**KAUNO TECHNOLOGIJOS UNIVERSITETAS**
**INFORMATIKOS FAKULTETAS**

**Marius Jokūbauskas**

# IŠMANIOJO NAMO PROGRAMŲ SISTEMŲ KŪRIMO, SKIRTŲ VEIKTI RIBOTŲ TECHNINIŲ RESURSŲ APLINKOJE, TYRIMAS

Baigiamasis magistro projektas

**Vadovas**
Doc. dr. Šarūnas Packevičius

**KAUNAS, 2016**

**KAUNO TECHNOLOGIJOS UNIVERSITETAS**
**INFORMATIKOS FAKULTETAS**

# IŠMANIOJO NAMO PROGRAMŲ SISTEMŲ KŪRIMO, SKIRTŲ VEIKTI RIBOTŲ TECHNINIŲ RESURSŲ APLINKOJE, TYRIMAS

Baigiamasis magistro projektas
**Programų sistemų inžinerija (kodas 621E16001)**

**Vadovas**
(parašas) Doc. dr. Šarūnas Packevičius
(data)
**Recenzentas**
(parašas) Doc. dr. Armantas Ostreika
(data)

**Projektą atliko**
(parašas) Marius Jokūbauskas
(data)

**KAUNAS, 2016**

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

(Fakultetas)

Marius Jokūbauskas

(Studento vardas, pavardė)

Programų sistemų inžinerija (kodas M4046M21)

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „išmaniojo namo programų sistemų kūrimo, skirtų veikti ribotų techninių resursų aplinkoje, tyrimas“

**AKADEMINIO SĄŽININGUMO DEKLARACIJA**

20 ____ m. _____ ____ d.

Kaunas

Patvirtinu, kad mano, **Mariaus Jokūbausko**, baigiamasis projektas tema „ išmaniojo namo programų sistemų kūrimo, skirtų veikti ribotų techninių resursų aplinkoje, tyrimas“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____ _____

*(vardą ir pavardę įrašyti ranka)* *(parašas)*

**TURINYS**

## PAVEIKSLŲ SĄRAŠAS



## LENTELIŲ SĄRAŠAS

Jokūbauskas, M. Išmaniojo namo programų sistemų kūrimo, skirtų veikti ribotų techninių resursų aplinkoje, tyrimas. *Programų sistemų inžinerijos magistro* baigiamasis projektas / vadovas doc. dr. Šarūnas Packevičius; Kauno technologijos universitetas, Informatikos fakultetas.

Mokslo kryptis ir sritis: Technologijos mokslų sritis, Informatikos inžinerija

Reikšminiai žodžiai: išmanusis namas, namų automatika, Raspberry Pi, Java, aparatūriniai ištekliai.

Kaunas, 2016. 63p.

## SANTRAUKA

Šių dienų prietaisai sudėtingėja, pingant galingesniems mikrovaldikliams, atsiranda galimybė juos jungti į tinklą, dalintis informacija. Išmaniųjų namų koncepcija nėra nauja, tačiau techninės sąlygos prieinamos kiekvienam prietaisui įsijungti į tinklą atsiranda tik dabar. Stebint šias tendencijas, atsiranda būtinybė kurti naujus įrenginius su dedikuota programine įranga šiam tinklui palaikyti ir valdyti. Šiame darbe nagrinėjamos tokių prietaisų programinės įrangos kūrimo, diegimo ir pritaikymo problemos. Darbe aprašytos išmaniųjų namų atsiradimo prielaidos, sukurti reikalavimai šiai sistemai. Pagal šiuos reikalavimus sukurta programinė įranga skirta išmaniųjų namų valdymui. Ši sistema veikia integruotame kompiuteryje „Raspberry Pi" veikiančio „Apache Tomcat®" aplikacijų serveryje. Tyrimo dalyje nagrinėjama šios sistemos greitaveika, siekiant išsiaiškinti, šios ir panašių, „Java" virtualioje mašinoje veikiančių, programų galimybes veikti riboto našumo integruotuose kompiuteriuose. Siekiama išsiaiškinti, kokių techninių parametrų kompiuteriuose sistemos greitaveika bus priimtina vartotojui bei ateityje padėti panašių sistemų architektams pasirinkti aparatūrinę įrangą, kurioje skandžiai ir patikimai veiktų sudėtinga virtualioje mašinoje veikianti programinė įranga. Buvo nustatyta, kad:

1. Kurta programų sistema veikia pakankamai greitai, kad patenkintų vartotojų lūkesčius, naudojant „Raspberry Pi 2B" integruotą kompiuterį.
2. Pagrindinis veiksnys lemiantis programų sistemos greitaveiką bei vartotojo laukimo trukmę – aparatūrinės įrangos lygiagrečių užduočių atlikimo galimybė daugiabranduoliuose procesuose.
3. Operatyvios atminties greitis, programinei įrangai veikiant Java virtualioje mašinoje, turį didelę įtaką sistemos greitaveikai.
4. Įtakos programų sistemos greičiui neturi pasirinktos atminties kortelės greitis.
5. Norint išnaudoti daugiabranduolių procesorių galimybes., lygiagretūs procesai turi būti tinkamai koordinuojami virtualios mašinos lygyje, t.y. parinkta tinkama VM implementacija.

Jokūbauskas, Marius. *Research On Developing Smart House Software In Limited Hardware Resource Environment*: *Master's* thesis in Software Engineering/ supervisor assoc. prof. Šarunas Packevičius. The Faculty of Informatics, Kaunas University of Technology.
Research area and field: Technology science, Software Engineering
Key words: smart home, home automation, Raspberry Pi, Java, hardware resources
Kaunas, 2016. 63p.

## SUMMARY

Nowadays, as home devices have become more and more complex and the price of powerful microprocessors have gone down, it has become a standard to connect devices to a single network so that they can share information. As a result, engineers are making devices with dedicated software to support such networks. In this thesis I will describe such software development and the problems associated with deploying and adapting it. This work used some available assumptions about the smart home to determine what the requirements of such system should be. We then developed software that satisfied these requirements. This software was deployed to a system-on-chip computer on a Raspberry Pi's Apache Tomcat Server. We then researched the system performance, when working on the Raspberry Pi (with its limited hardware). It is hoped that this research will help software architects to make an easy choice on platforms when they chose to deploy similar systems. The findings are as follows:

1. Our software works well on the Raspberry Pi 2B and satisfies customer needs.
2. When running on virtual machines, multitasking on multicore processors makes a huge impact on system performance.
3. Memory speed greatly effects systems performance.
4. The speed of SD memory cards has no effect on system performance.
5. If multitasking is available on a virtual machine, it should be utilized for optimal performance.

# 1. ĮŽANGA

## 1.1. Dokumento paskirtis

Šio dokumento tikslas yra magistrantūros studijų programos Programų sistemų inžinerija baigiamojo projekto ir jo tiriamosios dalies aprašymas.

Analitinėje dalyje aprašoma projektuojamos sistemos problemos ir spendimo būdai. Šioje dalyje suformuojama projektavimo ir programavimo problema bei uždaviniai. Analizės tikslas – išsiaiškinti, kokie yra vartotojų reikalavimai išmaniojo namo sistemai, kokios komunikavimo technologijos gali būti naudojamos ir apžvelgi aparatūrinė įrangą, kurioje galėtų veikti išmaniojo namo programų sistema.

Projektinėje dalyje aprašoma magistratūros studijų metu sukurtos programinės įrangos techninės-projektinės dokumentacijos esminiai aspektai. Projektinės dalies tikslas detalizuoti kuriamos sistemos reikalavimus ir aprašyti sukurtos programų sistemos architektūrą, bei kokybės kontrolę. Čia detalizuojamos išmaniojo namo programų sistemoje panaudotos technologijos.

Eksperimentinėje dalyje yra atliekamas magistratūros studijų metu sukurtos ir įdiegtos programinės įrangos bei jos patobulinimų eksperimentinis tyrimas. Šioje dalyje bandoma atsakyti į klausimą, kilusį analizės dalyje – kokių specifikacijų aparatūrinės įrangos reikia, kad išmaniojo namo sistema, sukurta pagal projektinėje dalyje aprašytas panaudotas technologijas, sėkmingai veiktų ir šios sistemos greitaveika tenkintų vartotoją.

## 1.2. Santrauka

Šių dienų prietaisai sudėtingėja, pingant galingesniems mikrovaldikliams, atsiranda galimybė juos jungti į tinklą, dalintis informacija. Išmaniųjų namų koncepcija nėra nauja, tačiau techninės sąlygos prieinamos kiekvienam prietaisui įsijungti į tinklą atsiranda tik dabar. Stebint šias tendencijas, atsiranda būtinybė kurti naujus įrenginius su dedikuota programine įranga šiam tinklui palaikyti ir valdyti. Šiame darbe nagrinėjamos tokių prietaisų programinės įrangos kūrimo, diegimo ir pritaikymo problemos. Darbe aprašytos išmaniųjų namų atsiradimo prielaidos, sukurti reikalavimai šiai sistemai, pagal šiuos reikalavimus sukurta programinė įranga skirta išmaniųjų namų valdymui. Ši sistema veikia integruotame kompiuteryje „Raspberry Pi" veikiančio „Apache Tomcat®" aplikacijų serveryje. Tyrimo dalyje nagrinėjama šios sistemos greitaveika, siekiant išsiaiškinti, šios ir panašių, „Java" virtualioje mašinoje veikiančių, programų galimybes veikti riboto našumo integruotuose kompiuteriuose. Siekiama išsiaiškinti, kokių techninių parametrų kompiuteriuose sistemos greitaveika bus

priimtina vartotojui, bei ateityje padėti panašių sistemų architektams pasirinkti aparatūrinę įrangą, kurioje skandžiai ir patikimai veiktų sudėtinga virtualioje mašinoje veikianti programinė įranga.

# 2. ANALITINĖ DALIS

## 2.1. Įvadas

Šios dalies tikslas yra suformuoti kuriamos sistemos problemą ir jos sprendimo būdus. Čia suformuojamas projektavimo ir programavimo problema bei uždaviniai.

Šioje dalyje aprašoma išmaniojo namo sistema, kam ji reikalinga ir kam gali būti naudojama. Supažindinama su egzistuojančiais sprendimais bei suformuojamos kuriamos išmaniojo namo sistemos projektavimo gairės, galimos realizavimo technologijos ir būdai.

Analizės tikslas – išsiaiškinti, kokie yra vartotojų reikalavimai išmaniojo namo sistemai, kokios komunikavimo technologijos gali būti naudojamos ir apžvelgi aparatūrinė įrangą, kurioje galėtų veikti išmaniojo namo programų sistema.

## 2.2. Išmaniojo namo sistemos paskirtis

Šiuolaikiniame automobilyje yra apie 90 įvairių rūšių jutiklių ir vykdančių prietaisų, kurie sujungti į tinklą geba reaguoti į aplinkos parametrus ir padaryti vairavimą saugesnį, ekologiškesnį, pigesnį ir patogesnį. Tuo tarpu šiuolaikiniuose namuose dažnai aptiksime bent 4–9 valdiklius, tačiau kiekvienas jų veiks atskirai, valdydamas savo „bloką“, pvz. termostatas, ventiliacijos valdiklis, kondicionierius. Išmaniųjų namų koncepcija numato, kad šie prietaisai bus sujungti į tinklą ir galės keistis informacija bendraudami tarpusavyje (M2M). Pavyzdžiui informacija apie patalpos temperatūrą iš ventiliacijos įrenginio bus perduodama kondicionieriaus įrenginiui.

Išmaniųjų namų koncepcija nėra nauja. Technologijų entuziastai jau 6-ame praeito amžiaus dešimtmetyje svajojo ir kūrė automatizuotų namų sistemas. Pingant technologijai, iš pavienių entuziastų namų, išmaniojo namo technologijos veržiasi į kiekvienus namus.

Norint įdiegti išmanųjį namą reikalinga pragmatinė motyvacija. Toliau bus bandoma apžvelgti kelis šios motyvacijos aspektus.

**Sumažinti energijos nuostolius**. Išmaniajame name energijos reikalaujantys veiksniai: šildymas, šviesa, vėdinimas, kondicionavimas paskirstomas pagal poreikius, paros metą, patalpų „užimtumą“. Išmaniajame name ne tik bus išjungtos šviesos patalpose, kuriose nėra

žmonių, bet ir atsikėlus vėlai naktį einant atsigerti, vandens šviesos intensyvumas bus pritaikytas taip, kad neakintų šeimininko [1].

**Vartotojų ramybė**. Ar išmaniojo namo vartotojas pamiršo neišjungtą lygintuvą? Nieko tokio, sistema pati išjungs, arba naudotojas prisiminęs tai autobuse, išjungs savo telefonu. Išmaniojo namo sistemos plačiai integruojamos su saugos sistemomis Jos ne tik informuoja saugos kompaniją apie keistą veikseną, bet ir imituoja žmonių buvimą, kad atbaidytų potencialius vagis.

**Ekologija**. Klimato atšilimas tampa realiu pavojumi, todėl tikslingas energijos panaudojimas ir atsinaujinančios energijos „integravimas“ į išmanųjį namą tampa realiu uždaviniu. Protingas namas apskaičiuoja savo, kaip namo, ar jo šeimininkų anglies pėdsaką ir taip didindama namo gyventojų sąmoningumą.

**Gerovė. M**okesčiai sudaro didelę dalį šeimos pajamų. Investicijos į išmaniojo namo energijos sistemą galėtų sumažinti sąnaudas iki 23 % [2].

**Visuomenės nauda**. Žemų ir vidutinių pajamų gyventojams sutaupymas už komunalines sąskaitas gali būti vienas iš faktorių, lemiančių geresnę gyvenimo kokybę, sutaupyti pinigai gali būti panaudoti vietinei ekonomikai kelti, o ne didinti sąskaitą už importuojamą kurą.

**Pagalba negalios ir senatvės atveju.** Išmaniojo namo ir jutiklių tinklų panaudojimas negalios ir senatvės atveju šiuo metu įgauna vis didesnį patrauklumą. Dėl didelės ir augančios senolių slaugos kainos, siekiama, kad senyvo amžiaus žmonės kuo ilgiau galėtų pasilikti savo namuose. Siekiant užtikrinti saugumą ir gyvenimo kokybę, naudojami jutiklių tinklai, integruoti į išmaniojo namo sistemą, kurie gali pagal poreikį pranešti artimiesiems, gydytojui ar greitosios pagalbos tarnybai apie esamą ar pasikeitusią būklę [3].

Kaip matome iš pateiktų motyvacijų, kiekvienas žmogus gali rasti naudą, kurią jam galėtų suteikti išmaniųjų namų sistema.

## 2.3. Reikalavimai šiuolaikinei išmaniojo namo sistemai

Remiantis ZigBee asociacija [4] šiandien išmaniojo namo sistemai keliami tokie reikalavimai:

Lengvas įrengimas:

- Paprastas sistemos įrengimas – pasidaryk pats;
- Tinkama tiek naujai statybai, tiek renovacijai;

- Savaime komutuojami tinklai lengvam įrengimui;
- Veikimas „be rūpesčių", minimali priežiūra.

Internetinė prieiga:

- Galimybė valdyti įrenginius per atstumą iš bet kurios pasaulio vietos;
- Mobilių įrenginių panaudojimas išmaniojo namo valdymui.

Galios kontrolė:

- Energijos panaudojimo stebėsena;
- Nuotolinis prietaisų įjungimas / išjungimas.

Saugumas:

- Lengvai pridedami įrenginiai, norint sukurti integruotą išmaniojo namo sistemą;
- Įdiegta saugumo ir šifravimo sistema.

Nekomplikuotos išmaniojo namo sistemos sumažina sistemos kainą galutiniam vartotojui, todėl tai yra svarbus faktorius, lemiantis išmaniojo namo sistemos populiarumą. Norint, kad vartotojas galėtų pats įsidiegti tokią sistemą, reikalinga išsami dokumentacija, kuri būtų pritaikyta paprastam vartotojui. Čia neturėtų būti minimi standartai ar protokolai, kuriais remdamasi sistema veikia, bet, galbūt, tiesiog apsiribojama įrenginiais, su kuriais tokia sistema yra pasiruošusi dirbti kartu.

Naujos statybos namų yra palyginti nedaug, todėl sudarant sąlygas įdiegti išmaniojo namo sistemą esamuose būstuose be didesnių statybos ir remonto darbų, padidinamas tokios sistemos potencialių vartotojų skaičių. Sudėtingos instaliacijos nereikalauja sistemos, kurios geba veikti su šiai įprastiniais belaidžiais „WiFi" ir „Bluetooth" protokolais. Kiti belaidžiai protokolai, kaip „ZigBee" ir pan., taip pat nereikalauja sunkių diegimo procedūrų, tačiau gali sąveikauti su, jau įprastu namuose, „WiFi" tinklu [5], [6].

Tinklai, kuriuose pritaisai įjungiami automatiškai, savaime turi privalumų prieš tuos, kuriuose reikia nustatyti įvairius parametrus – tokius, kaip adresas, greitis ar pan. Tokius prietaisus lengviau integruoti į bendrą sistemą. Tokius tinklus gali pasiūlyti „WiFi", „Bluetooth" ir „ZigBee" technologijos. Iš dalies ši problema yra sprendžiama ir „1-wire" tinkluose, nes kiekvienas prietaisas turi unikalų adresą, o šio tinklo greitis yra nustatomas automatiškai.

Išmaniojo namo sistemų priežiūra neturėtų kelti vartotojams naujų rūpesčių. Sistemos aptarnavimo periodai turėtų būti, kiek įmanoma ilgesni. Sudėtingiems pramoniniams

įrenginiams yra įprasta mėnesinė, ar metinė priežiūra, automobilių sektoriuje – taip pat dažnai rekomenduojama metinė arba priežiūra pagal nuvažiuotą atstumą. Namų vartotojai nėra prie to įpratę, todėl priežiūros ar derinimo laikotarpiai turėtų būti įmanomai ilgi. Pageidautina, kad bent jau programinės įrangos atnaujinimai išmaniųjų namų programinėse sistemose vyktų automatiškai. Deja tokia praktika, kaip rodo plačiai nuskambėję pavyzdžiai [7], [8], susijusi su didesne rizika sugadinti (angl. bricking) sistemą atnaujinimo metu. Taigi, jeigu programų sistemoje numatyta automatinio atnaujinimo galimybė, reikėtų atsižvelgti į šią riziką ir takyti griežtas programų sistemos kokybės procedūras.

Dar nesenai į išmaniojo namo sistemas buvo diegiamos belaidžio korinio ryšio technologijos, valdymas trumposiomis žinutėmis [9], [10]. Šiandien, plačiai paplitus išmaniesiems telefonams su interneto prieiga, dažnai tokių brangių technologijų yra atsisakoma, jungiant išmaniojo namo sistemas į interneto tinklą. Išmaniojo namo sistemos įjungimas į tinklą, esant įgyvendintai vartotojo sąsajai, suteikia galimybes valdyti prietaisus iš bet kurio pasaulio taško, kur yra interneto prieiga.

Pagal Europos IoT strategiją [11] sunaudojamos energijos matavimas naudojant išmaniuosius prietaisus, kurie gali perduoti duomenis nuotoliniu būdu, yra vienas iš prioritetų. Be galimybės perduoti suvartojimo duomenis tiekėjui automatiškai, išmaniojo namo sistema gali šiuos duomenis patogiai nuolat atvaizduoti vartotojui, kuris gali stebėti resursų suvartojimo dinamiką, ir, galbūt, priimti atitinkamus sprendimus, kaip sumažinti energijos ar išteklių (vanduo, dujos) sunaudojimą.

Išmaniųjų namų saugumui skiriamas didelis dėmesys [12]–[14]. Saugumas apima tiek prietaisų bendravimo protokolų saugumą, tiek vartotojo identifikavimo problemas. Išmanusis namas turi būti prieinamas internetu, o tai yra didžiausia saugumo rizika. Programišiai, kenkėjiškos programos gali sugadinti išmaniojo namo sistemą, paveikti ar sugadinti prie jos prijungtus prietaisus, nutekinti informaciją apie naudotojų asmeninį gyvenimą. Tai, suprantama, nėra pageidaujama nei iš protingų namų kūrėjų perspektyvos, nei iš šių namų naudotojų pusės.

Potencialūs vartotojai ypač jautriai reaguoja į sistemos kainą. Remiantis [15] straipsniu, vartotojams prieinama sistemos kaina yra ypač svarbi. Iki šiol išmaniojo namo sistemų kaina buvo didelė. Dėl atvirų standartų paplitimo, įrangos pigimo, masinio efekto matomos bendros tendencijos kainai smarkiai sumažėti.

