TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mihkel Reinart 083855IAPB

Targa ehitise siseruumide andmete visualiseerimine

Bakalaureusetöö

Juhendaja: Jaanus Kaugerand
Doktorant-nooremteadur

Tallinn 2021

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: Mihkel Reinart

04.01.2021

Annotatsioon

Antud lõputöö käigus kirjeldatakse rakenduse arendust, mis võimaldab visualiseerida olemasolevasse Thinnect OÜ platvormi kogutud siseruumide sensorite andmeid.

Töös antakse ülevaade rakenduse arendamisele eelnevast analüüsist alternatiivsete süsteemide kohta. Analüüsi tulemusena selgus, et hetkel puuduvad nõuetele vastavad lahendused. Seejärel analüüsitakse valminud rakenduse arendamiseks kasutatud tehnoloogiaid ja kirjeldatakse valminud rakenduse prototüüp lahendust.

Valminud üheleherakendus (SPA) on ehitatud Vue.js raamistikule ja kasutab andmete visualiseerimiseks hüpertekst edastusprotokolli (HTML) Canvas tehnoloogiat.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 32 leheküljel, 7 peatükki, 6 joonist.

Abstract

Smart building indoor data visualization

IoT solutions are becoming more and more affordable making smart buildings more common around us. Thinnect OÜ, a company that develops and implements sensor networks, is also activly working with smart buildings. One of the problems they face is a lack of open source software that would make it easy to visualize indoor sensor data.

The goal of this thisis is to descibe the development of an application that allows to visualize the indoor sensor data already stored in Thinnect platform.

This thisis gives an overview of the analysis of alternative solutions preceding the development cycle. The result of the analysis reveals that there are no existing solutions meeting all of the requirements. The Author then analyses the technologies used for developing the application and provides an overview of the prototype.

The final solution is a Single Page Application (SPA) built with Vue.js framework that uses Hypertext Transfer Protocol (HTML) Canvas technology to visualize data.

The application fetches all the data from existing Thinnect platform through an API and draws the sensor readings on the image displaying the floor plan. It is possible to filter the data by dates and by sensor types. The application also makes it possible to move back and forward in the selected timeframe by using a timeline component.

The thesis is in Estonian and contains 32 pages of text, 7 chapters, 6 figures.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendusliides.
CSS	*Cascading Style Sheets*, veebilehtede valmistajatele ja kasutajatele mõeldud laadistik.
CO_2	Süsinikdioksiid ehk süsihappegaas.
IDE	*Integrated Development Environment*, integreeritud programmeerimiskeskkond. Harilikult koosneb integreeritud programmeerimiskeskkond lähtekoodi redaktorist, kompilaatorist või interpretaatorist või mõlemast ning programmeerimise automatiseerimise abivahenditest.
IoT	*Internet of Things*, Asjade Internet, ka Värkvõrk või Nutistu, interneti kaudu seotud asjade võrk.
JSON	*JavaScript Object Notation*, programmeerimiskeelest sõltumatu andmevahetusformaat, mis pärineb JavaScriptist.
Heatmap	Kuumakaart, Andmete visualiseerimise tehnika, mis näitab väärtuse suurust kahemõõtmelises ruumis värvi abil.
HTML	*Hypertext Transfer Protocol*, hüperteksti edastusprotokoll.
Smart building	Tark ehitis. Hoone, mis on varustatud sensoritega võimaldamaks sisekliima, kütte, valgustus ja muude süsteemide automatiseerimist.
SPA	*Single page application*, üheleherakendus.

Sisukord

Jooniste loetelu

1 Sissejuhatus

Elektroonika ja infosüsteemide areng on jõudnud järjega piisavalt kaugele, et andmete kogumine meid kõiki ümbritsevatest keskkondadest on piisavalt taskukohane, tegemaks seda peaaegu kõikjal. Kõiksugu sensoritest saadava info analüüs ja selle rakendused ehitiste paremaks haldamiseks on aktuaalne teema juba praegu ja tõenäoliselt veelgi aktuaalsem järgmistel kümnenditel [1], sest nii on võimalik inimestel teha otsuseid parema informatsiooni baasil. Selleks, et suure hulga heterogeensete sensorite poolt toodetud andmeid inimestele mõistetavamaks teha, kasutatakse paljudes valdkondades andmete visualiseerimist [2][3].

Üks sensorite andmete kasutusvaldkond on "targad ehitised" (*smart buildings*). Ehitiste "targaks muutmine" (ehitiste elektroonikaga ühendamine) võimaldab hoida kokku energiat, muuta elanike eluolu mugavamaks, turvalisemaks ja efektiivsemaks [4].

1.1 Taust ja probleem

Üks firma, mis IoT lahendustega igapäevaselt tegeleb, on Thinnect OÜ [5]. Näiteks võimaldavad nende lahendused jälgida köökides ja kauplustes olevate külmikute temperatuure, et veenduda toidu ohutuses. Samuti mõõdavad nende seadmed linna tänavatel liikluse kiirust, tihedust ja müra, mille analüüsimisel on võimalik paremini linna keskkonda planeerida.

Samuti toimetab Thinnect tarkade majade valdkonnas, olles paigaldanud erinevaid sensoreid ka siseruumidesse, mõõtmaks näiteks temperatuuri, CO_2 sisaldust, ruumide kasutatavust liikumissensorite abil jms. Paraku ei ole Thinnect välja töötanud tarkvara, mille abil neid andmeid visualiseerida. Kuigi leidub teiste ettevõtete poolt loodud IoT seadmete kuvamiseks ja haldamiseks loodud rakendusi, siis kahjuks on neist enamus kinnise lähtekoodiga, mis muudab nende modifitseerimise raskeks või võimatuks.

1.2 Nõuded rakendusele

Töö eesmärgiks on luua rakendus, mis võimaldaks anda ülevaadet siseruumide sensorite poolt kogutud andmetest.

Käesoleva töö käigus loodud rakendus tegeleb olemasolevasse Thinnect platvormi kogutud andmete visualiseerimisega.

Kuna rakenduse eesmärk on visualiseerida olemasolevaid andmeid ja rakenduse kaudu uusi andmeid juurde ei genereerita, siis on võimalik kogu rakendus ehitada *front end* veebirakendusena.

