TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Rene Prähla 142353IAPB

VIRTUAALSETE ANDURITE LOOMINE JA LISAMINE KNX STANDARDILE VASTAVA AUTOMATISEERITUD HOONE LAHENDUSSE

Bakalaureusetöö

Juhendaja: Tarvo Treier
MSc

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: Rene Prähla

20.05.2019

Annotatsioon

Käesoleva bakalaureuse töö põhieesmärgiks on analüüsida KNX standardi võimalusi virtuaalsete seadmete loomiseks ja realiseerida virtuaalsed seadmed vastavalt KNX standardile vastavalt automatiseeritud hoone lahendusse. Töös antakse ülevaade KNX standardist ning luuakse virtuaalse anduri rakendus tõestamaks virtuaalsete andurite loomise ja kasutamise võimalust KNX standardile vastava automatiseeritud hoone lahenduses.

Bakalaureuse töö põhiliseks probleemiks on leida asendus kallitele KNX standardile vastavatele automatiseeritud hoone komponentidele ja lahendus automatiseeritud hoone lahenduse laiendamisel tekkivatele komplikatsioonidele.

Probleemi lahendamiseks on loodud kaks erinevat rakendust virtuaalsete seadmete loomise ja kasutamise tõestamiseks ning väiksemas suuruses automatiseeritud hoone lahendus loodud rakenduste testimiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 28 leheküljel, 4 peatükki, 22 joonist, 0 tabelit.

Abstract

Creation and addition of virtual sensors to KNX automated building solution

The purpose of this thesis is to analyze KNX automated building standard for creation of virtual sensors and to create virtual sensor solutions to use with KNX automated building solution. In this thesis, the author has given an overview of KNX standard and created virtual sensor solutions to prove the capability of creating virtual sensors for KNX automated building solution.

The main problem of this thesis is to find a substitute for expensive sensors that created according to the KNX standard and to find a solution for problems that may arise when expanding the current KNX automated building solution.

To solve the problem, the author has created two different software applications to prove that the creation of virtual sensors is possible for KNX automated building solutions and has created a small scale KNX automated building solution to test those software applications.

The thesis is in Estonian and contains 28 pages of text, 4 chapters, 22 figures, 0 tables.

Lühendite ja mõistete sõnastik

API	*Application programming interface*
cEMI	*Common external message interface,* KNX tunnelühendusel kasutatav ühine väliste sõnumite liides.
Docker	Tarkvarapakettide ehk konteinerite virtualiseerimise rakendus
ETS	*Engineering Tool Software,* KNX Assotsiatsiooni poolt loodud tarkvara KNX standardile vastava hoone süsteemi seadistamiseks
GET	Hüperteksti edastusprotokolli meetod andmeedastuseks
ID	Identifikaator
IP	*Internet protocol,* Interneti protokoll
JSON	*JavaScript Object Notation,* JavaScripti objekti esitlusviis
Konteiner	Tarkvarapakett, mis käivitatakse eraldatud keskkonnas Docker rakenduse kaudu. Sisaldab lisaks loodud rakendusele ka vajalikke tööriistu, teeke ja konfiguratsioonifaile.
POST	Hüperteksti edastusprotokolli meetod andmeedastuseks
REST	*Representational State Transfer,* Kujutava olekuedastuse protokoll
SaaS	*Software as a Service,* Tarkvara teenusena
SDK	*Software development kit,* tarkvara arenduskomplekt
SQL	*Structured Query Language*, Struktuuripäringukeel
UI	*User interface,* Kasutajaliides

Sisukord

Jooniste loetelu

1 Sissejuhatus

Aastate jooksul on inimesed muutunud üha mugavamaks ja järjest rohkem soovitakse hooneid automatiseerida. Üheks automatiseerimise standardiks on KNX, mis näeb ette, et kõik omavahel kokku ühendatud seadmed ja andurid peavad oskama suhelda üksteisega kasutades KNX protokolli. Selliseid seadmeid ja andureid on erinevate tootjate poolt tehtud väga palju, kuid siiski võib kogu automatiseerimise lahendamine nendega tähendada mõningaid probleeme:

- KNX seadmed ja andurid maksavad palju.
- Kuigi tootjaid ja erinevaid seadmeid ja andureid on palju, siis päris kõike ei pruugi veel olemas olla.
- Kõik KNX seadmed ja andurid peavad olema ühendatud KNX-i võrku kas füüsiliselt kaabliga või juhtmevabalt. Kõigist seadmetest ja anduritest pole loodud juhtmevabasid versioone. Kui ehituse käigus pole kohe vajalikke kaableid paika pandud, siis nende hilisem paigaldamine võib väga kulukaks osutuda.

Käesoleva töö autor on kokku puutunud valmimisjärgus eramajaga, kus ta hoone automatiseerimisel avastas, et nii välise ilmajaama kui ka autoväravа asendi anduri jaoks polnud veetud kaabeldust. Ilma kaablita neid andureid kasutada ei saa, seega antud olukorras olid mõlemad seadmed kasutud. Lähtudes teadmistest, et informatsiooni ilma kohta on võimalik leida internetist ja autoväravale oli suunatud turvakaamera, tekkis autoril küsimus „Kas on võimalik ära kasutada olemasolevaid ressursse puuduva funktsionaalsuse asendamiseks?“ Käesolev töö püüab leida lahendust eelnimetatud küsimusele luues virtuaalseid seadmeid eelnimetatud seadmete asendamiseks.

Antud bakalaureuse töö põhieesmärgiks on analüüsida KNX standardi võimalusi virtuaalsete seadmete loomiseks ja realiseerida virtuaalsed seadmed vastavalt KNX standardile vastavalt automatiseeritud hoone lahendusse.

1.1 Metoodika

Bakalaureusetöö valmib järgmistes etappides:

1. KNX standardi uurimine ja sellel baseeruva lahenduse realiseerimine
2. KNX standardile vastavate olemasolevate ühendusteekide uurimine ja sobivaima teegi ja arendusplatvormi valimine
3. Lihtsama virtuaalse seadme realiseerimine võimalikkuse tõestamiseks
4. Lokaalse ja pilvepõhise rakenduse loomine, mis võimaldab luua ühendust mitme automatiseeritud hoone lahendusega ja lubab kasutajal luua endale sobilikud virtuaalsed seadmed.
5. Loodud rakenduse funktsionaalsuse testimine ja dokumenteerimine

1.2 Ülesehitus

Selleks, et oleks arusaam, kuidas automatiseeritud hoone lahendus töötama peaks, annab käesoleva töö teine peatükk ülevaadet KNX standardist ning KNX standardil põhineva automatiseeritud hoone lahenduse seadistamisest. Kolmandas peatükis võrdleb autor KNX standardile vastavaid ühendusmeetodeid ning tarkvaralahendusi, millega on võimalik luua virtuaalseid seadmeid ja ühendada neid automatiseeritud hoone lahendusega. Neljandas peatükis on lahti seletatud kahe autori poolt loodud lahenduse funktsionaalsus, mis tõestavad virtuaalsete andurite loomise ja kasutamise võimalust KNX standardile vastava automatiseeritud hoone süsteemis.

2 KNX ja selle seadistamine

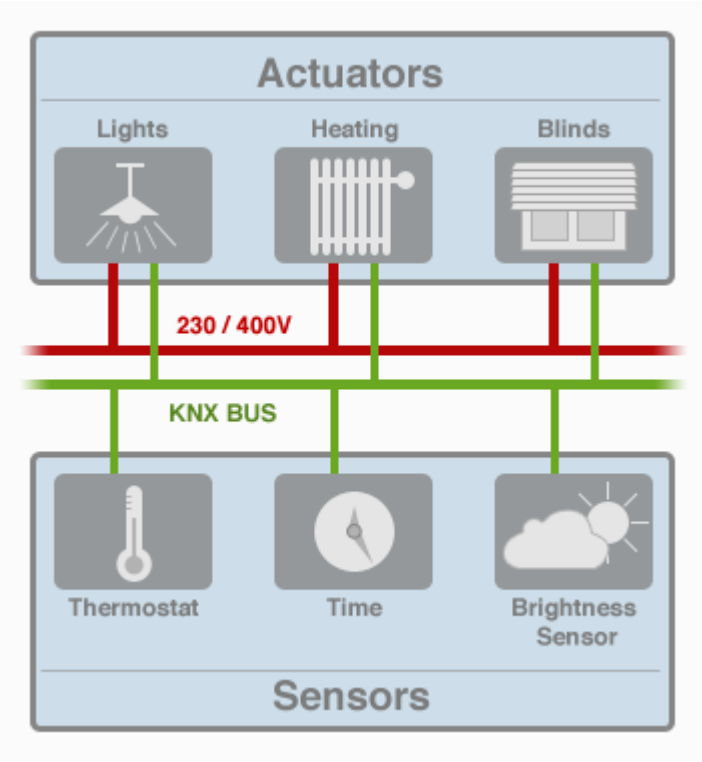
Enne virtuaalse anduri loomist, tuleks omada arusaama, kuidas töötab KNX standardile vastav automaatse hoone lahendus. Selle jaoks sisaldab järgnev peatükk üldist tutvustavat informatsiooni KNX standardist, KNX hoone lahenduse ülesehitusest ja ka informatsiooni süsteemi seadistamisest kasutades KNX assotsiatsiooni poolt loodud tarkvara nimega ETS5.

2.1 KNX standard

KNX standard [1] on KNX assotsiatsiooni poolt loodud ülemaailmselt kasutusel olev standard hoonete kontrollimiseks ja automatiseerimiseks. Tegemist on detsentraliseeritud süsteemiga, mis ei nõua juhtivat kontrolliva loogikaga seadet. KNX standardi kohaselt on kõik seadmed omavahel ühendatud siini kaudu ja igale seadmele on antud oma käitumisloogika. [2] Siiniks, millega seadmed on omavahel ühendatud, võib olla keerdpaarjuhe, voolujuhe, raadiovõrk või interneti protokollil põhinev andmeside võrk. Kuna kõik seadmed omavad oma käitumisloogikat, siis peavad seadmed oskama omavahel suhelda. Selleks kasutab KNX süsteem unifitseeritud suhtluskeelt, mida kutsutakse KNX telegrammideks.

Arhitektuurilt koosneb süsteem järgnevatest komponentidest:

- Toiteallikas, mis annab töötamiseks vajaliku elektritoite kõikidele ühendatud seadmetele.
- Andurid, mis koguvad informatsiooni ümbritsevast keskkonnast ja saadavad vastava informatsiooni laiali telegrammide kujul.
    - Näiteks erinevad lülitid, temperatuuri-, liikumis- ja hämarusandurid
- Täiturid, mis loevad siinilt informatsiooni sisse telegrammidena ja rakendavad oma funktsionaalsust vastavalt loetud informatsioonile.
    - Näiteks kütte- ja ventilatsioonisüsteemid, valgustid, heli ja videoseadmed
- Süsteemi komponendid, mis lubavad luua ühendusi välisvõrkudega või teiste siinidega.



