TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mattias Kõosalu 134911IAPB

VÄLISE LOOGIKAKOMPONENDI ARENDAMINE KNX VÕRGULE

bakalaureusetöö

Juhendaja: Tarvo Treier
Magister MSc

Tallinn 2020

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: Mattias Kõosalu

08.01.2020

Annotatsioon

Töö eesmärk on luua reeglimootori prototüüp KNX standardile vastavale hooneautomaatika lahendusele. Reeglimootor kontrollib ja käivitab salvestatud reegleid, mis on kujul kui…siis.

Reeglimootor pakub kasutajatele võimalust hooneautomaatika lahenduse loogikat lihtsasti ümber seadistata ning koostada keerulisi reegleid lihtsal kujul ilma kallite lisakomponentideta. Reeglimootoriga on võimalik kasutajal ümber seadistada olemasolev hooneautomaatika lahendus vastavalt muutuvatele vajadustele.

Töö tulemusena on loodud reeglimootori prototüüp, millesse saab sisestada reegleid läbi REST API. Reeglimootor tagab reeglite täitmise suheldes KNX võrguga.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 41 leheküljel, 3 peatükki, 18 joonist, 0 tabelit.

Abstract

Developing an external logic component for a KNX network

The purpose of this thesis is to develop a prototype of a rule engine to Control a KNX standard based automated building solution. The rule engine evaluates and executes rules in the form if…then.

The rule engine offers users the opportunity to easily reconfigure their automated building solution and easily create complicated rules without expensive external devices. Users can use the rule engine to reconfigure an existing building automation solution according to their changing needs.

The result of this thesis is a prototype rule engine. Rules can be added to the rule engine using a REST API. The rule engine implements the behaviour described through rules by communicating with a KNX network.

The thesis discusses the KNX standard and finds a method to control the behaviour of KNX devices and analyzes the principles of the rule engine. As part of the thesis an importer for KNX project data is developed along with a method of adding and evaluating rules that may use data from a KNX solution.

The thesis is in Estonian and contains 41 pages of text, 3 chapters, 18 figures, 0 tables.

Lühendite ja mõistete sõnastik

API	*Application programming interface*, rakendusliides
Domeenispetsiifiline keel	*Domain-specific language*, konkreetsele domeeni keskenduv programmeerimiskeel
ETS	*Engineering Tool Software*, rakendus KNX lahenduse seadistamiseks
Grupiaadress	*Group address*, aadress, mis määrab KNX lahenduses sõnumi sihtkoha
Grupiväärtus	*Group value*, KNX lahenduses sisalduva sõnumi väärtus, mida tõlgendavad KNX seadmed
IP	*Internet protocol*, interneti protokoll
KNX	Hooneautomaatika standard
OpenHAB	Tarkvara hooneautomaatika programmeerimiseks
Reeglimootor	Reeglite käivitamise keskkond
REST	*Representational state transfer*, tarkvaraarhitektuur
SaaS	*Software as a service*, tarkvara kui teenus
URL	*Uniform resource locator*, viide veebis olevale ressursile
XML	*Extended markup language*, keel andmete hoidmiseks

Sisukord

Jooniste loetelu

1 Sissejuhatus

Hoonete automatiseerimise lahendused on muutumas aina levinumaks, kuna nende kasutamisega saab muuta rutiinsed tegevused aja- ja kuluefektiivsemaks. Hoone automatiseerimisega saab muuta hoone kasutamise mugavamaks kaugjuhtimise või reeglite alusel automatiseerimise abil.

Hoonet automatiseerides on võimalik luua lahendusi, mis ei oleks hoonet käsitsi juhtides praktilised. Näiteks saab seadistada valgusti ereduse olenevalt naturaalse valguse tugevusest, arvutuste kohaselt võib sellise lahenduse abil säästa energiakulu kuni 52% võrra [1].

KNX standard on levinud ja stabiilne hoone automatiseerimise standard. Bakalaureusetöö raames on eeldatud, et kasutaja on otsustanud hoone automatiseerimisel kasutada KNX standardit. KNX lahendusi on keeruline seadistada. KNX lahendusi seadistatakse kasutades tarkvara ETS. ETS-i abil KNX lahenduse seadistamiseks on vaja teadmisi KNX standardist ning tarkvarast ETS. Tavakasutajale võib osutuda liiga keeruliseks KNX lahendusse väikeste ümberseadistuste tegemine isegi siis, kui ekspert on esialgse seadistamise korralikult ära teinud.

Kasutajal võib tekkida näiteks järgmine olukord. Kasutajal on õues neli valgustit. Kasutaja soovib, et talvel õhtul põleksid kõik valgustid, aga suvel põleks ainult kaks valgustit. Hetkel peaks sellise lihtsa muudatuse tegemiseks kasutama ETS-i.

Bakalaureusetöö eesmärk on luua lihtne viis hooneautomaatika loogika ümber seadistamiseks. Võiks pakkuda täiendavaid võimalusi, et loogika ümber seadistamise asemel saaks eelkirjeldatud loogika muudatuse ning muud sarnased olukorrad lahendada automaatselt. Sellel eesmärgil luuakse bakalaureusetöö raames reeglimootor, mis käivitab reegleid kujul kui…siis.

1.1 Ülesanded

Reeglimootori arendamise võib jagada alamülesanneteks järgnevalt.

1. Uurida KNX standardit. Tuleb aru saada, kuidas KNX süsteemi seadistada ning kuidas reeglimootor saaks KNX süsteemi kirjutada ja lugeda.

2. Seada üles lihtne KNX süsteem reeglimootori arendamiseks.

3. Uurida reeglimootorite põhimõtteid.

4. Tuleb määrata reeglite ülesehitus ehk kuidas kasutaja saab määrata reegli tingimused ning tagajärjed.

5. Leida viis edastada reeglimootorile KNX projekti andmed.

6. Otsustada, kuidas toimub reeglite hindamine ja käivitamine.

1.2 Ülevaade tööst

Teises peatükis on seletatud KNX standardi tausta ning arutletud, kuidas reeglimootor saaks juhtida KNX võrku.

Kolmandas peatükis on analüüsitud, millisel kujul kasutaja saaks reeglimootorisse reegleid sisestada. Uuritakse olemasolevaid reeglimootoreid ning reeglimootorite teooriat. Leitakse viis KNX väärtuste võrdlemiste ja muude operatsioonide tegemiseks. Arutletakse, milliseid andmeid on reegliga vaja määrata, et reeglimootor saaks reegleid hinnata.

Neljandas peatükis on käsitletud reeglimootori realisatsiooni. Peatüki raames on lahti seletatud autori nägemus rakenduse lõplikust arhitektuurist. Kirjeldatakse, kuidas reeglimootor suhtleb KNX võrguga. Luuakse ETS projekti andmete importer, et leida KNX väärtuste kasutamiseks vajalik info. Selgitatakse reeglite lisamist. Vaadeldakse reeglite uuendamise protsessi. Räägitakse reeglimootori testimisest.

1.3 Alternatiivsed lahendused

Leidub mitmeid loogikaelemente, mis pakuvad kasutajasõbralikku kasutajaliidest hooneautomaatika lahenduse reeglite seadistamiseks, kuid eeldavad, et kasutaja kasutab KNX-ga mitteühilduvat hooneautomaatika standardit [2].

Peamine meetod KNX lahenduse loogikareeglite seadistamiseks on ETS. ETS on KNX Foundationi poolt loodud tööriist KNX lahenduse seadistamiseks. Kasutajal on vaja detailseid teadmisi KNX standardist ja tarkvarast ETS, et seadistada hooneautomaatika lahenduse loogika vastavalt oma vajadustele. Loogika ümber seadistamiseks on vaja laadida üles kõik seadistuse muudatused igasse seadmesse, mille seadistust muudeti. Keerulisemate loogikareeglite kasutamiseks võib olla vaja eraldiseisvaid loogikaelemente. Arendatava reeglimootoriga peaks olema kasutajal lihtsam loogikat ümber seadistada.

Leidub hooneautomaatika seadistamise tarkvara, mis ei olene tehnoloogiast, järgnevalt on tähelepanu pööratud OpenHAB lahendusele.

OpenHAB-i kasutamiseks on vaja kasutajal luua OpenHAB-i süsteemis virtuaalsed seadmed ja sisestada seadmete andmed. OpenHAB jälgib süsteemis toimuvaid sündmusi. Sündmus võib olla näiteks mõne seadme oleku muutumine, kellaaja muutumine või OpenHAB-i süsteemi käivitumine. Seejärel saab defineerida reeglid. Reeglitel on tingimus ja tulemus. Tingimus on ühe või rohkema sündmuse toimumine, kui üks sündmustest toimub käivitatakse tulemuse blokk. Tulemuseks on Java kood, kus saab kasutada OpenHAB-i süsteemi lisatud seadmeid ning paljusid OpenHAB-i ja Java teeke. Alloleval joonisel on näha OpenHAB-i reegli loomise malli.

```
rule "<RULE_NAME>"
when
    <TRIGGER_CONDITION> [or <TRIGGER_CONDITION2> [or ...]]
then
    <SCRIPT_BLOCK>
end
```

Joonis 1. OpenHAB-i reegel [3].

OpenHAB lahenduse kasutamine eeldab eraldi kasutajaliidese leidmist või koodi kirjutamise oskust [2]. OpenHAB lahendusele loodud kasutajaliideste miinuseks on, et need ei saa teha lihtsustusi KNX standardi jaoks.

2 KNX

Reeglimootori arendamiseks on vaja omada ettekujutust sellest, kuidas KNX standardile vastav lahendus töötab. Oluline on mõista KNX võrgu sisest kommunikatsiooni, sest suhtlus välisvõrguga töötab sama moodi. Järgnevas peatükis selgitatakse KNX standardi põhimõtteid, seatakse üles KNX võrgu näidis ja analüüsitakse, mida need põhimõtted tähendavad reeglimootori kontekstis.

2.1 Põhimõtted

KNX standardile vastav lahendus koosneb paljudest seadmetest. Iga seade peab vastama KNX standardile. Minimaalne KNX lahendus sisaldab ruuterit ja toiteallikat. Ruuter korraldab suhtlust KNX võrgu sees ja suhtleb välisvõrguga. Toiteallikas pakub lahendusele voolu, aga ei ole muul viisil oluline. Seadmetel võib olla mitu funktsiooni. Näiteks võib olla ühte seadmesse kombineeritud liikumisanduri ja temperatuuri lugemise funktsioonid. Seadmetel võivad olla anduri funktsioonid ja täituri funktsioonid. Täitur võib olla näiteks valgusti, mis lülitub sisse vastavalt sisendile.

KNX lahendust seadistatakse läbi spetsiaalse tarkvara ETS. ETS salvestab kogu informatsiooni KNX lahenduse seadistuse kohta projekti. ETS edastab projekti seadistuse igale seadmele eraldi.

Seadmeid saab seadistada läbi ETS-i kasutades parameetreid. Mõned seadmed ei saa oma olekut süsteemis uuendada iga muutuse järel ja uuendavad oma olekut hoopis iga kindla ajavahemiku järel. Ajavahemiku saab määrata läbi parameetrite.

2.2 Näidislahendus

Järgnevalt on kirjeldatud näidisvõrku, millel põhinevad hiljem kasutatavad reeglite näited. Näidisvõrgu seadistamiseks on kasutatud tarkvara ETS5 professionaalse litsentsiga.

Näidisvõrgus on sisend-väljund plokk, lüliti, valgusti, liikumisandur, dimmer, multiandur, KNX ruuter ning toiteallikas. Seadmed, mida saab läbi ETS-i seadistada, on kujutatud järgmisel joonisel.