1.2.1 Funktsionaalsed nõuded

Rakenduse abil peaks kasutaja saama teha järgnevaid tegevusi:

- näha hoone siseruumide plaani või selle mingisugust alamosa. Näiteks ühte konkreetset ruumi või mitmekorruseliste ehitiste puhul ühe korruse plaani
- valida ühe või mitu sensori tüüpi, mille kohta andmeid vaadelda
- valida konkreetne ajahetk või -vahemik, mille kohta sensorite kogutud andmeid vaadelda
- valides ajavahemiku, peab olema võimalik visualiseerida andmete muutused ajas

1.2.2 Mittefunktsionaalsed nõuded

On eeldatav, et erinevaid sensorite tüüpe ja erinevaid viise andmete visualiseerimiseks on rohkem kui antud töö raames on võimalik rakendusse implementeerida. Seega on rakenduse mittefunktsionaalseteks nõueteks järgnevad:

- süsteemi arhitektuur peab võimaldama lisada kihte uute sensoritüüpide visualiseerimiseks
- rakenduse lähtekood peab olema dokumenteeritud piisaval tasemel, et seda on võimalik tulevikus muuta ja täiendada
- kasutatud tehnoloogiate dokumentatsioon peab olema heal tasemel
- süsteem peab olema integreeritav Thinnecti teiste rakendustega
- lähtekood peab olema avalik

1.3 Ülevaade tööst

Antud bakalaureusetöö annab ülevaate nõuetele vastava rakenduse prototüübi arendamisest, selle käigus tekkinud probleemidest ja nende lahendustest. Samuti annab töö ülevaate alternatiivsetest süsteemidest, et anda selgem pilt rakenduse olemusest ja selgitada tehtud otsuseid.

Antud töö kirjeldab rakenduse esmast versiooni, mille raames visualiseeritakse ruumi CO_2 kontsentratsiooni ja ruumi kasutatavust mõõtvate sensorite andmed.

Valminud rakenduse lähtekood on leitav autori repositooriumist [6].

2 Ülevaade eksisteerivatest süsteemidest

Hetkel on teada suhteliselt vähe rakendusi, mis ruumide sisekliimat visualiseerida võimaldavad.

Avatud lähtekoodiga IoT lahendustele mõeldud süsteeme uurides jäid silma mõned, mis osaliselt nõuetele vastavad.

OpenRemote nimeline IoT platvorm tundus esmapilgul paljulubav. Platvormil on avatud lähtekood ja kasutusjuhtusid uurides jäi silma üks [7], mis kuvas CO_2 taseme muutuseid, täites sellega ka otseselt nõuetes ettenähtud sensoritüübi nõude. Kahjuks ei võimalda lahendus aga kuvada siseruumide korruseplaani ja samuti on lähtekoodi litsentsiks valitud GPLv3 (*GNU General Public License version 3*), mille kasutamine kommertsrakendustes on äärmiselt tülikas, sest kõik lähtekoodi muudatused tuleb dokumenteerida ja avalikustada.

Järgmine lootustandev IoT platvorm oli Thingsboard [8]. Thingsboard platvorm on samuti avatud lähtekoodiga, kuid Apache 2.0 litsentsiga, mis on kommerts rakenduste vaatepunktist märksa mõistlikum - muudetud tarkvara ei ole kohustatud sama litsentsi järgima. Thingsboard rakendus võimaldab äärmiselt lihtsalt seada üles ülevaatliku näidikupaneeli (ik *dashboard*) IoT seadmete jälgimiseks, kuid kahjuks puudus olemasolev lahendus siseruumide kuvamiseks. Samuti on platvorm üles ehitatud kasutades programmeerimiskeelena Javat, mis ei ole autori teada hetkel Thinnect meeskonna poolt kasutatavate keelte hulgas.

Lisaks leidis autor otsingu tulemusena veel mitmeid lahendusi. Osad neist tundusid omavat funktsionaalsust siseruumide plaanide kuvamiseks (https://www.infsoft.com, https://www.esri.com/) ja osad, mis olid avatud lähtekoodiga IoT platvormid (https://thinger.io), kuid süsteeme, mis täidaksid mõlemad neist nõuetest leida ei õnnestunud.

Autor julgeb otsingu tulemustele põhinedes väita, et käesoleva töö kirjutamise hetkel ei eksisteeri konkureerivat süsteemi, mis vastaks kõikidele esitatud nõuetele.

3 Kasutatud tehnoloogiad

Rakendus saab oma andmed Thinnecti API kaudu JSON formaadis. See tähendab, et puudub vajadus eraldi serveripoolse komponendi järele.

Tegemist on veebirakendusega, sest puuduvad platvormi spetsiifiliste funktsionaalsuste vajadused – kõik vajaminev on toetatud veebilehitsejates.

Rakendus on üles ehitatud Vue.js raamistikule. Programmeerimiskeelena on kasutatud Typescripti. Veebiliidese kasutusmugavuse tõstmiseks ja süsteemi visuaalseks disainiks on kasutatud Tailwind CSS raamistikku.

Tehnoloogiate valikul sai määravaks peamiselt kolm kriteeriumi:

- Esiteks soov integreerida valmiv rakendus võimalusel Thinnecti teiste rakendustega. Mõned firma teised projektid on valminud samu tehnoloogiaid kasutades. Tundub loogiline, et valides samad tehnoloogiad on arendajatel lihtsam koodibaasis navigeerida ja vajadusel täiendusi teha.
- Teiseks, valitud tehnoloogiad on kõik hetkel pigem populaarsed [9]. Nende kohta leidub palju teavet ja nende dokumentatsioon on väga hea.
- Kolmandaks, autori soov kasutada tehnoloogiad, mis oleksid tema jaoks madala õppimiskõveraga, kuid siiski osaliselt tundmatud.

3.1 Vue.js

Vue.js [10] on populaarne Javascripti raamistik kasutajaliideste ehitamiseks.

Tegemist on ühega kolmest hetkel kõige populaarsemast *front end* raamistikust – hetkel on kõige populaarsemaks *front end* raamistikuks ReactJs [11], millele järgnevad

olenevalt konkreetsest mõõdupuust kas Vue või Angular [12]. Kõik kolm raamistiku on vabavaralised ja nende lähtekood asub Github keskkonnas [13].

Võttes aluseks Githubi tähekeste põhist kasutajapoolset hinnangut [14] – tähekesi saavad anda Githubi kasutajad märkides niimoodi valitud repositooriumid enda jaoks tähtsaks, et projekti muudatustega kursis olla – on Vue samas nii Reactist kui Angularist rohkem jälgitav ehk populaarsem.

Vue, nagu ka teised tänapäevased raamistikud, püüab arendajat suunata koodi eraldama komponentideks. Komponendid võimaldavad hoida koodibaasis eraldatud osakesi, mida on lihtsam isolatsioonis vaadelda, mõista ja testida. See teeb komponentide implementeerimise ja ülejäänud rakenduses kasutuselevõtmise märksa lihtsamaks.

