TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mihkel Jõgeda 202875IADB

Rakendus veearvestite näitude automaatseks edastamiseks

Bakalaureusetöö

Juhendaja: Märt Kalmo
Magistrikraad

Tallinn 2023

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 Jõgeda

11.05.2023

Annotatsioon

Kortermaja rekonstrueerimise käigus on korteritesse paigaldatud kaugloetavad arvestid ning keskseade. Kord kuus käib kortermajas haldusfirma tehnik, kes vaatab süsteemist veemõõtjate näite, sisestab need käsitsi majas kasutusel olevasse arvelduskeskkonda ning elanikele esitatakse tarbimisarve tehniku sisestatud näidu põhjal. Kortermajale teenust pakkuv haldusfirma soovib arvelduprotsessi muuta kiiremaks, vähendada andmete käsitsi sisestamisel tehtavate vigade võimalust ning vabastada töötajad tööülesannetest, mis oleksid automatiseerimisel kuluefektiivsemad nii haldusteenuse pakkujale kui ka kortermaja elanikele.

Bakalaureusetöös analüüsitakse võimalusi mõõteandmete edastamiseks üle Interneti kortermajas kasutusel olevasse arvelduskeskkonda.

Analüüsi järel teostatakse vastavalt analüüsi tulemustele pilootprojekt ühe kortermajaga vastavalt hindamaks, kas lahendus on sobilik kasutuselevõtuks kõikides haldusfirma poolt hallatavates majades

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 22 leheküljel, 6 peatükki, 9 joonist, 2 tabelit.

Abstract

Application for Automatic Forwarding of Water Meter Readings

During the reconstruction of the apartment building, remotely readable meters have been installed in apartments along with a central device that has been installed in the building. Once a month, a technician from the property management company visits the apartment building, manually takes the readings of water meters connected to the system and enters manually enters them into the billing environment used by the apartment association. The residents are presented with a consumption bill based on the reading entered by the technician. The management company providing service to the apartment building wants to speed up the billing process, reduce the possibility of errors caused by manual data input , and release the employees from tasks that would be more cost-effective for both the management service provider and the residents of the apartment building if automated.

The bachelor's thesis analyzes the possibilities of transmitting measurement data over the internet to the billing environment used in the apartment association.

After the analysis, a pilot project will be carried out with one apartment building based on the results of the analysis in order to assess whether the developed solution is suitable for use in all buildings managed by the management company.

The thesis is written in Estonian and contains 22 pages of text, 6 chapters, 9 figures, 2 tables.

Lühendite ja mõistete sõnastik

API	Application Programming Interface on liides, mis võimaldab erinevatel rakendustel omavahel suhelda ja andmeid vahetada. API-d kasutatakse laialdaselt tarkvaraarenduses ja andmete jagamisel erinevate platvormide vahel.
DHCP	Dynamic Host Configuration Protocol on võrguprotokoll, mis võimaldab arvutitel ja muudel seadmetel automaatselt saada IP-aadressid, võrguparameetrid ja muud seadistused võrgus.
DNS	Domain Name System on protokoll, mis tõlgib inimkeelsed domeeninimede aadressid masinatele mõistetavateks IP-aadressideks
FTP	File Transfer Protocol on võrguprotokoll, mida kasutatakse failide edastamiseks arvutite vahel internetis.
HTTP	Hypertext Transfer Protocol on andmevahetuse protokoll, mida kasutatakse andmete edastamiseks veebiserverite ja klientrakenduste vahel.
IIoT	Industrial Internet of Things ehk tööstuslik asjade internet viitab seadmetele ja seadmesüsteemidele tööstuslikus keskkonnas, mis on ühendatud internetiga ning võimelised omavahel suhtlema ja andmeid vahetama.
IoT	Internet of Things ehk asjade internet viitab ühendatud seadmetele, mis on ühendatud võrku ning on võimelised omavahel suhtlema ja andmeid vahetama ilma inimese sekkumiseta.
M2M	"machine-to-machine" ehk „masinalt-masinale“ tähendab masinate omavahelist suhtlust, mis toimub ilma inimese sekkumiseta. See viitab kahe või enama seadme omavahelisele suhtlusele, kus üks seade võib saata andmeid või käsklusi teisele seadmele ilma inimese sekkumiseta.
MBUS	Meter-Bus on kommunikatsiooniprotokoll, mis on loodud spetsiaalselt kaugloetavate mõõteseadmete jaoks. See võimaldab andmete kauglugemist ja -kontrolli ning on mõeldud kasutamiseks kütte-, vee-, gaasi- ja elektriarvestite ning muude sarnaste seadmete jaoks.
MQTT	Message Queuing Telemetry Transport on lihtne ja kerge protokoll sõnumite vahetamiseks IoT seadmete vahel

PLC	Programmable Logic Controller ehk Programmeeritav loogikakontroller on elektrooniline seade, mida kasutatakse tööstuslikes automaatikasüsteemides ning mis on mõeldud erinevate automaatsete protsesside juhtimiseks
REST	Representational State Transfer on tarkvaraarhitektuuri stiil andmete vahetamiseks teenuste vahel
SMS	Short Message Service on mobiilsidevõrgu teenus, mis võimaldab saata ja vastu võtta lühisõnumeid.
TLS	Transport Layer Security on turvaprotokoll, mida kasutatakse andmete turvaliseks edastamiseks võrgus
VPN	Virtual Private Network on privaatne võrguühendus, mis võimaldab turvalist andmeedastust interneti kaudu
WIFI	Wireless Fidelity on traadita andmeside tehnoloogia, mis võimaldab arvutite ja muude seadmete juhtmevabalt arvutivõrku ühendamist

Sisukord

1 Sissejuhatus 10
1.1 Taust 10
1.2 Probleemi kirjeldus 10
1.3 Töö eesmärk 11
2 Probleemi analüüs 12
2.1 Paigaldatud seadmed 12
2.2 Andmete edastusvõimalused 13
2.3 Planeeritava süsteemi ülevaade 14
3 Tehnoloogiate analüüs 16
3.1 MQTT Brokeri valik 16
3.2 Andmebaasiserverite valik 20
3.2.1 Kronoloogilise andmebaasi valik 20
3.2.2 Relatsioonilise andmebaasi valik 21
3.3 Tagarakenduse tehnoloogiad 21
3.4 Kasutajaliidese tehnoloogiad 23
4 Teostus 23
4.1 Kontrolleri tarkvara täiendamine 23
4.2 Serveri seadistamine 25
4.3 Näitude edastamine kasutades API-liidest 26
4.4 Rakenduse kirjeldus 27
4.5 Testimine 28
5 Tulemused 29
5.1 Edasiste arenduste võimalused 30
6 Kokkuvõte 30
Kasutatud kirjandus 32
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks 36
Lisa 2 – Korto mõõtmisandmete api kirjeldus 37

Jooniste loetelu

Joonis 1 Tarkvara täiendamise seadmete testkomplekt....12
Joonis 2 Internetiruuter....13
Joonis 3 Andmete liikumise põhimõte....15
Joonis 4 MQTT tööpõhimõte....17
Joonis 5 Valmiva rakenduse andmete liikumise arhitektuur....22
Joonis 6 WAGO e!Cockpit tarkvara....24
Joonis 7 WAGO e!Cockpit tarkvara IDE ülevaade....25
Joonis 8 Andmebaasi diagramm tähtsamate olemitega....28
Joonis 9 Pilootprojekti käigus paigaldatud seadmed....29

Tabelite loetelu

Tabel 1 MQTT Brokeri vastavus seatud tingimustele 18
Tabel 2 MQTT Brokeri ressursikasutus 19

1 Sissejuhatus

1.1 Taust

Vee ettevõtted paigaldavad üldjuhul vaid ühe arvesti tervele majale ning tarbitud veekoguse arveldamine toimub majaomaniku või korteriühistuga. Kortermajades on kulude jagamiseks paigaldatud igasse korterisse veemõõtjad, mille näitude alusel toimub vee ettevõtte väljastatud arve jagamine majaelanike vahel. Veenäitude kogumise võimalusi on erinevaid – lihtsaim ja pikalt kõige laiemalt kasutusel olnud võimalus oli kõikide ühistusse kuuluvate korteri elanike poolt näitude käsitsi kirjapanek ning nende edastamine paberil kindlaksmääratud ajaks.

