Kauno technologijos universitetas

Informatikos fakultetas

Išmaniojo virdulio valdymo tyrimas, naudojant įvairias belaidžio ryšio technologijas

Baigiamasis magistro studijų projektas

Aldas Jonauskis

Projekto autorius

Dr. Šarūnas Packevičius

Vadovas

Kaunas, 2023

Kauno technologijos universitetas

Informatikos fakultetas

Išmaniojo virdulio valdymo tyrimas naudojant įvairias belaidžio ryšio technologijas

Baigiamasis magistro studijų projektas

T00M038

Aldas Jonauskis

Projekto autorius

Dr. Šarūnas Packevičius

Vadovas

Prof. Tomas Blažauskas

Recenzentas

Kaunas, 2023

Kauno technologijos universitetas

Informatikos fakultetas

Aldas Jonauskis

Išmaniojo virdulio valdymo tyrimas naudojant įvairias belaidžio ryšio technologijas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs (-usi);

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Aldas Jonauskis

Patvirtinta elektroniniu būdu

Kauno technologijos universitetas

Informatikos fakultetas

Išmaniojo virdulio valdymo tyrimas naudojant įvairias belaidžio ryšio technologijas

Projekto tema	Išmaniojo virdulio valdymo tyrimas, naudojant įvairias belaidžio ryšio technologijas
Reikalavimai ir sąlygos	1. Sistema turi veikti naudojant LoRaWan, Bluetooth, *Wi-Fi* ryšį 2. Sistema turi veikti naudojant MQTT, Things Industries WebHook, metodu. 3. Ištirti LoRa esamas ir ateityje įmanomas Google Cloud integracijas
Vadovas / Vadovė	Šarūnas Packevičius 2023-04-17
	(vadovo pareigos, vardas, pavardė, parašas) (data)

Jonauskis, Aldas. Išmaniojo virdulio valdymo tyrimas naudojant įvairias belaidžio ryšio technologijas. Magistro baigiamasis projektas / vadovas / doc. dr. Šarūnas Packevičius; Kauno technologijos universitetas, Informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Informatikos mokslai

Reikšminiai žodžiai: LoRa, Bluetooth, LoRaWAN, *Wi-Fi*, ESP32, Google Cloud, Firebase,

Kaunas, 2023. 138 p.

Santrauka

Magistro baigiamajame darbe yra kuriama išmanaus virdulio sistema, naudojant belaidžio ryšio technologijas. Sistemos kūrimo procese tai pat atliekamas belaidžių ryšių technologijų tyrimas. Pagrindinės tyrinėjamos belaidžio ryšio technologijos – LoRa, Wi-Fi ir Bluetooth belaidžio ryšio technologijos. Tyrime bus apžvelgiamos esamos ir potencialios LoRa integracijos su Google Cloud ir nagrinėjama kaip keičiasi LoRa veikimas keičiant integracijos.

Jonauskis, Aldas. Smart Kettle Control Analysis using Wireless Network Technologies. Master's Final Degree Project / supervisor Dr., associate professor, Šarūnas Packevičius; Faculty of Informatics, Kaunas University of Technology.

Study field and area (study field group): Informatics

Keywords: LoRa, Bluetooth, LoRaWAN, *Wi-Fi*, ESP32, Google Cloud, Firebase

Kaunas, 2023. 138 pages.

Summary

In this master‘s degree project a smart kettle system will be created using various wireless network technologies. During the development phase of this project, an additional analysis will be done for wireless network technologies. The main technologies that will be analyzed – LoRa, *Wi-Fi*, Bluetooth wireless networks. The analysis will consist of determining existing and potential LoRa integrations with Google Cloud and how LoRa operating principle changes depending on these integrations.

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

SANTRUMPŲ IR TERMINŲ SĄRAŠAS

Santrumpos:

BLE (angl. *Bluetooth*) – Belaidės technologijos standartas naudojamas apsikeisti duomenimis;

Wi-Fi (angl. *Wireless Fidelity*) – Belaidžių tinklo protokolų šeima, paremta IEEE 802.11 standartų šeima.

SLAM (angl. *Short for long range*) – Žemos galios plačios srities protokolas;

Debesija (angl. *Cloud*) – Tinklas kompiuterinių įrenginių suteikiančių nuotolinį duomenų laikymą ir paslaugų apdorojimą internetu.

ESS – Nervų praplėsto aptarnavimo rinkinys (angl. An extended service set).

AP – Prieigos taškas (angl. *AP – Access point*).

Terminai:

6LoWPAN (angl. *6LoWPAN*) IPV6 adresas, esantis virš žemos galio belaidžio privataus lauko tinklo

Pikonet – Technologija, sujungianti belaidžių naudotojų grupes, naudojant Bluetooth technologiją.

Adhoc – Komunikacijos nustatymas, leidžiantis kompiuteriams tiesiogiai komunikuoti vienas su kitu.

Raktiniai žodžiai:
Keywords: LoRa, *Wi-Fi*, Bluetooth 4.2, smart kettle, drink preference, system base, BLE.

ĮVADAS

1 Įvadas

Dokumente pateiktas tarptautinėje verslo praktikoje naudojamas vertėmis grįstas požiūris (*angl. NABC- Needs, Approach, Benefits, Competition*) į projekto pasiūlymą. Kartu dokumentas yra ir Kauno technologijos universiteto Informatikos fakulteto Programų inžinerijos katedros studijų programos T000M241 baigiamojo darbo įvadas.

Pasaulyje nesustabdomai augant išmaniųjų technologijų rinkai, vis labiau auga poreikis viską automatizuoti. Šis procesas neprasilenkė ir su nuosavais namais. Nepaliaujamai augant poreikiui visus atskirus namų apyvokos įtaisus valdyti iš vienos centrinės lokacijos, namuose nebelieka vietos rankiniu būdu valdomiems buitiniams prietaisams ir siekiama paversti namą išmaniaisiais namais.

Išmanusis namas – tai visiško kompiuterizavimo taikymas, kuriame namų aplinka yra stebima aplinkos intelekto, siekiant suteikti kontekstą atpažįstančias paslaugas ir palengvinti nuotolinę namų kontrolę [1].

Vienas populiariausių būdų automatizuoti išmanų namą yra naudojant namų apyvokos priemones. Vienas geriausių visiško automatizavimo trūkumo pavyzdžių namų apyvokos sferoje yra rankiniu būdu valdomas virdulys. Šio prietaiso rankinis valdymas gaišina žmogaus brangų laiką, virdulio vandens virinimo procesas yra valdomas pagal asmens netobulą nuožiūrą arba pagal vieną nustatytą virdulio visiško įkaitinimo temperatūrą, kuri nėra visada idealiai tinkamai suderinama su žmogaus trokštamu gėrimu. Šio prietaiso automatizavimas ir prijungimas prie vieno centrinio tinklo atneštų vartotojui nemažai privalumų. Pirmiausiai, vartotojui nebereikėtų eiti iki virdulio ir rankiniu būdu jo įjungti – visa tai būtų galima atlikti iš mobiliosios programos. Taip pat iš tos pačios programėlės žmogus galėtų pasirinkti, kokio gėrimo trokšta. Taigi pagal pasirinkimą virdulys kaitintų vandenį iki tam gėrimui rekomenduojamos temperatūros. Sistemos sėkmingo veikimo atveju, būtų išspręstos dvi problemos – sutrumpinamas ir automatizuojamas vandens užvirinimo procesas ir taip pat virdulio kaitinamas vanduo bus geresnės kokybės gaminamo gėrimo atžvilgiu. Jei kuriama sistema sugebės išspręsti minėtas problemas vartotojui, yra pagrindo galvoti, kad išmanusis arbatinukas galės būti sėkmingai įsitrauktas į rinką. Taip pat šios sistemos kūrimas pasitarnaus tuo, kad bus galima tuo pačiu metu ištirti įvairias komunikacijos priemones, tokias kaip LoRa, Wi-Fi ir *Bluetooth* ir pilnavertiškai patikrinti jų veikimo galimybes bei panaudojimo vertingumą išmaniose sistemose. Tačiau tokio tyrimo sėkmė priklausys nuo sistemos pagrindo išplėtimo ir sukūrimo.

Projekto pasiūlymas yra suformuluotas pagal Šarūno Packevičiaus užsakymą ir bus įgyvendinamas Aldo Jonauskio kaip magistrinio darbo dalis. Projektą planuojama įgyvendinti iki 2022 metų birželio mėnesio. Sistemos rezultatas bus programos valdomas išmanusis virdulys, į kurį iš vartotojo mobilios programėlės įjungus trokštamo gėrimo komandą, kaitins vandenį iki pasirinkto gėrimo temperatūros. Taip pat planuojama į sistema įterpti LoRa, *Wi-Fi* ir *Bluetooth* protokolus, kad būtų galima juos ištirti ir nustatyti, kuris geriausiai tinka komunikuoti tarp sistemos mikrovaldiklio ir mobilios programėlės.

1.1. Poreikis

1.1.1. Projekto vartotojai ir klientai

Prietaisams tampant išmaniems, vis didesnė rinkos dalis susižavi automatizavimu: paprasti rinkos vartotojai, kompanijos. Jei sistema sėkmingai įsitrauks bent į vieną iš šių rinkų, tuomet atsiranda didelė tikimybė, kad produktas taps labai sėkmingu. Jei bus įsitraukta į buitinių vartotojų rinką, tuomet produktą bus galima panaudoti privačiuose namuose. Tai leis vartotojams sutaupyti laiko ir energijos kaitinant virdulį. Taip pat sistema gali būti panaudota įmonėse, kur darbuotojai jau dirbdami galės užkaitinti gėrimą per mobilią programėlę. Tai leis per pertraukėlę eiti pasiimti gėrimą ir tada turėti daugiau laisvo laiko nelaukiant gėrimo arba kaip tik grįžti į darbą anksčiau laiko ir taip padidinti kompanijos našumą. Tačiau ir be šitų rinkų yra potencialūs klientai – *Alexa, Google home* ir kt. Šie klientai, pasirašę sutartį, galės integruoti kuriamą sistemą į bendrą išmaniųjų namų ekosistemą, praplečiant jų galimybes.

1.1.2. Vartotojo problemos

Vis sparčiau augant automatizavimo tempams, vis didesnė žmonių dalis siekia valdyti visus prietaisus iš vienos bendros lokacijos. Tai itin opi problema namų apyvokos prietaisams. Vartotojai nori per savo programėlę ar kitą išmanųjį įrankį valdyti visus namų apyvokos prietaisus ir išvengti kuo daugiau rankinio valdymo. Ne išimtis ir virdulys. Įprastai vartotojai turi nueiti iki virdulio, įjungti jį ir tada nuojautos būdu jį sustabdyti, kuomet yra pasiekiama tinkama temperatūra planuojamam gėrimui. Šis kaitinimo būdas sukelia dvi problemas – reikia rankiniu būdu įjungti virdulį ir netiksliai jį sustabdyti kada manoma, kad temperatūra yra tinkama. Dar viena problema išryškėja labiau kai atsižvelgiama, kad ne visi vartotojai turi laiko laukti prie virdulio, kol tinkamai užkais vanduo.

Ir nors rinkoje yra sistemų, kurios automatizuoja virdulio kaitinimą, jos yra gana brangios ir dažnai turi esminių trūkumų.

1.1.3. Rinkos tyrimas

Išmaniojo virdulio sistema gali būti skirta trims pagrindiniams rinkos segmentams: išmaniojo namo, namų apyvokos įrankių valdymo sistemoms, kompanijoms ir / arba išmaniųjų namų savininkams. Namų apyvokos įrankių valdymo sistemų rinkoje galima įsitraukti pasiūlant jiems kuriamą sistemą integruoti į bendrą sistemą.

Išmanaus namo, namų apyvokos įrankių valdymo sistemos:

- *Alexa* – šiuo metu didžiausias namų apyvokos įrankių valdymo sistemos rinkos gaminys pasaulyje. Įmonė paskutiniame metų ketvirtyje pardavė apie 15,8 milijonų šios sistemos vienetų ir šios kategorijos sistema užima apie 28 3 % rinkos, 16 % prieaugis palyginus su praeitų metų paskutinio ketvirčio duomenimis (2019 paskutinio ketvirčio rinkos duomenys) [2].
    - *Google Nest* – antras didžiausias namų apyvokos įrankių valdymo sistemos rinkos atstovas pasaulyje. Įmonė paskutiniame ketvirtyje užfiksavo 13,9 milijonų šios sistemos pardavimų ir šios kategorijos užėmė 24,9 % rinkos, 20 % prieaugis palyginus su praeitų metų paskutinio ketvirčio duomenimis (2019 paskutinio ketvirčio rinkos duomenys) [2].
    - *Baidu* – Kinijos kompanija, kuri taip pat užsiima namų apyvokos įrankių valdymo sistemų gaminimu ir yra trečia pasaulyje pagal rinkoje užimamą vietą. Įmonė paskutiniame ketvirtyje pardavė 5,9 milijonų pardavimų ir užėmė 10,6 % rinkos (trečia pasaulyje) ir užfiksavo 171 % augimą palyginus su praeitų metų paskutinio ketvirčio duomenimis (2019 paskutinio ketvirčio rinkos duomenys) [2].
    - *Baidu* – dar viena Kinijos kompanija, kuri taip pat užsiima namų apyvokos įrankių valdymo sistemų gaminimu ir yra ketvirta pasaulyje pagal rinkoje užimamą vietą. Įmonė paskutiniame ketvirtyje pardavė 5,5 milijonų pardavimų ir užėmė 9,8 % rinkos (trečia pasaulyje) ir užfiksavo 94 % augimą palyginus su praeitų metų paskutinio ketvirčio duomenimis (2019 paskutinio ketvirčio rinkos duomenys) [2].

Reikia pastebėti, kad įsitraukti į šią rinką yra itin sunku, nes šios kompanijos yra globalios ir turi daugybę išteklių, todėl turi visas galimybes sukurti savo išmaniojo arbatinuko sistemą ir pasamdyti pačius protingiausius rinkos darbuotojus tą sistemą įgyvendinti. Į tai atsižvelgus, būtų neprotinga tiesiogiai konkuruoti su šių kompanijų kuriamomis sistemomis, todėl labiau verta su šiomis kompanijomis sudaryti kontraktą, kur jų kuriamos išmaniojo namo, namų apyvokos įrankių valdymo sistemos bus tiesiogiai suderinamos su kuriama sistema už mėnesinį mokestį.

Taip pat yra galimybė ir nesikoncentruoti į galimybę suderinti sistemą su esamais išmaniųjų namų rinkos atstovas, bet koncentruotis į atskirų vartotojų segmentą. Pagal 2020 metų gruodžio 10 duomenis, išmaniųjų mobiliųjų prietaisų vartotojų skaičius perkopė 3,8 milijardus [3]. Absoliuti dauguma iš jų turi arba *IOS* arba *Android* operacinėmis sistemomis paremtas sistemas [4]. O dėl to, kad kuriamos sistemos programa dirbs pagal *Android,* daug šių vartotojų taps potencialūs sistemos pirkėjai. Bet ir šiuo atveju verta atkreipti dėmesį, kad rinkoje jau yra kuriamos sistemos alternatyvos.

1.1.4. Informacija apie klientus

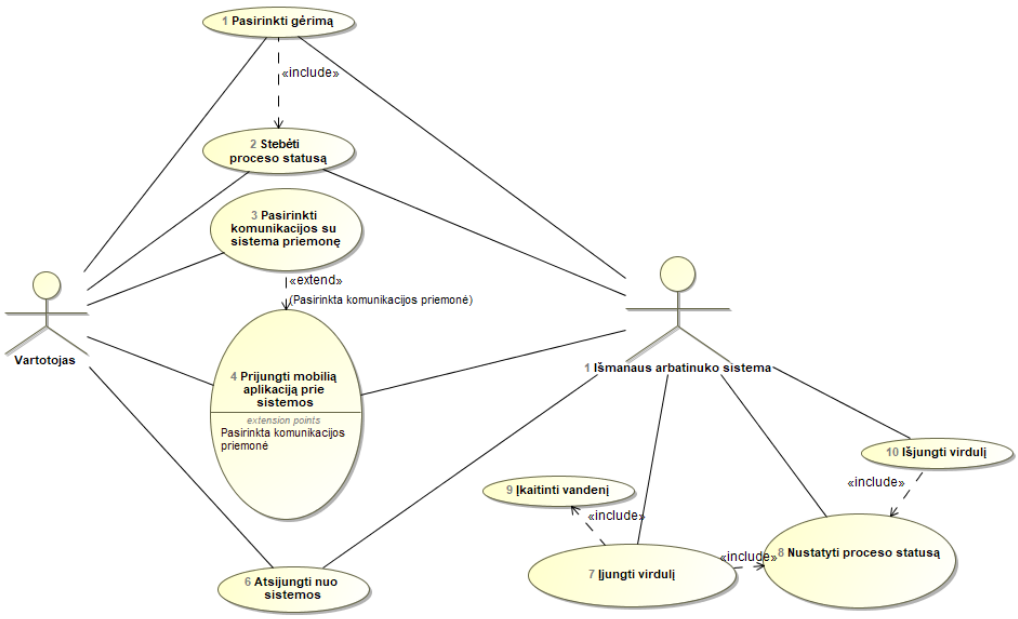
Potencialūs šios sistemos pirkėjai gali būti išmaniojo namo, namų apyvokos įrankių valdymo sistemų kūrėjai arba atskiros įvairių paskirčių kompanijos, arba patys vartotojai. Vartotojai –

individai, turintys išmanųjį įrenginį ir norintys jį automatizuoti. Kompanijos gali būti įvairaus pobūdžio, nes ši sistema bus naudojama darbuotojų komfortui ir galbūt net produktyvumo didinimui.

1.2. Pasiūlymas

1.2.1. Produkto ar paslaugos apibūdinimas

Apibūdinant sistemą, ji yra skirta ištirti *Wi-Fi, LoRa, Bluetooth* komunikacijos protokolus ir taip pat virdulyje užkaitinti gėrimą pagal vartotojo gėrimo temperatūros pasirinkimą. Sistemos detalesnės funkcijos yra aprašomos panaudojimo atvejų diagramoje (žr. 1 pav.).



pav. 1 Panaudojimo atvejų diagrama

Sistema bus valdoma dviejų aktorių: žmogaus (vartotojo) ir mikrovaldiklio (išmaniojo arbatinuko sistema).

Vartotojas – duoda komandas mikrovaldikliui vykdyti ir išpildyti, kurio rezultatas yra užkaitintas virdulys iki trokštamos temperatūros, prijungimas / atjungimas mobilios programėlės prie sistemos, arba virdulio virinimo statusas. Vartotojo panaudojimo atvejai:

- pasirinkti gėrimą;
- stebėti proceso statusą;
- pasirinkti komunikacijos su sistema priemonę;
- prijungti mobiliąją programą prie sistemos;

- atsijungti nuo sistemos.

Mikrovaldiklis – valdo sistemos atsaką į vartotojo užklausas ir stebi situaciją virdulyje:

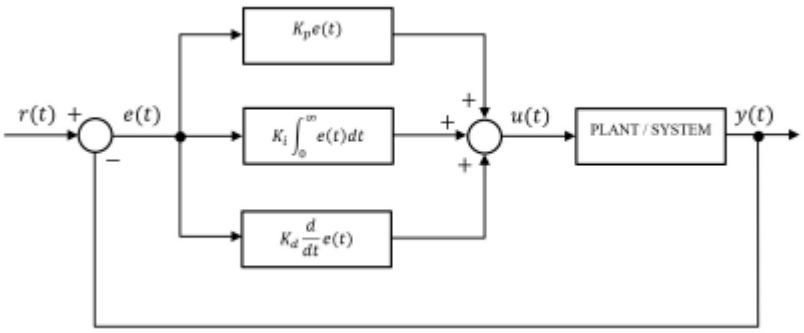
- įjungti virdulį;
- įkaitinti vandenį;
- nustatyti proceso statusą;
- išjungti virdulį;

1.2.2. Sistemos kontekstas

Kuriama sistema bus sudaryta iš šių elementų: mikrovaldiklio, maitinimo šaltinio, temperatūros sensorių, temperatūros matavimo skydo *Arduino* valdikliui, programėlės, komunikacijos protokolų ir valdymo algoritmo. Valdiklis dirbs kaip sistemos smegenys. Jis suteiks automatizuotą temperatūros skaitymą, temperatūros vaizdavimą ir šildymo valdymą ir leis perteikti duomenis į mobilią programėlę. *Arduino* mikrovaldiklis buvo parinktas, nes jis turi didžiulę biblioteką ir didelę aparatinės įrangos paramą.

Sistemai suteikiant automatinį vandens temperatūros valdymą, bus panaudota PID valdymo sistema, kuri buvo pasirinkta, nes su šia sistema galima tiksliai valdyti vandens temperatūrą su daug mažesniais nukrypimais nei kitų sistemų [5]

Taip pat ši technologija buvo pasirinkta, nes pramonėje daugiau nei 90 % visų valdymo grandinių yra proporciniai integraliniai išvestiniai (PID) [6] [7]. Jie naudojami, nes juos lengva gaminti, yra efektyvūs ir yra sukaupta itin daug valdymo veiksmų, žinių ir diskretinių PID realizacijos. Taip pat apie šį valdymo būdą yra daugybė literatūros, todėl toks valdymas yra itin naudingas [8].



pav. 2 PID valdymo blokinė diagrama [4]

Norint valdyti mikrovaldiklį ir duoti jam komandas reikės mobiliosios programėlės, iš kurios žmogus galės siųsti jam komandas ir atvaizduoti statusą vartotojui ir taip reikės komunikacijos protokolo sukurti komunikacijos ryšiui tarp mikrovaldiklio ir programėlės. Siekiant sukurti sistemą, kuri yra efektyvi ir itin greitai vykdo visas komandas, bus naudojami visi trys komunikacijos protokolai: *Wi-Fi, LoRa* ir *Bluetooth*.

Pagrindinės užduotys, kurias atliks mikrovaldiklis sistemoje:

- įjungs ir išjungs arbatinuką;

• matuos arbatinuko temperatūrą;
• nustatys vandens kaitinimo statusą (kaitinamas ar pabaigtas);
• siųs statuso duomenis į programėlę;
• atrinks ir prijungs komunikacijos priemonę, su kuria mobili programėlė kontaktuoja;
• priims iš programėlės komandas.

Aplikacijos užduotys, kurias vykdys, sukūrus sistemą:
• Rodys gėrimo statusą vartotojui ir esant poreikiui, siųs prašymą mikrovaldikliui rodyti statusą.
• Siųs gėrimo pasirinkimą mikrovaldikliui vykdymui
• Siųs užklausą prisijungti su vartotojo nurodyta komunikacija
• Atjungs sistemą

„*Android Studio integrated environment*“ sistemoje bus naudojamas, kaip įrankis, leisiantis sukurti Android programėlę, nes šis įrankis pasižymi:
• Šaltinio kodo redaktoriumi, kuris pasižymi galimybe pažengusiai pabaigti kodą, sutvarkymu (restruktūrizuoti), ir analizuoti kodą [9]
• Kodo statymo galimybe (angl. build) [9]
• Derintoju (angl. debugger), kurio pavadinimas yra Android derintojo stebėtojas [9]
• JAVA palaikymas [9]

Taip pat bus reikalinga Atmel Studio 7.0 aplinka, kuri bus naudojama programuoti vienareikšmiškai sistemą valdantį mikrovaldiklį. Ši technologija buvo pasirinkta, nes palaiko įvairius populiariausius mikrovaldiklius ir turi didžiulę nuolat pildomą biblioteką. [11]

Ši sistema savo pagrindinėje formoje nebus kitos sistemos dalis, bet pardavus šią sistemą namų apyvokos priemonių valdymų sistemų kompanijoms, jie šią sistema galės integruoti į savo bendrą sistemą. Vartotojui iš programėlės, pasirinkus trokštamą gėrimą per komunikacijos protokolą yra, siunčiama komanda mikrovaldikliui įvykdyti komandą. To pasėkoje, mikrovaldiklis kaitina vandenį ir siunčia vandens statusą programėlei.

1.2.3. Bendri apribojimai

Vartotojas nori pasirinkti sistemoje nenumatytą gėrimą. Šią situaciją galima spręsti vartotojui atsisakant gėrimo ir pasirenkant gėrimą, kurio rekomenduojama kaitinimo temperatūra yra panaši ar lygiavertė kitiems, jau sistemoje esantiems, gėrimams. Taip pat šią situaciją galima spręsti į sistemos mobilią programėlę integruojant gėrimo įtraukimo funkciją, kuria naudojantysis žmogus gali įtraukti gėrimą ir jo rekomenduojamą temperatūrą.

Sistemoje pasirinkti gėrimo temperatūra viršija maksimalią galimą – šią situaciją galima spręsti panaudojant aukštesnę temperatūrą matuojantį jutiklį, jei temperatūra viršija nustatytas ribas arba programėlėje nustatyti ribas, iki kokios temperatūros vartotojas gali kaitinti vandenį. Esant situacijai, kai viršijama mikrovaldiklio maksimali nustatyta temperatūra, tada šiuo atžvilgiu reikia

arba kelti mikrovaldiklio maksimalią priimamą temperatūrą jo programoje, arba vėl nustatyti temperatūrų ribas, iš kurių vartotojas rinkdamasis gėrimus negali išeiti.

1.2.4. Projekto įgyvendinimo planai ir kokybės vertinimas

Darbų seka ir terminai:

1. Projekto informacinė sistema (2020 – 10 – 13)
2. Projektavimas metodologijos ir technologijų analizė (2020 – 11 – 15)
3. Projekto paraiška (2020 – 11- 25)
4. Projektavimo planas (2020 – 12 – 15)
5. Reikalavimų specifikavimas (2020 – 03 – 10)
6. Architektūros specifikavimas (2020 – 04 – 15)
7. Darbų viešo pristatymo medžiagos paruošimas (2020 – 05 -03
8. Prototipas (2020 – 05 – 17)
9.Magistrinio projekto programų sistemos testavimo planavimas (2020 – 10 – 01)
10. Inžinerinis projektas – Programų sistemos prototipas 30% (2020 – 11 – (3 – 15))
11. Testavimo realizavimas – (2020 – 11 – 15)
12. Projekto kokybės tyrimas – (2020 – 12 – 05)
13. Inžinerinis projektas (2020 – 12 – 23)
14. Projekto ataskaita (2020 – 01 – 15)
15. Programų sistemos gynimas (2020 – 01 – 23)

Kuriamos sistemos kokybės įvertinimo būdai pateikiami 1 lentelėje:

lentelė 1 Sistemos vertinimo kriterijai

Nr.	Kriterijus	
1	Kaitinamas gėrimas iki mobiliojoje programėlėje pasirinkto gėrimo rekomenduojamą temperatūrą	Siekiama, kad kiekviena pasirinkto gėrimo komanda būtu 100 % tinkamai išpildoma, ir vanduo yra užkaitinamas pagal pasirinkto gėrimo rekomenduojamą temperatūrą
2	Teisinga užkaitinta temperatūra	Siekiama, kad užkaitinus vandenį pagal pasirinkto gėrimo užklausą, temperatūros skirtumas tarp trokštamos ir realios temperatūros būtų tik 2 -3 C°
3	Komunikacijos protokolų veikimas	Siekiama, kad sistema veiktų su *Wi-Fi,* LoRa ir *Bluetooth* komunikacijos protokolais
4	Universalumas	Nors sistema nebus integruojama į didesnę sistemą, siekiama, kad bent 70% sistemos programinio kodo, būtų galima perkelti į kitą sistemą

5	Gėrimo statusas mobiliojoje programėlėje	Aplikacijoje, turi visada rodyti teisingą gėrimo statusą pagal virduly kaitinamo vandens statusą (kaitinamas, užkaitintas)

1.3. Nauda

Šio prietaiso pagrindiniai konkurentai yra *ikettle, appkettle* ir *Fellow stag ekg* sistemas kuriančios įmonės. Šių įmonių parduodamas arbatinukas kainuoja 99£ (*ikettle*), £267.45 (*Fellow stag ekg*), £149.00 (*appkettle*), ir nors ir jų kainos nėra didelės (ypatingai *ikettle* ir *appkettle*), bet sukūrus šio projekto sistemą ir pardavus pakankamai sistemos vienetų 416.66 už 60 eur., bus galima susigrąžinti projekto savikainą + naujiems užsakymams reikalingos įrangos ir darbuotojų kaštus (planuojama, kad ši suma bus apie 25000) . Nuo tada, bus galima pardavinėti prietaisą 30% žemesne nei rinkos kaina. Taip pat yra galimybė parduoti sistemą vienam išmanaus namo, namų apyvokos įrankių valdymo sistemų gamintojams už 30000 eur. (taip susigrąžinant savikainą ir gauti papildomą atlygį) už galimybę integruoti kuriamą sistemą į gamintojo. Dar viena galimybė atsiras, jeigu į kuriamą sistemą bus integruojami kartu *Wi-Fi, Bluetooth* ir *LoRa,* leidžiantys vartotojui pasirinkti jam tinkamą komunikacijos protokolą su virduliu skirtingai nei konkurentai, kurie tokio funkcionalumo neturi ir siūlo tik vieną iš šių alternatyvų. Tai pasiteisins šiais atvejais:

- Jei klientai norės prietaiso, kuriam nereikia *Wi-Fi* interneto, jie galės pasirinkti *Bluetooth,* kuriai interneto nereikia

- Jei klientai nori greičio ir greitesnio duomenų perdavimo greičio jie galės tada pasirinkti *Wi-Fi* kaip komunikacijos protokolą.

- Jei klientai gyvena vietoje, kur yra didelis atstumas tarp vartotojo ir prietaiso, jie galės panaudoti *LoRa*. Šis scenarijus galimas pvz.: bendrabutyje, kur prietaisas yra apatinio aukšto bendroje virtuvėje, o vartotojas 5 aukšte.

Galimas variantas ir siūlyti produktą tik su vienu protokolu ir pardavinėti trijų rūšių produktus (su *Bluetooth,* su *Wi-Fi,* su *LoRa*) taip leidžiant pasirinkti pageidaujamą sistemą, ir galbūt sumažinti savikainą. Siekiama parduoti sistemos vienetų tokiu kiekiu, kad būtų galima susigrąžinti išlaidas + 20% produkto pajamų.