Rakenduse jaoks sai konkreetselt valitud Vue.js versioon 3. Vue 3 avalikustati küll alles 18. septembril 2020 ja seega on probleemne leida väliseid vabavaralisi komponente, mis aitaks rakenduse valmimisele kaasa.

Samas on Vue 3 eeliseks varasema versioon 2-ga võrreldes uus Composition API [15], mis võimaldab koodi veelgi konkreetsemalt loogilisteks alam-osadeks tükeldada. Rakenduse arendamisel on seda raamistiku võimalust läbivalt kasutatud.

3.2 Typescript

Typescript [16] on avatud lähtekoodiga programmeerimiskeel, mis on ehitatud Javascripti peale. See tähendab, et Typescripti kompileerimisel on tulemuseks Javascript ja seega saab Typescriptist kirjutatud rakendusi jooksutada täpselt samamoodi nagu otse Javascriptis kooditud rakendusi.

Typescript lisab omalt poolt toe tüüpidele – võimalik on defineerida objektide tüübi struktuure; defineeritud peavad olema nii funktsioonide argumendid kui ka tagastustüübid.

Kuna Javascript ise on nõrgalt tüübitud, siis on sagedased olukorrad, kus näiteks funktsiooni väljakutsuja annab kaasa argumendi, mida funktsioon tegelikult ei eelda ja

olukord tekitab seega vea. Tüüpide lisandumine võimaldab kompilaatoril arendajat teavitada kui koodis taolisi olukordi leidub.

Lisaks muudavad Typescripti tüübid arendaja töö lihtsamaks, sest arenduskeskkonnad (IDE) oskavad tüüpide põhjal automaatselt arendajat abistada, näiteks *autocomplete* võimaluse abil pakkuda tüübis leiduvaid väljasid.

Rakenduse kirjutamisel on Typescripti tüüpe kasutatud eelkõige API kaudu küsitavatele andmetele struktuuri andmiseks. Samuti on tüüpidega ära defineeritud liidesed (ik *interface*) rakenduse sisese informatsiooni hoidmiseks (Vuex *store states)*.

3.3 Tailwind CSS

Tänapäeva veebirakendused põhinevad kolmel baastehnoloogial, mille hulgast CSS on kasutusel veebilehtedele disaini lisamiseks [17]. Kuna rakenduse disain ei olnud otseselt ette seatud nõue ja autoril on kombeks CSSi kallal nokitseda kauem kui ajaliselt mõistlik, siis sai otsustatud kasutusele võtta ka mõni CSS raamistik.

Valituks osutus Tailwind CSS [18]. Tegemist on raamistikuga, mis reklaamib ennast kui utiliitidele keskendunud (*utily-first*) lahendusena. See tähendab seda, et raamistik ise ei määra mitte ühelegi veebilehe osale stiile. Selle asemel koosneb raamistik väga paljudest klassidest, millest igaüks on loodud ainult ühe kindla atribuudi määramisele. Näiteks selleks et tekst värvida valgeks on olemas *text-white* klass. Tausta siniseks värvimiseks on näiteks *bg-blue-500* klass. Ja kui arendajal on soov luua näiteks sinine, valge tekstiga ja ümarate nurkadega nupp, siis tulebki <*button*> elemendile lisada järgnevate väärtustega klasside atribuut: *class="text-white bg-blue-500 rounded"*.

Antud lähenemine on justkui vastuolus levinud tööstusharu praktikatega, mis ütlevad, et veebirakenduste puhul tuleks Javascript, CSS ja HTML hoida eraldatult, kuna neist igaüks tegeleb erineva aspektiga koodibaasist. Samuti tähendab antud lähenemine, et mõnede HTML elementide puhul lisandub kümneid erinevaid klasse, mis omakorda muudab HTML märgistuskeele natukene raskemini loetavaks.

Nende samade eelarvamustega lähenes raamistikule ka autor. Kasutamise käigus sai aga selgeks, et HTML lugemine ei muutunud oluliselt keerulisemaks. Küll aga oli uute stiilide lisamine märksa kiirem, sest jäi ära tavapärane nokitsemise osa tänu eeldefineeritud suurustele ja värvidele.

Kuigi kogemuse põhjal ei saa olla kindel, kas see raamistik sobiks suurematele ja nõudlikuma disainiga rakendustele, siis antud rakenduse puhul aitas antud raamistiku valimine kindlasti arendamisel aega kokku hoida.

3.4 HTML *Canvas*

Vajalikul kujul ruumide plaani ja sensori näitude joonistamiseks veebirakenduses oli kaalumisel mitu viisi.

Thinnecti eelnevad lahendused olid ehitatud kasutades LeafletJS raamistikku. Antud raamistik on eelkõige mõeldud kaardirakenduste visualiseerimiseks ja seega olid sel tegelikult juba olemas võimekused nii taustale plaani joonistamiseks kui ka koordinaatsüsteemis liikumiseks. Samuti sisse ehitatud pildi suurendamise ja vähendamise võimekus. Küll aga tundus antud raamistik eelkõige orienteeritud kaardirakenduste visualiseerimistele ja seal oli natuke liiga palju võimalusi, mida otseselt siseruumi visualiseerimise rakenduses vaja ei lähe.

Selle asemel osutus hetkel valituks otse HTML *Canvas*'el põhinev lahendus. Canvas API [19] on veebitehnoloogia, mis lubab HTML <*canvas*> elemendile joonistada Javascripti abil.

Canvas API puuduseks võib lugeda seda, et tegemist on rastergraafikaga. Rastergraafikas pildid näevad parimad välja originaal mõõtmetes. Pildi suuruse muutmisel võib pilt muutuda sakiliseks.

Eeliseks võib samas pidada seda, et *Canvas* API on äärmiselt lihtsalt mõistetav, tänapäevastes veebilehitsejates laialdaselt juba toetatud ja detailne graafika joonistamise API võimaldab joonistada just seda, mis vaja on.

4 Rakenduse arhitektuur

Tegemist on *Single Page Application* (SPA) rakendusega. SPA tüüpi rakendused laevad mällu esmase pöördumisega kogu töötamiseks vajaliku HTML, Javascripti ja CSSi. Järgnevate pöördumistega uuendatakse enamasti vaid lehe kuvamiseks vajaminev dokumendi sisu. Nii tehes välditakse kogu lehekülje uuesti laadimist – järgnevatele lehtedele navigeerimine on kiirem ja server peab saatma vähem andmeid.

Vue raamistikuga ehitatud SPA rakendused on vaikimisi struktureeritud nii, et äriloogika on eraldatud hoidlatesse (ik *store*), vaadetesse (ik *view*) ja komponentidesse.

4.1 Hoidla

Store ehk hoidla on tsentraalne andmete haldamise objekt. Vue standard hoidla kasutab Vuex teeki [20].

