TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Silver David Iling 179620IACB

VALGUSTI NUTIKODU KESKSEADME FUNKTSIOONIDES

Bakalaureusetöö

Juhendaja: Andres Rähni
MSc

Tallinn 2022

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Silver David Iling

03.01.2022

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks oli luua targa valgusti ja nutikodu keskseadme kombinatsioon. Tulemuseks on tehtud töötav prototüüp, keskseadmele on installeeritud openHAB'i kodu automaatika tarkvara süsteemi juhtimiseks ja monitoorimiseks. Keskseadmega suhtleb juhtmevaba sensoorika ning Wi-Fi baasil pistikupesa. Kasutajale on tehtud kasutajaliides kasutades openHAB'i tööriista HABPanel.

Seadme ümber on 3D modelleeritud ja prinditud korpus, kuhu on ehitatud tark lamp. 3D mudel on koostatud kasutades SolidWorks'i modelleerimistarkvara. Lamp on RGB võimekusega ja saadaval olevate värvivarjundite valiku suutlikkus on teostatud kasutades PWM'i.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 26 leheküljel, 6 peatükki ja 23 joonist.

Abstract

Lighting as a part of an integrated smart home system

The purpose for this Bachelor's thesis was to create a combination of a smart lamp and a smart home hub. As a result a working prototype has been built using a Raspberry Pi 3 Model B computer, a home automation software called openHAB has been installed onto the smart home system for controlling and monitoring the work of the system and a user interface has been implemented for the user to interact with using the user interface building tool in openHAB called HABPanel. The system is also being interacted with wireless sensors and a smart wall plug based on Wi-Fi communication.

There has also been a 3D model printed for the casing for the smart lamp and smart home system using SolidWorks solid modeling computer-aided-design software. The smart lamp is RGB capable and the availability of different colour shades outside the three main colours of the LED is done by using a method called pulse-width modulation, also known as PWM.

The thesis is in Estonian and contains 26 pages of text, 6 chapters and 23 figures.

Lühendite ja mõistete sõnastik

CLI	*Command-Line Interface,* käsurida.
Wi-Fi	Juhtmevaba võrk IEEE 802.11x standarditel.
RGB	*Red Green Blue* ehk liitvärvimudel, mis koosneb punasest, rohelisest ja sinisest värvist.
PWM	*Pulse-width modulation* ehk pulsilaiusmodulatsioon on meetod väljundpinge reguleerimiseks, kus muudetakse impulsside laiust.
DSL	*Domain-specific language* ehk domeenispetsiifiline keel on antud süsteemis kasutatav programmeerimiskeel spetsiifiliselt selles rakenduses.
USB	*Universal Serial Bus* ehk universaalne järjestiksiin on standard seadmete ühendamiseks arvutiga.
PIR sensor	*Passive infrared sensor* ehk liikumisandur
DC	*Direct current* ehk alalisvool
MOSFET	*Metal-oxide-semiconductor field-effect transistor* ehk isoleeritud paisuga väljatransistor
GPIO	*General-Purpose Input/Output* ehk üldotstarbeline sisend/väljund
LED	*Light-Emitting Diode* ehk valgusdiood
FDM	*Fused Deposition Modeling,* 3D printimise meetod
API	*Application Programming Interface*, rakendusliides
SSH	*Secure Shell,* turvakest

Sisukord

Jooniste loetelu

1 Sissejuhatus

Bakalaureuse lõputöö eesmärgiks oli luua töötav prototüüp nutikodu keskseadme ja targa lambi kombinatsiooni näol. Põhiline eesmärk oli luua süsteem, millega juhtida targa kodu seadmeid läbi keskkonna, mis on suuteline võtma vastu andmeid erinevatelt andmeside protokollidelt. Idee sellise kombinatsiooni jaoks oli nutikodu seadmete integreerimine ühtseks seadmeks vähendamaks vooluvõrku ühendatavate seadmete hulka.

Kodu automaatika tarkvaraks valiti vabavaraline openHAB, läbi mille on võimalik korraldada erinevate seadmete tööd ja töödelda andmeid vastavalt ühendatud seadmete võimekuse ja kokkusobivusele openHAB'i süsteemiga. Autor koostas targa lambi ehitades elektriskeemi, millega toidetakse nii nutikodu keskseadet kui ka lampi ennast.

Sellise toote valmistamiseks oli vaja leida alternatiiv keskseadme paigutuseks, seda siis turvalisust puudutavatel põhjustel. Turvalisust puudutavad põhjused hõlmavad endas koju sissemurdmist, kus olemasolevate liikumisandurite töö on kergesti peatatud, kui kõrvaldatakse neid juhtiv keskseade. Eesmärgiks on luua nutikodu keskseade, mis näeb välja nagu lamp ning seega ei reeda oma asukohta.

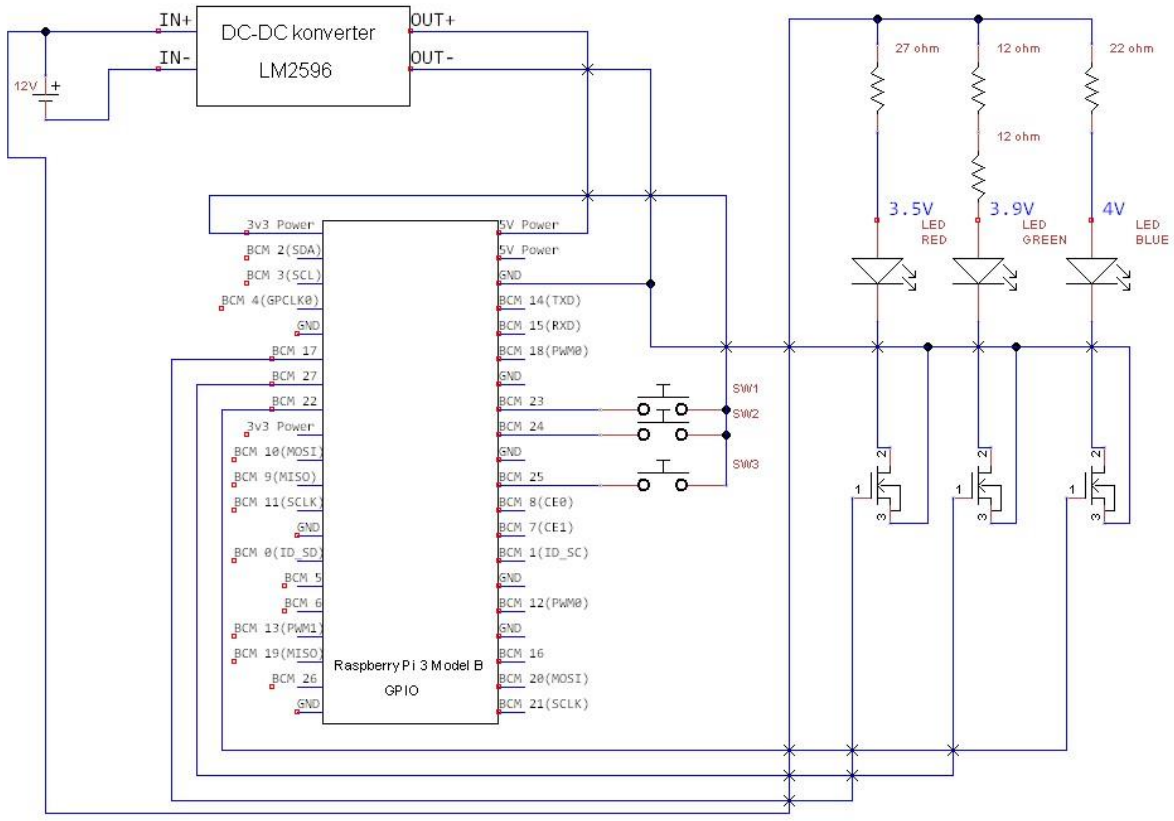
Nutikodu keskseadme kontrolleriks on võetud kasutusele Raspberry Pi 3 Model B, keskseade peab olema erinevate andmesideprotokollidega laiendatav, valgusti peab peitma keskseadet ning kasutama teatud andureid ja täitureid kontseptsiooni tõestuseks. Lisaks sellele peab keskseade pakkuma kasutajale suhtlemisviisi läbi kasutajaliidese ning teavitama erinevatest sündmustest. Lambil on võimalik sisse lülitada automaatvalgustusrežiim, mis vastavalt multisensori valguse intensiivsuse andmetele reguleerib lambi valgusviljakust.

Lisaks sellele suhtleb nutikodu keskseade nutipistikuga TP-LINK HS100, mida kasutab autor, et juhtida ruumikütteseadet. Nutipistikuga suheldakse kasutades Wi-Fi andmesidet näitamaks, et keskseade võimekus avaldub selles, et on võimalik ühendada erinevatel andmesideprotokollidel põhinevaid seadmeid ühte kesksesse süsteemi.