1.2 Probleemi kirjeldus

Kortermaja rekonstrueerimise käigus on korteritesse paigaldatud kaugloetavad arvestid ning keskseade. Keskseadmele on lisaks veemõõtjatele võimalik ühendada veel erinevaid andureid ja täitureid maja automaatika juhtimiseks. Kord kuus käib kortermajas haldusfirma tehnik, kes süsteemist veemõõtjate näite vaatab, sisestab need käsitsi majas kasutusel olevasse arvelduskeskkonda ning elanikele esitatakse tarbimisarve tehniku sisestatud näidu põhjal. Kortermajale teenust pakkuv haldusfirma soovib arveldusprotsessi muuta kiiremaks, vähendada andmete käsitsi sisestamisel tehtavate vigade võimalust ning ning vabastada töötajad tööülesannetest, mis oleksid automatiseerimisel kuluefektiivsemad nii haldusteenuse pakkujale kui ka kortermaja elanikele. Soovitud eesmärkide saavutamiseks soovib haldusfirma hakata veenäite automaatselt arveldussüsteemi edastama. Lisaks konkreetsele kortermajale soovib haldusfirma oma tööprotsesside ühtlustamiseks automatiseerida näitude kogumise protsessika teistes ettevõtte poolt hallatavates kortermajades. Seatud eesmärgi saavutamiseks plaanitakse paigaldada kasutuselevõetav automatiseeritud lahendus ka teistesse ettevõtte halduses olevatesse kortermajadesse.

1.3 Töö eesmärk

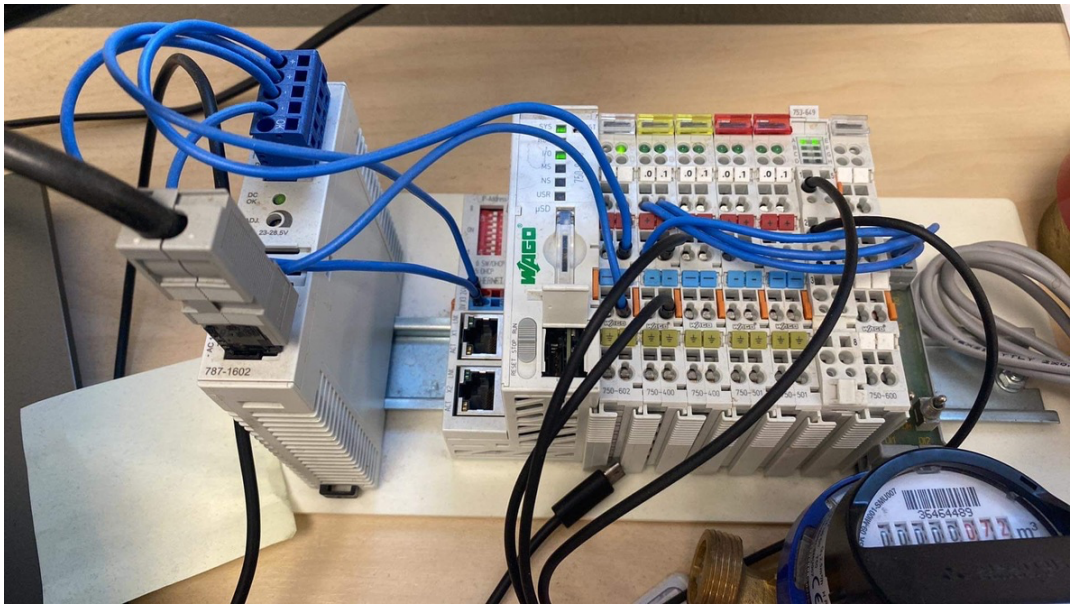
Bakalaureusetöö eesmärk on analüüsida võimalusi mõõteandmete edastamiseks üle Interneti kasutusel olevasse arvelduskeskkonda.
Analüüsi järel teostatakse pilootprojekt ühe kortermajaga vastavalt analüüsitulemustele ning hinnatakse, kas lahendus on sobilik kasutuselevõtuks kõikides haldusfirma poolt hallatavates majades.

2 Probleemi analüüs

Järgnevas osas analüüsitakse bakalaureusetöös käsitletavat probleemi koos paigaldatud seadmetega, mis arendatavale süsteemile piirangud seavad. Lisaks antakse ülevaade planeeritavast rakendusest probleemi lahendamiseks.

2.1 Paigaldatud seadmed

Kortermajja on maja renoveerimise käigus paigaldatud automaatika keskseade Wago 750-8100, edasises kontekstis „kontroller“ ja veemõõtjad Apator. Veemõõtjad on juhtmetega ühendatud keskseadme külge, keskseade ei ole töö kirjutamise alguse ajaks Internetiga ühendatud. Kuna sellisele kontrollerile on võimalik lisaks veemõõtjatele külge ühendada ka muid hooneautomaatikaga seotud andureid nagu näiteks soojusarvestid, gaasiarvestid, õhukvaliteedi andurid ja täitureid nagu näiteks mootorid, juhitavad klapid, ventiilid, läbipääsud on haldusfirma avaldanud soovi sarnane komplekt paigaldada ka teistesse enda poolt hallatavatesse ja rekonstrueeritavatesse majadesse.



Joonis 1 Tarkvara täiendamise seadmete testkomplekt

2.2 Andmete edastusvõimalused

Vastavalt kasutusjuhendile[4] toetab paigaldatud kontroller järgmisi võrguprotokolle: DHCP, DNS, HTTP(S), FTP(S), MQTT. DHCP ja DNS on mõeldud kontrolleri võrgusuhtluse tagamiseks, HTTP ja FTP on mõeldud kontrolleri seadistamiseks ja kontrollerist andmete kätte saamiseks võrgu kaudu. Neid protokolle oleks võimalik kasutada veemõõtjate lugemite kätte saamiseks, kuid nende kasutamine sel eesmärgil ei ole mõistlik, kuna sellisel juhul käituks kontroller serverina ning näitude lugemiseks oleks tarvilik paigaldada lisaks seadmeid mis kohapeal andmeid loeksid ning võrku edastaksid või kasutada kontrollerisse jäigalt kodeeritud salasõnasid mis süsteemi eluea jooksul tõenäoliselt ei muutuks ning mis seetõttu võiks tekitada potentsiaalse andmete lekkimise ohu. MQTT'd on võimalik kasutada andmete edastamiseks võrku otse kontrolleri poolt, kontrollerisse pöördumata. Kasutades MQTT protokolli on kliendi (kontrolleri) ja serveri vaheline autentimine võimalik kliendisertifikaati, mis aitab vältida paroolide salvestamist

Joonis 2 Internetiruuter

ja võimalikku korduvkasutust erinevate majade vahel. Selleks, et andmeid oleks võimalik automaatselt saata kesksesse serverisse tuleb kontroller ühendada Internetiga. Kuna kontrolleril on vaid Ethernet võrguliides ning korteriühistu ei taga kontrollerile kaabliga Internetiühendust ühendatakse kontroller võrku kasutades mobiilset Internetiühendust. Internetiühenust jagava seadme ehk ruuterina oleks antud kontekstis võimalik kasutada erinevaid internetiteenusepakkuja poolt müügiks pakutavaid seadmeid, kuid kuna need on suunatud pigem kodukasutuseks, pakuvad vähe iseseisvat ümberseadistusvõimalust ning omades antud olukorras ebavajalikku Wi-Fi funktsionaalsust otsustab töö autor koostöös haldusfirmaga kasutada Teltonika ruuterit TRB140. Valik langes konkreetsele

tootele, kuna toodet on võimalik sarnaselt kontrolleriga paigaldada automaatikakilbi DIN-liistule ja ruuteri toitepinge (9-30V) vahemik võimaldab seadme toitena kasutadakontrolleri toiteallikat. Lisaks võimaldab TRB140 luua VPN ühendusi ning teda on võimalik kaugelt SMS käsklustega juhtida. Kõik see tagab süsteemi kompaktsuse ning annab haldusfirma tehnikule võimaluse vajadusel teostada esmaseid veaolukordade tuvastamisi ja lahendamisi objektile kohale sõitmata.

2.3 Planeeritava süsteemi ülevaade

Korteritesse on paigaldatud veemõõtjad mis omakorda on juhtmetega ühendatud kortermaja tehnoruumis asuva keskse kontrolleri külge. Kontroller saab veemõõtjatest info kasutades MBUS protokolli. Kontrolleri poolt veemõõtjast väljaloetav informatsioon sisaldab veemõõtja seerianumbrit, hetkenäitu, hetke vooluhulka ning juhtudel kui mõõtja on tuvastanud veaolukorra siis ka konkreetse veakoodi. Veaolukorraks võib olla näiteks vee voolusuuna muutumine või mõõtja varupatarei tühjenemine.

