TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Erik Jüri Nurmela 155721

TARGA KODU LAHENDUSTE VÕIMALUSED SAMSUNG SMARTTHINGS PLATVORMIL

Bakalaureusetöö

Juhendaja: Andres Rähni
tehnikateaduste magister

Tallinn 2019

Autorideklaratsioon

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

Autor: Erik Jüri Nurmela

20.05.2019

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on uurida arukate hoonete lahendusi, erinevate firmade koduautomaatika seadmete ühilduvust ja nende käitumist Samsung SmartThings platvormil.

Töö alguses tutvustatakse Samsung SmartThingsi ning tema arenduskeskkonda seadmete ja automatsioonide haldamiseks. Võrreldakse SmartThingsi ametlikke nutitelefoni rakendusi Android operatsioonisüsteemil ning kirjeldatakse Groovy programmeerimiskeelt SmartThingsi kontekstis.

Lõputöö käigus luuakse webCoRE reeglimootoriga juhtimisalgoritm targa pirni ja hämarduslüliti vahel. Parandatakse SmartThingsi poolt ametlikult toetamata kahe releega täituri funktsionaalsust.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 28 leheküljel, 21 peatükki, 21 joonist.

Abstract

Smart home solution possibilities on Samsung SmartThings platform

The aim of this bachelor thesis is to examine smart home possibilities, compatibility of various home automation devices and their behaviour on the Samsung SmartThings platform.

In the beginning, there is an introduction to Samsung SmartThings. Following that, an overview of the SmartThings IDE for managing devices and automations is given. The author also analyses differences in functionality and practicality of official SmartThings smartphone applications. A part of this thesis briefly covers the Groovy programming language and its usage in SmartThings.

The result is an algorithm to control a smart bulb with a dimmer switch on webCoRE rule engine piston on Samsung SmartThings. Additionally, full functionality is granted for a double relay switch, which is not officially supported on the Samsung SmartThings platform.

The thesis is in Estonian and contains 28 pages of text, 21 chapters, 21 figures.

Lühendite ja mõistete sõnastik

Android	Google'i arendatav mobiil-operatsioonisüsteem
Device Handler	Seadme käitumist defineeriv programmikood Groovy keeles
Groovy	Programmeerimiskeel
Hub	Ruuteri külge ühenduv jaotur, mis vahendab juhtmevabalt targa kodu seadmete vahel andmeid
IDE	*Integrated development environment*, integreeritud programmeerimiskeskkond
iOS	Apple'i arendatav operatsioonisüsteem oma seadmetele
JVM	*Java Virtual Machine*, abstraktne masin, mis interpreteerib kompileeritud Java baitkoodi arvuti protsessorile arusaadavateks instruktsioonideks
SmartApp	Groovy keeles kirjutatud või šablooni järgi telefoni rakenduses tehtud automatsioon
webCoRE	*web Community's own Rule Engine*, automatsioonide reeglimootor SmartThingsile
Zigbee	Üks andmesideprotokollidest mida kasutavad targa kodu seadmed
Z-Wave	Üks andmesideprotokollidest mida kasutavad targa kodu seadmed

Sisukord

1 Sissejuhatus 8
2 SmartThings platvorm 9
2.1 Näiteid SmartThingsi kasutusest 10
2.2 SmartThings Groovy IDE 10
2.2.1 My Locations vahekaart SmartThings Groovy IDEs 11
2.2.2 My Hubs vahekaart SmartThings Groovy IDEs 11
2.2.3 My Devices vahekaart SmartThings Groovy IDEs 12
2.2.4 My SmartApps vahekaart SmartThings Groovy IDEs 13
2.2.5 My Device Handlers vahekaart SmartThings Groovy IDEs 13
2.2.6 My Publication Requests vahekaart SmartThings Groovy IDEs 14
2.2.7 Live Logging ja Documentation vahekaardid SmartThings Groovy IDEs 15
2.3 Telefoni rakenduste SmartThings ja SmartThings Classic võrdlus 15
2.3.1 Automatsioonid SmartThings ja SmartThings Classic rakendustes 18
3 Groovy programmeerimiskeel 22
3.1 Groovy SmartThingsis 22
3.1.1 Groovy keele lihtsustused 22
4 webCoRE reeglimootor 24
4.1 Ettevalmistused webCoREi kasutamiseks 25
4.2 webCoRE kolvi loomine 28
4.2.1 Loodud kolvi testimine 31
5 Fibaro kahe releega täitur 33
5.1 Fibaro FGS-222 testimine 35
6 Kokkuvõte 36
Kasutatud kirjandus 37
Lisa 1 – Classic rakenduse liidese kirjeldus handleri koodis 39
Lisa 2 – Digitalgecko puldi handler 40

Jooniste loetelu

Joonis 1. SmartThings Groovy IDE avaleht. 11
Joonis 2. Samsung SmartThings Hub [9]. 12
Joonis 3. My Devices vahekaart SmartThings Groovy IDEs. 13
Joonis 4. Aeotec Multisensor 6 füüsiline seade [2]. 14
Joonis 5. Rakenduste avaleht. 16
Joonis 6. Aeotec Multisensor 6 detailvaade SmartThings rakenduses. 17
Joonis 7. Aeotec Multisensor 6 detailvaade SmartThings Classic rakenduses. 18
Joonis 8. Automatsiooni loomine SmartThings rakenduses. 19
Joonis 9. Automatsiooni loomine SmartThings Classic rakenduses. 20
Joonis 10. Automatsioonide info kandumine kahe rakenduse vahel. 21
Joonis 11. SmartThingsi andmevoo ja suhtluse visualiseerimine [17]. 25
Joonis 12. webCoRE katsetuse käigus kasutatud seadmed. 25
Joonis 13. Puldi sündmused (a) vaikimisi handleriga, (b) uue handleriga. 26
Joonis 14. IDE logisse saadetavad sündmused 27
Joonis 15. IDE logisse saadetavad sündmused nii enne kui peale modifitseerimist, nupu all hoidmisega. 27
Joonis 16. Muutus digitalgecko handleri koodis. 28
Joonis 17. Lambi juhtimiseks valminud kolb 30
Joonis 18. webCoRE kolvi logi väljavõte. 32
Joonis 19. Harutoosi sisse installerimise illustratsioon [19] 33
Joonis 20. Releemooduli elektriskeemid. (a) Ühe sisend-väljund paariga. (b) Kahe sisend-väljund paariga. 34
Joonis 21. Rakenduses multisensori graafilist liidest kirjeldav osa koodist. 39

1 Sissejuhatus

Lõputöö on koostatud juhendiks ning annab ülevaate ühest koduautomaatika ökosüsteemist. See on valminud autori huvist targa kodu seadmete vastu. Arvutisüsteemide instituudi laboris olemasolevatest lahendustest sai läbi töötamiseks valitud Eesti turul figureeriv platvorm Samsung SmartThings. Platvorm on suunatud elamutele nagu korter ja eramaja, mitte suurtele lao- või kontorihoonetele ja on kasutajasõbralik ametlikult toetatud tooteid kasutades.

Tarkadest seadmetest olid vaatluse all *hub*, *bridge*, tark pirn, pult, kahe releega lüliti ja multisensor. SmartThings Hub (versioon 2, alates 2015. a.) ja Philips Hue Bridge ühendati ruuteri külge ja vooluvõrku, rohkem nendega töö käigus toimetada vaja ei olnud. Philips Hue White Bulb, Philips Hue Dimmer Switch, Fibaro FGS-222 ja Aeotec Multisensor 6 töötati läbi IDEs ja telefonirakendustes. Uuriti seadmete seadistuste ja omaduste kirjeldamist programmikoodis (edaspidi *handler*).

Loodi webCoRE reeglimootoriga automatsioon SmartThings platvormil.

Nutitelefonidele on SmartThings rakendusi kaks, üks uuem ja teine vanem. Võrreldi omavahel kahte rakendust, nende erinevusi ja sarnasusi automatsioonide tegemisel ning tavakasutusel.

2 SmartThings platvorm

Targa kodu lahenduste üldised eesmärgid on ressursside säästlik kasutamine, turvalisuse ja ohutuse suurendamine, mugavuse tagamine ning alamsüsteemide eesmärkide täitmine (ventilatsioon, küte jne) [1].

SmartThings on lihtne viis muuta oma kodu targaks koduks. Ta suudab monitoorida, kontrollida ja automatiseerida koduseadmeid, et elu lihtsamaks teha [2].

Iga element kasvavas SmartThings ökosüsteemis lubab kasutajatel SmartThings pilve (*Cloud*) ehitada ja integreerida asjade interneti (IoT, *Internet of Things*) seadmeid, teenuseid ja eksisteerivaid lahendusi [3].

Paljudele SmartThings kasutajate kogukonna liikmetele meeldib kirjutada *SmartApp* programme ja seadme tüübi *handler*eid ning neid teistega jagada. Ametlikus SmartThings *community* foorumis on just selleks koht. Nii mõnedki liikmed postitavad tulemusi spetsiifilistest rakenduste ja seadmete kombinatsioonidest, mis tunduvad neile kasulikuna [4].

SmartThings *Community* foorum on koht õppimiseks, aitamiseks ja kogemuste jagamiseks SmartThingsi, SmartAppide, IoT ja automatsioonide kohta. Avalikus foorumis saavad kõik näha erinevaid arutelusid.

Ometi on raske leida täpselt seda mida vaja on, sest informatsiooni on palju ja kogus ainult kasvab. Õnneks on foorumites liikumise lihtsustamiseks kasutatud silte. Siltide abil saab jaotada postitusi vastavalt sisule erinevate kategooriate alla [5].

SmartThings Inc. on Ameerika tehnoloogiafirma, mille peamaja on Mountain View, Californias. SmartThings osteti Samsung Electronics poolt 2014 aasta augustis [6].

SmartThings toetab paljude firmade erinevaid seadmeid oma platvormil: Aeotec, Amazon, Arlo, August, Belkin, BeSense, Bosch, Bose, Centralite, Cree, CURB, Danalock, Dome, ecobee, Eaton, Ecolink, Enerwave, Everspring, Fibaro, First Alert, FortrezZ, GE, Google, Honeywell, iHome, IKEA, Iris, JASCO, KeyWe, Kwikset, Leak Intelligence, LeakSMART, Leviton, LiFi Labs, Go Control, Lutron, Neo Coolcam, Nortek, Nyce, Philips Hue, Remotec Technology, Ring, Samsung, Satco, Schlage,

Sengled, Sensative, SmartThings, Springs Window Fashions, SYLVANIA, Yale, Zen, Zooz [2].

2.1 Näiteid SmartThingsi kasutusest

Kui hommikul üles ärkad saab SmartThings tuled põlema panna, kütet juurde keerata, raadio käima panna ja kohvimasina käivitada.