Addres	Room	Description	Application Program	Adr Prg Par Grp Cfg	Manufacturer
1.1.1	Demo room		MINiBOX 45 1.4	✓ ✓ ✓ ✓ ✓	Zennio
1.1.2	Demo room		REKNT004	✓ ✓ ✓ ✓ ✓	Dinuy
1.1.8	Demo room		Auro KNX app v1.1	✓ ✓ ✓ - ✓	BASALTE
1.1.12	Demo room		Sewi KNX AQS/TH-D L-Pr	✓ ✓ ✓ ✓ ✓	Elsner Elektronik GmbH
1.1.255	Demo room		KNX IP Router 751	✓ ✓ ✓ - ✓	Weinzierl Engineering GmbH

Joonis 2. Näidislahenduses kasutatud seadmed ETS-is.

Kõik seadmed on ühendatud ruuteri ja toiteallika külge. KNX ruuter loob võrgu, mille kaudu KNX seadmed saavad üksteisega suhelda. KNX ruuter on ühendatud kohtvõrku. Läbi kohtvõrgu saab ligipääsu KNX võrgule. Kasutades KNX ruuteri IP (*Internet protocol*, interneti protokoll) aadressi saab KNX võrguga ühenduda, ühenduse saab luua näiteks läbi ETS-i.

Lüliti on ühendatud sisend-väljund plokki, kuna lahenduses kasutatud lülitit ei saa otse KNX võrku ühendada. Sisend-väljund plokk kirjutab lüliti oleku muutumisel uue lüliti oleku süsteemi.

Dimmer on testsüsteemi ainuke täitur. Dimmer on ühendatud valgustiga ning juhib valgusti käitumist. Valgusti saab sisse või välja lülitada binaarse signaaliga. Alternatiivselt saab valgusti intensiivsuse määrata protsendiväärtusega.

Termomeeter loeb perioodiliselt temperatuuri.

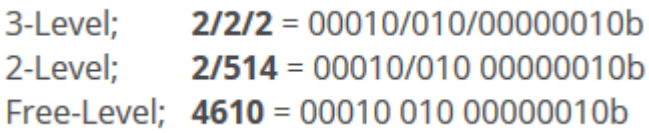
Multianduril on mitu sensori funktsiooni. Multiandur oskab lugeda temperatuuri. Lisaks on multianduril hämarusanduri funktsioon. Valguse intensiivsuse näidu võib saada absoluutse mõõduna või protsendina maksimumist.

2.3 Võrk

KNX seadmete vaheline suhtlus põhineb grupiaadressidel. Seadme funktsioonid saab siduda grupiaadressidega. Täiturid kuulavad seotud grupiaadresse ja käituvad vastavalt grupiaadressile kirjutatud väärtustele. Andurid loevad väärtusi ja kirjutavad grupiaadressidele, millega nad seotud on. Grupiaadressile saadetud sõnumist saab sisend grupiobjektidele, mis seda aadressi kuulavad. Ühel seadmel võib olla nii

grupiobjekte, mis vajavad sisendit, kui ka grupiobjekte, mis saadavad väljundi. Seadme funktsioone seotakse grupiaadressidega läbi ETS-i. Grupiaadressidele saadetavates sõnumites sisaldub uus väärtus, mida nimetatakse grupiväärtuseks.

Grupiaadresse eristatakse üksteisest sõne kujulise aadressi ja nime alusel. Grupiaadresse saab grupeerida vabalt, kahetasemeliselt või kolmetasemeliselt. Grupiaadressi aadressi saab kujutada sõnena, siis eraldatakse iga taseme grupp kaldkriipsuga. Reeglimootor eeldab, et on kasutatud kolmetasemelisi grupiaadresse. Järgneval joonisel on näha näide igast taseme kujutusest. Joonisel on näha ka grupiaadresside binaarset kujutust. Antud grupiaadressi kujutused on sama binaarse grupiaadressi erinevad kujutused.



Joonis 3. Grupiaadresside tasemed [4].

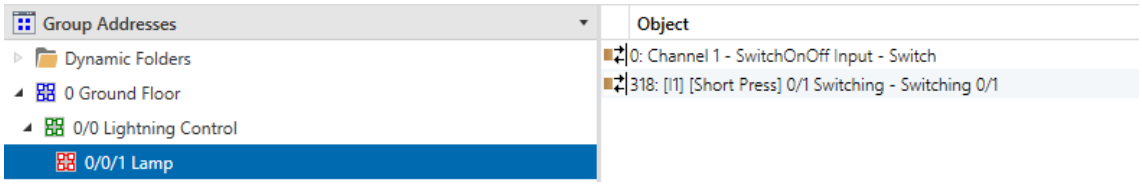
KNX standard defineerib suure hulga andmetüüpe. Paljud nendest tüüpidest on analoogsed programmeerimiskeeltest tuttavatele tüüpidele nagu täisarv, binaarne väärtus, sõne ja ujuvarv. Sellistel üldistel tüüpidel võivad olla alamtüübid rangemate piirangutega. Näiteks 16-bitisel ujuvarvul on alamtüübid Celsius ja Kelvin temperatuuride hoidmiseks. 16-bitine ujuvarv jääb vahemikku -671 088,64 kuni 670 760,96. Selle alamtüüp Celsius jääb vahemikku -273 kuni 670 760 **Error! Reference source not found.**.

Igal grupiaadressil on andmetüüp, Grupiaadressile saab saata ainult seda tüüpi väärtuseid. ETS kontrollib projekti seadistamisel, et kõigil grupiaadressiga ühendatud grupiobjektidel oleks sama andmetüüp.

Sõnumites, mida seadmed üksteisele saadavad, ei ole informatsiooni sõnumis sisalduva väärtuse andmetüübi kohta. Reeglimootoril on vaja ligipääsu ETS projekti andmetele, et aru saada, mis tüüpi KNX võrgust loetud grupiväärtused on.

KNX seadmete vahelise suhtluse illustreerimiseks on toodud lihtne näide grupiaadressist, kus on ühendatud lüliti ja valgusti. Lüliti ja valgusti kasutavad mõlemad binaarset andmetüüpi. Lüliti sisse lülitamine saadab lüliti grupiaadressile sõnumi väärtusega „tõene“ ja lüliti välja lülitamisel saadab lüliti sõnumi „väär“. Lüliti lülitab valguse sisse, kui grupiaadressi väärtus on „tõene“ ja lülitab valguse välja, kui

grupiaadressi väärtus on väär. Joonisel on näha, kuidas ühendada valgusti ja lüliti omavahel kasutades ETS-i.



Joonis 4.Valgusti lülitiga ühendamine kasutades grupiaadresse.

2.4 Järeldused

Kasutades teadmisi KNX võrgust saame esialgse arusaama reeglimootori tööpõhimõtetest. Reeglimootor saab kuulata kõikidele grupiaadressidele saadetavaid sõnumeid. Sõnumite alusel saab reeglimootor otsustada, kas mõne reegli tulemust on vaja käivitada. Reegli tulemuse käivitamine kujutab endast grupiaadressile mõne väärtuse kirjutamist. Järgnevates peatükkides uuritakse, kuidas kasutaja saaks määrata grupiaadressile kirjutatava väärtuse.

Reeglimootoril on vaja, et KNX süsteem oleks õigesti seadistatud. Kõige tähtsam on, et seadme funktsioonid oleks grupiaadressidega seotud. Algne KNX süsteemi seadistamine on lihtkasutajale liiga keeruline. Reeglimootori arendamisel on eeldatud, et mõni ekspert algseadistab KNX süsteemi kasutaja jaoks. Seejärel süsteemi juhtimine reeglimootori abil peaks olema lihtkasutajale juba jõukohane. Seadmete funktsioonide sidumiseks grupiaadressidega on kaks varianti, kas siduda iga grupiaadressiga üks seadme funktsioon või siduda iga grupiaadressiga mitu seadme funktsiooni.

Tavalises KNX süsteemis on iga grupiaadressiga seotud mitu seadme funktsiooni ning omavahel suhtlevad ainult seadmed, mis on samal grupiaadressil. Reeglimootori kasutamine sellises KNX lahenduses on keerulisem. KNX lahenduse suhtlus käib läbi grupiaadresside ning reeglimootori reeglid loevad ja kirjutavad omakorda grupiaadressidelt. ETS-is seadistatud loogika ja reeglimootori loogika ei pruugi olla kooskõlas ja võib tekkida segadus.

Reeglimootori tööks oleks parem, kui KNX lahendus seadistataks nii, et iga seadme funktsioon on eraldi grupiaadressil. Kasutajale on selgem, mida iga reegel teeb. Seadme funktsioonid, mis on eraldi grupiaadressidel, ei suhtle omavahel, aga reeglimootoriga on

võimalik luua reeglid, mis ühendaksid grupiaadressid omavahel virtuaalselt. Sellise seadistusega süsteem toimib ainult koos reeglimootoriga.

Kasutajale on kõige mugavam nende kahe võimaluse vaheline kompromiss. Eelkõige suurt töökindlust vajava loogika saab seadistada läbi ETS-i. Loogika, mida on vaja tihti muuta, saab seadistada läbi reeglimootori. Reeglimootor ja ETS kasutaksid samas süsteemis erinevaid grupiaadresse ja seetõttu ei sega üksteist.

Kasutajaliidese ja programmi lihtsustamiseks on võimalik eeldada, et igal grupiaadressil on üks seade. Siis töötaks kogu suhtlus KNX võrgus läbi reeglimootori. Selline lihtsustus tähendaks, et kasutajal ei ole vaja teada midagi grupiaadressidest ja saab ainult ühendada seadmeid omavahel, mis on intuitiivsem. Seda lähenemist ei ole kasutatud, sest siis ei saaks reeglimootorit kasutusele võtta olemasolevas KNX süsteemis ilma väga suurte muudatusteta KNX seadistuses.

3 Reeglite ülesehituse analüüs

Järgnevas peatükis analüüsitakse, millisel kujul reegleid saaks sisestada ja salvestada. Alustuseks uuritakse olemasolevates reeglimootorites kasutatud põhimõtteid. Edasi leitakse viis väärtuste kujutamiseks reegli sees. Lõpuks kirjeldatakse, millise kujuga reegleid saab reeglimootorisse sisestada ning miks reeglitele selline kuju anti.

3.1 Reeglimootori teooria

Järgnevalt analüüsitakse, kuidas kasutaja saaks defineerida reegli käivitumise tingimuse ja tagajärje. Tuuakse välja reeglite defineerimise programmeerimiskeele abil, domeenispetsiifilise keele abil ning struktureeritud kujul eelised ja nõrkused. Viimasena otsustatakse, millist lähenemist kasutada.

Reeglimootor on programm, milles on salvestatud hulk reegleid kujul kui… siis. Kasutaja defineerib reegli käivitumise tingimuse ja tagajärje.

Käivitumise tingimused või tulemused või mõlemad saaks määrata programmeerimiskeele abil. Näiteks OpenHAB lahenduse puhul saab tingimused määrata rangelt struktureeritud kujul ja tulemust kirjeldada Java koodi kirjutades.

Programmeerimiskeele alusel reeglite kirjeldamise puhul on peamine probleem, et kasutaja peab oskama koodi kirjutada. Lähenemise eelis on, et programmeerimiskeele kasutamine pakub erakordselt palju võimalusi reeglite defineerimisel.

Reeglit saaks kirjeldada domeenispetsiifilist keelt kasutades. Domeenispetsiifiline keel on programmeerimiskeel, mis on disainitud mõne konkreetse probleemi valdkonna jaoks [6]. Üldised programmeerimiskeeled on loodud pidades silmas keele kasutusvõimalust paljudes valdkondades. Domeenispetsiifilised keeled on näiteks HTML, mida kasutatakse dokumentide struktuuri defineerimiseks, ning SQL, mida kasutatakse andmebaasi päringute koostamiseks.