Lõputöö teises peatükis tutvustatakse seadme elektriskeemi, kolmandas peatükis lambi lülitamist, neljandas peatükis openHAB'i süsteemi ja tarkvaraarendust ning viiendas peatükis 3D modelleerimist.

2 Seadme elektriskeem

Elektriskeem (joonis 1) koosneb kolmest osast: lambi toitelahendus, Raspberry Pi toitelahendus ja lambi lülitusskeem. Lambi toitelahenduseks kasutas autor reguleeritava väljundpingega toiteadapterit vahemikus 3-12V ning toiteadapteri väljundvool on 2.25A, seega Raspberry Pi 3 Model B ning lambi toitmiseks on seda piisavalt.



Joonis 1. Keskseadme ja lambi elektriskeem, joonistatud TinyCAD tarkvaraga [1].

2.1 Lambi toitelahendus

Toiteadapteri väljundpinge on reguleeritud 12V peale täpsemaks voolu reguleerimiseks lambile, siin ei kasutata LM2596 pingeregulaatorit, kuna see on ainult kasutusel Raspberry Pi toitelahendusel seega lambi toide on 12V pingel. Lambi toide võetakse otse adapterist läbi arvutatud takistite. Takistite arvutus viidi läbi kasutades ülikoolis olevat toiteplokki. Toiteplokki kasutades reguleeriti LED'ile rakenduvat voolutugevust 350

milliamprini iga värvi kiibile. Voolutugevuse suurus ja muud atribuudid on võetud tootja poolt esitatud andmelehelt [2]. Toiteplokk kuvas LED'ile rakenduvat pinget ning kasutades Ohmi seadust leiti vajalik takistus, et 12V sisendpinge puhul rakenduks igale LED'ile 350 milliamprit. Takistite väärtused on seega punasele värvile 27Ω, rohelisele värvile 24Ω ja sinisele värvile 22Ω. LED'i erinevate kiipide päripinged on seega järgmised: punasel värvil 3.5V, rohelisel värvil 3.9V ja sinisel värvil 4V. Kuna toiteplokk kuvas pingeid ühe komakohaga, siis täpsemat väärtust ei olnud võimalik välja selgitada antud hetkel. See mängib aga väikest rolli, kuna takistite ebatäpsus tuleb arvesse võtta. Esialgses lahenduses oli LED'ide toide viidud 5V peale, kuid kuna takistite tolerants on 5%, siis sellise ebatäpsusega oli mõistlikum viia pinge 12V peale, et suuremate takistite väärtuse juures mängiks 5% tolerants väiksemat rolli.

2.1.1 Lambi takistite võimsus

LED'i iga kiibi eraldatav võimsus 350 milliampri juures on 1.4W, seega takistid on valitud võimsusega 2W. Paraku takistite saadavus ostmise ajal ei võimaldanud 2W takistite ostu, siis on takistiteks määratud hoopis 5W takistid. See ei ole probleem, kuna ainus oluline faktor takistite võimsuse hinnangu juures on see, et takistitel eralduv võimsus ei oleks hinnangust suurem.

2.1.2 Lambi jahutuslahendus

LED'i jahutuseks on tootja määranud soovituslikuks kasutada jahutuslahendust, et LED'i eluiga ei väheneks kõrge temperatuuri tõttu. Kasutusele on võetud Ohmite poolt toodetud alumiiniumist tähekujuline jahutusradiaator [3]. Radiaator on võimeline vähendama ühendatud valgusti temperatuuri 4.33 kraadi Celsiuse järgi iga vati kohta. Seega vähendatakse lambi soojust ca 18°C. Jahutusradiaator on lihtsuse mõttes monteeritud LED'iga kokku kasutades kiirliimi.

2.2 Raspberry Pi toitelahendus

Kuna kasutusel olev toiteadapter on reguleeritud 12V juurde, siis Raspberry Pi'le minevat sisendpinget on vaja oluliselt vähendada. Selle saavutamiseks on elektriskeemi lisatud Texas Instruments'i poolt toodetud DC-DC pingeregulaator LM2596. LM2596 (joonis 2) on pingeregulaator, mis võimaldab pinget vähendada, võimaldab vastu võtta

sisendpingeid kuni 40V ja suudab väljundvoolu rakendada kuni 3A [4]. Moodulit häälestatakse kruvikeerajaga keerates määratud kohas, mis reguleerib väljundpinget. Kasutades pingeregulaatorit on 12V pinge langetatud 5.2V peale, põhiliselt kuna seda pinget väljastab ka Raspberry Pi ametlik toiteadapter. Lisaks sellele on autor märganud kasutades Raspberry Pi 3 Model B toitmisel 5V Samsungi toiteadapteriga, et suure koormuse all esineb Raspberry Pi'l pingelang mis häirib selle tööd. Tavaliselt toidetakse Raspberry Pi'd läbi USB ühenduse, kuid lõputöö 3D mudeli kompaktse disaini tõttu ei mahu seda toitma läbi USB. Selle asemel on võimalik toidet suunata ka muul viisil, nimelt saab Raspberry Pi'd toita läbi 5V siini, sellisel juhul on oht kahjustada Raspberryt, kuna liiga suuure voolu tarbimise puhul ei ole kaitset, mis rakenduks. Antud tööks sobib see lahendusena, kuna Raspberry Pi ise ei ole üle koormatud ning seega puudub oht selle kahjustamiseks.

Joonis 2. LM2596 DC-DC pingeregulaator [5]

2.3 Lambi lülitusskeem

Lamp on RGB võimekusega ja seega on võimalik muutes kolme eraldi kiibi pingeid, et saavutada erinevaid värvivarjundeid. Autor lahendas selle kasutades pulsilaiusmodulatsiooni, et saavutada kontrolli lambi lülituse üle. Pulsilaiusmodulatsioon võimaldab reguleerida pinget muutes impulsi laiust. Laiuse muutus mõjutab töötsüklit perioodis. Pinge reguleerimine võimaldab esile tuua palju rohkem värvivarjundeid igal individuaalsel LED kiibil kui ka värvikiipide erinevate pingete kombinatsioonil. Kiireks lülituseks on võetud kasutusele kolm Infineon'i poolt toodetud IRLZ44NPBF MOSFET'i. Kuna kiire lülituse puhul tekkivad järsud tõusud voolus, mis kestavad väga lühikest aega, siis see võib mõjutada MOSFET'i temperatuuri kiirel lülitusel. Antud transistorid on aga maksimaalse töötemperatuuriga 175°C [6] ja kannatavad seetõttu võimaliku temperatuuritõusu ära.

3 Lambi valguse toonide ja valgustugevuse juhtimine

Targa lambi lülitamine toimub tarkvaraliselt kasutades Python'i teeki *pigpio* [7], läbi mille on riistvara taimereid kasutades tehtud võimalikuks PWM kasutamine. Olles sisestanud keskseadme vooluvõrku, käivitatakse pärast üldisi süsteemiteenuseid skript */etc/rc.local*, mis lubab süsteemi administraatoril käivitada skripte. Antud töös kasutatakse seda skripti, et käivitada *pigpiod* utiliit, mis käivitab *pigpio* teegi madala taseme taustprogrammina, et seda oleks võimalik peamises skriptis kasutada. Lisaks sellele käivitatakse süsteemi käivitamisel lambi lülitamiseks mõeldud skript.

3.1 GPIO initsialiseerimine

Lambi lülitamise teeb võimalikuks Raspberry Pi sisendite ja väljundite seadistamise võimalus. Konfigureerimisel kasutatakse Raspberry Pi *Broadcom pin* numbreid [8] füüsiliste sisendite-väljundite asukohtade asemel sellel põhjusel, et *pigpio* teek kasutab neid, et aru saada, millist sisendit-väljundit tahetakse konfigureerida. Väljunditeks on valitud *Broadcom pin* arvud 17 punasele, 22 sinisele ja 27 rohelisele. Lambi füüsilistele nuppudele on määratud sisendid 23, 24 ja 25.

3.1.1 PWM seadistamine

PWM kasutamine nõuab *pigpio* teegiga väljundite seadistamist. See hõlmab valimi moodustamist ja sageduse määramist. Sageduseks on valitud 8000Hz ja valimi suuruseks 256. Seda saavutatakse kasutades *pi.set_PWM_frequency(17, Freq)* ja *pi.set_PWM_range(17, 256)*, seda siis iga väljundi kohta.