## 2.4. Potencialūs vartotojai

Iki šiol išmaniojo namo technologijos buvo išskirtinai turtingų asmenų, kaip Bill'o Gates'o sodyba Vašingtono valstijoje, arba pamišusių entuziastų namų projektai, arba technologijų demonstravimo projektai kuriami universitetuose. Pingant technologijoms, protingų namų sistemomis galės naudotis vis daugiau ir daugiau žmonių. Pagal Philippopoulos'ą (2004), artimiausioje ateityje savo namuose protingų namų technologijomis naudosis su naujausiomis technologijomis dirbantys žmonės – programuotojai, elektros ir automatikos inžinieriai, bankų sektoriaus tarnautojai ir kt., bei negalią turintys žmonės. Remiantis „Dzone" žurnalu [16], dauguma programuotų norėtų "daiktų interneto, kaip hobio projekto.

Remiantis Philippopoulos'u (2004), protingų namų vartotojai nėra pasiruošę lengvai priimti tam tikras intervencijas savo namuose. Autoriai pateikia sprendimą, kad jokių naujų kabelinių sistemų montuoti nereikės.

Sprendimas buvo pateiktas 2006, kai Elektros ir elektronikos inžinierių institutas (angl. IEEE - Institute of Electrical and Electronics Engineers) 802.15.4 standartu apibrėžė belaidžių mažos galios radijo ryšių specifikaciją. Šis tarptautinis standartas sudarė sąlygas sukurti galios nebrangius prietaisus, kurie belaidžiu ryšiu komunikuotu tarpusavyje. Tokių prietaisų energijos resursas, jei jie maitinami iš baterijų, gali siekti 3 ir daugiau metų [17]. Tačiau ši technologija, kol kas, nėra labai populiari. Smarkiai atpigus vietinių belaidžių tinklų (angl. WiFi) bei „Bluetooth" sistema luste (SoC) modulių kainai, tikėtina, kad daiktų internetas remsis šiais, jau ilgą laiką egzistuojančiais protokolais. Šios technologijos egzistuoja dešimtmečius ir yra pasitvirtinusios savo patikimumą ir išplečiamumą.

## 2.5. Protokolų ir technologijų apžvalga

Vienas iš išmaniojo namo paplitimą ribojančių veiksnių gali būti suvienodintų technologijų sklaida. Nėra nusistovėjusių aiškių standartų, kai kurie naudojami protokolai yra firminiai ir patentuoti. Nors šiuo metu pareinama prie viešai paskelbtų standartų (kaip „Zigbee"), vis dėlto atsiranda ir panašių-skirtingų technologijų, kurios kai kuriais parametrais gali lenkti esamus („Bluetooth LE", „6LoWPAN").

### 2.5.1. I2C sąsaja

Tai mažo greičio sąsaja, skirta sujungti periferiją su integruotais kompiuteriais, mikrovaldikliais [18]. Nuo 2006 nereikia licencijavimo. Tačiau periferijos gamintojams reikia išsipirkti adresus iš „NXP".

I2C sąsaja sujungimui naudoja du laidus, t. y. duomenų liniją ir laikrodžio liniją (taktiniai impulsai). Duomenų mainai vyksta pagal vedantysis-vedamasis (master-slave) bendravimo logiką. Naudojama įrenginių adresacija. Duomenys apdorojami tuo pačiu laidu abejomis kryptimis. Įprastiniu rėžimu veikia 100 kb/s greičiu. Paskutiniai standarto pakeitimai leidžia įrenginiams veikti iki 3,4 mb/s greičiu [19].

### 2.5.2. „1-Wire“ sąsaja

„1-Wire“ tai „Maxim“ kompanijos vieno duomenų srauto sąsaja, naudojama „Maxim“ gaminamuose temperatūros jutikliuose. Ši sąsaja leidžia prijungti neribotą kiekį jutiklių prie vieno valdančiojo įrenginio [20]. Duomenų perdavimui ir maitinimui gali būti naudojamas vienas kabelis, t. y. „parazitinės” srovės maitinimas. Duomenų transakcijos metu įkraunamas vidinis kondensatorius, kurio energijos pakanka išsiųsti duomenis valdančiajam įrenginiui [21]. Įrenginiai turi unikalius adresus, todėl šiuo protokolu veikiančiuosius įrenginius patogu naudoti, kai visam įrenginiui reikia priskirti adresą [22].

Atstumas tarp valdančiojo įrenginio ir tolimiausio jutiklio priklauso nuo prie linijos prijungtų jutiklių kiekio, didžiausias atstumas tarp valdančiojo įrenginio ir tolimiausio jutiklio – 750 metrų. Duomenų perdavimo greitis – 14 kilobitų/s su galimybe padidinti greitį iki 140 kilobitų/s [21].

„1-Wire” protokolas plačiai paplitęs naudojant šiuo standartu veikiančių termometrų tinklus. Šis tinklas gali veikti iki 500 m atstumu [23] ir gali būti naudojamas unikaliai įrenginių adresacijai [24].

### 2.5.3. SPI sąsaja

SPI (angl. serial peripherial interface) – nuosekli periferinė sąsaja. Duomenys tuo pačiu metu gali būti perduodami iš vedamojo įrenginio į vedantįjį ir atvirkščiai. Naudojami 4 laidai SS – vedamojo aktyvinimas, SCLK – laikrodis (taktiniai impulsai), MOSI (angl. master out slave in) ir MISO (angl. master in slave out) duomenų laidai.

Trūkumai – nėra patvirtinimo, kad duomenys gauti. Greitis iki 4 mb/s [19].

### 2.5.4. RS-485

Tai laidinis protokolas, kuriuo įrenginiai bendrauja serijiniu ryšiu naudodami dviejų laidų sistemą. Bendravimas šiame tinkle gali būti inicijuojamas tik pagrindinio (angl. Master) prietaiso. Duomenys gali būti perduodami aštuoniais bitais. Šis protokolas lengvai suderinamas su kitais serijiniais protokolais: „RS422”, „ModBus”.

### 2.5.5. Serijinių protokolų apibendrinimas

Geriausiai išmaniųjų namų poreikius tenkina „1-wire" ir „RS-485" sąsajos. Atstumas nuo informacijos šaltinio yra pakankamai didelis ir nereikia daugelio laidų duomenų linijų. Tai sudaro sąlygas lengvesnei ir pigesnei instaliacijai. SPI ir I2C protokolas yra tinkamas įrenginiams, kurie nėra nutolę nuo valdančiojo įrenginio.

**2.1 lentelė** Laidinių komunikacijų protokolų, naudojamų išmaniuose namuose, palyginimas

| | **1-Wire** | **I2C** | **SPI** | **RS-485** |
|---|---|---|---|---|
| **Tinklo koncepcija** | Vienas šeimininkas, daug priklausomų (slave) | Daug šeimininkas, daug priklausomų | Vienas šeimininkas, daug priklausomų | Vienas šeimininkas, daug priklausomų |
| **Signalinių linijų kiekis** | 1 (IO) | 2 (SCL, SDA) | 4 (CS, SI, SO, SCK) | 2 (A,B) |
| **Tinklo dydis** | Iki 300m | 400pF talpumo reikalavimas | 2-3cm | 400m |
| **Įtampa** | 2,8 – 6,0 V | 1,8 – 5,5V | 1,8 – 5,5V | 3-12V |

## 2.6. Belaidžiai protokolai

### 2.6.1. Blueotooth low energy

„Bluetooth low energy" arba „Bluetooth 4" – tai taip pat 2.4GHz dažniu veikiantis protokolas, skirtas mažiems, mažai energijos reikalaujantiems prietaisams. Nors su „Bluetooth" dalinasi tais pačiais radijo dažniais, tačiau nėra suderinamas. Šis protokolas yra labai plačiai paplitęs ir naudojamas daugumoje šiuolaikinių mobiliųjų telefonų, nešiojamų ir stacionarių kompiuterių bei kituose prietaisuose.

„Bluetooth 4" radijo modulis naudoja labai mažai energijos, todėl gali sėkmingai veikti baterijų pagalba.

### 2.6.2. Zigbee

Tai laisvai prieinamas belaidžio ryšio standartas, priklausantis „ZigBee" aljansui , kuris paremtas IEE 802.15.4 standartu. IEE 802.15.4 standartas apima žemesniuosius lygmenis, tuo tarpu aukštesniuosius protokolo lygmenis nustato „Zigbee" standartas.

„ZigBee" protokolu veikiantys belaidžiai prietaisai naudojami namų, protingų sandėlių [25], žemės ūkio [26] valdyme, o taip pat žaliosios energijos [27] gavyboje, medicinoje. Diegėjai ypač pabrėžia šio protokolo ryškiausias teigiamas savybes – mažą kaina ir mažą energijos suvartojimą, kad sąlygoja ir mažą tokių prietaisų aptarnavimo ir diegimo kainą.

Prietaisai šiame tinkle, skirtingai nei „WiFi" tinkluose, patys suformuoja tinklą ir jiems nereikia papildomos stotelės. Kol kas šis protokolas nėra itin palčiai paplitęs, be to norint šiuo protokolu prietaisus "matyti" internete, reikia naudoti tarpines stoteles.

Norint sistemose diegti „Zigbee" protokolą reikia tapti asociacijos nariu (kaina ~2000 eurų metams. Norint įrenginius parduoti, jie turi būti testuojami atestuotose laboratorijose (kaina iki 1000 eurų). Dėl šios priežasties, kuriamoje sistemoje šis protokolas tiesiogiai naudojamas. Bus naudojamas „Hue Hub" tarpinė stotelė „Zigbee" apšvietimo sistemoms, ji veikia „Rest" protokolu.

### 2.6.3. 6lowpan

Belaidis protokolas „6lowpan" (angl. IPv6 enabled low power personal area network) veikia IEEE 802.15.4 standartu, kaip ir „ZigBee" protokolas, tačiau pašalina „ZigBee" protokolo limitą – įrenginiai turi IP adresą [28]. IP įgalinti prietaisai gali lengvai jungtis prie interneto naudodami egzistuojančius protokolus, be tinklų sąsajos (angl. gateway). Šie tinklai įgalina naudoti egzistuojančią tinklo infrastruktūrą. IP pagrįstos technologijos egzistuoja dešimtmečius, yra gerai žinomos, ir pasiteisino pasauliniu mastu, be to IP technologija specifikuota ir atvira, šio standarto dokumentacija prieinama bet kam. Tačiau pasiekti internetą vis tiek turi būti naudojamos tarpinės stotelės. „6lowpan" papildo „ZigBee" protokolą, bet kol kas nėra naudojamas komerciniuose įrenginiuose. Todėl šis protokolas taip pat nebus palaikomas kuriamoje sistemoje.

### 2.6.4. Z-wave

„Z-Wave" tai Europoje populiarus uždaras belaidžio ryšio standartas, priklausantis „Z-Wave" aljansui, prie kurio prisijungę daugiau kaip 160 skirtingų kompanijų. Šis standartas skirtas protingo namo įrenginių valdymui nuotoliniu būdu, naudojamas gyvenamųjų namų bei mažų komercinių pastatų valdymui. Ši technologija naudoja mažo galingumo, mažai energijos

vartojančius belaidžio ryšio modulius, kurios galima naudoti apšvietimo, pramogų sistemų, buitinės technikos ir t.t. valdymui.

Didžiausias leistinas atstumas tarp atskirų įrenginių negali būti didesnis negu 30 metrų. Įrenginių apjungimui į tinklą naudojama „Mesh" technologija. Viename tinkle gali būti iki 232 įrenginių, tačiau gamintojai rekomenduoja viename tinkle naudoti ne daugiau kaip 30-50 įrenginių [29].

Norint padidinti valdomų įrenginių kiekį, juos galima suskirstyti į atskirus tinklus. Duomenų perdavimo greitis svyruoja nuo 9600 bitų/s iki 100 kilobitų/s. „Z-Wave" taip pat palaiko duomenų šifravimą, tačiau jis nėra privalomas. Belaidžio ryšio modulius gamina tik 2 gamintojai. Norint gaminti „Z-Wave" reikalingas brangus licencijavimas. Gamintojas kainos viešai nepateikia.

### 2.6.5. Insteon

„Insteon" kompanijos sukurta technologija, skirta protingo namo įrenginių valdymui nuotoliniu būdu. Ši technologija yra dvejopa – įrenginiai vienu metu gali bendrauti ne tik belaidžiu ryšiu, bet ir per elektros instaliaciją, taip padidinant ryšio tarp įrenginių patikimumą. Didžiausias leistinas atstumas tarp atskirų belaidžių įrenginių negali būti didesnis negu 45 metrai, įrenginių apjungimui į tinklą naudojama „Mesh" technologija. Viename tinkle gali būti iki 16,7 milijono įrenginių. Duomenų perdavimo greitis svyruoja nuo 2880 bitų/s iki 13165 bitų/s, elektros instaliacija – iki 38400 bitų/s belaidžiu ryšiu, taip pat „Insteon" palaiko duomenų šifravimą, tačiau jis nėra privalomas. Taip pat šis standartas yra suderinamas su X-10 standarto įrenginiais. Pagrindinis šios technologijos trūkumas, kad „Insteon" įrenginius ir belaidžius modulius gamina tik viena kompanija – taigi ir šis protokolas nebus palaikomas.

## 2.7. Aparatūrinės įrangos apžvalga

Renkantis kompiuterį, kuriame veiks išmaniojo namo sistema reikėtų atsižvelgi į:

- Kainą;
- populiarumą;
- skaičiavimų pajėgumą;
- sujungimo galimybes;
- suvartojamą energijos kiekį;
- dydį.

Naudojant jau pagamintas aparatūrines sistemas, tokias kaip „Raspberri Pi" ar „Beagle bone", svarbu atsižvelgi į sistemos kainą. Kompiuteris, kuriame veikia sistema, gali sudaryti

iki 50 % visos sistemos kainos. Pasirinkus per daug brangų sprendimą, ir norint tilpti į panašių produktų kainą (200-300$), reikėtų taupyti programinės įrangos sąskaita.

Sistemos populiarumas, taip pat yra labai svarbus. Populiariai sistemai, tokiai kaip „Raspberry Pi", yra sukurta daug programinės įrangos komponentų, plokštė turi išsamią dokumentaciją ir daug kitų informacijos šaltinių – diskusijų lentos ar tinklaraščiai, kuriuose galima rasti įvairių programinės įrangos kūrimo procesui reikalingų žinių ir problemų sprendimo būdų. Be abejo, žinoma aparatūrine įranga būsimi pirkėjai linkę pasitikėti labiau. Populiarių sistemų, kaip „Raspberry Pi", išleidžiamos patobulintos versijos, dažnai tarpusavyje suderinamos, todėl galima lengvai migruoti programinę įranga į pasikeitusią versiją.

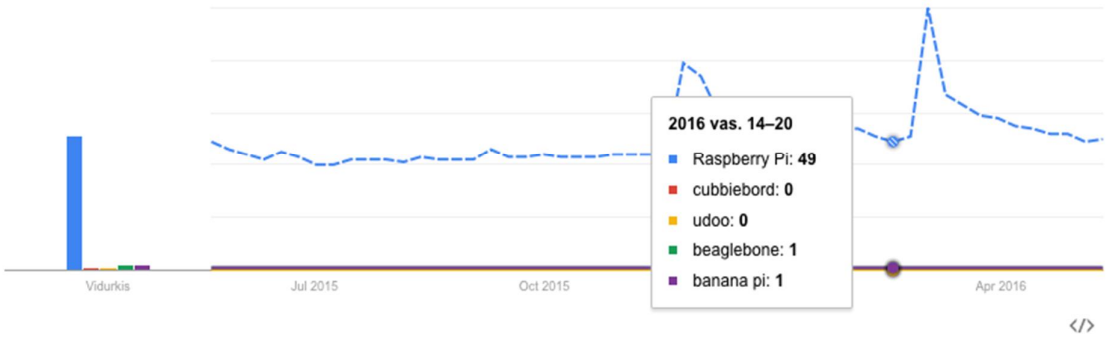
Galingesnės sistemos, kaip „Raspberry Pi", suteikia daugiau lankstumo lyginat su „Arduino". Pavyzdžiui, leidžia naudoti aukštesnio lygio programavimo kalbas – tokias, kaip „Python" ar „Java". Tai savo ruožtu leidžia naudoti vietinius aplikacijų serverius vartotojo sąsajai.

Tinklo, įskaitant belaidį, „Bluetooth", serijinio ryšio, SPI ir I2C, bei GPIO sąsajos suteikia galimybes išmaniojo namo prietaisus sujungti į bendrą tinklą. Kuo šių komunikacijų naudojamame integruotame kompiuteryje palaikoma daugiau, tuo lengvesnis tokių prietaisų diegimas ir palaikymas.

Kompiuterio, į kurį bus diegiama išmaniojo namo sistema, dydis turi didelę reikšmę – mažesnes kompiuterio plokštes lengviau integruoti į komunikacijų spintas išmaniuose namuose. Dažniausiai tokiuose kompiuteriuose naudojamas sistemos luste technologija (angl. System on chip - SoC), todėl jos tampa ne didesnės nei kreditinė kortelė.

Maksimovič [30] konstatuoja, kad „Raspberry Pi" yra geriausias kompiuteris daiktų internetui, vertinat plokštės kainą, sunaudojamą galią, skaičiavimo pajėgumą ir galimybe jungtis su kitomis sistemomis. Belieka pritarti šiai nuomonei, pabrėžiant, kad sistemos resursų užtenka ne tik sukurtai programinei įrangai, bet ir atsiradus naujajai – trečiajai – šio kompiuterio versijai migravimas į pastarąją neįnešė į programų sistemą nei vieno pakeitimo.

Maksimovič be Raspberry Pi aprašė „Arduino", „BeagleBone", „Phidgets" ir „Udoo" plokštes, prieduose papildomai apžvelgta „Cubbieboard", „Banana Pi" ir „pcDuiNe3" kompiuterius. Atmetus „Arduino", kaip visiškai netinkamą aplinką tokios sistemos diegimui, pastebėsime, kad kiti integruoti kompiuteriai pasižymi gana panašiomis techninėmis charakteristikomis.




**2.1 pav.** „Google™ Trends" pateikta ataskaita, kurioje pavaizduota paieškų, kuriose naudoti aprašytų integruotų kompiuterių vardai raktažodžiais, kiekis per pastaruosius metus

Naudodami „Google™ Trends" paieškos tendencijų variklį, buvo palygintas šių kompiuterių populiarumas. Pateiktoje iliustracijoje matyti, kad paieškų „Google" paieškos sistemoje naudojant raktažodį „Raspberry Pi" yra gerokai daugiau, nei kitų minėtų kompiuterių. Tai, kaip minėta anksčiau, turi daug privalumų.

## 2.8. Raspberry Pi apžvalga

2016 gegužę Raspberry Pi fondas yra išleidęs penkis Raspberry Pi modelius: „Raspbery Pi 1 model A+", „Raspberry Pi 1 model B+", „Raspberry Pi 2 model B" ir „Raspberry Pi 3 model B". Toliau apžvelgiami„B" modeliai, nes jie pagal savo išvadų išdėstymą tinka kuriamai sistemai.

Modelis 1B+ turi „Armv6" architektūros vieno branduolio procesorių veikiančiu 700Mgh dažniu. Operatyvioji atmintis – 512Mb. Kompiuterio atmintis – keičiama SD kortelė. Šis kompiuteris turi aštuonis GPIO išvadus.

Modelis 2B turi keturių branduolių „Armv7l" architektūros procesorių veikiančiu 900 Mhz taktiniu dažniu ir, beveik dvigubai daugiau operatyviosios atminties nei ankstesnis modelis. Šis modelis buvo išleistas su daugiau nei dvigubai didesniu GPIO išvadų skaičiumi.

Modelis 3B taip pat turi keturių branduolių „Armv7l" architektūros procesorių, kuris veikia 1200 Mhz dažniu ir turi 1024 Mb operatyviosios atminties. Šis kompiuteris, skirtingai nei ankstesnės versijos, turi belaidžių „WiFI" ir „Bluetooth" ryšių galimybes.

**2.2 lentelė** lentelė Raspberry Pi „B" modelių techninių savybių palyginimas.

| | **Raspberry Pi 3 model B** | **Raspberry Pi 2 model B** | **Raspbery Pi 1 model B+** |
|---|---|---|---|
| *Procesoriaus architektūra* | Armv7l | Armv7l | Armv6l |
| *Pagrindinis lustas* | Broadcom BCM2837 | Broadcom BCM2836 | Broadcom BCM2835 |
| *Procesoriai/ branduoliai* | 4 | 4 | 1 |
| *Operatyvioji atmintis, Mb* | 1024 | 1024 | 512 |
| *Taktinis dažnis, Mhz* | 1200 | 900 | 700 |
| *Tinklo komunikacijos* | 10/100 mb/s tinko adapteris, WiFi, BlueTooth | 10/100 mb/s tinko adapteris | 10/100 mb/s tinko adapteris |
| *Kitos komunikacijos* | SPI, I2C, 17 GPIO | SPI, I2C, 17 GPIO | SPI, I2C, 8 GPIO |
| *USB, kiekis vnt.* | 4 | 4 | 2 |
| *Garso, vaizdo išėjimai* | HDMI, kombinuotas vaizdo (TRRS), bei 3,5 mm garso išėjimas | HDMI, kombinuotas vaizdo (RCA), 3,5 mm garso išėjimas | HDMI, kombinuotas vaizdo (RCA), 3,5 mm garso išėjimas |
| *Atmintis* | microSD | microSD | SD |
| *Dydis, mm* | 85,60 × 56,5 | 85,60 × 56,5 | 85,60 × 56,5 |
| *Išleidimo data* | 2016 vasaris | 2015 vasaris | 2014 liepa |
| *Sunaudojama galia, W* | 1,141 | 1,038 | 0,986 |

Vis kompiuteriai yra vienodo dydžio 85,6 mm x 56,5, bei turi aparatūrinius SPI ir I2C komunikacijos protokolus. Visuose kompiuteriuose integruota 10/100 Mb/s tinklo sąsaja. Visos plokštės turi HDMI ir kombinuotus vaizdo išėjimus. Šių kompiuterių kaina – nepriklausomai nuo modelio – 35 JAV doleriai.

