TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Rain Vink

TARGA KODU LAHENDUS KASUTADES IEEE 802.11 RAADIOKOHTVÕRGU STANDARDIT

Bakalaureusetöö

Juhendaja: Ivo Müürsepp
PhD

Tallinn 2017

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: Rain Vink

11.05.2017

Annotatsioon

Antud lõputöö eesmärk on luua võimalikult odav ja turvaline targa maja lahendus, kasutades selleks IEEE 802.11 raadiokohtvõrgu standardit. Eeldatavalt võiks selline lahendus anda suuremat kokkuhoidu ning suuremat kättesaadavust teiste tehnoloogiate ees ning aidata lisaks sellele ka kaasa kliima soojenemise aeglustumisele kasutades paremini juba olemasolevaid ressursse.

Töö tulemusena valmib targa kodu prototüüp, mis kasutab andmevahetuseks IEEE 802.11 raadiokohtvõrgu standardit. Prototüüp koosneb kolmest komponendist: kasutajaliidesest, keskjaamast ja kontrollitavast seadmest. Lisaks sellele võimaldab esialgne rakendus:

- sisse logimist kasutajanime ja parooliga
- otsida seadmeid kohalikust võrgust
- muuta kontrollitavate seadmete olekut

Töö eesmärgi tulemuse kindlaks tegemisel luuakse kõige pealt prototüüp kasutades IEEE 802.11 standardit ning see järel koostatakse sarnased süsteemid, juba turul olemas olevaid tehnoloogiate abil. Seejärel võrreldakse neid omavahel ning kirjutatakse selle alusel tulemus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 27 leheküljel, 4 peatükki, 10 joonist, 4 tabelit.

Abstract

A smart home solution using the IEEE 802.11 wireless local area network standard

The purpose of this thesis is to create the low cost and secure smart home solution using for it the IEEE 802.11 wireless local area network standard. Presumably, this solution should provide greater savings and better availability compared with other technologies already in the market. In addition, by making smart home solution more affordable to most of people in the world, should slow down global warming by making better use of the already available resources.

As a result of the work there will be a smart home prototype that uses to exchange data using the IEEE 802.11 wireless local area network standard. The prototype consists of three different components: user interface or web application, central control station and controllable device. In addition, preliminary system should allow next actions:

- login with username and password
- search devices in local area network
- change state of different devices

To find out the results of this thesis, there must be a smart home prototype that is using IEEE 802.11 wireless local area network standard. If this exist, then it is required to create same type of systems by using different technologies that are already developed and in the market. After this, it possible to compare all the solutions and write a result by using the data got from the comparison.

The thesis is in Estonian and contains 27 pages of text, 4 chapters, 10 figures, 4 tables.

Lühendite ja mõistete sõnastik

API	*Application programming interface* ehk rakendusliides, mille abil on võimalik erinvatel programmidel omavahel suhelda.
CORS	*Cross-origin resource sharing* ehk lähtekohtade vaheline ressursside jagamine, on mehhanism, mille abil on võimalik rakendustel omavahel andmeid jagada.
HTTP	*Hypertext Transfer Protocol* ehk hüperteksti edastusprotokoll on protokoll, mida kasutatakse andmete saatmiseks arvutivõrkudes. [1]
IEEE 802.11	Raadiokohtvõrgu standard mille sagedusaladeks on 2,4-2,4835 GHz ja 5.725-5.850 GHz [2].
ISO OSI	OSI on andmesideprotokollide kontseptuaalne mudel, mis jagab sõnumi edastamiseks vajaminevad funktsioonid 7 kihi vahel [3] ning mille on heaks kiitnud rahvusvaheline Standardimisorganisatsioon ehk ISO [4].
JSON	*JavaScript Object Notation* on andmevahetusvorming.
JWT	*Json Web Token* on JSON-i baasil kirjutatud turvaline autentimise meetod. [5]
MVC	*Model-view-controller* on tarkvara arhitektuurimuster, mis jagab rakenduse kolmeks erinevaks osaks: mudel, vaade ja kontroller. [6]
POST	HTTP protokolli päringu meetod.
REST	*Representation state transfer* on tarkvaraarhitektuuri stiil.
RF	*Radio frequency* ehk raadiosagedus.
SSL	*Secure Sockets Layer* ehk turvasoklite kiht on krüptograafiline protokoll, mis võimaldab turvatud võrgusuhtlust internetis. [7]
URL	*Uniform resource locator* ehk internetiaadress, mida kasutatakse ressursside leidmiseks ja kasutamiseks arvutivõrkudes.
Wi-Fi	Kaubamärk, mis kasutatakse seadmete puhul, mis põhinevad IEEE 802.11 standardil. [8]
IP aadress	*Internet protocol address* ehk internetiaadress on arvutite või muudeseadmete puhul kasutatav unikaalne aadress, selleks et leida teatud seade võrgust ülesse. [9]

WPA2	*Wi-Fi Protected Access II* on turvaprotokoll, mida kasutatakse turvalisuse tagamiseks traadita arvutivõrgus läbi seal oleva andmevahetuse krüpteerimise [10].
MAC aadress	MAC-aadress ehk meediumipöörduse juhtimise aadress on füüsilise võrguliidese unikaalne identifitseerija [11].

Sisukord

1.1 Ajalugu 11
1.2 Töö taust ja eesmärk 11
1.3 Metoodika 12
1.4 Ülevaade tööst 12
2.1 Targa maja eelised 13
2.2 Targa maja miinused 13
2.3 Olemasolevad tehnoloogiad 14
2.3.1 X10 14
2.3.2 Zigbee 15
2.3.3 KNX 16
3.1 Ülevaade 18
3.2 Keskjaam 19
3.2.1 Json Web Token 19
3.2.2 Andmebaas SQLite 20
3.2.3 Django Cors 21
3.2.4 TP-Link HS100 API 21
3.2.5 Arenduse käik 22
3.3 Kasutajaliides 26
3.3.1 Arenduse käik 27
3.4 Prototüübi installeerimine 28
4.1 Autori prototüüp 30
4.1.1 Skaleeritavus ja paigaldamise lihtsus 31
4.1.2 Turvalisus ja kvaliteet 31
4.2 X10 32
4.2.1 Skaleeritavus ja paigaldamise lihtsus 32
4.2.2 Turvalisus ja kvaliteet 32
4.3 Zigbee 33
4.3.1 Skaleeritavus ja paigaldamise lihtsus 33
4.3.2 Turvalisus ja kvaliteet 33

4.4 KNX 34
4.4.1 Skaleeritavus ja paigaldamise lihtsus 34
4.4.2 Turvalisus ja kvaliteet 35
4.5 Tulemus 35

Jooniste loetelu

Joonis 1. Prototüübi skeem 19
Joonis 2. JWT autentimis näide [26] 20
Joonis 3. Andmebaasi tabelite skeem 21
Joonis 4. TP-LINK HS100 nutilüliti [31] 22
Joonis 5. Projekti struktuur [32] 23
Joonis 6. JWT implementeerimine koodis 23
Joonis 7. Vastus päringule, kui kontrollitakse kasutaja autentsust 24
Joonis 8. Otsingul leitud seadme info 26
Joonis 9. Kasutaja tuvastamise vaade 38
Joonis 10. Kasutaja seadmete vaade 38

Tabelite loetelu

Tabel 1. Autori prototüübi hinna arvutamine 31
Tabel 2. X10 süsteemi hinna arvutamine 32
Tabel 3. Zigbee süsteemi hinna arvutamine 33
Tabel 4. KNX süsteemi hinna arvutamine 34

1 Sissejuhatus

Tark maja ehk nutimaja tähendab tänapäevases mõistes elektrooniliselt automatiseeritud hoonet, mis võimaldab endast näiteks valgustuse või temperatuuri juhtimist, erinevate seadmete sisse ja välja lülitamist ning turvasüsteemide haldamist. Siinkohal tuleb lisada, et tark maja on terviklik süsteem, kus pole lihtsalt loodud lisa võimalus seadmete kontrollimiseks, vaid kus juhtimine käib automaatselt sensoritelt ja loenduritelt tuleva info abil [12], [13].

1.1 Ajalugu

Automatiseerimise alguseks loetakse aastat 1885, sest just sellel ajal lõi Albert Butz termostaadi, mis suutis reguleerida kivisöest ahjude põletamis intensiivsust. Järgmine suur samm saavutati aastal 1975 kui valmistati esimene targa kodu internetipõhine tehnoloogia X10, mis on kuni tänaseni üks populaarsemaid lahendusviise kodustest süsteemides tänu taskukohastele hindadele. 1978 aastaks sisaldas X10 16 kanalilist juhtkonsooli, lambi- ja seadme moodulit, ning natukene hiljem lisandus ka nende hulka seinavalgustusmoodul ja esimene taimer. Tänapäeval on lisandunud X10 kõrvale ka teised targa kodu tehnoloogiad nagu Zigbee või Z-wave [12], [13].

1.2 Töö taust ja eesmärk

Tänapäeval räägitakse üha rohkem kliimasoojenemisest ja sellega kaasnevatest tagajärgedest. Leidub väga palju põhjuseid, miks selline probleem on viimasel ajal esile kerkinud. Näiteks rahvastiku kasvust tingitud suurem energia või toidu vajadusest, keskkonnale kahjulikud energia tootmis viisist ning metsade vähenemisest. Kuigi esialgu tundub, et antud probleem pole Eesti jaoks kriitilise tähtsusega, tuleks võtta elus loodust nagu hammasratastest, mutritest ja poltidest koosnevat süsteemi, kus ühe osa katki minekul langeb raskus teistele osadele. Jättes tähelepanu pööramata katkistele elementidele võib olukord eskaleeruda nii kaugele, et kogu süsteemi pole võimalik enam parandada ning kõige lõpuks lakkab ka kogu töö. Vaatamata sellele, et antud

probleemi lahendamisega tegelevad väga paljud inimesed toimub siiamaani pidev soojenemine. Niikaua kuni puudub piisavalt hea ja püsiv lahendus, peab inimkond rohkem panustama ressursside kokkuhoidu.

Rakendades siinkohal targa maja süsteemi võime vähendada elektrikulu kuni 20% ja saavutada 15% kokkuhoidu küttekuludelt ühe maja kohta. Vaatamata sellisetele näitajatele pole targa maja lahendused leidnud ühiskonnas laiemat kasutust. Peamine põhjus siinkohal on hind. Z500 kodulehe põhjal läheb tervik lahendus maksma umbes €1000 – 2000 [14], mis esialgu ei tundu olevat väga suur summa, aga kui arvestada, et enamik inimesi teenib Eesti keskmine bruto palka ehk umbes 1000€ kuus, siis lihtne matemaatika näitab, et süsteemi rakendamine on väga paljudele kallis ja ebatasuv.