1.4. Konkurencija ir alternatyvos

Kaip atskleista praeitame skyriuje, šio prietaiso pagrindiniai konkurentai yra *ikettle,* appkettle ir *Fellow stag ekg*. Kompanijos *Smarter* sukurtas *Ikettle* yra trečios iteracijos virdulys, jį galima valdyti iš mobilaus įrenginio programėlės, esant bet kurioje namų vietoje. Šio virdulio temperatūrą galima valdyti pagal vartotojo norus ir galiausiai virdulys yra lengvai integruojamas į *Alexa* ir *Google* asistentų valdomus protingus namus, ko, verta paminėti, kuriama sava sistema pirmojoje iteracijoje nepasieks. Šis konkurento gaminys, galima paminėti, yra arčiausias kuriamai sistemai iš techninės pusės, bet vis dėlto, kuriamas virdulys turės ir keletą pranašumą, palyginus su konkurento produktu. Pirmiausiai, *Ikettle* yra Android operacinėms sistemoms skirtas įrenginys, kol tuo tarpu kuriamas

išmanusis virdulys bus valdomas Android mobilių įrenginių. Taip pat į kuriamą sistemą bus integruojami *Wi-Fi, Bluetooth* ir *LoRa*, leidžiantys vartotojui pasirinkti jam tinkamą komunikacijos protokolą su virduliu skirtingai nei *Ikettle*, kurio veikimas pagrįstas pagrinde *Wi-Fi* komunikacija. *Ikettle* taip pat yra reliatyviai brangus palyginus su kuriama sistema – *Ikettle* kainuoja apie 99£, o tuo tarpu kuriamą sistemą planuojama įkainoti apie per pusę tiek.

Fellowpruducts kompanijos *Fellow stag ekg* virdulys pasižymi dauguma jau minėtomis *Ikettle* savybėmis, bet ir gali, skirtingai nei kuriama sistema, išlaikyti iki valandos trokštamą temperatūrą, rodyti LCD ekrane siekiamą ir užkaitintą vandens temperatūrą, bei leidžia celcijiniu tikslumu reguliuoti užkaitinimo temperatūrą. Skirtingai nei Ikettle ir kuriama sistema, šis prietaisas nėra valdomas iš mobilios programėlės, todėl žmogus turi prieiti prie prietaiso ir pasinaudojant prietaise integruotu valdikliu pasirinkti ir užkaitinti vandenį iki trokštamos temperatūros. Prietaiso kaina, esanti 129£, taip pat nėra pati patraukliausia, palyginus su planuojamos sistemos 60 eur įkainiu.

Appkettle kompanijos *Appkettle* virdulys, dar vienas kuriamos sistemos konkurentas funkciškai visiškai atitinka Ikettle. Dėl šios priežasties, *Appkettle* turi ne tik *Ikettle* tyrime išrinktus pranašumus prieš kuriamą sistemą, bet ir visus *Ikettle* turimus trūkumus palyginus su kuriama sistema – galima paminėti, kad neturi *Bluetooth*, palaiko tik Android sistemą, *Wi-Fi* ir yra brangus.
Apžiūrėjus visus pagrindinius konkurentus, buvo pastebėta, kad kuriama sistema, nors į rinką dar nėra paleista, vis dėlto, pasižymi kai kuriais pranašumais palyginus su jau minėtais rinkoje pasireiškusiais konkurentais. Dėl šios priežasties, egzistuoja itin geros perspektyvos įsitraukti į rinką.

Kaip anksčiau pasakota, išmanaus namo, namų apyvokos įrankių valdymo sistemų gamintojai (Amazon, Google), gamina sistemas, kurios valdo namų apyvokos prietaisus ir, galima teigti, yra sudarę monopolijas rinkose. Jie kuriamai sistemai, nors ir yra toje pačioje rinkoje, pagrindiniu atžvilgiu yra ne konkurentai, o potencialūs partneriai arba sistemos klientai. Nors jie turi išmanaus virdulio sistemas, ne visi vartotojai būtent turės ar naudos didžiųjų kompanijų išmanaus virdulio sistemas. Todėl šioms valdymo sistemoms itin būtinas suderinamumas su kitomis sistemomis. Yra nebloga galimybė už mokestį iš šių bendrovių, sukurti galimybę jų valdymo sistemoms prisijungti prie šiame projekte kuriamos sistemos ir taip padidinti susietumą su kitomis programomis.. Tokiu atžvilgiu, ateis pajamos ne tik iš sistemos pirkėjų, bet ir licencijos mokestis iš valdymo sistemų gamintojų. Taip pat šios kompanijos galėtų ir nusipirkti sistemą, kaip alternatyva, jų jau turimoms sistemoms. Todėl galima teigti, kad sistemai yra ne vienas kelias prasibrauti į rinką.

ANALITINĖ DALIS

2 Tikslas

Darbo tikslas – Sukurti išmaniojo virdulio sistemą valdomą programėlės, kuri skirtingai kaitintų pagal naudotojo planuojamo gėrimo poreikį ir ištirti sistemą ir programėlę jungiančius komunikacijos protokolus. To pasėkoje vartotojas galės iš programėlės užkaitinti jam tinkantį gėrimą ir valdyti sistemą su efektyviausiu komunikacijos protokolu.

Sistema susidės iš virdulį valdančio mikrovaldiklio. Šis valdiklis bus užprogramuotas C programine kalba ir tokiu būdu bus sudaryta efektyviausia (pagal tyrimo rezultatus) komunikacija tarp sistemos ir kuriamos programėlės. Ši sistema nebus priklausoma nuo išorinių sistemų, bet bus aptariamos sistemos integravimo galimybes į išmanaus namo sistemą.

2.1 Komunikacijos protokolų apžvalga kuriamai sistemai

Daiktų internetas pasaulyje yra įsivaizduojamas kaip didelis tinklas fizinių prietaisų, kuriame internetas išsiplečia į fizinį pasaulį apimančius kasdienius prietaisus, kurie prasideda nuo paprasto vienatinio prietaiso iki didžiulio tarp-platforminių įterptinių technologijų ir „Cloud“ sistemų, kurie tarpusavyje komunikuoja realiu laiku [1]. Bet koks daiktų interneto kūrimo klausimas prasideda nuo komunikacijos sistemų išrinkimo. Šiame darbe bus apžvelgiamos šios technologijos kuriamai sistemai: *Bluetooth* 4.2 (BLE), protokolas 802.11 (*Wi-Fi*) ir *LoRa*.

Mažai energijos eikvojantis *Bluetooth* (BLE) buvo pristatytas kaip viena geriausių alternatyvų daiktų interneto rinkoje. BLE reklamuojamas kaip išmanus *Bluetooth* ir yra mažai energijos eikvojantis klasikinio *Bluetooth* variantas nukreiptas į mažai energijos ir resursus eikvojančius prietaisus . Po BLE pristatymo rinkoje įvyko pilnavertis *Bluetooth* 4.0.3 technologijos įtraukimas į *Bluetooth* technologiją turinčių prietaisų architektūrą [2].

802.11 protokolas su a/b/g/n/ac technologijos versijomis yra vienas iš pagrindinių akivaizdžių technologinių kandidatų, kai kuriama daiktų internetų sistema. Šiandien, *Wi-Fi* yra de facto naudojama technologija, kai turima omenyje prisijungimą prie interneto iš belaidžio prieigos taško. Ši sistema yra labiausiai naudojama kai kalbama apie namų apyvokos prietaisus, išmanaus namo saugumo sistemas ir belaidį internetą. Šiais laikais praktiškai kiekvienas privatus namas, darbo vieta, kavinė ir universitetas turi nuosavą *Wi-Fi* tinklą. [3]

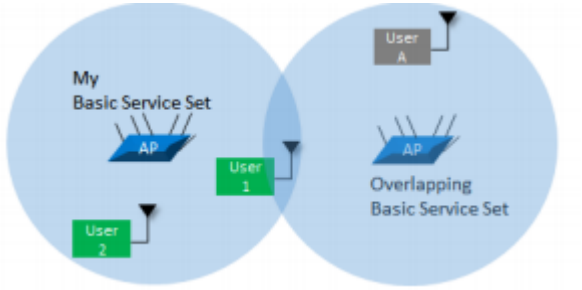
LoRa yra ilgos distancijos mažai energijos naudojanti belaidė technologijos platforma, kuri naudoja nelicencijuotą radijo spektrą industrinėje, mokslinėje ir medicininėje radijo juostoje [4]. *LoRa* išradimo tikslas buvo pašalinti kartotuvus, sumažinti prietaisų kainas, padidinti baterijos gyvavimo laiką prietaisuose, išplėsti tinklo dydį ir palaikyti didžiulį kiekį prietaisų. Tai yra fizinis sluoksnis naudojamas ilgo nuotolio komunikacijoje [5]

2.1.1 Analitiniai modeliai

Didžioji dalis IOT prietaisų yra paremti 3 didžiausių rinkoje komunikacijos protokolų modelių – *Wi-Fi, Bluetooth, LoRa*. Ne išimtis bus ir kuriama išmanaus virdulio sistema. Todėl šiame skyriuje bus detaliai aprašomi populiariausi analitiniai komunikacijos protokolų modeliai naudojami IOT prietaisuose.

2.1.1.1 Wi-fi komunikacijos protokolas

Wi-Fi, kurio pilnas pavadinamas yra belaidis nelicencijuojamas ryšys (*wireless fidelity*) yra paremtas IEEE 802.11 šeimos standartais ir yra pagrinde vietinio lauko tinklo technologija sukurta suteikti pastato viduje plačiajuosčio radijo signalų sistemos veikimo zoną su duomenų perdavimo greičiu 54 Mbps. Ši technologija tipiškai padengia viduje esančią veikimo zoną apie 100 metrų spinduliu. Visi belaidžiai prietaisai, kurie prisijungia prie *Wi-Fi* tinklo, jie sudaro paprastų paslaugų rinkinį (*BSS*). Plačiau BSS tai yra rinkinys belaidžių stotelių (*STAs),* ar mobilių, nešiojamų ar fiksuotų valdomų vienos koordinacinės funkcijos (*CF*), kuri nusprendžia kada STA perduos ir kada gaus duomenis [2]. Taip pat dideliuose *Wi-Fi* tinkluose daugybe BSS gali būti sujungti panaudojant skirstymo sistemas (DS) taip suteikiant pakankamą veikimo zoną didesniam kiekiui stotelių. Ši konfigūracija turinti dvi ar daugiau BSS yra vadinama praplėsto aptarnavimo rinkinys (ESS). DS yra laidinė pagrindinė jėga sujungianti prieigos taškus (AP) ir leidžianti kartu veikiančioms stotelėms prisijungti prie paslaugų kurios buvo ankščiau prieinamos tik laidinėje architektūroje [4]

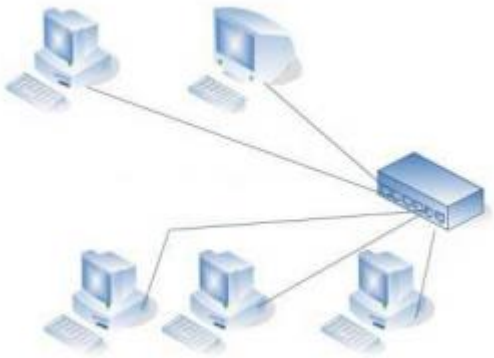


pav. 3 supaprastintas statybinis Wi-Fi modelio blokas [2]

Galima taip pat paminėti, kad neseniai, *Wi-Fi* propagavimas kaip plėtinio laidinių tinklų paaugo dramatiškai ir todėl tai suteikė prietaisams gebančius dirbti su belaide sąsaja daug didesnį mobilumą. DS yra pagrindinė laidinė jėga sujungianti prieigos taškus (AP) ir leidžianti kartu veikiančioms stotelėms prisijungti prie paslaugų ankščiau prieinamų tik laidinėje architektūroje. [3]

Todėl, *Wi-Fi* prietaisai gali suformuoti žvaigždės topologiją su jos prieigos taškais atliekančiais interneto vartų vaidmenį. Verta paminėti, kad *Wi-Fi* išėjimo jėga yra didesnė nei panašių belaidžių komunikacijos protokolų kaip *Bluetooth*, LoRa, bet reikia, kad būtų pasiekta pilna interneto prijungimo veikimo zona kitaip gali atsirasti negyvų zonų, kur *Wi-Fi* ryšys nesuveiks. Šią problemą

dalinai galima išspręsti panaudojant daugiau nei vieną anteną prieigos taškuose. *Wi-Fi* dirba 2.4 GHz ir 5 GHz darbo ruožuose.



pav. 4 Wi-Fi protokolo žvaigždės topologijos pavyzdys [6]

Wi-Fi operacija 5 GHz juostoje leidžia panaudoti daugiau kanalų ir suteikia didesnį duomenų perdavimo greitį. Tiesa, atstumas 5 GHz radijo pastato viduje yra trumpesnis negu 2.4 GHz. IEEE 802.11b ir IEEE 802.11g dirba 2.4 GHz industrinio, mokslinio ir medicininio dažnių juostoje (ISM). IEEE 802.11n patobulėjo nuo ankstesnių versijų įterpdamas daugybės įėjimų ir daugybės išėjimų metodus (MIMO) [7]. Šis standartas palaiko duomenų perdavimo greitį nuo 54Mbit/s iki 600 Mbit/s [8]. Tuo tarpu IEE 802.11ac yra patobulinta versija IEE 802.11n. Jis suteikia didelio pralaidumo belaidžius vietinės zonos tinklus (WLANs) 5 GHz juostoje su didesniais erdviniais srautais ir didesne moduliacija su MIMO atiduodančiais duomenų perdavimais siekiančiais 433.33 Mbps [9].

2.1.1.2 Bluetooth komunikacijos protokolas

2.1.1.2.1 Supažindinimas

Bluetooth yra trumpos distancijos, mažai energijos IEEE atviro standarto sukuriantis belaidžius asmeninės zonos tinklus. *Bluetooth* dirba nelicencijuojame 2.4 GHz trumpos distancijos radijo dažnio spektre skirtame trumpos distancijos komunikacijos prietaisams. Šis spektras yra tinkamas pakeisti laidus spausdintuvams, faksams, valdymo pulteliams, kompiuterio pelėms, klaviatūroms ir t.t. Prietaisai gali būti skirti komunikacijai tarp nešiojamų kompiuterių, tiltai tarp kitų tinklų, arba dirbti kaip mazgai *adhoc* tinklų. Toks diapazonas aplikacijų yra žinomas kaip vietinis asmeninės zonos tinklas. *Bluetooth* 4.2 išleistas 2014 gruodį žinomas kaip BLE tapo pagrindinis protokolas IOT prietaisams. BLE pašalino vartų tokių kaip 6LoWPAN ribinių maršrutizatorių [10] reikalavimą sujungti *Bluetooth* LE prietaisus su internetu. Tai reiškia, kad galima lengvai panaudoti bet kokius *Bluetooth* LE palaikomus išmaniusiuosius telefonus. *Bluetooth* LE gali veikti kaip vartai į internetą.

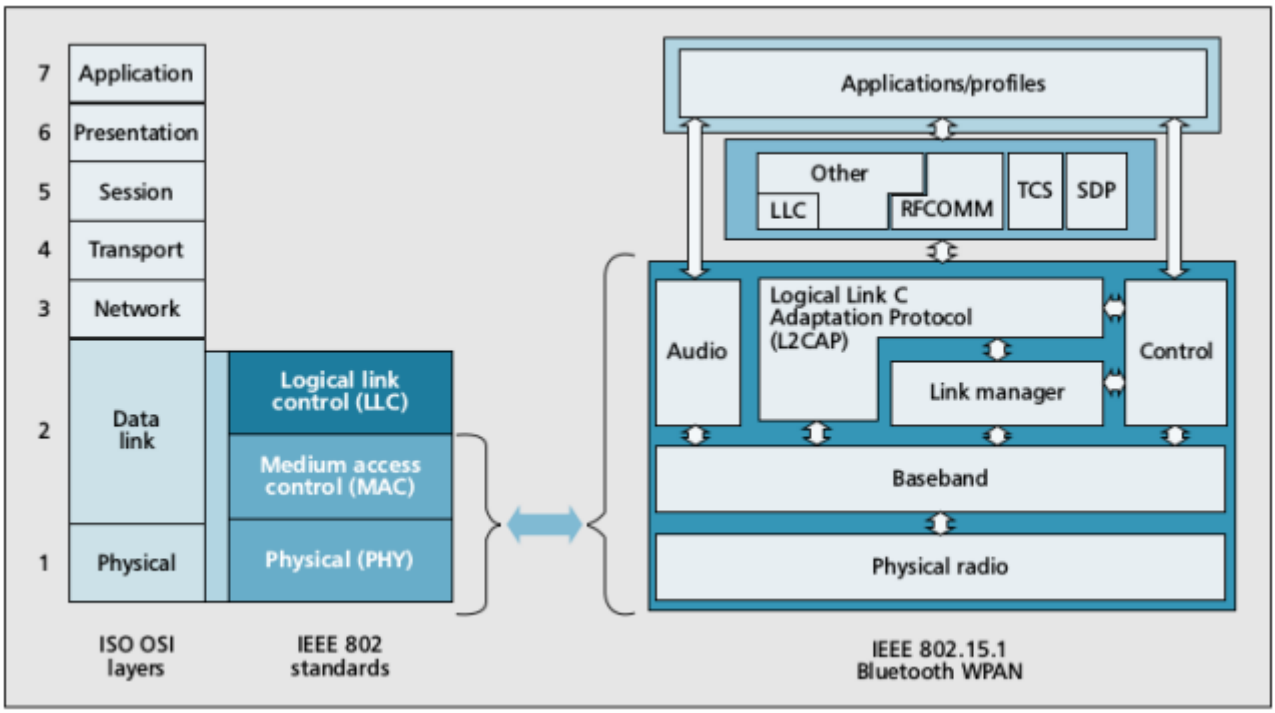
2.1.1.2.2 Veikimo principas

Kai *Bluetooth* prietaisas yra įjungiamas, jis gali dirbti kaip vienas iš valdomų prietaisų, gaunantis komandas iš valdančio prietaiso. Šis pavaldus prietaisas tada pradeda klausytis valdančio prietaiso užklausos apie pridėtus naujus prietaisus ir atsako į minėtą užklausą. Užklausos fazė leidžia vadovaujančiam prietaisui sužinoti valdomo prietaiso adresą. Verta paminėti, kad ši fazė nėra reikalinga elementariai sujungtiems prietaisams, kurie iš karto gauna vienas kito adresą. Kai

vadovaujantis prietaisas sužino valdomo elemento adresą, jis gali atidaryti ryšį su juo jei valdomas prietaisas klausosi prisijungimo užklausų. Jei tai įvyksta, tada valdomas elementas atsako į valdančio prijungimo užklausą ir šie du prietaisai sinchronizuoja virš dažnio šuolio sekos, kuri yra unikali kiekvienam pikonetui ir kuri yra nusprendžiama valdančio elemento. Taip pat *Bluetooth* iš anksto aprašo įvairius prijungimo būdus, kiekvienas su skirtinga kombinacija pralaidumo, apsauga nuo klaidų ir aptarnavimo patikimumo. Kai tik junginys yra sudaromas, prietaisai savo ruožtu gali patvirtinti vienas kitą ir tada komunikuoti. Prietaisai, kurie nėra suderinti, į transmisijas gali įeiti į vieną iš keleto galią ir pralaidumą saugančių režimų arba tiesiog nutraukti ryšį. Galima taip pat paminėti, kad valdantis prietaisas ir valdomas elementas, gali susikeisti rolėmis, kai prietaisas nori dalyvauti daugiau negu viename pikonete.

2.1.1.2.3 Protokolo apžvalga

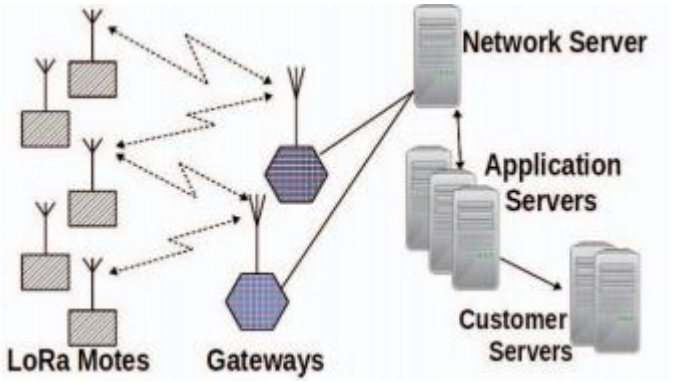
Bluetooth ne tik aprašo radijo sąsaja, bet taip pat leidžia prietaisams surasti vienas kitą ir aprašyti jų atliekamas funkcijas. *Bluetooth* MAC protokolas yra sukurtas sudaryti *adhoc* tinklus, be rankinės konfigūracijos, laidų ir laidinės infrastruktūros. Jis yra paremtas ne pagal eiliškumo sprendimą kaip tradiciniuose belaidžiuose LAN, bet pagal valdančio ir valdomo elementų ryšį. *Bluetooth* pikonetas yra sudarytas iš vieno valdančio prietaiso ir iki septynių valdomų elementų. Valdantis elementas skiria perdavimo lizdus (ir tuo pačiu kanalo pralaidumą) valdomiems elementams pikonete [11]. *Bluetooth* radijo modelis naudoja mažos galios radijo siųstuvus-imtuvus dirbančius nuo 10m iki 100m diapazone ir, kurie dirba 2.45Ghz dažnyje. Galima radijo 83.5 MHz juosta yra padalinama į 79 kanalus, kurie yra po 1Mhz pločio [12]. *Bluetooth* panaudoja vieną kanalą ir kanalas yra keičiamas 1600 kartų ryšium pagal kanalų šuolių schemą išgautą iš pikoneto valdančio elemento adreso [13] .



pav. 5 Supaprastintas statybinis Wi-Fi modelio blokas [13]

2.2.1.3 LoRa ir LoRaWAN komunikacijos protokolas

LoRa belaidė technologija naudoja čiulbesio – plitimo – spektro (CSS) moduliacija su plitimo faktorių ir pralaidumo pasirinkimais, kad būtų galima optimizuoti moduliaciją ir pasiekti optimalų komunikacijos diapazono ir duomenų reikalavimus. *LoRa* naudoja ISM (juosta skirta pramoniniams, moksliniams ir medicininiams tikslams) juostas 433MHz, 868MHz arba 915 MHz priklausomai nuo jurisdikcijos juosta yra padalinama į kanalus. SF ir pralaidumo kombinacija iškeičia greitį į diapazoną. Europoje perduodama galia yra limituota iki 14dBm maksimalaus dydžio galios, kuri gali būti išspinduliuota iš antenos (EIRP) su vieno procento ore darbo ciklu. Priklausomai nuo moduliacijos transmisijos galia gali pasiekti iki 156 dB [5].
LoRaWAN (LoRa sąjungos prekės ženklas) panaudoja protokolų šūsnį, kuriame belaidė *LoRa* yra fizinis sluoksnis. *LoRaWAN* taškas komunikuoja gyvai su vartais, kurie inkorporuoja imtuvo koncentratorių (dekoderį) galintį dekoduoti 10 kartų pasiųstas transmisijas. Vartai taip pat komunikuoja su „Tinklo serveriu“.



pav. 6 LoraWAN *topologija. Taškai gali priklausyti bet kam [5]*

Jei *LoRaWAN* taško transmisija yra aptinkama, tai tinklo serveris nusprendžia, kuriuos vartus panaudoti pripažinimo išsiuntimui (jeigu reikia). Tinklo serveris pasiunčia duomenų paketą į „Aplikacijos Serverį“ ir programėlės serveris persiunčia duomenis į „Kliento Serverį“. Kai naudojamas per orą (angl. over the air) autentikavimo metodas, autentikavimas įvyksta programėlės serveryje. Programėlės serveris leidžia sugrupuoti daugybę taškų į „programėlė“. Tuo tarpu nors klientas gali turėti daugybę ir skirtingų rūšių taškų skirtingose lokacijose, „programėlė“ yra tik administracinis grupavimas panašių taškų. „Programėlė“ taip pat turi savo šifravimo raktą.

2.2.2 Taikomos technologijos, jų veikimo principai ir apribojimai

2.2.2.1 Pagrindinės technologijos

Šioje dalyje bus aprašomos technologijos, kurios bus naudojamos realizuoti kuriamai sistemai. Sistemos Android mobilios programėlės dalies kūrimui bus naudojama Android Studio integruota realizavimo aplinka. Ši aplinka buvo pasirinkta nes ji yra pagrindinis įrankis naudojamas kurti Android programėlės. Šis įrankis pasižymi:

• Šaltinio kodo redaktoriumi, kuris pasižymi galimybe nuspėti planuojamą funkciją, refaktoravimą (restruktūrizuoti), ir taip pat gali analizuoti kodą.[14]
• Kodo paruošimo galimybe (angl. *build*) [14]
• Derintoju (angl. *debugger*), kurio pavadinimas yra Android derintojo stebėtojas [14]

Taip pat šis įrankis buvo pasirinktas nes jis palaiko JAVA ir iš bėdos c++ kalbą, kurios jau yra suplanuotos kaip pagrindinės sistemos kalbos [14]

2.2.2.2 Papildomos technologijos

Mikrovaldiklis bus ESP32 mikrovaldiklis, iš jo nėra tikimasi didelės galios, bet didelio integralumo dėl to šis mikrovaldiklis ir buvo parinktas kuriamai sistemai. [17]
Virdulio temperatūrai matuoti bus naudojamas temperatūros sensorius, kuris yra suderinamas su mikrovaldikliu. Dėl to, kad mikrovaldiklis, kuris valdys sistemą jau yra išrinktas ESP32, sistemai bus naudojami DS180B20 temperatūros sensoriai, kurie yra lengvai suderinami su ESP32 mikrovaldikliu. Komunikuoti sistemai su mobilia programėle reikės *Bluetooth, Wi-Fi* ir *LoRa* komponentų ir reikės juos pridėti prie ESP32 mikrovaldiklio. *Bluetooth* komunikacijos sukūrimui bus naudojamas populiariausiai rinkoje naudojamas *HC-05 Bluetooth* modulis, kuris yra suderinamas su ESP32 prietaisu. *Wi-Fi* komunikacijai sukurti bus naudojamas Espressif ESP8266 *Wi-Fi* modelis, kuris vėlgi yra suderinamas su ESP32 ir lengvai integruojamas į sistemą. Ir galiausiai *LoRa* komunikacijai sukurti, bus naudojamas *Sparkfun* 1 kanalo siųstuvas/vartai modulis. Jis nėra taip lengvai integruojamas į ESP32 kaip *Bluetooth* ir *Wi-Fi*, bet šis modulis yra geriausias pasirinkimas norint integruoti *LoRa* protokolą.

2.2.3 Naudojami skaičiavimo metodai

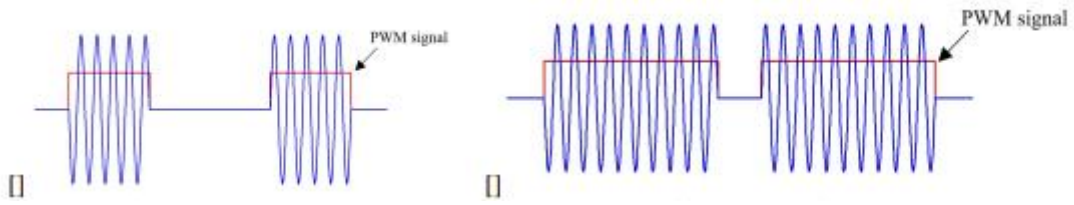
2.2.3.1 Grandinės metodo aprašymas

Kuriama sistema bus sudaryta iš šių elementų: mikrovaldiklio, maitinimo šaltinio, temperatūros sensorių, temperatūros matavimo skydo ESP32 valdikliui, programėlės, komunikacijos protokolų ir valdymo algoritmo. Valdiklis dirbs kaip sistemos smegenys. Jis suteiks automatizuotą temperatūros skaitymą, temperatūros vaizdavimą ir šildymo valdymą. ESP32 mikrovaldiklis buvo parinktas, nes jis turi didžiulę biblioteką ir didelę aparatinės įrangos paramą [17]

Kad būtų paleista temperatūros kontrolė arbatinuko viduje, bus panaudojamas temperatūros sensorius. DS180B20 naudojamas sensorius, kuris yra tinkamas naudoti vandenyje ir gali dirbti su ESP32 mikrovaldikliu. Maitinimo šaltinis jutikliui bus 3.0 – 5V ir galintis matuoti temperatūrai nuo -55 ◦C iki 125◦C [18]

Šildymo įjungimo grandinei, bus naudojamas kietos būsenos relinis jungiklis nes lengva sukurti sąsają tarp mikrovaldiklio ir kintamosios srovės šaltinio. Reikia pastebėti, kad paveikti SSR ir valdyti kintamosios srovės įtampą, žemo dažnio PWM (0.1 kintamosios srovės dažnio) turėtų būti

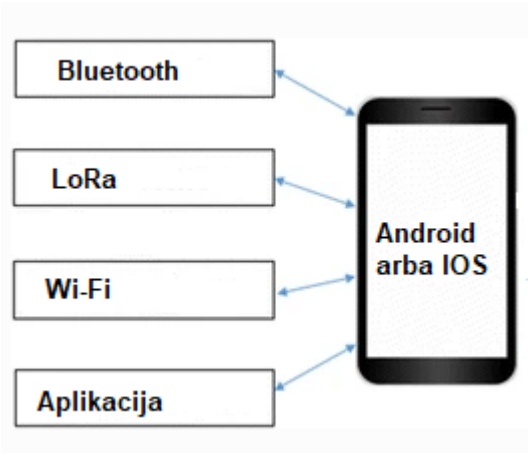
naudojamas, kad būtų gaunamas vienodas ciklas kintamosios srovės kaip parodyta (žr. 7 Pav.).. Aukšto dažnio PWM verčianti SSR privers kintamosios srovės nevienodus fazės kampus judėti, taip padarant tiesinius ryšius tarp virdulio jėgos ir PWM signalo neįmanomu. (7-8 pav.). parodo kintamosios srovės išvestį kai yra valdomas žemos ir aukštos PWM dažnio maitinantis SSR. Šiame darbe, žemo dažnio PWM gali būti panaudotas vandens temperatūros kontrolei nes vandens temperatūra turi lėtus žingsnio atsakus. [18]



pav. 7 Žemo dažnio PWM valdantis SSR relinį jungiklį sukurs kintamosios srovės ciklą išeitį su (a) 40% darbo ciklu b) su 80% darbo ciklu [18]

2.2.3.2 Pirmo metodo valdyti temperatūrą aprašymas

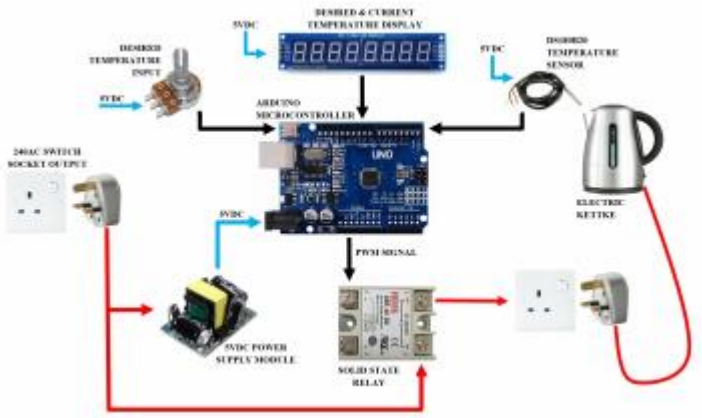
Mobilieji telefonai turi įvairius sensorius ir funkcijas, kurias jie gali atlikti. Tai pats svarbiausias įtaisas norint gauti komandas iš naudotojo. Šiame darbe, mobilus telefonas gali būti naudojamas, kaip komandas duodantis prietaisas ir nurodantis kokia temperatūra mikrovaldiklis dirbs. Šie duomenys/komandos yra siunčiami į serverį (mikrovaldiklį) apdorojimui [19]



pav. 8 Pagrindinės funkcijos, kurias atliks mobilus įtaisas [19]

2.2.3.3 Antro metodo valdyti temperatūrą aprašymas

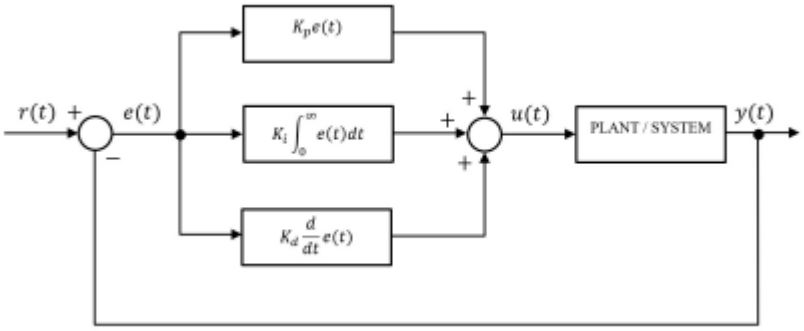
Potenciometras yra naudojamas, leisti naudotojui pasirinkti trokštamą momentinę temperatūrą virdulyje. Valdymo algoritmui, PID valdiklis yra naudojamas suteikti automatizuotą temperatūros kontrolę, kur PID yra užkoduotas mikrovaldiklyje. PID valdiklio detalės ir jo dizainas yra aptariami kitame skyrelyje. Pavaizduota sistemos grandinė (žr. 9 pav.). rodo diagramą mikrovaldiklį sujungtą su įėjimu ir išėjimo komponentais.[18]



8 Pav. sistemos grandinė [18]

2.2.3.4 PID valdiklio aprašymas

PID yra proporcinis integrinis valdiklis, kuris valdo ciklo grįžtamojo ryšio mechanizmą. Tai dažnai naudojamas prietaisas pramoniniuose valdymo sistemose. Pirmas PID valdiklis buvo sukurtas Elmerio Sperio 1911 metais ir, kuris buvo panaudotas automatizuoti laivo valdymą [20]. PID blokinė diagrama (žr. 10 pav.). Paveiksle yra pavaizduota blokinė diagrama sistemos valdomos PID valdiklio, kur y(t) yra valdomas išėjimas, r(t) yra etaloninis signalas, e(t) yra klaidos signalas ir u(t) yra valdymo signalas. [18]



pav. 9 PID valdymo blokinė diagrama [18]

PID valdymo signalas yra suma trijų kintamųjų, kurie yra paremti šiuo klaidos matavimu:

$$u(t) = K_p + K_i \int_0^\infty e(t)dt + K_d \frac{d}{dt} e(t) \quad (1)$$

kur, e(t) = r(t) – y(t), K_p yra proporcinė konstanta, K_i yra integralinė konstanta ir K_d yra išvestinė konstanta. K_p, K_i, K_d reikšmės yra svarbiausios norint suteikti stabilų ir trokštamą trumpalaikį atsaką, kuris gali būti gaunamas naudojant euristinius metodus, analitinius metodus, dažnio atsako metodą ir optimizacijos metodą [21].