3.2 Funktsioonid

Paremaks koodi struktureerimiseks loodi funktsioonid igale väljundile. Neid kasutatakse LED'i kiipide pinge reguleerimiseks. Funktsioonides rakendatakse tsüklilist väärtuste määramist, et muuta värvide üleminek sujuvamaks. Sujuvus on saavutatud, kuna programmil võtab tsükli itereerimiseks piisavalt kaua aega, et inimesele tunduks värvide

üleminek sujuv. On toodud ka vastassuunaline tsükkel, et väljendada lambi valgusviljakuse vähenemist. *Duty cycle* ehk töötsükkel on osa perioodist, kus on signaal aktiivne. Näiteks, signaal töötsükliga 40% tähendaks, et signal on aktiivne 40% ajast ja 60% inaktiivne. Aeg, mil signaal on aktiivne on määratud perioodi pikkusest. *Pigpio* teegis kasutatav rida töötsükli muutmiseks on *pi.set_PWM_dutycycle(RED, x)*, kus *RED* on antud värv, mille all on mõeldud *Broadcom pin* arvu ja *x* on soovitud töötsükli väärtus väljendatud arvuna 0-256. Allpool on toodud näide (joonis 3) ühest koostatud funktsioonist.

```
def setREDdc(cycle, old, c):
	REDnew = cycle
	if REDnew > old:
		for x in range (old, REDnew):
			pi.set_PWM_dutycycle(RED, x)
			for y in range(1, 4000):
				c = c + 1
	time.sleep(0.025)
	else:
		for x in reversed(range(REDnew, old)):
			pi.set_PWM_dutycycle(RED, x)
			for y in range(1, 4000):
				c = c + 1
		time.sleep(0.025)
	return REDnew
```

Joonis 3. Näide töötsükli funktsioonist.

3.3 Failist lugemine

OpenHAB'i süsteem kirjutab vastavalt kasutaja sisendile tekstifaili lambi töötsükli väärtuseid, vastavalt sellele töötab skript selliselt, et seda sama tekstifaili avatakse ning sellest loetakse sõne sisse ning sõne sisu jagatakse iga värvi jaoks eraldi ära kasutades Python'i *split()* funktsiooni [9]. Jaotatud sõne osad muudetakse täisarvudeks ning seejärel kutsutakse välja funktsioonid töötsükli muutmiseks. Allpool on välja toodud kood (joonis 4) failist lugemiseks.

```
try:
    while RUNNING:
        try:
            with open('/etc/openhab2/scripts/dutycycle.txt', 'r') as myFile:
                try:
                    dutycycle = myFile.read().split(" ")

                    newRedCycle = int(dutycycle[0])
                    newGreenCycle = int(dutycycle[1])
                    newBlueCycle = int(dutycycle[2].split()[0])
                    myFile.close()

                    REDold = setREDdc(newRedCycle, REDold, 0)
                    GREENold = setGREENdc(newGreenCycle, GREENold, 0)
                    BLUEold = setBLUEdc(newBlueCycle, BLUEold, 0)
                except ValueError:
                    print ("Value error, trying again")
        except OSError:
            print ("File is empty")
    time.sleep(t)

except KeyboardInterrupt:
    print ("Program stopped")
    RUNNING = False
```

Joonis 4. Failist lugemise kood.

4 openHAB’i süsteem

OpenHAB ehk *open Home Automation Bus* on avatud lähtekoodiga kodu automaatika tarkvara. OpenHAB võimaldab kasutajale ühendada kokku erinevaid openHAB’i toetavaid seadmeid. Sündmusi on võimalik esile kutsuda kasutades reegleid, heli käsklusi või käsklusi läbi openHAB’i kasutajaliidese. Antud töös on tarkvara implementatsioon läbi viidud lähtudes openHAB’i dokumentatsioonile [10]. OpenHAB sai võtta kasutusele tänu selle tarkvara stabiilsuse tõttu, tasub mainida, et alternatiivseid kodu automaatika tarkvara võimalusi on veel, näiteks vabavaraline Home Assistant, kuid openHAB antud kontekstis on sobilikum kuna soovitud valvesüsteemi funktsionaalsus on paremini realiseeritav tarkvaraga, millel ei tule nii tihti uuendusi ning mis uuendustejärgselt säilitab oma funktsionaalsuse suurema tõenäosusega. Antud lahenduses on openHAB’iga liidestatud töös kasutatud multisensor Fibaro FSGM-001, mis töötab Z-Wave andmesideprotokollil, TP-LINK HS100 nutipistik, mis töötab IEEE 802.11b/g/n protokollidel ning Raspberry Pi kaudu juhitav tark lamp. Lisaks sellele on openHAB’il suur valik erinevaid ühilduvaid seadmeid, mis võimaldavad kasutada täiendavalt erinevaid andmesideprotokolle [11].

4.1 openHAB’is orienteerumine

OpenHAB’i tarkvaras on erinevad kontseptsioonid, kuidas esitada seadmete kooskõlastamist tarkvaraga. Need kontseptioonid on alustaladeks, kuidas kasutada openHAB’i tarkvara [12].

4.1.1 *Thing*

Thing tüüpi atribuut on üksus, mida saab füüsiliselt ühendada süsteemi. Sellist tüüpi üksused ei pea alati olema füüsilised esemed, vaid *Thing* tüüp võib esindada ka veebiteenust või muud haldavat infoallikat. *Thing* avalikustab oma võimekuse läbi *Channel*’ite.

4.1.2 *Binding*

Binding tüüpi atribuudist saab mõelda kui tarkvara adapterist, mis võimaldab *Thing*’i teha saadaval olevaks kodu automaatikasüsteemile. See on lisandmoodul, mis võimaldab linkida *Item* tüüpi füüsiliste seadetega.

4.1.3 *Item*

Item tüüp esindab võimekusi, mida rakendused saavad kasutada. Sellist tüüpi üksuseid saab ära kasutada kas kasutajaliidestes või automaatikaloogikas. *Item* tüüpi üksustel on atribuut *State* ehk olek, millele on võimalik käsklusi saata.

4.2 openHAB'i failid

Kõik failid, mida redigeeritakse on sellise faililaiendiga, millist tüüpi see fail haldab. OpenHAB'is programmeeritakse skripte DSL ehk domeenispetsiifilises keeles. Kasutusel olev keel põhineb Xbase'l ning seetõttu on see sarnane ja jagab mitmeid aspekte Xtend keelega, mille juured on Java programmeerimiskeeles, kuid keskendub rohkem sisutihedale süntaksile. Selles peatükis kirjeldatakse failie, mis on loodud süsteemi töös kasutamiseks.

4.2.1 Tüübifail *projekt.items*

Antud failis on deklareeritud kõik andmetüübid, mis on süsteemis kasutusel. Siia kuuluvad multisensori tüübid, nutipistiku tüüp kui ka kõik lambiga seotud muutujad. Selles failis välja toodud tüübid ja muutujad võetakse kasutusele kasutajaliideses ja *.rules* failides. Deklareeritud andmetüübid on *Switch*, mis on lüliti olekutega *ON* ja *OFF*. Järgmisena *Number*, millele saab numbrilist väärtust lisada, *String*, millele saab sõne lisada ja *Contact*, millele ei saa väärtuseid lisada, vaid saab ainult staatust lugeda. *Contact* tüübid on antud töös seotud Raspberry Pi sisend/väljunditega, mis juhivad lambi lülitust MOSFET'e kasutades. Lisaks sellele on ka töös kasutusel fail *projekt.things*, mis koosneb kõigest ühest reast ning selles luuakse võimekus käivitada käsku läbi CLI.

4.2.2 Reeglite fail *default.rules*

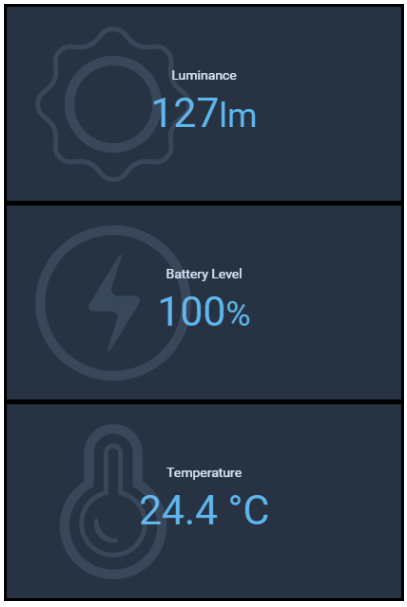
Rules failis käsitletakse kõiki reegleid openHAB'is. Sellises failis saab skriptida sündmusi vastavalt seadete värskendustele ja lülitustele kui ka kasutajaliidese elementidega suheldes. Failis *default.rules* on läbi viidud kõik targa lambi juhtimine ja sensoorika ja nutipistiku kontroll. Siin on ka ära määratud, mis juhtub, kui lambi peal olevaid füüsilisi surunuppe lülitatakse.