Joonis 1. Lihtsam KNX siini visualiseeriv joonis [2]

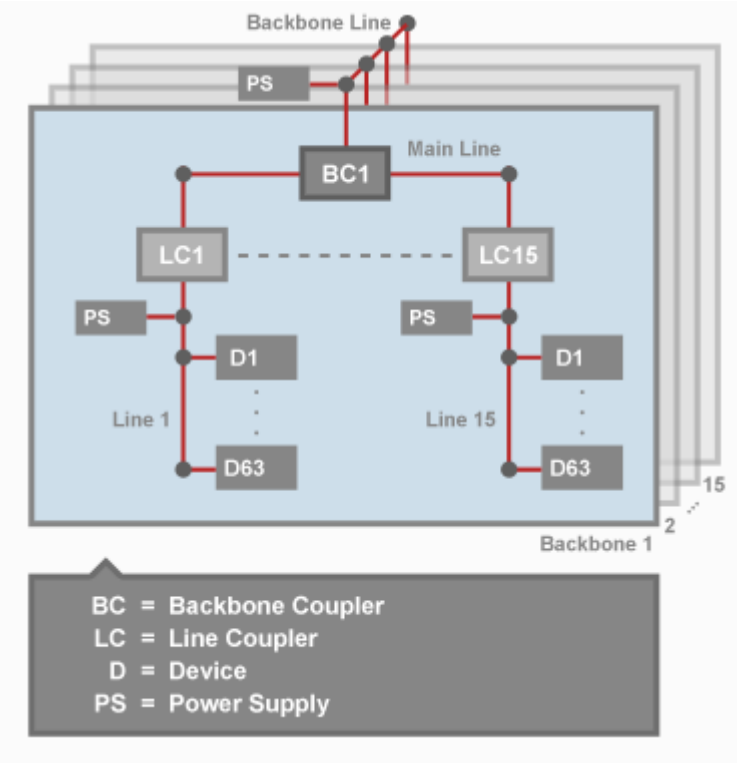
2.2 KNX topoloogia

Väiksemate projektide puhul võib KNX süsteem tunduda vägagi lihtne, aga suuremate projektide puhul võib süsteem muutuda väga kiiresti väga keeruliseks. Selle jaoks on kasutusele võetud hierarhiline süsteem erinevate süsteemiosade juhtimiseks.

Kõige väikseim üksus KNX hierarhias on liin (ingl k: *line*). Ühele liinile on võimalik ühendada kokku maksimaalselt 64 erinevat seadet. Järgnevalt suurem üksus on pealiin (ingl k: *main line*), mis mahutab kokku kuni 15 erinevat liini. Liinid on omavahel seotud liiniühendajaga (ingl k: *line coupler*). Kõige suurem üksus on magistraalliin (ingl k: *backbone line*), millesse on võimalik ühendada 15 pealiini magistraalühendajaga. Igal liinil ja magistraalliinil peab olema oma vooluallikas ja siin. [3]

Igale liinil olevale seadmele on määratud oma individuaalne aadress, millega on võimalik võtta ühendust kindla seadmega. Seadme individuaalne aadress koosneb pealiini, liini ja seadme numbrist. Individuaalne aadress esitatakse järgneval kujul: 1.2.123. Individuaalne aadress näitab esiteks millisel pealiinil seade asub, siis millisel liinil seade asub ja lõpuks on seadmele antud numbriline väärtus. Eelnevalt nimetatud aadressi järgi, asub seade

esimesel pealiinil, teisel liinil ja seadme leiab 123-ndalt kohalt. Individuaalset aadressi kasutatakse seadmetele loogikareeglite seadistamiseks. [2]



Joonis 2. KNX arhitektuuri visualiseeriv joonis [2]

2.3 KNX standardil põhineva süsteemi seadistamine

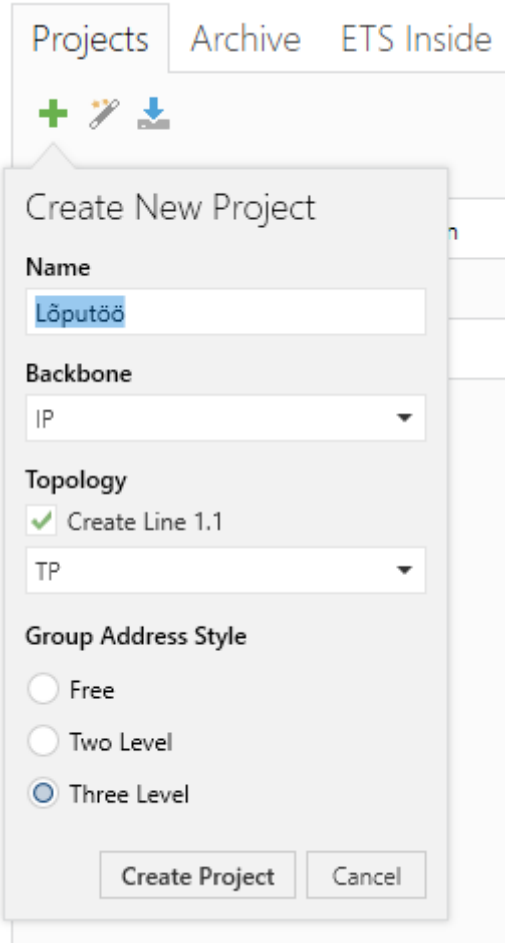
Kujutagem ette, et olete kokku ühendanud automatiseeritud hoone lahenduse, kuid süsteem ei tööta veel nii kuidas te soovite. Mitte töötamise põhjuseks on see, et süsteemist on puudu loogika reeglite seadistused. Selleks, et loogikareegleid seadistada, on KNX Assotsiatsioon loonud rakenduse, mida nimetatakse ETS-iks. ETS võimaldab seadistada kõiki KNX standardile vastavaid seadmeid, mis võivad olla loodud erinevate tootjate poolt. Standardile vastavus tagab komponentide töökindluse ning tootjate rohkus lubab hoone süsteemi loojal kasutada kõige sobilikemaid komponente oma lahenduses.

Antud lõputööga seoses, on lõputöö autor loonud lihtsama hoone näitelahenduse ning näitab ette, kuidas käib KNX lahenduse automatiseerimine kasutades ETS5 rakendust.

Kui ETS rakendus tööle panna, siis esimene ekraan, mida kasutaja näeb on ülevaate ekraan. Ülevaate ekraanil saab kasutaja luua ja avada projekte, vaadata arhiveeritud

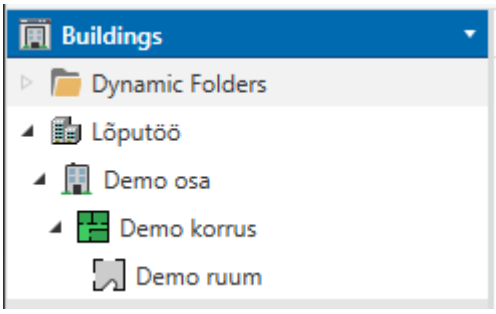
projekte ning luua projekte kasutades ETS Inside rakendust. Käesolevas töös on kasutatud ainult projekti loomise võimalust.

Kui projekti luua, siis näeb kasutaja sellist akent nagu on kujutatud joonisel 3. Kasutaja saab seal anda nimetuse oma projektile ja valida, kas magistraalliin on interneti protokolli (IP) või keerdpaarjuhtme (TP) baasil, kas topoloogia on loodud keerdpaarjuhtmele (TP), voolujuhtmele (PL), raadiovõrgule (RF) või internetiühendusele (IP) ja ka milline on grupiaadresside stiil. Antud töös on seaded valitud nii, nagu joonisel 3 on näidatud.



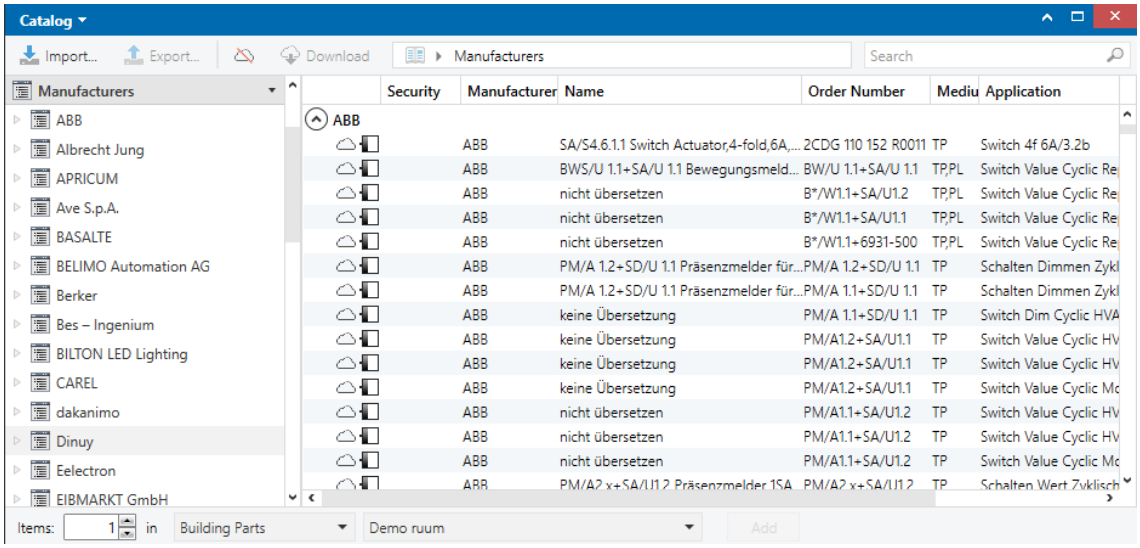
Joonis 3. ETS projekti loomine

Peale projekti loomist avaneb vaade, mille pealkirjaks on „hooned“ (ingl k. *buildings*). Tavaliselt luuakse projektis hoonete struktuur, kus määratakse ära hierarhiliselt suuremast väiksemaks hooned, hoonete osad, korrused ja ruumid. See muudab süsteemi lihtsamini hallatavaks, kuna kõik hoones kasutatavad seadmed on võimalik grupeerida vastavalt hoone osale. Kuna käesoleva töö lahendus on mõeldud katsetamiseks, siis on hooneosad määratud vastavalt joonisele 4.



Joonis 4. Näiteprojekti hoone struktuur

Järgnevalt tuleb ruumi lisada realiseeritud süsteemi komponentide mudelid. Komponentide mudelid on tootjate poolt valmistatud komponentide seadistused, mis määravad ära füüsilise seadme funktsionaalsuse võimalused. Mudelid on saadaval kataloogi (ingl k. *catalog*) aknas, mis on kujutatud joonisel 5.