Antud töö eesmärk on luua võimalikult odav ja turvaline targa maja lahendus, kasutades selleks IEEE 802.11 raadiokohtvõrgu standardit, mille tulemusena peaks lahendus pakkuma taskukohast automatiseerimist suuremale osale rahvastikust ning läbi selle ka andma inimestele võimaluse panustada rohkem võitlusesse kliimasoojenemisse hoides kokku kütte ja elektri peale kuluvat energiat.

1.3 Metoodika

Selleks, et jõuda eesmärgini analüüsitakse esmalt juba turul olevad tehnoloogiaid. Peale seda luuakse prototüüp ning see järel koostatakse sarnased lahendused, kasutades juba olemas olevaid tehnoloogiaid. Prototüübi arendamisel kasutatakse järgnevaid vahendeid: Node.js [15], Django [16] ja React [17].

Eesmärgi tulemuse saamiseks koostatakse erinevate tehnoloogiate abil samasugune süsteem. See järel võrreldakse neid erinevate kriteeriumite alusel.

1.4 Ülevaade tööst

Antud töö koosneb neljast osast. Esimeses osas analüüsitakse juba turul olevate targa maja tehnoloogiate üle. Teises osas kirjeldatakse prototüübi loomis protsessi. Kolmandas peatükis koostatakse prototüübile sarnane süsteem kasutades juba eelnevalt analüüsitud tehnoloogiaid ning neljandas osas jõutakse tulemuseni.

2 Analüüs

Antud peatükis tuuakse välja juba turul olevad tehnoloogiad ning analüüsitakse nende positiivseid ja negatiivseid külgi. Lisaks sellele räägitakse targa maja plussidest ja miinustest. Analüüsi tulemusi kasutatakse pärast lõpp tulemuse koostamisel.

2.1 Targa maja eelised

Peamiseks eeliseks, mida tark maja võimaldab on kokkuhoidu nii elektri, kütte ja ka vee pealt. Siinkohal tuleb võtta loomulikult arvesse, mis mastaabiga on tegemist. Suuremate hoonete puhul on kasutegur märksa suurem kui väiksemate puhul kuid vaatamata suurusele on säästlikus kindlasti tagatud.

Energiasäästlikkuse tagab süsteem peamiselt selliselt, et hoidutakse liigsest kulutamisest ehk kasutatakse ainult neid mooduleid, mis on antud hetkel vajalikud või reguleeritakse seadmeid vastavalt olukorrale. Järgmine näide kirjeldab, millise stsenaariumi puhul võib süsteem osutuda kasulikuks. Näiteks lahkub maja omanik terveks päevaks kiiruga kodust jättes põlema mõned valgustid ning samuti ka vannitoa kraani. Targa maja lahenduse korral tuvastaks süsteem, et inimene on majast lahkunud ning seejärel kõrvaldaks kulu allikad. Vastasel juhul saabub inimene õhtul koju ning märkab, et eelnevalt nimetatud seadmed on tööle jäetud ning seda on ka tunda maksude maksmisel [12], [13].

Lisaks sellele, kui tegemist on süsteemiga, mis mitte lihtsalt ei lülita valgusteid sisse ja välja, vaid suudab ka tuvastada lekkeid ja rikkeid võib see aidata hoiduda väga ebameeldivatest remondi kulutustest andes lisaks hoonele märksa pikema eluea [12], [13].

2.2 Targa maja miinused

Nagu igal asjal on kaks erinevat poolt, siis ei ole ka tark maja siinkohal erand. Rääkides miinustest, siis üheks suurimaks probleemiks siinkohal on kogu projekti maksumus. See

on üks peamisi põhjuseid, miks tänapäeval pole veel nutimajad nii Eestis kui ka mujal riikides väga levinud. Seda peetakse siiamaani veel luksuseks. Selle tõestuseks võib koostada järgmise matemaatilise näite. Ütleme, et Eestis elav kodanik saab keskmist palka, milleks on umbes 1000€ kuus. Sellest kätte saab ta umbes 807.20€. Kuid sellest kulutab ta 100€ söögi, 65€ elektri ja sama palju kütte peale. Lisaks sellele lisandub 100€ täiendavaid kulusid. Kätte jääb siis inimesel 477.20€. Juhul kui süsteem maksab 1000€, mis omakorda viib nii kütte kui ka elektri kulu 55 euroni, siis sellisel juhul jääks inimesele kätte iga kuu 20€ rohkem, näidates veel lisaks, et kogu süsteem tasuks ära umbes nelja aasta pärast. Sellised näitajad on ainult eeldusel, et kõik läheb hästi ja lisakulutusi pole vaja teha. Antud arvutus näitab, et targa maja tehnoloogia rakendamine siiamaani kallis ning üldjuhul ei õigusta see enda ära [12], [13].

Lisaks hinnale kerkib ülesse veel teinegi oluline küsimus. Kui turvaline kogu süsteem on ? Nagu me teame, et kõik mis on ühendatud võrku võib mingil hetkel sattuda pahalaste huvi orbiiti ja isegi ka nende käsutusse. Saades ligi kogu hoone andmetele ja õigustele on võimalik korda saata mitmeid erinevad kuritegusid. Alustades lihtsalt jälgimisest ja andmete kogumisest lõpetades inimelude ohtu panemisega.

2.3 Olemasolevad tehnoloogiad

Tänapäeval leidub väga palju erinevaid targa kodu realiseerimise võimalusi. Siinkohal tuleks jagada lahendused kahte suurde rühma. On olemas tehnoloogiaid, mis pakuvad kaugjuhtimist täielikult läbi raadiosignaalide ning lahendusi, mille korral luuakse ühenduse kasutades juhtmeid.

2.3.1 X10

X10 on avatud lähtekoodiga protokoll, mille abil on võimalik suhelda kodutehnika seadmete vahel läbi elektrijuhtmete või raadiolainete abil. Leiutati aastal 1975. Ettevõtte Pico Electronics kodutehnika juhtimiseks. Tänapäeval on X10 üks populaarsemaid lahendusi, mida kodu automatiseerimisel kasutatakse. Populaarsuse põhjuseks on erinevate moodulite suhteliselt odav hind ning lisaks on paigaldus ja skaleeritavus lihtne. Kogu süsteemi on võimalik ühendada kuni 256 erinevat moodulit või seadet.

X10 peamised miinused on:

- Käsklused võivad minna kaduma, kuna X10 saab ainult edastada 1 käskluse korraga. Juhul kui saadetakse kaks käsklust samal ajal, siis võib juhtuda, et käsklused põrkuvad omavahel ning sel juhul on neid võimatu dekodeerida. Võib juhtuda ka olukord, kus pannakse käima vale operatsioon.

- X10 on aeglane. Probleem tõuseb esile just siis, kui kasutatakse mõlemasuunalisi lüliteid või kui kasutada näiteks digitaalseid kontrollereid. Siinkohal on probleemi lahenduseks pakutud seadmete grupeeritust ja ka aeglasemaid muutuseid.

- Piiratud funktsionaalsus. See protokoll ei toeta näiteks timmimis kiirust, otsest trimmeri taseme seadistamist ning ka vooluringi grupeerimist.

- Juhul kui kaks kõrvuti olevat maja kasutab samu aadresse, võib tekkida olukordi, kus signaalid omavahel põrkuvad, mille tulemusena tekkib olukord, kus süsteemi töö häiritud. Lisaks sellele, ei ole välistatud, et ühe maja kontroller võib ka kontrollida teise maja mooduleid. Sama olukord kehtib ka juhul, kui kasutatakse raadiosignaale.

- Kõik X10 signaalid, kas siis läbi raadiolainete või vooluliinide on krüpteerimata. Siinkohal tõuseb väga olulisele kohale turvalisus, kuna antud süsteemi pole väga raske sel juhul häirida ja ka tekitada omanikule peavalu lülitades seadmeid ilma tema teadmata vastavalt oma tahtmisele [18], [19].

2.3.2 Zigbee

Zigbee on IEEE 802.15-4 põhineval standardil olev kõrgetasemeline sideprotokoll, mida kasutatakse peamiselt andmete edastamisel madala energiatarbega ja väikeste kiirustega seadmete puhul. Antud juhul ka siis kodu automatiseerimisel. See loodi aastal 1998 ning standardiseeriti aastal 2003. Selle leviala on umbes 10 - 100 meetrit sõltuvalt keskkonnast ning andmeedastuskiirus on defineeritud 250 kbit/s. Võrreldes nüüd X10-ga siis Zigbee võrgud on turvatud 128-bitise sümmeetrilise krüpteerimisvõtmega AES-128.

Tema peamised eelised on:

- Madal energia kasutus. See tagab pikaajalise kasutatavuse erinevatele seadmetele. Eelis tekib paikades, kus seadeldisi pole võimalik ühendada vooluvõrku.

- Kasutades silmusvõrgu topoloogiat suurendab see kogu süsteemi töökindlust. Kuna võrgus pole tsentraliseeritud seadmeid, tähendab see, et iga moodul võib olla vahendaja. Juhul kui üks masin lakkab töötamast, valitakse sõnumi saatmiseks mingi teine tee, tagades see läbi parema töökindluse. See eeldab, et seadmed on paigutatud õigesti.

- Turvalisuse tagab sümmeetriliste võtmete korrektne kasutamine. Kasutatakse AES-128 krüpteerimist.

- Zigbee seadmed on taskukohaste hindadega.

- Seadmeid on lihtne nii võrku lisada kui ka eemaldada.

Negatiivsed küljed:

- Zigbee piiranguks on mälu suurus ja andmete töötlemise kiirus [20].

- Selleks, et arendada Zigbee-d ärilistel eesmärkidel tuleb tasuda aasta maksu.

- Võib tekkida probleeme signaali edastamisel ja vastuvõtmisel, kuna ühe seadme leviala pole väga suur. Selleks, et seda probleemi vältida tuleb seadmed paigutada vastavalt [21], [22].