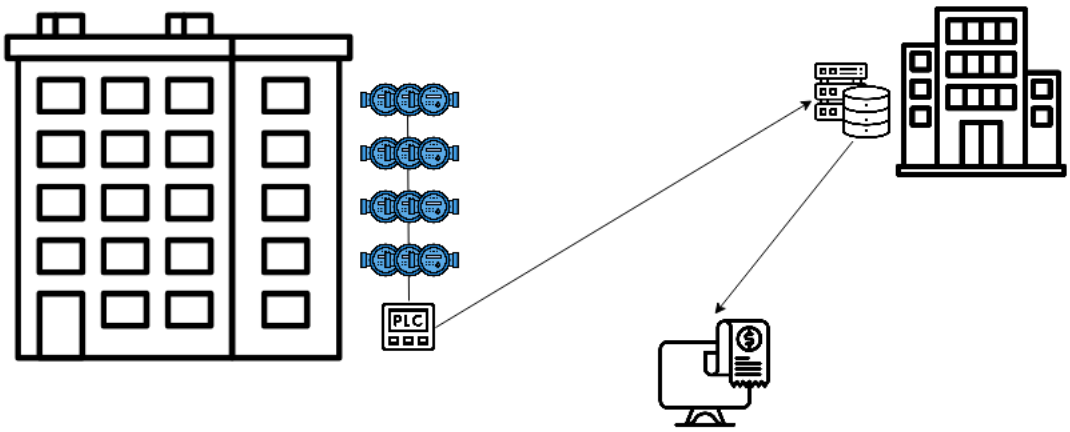
Bakalaureusetöö teostamiseks on vajalik täiendada juba paigaldatud kontrolleri tarkvara viisil, mis võimaldab kontrolleril edastada info veenäitude kohta kindlaksmääratud intervalli järel keskserverisse.
Kuna kontroller toetab info edastamist kasutades MQTT protokolli on mõistlik valida just see protokoll info edastamiseks, sest MQTT protokoll on üks vanimaid ja levinumaid masinalt-masinale (M2M) sõnumsideprotokolle IoT (asjade Interneti) ja IIoT (Industrial IoT/tööstuslik asjade internet) maailmas[1]. Seega võib järeldada, et sellisesse süsteemi saab tulevikus liita ka teist tüüpi või teise tootja kontrollereid.

Kuna MQTT server, mis info vastu võtab ei salvesta vastuvõetavat infot tuleb mõõteandmete säilitamiseks keskses serveris salvestada mõõtjatest pärit andmed koos ajatempliga andmebaasi.

Lisaks tuleb haldusfirmale luua kasutajaliides, milles on võimalik kirjeldada maju ning seal kasutusel olevat arveldussüsteemi (Korto, Digimaja) ja siduda veemõõtja seerianumber konkreetse maja, korteri ja mõõtja tüübiga (soe, külm, peaarvesti). Seejärel on võimalik andmeid perioodiliselt arveldussüsteemi edastada. Kuna kortermaja, mille korterite veemõõtjate baasil käesolev töö teostatakse, kasutab arveldussüsteemina

veebipõhist rakendust Korto ning Korto võimaldab andmeid vastu võtta REST api kaudu[2] saame süsteemi testimise käigus näidud igakuiselt edastada kasutades seda kanalit.

Kogu süsteem peab olema laiendatav, et tulevikus oleks võimalik liita uusi majasid ning rakenduse programmeerimisel tuleb arvesse võtta, et tulevikus võib tekkida soov süsteemi lisada ka teist tüüpi (näiteks: soojus, gaas) arvesteid.



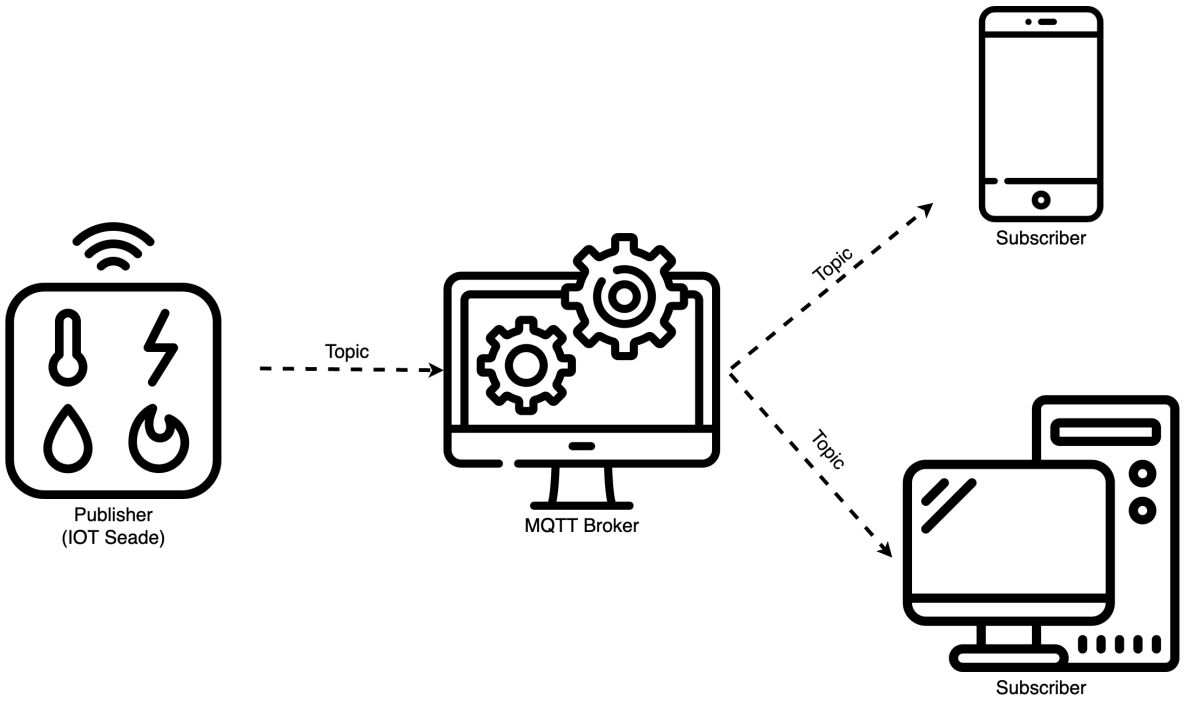
Joonis 3 Andmete liikumise põhimõte

3 Tehnoloogiate analüüs

Selles peatükis selgitab töö autor tarkvara valmistamisel kasutatavaid tehnoloogiaid ning valib välja projekti teostamiseks sobilikud tehnoloogiad.

3.1 MQTT Brokeri valik

MQTT on masinalt-masinale (M2M) kommunikatsiooniprokoll, mis on välja töötatud 1999 aastal Andy Stanford-Clark ja Arlen Nipper'i poolt[3]. MQTT loomise eesmärgiks oli luua võimalikult lihtne ja kerge protokoll nafta- ja gaasitööstuse ettevõtetes kasutusel olnud seadmetest info saamiseks üle aeglase ning kehva kvaliteediga Internetiühenduse. Tänapäevaks on MQTT kasutusele võetud paljudes IoT seadmetes kuid seda kasutatakse näiteks ka Meta (Facebook) poolt loodud suhtlusrakenduses Messenger[8] ning Netflixi voogedastusteenuses[13].
MQTT põhineb sõnumite vahetusel klientide vahel kasutades keskset serverit mida nimetatakse Broker'iks. Sõnumeid vahetatakse kasutades Publish/Subscribe (avalda/kuula) mudelit. Sõnumid jagatakse teemadesse (topic) mille kaudu saab klient infot avaldada (publish) või kuulata (subscribe).



Joonis 4 MQTT tööpõhimõte

Oluline on märkida, et üldjuhul MQTT broker sõnumeid ei salvesta ning kui teemal puuduvad kuulajad , ei ole hilisem sinna saadud sõnumi kättesaamine võimalik. Lisaks pole hiljem teemat hiljem teemat kuulama asunud kliendil võimalik kätte saada varasemaid teemasse saadetud sõnumeid. Juhtudel, mil varasema sõnumi säilitamine on siiski oluline saab sõnumi avaldaja sellekohase lipu märkida sõnumi päisesse[10]. Lisaks on avaldajal võimalik saata brokerile ka Will tüüpi[12] sõnum, mille broker salvestab ning üldjuhul edastab selle teemat kuulavatele klientidele avaldaja ühenduse katkemisel.

Protokolli on sisse ehitatud ka autentimisvõimalus mille abil on võimalik piirata kliendi ligipääsu teemapõhiselt. Lisaks on võimalik kasutusele võtta ka TLS krüpteerimine, mis võimaldab andmeid edastada üle turvalise kanali.