## 2.9. Panašių programinių sistemų apžvalga

Per pastaruosius metus kompiuterio „Raspberri Pi" pagrindu kurtų sistemų atsirado nemažai. Jain su bendraautoriais [31] sukūrė „Python" pagrindu veikiančią sistemą, kuri valdoma naudojant elektroninį paštą. Patel su bendraautoriais [32] taip pat sukūrė „Python" pagrindu sukurta sistema veikiančią Raspberri Pi, kurioje integravo kamerą, drėgmės ir atstumo sensorius.

Visi paminėti projektai, panašu, buvo sukurti daugiau technologijos pademonstravimui, nei naudoti galutiniam vartotojui. Aprašytos sistemos buvo sujungtos laboratorijos sąlygomis naudojant maketines plokštes, tuo tarpu mūsų sukurta sistema yra paruošta diegimui ir jau

įdiegta objektuose. Mūsų sistema naudoja rinkoje esančius įrenginius („Phillips Hue", „ICPconn 7000" serijos įrenginius ir kt.), turi galutiniam vartotojui skirtą *Web* sąsają ir Administravimo klientą, kuriuo galima patogiai įdiegti ir sujungti išmaniųjų namų prietaisus.

## 2.10. Išvados

Analitinėje dalyje aprašyta kodėl išmaniojo namo sistemas reikia kurti ir kuo jos bus naudingos vartotojams. Nagrinėjant literatūrą buvo sudarytos programų sistemos reikalavimų prielaidos, t.y. nuspręsta, koks išmaniojo namo funkcionalumas bus norimas vartotojo. Apžvelgti pagrindiniai laidiniai ir belaidžiai prietaisų bendravimo protokolai naudojami išmaniųjų namų sistemose. Išmaniojo namo programų sistemoje numatytas SPI, I2C, „ModBus" ir „1-Wire" laidinių protokolų palaikymas. Programų sistemoje taip pat bus palaikomas „Bluetooth" protokolas. „Zigbee" protokolas palaikomas per „Philips Hue Hub", kuris įgalina valdyti šiuo protokolu veikiančias apšvietimo sistemas.

Šioje dalyje taip pat buvo aprašyti ir palyginti integruoti kompiuteriai, kuriuose galima diegti išmaniojo namo programų sistemas. Išanalizavus šių kompiuterių specifikacijas, bei remiantis kitų autorių rekomendacijomis išmaniojo namo sistemos diegimui buvo pasirinktas kompiuteris „Raspberry Pi".

# 3. PROJEKTINĖ DALIS

## 3.1. Įžanga

Projektas buvo atliktas pagal tradicinę programų sistemų kūrimo „krioklio proceso" metodiką. Jos esmė yra projekto kūrimas palaipsniui, pradedant nuo reikalavimų sistemai, sistemos architektūros, produkto gamybos (kodavimo) ir užbaigiant testavimu. Šioje dalyje aprašysime esminius šio proceso metu gautus rezultatus. Pateikiami tik esminės sistemos reikalavimų ir architektūros dalys. Išsami sistemos architektūra ir reikalavimai, pagal kuriuos sistema buvo įgyvendinama, pateikta projekto dokumentacijoje.

## 3.2. Sistemos reikalavimai

Sistemos reikalavimai aprašyti remiantis Volere [33] metodika. Skyrių pavadinimo skliausteliuose nurodytas Volere skyriaus numeris.

### 3.2.1. Sistemos paskirtis (1)

Sistema skirta išmaniųjų namų automatizavimui ir namų bei namų daiktų tinko (angl. HAN – home area network) įjungimui į tinklą. Sistema jungia belaidžius prietaisus ir jutiklius į bendrą HAN, kurie sąveikauja tarpusavyje (angl. M2M – machine to machine).

## 3.3. Problema (1a)

Išmaniuose namuose yra daugybė į tinklą galinčių įeiti prietaisų, kurie susijungę tarpusavyje gali pasiūlyti naują funkcionalumą. Programų sistemai šiuos prietaisus reikia įjungti į vieną tinklą ir, esant reikalui, leisti apsikeisti duomenimis su vartotoju ar kitais prietaisais.

### 3.3.1. Tikslai (1b)

Išmaniojo namo sistemos tikslai:

- Palengvinti protingų namų automatizavimo procesą
- Protingų namų instaliaciją padaryti mažiau sudėtingą
- Suteikti vartotojui galimybę pačiam diegti sistemą
- Reguliavimo ir pritaikymo poreikiams funkcijas padaryti paprastesnes
- Sumažinti išlaidas aparatinei įrangai, naudojant atvirą aparatūrinę įrangą (Raspberry Pi)

- Sumažinti bendrąsias namo energijos sąnaudas, naudojant protingą apšvietimą, protingą šildymą ir kondicionavimą, vėdinimą.

### 3.3.2. Personažai (2e)

Personažai yra sistemos vartotojai, kurie vienokiu ar kitokių būdu sąveikauja su išmaniojo namo programų sistema. Išskiriami šie personažai:

- Vartotojas – sistemos naudotojas turintis bendras teises valdyti prietaisus ir matyti informaciją, kurti scenarijus.
- Administratorius – sistemos naudotojas, kuris konfigūruoja protingų namų sistemą, priskiria prietaisus, juos sujungia.
- Sistema – M2M operacijos yra atliekamos ir daug užduočių vykdo pati sistema.

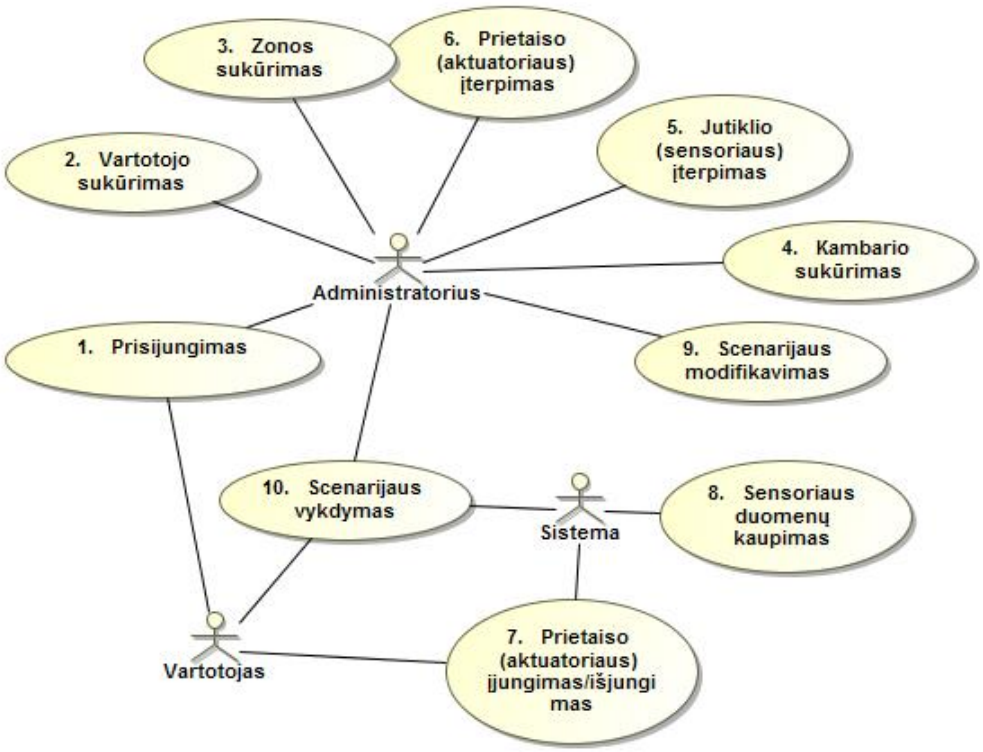
### 3.3.3. Komunikuojančios sistemos (3c)

„Raspberry Pi" išplėtimo plokštė (iliustracija priede) sukurta UAB „Singletonas" yra pagrindinė platforma, kuria integruotas kompiuteris bendrauja su išoriniais įrenginiais. Šioje plokštėje įdiegta „ModBus" sąsaja, „OneWire" sąsaja. Taip pat šioje plokštėje naudojant SPI protokolą įdiegti 16-a išeinančių kontaktų. 4 įeinantys kontaktai prijungti prie „Raspberri Pi" GPIO išvadų. I2C sąsaja, esančia „Raspberry Pi", pajungtas realaus laiko laikrodis, kuris skirtas išsaugoti laiko parametrus po sistemos išjungimo. Šioje plokštėję yra integruotas „Raspberry Pi" energijos tiekimas.

Sistema turėti sąsają su prietaisais pasiekiamais interneto tinklu (pvz. „Phillips Hue Hub"); turi bendrauti su prietaisais prijungtais per „ModBus" sąsają (IcpConn M-7000 serijos įrenginiai); pasiekti „1-Wire" protokolu veikiančių termometrų duomenis; mokėti valdyti „Raspberry Pi" išplėtimo plokštės įėjimus ir išėjimus. Sistema tai pat palaiko bevielį „Bluetooth".

### 3.3.4. Sistemos ribos (8a)

Iliustracijoje pateikiamos sistemos ribos, ankstesniame skyriuje aprašyti personažai ir galimi panaudojimo atvejai.




**3.1 pav.** Išmaniojo namo programų sistemos panaudos atvejų diagrama

### 3.3.5. Sistemos architektūra

Projekto architektūros specifikacijos skyriaus paskirtis – pateikti ir išanalizuoti išmaniojo namo sistemos statinius (panaudojimo atvejus, sistemos išskaidymo į paketus, klasių diagramas) ir dinaminius (sąveikos, būsenų, veiklos diagramas) architektūros modelius.

Projektuojamos išmaniojo namo sistemos architektūra apibrėžiama iš įvairių požiūrio taškų. Kiekvienam požiūriui pavaizduoti naudojamas atskiras modelis. Parenkant sistemos architektūros modelius taikomi tipiniai architektūros šablonai ir unifikuota modeliavimo kalba UML (angl. Unified Modeling Language).

Siekiant surinkti ir pateikti svarbius programų įrangos architektūrinius sprendimus, kuriuos galima panaudoti kuriamoje sistemoje, buvo naudotasi projekto reikalavimų specifikacijoje nurodytais užsakovo reikalavimais sistemai. Siekiant nustatyti protingo namo modelį, išsiaiškinti ir apibrėžti būtinus reikalavimus sistemai, vykdyta prototipų projektavimą ir greitą realizavimą. Išanalizavus architektūros modelius, specifikacijos dokumentas bus naudojamas geriau suprasti architektūrą, pasirinkti tinkamus sprendimus realizuojant ir vystant sistemą.

### 3.3.6. Architektūriniai apribojimai

Sistema veikia „Raspberry Pi" integruotame kompiuteryje. Tai ribotų resursų integruotas kompiuteris. Naudotinoje „Raspberry Pi 2 B+" versijoje yra „Brodacom BCM2836" 32 bitų „Arm v7" procesorius, kurio taktinis dažnis 900Mhz. „Raspberry Pi" turi 1Gb operatyvinės atminties, ir SD kortelių lizdą. Keturis USB išplėtimus, aparatūrines I2C, SPI sąsajas.

Reikalavimai ir apribojimai, kurie turi įtakos sistemos architektūrai: (1) sistema turi būti mobili, veikti žiniatinklio serveryje (2) Sistema neturi reikalauti papildomo diegimo vartotojo kompiuteryje.

## 3.4. Architektūrinis sprendimas

Sistemos kūrimui ir diegimui naudojamas „Apache TomCat® 7" žiniatinklio serveris. Duomenų bazė patalpinta „Oracle MySQL" serveryje. Sistemą sukurta naudojant Java programavimo kalba, todėl veikia skirtingose operacinėse sistemose.

Vartotojo sistema kuriama pagal trijų sluoksnių architektūrą. Pirmasis sluoksnis tai duomenų bazės serveris, antras sluoksnis – WEB aplikacijų serveris, kuris bendrauja su duomenų baze, WEB serveryje atliekamos ir nuolatinės vartotojo prietaisų parametrų užklausos, bei šių prietaisų valdymas, nedalyvaujant vartotojui (M2M). Kliento pusėje (trečias sluoksnis) yra tik interneto naršyklė (pvz., „Internet Explorer"). Jos pagalba vykdomas apsikeitimas duomenimis tarp naudotojų ir sistemos.

Vartojo sąsaja klientui pateikiama kaip vieno puslapio aplikacija (angl. single page application), kurioje vartojo užklausoms perduoti serveriui naudojamas AJAX protokolas. Vartotojo sąsajai įgyvendinti panaudotas „Primefaces" komponentas. Aplikacijų serveris sukurtas naudojant „Java Faces" technologiją. Bendravimui su duomenų baze naudojamas „EclipseLink" JPA komponentas. Aplikacijų serveris su administravimo klientu bendrauja naudodamas „WebService" sąsają.

Komunikacijai su „Zigbee" bevielį ryšį naudojančiomis lempomis naudojamas „PhillipsHue Hub". Šis krašto maršrutizatorius (angl. edge router) komutuoja REST užklausas ir valdo belaidžias lempas.

„ModBus" protokolas įgyvendintas per „Raspberry Pi" nuosekliąją sąsają. UAB „Singletonas" sukurtoje išplėtimo plokštėje šis serijinis signalas aparatūrinės įrangos pagalba verčiamas į RS485 (ModBus) signalą. Išmaniojo namo programų sistema sistema naudoja „Pi4J" komponentą bendrauti su nuosekliąja sąsają. Per JNA (angl. Java native access) naudojamas „libmodbus" testuojant sistemą pasirodė nepakankamai stabilus.

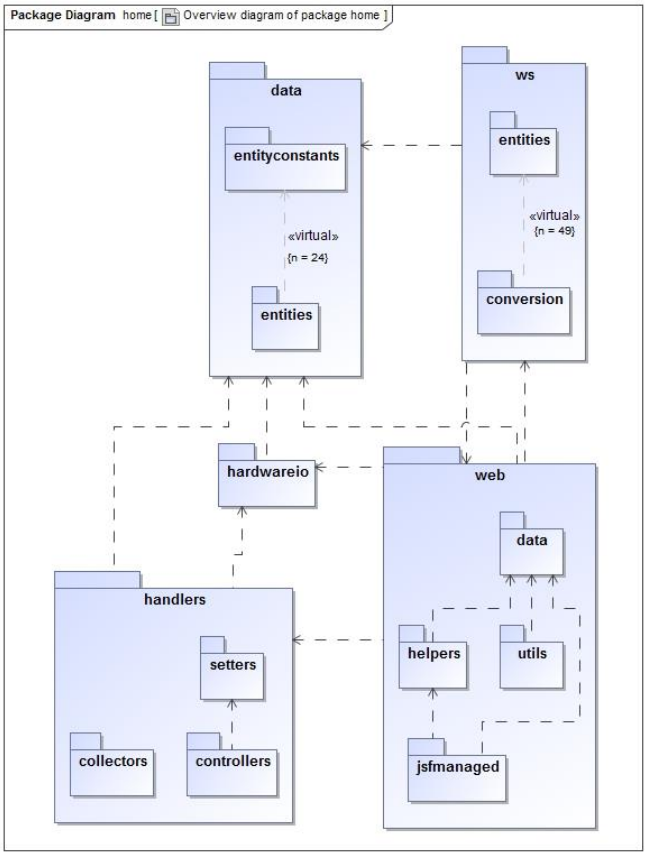
Su minėtos išplėtimo plokštės įėjimais, kurie sujungti su „Raspberri Pi" GPIO išvadais sistema bendrauja per „Pi4J" komponentą. Išėjimams komutuoti naudojamas SPI protokolas, kuris tai pat valdomas naudojant „Pi4J" komponentą. Iš viso naudojama 16 išėjimu, kurie po 8 prijungti prie SPI1 ir SPI2 išvadų "Raspberry Pi" plokštėje.

„1-Wire" pajungti termometrai „owfs" įrankio pagalba susiejami su failine sistema. Šios sistemos failai, kuriuose yra termometrų duomenys, skaitomi JDK įrankiais.

„Bluetooth" prietaisų palaikymas implementuotas naudojant „bluez" įrankius.

## 3.5. Sistemos statinis vaizdas

Šiame skyriuje bus aprašytas statinis sistemos vaizdas. Išmaniojo namo sistemos loginė struktūra sudaryta iš sistemą sudarančių paketų ir juos sudarančių klasių. Sistema išskaidyta į paketus ir subpaketus:




**3.2 pav.** Išmaniojo namo programų sistemos suskirstymo į paketus diagrama

Pakete „data" yra duomenų esybių klases ir klasės skirtos sąveikai su „MySQL" serveriu naudojant JAXB. Visos klasės paveldi DataItemEntity abstrakčią klasę. Kiekviena

klasė duomenų bazeje turi atitinkamą lentelę, kuri yra sukuriama komponento „EclipseLink", pagal esybių klases.

Pakete „ws" yra klasės skirtos bendradarbiavimui su nutolusiu administravimo klientu per WEB naudojant SOAP.

Pakete „handlers" yra pagrindinė išmaniojo namo sistemos logika, prietaisų, ju parametrų ir ryšių klasės. Pakete „handlers" yra trys subpaketai: „collectors", „controllers" ir „setters". Pakete „controllers" yra klasės, kurios atsakingos už veiklos prietaisų ir kontroliuojančių prietaisų tarpusavio veikimą. Pakete „setters" yra klasės kurios pakeičia parametrų reikšmes. Pakete „colectors" yra klasės kurios perduoda sistemai informaciją apie namo aplinką.

Paketas „HardwareIO" skirtas bendravimui su aparatine įranga, prievadais. Pakete „hardwareIO" yra klasės kuriose yra pagalbiniai metodai dirbantys su išoriniais įrenginiais, tai „Bluetooth", Modbus (FN485IO), OneWire, „Raspberry Pi" GPIO išėjimais (PIIO).

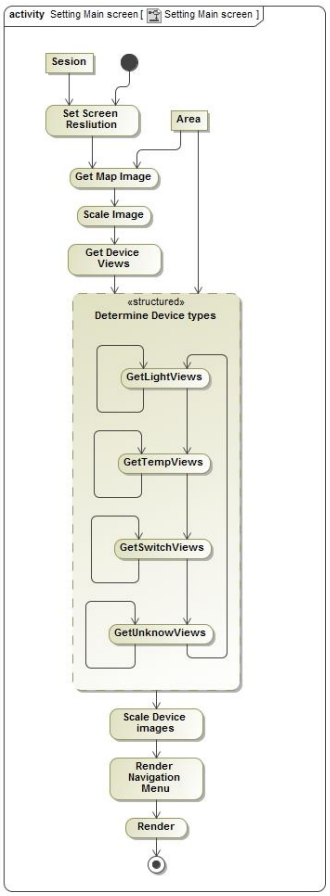
Pakete WEB yra „JavaBeans™" klasės. Šis paketas skirtas vartotojo grafinės sąsajos sukūrimui ir vaizdavimui.

## 3.6. Sistemos dinaminis vaizdas

Šioje dalyje aprašyti pagrindinių arba tyrime naudojamų funkcijų architektūra, siekiama geriau perteikti sistemos funkcionalumą. Detali visos sistemos architektūra pateikta projekto dokumentacijoje.

*3.6.1.1 Pagrindinio puslapio įkrovimas*

WEB naršyklei užklausus pagrindinio puslapio, sukuriama sesija. Iš sesijos duomenų gaunama vartotojo naršyklės rezoliucija, kuri naudojama sumažinti plano vaizdą, gaunami plano duomenys, gaunami plane esantys prietaisai, kurie sugrupuojami į temperaturos, apšvietimo, mygtukų ir kt. tipus, bei atitinkamai atvaizduojami, sukuriamas meniu, ir gautas rezultatas perduodamas naršyklei atvaizduoti naudojant standartinę „Java Faces" biblioteką.