2.3.3 KNX

KNX protokoll on ISO OSI mudelile baseerunud võrguprotokoll, mida kasutatakse hoonete automatiseerimisel. KNX on kokku pandud kolmest standartist, mida kasutati eelnevalt. Nendeks on *European Home Systems Protocol* (EHS), *BatiBUS* ja *European Installation* Bus (EIB või Instabus). KNX-iga võimaldab pidada suhtlust läbi erinevate sidevahendite. Peamiseks ühendus viisiks seadmete vahel kasutatakse näiteks koaksiaalkaableid, elektrijuhtmeid ja Etherneti kaableid. KNX-i puhul on võimalik kasutada juhtimiseks raadiosignaale ja infrapunakiirgust, aga kuna nende komponentide

kasutamine tervik süsteemis on väga kallis, siis jäetakse need sidevahendid eesmärgi tõttu siinkohal välja. KNX-iga on võimalik koostada puu, siin või tähttopoloogiat.

KNX-i peamised eelised:

- KNX tagab suurepärase töökindluse ja kiiruse, tingituna sellest, et ühendused on loodud kaabli abil, mis kõrvaldab omakorda ka müra probleemid.
- Sobib suurtesse kohtadesse. KNX on täielikult jagatud võrk, mis suudab hallata 65536 seadet. Igasse alamvõrku on võimalik ühendada kuni 256 seadet.
- Testitud ja sertifitseeritud toodete valik on väga suur.
- Võimalik juhtida alates 8 bitilistest seadmetest kuni personaalarvutiteni välja.

Negatiivsed pooled:

- Võrgu ehitamisel tuleb tähelepanu panna suuresti projekteerimisele. Kuna valmis majadele on väga raske kaableid paigaldada, tuleks see töö ära teha hoone valmimise käigus. Leidub võimalus paigaldada valmis hoonetele, aga enamus juhtudel on see väga keerukas ning võib minna väga kulukaks.
- Võrgu parandamine võib osutada kulukaks. Sõltub see hoonest endast, keskkonnast ja probleemist. Kuna tavaliselt on juhtmed paigutatud peidetud kohtadesse, siis nende parandamine on kindlasti raskendatud ning võrreldes RF-mooduli vahetamine ka kallim.
- Paigaldamine on keerukas ja kallis.
- Seadmed on kallid [23], [24].

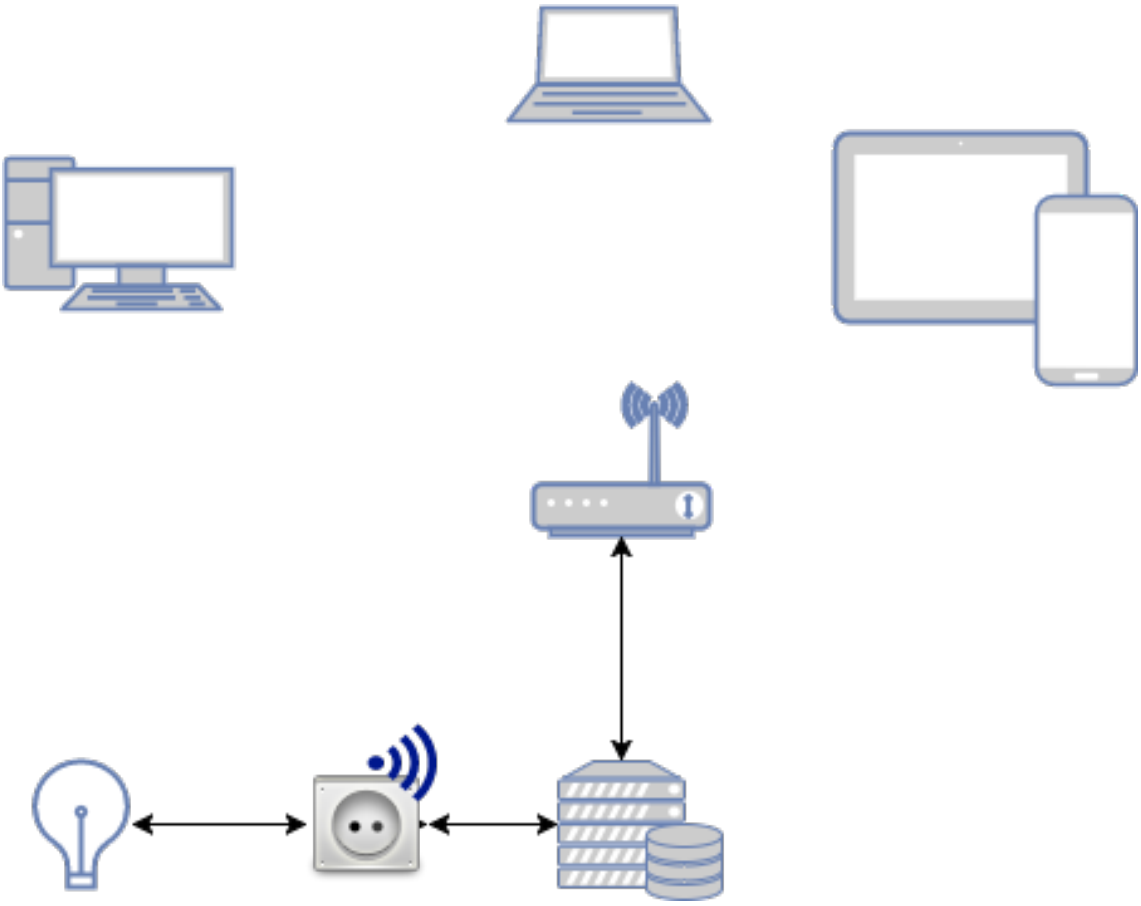
3 Prototüüp

Lõputöö käigus valmis targa kodu lahendus kasutades IEEE 802.11 raadiokohtvõrgu standardit, mis eelduste kohaselt võiks võrreldes teiste tehnoloogiatega olla odavam. Selles peatükis räägitakse lähemalt prototüübi juures kasutatavatest vahenditest ning ka valmimisest.

3.1 Ülevaade

Kogu prototüübi ülesehitus on väga lihtne. Kesksel kohal asub seade, milleks võib olla isegi lauaarvuti, milles on tööle pandud andmebaas, põhiserver ja kasutajaliides. Käskude jagamiseks kasutatakse HTTP [1] protokolli. Lisaks sellele on mõeldud kasutajaliides kõikidele seadmetele, milles on võimalik kasutada veebilehitsejat. Kuna sellisel juhul ei pea muretsema erinevate seadmete eripärade üle, siis on ka arendus kulud märksa odavamad.

Põhi server ehk keskjaam on loodud selleks, et oleks tsentraalne koht, mis omab ülevaadet kõikidest selles hoones olevatest võrku ühendatut seadmetest ning olla samuti vahendaja rollis. Lihtne näide kuidas käib juhtimine. Näiteks soovib kasutaja lülitada sisse laevalgustit. Selleks avab ta oma nutitelefonis oleva veebirakenduse, tuvastab ennast ning muudab vastava lüliti olekut telefonis. See järele tehakse päring serverile, kus vaadatakse, et kas antud kasutajal on õigust teha seda. Positiivse vastuse korral saadetakse sisse lülitamise päring vastavasse lülitisse. Pärast seda, kui lüliti on päringu kätte saanud ennast vastavalt reguleerinud, saadetud serverile tagasi vastus, mis lubab muuta ka andmebaasis oleva seadme olekut.



Joonis 1. Prototüübi skeem

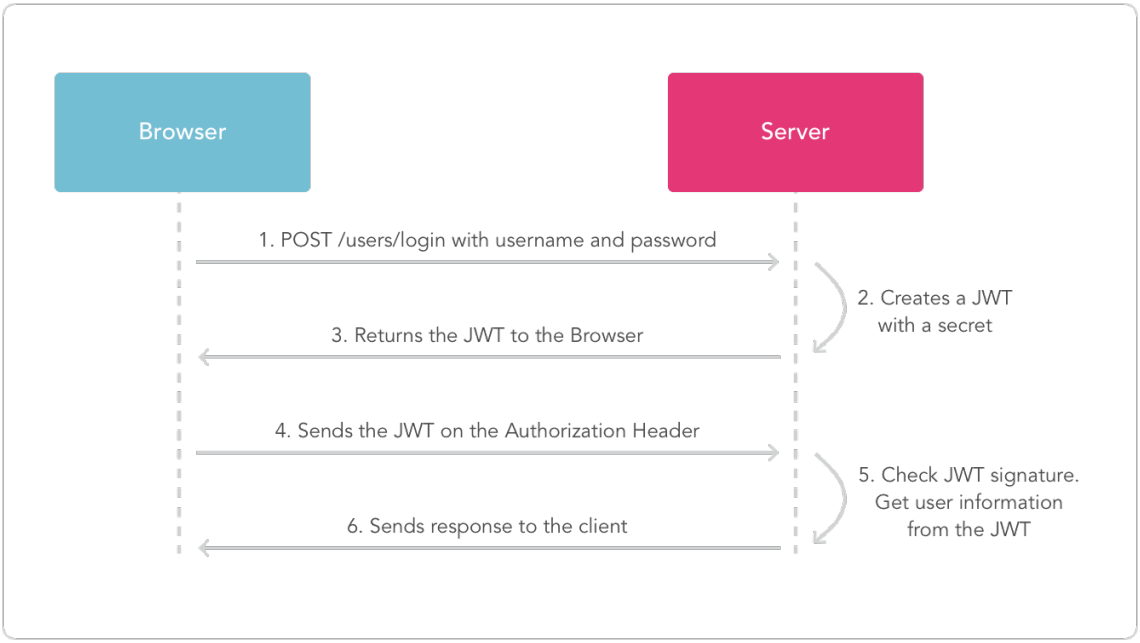
3.2 Keskjaam

Kogu süsteem suudab töötada igas võimalikus arvutis, mis on ühendatud kohalikku võrku ning millel on piisavalt võimsust ja mälu jooksutada Django serverit. Järgneva arendus keskkonna valiku juures on lähtutud põhiliselt keskkonna üles seadmise lihtsusest ja kulukusest ning Django vastab nendele nõutele. Sellel on olemas väga hea dokumentatsioon ning lisaks sellele ka kiire ja turvaline. Seda kasutavad näiteks Instagram, NASA ja Bitbucket [25]. Üks peamine tegevusi, mida server teeb on pakkuda API-teenust ehk olla vahendaja kasutaja ja kontrollitava seadme vahel. Lisaks sellele hoiab ta endas veebirakendust, mille kaudu on võimalik kogu süsteemi juhtida.