Reeglimootoris reeglite kirjeldamiseks oleks võimalik luua domeenispetsiifiline keel. Domeenispetsiifilise keele loomisega oleks võimalik paremini abstraheerida hooneautomaatika standardite ja lahendust pakkujate vahelised erinevused ühise süntaksi taha ning oleks lihtsam luua kasutajale intuitiivne kasutajaliides [7]. Samas

kaasneksid üldised domeenispetsiifilise keele kasutamisega seonduvad probleemid: kasutajale mugava domeenispetsiifilise keele süntaksi välja töötamine on keerukas ning keele parsimine võib osutuda keerukaks [8].

Reegleid saaks defineerida struktureeritud kujul. Reeglid saaks defineerida andes kasutajale võimaluse kirjeldada reeglite käivitamise tingimus ja tulemus programmile arusaadaval kujul. Reegli tingimusel ja tulemusel oleks range formaat, mida reeglimootor oskaks tõlgendada.

Lähenemise eeliseks on, et kasutajal on võimalik reegleid kirjutada ilma programmeerimisoskuseta ning kasutajaliidese arendamine on suhteliselt lihtne. Lähenemise puudusena saab välja tuua, et ranges formaadis reeglite sisendamine ei paku kasutajale kaugeltki nii palju võimalusi kui programmeerimiskeele või domeenispetsiifilise keele kasutamine.

Lõputöö raames valmiva reeglimootori arendamisel on kasutatud reeglite defineerimist rangelt struktureeritud kujul. Arendatava reeglimootori peamine eesmärk on muuta kasutajale reeglite defineerimine võimalikult lihtsaks. Reeglite defineerimine programmeerimiskeele abil muudaks reeglite kirjeldamise kasutajale liiga keeruliseks. Reeglite defineerimine domeenispetsiifilise keele abil muudaks reeglimootori arendamise liiga keeruliseks. Autori hinnangul on võimalik struktureeritud reeglitega pakkuda kasutajale piisavalt võimalusi, et kasutaja saaks kõik vajalikud reeglid defineerida.

3.2 Avaldised

Kasutajal peab olema viis tingimuse defineerimiseks. Samuti peab olema võimalik leida, mis väärtuse peab tulemusena grupiaadressile kirjutama. Selleks peaks kasutaja saama väärtusi omavahel võrrelda

Eeltingimus peab lõpuks tagastama kas tõese või väära väärtuse. Kasutaja peaks saama vastavalt vajadusele luua eeltingimusi, mille väärtus oleneb KNX süsteemist. Näiteks võib eeltingimuseks olla, et lüliti peab olema sisse lülitatud.

Võib olla vajalik, et reegli käivitumine oleneks mitmest tingimusest. Näiteks soovib kasutaja, et eeltingimus oleks täidetud, kui elutoa lüliti on sisse lülitatud või köögi lüliti

on sisse lülitatud. Tingimusi võib üksteisega ühendada loogiliste „ja“ või „või“ operaatoritega. Näiteks tahab kasutaja, et valgusti lülituks sisse koos liikumisanduriga, aga kui lüliti on välja lülitatud peaks valgusti olema alati välja lülitatud. Teisisõnu, lüliti peab olema sisse lülitatud ja liikumisandur peab andma signaali, et valgusti lülituks sisse. Lisaks võib eeltingimus koosneda veel rohkematest alamtingimustest, mis on kõik ühendatud „ja“ või „või“ operaatoritega. Siis on võimalus kasutada segamini „ja“ ning „või“ operaatoreid. Tulemuse arvutamisel peaks arvestama, et „ja“ operaatoreid peab hindama enne „või“ operaatoreid. Autor otsustas, et mitme tingimuse kasutamine on kasutajale oluline. Reeglimootor võiks toetada ükskõik kui mitut alamtingimust ning võib kasutada nii „ja“ kui ka „või“ operaatoreid.

Reeglimootori lahendust oleks võimalik lihtsustada, kui toetada ainult KNX binaarseid andmetüüpe või toetada ainult kindlaid andmetüüpe. Tõenäoliselt oleks kasutajale kasulik võrrelda näiteks temperatuuri, kellaaja või protsendi põhiseid väärtuseid . Ainult binaarse andmetüübi toetamine lihtsustaks reeglimootori arendamist palju, sest ei oleks vajadust tuvastada KNX süsteemist loetud väärtuste tüüpi, kuna saaks eeldada, et väärtus on binaarset tüüpi. Lisaks oleks võimalik luua lihtsam kasutajaliides. Ainult binaarse andmetüübi toetamine piiraks kasutaja võimalusi väga palju. Võiks toetada binaarset tüüpi ning veel mõnda loetud tüüpi, mille järele on ilmnenud konkreetne vajadus, aga siis ei ole enam võimalik eeldada KNX süsteemist loetud väärtuste tüüpi ning kasutajaliideses ei saaks teha suuremaid lihtsustusi. Seega ei ole reeglimootori keerukuse seisukohalt eriti oluline, kas toetatakse kahte andmetüüpi või kõiki andmetüüpe. Seetõttu otsustas autor toetada kõiki andmetüüpe.

Reeglite loomisel oleks kasulik võimalus võrrelda omavahel erinevaid KNX süsteemi väärtusi või võrrelda KNX süsteemi väärtust kasutaja poolt valitud konstantse väärtusega. Seega eeltingimuses peaks toetama võrratuse funktsioone ehk vähemalt „=“, „<“, „>“, „<=“, „>=“ operaatoreid.

Reegli tulemuse hindamine on üldiselt analoogne reegli eeltingimuse hindamisele, aga reegli tulemuse hindamise lõpptulemuseks peaks olema grupiväärtus.

Tulemuses saaks väärtustega tehteid teha. Näiteks määrata, et lamp põleks veidi eredamalt kui tavaliselt hämarusanduriga põleks. Selliste tehete toetamine ei ole eriti

oluline, aga matemaatiliste tehete toetamine on üsna lihtne, kuna nad on väga sarnased loogilistele operatsioonidele.

Eeltingimus ja tulemus on oma olemuselt väga sarnased, ainuke vahe on, et tulemuse tüüp peaks olema erinev. Seega peaks saama välja töötada lahenduse, mis töötab nii eeltingimuse kui ka tulemuse puhul.

3.2.1 Lahendusvõimaluste võrdlus

Avaldiste hindamiseks on kaks varianti. Saaks hinnata avaldisi, mis on rangel programmile tuttaval kujul. Alternatiivselt saaks hinnata vabateksti kujul avaldisi. Järgnevalt on võrreldud nende kahe lähenemise tugevusi ja nõrkusi ning selgitatud, millise valiku autor tegi.

Andmete hoidmiseks saaks luua avaldise objekti. Objekt peaks sisaldama informatsiooni, kuidas arvutada avaldise lõppväärtus. Avaldises võib olla konstant, KNX väärtus või operatsioon. Konstant on antud kontekstis väärtus, mis ei olene KNX süsteemist. KNX väärtus loetakse reegli hindamisel grupiaadressilt, mis on kirjeldatud KNX väärtuses. Operatsioonis on kirjeldatud operatsiooni tüüp ja nimekiri kasutatavatest väärtustest. Operatsiooni väärtused võivad olla rekursiivsed ehk iga operatsiooni väärtus võib olla konstant, KNX väärtus või operatsioon. Konstant ja KNX väärtus on rekursiooni baasjuhud.

Operatsiooni tüübid võivad olla näiteks võrdus, suurem kui, liitmine, lahutamine ja nii edasi olenevalt andmetüübist. Iga väärtuse tüübi peab leidma enne, kui seda on võimalik operatsioonis kasutada. Konstandi puhul on tüübi leidmine lihtne läbi süsteemi teekide. KNX väärtuse tüübi saab leida projekti andmeid kasutades. Reeglimootorisse saaks sisse kirjutada, mis tüüpi väärtuse iga operatsiooni tüüp tagastab.

Lahenduse puudusteks on arendamise keerukus ja kohmakas kuju. Sellisel kujul on avaldiste hindamisel liiga spetsiifilised nõuded, et teeke kasutada. Iga operatsiooni peab iga tüübi kohta ise implementeerima.

Avaldisi saaks hoida vabateksti kujul. Avaldis oleks sõne, kuhu saab kirjutada matemaatilisi avaldisi. Näiteks “1 + 1” või “true or false”. Sõne alusel matemaatilise avaldise hindamise jaoks on võimalik leida teeke. Autor eelistas avaldise hindamise jaoks mõeldud teeke otsides teeki *Flee* (*Fast Lightweigh Expression Evaluator*).

KNX väärtusi saaks avaldisse sisestada; kas funktsioone kasutades või muutujaid kasutades. Saaks luua funktsiooni, millele antakse parameetriga kaasa grupiaadress. Funktsiooni oleks võimalik avaldise sõnes kasutada. Avaldise hindamisel hinnataks funktsiooni ja see käituks nagu tavaline väärtus. Alternatiivselt saaks avaldisse anda kaasa nimekirja muutujaid. Muutujal oleks nimi ja grupiaadress. Enne avaldise hindamist, hindaks reeglimootor kõik muutujad ja asendaks muutuja nime kasutused avaldise sõnes vastavate väärtustega. Kasutades funktsiooni lähenemist ei teaks reeglimootor, kuidas KNX väärtusi avaldises kasutatakse ilma reeglimootorile iseseisva parseri arendamiseta. Muutujate põhise lähenemise puhul oleks teada KNX väärtused, mida tohib avaldises kasutada, aga ei oleks teada, kas kõiki neid muutujaid avaldises tegelikult kasutati. Reeglimootorile iseseisva parseri kirjutamine nõuaks liiga palju tööd. Muutujate põhist lähenemist kasutades oleks reeglimootoril rohkem informatsiooni kasutatud KNX väärtuste kohta, mis oleks oluline reeglite esmasel valideerimisel ning reeglite käivitamisel.