Järgnevas osas võrdleb autor Mosquitto, EMQ X ja HiveMQ serveritarkvarasid. Võrdluse aluseks on järgnevad tingimused:

- Server peab võimaldama üheaegset suhtlust vähemalt 25 kliendiga
- Server peab võimaldama kliente autentida kasutajatunnuse ja parooliga
- Server peab võimaldama piirata ligipääsu teemadele vastavalt kasutajatunnusele (autoriseerimine)
- Server peab toetama TLS ühendusi

- Serverit peab olema võimalik paigaldada Docker konteinerisse.

- Tabel 1 MQTT Brokeri vastavus seatud tingimustele

	Mosquitto	EMQ X	HiveMQ
Max üheaegne ühenduste arv	Täidab nõuet	Täidab nõuet	Täidab nõuet
Autentimine	Jah	Jah	Jah
Autoriseerimine	Jah	Jah	Jah
TLS	Jah	Jah	Jah
Docker (kas on toetatud, allalaadimiste arv hub.docker.com alusel)	Jah, (>500 miljonit allalaadimimist)	Jah (>10 miljonit allalaadimist)	Jah (>5 miljonit allalaadimist)

Kuna üheaegsete lubatud ühenduste arv kõikidel analüüsitud serveritarkvaradel on piisav, ei saa see valitava tarkvara puhul määravaks. Parema valiku tegemiseks analüüsis autor veel riistvarale esitatavaid miinimumnõudeid. EMQ X ja HiveMQ riistvarale esitatavad miinimumõuded avaldanud oma kodulehtedel[15][14] on võrdluses välja toodud just need andmed. Mosquitto puhul selliseid miinimumnõudeid riistvarale kättesaadaval ei olnud. Riistvara nõuete määramiseks teostas töö autor testimise olemasoleva riistvara peal.

Tabel 2 MQTT Brokeri ressursikasutus

	Mosquitto	EMQ X	HiveMQ
Protsessor	Pole avaldatud. Testimisel 10 kliendiga ca 1% 1xCPU	2CPU	4CPU
Operatiivmälu	Ei ole avaldatud. testimisel 10 kliendiga ca 15MB	4GB	4GB
Kõvakettapind	Info puudub/ pole ilmnenud	Info puudub/ pole ilmnenud	100GB
Muu	Info puudub/ pole ilmnenud	Info puudub/ pole ilmnenud	OpenJDK 11 või uuem

Võrdluse tulemustest selgub, et kõik võrdluses olnud tarkvarad on sobilikud ülesande lahendamiseks. Autor valis ülesande lahendamiseks Mosquitto MQTT broker'i, sest sellel on suurim kasutajate arv ning testimisel ilmnes väiksem riistvara ressursside nõudlus. Lisapõhjuseks Mosquitto kasuks osutus litsentsitüüp ning majanduslik taust kuna erinevalt teistest kahest valikus olnud tarkvarast ei ole see seotud ühegi äriettevõttega. See aspekt on oluline, kuna äriettevõtted võivad tulevikus oma tasuta jagatava tarkvara litsentsitingimusi muuta ning tasuta toote tasuliseks muuta.

3.2 Andmebaasiserverite valik

Loodav rakendus tegeleb peamiselt kahte tüüpi andmetega

- Mõõtmisandmed – on ajalises järjestuses andmed kindla veemõõtja näidu kohta
- Rakenduse halduseks ja mõõteandmete edastamiseks vajalikud andmed – nende andmete hulka kuuluvad näiteks mõõtjate sidumine maja, korteri ja mõõtjatüübiga ja rakenduse ligipääsuõiguste haldus.

Kuna andmed on eritüübilised tuleb ka nende säilitamisele läheneda erinevalt. Mõõtmisandmed peavad olema ajaliselt järjestatud, seotud kindla punktiga ajas ning nad on pidevas muutuses. Samas väheneb ajas vanemate mõõteandmete aktuaalsus ning serveripinna ökonoomsekaks kasutamiseks on mõistlik neid andmeid teatud aja möödudes koondada vähendades andmepunktide tihedust. Näiteks puudub vajadus omada 5 minutilise täpsusega andmeid rohkem kui pool aastat peale mõõtmist ning rohkem kui 6 tunnise täpsusega andmeid kahe aasta pärast peale mõõtmist. Samas on oluline osa andmeid säilitada, et säiliks mõõtmisandmete ajalugu ning oleks võimalik arvutada tarbimistrende. Kronoloogilistesse andmebaasisüsteemidesse (Time Series Database, TSDB) on selline funktsionaalsus juba sisse ehitatud, seega oleks mõistlik neid andmeid hoida just sellises andmebaasis.

Näitude edastamiseks ja rakenduse toimimiseks vajalikud andmed muutuvad harva ning neid ei ole eriti palju, seega on neid mõistlik hoida mõõteandmetest eraldi relatsioonilises andmebaasis.

3.2.1 Kronoloogilise andmebaasi valik

Veemõõtjate näitusid edastatakse kontrollerisse ning kontrollerist kesksesse serverisse kindla ajavahemiku järel. Aegrida on andmepunktide jada, mis esinevad kindlas järjekorras kindla ajavahemiku jooksul[16]. Seega saab öelda, et mõõtmisandmete puhul on tegu aegreaga. Aegridade salvestamiseks ja töötlemiseks on loodud kronoogilised andmebaasid. Kuigi aegridadel põhinevaid andmeid on võimalik salvestada ja töödelda ka relatsioonilises andmebaasis pakuvad kronoloogilised andmebaasisüsteemid paremaid

statistilisi funktsioone ning optimaalsemat ressursikasutust justnimelt aegridade salvestamiseks.

Andmebaaside kasutusstatistikat koguva veebilehe DB-Engines andmetest[19] lähtub, et InfluxDB[20] on hetkel enamlevinud kronoloogiline andmebaas. Kuigi samade andmete põhjal on näha, et TDEngine[21] on kiiresti populaarsust kasvatamas valitakse proleemi lahendamiseks siiski hetkel populaarseim vababavaraline andmebaasisüsteem InfluxDB

3.2.2 Relatsioonilise andmebaasi valik

Relatsiooniline andmebaas on andmebaasi tüüp, kus andmed on organiseeritud tabelite kujul. Iga tabel koosneb ridadest ja veergudest ning igal veerul on kindel andmetüüp ja nimi[17]. Andmebaasi tabelites olevaid ridu on omavahel võimalik siduda võtmete kaudu[18].

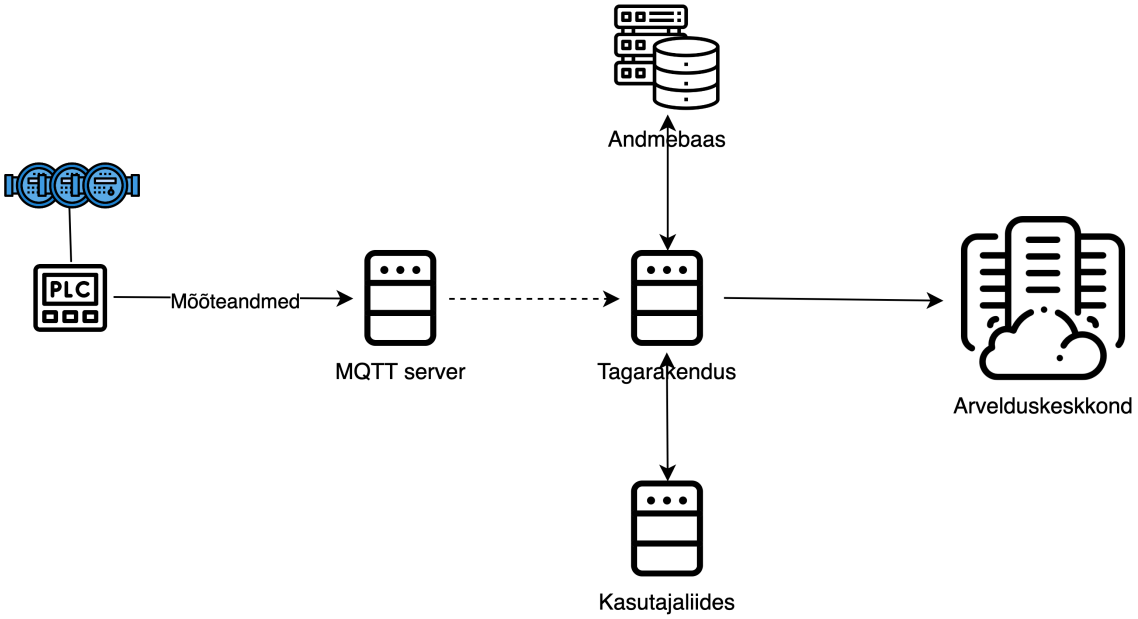
Haldusfirma on väljendanud, et ei sooviks võtta lisakohustust tarkvara litsentsitasu näol, seega on tarvis valida vabavaraline andmebaasisüsteem. Andmebaaside kasutusstatistikat koguva veebilehe DB-Engines andmetest [19] lähtub, et enimkasutatavad vabavaralised andmebaasiserveri tarkvarad on MySQL ja PostgreSQL.