Šiais laikais, PID valdymo algoritmas yra visados įgyvendintas mikrovaldiklyje, kad būtų galimas kompaktiškas produkto dizainas. Tiesa, integralinė ir išvestinė PID (1) formulės operacija negali būti taip atlikta mikrovaldiklio. Todėl, integralinė ir išvestinė sąlyga (1) turi būti perkeista į diskretinę formą. Pradedant su išvestine forma, mes galime panaudoti sekančią išvestinė formulę kaip aproksimaciją [18]:

$$\frac{d}{dt}e(t) = \frac{e(k)-e(k-1)}{T_s} \quad (2)$$

e(k) yra klaidos signalas diskrečiojoje srityje, e(k-1) yra ankstesnis signalas ir T_s yra ėminių ėmimo laikas. Formulė (2) yra apytikslis nuolydis tangentinės linijos e(t). Apytikslė integralinė sąlyga gali būti užrašyta šitaip [18]:

$$\int_0^\infty \quad e(t)dt \approx T_s \sum_0^\infty \quad e(k) \quad (3)$$

Šių aproksimacijų pagalba, mes galime perrašyti PID valdymo algoritmą diskretinėje formoje [18] :

$$u(k) = K_p e(k) + K_i\left(T_s \sum_0^\infty \quad e(k)\right) + K_d\left(\frac{e(k)-e(k-1)}{Ts}\right) \quad (4)$$

Kernelio kodo įgyvendinimas diskrečiajame PID (4) formulės *Arduino* IDE aplinkos programinė įranga yra parodyta 8 figūroje. *Arduino* IDE yra programinė įranga sukurta *Arduino*, ESP32 ir kitiems mikrovaldikliams, kuris leidžia naudotojui parašyti kodą C kalba, sukompiliuoti, išsiųsti/priimti į/iš mikrovaldikliui. PID reikšmės K_p, K_i, K_d yra parametrai, kurie turi būti sureguliuojami. Kuriamoje sistemoje, euristinis būdas bus naudojamas nustatyti tinkamas vertes PID stiprinimui. Procedūrą nustatanti PID stiprinimą yra aprašoma taip:

(a) Nustatyti stiprinimą į 0.
(b) Didinti K_p stiprinimą, kol atsakas pradeda osciliuoti.
(c) Didinti K_d stiprinimą, kol osciliavimai pradeda silpnėti.
(d) Karoti (b) ir (c) žingsnius kol atsakas yra stabilus su minimaliai osciliavimais.
(e) Didinti K_i stiprinimą, kad priartinti atsaką iki nustatyto taško.

```
unsigned long Now;
unsigned long LastTime;
double SamplingTime=0;
double Error;
double SumError = 0;
double RateError;
double LastError = 0;
void PID(float setpoint, float output, float kp,
float ki, float kd)
{
Now = millis();
SamplingTime = (double)(Now - LastTime)*pow(10,-3);
Error = setpoint - output;
SumError = SumError + (Error*SamplingTime);
RateError = (Error - LastError)/SamplingTime;
LastError = Error;
LastTime = Now;
Output = kp*Error + ki*SumError + kd*RateError;
return Output;
}
```

pav. 10 PID valdymo algoritmo kodas [18]

2.2.4 Atrinkti matematiniai metodai , technologijos, sprendimai ir /ar programiniai analogai

Taigi atlikus technologijų apžvalgą pareitame skyriuje, buvo sudarytas sąrašas elementų, kurie bus naudojami, kuriant sistemą:

lentelė 2 Naudojamų technologijų sąrašas

Technologija/Matematinis metodas	Pavadinimas/Modelis
Mikrovaldiklis	ESP32
Temperatūros sensorius	DS180B20
Bluetooth	HC-05
Wi-Fi	*Espressif* ESP8266 ESP-01
LoRa	*Sparkfun 1 channel gateway*
Relinis jungiklis	SSR
Android programėlės kūrimo įrankis	Android Studio integrated environment
Maitinimo šaltinis	3.0 – 5V
IOS programėlės kūrimo įrankis	SWIFT
Temperatūros rodymo elementas	Standard LCD
Virdulys	Elektrinis virdulys
Programavimo kalba	C, JAVA, SWIFT
Temperatūros valdymo prietaisas	Aplikacija

2.2.4.1 Atrinktų matematinių metodų pagrindimas

Sistemai suteikti automatinį vandens temperatūros valdymą, buvo panaudotas PID valdymo sistema, kuri buvo pasirinkta nes naudojantis šia sistema galima tiksliai valdyti vandens temperatūrą su daug mažesniais nukrypimais nei kitų sistemų [18]

Taip pat ši technologija buvo pasirinkta nes pramonėje daugiau nei 90% visų valdymo grandinių yra Proporciniai Integraliniai išvestiniai (PID). [22] [23]. Jie naudojami nes juos lengva gaminti, yra efektyvūs ir yra sukaupta itin daug valdymo veiksmų žinių ir diskretinių PID realizacijos. Taip pat apie šį valdymo būdą yra daugybė literatūros, todėl toks valdymas yra itin naudingas [24].

2.2.4.2 Technologijų pasirinkimo pagrindimas

Mikrovaldiklis ESP32 buvo pasirinktas nes iš jo nebus tikimasis didelės galios, jis yra lengvai integruojamas ir galima į jį lengvai įdiegti reikiamas komunikacijas. Taip pat ši technologija turi itin išplėtotą biblioteką, kuria bus galima pasinaudoti, kuriant ryšį tarp mikrovaldiklio ir programėlės [17]

Temperatūros sensoriui bus naudojamas DS180B20, kuris yra vandeniui atsparus ir gali būti sujungtas su bet kokiu mikrovaldikliu. Nors jis turi vieną diskretinę kojelę prie jos galima prijungti keletą jutiklių. Taip pat šis sensorius gali matuoti nuo -55 ◦C iki 125◦C temperatūrą [18] idealiai tinkantis virduliui planuojamiems gėrimams.

Bluetooth komunikacijai buvo pasirinktas HC-05 nes jis yra pats populiariausias modulis rinkoje ir itin dažnai naudojamas įterptiniuose projektuose. Taip pat jo kaina yra itin žema ir jis yra lengvai suderinamas su ESP32 mikrovaldikliu [25].

Wi-Fi komunikacijai buvo pasirinktas Espressif ESP8266 ESP-01 modulis, kuris yra žemos kainos ir žemai energijos išnaudojantis modulis, idealiai tinkantis kuriant IOT sistemą. [26]

Tuo tarpu *LoRa* komunikacijai buvo pasirinktas RYLR896 modulis nes jis puikai tinka IOT sistemoms, dirba su ESP32 sistema ir yra reliatyviai žemos kainos. [27]

SSR relinis jungiklis, kuris sistemoje bus naudojamas kaip tarpininkas tarp mikrovaldiklio ir virdulio, buvo parinktas nes jį yra lengva sujungti tarp mikrovaldiklio ir kintamosios srovės šaltinio [18]

Android Studio įrankis leis sukurti sistemoje *Android* programėlę. Šis įrankis buvo pasirinktas nes pasižymi:
• Šaltinio kodo redaktoriumi, kuris turi galimybe pažengusiai pabaigti kodą, gali restruktūrizuoti ir analizuoti kodą.[14]
• Kodo statymo galimybe (angl. *build*) [14]
• Derintoju (angl. *debugger*), kurio pavadinimas yra Android derintojo stebėtojas [14]

Verta paminėti, kad būtent šis įrankis buvo pasirinktas nes jis palaiko JAVA ir iš bėdos c kalbą, kurios jau yra suplanuotos kaip pagrindinės sistemos programavimo kalbos [14]

2.2.4 Egzistuojančių rinkoje metodų, kiekybinis ir kokybinis palyginimas

lentelė 3 Sistemų palyginimas

Kriterijus/Programa	*iKettle* {1}	*Apkettle* {3}	Fellow stagg ekg {2}
Komunikacijos priemonė	*Wi-FI*	*Wi-Fi*	Nėra
Integravimo į išmanaus namo galimybė	Yra	Yra	Nėra
Priemonės į kurias integruojama sistema	Alexa, Google	Alexa. Google	Nėra
Programėlės	Android	Android	Nėra
Kaina	99£	£267.45	£149.00
Temperatūros ruožas	20 – 100 ºC	0 - 100 ºC	57 - 100 ºC
Tūris	1.8 litrai	1.7 litrai	0/9
Galingumas	3000 vatai	2400 vatai	1200 vatai
Papildomos savybės		Balsu valdoma programėlė	Išlaiko iki valandos temperatūrą, rodo LCD ekrane siekiamą ir esamą temperatūrą.
Tolerancija	(~1ºC)	(~1ºC)	(~0.1 ºC)

2.3 Įgyvendinimo problemos

Kuriamai sistemai gali iškilti šios problemos:

1. Vartotojas nori pasirinkti sistemoje nenumatytą gėrimą – ši nenumatyta situacija iškiltų jeigu vartotojas sugalvotų pasirinkti gėrimą, kuris nėra vienas iš numatytų pasirinkimų sistemoje. Šią situaciją galima spręsti vartotojui pasirenkant gėrimą, kurio rekomenduojama kaitinimo temperatūra yra panaši ar lygiavertė kitiems jau sistemoje esantiems gėrimams. Taip pat šią situaciją galima spręsti į sistemos mobilią programėlę integruojant gėrimo įtraukimo funkciją, kurią naudojantis žmogus gali įtraukti gėrimą ir gėrimui rekomenduojamą temperatūrą.

2. Sistemoje pasirinkti gėrimo temperatūra viršija maksimalią galimą – sistemoje ši situacija gali iškilti jei bus sukurta gėrimo įtraukimo funkcija. Ji gali pasireikšti jei šios funkcijos įtrauktas gėrimas reikalauja didesnės temperatūros nei gali būti matuojama temperatūros jutiklio ar pasieks didesnės nei mikrovaldiklio nustatytas temperatūros ribas. Jeigu viršijama temperatūros jutiklio maksimali temperatūra tada galima situaciją spręsti panaudojant aukštesnę temperatūrą matuojantį jutiklį arba programėlėje nustatyti ribas iki kokios temperatūros vartotojas gali kaitinti vandenį. Esant situacijai kai viršijama mikrovaldiklio maksimali nustatyta temperatūra tada šiuo atžvilgiu reikia arba kelti mikrovaldiklio maksimalią priimamą temperatūrą jo programoje arba vėl nustatyti temperatūrų ribas iš kurių vartotojas rinkdamasis gėrimus negali išeiti.

3. Mikrovaldiklis nekomunikuoja su mobilios programėles *Bluetooth* (ar kito protokolo) ryšiu – ši situacija gali iškilti kai mobilaus įrenginio *Bluetooth* ar kt. protokolai nekomunikuoja su mikrovaldiklio *Bluetooth* (ar kitu protokolu). Situacija gali būti sprendžiama perkraunant visą sistemą arba sprendžiama vartotojui pasirenkant kitokį komunikacijos metodą.

4. Mikrovaldiklis gaus dvi komandas vienu metu – ši situacija gali iškilti kai vartotojas per atsitiktinumą perduoda dvi komandas vienu metu į mikrovaldiklį vykdymui. Ji gali būti sprendžiama sistemoje (mikrovaldiklyje) integruojant filtravimo sistemą, kurios dėka galima po vieną filtruoti komandas ir mikrovaldiklis vykdys tik pirmą patvirtintą vartotojo pasirinkimą . Taip pat ši situacija gali būti sprendžiama mobiliojoje programėlėje integruojant funkciją, kuri vartotojui paspaudžius ant gėrimo pasirinkimo, automatiškai įterpia gėrimo pasirinkimo pauzę, kurios metu žmogus negali pasirinkti jokio kito gėrimo. Ši pauzė galios tol, kol bus vykdoma pirma pasirinkto gėrimo komanda.

5. Mikrovaldiklis priima nepilną komandą iš mobilios programėlės – ši situacija gali kilti jei vartotojas pasirenka gėrimą, bet dėl ryšio nesklandumų, komanda yra nepilnai perduodama į mikrovaldiklį. Situacijai spręsti sistemoje (mikrovaldiklyje) galima įdiegti komandos tikrinimo funkciją – komanda yra patikrinama su mikrovaldiklyje numatytomis komandomis ir jei komanda neatitinka nei vienos iš jų, tada mikrovaldiklis atsikrato neatitikusios komandos ir vartotojui per mobilią programėlę siunčia komandos pakartojimo prašymą.

6. Sistemos veikimo sustojimas integravus į išmaniojo namo sistemą – ši situacija gali kilti jei sistema į išmanųjį namą buvo teisingai integruota, bet sistema nebeveikia. Tokiu atveju reiktų pertikrinti išmanaus namo ir išmaniojo arbatinuko sistemas ir ieškoti klaidingų ar tarp sistemų nesuderinamų funkcijų.

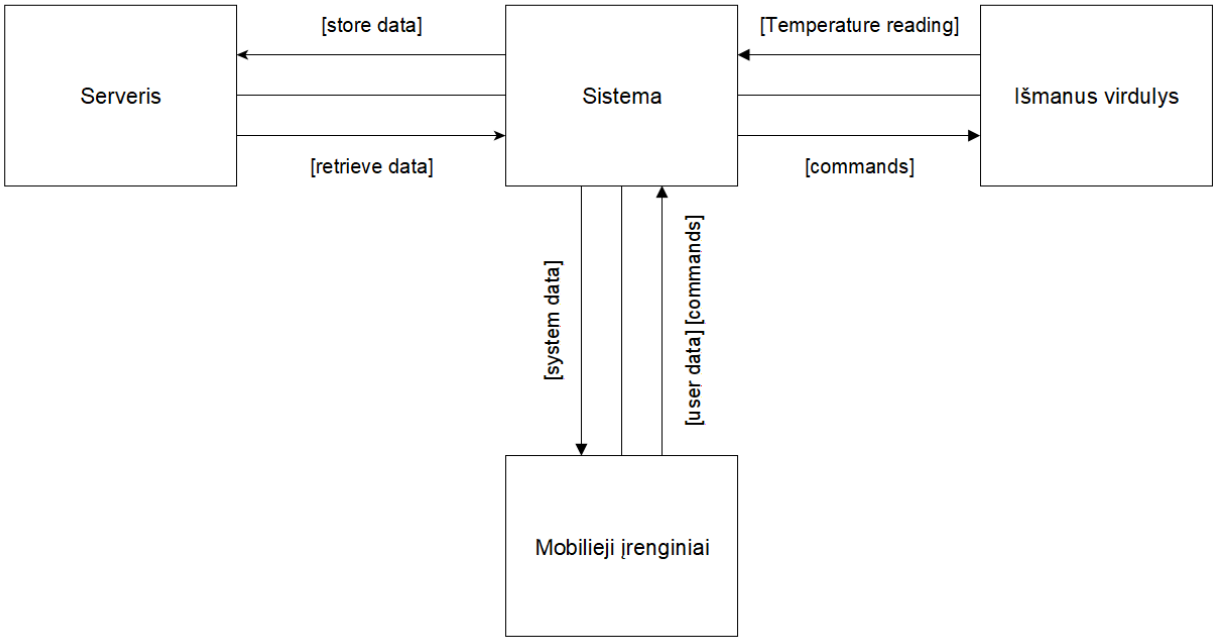
2.4 Projektavimo technologijos projektavimo santrauka

1. Pasinaudojant įvairiais su magistriniu darbu susijusiais literatūros šaltiniais, buvo plačiau susipažinta su, kuriamos sistemos ypatumais.
2. Analizuojant autorių straipsnius ir informacinius šaltinius pastebėtos potencialios bėdos, kurios gali iškilti: PID signalo nukrypimai, nenumatytas gėrimo pasirinkimas, nevyksta komunikacija, mikrovaldiklis priima nepilną komandą.
3. Taip pat pastebėti sistemų privalumai: su šia sistema vartotojui nereikės eiti iki virdulio, ir laukti kol bus paruošiamas gėrimas - gali tiesiog pasirinkti gėrimą programėlėje ir arbatinukas užkaitins.

4. Jei įmanoma sistemoje reikėtų įdiegti 3 komunikacijos protokolus – *Bluetooth*, *Wi-Fi* ir LoRa, kad būtų projekte patikrinta, kuri komunikacija labiausiai tinka sistemai.
5. Temperatūrai valdyti geriausia naudoti programėle valdoma iš vartotojo mobilaus įrenginio.
6. Pastebėtų problemų sprendimus ir privalumus bus galima pasitelkti savo kuriamoje sistemoje.

3. Sistemos paskirtis

3.6.1 Diegimo aplinka



pav. 11 Sistemos diegimo aplinka

3.7 Sąvokos ir santrumpos

Terminų, kurie bus naudojami specifikuojant reikalavimus žodynas. Čia galima rasti dažniausiai naudojamų žodžių, sutrumpinimų paaiškinimus naudojamus šioje reikalavimų specifikacijoje.

lentelė 4 Terminų ir sąvokų lentelė reikalavimams.

Pavadinimas	Apibūdinimas
IOT (Internet Of Things)	Išmanių daiktų sistema, kurioje interneto „daiktai“ gali tarpusavyje komunikuotis, siųsti, ir priimti duomenis.
LoRa	LoRa yra ilgos distancijos mažai energijos naudojantis belaidis protokolas, kuris naudoja nelicencijuotą radijo spektrą industrinėje, mokslinėje ir medicininėje radijo juostoje.
BLE (*Bluetooth*)	Mažai energijos eikvojantis *Bluetooth* (BLE) protokolas buvo pristatytas kaip viena geriausių alternatyvų daiktų interneto rinkoje.

802.11 (*Wi-Fi*)	802.11 protokolas yra viena pagrindinių belaidžių technologijų, kai kuriama daiktų internetų sistema.
Raspberry Pi	Mini kompiuteris kuris gali atlikti paprastas darbines funkcijas
Serveris	Kompiuterio aparatinės įrangos dalis, kuri apdoroja gaunamas komandas ir laikomus duomenis.
IOS programėlė	Programa veikianti pagal specializuotos mobilios operacinės sistemos pagrindą.
Android programėlė	Programa veikianti pagal specializuotos mobilios operacinės sistemos pagrindą.
Google Home	„Google Home“, yra išmaniųjų garsiakalbių linija, kurią „Google“ sukūrė pagal „Google home“ prekės ženklą. Šie įrenginiai leidžia vartotojams perduoti balso komandas ir sąveikauti su paslaugomis per „Google Assistant“, įmonės virtualų asistentą.
Amazon Echo	„Amazon Echo“ yra laisvų rankų įrangos garsiakalbių ir virtualių pagalbinių įrenginių grupė, kurie bendrauja su galutiniu vartotoju per „Amazon Alexa“ debesies balso paslaugą. Produktų linija „Echo“ įrenginiai prisijungia prie „Alexa“ paslaugos internetu.

3.8 Svarbūs faktai ir prielaidos

3.8.1 Faktai

1. Pagal 2021 metų statistiką pasaulyje buvo apie 46 milijardų išmanių prietaisų, kurie gali būti prijungti prie IOT sistemos
2. Wi – Fi sistema vidutiniškai gali dirbti iki 32 metrų atstumu
3. 2021-aisiais pasaulio IOT rinka yra apie 310 milijardų dydžio.
4. LoRa sistema vidutiniškai gali dirbti iki 20 km atstumu
5. Bluettoth sistema vidutiniškai gali dirbti 10 m atstumu

3.8.2 Veiklos taisyklės

V1. Sistema naudojanti *Wi-Fi* turi veiki 32 metrų ribose
V2. Sistema naudojanti LoRa turi veiki 20 Km ribose
V3. Sistema naudojanti *Bluetooth* turi veiki 10 metrų ribose
V4. Programėlė dėl sisteminės apžiūros nedirbs kelias valandas vieną kartą į savaitę

V5. Sistemai gali tekti konkuruoti su kitomis Išmanių virdulių sistemomis (*ikettle*, *appkettle* ir *Fellow stag ekg*)

3.8.3 Prielaidos

Kuriant sistemą, daroma prielaida, kad IOT rinka artimojoje ateityje plėsis toliau ir atsiras vis naujų vartotojų, kuriuos bus galima pritraukti pasinaudojant kuriama sistema
Kuriant sistemą daroma prielaida, kad naudotojai turės prieigą prie *Wi-Fi* ir būtent naudojantis šiuo protokolu, galės prisijungti prie sistemos ir ja naudotis.
Kuriant sistemą, daroma prielaida, kad bendrų IOT sistemų rinka artimojoje ateityje plėsis toliau ir atsiras vis naujų IOT sistemų autorių, kurie norės prijungti šią sistemų prie jų.

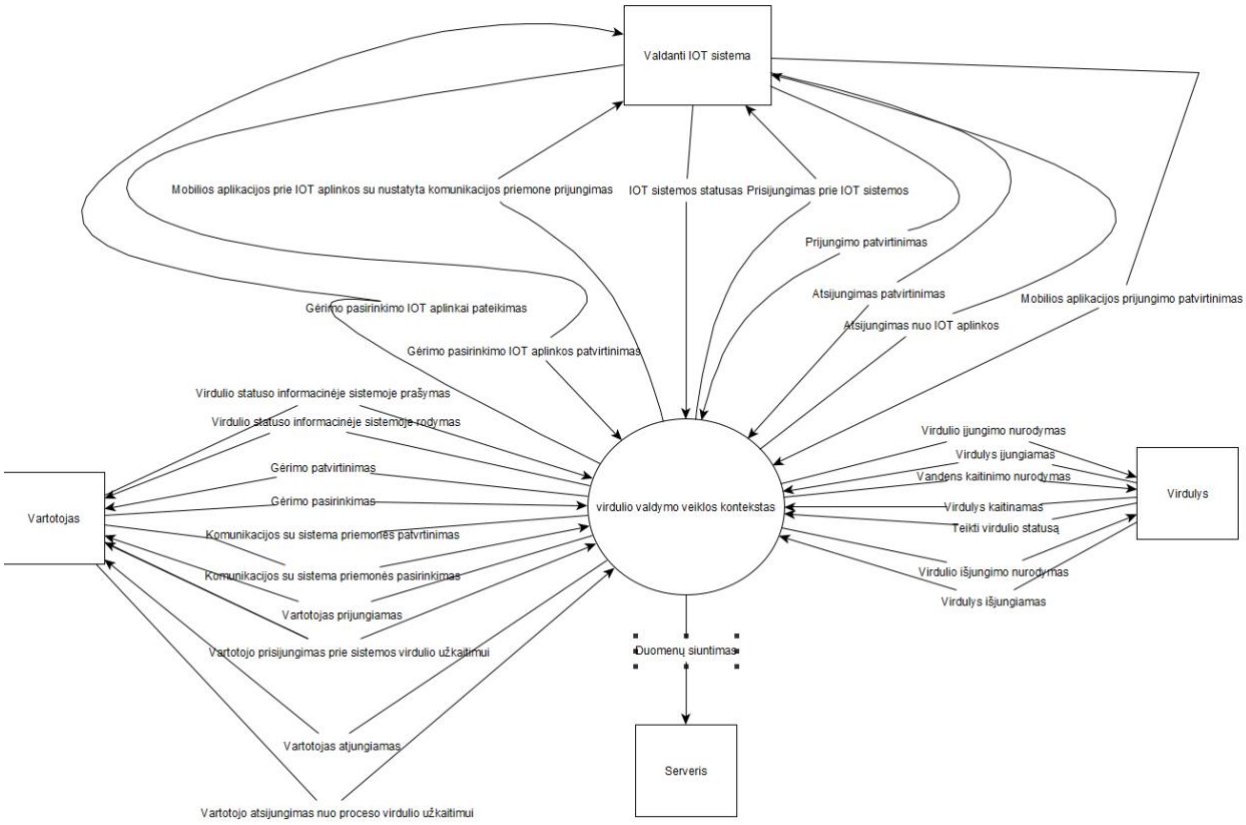
3.9 Veiklos sudėtis

3.9.1 Esama padėtis

Šiuo metu dauguma elektrinių virdulių savininkų turi rankiniu būdu įjungti ir užkaisti vandenį virdulyje. Tai daroma, priėjus prie virdulio, pripylus vandens ir tada jis yra įjungiamas ir užkaitomas. Taip pat galimas variantas, kad virdulys yra iš dalies yra išmanusis, rodo temperatūrą ir rodo vandens statusą, bet viską pamatyti reikia vėl prieiti prie virdulio. Kitais dar atvejais yra galimas variantas, kad yra virdulių, kurie yra įjungiami iš nuotolio, bet dažniausiai yra pasiūlomos tik viena komunikacijos priemonė (šio projekto įgyvendinimo atveju, bus trys komunikacijos priemonės – BLE, *Wi-Fi, LoRa*). Ir net esant galimybei, matuoti temperatūrai iš nuotolio ir prisijungti trimis ar daugiau komunikacijos priemonių, šios sistemos yra visiškai atskiros sistemos ir neintegruojamos kaip posistemė į kitas sistemas – Pvz. *Google Home* ar *Amazon*.