Joonis 5. ETS5 kataloogi aken

Kataloogi akna kaudu on võimalik otsida seadet käsitsi kasutades tootjate nimekirja, mis on märgitud vasakul servas ja siis otsida läbi tabeli kasutatavat seadet. Lisaks on olemas ka otsingusüsteem, mille kaudu on võimalik seadme nime, tootja või tootekoodi järgi otsida kindlat seadet.

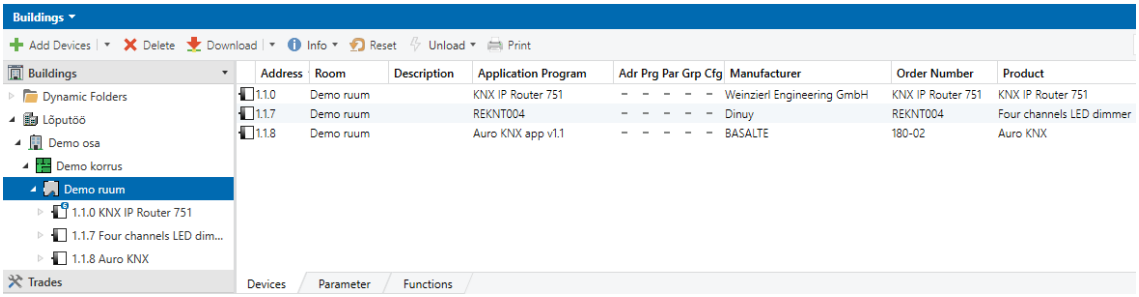
Autori poolt loodud KNX lahendus koosneb järgmistest komponentidest:

- KNX Toiteplokk

- KNX IP ruuter, võrguühenduse loomiseks väliste seadmetega kasutades interneti protokolli.
- KNX *dimmer* seade valgustitele ja LED valgusti
- KNX multifunktsionaalne andur, mis sisaldab liikumisandurit ja temperatuuriandurit
- Halogeenivaba keerdpaarjuhe

Nimekiri koos seadmete nimetustega asub Lisa 1.

Kuna toiteplokki ega muid KNX standardile mittevajalikke seadmeid pole võimalik süsteemi lisada, siis lõpptulemusena on autori näites lisatud ainult 4 seadet: IP ruuter, universaalne sisend-/väljundseade, valgustitele mõeldud *dimmer* ja multifunktsionaalne andur.



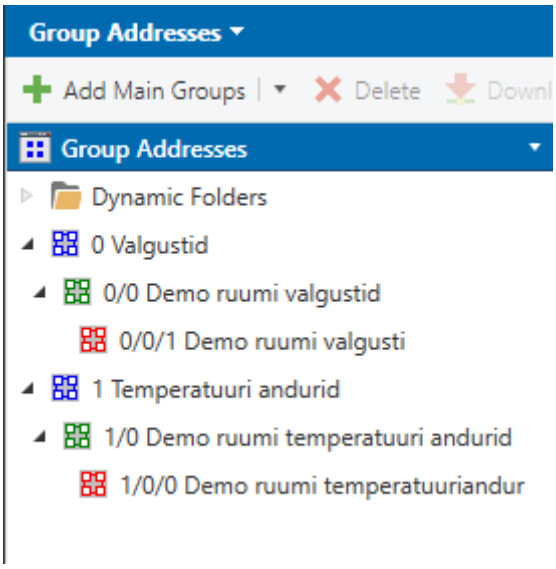
Joonis 6. ETS projekti lisatud seadmed

Igale lisatud seadmele on määratud individuaalne aadress, mida on võimalik muuta vastavalt oma lahenduse topoloogilistele otsusele. Lisaks on võimalik lisada ka juurde kirjeldus, mida saab kasutada seadmete eristamiseks, kui sama seadmemudelit on kasutuses rohkem kui üks.

Peale seadmete lisamist hoone skeemi tuleb muuta seadmete parameetreid vastavalt seadme kasutuse eesmärkidele. Selleks, et saaks seadme parameetreid muuta, tuleb valida seade hoonete vaatest ja seadme valimisel peaks tekkima seadme alla serva valik nimega parameeter (ingl k *parameter*). Vajutades seda avaneb vaade, kus on üles märgitud kõik seadme võimalused, mida on võimalik aktiveerida vastavalt vajadusele. Peale parameetrite seadistamist on kõik seadistatud funktsioonid kättesaadavad grupiobjektide vaates (ingl. k *Group Objects*). Kuna tegemist on aega nõudva tegevusega ja igal seadmel

on erinevad võimalused, siis käesoleva töö autor ei hakka parameetrite seadistamist siin töös detailsemalt lahti seletama.

Järgnevalt, kui grupiobjektid on olemas grupi objektide vaates, on võimalik hakata looma grupi aadresse. Grupiaadresside eesmärk on lihtsustada seadmete suhtlust siinil, lubades luua automatiseerimisel olukordi, kus üks andur võib kontrollida mitut täiturit, mitu andurit kontrollivad ühte täiturit ja mitu andurit kontrollivad mitut täiturit. [4] Grupiaadresside loomiseks tuleb avada grupi aadresside vaade. Antud vaates on võimalik luua seadmegruppe, mis annavad seadmetele loogika, mille alusel seadmed hakkavad käituma. Antud töös kasutatud grupiaadressid on kolme tasemelised ja selle tõttu koosnevad kolmest osast: peagrupist (ingl. k *Main Group*), keskmisest grupist (ingl. k *Middle Group*) ja siis grupi aadressi individuaalsest identifikaatorist. Näiteks grupi aadress, mis on tähistatud väärtusega 0/0/1, näitab, et peagrupi tähis on null, keskmise grupi tähis on 0 ja individuaalne identifikaator on 1. Autori näidisprojektis on grupi aadressid kujutatud vastavalt joonisele 7.

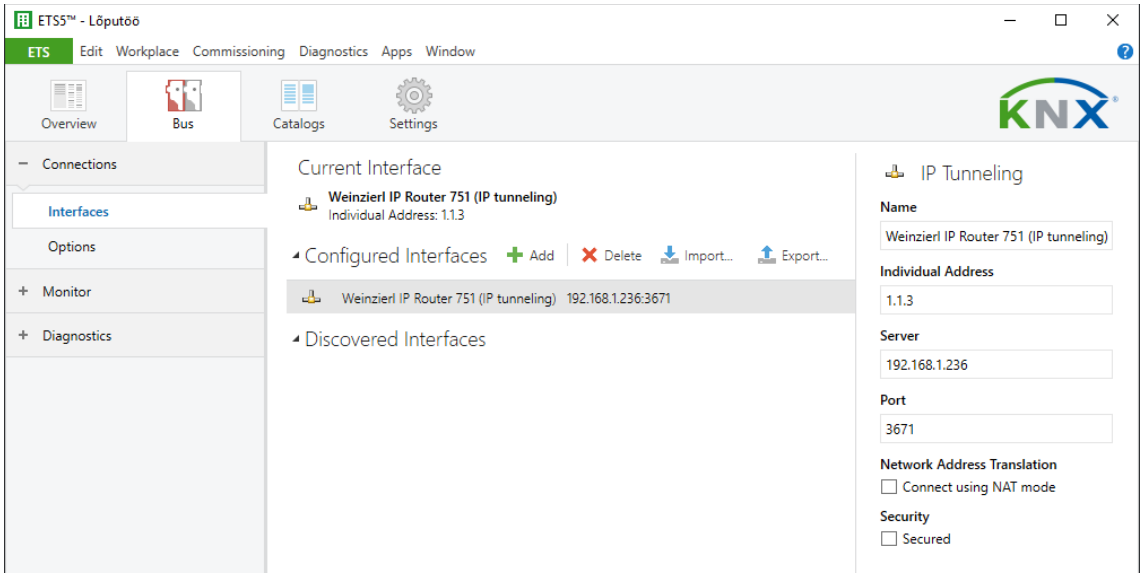


Joonis 7. Autori projekti grupiaadresside tähistus

Peale grupiaadresside loomist, on vaja lisada grupiaadressidele vastavalt hoonete vaatest seadmete funktsionaalsus. Selleks on vaja otsustada, mis on iga grupiaadressi eesmärk. Autori poolt loodud lahenduse eesmärk on kasutada seda virtuaalsete seadmete kasutuse demonstreerimiseks. Seega aadressi 0/0/1 alla on seadistatud ainult lambile mõeldud lüliti funktsionaalsus, mida on võimalik käivitada, luues telegrammi antud aadressile ja edastada see füüsilisele süsteemile. Lambi töötamise testimiseks on lisatud sellele

aadressile ka liikumisanduri väljund. Aadressi 1/0/0 eesmärk on tekitada loetav informatsiooni allikas temperatuuri lugemiseks süsteemist. Seetõttu on eelnimetatud aadressile seadistatud temperatuurianduri väljund.

Peale grupiaadresside lisamist, on järgnevalt vaja laadida loodud seadistus füüsilisse süsteemi. Autori lahenduses on kasutusel IP ruuter. See seade lubab meil luua ühendust füüsilise lahendusega kasutades selleks interneti võrku. Selleks tuleb esmalt luua ühendus KNX füüsilise lahendusega. Ühenduse loomiseks tuleb ETS rakenduses liikuda avalehele ja avada siini vaade. Siini vaates on võimalik konfigureerida liidese (ingl. k *interface*) ühenduvust, jälgida telegrammide liikumist füüsilises lahenduses ning tegeleda seadmete diagnostikaga. Selleks, et saaks luua ühendust füüsilise seadmega, on vaja teada serveri võrguaadressi ja porti. KNX serveri võrguaadressi saab tuvastada interneti ruuteri kasutajaliideses, kui lahendus on ühendatud internetti läbi ruuteri. Joonisel 8 on näidatud autori lahenduses kasutatud ühenduvuse sätteid.



Joonis 8. KNX liidesega ühenduse loomine

Nüüd, kui ühendus füüsilise lahendusega on loodud, tuleb laadida loodud seadistus füüsilisele lahendusele. Selleks tuleb avada uuesti projektis hoonete vaade ja vajutada allalaadimise nuppu. Selle protsessi käigus saadetakse kõikidele seadmetele loodud seadistused vastavatele seadmetele kasutades loodud ühendust. Protsessi lõppedes on seadistatud automatiseerimine KNX standardile vastavale hoone lahendusele.

3 KNX standardil baseeruva lahendusega ühenduse loomine

Järgnevas peatükis tutvustab autor KNX standardile vastava automatiseeritud hoone lahenduse võimalusi luua ühendust välise võrguga kasutades interneti protokolli ja tutvustab olemasolevaid tarkvarateeke, millega on võimalik luua ühendus automatiseeritud hoone lahendusega. Informatsioon on vajalik virtuaalsete seadmete loomisel ja kasutamisel KNX standardile vastava automatiseeritud hoone lahenduses.

