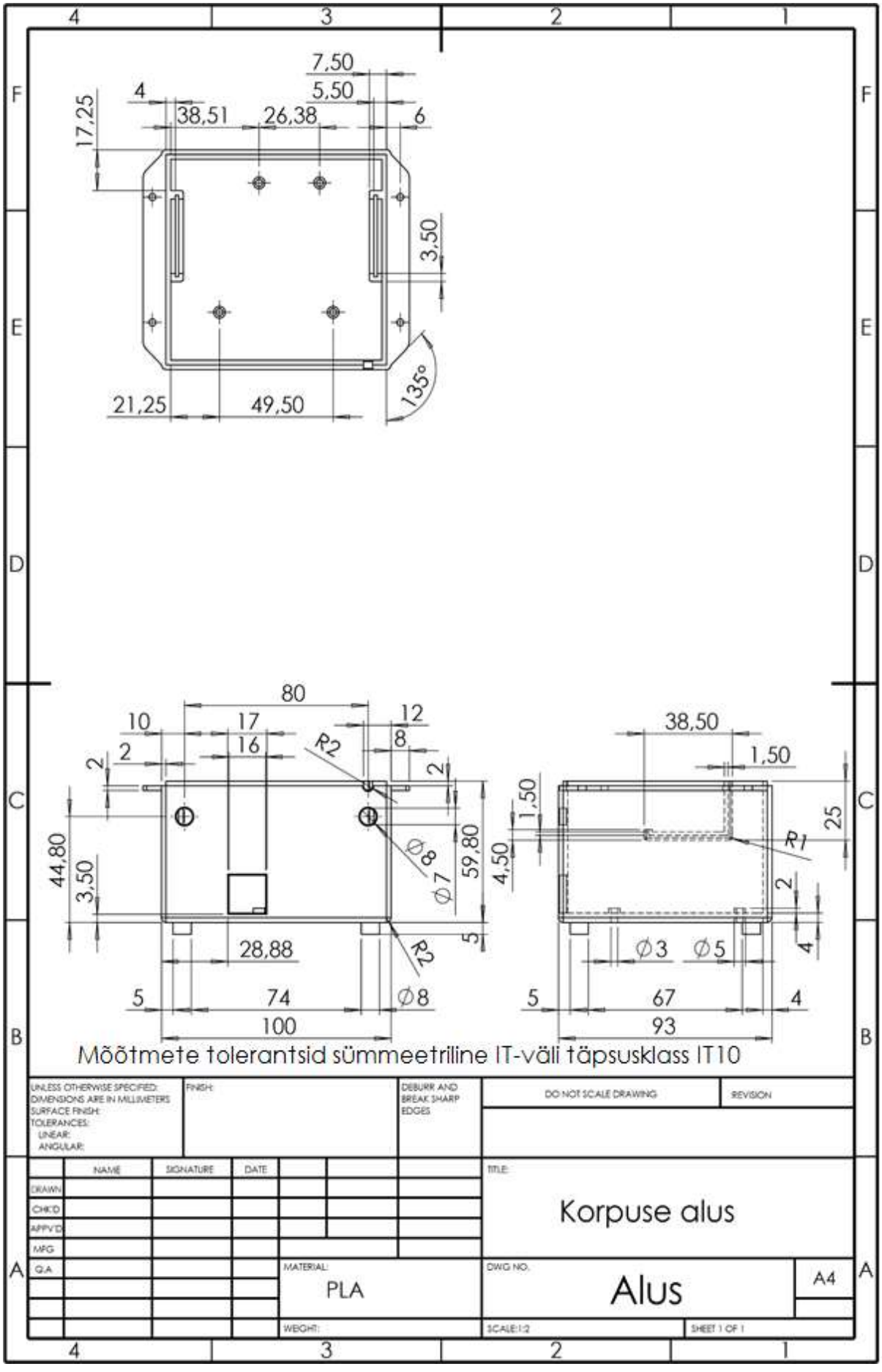
## Lisa 2 – Optimeerimispäringu tegemise skript