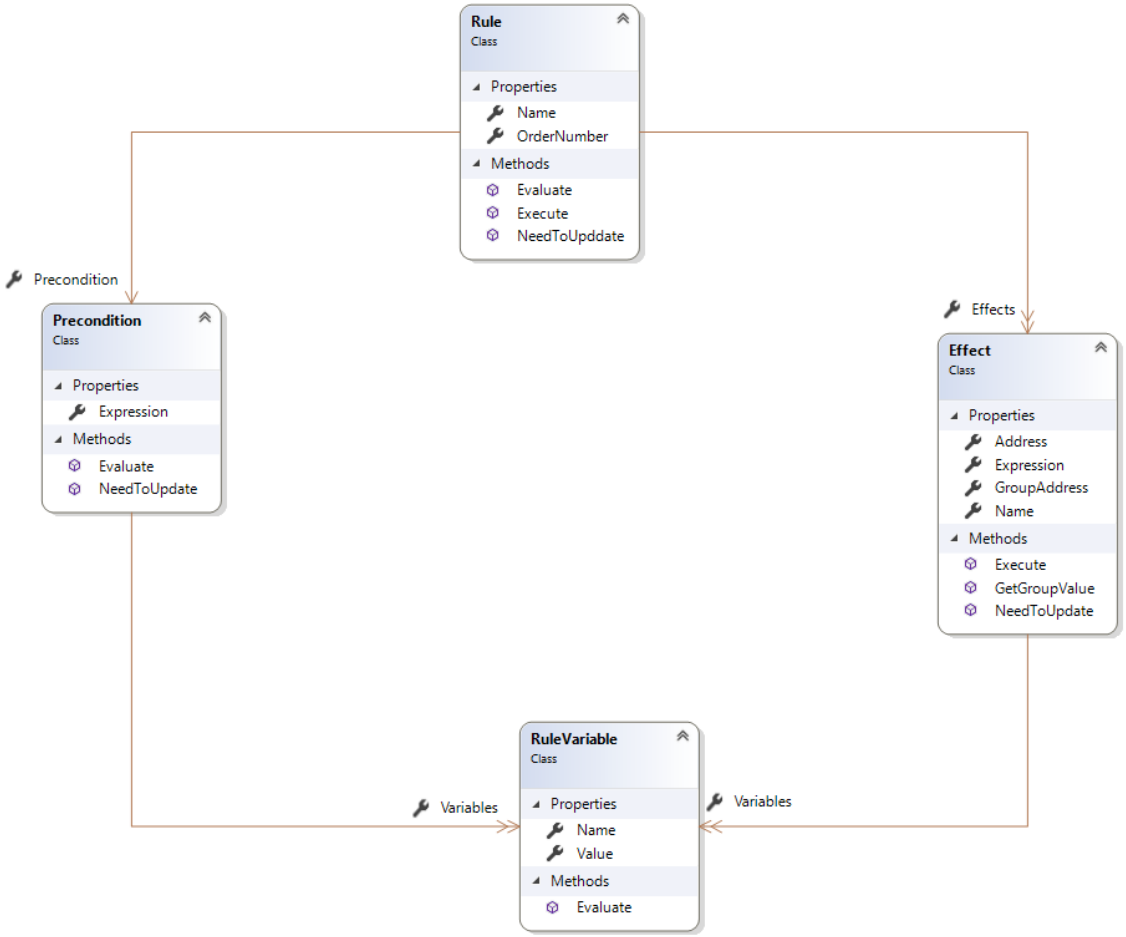
Lahenduse peamine puudus on ebatäielik informatsioon KNX väärtuste kasutamise osas isegi kui kasutada muutujate põhist lähenemist. Struktureeritud lahenduse puhul on reeglimootoril kogu informatsioon selle kohta milliseid KNX väärtuseid loetakse ja mida nendega edasi tehakse. Vabateksti puhul ei ole teada, kuidas KNX väärtuseid kasutatakse.

Avaldise hindamiseks peab sõnesid töötlema, struktureeritud lahenduse puhul on kõik andmed juba hindamiseks sobilikul kujul. Oletatavasti on struktureeritud avaldise hindamisel veidi parem jõudlus. Autori hinnangul on jõudluse kadu piisavalt väike, et see ei häiriks kasutajat, seetõttu on eelistatud avaldise sisestamist vabatekstina.

Autor otsustas, et avaldisi peaks saama sisestada vabateksti kujul. Struktureeritud avaldiste hindamise süsteemi arendamine oleks palju keerulisem, kuna ei ole võimalik kasutada teeke.

3.3 Reeglid

Vajalik on vaja otsustada, kuidas hoida reeglite eeltingimusi ja tulemusi ning muid toetavaid andmeid. Joonisel on näha reegli ülesehitus koodis.



Joonis 5. Reegli klassidiagramm.

Reeglil on üks eeltingimus, kuna eeltingimuse avaldisse saab panna ükskõik palju tingimusi, liites need „ja“ või „või“ operaatoritega, siis ei ole vajadust toetada mitut eeltingimust.

Tulemusi võib olla mitu. Juhul, kui eeltingimus on tõene, täidetakse kõik tulemused. Reeglimootoril ei ole hädavajalik toetada mitut tulemust. Ainult ühe tulemuse toetamisel saaks kasutaja luua mitu reeglit sama eeltingimusega, et kõik tulemused saaksid täidetud. Hetkel on otsustatud toetada mitut tulemust, sest ühele reeglile mitme tulemuse lisamine on kasutajasõbralikum.

Lisaks eeltingimusele ja tulemustele on reeglitel nimi ja järjekorranumber. Nime on vaja, et kasutaja saaks kergemini aru, mida reegel teeb, nimel ei ole programmi sees funktsiooni. Nimi on kohustuslik reegli salvestamisel kaasa anda ja peab olema unikaalne, sest isegi halvasti nimetatud reegel aitab kasutajal reeglite üle järge pidada. Kui mitme reegli tulemused lähevad omavahel konflikti, siis otsustatakse

järjekorranumbri alusel, millised reeglid käivitatakse. Järjekorranumber peab olema unikaalne, muidu on reeglite täitmise järjekord ebamäärane.

Eeltingimuses on avaldis vabateksti kujul ning muutujad.

Tulemuses on avaldis vabateksti kujul, muutujad ning grupiaadress, kuhu avaldise lõpptulemus kirjutada. Tulemusel võib olla nimi, aga erinevalt reegli nimest ei ole tulemusele nime panemine kohustuslik. Tulemused on vähem keerulised kui reeglid, seetõttu on tulemustel lihtsam vahet teha.

Muutujad on eraldi iga eeltingimuse ja tulemuse küljes, mitte üldiselt reegli kohta. Muutujate eraldi hoidmine on kasulik, sest eeltingimusi ja tulemusi on tihti vaja eraldi hinnata, selleks tuleb kõik avaldises kasutatud muutujad hinnata. Juhul, kui muutujad oleks reegli küljes, peaks ühe eeltingimuse või tulemuse hindamisel leidma kõikide muutujate väärtused või peaks kuidagi aru saama, milliseid muutujaid kasutati konkreetses eeltingimuses või tulemuses. Ainuke viis aru saada, mis muutujaid mõni eeltingimus või tulemus kasutab, oleks sõne töötlemine väljaspool avaldise hindamise teeki. Autor soovib avaldise sõne töötlemise jätta võimalikult palju avaldise hindamise teegi hooleks.

Järgnevalt on toodud näide lihtsast reeglist, mis lülitab sisse valgusti, kui lüliti sisse lülitatakse.

```
{
    "Name": "Valgusti lülitiga",
    "Precondition": {
        "Variables": [
            {
                "Name": "lüliti",
                "Value": "0/0/1"
            }
        ],
        "Expression": "lüliti"
    },
    "Effects": [
        {

 "Variables": [],
            "Expression": "100",
            "Address": "0/0/4"
        }
    ]
}
```

Joonis 6. Lihtne reegel JSON kujul.

Reegli eeltingimus kontrollib, et lüliti on sisse lülitatud. Kui lüliti on sisse lülitatud, siis kirjutatakse valgusti grupiaadressile „0/0/4“, et valgusti peaks põlema 100% eredusega ehk täisvõimsusel.

Saab luua reegleid, mille eeltingimus on alati tõene. Alati tõeste reeglitega, mille tulemus suunatakse teisele grupiaadressile, saab luua virtuaalsed grupiaadresside ühendused. Seadmete funktsioonid on ühendatud sama moodi, kui nad oleksid ühe grupiaadressiga seotud. Selline võimalus on kasulik, sest reeglimootori reegleid on lihtsam luua, kui igal seadme funktsioonil on eraldi grupiaadress. Seadmed üksteise vahel ei suhtle, kui kõik funktsioonid on erinevatel aadressidel. Järgnevalt on esitatud näide sellisest reeglist, millega luuakse grupiaadresside vahel virtuaalsed ühendused, et seda piirangut vältida.

```
{
    "Name": "Lüliti ümber suunamine",
    "Precondition": {
        "Variables": [],
        "Expression": "true"
    },
    "Effects": [
        {
            "Variables": [
                {
                    "Name": "lüliti",
                    "Value": "0/0/2"
                }
            ],
            "Expression": "lüliti",
            "Address": "0/0/3"
        }
    ]
}
```

Joonis 7. Reegliga grupiaadressi tulemuse ümber suunamine.

Näites on loodud reegel, mis loeb lüliti grupiaadressilt lüliti seisu ning kirjutab tulemuse valgusti grupiaadressile. Tulemus on sama kui lüliti ja valgusti otse grupiaadresside kaudu ühendamine.

Sarnaselt saab luua reegli, mille eeltingimus on alati väär. Selliseid reegleid ei käivitata kunagi, seetõttu on need kasutud. Võimalusel eelistaks autor keelata alati väära eeltingimusega reeglite sisestamise. Kahjuks on selliste reeglite keelamine hetkel liiga keeruline. Eeltingimuse avaldisele võib anda keerulise kuju, mis on alati väär, nagu järgnevas näites.

```
{
    "Name": "Alati väär",
    "Precondition": {
        "Variables": [
            {
                "Name": "lüliti",
                "Value": "0/0/1"
            }
        ],
        "Expression": "lüliti and not lüliti"
    },
    "Effects": [
        {
            "Variables": [],
            "Expression": "0",
            "Address": "0/0/4"
        }
    ]
}
```

Joonis 8. Reegel, mis kunagi ei käivitu.

Reegli avaldises kasutatakse KNX võrgu väärtust, aga sellisel viisil, et tulemus saab olla ainult väär. Kontrollitakse, kas lüliti on samal ajal nii sisse lülitatud kui ka välja lülitatud. Selgelt sellist olukorda ei saa kunagi juhtuda ning seetõttu ei käivitata kunagi reegli tulemust.

Näites toodud avaldise saaks lihtsustada avaldiseks „false“. Avaldise lihtsustamine võimaldaks välistada alati väärade eeltingimustega reegleid. Avaldise hindamise teek ei paku võimalust selliselt lihtsustada ning autor hindas, et avaldise lihtsustamise arendamine oleks liiga keeruline. Seetõttu saab praegu sisestada reegleid, mis kunagi ei käivitu.

4 Reeglimootori realisatsioon

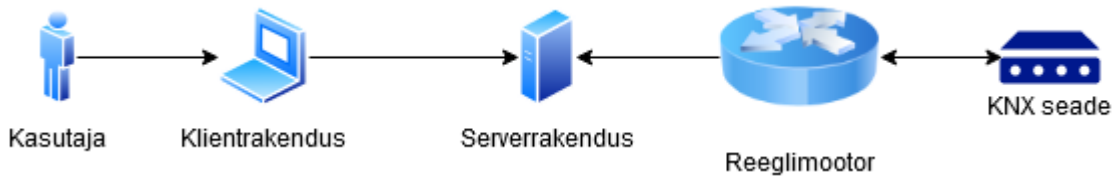
Lõputöö raames luuakse reeglimootor, mis juhib suhtlust KNX võrgus läbi grupiaadresside. Esimese sammuna leitakse viis ühendada reeglimootor KNX võrguga ja KNX võrku sõnumeid saata. Arendatakse viis reeglimootorisse projekti andmed sisestada. Järgnevalt otsustatakse, kuidas kasutaja saab reegleid reeglimootorile edastada. Kirjeldatakse, kuidas käib reeglite uuendamine.

Eelnevate peatükkide põhjal saab esitada arendatavale reeglimootorile peamised nõuded. Reeglimootor peab salvestama reegleid kujul kui…siis ja oskama nende alusel KNX lahendust juhtida. Kasutaja peab saama reegli tingimuste ja tulemuste defineerimiseks kasutada keerukaid avaldisi. Reeglimootor peaks toetama kõiki KNX tüüpe.

Rakenduse arendamiseks on kasutatud programmeerimiskeelt C# ja .NET Core 3.0 raamistikku. C# programmeerimiskeel on valitud, sest autor on C#-ga tuttav. Rakendus kasutab .net Core-i mitte .net-i, kuna kõik vajalikud teegid toetavad .net Core-i ning võimalus käivitada rakendus lihtsasti Linux või Mac operatsioonisüsteemis võib osutuda kasulikuks.