Kui kodust ära lähed on võimalik SmartThingsi abil kõik uksed lukustada, tuled kustutada, kütet väiksemaks keerata ja turvakaamera tööle panna.

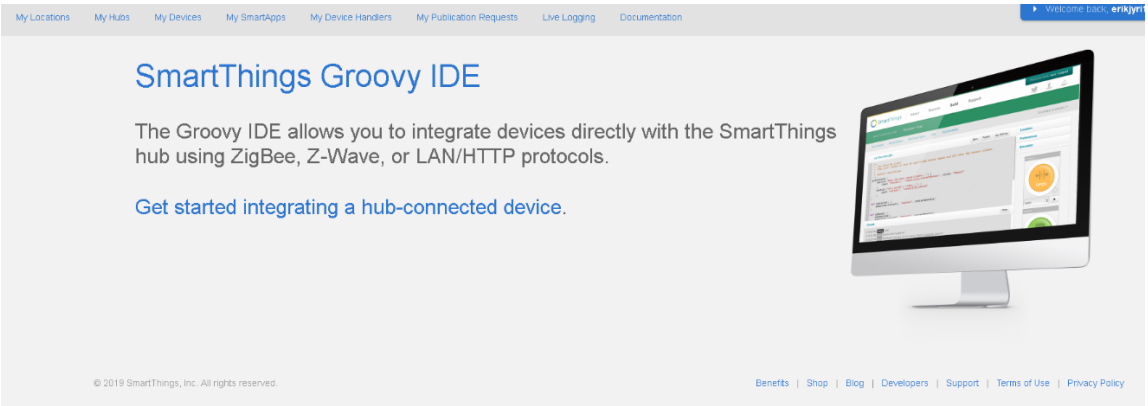
Olles kodust eemal saab SmartThings saata videoteavitusi kui kodus midagi ootamatut juhtub, hoiatada veelekke korral ja palju muud.

Õhtul koju jõudes võimaldab SmartThings automaatselt avada garaaži ukse, kohandada valguse temperatuuri lampides, käima panna lemmik muusika ja kontrollida teisi koduseadmeid.

Magama minnes teab SmartThings hoiatada kui mõni uks või aken on jäänud lahti, enne kui kõik tuled kustu paneb ja termostaadi madalama kütte peale reguleerib [2].

2.2 SmartThings Groovy IDE

Põhiline tööriist edasijõudnud kasutajatele Samsung SmartThings seadmetega töötamiseks on IDE (Joonis 1).Selle graafiline liides on tagasihoidlik, pigem informatiivne.
Veebilehel näeb andmeid hubi ja seadmete kohta. Kasulik silumiseks ja seadmete korrektses töötamises veendumiseks. See sisaldab kõiki SmartThings telefonirakenduste funktsioone.



Joonis 1. SmartThings Groovy IDE avaleht.

2.2.1 My Locations vahekaart SmartThings Groovy IDEs

Minu asukohad (*My Locations*) vahekaardil näeb kõiki kasutaja poolt tekitatud või talle lisatud asukohti. Vahekaardil on viited logidele, ning kõigele muule mida näeb SmartThings rakenduses. Funktsionaalsuse poolest katab *My Locations* kaart IDE 100% ulatuses, viitades kõikidele teistele vahekaartidele kui vajutada mõne asukoha peale. Kõigepealt kuvatakse asukoha nimi kuhu peale vajutades näeb asukoha infot, seal hulgas omaniku Samsungi kontot, *hub*i töörežiimi, temaga ühendatud seadmeid, sündmusi, automatsioone ja seadmegruppe.

Asukoha all mõeldakse ühe Samsungi *hub*i paiknemist, igal *hub*il on oma asukoht. Kui *hub* on juba ühele kasutajale lisatud, saab tema asukohta teiste kasutajatega jagada. Kõik kellega asukoht on jagatud saavad *hub*i ja tarkvara kontrollida omanikuga võrdselt: lisada ja eemaldada seadmeid, konfigureerida automatsioone, lisada asukohta uusi kasutajaid ning neid sealt eemaldada. Omanikku ei saa tema enda asukohast eemaldada ja privileegiks jääb enda asukoha kustutamine.

Asukohta saab kutsuda kasutajaid, kellel on Samsungi konto samas regioonis ning kes on eelnevalt vähemalt korra Samsung SmartThings rakendusse sisse loginud. Asukoha jagamine pole vastastikune. Kasutaja lisamisel ei lisa ta sind automaatselt enda asukohta vastu [7].

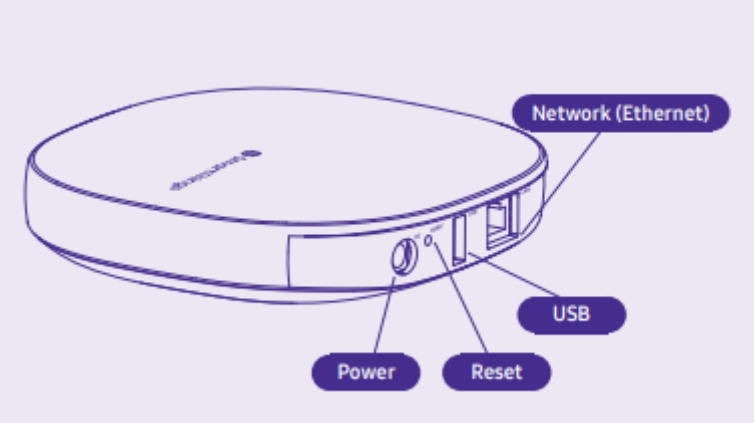
2.2.2 My Hubs vahekaart SmartThings Groovy IDEs

Minu *hub*id (*My Hubs)* vahekaardil kuvatakse kõik kasutaja enda ja kasutajaga läbi asukoha jagatud *hub*id ning nende külge ühendatud seadmed. *Hub* on vajalik targa kodu

seadmete omavaheliseks suhtlemiseks. Ta töötab kui vahendaja, tõlkides erinevates andmesideprotokollides suhtlevate seadmete sõnumeid üksteisele loetavaks. Ühendatakse ruuteri külge või Wi-Fi võrku ja vooluallikaks on pistik (Joonis 2).

Kasutaja saab kontrollida mitut *hub*i ühe konto alt, eeldusel et iga *hub* on lisatud kuhugi asukohta. Samu seadmeid aga mitme *hub*iga kontrollida ei saa, kuna iga seade võib korraga ainult ühe *hub*i külge ühendatud olla [8].

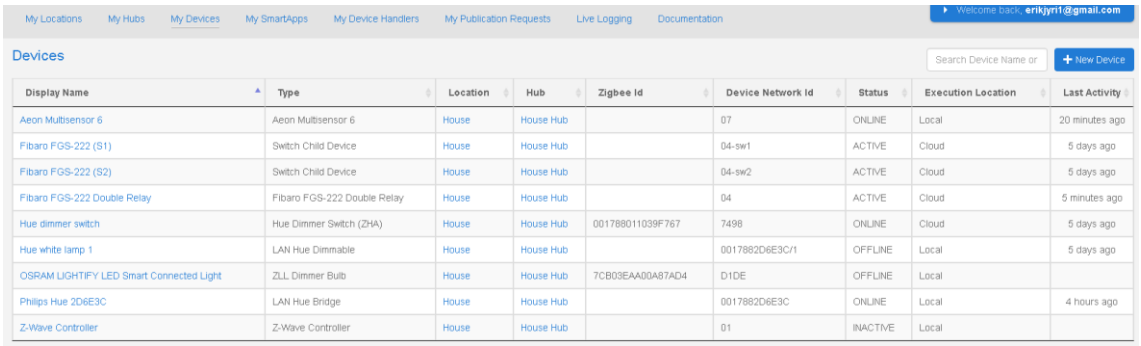
*Hub*i kohta kuvatakse: *hub* ID, staatus, tarkvara ja riistvara versioon, asukoht, viimati aktiivne, loomiskuupäev, viimati uuendatud, IP aadress, MAC aadress, viimati käivitatud, toite staatus, seadistus, Zigbee protokolli info ja staatus, Z-Wave protokolli info ja staatus, viide sündmustele.



Joonis 2. Samsung SmartThings Hub [9].

2.2.3 My Devices vahekaart SmartThings Groovy IDEs

Minu seadmed (*My Devices*) vahekaardil on näha kõik seadmed mis on seotud mõne kasutajale lisatud asukoha või *hub*iga (Joonis 3). Kuvatakse info: Nimi, tüüp(*handler*), asukoht, staatus jms. Näeb ka eelnevaid sündmusi, näiteks millal liikumisandur liikumise tuvastas või millal mis nuppu vajutati. Kõik sündmused defineeritakse programmeeritud *handler*iga.



Joonis 3. My Devices vahekaart SmartThings Groovy IDEs.

Olles *My Device Handler*s vahekaardil oma *handler*i lisanud, saab selle kasutusele võtta *My Devices* kaardil. Mõne seadme peale vajutades ja edasi *Edit* vajutades tuleb muuta rippmenüüst seadme tüüpi. Enda lisatud *handler*id on valikus tavaliselt kõige all.

2.2.4 My SmartApps vahekaart SmartThings Groovy IDEs

My SmartApps valikus on kuvatud kõik veebi kasutajaliideses käsitsi lisatud *SmartApp*id. Telefonirakenduses saab teha erinevate šabloonide põhjal automatsioone kuid neid siin vahekaardil ei kuvata. Siin on ainult programmeerimise tulemusel valmivad rakendused.

Uue loomiseks saab paremal üleval vajutada "*+New SmartApp*". Võimalik on Gitist(versioonihaldustarkvara) või koodifailist programm importida. Vajutades esimesele variandile avaneb menüü uue programmi loomise kohta. Programmi saab teha malli järgi, kuid see eeldab Groovy programmeerimiskeele oskust ja modifitseerimist. Algajale on see väga raske ülesanne. *SmartApp*ide programmikoodis kirjutamine on sobiv eelkõige edasijõudnud kasutajatele.

2.2.5 My Device Handlers vahekaart SmartThings Groovy IDEs

Minu seadmetüübid (*My Device Handlers*) all antakse kasutajale võimalus lisada enda koodi uuteks seadmetüüpideks. *Handler* defineerib seadme tüübi ehk käitumise, mis on vajalik seadme korrektseks töötamiseks. Ilma korrektse *handler*ita võib kaduda seadme funktsionaalsus 100% ulatuses. Asja teevad veel keerulisemaks andmesideprotokollid: näiteks Zigbee, Z-Wave, Wi-Fi jne. Erinevate andmesideprotokollidega suhtlevad seadmed ei oska omavahel otse kommunikeerida, infot võidakse saata aga aru sellest ei saada.