Vuex poolt kasutatav muster määrab reegli, et andmete hoidlasse kirjutamine peab alati toimima läbi kindlate sünkroniseeritud käitumisega meetodite. See tagab, et andmete muudatused on alati tehtud ennustataval viisil. Meetodeid, mis muudatusi teevad kutsutakse mutatsioonideks (ik *mutation*). Kuna mutatsioonid on sünkroniseeritud toimingud, siis on tavaline, et nende käivitamiseks kasutatakse asünkroonseid meetodeid. Neid kutsutakse tegevusteks (ik *action*).

Andmete hoidlast küsimiseks on kasutusel *getter* meetodid, mis enamasti tagastavad hoidlast otse mõne andmeelemendi või hulga, kuid võivad vajadusel väärtuse genereerida ka hoidlas leiduvate andmete põhjal.

Antud rakenduses on hoidla eraldatud kaheks kihiks.

Esimene kiht koosneb eraldatud alamhoidlatest, millest igaüks suhtleb ühe või kahe API otsaga (ik *endpoint*). Iga selline alamhoidla defineerib talletavate andmete struktuuri, tegevused, mutatsioonid ja *getter*'id.

Eraldi hoidlad on olemas näiteks kasutaja autentimiseks, sensor-seadmete andmete ja erinevate mõõtmissündmuste talletamise jaoks.

Teine kiht tegeleb lehel olevate komponentide olekute talletamisega ja jagab vastavalt kasutaja tegevustele esimese kihis olevatele alamhoidlatele käsklusi andmete pärimiseks. Siin hoitakse informatsiooni näiteks selle kohta, mis ajavahemik ja missugused sensoritüübid on hetkel kasutaja poolt valitud.

4.2 Vaade

View ehk vaade ehk lehekülg on Vue rakendustes komponent, mis omab infot selle kohta, mis andmeid ja alamkomponente peab antud lehekülje jaoks kuvama.

Antud rakenduse peamine vaade seob omavahel filtrites kasutatavad komponendid ja ruumide plaani ning sellele sündmusi visualiseeriva *canvas* elemendi komponendi.

4.3 Komponendid

Komponendid on *front end* raamistikes kõige tavapärasemad koodihaldus üksused, mis koosnevad enamasti märgistuskeeles defineeritud mallist, Javascriptis kirjutatud loogikast ja CSSiga lisatud stiilidest.

Vue komponendid koondavad kõik need kolm osa samasse faililaiendiga ".vue" lõppevase faili. Selles failis kirjeldatakse <*template*> elemendis HTML koos Vue-põhiste attribuutidega. Loogika läheb <*script lang="ts"/*> elemendi sisse. Seejuures atribuut *lang* märgib ära, et kasutatakse Typescripti. Juhul kui komponent vajab stiile, siis need lisatakse faili lõppu <*style*> elemendi sisusse.

Loodud rakenduses on mitmeid komponente. Nendest kaks tähtsamat on EventSequencePlayer ja MapCanvas.

EventSequencePlayer komponent võtab sisendiks talle läbi vaate ja hoidla edastatud andmeid sündmustest, mis päritakse API-st. Sündmuste andmed töödeldakse nii, et neid oleks võimalik läbi mängida ajateljel. Juhul kui ajavahemikku kuulub rohkem kui üks sündmus, siis toob komponent nähtavale *play* ja *pause* nupud, mis võimaldavad ajateljel liikumist käivitada ja peatada. Lisaks kuulab komponent klikke ajateljel ja oskab nende põhjal teljel edasi ja tagasi hüpata.

EventSequencePlayer komponent ümbritseb MapCanvas komponenti ja oskab igas mängitavas ajahetkes leiduvate sensor-seadmete mõõtetulemused MapCanvas komponendile edasi anda.

MapCanvas komponent on rakenduse kõige tähtsam osa. Tema ülesandeks on visualiseerida ehitise plaani pilti, joonistada sellele sensor-seadmed ja visualiseerida mõõtetulemusi. Komponent teeb kõike seda joonistades HTML *canvas* elemendile erinevaid kujutisi.

MapCanvas komponent on omakorda jaotatud erinevateks loogika tükkideks, millest tähtsamad on:

- AnimationLoop.ts. See tükk hoolitseb selle eest, et *canvas* elemendi pilt joonistatakse konstantselt uuesti. Et seda teha efektiivselt kasutatakse veebilehitsejates toetatud *requestAnimationFrame* funktsiooni. Inimsilmale tunduvad sujuvana animatsioonid, mis on vähemalt kiirusega 60 kaadrit sekundis. Sellise animatsiooni sageduse saavutamine on keskmise jõudlusega sülearvutil tavapärane.

- CoordinateConverter.ts, mis hoolitseb selle eest, et alati konverteerida kõikide kujutavate olemite koordinaadid laius- ja pikkuskraadidelt (mida kasutab ja tagastab API) *canvas* elemendi poolt kasutatud lokaalsetele koordinaatidele (x ja y-telgede 0-punktid *canvas* elemendi ülemises vasakus nurgas).

- moveOnCanvas.ts ja zoomOnCanvas.ts, mis võimaldavad kasutajal komponendi kuvatavat pilti liigutada ja suurendada-vähendada.
- Erinevad kihid (backgroundLayer.ts, heatmapLayer.ts, colorbarLayer.ts), mis tegelevad üksnes *canvas* elemendile joonistamisega. Siin asub loogika, mis teab millise värvusega ja kuidas kujutada erinevaid mõõtetulemusi.