4.1 Arhitektuur

Reeglimootori prototüüpi on kavas kasutada koos klientrakenduse ja SaaS (*Software as a service*, tarkvara teenusena) serverrakenduse komponentidega täieliku reeglimootori lahenduse loomiseks. Lõputöö raames valmis ainult reeglimootori prototüüp. Järgnevalt on läbi mõeldud terviklik lahendus, et mõista, kas reeglimootori prototüüp sobib suuremasse pilti. Joonisel on näha reeglimootori lõpliku lahenduse arhitektuuri visualisatsioon.



Joonis 9. Reeglimootori lõpliku lahenduse arhitektuuri visualisatsioon.

Klientrakendus on veebisait, mis suhtleb ainult serveriga. Klientrakenduse kaudu saaks klient reegleid sisestada ja reegleid hallata. Kasutajale on mugavam suhelda serveriga, sest sellele on läbi interneti lihtsam ligipääs.

SaaS serverrakenduses hoitakse ja hallatakse hoonete andmeid. Ühel hoonel peaks olema üks reeglimootor. Iga hoone kohta võib olla mitu kasutajat. Serverrakendusel peaks olema andmebaas. Kavas on hoida võimalikult palju funktsionaalsust serveris ja võimalikult vähe funktsionaalsust reeglimootoris, sest serverrakendust on lihtsam uuendada ning seal saab hallata keskses kohas kõikide klientide andmeid.

Reeglimootor peab olema võimeline vähemalt reegleid hindama ja KNX võrguga suhtlema. Üks reeglimootor töötab ainult ühe konkreetse hoonega. Reeglimootor on üles seatud hoone kohalikku võrku, mille kaudu on ligipääs KNX võrgule. Reeglimootor ei pruugi olla välisvõrgust lihtsasti kättesaadav. Reeglimootor peaks küsima kõik vajalikud andmed serverrakenduse käest, aga samas võiks suuta olemasolevaid reegleid hinnata ilma internetiühenduseta. Selle jaoks peaks reeglimootor salvestama piisavalt andmeid. Eesmärk on hoida salvestatavate andmete hulk väike ja vältida vajadust andmebaasi järele. Hetkel salvestab reeglimootor andmed ainult mällu.

4.2 KNX võrguga ühendumine

KNX võrguga suhtlemiseks on vaja leida võimalus, mis abstraheerib võimalikult palju KNX-ga suhtlemise detaile.

Reeglimootor kasutab teeki *Falcon SDK*. Teeki *Falcon SDK* on eelistatud põhjusel, et teek on loodud sama organisatsiooni poolt, mis haldab KNX standardit. Autor eeldab, et seetõttu on teek töökindlam kui teised valikud.

Teek võimaldab kuulata pealt KNX-i võrku ning lugeda ja kirjutada grupiväärtuseid. *Falcon SDK* abil on võimalik konverteerida grupiväärtused tavalisteks objektideks, aga selleks on vaja ette anda tüüp, milleks grupiväärtus konverteerida.

KNX-ga on võimalik ühendus luua kasutades tunnelühendust või ruutimist. Reeglimootori arendamisel on eelistatud kasutada tunnelühendust turvalisuse ja töökindluse kaalutlustel [9].

Grupiväärtuse lugemiseks on kasutusel timeout väärtusega 500 millisekundit. Väärtus võiks olla võimalikult väike, et vastuse puudumise korral reeglimootori tööd vähem segada. Autori kogemuse põhjal on 500 millisekundit sobilik väärtus, sest see on piisav aeg, et seadmed peaksid suutma vastata ja see ei häiri reeglimootori tööd.

Reeglimootori arendamise lihtsustamiseks on loodud *mock* KNX liides. Arendades ei ole alati ligipääsu KNX süsteemile. Seega oleks reeglimootori arendamisel abiks virtuaalne liides, mis käitub võimalikult sarnaselt reaalsele KNX võrgule, aga ei vaja ühendust füüsilise KNX süsteemiga.

Mock liidesele saab väärtuseid kirjutada ja lugeda sarnaselt KNX võrgule. Erinevalt KNX võrgust ei oska *mock* liides grupiaadressi väärtuse muutumise sündmusi luua. Liides hoiab taustal nimekirja grupiaadressidest ja nende viimastest väärtustest. Selleks, et muuta mõne liidese grupiaadressi väärtust, saab saata läbi API (*Application programming interface*, rakendusliides) kirjutamise päringu. Kirjutamise päring käitub sama moodi kui grupiaadressi kuulamine. Lugemiseks saadetakse API päring koos soovitava grupiaadressiga. Liides tagastab grupiaadressile viimasena salvestatud väärtuse.

Mock liideselt saab väärtuseid pärida läbi GET tüüpi API päringu. Päringus saab määrata grupiaadressi, millelt lugeda. URL-s ei saa kasutada kaldkriipse, seetõttu grupiaadressi määramiseks tuleb API päringus asendada grupiaadressis kaldkriipsud sidekriipsudega. Peale päringu kätte saamist asendab liides sidekriipsud uuesti kaldkriipsudega. Päring tagastab grupiaadressi hetke väärtuse. Tagastatav väärtus konverteeritakse KNX tüübist süsteemi tüübiks ja esitatakse lõpuks sõne kujul. Järgnevalt on välja toodud väärtuse päringu koostamise mall.

```
GET /api/groupaddress/{groupAddress}
Response: ""
```

Joonis 10. Virtuaalse grupiaadressi väärtuse pärimine.

Mock liidesele saab väärtuseid kirjutada POST tüüpi API päringuga. Kirjutamise päring taaskasutab reegli tulemuse andmetüüpi. Kirjutamise päring on seega analoogne reeglile, mis alati õnnestub, aga käivitub ainult üks kord. Päring tagastab binaarse väärtuse, kui kirjutamine õnnestus tagastatakse väärtus „tõene“, kui mitte, tagastatakse väärtus „väär“.

```
POST /api/groupaddress
Body:
{
    "Variables": [
        {
            "Name": "",
            "Value": ""
        }
    ],
    "Expression": "",
    "Address": ""
}
Response: true
```

Joonis 11. Virtuaalsele grupiaadressile kirjutamine.

4.3 Projekti andmed

Selleks, et oleks võimalik konverteerida süsteemi tüüpide ja KNX tüüpide vahel, on reeglimootoril vaja teada grupiaadresside andmetüüpe. See informatsioon on kättesaadav läbi ETS projekti.

4.3.1 Analüüs

Vajaliku info saaks sisestada käsitsi läbi kasutajaliidese ja API ning need andmed salvestada. Andmete sisestamine on vajalik ainult KNX-i võrgu algsel seadistamisel või ümberseadistamisel.

Andmed saaks importida kasutades ETS-i projektifaili. Kõik vajalikud andmed on olemas projektifailides sisalduvates XML (*Extended markup language*, keel andmete hoidmiseks) failides. Lisaks jääb reeglimootorile võimalus salvestada kogu projektifail. Kogu projektifaili salvestamine on kasulik, kui tulevikus ilmneb vajadus täiendavate

andmete järele. Oleks võimalik täiendavad andmed importida projektifailis ja puuduks vajadus kasutajat häirida.

Grupiaadressi andmetüübi saaks kaasa anda iga kord, kui reeglil on vaja kirjutada või lugeda grupiaadressilt. Seadmete ja kasutuses olevate grupiaadresside informatsiooni ei ole ilmtingimata vaja ja selle lahenduse puhul need puuduksid.

Alati grupiaadressi andmetüübi kaasa andmine oleks kasutajale väga koormav ja lisainformatsiooni puudumine on tõsine probleem, aga see lahendus oli kasulik vaheetapina parema lahenduse välja töötamiseks. Käsitsi sisestatud andmete salvestamine oleks parem võimalus. Andmete automaatne import oleks kõige parem lahendus, kuna see nõuab minimaalselt käsitööd. Korraliku API ja kasutajaliidese välja töötamine võib olla keerulisem, kui andmete importimine XML failidest.

Projektifaili import on realiseeritud reeglimootori osana, aga tulevikus võiks projektifaili import olla serverrakenduse funktsioon. Serverrakendus võiks täita võimalikult palju ülesandeid ja reeglimootorile võiks jätta võimalikult vähe ülesandeid. Reeglimootor saaks serverrakenduselt pärida vajalikud projekti andmed.

Hetkel peab projektifaili peale iga reeglimootori taaskäivitamist uuesti importima. Tulevikus peaks andmed salvestama. Kui projekt andmed on korra juba salvestatud ning projektifaili imporditakse uuesti, siis importer peaks aru saama, kas projektifail on vahepeal muutunud. Seadmeid või grupiaadresse võib kustutada või lisada ning nende andmeid saab muuta.

4.3.2 Realisatsioon

Projektifaili on võimalik importida läbi API.

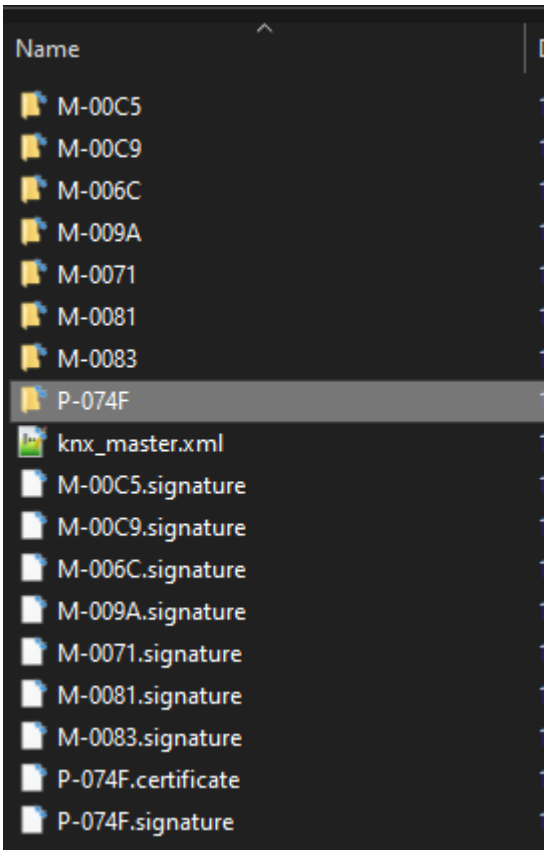
```
POST /api/project
Body:
[
    Files: ""
]
Response: true
```

Joonis 12. Projekti impordi API päring.

API päringule tuleb kaasa anda projektifail. Import õnnestub ainult siis, kui on kaasa antud täpselt üks projektifail. Tulemusena tagastab API päring binaarse väärtuse. Kui

importimine õnnestus, tagastatakse väärtus „tõene“, kui importimisel tekkis viga, tagastatakse väärtus „väär“.

Projektifaili importimise esimese sammuna on vaja ZIP arhiiv lahti pakkida. ZIP arhiivi sisu on järgmine.



Joonis 13. ETS projektifaili sisu.

Reeglimootori jaoks on olulised ainult kaustad prefiksiga „P-“ ning „M-“ . Prefiksiga „P-“ algavates kaustades on projekti andmed. Prefiksiga „M-“ algavates kaustades on seadmete andmed. Muud failid ei ole reeglimootori jaoks olulised. Projekti kaustasid võib väga keeruliste projektide korral olla mitu. Reeglimootori import hetkel selliseid projektifaile ei toeta.

Projekti kaustas on iga installatsiooni kohta eraldi fail n.xml, kus n on projekti id number. Installatsioone on tavaliselt üks **Error! Reference source not found.**. Reeglimootori import toetab ainult ühte installatsiooni.