3.9.2 Veiklos kontekstas

Čia bus pavaizduota esama situacija pagal kurią bus galima išanalizuoti esamą EV elektros pirkimo ir pardavimo situacija ir jos trūkumus ir parinkti geriausią iš kompiuterizavimo alternatyvų.



pav. 12 Sistemos veiklos kontekstas

3.9.3 Veiklos suskaidymas

Įvykis:
1A. Vartotojas nori prisijungti prie virdulio užkaitimo proceso;
Įeinantis srautas
1B. Vartotojo prisijungimas prie sistemos virdulio užkaitimui;
Išeinantis srautas:
1C. Vartotojas prijungiamas;
Įvykis:
2A. Vartotojas pasirenka komunikacijos priemonę virdulio valdymu;
Įeinantis srautas:
2B. Komunikacijos su sistema priemonės pasirinkimas;
Išeinantis srautas:
2C. Komunikacijos su sistema priemonės patvirtinimas;
Įvykis:
3A. Vartotojas pasirenka gėrimą;
Įeinantis srautas:
3B. Gėrimo pasirinkimas;
Išeinantis srautas:
3C. Gėrimo patvirtinimas;
Įvykis:
4A. Vartotojas nori pamatyti virdulio statusą informacinėje sistemoje;
Įeinantis srautas:

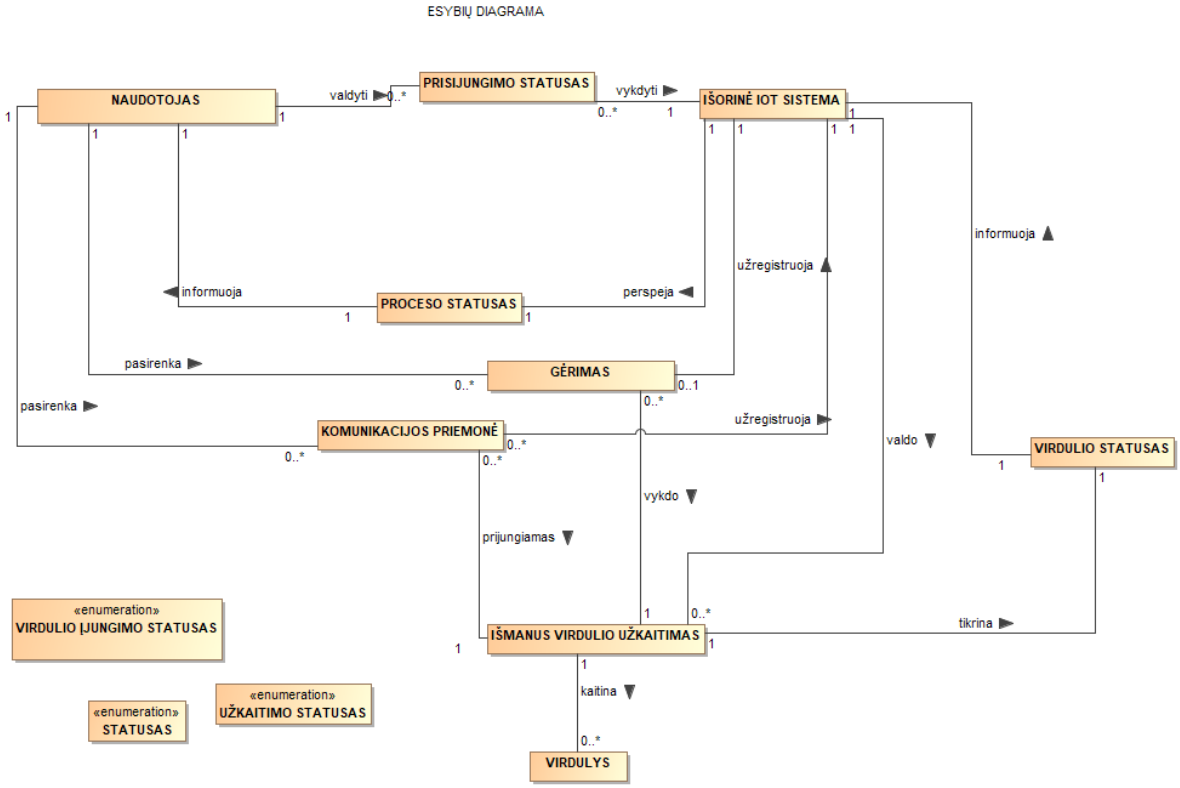
4B. Virdulio statuso informacinėje sistemoje prašymas;
Išeinantis srautas:
4C. Virdulio statuso informacinėje sistemoje rodymas;
Įvykis:
5A. Vartotojas nori prisijungti prie virdulio užkaitimo proceso;
Įeinantis srautas:
5B. Vartotojo atsijungimas nuo proceso virdulio užkaitimui;
Išeinantis srautas:
5C. Vartotojas atjungiamas;
Įvykis:
6A. Virdulio veiklos kontekstas pateikia gėrimo pasirinkimą IOT aplinkai;
Įeinantis srautas:
6B. Gėrimo pasirinkimo IOT aplinkai pateikimas;
Išeinantis srautas:
6C. Gėrimo pasirinkimo IOT aplinkos patvirtinimas;
Įvykis:
7A. Virdulio veiklos kontekstas pateikia mobilios programėlės prijungimą prie IOT aplinkos;
Įeinantis srautas:
7B. Mobilios programėlės prie IOT aplinkos su nustatyta komunikacijos priemone prijungimas;
Išeinantis srautas:
7C. Mobilios programėlės prijungimo patvirtinimas;
Įvykis:
8A. IOT pateikia savo sistemos statusą;
Įeinantis srautas:
8B. IOT sistemos statusas;
Įvykis:
9A. Virdulio veiklos kontekstas prisijungia prie IOT aplinkos;
Įeinantis srautas:
9B. Prisijungimas prie IOT sistemos;
Išeinantis srautas:
9C. Prijungimo patvirtinimas;
Įvykis:
10A. Virdulio veiklos kontekstas atsijungia nuo IOT aplinkos;
Įeinantis srautas:
10B. Atsijungimas nuo IOT sistemos;
Išeinantis srautas:
10C. Atjungimo patvirtinimas;
Įvykis:
11A. Virdulio veiklos kontekstas įjungia virdulį;
Įeinantis srautas:
11B. Virdulio įjungimo nurodymas;
Išeinantis srautas:
11C. Virdulys įjungiamas;
Įvykis:
12A. Virdulio veiklos kontekstas nurodo kaitinti virdulį;

Įeinantis srautas:
12B. Virdulio kaitinimo nurodymas;
Išeinantis srautas:
12C. Virdulys kaitinamas;
Įvykis:
13A. Virdulys teikia savo statusą ir informaciją;
Įeinantis srautas:
12B. Teikti virdulio statusą
Įvykis:
12A. Virdulio veiklos kontekstas nurodo išjungti virdulį;
Įeinantis srautas:
12B. Virdulio išjungimas;
Išeinantis srautas:
12C. Virdulys išjungiamas;
Įvykis:
13A. Virdulio veiklos kontekstas siunčia duomenis į serverį;
Įeinantis srautas:
12B. Duomenų siuntimas;

3.10 Duomenų modelis ir vienetų žodynas

3.10.1 Esybių duomenų modelis

Čia vaizduojama esybių diagrama, kurioje galima pamatyti ryšius tarp įvairių esybių narių.



pav. 13 Esybių duomenų modelis

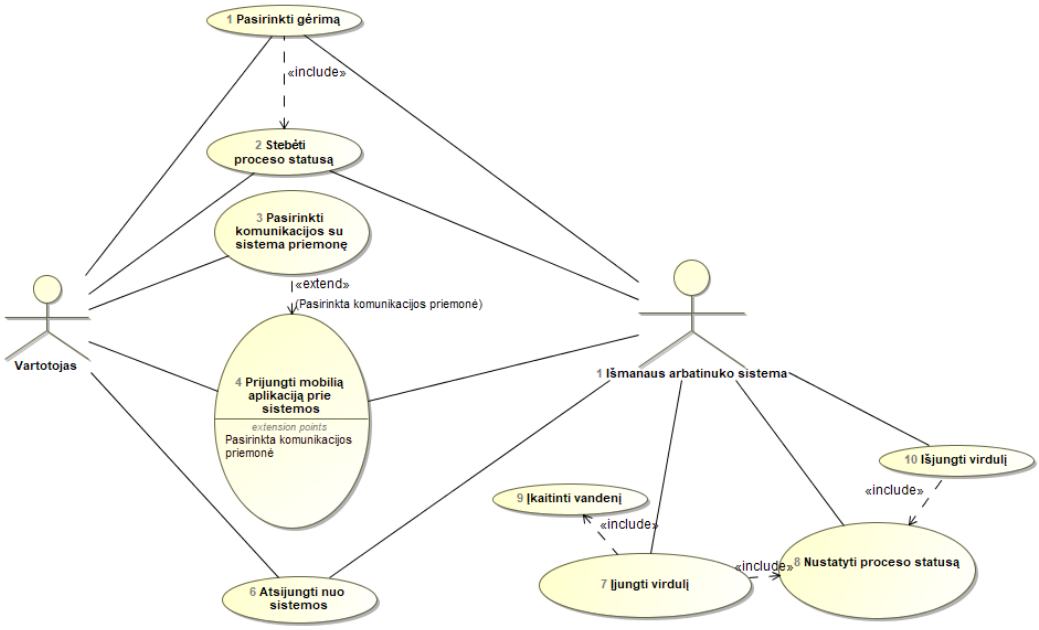
3.10.2 Duomenų žodynas (duomenų modelio specifikacija)

lentelė 5 Duomenų žodynas

Duomenų žodyno elemento pavadinimas	Turinys	Tipas
Naudotojas	Telefono tipas + Vardas	*Class*
Prisijungimo statusas	Statusas	*Class*
Statusas	Prisijungta, atsijungta	*Enumeration*
Išorinė IOT sistema	IOT platforma	*Class*
Virdulio statusas	Temperatūra	*Class*
Proceso statusas	Užkaitimo statusas + temperatūra	*Class*
Užkaitimo statusas	Ruošiamas gėrimas + paruoštas gėrimas	*Class*
Gėrimas	Gėrimo pavadinimų + temperatūra	*Class*
Komunikacijos priemonės	Belaidės komunikacijos protokolo pavadinimas	*Class*
Išmaniojo virdulio užkaitimas	Virdulio įjungimo statusas + temperatūra	*Class*
Virdulio įjungimo statusas	Įjungtas + išjungtas	*Enumeration*
Virdulys	Virdulio tipas	*Class*

3.11 Sistemos sudėtis

3.11.1 Panaudojimo atvejų diagrama



pav. 14 Panaudojimo atvejų diagrama

3.11.2 Panaudojimo atvejų specifikacijos

PA 1 panaudojimo atvejis

1. PANAUDOJIMO ATVEJIS: Pasirinkti gėrimą **Vartotojas/Aktorius:** Išorinė IOT sistema, vartotojas **Aprašas:** Vykdant šį PA, galima pasirinkti patinkantį gėrimą, stebėti pasirinkto gėrimo statusą. **Prieš sąlyga:** Prijungti mobilią programėlę prie sistemos;

Sužadinimo sąlyga: Vartotojas nori pasirinkti gėrimą Vartotojas nori matyti gėrimo statusą

Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Naudotojas pasirenka gėrimą funkcija sistemoje „pasirinkti gėrimą“	1.1 Sistema išmeta visų galimų gėrimų pasirinkimus

2. Naudotojas pasirenka norimą gėrimą funkcija „patvirtinti“	2.1. Naudotojas pasirenka norimą gėrimą funkcija „patvirtinti“
3. Vartotojas nori matyti gėrimo statusą	3. Vykdomas „Stebėti gėrimo statusą“ PA
4. PA pabaiga	

Po-sąlyga: Jei vykdomas tik 1 žingsnis – naudotojui išmetamas visų galimų gėrimų pasirinkimus. Jei vykdomas 2 žingsnis – siunčiama komanda į mikrovaldiklį ir pradedamas gėrimo gaminimas. Jei vykdomas 3 žingsnis – vaizduojamas gėrimo statusas.

PA 1.1 panaudojimo atvejis

PA 1.1 stebėti proceso statusą

Tikslas/uždavinys: Vartotojas gali matyti gėrimo virimo statusą
Vartotojas/Aktorius: Išorinė IOT sistema, vartotojas
Aprašas: Šiame etape vartotojas gali stebėti pasirinkto gėrimo statusą ir eiti jį pasiimti jam būnant paruoštam ir šia PA yra papildoma PA 1
Prieš sąlyga: Prijungta mobili programėlė prie sistemos ir buvo atliktas gėrimo pasirinkimas

Sužadinimo sąlyga: Vartotojas nori pasirinkti gėrimą
Vartotojas nori matyti gėrimo statusą

Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Vartotojas pasirenka funkciją „stebėti gėrimo statusą“	1.1 Išorinė IOT sistema pradeda PA 4.2
	1.2 Išorinė IOT sistema gavusi iš mikrovaldiklio informaciją, kad vis dar kaitinamas, nusiunčia statusą mobilios programėlės sąsaja „ruošiamas gėrimas“
	1.3 Išorinė IOT sistema gavusi iš mikrovaldiklio patvirtinimą, kad vanduo buvo sėkmingai užvirintas, nusiunčia statuso atnaujinimą į sistemos mobilios programėlės sąsaja „paruoštas gėrimas“
2. Vartotojas eina prie virdulio pasiimti užkaitintą vandenį atsiradus statusui „paruoštas gėrimas“	2.0 Išorinė IOT sistema laukia naujo PA 1 „Pasirinkti gėrimą“
PA pabaiga	

Po-sąlyga: Jei vykdomas tik 1 žingsnis – naudotojui išmetama kaitinamo vandens statusas – ruošiamas arba paruoštas gėrimas (Užkaitimo progresas). Jei vykdomas 2 žingsnis – Išorinė IOT sistema laukia naujos komandos.

PA 2 panaudojimo atvejis

PA 2 Pasirinkti komunikacijos su sistema priemonę

Tikslas/uždavinys: PA 2 Pasirinkti komunikacijos su išorine sistema priemonę
Vartotojas/Aktorius: vartotojas
Aprašas: Šiame etape vartotojas gali pasirinkti komunikacijos priemonę (*Bluetooth, Wi-Fi, LoRa*)
Prieš sąlyga: Vartotojas nepasirinko komunikacijos sistemos

Sužadinimo sąlyga: Vartotojas nori pasirinkti komunikacijos protokolą sujungti su išorine sistema

Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Vartotojas paleidžia mobilią programėlę	1.1. Mobili programėlė atvaizduoja pasirinkimus vartotojui prisijungti prie sistemos.
2. Vartotojas pasirenka *Bluetooth* protokolo prijungimo funkciją „Prijungti su *Bluetooth*“	2.1. Užregistruojamas pasirinkimas ir pradedamas PA 2.1 su *Bluetooth* protokolu
3. Vartotojas pasirenka *Wi-Fi* protokolo prijungimo funkciją „Prijungti su *Wi-Fi*“	3.1. Užregistruojamas pasirinkimas ir pradedamas PA 2.1 su *Wi-Fi* protokolu
4. Vartotojas pasirenka LoRa protokolo prijungimo funkciją „Prijungti su LoRa“	4.1. Užregistruojamas pasirinkimas ir pradedamas PA 2.1 su LoRa protokolu
PA pabaiga	

Po-sąlyga: Jei vykdomas tik 1 žingsnis – Mobili programėlė atvaizduoja pasirinkimus vartotojui prisijungti prie sistemos. Jei vykdomas 2 žingsnis – Mobili programėlė susikomunikuoja su sistema naudojant *Bluetooth* protokolą. Jei vykdomas 3 žingsnis – Mobili programėlė susikomunikuoja su sistema naudojant *Wi-Fi* protokolą. Jei vykdomas 4 žingsnis - Mobili programėlė susikomunikuoja su sistema naudojant *LoRa* protokolą

PA 3 panaudojimo atvejis

PA 3 Atjungti mobilią programėlė nuo sistemos

Tikslas/uždavinys: Vartotojas gali atsijungti nuo sistemos
Vartotojas/Aktorius: Išorinė IOT sistema, vartotojas
Aprašas: Šiame etape vartotojas gali atsijungti nuo išorinės IOT sistemos.
Prieš sąlyga: Vartotojas atsijungė nuo Išorinės IOT sistema

Sužadinimo sąlyga: Vartotojas nori atsijungti nuo sistemos

Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Vartotojas paleidžia funkciją „atsijungti nuo sistemos“	1.1. Mobili programėlė nutraukia ryšį su su išorine IOT sistema
	2.1. Išorinė IOT sistema atpažįsta , kad nuo jos atsijungta ir pabaigia vykdyti komandą, bet nebevykdo PA 1.1
PA pabaiga	

Po-sąlyga: Mobili programėlė nutraukia ryšį su sistema ir sistema nebesiunčia signalo, kad buvo įvykdyta komanda dėl užkaitinti vandens

PA 4 panaudojimo atvejis

PA 4 įjungti virdulį

Tikslas/uždavinys: Išmani virdulio sistema įjungia virdulį
Vartotojas/Aktorius: Išorinė IOT sistema
Aprašas: Šiame etape Išorinė IOT sistema įjungia virdulį
Prieš sąlyga: Vartotojas pasirinko gėrima ir prisijungė prie Išorinės IOT sistemos

Sužadinimo sąlyga: Išorinė IOT sistema gavo signalą apie pasirinktą gėrimą

Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**

1. Išorinė IOT sistema gauna pasirinkto gėrimo signalą	1.1. Sistema įjungia virdulį
	1.2. Pradedamas PA 4.1
	1.3. Sistema matuoja vandens temperatūrą
	1.4. Pradedamas PA 4.2
	1.5. Sistema užfiksuoja, kad pasiekta reikalaujama vandens temperatūra
	1.6 Pradedamas PA 4.3
PA pabaiga	

Po-sąlyga: Jei vykdomas tik 1.1 žingsnis – Sistema įjungia virdulį. Jei vykdomas tik 1.2 žingsnis – Pradedamas vandens kaitinimas. Jei vykdomas tik 1.3 žingsnis – Sistema matuoja temperatūrą. Jei vykdomas tik 1.4 žingsnis – Pradedamas statuso nustatymo procesas Jei vykdomas tik 1.5 žingsnis – Sistema užfiksuoja, kad pasiekta reikalaujama vandens temperatūra. Jei vykdomas tik 1.6 žingsnis – Pradedamas virdulio išjungimas

PA 4.1 panaudojimo atvejis

PA 4.1 Įkaitinti vandenį

Tikslas/uždavinys: Kaitinti vandenį
Vartotojas/Aktorius: Išorinė IOT sistema
Aprašas: Šiame etape išplečiama virdulio įjungimo panaudojimo atvejis
Prieš sąlyga: sistema įjungė virdulį

Sužadinimo sąlyga: Virdulys buvo įjungtas

Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Sistema įjungė virdulį	1.1. Vanduo yra kaitinamas
PA pabaiga	

Po-sąlyga: Vanduo yra kaitinamas

PA 4.2 panaudojimo atvejis

PA 4.2 Nustatyti proceso statusą

Tikslas/uždavinys: Nustatyti proceso statusą
Vartotojas/Aktorius: Išorinė IOT sistema
Aprašas: Šiame etape nustatomas proceso statusas ir išplečiamas virdulio įjungimo panaudojimo atvejis
Prieš sąlyga: Išorinė sistema įjungė virdulį ir yra kaitinamas

Sužadinimo sąlyga: Išorinė sistema pradeda matuoti vandens temperatūrą

Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Išorinė sistema pasiunčia signalą sistemai išmatuoti temperatūrą	1.1. Sistema išmatuoja virdulio temperatūrą
2. Išorinė sistema gavo pranešimą, kad temperatūra yra mažiau vartotojo trokštamos temperatūros.	2.1. Virdulio kaitinimo procesas yra tęsiamas ir nustatomas „ruošiamas gėrimas“ statusas
	1.2 Sistema išsiunčia nustatytą statusą į išorinę sistemą siuntimui į mobilią programėlę
3. Išorinė sistema gavo informaciją, kad temperatūra yra lygi ar daugiau negu vartotojo trokštamos temperatūros.	2.1 Virdulio kaitinimo procesas yra pabaigiamas ir nustatomas „paruoštas gėrimas“ statusas ir pradedamas 4.3 PA
	2.2 Sistema išsiunčia nustatytą statusą į mobilią programėlę
PA pabaiga	

Po-sąlyga: Jei vykdomas 1 žingsnis - Virdulio kaitinimo procesas yra tęsiamas ir nustatomas „ruošiamas gėrimas“ statusas ir taip pat sistema išsiunčia nustatytą statusą į mobilią programėlę. Jei vykdomas tik 2 žingsnis - Virdulio kaitinimo procesas yra pabaigiamas ir nustatomas „paruoštas gėrimas“ statusas ir pradedamas 4.3 PA ir sistema išsiunčia nustatytą statusą į mobilią programėlė

PA 4.3 panaudojimo atvejis

PA 4.3 Išjungti virdulį

Tikslas/uždavinys: Išjungti virdulį
Vartotojas/Aktorius:
Aprašas: Šiame etape išjungiamas virdulys ir išplečiamas PA 4.2

Prieš sąlyga: Įjungtas virdulys ir nustatytas proceso statusas

Sužadinimo sąlyga: Sistema nustatė temperatūrą aukštesnę arba lygią vartotojo trokštamai.

Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Sistema nustatė temperatūrą aukštesnę arba lygią vartotojo trokštamai	1.1. Virdulys yra išjungiamas
PA pabaiga	

Po-sąlyga: Virdulys yra išjungiamas

3.12 Funkciniai ir nefunkciniai reikalavimai

lentelė 6 Sistemos funkciniai reikalavimai

Reikalavimo numeris	Aprašymas	Tinkamumo kriterijus
1	Sistema turi atitinkamai rodyti ar mobili aplikacija yra prisijungusi prie serverio.	Paleidus programėlę, vartotojui iš karto aišku, ar jis jau prisijungė prie išorinės IOT sistemos.
2	Vartotojas gali bet kada esant poreikiui prisijungti, atsijungti nuo išorinės IOT sistemos.	Naudotojas sėkmingai prijungia/atjungia programėlę prie/nuo išorinės IOT sistemos.
3	Programėlė, visada rodo kokiu komunikacijos protokolu prisijungė prie sistemos ir ar yra laisvų veikiančių protokolų aplinkoje.	Programėlė rodo laisvų ir užimtų protokolų sąrašą.
4	Vartotojas žino koks yra kaitinamo gėrimo statusas.	Programėlė parodo ar gėrimas yra paruoštas, ar vis dar ruošiamas.
5	Vartotojas gali bet kada iš programėlės pasirinkti trokštamą gėrimą.	Vartotojas sėkmingai pasirinko trokštamą gėrimą.
6	Išorinė IOT sistema turi automatiškai prisijungti ar prisijungti prie programėlės, atėjus atitinkamai komandai.	Naudotojui pasirinkus protokolą ir prisijungimo statusą, išorinė IOT sistema, automatiškai vykdo šią komandą.
7	Jei išorinė IOT sistema gavo komandą iš programėlės, ji turi automatiškai prisijungti prie serverio, ir siųsti komandą pradėti kaitinti virdulį iki nurodytos temperatūros.	Išorinei IOT sistemai, gavus iš programėlės komandą, automatiškai jungiasi prie serverio ir siunčia komandą.

8	Serveris, gavęs instrukciją iš išorinės IOT sistemos, automatiškai turi kaitinti vandenį iki nurodytos temperatūros.	Atėjus komandai iš išorinės IOT sistemos, serveris įjungia virdulį ir kaitina jį iki nurodytos temperatūros.
9	Serveris, turi nesustoti kaitinti virdulio iki bus pasiekta siekiama temperatūra, net ir praradus ryšį su išorine IOT sistema arba programėle.	Serveris nesustoja kaitinti virdulį iki kol įvykdys komandą.
10	Kaitinant virdulį, serveris turi matuoti temperatūrą.	Serveris išmatuoja kaitinamo virdulio temperatūrą.
11	Virdulio temperatūra turi būti matuojama kas 1 s ir siunčiama į programėlę arba į Išorinę IOT sistemą.	Virdulio temperatūrą serveris išmatuoja kas 1 s ir informacija siunčiama į išorinę IOT sistemą arba programėlę.
12	Virdulio kaitinimo metu, išmatuota temperatūra be perstojo turi būti siunčiama į išorinę IOT sistemą arba į programėlę.	Nustačius temperatūrą virdulyje, serveris iš karto ją siunčia į išorinę IOT sistemą arba į programėlę.
13	Jei išorinė IOT sistema, gauna nustatytą temperatūrą iš serverio, ji turi prisijungus prie programėlės, turi ją persiųsti.	Programėlė, užkaitus virduliui iki tinkamos temperatūros, atvaizduoja naudotojui „gėrimas paruoštas“ statusą.
14	Virdulys turi išsijungti, pasiekus pasirinktą temperatūrą.	Virdulys atsijungia, užkaitinus vandenį iki reikiamos temperatūros.
15	Vartotojui pasirinkus, stabdyti komunikaciją, virdulys turi išsijungti.	Virdulys atsijungia, iš aplikacijos arba išorinės IOT sistemos gavus komandą, stabdyti kaitinimą.

lentelė 7 Sistemos nefunkciniai reikalavimai

Reikalavimo numeris	Aprašymas	Tinkamumo kriterijus
1	Mobilios programėlės išvaizda turi būti IOT tematikos.	Daugumą vartotojų, paleidę programėlę, supranta programėlės paskirtį.
2	Visos sistemos funkcijos turi būti matomos paleidus mobilią programėlę.	Vartotojas prisijungęs prie programėlės mato šias komandas: „prisijungti prie virdulio“, „pasirinkti gėrimą“ „stebėti gėrimo statusą“
3	Mobili programėlė, turi greitai prisijungti prie išorinės IOT sistemos arba serverio.	Programėlėje paleidus funkciją „patvirtinti prisijungimą prie virdulio“, programėlė prisijungia prie išorinės IOT sistemos arba serverio per 10 s.
4	Programėlė turi atvaizduoti tik pasirinkto gėrimo statusą.	Mobili programėlė atvaizduoja pasirinkto gėrimo kaitinimo proceso etapą.

5	Užkaisto vandens temperatūrą turi būti tokia pati, kokia buvo nustatyta vartotojo programėlėje.	Vartotojas, jei programėlėje pasirinko gėrimą turintį 70 °C, tokia pati temperatūra turi būti ir ką tik užkaistame virdulyje.
6	Pasirinktas komunikacijos protokolas turi būti naudojamas jungimuisi prie išorinės sistemos ir kuriamos sistemos.	Atjungus pasirinktą komunikaciją, sistemos ryšys yra nutraukiamas.
7	Egzistuojant komunikacijos protokolui, jis turi būti atvaizduojamas mobiliojoje programėlėje kaip pasiekiamas.	Visi protokolas prie kurių įmanoma prisijungti mobiliojoje programėlėje, turi būti pasiekiami vartotojui ir pasirenkami.
8	Neegzistuojant komunikacijos protokolui, jis neturi būti atvaizduojamas mobiliojoje programėlėje kaip pasiekiamas.	Visi protokolas prie kurių neįmanoma prisijungti mobiliojoje programėlėje, turi būti nepasiekiami vartotojui ir nepasirenkami.
9	Sistema ir programėlė turi atnaujinti naują įkeltą informaciją	Įkėlus naujų gėrimų ir jų temperatūrų ekvivalentų, jie turi būti atnaujinti sistemoje ir mobiliojoje programėlėje
10	Turi būti galimybė plėsti sistemos funkcionalumą ir didinti jos galimybes.	Naujas funkcija į sistemą integruojama nekeičiant pagrindo kodo.
11	Sistema gali dirbti su skirtingomis išorinėmis IOT sistemomis.	Prijungus programėlę prie išorinės IOT sistemos gaunami pasirinkimai ir kitiems IOT prietaisams.
12	Sistemos serveriai turi būti nuolatos tikrinami.	Serveriai yra kas savaitę tikrinami ir užtikrinama, kad iki kitos savaitės veiks tinkamai.
13	Sistema turi dirbti su Android platformomis.	Naudotojas per bet kurią iš šių operacinių aplinkų gali pasiekti sistemą.

3.15.2 Reikalavimai darbui su gretimomis sistemomis

Atviros problemos ir klausimai
Vis dar neaišku ar viena iš komunikacijos priemonių naudojamų su mikrovaldikliu pasirodys, kad yra netinkama prijungimui prie išmanaus virdulio. Dar nebuvo padaryti pirmieji testai tam patikrinti.

Neaišku koks yra poreikis išmaniai virdulio sistemai, kuri kaitina iki norimos temperatūros.
Neaišku kuri išorinė sistema (*Google Home* ar *Amazon Echo*), būtų tinkamiausi kuriamai sistemai ir ar visos išorinės sistemos galės integruoti kuriamą sistemą.

3.13 Egzistuojantys sprendimai

3.13.1 Prieinamos sistemos

OpenHab – Ši sistema yra puiki platforma integruoti įvairių rūšių išmanius prietaisus prie kuriamos sistemos. Turėdama beveik 33 000 narių, *openHAB* yra lengva atsisiųsti į populiariausias operacines sistemas.

Google Home – Šia viešai prieinama sistema galima pasinaudoti kuriant ryšį tarp išmanaus prietaiso ir serverio. *Google Home* jau turi paruoštą sąrašų „veiksmų“ – funkcijų, kurios jau yra užprogramuotos ir galima pasinaudoti paleidžiant valdant išmanų arbatinuką.

Amazon Skils – Taip pat viešai prieinama sistema, kurioje galima kurti „*skills*“ – balsu valdomas komandos kuriomis prie Amazon sistemos prijungtas prietaisas bus valdomas. Pagrindinė šios sistemos savybė, kad tokiu atveju nereikėtų savininkui išmanaus telefono, ir tik Amazon balsu valdomu prietaisu valdyti arbatinuką.

3.13.2 Prieinami komponentai

Yra puiki galimybė integruoti išorinius komponentus, kurie kontroliuoja kitus gėrimo aspektus, pavyzdžiui – išmanusis reguliuojamos temperatūros termosas. Jį būtų galima panaudoti sistemoje kai jau gėrimas yra užkaistas ir yra užpilamas į puodelį. Puikus šio projekto tęsinys integruoti komponentą kuris kaitina patį puodelį.

3.13.3 Kopijuotini sprendimai

Ikettle sistemą, galima valdyti iš mobilaus įrenginio esant bet kurioje namų vietoje, virdulio temperatūrą galima valdyti pagal vartotojo norus ir galiausiai virdulys bus lengvai integruojamas į *Alexa* ir *Google* asistentų valdomus išmanius namus. Stebint šios kompanijos produktus galima perprasti geriausią sprendimą integruoti kuriamą sistemą į *Alexa* ir Google platformas, ir integruoti funkciją, kur vartotojo gėrimas kaitinamas iki jo pasirinktos temperatūros ir nebūtinai optimalios gėrimui.
Fellowpruducts kompanijos *Fellow stag ekg* virdulys pasižymi dauguma jau minėtomis *Ikettle* savybėmis, bet ir gali, skirtingai nei kuriama sistema, išlaikyti iki valandos trokštamą temperatūrą, rodyti LCD ekrane siekiamą ir užkaitintą vandens temperatūrą, bei leidžia celcijiniu tikslumu reguliuoti užkaitinimo temperatūrą kas gali būti labai kopijuotina kuriamai sistemai ateityje.

3.14 Naujos problemos

3.14.1 Poveikis diegimo aplinkai

Bet koks sistemos duomenų bazių pakaitimas išskirs vartotojo matomą informaciją mobiliojoje programėlėje.
Bet kokie sistemos virdulio kaitinimo pakeitimai sukels sunkumų automatizuojant procesą tarp sistemos, tarpinės išorinės sistemos ir vartotojo.

3.14.2 Probleminė naudotojų reakcija

Vartotojai gali nenorėti pasirinkti gėrimą numatytomis temperatūromis ir gali palikti sistemą ir pasirinkti konkurentus, nes norėtų turėti savo norimos temperatūros pasirinkimo galimybę.

3.14.3 Apribojimai diegimo aplinkoje

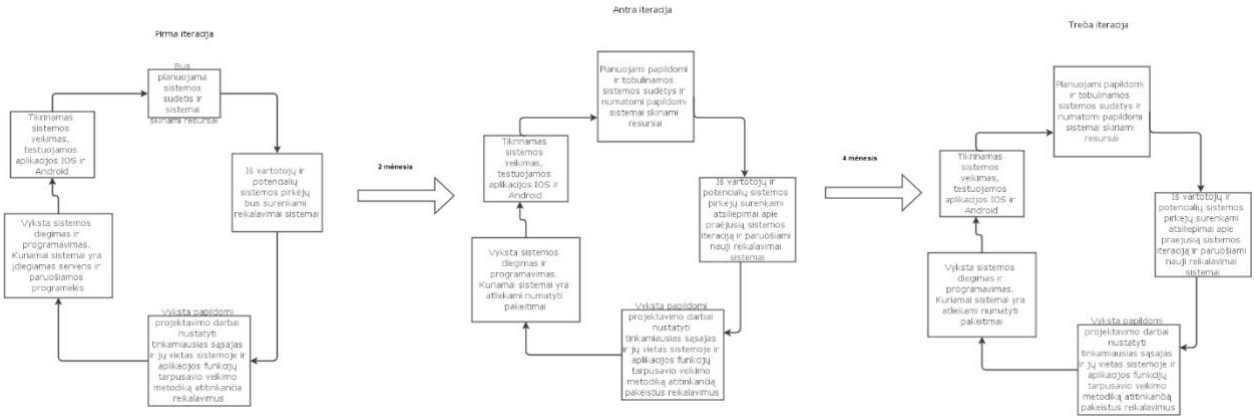
Planuojamas serveris nėra pakankamai galingas prijungti daugiau nei vieną prisijungusį vartotoją.