3.1 Ühendusmeetodid

KNX standardi kohaselt on võimalik automatiseeritud hoone lahendusega luua kahte tüüpi ühendusi kasutades IP protokolli. Nendeks on tunnelühendus ja ruutimine. Selleks, et luua neid ühendusi, on vaja paigaldada hoone lahendusse kas IP ruuter või IP liides.

Tunnelühendus on interneti ühendusetüüp, kus on võimalik luua krüpteeritud ühendus kahe seadme vahel läbi avaliku internetivõrgu. Tunnelühenduse eripäraks on privaatsus ja võime saata informatsiooni kasutades kapseldamise meetodit. Tunnelühendus luuakse OSI mudeli teise kihi ehk kanalikihi tasemel. Saatmisel informatsioon kapseldatakse interneti protokollile vastavaks ja vastuvõtmisel dekodeeritakse informatsioon ja edastatakse vastavale rakendusele või süsteemile. [5] KNX lahenduses kasutatakse tunnelühendust cEMI formaadis telegrammide edastamiseks KNXnet/IP tunneli protokolli kapseldatuna üle interneti võrgu. [6] KNX lahenduses tunnelühenduse loomiseks on vaja lisada lahendusse kas IP ruuter või IP liides. Tunnelühendus on primaarselt kasutuses ETS rakendusega suhtlemisel, kuid seda kasutatakse ka väliste rakendustega suhtlemiseks.

Ruutimine on kindla ühenduseta meetod informatsiooni korraga saatmiseks kõikidele seadmetele, mis on ühenduse grupiga liitunud. Ruutimisel saadetakse multiedastus sõnum ruuteri kaudu teistele seadmetele, kui ruuteri filtreerimistabelis on määratud sobiva seadme olemasolu. KNXnet/IP ruutimise protokolli alusel saadetakse telegramm interneti protokolli kaudu võrku ainult siis, kui ruuteri filtreerimistabelis on määratud ruuteri aadress vastavalt grupiaadressile, millele saadetav telegramm on mõeldud. [7] KNX lahendustes kasutatakse ruutimist magistraalliinina, kuna internetiühendus võimaldab kiiremat ühendust pealiinide vahel. Ruutimise kasutamiseks peab olema KNX lahendusse lisatud IP ruuter. [6]

Autori lahenduses on kasutatakse tunnelühendust, kuna lahenduskäigus on mõeldud lahenduse turvalisuse ja töökindluse peale. Tunnelühenduse krüpteeritud ühendus tagab turvalisuse ning ühendus kahe seadme vahel takistab liigse käskude ahela teket, mis tagab töökindluse.

3.2 Tarkvarateegid ühenduse loomiseks

Selleks, et saaks virtuaalset seadet luua, oleks vaja kuidagi luua ühendus füüsilise lahendusega. Ühenduse loomiseks on loodud mitmed erinevad tarkvarateegid, mis haldavad kogu ühenduse protsessi. Käesoleva töö raames on autor uurinud kahte tarkvarateeki: Java teek „Calimero Project“ [8] ning C# ja C++ teek „Falcon SDK“ [9]. Mõlemad teegid võimaldavad luua nii tunnelühendust kui ka ruutimisühendust ning võimaldavad siini jälgimist, süsteemi tuvastust ja seadme haldamist. Katsetuste tulemusena on autor jõudnud järeldusele, et funktsionaalsuse ja kasutuse poolelt on mõlemad teegid vägagi sarnased. Suurim erinevus seisneb programmeerimiskeeltes, milles neid teeke kasutada saab.

Suure sarnasuse tõttu on autor otsustanud Falcon SDK kasuks C# programmeerimise keele eelistuse tõttu.

4 Virtuaalsete andurite lahenduse realiseerimine

Sissejuhatuses analüüsis autor olukorda, kus on olemas automatiseeritud hoone lahendus aga lahenduse ülesseadmisel ei arvestatud hämarusanduri lisamisega. Nüüdseks peaks lugejal olema lihtne arusaam KNX standardil põhineva automatiseeritud hoone ehitusest ning ka erinevatest ühenduse loomise meetoditest, mille alusel saab luua virtuaalseid andureid. Kasutades eelnevates peatükkides analüüsitud informatsiooni, on autor loonud kaks erinevat rakendust virtuaalsete seadmete loomiseks ja kasutamiseks KNX hoone lahenduses.

Esimese rakenduse eesmärgiks on tõestada, et on võimalik luua ja kasutada virtuaalseid andureid automatiseeritud hoone lahenduse juhtimiseks. Antud rakendus kujutab endast lihtsamat hämarusandurit, mis rakendab internetist päritud päikese informatsiooni valgustite kontrollimiseks.

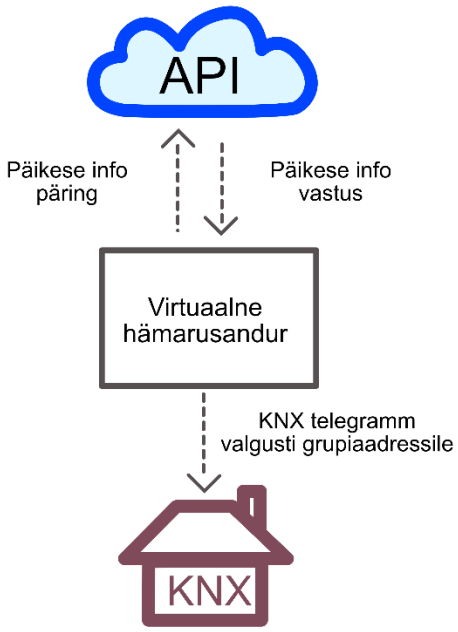
Esimene rakenduse lahendus tõestab ära, et on võimalik rakendada virtuaalsete andurite loogikat, kuid selle rakendamisel peab omama teadmisi KNX standardist. Sellest tulenevalt tekkis autoril küsimus: Kas kuidagi on võimalik luua rakendus, mis lubab arendajatel luua virtuaalseid andureid nii, et arendaja ei pea omama täielikke teadmisi KNX tarkvarateekidest? Teine autori poolt loodud rakendus vastab sellele küsimusele.

Teine autori poolt loodud rakendus on RESTi liides, kus RESTi meetodite välja kutsumine kasutab KNX tarkvarateegi funktsionaalsust käskluste edastamiseks automatiseeritud hoone lahendusse. Selle rakenduse kasutamisel ei pea otseselt teadma, kuidas KNX tarkvarateegid töötavad vaid kogu vajalik informatsioon on automatiseeritud hoone seadistus, mis loetakse välja andmebaasist. Antud rakendus muudab KNX lahendustele rakenduste loomise lihtsamaks ja arendajasõbralikumaks.

Mõlemad rakendused on loodud Visual Studio keskkonnas kasutades C# programmeerimiskeelt, .NET raamistikku ja Falcon SDK tarkvarateeki. Teine rakendus kasutab lisaks eelnevalt nimetatule Swagger tarkvarateeki RESTi liidese funktsionaalsuse dokumenteerimiseks, Microsoft Azure keskkonnas loodud Azure SQL andmebaasi ja Docker konteineri loomise funktsionaalsust lihtsaks seadistamiseks.

4.1 Virtuaalne hämarusandur

Tõestamaks, et on võimalik luua virtuaalseid andureid KNX standardile vastavale automatiseeritud hoone lahendusele, on autor loonud väikese konsoolirakenduse, mis funktsionaalsuse poolest emuleerib hämarusandurit. Joonisel 9 kujutatud rakendus loeb sisse päikese informatsiooni internetist, kasutades Sunrise-Sunset API käsklusi ning päikese tõusu ja loojumise aja järgi lülitab peatükis 2.3 kirjeldatud näitesüsteemis valgustit sisse ja välja. Joonisel 10 on kujutatud klassi diagramm, mille baasil on kirjeldatud rakenduse töökäiku.



Joonis 9 Virtuaalse hämarusanduri lahendust visualiseeriv mudel

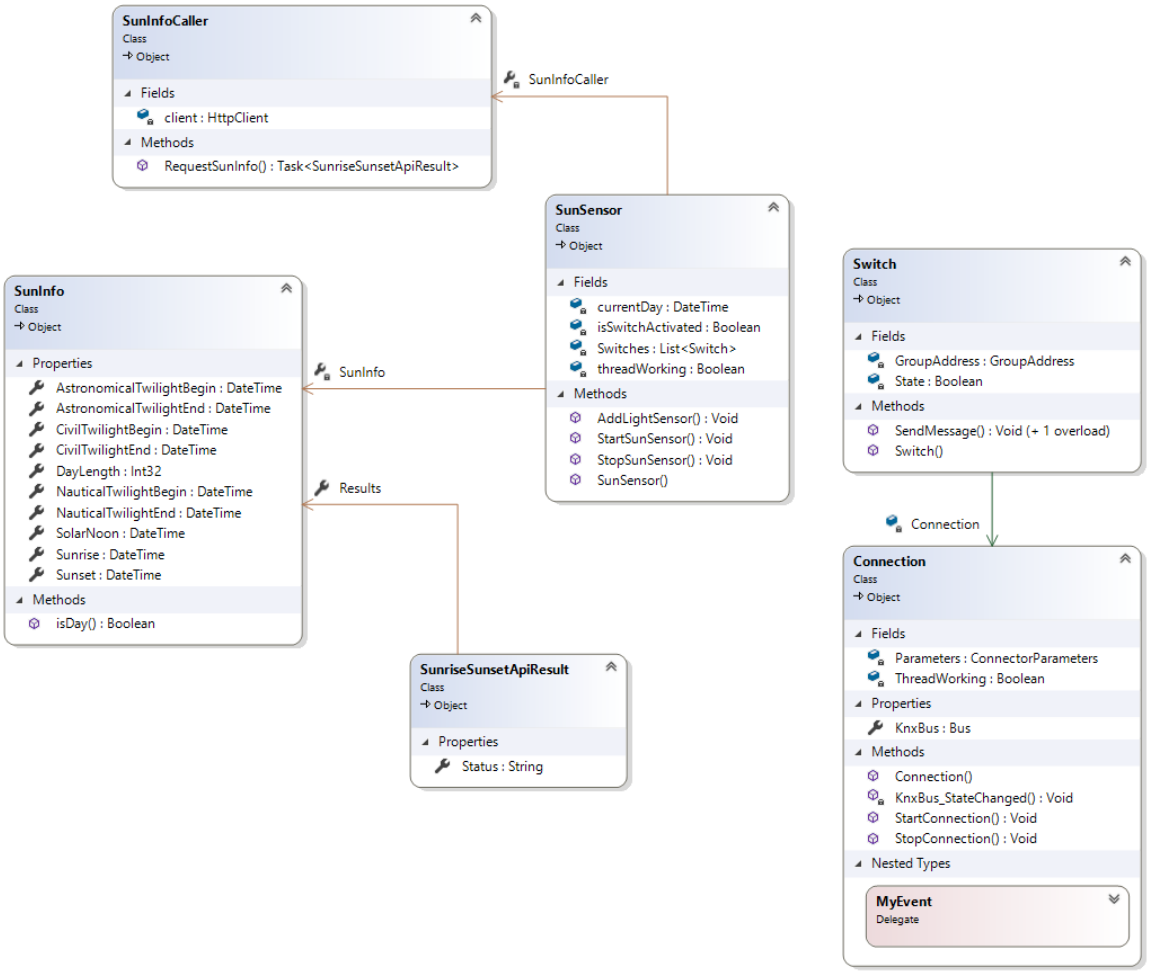
Rakenduse käivitamisel luuakse algselt ühenduse loomise jaoks lõim, kasutades „Connection“ klassi objekti. „Connection“ klassi eesmärk on hoida elus tunnelühendust automatiseeritud hoone lahendusega. Selle jaoks, et ühendust luua, antakse „Connection“ klassi objektile sisendiks ühenduse parameetrid, mis koosnevad füüsilise lahenduse IP aadressist, ühendusele mõeldud pordist ja võrguaadresside teisenduse kasutuse vajaduse tõeväärtusobjektist. Ühenduse elus hoidmiseks, jälgib selle klassi lõim pealt ühenduse seisundit. Juhul kui ühendus peaks sulguma, toimub ühenduse seisundi muutus ja selle

tuvastamisel luuakse ühendus uuesti kasutades objekti loomisel sisendiks antud parameetreid.