3.2.1 Json Web Token

Turvalisus on kliendi ja serveri vahel on tagatud JWT-ga. JWT loogika on väga lihtne. Kasutaja sisestab oma nime ja parooli. See järel saadetakse need serverisse, kus toimub kindlaks tegemine, et kas inimene on see kelleks ta ennast väidab olevat. Kui ei, siis kuvatakse vea teade. Vastasel juhul genereerib server pika sõne tüüpi väärtuse ehk

token-i oma salajase võtmega ning saadetakse see vastusena tagasi kasutajale. Pärast seda pannakse token päringu päisesse ning saadetakse iga korraga serverisse, kus siis kontrollitakse, kas see on valideeritud ja autentne.

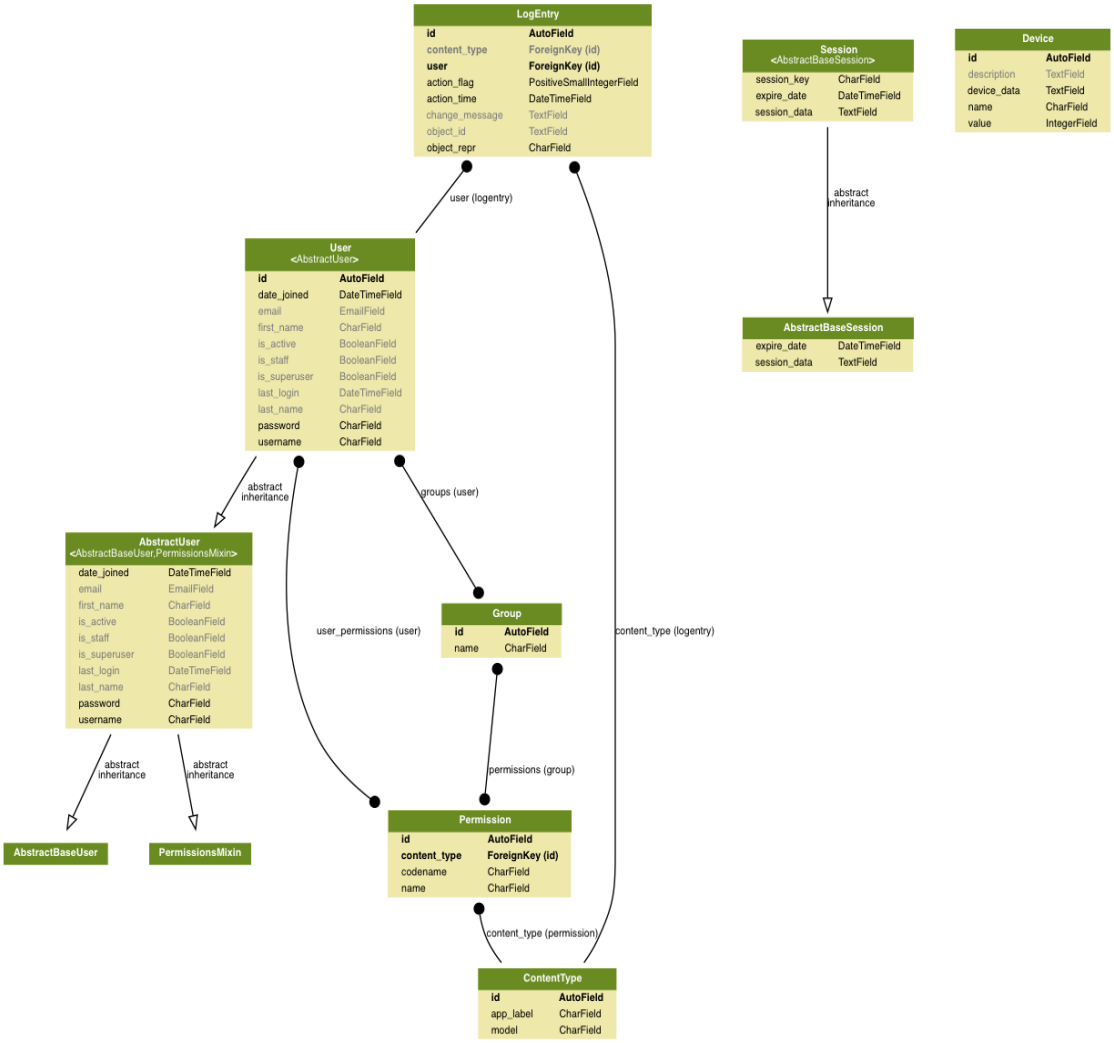


Joonis 2. JWT autentimis näide [26]

3.2.2 Andmebaas SQLite

Andmete hoidmiseks kasutakse andmebaasi süsteemi nimega SQLite-i [27]. Kuna töö eesmärgiks oli pakkuda võimalikult odavat lahendust, siis on arvestatud, et iga keskus asub inimese enda kodus, siis selle tõttu on ka antud hetkel SQLite piisav valik. SQLite on serverita toimiv andmebaasi mootor, kus kogu informatsioon on salvestatud ühte ainsasse faili. Ta on esiteks piisavalt kiire, kasutab vähe ressursse ning lihtne ülesse seada. Tema puudusteks aeglust suurte migratsioonide ja transaktsioonide arvu juures. Lisaks sellele ei sobi kohtadesse, kus on väga palju võrguliiklust ja andmete muutmist [28].

Kogu prototüübi lihtsuse mõttes on andmebaasi kasutatud ainult kahte peamist tabelit. Üks on kasutaja tabel ja teine on seadmete tabel. Seadmete tabelis on hetkel kirjeldatud ainult viis välja: seadme kood, nimi, kirjeldus, olek ja aadress. Kasutaja tabelis on peamisteks väljadeks on kasutajanimi ja räsitud ning soolatud parool.



Joonis 3. Andmebaasi tabelite skeem

3.2.3 Django Cors

Kuna veebirakendus ja keskjaam on üksteisest täielikul eraldatud, siis tekib probleeme sellega, et kaks rakendust ei saa omavahel andmeid vahetada just Cors-i piirangu tõttu. Selleks, et seda probleemi kõrvaldada kasutas autor juba olemas olevat raamistikku Django Cors [29].

3.2.4 TP-Link HS100 API

Prototüübis kasutatakse kontrollitavaks seadmeks nutilülitit TP-Link HS100. Selleks, et kohtvõrgust leida ja kontrollida vastavat seadet, võeti kasutusele Javascript-is kirjutatud moodul nimega *hs100-api* ning seda kasutati kohaliku uue Javascript-is kirjutatud API serveri tegemisel. Kuid siinkohal võib ligi pääseda ainult keskjaam [30].

Joonis 4. TP-LINK HS100 nutiüliti [31]

3.2.5 Arenduse käik

Arenduse esimese käiguna tuli luua uus Django projekti ning virtuaalne keskkond. Virtuaalse keskkonna loomiseks kasutati Pythoni paketti nimega virtualenv, mis tagab kogu projekti arendus keskkonna isoleerituse teistest süsteemidest. Näiteks võimaldab see kasutada neid lisasid, mis tegelikult arenduse jaoks vaja on. Lisades veel siia ka võimaluse installeerida kogu süsteem lihtsalt. Virtuaalse keskkonna loomisel kasutamiseks tuleb kõige pealt installeerida virtualenv. Selleks kasutati Pythoni paketi haldurit nimega *pip*. Kogu protsess alustamiseks tuleb käsureale sisestada järgmine rida "pip install virtualenv". Pärast seda tuleb liikuda keskkonna lähtekoodi kausta ja sisestada järgmine käsk "virtualenv venv". See järel luuakse samasse kohta virtuaalne keskkond, mille aktiveerimiseks tuleb kasutada käsku "source bin/activate".

Juhul, kui virtuaal keskkond on installeeritud ja käivitatud tuli edasi liikuda Django projekti loomise juurde. Täpselt sama moodi kasutatakse siinkohal uuesti pip-i selleks, et alla laadida Django raamistiks. Protsessi lõppedes avaldub võimaldus luua uus projekt käsuga "django-admin startproject switcher", mille tulemusena genereeriti järgnev struktuur [32].

```
switchcer/
    manage.py
    switchcer/
        __init__.py
        settings.py
        urls.py
        wsgi.py
```

Joonis 5. Projekti struktuur [32]

Manage.py - fail, mille abil on võimalik käivitada serverit ja teha muid tegevusi.

Settings.py - fail, kus on kirjas missuguseid lisapakette kasutatakse, serveri ja andmebaasi seadistused.

Urls.py - pannakse kirja serveri teekonnad.

Arendustöö esimese punktina realiseeriti autentimissüsteem, ehk kasutaja tuvastamine, JWT token-i valideerimine ning ka selle uuendamine. Siinkohal võeti kasutusele Django REST [33], mis on Python-is kirjutatud spetsiaalselt Djangole mõeldud teek, et luua lihtsalt ja kiirelt rakenduseliideseid kasutades REST-i põhimõtteid. Selleks, et rakendada JWT-d tuli lisaks Django REST-ile ka alla laadida Django REST JWT teek [34]. Kuigi mõlemad kõlavad üksteisele väga sarnaselt on tegemist ikkagi kahe erineva mooduliga. Django REST-i ainukene kasutusala antud juhul on see, et JWT-i teek kasutab seda enda koodis. Vaja minevate moodulite olemas olu korral tuli lisada url-i failis *urlpatterni* massiivi kolm uut elementi. Need on kuvatud järgmisel joonisel 4.

```
23  urlpatterns = [
24      url(r'^admin/', admin.site.urls),
25      url(r'^api/auth/login', obtain_jwt_token),
26      url(r'^api/auth/token/verify', verify_jwt_token),
27      url(r'^api/auth/token/refresh', refresh_jwt_token),
28  ]
```

Joonis 6. JWT implementeerimine koodis

Real 25 kirjeldatud objekt tegeleb kasutaja sisse logimisega. Kui kasutaja soovib ennast tuvastada, siis saadetakse vastavale aadressile POST tüüp päring, millele on kaasa antud nii kasutaja nimi kui ka parool. Juhul kui kasutaja on registreeritud andmebaasi, siis

tagastab server valideeritud token-i. Ebakorrektse päringu või registreerimata kasutaja korral tagastatakse vastav veateade.