4.2.3 Telegram'i fail *telegram.rules*

Siin käsitletakse kõik sündmused, mis on seotud kasutaja teavitamisega läbi Telegram rakenduse. Kasutajale saadetakse teavitusi sensori häiringutest ja liikumistuvastusest, kui valverežiim on sisselülitatud. Lisaks sellele teavitatakse üldisemate keskkonnaliste andmemuutuste kohta nagu näiteks temperatuur, valguse intensiivsus ja sensori akutase. Samuti saadetakse infot valverežiimi sisse- ja väljalülitamise kohta.

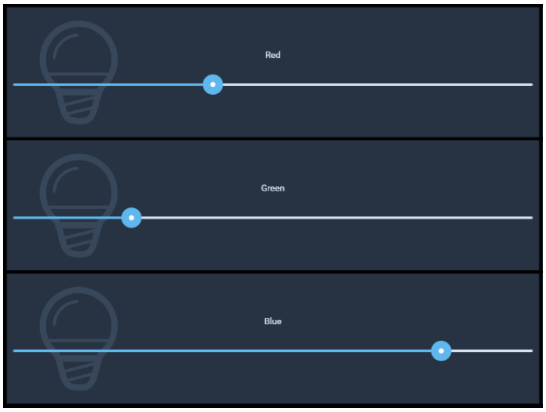
4.3 Kasutajaliides

Läbi kasutajaliidese toimub põhiliselt sensoorika andmete kuvamine ja lambi juhtimine. Kasutajaliides on loodud kasutatud openHAB'i tööriista HABPanel ning on esitatud inglise keeles. Kasutajaliidese peab seadistama kasutades HABPanel'it igal seadme tüübil eraldi. Samas võrgus ühendatud erinevad seadmed ei kuva sama kasutajaliidest universaalselt. Seega on kasutajaliides loodud eraldi nii telefonile kui ka arvutile. Kuvatõmmiste näidistena tuuakse esile arvutis olev kasutajaliides. HABPanel'is saab valida rakenduse poolt ära määratud nii-öelda vidinad. Nende disain on arendajate poolt kindlaks määratud ning kasutajale on mõned kohandamisvõimalused antud. Kasutajale kuvatakse infot multisensori kohta, nimelt kuvatakse kasutajale valguse intensiivsust, multisensori akutaset ja temperatuuri (joonis 5).



Joonis 5. Kasutajaliidese kuvatõmmis kasutajale kuvatud multisensori andmetest.

Lisaks sellele on kasutajal võimalik muuta lambi eredust ning seeläbi kombineerida erinevaid värvuseid läbi kolme rullikutüüpi elemendi. Elementidel saab hiirega liigutades või sõrmega viibates muuta liuguri kursori hetkeasukohta ning seeläbi manipuleerida lambi tööd. Allpool on välja toodud kuvatõmmis nendest elementidest (joonis 6).



Joonis 6. Kuvatõmmis kasutajale kuvatud lambi erinevate värvide muutmise elementidest.

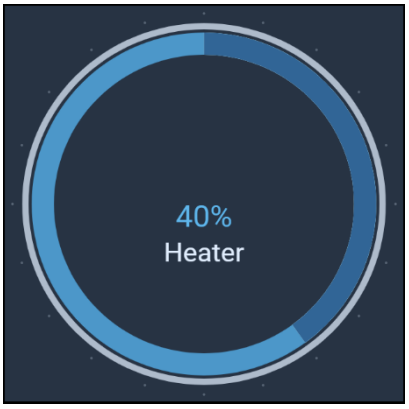
Kasutajaliidesel on veel võimalik lülitada sisse ja välja automaatvalgustusrežiimi (joonis 7).

Joonis 7. Kuvatõmmis kasutajale kuvatud automaatvalgustusrežiimi lüliti.

Lisaks sellele on läbi kasutajaliidese võimalik lülitada sisse ja välja valverežiimi (joonis 8).

Joonis 8. Kuvatõmmis kasutajale kuvatud valverežiimi lülitist.

Viimaseks seni realiseeritud funktsionaalsuseks on kasutajal võimalik määrata autoril olemasoleva 2000W ruumikütteseadme töötsükkel protsentides, kasutaja saab elemendil viibata pööramist või siis hiirega ringiratast liigutada (joonis 9).



Joonis 9. Kuvatõmmis kasutajale kuvatud ruumikütteseadme juhtimise elemendist.

4.4 Z-Wave ja multisensori integreerimine

Antud töös on kasutusele võetud Z-Wave andmesideks Aeotec Z-Stick Gen5 USB kontroller. Kontroller ühendub Raspberry Pi külge ning on seotud Fibaro FGMS-001 multisensoriga kasutajale info edastamiseks ja valverežiimi ning automaatvalgustusrežiimi funktsionaalsuse realiseerimiseks. Z-Wave USB kontrolleriga seadme ühildumisel on lähtutud Aeotec'i kontrolleri kasutajajuhendist [13]. OpenHAB pakub Z-Wave *Bindingut* [14], mille kaudu saab Z-Wave seadmeid integreerida openHAB'iga. Z-Wave kontroller tuleb manuaalselt ühildada süsteemiga ning konfigureerida openHAB'i jaoks, millisesse porti USB kontroller on ühendatud.

4.5 Tarkvaraarendus

OpenHAB'i *default.rules* failis toimub põhiline tarkvaraarendus. Nagu eelnevalt mainitud käsitletakse selles failis sündmusi olenevalt, kuidas süsteemiga suheldakse. Läbi openHAB'i on realiseeritud järgmised funktsionaalsused: lambi ereduse muutmine iga värviga ehk värvuste kombineerimine, lambile automaatvalgustusrežiim, et vastavalt valgustatusele muuta lambi eredust, ruumikütteseadme juhtimine, liikumissensorit kasutades valverežiim ja kasutajale märku andmine, kui viiakse lamp sisse ja välja

valverežiimist, samuti kui tuvastatakse liikumist või kui multisensorit füüsiliselt häiritakse. Füüsiliste nuppudega saab lülitada lampi sisse ja välja, rakendada valverežiim ja rakendada automaatvalgustusrežiim.

4.5.1 Süsteemi algväärtused

Süsteemi käivitamisel on vajalik erinevate väärtuste seadistamine (joonis 10). Nende alla kuuluvad kõik lambi väärtused, vastasel juhul on lambil tühjad väärtused ning süsteem ei tööta korrektselt. Lisaks sellele lülitatakse automaatvalgustusrežiim välja, kuid valverežiim on vaikimisi sisselülitatud, selle kaalutlusel, et kui toitepinge kaob ning seade taastub, siis on võimalik liikumisanduriga liikumist siiski tuvastada, kui enne toitepinge kadumist oli see režiim sisselülitatud .

```
rule "Set Default values"
when
    System started
then
    TamperSwitch.postUpdate(ON)
    AutomatedLighting.postUpdate(OFF)
    DutyCycleBlue.postUpdate("25")
    DutyCycleGreen.postUpdate("25")
    DutyCycleRed.postUpdate("25")

    executeCommandLine("/bin/sh@@-
c@@echo " + DutyCycleRed.state + " " + DutyCycleGreen.state + " " + DutyCycle
Blue.state +" > /etc/openhab2/scripts/dutycycle.txt; echo $?", 100)
end
```

Joonis 10. Süsteemi algväärtustamine.

4.5.2 Tarkvaraline lambi lülitamine

Lambi ereduse reguleerimine ja värvuste kombineerimine käib läbi kasutajaliidese kolme elemendi. Allpool on välja toodud kood (joonis 11), mis võimaldab läbi kasutajaliidese lambi eredust ja värvust muuta. Reeglid töötavad sündmuste alusel, ehk kood viiakse läbi siis, kui antud sündmus on juhtunud. *Dimmer* reeglis täidetakse kood, kui kasutajaliidesega suhtlemisest on toimunud muutus ehk *received update* numbrilisel andmetüübil. Antud reeglis on võetud kasutusele *Exec Binding*, mis võimaldab Raspberry Pi käsureal käsklusi läbi viia läbi openHAB'i süsteemi. Läbi kasutajaliidese määrab kasutaja väärtuse ning süsteem kirjutab selle faili. Monitoorimiseks kirjutatakse lambi värvuste väärtused openHAB'i poolt pakutavasse logivaatesse.

```
rule "Dimmer"
when
    Item DutyCycleRed received update or Item DutyCycleGreen received update
or Item DutyCycleBlue received update
then
    if(AutomatedLighting.state == OFF) {
        executeCommandLine("/bin/sh@@-
c@@echo " + DutyCycleRed.state + " " + DutyCycleGreen.state + " " + DutyCycle
Blue.state +" > /etc/openhab2/scripts/dutycycle.txt; echo $?", 100)
        logInfo("default.rules", "Values are: " + DutyCycleRed.state + " " +
DutyCycleGreen.state + " " + DutyCycleBlue.state)
    }
end
```

Joonis 11. Lambi valgusviljakuse muutmise kood.