4.3.3 Grupiaadresside import

Projektifailis on vaja leida grupiaadresside info. Installatsiooni failist võib otsida nime alusel elementi „GroupAddresses“. Edasi on võimalik leida selle alamelemendid „GroupRange“. Iga aadressi taseme kohta on üks kiht „GroupRange“ elemente. Igal „GroupRange“ elemendil on nimi ning lubatud grupiaadresside numbrite algus ja lõpp. Iga taseme nimi tuleks salvestada.

XML failis on grupiaadressi aadress salvestatud ühe numbrina. Reeglimootoril oleks vaja teada grupiaadressi kolmekohalist kuju. Selle kuju leidmiseks on kõigepealt vaja sisse lugeda elemendi „GroupAddress“ atribuut „Address“. Vaja on teada iga „GroupRange“ elemendi positsiooni „GroupRanges“ elemendi sees. Kolmekohalise aadressi puhul käib aadressi leidmine järgmiselt. Oletame, et grupiaadress on kujul x/y/z. Väärtus x on esimese taseme „GroupRange“ elemendi indeks alates nullist, mille sees on otsitav grupiaadress. Sarnaselt väärtus y on esimese taseme „GroupRange“ elemendi indeks, mille sees on otsitav grupiaadress. Väärtuse z leidmiseks tuleb elemendi „GroupAddress“ parameetri „Address“ väärtusest lahutada elemendi „GroupRange“, mille sees element „GroupAddress“ on, parameetri „RangeStart“ väärtus ning liita 1. Järgneval joonisel on näha elemendi „GroupAddresses“ struktuur.

```
<GroupAddresses>
    <GroupRanges>
        <GroupRange  Id="P-074F-0_GR-1" RangeStart="1" RangeEnd="2047"
                     Name="Ground Floor" Puid="28">
            <GroupRange  Id="P-074F-0_GR-2" RangeStart="1" RangeEnd="255"
                         Name="Main" Puid="72">
                <GroupAddress  Id="P-074F-0_GA-1" Address="1" Name="Switch"
                               DatapointType="DPST-1-1" Puid="73" />
                <GroupAddress  Id="P-074F-0_GA-2" Address="2" Name="Switch 2"
                               DatapointType="DPST-1-1" Puid="74" />
                <GroupAddress  Id="P-074F-0_GA-3" Address="3"
                               Name="Lamp boolean"
                               DatapointType="DPST-1-1" Puid="75" />
                <GroupAddress  Id="P-074F-0_GA-4" Address="4"
                               Name="Lamp relative"
                               DatapointType="DPST-5-1" Puid="76" />
                <GroupAddress  Id="P-074F-0_GA-5" Address="5" Name="Motion"
                               DatapointType="DPST-1-1" Puid="77" />
                <GroupAddress  Id="P-074F-0_GA-6" Address="6"
                               Name="Brightness in %"
                               DatapointType="DPST-5-1" Puid="78" />
                <GroupAddress  Id="P-074F-0_GA-7" Address="7"
                               Name="Temperature"
                               DatapointType="DPST-9-1" Puid="79" />
            </GroupRange>
        </GroupRange>
    </GroupRanges>
</GroupAddresses>
```

Joonis 14. Grupiaadresside andmed projektifailis.

Näiteks, kui teha aadress 6 kolmetasemeliseks, siis esimese taseme „GroupRange“ on esimeses positsioonis ehk x väärtus on 0. Teise taseme „GroupRange“ on samuti esimeses positsioonis ehk y väärtus on 0. Lahutades aadressist 6 „GroupRange“ elemendi „RangeStart“ väärtuse 1, leiame, et z väärtus on 6. Kolmetasemeline grupiaadress on 0/0/6. Tulevikus peaks importeri võimalusi laiendama ja toetama kahetasemeliseid ning vaba tasemiliseid grupiaadresse.

Grupiaadresside „Puid“ element on vaja salvestada. Elemendi väärtus on projekti raames unikaalne, seega saab seda kasutada, et tunda projekti mitmekordsel importimisel ära juba salvestatud grupiaadressid. Grupiaadressi andmetüübi leiab parameetrist „DatapointType“, kui eemaldada prefiks „DPST“. Selle koodi alusel saab tuvastada KNX andmetüübi. Näiteks „DPST-1-1“ ehk KNX andmetüüp põhigrupiga 1 ja alamgrupiga 1 on binaarset tüüpi.

4.3.4 Seadmete import

Järgmisena on vaja projektifailist sisse lugeda projektis kasutatud seadmed. Seadmed leiab „DeviceInstance“ elementidest. Seadme andmete import on arendatud kasutajaliidese loomise ettevalmistusena. Praegune reeglimootor seadmete andmeid ei kasuta. Seadmete infost on vaja toote ID numbrit, et selle abil leida seadme tooteinfo. Lisaks on vaja salvestada seadme „Puid“.

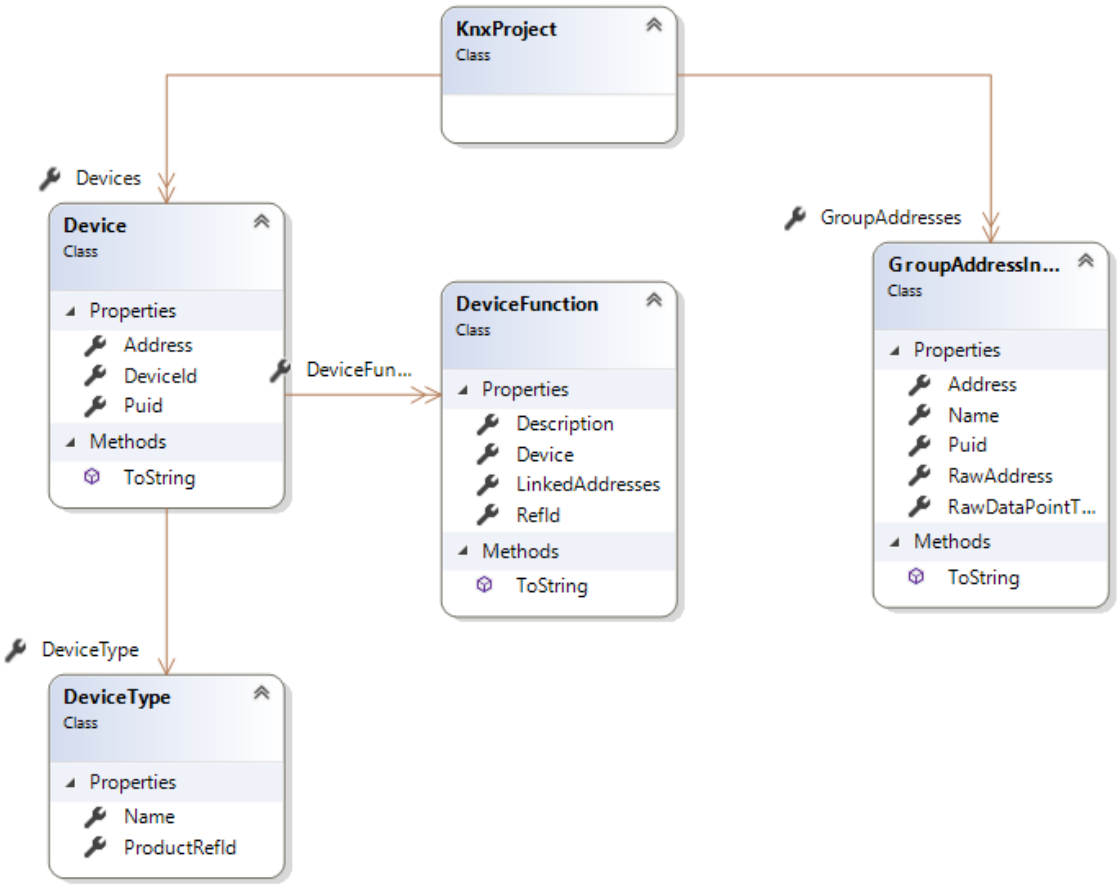
Edasi loeme sisse seadme funktsioonide nimed ja grupiaadressid, millega seadme funktsioonid seotud on. Seotud grupiaadressid on vaja leida läbi grupiobjektide ehk esimesena on vaja viidet grupiobjektidele. Leides elemendid „Node“, saab kätte seadme funktsioonid. Seadme funktsioonide elemendid ei pruugi olemas olla. Viited grupiobjektidele on „Node“ elementide atribuudi „GroupObjectInstances“ väärtuses.

Kasutades seadme funktsioonidest loetud grupiobjektide viiteid saab kokku panna seadme funktsioonid grupiaadressidega. Elemendid „ComObjectInstanceRef“ saab kokku panna seadme funktsioonidega läbi atribuudi „ChannelId“. Grupidaadressid saab panna kokku elemendi „ComObjectInstanceRef“ parameetriga „Links“, kus number prefiksi „GA-“ järel on sama kui elemendi „GroupAddress“ atribuut „Address“.

Seadme tooteinfo leidmiseks otsitakse projektifailist kõik failid nimega „Catalog.xml“. Nendest failidest otsitakse element „CatalogItem“ ning loetakse sisse selle atribuudid „ProductRefId“ ehk toote id ning „Name“ ehk toote nimi. Toote id abil seotakse toote info seadmetega.

4.3.5 Tulemused

Projektifaili impordi tulemusena on reeglimootoris nimekiri projektis kasutatavatest grupiaadressidest ja seadmetest. Nimekirja grupiaadressidest ja seadmetest hoitakse „KnxProject“ klassis. Järgneval joonisel on kujutatud projektifaili impordi tulemuse struktuuri.



Joonis 15. Projektifaili impordi tulemuse klassidiagramm.

Grupiaadressi kohta on salvestatud aadress samal kujul nii kolmetasemeliselt kui ka ühe numbrina nagu failis, grupiaadressi identifikaator projekti raames, grupiaadressi nimi koos eelnevate tasemete nimedega ning grupiaadressi andmetüüp.

Seadme kohta on salvestatud seadme tüübi identifikaator failis, seadme identifikaator projekti raames, seadme tüüp ning nimekiri seadme funktsioonidest.

Seadmetüübi nimi ja identifikaator on salvestatud. Seadmetüübi identifikaatori abil saab leida, mis tüüpi mõni seade on.

Seadme funktsioonide kirjeldus ja viide grupiaadressile on salvestatud. Lisaks on salvestatud seadme funktsiooni omav seade ning nimekiri grupiaadressidest, mis kasutavad seadme funktsioone.

4.4 Reeglite lisamine

Reegleid lisatakse läbi API. Lisamise päringule antakse kaasa reegli andmed sellisel kujul nagu eelnevas analüüsis otsustatud. Järgnevalt on kirjeldatud reegli lisamise

protsessi ja reegli valideerimist. Alloleval joonisel on kujutatud ühe reegli lisamise päringu malli.

```
POST /api/rule
Body:
{
    "Name": "",
    "Precondition": {
        "Variables": [
                {
                    "Name": "",
                    "Value": ""
                }
        ],
        "Expression": ""
    },
    "Effects": [
        {
            "Variables": [
                {
                    "Name": "",
                    "Value": ""
                }
            ],
            "Expression": "",
            "Address": ""
        }
    ]
}
Response: ""
```

Joonis 16. Reegli lisamise päring.

Reeglite lisamine eeldab, et projektifail on sisse tõmmatud. Juhul, kui üritatakse reeglit lisada, aga projekti andmeid ei ole veel reeglimootorisse sisestatud, antakse veateade ja reeglit ei lisata. Projekti andmed on vajalikud reegli valideerimiseks.