Real 26 on kirjeldatud token-i autentsuse kontrollimine. Seda kasutatakse kontrollimaks, et kas antud kasutajal on õigus muuta ja lugeda seadmete andmeid. Teostatakse iga päringu korral kohtades, kus on tegemist kaitstud teekondadega. Juhul kui token pole autentne tagastatakse serveri poolt vastav veateade ning kasutaja peab ennast uuesti tuvastama. Kui token kehtib tagatakse serveri poolt olemas olev token. Näide vastusest joonisel number 6.


```
{
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9
      .eyJ1c2VyX2lkIjoxLCJlbWFpbCI6InJhaW4udmlua0BnbWFpbC5jb20iLCJ1c2VybmFtZSI6InJhaW4iLCJleHAiOjE0OTM3MTY5MTd9
      .-n4ZjWGtZYsCAIqhxLiGO5tL_Kg0P7XwRn0ieDgs1Gk"
}
```


Joonis 7. Vastus päringule, kui kontrollitakse kasutaja autentsust

Kuna veebirakendus ja server töötavad erinevatel aadressidel, siis tuli siinkohal pöörata tähelepanu Cors-ile, sest kõik päringud on vastuvõetamatud, mis on põhiserveri aadressist erinev. Selle probleemi likvideerimiseks kasutatakse Django Cors-i. Lisades Setting.py alla veebirakenduse aadressid ja teised parameetrid oli suhtlus kahe rakenduse vahel võimalik. [35]

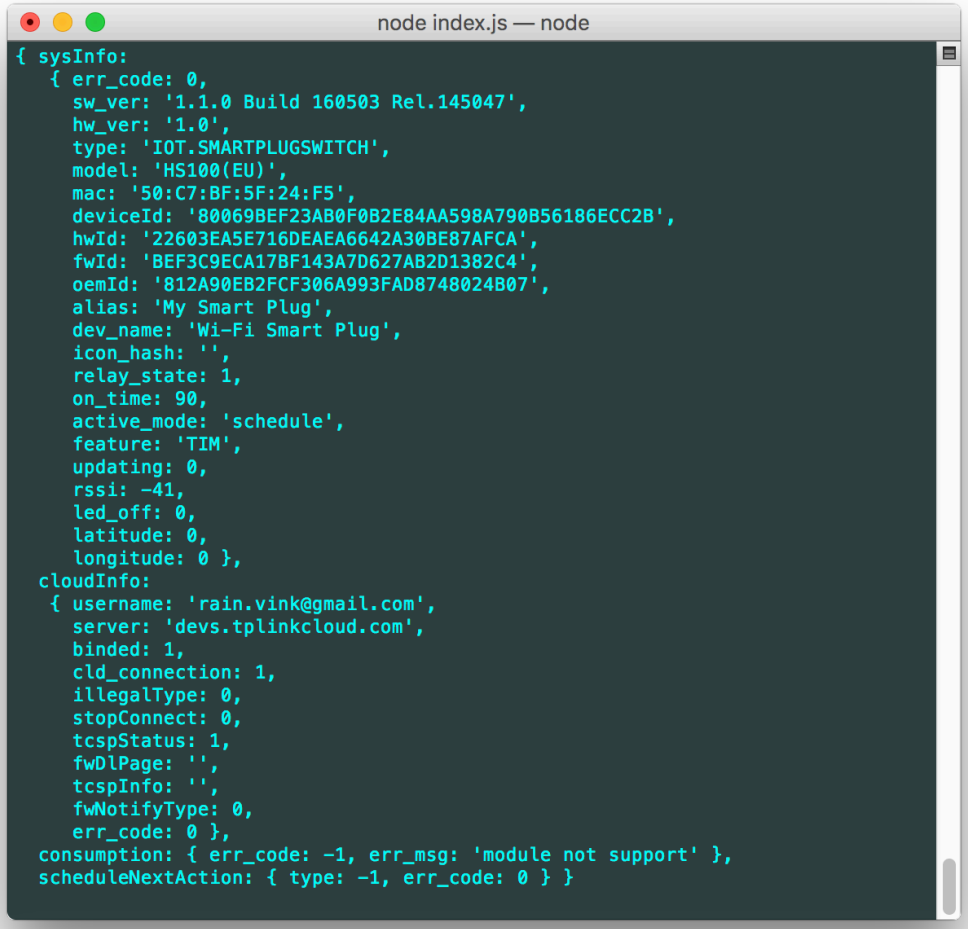
Järgmise samm projektis tuli autoril luua andmebaasi tabelid kui ka seadmete kontrollimiseks uued kaitstud teekonnad, millele on õigus ligi pääseda ainult tuvastatud kasutajatel. Esimese lingi eesmärgiks oli tagastada kõigi seadmete loetelu, mis on seotud kasutajaga ning teise lingi eesmärk on muuta kindla seadme olekut.

Seadme oleku muutmiseks tuleb kaasa anda kaks lisa parameetrid. Esimene parameeter on seadme kood, mis antakse kaasa juba url-is ning teiseks järgmise oleku väärtus. Minu prototüübi puhul saab olla nendeks väärtusteks 0 ja 1 kuna kasutatakse lihtsalt sisse välja lülitit. Timmeri olemas olekul võivad väärtusi olla rohkem. Juhul kui tuleb päring kasutajaliidesest, et tuleks muuta seadme olekut mille identifitseerimis koodiks on näiteks 1, siis see järel teostab server kontrolli, kas antud kasutajal on õigus seda teha, kas seade on registreeritud andmebaasi ning kas seadme järgmine väärtus on lubatud. Kriteeriumite ebaõnnestumiste korral tagastatakse kasutajale vastavad teated.

Vastasel korral toimub saadetakse käsk edasi seadmele ning seejärel muudetakse ka olekut andmebaasis.

Selleks, et andmebaasi lisada uus seadmete tabel tuli lähtekoodi kaustas käivitada järgmine käsk “python3 manage.py startapp devices”. Seejärel genereeriti uus kaust, kuhu oli autoril võimalik kirjutada seadmete otsimise, salvestamise ja kontrollimise loogika. Kuna Django kuulub MVC [6] mustri alla, siis on ka kaustas selle jaoks vastavad fail models.py loodud, kus saab kirjeldada seadmete andmebaasi tabelit. Selleks, et server aru saaks, et soovitakse luua uus tabel lisati settings.py all olevasse “INSTALLED_APPS” massiivi kirje *devices*.

Viimase sammuna keskjaamas oli puudu veel eelnevalt kirjeldatud API server seadmete otsimiseks ja kontrollimiseks. Kuna HS-100 node-i moodul oli kirjutatud juba varem Javascriptis, siis selle tõttu on ka vastav teenus kirjutatud samas keeles. Nagu ka teises sammus kirjeldatud on ka siinkohal loodud kaks vastavat teekonda. Esimene tagastab kõik võrgus olevate seadmete andmed ja teine suudab kindlat seadet kontrollida. Kogu serveri turvalisuse tagab asjaolu, et ainult keskjaama aadressilt on võimalik andmeid pärida. Juhul kui kõik komponendid olid valmis kirjutatud, tuli teha kogu süsteem terviklikuks. See tähendas, et Djangos tuli luua suhtlus kahe teenuse vahel.



Joonis 8. Otsingul leitud seadme info

3.3 Kasutajaliides

Veebirakenduse ehitamisel on kasutatakse React-i. React on Javascripti teek, mida kasutakse dünaamiliste veebirakenduste loomisel. Loodud on see Facebooki poolt ning baseerub komponentidele, mis annab eelise just arenduse kiirusele ning ka koodi loetavusele. Lisaks sellele pakitakse kogu veebirakendus kokku ühte faili, mis tagab selle, et iga uue lehekülje avamisel ei teha päringut serverile vaid see on juba kirja pandud vastavas failis. Vähendab oluliselt päringute arvu ning tagab ka parema kiiruse ja sujuvuse [17].

Lisaks sellele on kasutatud veebirakenduse loomisel veel selliseid vahendeid nagu Node, Npm [36] ja Sass [37]. Node on brauserist sõltumatu Javarscipti koodi käivitamiseks vajalik tööriist. Antud juhul kasutakse seda React-i ja Sass-i koodi

kompileerimiseks, käivitamiseks ja koodi kokku surumiseks. *Node package manager* ehk lühidalt öeldes npm on Node-i moodulite haldur, mida kasutatakse erinevate moodulite alla laadimiseks või kustutamiseks [36].

Sassi kasutatakse selleks, et kõrvaldada CSS-i teatud funktsionaalsuse puudujäägid võimaldades kirjutada paremini koodi ja teatud funktsioone. Näiteks üks eeliseid seisneb selles, et sass-is on võimalik ülem objekti sees kirjeldada vastavalt tema alam objekti kujundust muutmata seda teiste samanimeliste objektide juures [37].

3.3.1 Arenduse käik

Siinkohal sai arendus käik sai alguses sellega, et pandi kokku arendus keskkond. Kõige pealt uuendati node-i ning see järel laaditi alla Facebooki poolt loodud React-i arendus keskkond. Selleks, et seda teha tuli käsureal käivitada järgmine käsk "npm install -g create-react-app", mille peale tõmmati alla mooduli, mille olemasolul on võimalik arendus keskkond püsti panna.

Peale seda tuli luua projekti kaust käivitades järgmise käsu "create-react-app", mis siis kasutas eelnevalt installerituid mooduleid, et genereerida arendamiseks vajalikud kaustad ja failid. Kuna antud juhul kasutatakse Facebooki poolt olevat seadistust, mis teatud piirangute tõttu ei sobinud, siis tuli kloonida kõik moodulid ja vajalikud koodiread veebirakenduse kausta. Seda tehti lähtekoodi kaustas käsuga "npm eject". Probleem seisnes selles, kuna oli vaja lisada juurde teatuid lisasi, nagu näiteks Sass, siis seda polnud võimalik mugavamalt.

Selleks, et mitte panna väga palju rõhku kasutajaliideses disainimisele võeti kasutusele Bootstrap. Bootstrap on CSS-i teek, mida kasutatakse veebilehtede stiili arendamisel ja erinevate komponentide paigutamisel [38]. Pärast kõiki neid samme arendus keskkond valmis ning töö võis jätkuda arendusega.

Nagu ka keskjaama arendusel oli siis esimeseks eesmärgiks oli implementeerida kasutaja tuvastamine ning vastavalt ka sellele kasutaja vaate loomine. Selles faasis loodi kaks vaadet ehk lehe komponenti. Ühes komponendis oli kirjas väike tutvustus ja ka kasutaja tuvastamise väli ning teises ülevaade võrgus olevates seadmetest. Siinkohal tuli tähelepanu panna asünkroonsele Javascriptile, et ei tekiks juhuseid, kus päringute

vastused võiksid jääb hiljaks. Sellise probleemi tekkimisel, oleks kogu veebirakenduse töö häiritud.

Lisaks sellele tuli vaadata üle, et ainult autenditud kasutajal on õigus liikuda edasi järgmisele komponendile. Kui brauseri küpsistesse poldud salvestatud kehtivat token-it, siis juhatakse tagasi avalehele, kus siis kasutaja peab sisestama oma tunnused.