Tavaliselt on funktsionaalsuse kadu väiksem kui 100%. Kui seade õnnestub telefoni rakenduses lisada, on ta leitud täitma vähemalt ühte funktsiooni. Seadme lisamisel

defineerib rakendus ta ühe *handler*iga mis suudab SmartThingsi arvates kõige paremini esindada konkreetset seadet. Vähempopulaarsetele toodetele ja seadmetele mis pole ametlikult SmartThingsi poolt toetatud leidub sageli paremini töötavaid, kogukonna loodud *handler*eid.

2.2.6 My Publication Requests vahekaart SmartThings Groovy IDEs

Minu esitatud *handler*id (*My Publication Requests*) leht lubab esitada enda kirjutatud *Device Handler*i kõigile kasutamiseks SmartThings platvormil. Võimalik, et lisatud *handler* määratakse vaikimisi kõigile uutele seadmetele, kui varem samal seadmel ametlikult *handler*it polnud või on loodu praegusest vaikimisi *handler*ist parem.

Vaheleht on kasutamiseks mõeldud pigem firmadele kes toodavad seadmeid ja tahavad, et need oleksid kohe SmartThingsi poolt toetatud. Täpsustada saab, kas lähtekood jäetakse nähtavaks või hoitakse salajas. Lähtekoodi lubatakse salajasena hoida ainult konkreetse seadme tootja- ja müügifirmal. Kõik taotlused vaadatakse üle ja hinnatakse *SmartThings*i poolt enne üldkasutatavaks tegemist. Kui midagi on hindamiseks saadetud, näeb lehel esitamise staatust.

Vaadeldes multisensorit (Aeotec Multisensor 6) (Joonis 4) tekkis mõte, et äkki on seadmel paremaid, SmartThings kogukonna poolt tehtud *handler*eid, mis hoiaks näiteks rohkem patareid. Multisensor on täielikult patareitoitel, mis tähendab, et optimaalne suhtlus võrguga on väga oluline patarei eluea pikendamisel.

Joonis 4. Aeotec Multisensor 6 füüsiline seade [2].

Olles ühe handleri leidnud, lisati see enda seadmetüüpide alla (*My Device Handlers*). IDE näitas autori nime koha peal SmartThings. Ilmnes, et kogukonnas on inimesed jõud ühendanud ja loonud põhjaliku *Device Handler*i seadmele, millel enne ametlikult polnud.

Nad esitasid handleri (*My Publication Requests* alt) ja see saigi peale üle vaatamist vaikimisi seadmetüübiks.

2.2.7 Live Logging ja Documentation vahekaardid SmartThings Groovy IDEs

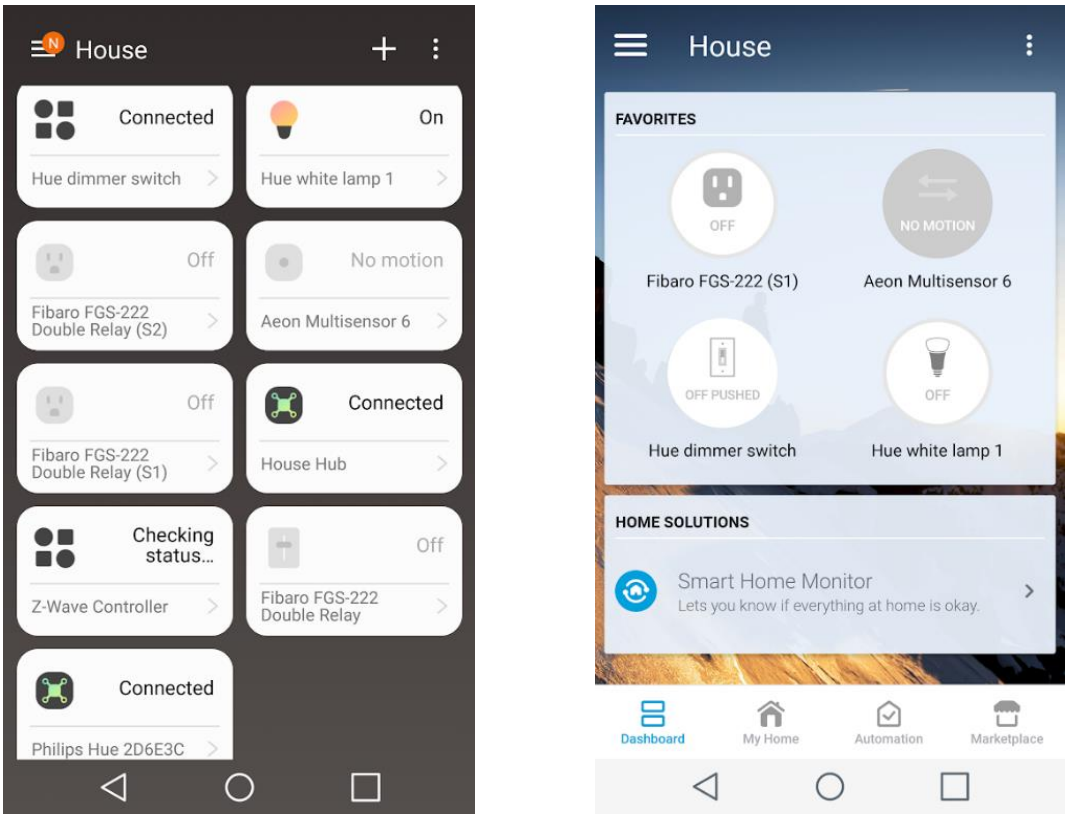
Live Logging konsooli kasutatakse *handler*ite ja *SmartApp*ide töö silumiseks. *Handler*isse saab kirja panna erinevates olukordades logisse kirjutamise käske. Näiteks kontrollitakse muutujate väärtusi seadme töö käigus. Kui *handler* või *SmartApp* on valmis kirjutatud ja enam silumist ei vaja, eemaldatakse reeglina koodist logi väljatrükid. Kustutatakse täienisti või kommenteeritakse välja kui võib tulevikus vaja minna.

Documentation on otsetee Samsung SmartThings dokumentatsiooni internetilehele, mis avaneb uues brauseri aknas. Dokumentatsiooni vahekaarti kui sellist ei eksisteeri.

2.3 Telefoni rakenduste SmartThings ja SmartThings Classic võrdlus

Samsungi targa kodu rakendusi saab alla laadida nii Android kui ka iOS operatsioonisüsteemiga nutitelefonidesse. Kõik ühes rakendusi on kokku kaks, uuem (Samsung SmartThings) ja vanem (Samsung SmartThings *Classic*). Funktsionaalsuse mõttes täidavad enamjaolt samu ülesandeid kuid kasutamise koha pealt on nii mõnigi asi erinev.

Juba avalehel üritatakse uuemas rakendus moodsama disaini peale mängida (Joonis 5).



(a) (b)

Joonis 5. Rakenduste avaleht.

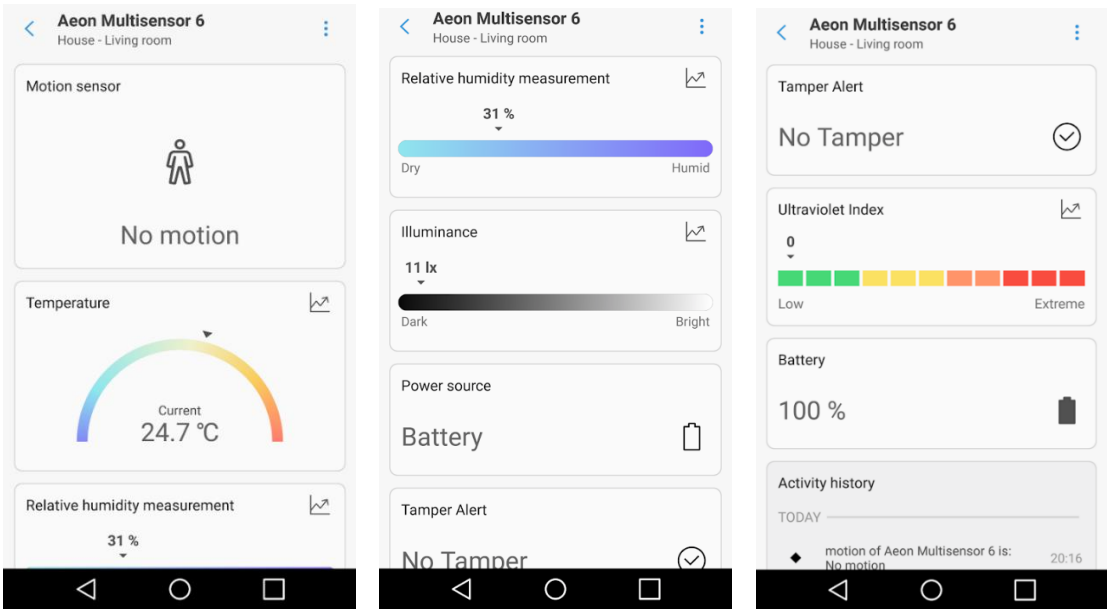
(a) Uuem rakendus Samsung SmartThings. (b) Vanem Samsung SmartThings Classic.

Samsungi põhirõhk läheb hetkel uue rakenduse arendamiseks, viimane uuendus(1.6.29) tuli iOSile 09.04.2019 [10]. Teisele rakendusele tuli viimane uuendus iOSil(2.17.1) pool aastat tagasi, 04.10.2018 [11].

Mõlemad rakendused on vaadeldud Android 6.0 operatsioonisüsteemiga telefonis LG G3.

Uuem rakendus on suunatud rohkem tavakasutajale. Kena kujundus ja suured ümarad nupud annavad edasi lihtsust ja mugavust. Nii võib tunduda inimesele kes ühendab oma asjad ja juhib neid ainult rakendusest. Autori kasutuskogemusest on uuem rakendus piiratuma funktsionaalsusega, kuid kasutamiseks intuitiivsem. Populaarsematele seadmetüüpidele ja nende võimekustele (*Capability*) on SmartThingsi enda poolt arendatud kasutajaliidesed, et kuvada värvilised, informatiivsed graafikud. Graafikud ilmuvad nähtavale iga näidu paremal üleval aja funktsiooni joonele vajutades (Joonis 6). Alati näidatakse ka sündmuste ajalugu (tuvastati liikumine, vajutati nuppu).

Aeotec Multisensor 6 on selge näide graafilise liidese erinevustest kahes rakenduses (Joonis 6) (Joonis 7).