**3.3 pav.** Puslapio generavimo serveryje sekų diagrama

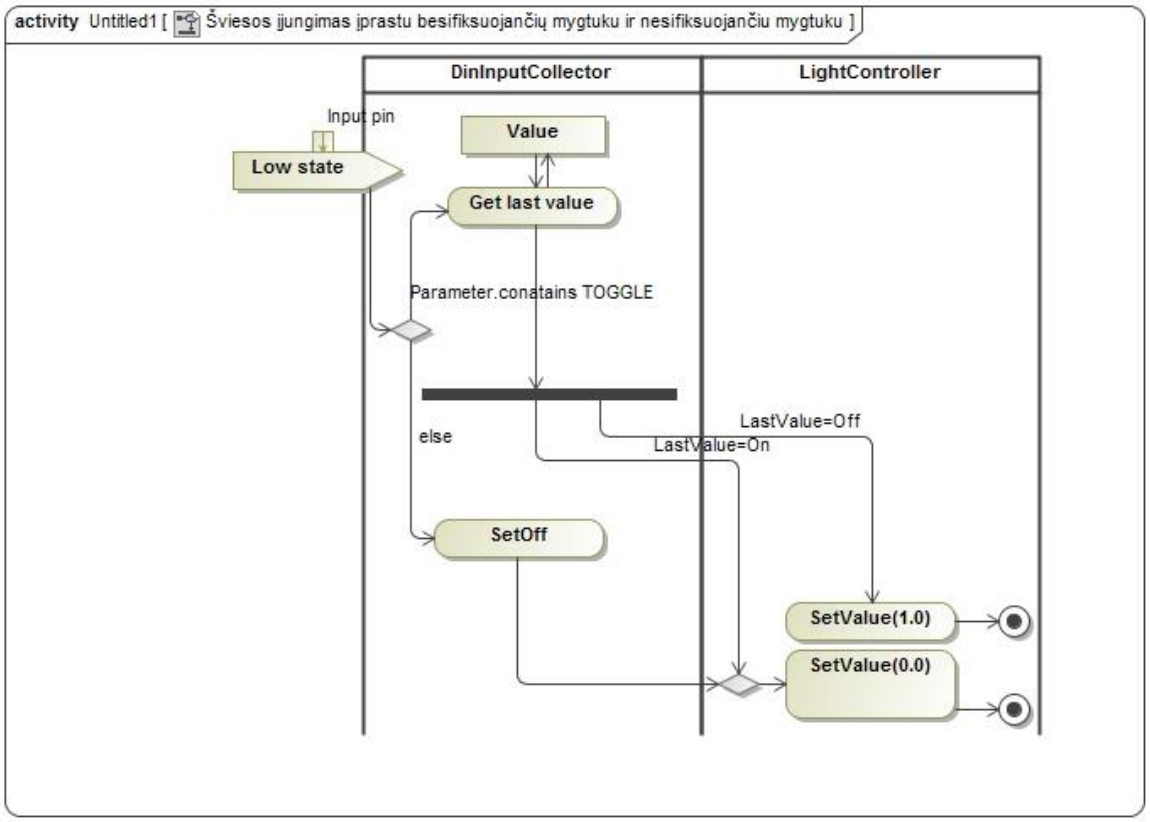
*AreaView* „JavaBean™„ klasė kreipiasi į sesijos kontrolerį, kuris grąžina esamą zoną, jos paveikslą, bei vartotojo naršyklės dydžio parametrus. *AreaEntity* klasėje grąžinami toje zonoje esantys prietaisai, kurio tipą grąžina metodas *getDeviceTypes()*. Pagal šio prietaiso tipą *deviceViewEntity* "Factory" klasė nustato norimo vaizduoti prietaiso tipą, priskiria jam funcijas ir ikonas.

### *3.6.1.2 Prietaiso įjungimas naršyklėje*

*AreaDeviceSwitch* JavaBean™ klasė gauna asinchroninę vartotojo užklausą iš vartotojo naršklės įjungti prietaisą (paspaudžiamas virtualus mygtukas), *AreaDeviceViews* klasė iš klasės *AreaEntity* gauna prietaisų objektus, kurių parametrus norima pakeisti, pagal prietaiso ID kurį perduoda *AreaDevicesSwitch* klasė. Tada gaunami prietaiso parametrai, kuriame nurodyta į kokį „kolektorių" kreiptis. „Kolektoriui" perduodama žinutė apie pasikeitusį parametrą, ir kolektorius aktyvuoja „seterį", kuris įjungia arba išungia lemputę –

šiuo atveju *HueLampOnOffSetter*. Analogiškai ir su kitų paramtrų pakeitimu, šviesos spalvos ar intensyvumo.

### *3.6.1.3 Prietaiso išjungimas jungikliu*




**3.4 pav.** Prietaiso įjungimo sekų diagrama

Iliustracijoje pateikta veiksmų diagrama, parodo, kokia įvykių seka programų sistemoje vyksta po fizinio mygtuko paspaudimo, prijungto prie vieno iš „Raspberry Pi" įėjimų. „Kolektorius" *DinInputCollector* sužadinamas, kai aktyvuojamas mygtuko kontaktas. Jeigu prietaiso parametras yra SWITCH_TOGGLE, t.y. mygtukas nėra fiksuojamas, o įjungiamas vieną kartą paspaudus, išjungiamas antrą kartą paspaudus, užklausiama paskutinė lemputės būsena. Jei paskutinė būsena yra išjungta, „seterio" metodui perduodama reikšmė 1.0, jei buvo įjungtas perduodama reikšmė 0.0.

Kitu atveju jei prijungtas mygtukas fiksuojamas, t.y. parametras SWITCH_ON_OFF, paspaudus mygtuką įtampa įėjime nukrenta ir kontroleris seteriui perduoda funcijai SetValue perduoda argumentą 0.0.

#### *3.6.1.4 Plano įkrovimas administravimo kliente*

Administravimo klientas paleidimo metu kreipiasi į „WebService" metodą *loadMapsIds*, naudodamas įprastinį objektų pasiekimo protokolą (angl. SOAP simple object access protocol). Sugeneruojama standartinė *LoadMapsIdsReaquest* klasė, kuri nuoseklinama į XML dokumentą. Pastarasis persiunčiamas interneto protokolu „Apache Tomcat®" veikiančiai „Raspberri Pi" išmaniojo namo aplikacijai. Čia XML dokumentas apdorojamas ir iškviečiamas metodas *loadMapsIds*, kuris kreipiasi į duomenų bazę bei grąžina visus duomenų bazėje esančius planų unikalius numerius XML dokumentu. XML dokumentas aplikacijoje atstatomas į „List" objektą. Administravimo klientas visiems gautiems unikaliems planų numeriams kviečia „WebService" metodą *loadMap(Long id)* – generuojama klasė *LoadMapRequest*, kuri nuoseklinama į XML dokumentą ir interneto protokolu persiunčiama serveriui, kur ši užklausa iškviečia metodą *loadMap*. Serverio metodas *loadMap(Long id)* kreipiasi į „MySQL" duomenų bazę, kur iš atitinkamų lentelių yra sugeneruojamos esybių klasių objektai. Šie objektai konvertuojami į „WebService" esybių objektus naudojat pagalbinius metodus klasėje *WSEntityMerger.* Pastarieji serializuojami į XML dokumentą, kuris interneto protokolu grąžinamas klientui.

## 3.7. Sistemos testavimas

Sistemos testavimas buvo vienas išmaniojo namo kūrimo ciklo etapų. Testavimo metu buvo bandoma surasti klaidas, taip užtikrinat programų sistemos kokybę ir tinkamumą galutiniams vartotojams.

### 3.7.1. Vienetų testavimas

Pagal testavimo aplinką testuojami komponentai buvo atskirti į dvi dalis:

- Kodas kuris gali būti testuojamas „Eclipse" aplinkoje
- Kodas kuris gali būti testuojamas, tik „Raspberry Pi"

Klasės ir komponentai, kurie skirti dirbti su aparatūrine įrangą, negali būti testuojami aplinkoje, kurioje koduojama, taigi, svarbu šiuos modulius testuoti natyvioje aplinkoje. Klasės pakete *HardwareIOManagers* (žr. Detalios architektūros specifikaciją) gali būti testuojamos tik „Raspberry Pi" prijungus atitinkamą įrangą, su kuria klasė skirta dirbti.

Testavimui palengvinti sukurtas įrankio „ant" scenarijus, kuris SSH protokolu kopijuoja testus ir juos paleidžia kompiuteryje „Raspberry Pi".

Funkciniai vienetai – vienetas, kuris suteikia tam tikra funkciją prietaisams ar jų grupėms. Sistemoje įgyvendinta kaip *setter*, *controller* ir *collector* klasių sąveika. Šie elementai

atitinka kontroliuojamų ir kontroliuojančių prietaisų veiklą: pvz., po mygtuko paspaudimo įsijungia apšvietimo lempos.

Šių klasių sąveikos testuojamos testuose sukuriant virtualius prietaisų tipus bei priskiriant jiems parametrus, kurie valdomi atitinkamų „seterių" ar „kolektorių".

Testo metu tikrinama:

- Ar veikia pagal numatytą specifikaciją;
- Ar logika veikia korektiškai;
- Ar „seteriai" atlieka aparatūros operaciją;
- Ar „kolektorius" gauna duomenis iš aparatūros;

Kiekvienas „seteris", kolektorius ar kontroleris buvo ištestuotas bent viename funkcinių vienetų „Unit" teste.

### 3.7.2. Integravimo testavimas

Sistemą sudaro pagrindinis ir administravimo moduliai, kurie bendrauja per „WebService" naudojant XML. Daroma prielaida, kad JAXB XML (angl. parser) analizatorius-keitiklis veikia tinkamai. Buvo svarbu įsitikinti, kad ši sąsaja neturi trūkumų:

- visi *Web* metodai veikia, kaip aprašyta detalioje architektūros specifikacijoje pagrindiniame modulyje;
- visi *Web* metodai pasiekiami administravimo kliente;
- visos *Web* esybių klasės atitinka klases, kurios naudojamos, atitinka esybių klases naudojamas JPA.

### 3.7.3. Patikimumo testavimas

Protingo namo sistema turi veikti nepertraukiamai, todėl patikimumo testavimui buvo skirtas dideli dėmesys. Protingo namo sistema turi veikti brandžiai, t.y. PĮ gedimo dažnumas, t.y. trikis ar kitas veikimas negali atsirasti dažniau nei kartą per mėnesį.

Testuojant protingo namo sistemą buvo tikrinamas atsparumas defektams. Iškeltas tikslas, kad programinė įranga nesustos sutrikus neesminių komponentų veikimui: ryšio su prietaisu nubuvimui; esant ryšio su vartotojo sąsaja arba administravimo klientu klaidai; duomenų bazės ar „WebService" klaidos atveju.

Sistemos atgautinumas (angl. recoverability) buvo vienas iš testuojamų etapų. Buvo įsitikinta, kad programa geba grįžti prie pradinės būsenos, esant ryšio su prietaisu dingimui, arba laikiniems trikdžiams; programa geba grįžti prie pradinės būsenos esant ryšio trikdžiams

tarp vartotojo naršyklės arba administravimo kliento; programa atkuria veikimą esant „PeriodicUpdater", kuri skirta nuolatiniam jutiklių atnaujinimui vykdyti, gijos (angl. thread) sutrikimui.

## 3.8. Išvada

Projektinėje dalyje aprašyti išmaniojo namo sistemos reikalavimai, kuriais buvo grindžiama projekto realizacija. Sistema realizuota „Apache Tomcat®" aplikacijų serveryje programavimo kalba „Java" panaudojant JPA, „Java Faces" technologijas. Vartotojo sąsaja sukurta vieno puslapio principu (angl. single page aplication), bei naudoja asinchronines „JavaScript" ir XML (AJAX) serverio užklausas, naudodama komponentą „Primefaces". Administravimo sąsaja bendrauja su administravimo klientu naudodama „WebService" užklausas per SOAP.

Programų sistemos kokybei užtikrinti programų sistema buvo testuojama naudojant vienetų testavimą programavimo aplinkoje, įrenginių klasės ir metodai buvo testuojami diegimo aplinkoje. Didelis dėmesys buvo skirtas sistemos atgautinumo ir atsparumo testavimui, bei klasėms, kurios skirtos valdyti prietaisus.

Siekiant patenkinti vartotojų reikalavimus buvo sukurta sudėtinga programų sistema, kuriai veikti reikalingi pakankamai dideli aparatūriniai diegimo aplinkos resursai, todėl nuspręsta ištirti šios programinės įrangos greitaveiką kompiuterio „Raspberry Pi" versijose. Tai pat nustatyti kokie aparatūrinės įrangos parametrai labiausiai veikia programų sistemos spartą.

# 4. EKSPERIMENTINĖ DALIS

## 4.1. Įvadas

Sistemos, veikiančios ribotų resursų kompiuteriuose, arba integruotuose kompiuteriuose, nepriklausomai nuo jų galimybių, turi veikti vartotojui priimtinu greičiu. Sunku pasakyti, kiek įprastinis vartotojas yra linkęs laukti puslapio (programos) atsidarymo. Kai kurie autoriai [34] teigia, kad iki 1,2 sekundės greitis yra priimtinas visoms svetainėms, tačiau kiti teigia, kad 100ms puslapio pagreitėjimas atneša 1% daugiau pajamų, o 400ms pagreitėjimas susijęs su 12% didesniu srautu [35]. Šie tyrimai atlikti internetinėse svetainėse. Nėra aišku, kiek protingų namų vartotojai nusiteikę laukti kol atsidarys išmaniojo namo valdymo programa (puslapis), tačiau, bent jau architektūriniu požiūriu – kuo greičiau tuo geriau. Visgi nauja programų sistema neturėtų pabloginti esamos padėties, nors ir siūlo papildomą funkcionalumą, t.y. protingų namų vartotojas nori, kad paspaudus mygtuką šviesa įsijungtų nedelsiant. O sistemos valdymo puslapio (programos) atsidarymo sulauktų kiek įmanoma greičiau.

Sukurta sistema įdiegta į „Raspberry Pi" kompiuterį, kuris neturi našių aparatūrinių resursų, todėl tikėtina, kad programos greitaveika nebus labai gera, jei nebus imtasi tam tikrų optimizacijos veiksmų.

Išmaniojo namo programos kūrimo proceso metu, mažiau nei dviejų metų laikotarpyje, „Raspberry Pi" išleido dar dvi patobulintas integruoto kompiuterio versijas. Šios versijos yra tarpusavyje suderinamos[1] tiek operacinės sistemos, tiek aparatūrinės periferijos požiūriu. Tačiau buvo pastebėta, kad „Raspberry Pi" antrojoje (2 model B) versijoje, kurta išmaniojo namo programų sistema veikia nepalyginamai greičiau nei pirmojoje kompiuterio versijoje. Be „Raspberry Pi" egzistuoja daugybė integruotų kompiuterių (daugiau žr. sk. Apžvalga), kurie skiriasi tiek savo parametrais, tiek kaina. Be to, kai kurias dalis galima patobulinti (pvz. atminties kortelė).

Be abejo, panašaus sudėtingumo programų sistemas galima optimizuoti, keičiant operacinės sistemos parametrus ar pačios programų sistemos parametrus, ar sprendimo architektūrą. Tačiau tyrimo metu buvo orientuotasi į aparatūrinius parametrus ir VM.

[1] Su išimtimis, gamintojas negruntuoja visiško sistemos suderinamumo

## 4.2. Tikslas

Ištirti sukurtos programos greitaveiką, siekiant nustatyti kokie pagrindiniai aparatūriniai faktoriai lemia sistemos atsako greitį galutiniam vartotojui siekiant padėti sistemų architektams pasirinkti diegimo aplinką.

## 4.3. Uždaviniai

Tyrimui yra keliami šie uždaviniai:

1. Įvertinti esamos programų sistemos spartą: nustatyti kiek laiko trunka puslapio atvaizdavimas vartotojui, ir kas sudaro šį vartotojo laukimo laiką; nustatyti kiek laiko trunka plano saugojimo operacija;
2. Nustatyti lygiagrečių procesų vykdymo daugiabranduoliusoe procesoriuose įtaką sistemos greitaveikai
3. Nustatyti operatyvios atminties greičio įtaką programų sistemos greitaveikai.
4. Įvertinti skirtingų greičių SD atminties kortelių įtaką sistemos greičiui.
5. Įvertinti VM implementacijos įtaką sukurtos programų sistemos greičiui

## 4.4. Tyrimo metodai ir įrankiai

Siekiant atsakyti į klausimus, kuriuos kelia tyrimo suformuoti uždaviniai, pirmiausiai reikia apibrėžti tyrimo eigą ir jos griežtai laikytis, tam kad tyrimas būtų validus, o gauti rezultatai būtų patikimi.

### 4.4.1. Tyrimo aplinka

Programų sistemos veikimo greitį įtakoja daugybė faktorių. Objektyviai nustatyti programos veikimo greitį reikia suvienodinti tyrimo sąlygas. Tyrimui naudojama ta pati programų sistemos versija (kompiliavimo numeris 367), veikianti „Apache Tomcat®" aplikacijų serveryje (angl. application server), paleista „Oracle Java JDK" aplinkoje (versija 1.8.0_65) su „Java HotSpot™ Client VM" (build 1.8.0_65-b17, interpreted mode), veikiančia „Raspbian GNU/Linux 8.0 (jessie)", „Linux" branduolio versija 4.4.7+. Virtualioje mašinoje veikė Java agentas „Zorka". Taip pat įdiegta „MySql®" duomenų bazė (Versija 14.14), kurią naudoja sukurta išmaniojo namo sistema.

Išmaniojo namo programų sistema veikia naudojant planą, kuriame yra keturios zonos: II aukštas, rekuperatoriaus patalpa, įvykių demonstracija, šildymas.

Ši instaliacija buvo replikuota į visas naudojamas SD atminties korteles naudojant komandą "*dd bs=16m of=piHomeWZorkaYessie.img if=/dev/rdisk2"* MacOS aplinkoje, ir

nukopijuota į visas naudojamas atminties korteles komanda "*dd bs=16m of=piHomeWZorkaYessie.img if=/dev/rdisk2"* [36].

Tyrimas atliekamas „Raspberry Pi" integruotuose kompiuteriuose, t.y. versijose „1B", „2B" ir „3B". Buvo naudojama ta pati tinklo infrastruktūra.

## 4.5. Atliekami matavimai

Siekiant išsiaiškinti kokie aparatūriniai parametrai lemia sistemos greitaveiką galutiniam vartotojui buvo matuotas programos įkrovimo laikas, plano įkrovimo greitis

### 4.5.1. Programos įkrovimo laikas

Siekiant išsiaiškinti programų sistemos greitaveiką buvo matuojamas programos (puslapio) atidarymo greitis viename įprastiniame namų kompiuteryje (Lenovo X230, i5 Intel® procesorius, 8gb RAM) naudojant „GoogleChrome" naršyklę (versija 50.0.2661.94). Šiame kompiuteryje taip pat veikė Java agento „Zorka" duomenų rinkimo programa „Zico" (versija 1.0.15).

Serverio apdorojimo laikas (SAL). Serverio laiku laikysime įrankio Zorka pateikiamą HTTP reikšmę, kuri gaunama instrumentuojant „Apache Tomcat®" klasės *org.apache.catalina.core.StandardEngineValve* metodą *invoke(),* įrankio „Zorka" pagalba. Šis laikas yra tas, per kurį apdorojama kliento užklausa (angl. request) ir suformuojamas atsakymas (angl. response). Į šį laiką įeina visos puslapį krovimo metu suformuojamos SQL užklausos, prietaisų vaizdavimo generavimas ir kiti[2] veiksmai.

Pagrindinio išteklio įkrovimas (PIĮL). Šis laikas apima SAL laiką ir laiką per kurį jis persiunčiamas vartotojui.

Puslapio pilnas įkrovimo laikas. Buvo matuojamas puslapio pilnas puslapio įkrovos laikas (PPL), kuris pateikiamas Chrome naršyklės Develooper Tools skiltyje Network, parametro vardas "Load". Šis laikas susideda iš SAL ir PIĮL laiko, laiko reikalingo visų išteklių (angl. resource) įkrvimui bei atvaizdavimui vartotojo naršyklėje.

### 4.5.2. Plano įkrovimo laikas

Siekiant išsiaiškinti programos greitaveiką buvo matuojamas viso plano pakrovimo laikas administravimo programoje. Šis matavimas parodo, kaip greitai visas administratoriaus sukurtas žemėlapis yra įkraunamas administravimo programoje. Šį laiką sudaro tinklo

[2] Detaliau skyriuje „Sistemos dinaminis vaizdas"

užklausos serveriui išsiuntimas, laikas, kurio reikia bendravimui tinkle, XML užklausos nagrinėjimo laikas, žemėlapio išrinkimo iš duomenų bazės laikas, XML atsakymo sudarymo laikas, atsakymo suformavimo laikas bei laikas, kurio reikia žemėlapiui atvaizduoti administravimo kliente. Išskirsime tris pagrindinius laiko momentus, norint detaliau panagrinėti programų sistemą ir našumą integruotame kompiuteryje.

Bendras SQL duomenų bazės užklausos laikas (SQLL). Tai suminis laikas reikalingas užkrauti žemėlapio duomenis iš duomenų bazės. Šis laikas taip pat apima laiką, kuris reikalingas užklausos atsakymui nuoseklinti (angl. serialize) į Java objektus naudojant JAXB.