Tulevikus peaks reeglite salvestamine olema peamiselt serverrakenduse vastutus. Reeglimootor peaks serverrakenduselt vajadusel projektiga seotud reegleid pärima. Sedasi on kasutajal võimalik reegleid lisada kodust väljas, sest reeglimootorile on ligipääs ainult läbi kohaliku võrgu.

4.4.1 Valideerimine

API päringu kätte saamisel kontrollib .NET Core automaatselt, et kaasa antav objekt vastaks reegli ülesehitusele. Seejärel valideeritakse muud reeglite korrektsuse nõuded.

Juhul, kui valideerimine õnnestub, lisatakse reegel salvestatud reeglite hulka ning reeglile antakse vaikimisi järjekorranumber. Valideerimise ebaõnnestumisel antakse veateade ja reeglit salvestatud reeglite hulka ei lisata.

Reeglite valideerimisel kontrollitakse järgmiseid nõudeid.

```
private RuleSaveResultCode Validate(Rule rule)
{
    if (rule == null)
        return RuleSaveResultCode.RULE_NULL;

    if (KnxConnectionManager.Project == null)
        return RuleSaveResultCode.PROJECT_MISSING;

    if (rule.Precondition == null
        || string.IsNullOrEmpty(rule.Precondition.Expression))
        return RuleSaveResultCode.PRECONDITION_MISSING;

    if (rule.Effects == null || !rule.Effects.Any())
        return RuleSaveResultCode.EFFECTS_MISSING;

    if (string.IsNullOrEmpty(rule.Name))
        return RuleSaveResultCode.MISSING_NAME;

    if (_rules.Any(x => x.Name == rule.Name))
        return RuleSaveResultCode.DUPLICATE_NAME;

    if (rule.OrderNumber.HasValue
        && _rules.Any(x => x.OrderNumber == rule.OrderNumber))
        return RuleSaveResultCode.DUPLICATE_ORDER_NUMBER;

    if (HasDuplicateEffect(rule))
        return RuleSaveResultCode.DUPLICATE_EFFECT_WRITE_ADDRESS;

    if (!AreVariablesDefined(rule.Precondition.Variables)
        || rule.Effects.Any(x => !AreVariablesDefined(x.Variables)))
        return RuleSaveResultCode.UNDEFINED_VARIABLE;

    if (HasVariableReferencingWriteAddress(rule))
        return RuleSaveResultCode.RECURSIVE_RULE;

    return RuleSaveResultCode.SUCCESS;
}
```

Joonis 17. Reegli valideerimise kood.

Kontrollitakse, et salvestataval reeglil on eeltingimus ja tulemus ning eeltingimusel ja tulemustel peavad olema avaldise sõned. Reeglil peab olema projekti piires unikaalne nimi ning järjekorranumber.

Kirjutamine grupiaadressidele, mida ei ole ETS projektis kirjeldatud, on samuti keelatud, sest ainuke põhjus kirjutada tundmatutele grupiaadressidele on salvestada vahetulemusi. Võib juhtuda, et reegel kasutab tundmatuid grupiaadresse selle pärast, et imporditi projektifail, mis ei ole kooskõlas KNX süsteemiga. Sellisel juhul on vea andmine kõige parem võimalus. Reeglimootor toetab ükskõik kui keerulisi tehteid matemaatiliste ja loogiliste operaatoritega, seega peaks puuduma vajadus vahetulemusi kasutada. Võib tekkida segadus, kui olematu grupiaadress läheb tulevikus KNX-i poolt kasutusse.

Enne reegli lisamist kontrollitakse, et ühe reegli tulemused ei üritaks mitu korda sama grupiaadressi väärtust muuta. Ei oleks selge, millises järjekorras tulemused peaks käivitama ehk milline väärtus peaks lõpuks grupiaadressil olema. Tõenäoliselt on selline reegel vigaselt sisestatud.

Reegel, mis kirjutab ja loeb samalt aadressilt võib sattuda lõputusse tsüklisse. Sellised reeglid peaks keelama. Järgnevalt on toodud näide reeglist, mis kirjutab ja loeb samalt aadressilt.

```
{
    "Name": "Rekursiivne reegel",
    "Precondition": {
        "Variables": [],
        "Expression": "true"
    },
    "Effects": [
        {
            "Variables": [
                {
                    "Name": "lüliti",
                    "Value": "0/0/1"
                }
            ],
            "Expression": "lüliti",
            "Address": "0/0/1"
        }
    ]
}
```

Joonis 18. Reegel, mis viitab samale aadressile kui kirjutab.

Reegli eeltingimus on alati tõene. Reegli tulemus käivitatakse kohe kui reeglit kontrollitakse. Reegli käivitamisel loetakse lüliti grupiaadressi väärtus ning kirjutatakse see uuesti samale aadressile, kust see loeti. Reeglimootor kontrollib peale grupiaadressile kirjutamist, kas on vaja seda reeglit uuesti käivitada, kuna reeglit tulemuse väärtus oleneb lüliti grupiaadressi väärtusest, otsustab reeglimootor sama reegli uuesti käivitada. Reegli uuesti käivitamisel jätkuks samasugune prosess lõputult. Tsükli vältimiseks saab keelata sellised reeglid, kus on kasutatud kirjutamise aadressi eeltingimuse või tulemuse arvutamiseks.

4.5 Vahemälu

KNX võrgust väärtuste pärimine on aeglane tegevus, eriti kui pärimine ebaõnnestub. Igal reeglil võib olla vaja pärida mitut väärtust ning iga väärtust võib olla vaja kasutada mitmel erineval reeglil, seega võib päringute arv kasvada kiiresti suureks. Selle probleemi lahendamiseks on reeglimootoril vahemälu, kuhu salvestatakse grupiaadressi väärtus viimase uuendamise hetkel.

Vahemälu uuendamine saaks toimuda kas ette määratud ajavahemiku või KNX võrgu sündmuse järel. Praeguses rakenduses uuendatakse vahemälu ainult sündmuse peale. Juhul, kui KNX seadmete parameetrid on reegli kasutamiseks hästi üles seatud, ei ole sellest probleemi ja väärtused uuenevad ilma, et tekiks suur viide. Samas ei pruugi

seadmete parameetrid olla üles seatud reeglimootorit silmas pidades. Grupiaadressi uuendamise sündmus võib tulla palju hiljem, kui on tekkinud vajadus väärtust uuendada. Eelistatud oleks, et reeglimootor töötaks iga KNX konfiguratsiooniga, seega võiks tulevikus vahemälu uuendada nii grupiaadressi väärtuse muutumise peale kui ka perioodiliselt.

Tulevikus peaks vahemälu algväärtustama projekti andmete põhjal. Projekti andmete põhjal saab reeglimootor pärida kõikidelt kasutatavatelt grupiaadressidelt nende väärtust ning selle seada vahemälus grupiaadressi väärtuseks. Hetkel vahemälu ei algväärtustata, vaid sisestatakse väärtused ainult siis, kui KNX süsteemis väärtust uuendatakse.

Vahemälu on realiseeritud sõnastiku kaudu. Vahemälus on grupiaadressid ja iga grupiaadressi viimane teadaolev grupiväärtus. Vahemällu salvestatakse kanne ainult siis, kui grupiaadressile on leitud mõni väärtus. Grupiväärtuse uuendamise sündmuse, grupiaadressilt lugemise või grupiaadressile kirjutamise peale uuendatakse vahemälu.

4.6 Reeglite uuendamine

Reeglimootor jälgib jooksvalt KNX võrku ja sõnumeid, mida KNX võrgus saadetakse. Reeglimootor uuendab reegleid grupiväärtuse muutumise sõnumite peale. Alternatiivselt saaks reegleid uuendada perioodiliselt või vahemälu uuenemise peale. Vahemälu uuenemise peale reeglite uuendamine oleks kasulik, kui vahemälu uuendaks ennast nii perioodiliselt kui ka sündmuste peale, kuna hetkel vahemälu uueneb ainult sündmuste peale, on see võimalus samaväärne reeglite uuendamisega grupiväärtuse muutumise sõnumi peale. Reegleid saaks uuendada perioodiliselt. Perioodilisel uuendamisel on probleem, et seadme oleku muutumise ja reegli uuendamise vahele võib jääda viide, seetõttu on eelistatud hetkel sündmuste alusel reeglite uuendamist.

Kui KNX süsteemist saadakse grupiväärtuse muutumise sõnum, otsustatakse iga salvestatud reegli puhul, kas seda on vaja uuendada. Reeglit on vaja uuendada, kui selle eeltingimust või mõnda tulemust on vaja uuendada. Eeltingimust on vaja uuendada, kui mõni selle muutuja grupiaadress on sama kui grupiaadress, millelt tuli muutumise sõnum. Tulemuse uuendamise vajadus on analoogne eeltingimuse uuendamisele.

Peale uuendatavate reeglite leidmist kontrollitakse, milliste reeglite eeltingimused on täidetud. Eeltingimuste täitmist kontrollitakse reeglite järjekorranumbri alusel, siis

käivituvad ka reegli tulemused järjekorranumbri alusel. Selleks hinnatakse kõigepealt eeltingimuses kasutatavad muutujad. Muutuja väärtus on mõne grupiaadressi hetkeväärtus, seega muutuja väärtuse leidmiseks küsitakse võimalusel vahemälust vajaliku grupiaadressi väärtus või selle puudumisel üritatakse lugeda see väärtus otse KNX võrgust.

Muutujate väärtused tulevad reeglimootorisse grupiväärtuse tüübina. Selleks, et neid kasutada koos konstantidega nagu tavalisi süsteemi väärtusi, tuleb KNX võrgust loetud väärtused konverteerida süsteemi tüübiks. Kasutades projekti andmeid leitakse, mis tüüpi andmed sellele grupiaadressile saadetakse. Andes *Falcon SDK* teegile grupiväärtuse tüüpi objekti ja vastava tüübi, oskab teek konverteerida grupiväärtuse süsteemi tüüpi väärtuseks.

Avaldise hindamiseks on vaja avaldise hindaja korrektselt seadistada. Teegile antakse ette kultuurist olenevad formaadid. Näiteks millist märki kasutada komakohana ning milline on kuupäeva formaat. Lisaks on vaja teada anda, milliseid staatilisi funktsioone on võimalik avaldistes kasutada. Hetkel on avaldiste hindaja seadistatud nii, et saaks kasutada kõiki staatilisi funktsioone .NET Core süsteemi teegist „Math“. Avaldise hindajale antakse ette muutujate nimed ja nende hetkeväärtused.

Järgmisena eeltingimuse avaldis kompileeritakse ja hinnatakse. Avaldise kompileerimisega tehakse kindlaks, kas avaldise struktuur on korrektne. Teek hindab avaldise tulemuse ja asendab automaatselt muutujad avaldise tekstikujus õigete väärtustega. Hindamise tulemuseks on binaarne väärtus.

Reegleid, mille eeltingimus on väär, ei käivitata. Kui reegli eeltingimus on tõene, siis käivitatakse kõik reegli tulemused.

Reegli tulemuse täitmiseks hinnatakse reegli tulemuse avaldis analoogselt eeltingimuse hindamisega. Erinevuseks on, et avaldise lõpptulemus võib olla ükskõik milline objekt. Avaldise lõpptulemus muudetakse tagasi grupiväärtuse tüüpi objektiks, et see väärtus oleks võimalik kirjutada KNX võrku. Viimasena kirjutatakse avaldise lõpptulemus KNX võrku ning seadmed, mis kasutavad seda grupiaadressi oma sisendina, muudavad oma käitumist vastavalt. Reeglimootor kontrollib, kas enda poolt kirjutatud väärtuse tulemusena peab reegleid uuendama. Kui mitu reeglit kirjutavad samale grupiaadressile, siis lõpuks jääb grupiaadressile viimase järjekorranumbriga reegli tulemus.