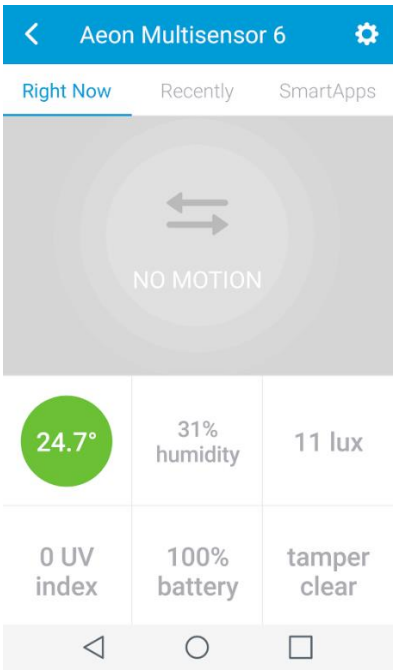


(a) (b) (c)

Joonis 6. Aeotec Multisensor 6 detailvaade SmartThings rakenduses.

(a) Vaate ülemine osa, (b) vaate keskmine osa, (c) vaate alumine osa.

Classic rakenduse kasutajaliideses näeb seadme vaates ainult *handler*is kirjeldatud ruutudel vastavat informatsiooni. Vanem rakendus kannatab ainult graafikute puudumisega. Sündmuste ajalugu on nähtav vajutades üleval nupule *Recently* (Joonis 7). Kasutajaliidese kirjeldus seadmevaates lähemalt lisade all (Vt Lisa 1, lk 40).

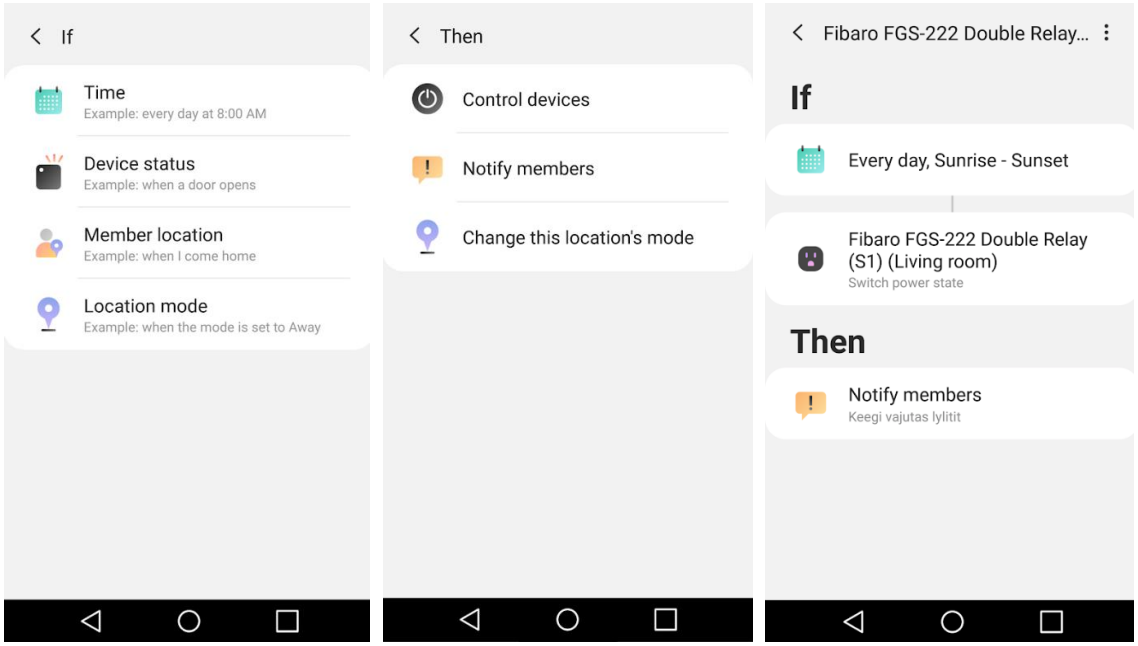


Joonis 7. Aeotec Multisensor 6 detailvaade SmartThings Classic rakenduses.

Samsung SmartThings *Classic* rakendus on edasijõudnud kasutaja tööriist nutitelefonis. Selles saab konfigureerida veebiliideses (IDEs) lisatud *SmartAppe* ja näha ise kirjutatud *handler*i manuaalset asetust seadmevaates, mõlemat mida SmartThings rakenduses teha ei saa. Küll aga saab eelmainitud *SmartAppe* näha ja konfigureerida SmartThings rakenduses peale nende lisamist Samsung SmartThings *Classicus*. Rakendused vahetavad omavahel informatsiooni. Enamus mis ühes tehtud kajastub ka teises [12].

2.3.1 Automatsioonid SmartThings ja SmartThings Classic rakendustes

Automatsioonide tegemine SmartThings rakenduses käib puhtalt *if-then* loogika järgi. Saab valida tingimusi (*if*) millal mingi tegevus aktiveeritakse (*then*) (Joonis 8).



(a) (b) (c)

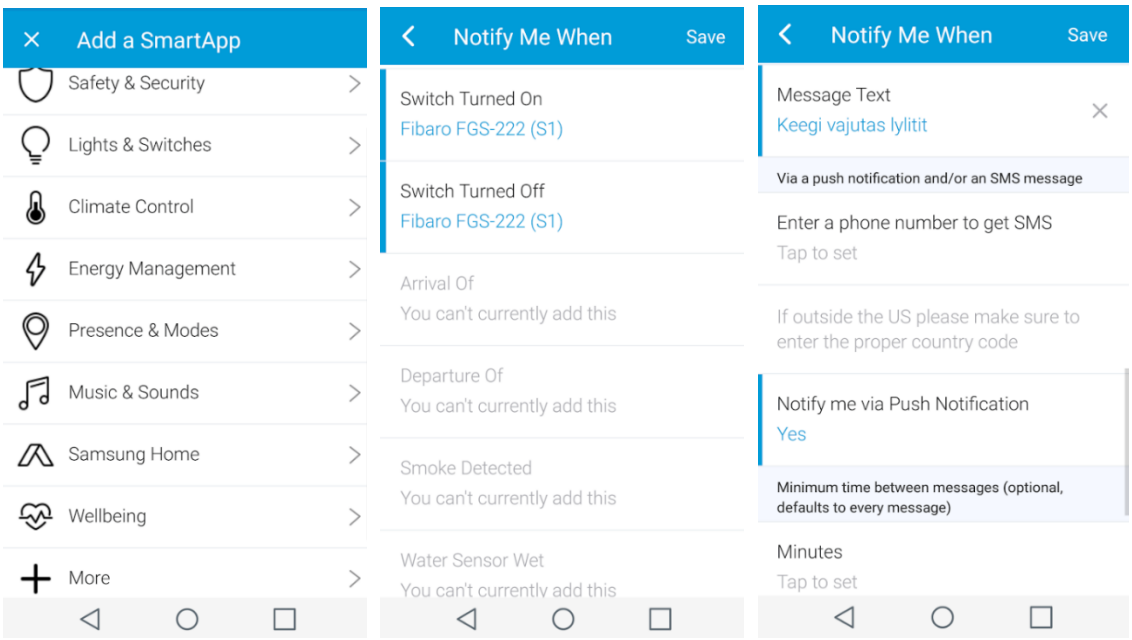
Joonis 8. Automatsiooni loomine SmartThings rakenduses.

(a) Kõigepealt valitakse esimene tingimus (if) ehk aeg (Time). Seejärel kontrollitakse teist tingimust, kas tütarseadmel (Device status), ühe seinakontakti voolu lülitil on toimunud muutus.

(b) Kui toimub muutus, siis tuleb nutitelefoni tõuketeavitus sisuga „Keegi vajutas lylitit“.

(c) Automatsiooni lõppkuva

Classic rakenduses tehakse automatsioone mudelite järgi või kasutatakse veebis programmeeritud *SmartApp*e (Joonis 9).



(a) (b) (c)

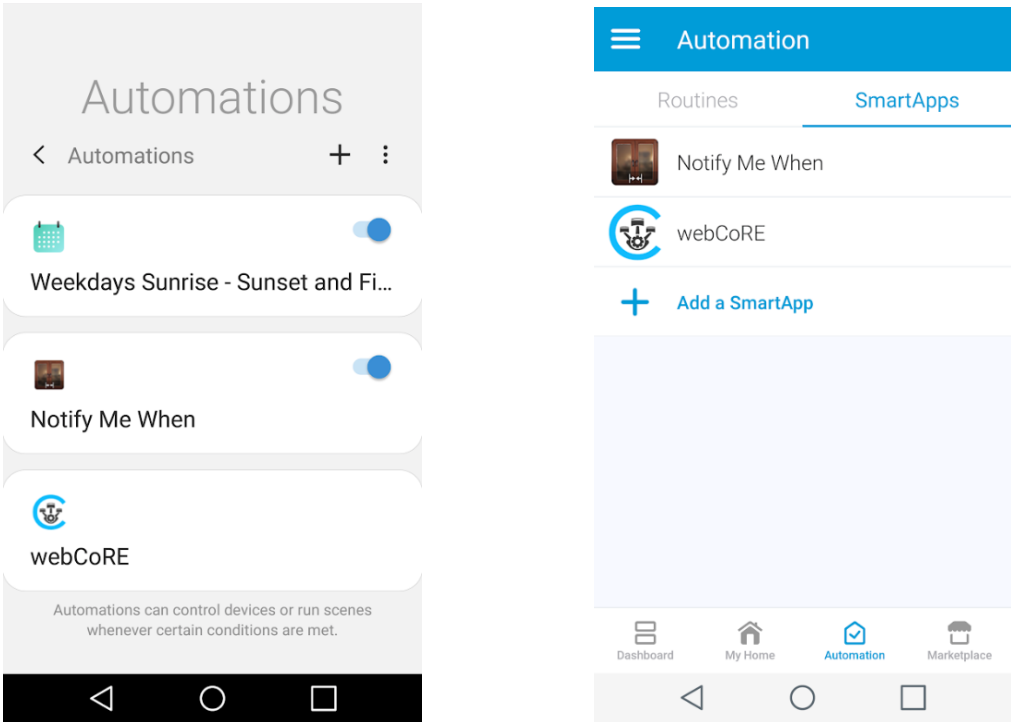
Joonis 9. Automatsiooni loomine SmartThings Classic rakenduses.

(a) Vajutatakse valitakse *Add a SmartApp* ja *Safety & Security*.

(b) Kontrollitakse kas tütarseadmel – seinakontakti voolu lülitil on toimunud lülitus.

(c) Saadetakse telefoni tõuketeavitus sisuga „Keegi vajutas lylitit“

Šablooni järgi tehtud automatsioonid kanduvad sama nimega üle uuemasse rakendusse. *If-then* mudeliga uuemas rakenduses tehtud automatsioonid *Classic* rakendusse üle ei tule (Joonis 10). IDE’s on näha mõlemas rakenduses tehtud automatsioone.



(a) (b)

Joonis 10. Automatsioonide info kandumine kahe rakenduse vahel.

(a) SmartThingsis on näha Classicus tehtud automatsiooni *(Notify Me When*).

(b) Classicus ei näe SmartThingsis loodud automatsiooni (*Weekdays Sunrise – Sunset*...).