Planuojamas serveris nebus pasirengęs priimti daugiau nei vieną komandą iki jos išpildymo pabaigos.

3.15 Uždaviniai

3.15.1 Sistemos kūrimo procesas

Sistemai kurti, dėl to, kad pilnai yra dar neaiškūs sistemos reikalavimai ir norima rasti vartotojams priimtiną sprendimą iteracijomis, bus naudojamas Agile sistemos modelis.



pav. 15 Sistemos modelis

3.15.2 Detalus kūrimo planas

1 iteracijos planavimo etapas - nustatomi sistemos tikslai ir uždaviniai.
1 iteracijos projektavimo etapas - vykdomas sistemos projektavimas.
1 Iteracijos sistemos diegimo etapas - vyksta sistemos diegimas ir kūrimas.
1 Iteracijos sistemos testavimo etapas - vyksta sistemos testavimas ir tikrinimas.
2 iteracijos planavimo etapas - nustatomi nauji sistemos tikslai ir uždaviniai.
2 iteracijos reikalavimų analizės etapas - iš vartotojų, potencialių sistemos pirkėjų ir pagal gautą pirmos iteracijos priėmimą rinkoje, surenkami nauji ar papildomi egzistuojantys reikalavimai sistemai.
2 iteracijos projektavimo etapas - pagal naujus reikalavimus vykdomas sistemos projektavimas.
2 Iteracijos sistemos diegimo etapas - vyksta papildomi sistemos diegimai ir pakeitimai.
2 Iteracijos sistemos testavimo etapas - vyksta sistemos testavimas ir tikrinimas.
3 iteracijos planavimo etapas - nustatomi nauji sistemos tikslai ir uždaviniai.

3 iteracijos reikalavimų analizės etapas - iš vartotojų, potencialių sistemos pirkėjų ir pagal gautą pirmos ir antros iteracijos priėmimą rinkoje, surenkami nauji ar papildomi egzistuojantys reikalavimai sistemai.
3 iteracijos projektavimo etapas - pagal naujus reikalavimus vykdomas sistemos projektavimas.
3 Iteracijos sistemos diegimo etapas - vyksta papildomi sistemos diegimai ir pakeitimai.
3 Iteracijos sistemos testavimo etapas - vyksta sistemos testavimas ir tikrinimas.

3.16 Rizikų įvertinimas

3.16.1 Su kokia rizika susiduriame kurdami sistemą

lentelė 8 Rizikos faktorių lentelė

Rizikos faktorius	Veiksmų aprašas nepasireiškus faktoriui	Veiksmų aprašas nepasirėškus faktoriui
Vis dar neaišku kaip keisis išorinių IOT sistemų standartai ir jų veikimo būdai. Tai gali stipriai paveikti mobilios programėlės reikalingumą.	Rizikos faktorius yra mažinamas, didinant sistemos naudotojų skaičių, kad būtų galima pereiti į kitą labiau suderinamą sistemą	Rizikos faktorių bandoma valdyti aktyviai ieškant naujų sistemų, kurios būtų labiau suderinamos su kuriama.
Ar atsiras komunikacijos protokolas, kuris bus visais atžvilgiais geresnis nei esami komunikacijos protokolai	Rizikos faktorius yra mažinamas, pamažu integruojant naujai išleistą protokolą į sistemą ir padarant jį suderinamą su sistemą.	Įvykus šiam scenarijui aktyviai bandoma persiorientuoti į naują protokolą.
Ar rinkoje neiškils produktas kuris nurungs IOT sistemą savo galimybėmis.	Bus ieškoma būdų kaip perorientuoti kuriamą sistemą į naujo produkto aplinka.	Bus siekiama kuo greičiau persikelti ar įterpti funkcionalumą sistemoje, kurio atžvilgiu naujas produktas bus suderinamas su kuriama sistema.

3.16.2 Kokį atsitiktinumų (rizikų) įvertinimo planą reikia sudaryti

3.16.3 Kaštai

Sistemos kūrimui planuojama skirti 18000 € - 17000 yra skirta 3 darbuotojams už trijų mėnesių darbą ir 1000 skiriama visoms kitoms išlaidoms (įrangai, licencijoms, panašių sistemų pirkimams ar kitiems nenumatytiems atvejams)

lentelė 9 Išlaidų suvestinė

Išlaidas sudarantys elementai	1 mėnesio išlaidos	2 mėnesio išlaidos	3 mėnesio išlaidos	4 mėnesio išlaidos	5 mėnesio išlaidos	6 mėnesio išlaidos	Galimai 7 mėnesio	Visų mėnesių
Elektronikos inžinierius	1500	1500	1500	1500	1500	1500	1500	
Programuotojas				1500	1500	1500	1500	
Komandos lyderis	1500	1500	1500	1500	1500	1500	1500	
Mikrovaldiklis	30	25	20	15	15	15	15	
BLE modulis	5	5	5	5	5	5	5	
LoRa modulis	5	5	5	5	5	5	5	
Virdulys	20	20	15	15	15	15	15	
Relinis jungiklis	2	2	2	2	2	2	2	
Temperatūros jutiklis	2	2	2	2	2	2	2	
Patalpų nuoma	300	300	300	300	300	300	300	
Viso mėnesio išlaidos $	3364	3359	1849	3344	4844	4844	4844	26448

3.17 Naudotojo dokumentacija ir apmokymas

3.17.1 Reikalavimai naudotojų dokumentacijai

Aprašymas - sudaryti sistemos techninę specifikaciją
Tenkinimo kriterijus - Yra sudaryta sistemos techninė specifikacija

3.17.2 Reikalavimai naudotojų apmokymui

Aprašymas - sistemos sąsajoje sukurti trumpą tekstą vartotojams apmokyti
Tenkinimo kriterijus - Įėjus į mobilią programėlę yra atvaizduojama informacija kaip naudotis sistema

3.17.3 Perspektyviniai reikalavimai

Aprašymas - Įdiegti sistemoje funkciją leidžiančią vartotojams pasirinkti norimą temperatūrą
Tenkinimo kriterijus - Sistemos pasirinkti gėrimą puslapyje, yra mygtukas „pasirinkti temperatūrą“

Aprašymas -Įdiegti sistemoje mašininio mokymosi elementų, kurie autonomiškai parinktų tinkamą temperatūrą gėrimui pagal išorinius informacinius šaltinius
Tenkinimo kriterijus - Nebereikės kelti duomenų bazių į sistemą rankiniu būdu

3.18 Idėjos sprendimams

Ateityje panaudoti galingesnį serverį nei *Raspberry PI*
Prijungti kaitinamo termoso funkciją sistemoje.
Balsu valdoma sistema.

ARCHITEKTŪROS SPECIFIKAVIMAS

4.1 Įvadas

4.1.1 Dokumento paskirtis

Šio dokumento paskirtis yra pateikti planuojamo projekto kuriamos sistemos vaizdą aprašant ir pavaizduojant projekto architektūrą taip užtikrinant kuriamos architektūros potencialo suderinamumui tarp projekto užsakovo, projekto vadovo ir projekto kūrėjų. Šis dokumentas detalizuos projekto architektūros realizacijos svarbius aspektus ir sprendimus, kurie buvo priimti kuriant sistemą.

Šis dokumentas taip pat atliks Kauno Informatikos fakulteto Programų inžinerijos katedros „Projekto architektūros“ magistrinio projekto ataskaitos funkciją. Dokumento turinys apibrėš išmaniojo virdulio kaitinančio gėrimą pagal vartotojo poreikį architektūrinę sistema.

4.1.2 Apibrėžimai ir sutrumpinimai

Terminų, kurie bus naudojami specifikuojant reikalavimus žodynas. Čia galima rasti dažniausiai naudojamų žodžių, sutrumpinimų paaiškinimus naudojamus šioje architektūros specifikacijoje:

lentelė 10 Apibrėžimai ir sutrumpinimai

Pavadinimas	Apibūdinimas
IOT (*Internet Of Things*)	Išmanių daiktų sistema, kurioje daiktai gali tarpusavyje komunikuotis, ir siųsti ir priimti vienas kitam duomenis.
LoRa	LoRa yra ilgos distancijos mažai energijos naudojanti belaidė technologijos platforma, kuri naudoja nelicencijuotą radijo spektrą industrinėje, mokslinėje ir medicininėje radijo juostoje
BLE (*Bluetooth*)	Mažai energijos eikvojantis *Bluetooth* (BLE) buvo pristatytas kaip viena geriausių alternatyvų daiktų interneto rinkoje
802.11 (*Wi-Fi*)	802.11 protokolas yra viena pagrindinių belaidžių technologijų akivaizdžių technologinių kandidatų, kai kuriama daiktų internetų sistema.

Raspberry Pi	Mažas kompiuteris, kuris gali atlikti paprastas darbines funkcijas
Serveris	Kompiuterio aparatinės įrangos dalis, kuri suteikia funkcionalumą programoms ar prietaisams
IOS programėlė	Programa veikianti pagal specializuotos mobilios operacinės sistemos pagrindą
Android programėlė	Programa veikianti pagal specializuotos mobilios operacinės sistemos pagrindą
Google Home,	„Google Home“, yra išmaniųjų garsiakalbių linija, kurią „Google“ sukūrė pagal „Google home“ prekės ženklą. Įrenginiai leidžia vartotojams kalbėti balso komandomis ir sąveikauti su paslaugomis per „Google Assistant“, įmonės virtualų asistentą.
Amazon Echo	„Amazon Echo“ yra laisvų rankų įrangos garsiakalbių ir virtualių pagalbinių įrenginių grupė, kurie bendrauja su galutiniu vartotoju per „Amazon Alexa“ debesies balso paslaugą, produktų linija. „Echo“ įrenginiai prisijungia prie „Alexa“ paslaugos internetu.
MQTT	Publikavimos ir prenumeratos tinklo protokolas, skirtas siųsti žinutes iš vienos sistemos į kitą.

4.1.3 Apžvalga

Šis dokumentas aprašo išmanaus virdulio kaitinančio gėrimą pagal vartotojo poreikį architektūrą. Skyriuje „architektūros pateikimas“, bus nurodomi reikalingi vaizdai ir šiems vaizdams bus nurodomi elementai iš kurių yra sudaryti. Taip pat dokumente galima rasti „Architektūros tikslai ir apribojimai“ skyrius, kuriame bus nurodomi tikslai ir reikalavimai turintys esminę įtaką sistemos kūrimui ir šiai sistemai taikomi apribojimai. Taip pat yra pateikiama sistemos panaudojimo atvejai (skyrius „Panaudojimo atvejų vaizdas“). Po panaudojimų atvejų, kitame skyriuje bus nurodoma sistemos architektūra tiek statiniu, tiek dinaminiu aspektais. Taip pat architektūros specifikacijos „Duomenų vaizdas“ bus pateikiamas duomenų bazės modelis, pagal kurį dirbs išmanaus arbatinuko sistema“.

4.2 Architektūros pristatymas

Dokumente kuriamos sistemos architektūra pateikiama keliais vaizdais, vaizduojančiais sistemą skirtingais aspektais: panaudojimo atvejų vaizdu, statiniu vaizdu kuriame yra išskaidoma sistema į paketus ir pateikiama klasių diagrama, dinaminiu arba procesų vaizdu, kuriama atvaizduojama sekų diagrama, ir išdėstymo vaizdu. Vaizdus galima matyti *MagicDraw* modeliais ir naudojama unifikuota modeliavimo kalba (UML). Šie vaizdai atvaizduos sekančias diagramas:

- Panaudojimo atvejų:
  - Panaudojimo atvejų diagrama;
- Statinis vaizdas:
  - Sistemos skaidymo diagrama;
  - Klasių diagrama;
- Dinaminis vaizdas:
  - Bendradarbiavimo diagrama;
  - Sekų diagrama;
- Išdėstymo vaizdas:
  - Išdėstymo diagrama;

4.3 Architektūros tikslai ir apribojimai

Architektūrinius sprendimus pakeičiantys reikalavimai:

- Sistema turi veikti su mobiliu įrenginiu.
- Sistema turi būti naudojama tik asmenų, kurie turi virdulį.
- Sistemos kode neturi būti konfliktų *Android Studio* aplinkoje.
- Turi būti galimybė lengvai įterpti naujų komponentų algoritmų sistemos išmanaus virdulio kaitinančio pagal vartotojo gėrimo poreikį algoritme.
- Kuriamos sistemos algoritmas turi būti lengvai suderinamas su išorinė (Google *Home* ar *Amazon Echo*) sistema.
- Programėlės vaizdas (GUI), turi būti suprantamas ir mobilios programėlės funkcijos lengvai valdomas vartotojo.
- Vartotojo gėrimo pasirinkimas turi būti išsaugojamas mobiliojoje programėlėje
- Sistemos serverio Raspberry PI resursų turi užtekti prijungti sistemą *Bluetooth, LoRa* ir *Wi-FI* metodais.

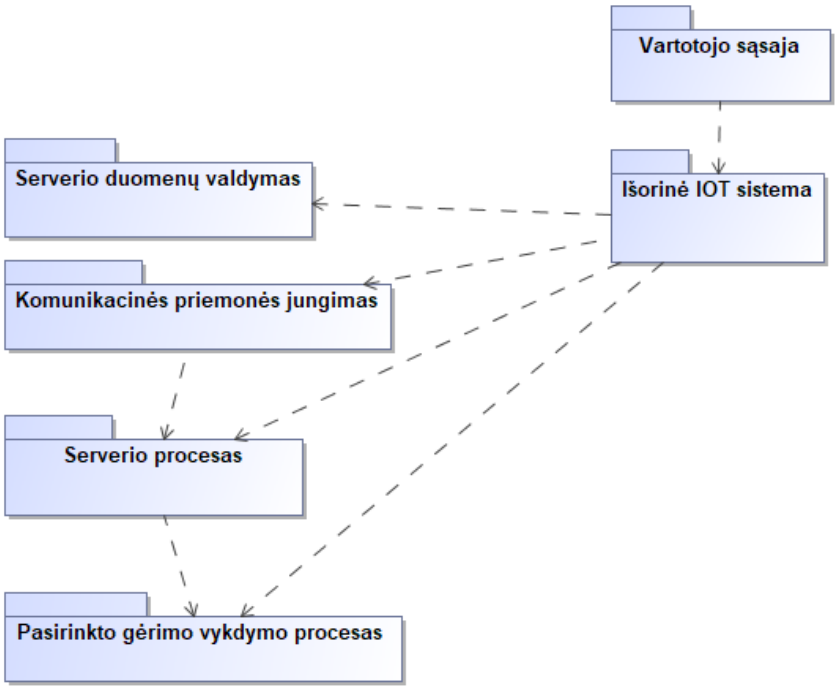
4.4 Panaudojimo atvejų vaizdas

Detalūs scenarijai ir panaudojimo atvejai yra aprašomi “sistemos paskirtis” skyrelyje.

4.5 Sistemos statinis vaizdas

4.5.1 Apžvalga

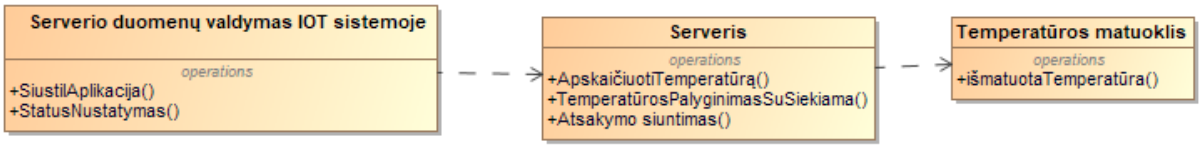
Sistemą galima suskaidyti į paketus (žr. pav. 16).



pav. 16 Sistemos paketai

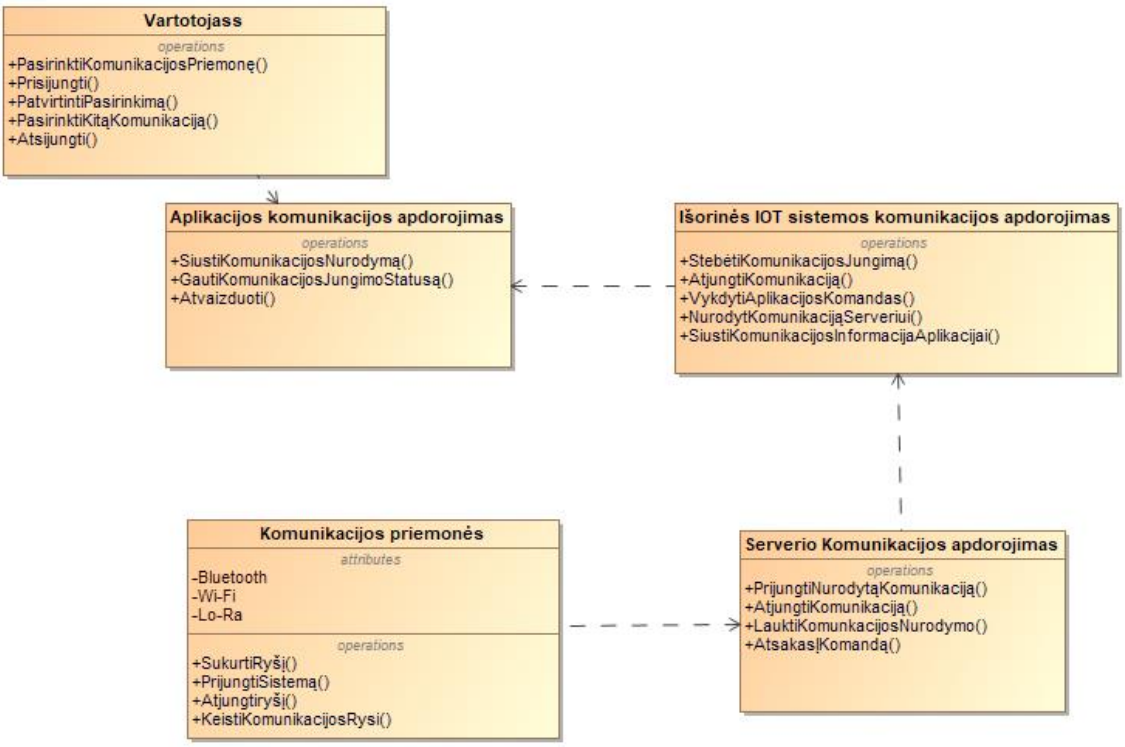
4.5.2 Paketų detalizavimas

Paketas Serverio duomenų valdymas yra klasės, kuriose vyksta serverio duomenų perdavimas išorinei IOT sistemai, kuri juos apdoroja ir siunčia programėlei. Pakete esančių klasių struktūra pateikiama valdymo klasės diagramoje (žr. 17 Pav.):



pav. 17 Serverio duomenų valdymo klasės diagrama

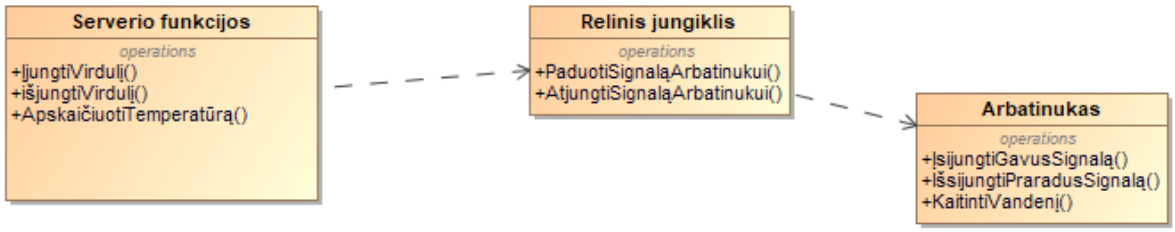
Paketas Komunikacinės priemonės jungimas skirtas nurodyti komunikacinių priemonių klasių procesus vykstančius tarp programėlės, Išorinės IOT sistemos ir serverio. Ši diagrama yra pavaizduota komunikacinės priemonės (žr. 18 Pav.) diagramoje:



pav. 18 Komunikacinės priemonės jungimas klasės diagrama

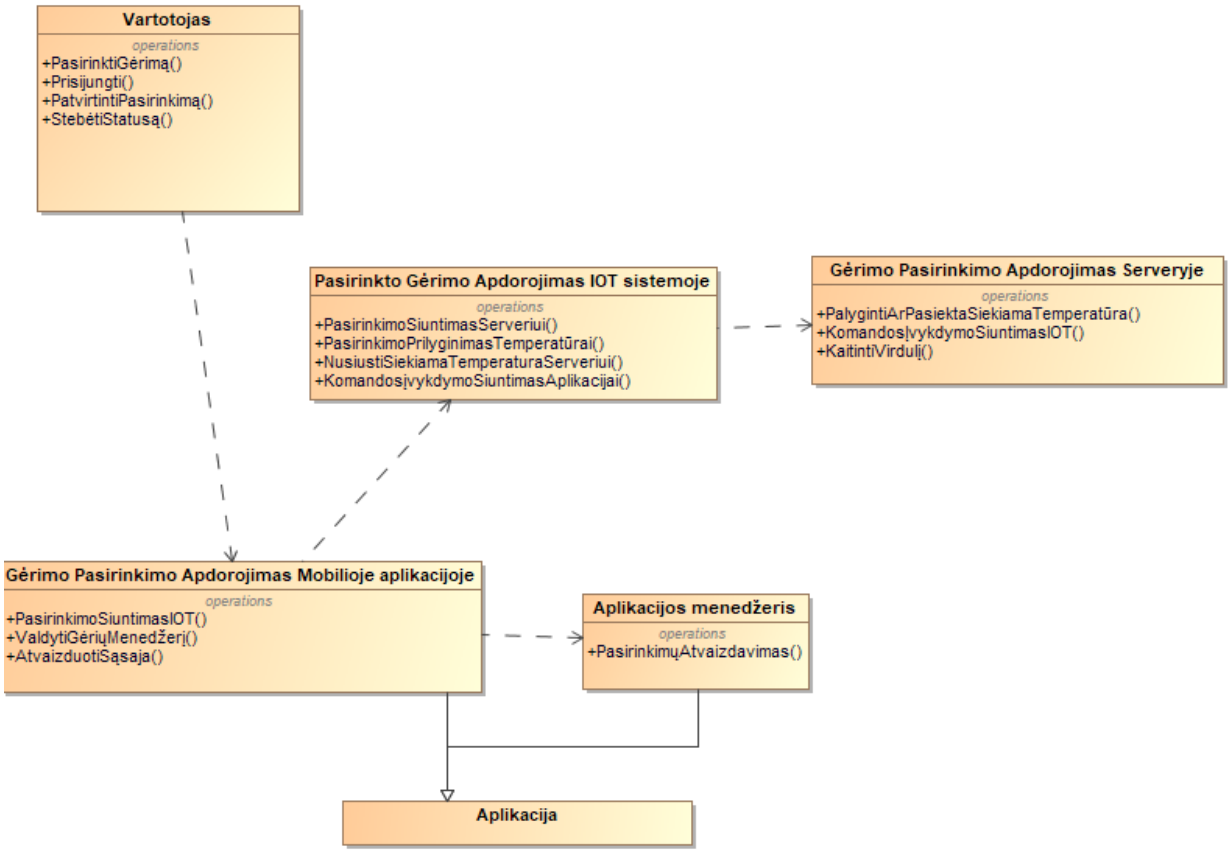
4.5.3 Serverio procesas

Paketas Serverio procesas skirtas nurodyti serverio klases, kurios vyksta serverio planuojamoje *Raspberry Pi* formoje, kurios funkcijomis gali pasinaudoti keli panaudojimo atvejai.



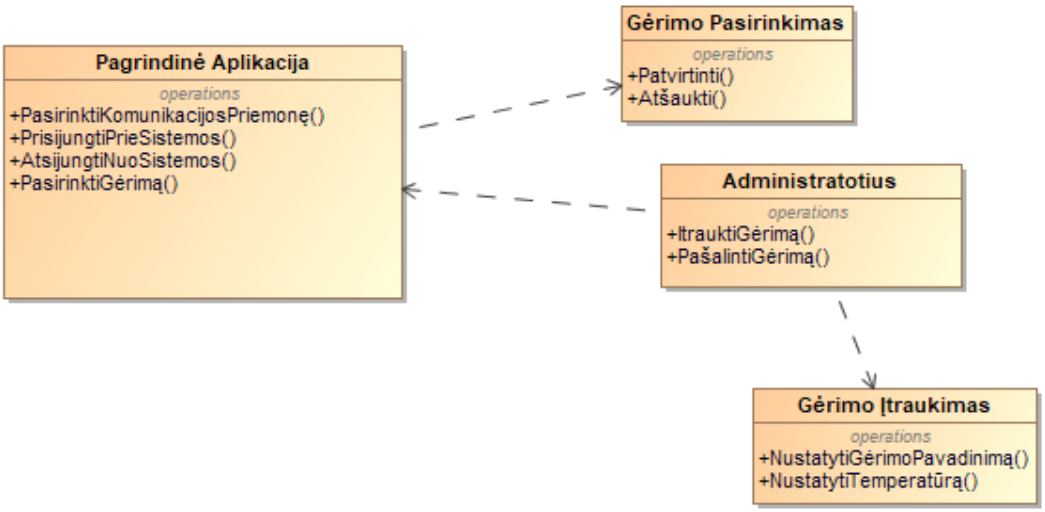
pav. 19 Serverio procesas klasės diagrama

Paketas pasirinkto gėrimo vykdymo procesas skirtas parodyti kaip pasirinkto gėrimo informacija pasieks serverį ir kaip bus pasiekiamas (žr. pav. 20).



pav. 20 Pasirinkto gėrimo vykdymo proceso klasės diagrama

Paketas „Vartotojo sąsaja“ skirtas parodyti kas bus matoma vartotojo mobiliojoje programėlėje klasėmis.



pav. 21 Vartotojo sąsajos klasės diagrama

Paketas „Išorinė IOT sistema“ skirta parodyti papildomas klasės ir yra skirta valdyti ir administruoti kuriamą sistemą, iš išorinės IOT sistemos.

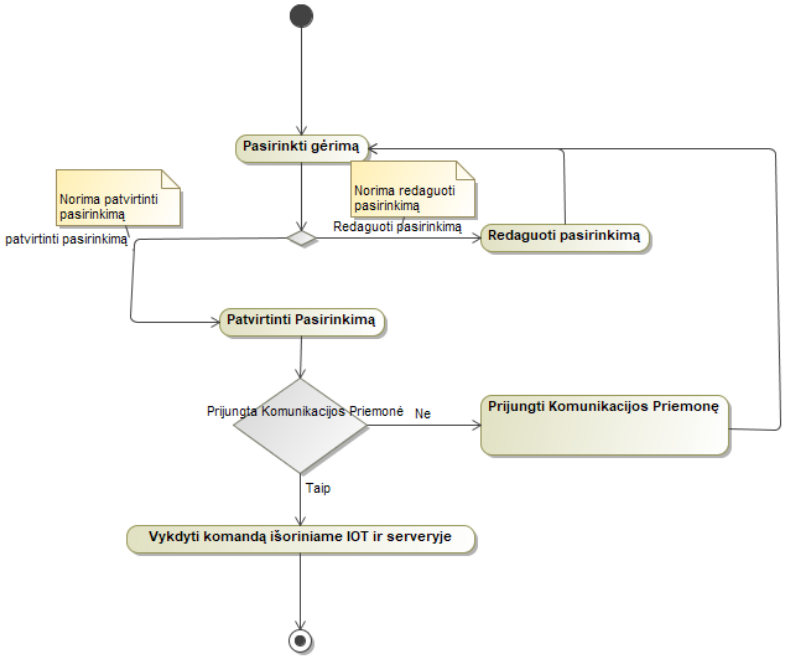
pav. 22 Išorinės IOT sistemos klasės diagrama

4.6 Sistemos dinamika

Šiame skyriuje pateikiamos sąveikos, būsenų bei veiklos diagramos. Šios diagramos padengia bendrą procesą naudojimosi kuriamu įrankiu veiklos procesą, bei leidžia projektą stebintiems ir sistemą kuriantiems žmonės geriau matyti koks funkcionalumas bus pateikiamas sukurtoje sistemoje.

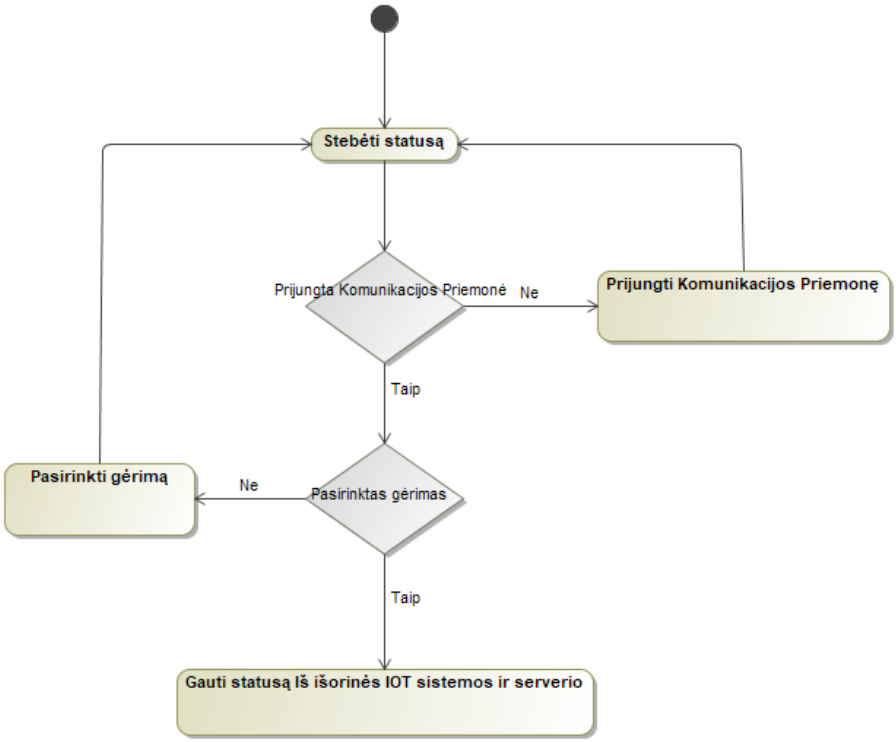
4.6.1 Veiklos diagramos

Veiklos diagramoje pateikiama „Pasirinkti Gėrimą“ ir „Stebėti statusą“ panaudojimo atvejai. Pirmą diagramą, kuriame atvaizduojamos gėrimo pasirinkimo funkcijos. Šią veiklos diagramą, galima vykdyti neribotą skaičių kartų ir jos struktūra yra nurodyta žemiau (žr. 23 Pav.).