Metodo *loadMap()* laikas. *loadMap* tai metodas su antoacija *javax.ws.Webservice*: pakete *com.slabs.pi.home.ws*, klasėje *HomeServiceImpl*. Tai *WebService* metodas, kuris grąžina *WsMap* objektą, t.y. visą planą, su jame esančiomis zonomis, prietaisais ir kt. Į šio metodo laiką įeina SQLL.

Plano gavimo laikas (PGL). Tai laikas, reikalingas suformuoti užklausą ir atsiųsti išmaniojo namo planą į administravimo klientą. Šis laikas yra laikas, reikalingas aplikacijų serveriui priimti ir apdoroti *WebService* užklausą bei atsiųsti atsakymą administravimo klientui. Matuojamas įrankio "Zorka" pagalba kaip *org.apache.catalina.core.StandardEngineValve* metodo *invoke()* laikas.

Plano įkrovimo laikas (PĮL). Bendras laikas reikalingas administravimo kliente atvaizduoti žemėlapį. Tai yra nutolusio *WebService* metodo iškvietimas, duomenų nuoseklinimo laikas bei atvaizdavimo administravimo programoje laikas.

### 4.5.3. SD atminties kortelės greitis

Siekiant išsiaiškinti atminties kortelės veikimo laiką, reikia patikrinti, koks yra tam tikros atminties kortelės greitis. SD (angl. Secure digital) atminties kortelės pasižymi skirtingu maksimaliu rašymo ir skaitymo greičiu. SD kortelės yra pagrindinis kompiuterio atminties elementas. SD kortelės šiame kompiuteryje yra keičiamos, todėl nagrinėjant programų sistemos greitį, reikia patikrinti kokiu greičiu jos gali veikti.

Bendras skaitymo greitis kompiuteryje. Tai greitis reikalingas įrašyti SD atminties kortelės atvaizdą į kompiuterio kietąjį diską.

Bendras rašymo greitis kompiuteryje. Tai greitis reikalingas duomenimis užpildyti kortelės atminties elementus informacija. . Šis greitis buvo matuojamas perkeliant disko atvaizdą (angl. disk image) iš kompiuterio kietojo disko į atminties kortelę. Įrašymui buvo naudojamas kompiuteris MAC mini (vėlyvas 2014 modelis). Kadangi SD atminties kortelės įrašymo/skaitymo greitis yra daug mažesnis nei skaitymas/rašymas iš kietojo disko

kompiuteryje, ar procesoriaus apdorojimo galimybės į šiuos faktorius matuojant atsižvelgta nebuvo.

Užpildymo “0” greitis toliau minimas kaip rašymo greitis „Raspberry Pi”. Buvo matuojamas greitis, užpildant laikiną failą 419Mb nulių greitis kompiuteryje „Raspberry Pi” vykdant komandą *dd if=/dev/zero of=/drive/output bs=8k count=50k conv=fsync*.

Skaitymo greitis „Raspberry Pi” buvo matuojamas kopijuojant sukurtą failą */drive/output* į operatyviąją atmintį. Operatyviosios atminties dalis buvo susietas su */mnt/test1* komanda *mount -tmpfs /mnt/test1 /mnt/test1*, ir į ja perkeltas minėtas failas komanda *dd if=/drive/output of=/mnt/test1/temp*. Laikas ir greitis, per kurį sistema perkėlė failą, buvo programos *dd* atspausdintas konsolėje programos baigimo metu.

### 4.5.4. Įkrovimo laikas

Buvo matuojamas aplikacijų serverio „Apache Tomcat®” įkrovimo laikas. Šis laikas buvo gaunamas iš aplikacijų serverio žurnalų failo (angl. log files) “cathalina.out” eilutės “INFO: Server startup in XXms” kur XX nurodytas laikas, reikalingas įkrauti aplikacijų serverį „Tomcat®” su išmaniojo namo programų sistema.

Kadangi aplikacijų serveris „Tomcat®” yra kraunamas sistemos įkrovimo metu, kaip sistema yra ir taip stipriai apkraunama, šis laikas buvo matuojamas ir, sistemai pilnai įsikrovus, paleidžiant programą iš naujo komanda */etc/init.d/tomcar restart.*

### 4.5.5. Operatyviosios atminties greitis

Norint gauti operatyviosios atminties greitį Linux operacinėje sistemoje buvo sukurtas laikinas diskas operatyviojoje atmintyje naudojant komandą *mount -t tmpfs /mnt/test1 /mnt/test1*. Komanda tokiais parametrais sukuria pusės operatyviosios atminties talpos virtualų diską operatyviojoje atmintyje ir priskiria jam vardą failų sistemoje */mnt/test1*.

Sukūrus diską operatyviojoje atmintyje jis užpildomais 0 reikšme komanda *dd if=/dev/zero of=/mnt/test1/test bs=1M.* Programos pateikiamas greitis laikomas operatyviosios atminties greičiu.

### 4.5.6. Įrankiai

Puslapio įkrovimui matuoti buvo naudojama „Google™” sukurta naršyklė „Chrome” versija 50.0.2661.94. Ši naršyklė turi įrankius “Developer Tools”, kurių pagalba galima matyti, kiek laiko trunka įkrauti puslapi, įskaitant atskirų puslapio resursų, įkrovimo greitį.

Įrankis „Zorka” (versija 1.015) , buvo įdiegtas kompiuteriuose „Raspberry Pi”. Šio įrankio paskirtis – instrumentuoti „Java” VM veikiantį programos kodą, pagal iš anksto aprašytus programos scenarijus. Šis įrankis leidžia stebėti ir matuoti VM veikiantį kodą jo nekeičiant. Šio įrankio pagalba buvo matuoti „Apache Tomcat®” serveryje veikiančios išmaniojo namo programų sistemos tam tikrų metodų vykdymo laikai.

Įrankiu „IBM SPSS” analizavome kai kurių gautų duomenų statistinius rodiklius.

### 4.5.7. Taikomi statistiniai metodai

Spartos matavimai gali būti sąlygojami įvairių faktorių. Kiekvienas matavimas atliekamas bent tris kartus, ir rezultatas pateikiamas kaip šių rezultatų vidurki. Siekiant palyginti vidurkius, kurių skirtumai nebuvo akivaizdūs, naudojame statistinę analizę. Patvirtinti iškeltas hipotezes naudojomės neparametriniais kriterijais, kurie rodo, kad matuojamos imtys yra homogeniškos. Patikrinti hipotezės buvo naudojamas „Kruskal-Wals” nepriklausomų imčių testas.

## 4.6. Rezultatai

### 4.6.1. Išmaniojo namo sistemos veikimas skirtingose RPi versijose

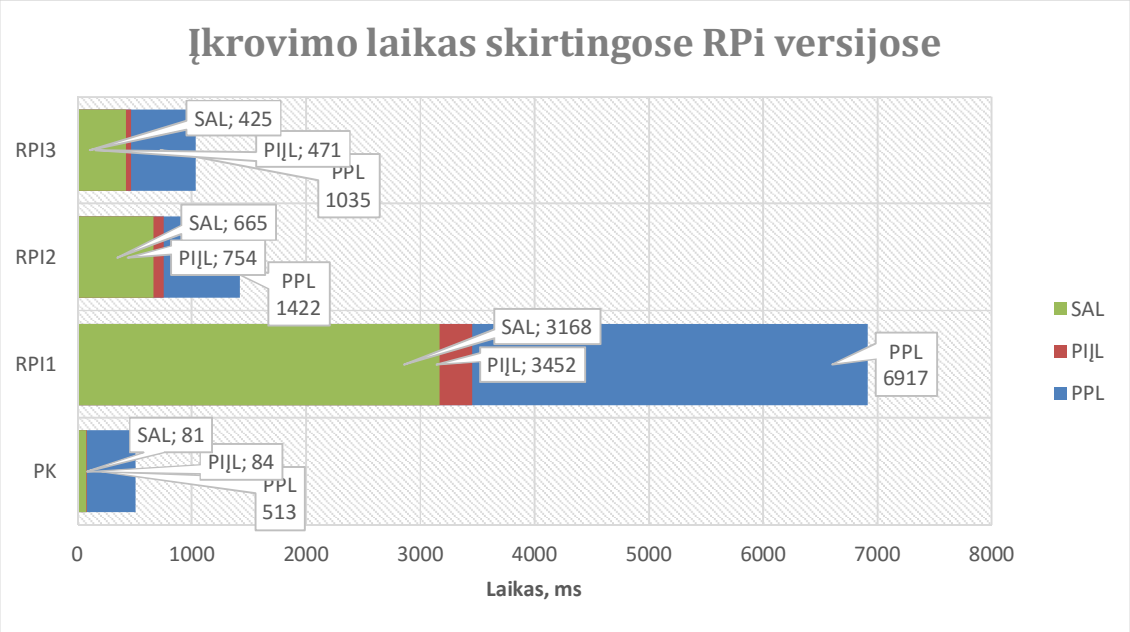
Kaip ir buvo manyta, naujesnėse kompiuterio versijose, pagrindinės zonos įkrovimas vartotojui buvo greitesnis.

„Raspberry Pi” versijoje 1B zonos įkrovimas vidutiniškai truko 6,9 sekundes, tačiau modeliuose 2B ir 3B, šis laikas buvo atitinkamai 1,4 s ir 1,0 sekundės.

Pirmojoje kompiuterio versijoje vidutinis serverio apdorojimo laikas pagrindiniam puslapiui truko 3,2 s, naršyklei šis išteklius (angl. resource) buvo pateiktas po 3,4 sekundžių, o visi ištekliai galutinai atsiųsti bei įkrautas puslapis tik po 6,9 sekundžių

Antrojoje kompiuterio versijoje programų sistemos greitaveika buvo žymiai geresnė, ir vartotojo reikalautą pagrindinės zonos puslapį vidutiniškai naršyklėje galima išvysti po 1,4sekundės. Tai yra beveik 5 kartus greičiau nei pirmojoje versijoje. Atitinkamai sumažėjo ir laikas reikalingas įkrauti pagrindinį išteklį iki 754 ms, bei laikas reikalingas serveriui apdoroti pagrindinį puslapį iki 665ms.

Trečiojoje kompiuterio versijoje (3B) matuojami puslapio įkrovimo laikai taip pat pagerėjo, nors ir ne taip žymiai kaip pereinant nuo pirmosios į antrąją versijas. Pagrindinį puslapį vartotojas gali išvysti po 1,0 sekundės. Pagrindinį išteklį užtrunka atsiųsti 471ms, o jį apdoroti aplikacijų serveryje tik 425 ms.

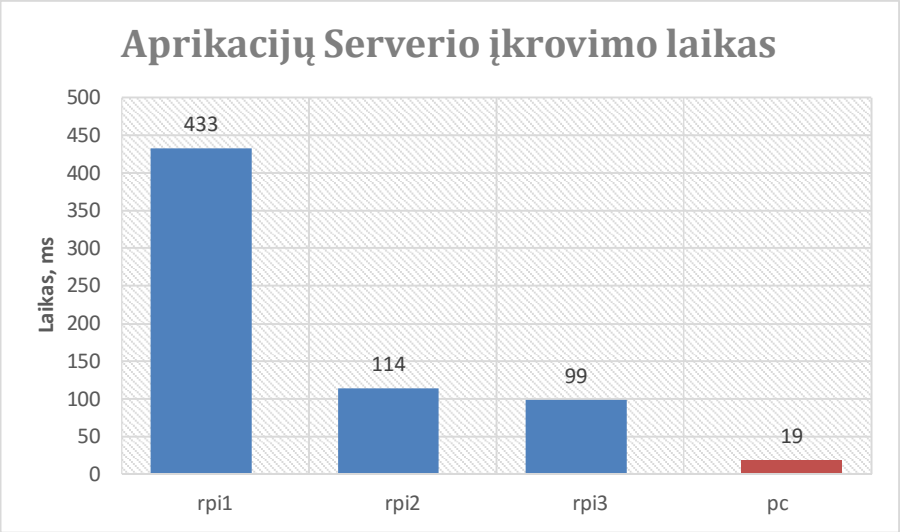



**4.1 pav.** Sistemos pagrindinio vartotojo lango įkrovimo laido diagrama skirtingose kompiuterio „Raspberry Pi" versijose

Čia ir kitur, kur matuojamas skirtingų operacijų laikas, naudosime Horizontalias stulpelines diagramas, kur X ašyje atidėtas laikas milisekundėmis. Ašis prasideda 0 reikšme ir tai yra tam tikros operacijos pradžia, šiuo atveju puslapio reikalavimas arba atnaujinimas naršyklėje. Y ašyje – tiriama grupė, šiuo atveju – kompiuterio „Raspberry Pi" versijos. Stulpeliuose vaizduojamas tam tikros operacijos laikas, pavyzdžiui, šiuo atveju žalia spalva pažymėtas Serverio apdorojimo laikas (SAL), t.y. laikas nuo vartotojo reikalavimo pradžios iki atsakymo (angl. response) suformavimo serveryje. Raudona spalva pažymėtas pagrindinio išteklio įkrovimo laikas, šį stulpelį dalinai dengia SAL laikas. Atvaizduojama tik skirtumo dalis, tokiu būdu galima matyti, kiek ši operacija truko atskirai, t.y laikas, per kurį suformuojama užklausa ir ji gaunama, be serverio apdorojimo laiko. Tai nėra tikslus atvaizdavimas laiko bėgyje, nes dalis laiko yra skiriamas užklausos suformavimui po vartotojo užklausos pateikimo (mygtuko paspaudimo). Veiksmo, t.y. puslapio reikalavimas, įvyksta prieš tai, kai pateikiama užklausa serveriui. Dalis laiko užtrunka, kol serverio puslapis yra atsiunčiamas vartotojo naršyklei. Taigi jeigu vaizduotume tiksliai laiko skalėje, dalis raudono stulpelio būtų prieš žaliąjį. Tačiau toks vaizdavimas vistiek leidžia susidaryti aiškiam puslapio įkrovimo laiko vaizdui, įskaitant tinklo operacijų laiką. Mėlyna spalva pažymėtas puslapio įkrovimo laikas, t.y. visas laikas reikalingas puslapiui atvaizduoti. Skirtumas kuris nėra dengiamas SAL ir PIĮL laikų, rodo kiek laiko trunka kitų resursų (pvz. paveikslėlių) atsiuntimas, bei atvaizdavimas kliento naršyklėje. Turint omenyje, kad puslapio atvaizdavimo

laikas yra konstanta, nes naudojamas tas pats kompiuteris ir ta pati interneto naršyklė visame tyrime, ši, neuždengta dalis mėlynojo stulpelio dalis, parodo, kiek laiko trunka iš integruoto kompiuterio „Raspberry Pi" atsiųsti visus resursus – t.y. paveikslėlius, pakopinių stilių šablonų rinkmenas (css), JavaScript failus ir kt.

Skirtingose kompiuterio „Raspberry Pi" versijoje taip pat skiriasi ir laikas kurio reikia, kad butų užkrautas aplikacijų serveris „Apache Tomcat®". „Raspberry Pi 1B" šis laikas vidutiniškai yra 7 minutės ir 10 sekundžių, tuo tarpu Rpi antrojoje versijoje (2b) aplikacijų serveris įkraunamas nepalyginamai greičiau – per 110,4 sekundes. Tai yra beveik keturis kartus greičiau. Trečiojoje versijoje šis laikas yra 98,8 sekundės. Palyginimui įprastiniame kompiuteryje, tam reikia tik 18,6 sekundės.

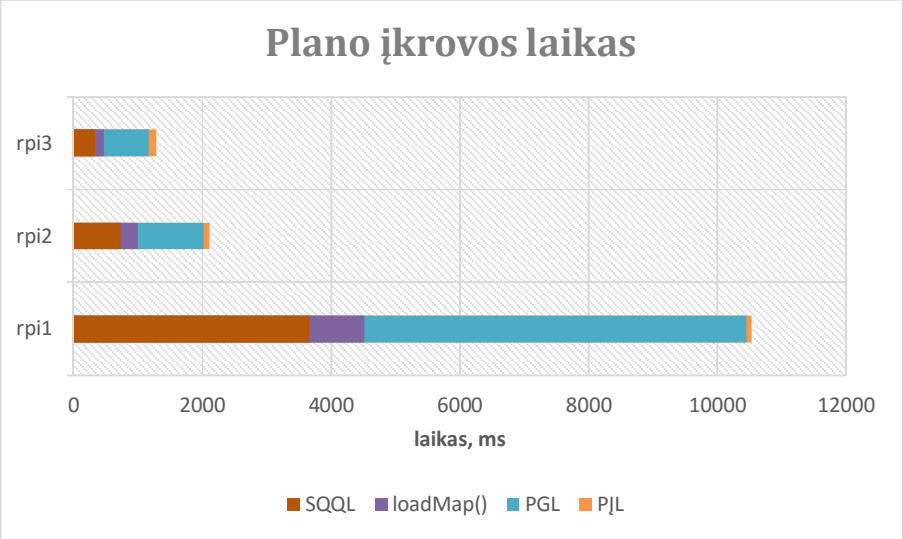



**4.2 pav.** Aplikacijų serveri įkrovimo laiko skirtingose kompiuterio „Raspberry Pi" versijose

Programų sistemos greitaveikai patikrinti taip pat buvo tirta kiek laiko trunka įkrauti visą planą administravimo kliente.

Pirmojoje kompiuterio versijoje įkrovos laikas buvo pats lėčiausias ir siekė 10,5 sekundes, iš kurių užklausos atsakymui suformuoti reikėjo 10,4 s. Metodas *laodMap()* užtruko 4,5 sekundės, o SQL užklausų laikas siekė 3,6 sekundes.

„Raspberry Pi" versijoje 2B, matomas drastiškas laiko, reikalingo užkrauti planą sumažėjimas, šis laikas siekė tik 2,1 sekundės, šio laiko dalį (2,0 s) užtruko plano atsiuntimas, aplikacijų serverio *loadMap()* metodas truko 1,0 s. SQL serverio užklausos truko tik 328 ms.




**4.3 pav.** Plano įkrovimo administravimo kliente laikas skirtingose kompiuterio „Raspberry Pi" versijose

Kaip matome iš pateiktų rezultatų, lyginat kompiuterio „Raspberry Pi" versiją 1B su vėlesnėmis versijomis, pastebimas didelis išmaniojo namo sistemos pagreitėjimas. Šis greitaveikos gerėjimas susijęs tik su aparatūrinės įrangos parametrais, nes tyrime buvo naudoti tie patys operacinės sistemos ir programinės įrangos versijos ir parametrai. „Raspberry Pi 2B" versijoje, lyginant su „1B", pasikeitė procesorius, vietoje vieno branduolio ARMv6 procesoriaus naudojamas ARMv7 keturių branduolių procesorius. Tai leidžia atlikti daugiagijas operacijas, taip paspartinant OS ir sudėtingų programinių sistemų veikimą. Taip pat pagerėjo ir operatyvinės atminties kiekis bei jos sparta. Sudėtingoms ir didelėms Java virtualioje mašinoje veikiančioms programoms šis aparatūros spartos pagerėjimas turi didelę įtaką.

### 4.6.2. Kortelės greičio įtaka sistemos veikimui

Įtaką programų sistemų veikimui galėtų turėti ir integruotos SD atminties kortelės greitis. Bendruomenės dokumentacijoje [37] nurodoma, kad su daugeliu atminties kortelių integruotas „Raspberry Pi" kompiuteris iš viso nėra suderinamas. O ir su tomis kortelėmis, su kuriomis Rpi veikia, nurodoma sparta žymiai skiriasi. Darytina prielaida, kad ir išmaniojo namo programų sistemos greitaveika gali skirtis. Programų sistema startuoja iš šios atminties, duomenų bazės įrašai taip pat nebūtinai yra įkrauti į operatyvinę atmintį.