Igapäevaseks kasutamiseks tuleks valida üks kahest rakendusest. Kui mõlemad on korraga installeeritud kopeerivad nad üksteise sisu ja kõik teavitused mis seadistatakse tulevad kahekordselt. Vähemalt on nii nutitelefoni enda teavitustega. Kui teavitust nõutakse seadme staatuse muutumisel, näiteks lüliti sisselülitamisel, tuleb neid kaks tükki. Sama sisuga üks Samsung SmartThingsilt ja teine Samsung SmartThings *Classic*ult.

Kasutusmugavuse vaatepunktist nautis autor rohkem *Classic* rakenduse kasutamist. Viide andmete laadimisel on väiksem, seadmete kohta näidatakse koguaeg korrektseid andmeid ja on kuvatud *handler*iga kirjeldatud praktilise välimusega seadme detailvaated. *Classic* rakendus osutus testimise käigus töökindlamaks. Oli hetki, mil SmartThings rakendus ei laadinud seadmeid ja väitis, et puudub võrguühendus. Samal ajahetkel aga *Classic*us kõik töötas.

3 Groovy programmeerimiskeel

Apache Groovy on objektorienteeritud programmeerimiskeel Java platvormil. Dünaamiline keel on sarnaste võimalustega Pythonile, Rubyle, Perlile ja Smalltalkile. Olles tuttav Java, C, C++, Pythoni, Ruby või JavaScriptiga on näha Groovys analoogiat. Groovy kood on kompileeritud bait koodiks, mis täidetakse Java *Virtual Machine*is.

Groovy valiti SmartThingsi programmeerimiskeeleks oma lihtsuse ja paindlikkuse ning JVMi jõudluse ja stabiilsuse tõttu. Kuna Groovy kompileeritakse bait koodi mis täidetakse JVMis on 99% Java koodist korrektne Groovy. Standardsed Java teegid on saadaval Groovy programmides. Groovy laiendab Javat paljudel kasulikel viisidel [13], [14].

3.1 Groovy SmartThingsis

Groovy töötab SmartThingsis „liivakastis“, isoleerituna osadest oma võimekustest. Kogu SmartThingsi kood käivitatakse SmartThingsi ökosüsteemis. Kui kirjutada *SmartApp* või *Device Handler*, käivitatakse kood lõpuks ikkagi SmartThingsi platvormil. Seepärast keelab SmartThings teatud meetodite ja funktsioonide kasutuse. Näiteks ei saa programmikoodis luua uut faili või avada olemasolevat faili.
SmartThingsil arendades on üks esimesi märgatavaid asju, et kasutada on palju funktsioone ja meetodeid ilma neid importimata. On väga haruldane üldse mõnda import lauset SmartThingsis näha.
Iga *SmartApp* või *Device Handler* on tegelikult abstraktse *Executor* klassi eksemplar SmartThings platvormil. Konkreetne *Executor* klass defineerib ja sisaldab palju meetodeid. Selle tulemusena ei pea igas *SmartApp*is ega *Device Handler*is mitte midagi importima, et pea kõiki võimalikke funktsionaalsusi kasutada. Taolise mudeliga saab koodi sees ilma importimata kutsuda paljusid eeldefineeritud meetodeid [15].

3.1.1 Groovy keele lihtsustused

*SmartApp*i või *Device Handler*i autorina on keelatud oma klasside loomine ja kohandatud .jar failide importimine. See võib alguses arendajale suure piiranguna tunduda kuid kasutamise käigus veendutakse, et tegelikult nii ei ole. Enamuse meetodite automaatse kaasamise tulemusena vajadust oma klasside või objekti struktuuride loomise järgi ei teki.

Mõned meetodid on potentsiaalse turvariskina kasutamiseks keelatud. Nii mõnigi nendest tegeleb Groovy täiustatud metaprogrammi kontseptsioonidega. Need lubavad arendajatel saada ja muuta programmi käitlusteavet. SmartThingsis pole see vajalik, nii et need on keelatud.

Veel on piiratud oma lõimede loomine ja süsteemi meetodite kasutamine nagu *System.out*(). Lisaks looklevate sulgude kaasamine defineerimisel, näiteks *def squareItClosure = {it * it}.*

Neid programmis kasutades kuvatakse käivitamisel *SecurityException* veateade. Keelatud meetodid on: *addShutdownHook*(), *execute*(), *getClass*(), *getMetaClass*(), *setMetaClass*(), *propertyMissing*(), *methodMissing*(), *invokeMethod*(), *mixin*(), *print*(), *printf*(), *println*(), *sleep*() [15].

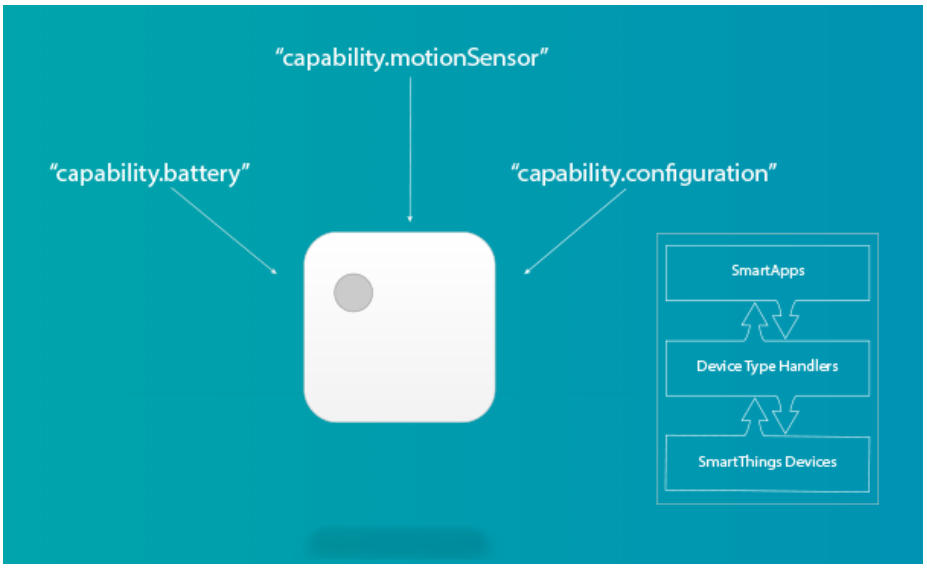
4 webCoRE reeglimootor

web Community's own Rule Engine on kogukonna poolt loodud reeglimootor SmartThingsile. Mootor laseb kasutajatel skripte teha, mida interpreteerib ja teostab SmartThingsi paigaldatav *SmartApp*. webCoRE arenduskeskkond (IDE) asub brauseris. Nende kodulehel on peale tutvustuse ja võimaluste kirjelduse ka paigaldusjuhend koos videoga. webCoREis loodud programmid on nime saanud sisepõlemismootoris töötavate kolvide järgi (*piston*).

webCoRE on veel arengustaadiumis, esimene beetaversioon oli aprillis aastal 2017. Praegu on reeglimootor teises beetaversioonis 2017 juunist. Juba selle ajaga on palju programmivigu likvideeritud ja lisatud uusi funktsionaalsusi. Veebilehitsejas paiknev arenduskeskkond on ligipääsetav nii arvutist kui ka nutitelefonist [16].

Kasutajate poolt tehtud kolvid võivad olla üsna keerulised, sest webCoRE on loonud justkui omaenda programmeerimiskeele. Semantika ja süntaks on sarnased teistele keeltele aga programmeerimine on olemuselt hoopis teistsugune.
Ei kirjutata rea kaupa programmikoodi nagu tavaliselt. Selle asemel saab peaaegu kõik ära teha hiirevajutustega: lisada tingimuslikke lauseid valides kõik muutujad, mõjutada seadmeid, lugeda seadmete olekuid jne. Muutujate kasutamine, konsooli trükk ja muu tavapärane programmeerimine käib läbi väljendite (*expression*). Väljendid on üks rida koodi lünga sees.

webCoREi teeb võimsaks kogu loogika mida inimene suudab programmile välja mõtelda. Lisaks sellele saavad erinevad kolvid omavahel suhelda, käitudes üksteise järgi või vahetades andmeid läbi globaalmuutujate. Kasutamiseks on oluline aru saada kuidas näeb välja seadmete, *handler*ite ja *SmartApp*ide vaheline hierarhia (Joonis 11).



Joonis 11. SmartThingsi andmevoo ja suhtluse visualiseerimine [17].

4.1 Ettevalmistused webCoREi kasutamiseks

Katsetuse käigus õpiti Philips *Hue Dimmer Switch* puldiga kontrollima Philips *Hue White Bulb* lambipirni läbi Philips *Hue Bridge*i (Joonis 12). Kõik Samsung SmartThings platvormil ja *hub*il.

(a) (b) (c)

Joonis 12. webCoRE katsetuse käigus kasutatud seadmed.

(a) Philips *Hue Bridge* [18]. (b) Philips *Hue Dimmer Switch* koos seinahoidjaga [18].

(c) Philips *Hue White Bulb* [18].

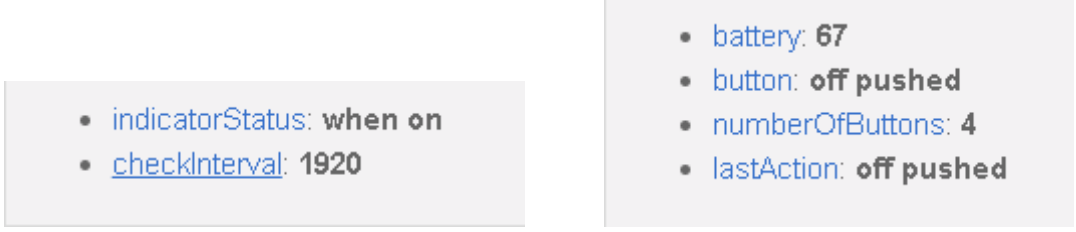
Oluline on teada, et kui lampi mille sisse on keeratud tark pirn läbib vool, on pirn võimeline teiste seadmetega juhtmevabalt suhtlema. Targal pirnil patareid sees pole, ehk kasutamisel peab lamp koguaeg vooluvõrgus olema. Lamp tuleb välja lülitada virtuaalselt (rakendusest) mitte füüsilisest lülitist. Muidu kaob juhtimise võimalus ja seadme staatuseks kuvatakse *offline*.

Kõigepealt installeeriti webCoRE SmartThingsi. Lisati *hub*i külge SmartThings *Classic* rakenduses *bridge*, siis pirn ja seejärel pult. Nutitelefonis oli näha pirni eredus ja staatus. Peale puldi ühendamist hakkas kõik justkui iseenesest tööle. Lambi pirni oli võimalik puldi abil heledamaks ja tumedamaks muuta ning sisse-välja lülitada. Pirnile omistati vaikimisi *LAN Hue Dimmable*, puldile *Dimmer Switch* ja *bridge*ile *LAN Hue Bridge handler*. SmartThings sai nende suhtlema panemisega hakkama kuid tegi tee peal lihtsustusi.