Kui ühendus on loodud, luuakse järgmine objekt, milleks on „SunSensor“ klassi objekt. Selle klassi eesmärgiks on kasutada päikesetõusu ja loojangu informatsiooni valgustite sisse ja välja lülitamiseks. Objekti loomisele järgnevalt tuleb lisada objektile „Switch“ klassi objektid vastavalt valgustite kontrolli grupiaadressidele. Kui „SunSensor“ objektiga lõim käivitada, kontrollib antud lõim iga 100 millisekundi järel, kas vaja on muuta lüliti väärtust vastavalt kellaajale ja Sunrise-Sunset API poolt saadud informatsioonile. Näiteks, kui tegemist on päevase kellaajaga ja „Switch“ objekti „State“ väärtus näitab, et tõest väärtust, saadetakse „Switch“ objekti kaudu KNX telegramm automatiseeritud hoone lahendusse, kasutades eelnevalt loodud „Connection“ objekti ühendust. Öisel kellaajal lülituse toimumiseks peab „Switch“ klassi objekti „State“ võrduma väärtusega väär (ingl. k *false*), kuna sellisel juhul ei ole virtuaalses anduris toimunud lüliti oleku muudatus sisse lülituseks. Lülituse toimumisel, muudetakse „State“ muutuja väärtus tõeseks, mis näitab, et öisel perioodil on lüliti staatus muudetud sisse lülituseks. Lõime poolt toimuv kontroll on 100 millisekundi peale määratud, kuna pikemat ajaperioodi kasutades toimub rakenduse sulgedes lõime sulgemine liiga hilja ja võib juhtuda, et ühenduselõime ei suleta. Kui ühenduselõim jääb sulgemata, siis ei saa hoone lahendusega enne ühenduse aegumist ühendust luua.

Eelnevalt mainitud „Switch“ klassi objekt on loodud lülituse emuleerimiseks. „Switch“ klassi objekti loomisel on vaja anda argumendiks ühenduse objekt, läbi mille käib lüliti funktsionaalsus. Lülitusel saadetakse KNX telegramm, kasutades Falcon SDK „Bus“ objekti „WriteValue“ meetodit. Selle meetodi argumendiks on antud grupiaadress ja lüliti soovitud asendi tõeväärtusobjekt. Argumendiks antud grupiaadress peab vastama automatiseeritud hoone lahenduses olevale grupiaadressile, mida kasutatakse lüliti funktsionaalsuse käivitamiseks. Antud lahenduses on lülitiga seotud grupiaadress määratud LED lambi sisse ja välja lülitamiseks.

„SunSensor“ klassi objekti puhul sai mainitud Sunrise-Sunset API poolt saadud informatsiooni. Selle informatsiooni saamiseks on loodud abiklass nimega „SunInfoCaller“. Antud klassi objekti eesmärk on saata päring Sunrise-Sunset API serverisse ja vastuseks saadetud informatsioon kasutada virtuaalse hämarusanduri lahenduses „SunriseSunsetApiResult“ ja „SunInfo” objektide kaudu.



Joonis 10 Lihtsama anduri lahenduse klassi diagramm

Eelnevalt kirjeldatud rakendust kasutades, on näha, et on võimalik luua virtuaalset seadet, millega saab kontrollida KNX standardile vastavat automatiseeritud hoone lahendust.

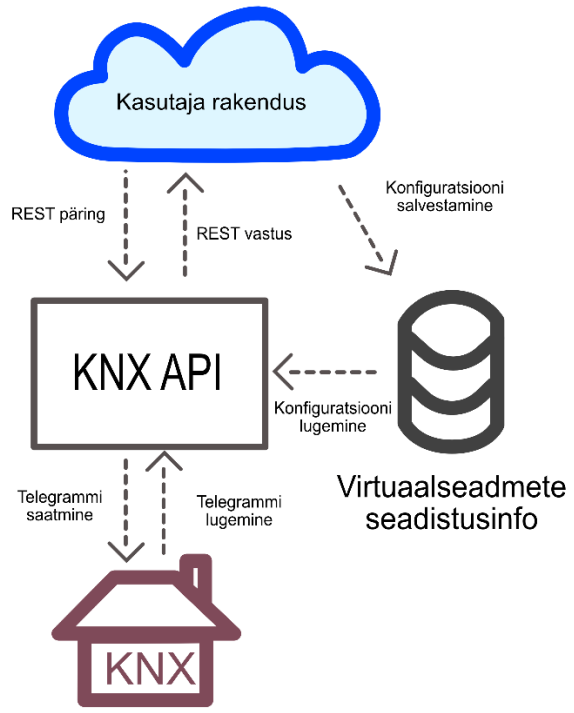
4.2 REST API virtuaalsete andurite loomiseks ja kasutamiseks

Teine lahendus kujutab endast REST API rakendust, mille eesmärgiks on lihtsustada virtuaalsete seadmete loomise protsessi, asendades virtuaalse anduri lahendusest KNX standardile vastava ühenduse loogika veebiteenusele vastavate päringutega.

Lahenduse loomisel on lisaks kasutatud Docker konteinerite tehnoloogiat ja SQL andmebaasi. Docker konteinerite tehnoloogia võimaldab antud rakendust seadistada lokaalsesse või pilve seadmesse vähese vaevaga. SQL andmebaas võimaldab turvalisemat automatiseeritud hoone lahenduse seadistuse kasutamist, kuna andmebaasi kasutades saab REST päringutes kaasa anda seadistuse informatsiooni asemel

andmebaasi viited seadistuse informatsioonile. Tänu Dockeri ja andmebaasi toele, on võimalik edasiarendusena luua SaaS mudelil põhinev veebirakendus, kus mitmed kliendid saaksid juhtida oma automatiseeritud hoone lahendusi kasutades autori poolset API-t. Kuna leidub ka inimesi, kes ei usalda oma hoonete lahendusi pilve põhisesse rakendusse näiteks turvariskide poolest, on rakendust võimalik kasutada ka lokaalses serveris.

Lahenduse töökäiku on kujutatud joonisel 11. Päringu tegemisel antakse kaasa andmebaasi viited, mille alusel on süsteem suhtleb automatiseeritud hoone lahendusega. Vastavalt päringu tüübile loetakse automatiseeritud hoone lahendusest informatsioon sisse või saadetakse hoone lahendusse funktsionaalsust käivitav telegramm. Seejärel saadetakse rakendusest välja vastus vastavalt päringule. Kui tegemist on funktsionaalsust käivitava päringuga, siis vastatakse lühidalt, et valitud ülesanne on täidetud. Informatsiooni sisse lugeva päringu puhul lisatakse vastusesse hoone lahendusest sisse loetud informatsioon. Vea tekkel edastatakse vastuses vea kirjeldus.



Joonis 11 REST liidese lahendust visualiseeriv mudel

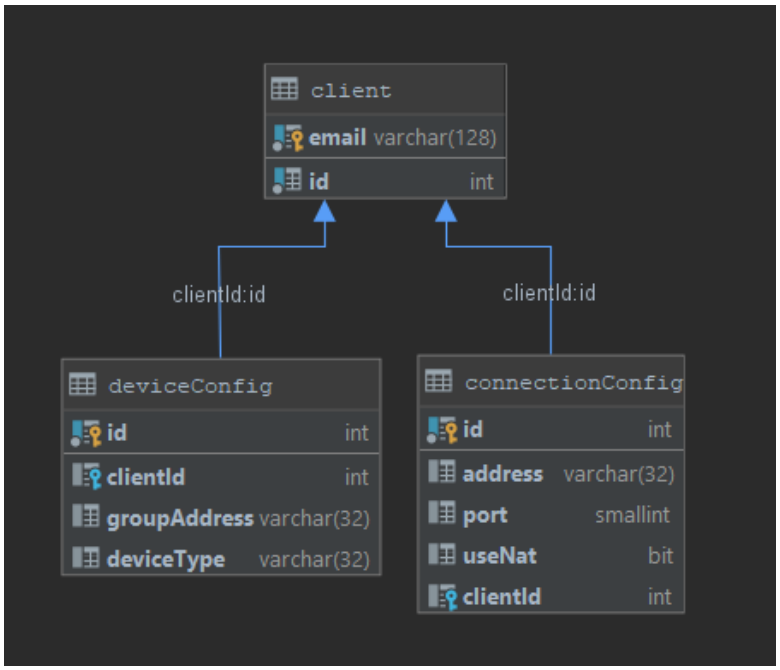
Loodud rakenduse seletuse saab jagada neljaks:

- Andmebaasi seletus
- Ühenduse loomise seletus
- Andurite funktsionaalsuse käivitamise seletus
- Rakenduse seadistamine ja kasutamine

4.2.1 Andmebaas

Antud rakenduses on kasutusele võetud andmebaas turvalisuse eesmärgil. Andmebaasi kasutusega saab vältida hoone lahenduses kasutusel olevate parameetrite leket andes REST päringutega kaasa andmebaasi viited. Lisaks on võimalik andmebaasi kasutamisel luua SaaS arhitektuuril põhinev rakendus, mida on plaanis luua edasiarendusena. Andmebaasis talletatakse iga kasutaja seadistuse informatsioon ja see võimaldab kasutada teenust sõltumatult teistest kasutajatest.

Rakenduses kasutatud andmebaas on realiseeritud kasutades kolme tabelit. Andmebaasi tabelid on kujutatud joonisel 12.