4.5.3 Lambi lülitamine kasutades surunuppu

Lambi korpusele on paigutatud kolm nuppu, üks neist on mõeldud lambi sisse- ja väljalülitamiseks. See realiseeritakse kasutades reeglit *Toggle lamp with button* mis käivitatakse, kui süsteem tuvastab signaali surunupult. Surunuppudele on määratud 20ms pikk *debounce* periood ehk periood, kus nupuvajutusel tekivad esile palju ebakindlaid lugemeid kui nupuvajutus kestab, kui nupp on lõpuni alla surutud, siis tuvastatakse õige väärtus. Lambile rakendatavad väärtused ei ole nupuga sisselülitades maksimaalsed, seda sellel põhjusel, et kuna LED'i iga kiip on erinev, siis sama voolutugevuse juures annavad värvid erinevat eredust, seega on silma järgi võetud väärtused, mis esitaksid valget värvust. Allpool on toodud kood (joonis 12), mis näitab, kuidas on programmis realiseeritud lambi sisse ja välja lülitamine, kontrollitakse kas üks LED'i kiipidest on välja lülitatud, pärast mida lülitatakse valge värvuse peale ning kui ükski LED'i kiipidest pole välja lülitatud, siis lülitatakse lamp välja. Lisades 4 ja 5 on välja toodud ülejäänud nuppude funktsionaalsuse kood.

```
rule "Toggle lamp with button"
when
    Item OnOffButton changed from CLOSED to OPEN
then
    if((DutyCycleRed.state.toString == "10") || (DutyCycleGreen.state.
toString == "10") || (DutyCycleBlue.state.toString == "10")) {
        DutyCycleRed.postUpdate("175")
        Thread::sleep(1000)
        DutyCycleGreen.postUpdate("210")
        Thread::sleep(1000)
        DutyCycleBlue.postUpdate("135")
        Thread::sleep(1000)

    }
    else{
        DutyCycleRed.postUpdate("10")
        Thread::sleep(1000)
        DutyCycleGreen.postUpdate("10")
        Thread::sleep(1000)
        DutyCycleBlue.postUpdate("10")
        Thread::sleep(1000)
    }
end
```

Joonis 12. Lambi sisse- ja väljalülitamine kasutades surunuppu.

4.5.4 Automaatvalgustusrežiim

Automaatvalgustusrežiim hõlmab endas automaatset lambi ereduse muutmist vastavalt valguse intensiivsuse andurile. Kuigi sensor registreerib ca 15000lm kui sensor suunata otse päikesesse, siis selle väärtuse määramine maksimumiks pole mõistlik arvestades multisensori teist funktsionaalsust valverežiimi näol. Olles märganud, et aknast välja suunatud sensor, kuhu ei paista otsest päikesevalgust ei registreeri üle 4000lm, sai määratud see väärtus maksimumiks. Maksimumiks siinkohal mõeldakse skaleeritava suuruse maksimumi, mis võetakse arvesse lambi ereduse arvutamisel. Ereduse arvutamine toimub järgmiselt: salvestatakse muutujatesse lambi miinimum- ja maksimumväärtused ja sensori poolt mõõdetav maksimum, registreeritakse valguse intensiivsuse sensori ja lambi LED kiipide hetkeväärtused ning teisendatakse need ujukomaarvudeks. Järgmiselt skaleeritakse saadud valguse intensiivsuse sensori andmed selliselt, et tulemus oleks määratav lambile ning seejärel ümardatakse arv lähima täisarvuni ning teisendatakse sõne kujule ja määratakse lambile saadud väärtus. Järgnevalt tuuakse välja automaatvalgustusrežiimi kood (joonis 13).

```
rule "Automated Lighting"
when
    Item ZWaveNode009FGMS001MotionSensorSensorLuminance received update
then
    if(AutomatedLighting.state == ON) {
        var lampMin = 10.0
        var lampMax = 250.0
        var sensorMax = 4000.0
        var sensorCurrent = ZWaveNode009FGMS001MotionSensorSensorLuminance
        var redOld = Float::parseFloat(DutyCycleRed.state.toString)
        var greenOld = Float::parseFloat(DutyCycleGreen.state.toString)
        var blueOld = Float::parseFloat(DutyCycleBlue.state.toString)
        logInfo("default.rules",     "toString     values     are:     "     +
DutyCycleRed.state.toString  +  "  "  +  DutyCycleGreen.state.toString  +  "  "  +
DutyCycleBlue.state.toString)
        var sensorCurrentFloat = Float::parseFloat(sensorCurrent.state.toStri
ng)
        logInfo("default.rules", "sensorCurrentFloat: " + sensorCurrentFloat)
        if(sensorCurrentFloat > 4000.0) {
             sensorCurrentFloat = 4000.0
        }
        logInfo("default.rules", "sensor state is: " + sensorCurrent.state +
" old colours: " + redOld + " " + greenOld + " " + blueOld)
        var sensorScaling = (sensorCurrentFloat / sensorMax)  * (lampMax - la
mpMin) + lampMin
        var lampNew = Math::round(250 - sensorScaling)
        var lampNewStr = lampNew.toString
        // 250 for thread sleep gives enough time to apply changes to each LED
        DutyCycleRed.postUpdate(lampNewStr)
        Thread::sleep(250)
        DutyCycleGreen.postUpdate(lampNewStr)
        Thread::sleep(250)
        DutyCycleBlue.postUpdate(lampNewStr)
        Thread::sleep(250)
        executeCommandLine("/bin/sh@@-
c@@echo " + DutyCycleRed.state + " " + DutyCycleGreen.state + " " + DutyCycle
Blue.state +" > /etc/openhab2/scripts/dutycycle.txt; echo $?", 100)
        logInfo("default.rules", "sensor state is: " + sensorCurrent.state +
" new colours: " + DutyCycleRed.state + " " + DutyCycleRed.state + " " + Duty
CycleRed.state)
    }
    else {
             logInfo("default.rules", "Doing nothing, automated light switch
not turned on.")
    }
end
```

Joonis 13. Automaatvalgustusrežiimi kood.

4.5.5 Ruumikütteseadme juhtimine

Töös kasutusel olev TP-LINK nutipistik suhtleb üle Wi-Fi võrgu keskseadmega. Antud TP-LINK HS100 nutipistik suhtleb keskseadmega kasutades IEEE 802.11b/g/n protokolle. Nutipistiku kõige olulisem muudetav suurus on lüliti tüüp, kuna see võimaldab meil seadme põhilist funktsionaalsust juhtida. Sellist seadet saab kasutada paljude kodutehnika seadmete energiakulu vähendamiseks lülitades ühenduse välja ning seeläbi säästes energiat, mis kuluks vastasel juhul kõigest ooterežiimil olemisele. Antud töös saavutatakse sama funktsionaalsus, mis on olemas olemasoleval ruumikütteseadmel. Seade töötab selliselt: seadmel on pööratav nupp, mis määrab intervalli, kui tihti kütteseade end sisse ja välja lülitab. Kasutades Wi-Fi suhtlusega seinakontakti on seda võimalik läbi keskseadme juhtida. Nimelt on kasutajal võimalik määrata kümne minutilises vahemikus kui kaua veedab aega kütmisele seade olles sisse lülitatud. Kui aeg saab täis, siis lülitatakse ülejäänud ajaks välja kuniks saab kümme minutit täis, seejärel korratakse tegevust seni, kuni kasutaja muudab seadme töötsüklit. Programmis on see realiseeritud nii, et kasutajaliideses oleva numbrilise elemendi uuendusel käivitatakse kood. Reeglis viiakse läbi vajalikud teisendused ning saadetakse seinakontaktile vajalikud signaalid korrektseks tööks. Implementatsioonis oli vaja kasutusele tuua kaks reeglit, et kasutajaliidese elemendi muutusel lõpetatakse eelmise seadistuse töö. Allpool on toodud ruumikütteseadme juhtimise kood (joonis 14).

```
rule "Heater1"
when
    Item HeaterLevel received update
then
    var HeaterLevelFormat = Integer::parseInt(HeaterLevel.state.toString)
    var onTimer = HeaterLevelFormat * 6000
    var offTimer = (100 - HeaterLevelFormat) * 6000
    logInfo("default.rules", "States are: " + onTimer + " " + offTimer + " "
+ HeaterLevelFormat + " " + tplinksmarthome_hs100_866124_switch)

    if(HeaterLevelFormat == 0){
        tplinksmarthome_hs100_866124_switch.sendCommand(OFF)
        logInfo("default.rules", "Heater turned off. " + tplinksmarthome_hs10
0_866124_switch)
    }
    else if(HeaterLevelFormat == 100){
        tplinksmarthome_hs100_866124_switch.sendCommand(ON)
        logInfo("default.rules", "Heater turned on. " + tplinksmarthome_hs100
_866124_switch)
    }
    else{
        exitFlag = false
        while(!exitFlag){
            tplinksmarthome_hs100_866124_switch.sendCommand(ON)
            Thread::sleep(onTimer)
            tplinksmarthome_hs100_866124_switch.sendCommand(OFF)
            Thread::sleep(offTimer)
        }
    }
end
```

Joonis 14. Ruumikütteseadme juhtimise kood.