Puudusteks olid puldi vajutuste nähtavus ja vähene info telefonirakenduses. Kuskil polnud jälge, et puldil oleks mõnda nuppu vajutatud. Pult oli SmartThings *Classic* rakenduses seadmete all ikoonina olemas aga rakenduses temaga teha midagi ei saanud, ainult füüsiline seade töötas.

Otsiti internetist puldile paremat *handler*it, ning leiti üks üsna populaarne kogukonna liikme (digitalgecko) poolt postitatud *handler* (Vt Lisa 2, lk 40). Leitud *handler* on teoreetiliselt võimeline teenindama igat Zigbee andmesideprotokolliga pulti millel on vähemalt üks nupp.

Vaikimisi omistatud *handler* puldi kohta eriti kasulikku informatsiooni ei andnud (Joonis 13). Pärast *handler*i muutmist on näha puldi aku taset, nupu vajutuse staatust, nuppude arvu ja viimast tegevust (*lastAction*).



(a) (b)

Joonis 13. Puldi sündmused (a) vaikimisi handleriga, (b) uue handleriga.

lastAction on põhimõttelt sama mis *button*. *lastAction*il on ainult nimi ja väärtus, ning ta uuendab ennast ainult *button*i staatuse muutumisel. Kui vajutatakse kümme korda järjest *off* nuppu, uuendab *lastAction* ennast ainult esimesel korral.

Kuna taheti nuppude poolt saadetud sündmuste peale webCoREi kolvi ehitada, modifitseeriti veidi *handler*it (Joonis 16). *Handler*i autori (digitalgecko) versioonis tuli lühikese vajutuse peale korraga kaks sündmust, vajutati ja lasti lahti. Nende peale oli keeruline automatsiooni ehitada, sest nad registreeriti tihti erinevas järjekorras. Tema versioonis oli samuti vahe sellel, kas nuppu vajutati hetkeks või kuni sekundilise all hoidmisega. Testimise tulemusena leiti, et lihtsam ja arusaadavam on kui iga nupu vajutuse kohta tuleks seadmelt ainult üks sündmus (Joonis 14).

Nupu all hoidmisel tuleb kõige pealt sündmus 1: nuppu on vajutatud, siis sündmus 2: nuppu hoitakse all ja sündmus 3: nupp vabastatud. *Handler*i autori (digitalgecko) versiooniga võrreldes muudeti ainult vajutuse kirjeldust. Enne oli *pressed_down*, nüüd on *down pushed* (Joonis 15).

button	**released_down**
button	**pressed_down**

(a)

button	**off pushed**
button	**up pushed**

(b)

Joonis 14. IDE logisse saadetavad sündmused .

(a) Enne modifitseerimist, ühe nupu vajutusega. (b) Peale modifitseerimist, kahe erineva nupu vajutus.

button	**released_down**
button	**held_down**
button	**down pushed**

Joonis 15. IDE logisse saadetavad sündmused nii enne kui peale modifitseerimist, nupu all hoidmisega.

```
// ENNE
    if ( buttonState == 0 ) {
        result = [createEvent(name: "button", value: "pushed", \
        data: [buttonNumber: buttonName], descriptionText: \
        "$device.displayName button $button was pushed", isStateChange: true)]
        sendEvent(name: "lastAction", value: buttonName + " pressed")
    }
    else if ( buttonState == 2 ) {
        result = [
        createEvent(name: "button", value: "pushed_" + buttonName, \
        data: [buttonNumber: buttonName], descriptionText: \
        "$device.displayName button $button was pushed", isStateChange: true),
        createEvent(name: "button", value: "released_" + buttonName, \
        data: [buttonNumber: buttonName], descriptionText: \
        "$device.displayName button $button was pushed", isStateChange: true)
        ]
        sendEvent(name: "lastAction", value: buttonName + " pushed")
        sendEvent(name: "lastAction", value: buttonName + " released")
    }

// PÄRAST
    if ( buttonHoldTime == 0 ) {
        result = [createEvent(name: "button", value: buttonName + " pushed", \
        data: [buttonNumber: buttonName], descriptionText: \
        "$device.displayName button $button was pushed", isStateChange: true)]
        sendEvent(name: "lastAction", value: buttonName + " pushed")
    }
```

Joonis 16. Muutus digitalgecko handleri koodis.

4.2 webCoRE kolvi loomine

Järgnevalt loodi kolb, et webCoRE abiga puldi nupu vajutused valgustiga siduda. webCoREi esilehel tuleb valida "+ *New Piston*". Seejärel avaneb valik kolvi loomiseks, kas luua uus täiesti tühjalt lehelt, kopeerida enda varasem kolb, taastada kustutatud kolb või importida. Tuleb valida esimene variant ja luua uus.

Soovitatav on sisse lülitada automaatne varundamine, siis luuakse brauseri vahemällu fail kus hoitakse viimati muudetud koodi. Kui minnakse kolvi muutma ja webCoRE leiab arvutist hilisema versiooniga varukoopia, pakutakse automaatselt taastamist.

Keypad 1 on pult (Philips *Hue Dimmer Switch*) ja *Dimmer* 1 on tark pirn (Philips *Hue White Bulb*).

Kolb ehitatakse puldi poolt saadetud sündmuste peale, ehk programmi jooksutatakse iga kord kui tuleb puldilt sündmus (Joonis 17, Rida 21–23). Edasi kontrollitakse mis sisuga sündmus tuleb (Joonis 17, Read 26, 33, 40, 57, 74, 86) ja defineeritakse iga juhtumi (*case*) kohta eraldi tegevus.

Sündmuse „*on pushed*“ korral saadetakse pirnile sisse lülitamise käsk (Joonis 17, Rida 27–31). Sündmuse „*off pushed*“ korral saadetakse pirnile väljalülitamise käsk (Joonis 17, Rida 34–38). Mõlemat nuppu mitmekordselt järjest vajutades midagi topelt ei juhtu, sisse- või väljalülitatud seade jääb samasse staatusesse.

Kui vajutatakse pirni eredamaks ehk tuleb sündmus „*up pushed*“, siis kontrollitakse enne kas pirni eredus jääb 75% ja 100% vahele (Joonis 17, Rida 41–42). Kui jääb siis lülitatakse pirn maksimum ereduse peale (Joonis 17, Rida 44–48), kui mitte siis tehakse 20% eredamaks (Joonis 17, Rida 50–54).

Kui vajutatakse pirni tumedamaks, ehk tuleb sündmus „*down pushed*“, siis tehakse täpselt samamoodi nagu eredamaks muutmisel, ainult tagurpidi. Kontrollitakse kas eredus on madalama 25% sees (Joonis 17, Rida 59). Kui on siis viiakse kõige tumedama 5% peale (Joonis 17, Rida 61–65), kui mitte, siis tehakse 20% tumedamaks (Joonis 17, Rida 67–71).

Olles saanud puldilt sündmuse, et eredamaks või tumedamaks nupp on all hoitud, minnakse tsükli sisse kuni all hoidmise lõpuni (Joonis 17, Read 75–84, 87–96). Tsüklis reguleeritakse iga 50 millisekundi tagant valgustit 10% kaupa eredamaks või tumedamaks. Tsükkel lõpetatakse kui jõutakse kas kõige eredamani (100%), kõige tumedamani (5%) või lastakse nupp lahti.

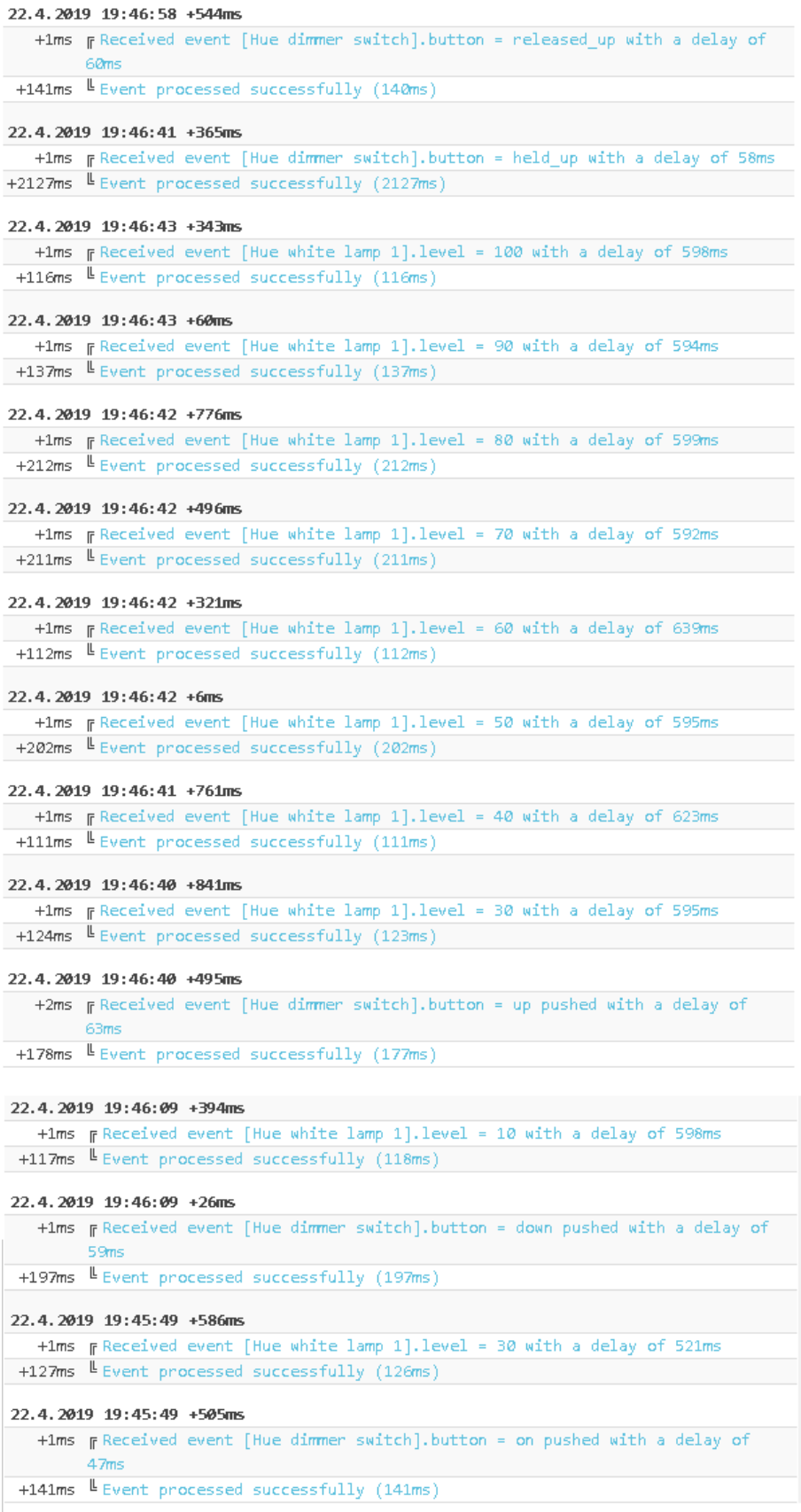
```
settings
    enable parallelism;
end settings;

execute
    on events from
        Keypad 1's button
    do
        switch (Keypad 1's button)
            /* Sisse nupp, lylitab sisse kui lamp ei põle */
            case {"on pushed"}:
                with
                    Dimmer 1
                do
                    Turn on;
                end with;
            /* Välja nupp, lylitab välja kui lamp põleb */
            case {"off pushed"}:
                with
                    Dimmer 1
                do
                    Turn off;
                end with;
            /* Dimm up, teeb eredamaks kui lamp põleb */
            case {"up pushed"}:
                if
                    Dimmer 1's level is inside of range 75% and 100%
                then
                    with
                        Dimmer 1
                    do
                        Fade level to 100% in 0 seconds;
                    end with;
                else
                    with
                        Dimmer 1
                    do
                        Adjust level by 20%;
                    end with;
                end if;
            /* Dimm down, teeb pimedamaks kui lamp põleb */
            case {"down pushed"}:
                if
                    Dimmer 1's level is inside of range 0% and 25%
                then
                    with
                        Dimmer 1
                    do
                        Fade level to 5% in 0 seconds;
                    end with;
                else
                    with
                        Dimmer 1
                    do
                        Adjust level by -20%;
                    end with;
                end if;
            /* Dimm up (nuppu all hoides), teeb eredamaks kui lamp põleb */
            case {"held_up"}:
                while
                    Dimmer 1's level is inside of range 5% and 90%
                do
                    with
                        Dimmer 1
                    do
                        Adjust level by 10%;
                        Wait 50 milliseconds;
                    end with;
                end while;
            /* Dimm down (nuppu all hoides), teeb pimedamaks kui lamp põleb */
            case {"held_down"}:
                while
                    Dimmer 1's level is inside of range 15% and 100%
                do
                    with
                        Dimmer 1
                    do
                        Adjust level by -10%;
                        Wait 50 milliseconds;
                    end with;
                end while;
        end switch;
    end on;
end execute;
```