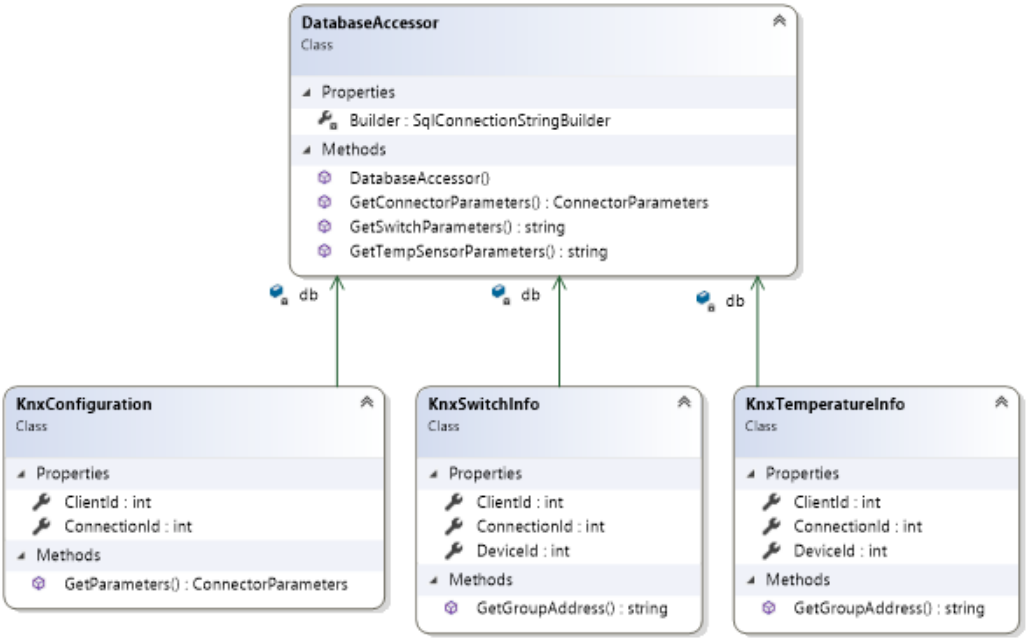
Joonis 12 Andmebaasi skeem

Tabel nimetusega „client“ on mõeldud kliendi andmete talletamisega ning seob kliendi automatiseeritud hoone lahenduse seadistuse kliendi andmetega kasutades primaarset võtit nimega „id“.

Tabel „deviceConfig“ on mõeldud hoone lahenduse automatiseerimise seadetega. Andmeväli „groupAddress“ omab väärtust hoone lahenduses kasutatud grupiaadressi väärtust, mida kasutades luuakse telegrammid funktsionaalsuse täitmiseks. Väli „deviceType“ on loodud tähistamaks talletatud grupiaadressi funktsiooni.

Tabelis „connectionConfig“ hoitakse ühenduse loomiseks vajalikku informatsiooni. Väli „address“ omab automatiseeritud hoonega ühendamiseks vajalikku IP aadressi. Väli „port“ määrab ühendusepordi, mille kaudu ühendus luuakse ja väli „useNat“ on määrab ära, kas antud ühenduse loomiseks on vaja kasutada võrguaadresside tõlkimise tehnikat.

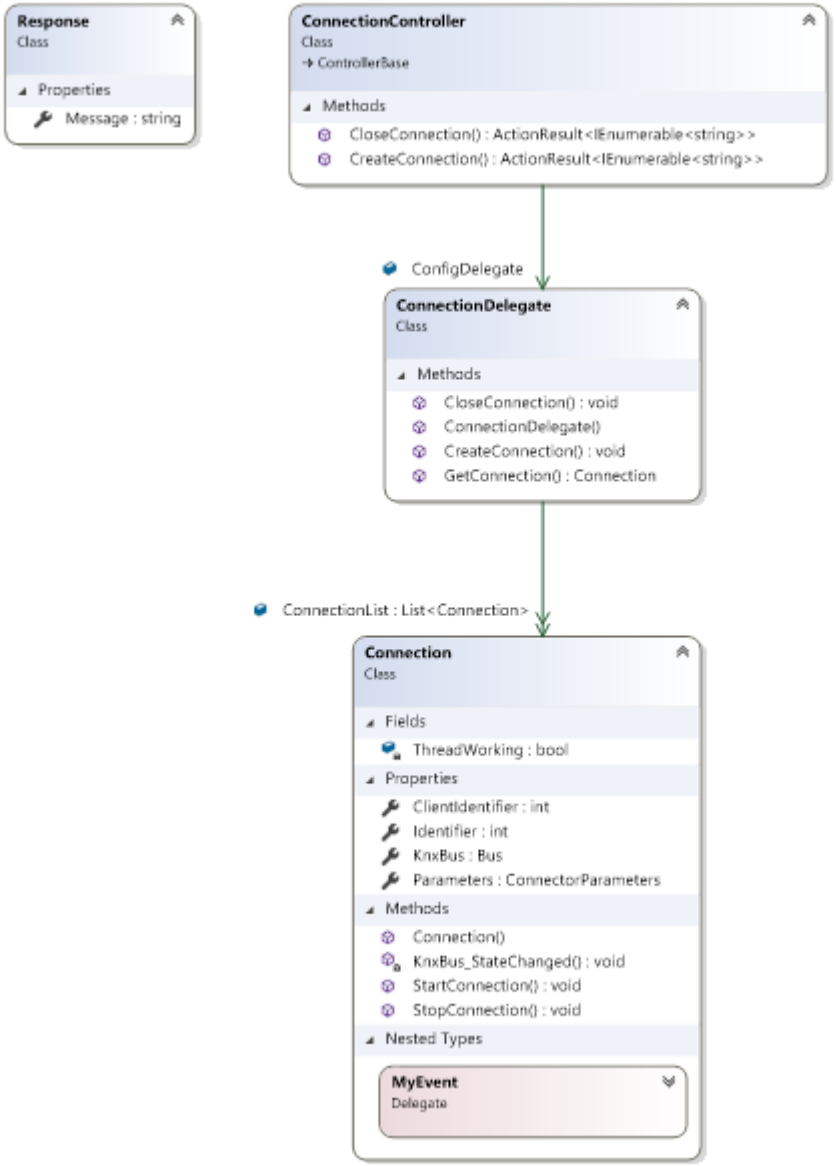
Rakenduses kasutamiseks on iga andmebaasi tabeli jaoks loodud eraldi objektid. Need objektid on kasutusel REST päringu sisendiks. Päringu kaudu antakse kaasa viited tabeli informatsioonile ning kasutades viiteid ja „DatabaseAccessor” klassi, loetakse sisse funktsionaalsuse jaoks vajalik informatsioon andmebaasist. Andmebaasist lugemise funktsionaalsus on kujutatud järgneval klassidiagrammil.



Joonis 13 Andmebaasi klassidiagramm

4.2.2 Ühendumine automatiseeritud hoone lahendusega

Hoonega ühendumise jaoks on kasutatud modifitseeritud lahendust esimesest rakendusest. Ühenduse tegevused on lahti seletatud kasutades joonisel 14 kujutatud klassidiagrammi.

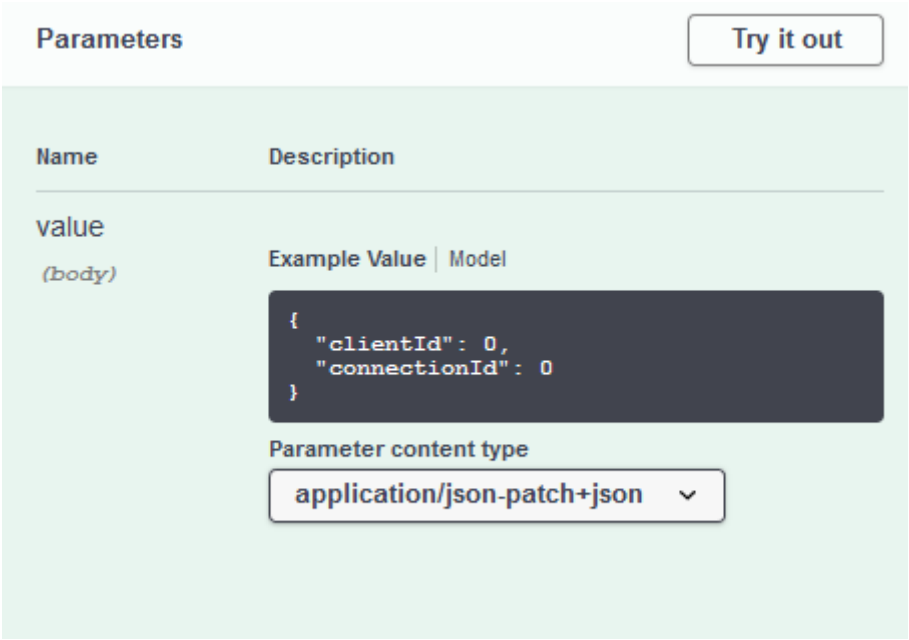


Joonis 14 REST liidese klassidiagramm ühenduse loomiseks

Ühenduse loomiseks on loodud kaks päringut: create_connection ja close_connection. (Joonis 15)

Joonis 15 Ühenduse REST päringud

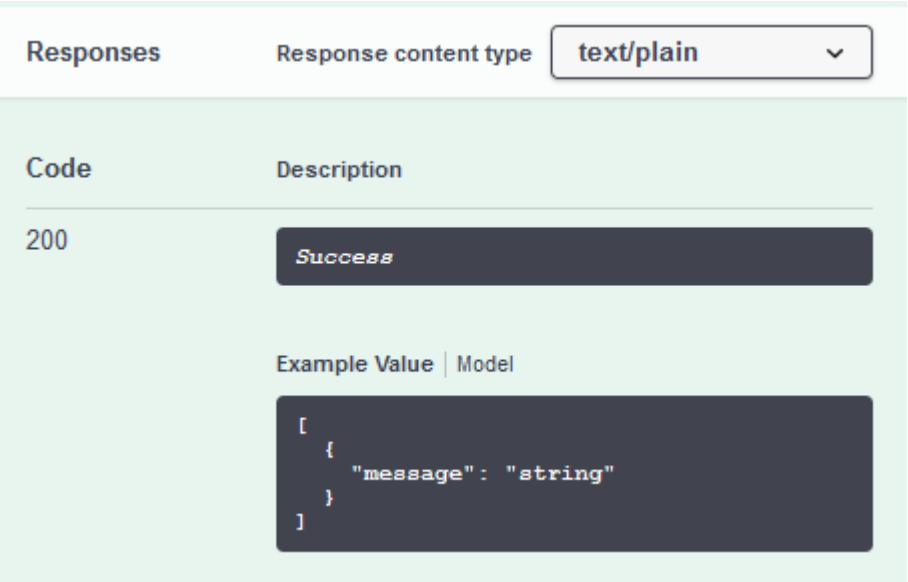
Mõlema päringu tegemisel on vaja anda kaasa JSON andmetüübi kujul objekt, mis koosneb „clientId“ ja „connectionId“ väärtusest. Nende väärtuste järgi loetakse ühenduse loomisel andmebaasist. Objekti ülesehitus on kujutatud joonisel 16.