4.5.6 Valverežiim

Valverežiimi saab lülitada nii surunupuga kui ka läbi kasutajaliidese. Mõlemal juhul on sündmus seotud ühe lüliti tüübiga openHAB'is. Kui valverežiim lülitatakse sisse, siis vilgutatakse kasutajale punast, kollast ja rohelist värvi. Kui valverežiim lülitatakse välja, siis vilgutatakse kasutajale rohelist värvi. Sellisel juhul on kasutajale pakutud kinnitust sooritatud tegevusest. Liikumise tuvastamisel lülitatakse lüliti tüüpi muutuja sisse multisensori PIR sensori poolt ning jääb sisse lülitatuks, niikaua kuni antud liikumine, mis tuvastati on lõppenud. Valverežiim tuvastab liikumist ning liikumise tuvastamisel vilgutab valgusti punast värvust, lisaks sellele saadetakse sellest sõnum kasutajale. Kui valverežiimil füüsiliselt liigutatakse multisensorit, siis vilgutatakse punast, lillat ja sinist värvust ning ka selle kohta saadetakse kasutajale sõnum Telegram'i. Valverežiimi kasutamise koodilõiked on välja toodud lisades 1, 2 ja 3.

4.5.7 Rakenduse Telegram kasutamine

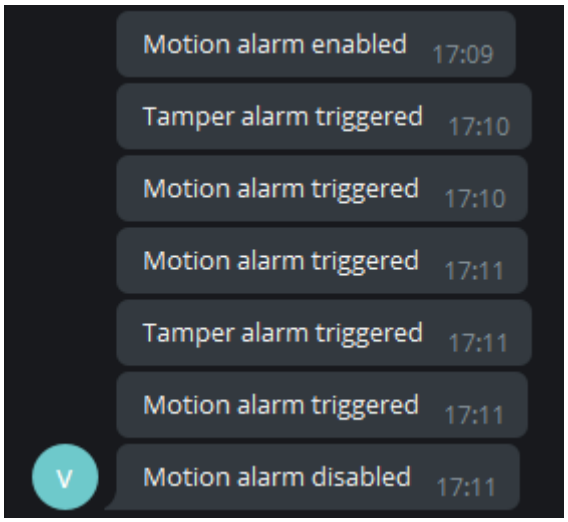
Olulisematest sündmustest antakse kasutajale märku läbi kiirsuhtlusrakenduse Telegram [15]. Teavitatakse järgmistest sündmustest: liikumise tuvastus valverežiimil, sensori häirimine valverežiimil, multisensori akutaseme muutus, valverežiimi sisse- ja väljalülitamisest, temperatuuri ja valguse intensiivsuse muutumisel. Telegram rakenduse jaoks on tehtud openHAB'i keskkonnas *Binding*, mis võimaldab läbi *sendTelegram()* funktsiooni koodis saata kasutajale tekstisõnumeid. Sõnumite edastamine käib läbi veebiroboti kasutades Telegram API'd. Robotil on oma vestluse ID ja juurdepääsuluba, mille kaudu tehakse päringuid, et teateid edastada. Sõnumeid saadetakse kasutajale olenemata sellest, kas vastuvõttev seade on samasse kohtvõrku ühendatud, kui keskseade. Seega on võimalik kasutajat teavitada liikumisanduri tegevusest isegi siis, kui seadme omanik on kodust eemal. Allpool tuuakse paar näidet koodist Telegram rakendusega suhtlemise arendamisest (joonised 15 ja 16) ning näide kasutajale saadetud Telegram sõnumist (joonis 17).

```
rule "Notify on movement"
when
    Item ZWaveNode009FGMS001MotionSensorAlarmMotion received update
then
    if (TamperSwitch.state == ON) {
        // bot1 is the Telegram bot to use, as per the configuration
        sendTelegram("bot1", "Motion alarm triggered")
    }
end
```

Joonis 15. Koodinäide liikumise tuvastuse teavitamisest.

```
rule "Notify on Motion enable/disable"
when
    Item TamperSwitch received update
then
    if (TamperSwitch.state == ON) {
        sendTelegram("bot1", "Motion alarm enabled")
    }
    else{
        sendTelegram("bot1", "Motion alarm disabled")
    }
end
```

Joonis 16. Koodinäide valverežiimi lülitamise teavitamisest.



Joonis 17. Näide Telegram'i teavitustest.

4.5.8 Logimine

Süsteemi töö haldamiseks logiti openHAB'i poolt pakutavat logivaadet erinevaid sündmusi, et näha, kuidas süsteem käitub erinevates olukordades. See tuli eriti kasulikuks probleemide tuvastamisel. Logivaates saab kuvada iga asja olekut ning iga muutuja väärtust. Lisaks koodis määratud logiinfole väljastab süsteem ise infot koodis esinevate tõrgete ja muude sündmuste kohta. Allpool tuuakse esile näide logist, kus salvestati lambi värvuste muutmine (joonis 18).

```
2020-05-13 00:33:21.143 [INFO ] [smarthome.model.script.default.rules] - Values are: 10 10 10

2020-05-13 00:38:39.036 [ome.event.ItemCommandEvent] - Item 'DutyCycleRed' received command 165
2020-05-13 00:38:39.051 [nt.ItemStatePredictedEvent] - DutyCycleRed predicted to become 165
2020-05-13 00:38:39.074 [vent.ItemStateChangedEvent] - DutyCycleRed changed from 10 to 165

2020-05-13 00:38:39.172 [INFO ] [smarthome.model.script.default.rules] - Values are: 165 10 10

2020-05-13 00:38:39.554 [ome.event.ItemCommandEvent] - Item 'DutyCycleGreen' received command 171
2020-05-13 00:38:39.562 [nt.ItemStatePredictedEvent] - DutyCycleGreen predicted to become 171
2020-05-13 00:38:39.573 [vent.ItemStateChangedEvent] - DutyCycleGreen changed from 10 to 171

2020-05-13 00:38:39.688 [INFO ] [smarthome.model.script.default.rules] - Values are: 165 171 10

2020-05-13 00:38:40.194 [ome.event.ItemCommandEvent] - Item 'DutyCycleBlue' received command 195
2020-05-13 00:38:40.212 [nt.ItemStatePredictedEvent] - DutyCycleBlue predicted to become 195
2020-05-13 00:38:40.249 [vent.ItemStateChangedEvent] - DutyCycleBlue changed from 10 to 195

2020-05-13 00:38:40.338 [INFO ] [smarthome.model.script.default.rules] - Values are: 165 171 195
```

Joonis 18. Kuvatõmmis logivaatest lambi ereduse muutmisel.

4.6 Nutikodu keskseadme küberturvalisus

Võttes arvesse nutikodu keskseadme valverežiimi funktsionaalsust tuli veenduda, et süsteem ei oleks küberrünnakutele haavatav. Kuna openHAB installeerimine on läbi viidud openHABian distributsiooni installeerimisega, mis on sisuliselt Linux Debian distributsioon, siis kasutades küberturvalisuse poolelt teadlikke meetodeid nii kohaliku võrgu poolelt kui ka välisvõrgu poolelt on turvalisus tagatud. Lähtutud on openHAB turvalisuse dokumentatsioonist [16]

4.6.1 Raspberry Pi haldamine

Süsteemi töö haldus Raspberry Pi peal on läbi viidud kaughaldusega kasutades asümmeetrilise krüptograafia avalik-salajasi SSH võtmeid [17]. Avalik võti on jagatud Raspberry Pi'le ning salajast võtit ei jagata, selle kaudu on võimalik turvaline paroolita kaugligipääs mis tähendab, et parooli ära arvamine ei ole võimalik ründevektor süsteemile.