4.6.1 Lõputu tsükkel

Kasutaja saab vabalt valida, kuidas reegleid koostada. Halvasti koostatud reeglid võivad sattuda lõputusse tsüklisse, kuna reegleid peab uuesti kontrollima ka siis kui reeglimootori tulemus tuleb kirjutatakse KNX süsteemi.

Lõputu tsükkel võib tekkida, kui mitu reeglit üksteise grupiaadressidele viitavad nagu järgnevas näites.

```
{
    "Name": "Tsükkel 1",
    "Precondition": {
        "Variables": [
            {
                "Name": "lüliti",
                "Value": "0/0/1"
            }
        ],
        "Expression": "lüliti"
    },
    "Effects": [
        {
            "Variables": [],
            "Expression": "true",
            "Address": "0/0/2"
        }
    ]
}

{
    "Name": "Tsükkel 2",
    "Precondition": {
        "Variables": [
            {
                "Name": "lüliti",
                "Value": "0/0/2"
            }
        ],
        "Expression": "lüliti"
    },
    "Effects": [
        {
            "Variables": [],
            "Expression": "true",
            "Address": "0/0/1"
        }
    ]
}
```

Joonis 19. Kaks reeglit, mis tekitavad lõputu tsükli.

Kui eelnevad kaks reeglit on reeglimootorisse korraga salvestatud, siis võib tekkida lõputu tsükkel. Oletame, et alguses on grupiaadresside „0/0/1“ ja „0/0/2“ väärtused mõlemad väärad. Selles seisus salvestatakse mõlemad reeglid. Peale reeglite salvestamist lülitatakse sisse lüliti, mille grupiaadress on „0/0/1“. Grupiaadressi „0/0/1“ muutumine käivitab reegli „Tsükkel 1“. Reegli eeltingimus ehk grupiaadressi „0/0/1“ väärtus on „tõene“, seega käivitatakse tulemus. Tulemuse käivitamine tähendab, et grupiaadressile „0/0/2“ kirjutatakse väärtus „tõene“. Sellega muutub grupiaadressi

„0/0/2“ väärtus ning on vaja uuendada reeglit „Tsükkel 2“. Selle reegli eeltingimus ehk grupiaadressi „0/0/2“ väärtus on „tõene“. Käivitatakse reegli tulemus, mis kirjutab uue väärtuse grupiaadressile „0/0/1“. Grupiaadressi väärtuse muutuse tõttu on vaja uuesti hinnata reeglit „Tsükkel 1“. Reegel „Tsükkel 1“ käivitab iga kord reegli „Tsükkel 2“ ning iga reegli „Tsükkel 2“ käivitumine põhjustab reegli „Tsükkel 1“ käivitumise. Sellise tsükli ennetamiseks ei leidnud autor lihtsat lahendust. Järgnevalt on analüüsitud, kuidas sellise tsükli korral peaks käituma.

Tsüklil saaks lasta edasi minna. Lõputu tsükliga reeglid saadaksid välja lõputult sõnumeid, mis ujutaks üle teised KNX võrgu sõnumid. Teised sõnumid ei pruugiks kohale jõuda või kohale jõudmine võtaks kauem. Kasutaja saaks aru, et midagi on valesti, sest KNX võrk ei ole enam kättesaadav.

Tsüklil saaks lasta edasi minna, aga seada viide reegli kontrollimise aega, kui sõnumi saadab reeglimootor mitte KNX võrk. Tänu viitele saaksid teised KNX sõnumid jätkata nagu tavaliselt. Kasutaja saaks aru, et midagi on valesti, kui ta märkab, et mõne reegli tulemus on alati näha.

Annaks seada piiri, kui sügavale reeglite ahelas hindamisega võib minna. Reeglimootor võib jätta meelde reeglit, mis algselt tsükli põhjustas ei tohiks enam käivitada. Kasutaja saaks aru, et midagi on valesti, kui ta näeb, et tema tehtud reegel ei tööta.

Hetkel on sellise lõputu tsükli ennetamine lahendamata probleem. Autori arvates ei ole ükski pakutud lahendustest praegusel kujul sobilik ning peaks arendama edasi mainitud valikuid või leidma uue lähenemise. Praegu lõputu tsükli tekkimisel annab reeglimootor vea ja lõpetab töö.

4.7 Muud API päringud

Reeglimootorile on loodud testimise lihtsustamiseks veel mõned päringud. Reeglimootoril on võimalus pärida kõiki salvestatud reegleid ning reegleid kustutada.

Kõikide salvestatud reeglite päringu mall on järgmine.

```
GET api/rule
response:
[
    {
        "name": "",
        "orderNumber": 0,
        "precondition": {
            "expression": "",
            "variables": [
                {
                    "name": "",
                    "variableType": 0,
                    "value": ""
                }
            ]
        },
        "effects": [
            {
                "name": null,
                "expression": "",
                "address": "",
                "groupAddress": {
                    "address": 0
                },
                "variables": []
            }
        ]
    }
]
```

Joonis 20. Kõikide reeglite läbi API pärimise mall.

Reeglite kustutamine käib järjekorranumbri alusel, sest järjekorranumber on unikaalne. Nime ei kasutatud, sest nimes võib olla URL-s lubamatuid märke. Järgnevalt joonisel on kujutatud ühe reegli kustutamise mall.

```
DELETE api/rule/{orderNumber}
response: true
```

Joonis 21. Reegli kustutamine läbi API.

Päringu tulemusena kustutatakse reegel, millel on päringus täpsustatud järjekorranumber. Päring tagastab väärtuse „tõene“, kui kustutamine õnnestus ning väärtuse „väär“, kui kustutamine ebaõnnestus või päringut ei olnud olemas.

4.8 Testimine

Testimiseks on kasutatud nii reaalset KNX võrku kui ka virtuaalset KNX võrku. Reeglimootori arendamise käigus kasutati manuaalseid teste. Manuaalseid teste kasutades on lihtsam jälgida, mis juhtub füüsilises KNX süsteemis.

Reeglimootori realisatsiooni nõuetele vastavuse kontrollimiseks on loodud hulk automaatseid ühikteste. Ühiktestide eesmärk on testida üksikute meetodite korrektsust. Ühiktestid on loodud kasutades tööriista Xunit. Tulevikus peaks API testimiseks lisama integratsioonitestid kasutades Newman tööriista ja muutma automaattestide kollektsiooni põhjalikumaks.

5 Kokkuvõte

Bakalaureusetöö peamine eesmärk oli muuta kasutajale lihtsamaks KNX standardil põhineva hooneautomaatika lahenduse seadistamine. Loodud reeglimootori abil on kindlasti lihtsam ümber seadistada valmis olevat KNX lahendust. Samas on veidi keerulisem üles seada uut KNX lahendust.

Reeglimootori arendamiseks leiti meetod KNX võrguga suhtlemiseks. Kahjuks peab leitud meetodi rakendamiseks KNX süsteem olema juba seadistatud läbi ETS-i. Reeglimootori tööks on vaja seadistada nii KNX süsteem kui ka reeglimootor. Seetõttu on reeglimootori üles seadmine keerulisem, kui autor oleks soovinud.

Reeglite defineerimiseks ja hindamiseks loodi avaldiste süsteem. Avaldiste süsteem pakub piisavalt palju võimalusi, et kasutajal peaks saama defineerida kõiki vajalikke reegleid. KNX väärtuste tõlgendamiseks ja KNX süsteemi informatsiooni kuvamiseks loodi projektifaili importer. Importer on kasutajale mugav lahendus reeglimootorisse ETS projekti andmete sisestamiseks.

Reeglimootori peamised tugevused on, et reeglimootoriga saab lihtsalt ja kiiresti KNX süsteemi loogikat ümber seadistada ning reeglimootoriga on tehtud võimalikult lihtsaks luua keerulisi reegleid.

Reeglimootori peamised miinused on, et see vähendab KNX lahenduse töökindlust ning muudab KNX lahenduse algse seadistamise keerukaks, kuna on vaja seadistata nii KNX lahendus ise kui ka reeglimootor.

5.1 Tulevik

Edasise sammuna peaks arendama serveripoolse SaaS rakenduse, mis suhtleb reeglimootoriga üle interneti. SaaS rakendus võtaks vastu kasutaja korraldused ja sünkroniseeriks need reeglimootoriga.

Reeglimootorile oleks vaja arendada kasutajaliideses veebilehe kujul. Kasutajaliideses võiks saada ülevaate kõikidest reeglitest ja KNX võrgu hetkeseisust. Kasutajaliidese ühe osana peaks looma veebilehe, kus saab sisestada reegleid. Reeglite sisestamise aknas saaks valida grupiaadressi. Grupiaadressi valimisel peaks olema kasutaja aitamiseks saadaval informatsioon iga grupiaadressi nimega ja grupiaadressi kasutavate grupiobjektidega ning seadmetega. Avaldisi peaks saama sisestada vabateksti kujul lahtrisse. Avaldise hetke tulemust peaks näitama enne reegli salvestamist kasutajaliideses.

Reeglimootori API-l on hetkel ainult sellised funktsioonid, mis on vajalikud testimiseks. Reeglimootori API-t peaks täiustama kasutajaliidesele vajalike funktsioonidega.

Kasutatud kirjandus

[1] Sławomir Sowa, Jarosław Gielniak. 2018. „Implementation of the lighting control algorithms in the KNX system", *ITM Web of Conferences*, Kd 19, pp 1-3, DOI: https://doi.org/10.1051/itmconf/20181901040.

[2] M. Talimets (2016). „Kasutajasõbraliku liidese analüüs ja väljatöötamine targa maja lahendusele" (Bakalaureusetöö). Tallinna Tehnikaülikool, Tallinn, Estonia. Loetud aadressil https://digikogu.taltech.ee/et/Item/c8655fc5-afce-40e1-9070-2dcffecc0a14.

[3] openHAB Community and the openHAB Foundation e.V. „ Working with rules and scripts", [Võrgumaterjal]. Loetud aadressil https://www.openhab.org/docs/tutorial/rules.html, [Kasutatud 05.01.2020].

[4] KNX Association cvba. „Group Address & Style", [Võrgumaterjal]. Loetud aadressil https://support.knx.org/hc/en-us/articles/115001825344-Group-Address-Style, [Kasutatud 07.01.2020].

[5] KNX Association cvba. KNX Specifications v2.1. 2014.

[6] Martin Fowler. „Domain-Specific Languages Guide ", [Võrgumaterjal]. Loetud aadressil https://martinfowler.com/dsl.html, [Kasutatud 05.01.2020].

[7] Manuel Jiménez, Francisca Rosique, Pedro Sánchez, Bárbara Álvarez, Andrés Iborra. 2009. „Habitation: A Domain-Specific Language for Home Automation", *IEEE Software*, Kd 26, Nr 4, pp 30-38, DOI: https://doi.org/10.1109/MS.2009.93.

[8] M. Mernik, J. Heering, A. M. Sloane. 2005. „ When and how to develop domain-specific languages ". *Journal ACM Computing Surveys*, Kd 37, Nr 4, pp 317-344, DOI: https://doi.org/10.1145/1118890.1118892.

[9] R. Prähla (2019). "Virtuaalsete andurite loomine ja lisamine KNX standardile vastava automatiseeritud hoone lahendusse" (Bakalaureusetöö). Tallinna Tehnikaülikool, Tallinn, Estonia. Loetud aadressil https://digikogu.taltech.ee/et/Item/035b0034-102b-4050-ba4f-dc56f17ad9aa.

[10] KNX Association cvba. ETS5 XML Project Format Description. 2019.