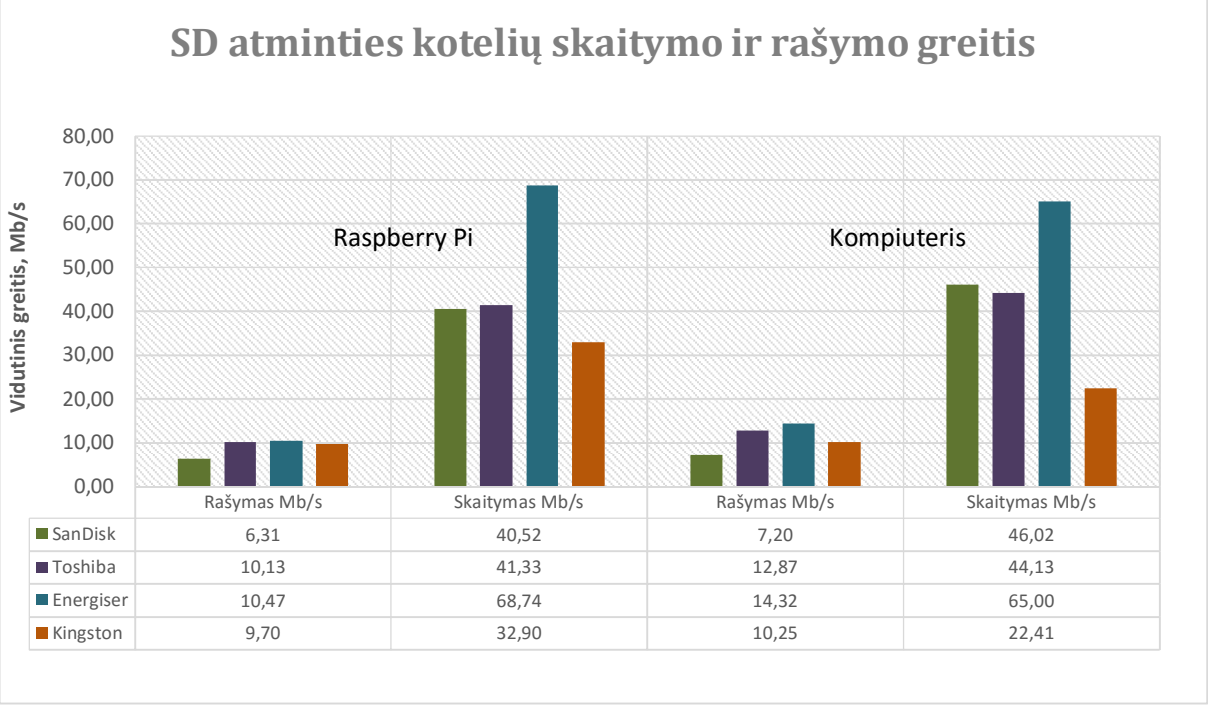
Buvo atliktas bandymas naudojant keturias atminties korteles, norint įsitinki, kokią įtaką daro pastoviosios atminties skaitymo ir rašymo greitis programų sistemos greitaveikai. Šios kortelės pasižymi skirtingu rašymo ir skaitymo greičiu. Pirmiausiai išbandėme kortelės

greitį idealiomis sąlygomis. T.y. failų užpildymo nuliais greitis buvo laikomas rašymo greičiu. Failo perkėlimas į operatyviąją atmintį laikomas rašymo greičiu.

**4.1 lentelė** Tyrime naudotų atminties kortelių lentelė

| *Gamintojas* | Talpa | Klasė | Gamintojo Nr. |
|---|---|---|---|
| ***SanDisk*** | 16Gb | 4 | SDSDQM-016G |
| ***Toshiba*** | 16Gb | 10 | SD-C016UHS1(6A |
| ***Energiser*** | 16Gb | 10[3] | FMDABH016A |
| ***Kingston*** | 16Gb | 10 | SDC4/16GB |

Tyrimui buvo pasitrinktos skirtingų gamintojų atminties kortelės. Jos pasižymėjo ir skirtingai deklaruojamais greičiais bei SD greičio klasėmis. Pagal deklaruojamą greitį, greičiausia atminties kortelė turėtų būti *Energiser*, o lėčiausia *SanDisk*.




| | Raspberry Pi Rašymas Mb/s | Raspberry Pi Skaitymas Mb/s | Kompiuteris Rašymas Mb/s | Kompiuteris Skaitymas Mb/s |
|---|---|---|---|---|
| SanDisk | 6,31 | 40,52 | 7,20 | 46,02 |
| Toshiba | 10,13 | 41,33 | 12,87 | 44,13 |
| Energiser | 10,47 | 68,74 | 14,32 | 65,00 |
| Kingston | 9,70 | 32,90 | 10,25 | 22,41 |

**4.4 pav.** Tyrime naudotų kortelių rašymo ir skaitymo greitis

[3] *Energiser* kortelė turėjo reitingą UHS1

Vidutinis *Energiser* skaitymo greitis „Raspberry Pi" buvo greičiausias ir siekė 68 Mb/s, *SanDisk* ir *Toshiba* greitis buvo apylygis ir siekė atitinkamai 40,52 ir 41,33 Mb/s, lėčiausiai testo metu veikė *Kingston* atminties kortelė, jos skaitymo greitis buvo 32,9Mb/s.

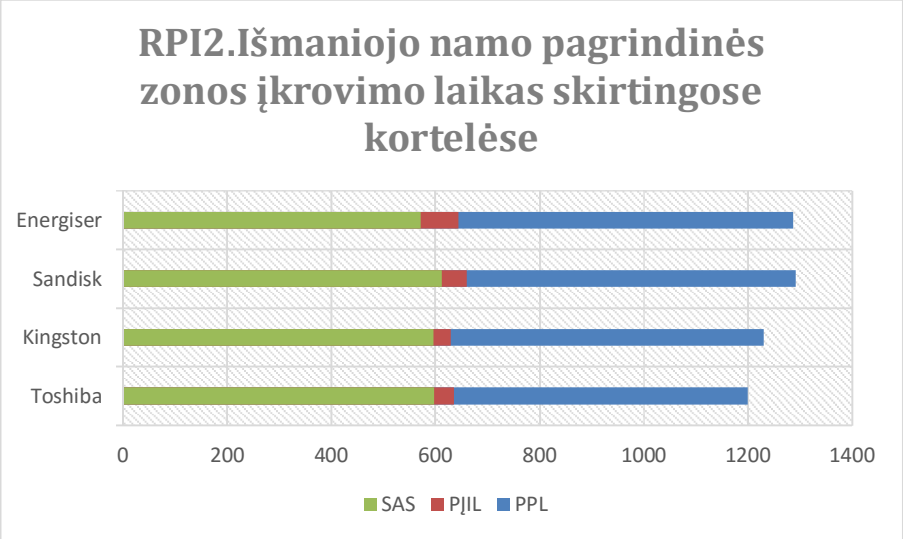
Rašyti greičiausiai galėjome į *Energiser* atminties kortelę, *Toshiba* ir *Kingston* greitis buvo atitinkamai 10,13 ir 9,70 Mb/s. Lėčiausiu rašymo greičiu pasižymėjo *SanDisk* atminties kortelė, tik 6,31Mb/s. Verta paminėti, įprastiniame namų kompiuteryje, tuo pačiu metodu bandytas rašymo į standųjį diską iš "/dev/null" greitis siekė 66Mb/s.

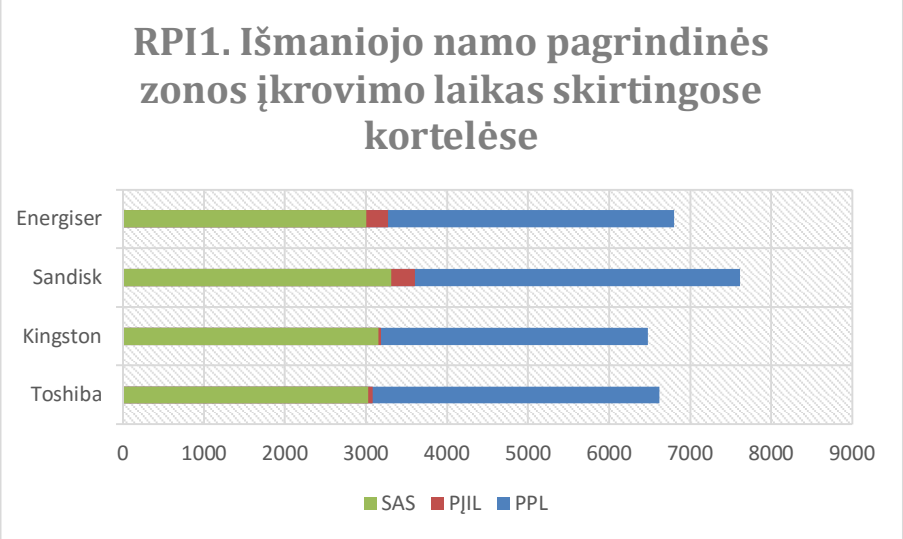
Palyginimui pateikiama atminties kortelių greičius kompiuteryje, kurie gauti perkeliant disko atvaizdą iš/į kortelę. Šie greičiai labai panašūs į tuos, kurie gauti testuojant atminties kortelės greitį „Raspberry Pi".

Pastebėsime, kad nors ir *SanDisk* klasifikuojama kaip 4-os greičio klasės, ir jos rašymo greitis daugiau kaip 1,5 karto lėtesnis, tačiau skaitymo greitis lenkia 10-os klasės *Kingston* kortelę. Rašymo greitis, skirtingai nei kompiuteryje, tarp greitesnių kortelių skiriasi labai nežymiai ir svyruoja ties 10Mb/s. Tikėtina, kad tai yra duomenų perdavimo riba į kortelę kompiuteryje „Raspberry Pi".

Skaitymo greitis integruotame ribotų resursų mažai skyrėsi nuo SD kortelės greičio kompiuteryje. Greičiausiai veikė, kaip ir nurodo gamintojų specifikacijos, *Energiser* kortelė. Šios kortelės skaitymo greitis buvo beveik dvigubai geresnis lyginat su mažiausio greičio kortelės skaitymo greičiu.

Įvertinus šias kortelių charakteristikas, vertinome kaip kortelės greitis įtakoja programų sistemos veikimą. Atminties kortelė yra vienintelis keičiamas kompiuterio „Raspberry Pi" elementas, taigi jeigu galima būtų pagerinti programų sistemos veikimą pakeičiant šią, nebrangią, SD atminties kortelę į geresnių parametrų kortelę, tai būtų labai didelis privalumas. Programų sistemos greitaveika buvo matuota tokiu pačiu būdu, kaip ir vertindami kompiuterio „Raspberry Pi" versijų įtaką. Pradžioje vertinome vartotojo pagrindinio puslapio, kuriame yra atvaizduotas kambarys su keletu prietaisų, įkrovimo laiką.

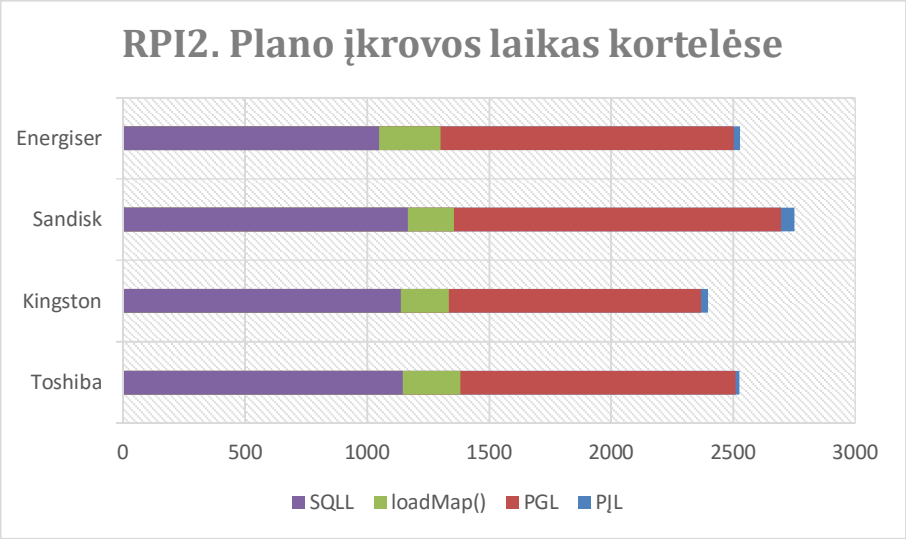



**4.5 pav.** Pagrindinės zonos įkrovimo laikas kompiuteriuose „Raspberry Pi 1B" ir „2B"

Pateiktuose grafikuose parodytas vidutinis puslapio įkrovimo laikas „Raspberry Pi 2B" versijoje ir 1 versijoje. 3-ios versijos įkrovimo laikas ir skirtumai tarp kortelių panašūs į antrosios versijos, todėl čia nepateikti. Kaip galima spręsti pagal įkrovimo laiką, skirtumai tarp kortelių nėra dideli, ir neturi ryšio su greičio matavimu. Tikėtina, kad atlikus daugiau eksperimentų, šie vidutiniai laikai taptų lygūs.

Vertinant šiuos duomenis statistiškai, įrankio IBM SPSS pagalba, „Kruskal-Wals" nepriklausomų imčių testu, SAS, PĮIL ir PPL laikų pasiskirstymas skirtingose kortelėse yra toks pat. Tai įrodo, kad SD atminties kortelės greitis nėra svarbus pagrindinio puslapio įkrovimo greičiui.




**4.6 pav.** Plano įkrovos laikas kompiuteryje „Raspberry Pi 2B"

Plano įkrovos laikas taip pat svyravo nuo 2,3 sekundės *Kingston* kortelėje iki 2,7 sekundės *SanDisk* kortelėje. SQL operacijų laikas, kur tikėtina, disko operacijos gali turėti didesnę įtaką, taip pat svyravo labai nedaug – nuo 1051ms *Energiser* kortelėje iki 1165 ms *SanDisk* kortelėje. Tačiau greičiausių parametrų kortelėje *Energiser* SQL operacijų laikas trumpiausias, o lėčiausioje *SanDisk* kortelėje ši operacija užtruko ilgiausiai ir šis skirtumas sudarė vidutiniškai 114 ms – t.y. apie 10 % SQL operacijos laiko. Vertinant, kad šių kortelių matuotas greičio skirtumas siekė 1,5 karto, šis sulėtėjimas yra mažai reikšmingas ir, tikėtina, yra atsitiktinis.

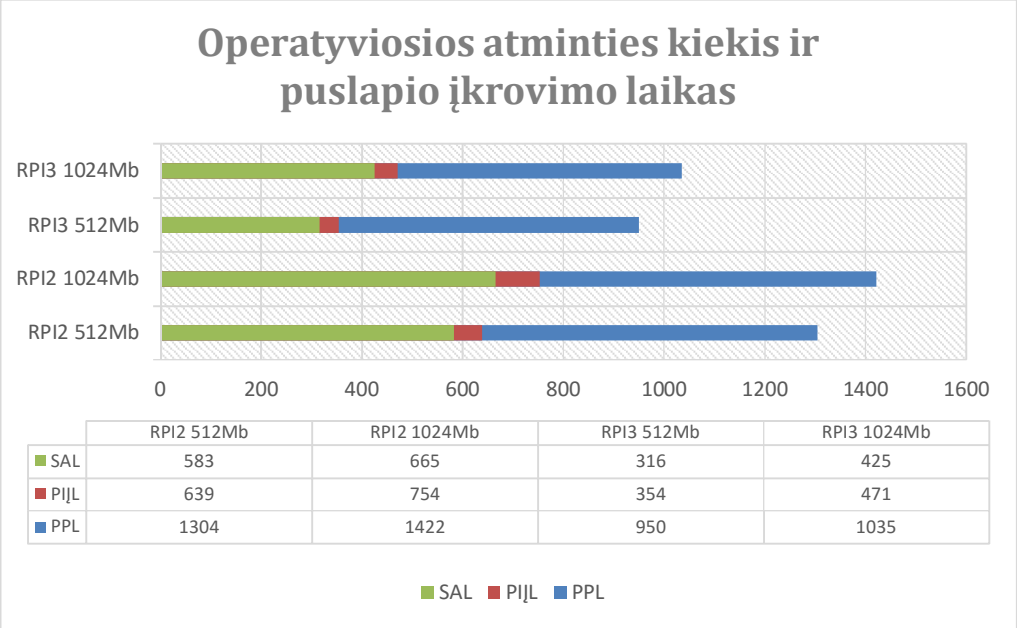
Tiek išmaniojo namo pagrindinio puslapio įkrovimo laikui, tiek viso plano administravimo kliente įkrovimui, kortelės greitis neturėjo jokios reikšmės, todėl darytina išvada, kad kuriant sudėtingas programų sistemas, skirtas veikti ribotų resursų kompiuteriuose, renkantis diegimo aplinką, nepriklausomai ar tai daroma reikalavimų specifikavimo metu ar vėliau, atminties kortelės greičio įtaka yra nereikšminga, todėl atminties kortelė reikėtų rinktis vadovaujantis kitais kriterijais.

### 4.6.3. Operatyviosios atminties talpos įtaka programų sistemos veikimui

Tirtose „Raspberry Pi" kompiuterio versijose operatyviosios atminties talpa svyruoja nuo 512 Mb pirmojoje versijoje  iki 1024 Mb antrojoje ir trečiojoje versijose. Kaip minėta, pirmojoje kompiuterio versijoje išmaniojo namo sistemos veikimas yra labai lėtas, lyginant su antrąja ir trečiąja versijomis. Trūkstant operatyviosios atminties, programų sistemų veikiamas

yra labai sulėtėja, nes dalis atminties turi būti nuolat perrašoma į mainų sritį (angl. swap), kurios greitis yra nepalyginamai mažesnis, nei RAM greitis.

Tam, kad atmesti arba patvirtinti, operatyviosios atminties kiekio įtaką sukurtos išmaniojo namo programų sistemos veikimui apribojome operatyviosios atminties kiekį komanda *mount –t tmpfs /mnt/test1/ /mnt/test1*, kuri sukuria ir prijungia prie failinės sistemos pusės (512 Mb) operatyviosios atminties didžio virtualų diską, kas neleidžia OS veikiančios programoms naudotis šiais atminties ištekliais.




| | RPI2 512Mb | RPI2 1024Mb | RPI3 512Mb | RPI3 1024Mb |
|---|---|---|---|---|
| SAL | 583 | 665 | 316 | 425 |
| PIĮL | 639 | 754 | 354 | 471 |
| PPL | 1304 | 1422 | 950 | 1035 |

**4.7 pav.** Puslapio įkrovimo laikas esant skirtingam operatyviosios atminties kiekiui kmpiuteriuose „Raspberry Pi 2B"

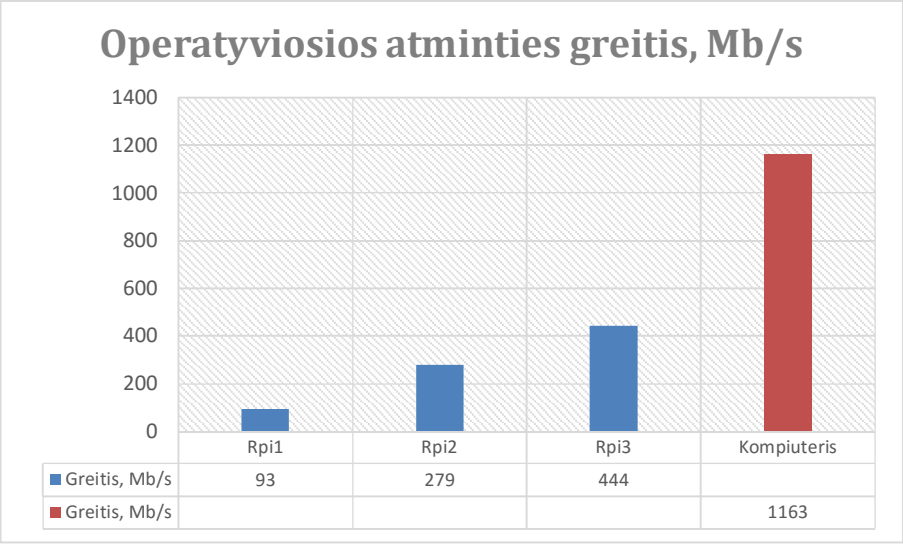
Kaip matome iš pateiktų duomenų, puslapio įkrovimo laikui atminties apribojimas iki 512 Mb neturėjo įtakos. „Raspberry Pi" antrojoje versijoje puslapis pilnai buvo įkraunamas per 1422 ms, o apribotais atminties resursais toje pačioje versijoje per 1304 s. Tokie patys rezultatai matyti ir „Raspberry Pi" trečiojoje versijoje, pagrindinio išmaniojo namo puslapio įkrovimo laikas be atminties ribojimo yra panašus, kaip ir laikas, per kurį puslapis įkraunamas neapribojus šio laiko.

„Raspberry Pi" veikiantis su išmaniojo namo programa tiek pirmojoje versijoje, tiek kitose, turi dar nepanaudotų atminties resursų. Aplikacijų serveris „Tomcat®", vertinant "ps aux" komandos pateiktas VSS reikšmę naudoja 175 Mb, duomenų bazės serveris „MySQL" naudoja 419 Mb. Dalis atminties naudojama kitiems operacinės sistemos procesams, tačiau 512 Mb operatyviosios atminties užtenka pilnai palaikyti išmaniojo namo sistemos funkcionalumą, neperkeliant atminties į lėtą mainų srities (angl. swap) sritį.

### 4.6.4. Atminties greičio įtaka programų sistemos veikimui

Testuojami „Raspberry Pi" modeliai pasižymi ne tik operatyviosios atminties talpos skirtumu, bet ir operatyviosios atminties greitis naujesniuose (2B ir 3B) modeliuose pastebimai skiriasi nuo pirmojo modelio.

Operatyviosios atminties greitis matuotas užpildant "0" reikšme failą sukurtą laikinoje operatyviosios atminties falinėje sistemoje komanda *mount –t tmpfs /mnt/test1 /mnt/tets1*, užpildymas vykdomas komanda *sudo dd if=/dev/zero of=/mnt/tets1 bs=8k count=50k*. Šiuo būdu operatyviojoje atmintyje sukuriamas 419 Mb failas. Šio failo sukūrimo greitis ir yra laikomas operatyviosios atminties greičiu.