4.6.2 Võrgu turve

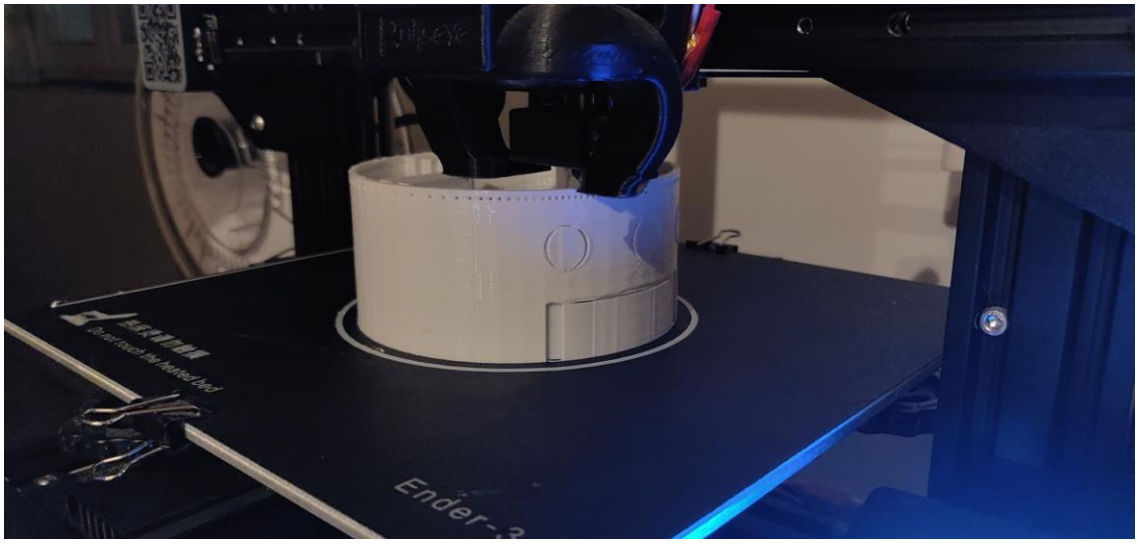
Kuna openHAB kasutab NGINX veebiserverit ning see ei ole avaliku domeeni ühendatud, siis veebi teel ligipääs openHAB kasutajaliidesele on ainult võimalik läbi kohaliku võrgu. Kohaliku võrgu turvalisuse tagab ruuter mis on parooliga turvatud kui ka Raspberry Pi enda tulemüür, kus saab filtreerida sissetulevat võrguliiklust ning lubada ainult openHAB'i poolne võrguliiklus teadete saatmiseks ja SSH port kaughalduseks.

4.6.3 Telegram'i veebirobot

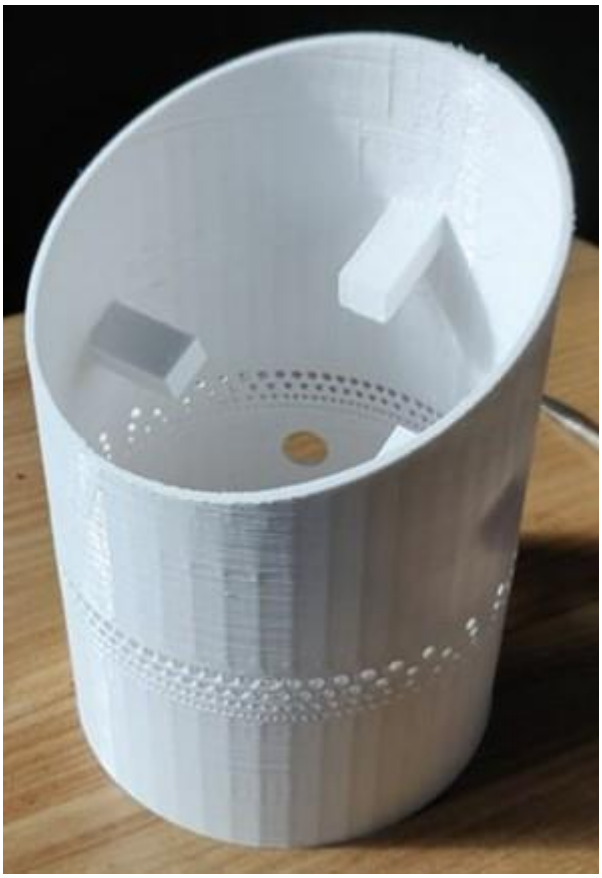
Telegram'i veebirobotit kasutatakse Telegram API kaudu, mille läbi teostab robot kasutajale teavituste saatmist, robotil on määratud vaid sõnumite saatmisfunktsionaalsus ning sõnumite lugemisõigust robotil ei ole, seega ei saa robotile sõnumite edastamisega haavata openHAB'i süsteemi.

5 Seadme 3D modelleerimine

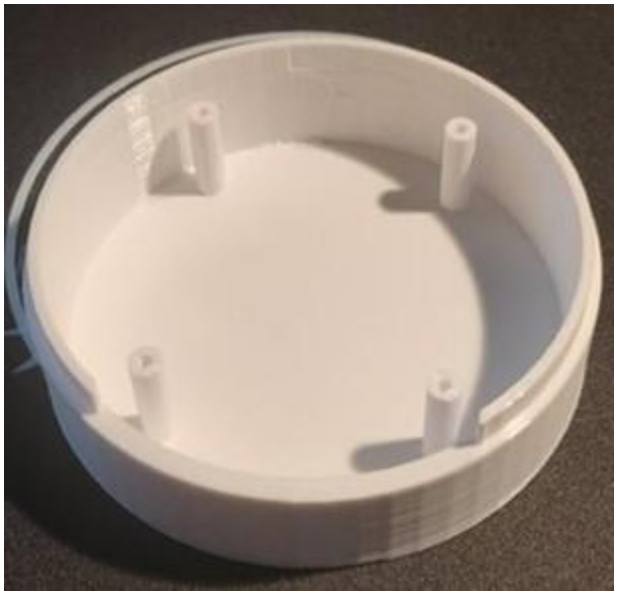
Nutikodu keskseadme ja valgusti paigutamiseks on valminud 3D prinditud korpus, mis on modelleeritud SolidWorks'i tarkvara Tallinna Tehnikaülikooli tudengilitsentsi alt. Modelleerimisel on võetud arvesse valguse hajutamist, nuppude paigutust, Raspberry Pi paigutust, protoplaadi, LED'i ja elektriskeemi paigutust. Printimine on läbi viidud FDM ehk *fused deposition modeling* meetodil ehk kiht-kihi haaval laotatakse kuuma filamenti tööpinnale arvuti poole juhitud printeri otsikuga. Mudel on prinditud kolmes osas: alumine osa, mis hoiab endas elektriskeemi ja Raspberry Pi'd, keskmine osa, kuhu on paigutatud nupud ja jahutusega LED ja ülemine osa, mis on mõeldud valguse hajutamiseks. Mudel on kolmes osas prinditud, et oleks võimalik korpust avada ja kuna filamendi paksust printimise ajal muuta ei saanud, siis õhema hajuti saamiseks prinditi see eraldi. Alumine osa koosneb poolikust avast ning neljast postist, kuhu Raspberry Pi kinni kruvida. Keskmisel osal on teine poolik ava, mis koos alumise osaga moodustab tervikliku ava Raspberry Pi I/O juurde ligipääsemiseks. Samuti on keskmisel osal kolm auku surunuppude paigutamiseks. Kuna silindriline mudel on pealt nurgaga, siis on prinditud elliptiline kujund, mis sobiks korpusele kaaneks. Järgmisena tuuakse esile 3D mudel fotode näol (joonised 19-23).



Joonis 19. 3D mudel printimises.



Joonis 20. Korpuse keskmine osa ilma kaaneta.



Joonis 21. Korpuse alumine osa.

Joonis 22. 3D mudel otsevaates.

Joonis 23. 3D mudel tagavaates.

6 Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli luua valgusti nutikodu keskseadme funktsioonides. Töö tulemusena on loodud töötav prototüüp, mis täidab nii targa valgusti kui ka nutikodu keskseadme funktsioone. Valgusti on RGB võimekusega, mis on teostatud läbi PWM kasutades Python'i teeki *pigpio*, mis teeb selle võimalikuks kasutades riistvara taimereid.

Kasutades vabavaralist kodu automaatika tarkvara openHAB loodi nutikodu keskseade, mis võimaldab ühendada kokku seadmeid, mille suhtlus põhineb erinevatel andmeside protokollidel, näiteks Z-Wave ja Wi-Fi. Sensoorikaks võeti kasutusele Fibaro FGMS-001 multisensor, mille andmeid kasutades loodi valverežiim ja automaatvalgustusrežiim. Multisensor suhtleb läbi Z-Wave USB pulga keskseadmega saates liikumisanduri, valguse intensiivsuse sensori, kiirendusanduri ja temperatuurianduri andmeid. Keskseadmega saavutati kontroll ruumiküttеseadme üle, mis teostati läbi TP-LINK HS100 targa seinakontakti.

Keskseadme kasutaja saab süsteemiga suhelda läbi HABPanel'i kasutajaliidese, kuhu on tekitatud ligipääs süsteemi erinevatele andmetüüpidele, ning mida kasutaja mõjutada saab. Kasutajaliideses on võimalik valgusti värve kombineerida ja eredust muuta, lülitada sisse nii valverežiim kui ka automaatvalgustusrežiim ning juhtida ruumiküttеseadet, muutes selle töötsüklit. Kogu süsteem on paigutatud 3D korpuse sisse, mis on disainitud kasutades SolidWorks'i modelleerimistarkvara.