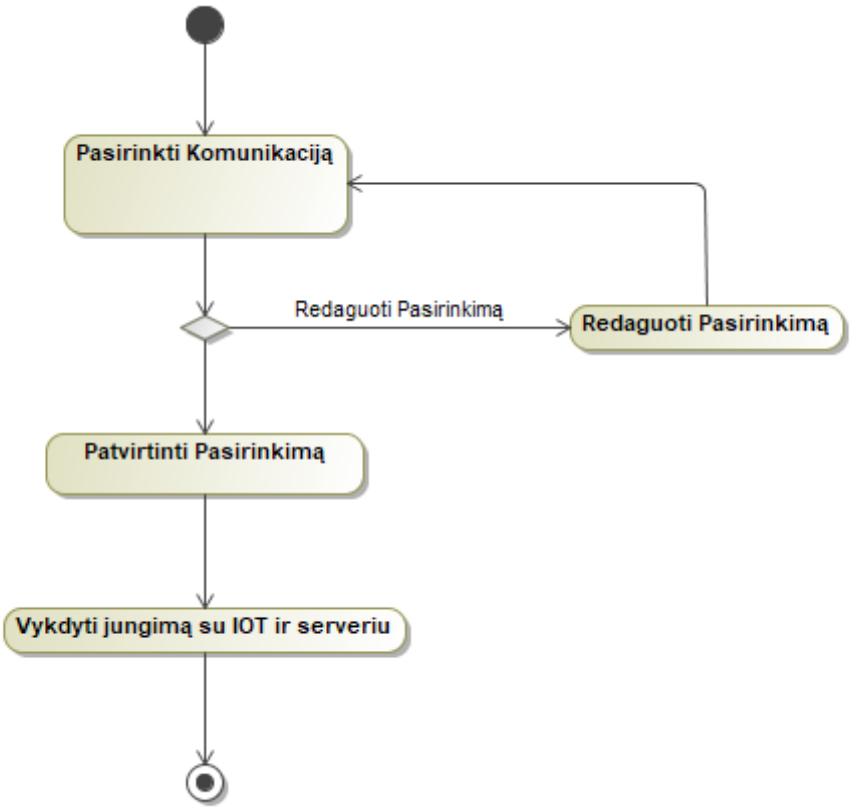
pav. 23 Pasirinkti gėrimą veiklos diagrama

Kitoje diagramoje yra nurodoma statuso stebėjimo panaudojimo atvejo veiklos diagrama, kurioje galima matyti Statuso pasirinkimo eiliškumas:



pav. 24 Stebėti statusą veiklos diagrama

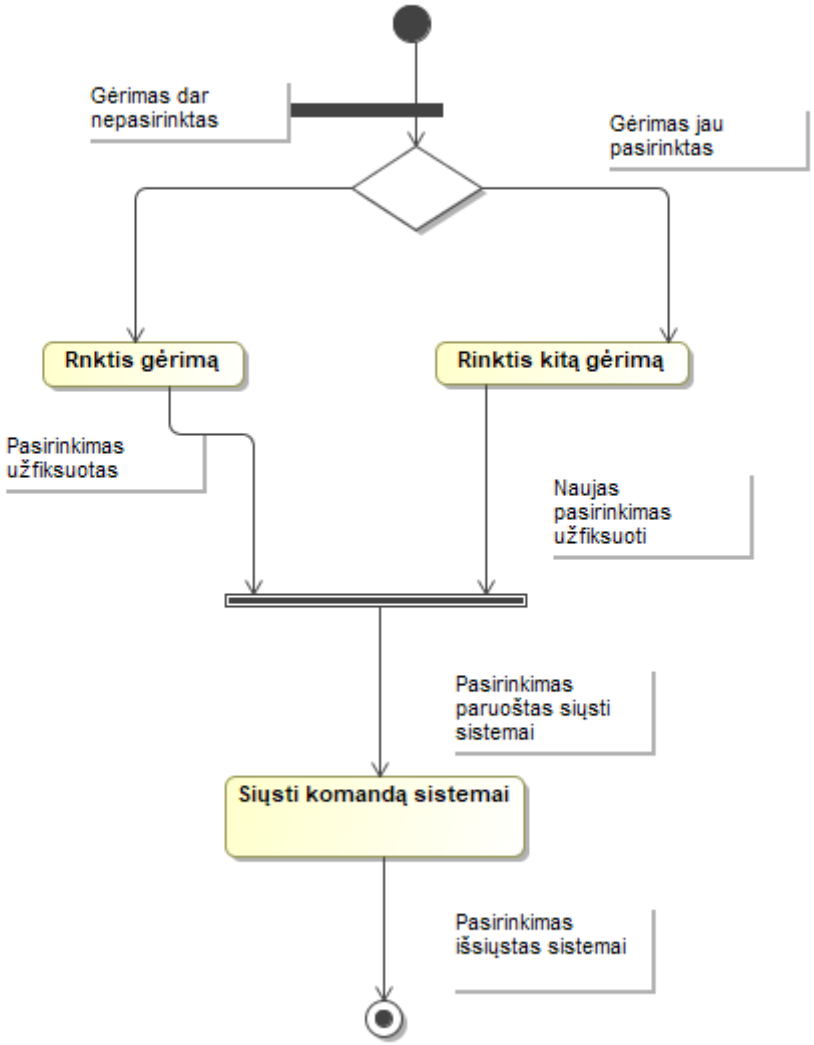
Ir paskutinėje diagramoje yra atvaizduojamas „pasirinkti komunikaciją“ veiklos diagrama, kurioje atliekamos ryšių jungimas prie sistemos (žr. 25 pav.).



pav. 25 Pasirinkti komunikaciją veiklos diagrama

4.7 Būsenų diagramos

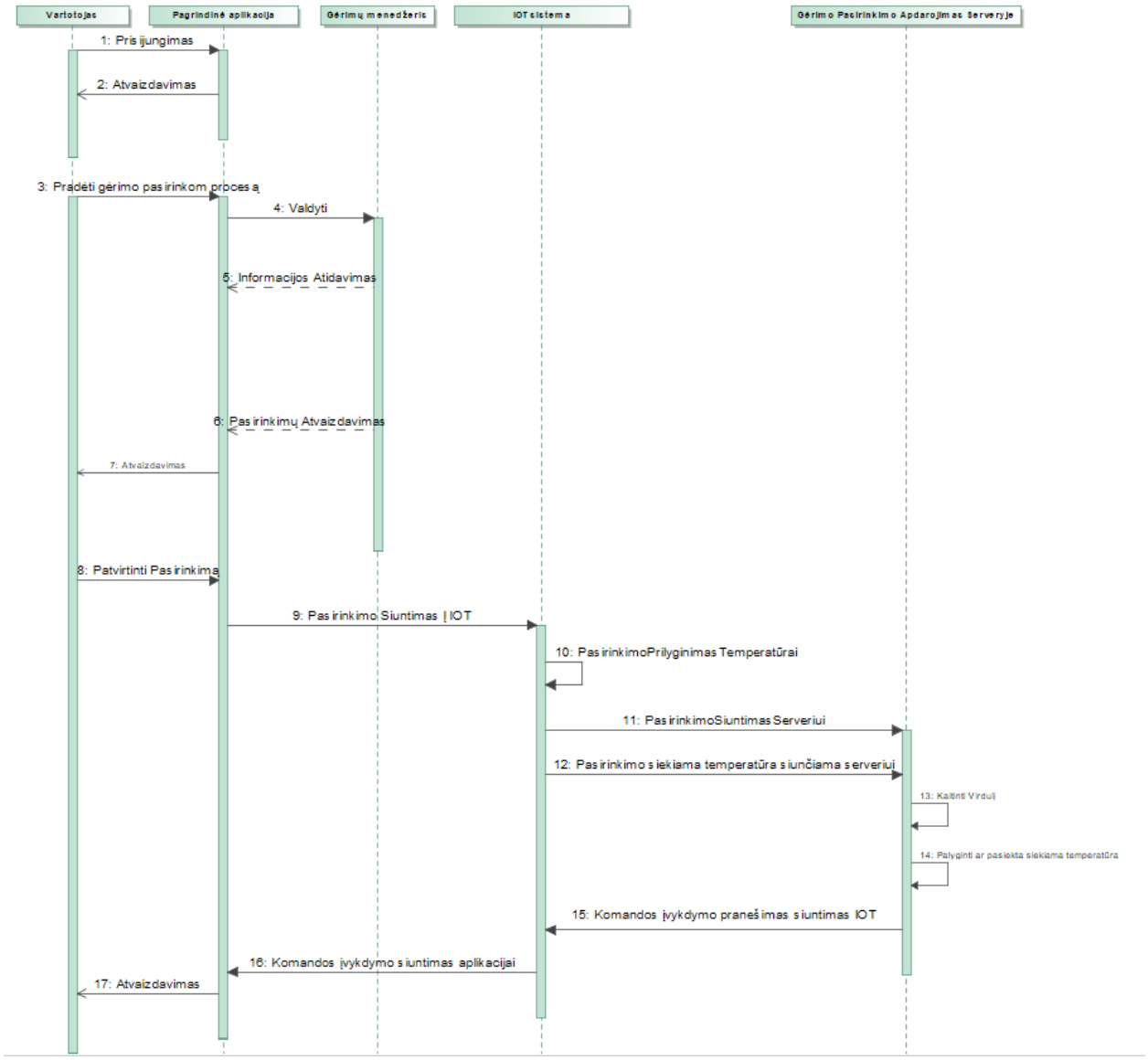
Būsenų diagramoje galima matyti, situacija, kai vartotojas bando pasirinkti gėrimą, jei prieš tai jau buvo pasirinkęs gėrimą. Šiuo atveju, vartotojui nereikia kurti naujos užklausos, o tik paspausti redaguoti pasirinkimą ir pasirinkti visai kitą gėrimą. Tai galima matyti atvaizduotą diagramoje (žr. 26 Pav.).



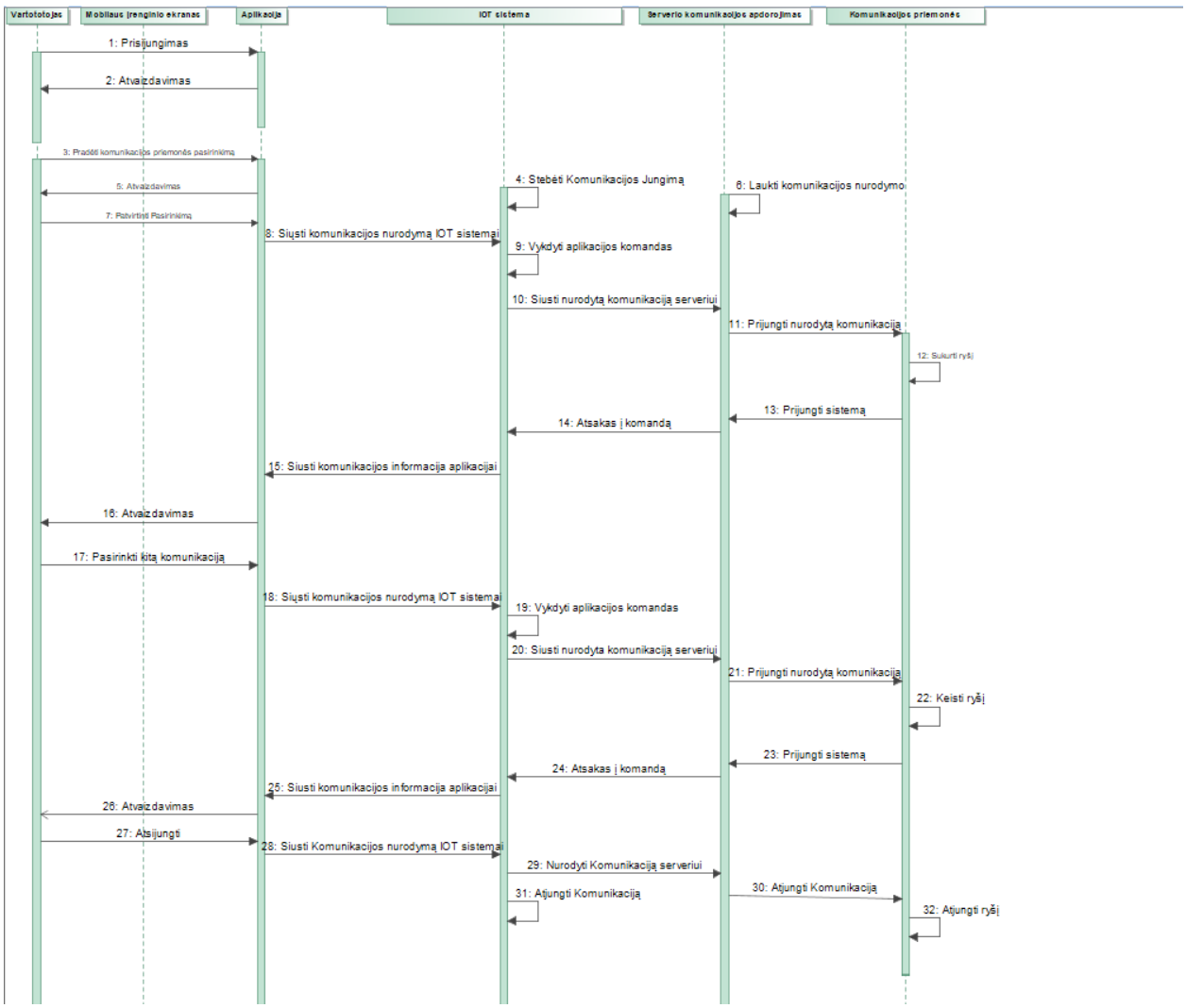
pav. 26 Pasirinkti gėrimą būsenos diagrama

Visiškai analogiškai veikia ir pasirinkti komunikacijos priemonę funkcija, todėl diagrama šiai funkcijai nebus atvaizduota.

4.8 Sekų diagrama

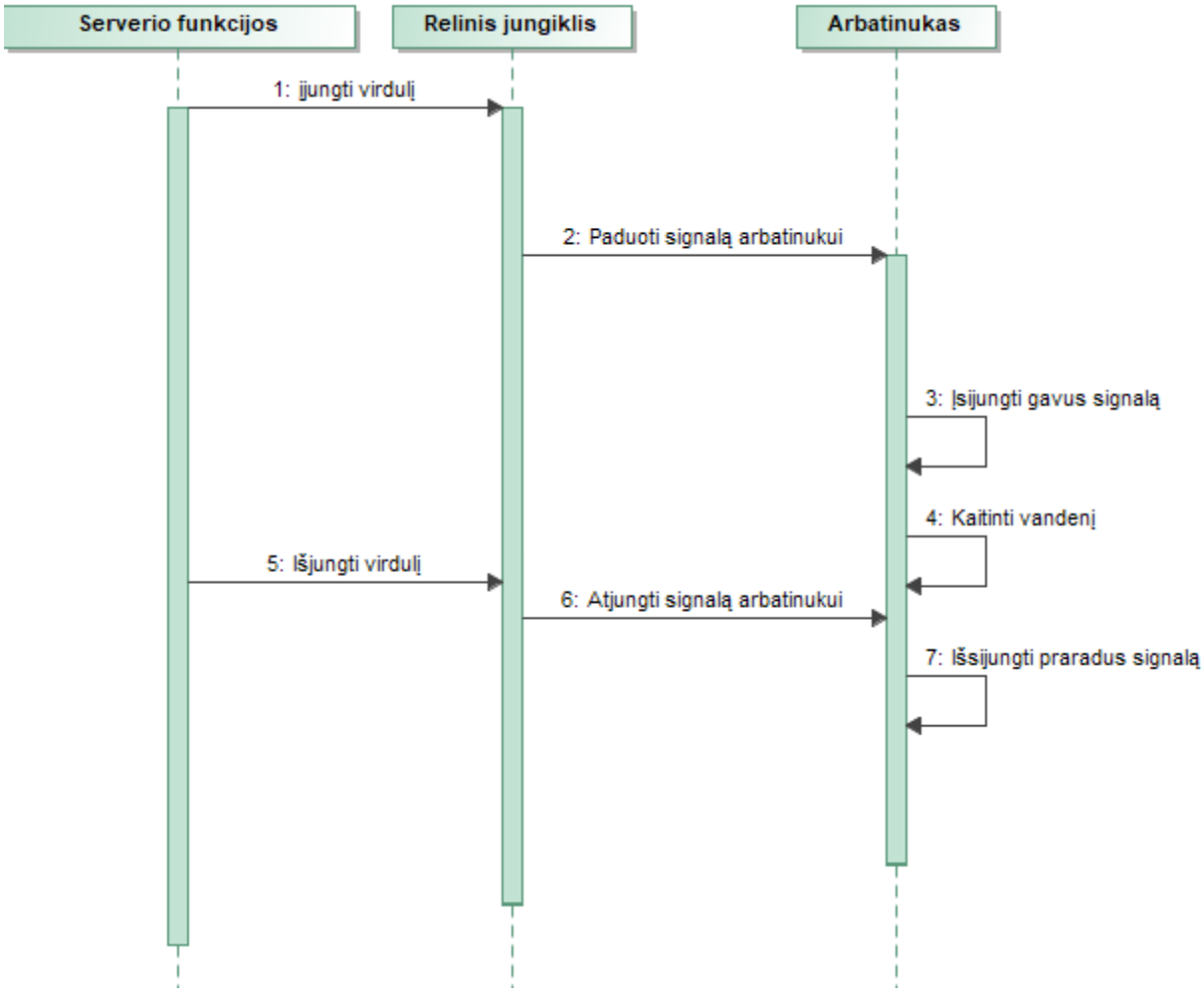


pav. 27 Pasirinkti gėrimą ir prisijungti prie sistemos sekos diagrama



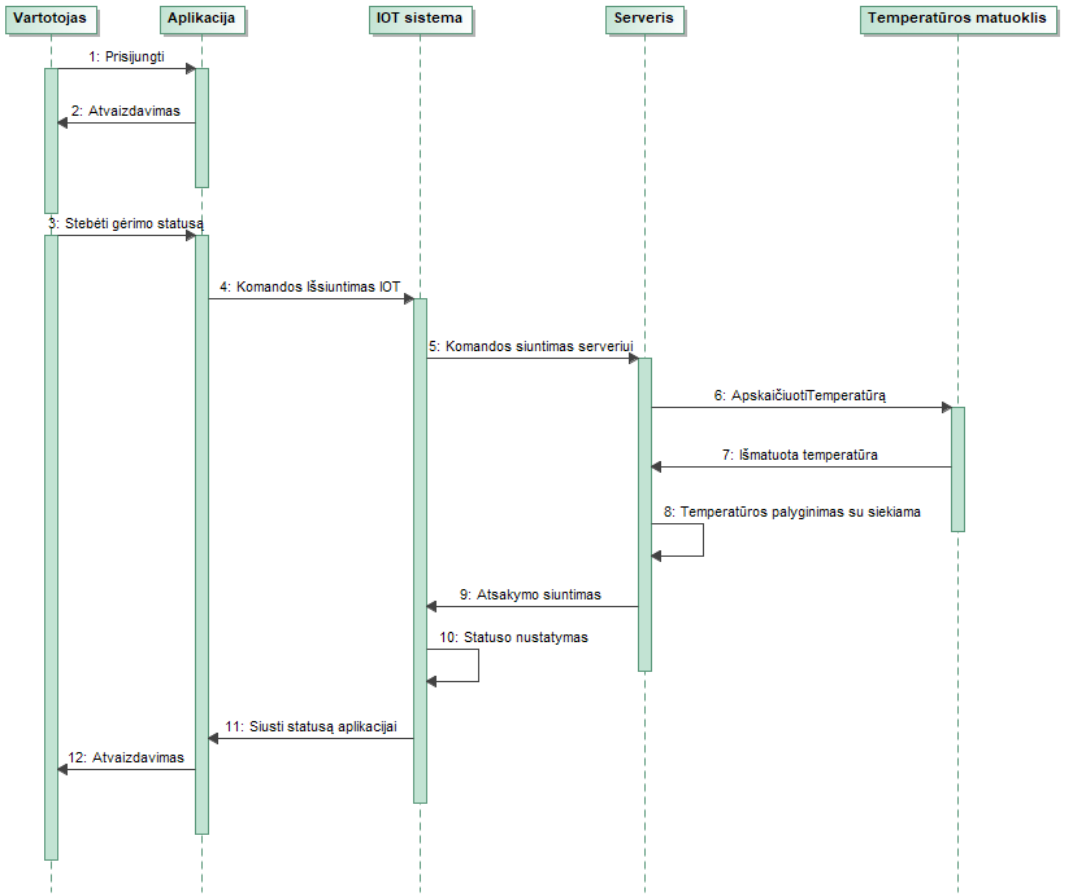
pav. 28 Pasirinkti ar pasirinkti kitą komunikaciją, atsijungti ir prisijungti prie sistemos sekos diagrama

Serverio arbatinuko (arbatinuko įjungimo ir išjungimo panaudojimo atvejai) (žr. pav. 29).



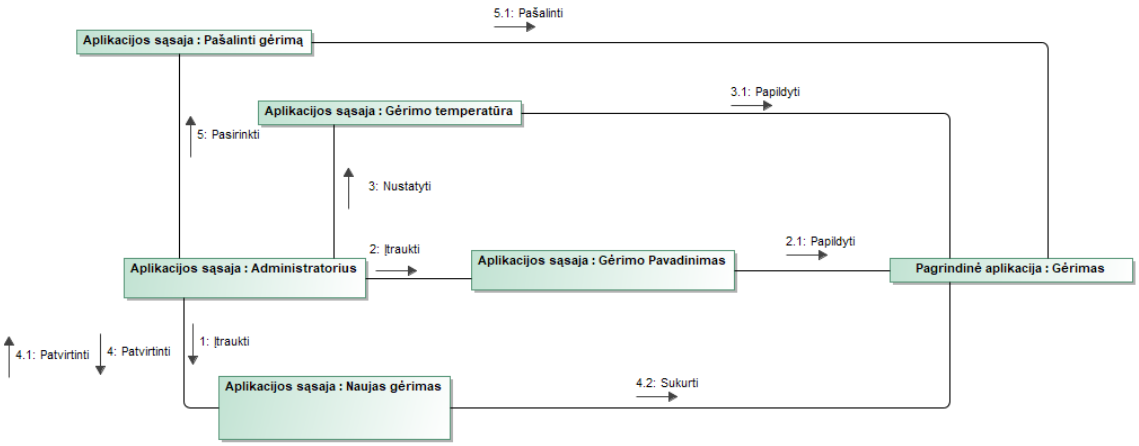
pav. 29 Įjungti ir išjungti virdulio sekos diagrama

Statuso stebėjimo trasos diagrama (žr. pav. 30).



pav. 30 Stebėti gėrimo statusą sekos diagrama

Buvo sukurta ir programėlės sąsajos administratoriaus bendradarbiavimo diagrama (žr. 31 pav.).



pav. 31 Stebėti gėrimo statusą sekos diagrama

4.9 Išdėstymo (*deployment*) vaizdas

Sistema veiks trijose platformose *Google Cloud, Android* mobiliojoje programėlėje kuri bus prijungta prie *Google assistant* ir mikrovaldiklyje. Jos yra išskiriamos į programėlę, IOT ir sistemą dėl aiškumo. Android programėlė bus sukurta naudojant *Android Studio* platforma. Mikrovaldiklis bus programuojamas *Visual studio platform*. Ir *Google Cloud* bus programuojama naudojant joje integruotomis „*Functions*“ funkcijomis. *Android* platformai reikės tenkinti šiuos reikalavimus sistemos veikimui:

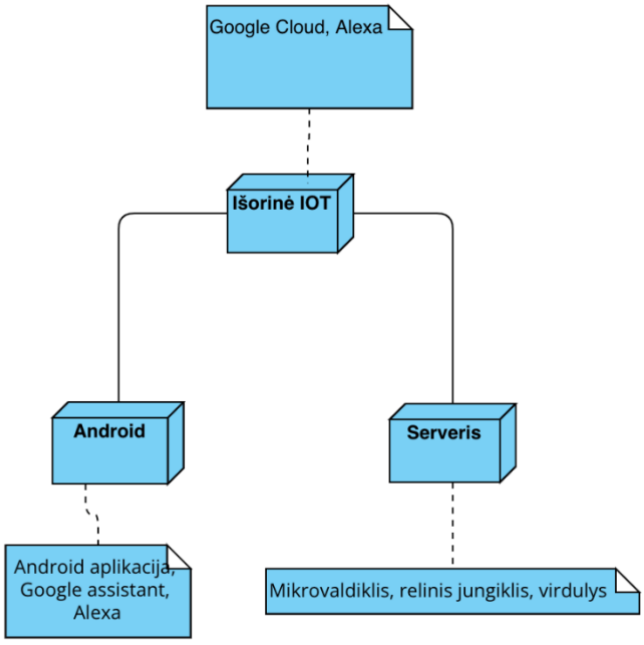
- *Android* 6.0+ nes *Android Studio* bus programuojama būtent šiai versijai.
- Daugiau nei 1 GB atminties (RAM).
- “*Play store*”.
- 1920 ekrano raiška.

Google Cloud nes platforma laikoma debesyje neturi specifinių reikalavimų įrangai.

Reikalavimai serveriui:

- Atmintis (RAM) 512 MB.

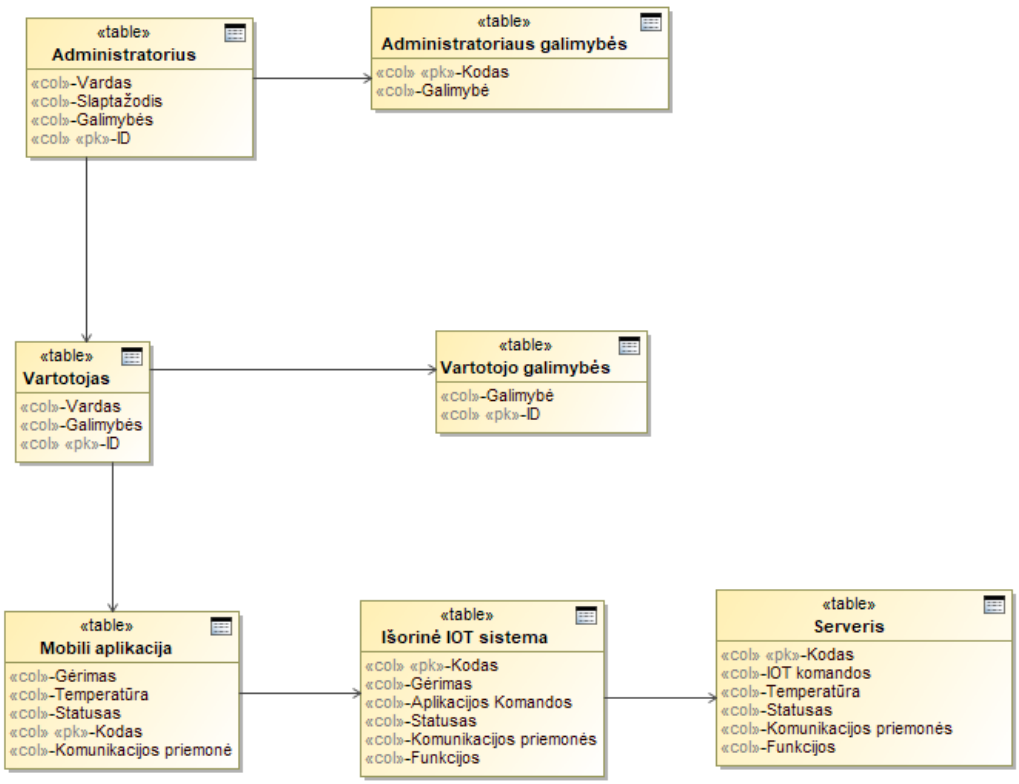
Programos išdėstymo vaizdas pavaizduotas žemiau (žr. 32 Pav.).



pav. 32 Sistemos išdėstymo vaizdas

4.10 Duomenų vaizdas

Pateikiamas duomenų bazės modelį galima stebėti žemiau pavaizduotame paveikslėlyje (žr. 33 Pav.).



pav. 33 Sistemos duomenų bazės modelis

4.11 Kokybė

Sistemos architektūra pagal poreikį leis proporcionaliai didinti jos galimybes ir sudėtingumą, kas reikš didesnį sistemos funkcionalumą. Sistema veiks principu, kad būtų lengvai integruojama į sudėtingesnę (*Google, Alexa*) sistemą, ko pasėkoje naudojant *Google „Functions“* bus įmanoma kurti naujas funkcijas serveriui.

Programėlė, IOT ir serveris būdami nepriklausomi vienas kito atžvilgiu leis lengviau didinti šių sistemos komponentų potencialą nepaveikiant kitų dviejų sistemos komponentų.

Google Cloud (*Amazon alexa*), būdama milijonų naudojama platforma ir turėdama daugybė serverių, leidžia užtikrinti sistemos našumą, saugumą ir patikimumą.

Visual studio suderinamumas su daugybę mikrovaldiklių platformų leis užtikrinti lengvą migravimą iš *Raspberry PI* į labiau pažengusį mikrovaldiklį. Taip pat ši platforma turėdama didžiulį įskiepių biblioteką, leis užtikrinti kodo naudojimą vienoje platformoje.

Android sistema, kuri yra pasaulyje plačiausiai naudojama platforma, užtikrins, kad sistema bus galima nuolatos palaikyti ir sumažins riziką poreikiui migruoti į kitą platformą.

Sistemos trijų komunikacijos priemonių naudojimas, užtikrins, kad egzistuos papildomų ryšių, kilus nesklandumų ar pasibaigus palaikymui vieno ryšio.

Sistemos mobili programėlė būdama APK formato, leis įkelti sistemą į „*Google play store*“ arba perkelti į bet kurią kitą *Android* platformą ir tęsti veiklą ten.

4.12.1 Testavimo apimtis ir tipai

Testavimas apima mobiliojoje programėlėje, serveryje veikiančios posistemės bei vizualizacijos posistemės testavimą, įskaitant diegimą. Pilna programinė įranga bus testuojama naudojant tokius testavimo būdus, kurie išvardinti chronologine tvarka:

1. Vienetų testavimas. Šio testavimo metu testuojami atskiri metodai bei klasės.

2. Integravimo testavimas. Jo metu užtikrinama, kad tarpusavyje bendradarbiaujantys sistemos komponentai bendrauja teisingai.

3. Validacijos testavimas. Tikrinimas sistemos atitikimas specifikacijai pagal vartotojo panaudojimo atvejus. Testuojamas virduliui perduodant duomenis programėlei.