Peale autentimist hakati koostama kasutaja vaadet ehk inglise keeles *dashboard*. Siinkohal oli loogika väga lihtne. Peale kasutaja sisse logimist tehakse päring automaatselt serverile, kus tagastatakse kõik selle kasutajaga seotud seadmed. Kuna antud juhul kasutame ainult ühte seadet, siis saadetakse tagasi ühe seadme andmed.

Selleks, et kasutaja saaks lülitada sisse mingisugust seadet tuleb tal, kasutajaliideses vajutada sellele seadmele määratud nuppu. Pärast nupu vajutamist saadetakse päring serverisse, mis koosneb seadme koodist ja soovitavast väärtusest. Juhul, kui päring õnnestub kuvatakse muudatus ka kasutajaliideses. Probleemi tekkimisel kuvatakse vastav veateade.

3.4 Prototüübi installeerimine

Kogu lahenduse implementeerimine on tehtud võimalikult lihtsaks. Selleks, et käivitada prototüüpi läheb vaja ühte arvutit, mis on mõeldud serveri jooksutamiseks ning ühte kontrollitavat seadet, millele on loodud tugi serveri poole pealt. Selleks, et käivitada kogu süsteem tuleb kõige pealt alla laadida lähtekood github-ist. Lähtekood asub aadressil https://github.com/AngryDogs/switcher. See järel tuleb liikuda sealt edasi *backend* kausta ning installeerida kõik vajalikud teegid (pip install -r requirements.txt). Pärast installeerimist on võimalik käivitada serverit kuid enne seda tuleks luua andmebaas koos vaja minevate andmebaasi tabelitega (python3 manage.py migrate) ja kasutaja kasutades käsku (python3 manage.py createsuperuser). Juhul kui arvutisse pole installeeritud Pythoni versiooni 3.X, tuleks seda teha enne ühegi käsu sisestamisel. See järel võib käivitada serveri kasutades käsku “python manage.py runserver &”. Nüüd peaks kogu rakendus töötama pordil 8000.

Selleks, et käivitada veebirakendust läheb vaja node.js. Järgmise sammuna tuleb liikuda *frontend*-i kausta. Seal käivitada käsk “npm install”, mis tõmbab alla kõik vaja minevad moodulid. Pärast seda on võimalik käivitada kasutajaliides. Selleks, et seda tuleb

sisestada käsuliidesesse käsk “npm run start”. See järel kompileeritakse kogu kood ühte faili ning kogu liides on kätte saadaval pordil 3000. Juhul, kui soovitakse kasutada näiteks nutitelefoni, tuleb selleks avada telefoni veebilehitseja ning server ip aadress lisades lõppu port 3000. See järel peaks kuvatama ekraanil avaleht, kus kasutajal on võimalik ennast tuvastada vahetult ennem loodud kasutajaga.

Siinkohal on tegemist ainult prototüübiga. Juhul kui on tegu oleks valmis lahendusega, peaks kogu rakenduse installeerimine töötama ainult ühe faili käivitamisega läbi kasutajaliidese. Lisaks sellele kasutatakse näiteks Nginx-i [39], et serveerida port 80 pealt juba ehitatud veebirakendus ning selleks, et teha otsene ligipääs Django serverile võimatuks. See tagab omakorda kogu süsteemile turvalisuse ja stabiilsuse.

4 Tulemus

Antud peatüki eesmärk on võrrelda erinevaid tehnoloogiad autori omaga ning jõuda välja mingisuguse tulemuseni. Selleks, et jõuda vastuseni, tuli kõige pealt kokku panna samalaadse skeemiga targa kodu lahendus kasutades erinevaid tehnoloogiaid. Välja arvatud autori prototüüp on teised süsteemid koostatud teoreetiliselt ning reaalset töötavad lahendust nende tehnoloogiate abil finantsilistel põhjustel koostatud pole.

Vahetult enne tehnoloogiate võrdlemist, tuli püstitata siinkohal teatud tingimused. Esiteks on kogu süsteem jaotatud kolmeks erinevaks osaks. Nendeks on kasutajaliides, keskjaam ja kontrollitav seade. Teiseks, tuleb eeldada, et on olemas juba ruuter ja laualamp ning kõige lõpuks toimub kogu andmeside ainult kohalikus võrgus.

Selleks, et anda hinnang järgnevatele lahendustele, tuli välja mõelda punktid, mis on targa kodu lahenduse puhul põhilised. Nendeks kriteeriumiteks on hind, paigaldamise lihtsus, turvalisus, skaleeritavus ja kvaliteet.

4.1 Autori prototüüp

Antud hetkel pole olemas mitte ühtegi suurt ettevõtet turul, kes pakuks koduautomatiseerimise tervik lahendust kasutades IEEE 802.11 raadiokohtvõrgu standardit. Peamisteks põhjusteks, miks seda ei kasutata seda laialdaselt on seotud energia tarbimise ja ka erineva kasutusvaldkonna tõttu. Nimelt on mõeldud kõik selle standardi seadmed olema pidevalt ühendatud vooluvõrku ja kohtades, kus tegevused on andmemahukad.

Leidub seadmeid, kus andmeside käib läbi sama standardi, kuid siinkohal puudub keskjaam ja juhtimiseks kasutatakse otsesuhtlust. Keskjaama puudumisel tekib olukord, kus kasutajal on väga raske saada ülevaadet tervest hoonest. Lisaks sellele on ka funktsionaalsus piiratud. Näiteks kütte ja termostaadi omavaheline koostöö pole siinkohal võimalik.

Kuna prototüübis kasutati keskjaamaks isiklikku arvutit, et lihtsustada arendus käiku, toote hinna arvutamisel on arvesse võetud järgnevaid seadmeid: Raspberry Pi 3 Mudel B [40] ja nutilüliti TP-link HS100 [41]. Raspberry Pi peal hakkab jooksma kogu rakendus, mille hulka kuulub nii veebirakendus kui ka tuumserver.

Tabel 1. Autori prototüübi hinna arvutamine

Seadme nimetus	**Hind**
Raspberry Pi 3 Mudel B	46,52€
TP-link HS100	28,48€
Kasutajaliides	Tasuta
Arenduse omakulu seadme kohta	10,00€
Kokku	**85,00€**

4.1.1 Skaleeritavus ja paigaldamise lihtsus

Süsteemi jooksutamiseks tuleks ühendada nutilüliti ja Raspberry Pi vooluvõrku. Lisaks sellele tuleb arvuti ühendada ka ruuteriga selleks, et oleks võimalik pidada vastavate seadmetega. See järel tuleb avada kasutajaliides ning käivitada protsess, mis leiab võrgust ülesse õiged seadmed ning lisab kasutaja konto alla. Juhul, kui lisatakse süsteemi uus seade, tuleks uuesti käivitada samasugune protsess ning selle lõppedes peaks olema kontroll uue komponendi üle. Selline lahendus garanteerib, et nii skaleeritavuse kui ka paigaldamine jääb lihtsaks vältides seejuures ka paigaldusega seotuid kulusid. Kohalikku võrku olevalt ruuteri võimekusest on võimalik paigaldada teoreetilisel umbes 250 seadet [42].

4.1.2 Turvalisus ja kvaliteet

Turvaliseks suhtluse pidamiseks on plaanitud tavalist SSL-i protokolli. See peaks tagama piisava kaitse selle eest, et võõrastel ei leiduks võimalus sekkuda süsteemi töösse. Arvestada asjaoluga, et kogu suhtlus toimub kohalikus võrgus, siis lisanduvad lisaks SSL-ile ka täiendavad turvaelemendid nagu WPA2 ja MAC aadresside filtreerimine.

Rääkides kvaliteedist, võib eeldada, et antud tehnoloogiaga on võimalik pakkuda samalaadset kvaliteeti nagu Zigbee või Z-wave puhul. Seda järgnevatel põhjustel: esiteks suudetakse kiirelt transportida kohalikus võrgus päringuid, teiseks on signaali levik erinevates keskkondades parem. Zigbee leviku raadius on umbes 10 – 100 meetrit, Z-Wave-il 30 – 100 meetrit [43] ja Wi-Fi-l 50 – 100 meetrit.

Probleeme võib tekkida olukordades, kus keskjaam on teatud põhjustel rivist välja läinud ning see tõttu pole kogu süsteemi enam võimalik kontrollida. Lisaks lisanduvad ka piirangud seadme asukoha valiku puhul. Nimelt pole mõistlik paigaldada seadmeid

kohtadesse, kus pole võimalik saada toidet vooluvõrgust, sest vastasel juhul peaksid seadmed patareide abil vastu ainult paar päeva.

4.2 X10

Nagu eelnevalt mainitud sai on X10 kõigist olemas olevatest tehnoloogiatest kõige vanem ning see omakorda seab siinkohal teatud piirangud. Esiteks pole olemas väga palju võimalusi realiseerida juhtimine läbi arvuti või nutitelefoni ning need lahendused, mis on loodud, on kas vananenud ning omakorda ka ebaturvalised.

Selleks, et realiseerida sarnane prototüüp läheb vaja järgnevaid elemente: vastuvõtjat ja konverterit Marmitek TIP10RF [44], mis suudab muuta liidesest tuleva signaali X10 raadio signaaliks, Haibrain TM13U [45], mis muundada X10 raadiosignaali elektriliini signaaliks ning lisaks selle on ta küljes pistik, kuhu saab ühendada ka laualambi.

Tabel 2. X10 süsteemi hinna arvutamine

Seadme nimetus	**Hind**
Marmitek TIP10RF	65,69€
Haibrain TM13U	28,50€
Kasutajaliides	Tasuta
Kokku	**94,19€**

4.2.1 Skaleeritavus ja paigaldamise lihtsus

X10 on võrdlemisi lihtne paigaldada. Antud juhul tuleb ühendad Marmitek TIP10RF ruuteriga ning pärast seda ka Haibrain TM13U vooluvõrku. See järel avada seadmes vastav tarkavara ning järgida sealseid juhendeid. Kasutajaliideses on loodud võimalus leida võrgus olevad X10 seadmeid automaatselt.

Siinkohal võib järeldada, et paigaldamine on lihtne ning inimestel ei tohiks siinkohal probleeme tekkida. Tänu automaatsele otsingule on ka kogu süsteemi skaleeritavus lihtne ning mugav.