5 Valminud rakenduse kirjeldus

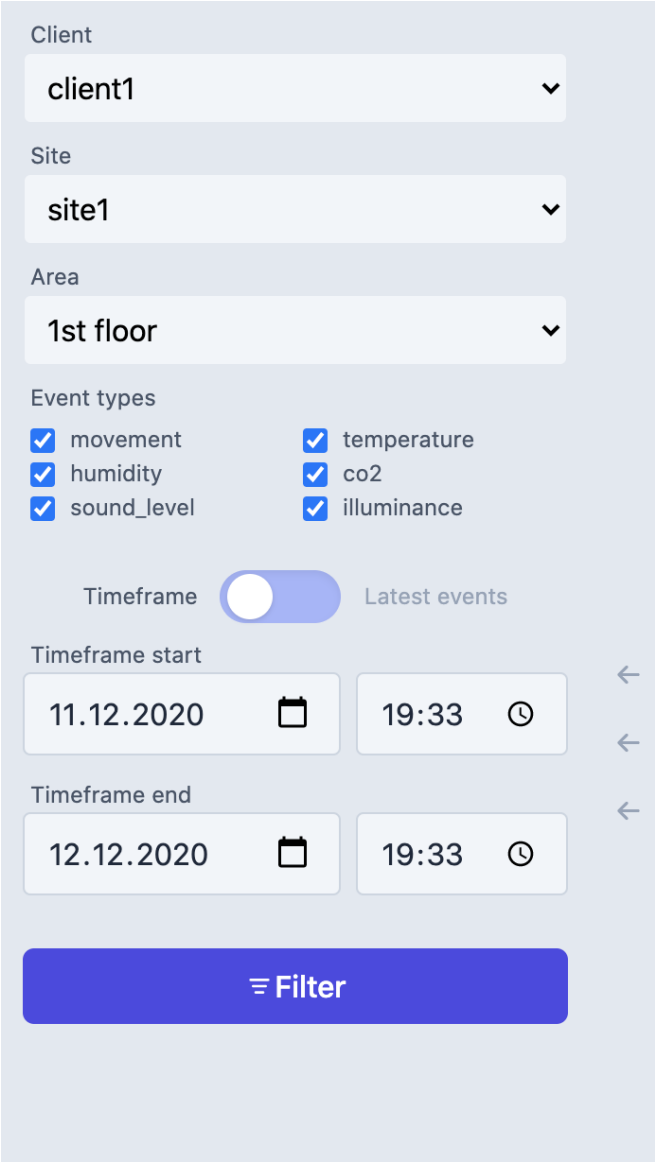
Valminud rakendus koosneb töö kirjutamise hetkel kolmest erinevast vaatest: lehekülg autentimiseks (*login* leht), lehekülg skaalade seadistamiseks (*settings* leht) ja peamine korruseplaani ja sensorite näitude kujutlusi kuvav lehekülg (*dashboard* leht). Järgnevalt ülevaade mida kasutaja nendes vaadetes teha saab.

5.1 Rakenduse kasutamine

Thinnecti API liides nõuab kasutaja autentimist. Autentimata kasutajal puudub ligipääs süsteemile ja ta suunatakse alati *login* leheküljele. Seal on võimalik kasutajal sisestada kasutajanimi ja parool, mille edukal sisestamisel tagastab API ligipääsu *token*'i.

Edukale autentimisele järgneb kasutaja suunamine süsteemi peamisesse vaatesse, mille keskosas asub korruseplaani kujutlus ja mille vasakust servast leiab kasutaja filtrid (Joonis 1).

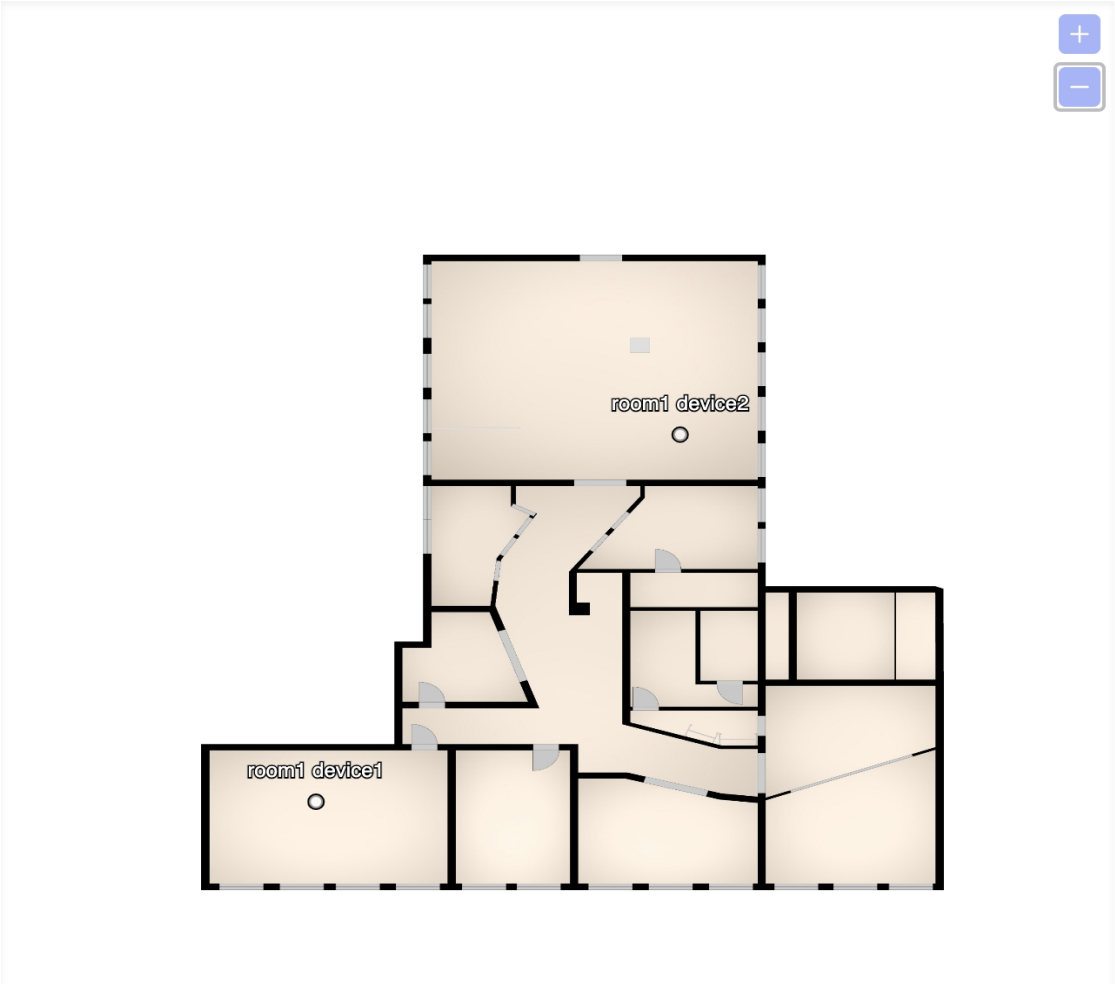
Filtrid võimaldavad kasutajal valida millise ehitise ja korruse plaani vaadeldakse. Samuti on filtritest võimalik valida, kas kasutaja soovib, et rakendus kuvaks pilti vastavalt sensorite kõige uuematele näitudele või on soov vaadelda mõnda kindlat varasemat ajaperioodi. Juhul kui kasutaja soovib vaadelda kõige uuemate näitude seisu, siis teostab rakendus korduvalt päringuid uute andmete küsimiseks.



Joonis 1. Ekraanipilt filtritest.

Rakenduse peamine funktsionaalsus on visualiseerida sensorite näitusid ehitise korruseplaanil (Joonis 2). Selleks ette nähtud alal on kuvatud korruseplaani pilt, mille peale kujutatakse sensorid.