**4.8 pav.** Operatyviosios atminties greitis kompiuteriuose „Raspberry Pi".

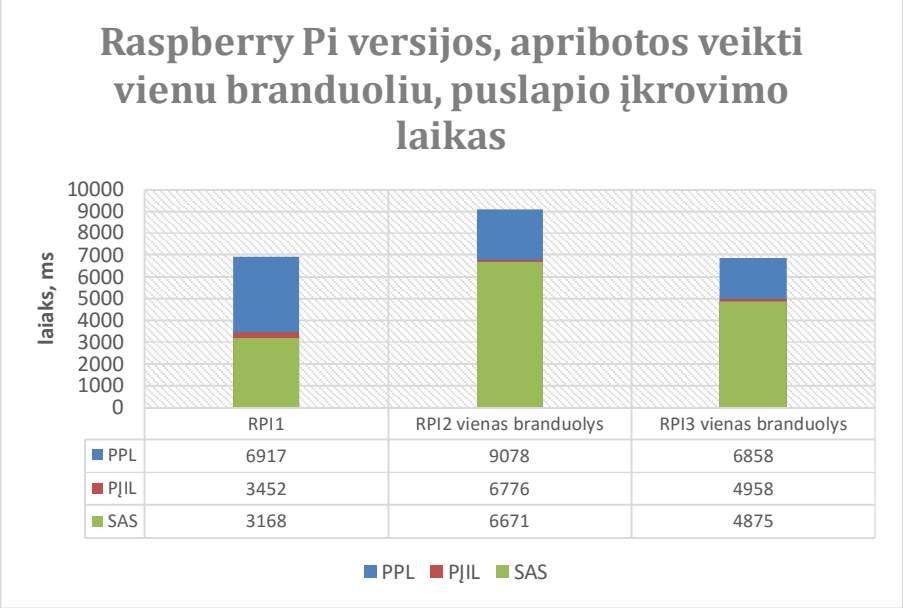
Kiekvienoje sekančioje „Raspberry Pi" versijoje atminties greitis didėja. Pirmojoje versijoje atminties greitis matuojant buvo 93 Mb/s, antrojoje 279 Mb/s t.y. beveik tris kartus didesnis. Trečiojoje – 444 Mb/s.

Pagrindinio puslapio įkrovimo laikas „Raspberry Pi 3B" yra greičiausias, „2B" – lėtesnis, ir lėčiausias pirmojoje versijoje, taip pat ir viso plano įkrovimo greitis, bei aplikacijų serverio „Apache Tomcat®" įkrovimo laikas. Ši seka, matoma ir „Raspberry Pi" atminties greičio matavimo rezultatuose, todėl, darytina prielaida, kad atminties greitis įtakoja išmaniojo namo programų sistemos veikimo greitį.

### 4.6.5. Procesoriaus įtaka programų sistemos veikimui

Pirmosios versijos integruotas kompiuteris „Raspberry Pi" turi „ArmV6" architektūros vieno branduolio procesorių veikiančiu 700 Mhz taktiniu dažniu. Antrosios ir trečiosios versijų Raspbery Pi turi „ArmV7" procesorių su keturių branduolių procesoriumi, veikiančiu atitinkamai 900 Mhz ir 1200 Mhz taktiniu dažniu.

Pabandėme apriboti Linux branduolį, kad „Raspberry Pi 2" ir „Raspberry Pi 3" veiktų naudodamas vieną procesoriaus branduolį, keisdami */boot/cmdline.txt* parametrą *maxcpu* į reikšmę "1".




**4.9 pav.** Puslapio įkrovimo laikas skirtingose „Raspberry Pi" kompiuterio versijose, kaip naudojamas vienas branduolys

Iš pateiktų rezultatų matome, kad apribotos „Raspberry Pi" veikti su vienu branduoliu, puslapio įkrovimo laikas ypač sulėtėja. „Raspberry Pi" 2B versijoje puslapis įkraunamas per 9 sekundes, iš kurių operacijos serveryje trunka 6,1 sekundes, ir per mažiau nei 100 ms sugeneruotas atsakymas atsiunčiamas kliento naršyklei. Tuo tarpu „Raspberry Pi" 3B versijoje veikė šiek tiek greičiau ir puslapis įkraunamas per 6,9 sekundes, iš kurių operacijos serveryje trunka 4,9 sekundes ir po 5 sekundžių šis išteklis pasiekiamas naršyklei.

Nepribotos Rpi3 ir Rpi2 šiuos veiksmus atlieka per 1,0 ir 1,4 sekundes[4]. Tai yra daugiau kaip 6 kartus greičiau. O operacijos serveryje skirtos sugeneruoti pagrindinį išteklį naršyklei trunka per 10 kartų lėčiau naudojant vieną branduolį.

### 4.6.6. Java virtualios mašinos įtaka programų sistemos veikimui

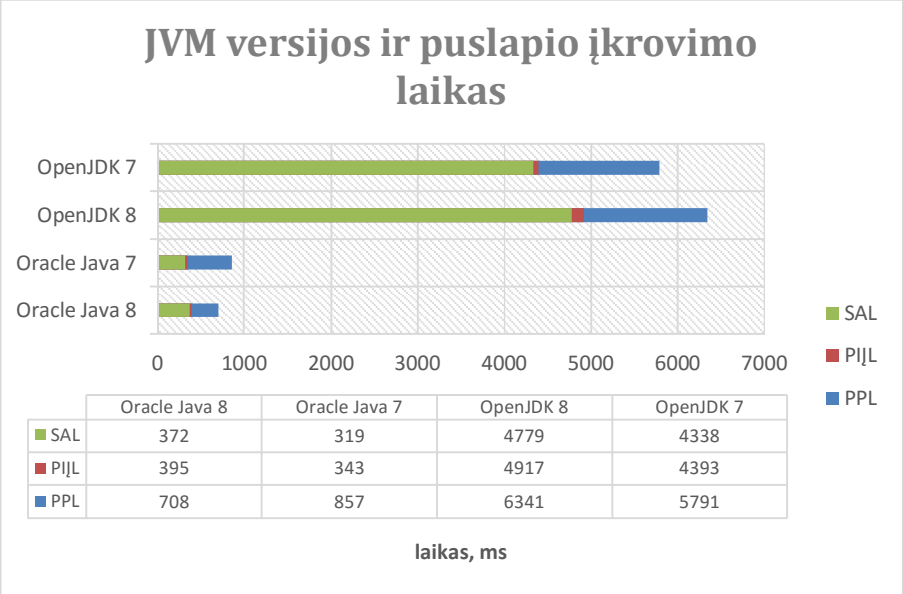
Virtualioji mašina, kurioje veikia aplikacijų serveris „Apache® Tomcat" su išmaniojo namo programų sistema, turi labai didelę įtaką programų sistemos veikimui. Išmaniojo namo aplikacija buvo kurta, kad palaikytų „Java 7" kalbos specifikaciją, todėl galime išbandyti kaip veikia skirtingose jave kalbos versijose (7 ir 8). „Arm" Procesoriui prieinamos kompanijų „Oracle" ir „OpenJDK" virtualios mašinos. Panagrinėsime programų sistemos greitaveiką šių VM septintoje ir aštuntose versijose.

**4.2 lentelė** Tyrime naudotos „Java" Virtualios mašinos implementacijos.

| ***Java VM*** | **Komandos "java –version" išvestis** |
|---|---|
| *„Oracle" Java 8* | java version "1.8.0_65"<br>Java(TM) SE Runtime Environment (build 1.8.0_65-b17)<br>Java HotSpot(TM) Client VM (build 25.65-b01, mixed mode) |
| *„Oracle" Java 7* | java version "1.7.0_60"<br>Java(TM) SE Runtime Environment (build 1.7.0_60-b19)<br>Java HotSpot(TM) Client VM (build 24.60-b09, mixed mode) |
| *„OpenJDK" 8* | „OpenJDK" version "1.8.0_40-internal"<br>„OpenJDK" Runtime Environment (build 1.8.0_40-internal-b04)<br>„OpenJDK" Zero VM (build 25.40-b08, interpreted mode) |
| *„OpenJDK" 7* | java version "1.7.0_95"<br>„OpenJDK" Runtime Environment (IcedTea 2.6.4) (7u95-2.6.4-1~deb8u1+rpi1)<br>„OpenJDK" Zero VM (build 24.95-b01, mixed mode) |

Lentelėje pateiktos tirtos Java versijos. Dešiniajame stulpelyje pateikiamas pavadinimas, kuris bus naudojamas toliau tekste. Kairiajame komandos *Java -version* išvestis, kuri nurodo, kokia VM mašinos versija veikia sistemoje.

[4] Detaliau skyriuje "Išmaniojo namo sistemos veikimas skirtingose RPi versijose"

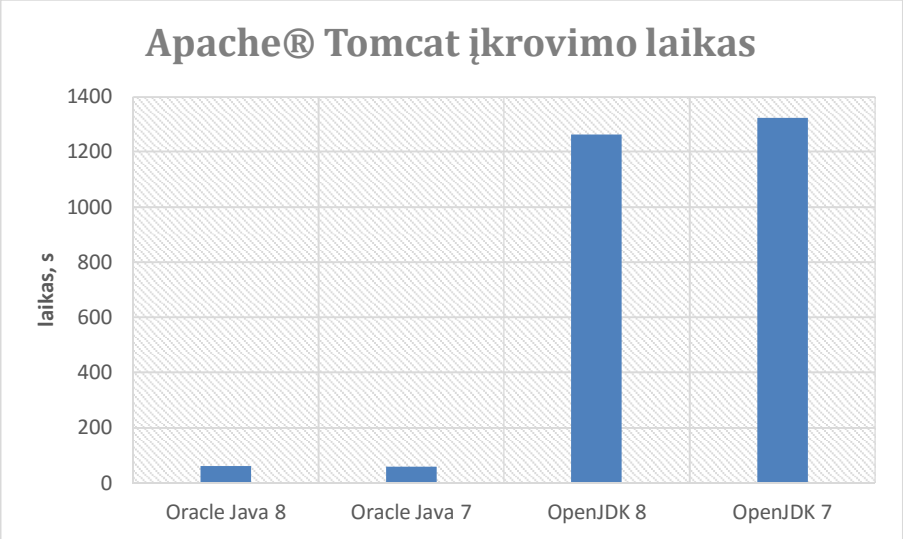



| | Oracle Java 8 | Oracle Java 7 | OpenJDK 8 | OpenJDK 7 |
|---|---|---|---|---|
| SAL | 372 | 319 | 4779 | 4338 |
| PIĮL | 395 | 343 | 4917 | 4393 |
| PPL | 708 | 857 | 6341 | 5791 |

**4.10 pav.** Puslapio įkrovimo laikas skirtingose virtualiose mašinose

Naudojant „Oracle" Java 8 pagrindinio išmaniojo namo puslapio vidutinis įkrovimo laikas siekė 708ms, iš šio laiko pagrindinis išteklius buvo įkrautas per 395ms, o kodo vykdymas serveryje truko 372 ms. Šiek tiek lėčiau veikė „Oracle" Java 7 versija, vidutinis įkrovimo laikas buvo 857 ms, tačiau pagrindinis išteklius įkrautas, bei operacijos serveryje vyko net greičiau nei Java 8 versijoje, atitinkamai 343 ir 319 ms. „OpenJDK" virtualios mašinos 8 versijoje įkrovimo laikas siekė net 6,3 sekundes, pagrindinis išteklius buvo įkrautas tik po 4,9s, operacijos serveryje truko didžiąją šio laiko dalį 4,8 sekundes. „OpenJDK" versijoje puslapis įkrautas per 5,7 sekundes, šio laiko 4,4 s truko pagrindinio resurso įkrovimas ir 4,3 sekundes truko suformuoti atsakymą (angl. response) serveryje.

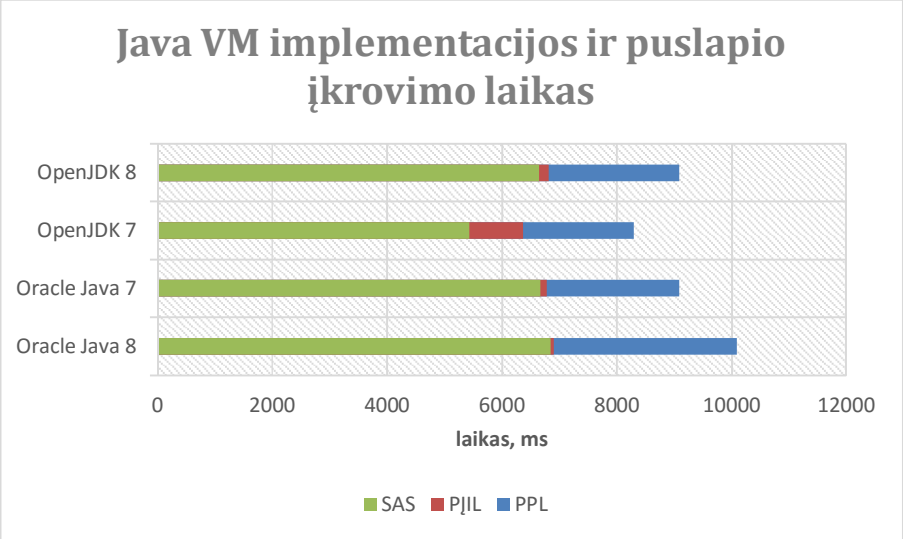
„OpenJDK" tyrimo metu "pasirodė" ypač lėta, lyginant su „Oracle" Java virtualios mašinos implementacija, išmaniojo namo pagrindinio puslapio įkrovimo greitis buvo nuo 6,6 iki 8,9 lėtesnis, o operacijos serveryje paruošti pagrindinį išteklį truko 11,6 iki 15 kartų lėčiau.




**4.11 pav.** Aplikacijų serverio įkrovimo laikas skirtingose virtualiose mašinose

Laikas, per kurį įkraunamas aplikacijų serveris „Apache® Tomcat" ir išmaniojo namo programų sistema paruošiama darbui, taip pat buvo panašiai pasiskirtęs. „OpenJDK" 8 versijoje aplikacija įkraunama per 21 minutes ir 3 sekundes, septintojoje versijoje – 22 minutes ir 2 sekundes. O „Oracle Java 8" virtualioje mašinoje serveris įkraunamas per 1 minutę, o septintoje versijoje, dar greičiau, per 57 sekundes. Taigi apliakcijų serveris „Apache® Tomcat" su išmaniojo namo sistema „Oracle" VM implementacija įkraunamas daugiau kaip 20 kartų greičiau, lyginant su „OpenJDK" *Arm* procesoriui skirta implementacija.

Yra pastebima, kad „OpenJDK" laikas yra panašus į laiką, kuris reikalingas puslapį įkrauti naudojant tik vieną procesoriaus branduolį, todėl gilinantis kodėl „OpenJDK" virtualioji mašina, lyginant su „Oracle" implementacija veikia taip prastai, buvo atliktas bandymas su „Raspberry Pi 2B" versija, kurioje, anksčiau minėtu būdu, "atjungti" visi 3 branduoliai ir paliktas tik vienas. Kaip įprasta, buvo matuotas puslapio įkrovimo laikas.




**4.12 pav.** Puslapio įkrovimo greitis skirtingose virtualiose mašinose, kai „Raspberry Pi" kompiuteryje naudojamas tik vienas procesoriaus branduolys

Kaip minėta apribotame „Raspberry Pi" 2B kompiuteryje, puslapis įkraunamas per ilgesnį laiką, nei naudojant visus branduolius. Naudojant „Oracle" Java 7 tai vidutiniškai truko 10,5 sekundžių. Naudojant „Oracle" Java 7 – vidutiniškai 9,0 sekundes. Su „OpenJDK" implementacijos 7 ir 8 versijomis atitinkamai 8,2 ir 9,0 sekundės. Panašūs rezultatai buvo gauti ir „Raspberry Pi" versijoje 1B, kuri turi tik vieną procesoriaus branduolį.

Taigi „OpenJDK" apribotame kompiuteryje, kaip matome, veikia panašiai, kaip ir „Oracle" implementacija. Veikiant visais branduoliais „Oracle" Java yra iki 10-ies kartų greitesnė. Darytina išvada, kad „OpenJDK" nepanaudoja kelių branduolių sistemos architektūros ir, nepaisant galimybės lygiagrečiai vykdyti procesus, veikia tik viename procesoriaus branduolyje, kas sąlygoja daug mažesnį sistemos našumą, kas ypač atsiliepia vykdant sudėtingą, daugelį gijų turinčią, protingo namo aplikaciją.

Gauti rezultatai dar kartą patvirtina, kad panašių sistemų architektams reikėtų atidžiai rinktis programinės įrangos komponentus, arba konkrečiai – virtualios mašinos implementaciją. Riboto našumo kompiuteriuose, tokiuose kaip „Raspberry Pi", netinkamas pasirinkimas gali sąlygoti žymų programų sistemų veikimo sulėtėjimą, kai šie aparatūriniai ištekliai tiesiog nėra išnaudojami. Šių dienų integruoti procesoriai dažnai veikia naudodami daugiabranduolę architektūrą, tokie procesoriai sutinkami ir mobiliuose telefonuose. Taigi esant galimybei rinktis tarp Virtualios mašinos implementacijų, reiktų atidžiai išnagrinėti pastarųjų specifikacijas, arba bandyti programų sistemos greitaveiką su skirtingomis versijomis.

## 4.7. Tyrimo išvados

Buvo tirta sukurta išmaniojo namų programų sistema, siekiant išsiaiškinti, kokių techninių parametrų kompiuteriuose sistemos greitaveika bus priimtina vartotojui bei ateityje padėti panašių sistemų architektams pasirinkti aparatūrinę įrangą, kurioje skandžiai ir patikimai veiktų sudėtinga virtualioje mašinoje veikianti sudėtinga programinė įranga.

Tyrimo metu paaiškėjo, kad didžiausią įtaką programų sistemos veikimui turi integruoto kompiuterio aparatūrinės įrangos gebėjimas atlikti lygiagrečius procesus daugiabranduoliuose procesoriuose. Didelę įtaką turėjo operatyviosios atminties greitis. Skirtingų greičių atminties kortelės nedaro įtakos programų sistemos veikimui. Tiriant virtualios mašinos įtaką paaiškėjo, kad su „Oracle Java 7" VM implementacija programų sistema veikia greičiausiai, o su „OpenJDK 8" VM versija programų sistemos veikimas sulėtėja daugiau kaip 10 kartų. Padaryta išvada, kad „OpenJDK" nepalaiko „Arm" procesoriaus lygiagrečių procesų vykdymo galimybės daugiabranduoliuose procesoriuose.

# 5. IŠVADOS

1. Apžvelgus publikuotus mokslinius straipsnius ir kitą literatūrą nustatyta, kad išmaniojo namo programų sistemos diegimo aplinkos reikalavimus geriausiai atitinka kompiuteris „Raspberry Pi" - jame galima implementuoti SPI, I2C, „ModBus" ir „1-wire" laidinius protokolus.
2. Suprojektuota išmaniojo namo sistema, susidedanti iš serverio, diegiamo kompiuteryje "Raspberry Pi" ir administravimo kliento. Serveryje veikia prietaisų valdymo logika ir vartotojo sąsaja, pasiekiama per „Web" naršyklę.
3. Kurta programų sistema veikia pakankamai greitai, kad patenkintų vartotojų lūkesčius, naudojant „Raspberry Pi 2B" integruotą kompiuterį. Šiame kompiuteryje pagrindinį vartotojo puslapį galima įkrauti iki 5 kartų greičiau.
4. Pagrindinis veiksnys lemiantis programų sistemos greitaveiką bei vartotojo laukimo trukmę – aparatūrinės įrangos lygiagrečių užduočių atlikimo galimybė daugiabranduoliuose procesuose. Vienu branduoliu veikiančio kompiuterio „Raspberry Pi" vartotojo aplikacijos įkrovimo greitis mažesnis 5 kartus.
5. Operatyvios atminties greitis, programinei įrangai veikiant Java virtualioje mašinoje, turį didelę įtaką sistemos greitaveikai.
6. Įtakos programų sistemos greičiui neturi pasirinktos atminties kortelės greitis.
7. Norint išnaudoti daugiabranduolių procesorių galimybes, lygiagretūs procesai turi būti tinkamai koordinuojami virtualios mašinos lygyje. „OpenJDK" nepalaiko daugiabranduolio "Arm" procesoriaus teikiamų lygiagrečių procesų vykdymo galimybės, todėl veikia iki 10 kartų lėčiau nei „Oracle" VM.

## 6. LITERATŪRA


[1] A. Kailas, V. Cecchi, ir A. Mukherjee, „A survey of communications and networking technologies for energy management in buildings and home automation“, *J. Comput. Networks …*, 2012.

[2] MiCasaVerde, „Energy“, 2012. [Interaktyvus]. Available at: http://wiki.micasaverde.com/index.php/Energy. [Žiūrėta: 30-lapkr-2014].