Joonis 16 Ühenduse päringute JSON objekti kuju

Päringu create_connection tegemisel loetakse sisse andmebaasist vajalikud parameetreid, ning „ConnectionDelegate“ klassis luuakse uus „Connection“ objekt, mis funktsionaalsuse poolest käitub sama moodi nagu esimeses lahenduses kirjeldatud „Connection“ objekt. Loodud „Connection“ klassi objekt talletatakse „ConnectionDelegate“ klassis staatilises listis, et oleks võimalik ühenduda rohkema kui ühe automatiseeritud hoone lahendusega. Korduva ühenduse loomise katsete takistamiseks ühe ja sama hoone parameetritega, on olemas kontroll, mis tuvastab, kas vastavatele parameetritele vastav hoone lahenduse ühendus on eelnevalt loodud. Kui vastav ühendus eksisteerib „ConnectionDelegate“ klassis olevas listis, kuid ühendus puudub, siis taaskäivitatakse ühendus. Käsujada lõppedes saadetakse tagasi vastus,

kinnitades meetodi korrektset käitumist. Vea korral sisaldab vastust veakirjeldust. Vastuse JSON objekti kuju on nähtaval joonisel 17.

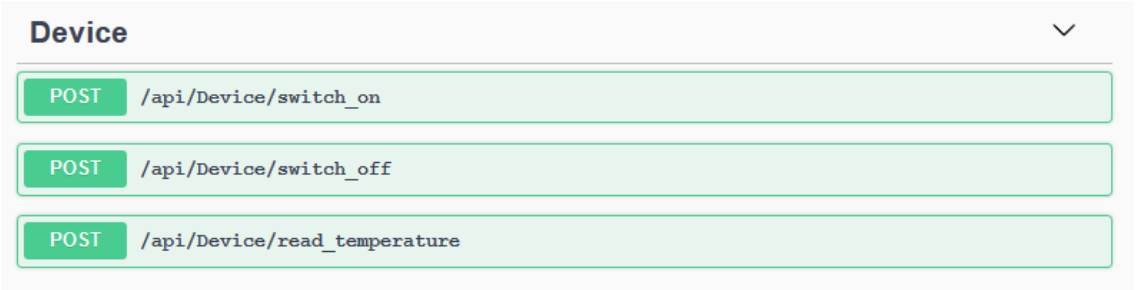


Joonis 17 Vastuse JSON objekti kuju

Päringu close_connection tegemisel antakse samuti kaasa andmebaasi viited, mille järgi tuvastatakse olemasoleva ühenduse klass „ConnectionDelegate“ klassiobjektis olevast listist. Ühenduse objekti leidmisel suletakse objektiga seotud lõim ning eemaldatakse avatud ühenduste listist.

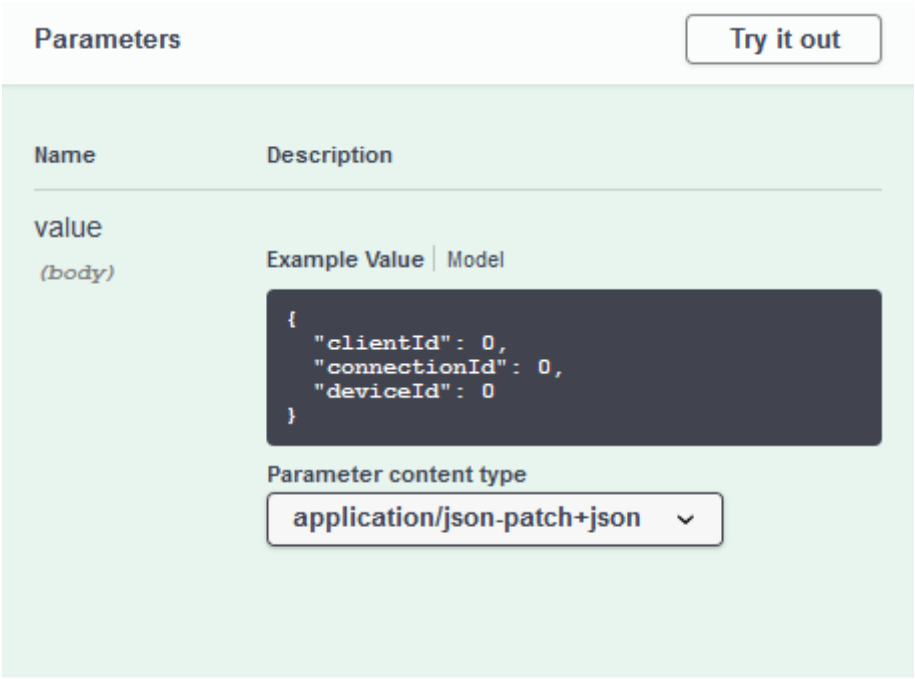
4.2.3 Virtuaalsete andurite kasutamine automatiseeritud hoone lahendusega

Andurite funktsionaalsuse rakendamiseks automatiseeritud hoone lahenduses on antud lahenduses loodud kahte sorti seadmete jaoks kokku kolm teenust, mis on kujutatud joonisel 18. Lüliti funktsionaalsuse täitmiseks on kasutusel päringud „switch_on“ ja „switch_off“. Temperatuuri anduri lugemiseks on kasutusel „read_temperature“ päring. Päringud on mõeldud kliendi rakendusega suhtluse korraldamiseks, kus kliendi rakendus esitab päringuid autori poolt loodud lahendusele. Päringud on esitatud POST meetodi kujul, kuna POST päringuid, võrreldes GET päringutega, ei talletata veebilehitseja ajaloos ega mälus ning selle tõttu on kasulikum tundliku informatsiooni saatmiseks. [10]



Joonis 18 Automatiseeritud hoone kontrollimise REST päringud

Kõigi päringute käivitamisel on kaasa anda JSON tüüpi andmeobjekt. Andmeobjekt on nende kolme päringu jaoks ühesugune. Andmeobjekt on kujutatud joonisel 19.

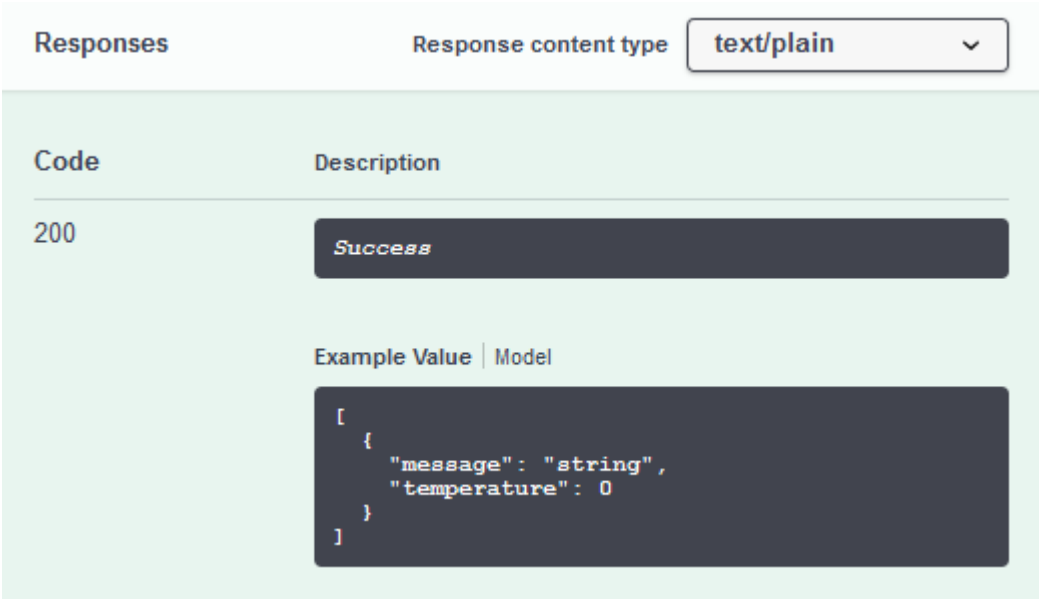


Joonis 19 Seadme päringute JSON objekti kuju

Päringute käivitamisel kasutatakse andmebaasi funktsionaalsust parameetrite sisse lugemiseks ning käivitatakse funktsionaalsus „DeviceDelegate“ klassi objektis. „DeviceDelegate“ klassiobjektis kutsutakse välja meetod vastava päringu järgi. Näiteks kasutades päringut „switch_on“, kasutatakse „DeviceDelegate“ klassi meetodit „SwitchOn“ kus vastavalt kaasaantud muutujatele, päritakse „ConnectionDelegate“ objektist vastav „Connection“ objekt. „Connection“ objekti kasutatakse lüliti telegrammi loova klassi „Switch“ meetodi „SwitchOn“ argumendina, koos andmebaasist loetud grupiaadressina, et luua kindlale ühendusele telegramm. Loodud telegramm informeerib automatiseeritud hoone lahenduses lülitama sisse seadmeid, mis on seadistatud kasutatud