4.2.2 Turvalisus ja kvaliteet

Turvalisuse koha pealt võib jääda X10 seoses oma protokolli tõttu hätta. Nagu analüüsi osas sai mainitud, siis X10 puhul ei toimu andmevahetuse puhul krüpteerimist. See

omakorda loob võimaluse teistel inimestel pealt kuulata maja sees olevad andmesidet. See omakorda annab võõrastele inimestele lihtsamad võimalused kontrollida hoones olevad seadmeid, rikkudes sellega nii seadust kui ka teise inimese privaatsust.

Kuna X10 põhiline eelis tuleneb tema paigaldamise lihtsusest kasutades juba olevas olevat elektrisüsteemi, siis sellest tingituna ei pruugi kvaliteet olla pidevalt samasugune. See võib olla tingituna mürast või mitu seadet üritavad saata andmeid samal ajal.

4.3 Zigbee

Leidub vähe ettevõtteid, kes on täielikult spetsialiseerunud kodu automatiseerimisele kasutades Zigbeed. Sellised firmad nagu Zipato või Wink pakuvad lisaks tuge ka Z-wave-i protokollile. Kuna puudus otsene valik Zigbee seadmetel, siis selle tulemusena valiti Wink-i kodulehelt kaks seadet: Levitoni nutilüliti [46] ja Wink-i keskjaam [47].

Tabel 3. Zigbee süsteemi hinna arvutamine

Seadme nimetus	**Hind**
Wink Smart Hub	44,99€
Leviton Decora Smart Plug-in Outlet	55,00€
Kasutajaliides	Tasuta
Kokku	**92,28€**

4.3.1 Skaleeritavus ja paigaldamise lihtsus

Nagu ka X10 või autori prototüübi puhul on ka Zigbee seadmeid paigaldada lihtne. Kõige pealt tuleb ühendada keskjaam ja lüliti vooluvõrku. Seejärel tuleb käivitada rakendus oma seadmes ning jooksutada otsing seadmete leidmiseks. Tingituna sellise süsteemi lihtsusest on skaleeritavus lihtne ja mugav.

4.3.2 Turvalisus ja kvaliteet

ZigBee kasutab oma võrgu turvamiseks AES-128 krüpteerimis standardit, millest võib järeldada, et süsteemi sisse murdmine on raskem kui näiteks X10 korral. Tingituna seadmete võimusest ja keskkonnast, siis andmeedastusvahemaad jäävad üldjuhul vahemikku 10-st kuni 100 meetrini. Võrreldes seda Wi-Fi standardiga, siis antud

tulemus on märksa halvem, sest Wi-Fi signaali raadiuseks peetakse umbes 50-st kuni 100 meetrini.

Üldjuhul pakub Zigbee head kvaliteedi taset eeldusel, et seadmed on paigutatud vastavalt. Näiteks tuleb vältida olukordi, kus seadmed asuvad üksteises liiga kaugel või kus teatud seadmed on ainukesed ligipääsu punktid teistesse seadmetesse, sest muidu võib juhtuda olukord, kus pool süsteemist pole lihtsalt kättesaadav või masinate ümberlülitus võtab märgatavalt rohkem aega. Seda põhjusel, et Zigbee seadmed on loodud kestma kaua ning seoses sellega on nende mälu kasutus ja arvutamise kiirus piiratud.

4.4 KNX

KNX tehnoloogiat kasutav süsteem erineb teistest sellega, et ühenduse loomiseks kasutatakse kontrolleri ja kontrollitava seadme vahel kas Etherneti või koaksiaal tüüpi kaablit. Siinkohal on leitud internetist järgnevad seadmed: Zennio KNX-IP Interface PLess [48], mille abil on võimalik luua ühendus KNX võrgu ja näiteks nutitelefoni vahel ning Gira Socket [49], mis on hetkel kontrollitavaks seadmeks.

Tabel 4. KNX süsteemi hinna arvutamine

Seadme nimetus	**Hind**
Zennio KNX-IP Interface PLess	194,24€
Gira Socket	13,14€
Kasutajaliides ETS5Lite [50]	200,00€
Kokku	**407,38€**

4.4.1 Skaleeritavus ja paigaldamise lihtsus

Nii skaleeritavus kui ka paigaldus on KNX puhul võrreldes teistega raskem. Esiteks selleks, et luua ühendus nii ruuteri ja Zennio seadme vahel ning see järel kõikide teiste kontrollitavate seadmete vahel, tuleb kõige pealt hakata paigaldama kaableid. Siinkohal tuleb arvestada, et paljud inimesed ei ole elektrikud ning ei soovi vedada juhtmeid üle toa, siis lisaks tabelis olevale hinnale, lisandub sinna ka paigaldus kulud. Paigaldus kulusid on väga raske hinnata kuna see oleneb hoone suurusest kui ka struktuurist. Kõige lihtsam oleks KNX-i kasutada hoonete puhul, mis on alles ehtitus järgus.

4.4.2 Turvalisus ja kvaliteet

Kvaliteedi puhul on KNX kindlasti kõige parem, seda just põhjusel, et kaablitega ühenduse korral on tagatud alati järgnevad omadused: kiire reageerimis aeg, pöördumised jõuavad kindlalt adressaadini, seadmed on pidevas ühenduses ning kahe seadme vaheline raadius on kordades suurem kui teistel tehnoloogiatel.

Lisaks sellele ei pea ka muretsema väga turvalisuse üle, kuna pealt kuulamine tähendaks võõra inimese reaalset kohalolekut kogu võrgu juures. Ei ole välistatud, et see on võimatud, kuid tegemist oleks ikkagi väga raske ülesandega.

4.5 Tulemus

Antud peatüki alguses öeldi, et hinnatakse erinevate lahenduste juures järgnevaid kriteeriume. Nendeks oli hind, paigaldamise lihtsus, turvalisus, skaleeritavus ja kvaliteet. Kõik tooted otsiti interneti poodidest ning valituks said need seadmed, mis olid kõige odavamad.

Hinna poolest jäid peale nii autori lahendus kui ka Zigbee. Zigbee kaasamine antud kriteeriumisse on seotud põhjusega kuna autori lahenduse hind on hetkel ainult hinnang.

Täpselt sama valik langetati ka ma paigaldamise lihtsuse ja skaleeritavuse punkti juures. X10 jäeti valikust välja järgneval põhjusel. Tingituna selle tehnoloogia vanusest, siis kogu süsteemi ülesseadmine on võrreldes selle punkti võtjatega natukene keerulisem.

Teenuse kvaliteedis tuleb tunnistada, et KNX on teistest kindlasti parem, kuna see lahendus viis suudab lihtsalt tagada parema ühenduvust erinevate seadmete vahel, pakkudes samas ka kõige paremat kvaliteeti.

X10-et ei valitud mitte üheski kategoorias tingituna tehnoloogia vanusest. Seoses sellega, ei suuda see tehnoloogia olla enam teistest parem.

Võrreldes nüüd autori prototüüpi teistega, siis võib öelda, et kasutades kodu automatiseerimiseks IEEE 802.11 raadiokohtvõrgu standardit võib osutuda esialgse hinnangu puhul teistest kasulikumaks. Seda järgnevatel põhjustel: suudab pakkuda piisavalt head ühenduvust ja kvaliteeti, lihtne paigaldada ning lisaks ka kõige soodsam.

Suurematekс negatiivseteks külgedeks võiks välja tuua asjaolud, et seadmed ei sobi kohtadesse, kus ainukeseks toite allikaks on patareid või tuumserveri kadumisel on kogu süsteemi töö häiritud. Kuid tegemist pole selliste probleemidega, millele ei ole võimalik leida lahendust. Näiteks saaks lahendada seadmete asukoha probleemi sellega, et lisada keskjaama juurde Zigbee tugi, kuna selle Zigbee masinate eluiga sellistes tingimustes on suhteliselt pikk. Keskjaama katki minekul on võimalik lisada juurde tagavara seadmeid, mis võtaksid kontrolli üle kui midagi peaks peamise jaamaga juhtuma.

5 Kokkuvõte

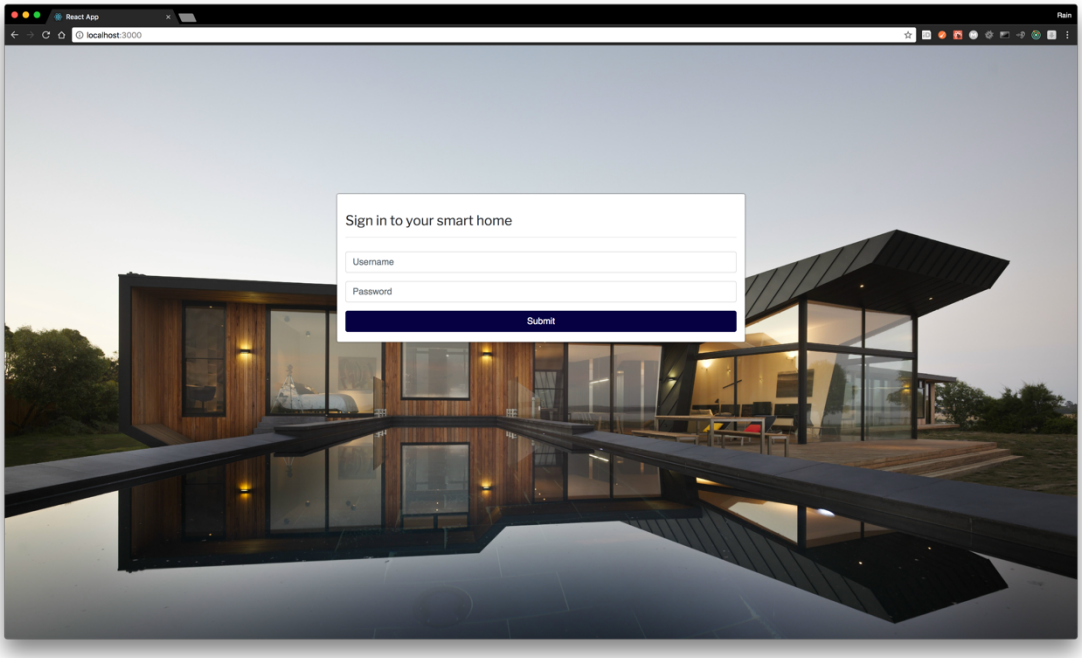
Antud töö eesmärgiks oli luua võimalikult odav ja turvaline targa maja lahendus, kasutades selleks IEEE 802.11 raadiokohtvõrgu standardit, mis annaks suuremat rahalist kokkuhoidu ning suuremat kättesaadavust teiste turul olevate tehnoloogiate ees.