[3] F. Viani, F. Robol, A. Polo, P. Rocca, G. Oliveri, ir A. Massa, „Wireless Architectures for Heterogeneous Sensing in Smart Home Applications: Concepts and Real Implementation“, *Proc. IEEE*, t. 101, nr. 11, p. 2381–2396, lapkr. 2013.

[4] Z. Alliance, „ZigBee specification“, 2006.

[5] D. Croce, D. Garlisi, F. Giuliano, ir I. Tinnirello, „Learning from errors: Detecting ZigBee interference in WiFi networks“, *2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, 2014, p. 158–163.

[6] S. Arif ir S. H. Supangkat, „Simulation and analysis of ZigBee - WiFi interference“, *2014 International Conference on ICT For Smart Society (ICISS)*, 2014, p. 206–210.

[7] R. Lawler, „Wink smart home hubs knocked out by security certificate (update)“, 2015.

[8] R. Price, „Google's parent company is deliberately disabling some of its customers' old smart-home devices“, 2016.

[9] S. I. Azid ir S. Kumar, „Analysis and Performance of a low cost SMS based Home security system“, *Int. J. Smart Home*, t. 5, nr. 3, p. 15–24, 2011.

[10] M. H. A. Wahab, N. Abdullah, A. Johari, ir H. A. Kadir, „GSM based electrical control system for smart home application“, *J. Converg. Inf. Technol.*, t. 5, nr. 1, p. 33–39, 2010.

[11] S. W. Harald Sundmaeker, Patrick Guillemin, Peter Friess, *Vision and Challenges for Realising the Internet of Things*. CERP-IoT, 2010.

[12] D. Pishva ir K. Takeda, „Product-based security model for smart home appliances“, *IEEE Aerosp. Electron. Syst. Mag.*, t. 23, nr. 10, p. 32–41, spal. 2008.

[13] D. Valtchev ir I. Frankov, „Service gateway architecture for a smart home“, *IEEE Commun. Mag.*, t. 40, nr. 4, p. 126–132, bal. 2002.

[14] M. Schneps-Schneppe, A. Maximenko, D. Namiot, ir D. Malov, „Wired Smart Home: Energy metering, security, and emergency issues“, *2012 IV International Congress on Ultra Modern Telecommunications and Control Systems*, 2012, p. 405–410.

[15] V. Philippopoulos ir C. Georgopoulos, „Smart Homes: a user perspective“, *hft.org*, 2004.

[16] DZone, „2014 Guide to Internet of Things“, 2014. [Interaktyvus]. Available at: file:///C:/Users/jokub_000/Downloads/DZone_GuideToInternetOfThings_6.pdf. [Žiūrėta: 30-lapkr-2014].

[17] B. Krishnamachari ir C. S. Raghavendra, „Performance evaluation of the IEEE 802.15.4 MAC for low-rate low-power wireless networks“, *IEEE International Conference on Performance, Computing, and Communications, 2004*, 2004, p. 701–706.

[18] P. Semiconductors, „The I2C-bus specificatuin“, 2000. [Interaktyvus]. Available at: http://151.100.120.244/personale/balsi/didattica/testi/testiSE-9.pdf. [Žiūrėta: 19-lapkr-2014].

[19] F. Leens, „An introduction to I 2 C and SPI protocols“, *Instrum. Meas. Mag. IEEE*, 2009.

[20] D. Awtrey ir D. Semiconductor, „Transmitting data and power over a one-wire bus“, *Sensors-The J. Appl. …*, 1997.

[21] Maxim, „Overview of 1-Wire Technology and Its Use - Tutorial - Maxim“, *Overview of 1-Wire Technology and Its Use*, 2008. [Interaktyvus]. Available at: http://www.maximintegrated.com/en/app-notes/index.mvp/id/1796. [Žiūrėta: 27-lapkr-2014].

[22] E. Diaconescu, „An identifying and authorizing application using 1-wire® technology“, *2010 IEEE 16th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, 2010, p. 243–248.

[23] A. M. N. Montoya, L. Giraldo, D. Aristizá bal, „Remote Monitoring and Control System of Physical Variables of a Greenhouse through a 1-Wire Network“, *Advances in Systems,*

*Computing Sciences and Software Engineering*, T. Sobh ir K. Elleithy, Sud. Dordrecht: Springer Netherlands, 2006.
[24] K.-X. Tee, M.-T. Chew, ir S. Demidenko, „An Intelligent Warehouse Stock Management and Tracking System Based on Silicon Identification Technology and 1-Wire Network Communication“, *2011 Sixth IEEE International Symposium on Electronic Design, Test and Application*, 2011, p. 110–115.
[25] X. Xu, W. W. Yao, ir L. Gong, „Information Technology in Intelligent Warehouse Management System Based on ZigBee“, *Advanced Materials Research*, 2014, t. 977, p. 468–471.
[26] J. Haule ir K. Michael, „Implementation of Zigbee based Wireless Automated Irrigation Management System for Small Scale Farmers“, *IJCER*, t. 3, nr. 5. p. 262–266, 01-lapkr-2014.
[27] S. Moon, S.-J. Kim, J.-W. Seo, J.-H. Park, C. Park, ir C.-S. Chung, „Maximum power point tracking without current sensor for photovoltaic module integrated converter using Zigbee wireless network“, *Int. J. Electr. Power Energy Syst.*, t. 56, p. 286–297, kovo 2014.
[28] Z. Shelby ir C. Bormann, *6LoWPAN: The Wireless Embedded Internet*. 2011.
[29] Z-wave, „Z-Wave Protocol Overview“, 20006.
[30] M. Maksimovic, V. Vujovic, N. Davidovic, V. Milosevic, ir B. Perisic, „Raspberry Pi as Internet of Things hardware: Performances and Constraints“, *IcETRAN 2014*, 2014.
[31] S. Jain, A. Vaibhav, ir L. Goyal, „Raspberry Pi based interactive home automation system through E-mail“, *2014 International Conference on Reliability Optimization and Information Technology (ICROIT)*, 2014, p. 277–280.
[32] J. A. Patel, A. Shubhangi, S. Joshi, A. Pawar, ir N. Bari, „Raspberry PI Based Smart Home“, *Int. J. Eng. Sci.*, t. 2800, 2016.
[33] S. Robertson ir J. Robertson, *Mastering the Requirements Process: Getting Requirements Right*. 2012.
[34] A. Bouch, A. Kuchinsky, ir N. Bhatti, „Quality is in the eye of the beholder“, *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '00*, 2000, p. 297–304.
[35] S. Avila, „The Impact Of Page Loading Times On Site Revenue“, 2015. [Interaktyvus]. Available at: http://www.surgedigital.co.uk/blog/the-impact-of-page-loading-times-on-site-revenue/. [Žiūrėta: 12-geg-2016].
[36] S. K. Paul Rubin, David MacKenzie, „dd - Linux manual page“, 2016. [Interaktyvus]. Available at: http://man7.org/linux/man-pages/man1/dd.1.html. [Žiūrėta: 12-geg-2016].
[37] „RPi SD cards - eLinux.org“, 2016. [Interaktyvus]. Available at: http://elinux.org/RPi_SD_cards. [Žiūrėta: 14-geg-2016].

# 7. TERMINŲ IR SANTRUMPŲ ŽODYNAS

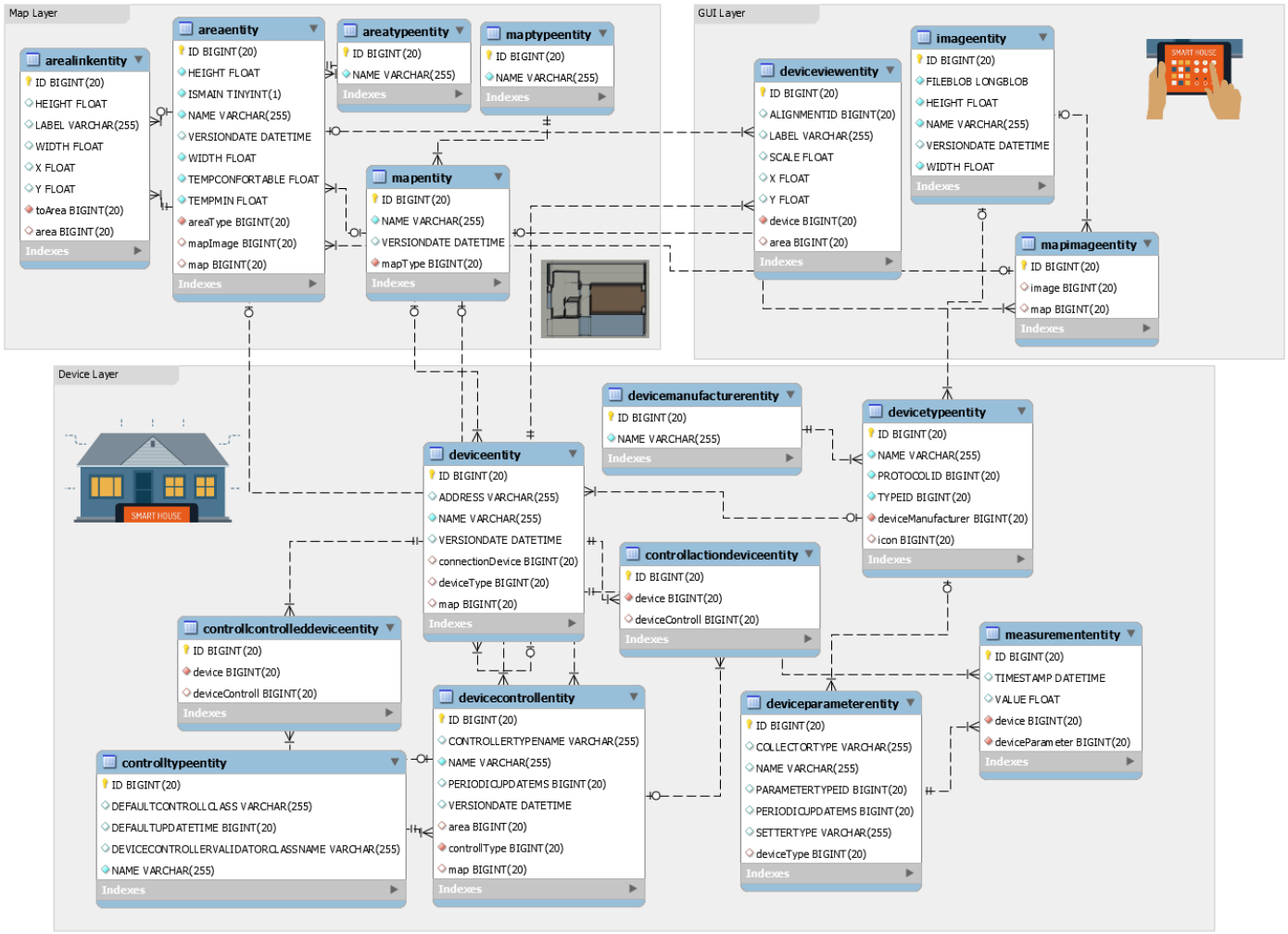
| Terminas arba santrumpa | Terminas anglų kalba | Apibūdinimas |
|---|---|---|
| GPIO | General purpose input/output | Tai integruoto kompiuterio išvadai skirti komunikuoti su prietaisais |
| IoT | Internet of things | Daiktų internetas tai informacinis tinklas , kuriame galima ieškoti informacijos apie realaus pasaulio objektus pagal unikalų ID (elektroninį produkto kodą) žinant sąveikos mechanizmą. |
| Java | | Aukšto lygio programavimo kalba. |
| JAXB | Java architecture for XML binding | Programavimo kalbos Java standartinė programavimo sąsaja skirta objektus nuoseklinti į XML dokumentus. |
| JPA | Java persistence API | Programavimo kalbos Java standartinė programavimo sąsaja skirta Java objektų saugojimui duomenų bazėse. |
| Kolektorius | Collector | Programoje realizuotas prietaiso vieno parametro gavimo apibrėžimas, veiksmų seka. |
| Kontroleris | Controller | Programoje realizuotas seterių ir geterių sujungimo vienetas, kuris apibrėžia, kaip bus vykdoma automatika. T.y. rinkdamas informaciją iš kolektorių valdo seterius. |
| Kontroliuojantys prietaisai | Control Device | Protingo namo elementas, galintis pateikti informaciją apie namo aplinką. Pvz: mygtukas, termometras |
| M2M | Machine to machine | Komunikacijos koncepcija, kuri pagrįsta mašininiu bendravimu be tarpinio žmogaus įsikišimo. |
| Python | | Aukšto lygio programavimo kalba. |
| Planas | Map | Namas arba arba didžiausias protingo namo suskirstymo elementas, jame yra prietaisai – sensoriai ir vykdomieji prietaisai, kurie bendradarbiauja tarpusavyje tenkindami vartotojų lūkesčius. Visi name esantys prietaisai gali bendradarbiauti tarpusavyje, gali būti daug žemėlapių, tačiau prietaisai priskirti žemėlapyje (name) negali bendrauti tarpusavyje tarp skirtingų namų. Žemėlapis (namas) turi savo grafinį vaizdą – planą, kuriame gali būti atvaizduojami prietaisai. |
| Prietaisas | Device | Protingo namo elementas gebantis priimti arba perduoti informaciją sistemai. Tai yra veikiantys ir/arba kontroliuojantys prietaisai. Kiekvienas prietaisas turi savo tipą, adresą. |
| Prietaiso parametras | Parameter | Funkcija, kurią atlieka prietaiso tipas. Funkcija apibrėžiama programoje realizuotais seteriais ir kolektoriais. |
| Prietaiso tipas | Device type | Prietaisui būdingų parametrų rinkinys. Prietaiso tipas susideda iš vieno ar kelių parametrų. |
| Prietaiso vaizdas | Device View | Tam tikro prietaiso atvaizdavimas vartotojo aplinkoje |

| | | |
|---|---|---|
| Rpi | Raspberry Pi | Kompiuteris išleistas Raspberry Pi fondo. |
| SD | Secure Digital | Atminties kortelės tipas, kuris buvo apibrėžtas „Secure Digital" kompanijos standartu. |
| Seteris | Setter | Programoje realizuotas prietaiso vienos funkcijos keitimo apibrėžimas, veiksmų seka. |
| SOAP | Simple Object Access protocol | Protokolas, kuris yra skirtas apsikeisti struktūrizuotais duomenimis tarp „WebService" kompiuterių tinkluose. |
| Sritis | Area | Kambarys arba jo dalis, mažiausias namo suskirstymo elementas, naudojamas suskirstyti prietaisus į grupes , bei jų atvaizdavimui. Kambarys gali būti ir neapribotas fizine siena, tačiau turėtų būti logiškai ar funkciškai apribotas. |
| URL | Uniform Resource locator | Suvienodinto struktūros adresas, kurio pagalba internete yra keičiamasi duomenimis. |
| Veiklos prietaisai | Action Device | Protingo namo elementas, galintis pakeisti ar koreguoti namo fizinę aplinką pvz. išmaniosios lemputės, reliniai kontaktai, sklendės, žaliuzės. |
| VM | Virtual Machine | Procesas veikiantis kompiuteryje, kuris įgalina veikti programinę įrangą skirtingose operacinėse sistemose, bei teikia kitas paslaugas joje veikiančioms aplikacijoms. |
| Web | World Wide Web | Informacinė erdvė, kurioje ištekliais keičiamasi pagal URL adresą. |
| WebService | | Paslauga, kuri teikiama elektroniniu prietaisų, ir skirta bendrauti šiems tarpusavyje: keistis duomenimis ar inicijuoti tam tikras funkcijas. |
| XML | Extensible Mark-up Language | Žymėjimo kalba, apibrėžianti tam tikras taisykles kurti dokumentus, kurie gali būti perskaityti kompiuterio arba žmogaus. |

# 8. PRIEDAI

## 8.1. UAB „Singletonas” skurta įėjimų išėjimų plokštė

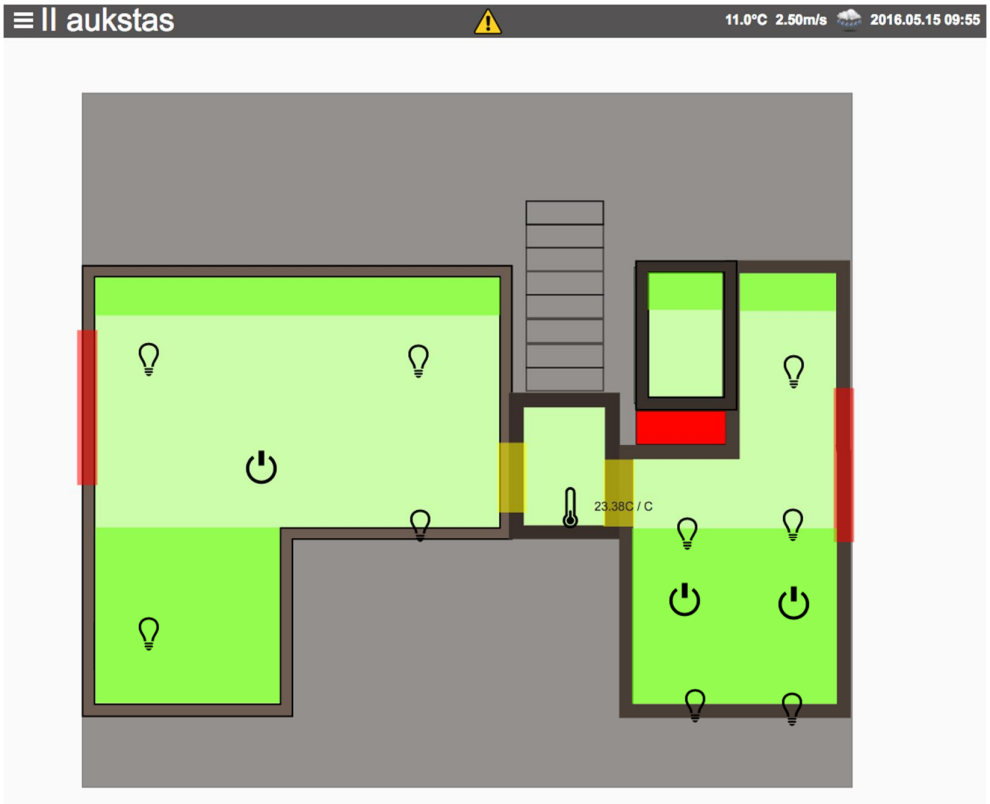
## 8.2. Duomenų modelis



## 8.3. Diegimo aplinkos apžvalga

| | | Arduino UNO | BeagleBone | Cubieboard 3 | pcDuiNE3 | Raspberry Pi B+ | Banana Pi |
|---|---|---|---|---|---|---|---|
| | |  |  |  |  |  |  |
| **SoC** | | Atmel ATmega328P | TI Sitara AM335x | Allwinner A20 | Allwinner A20 | Broadcom BCM2835 | Allwinner A20 |
| **CPU** | Architektūra | AVR | ARM Cortex-A8 | ARM Cortex-A7 | ARM Cortex-A7 | ARM11 | ARM Cortex-A7 |
| | Branduoliai | 1 | 1 | 2 | 2 | 1 | 2 |
| **RAM** | Dydis | 2 KB | 256 MB | 2 GB | 1 GB | 512 MB | 1 GB |
| **USB** | | NE | 1 | 3 | 1 | 2 | 2 |
| **Atmintis** | Plokštėje | 32 KB Flash + 1 KB EEPROM | 4 GB Flash | 8 GB Flash | 4 GB Flash | NE | NE |
| | Jungtys | Nėra | microSD | microSD, SATA 2.0 | microSD, SATA | SD | SD, SATA 2.0 |
| **Tinklas** | Eth. | NE | 10/100 | 10/100/1000 | 10/100 | 10/100 | 10/100/1000 |
| | Wi-Fi | NE | NE | a/b/g/n (BCM4329) | b/g/n (RTL8188) | NE | NE |
| **Komunik.** | Bluetooth | NE | NE | 2.1 + EDR | NE | NE | NE |
| | I²C | TAIP | TAIP | TAIP | TAIP | TAIP | TAIP |
| | SPI | TAIP | TAIP | TAIP | TAIP | TAIP | TAIP |
| **I/O** | GPIO | 22 | 66 | TAIP | 22 | 8 | 80 |
| | Analoginiai | 10-bit ADC, PWM | 12-bit ADC | ? | ADC, PWM | NE | 12-Bit-ADC |
| **Kitos** | | Arduino 1.0 headers | CAN, UART | IrDA, UART | Arduino 1.0 headers | UART | CSI, UART |
| **Operacinės sistemos** | Linux | NE | ? | TAIP | TAIP | TAIP | TAIP |
| **Dydis [mm]** | | 75 × 53 | 86 × 53 | 110 × 80 | ? | 85.6 × 54.0 × 19.5 | 92 × 60 |

## 8.4. Pagrindinio vartotojo puslapio vartotojo sąsajos vaizdas




## 8.5. Administravimo kliento vartotojo sąsajos vaizdas