Kasutajal on võimalik hiirega pilti lohistades kogu kujutlust liigutada. Sama saab teha ka klaviatuuril nooleklahve vajutades. Lisaks on võimalik kujutlust suurendada või vähendada kasutades klaviatuuril "+" ja "–" klahve.



Joonis 2. Ekraanipilt korruse plaanist.

Juhul kui kasutaja valis filtritest ajavahemiku, mille jooksul sensorite andmed muutuvad, siis kuvatakse kasutajale korruseplaani all sündmuste järjestuse riba. Selle riba abil on võimalik pildil läbi mängida sensorite muudatused vajutades *play* nupule. Juba käivitatud läbimängu puhul muutub *play* nupp *pause* nupuks, mille abil saab läbimängu vajadusel peatada.

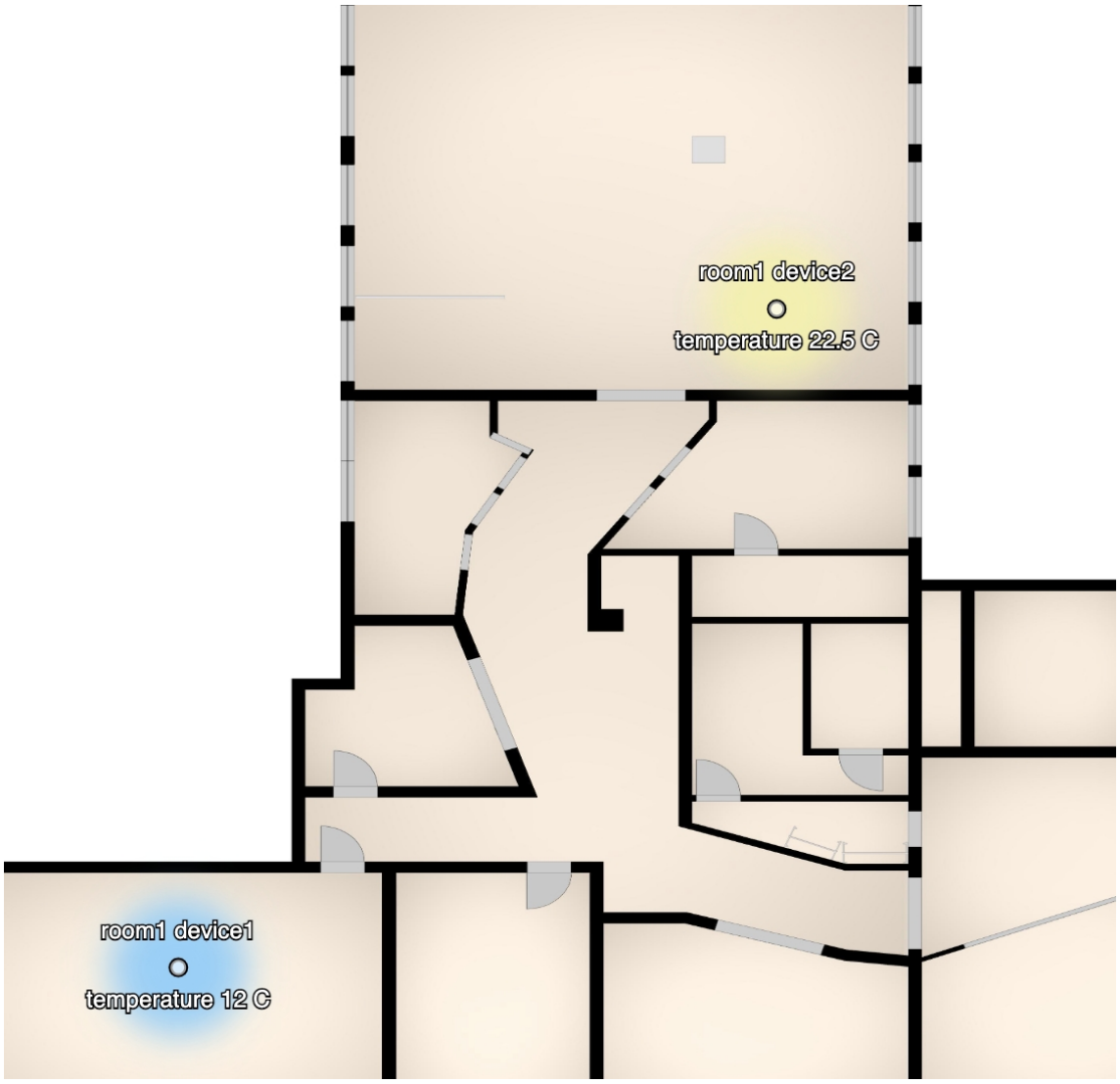
Kasutajal on võimalik klikkida ka ribal erinevaid ajahetki kujutatavatele aladele, et hüpata ajavahemikus edasi või tagasi (Joonis 3).

Joonis 3. *Play-pause* riba.

5.2 Sensorite näitude visualiseerimine

Valminud rakendus oskab hetkel sensorite tulemusi visualiseerida kahel viisil.

Juhul kui plaanil on kuvamiseks vaid ühe sensoritüübiga tulemused, kujutleb rakendus näitusid *heatmap*'ina. *Heatmap* kujutamine kasutab näitude eristamiseks erinevaid värvitoone ja on kasutajale väga lihtsasti mõistetav (Joonis 4).