Joonis 17. Lambi juhtimiseks valminud kolb

4.2.1 Loodud kolvi testimine

Loodud kolvi testimise logi on näha joonisel 18. Alguses vajutatakse pirn sisse 19:45:49, kuvatakse eredus 30%. Siis vajutatakse korra pirni tumedamaks 19:46:09, kuvatakse eredus 10%.
Peale seda hoitakse eredamaks nuppu all 19:46:40 kuni 19:46:58, kusjuures pirn saavutab 100% ereduse juba 19:46:43 ehk kolme sekundiga.

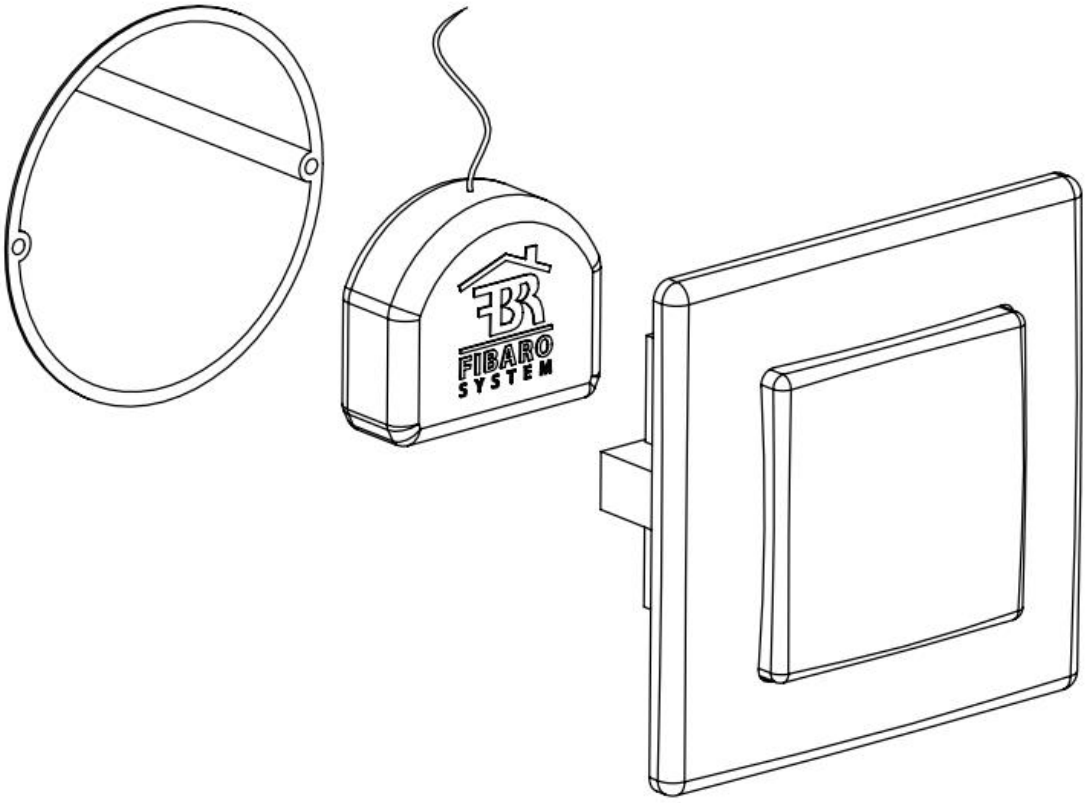
Kõik töötab nii nagu soovitud, pirn läheb eredamaks maksimaalselt 100% ja tumedamaks maksimaalselt 5%.
Pirni sisse ja välja lülitades säilib sama heledus. Pirni saab juhtida vajadusel ka SmartThings rakendustest.



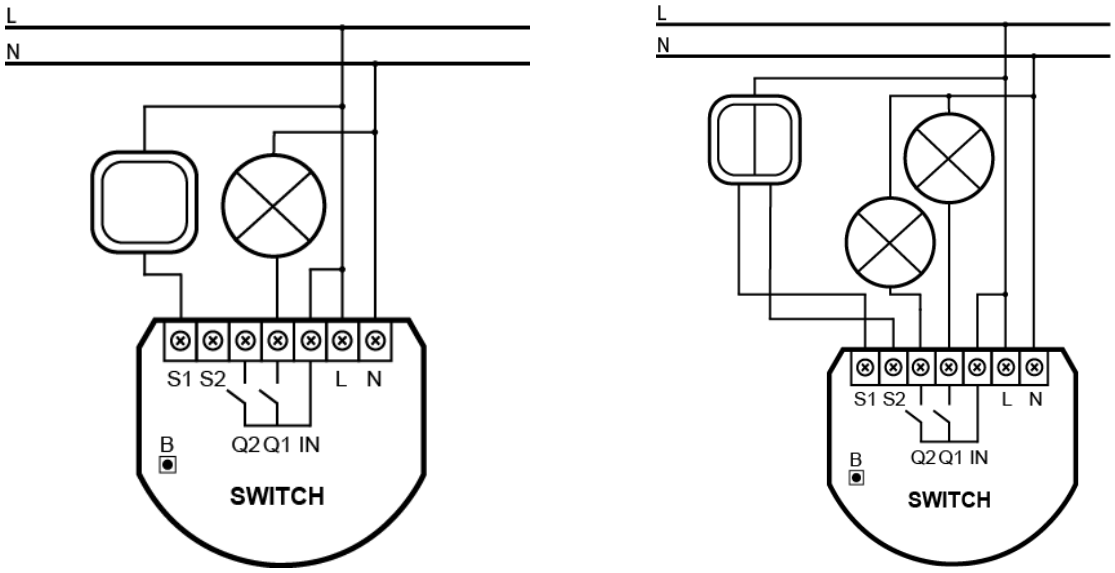
Joonis 18. webCoRE kolvi logi väljavõte.

5 Fibaro kahe releega täitur

Fibaro kahe releega täitur FGS-222 on disainitud harutoosi sisse installeerimiseks (Joonis 19) või kus vaja, kontrollimaks kahte eraldi seadet mis tarbivad kumbki kuni 1500W. Fibaro *Double Relay Switch*i saab kontrollida kas Z-Wave võrgus või otse külge ühendatud lülitiga. Töötamiseks tuleb anda vahelduvvoolu 110-240V (50/60Hz). Seadet saab ühendada kas ühe või kahe sisend-väljund paariga (Joonis 20) [19].



Joonis 19. Harutoosi sisse installerimise illustratsioon [19]



(a) (b)

Joonis 20. Releemooduli elektriskeemid. (a) Ühe sisend-väljund paariga. (b) Kahe sisend-väljund paariga.

L – faasi juhtme terminal.

N – neutraali juhtme terminal.

S1 – esimese lüliti terminal.

S2 – teise lüliti terminal.

IN – sisendterminal lülitava faasi jaoks.

B – teenindav nupp (Fibaro seadme lisamiseks või eemaldamiseks).

Q1 – väljundterminal, esimese koormuse pingestamise juhtimiseks.

Q2 – väljundterminal, teise koormuse pingestamise juhtimiseks.

5.1 Fibaro FGS-222 testimine

Fibaro relee sai ühendatud kahe sisend-väljund paariga maketi peale, lüliti taha. Sisendid on kaks lüliti klahvi ja väljundid on kaks pistikupesa. Pesadele antakse kas voolu või mitte, olenevalt klahvivajutusest või virtuaalselt (telefoni rakendusest) saadetavatest lülitustest.

Seadme lisamisel SmartThingsi leitakse ta üles, kuid nimetatakse vaikimisi „Z-Wave *Device Multichannel"*. Tuleb välja, et SmartThings platvormil konkreetset seadet ametlikult ei toetata. Vaikimisi omistatud *handler*iga seadme kasutamisel ilmneb, et lüliti kahest ühendatud klahvist töötab ainult üks.

Otsiti internetist, kas mõni SmartThingsi kasutaja on proovinud luua paremat *handler*it ning leiti üks katsetamiseks [20].