Haldusfirma esindaja on väljendanud soovi arendatavasse süsteemi tulevikus ettevõtte jaoks rohkemate vajalike funktsionaalsuste lisamiseks. Kuigi MySQL kasutajate arv hetkel on suurem, on PostgreSQL'i viimase kolme aasta kasutustrend pidevalt kasvav mis viitab just selle andmebaasisüsteemi kasutuselevõtule uutes, alustatavates projektides. Sellest lähtuvalt valis töö autor töös kasutatava lahendusena relatsiooniliseks andmebaasisüsteemiks PostgreSQL'i.

3.3 Tagarakenduse tehnoloogiad

Planeeritava lahenduse tagarakendus on andmete käitlemise keskpunkt. Tagarakendus jälgib mõõteandmete saabumist MQTT serverisse ja andmete laekumisel salvestab need andmebaasi. Samuti tegeleb tagarakendus kasutajaliidesest API-liidese kaudu tulevatele

andmepäringutele vastamisega ja näitude kindlal ajal arvelduskeskkonda edastamisega.



Joonis 5 Valmiva rakenduse andmete liikumise arhitektuur

Tagarakenduse raamistikena on kaalumise all Java programmeerimiskeelel põhinev Spring Boot[24] ning Microsofti poolt arendatud avatud lähtekoodiga .NET raamistikku[23], mis võimaldab C#, F# ja Visual Basic programmeerimiskeelte kasutust[23]. Haldusfirmal, mis valmivat rakendust kasutama hakkab, on juba olemasolev füüsiline server, millele on paigaldatud ettevõtte jaoks tarvilike teiste tarkvarade käitamiseks Docker konteineriseerimisteenus. Vältimaks ettevõtte jaoks lisanduvat halduskulu peab ka kogu planeeritav lahendus olema paigaldatav Dockeri konteineritena. Dockeri konteinerite kasutamine annab muuhulgas võimaluse tulevikus kogu rakendust või iga selle osa eraldi vastavalt vajadusele ümber tõsta, uuendada või välja vahetada. Et valmiv rakendus tarbiks serveris võimalikult vähe ressurssi, otsustab töö autor valida .NET Core raamistiku, mis on ülalmainitud kahest valikust kiirem ning väiksema protsessori- ja mälukasutusega[26]. Programmeerimiskeelena selles raamistikus kasutab autor C# programmeerimiskeelt. Valmiva koodi pakendab autor haldusfirma jaoks Docker'i pildina, mida haldusfirma serverit haldav administraator hõlpsasti paigaldada saab.

3.4 Kasutajaliidese tehnoloogiad

Haldusfirma soovib hooldustööde paremaks planeerimiseks saada ülevaadet mõõtjate näitudest ning tarbimisajaloost. Andmete analüüsimiseks tuleb luua ettevõttele haldusliides, milles kasutajale kuvatakse hetke tarbimisandmedd ja üleüldine tarbismisajalugu. Lisaks andmete nägemisele peab rakendus võimaldama arvesti identifikaatori sidumist majas kasutusel oleva arveldussüsteemiga ja arveldussüsteemis kirjeldatud arvesti identifikaatoriga. See võimaldabandmete automaatsel edastamisel tagarakendusel teada, missugust ja milliste parameetritega API-liidest kasutada tuleb.

Kasutajaliidese valmistamiseks on kaalumisel React ja Vue.JS Javascript raamistikud.

Kuna Vue.JS on pisut kiirem, tarbib vähem ressurssi ning võimaldab eraldada Javascript (ka TypeScript), HTML ja CSS eraldi komponentidesse[28][27] valib autor käesoleva töö kasutajaliidese valmistamise raamistikuks just Vue.JS. Kogu kasutajaliides on tagarakendusest eraldatud ning kõik toimimiseks vajaliku info päritakse API-liidese kaudu tagarakendusest. Valmiva kasutajaliidese pakendab töö autor samuti Docker pildina.

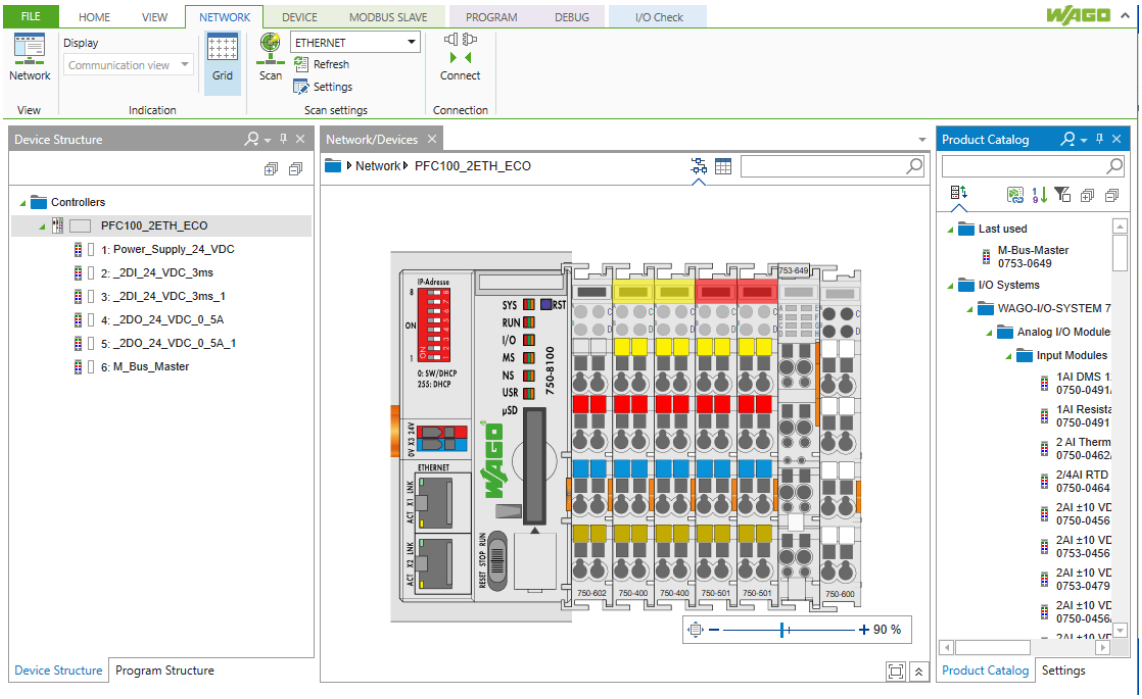
4 Teostus

Käesolevas peatükis kirjeldab autor kontrolleri tarkvara täiendamist, olulisi aspekte serveri seadistamiseks ja arendatavat rakendust. Samuti kirjeldatakse kogu arendatava süsteemi testimise põhimõtteid.

4.1 Kontrolleri tarkvara täiendamine

Kontrolleri programmeerimiseks kasutatakse IEC 61131 standardis kirjeldatud programmeerimiskeeli (redeldiagramm, funktsionaalblokid, struktuurtekst, juhiste loend, järjestikune funktsioonide diagramm). Kuna kontrolleri paigaldaja on keskseadme programmeerimisega algust teinud Structured Text'is ehk struktuurtekstis (ST) programmeerimiskeeles, on vajalik töös tehtavate täienduste tegemine ST keeles.. Struktuurtekst on tekstipõhine programmeerimiskeel, mille süntaks sarnaneb Pascal