Kasutajat teavitatakse sündmustest kasutades kiirsuhtlusrakendust Telegram. OpenHAB'i tarkvaraga saadetakse kasutajale sõnumeid, kui valverežiimil tuvastatakse liikumist või multisensori häirimist ja vähemolulisematest andmetest, näiteks temperatuurimuutused ning akutasememuutused. Kõik soovitud lahendused on läbi testitud ja töötavad.

Antud töö edasiarendusteks saavad olla erinevate andur- ja täiturseadmete lisamine, mis kasutavad antud realisatsioonis juba toetatud andmesideprotokolle või koguni lisada ka uusi suhtlusprotokolle laiendades veelgi kasutatavate liidesseadmete hulka.

Kasutatud kirjandus

[1] TinyCAD tarkvara
https://www.tinycad.net/ (26.12.21)

[2] LED'i andmeleht
http://www.farnell.com/datasheets/318442.pdf?_ga=2.50069285.2017181817.1589472294-1446951952.1588436330 (27.11.21)

[3] Jahutusradiaatori andmeleht
http://www.farnell.com/datasheets/1928727.pdf?_ga=2.50069285.2017181817.1589472294-1446951952.1588436330 (27.11.21)

[4] LM2596 pingeregulaatori andmeleht
https://www.ti.com/lit/ds/symlink/lm2596.pdf?ts=1637974633461&ref_url=https%253A%252F%252Fwww.google.com%252F (27.11.21)

[5] Pilt LM2596 pingeregulaatorist
https://m.media-amazon.com/images/I/41YI1K5raEL._SR500,500_.jpg (27.11.21)

[6] IRLZ44NPBF MOSFET'i andmeleht
http://www.farnell.com/datasheets/140906.pdf?_ga=2.208794385.2017181817.1589472294-1446951952.1588436330 (27.11.21)

[7] Pigpio teek
http://abyz.me.uk/rpi/pigpio/python.html (27.11.21)

[8] Raspberry Pi Pinout
https://pinout.xyz/ (27.11.21)

[9] Python split funktsioon
https://www.w3schools.com/python/ref_string_split.asp (26.12.21)

[10] openHAB dokumentatsioon
https://www.openhab.org/docs/ (27.11.21)

[11] openHAB ühilduvad seadmed
https://www.openhab.org/addons/ (26.12.21)

[12] Ülevaade openHAB mõistetest
https://www.openhab.org/docs/concepts/#things-channels-bindings-items-and-links (26.12.21)

[13] Aeotec Z-Stick Gen5 kasutusjuhend
https://help.aeotec.com/support/solutions/articles/6000242202 (27.12.21)

[14] openHAB Z-Wave Binding
https://www.openhab.org/addons/bindings/zwave/ (27.12.21)

[15] Kiirsuhtlusrakendus Telegram
https://telegram.org/ (27.12.21)

[16] openHAB turvalisus
https://www.openhab.org/docs/installation/security.html (27.12.21)

[17] SSH protokoll
https://www.ssh.com/academy/ssh (27.12.21)

Lisa 1 – Kasutajale valverežiimi lülitamisest märku andmine

```
rule "Blink lights on enable/disable"
when
    Item TamperSwitch received update
then
    if(TamperSwitch.state == ON) {
        var redOld = DutyCycleRed.state.toString
        var greenOld = DutyCycleGreen.state.toString
        var blueOld = DutyCycleBlue.state.toString
        logInfo("default.rules", "States are: " + redOld + " " + greenOld + "
 " + blueOld)
        DutyCycleRed.postUpdate("10")
        Thread::sleep(1000)
        DutyCycleGreen.postUpdate("10")
        Thread::sleep(1000)
        DutyCycleBlue.postUpdate("10")
        Thread::sleep(1000)
        var i = 1
        for(i=1; i<7; i++) {
                DutyCycleRed.postUpdate("250")
                Thread::sleep(1000)
                DutyCycleGreen.postUpdate("125")
                Thread::sleep(1500)
                DutyCycleRed.postUpdate("10")
                Thread::sleep(1000)
                DutyCycleGreen.postUpdate("10")
                Thread::sleep(1500)
        }
        DutyCycleRed.postUpdate(redOld)
        DutyCycleGreen.postUpdate(greenOld)
        DutyCycleBlue.postUpdate(blueOld)
    }
    else{
        var redOld = DutyCycleRed.state.toString
        var greenOld = DutyCycleGreen.state.toString
        var blueOld = DutyCycleBlue.state.toString
        logInfo("default.rules", "States are: " + redOld + " " + greenOld + "
 " + blueOld)
        DutyCycleRed.postUpdate("10")
        Thread::sleep(1000)
        DutyCycleGreen.postUpdate("10")
        Thread::sleep(1000)
        DutyCycleBlue.postUpdate("10")
        Thread::sleep(1000)
```

```
        var i = 1
        for(i=1; i<4; i++) {
                DutyCycleGreen.postUpdate("250")
                Thread::sleep(1000)
                DutyCycleGreen.postUpdate("10")
                Thread::sleep(500)
        }
        DutyCycleRed.postUpdate(redOld)
        DutyCycleGreen.postUpdate(greenOld)
        DutyCycleBlue.postUpdate(blueOld)
    }
end
```

Lisa 2 – Kasutajale liikumise tuvastamisest märku andmine

```
rule "Motion detected"
when
    Item ZWaveNode009FGMS001MotionSensorAlarmMotion changed from OFF to ON
then
    var redOld = "10"
    var greenOld = "10"
    var blueOld = "10"
    if(TamperSwitch.state == ON) {

        redOld = DutyCycleRed.state.toString
        greenOld = DutyCycleGreen.state.toString
        blueOld = DutyCycleBlue.state.toString
        logInfo("default.rules", "States are: " + redOld + " " + greenOld + "
 " + blueOld)
        DutyCycleGreen.postUpdate("10")
        Thread::sleep(1000)
        DutyCycleBlue.postUpdate("10")
        Thread::sleep(1000)
        DutyCycleRed.postUpdate("10")
        Thread::sleep(1000)

        var i = 1
        for(i=1; i<7; i++) {
               DutyCycleRed.postUpdate("250")
               Thread::sleep(1000)
               DutyCycleRed.postUpdate("10")
               Thread::sleep(1000)
        }
```

```
        DutyCycleRed.postUpdate(redOld)
        DutyCycleGreen.postUpdate(greenOld)
        DutyCycleBlue.postUpdate(blueOld)
    }
end
```

Lisa 3 – Kasutajale sensori häirimisest märku andmine

```
rule "Tampering detected"
when
    Item ZWaveNode009FGMS001MotionSensorAlarmTamper received update
then
    var redOld = "10"
    var greenOld = "10"
    var blueOld = "10"
    if(TamperSwitch.state == ON) {
        redOld = DutyCycleRed.state.toString
        greenOld = DutyCycleGreen.state.toString
        blueOld = DutyCycleBlue.state.toString
        logInfo("default.rules", "States are: " + redOld + " " + greenOld + "
 " + blueOld)
        var i = 1
        for(i=1; i<5; i++) {
                DutyCycleRed.postUpdate("250")
                Thread::sleep(500)
                DutyCycleBlue.postUpdate("250")
                Thread::sleep(500)
                DutyCycleRed.postUpdate("10")
                Thread::sleep(500)
                DutyCycleBlue.postUpdate("10")
                Thread::sleep(500)
        }
        DutyCycleRed.postUpdate(redOld)
        DutyCycleGreen.postUpdate(greenOld)
        DutyCycleBlue.postUpdate(blueOld)
    }
end
```

Lisa 4 – Surunupuga valverežiimi lülitamine

```
rule "Enable motion detection"
when
    Item MotionButton changed from CLOSED to OPEN
then
    if(TamperSwitch.state == ON) {
        TamperSwitch.postUpdate(OFF)
        logInfo("default.rules", "Tamper switch toggled OFF.")
    }
    else{
        TamperSwitch.postUpdate(ON)
        logInfo("default.rules", "Tamper switch toggled ON.")
    }
end
```

Lisa 5 – Surunupuga automaatvalgustusrežiimi lülitamine

```
rule "Enable automated lighting"
when
    Item AutoButton changed from CLOSED to OPEN
then
    if(AutomatedLighting.state == ON) {
        AutomatedLighting.postUpdate(OFF)
        logInfo("default.rules", "Automated lighting toggled OFF.")
    }
    else{
        AutomatedLighting.postUpdate(ON)
        logInfo("default.rules", "Automated lighting toggled ON.")
    }
end
```