4. Diegimo testavimas. Vartotojo telefonuose bus testavimo būdu diegiama sistemos programėlė ir tikrinamas procesas.

5. Našumo testavimas. Testuojamas virdulio atsako komandoms greitis ir informacijos pateikimas didėjant duomenų perdavimo sesijos ilgiui.

6. Konfigūracijos testavimas. Skirtingų konfigūracijų palaikymo testavimas vartotojo telefone iš programėlės.

4.12.2 Pagrindiniai apribojimai

Testavimo apribojimai:
• Vartotojui skirta programėlė bus testuojama naudojant Android operacinę sistemą.
• Serveriui bus naudojamas ESP32 prietaisas.
• Aplikacija ir serveris bus kuriami Java ir C kalbomis.

4.12.3 Pagrindiniai apribojimai

• Vartotojui skirta programėlė bus testuojama naudojant Android operacinę sistemą.
• Serveriui bus naudojamas ESP32 prietaisas.
• Aplikacija ir serveris bus kuriami Java ir C kalbomis.

4.12.4 Testuojama programų sistema

Testuojama išmanaus virdulio sistemą, sudaro trys posistemės. Pirma – Android programėlė veikianti vartotojo telefone valdanti išmanų virdulį ir serverį. Jos paskirtis – priimti vartotojo komandas, siųsti į serverį valdymui *Bluetooth, LoRa* ir *Wi-Fi* ryšiai ir valdyti duomenis *Firebase* duomenų bazėje. Antra – ESP32 pagrindu valdomas serveris. Jos paskirtis, yra priimti komandas iš programėlės ir perduoti komandas išmaniam virduliui bei siųsti surinktą virdulio informaciją programėlei. Trečioji posistemė yra išmanusis virdulys kuris turės temperatūros jutiklį, lygio matuoklį kurie bus naudojami matuoti temperatūrą ir lygį ir siųsti informaciją į serverį. Posistemė taip pat pasižymės vykdydama serverio komandas. Be šių atžvilgių, sistema taip pat kuriama su tikslu lengvai praplėsti ir integruoti į *Google Home* sistemą, kad būtų galima pridėti prie kitų išmanių prietaisų valdymo.

4.12.5 Testavimo ištekliai

Testavimui reikalingų programinių ir techninių resursų sąrašas.

• ESP32 sistema – aparatinė įranga.
• Išmaniojo virdulio valdymo sistema – programinė įranga.
• Programuotojas ir/ar testuotojas.

4.12.6 Testavimo rezultatai

Testavimo rezultatams kaupti naudojam *Github* platforma, kurioje yra išeities kodas. Tai patogus būdas vesti, sekti rezultatus naudojant sistemą ir susieti juos su konkrečiu kodu. Be to, testuotojas ir programuotojas gali nesudėtingai matyti testavimo progresą ar esamas klaidas. Nuoroda į platformą: https://github.com

4.12.7 Testavimo įrankiai ir aplinka

Automatiniams vienetų ir integraciniams testams bus naudojama *Android Studio* platforma bei *firebase database realtime API* duomenų bazių valdymo programa kurios pagalba valdoma Firebase. Detalesnės bibliotekos pateikiamos 3 skyriuje atskirai prie kiekvieno iš testų metodų.

1) Testavimo reikalavimai programinei aplinkai.
    - Android 6+;
    - Atmintis (RAM) ne mažesnė nei 1 GB;
    - 1 GB kietasis diskas

- „Google play“ servisai;
- 720p rezoliucija;

2) Produkcinė aplinka:
   - Android 6+ versija;
   - Atmintis (RAM) turėtų būti didesnė nei 1 GB;
   - 2 GB kietasis diskas;
   - „Play store“ servisai;
   - 1920p rezoliucija;

   Serveriui keliami reikalavimai:
   - Atmintis (RAM) bent turėtų būti 512 MB;
   - Reikalingas bent 450 Mhz spartos procesorius (CPU) apdoroti duomenis;

4.12.8 Testavimo tvarkaraštis

Testavimo darbų grafikas apima testavimo veiksmus ar užduotis, pradžios ir pabaigos datas bei atsakomybę. Jame taip pat turėtų būti aprašyta, kaip testas bus peržiūrėtas, stebimas ir patvirtintas.

lentelė 11 Testavimo tvarkaraštis

Testavimo užduotis	Pradžia	Pabaiga	Atsakomybė
Testavimų planas	2022-09-15	2022-10-01	Aldas Jonauskis, Virginija Limanauskienė
Vienetų testavimas	2022-10-01	2022-11-15	Aldas Jonauskis, Kęstutis Motiejūnas
Integravimo Testavimas	2022-10-10	2022-11-15	Aldas Jonauskis
Diegimo testavimas	2022-11-01	2022-12-15	Aldas Jonauskis, Vartotojas
Našumo testavimas	2022-10-15	2022-12-15	Aldas Jonauskis
Konfigūracijos testavimas	2022-11-01	2022-12-15	Aldas Jonauskis, Kęstutis Motiejūnas,
Priėmimo testavimas	2022-01-03	2022-01-10	Projekto vadovas, Komisija

• Testavimo planas bus įkeltas į IS ir “Moodle”.
• Vienetų testavimas atliekamas keičiant kodo eilutes sistemoje, programuotojo.
• Integravimo testavimas bus įvykdytas programuotojui testuojant kaip veikia integruojamos sistemos
• Planuojama atlikti diegimo testavimą programuotojui įdiegus sistemą

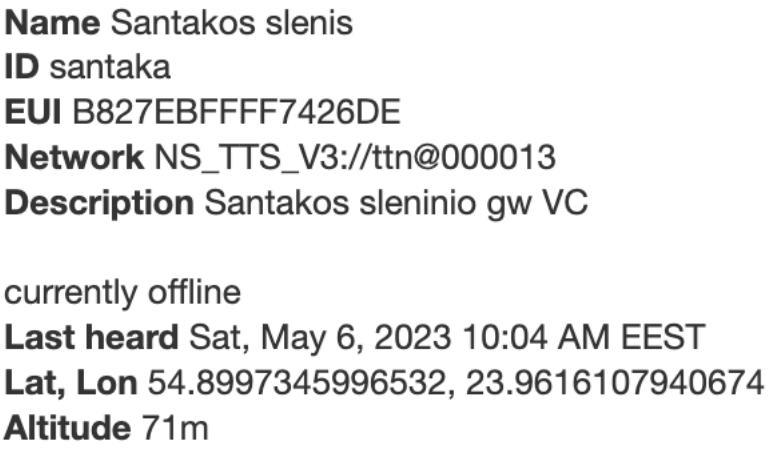
• Našumo testavimą bus atliekamas programuotojo, kuris tikrins sistemos spartą ir paleis įvairius scenarijus, kad būtų visų scenarijų našumas būtų ištestuotas.
• Programuotojas keis duomenų įvestį ir stebėdamas ir patvirtindamas rezultatus vykdys konfigūracinį testavimą.
• Priėmimo testavimas bus atliekamas pristatant galutinį produktą gavėjui ir gaunamas grįžtamasis ryšis produkto atžvilgiu.

5 Tiriamoji dalis

5.1 Įvadas

Sistemos tiriamoji dalis buvo atliekama naudojant trijų tipų aparatinės įrangos konfigūracijas:

- 1 kanalo “SparkFun LoRa gateway” – Šis 1 kanalo siųstuvas gali siųsti 1 komandą kas dvi minutes.
- 8 kanalų Rak2245 *Raspberry* PI 4 siųstuvas – Šis 8 kanalų siųstuvas gali gauti vienu metu informaciją iš 8 skirtingų šaltinių, bet tik gali išsiųsti 1 komandą tuo pačiu metu.
- „Santakos slėnio“ *LoRa* siųstuvas – KTU universitete esantis siųstuvas. Platesnę šio siųstuvo charakteristiką galima matyti iš TTN platformos {4}:



pav. 34 Santakos slėnio siųstuvo konfigūracija

Šios trys *LoRa* konfigūracijos atliko vartų siųstuvų (angl. *gateway*) funkciją. Prietaisas siųstuvas, kuris siųs duomenis į vartus, buvo pasirinktas „*Sparkfun 1 channel gateway*“. Šis *LoRa* siųstuvas buvo pasirinktas dėl nedidelio kiekio siunčiamų duomenų, ir dėl to, kad siųstuvą sudaro tas pats mikrovaldiklis, kurį naudoja ir serveris – ESP32. Tai užtikrino paprastą integralumą.

Pirmasis tyrimas atliktas 2023 m. sausio mėnesį naudojant šias tris aparatinių įrangų konfigūracijas, kurios buvo integruotos į bendrą suprojektuotą išmaniojo virdulio sistemą. Tai reiškia, kad sistemos *LoRa* komunikacija gali veikti naudojant bet kurią iš trijų skyriuje aprašytų konfigūracijų. Tyrimas atliktas KTU slėnyje, norint pasinaudoti TTN brokerio pagalba prieinamu „Santakos slėnio“ siųstuvu ir, kad šis siųstuvas būtų kuo įmanoma arčiau kitų naudojamų siųstuvų.

Antrasis tyrimas atliktas integravus ir pasiekus bazinį MQTT funkcionalumą sistemoje.

Eksperimentinis tyrimas buvo atliekamas paskutinis, pilnai integravus MQTT funkcionalumą į sistemą.

5.2 Pirmasis tyrimas

Ilgą laiką norint dirbti su *LoRa* technologijomis, buvo reikalinga sudėtinga ir reliatyviai brangi aparatinė ir programinė įranga. Bet tobulėjant technologijoms ir atsiradus atviriems tinklų serveriams kaip pvz. „The things network“, kaip niekada tapo lengva prijungti *LoRa* prietaisus prie serverio. Šiame tyrime buvo testuojamos trys aparatinės įrangos konfigūracijos. Šiai įrangai reikėjo siųsti pastovų 3 baitų pranešimą į TTN atvirą serverį – Išmatuotą gėrimo temperatūrą ir jo būseną. Žinutės pavyzdinį turinį siunčiamą naudojantis „Arduino“ IDE galima matyti žemiau (žr. 35 Pav.).

```
Statusas = "60°, kaitinamas"
```

pav. 35 Žinutė kuris bus siunčiama naudojantis LoRa *komunikacija (žinutė prasideda po lygu ženklo)*

Buvo atliktas tyrimas, kuriame tikrinama kaip sėkmingai šios žinutės pasiekia tinklą. Tyrimas atliktas keičiant tris parametrus:
Laiko intervalas - Laiko tarpas po kurio vėl siunčiama žinutė į *LoRa* serverį.
Sklidimo faktorius – Keičiamas sklidimo faktorius nuo 7 iki 12.
Aparatinė įranga – Tyrimui naudojamas trys skyriaus pradžioje minėtos aparatinės įrangos.

Keičiant šiuos tris parametrus buvo matuojama kokiu procentu žinutė gali sėkmingai pasiekti tinklą. Prieš tyrimą buvo suformuluota tyrimo hipotezė. Atlikti tyrimo rezultatai įtvirtinto hipotezę, kad daugiausiai ir stabiliausiai žinutes perduoda nuosavas siųstuvas/vartai.

5.2.1 Įžanga į tyrimą

LoRaWAN (angl. *Long Range Wide Area Network*) yra protokolas, sukurtas belaidžiams baterija įkraunamiems įrenginiams prijungtiems regioniniame, nacionaliniame ar pasauliniame tinkle. Dažnas šio protokolo panaudojimas atsiranda daiktų interneto (IoT) pritaikyme, nes jis gali siųsti nedidelius duomenų kiekius dideliais atstumais ir su minimaliomis energijos sąnaudomis [39]. Daiktų tinklas (angl. *The things network*) (TTN) yra pasaulinis, atviros prieigos, sutelktųjų šaltinių LoRaWAN tinklas, veikiantis kaip daiktų interneto įrenginių, naudojančių *LoRaWAN* protokolą, pagrindas {5}.

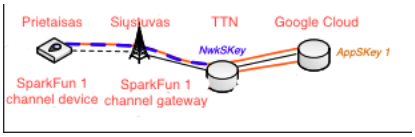
Siunčiant duomenis į šį tinklą, itin svarbu užtikrinti, kad visi duomenys siunčiami į tinklą jį sėkmingai pasiekia su skirtingomis konfigūracijomis ir skirtingo siuntimo dažniu. Svarbumas yra dar labiau įtvirtinamas, nes tai yra kertinis taškas norint suprasti *LoRaWAN* protokolo ir TTN veikimą ir patikimumą. Prarastų duomenų procentas gali būti paveiktas *LoRa* signalo stiprumo, numatyto platformos procentinio duomenų paketų praradimo ir pristatymo laiko. Analizė gali padėti nustatyti ir pašalinti galimus TTN sistemos trūkumus ar kliūtis, kurie vis dar nėra išspręsti. Šis tyrimas fokusuosis ties duomenų perdavimu kai visi siųstuvai yra 100 metrų atstumu vienas nuo kito.

Šio tyrimo poreikis yra įtvirtinamas dėl IOT augimo. IOT ir toliau augant, daugės įrenginių, kurie remiasi *LoRaWAN* ir tinklais, pvz. TTN. Suprasdami dabartinį duomenų praradimo mastelį, galima suprasti tinklo plėtimo potencialą ir nustatyti galimas problemas, kurios gali iškilti, kai bus prijungta daugiau įrenginių. Kaip buvo atlikta Jetmir tyrime, daugėjant įrenginių, didėja siųstuvų vartų poreikis [40].
Tai pat galima paminėti ir paslaugos kokybę. Daugeliui daiktų interneto prietaisų labai svarbus patikimas pranešimų pristatymas. Pavyzdžiui, naudojant aplinkos stebėjimą ar sveikatos priežiūros programas, didelis duomenų praradimų skaičius gali sukelti problemų. Todėl norint užtikrinti patikimą paslaugų kokybę, svarbu suprasti ir sumažinti duomenų praradimo skaičių.
Nesėkmingi perdavimai taip pat kartais gali reikšti saugumo problemas, pvz., trukdžius ar kitokio pobūdžio duomenų įterpimus. Todėl prarastų duomenų rodiklių analizė taip pat gali padėti pagerinti *LoRaWAN* ir TTN saugumą. Xueying Yang atliktoje analizėje pastebėta, kad galima įterpti žinutę į laukiamos žinutės vietą [41]. Ši įžvalga patvirtina, kad reikalinga nustatyti kiek duomenų yra prarandama, taip išvengiant netikėtos injekcijos.
Tyrimų rezultatai taip pat gali padėti tobulinti *LoRaWAN* protokolo dizainą ir prisidėti prie naujų standartų kūrimo. Ateityje tai gali padėti sukurti efektyvesnius, patikimesnius ir saugesnius daiktų interneto tinklus. Galima įžvelgti papildomai, kad įrenginio ir taikomųjų programų suderinamumas su „The things network“ turi įtakos *LoRa* veikimui. Skirtingi įrenginiai ir programos gali turėti skirtingą duomenų perdavimo gebėjimą ir toleranciją. Gedimų dažnio supratimas gali padėti tinklo inžinieriams ir kūrėjams sukurti ir pasirinkti tinkamus įrenginius ir programas, skirtas naudoti su *LoRaWAN* ir TTN. Todėl šios srities tyrimai yra ne tik problemų nustatymas pačiame tinkle, bet ir tobulinimo palengvinimas ir augimo rėmimas, taip pat ir ilgalaikės tokių tinklų, kaip TTN ir platesnės IoT ekosistemos, sėkmės užtikrinimas.

5.2.2 Duomenų siuntimo konfigūracijos

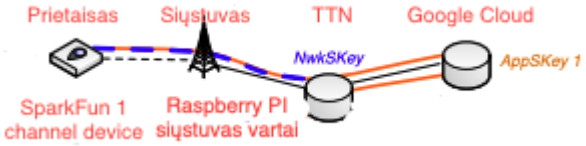
Tyrimą sudarys 3 *LoRa* duomenų siuntimo konfigūracijos:

1 Konfigūracijoje bus siunčiami duomenys naudojant „SparkFun LoRa gateway“ siųstuvas. Į jį iš prietaiso bus siunčiami 3 baitai duomenų kas 1 min., 2 min. ir 3 min. Taip pat siuntimas bus testuojamas skirtingais padengimo spektrais SF7-12. Šioje konfigūracijoje, duomenys bus siunčiami kai “Santakos” vartai yra atsijungę nuo TTN.



pav. 36 1 aparatinės įrangos konfigūracija

2 Konfigūracijoje bus siunčiami duomenys naudojant 8 kanalų Rak2245 *Raspberry* PI 4 siųstuvą. Konfigūracija galima matyti žemiau (žr. 37 pav.). Kaip ir 1 konfigūracijoje siunčiama bus laiku kai TTN “santaka” bus atjungta.



pav. 37 2 LoRaWAN *konfigūracija*

3 Konfigūracijoje bus naudojamas „The Things Network“ brokeris, kuris leis pasinaudoti tinkle esančiais prietaisais. Jo konfigūraciją galima matyti žemiau (žr. 38 Pav.) .

pav. 38 3 LoRaWAN *konfigūracija*

5.2.3 *LoRa* siuntimo duomenys

Vykdant *LoRa* duomenų siuntimus yra renkami duomenys apie sėkmingus siuntimus ir siuntimus, kurie nepasiekė tikslo. Nepasiekę tikslo bei pavykę siuntimai yra dokumentuojami ir supildomi lentelėse. Tikslo nepasiekęs siuntimas yra laikomas tuo siuntimu kuris buvo siunčiamas į atvirą „The Things Network“, bet šis siuntimas nebuvo užfiksuotas tinklo ir nėra jokių duomenų apie siunčiamą signalą. Užtikrinti, kad signalas iš tikro yra siunčiamas mikrovaldiklio, naudojamas „Sparkfun 1 channel gateway“ prietaise integruota lemputė ir 115200 sparta „BaudRate“ atvaizduojami komandų statusai (Pavykęs, nepavykęs siuntimas ir priežastis).

5.2.4 Tyrimo hipotezė

1 konfigūracijos siųstuvas/vartai, nuo 2022 metų tapo nebepalaikomo “The Things Network". Todėl pagal tai galima teigti, kad siuntimai naudojant šią konfigūraciją bus daugiausiai nesėkmingi. Remiantis šia prielaida suformuluojama pirma hipotezė.
H1: 1 kanalo siųstuvai nebegali siųsti duomenų į TTN tinklą.

Antra hipotezė, yra žinant kad kiti prietaisai irgi gali naudotis brokeriu, sėkmingiausiai duomenys bus siunčiami naudojant *RaspBerry* Pi paremtu siųstuvu, kuris atitinka visus standartus. Pagal tai sukuriama 2 hipotezė:
H2: Patikimiausiai duomenys siųs *RaspBerry* PI konfigūruotas siųstuvas.

5.2.5 Hipotezės patikrinimas

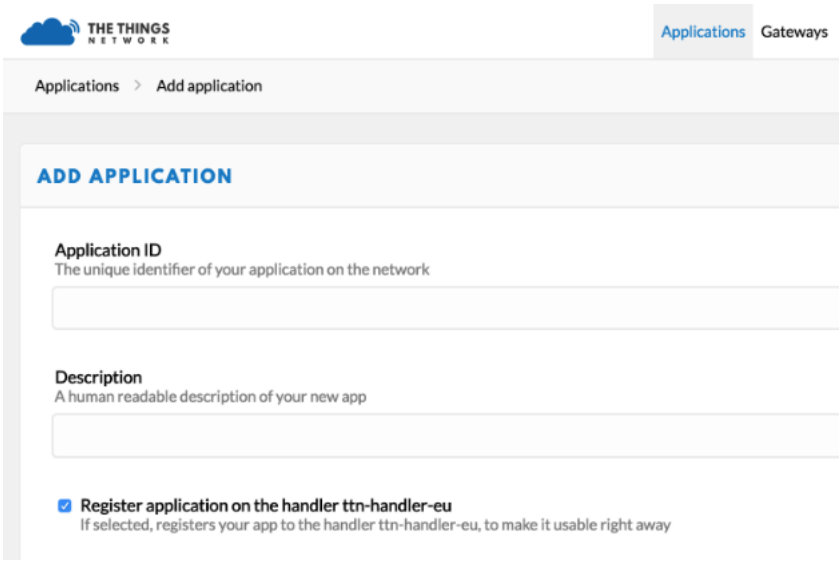
Hipotezės patikrinimui naudojamos gautų lentelių rezultatai. Pagal išbandytus siuntimo variantus ir pagal prarastų duomenų procentą, siuntimas kuris mažiausiai turės prarastų duomenų, patvirtins ar paneigs iškeltą hipotezę.

5.2.6 Tyrimo eiga

5.2.6.1 Tyrimo parametrai

Tyrimo pagrindą sudarė, 3 siųstuvų konfigūracijos, „Sparkfun 1 channel gateway“, ir TTN atviras tinklas. Visos trys konfigūracijos buvo ištestuotos išsiunčiant trumpus signalus į TTN. Naudojamas duomenų prietaisas buvo patikrintas išanalizavus dokumentaciją apie suderinamumą su kitais prietaisas ir pritaikius tą pati sklidimo faktorių su siųstuvais.

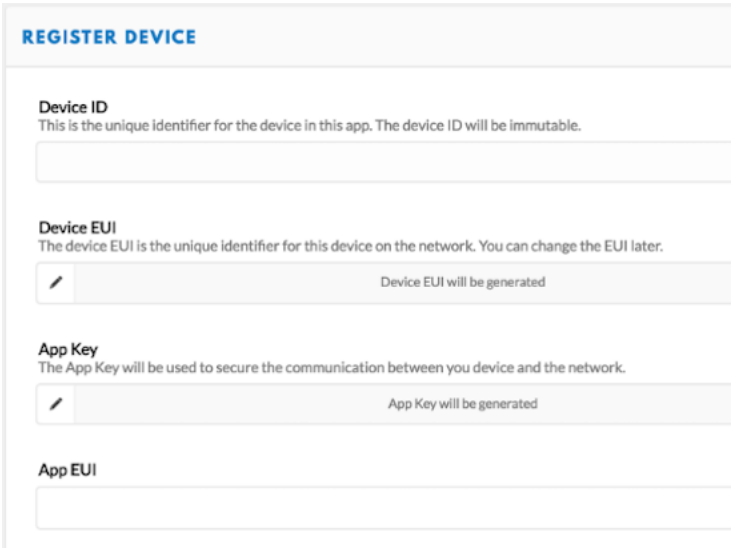
1 ir 2 konfigūracijos siųstuvai prie sistemos yra prijungiami naudojantis TTN egzistuojančia prijungimo architektūra. Pirmiausiai yra sukuriama programėlė TTN konsolėje kaip parodyta žemiau (žr. 39 Pav.).



pav. 39 Aplikacijos konsolė

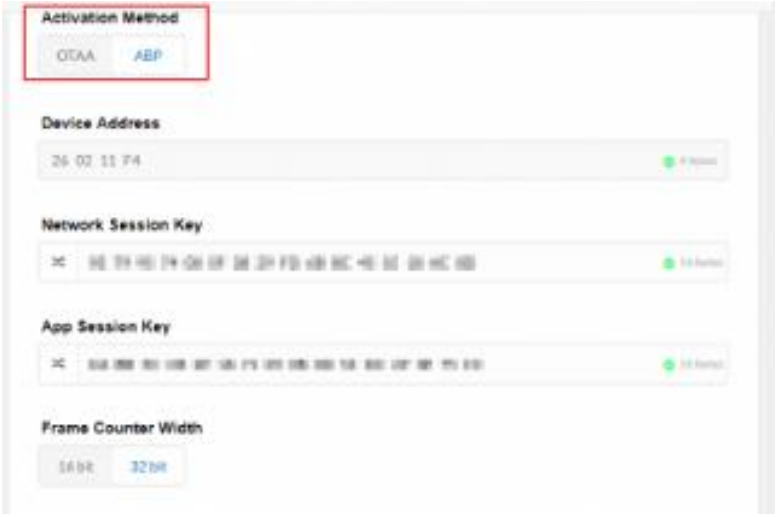
Pirmajame lange įrašoma programėlės pavadinimas. Šiuo atveju ji buvo pavadinta „LoRa failure rate analysis“. Antras langas, buvo paliktas tuščias.

Antrajame atsivėrusiame puslapyje, pridedamas naudojamas prietaisas siunčiantis duomenis į siųstuvus vartus. Lentelę galima pamatyti žemiau (žr. 40 Pav.).



pav. 40 Prietaiso registracijos langas

Pirmajame lange įrašomas prietaiso pavadinimas. Pavadinimas gali būti bet koks, tai prietaisui duodamas pavadinimas – smart-kettleESP32. Antrame lange reikia įrašyti prietaiso EUI, bet dėl to, kad šis prietaisas nėra registruotas TTN tinkle, registruojama prietaisą rankiniu būdu (žr. 41 Pav.).



pav. 41 Prietaiso registracijos langas ABP metodu

Šiame lange papildoma informacija, kurį naudos prietaisas. Adresas, „Network Session key“ yra automatiškai sugeneruojami ir tada įvedami į prietaiso programos kodą (žr. 42 Pav.).

```
static const PROGMEM u1_t NWKSKEY[16] = { 0x01, 0x23, 0x45, 0x67, 0x89, 0xAC, 0xBD, 0xEF, 0x01, 0x24, 0x45, 0x68, 0x89, 0xAB, 0xCD, 0xEF };
static const u1_t PROGMEM APPSKEY[16] = { 0xFE, 0xDC, 0xBA, 0x98, 0x76, 0x54, 0x32, 0x10, 0xEE, 0xDC, 0xAA, 0x98, 0x76, 0x54, 0x32, 0x10 };
static const u4_t DEVADDR = 0x01234567;
```

pav. 42 Prietaiso adresai naudojami prisijungimui prie TTN tinklo

Atlikus šį žingsnį, pradedamas siųstuvo vartų ruošimas. Šiame žingsnyje yra sukuriamas naujas siųstuvas/vartai, TTN platformoje. Ir sukūrus šiuos vartus, 2 vartų konfigūracijos galės prisijungti prie tinklo naudojant „smart-kettleESP32-eu1-cloud.eu1.cloud.thethings.industries“ adresą. Šiuo žingsniu pabaigiamas fizinių prietaisų ruošimas.

Norint pasinaudoti KTU „Santakos Slėnio“ vartais, reikia prisijungti prie TTN brokerio. Tai galima atlikti per TTN nustatymus prisiregistravus prie brokerio. Tai atlikus, galima siųsti duomenis ir nuotoliniu būdu.

5.2.6.2 Tyrime naudota programinė įranga

Tyrimo eigoje buvo naudojama TTN sistema skirta prijungti *LoRa* įrenginius prie tinklo ir programinė įranga, kuri buvo naudota kuriant magistrinį projektą (Siųstuvų vartų programinė įranga ir mikrovaldiklių programinė įranga).

Sistema siūlo nuosavą siųstuvų vartų naudojimą arba prisijungti prie TTN tinklo brokerio. Šiuo atveju bus naudojami abu variantai.