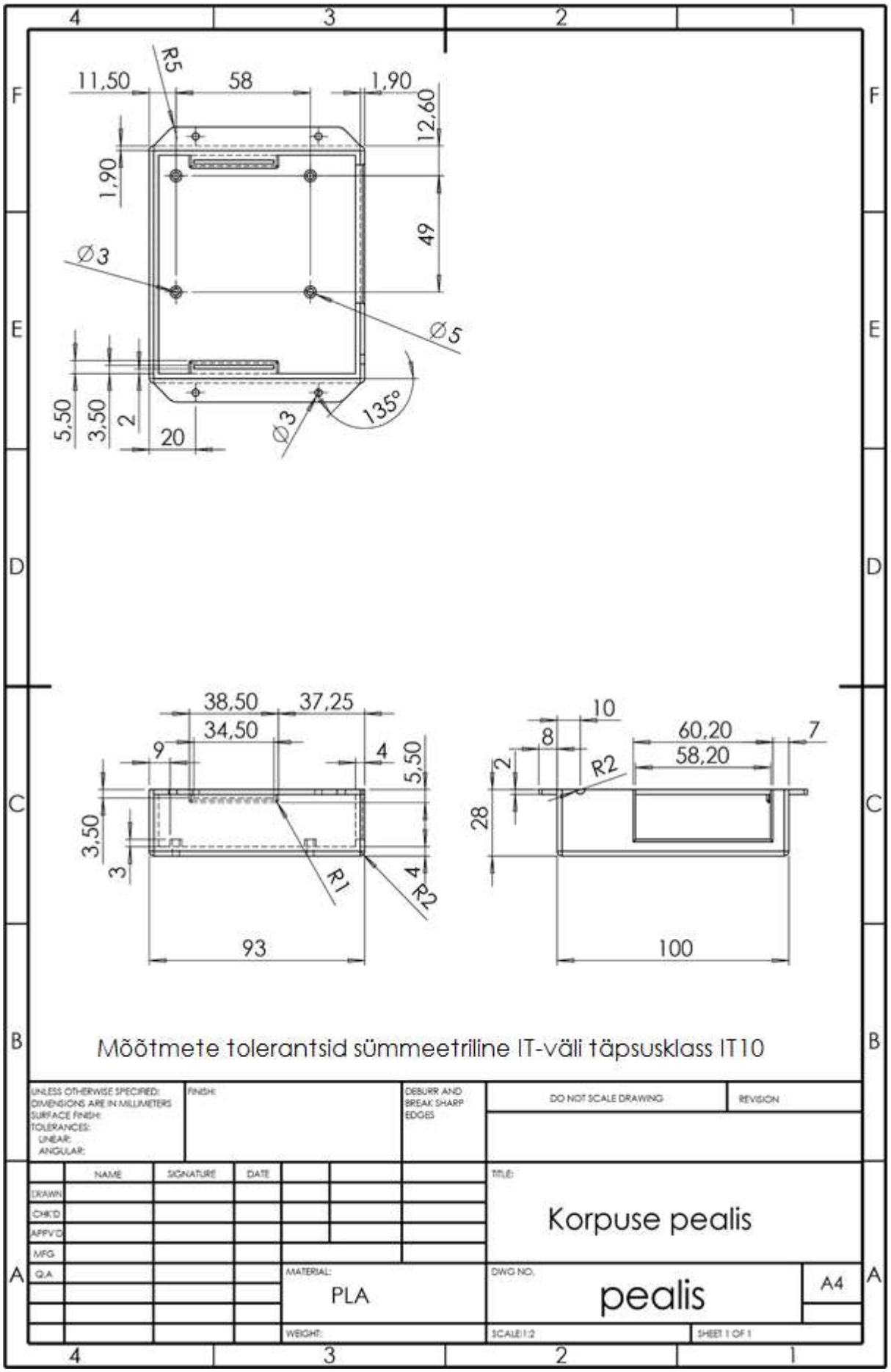
```
import ControllerFunctions as CF
import time

def Start(settings):
    logger = CF.logger
    settings = CF.GetJSON("/SLS/SLSPrototype/"+settings+".json")

    logger("Getting current temperature for optimization")
    InsideTemperature, OutsideTemperature = CF.GetTemperatures(settings["sensors"])
    logger(InsideTemperature)

    logger("Requesting new power schedule")
    response = CF.SLSOptimizationRequest(InsideTemperature, settings["minTemperature"],settings["maxTemperature"], settings["minPower"], settings["maxPower"], settings["deviceID"], OutsideTemperature = [0], StartUpCost = 0, ShutDownCost = 0, heatConductance = 0, heaterCoefficient = 0)

    powerForCurrentHour = response["power"][0]
    logger("Power for the current hour is " + str(powerForCurrentHour))
    minutes = CF.calculateMinutes(powerForCurrentHour, settings["maxPower"])

    #Switch according to current hour
    logger("Switching according to schedule")
    if(minutes < 60 and minutes > 0):
        CF.SwitchFromArduino(settings['PowerPIN'], 1)
        time.sleep(minutes * 60)
        CF.SwitchFromArduino(settings['PowerPIN'], 0)
    else:
        CF.SwitchFromArduino(settings['PowerPIN'], int(minutes != 0))
    return

Start('settings')
```