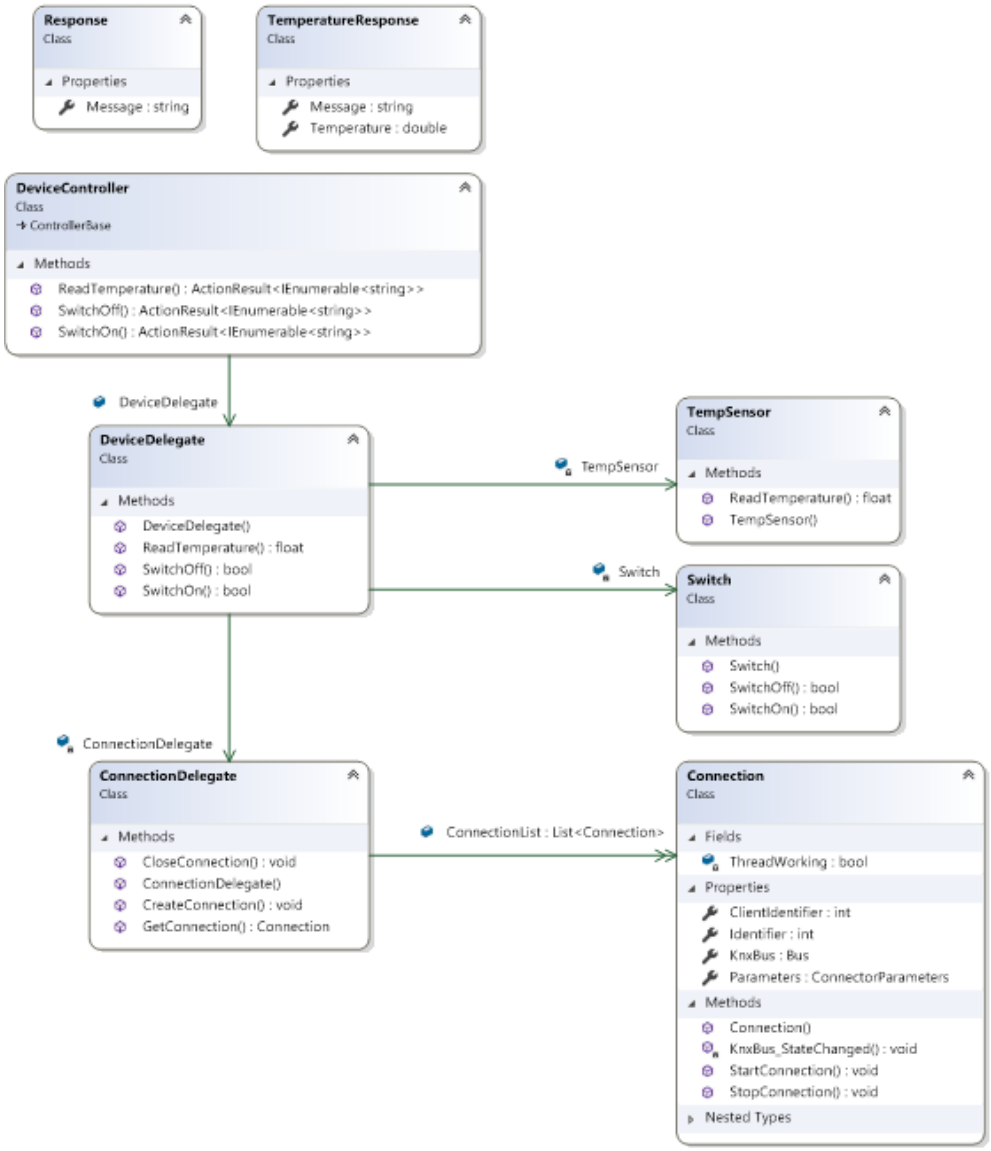
grupiaadressi alla. Päring „switch_off“ käitub sarnaselt, kuid sisse lülitava telegrammi asemel saadetakse hoone lahendusse välja lülitav telegramm. Peale telegrammi saatmist saadetakse päringu loojale sõnum JSON objekti kujul, mis on näidatud joonisel 17.

Temperatuuri lugemise funktsionaalsus loodi kompleksandurite loomise eesmärgil. Kompleksandurite all on mõeldud andureid, mis soovivad kasutada nii hoone lahenduses kasutatavat informatsiooni koos informatsiooniga, mis on võetud välisest keskkonnast nagu näiteks internetist. Automatiseeritud hoone lahenduses kasutatava temperatuuri anduri informatsiooni lugemiseks on kasutatud meetodit, mis käitub sarnaselt telegrammi saatmise meetodile, kuid eelnevalt saadetakse vastavale grupiaadressile informeeriv telegramm, et anduri informatsiooni saaks lugeda. Järgnevalt kasutatakse meetodit, mis kuulab vastavalt grupiaadressiga välja saadetud informatsiooni. Informatsioon saabub KNX standardile vastava „DataPoint“ andmetüübina, mis teisendatakse „float“ andmetüüpi kasutuse hõlbustamiseks. Vastuse kättesaamisel tagastatakse informatsioon päringu saatjale „TemperatureResponse“ objekti kujul, mis sisaldab sõnumit kinnitamaks funktsionaalsuse täitmist ja lisaks ka „float“ tüüpi temperatuuri väärtust. „TemperatureResponse“ JSON objekti kuju on näidatud joonisel 20



Joonis 20 Temperatuuriga vastus JSON objekti kujul

Nii lüliti kui ka temperatuuri anduri kasutus on kujutatud joonisel 21.



Joonis 21 Anduri funktsionaalsuse klassidiagramm

4.2.4 Rakenduse seadistamine ja kasutamine

Autori poolt kirjutatud rakendus on loodud kasutades Docker konteinerite tehnoloogiat ja ka Azure SQL andmebaasi. Docker konteinerite tehnoloogia võimaldab rakendust käivitada igas Docker rakendust jooksutavas seadmes väga vähese vaevaga. Lisaks on Docker funktsionaalsuse tõttu võimalik rakendust kasutada ka serveri keskkonnas tunduvalt lihtsamini. Andmebaasi kasutuse tõttu on rakenduse korrektse töötamise jaoks vaja tuua sisse mõningad muudatused. Esmalt on vaja ära muuta andmebaasi ligipääsu konfiguratsioon. Antud konfiguratsiooni andmed on „DatabaseAccessor“ klassis. Need tuleb muuta sobivaks vastavalt soovitud andmebaasi kasutamiseks. Lisaks, kui kasutada

mõnda muud andmebaasi süsteemi, on vaja ümber kirjutada ka olemasolevad SQL laused, mis on samuti leitavad „DatabaseAccessor“ klassis.

Olles muutnud andmebaasi konfiguratsiooni sobivaks, on võimalik ehitada Docker konteiner rakendusest. Autor kasutas selle jaoks Visual Studio Docker funktsionaalsust. Nimetatud funktsionaalsus võimaldab luua vähese vaevaga käivitatav Docker konteiner, mille seadistused on projektis Visual Studio rakenduse poolt genereeritud.

Docker konteineri käivitamisel, avaneb veebilehitseja aken Swagger UI rakenduse vaatega. Selles vaates on kuvatud kõik olemas olevad REST liidese päringud ning neid on võimalik manuaalselt välja kutsuda ja testida. Samuti on kõiki päringuid võimalik luua kasutades väliseid rakendusi nagu Postman ja ka arendaja loodud rakendus REST liidese kasutamiseks. Kõik autori poolt loodud päringud on nähtaval joonistel 15 ja 18.

Päringute tegemisel tuleb kaasa anda JSON formaadis andmed, mis kujutavad endast andmebaasi viiteid objektidele. JSON formaadis sisendid on kujutatud joonisel 22.

Joonis 22 JSON tüüpi sisendid päringutele

Kõikidel sisenditel on kaasa antud kliendi id ja ühenduse id. Kliendi id on kasutusel andmebaasist kliendile vastavate seadistuste leidmiseks. Ühenduse id on ühenduse päringutes kasutusel ühenduse seadistuse leidmiseks andmebaasist. Ühenduse loomisel seotakse id vastavalt ühenduse objektile, sest kasutusel olevaid ühenduse objekte saab

olla ainult üks. Andurite päringute juures on ühenduse id ühenduse objekti tuvastamiseks. Lisaks on andurite päringutele juurde lisatud ka seadme id, mille kaudu loetakse andmebaasist seadme info automatiseeritud hoone funktsionaalsuse käivitamiseks.

5 Kokkuvõte

Bakalaureuse töö eesmärgiks oli analüüsida KNX standardi võimalusi virtuaalsete seadmete loomiseks ja realiseerida virtuaalsed seadmed vastavalt KNX standardile vastavalt automatiseeritud hoone lahendusse.

Bakalaureusetöö tulemusena valminud rakendused vastavad püstitatud eesmärkidele. Töö käigus ühendas autor kokku ja seadistas lihtsama KNX standardile vastava automatiseeritud hoone lahenduse ning lõi kaks rakendust. Esimene rakendus kujutab endast virtuaalset hämarusandurit, mis tõestab virtuaalse anduri loomise võimalikkust KNX standardile vastava automatiseeritud hoone lahendusele. Teiseks rakenduseks on REST liides, mis võimaldab kasutada mitut lihtsama kui ka keerulisema ehitusega virtuaalset andurit.

REST liidese rakenduse edaspidiseks kasutamiseks reaalsetes tingimustes, oleks vaja täiendada antud rakenduse funktsionaalsust. Lisaks on vaja rakendusele teha koormustestid mitme automatiseeritud hoonega ühendamiseks. Antud töö raames oli autoril võimalik luua ainult kahe seadme funktsionaalsus ja testida ainult ühe automatiseeritud hoone lahendusega olemasolevate riistvara komponentide olemasolu tõttu. Kui rakendus võimaldab ühendust rohkemate automatiseeritud hoone lahendustega, siis võiks rakendust kasutada tarkvaralise teenusena KNX hoone juhtimiseks.

6 Kasutatud kirjandus

[1] KNX Assotsiatsioon, „What is KNX? - KNX Association,“ KNX Assotsiatsioon, [Võrgumaterjal]. Available: https://www2.knx.org/ae/knx/association/what-is-knx/index.php. [Kasutatud 13 mai 2019].

[2] KNX Assotsiatsioon, „KNX eCampus,“ KNX Assotsiatsioon, [Võrgumaterjal]. Available: http://wbt5.knx.org/. [Kasutatud 14 05 2019].

[3] M. Dostojevski, „Kodu juhtimise lahendused KNX'iga,“ Tallinna Tehnikaülikool, Tallinn, 2016.

[4] D.-F. Pang, S.-L. Lu ja Q.-Y. Zhu, „Design of Intelligent Home Control System Based on KNX/EIB Bus Network,“ %1 *2014 International Conference on Wireless Communication and Sensor Network*, Wuhan, 2014.

[5] Cisco Systems Inc., „Tunneling - Cisco,“ Cisco Systems Inc., [Võrgumaterjal]. Available: https://www.cisco.com/c/en/us/products/ios-nx-os-software/tunneling/index.html. [Kasutatud 15 mai 2019].

[6] KNX Assotsiatsioon, *KNX Specifications v2.1,* KNX Assotsiatsioon, 2014.

[7] KNX Assotsiatsioon, „KNX Basics,“ [Võrgumaterjal]. Available: http://knx.fi/doc/esitteet/KNX-Basics_en.pdf. [Kasutatud 15 mai 2019].

[8] Calimero Project, „The Calimero Project,“ [Võrgumaterjal]. Available: http://calimero-project.github.io/. [Kasutatud 15 mai 2019].

[9] KNX Assotsiatsioon, „Falcon Software Development Kit,“ KNX Assotsiatsioon, 2019.

[10] Refsnes Data, „HTTP Methods GET vs POST,“ Refsnes Data, [Võrgumaterjal]. Available: https://www.w3schools.com/tags/ref_httpmethods.asp. [Kasutatud 19 mai 2019].

Lisa 1 – Kasutatud KNX seadmete nimekiri

- KNX Toiteplokk
    - Eelectron PS00D03KNX
- KNX IP ruuter
    - Weinzierl KNX IP Router 751
- KNX *dimmer* seade valgustitele
    - DiNUY RE-KNT-004
- KNX multifunktsionaalne andur, mis sisaldab liikumisandurit ja temperatuuriandurit
    - Basalte Auro
- Halogeenivaba keerdpaarjuhe