Kõigepealt mindi Groovy IDE netilehele, kus salvestati uus *handler,* koodi kopeerides. Fibaro FGS-222 sai SmartThingsi eelnevalt lisatud ja kandis veel nime "*Z-Wave Device Multichannel*". *My Devices* alt otsiti seade üles ja vajutati *Edi*t. Seadme tüübiks muudeti "Fibaro FGS-222 *Double Relay*".
Telefoni rakendustes muutus nimi koheselt ja *Classic* rakenduses seadme peale vajutades oli näha *handler*is kirjeldatud graafilist kasutajaliidest. Nüüd tuli *handler*i looja kohaselt vajutada rakenduses (seadme peal olles) paremal üleval kolme punkti või hammasratast ning *Edit*. Siin sai muuta erinevaid parameetreid kuid see polnud kohustuslik ning vajutati *Save*. *Handler* sai *Edit* vajutusega konfigureerimiskäsu, et tütarseadmed tekitada.

Leitud *handler* tagas mõlema lüliti klahvi toimimise ning lasi konfigureerida kõiki tehase poolt loodud parameetreid. Ta tekitas kaks tütarseadet, üks mõlema lüliti klahvi kohta. Andis pistikupessa voolu nii virtuaalsete lülituste kui ka füüsiliste klahvide vajutuste peale.

6 Kokkuvõte

Samsung SmartThings on võimas platvorm millel saavad kasutajad palju ise ära teha. Konfigureerimisulatus on muljet avaldav, ning kogukonna poolt loodav on tasemel. Veidi aega panustades saab vähe toetatud seadmete kasutusvõimalusi tunduvalt parandada.

Automatsioonide loomine pole keeruline, kasutusel on intuitiivne *if-then* loogika. Vaid mõne vajutusega saab automatiseerida näiteks lambi tööle lülitamise liikumise tuvastamisel.

webCoRE reeglimootori abiga kolbe (ehk algoritme) luues on võimalik need keeruliseks ning isikupäraseks muuta. Neis saab tingimusi, tsükleid, muutujaid ja palju muud lisada ning seadmed omavahel suhtlema panna. Ainult SmartThingsi poolt pakutavad tööriistad kolbe ei asenda.

Kasutajatel on valida kahe telefonirakenduse vahel, kas uuem SmartThings või veidi vanem SmartThings *Classic*. Uuem rakendus sobib tavakasutuseks, kuid edasijõudnule jääb sellest väheks. Arendusrõhk on küll uuema peal kuid kõiki funktsionaalsusi *Classic* rakendusest veel ei täideta.

Nutikodu tehnoloogiad toodavad olemuselt mugavustooteid. Seadmete soetamisel peab mõtlema kas need tasuvad ära. Samsung SmartThingsi viimase põlvkonna seadmed on teiste Zigbee ja Z-Wave seadmete kõrval soodsamad. Mõne lambipirni valgusviljakuse muutmiseks on odavamaid alternatiive.
Nutikodu valdkond on asjade interneti (IoT, Internet of Things) arengutest tugevalt mõjutatud, ning SmartThings annab sellesse oma panuse.

Antud lõputööd saab kasutada õppematerjalina arukate hoonete praktikumides.

Kasutatud kirjandus

[1] A. Rähni, „Arukad hooned ja automaatika,“ 2019. [Võrgumaterjal]. Available: http://www.tud.ttu.ee/web/Andres.Rahni/Arukad_hooned_2.pdf. [Kasutatud 2 Aprill 2019].

[2] SmartThings Inc, „SmartThings. Add a little smartness to your things,“ 2019. [Võrgumaterjal]. Available: https://www.smartthings.com. [Kasutatud 5 Aprill 2019].

[3] SmartThings Inc, „SmartThings Developers | Documentation,“ 2019. [Võrgumaterjal]. Available: https://smartthings.developer.samsung.com/docs/index.html. [Kasutatud 24 Aprill 2019].

[4] SmartThings Inc, „SmartThings Community,“ 2019. [Võrgumaterjal]. Available: https://community.smartthings.com/. [Kasutatud 10 Aprill 2019].

[5] Things That Are Smart Wiki contributors, „How to Quick Browse the Community-Created SmartApps Forum Section,“ 20 Märts 2019. [Võrgumaterjal]. Available: http://thingsthataresmart.wiki/index.php?title=How_to_Quick_Browse_the_Community-Created_SmartApps_Forum_Section. [Kasutatud 18 Aprill 2019].

[6] A. Tilley, „Samsung Acquires SmartThings, A Fast-Growing Home Automation Startup,“ Forbes Media LLC, 30 detsember 2014. [Võrgumaterjal]. Available: https://www.forbes.com/sites/aarontilley/2014/08/14/samsung-smartthings-acquisition-2/#50ede0a33367. [Kasutatud 16 mai 2019].

[7] SmartThings Inc, „Location sharing FAQ - SmartThings UK Support,“ SmartThings Inc, 2017. [Võrgumaterjal]. Available: https://support.smartthings.com/hc/en-gb/articles/206531223-Location-sharing-FAQ. [Kasutatud 13 Mai 2019].

[8] SAMSUNG, „Using Multiple SmartThings Hubs,“ 2019. [Võrgumaterjal]. Available: https://www.samsung.com/us/support/answer/ANS00048960/. [Kasutatud 4 Aprill 2019].

[9] SmartThings Inc, „How to add a SmartThings hub - SmartThings Support,“ 2017. [Võrgumaterjal]. Available: https://support.smartthings.com/hc/en-gb/article_attachments/360002608703/HUB_QSG.pdf. [Kasutatud 13 Aprill 2019].

[10] Apple Inc, „SmartThings on the App Store,“ 2019. [Võrgumaterjal]. Available: https://itunes.apple.com/us/app/smartthings/id1222822904. [Kasutatud 24 Aprill 2019].

[11] Apple Inc, „SmartThings Classic on the App Store,“ 2019. [Võrgumaterjal]. Available: https://itunes.apple.com/us/app/smartthings-classic/id590800740. [Kasutatud 21 Aprill 2019].

[12] SmartThings, „Tiles - SmartThings Classic Developer Documentation,“ 2018. [Võrgumaterjal]. Available: https://docs.smartthings.com/en/latest/device-type-developers-guide/tiles-metadata.html. [Kasutatud 17 Aprill 2019].

[13] Apache Groovy project, „The Apache Groovy programming language - Documentation,“ 2019. [Võrgumaterjal]. Available: http://www.groovy-lang.org/documentation.html. [Kasutatud 24 Aprill 2019].

[14] SmartThings, „Groovy Basics - SmartThings Classic Developer Documentation,“ 2018. [Võrgumaterjal]. Available: https://docs.smartthings.com/en/latest/getting-started/groovy-basics.html. [Kasutatud 31 Märts 2019].

[15] SmartThings, „Groovy With SmartThings - SmartThings Classic Developer Documentation,“ 2018. [Võrgumaterjal]. Available: https://docs.smartthings.com/en/latest/getting-started/groovy-for-smartthings.html#groovy-for-smartthings. [Kasutatud 31 Märts 2019].

[16] webCoRE Wiki contributors, „webCoRE,“ 23 Veebruar 2019. [Võrgumaterjal]. Available: https://wiki.webcore.co/webCoRE. [Kasutatud 26 Aprill 2019].

[17] SmartThings Inc, „Overview | SmartThings Developers,“ SmartThings Inc, 2019. [Võrgumaterjal]. Available: https://developers.smartthings.com/ . [Kasutatud 31 Märts 2019].

[18] „The official site of Philips Hue,“ Signify Holding, 2019. [Võrgumaterjal]. Available: https://www2.meethue.com. [Kasutatud 4 Mai 2019].

[19] FIBAR GROUP S.A, „Double Relay Switch | FIBARO Manuals,“ 2019. [Võrgumaterjal]. Available: https://manuals.fibaro.com/double-relay-switch/. [Kasutatud 27 Märts 2019].

[20] SmartThings Inc, „[RELEASE] Fibaro FGS-222 Double Relay Device Handler (Also Philio PAN 02/04),“ Oktoober 2018. [Võrgumaterjal]. Available: https://community.smartthings.com/t/release-fibaro-fgs-222-double-relay-device-handler-also-philio-pan-02-04/92738. [Kasutatud 5 Aprill 2019].

Lisa 1 – Classic rakenduse liidese kirjeldus handleri koodis

```
tiles(scale: 2) {
    multiAttributeTile(name: "motion", type: "generic", width: 6, height: 4) {
        tileAttribute("device.motion", key: "PRIMARY_CONTROL") {
            attributeState "active", label: 'motion', icon: "st.motion.motion.active", backgroundColor: "#00A0DC"
            attributeState "inactive", label: 'no motion', icon: "st.motion.motion.inactive", backgroundColor: "#cccccc"
        }
    }
    valueTile("temperature", "device.temperature", inactiveLabel: false, width: 2, height: 2) {
        state "temperature", label: '${currentValue}°',
            backgroundColors: [
                [value: 32, color: "#153591"],
                [value: 44, color: "#1e9cbb"],
                [value: 59, color: "#90d2a7"],
                [value: 74, color: "#44b621"],
                [value: 84, color: "#f1d801"],
                [value: 92, color: "#d04e00"],
                [value: 98, color: "#bc2323"]
            ]
    }

    valueTile("humidity", "device.humidity", inactiveLabel: false, width: 2, height: 2) {
        state "humidity", label: '${currentValue}% humidity', unit: ""
    }

    valueTile("illuminance", "device.illuminance", inactiveLabel: false, width: 2, height: 2) {
        state "illuminance", label: '${currentValue} lux', unit: ""
    }

    valueTile("ultravioletIndex", "device.ultravioletIndex", inactiveLabel: false, width: 2, height: 2) {
        state "ultravioletIndex", label: '${currentValue} UV index', unit: ""
    }

    valueTile("battery", "device.battery", inactiveLabel: false, decoration: "flat", width: 2, height: 2) {
        state "battery", label: '${currentValue}% battery', unit: ""
    }

    valueTile("batteryStatus", "device.batteryStatus", inactiveLabel: false, decoration: "flat", width: 2, height: 2) {
        state "batteryStatus", label: '${currentValue}', unit: ""
    }

    valueTile("powerSource", "device.powerSource", height: 2, width: 2, decoration: "flat") {
        state "powerSource", label: '${currentValue} powered', backgroundColor: "#ffffff"
    }

    valueTile("tamper", "device.tamper", height: 2, width: 2, decoration: "flat") {
        state "clear", label: 'tamper clear', backgroundColor: "#ffffff"
        state "detected", label: 'tampered', backgroundColor: "#ff0000"
    }

    main(["motion", "temperature", "humidity", "illuminance", "ultravioletIndex"])
    details(["motion", "temperature", "humidity", "illuminance", "ultravioletIndex", "batteryStatus", "tamper"])
}
```

Joonis 21. Rakenduses multisensori graafilist liidest kirjeldav osa koodist.

Lisa 2 – Digitalgecko puldi handler

https://github.com/digitalgecko/mySmartThings/blob/master/devicetypes/digitalgecko/hue-dimmer-switch-zha.src/hue-dimmer-switch-zha.groovy