## Lisa 3 – Lokaalse kontrolli ja SLSi pilve andmete saatmise skript

```
import ControllerFunctions as CF
import json

def Start(settings):
    logger = CF.logger
    settings = CF.GetJSON("/SLS/SLSPrototype/"+settings+".json")
    logger(settings["deviceID"])
    logger("Getting temperatures from sensors")
    InsideTemperature, OutsideTemperature = CF.GetTemperatures(settings["sensors"])

    logger("Getting power from Arduino")
    Power = CF.GetPowerFromArduino(settings['PowerPORT'])

    logger(str.format("Sending data to SLS... IT: {0}, OT: {1}, P: {2}", InsideTemperature, OutsideTemperature, Power))
    CF.SLSSendData(InsideTemperature, OutsideTemperature, Power, settings["deviceID"], settings["authorization"])

    #Local control
    logger("Performing local control")
    flipSwitch, switchOn = CF.LimitationStatus(InsideTemperature, settings["minTemperature"], settings["maxTemperature"])
    if(flipSwitch):
        logger(str.format("Limit has been reached! T: {0}, MinT: {1}, MaxT: {2}, Switch:{3} ", InsideTemperature, settings["minTemperature"], settings["maxTemperature"],switchOn))
        CF.SwitchFromArduino(settings['PowerPIN'], switchOn)
    return

Start('settings')
```

## Lisa 4 – Korpuse joonised


7,50
5,50
4
38,51
26,38
6
17,25
3,50
135°
21,25
49,50
80
10
17
12
16
R2
8
2
2
2
44,80
3,50
59,80
Ø8
Ø7
5
28,88
R2
5
74
Ø8
100
38,50
1,50
1,50
25
R1
4,50
2
Ø3
Ø5
4
5
67
4
93
Mõõtmete tolerantsid sümmeetriline IT-väli täpsusklass IT10
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
CHK'D
APPV'D
MFG
Q.A
TITLE:
Korpuse alus
MATERIAL:
PLA
DWG NO.
Alus
A4
WEIGHT:
SCALE:1:2
SHEET 1 OF 1


11,50
58
1,90
12,60
1,90
49
R5
Ø3
Ø5
135°
5,50
3,50
2
20
Ø3
38,50
37,25
34,50
9
4
5,50
3,50
3
R1
R3
4
93
10
8
60,20
7
58,20
2
R2
28
100
Mõõtmete tolerantsid sümmeetriline IT-väli täpsusklass IT10
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
CHK'D
APPV'D
MFG
Q.A
TITLE:
Korpuse pealis
MATERIAL:
PLA
DWG NO.
pealis
A4
WEIGHT:
SCALE:1:2
SHEET 1 OF 1