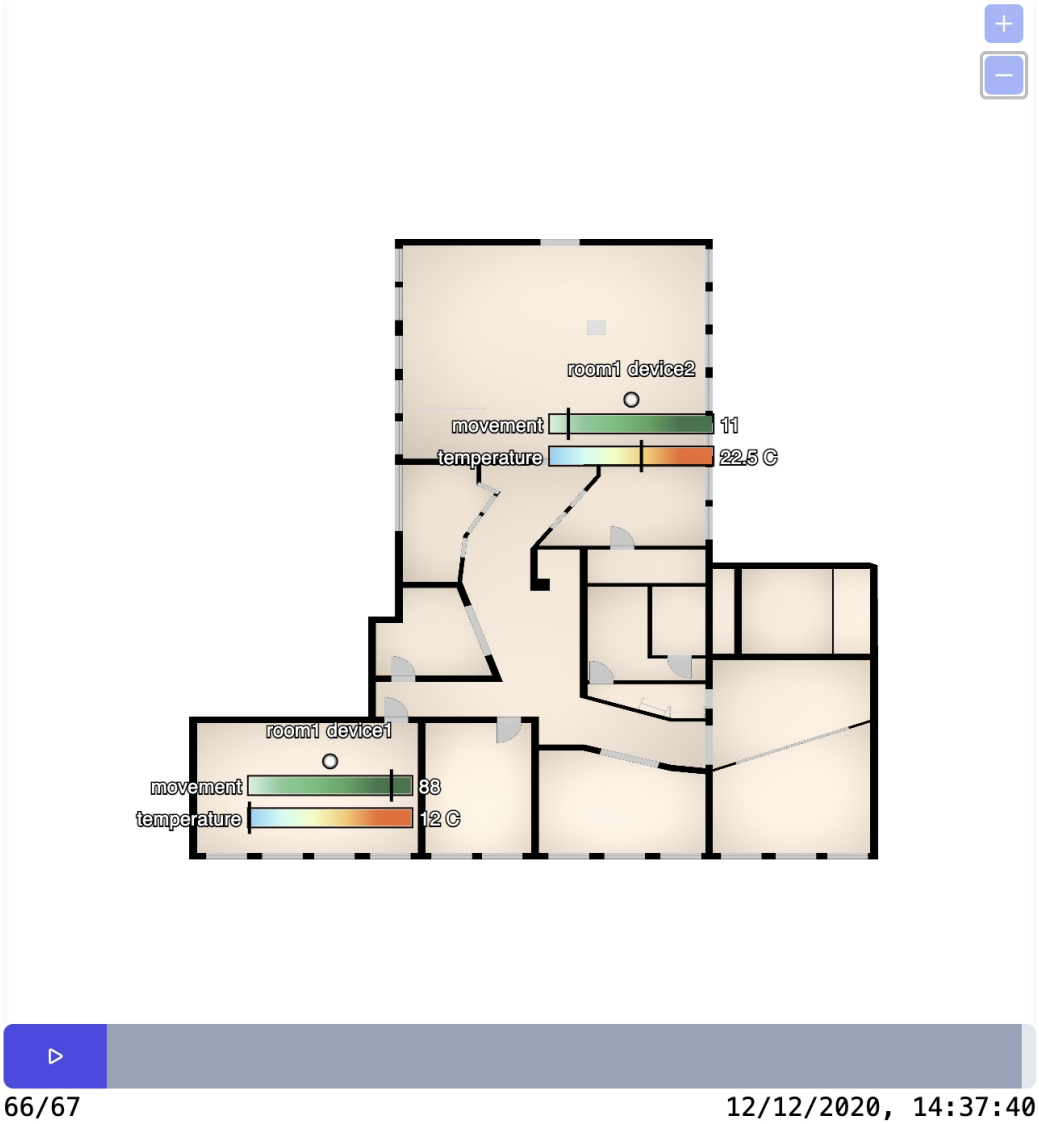


Joonis 4. *Heatmap* kujutlus süsteemis.

Kui erinevaid sensoritüüpe on korraga mitu, siis muutub *heatmap* lahendus aga raskemini mõistetavaks, sest erinevate sensorite visualiseerimiste kokkupuutepunktid tekitavad segadust. Sellele probleemile lahendust otsides proovis autor algselt mõne näidutüübi korral kuvada värviliste *heatmap* alade asemel kuvada erinevate mustritega alasid (triibud ja täpid), kuid kiirelt sai selgeks, et kui sensoritüüpe on rohkem kui 2, siis selline kujutamine ei ole enam kergesti mõistetav.

Selle probleemi lahenduseks sai otsustatud, et juhul kui sensoritüüpe on rohkem kui 1, siis kuvatakse *heatmap* kujutluste asemel tulemust näiduribana (Joonis 5). Selliseid

näiduribasid mahub plaanile rohkem ja nende puhul on kergemini mõistetav, mis sensoritüübiga sensori tulemust kujutatakse ja mis on kujutletav väärtus.

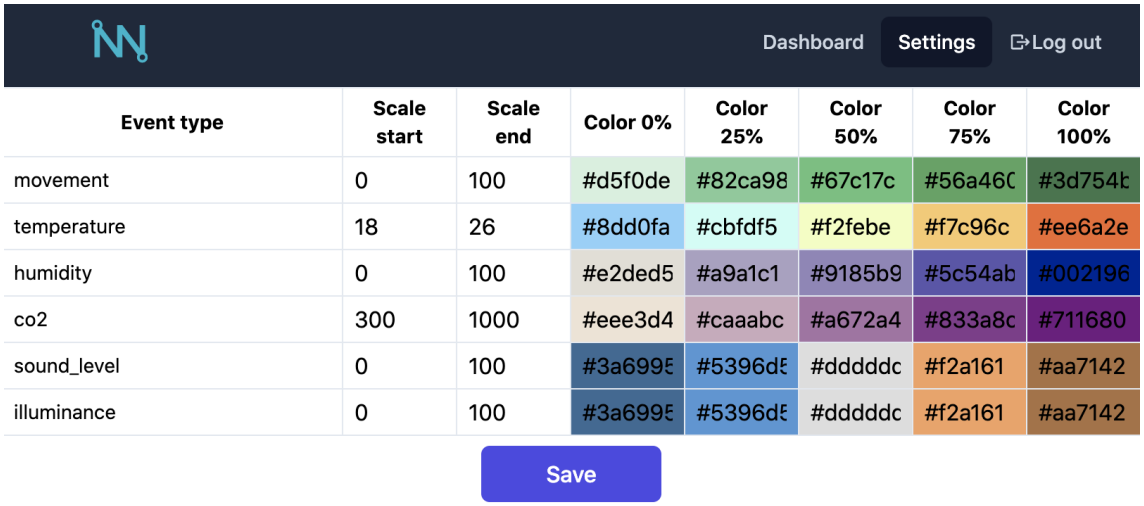


Joonis 5. Näiduriba kujutlus. Näiduribasid mahub plaanile rohkem.

Nii *heatmap* kui näiduriba kujutlused vajavad lisaks konkreetsele näidule visualiseerimiseks metaandmeid skaala algus- ja lõpp-punkti ning erinevate värvuste vastavuste kohta skaalal. Autori arvamusele tuginedes on need metaandmed seotud lõppkasutaja eelistustega, mistõttu sai loodud eraldi vaade erinevate sensoritüüpide jaoks skaalade seadistamiseks.

Skaala algus- ja lõpp-punktideks on võimalik määrata konkreetseid väärtusi. Alguspunktist väiksemad näidud kujutatakse sarnaselt alguspunkti väärtusega. Lõpp-punktist suuremad näidud aga samamoodi lõpp-punktis leiduva väärtusega.

Lisaks algus- ja lõpp-punktile on võimalik määrata värvused ka skaala 25%, 50% ja 75% tähiste jaoks (Joonis 6).