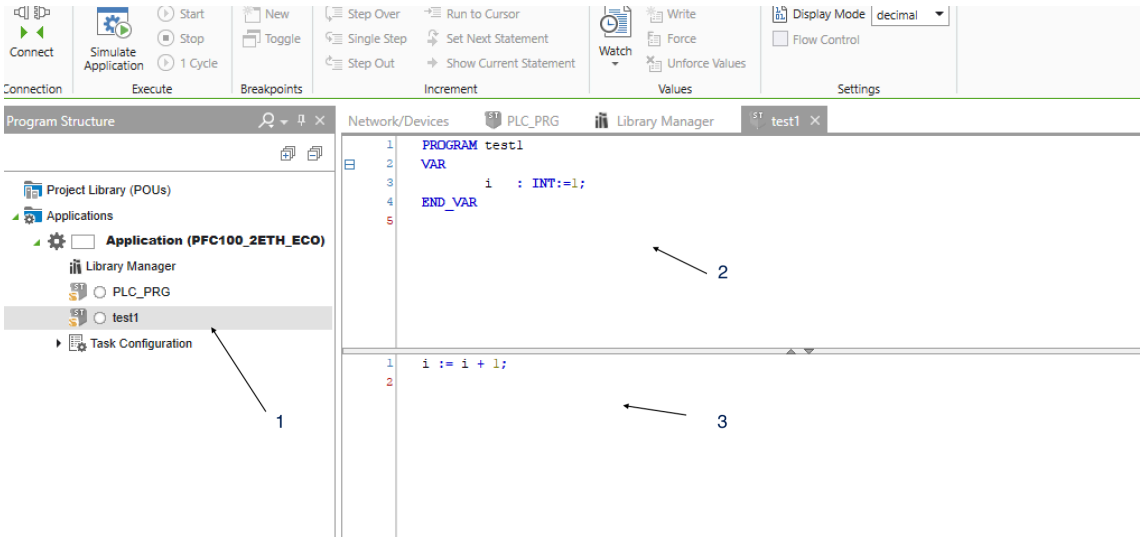
programmeerimiskeelele. WAGO kontrollerite programmeerimiseks kasutatakse e!Cockpit nimelist tarkvara. Tarkvara on tasuline, kuid töö kirjutamiseks sobib ka prooviversioon, mis toimib 30 päeva ning mida on võimalik alla laadida WAGO kodulehelt[25]. Prooviversioon on täisfunktsionaalne, ainsaks piiranguks on kasutatav aeg. Programmeerimise alustamiseks tuleb tarkvara käivitada ning luua uus projekt või avada eelneva projekti lähtekoodi fail. Projekti on vajalik kirjeldada kõik programmeeritavad kontrollerid ning iga kontrolleri külge ühendatud laienduskaardid. See annab Integrated Development Environment'ile (IDE-le) teada missuguseid konkreetseid funktsioone ja meetodeid programmeerimisel kasutada võimalik



Joonis 6 WAGO e!Cockpit tarkvara

Seejärel loob tarkvara tühja programmi skeleti, millesse on võimalik oma loogikat kirjutama hakata. Programmi põhivaade koosneb kolmest osast

1. Kirjeldatud alamprogrammid ja funktsioonid
2. Muutujate kirjeldamise ala
3. Programmi koodi ala



Joonis 7 WAGO e!Cockpit tarkvara IDE ülevaade

Kontrolleri paigaldaja edastas töö autorile enda poolt kirjutatud kontrolleri programmi mida kasutati kohapeal mõõtjate lugemiseks. Haldusfirma võimaldas töö kirjutamise ajaks töö autoril olemasoleva varuseadme kasutust. Töö autor on programmi täiendanud viisil, et kontroller edastaks iga 5 minuti järel kõik näidud MQTT serverisse.

Edastatava MQTT Sõnumi näide:

```
{
    "Name": "1. Köök Soe",
    "Readings": [
        {
            "ID": "00283253",
            "ValueVolume": "0.105",
            "ValueFlow": "0.0",
            "Error": "0"
        }
    ]
}
```

4.2 Serveri seadistamine

Haldusfirmal on juba olemas füüsiline server, mida kasutatakse teiste ettevõtte jaoks tarvilike tarkvarade käitamiseks. Serverile on paigaldatud Docker tarkvara ja haldusfirma esindaja on avaldanud soovi, et arendatav rakendus koosneks Docker'i konteineritest. Docker konteinerite kasutamine võimaldab nii arendatavat tarkvara ja kui ka iga selle komponenti vastavalt vajadusele uuendada, tõsta ümber uuele riistvarale või

pilvekeskkonda paigaldada. Docker piltidest loodud konteinereid kasutav rakendus on ühtlasi turvalisem, kuna internetti paistab vaid MQTT server ja veebipõhine kasutajaliides on kasutatav vaid haldusettevõtte sisevõrgust. Andmete turvaliseks edastamiseks kontrollerist MQTT serverisse seadistatakse MQTT server kasutama TLS krüpteeringut. TLS krüpteerimiseks soovib töö autor kasutada tasuta kättesaadaval olevaid Let's Encrypt sertifikaate. Kuna Let's Encrypt sertifikaatide maksimaalne kehtivusaeg on 90 päeva, tuleb serveri seadistustesse eraldi lisada sertifikaatide automaatne uuendamine. Automaatseks uuendamiseks on välja arendatud Certbot nimeline tarkvara, millel puudub vaikimisi MQTT serveri tugi. Bakalaureusetöö raames seadistas töö autor serveri sedasi, et peale igakordset automaatset sertifikaadi uuendust, taaskäivitatakse MQTT serveri konteiner uuendatud sertifikaadi kasutuselevõtuks. Bakalaureusetöö käigus koostatud lahendus koosneb viiest komponendist: MQTT server, PostgreSQL andmebaas, InfluxDB kronoloogiline andmebaas, .NET raamistikus kirjutatud tagarakendus ning Vue.JS raamistikus kirjutatud kasutajaliides. Iga komponent paigaldatakse Docker-pildist loodud konteinerina.

4.3 Näitude edastamine kasutades API-liidest

Korto arvelduskeskkond pakub mõõtmistulemuste edastamiseks haldusfirmadele ja teistele sarnase teenuse pakkujatele REST API liidest, mille otspunkt asub aadressil https://pro.korto.ee/api/readings. Testimiseks on võimalik kasutada ka testkeskkonda, kuhu ligipääsu saamiseks tuleb suhelda OÜ Tarkvaralaboriga, kes onKortot arendav ettevõte. API autentimine toimub JWT tokenit kasutades. Korto veebirakenduse API-s on võimalik vaid GET ja POST päringute kasutamine. GET päringut kasutades on võimalik API otspunkti dokumentatsiooni tuvastamine ja sedasi välja lugeda POST päringu koostamiseks vajalik info. Vastavalt dokumentatsioonile tuleb POST päringuga edastada massiiv mõõteandmetest, kus massiivi igas elemendis peab olema kirjas Korto keskkonnas seadistatud edastatava näidu mõõtja identifikaator, mõõtja näit ja näidu tüüp. Näidu tüüp võib olla „reading“ mis tähendab, et edastatakse mõõtja hetkenäitu või „consumption“ mis väljendab, et massiivis on tegu konkreetse mõõtja poolt mõõdetud kuluga ning kliendile esitataval arvel kuvatav arvutatakse selle põhjal. API täiskirjeldus on lisatud töösse lisana 2.

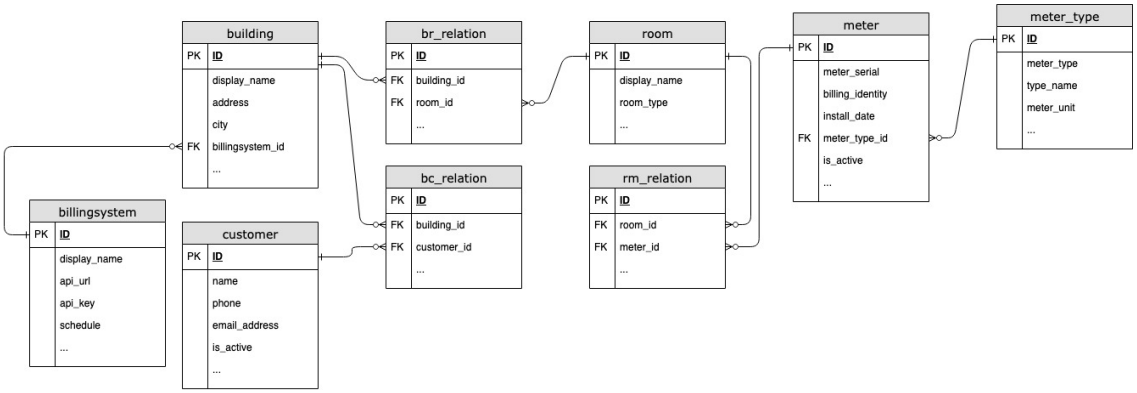
Saadetavas sõnumis peab sisalduma kirjas ka korteriühistu identifikaator. Seda on võimalik edastada kas massiivi sees, kasutades välja „client_id“ või päringu päises väljal „X-Client-ID“

Näide edastatavast sõnumist:

```
[
    {
    "client_id": "XXXXX",
    "meter_id": "11",
    "value": 178.4,
    "value_type": "reading"
    }
]
```

4.4 Rakenduse kirjeldus

Töö kirjutamise perioodil on arendatud rakendus, mis saab kortermajja paigaldatud kontrollerist infot veenäitude kohta, kasutades näitude vahendamiseks MQTT serverit. Tagarakendus ühendub käivitumisel MQTT serveri külge ning asub jälgima (subscribe) teemasid, mis tulenevad haldusfirma töötaja poolt relatsioonilises andmebaasis kirjeldatud seadistustest. Uue sõnumi saabumisel salvestab tagarakendus andmepunkti kronoloogilisse andmebaasi. Samuti kontrollib rakendus andmebaasist seadistatud näitude edastusgraafikut ja seadistab mälus vastavad taimerid. Taimeri täitumisel teeb tagarakendus päringu relatsioonilisse andmebaasi taimeri graafikuga seotud veemõõtjate identifikaatorite saamiseks ja seejärel pärib kronoloogilisest andmebaasist seotud mõõtjate näidud. Seejärel edastab rakendus näidud arveldussüsteemi, kasutades selleks konkreetse veearvestiga seotud konkreetse maja kohta andmebaasis kirjeldatud andmeid. Relatsioonilise andmebaasi diagramm koos tähtsamate olemite ja nendevaheliste seostega on kirjeldatud joonisel 8. Andmebaasimudeli loomisel on võetud arvesse võimalust tulevikus lisada ka muud tüüpi arvesteid.