Eesmärgi saavutamiseks tuli kõige pealt luua prototüüp, mis kasutab suhtluseks ainult eelnevalt nimetatud standardit. Süsteem koosnes kolmest peamisest komponendist: kasutajaliidesest, keskjaamast ja kontrollitavast seadmest. Arenduse käigus kasutati Javascripti teeki React, et töötada välja kasutajaliides. Tuum serveri loomiseks kasutati Node-i ja Django-t ning kontrollitavaks seadmeks leiti Internetist TP-Link HS100 nutilülitit.

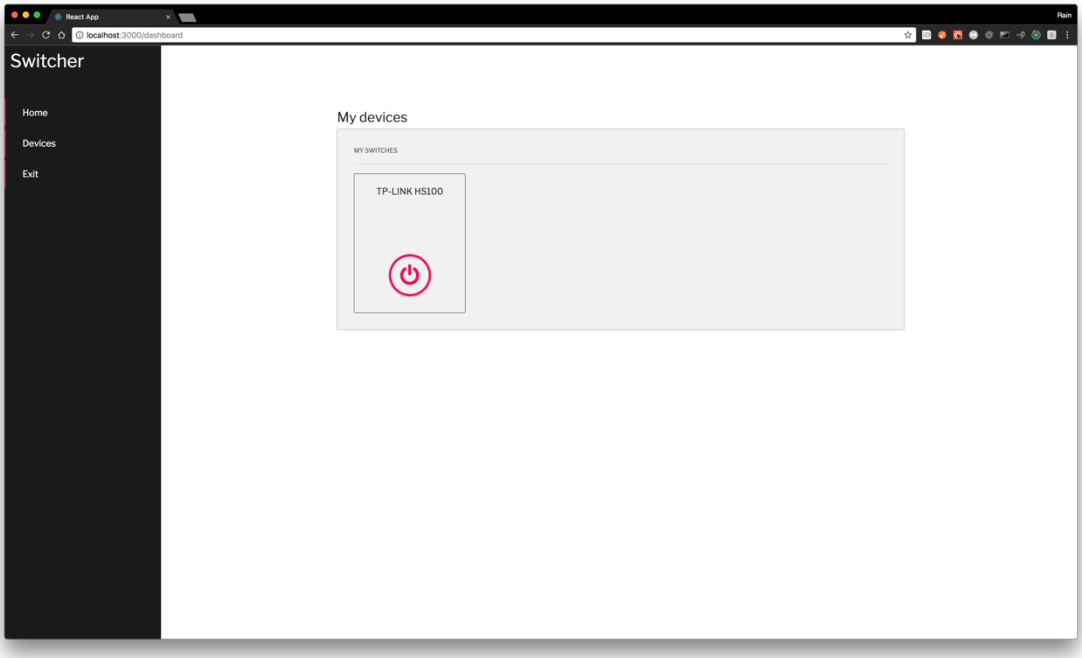
Prototüübi valmides oli võimalik luua teoreetiliselt samalaadsed süsteemid, mis kasutavad juba turul olevad tehnoloogiaid nagu Zigbee, KNX ja X10. See järel kirjutati välja nende tugevused ja nõrkused kasutades järgmisi kriteeriume: hind, paigaldamise lihtsus, turvalisus, skaleeritavus ja kvaliteet. Saadud informatsiooni kohaselt oli võimalik vaadata, kas antud töö täitis oma eesmärki või mitte.

Võrreldes nüüd loodud prototüüpi teiste lahendustega, siis võib öelda, et kasutades targa kodu süsteemi loomiseks IEEE 802.11 raadiokohtvõrgu standardit võib osutuda esialgse hinnangu puhul teistest kasulikumaks järgnevatel põhjustel. Esiteks suudab see pakkuda piisavalt head ühenduvust ja kvaliteeti. Teiseks ja kõige olulisemaks põhjuseks on süsteemi on odavus ja turvalisus. Lisaks sellele on seda võimalik ka väga lihtsalt paigaldada.

Lõppkokku võttes võib öelda, et antud töö tulemused oli positiivsed ja seoses sellega võib lugeda töö eesmärgi täidetuks.



Joonis 9. Kasutaja tuvastamise vaade



Joonis 10. Kasutaja seadmete vaade

Kasutatud kirjandus

[1] Hüperteksti edastusprotokoll. [WWW] https://et.wikipedia.org/wiki/H%C3%BCperteksti_edastusprotokoll (06.06.2008)

[2] IEEE 802.11. [WWW] https://et.wikipedia.org/wiki/IEEE_802.11 (20.01.2010)

[3] Avatud süsteemide sidumise arhitektuur. [WWW] https://et.wikipedia.org/wiki/Avatud_s%C3%BCsteemide_sidumise_arhitektuur (11.11.2008)

[4] Rahvusvaheline Standardiorganisatsioon. [WWW] https://et.wikipedia.org/wiki/Rahvusvaheline_Standardiorganisatsioon (15.04.2008)

[5] Jones M. B., Bradley J., Sakimura N. JSON Web Token (JWT). [WWW] https://tools.ietf.org/html/rfc7519 (05.2015)

[6] Model-View-Controller. [WWW] https://et.wikipedia.org/wiki/Model-View-Controller (27.08.2016)

[7] Transpordikihi turbeprotokoll. [WWW] https://et.wikipedia.org/wiki/Transpordikihi_turbeprotokoll (04.03.2010)

[8] Wi-Fi. [WWW] https://et.wikipedia.org/wiki/Wi-Fi (09.03.2011)

[9] IP-aadress. [WWW] https://et.wikipedia.org/wiki/IP-aadress (20.04.2011)

[10] Wi-Fi Protected Access. [WWW] https://en.wikipedia.org/wiki/Wi-Fi_Protected_Access (09.05.2017)

[11] MAC-aadress [WWW] https://et.wikipedia.org/wiki/MAC-aadress

[12] Tark kodu [WWW] https://et.wikipedia.org/wiki/Tark_kodu (19.04.2016)

[13] Home automation [WWW] https://en.wikipedia.org/wiki/Home_automation

[14] Tark maja – terviklik kodu automatiseerimine ja kontrollimine [WWW] http://z500.ee/tark-maja/

[15] About Node.js. [WWW] https://nodejs.org/en/about/

[16] Django. [WWW] https://www.djangoproject.com/start/overview/

[17] React. [WWW] https://facebook.github.io/react/

[18] X10 (industry standard). [WWW] https://en.wikipedia.org/wiki/X10_(industry_standard)

[19] X10 tehnoloogia. [WWW] http://www.dcc.ttu.ee/LAS/BCU3650/Lab_X10_EE_v1.02.pdf

[20] Zigbee Wireless Mesh Protocol. [WWW] https://sites.google.com/site/zigbeewirelessmeshprotocol/home/benefits-and-drawbacks

[21] Smart homes. [WWW] http://www.zigbee.org/what-is-zigbee/494-2/

[22] Zigbee. [WWW] https://en.wikipedia.org/wiki/Zigbee

[23] Introduction. [WWW] https://www.knx.org/knx-en/knx/technology/introduction/index.php

[24] KNX (standard) [WWW] https://en.wikipedia.org/wiki/KNX_(standard)

[25] Flagan M. 25 OF THE MOST POPULAR PYTHON AND DJANGO WEBSITES. [WWW] https://www.shuup.com/en/blog/25-of-the-most-popular-python-and-django-websites/ (07.08.2015)

[26] Introduction to JSON Web Tokens. [WWW] https://jwt.io/introduction/

[27] Features Of SQLite. [WWW] https://www.sqlite.org/features.html

[28] SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems. [WWW] https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems (21.02.2014)

[29] Ottoyiu. django-cors-headers. [WWW] https://github.com/ottoyiu/django-cors-headers

[30] Plasticrake. hs100-api. [WWW] https://github.com/plasticrake/hs100-api

[31] Wi-Fi Smart Plug HS100. [WWW] http://www.tp-link.com/us/products/details/cat-5516_HS100.html

[32] Writing your first Django app, part 1. [WWW] https://docs.djangoproject.com/en/1.11/intro/tutorial01/

[33] Django Rest Framework. [WWW] http://www.django-rest-framework.org/#

[34] GetBlimp. Django REST framework JWT. [WWW] http://getblimp.github.io/django-rest-framework-jwt/

[35] Pullen J. P. Everything You Need to Know About Smart Home Networking. [WWW] http://time.com/3745059/smart-home-wireless-networks/ (16.04.2015)

[36] These are the docs you're looking for. [WWW] https://docs.npmjs.com/

[37] Yard. Sass (Syntactically Awesome StyleSheets). [WWW] http://sass-lang.com/documentation/file.SASS_REFERENCE.html (28.04.2016)

[38] Otto M. Introduction. [WWW] https://v4-alpha.getbootstrap.com/getting-started/introduction/

[39] Welcome to NGINX Wiki!. [WWW] https://www.nginx.com/resources/wiki/

[40] Raspberry Pi 3 Mudel B. [WWW] http://www.ittgroup.ee/et/raspberry-pi-jt-miniarvutid/721-raspberry-pi-3-mudel-b.html

[41] TP-LINK HS100. [WWW] https://www.arvutitark.ee/est/tootekataloog/Vorguseadmed-Powerline449/TP-link-HS100-Smart-Plug-Wi-fi-216958

[42] B. Mitchell, „How Many Devices Can One Wireless Router Handle?. [WWW] https://www.lifewire.com/how-many-devices-can-share-a-wifi-network-818298 (19.04.2017)

[43] Z-Wave. [WWW] https://en.wikipedia.org/wiki/Z-Wave

[44] Marmitek TIP10RF Smartphone to X10 Interface. [WWW] https://www.uk-automation.co.uk/marmitek-tip10rf-smartphone-to-x10-interface/

[45] X10 Transceiver Module Single House Version TM13U. [WWW] https://www.uk-automation.co.uk/x10-transceiver-module-single-house-version-tm13u/

[46] Leviton Decora Smart Plug-in Outlet. [WWW] https://www.wink.com/products/leviton-decora-smart-plug-in-dimmer/

[47] Wink smart hub. [WWW] https://www.wink.com/products/wink-hub/

[48] Zennio KNX-IP Interface PLess. [WWW] http://www.myknxstore.co.uk/system-devices-c45/ip-interface-router-c242/knx-ip-interface-pless-p8683

[49] Gira Socket outlet with protective contact. [WWW] http://www.myknxstore.co.uk/electrical-accessories-c40/gira-socket-outlet-with-protective-contact-euro-us-16-a-250-v-p1685?attribute[1]=30

[50] ETS5 Prices. [WWW] https://www.knx.org/knx-en/software/ets/prices/index.php