Joonis 6. Ekraanipilt skaalade seadistamise vaatest.

6 Võimalikud tulevased arendused

Valminud rakendus täidab küll enamuse algselt seatud nõuetest, kuid autor leidis päris mitu funktsionaalsust, mida peaks tulevikus täiendama või muutma.

Rakenduse arenduse alguses puudus ülevaade, kuidas peaks välja nägema lõplik kasutajaliides. Seetõttu tuleb rakendust vaadelda eelkõige kui prototüüpi. Enne kui rakendus lõppkasutajale tarnida tuleks valminud komponendid katta üksustestidega – hetkel testid puuduvad.

Autori nägemuse järgi on lõppkasutajal sellisest siseruumi sensoreid kuvavast rakendusest kõige rohkem kasu kui see on kasutusel mõnes seadmes, mida on võimalik hõlpsasti kaasas kanda või siis ühel suurel ekraanil, kus rakendus oleks kõigile alati nähtav. Esimesel juhul oleks vaja, et rakendus oleks optimeeritud töötama väikeste ekraanidega seadmetes, teisel juhul väga suurte ekraanidega seadmetes. Kahjuks on rakendus hetkel arendatud üksnes sülearvuti ekraani suurusega arvestades ja ei järgi reageeriva veebidisaini (ik r*esponsive design*) põhimõtteid, mille järgi peaks lehe sisu kuvamine kohanema ekraani suuruse järgi.

Valminud *heatmap* lahendus on väga lihtne ja ei vasta alati sellele, mida kasutaja tõenäoliselt eeldab. Selleks, et kasutaja jaoks *heatmap*'ide kuvamine loogilisemaks muuta, peaks rakendus tulevikus võimaldama korruse plaanil ruumide seadistamist ja seejärel kuvama *heatmap*'il sensori näitu nii, et see ei ületaks seinade piirjooni.

7 Kokkuvõte

Käesoleva töö eesmärgiks oli luua rakendus, mis võimaldab anda ülevaadet siseruumide sensorite poolt kogutud andmetest. Selle saavutamiseks proovis autor esmalt analüüsida alternatiivseid süsteeme, mis võimaldaksid sama teha. Kahjuks peab tõdema, et töö ülesandepüstituse tingimusi rahuldavad alternatiivid töö kirjutamise hetkel puudusid.

Töö tulemusena valmis rakenduse esmane versioon, mille puhul on tegemist on SPA tüüpi rakendusega. Rakendus saab läbi API andmed korruseplaanide, sensor seadmete ja nende sensorite mõõtetulemuste kohta. Nimetatud andmete põhjal oskab rakendus joonistada korruseplaani pildi ja sellele kuvada sensorid koos mõõtetulemuste visualiseerimisega. Mõõtetulemusi on võimalik ajaliselt filtreerida. Mõõtetulemuste jada on võimalik ajaliselt läbi mängida ajajoonel.

Autor leidis, et tegemist on rakenduse esmase versiooniga, mida tuleks enne klientideni tarnimist veel edasi arendada ja katta juba arendatud funktsionaalsus üksustestidega.

Rakendus valideeriti kliendi poolt testkeskkonnas.

Kasutatud kirjandus

[1] A.S. Ali, Z. Zanzinger, D. Debose, B. Stephens, 2016. "Open Source Building Science Sensors (OSBSS): a low-cost Arduino-based platform for long-term indoor environmental data collection", Build. Environ., 100.

[2] A. F. M. Batista, P. L. P. Correa and G. Palanisamy, 2016. "Visual Analytics Improving Data Understandability in IoT Projects: An Overview of the U. S. DOE ARM Program Data Science Tools," IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Brasilia, 2016, pp. 349-354, doi: 10.1109/ MASS.2016.052.

[3] N. Iftikhar, B. P. Lachowicz, A. Madarasz, F. E. Nordbjerg, T. Baattrup-Andersen, K. Jeppesen, 2020. "Real-time Visualization of Sensor Data in Smart Manufacturing using Lambda Architecture". Proceedings of the 9th International Conference on Data Science, Technology and Applications - Volume 1: Data, 2020, p. 215-222.

[4] Energy conservation through smart homes in a smart city: A lesson for Singapore households [Võrgumaterjal]. https://www.sciencedirect.com/science/article/pii/S0301421517300393 [Vaadatud 07.12.2020].

[5] Thinnect OÜ [Võrgumaterjal]. https://www.thinnect.com [Vaadatud 22.11.2020].

[6] Indoor-sensor-visualizer [Võrgumaterjal]. https://bitbucket.org/mihkel_reinart/indoor-sensor-visualizer [Vaadatud 25.11.2020].

[7] OpenRemote "Blok61 Apartments" use-case [Võrgumaterjal]. https://openremote.io/solution/blok61/ [Vaadatud 13.12.2020].

[8] Thingsboard [Võrgumaterjal]. https://thingsboard.io/ [Vaadatud 13.12.2020].

[9] 2020 Developer Survey [Võrgumaterjal]. https://insights.stackoverflow.com/survey/2020#most-popular-technologies [Vaadatud 25.11.2020].

[10] Vue.js [Võrgumaterjal]. https://vuejs.org [Vaadatud 24.11.2020].

[11] "React - A JavaScript library for building user interfaces" [Võrgumaterjal]. https://reactjs.org [Vaadatud 25.11.2020].

[12] Angular [Võrgumaterjal]. https://angular.io [Vaadatud 25.11.2020].

[13] Github [Võrgumaterjal]. https://github.com [Vaadatud 10.12.2020].

[14] Front-end JavaScript frameworks [Võrgumaterjal]. https://github.com/collections/front-end-javascript-frameworks [Vaadatud 10.12.2020].

[15] Composition API [Võrgumaterjal].
https://v3.vuejs.org/guide/composition-api-introduction.html [Vaadatud 24.11.2020].

[16] Typescript [Võrgumaterjal]. https://www.typescriptlang.org [Vaadatud 24.11.2020].

[17] CSS - MDN Web Docs Glossary [Võrgumaterjal].
https://developer.mozilla.org/en-US/docs/Glossary/CSS [Vaadatud 10.12.2020].

[18] Tailwind CSS [Võrgumaterjal]. https://tailwindcss.com [Vaadatud 25.11.2020].

[19] Canvas API [Võrgumaterjal].
https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API [Vaadatud 10.12.2020].

[20] What is Vuex? [Võrgumaterjal]. https://vuex.vuejs.org [Vaadatud 24.11.2020].

Lisa 1– Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Mihkel Reinart

1 Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Targa ehitise siseruumide andmete visualiseerimine" mille juhendaja on Jaanus Kaugerand

1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2 Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3 Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

04.01.2021

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.