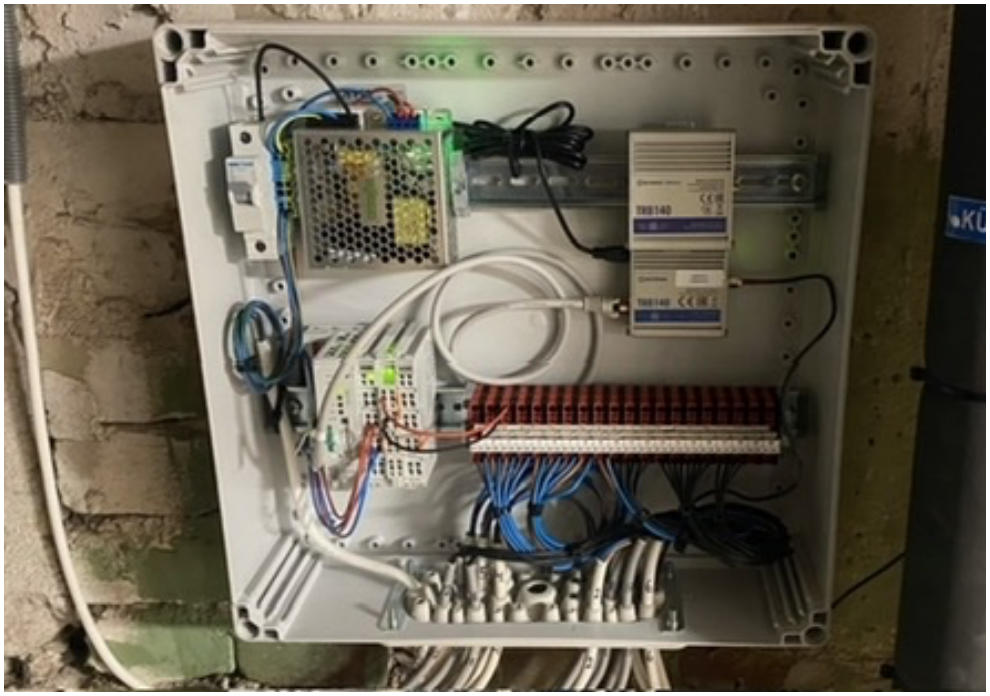
Joonis 8 Andmebaasi diagramm tähtsamate olemitega

Bakalureusetöö käigus on kortermajja paigaldatud kontrolleri tarkvara täiendatud sedasi, et kontroller edastaks perioodiliselt kõik näidud MQTT brokerisse.

Lisaks on haldusfirmale loodud kasutajaliides, kus haldusfirma töötajatel on võimalik lisada või muuta süsteemis maju, kortereid ning veemõõtjaid. Samuti saavad haldusfirma töötajad siduda kontrolleri poolt edastatud veemõõtja identifikaatorit arveldussüsteemis kirjeldatud veearvesti identifikaatoriga.

4.5 Testimine

Valmiva töö testimine toimus kahes faasis. Testimise eelduseks oli kontrolleri tarkvara täiendamine ja haldusfirma poolne internetilüüsi paigaldamine testimiseks valitud kortermajja. Paigaldatud seadmed on näha joonisel 9. Testimise esimeses faasis testitakse andmete automaatset edastamist arveldussüsteemi teenusepakkuja test-API kaudu. Testimine loetakse edukaks kui kõik näidud on edastatud teenusepakkuja testkeskkonda ja erinevus haldusfirma tehniku poolt kohapeal loetud andmetega puudub või on



Joonis 9 Pilootprojekti käigus paigaldatud seadmed

marginaalne. Marginaalne erinevus automaatselt loetud andmete ja haldusfirma tehniku poolt teostataval kohapeal loetud andmete vahel võib tekkida ajalisest nihkest lugemiste vahel. Teises faasis testiti andmete edastamist arvelduskeskkonna toodangukeskkonda. Testimine loetakse õnnestunuks, kui kõikide veemõõtjate näidud on automaatselt edastatud arvelduskeskkonda ja kortermaja elanikele on võimalik automaatselt esitatud andmete põhjal väljastada igakuised kommunaalteenuste arved ilma haldusfirma tehniku poolse käsitsi näitude võtmiseta.

5 Tulemused

Töö käigus on kortermaja kaugloetavate veemõõtjate keskseade ühendatud internetti ja tänu täiendatud kontrolleri tarkvarale on haldusfirma saanud oma töökorralduses teha soovitud muudatused. Kortermaja veenäidud on võimalik automaatselt edastada haldusfirma kesksesse serverisse ja haldusfirma tehnikul puudub vajadus käia igakuiselt objektil veenäitusid lugemas. Näidud on võimalik igakuiselt automaatselt edastada kortermajas kasutusel olevasse Korto arveldussüsteemi.

Haldusfirma esindaja hinnangul on töö käigus arendatud lahendusega võimalik liita ka teisi ettevõtte poolt hallatavaid kortermaju ja bakalaureusetöö käigus teostatud tööd on täitnud oma eesmärgid

5.1 Edasiste arenduste võimalused

Töö käigus valminud tarkvara on võimalik edasi arendada ka teist tüüpi mõõtjate näitude vastuvõtmiseks ja edastamiseks. Tarkvarasse on võimalik lisada uusi arveldussüsteeme, et haldusfirma kliendiks olevatel korteriühistutel oleks võimalik arveldus- ja raamatupidamissüsteeme vahetades näitusid automaatselt edastada. Samuti on võimalik tarkvara täiendada kontrollerist muud tüüpi info vastu võtmiseks. Haldusfirma esindaja on avaldanud soovi täiendada tarkvara häiresõnumite edastamise võimekusega ettevõtte tehnikutele, kes neile vastavalt reageerida saavaksid..

6 Kokkuvõte

Käesoleva töö eesmärgiks oli analüüsida võimalusi veemõõtjate mõõteandmete edastamiseks Interneti kaudu kortermajas kasutusel olevasse arvelduskeskkonda. Analüüsi järel teostati pilootprojekt ühe kortermajaga ning hinnati valminud lahenduse sobivust kasutuselevõtuks kõigis haldusfirma poolt hallatavates ja rekonstrueeritavates majades.

Kortermajja oli maja renoveerimise käigus paigaldatud automaatika keskseade Wago 750-8100 mis ühendati töö käigus Internetiga ning programmeeriti edastama näitusid kasutades MQTT protokolli.

Tarkvara arenduse käigus programmeeriti haldusfirma töötajatele kasutajaliides. Kasutajaliides võimaldab lisada maju, ruume ja arvesteid. Haldusfirma töötajad saavad kaugelt näha ka kortermaja ning iga korteri veetarbimise ajalugu. Valminud tagarakendus edastab kindlaksmääratud intervalli järel näidud JSON formaadis arvelduskeskkonda kasutades arvelduskeskkonna API-liidest ning ning neid andmeid kasutatakse majaelanikele tarbimisarvete väljastamiseks.

Kogu valminud lahendus paigaldati haldusettevõtte kasutuses olevasse serverisse kasutades Docker konteinereid. Valminud lahendus koosneb viiest konteinerist milleks on relatsiooniline andmebaasisüsteem PostgreSQL, kronoloogiline andmebaas InfluxDB, .NET raamistikus valminud tagarakendus, Vue.JS kasutajaliides ning Mosquitto MQTT

broker. Lisaks valiti kontrolleri Internetti ühendamiseks internetiruuter ning seadistati server automaatselt sertifikaate uuendama, et tagada andmete edastamise turvalisus.

Kortermaja veenäidud on võimalik automaatselt edastada haldusfirma kesksesse serverisse ja haldusfirma tehnikul puudub vajadus käia igakuiselt objektil veenäitusid lugemas. Näidud on võimalik igakuiselt automaatselt edastada kortermajas kasutusel olevasse Korto arveldussüsteemi..

Bakalaureusetöö käigus valminud lahendust on töö autoril koostöös haldusfirmaga plaanis edasi arendada ja lisada rakendusele uusi arvestite tüüpe ja häiresõnumite vastuvõtmise võimekus.

Kasutatud kirjandus

1. „MQTT Essentials“, HiveMQ GmbH [Võrgumaterjal] https://www.hivemq.com/mqtt-essentials/ [Vaadatud: 04.04.2023]
2. „Korto uuendused veebruaris“, OÜ Tarkvaralabor [Võrgumaterjal] https://korto.ee/public/news/et/2021/28-02-2021.md [Vaadatud: 04.04.2023]
3. „FAQ“, MQTT [Võrgumaterjal] https://mqtt.org/faq/ [Vaadatud: 04.04.2023]
4. „Manual“, WAGO Group [Võrgumaterjal] https://www.wago.com/global/d/9699 [Vaadatud: 25.02.2023]
5. „IT Explained: MQTT“, Paessler AG [Võrgumaterjal] https://www.paessler.com/it-explained/mqtt [Vaadatud: 04.04.2023]
6. „Understanding the MQTT Protocol Packet Structure“, Steve Cope [Võrgumaterjal] http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/ [Vaadatud: 05.04.2023]
7. „Comparing MQTT Brokers for the Industrial IoT “, Jeremy Theocharis [Võrgumaterjal] https://learn.umh.app/blog/comparing-mqtt-brokers-for-the-industrial-iot/ [Vaadatud: 05.04.2023]
8. „Building Facebook Messenger“, Lucy Zhang [Võrgumaterjal] https://engineering.fb.com/2011/08/12/android/building-facebook-messenger/ [Vaadatud: 04.04.2023]
9. „MQTT in an IoV scenario“, Guowei Li [Võrgumaterjal] https://www.emqx.com/en/blog/mqtt-for-internet-of-vehicles [Vaadatud: 04.04.2023]
10. MQTT Retained Messages Explained, Steve Cope [Võrgumaterjal] http://www.steves-internet-guide.com/mqtt-retained-messages-example/ [Vaadatud: 04.04.2023]

11. „MQTT Last Will and Testament Use and Examples“, Steve Cope [Võrgumaterjal] http://www.steves-internet-guide.com/mqtt-last-will-example/ [Vaadatud: 05.04.2023]
12. „MQTT Version 5.0“, Richard Coppen Andrew Banks Ed Briggs Ken Borgendale Rahul Gupta [Võrgumaterjal] https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html#_Toc3901060 [Vaadatud: 04.04.2023]
13. „Netflix relies on HiveMQ to run the Netflix app certification process“ HiveMQ GmbH [Võrgumaterjal] https://www.hivemq.com/case-studies/netflix/ [Vaadatud: 05.04.2023]
14. „Server Estimate“, EMQ [Võrgumaterjal] https://www.emqx.com/en/server-estimate [Vaadatud: 05.04.2023]
15. „System Requirements “, HiveMQ GmbH [Võrgumaterjal] https://www.hivemq.com/docs/hivemq/4.13/user-guide/system-requirements.html [Vaadatud: 05.04.2023]
16. [Võrgumaterjal] https://www.investopedia.com/terms/t/timeseries.asp
17. „What is a relational database?“, IBM [Võrgumaterjal] https://www.ibm.com/topics/relational-databases [Vaadatud: 06.04.2023]
18. „Why Do Relational Databases Use Primary Keys and Foreign Keys?“, Martyna Sławińska [Võrgumaterjal] https://learnsql.com/blog/why-use-primary-key-foreign-key/ [Vaadatud: 06.04.2023]
19. „DB-Engines Ranking of Relational DBMS“, solidIT consulting & software development gmbh [Võrgumaterjal] https://db-engines.com/en/ranking/relational+dbms [Vaadatud: 07.04.2023]
20. InfluxData Inc. [Võrgumaterjal] https://portal.influxdata.com/ [Vaadatud: 07.04.2023]
21. TDengine [Võrgumaterjal] https://tdengine.com/ [Vaadatud: 07.04.2023]
22. „PostgreSQL vs MySQL: The Critical Differences“, Mark Smallcombe [Võrgumaterjal] https://www.integrate.io/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/ [Vaadatud: 08.04.2023]
23. „What is .NET? Introduction and overview“, Microsoft [Võrgumaterjal] https://learn.microsoft.com/en-us/dotnet/core/introduction [Vaadatud: 14.04.2023]

24. VMware, Inc [Võrgumaterjal] https://spring.io/ [Vaadatud: 06.04.2023]
25. WAGO Group [Võrgumaterjal] https://www.wago.com/global/automation-technology/discover-software/ecockpit-engineering-software [Vaadatud: 25.02.2023]
26. „A Performance Comparison of RESTful Applications Implemented in Spring Boot Java and MS.NET Core“ Hardeep Kaur Dhalla 2021 J. Phys.: Conf. Ser. 1933 012041 [Võrgumaterjal] https://iopscience.iop.org/article/10.1088/1742-6596/1933/1/012041/pdf [Vaadatud: 15.03.2023]
27. „Comparative Web Performance evaluation of Vue & React using JSWFB“, Danial Eshete [Võrgumaterjal] https://medium.com/@danialeshete/comparative-web-performance-evaluation-benchmarking-of-vue-react-using-jswfb-a76982097225 [Vaadatud: 16.03.2023]
28. „Vue vs. React: Performance, Speed, Use Cases“, Vitaliy Ilyukha [Võrgumaterjal] https://jelvix.com/blog/js-frameworks-is-vuejs-better-than-react [Vaadatud: 16.03.2023]
29. „Vue vs. React in 2023“, Fireart Studio [Võrgumaterjal] https://fireart.studio/blog/vue-vs-react-in-2022/ [Vaadatud: 16.03.2023]
30. Joonistel 3, 4 5 kasutatud ikoonid:
    1. [Võrgumaterjal] https://www.flaticon.com/free-icon/settings_3067451
    2. [Võrgumaterjal] https://www.flaticon.com/free-icon/pc_2292112
    3. [Võrgumaterjal] https://www.flaticon.com/free-icon/mobile-phone_545245
    4. [Võrgumaterjal] https://www.flaticon.com/free-icon/temperature-control_3593297
    5. [Võrgumaterjal] https://www.flaticon.com/free-icon/settings_3067451
    6. [Võrgumaterjal] https://www.flaticon.com/free-icon/pc_2292112
    7. [Võrgumaterjal] https://www.flaticon.com/free-icon/mobile-phone_545245
    8. [Võrgumaterjal] https://www.flaticon.com/free-icon/temperature-control_3593297

9. [Võrgumaterjal] https://www.flaticon.com/free-icon/database-security_3161102

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

Mina, Mihkel Jõgeda

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Rakendus veearvestite näitude automaatseks edastamiseks“, mille juhendaja on Märt Kalmo
   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.

15.05.2023

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.

Lisa 2 – Korto mõõtmisandmete api kirjeldus


```
    "$schema": "http://json-schema.org/draft-07/schema",
    "title": "Korto remote meter reading API",
    "type": "array",
    "items": {
        "type": "object",
        "properties": {
            "meter_id": {
                "type": "string",
                "description": "Identifier of the meter in Korto database."
            },
            "client_id": {
                "type": "string",
                "description": "Identifier of the association in Korto
database. Can be omitted if sent in X-Client-ID header."
            },
            "value": {
                "type": "number",
                "description": "The submitted value."
            },
            "value2": {
                "type": "number",
                "description": "An optional secondary value, if the meter
measures multiple values (e.g. multi-tariff electricity meters)."
            },
            "value_type": {
                "type": "string",
                "description": "Type of the submitted value, either the
current reading of the meter, or measured consumption since last reading.",
                "enum": [
                    "reading",
                    "consumption"
                ]
            },
            "time": {
                "type": "string",
                "description": "Timestamp of the reading, in RFC3339 format.
If omitted, time of the request is used instead.",
                "format": "date-time"
            },
            "unit": {
                "type": "string",
                "description": "The unit that the submitted values are
measured in."
            }
        },
        "required": [
            "meter_id",
            "value",
```

```
            "value_type"
        ],
        "additionalProperties": false
    }
}
```