5.2.7 Tyrimo rezultatai

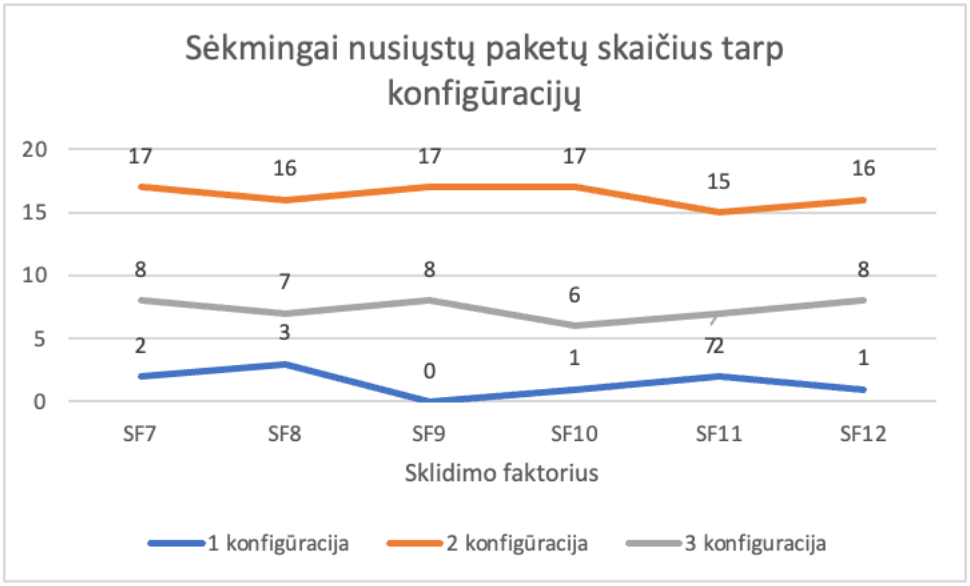
Pirmiausiai tyrimas buvo atliekamas siunčiant signalą kas 1 min. Tai yra virš TTN nustatytų ribų (reikalavimuose prašoma siųsti duomenis kas 2 min) nes tai sukelia didesnę apkrovą esantiems siųstuvams vartams ir gali sukelti trukdžius kitiems prietaisams siųstuvams. Bet norint ištestuoti limitus bus siunčiama laiku kai 5 km spinduliu siųstuvai/vartai yra atjungti ir bus siunčiama naudojant savus siųstuvus, ir tik 17 žinučių siunčiama naudojant “santakos” siųstuvą skirtingomis dienomis, skirtingais sklidimo faktoriais tuo tarpu šio limito laikinai buvo nesilaikoma. Tyrimų apibendrintus rezultatus su visais sklidimo faktoriais pirmai konfigūracijai galima matyti žemiau pavaizduotose diagramose (žr. 43-45 Pav.).

pav. 43 Sėkmingai nusiųstu paketų skaičius 1 konfigūracijos

pav. 44 Sėkmingai nusiųstų paketų skaičius 2 konfigūracijai

pav. 45 Sėkmingai nusiųstų paketų skaičius 3 konfigūracijai

Iš rezultatų galima matyti, kad stabiliausiai duomenis perdavė „Raspberry PI“ siųstuvas net ir viršijant TTN reikalavimus siuntimo dažniui (žr. 46 pav.):



pav. 46 Konfigūracijų rezultatų palyginimas

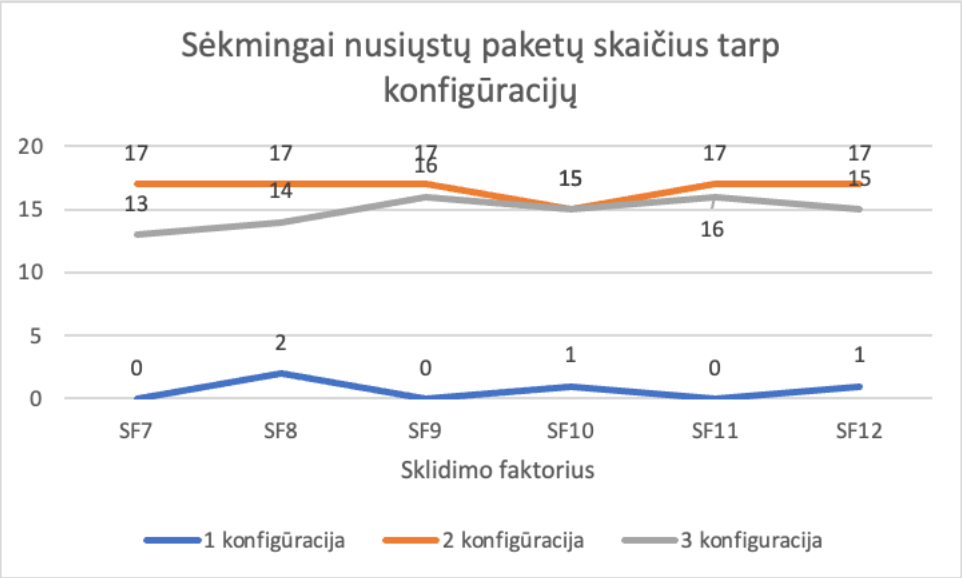
Antroje dalyje yra siunčiami tiek pat paketų ir per tokį patį sklidimo faktorių skaičių, tik šį kartą paketai yra siunčiami kas dvi minutes. Lenteles atspindinčias šiuos sklidimo faktorius, galima matyti žemiau (žr. 47-49 Pav.). Daugiau diagramų galima rasta 4 priede.

pav. 47 Sėkmingai nusiųstų paketų skaičius 1 konfigūracijai

pav. 48 Sėkmingai nusiųstų paketų skaičius 2 konfigūracijai

pav. 49 Sėkmingai nusiųstų paketų skaičius 3 konfigūracijai

Pagal gautus duomenis buvo sudaryta palyginamoji diagrama šioms trims konfigūracijom (žr. pav. 50). Daugiau palyginamųjų diagramų galima rasta 4 priede.



pav. 50 Konfigūracijų rezultatų palyginimas

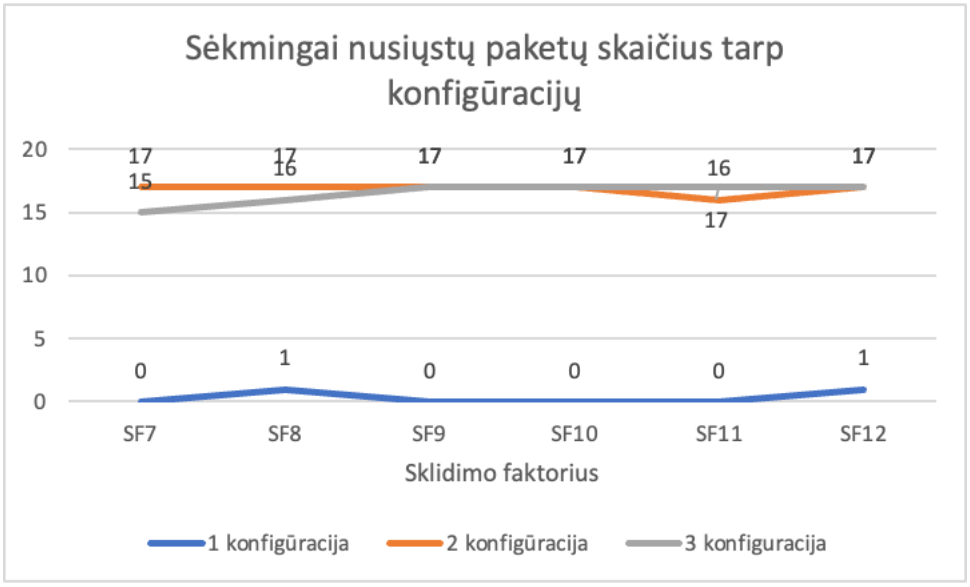
Trečioje tyrimo dalyje yra siunčiami tiek pat paketų ir per tokį patį sklidimo faktorių skaičių, tik šį kartą paketai yra siunčiami kas 5 minutes norint patikrinti, koks rezultatas siunčiant per didelį laiko tarpą. Lenteles atspindinčias šiuos sklidimo faktorius, galima matyti žemiau pavaizduotose diagramose (žr. 51-53 Pav.).

pav. 51 Sėkmingai nusiųstų paketų skaičius 1 konfigūracijai

pav. 52 Sėkmingai nusiųstų paketų skaičius 2 konfigūracijai

pav. 53 Sėkmingai nusiųstų paketų skaičius 3 konfigūracijai

Pagal tai buvo sudaryta palyginamoji diagrama, šioms trims konfigūracijoms (žr. pav. 54). Daugiau palyginamųjų diagramų galima rasta 4 priede.



pav. 54 Konfigūracijų rezultatų palyginimas

Šio tyrimo rezultatai parodė, kad 1 kanalo siųstuvas/vartai yra visiškai nepalaikomas TTN ir siunčiami duomenys iš šio siųstuvo, iš 17 pranešimų, tik keletas pasiekia tikslą. Itin svarbu paminėti, kad tyrimo pabaigoje atrasta, kad šie keli anomaliniai pranešimai, kurie sugebėjo pasiekti TTN, vis dėl to buvo siunčiami ne iš turimo 1 konfigūracijos siųstuvo, bet tikėtina iš aplinkoje esančio siųstuvo. Tai buvo patvirtinta, patikrinus TTN siųstuvo konsolę ir iš terminalo išgeneravus siųstų duomenų detalę informaciją. Svarbu pabrėžti, kad visos kitos konfigūracijos siuntė tik iš joms priskirto siųstuvo.

O tuo tarpu naudojant “packet forwarder” ir “Santaka” siųstuvą pastebėta, kad jis veikia jeigu laikomasi dviejų minučių siuntimo, o siunčiant kas 1 minutę sėkmingų siuntimų sumažėja per pusę. Pats stabiliausias siuntimas buvo pasiektas naudojant *RaspBerry* siųstuvą. Šis siųstuvus visomis sąlygomis puikiai pasirodė ir sugebėjo išsiųsti absoliučią daugumą pranešimų. Dėl šios priežasties, šis siųstuvas/vartai buvo naudojamas antro ir trečio tyrimo *LoRa* panaudojimo atvejais.

5.2.8 Tyrimo apribojimai ir grėsmės

Atlikus šį tyrimą, buvo pastebėta, kad vis dėl to, TTN pilnai nepalaiko 1 kanalo siųstuvų. Tikrinant TTN gautus duomenis šiam siųstuvui, pastebėta, kad jokių duomenų TTN siųstuvo informaciniame lange nebuvo rasta ir taip pat nebuvo rasta siuntimų bandymų turinčių „Sparkfun 1 channel gateway“ siųstuvo EUI (prietaiso adreso). Visi duomenys siųsti iš „Sparkfun 1 channel gateway“ kurie sėkmingai pasiekė TTN, buvo rasti TTN programos lange, bet ne siųstuvo lange. Tai reiškia, kad nors tyrimas atliktas, kai visi aplinkiniai 5 km spinduliu esantys siųstuvai nurodyti kaip neaktyvus, rasta kad vienas iš siųstuvų kai kuriais laiko tarpais aktyvavosi ir iš didelio nuotolio persiuntė prietaiso pranešimus. Pastebėta, kad kai ir būdavo siunčiami duomenys, užfiksuotas ir duomenų atvykimo vėlavimas, kas irgi patvirtino, kad siuntimus atliko kitas siųstuvas nei tyrime. Tai yra todėl nes visi siųstuvai yra mažiau nei 100 m. atstumo nuo prietaiso. Galiausiai patvirtinti, kad tyrime atlikti pranešimai nebuvo siųsti 1 konfigūracijos siųstuvo, buvo patikrinta siųstuvo duomenų istorija ir aplinkinių siųstuvų istorijos. Po šios patikros rasta, kad „lorix-iotlab“ persiuntė šiuos duomenis. Dėl šių priežasčių, galima pilnai patvirtinti teiginį, kad į TTN nebegalima persiųsti duomenų naudojant 1 kanalo siųstuvą. Tai yra įmanoma apeiti kuriant privatų tinklą *LoRa* protokolui arba sukuriant maskavimo programinę įrangą. Bet šis tyrimas nesiekia apeiti šio ribojimo. Išties 1 kanalo siųstuvų ribojimas yra teigiamas pokytis. Šie siųstuvai yra neefektyvūs, užkrauna tinklą, ir tik gali siųsti vieną pranešimą, palyginus su kitais *LoRa* palaikančius daugiakanaliais prietaisais
Antra įrangos konfigūracija su “Raspberry Pi” veikė be didesnių bėdų ir veikimas išliko pastovus. Prarasti duomenys nors ir numatytose ribose, gali reikšti, kad yra konfigūracijos trūkumų arba aplinkoje egzistavo trukdžiai. Svarbu paminėti, kad visi pranešimai siųsti matavimo prietaiso, skirtingai nei 1 konfigūracijos buvo siunčiami tik iš šio siųstuvo.
Trečia konfigūracija patvirtino, kad TTN limitai nėra rekomendacijos, bet reikalavimas ir naudojant kitų tinklo naudotojų siųstuvus/vartus siųsti pranešimus įmanoma tik kas 2 minutes. Dar vienas atrastas minusas yra šios konfigūracijos dažnas atsijungimas nuo tinklo. Palyginus su nuosavo *Raspberry* siųstuvu/vartais, „Santaka“ skirtingomis dienomis atsijunginėjo nuo TTN. Dėl šios priežasties nepatartina pasitikėti kitais tinkle esančiais prietaisais, jei būtina užtikrinti duomenų perdavimą. Visi pranešimai taip pat kaip ir 2 konfigūracijos, buvo persiunčiami tik šio siųstuvo.

H1 buvo patvirtinta, nes tyrime užfiksuoti keli pavykę siuntimai iš tikro buvo siunčiami „lorix-iotlab“siųstuvo, o ne 1 kanalo.
H2 buvo patvirtintas, nes *Raspberry* išsiuntė daug daugiau pranešimų nei 1 konfigūracija ir niekada neatsijunginėjo nuo tinklo palyginus su 3 konfigūracija.

5.2.9 Tyrimo išvados

Buvo atliktas tyrimas, su tikslu patikrint, ar siųstuvas/vartai, kurio prieiga dalijamasi yra toks pats patikimas siųsti žinutes kaip nuosavas siųstuvas/vartai. Taip pat šių atstumas iki prietaiso buvo mažesnis nei 100 metrų. Ir tyrimas parodė, kad daugiausiai pranešimų išsiuntė nuosavas siųstuvas/vartai. Tai ypatingai išryškėjo, siunčiant žinutes kas minutę. Tokiu nustatymu, nuosavas siųstuvas persiuntė dvigubai daugiau pranešimų. Ateityje šis tyrimas gali padėti kompanijoms pasirinkti tinkamiausią sprendimą *LoRa* duomenų persiuntimui.

Taip pat atliktas tyrimas patikrinti ar yra galimybė vis dar naudotis vieno kanalo siųstuvo vartais. Tyrimas nors ir nevisapusiškai, bet patvirtino TTN dokumentaciją kad tai nebėra sprendimas, kurį verta naudoti IOT prietaisams. Ateityje tai gali padėti, kitiems nepadaryti klaidos renkantis produktą persiųsti duomenims.

5.3 Antrasis tyrimas

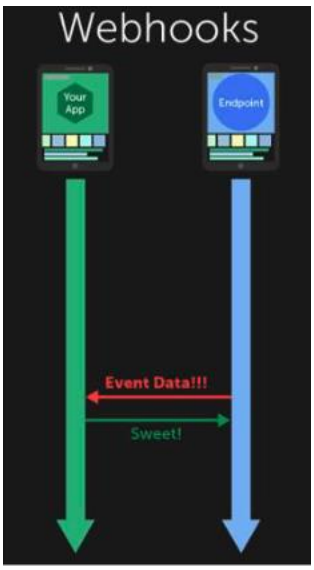
5.3.1 Įžanga į tyrimą

Pirmajame tyrime buvo patikrintos siuntimo duomenų praradimas su skirtingomis aparatinės įrangos konfigūracijomis. Minėtas tyrimas leis užtikrinti ateities naudotojams, kad bus pasirinkta tinkamiausia aparatinė įranga duomenų perdavimui *LoRa* protokolu. Antrasis tyrimas fokusuosis ties duomenų iš TTN tinklo perdavimo į *Google Cloud* sistemą. Buvo pastebėta, kad nors ir yra nemažai dokumentacijos lyginančius *Webhook* ir MQTT metodus tarpusavyje ir lyginančius kuris yra efektyvesnis metodas duomenų perdavimui, itin trūksta dokumentacijos kaip kalbama apie duomenų siuntimą šiais metodais tarp *Google Cloud* ir TTN. Dėl šios priežasties bus matuojama, kuris metodas yra efektyvesnis norint sumažinti siunčiamų duomenų dydį ir siunčiant didelius duomenų kiekius tarp *Google Cloud*. Pagal Maciej buvo pastebėta, kad siunčiant didelius kiekius duomenų, gali drastiškai pakilti naudojamų platformų kainos kaip Google Cloud Functions, *AWS lambda* ar *Azure functions* [41]
Dėl šios priežasties itin svarbu naudoti kuo efektyvesnį metodą duomenims perduoti, ypatingai jei yra didelis skaičius *LoRa* prietaisų, kurie pastoviai siunčia duomenis.

5.3.2 Duomenų siuntimo konfigūracijos

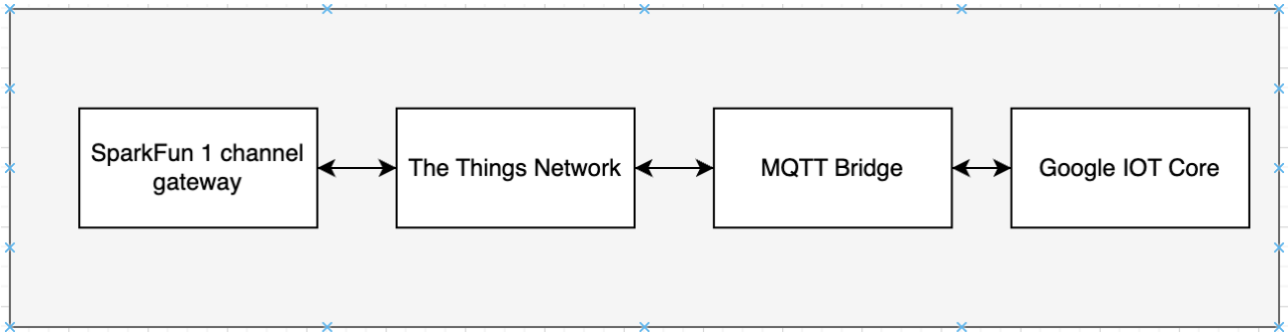
Šiame tyrime duomenys bus siunčiami dviem metodais – žinučių eilės sudarymo telemetrijos transporto protokolo (MQTT) ir „WebHook“ metodais.

„WebHook“ metodas buvo paruoštas sukuriant metodą TTN platformoje (angl. *Webhook integrations*) ir funkciją Google Cloud platformoje. Metodo veikimo principą galima matyti žemiau (žr. 55 Pav.).



pav. 55 "Webhook" metodo veikimo principas [42]

MQTT metodas buvo paruoštas tyrimui, sukuriant TTN ir MQTT tiltą, ir prie jo prijungiant TTN ir *Google Cloud IOT core* (žr. 56 Pav.).



pav. 56 MQTT sistemos struktūra

Prietaisas siųs *LoRa* duomenis į TTN kas 2 minutes ir duomenis sudarys išmatuota virdulio temperatūrą ir statusas.

5.3.3 Duomenų rinkimo metodas

Tyrimo duomenys bus renkami Google Cloud konsolėje „Subscriptions" lange. Šiame lange bus renkami duomenys siunčiami iš TTN MQTT metodu. Gale tyrimo visi šie duomenys bus atvaizduoti lentelėse ir bus atvaizduojamas žinučių dydis, prijungimo dydis baitais. Tuo tarpu „Webhook" metodo rezultatai bus renkami „Google Functions" konsolės lange. Kaip ir pirmajame laboratoriniame, užtikrinti, kad signalas iš tikro yra siunčiamas mikrovaldiklio, naudojama „Sparkfun 1 channel gateway" prietaise integruota lemputė ir konsolėje 115200 sparta „BaudRate" atvaizduojami komandų statusai (Pavykęs, nepavykęs siuntimas ir priežastis). Taip pat, užtikrinti kad žinutės buvo gautos TTN tinklo, bus tikrinama TTN konsolė ar žinutė sėkmingai pasiekia TTN ir ar TTN gali sėkmingai išsiųsti žinutes į „Google Cloud" „WebHook ar „MQTT" metodais.

5.3.4 Tyrimo hipotezė

Palyginus su *Webhook* metodu, MQTT metodui nereikės pastoviai prisijungti ir atsijungti nuo TTN ir „Google Cloud“ tinklo, nes MQTT tilto nustatymuose specialiai bus nustatomas nesibaigiantis gyvavimo laikas. Tai yra darom norint sukurti idealiausias sąlygas MQTT ir „Webhook“ metodo išnaudojimui. Dėl šios priežasties galima formuoti šią hipotezę:
H1: MQTT metodas bus efektyvesnis nei „Webhook“ siunčiant daug pranešimų dėl nesibaigiančio ryšio.

5.3.5 Hipotezės patikrinimas

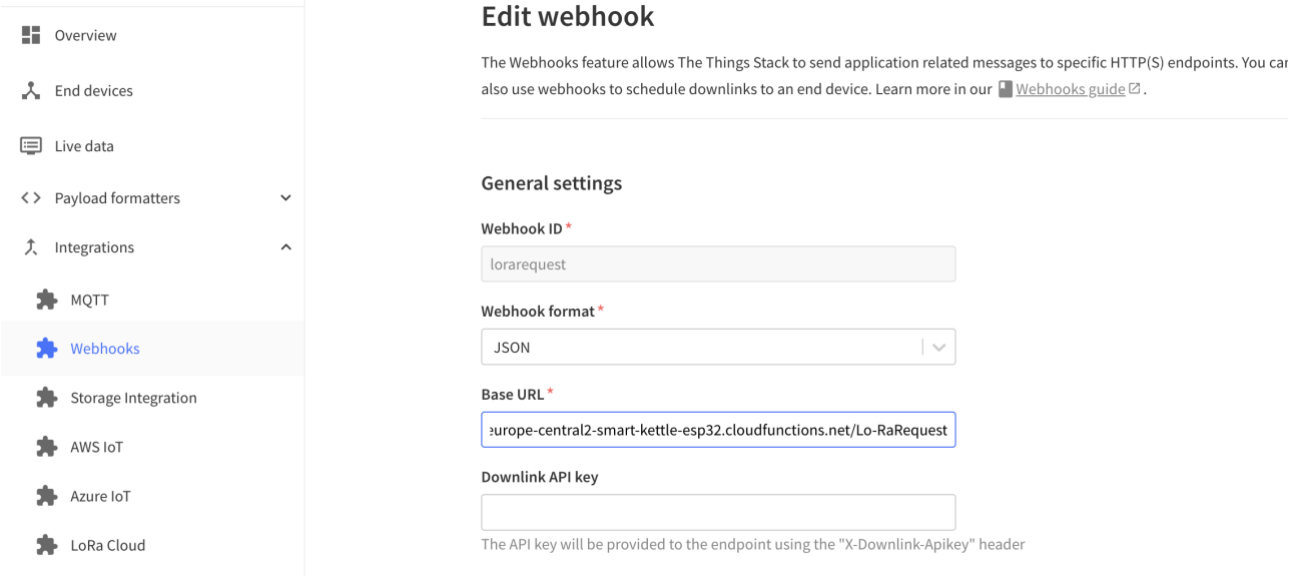
Hipotezės patikrinimui bus naudojami gauti duomenys išmatuoti „Google“ konsolėje ir po atitinkamo skaičiaus siuntimų, lentelėse bus palyginami MQTT metodo ir „WebHook“ metodo sėkmingos komunikacijos ir pranešimo dydžiai.

5.3.6 Tyrimo eiga

5.3.6.1 Tyrimo parametrai

Tyrimo pagrindą sudarė *WebHook* ir MQTT metodais duomenų apsikeitimai tarp TTN ir „Google Cloud“ sistemų. Konfigūracijos buvo ištestuotos paduodant pradinius duomenis iš *Postman* ir MQTT testavimo serverio. Duomenys iš *LoRa* prietaiso buvo siunčiami naudojant *Raspberry* siųstuvą – vartus.

„WebHook“ metodas yra paruošiamas siuntimui sukuriant „WebHook“ TTN platformoje kaip parodyta paveikslėlyje (žr. 57 Pav.).



pav. 57 WebHook *metodo paruošimas TTN platformoje*

Šis metodas yra pakurtas toje pačioje mobiliojoje programėlėje kaip ir pirmame tyrime. Atlikus šį veiksmą sukuriama *Webhook* programa priimti duomenis iš TTN, „Google Cloud“ platformoje. Kodo dalis kuri dekoduoja ateinantį „Webhook“ „JSON“ failą iš TTN (žr. 58 Pav.)

```php
$end_device_ids = $json['end_device_ids'];
    $device_id = $end_device_ids['device_id'];
    $application_id = $end_device_ids['application_ids']['application_id'];

$received_at = $json['received_at'];

$uplink_message = $json['uplink_message'];
    $f_port = $uplink_message['f_port'];
    $f_cnt = isset($uplink_message['f_cnt']) ? $uplink_message['f_cnt'] : 0;    // Zero & empty values are not included
    $frm_payload = $uplink_message['frm_payload'];
    $rssi = $uplink_message['rx_metadata'][0]['rssi'];
    $snr = $uplink_message['rx_metadata'][0]['snr'];
    $data_rate_index = isset($uplink_message['settings']['data_rate_index']) ? $uplink_message['settings']['data_rate_index'] : 0;
    $consumed_airtime = $uplink_message['consumed_airtime'];
```

pav. 58 Webhook *metodo dekodavimas PHP kodo iškarpa*

“WebHook” metodas dabar yra paruoštas veikimui ir galima testuoti.

Tolesnis tyrimo ruošimas liečia MQTT metodo rengimą. Šiai daliai, reikės sukurti MQTT klientus iš TTN ir Google brokerių. Tai bus galima padaryti naudojant MQTT tiltą. Pirmiausiai yra atsisiunčiamas „Mosquito“ klientas brokeris (puslapis randamas *mosquitto.rsmb/rsmb/src*) [44]

Tada atsisiunčiami konfigūraciniai failai iš „Google Cloud“ platformos prijungti prie MQTT „IOT core“ kliento [46]. Tai atlikus, sukuriamas naujas Pitono failas kuris sujungs TTN ir „Google Cloud“ klientus (žr. 59 Pav.).

```
private_key_file = './rsa_private.pem'
algorithm = 'RS256'
ca_certs = './roots.pem'
mqtt_bridge_hostname = 'mqtt.googleapis.com'
mqtt_bridge_port = 8883
message_type = 'event'
```

```
sub_topic = 'events' if message_type == 'event' else 'state'

mqtt_topic = '/devices/{}/{}'.format(device_id, sub_topic)

Gclient = google.get_client(project_id, cloud_region, registry_id,
                            device_id, private_key_file, algorithm,
                            ca_certs, mqtt_bridge_hostname, mqtt_bridge_port)
```

pav. 59 MQTT prisijungimas prie “Google IOT core”

Tada reikia sukurti funkciją, kuri konvertuoja ateinančius duomenis iš TTN naudojamo “base64” formato į “Google Cloud” naudojamą ASCII formatą. Tada siunčiami duomenys į „Google IOT core“ (žr. 60 Pav.).

```
21  def uplink_callback(self,msg, client);
22      # konvertuojam ateinancius duomenis is base64 i ASCII
23      duomenu_dekodavimas = base64.b64dekode(msg.payload_raw)
24      asciipayload = duomenu_dekodavimas.decode ("ascii")
25
26      #sutvarkom laiko timestamp, nes duomenys is prietaiso ateina netiksliu laiku
27      jsonpayload = json.loads(asciipayload)
28      sekunde = time.time()
29      jsonpayload["date"] = int(seconds)
30      payload = json.dumps(jsonpayload)
31      print ("gavom duomenis", jsonpayload)
32      #siunciam duomenis i IOT core, note to self - refactorinti Gclient loops (padaryta)
33      Gclient.publish(mqtt_topic, payload, qos=0)
34      Gclient.loop()
35      return True
```

pav. 60 Kodas konvertuojantis "base64" formatą š "Google" priimtiną ASCII formatą

Tada paleidžiamas failo scenarijus sukurti TTN ir „Google IOT Core" tiltą (žr. 61 Pav.).

```
aldasjonusauskis@Macbook-pro-3: python3 TTNbridge.py
TTN, IOT core bridge connection....
Connected
Device client_id is 'projects/TTNproject'
Subscribing to MQTT integration
aldasjonusauskis@Macbook-pro-3:
```

pav. 61 Terminalo langas paleidus tilto scenarijaus failą

Atlikus šiuos žingsnius, paleidžiama „Google Cloud" konsolė ir jos pagalba sukuriamas MQTT scenarijus. Po šio žingsnio galima pradėti tyrimą.

5.3.6.2 Tyrime naudota programinė įranga

Tyrimo metu buvo naudojamos TTN ir "Google Cloud" sistemos. Taip pat "Mosquito" MQTT programinė įranga, sukurti ryšį tarp sistemų.

5.3.7 Tyrimo rezultatai

Tyrimo metu buvo siunčiami "Webhook" metodu paketai iš TTN į "Google Functions". Šio metodo standartinė siuntimo formą, galima matyti 1 priede. Buvo siunčiami pastovūs paketai ir naudojant "Google Functions logs" matuojamas metodo paketo dydis. Išsiuntus 10 paketų, rezultatai pavaizduoti 20 lentelėje:

lentelė 12 "Webhook" metodo siuntimo rezultatai

Siuntimo bandymas	Paketo išmatuotas dydis bitais

1	1260
2	1266
3	1263
4	1260
5	1260
6	1260
7	1263
8	1266
9	1263
10	1260

TTN MQTT žinučių formatą galima matyti 2 ir 3 priede. Atlikus prijungimą tarp TTN ir “Google Cloud”, sistema buvo sėkmingai paleista ir kaip ir “Webhook” metodu, buvo siunčiama standartinė minimali TTN palaikoma žinutė nurodyta 2 ir 3 priede į “Google Cloud”. Kaip ir “Webhook” metodu, buvo siunčiama 10 žinučių ir buvo tikrinamas efektyvumas (žr. lentelė 13)

lentelė 13 “MQTT” metodo siuntimo rezultatai:

Siuntimo bandymas	Paketo išmatuotas dydis bitais
1	1441
2	988
3	988
4	985
5	982
6	985
7	988
8	982
9	988

10	985

Atlikus siuntimo testavimus, buvo pastebėta, kad siuntimas iš pradžių buvo efektyvesnis naudojant “Webhook” metodą, bet dėl to, kad MQTT metodui reikia tik pirminio prijungimo prašymo, o naujos žinutės naudoja tą patį prijungtą ryšį, duomenų dydis vienai žinutei sumažėjo nuo 1441 Baitų iki 988 – 982. Diagramą rodančią šį pokytį galima matyti čia:

5.3.8 Tyrimo apribojimai ir grėsmės

Atlikus tyrimą pasitvirtino iškelta hipotezė ir MQTT metodas iš tikro yra efektyvesnis metodas siųsti daug pranešimų vienu metu. Palyginus su “Webhook” metodu “MQTT” nereikia kiekvienam pranešimui siųsti prašymo prisijungti ir kai tik užmezgamas ryšys, galima siųsti neribotą kiekį mažesnių žinučių vienu metu. Dėl šios priežasties galima teigti, kad ryšys tarp TTN ir “Google Cloud” niekuo nesiskiria nuo kitų sistemų, ir MQTT metodas laikui bėgant tampa efektyvesniu.

Reikia paminėti, kad buvo tirta siunčiant žinutes tik iš TTN į „Google Cloud“. Todėl nors žinučių dydis neturėtų pastebimai keistis siunčiant duomenis iš „Google Cloud“ į TTN, šis tyrimas fokusavosi tik siuntimu iš TTN į „Google Cloud“.

H1 buvo dalinai patvirtina, nes duomenys buvo siunčiami tik iš TTN į „Google Cloud“

5.3.9 Tyrimo išvados

Buvo atliktas tyrimas, kurios pagrindinė užduotis buvo išsiaiškinti ar MQTT metodas yra efektyvesnis nei “Webhook” metodo ir ši prielaida pasitvirtino. Nors eksperimente buvo siunčiama tik 10 žinučių kiekvienam metodui, dėl to, kad šių metodų siunčiamų duomenų turinys ar dydis nesikeis, nebuvo pagrindo siųsti daugiau žinučių. Tiesa, buvo pastebėta, kad naudojamas tilto metodas sujungti MQTT brokerius tarp TTN ir “Google” nebuvo itin efektyvus ir persiunčiant duomenis, iš TTN atsiųsti 1330 baitų duomenų paaugo iki 1441 baitų. Tai gali būti formatų konvertavimo bėda iš “base64” į ASCII, arba tiesiog konvertavimo metu, atsitiktinai įterpta papildomų duomenų. Bet net ir tokiu atveju, tyrimo rezultatai vis tiek nesikeistų, ir MQTT pranešimas vis tiek keliais šimtais baitų būtų didesnis nei “Webhook” pradinio ryšio sukūrimo metu, o tada kaip ir tyrimo metu, sumažėtų 300 baitų. “Webhook” metodo rezultatai yra teisingi ir pranešimų dydis būtų gerokai didesnis nei MQTT metodo po ryšio sukūrimo. Ateityje šis tyrimas gali padėti organizacijoms ieškančioms efektyvaus ir kapitalo išsaugančio metodo perduoti daugybę duomenų.

EKSPERIMENTINĖ DALIS

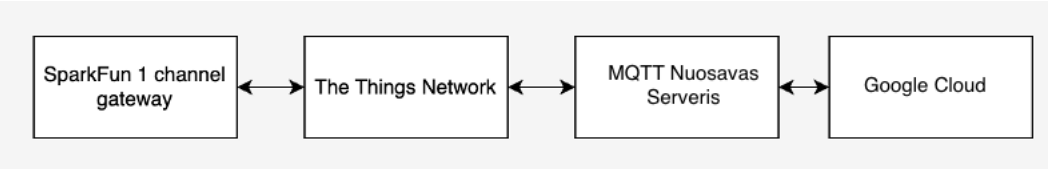
3 Eksperimentinis tyrimas

6.1 Įžanga į tyrimą

Antrasis tyrimas duomenų siuntimas MQTT metodu rėmėsi „Google Cloud IOT core“ brokeriu. Šis brokeris buvo naudojamas norint perduoti duomenis iš TTN į *Google Cloud*. Bet 2022 metų rugpjūtį, „Google“ paskelbė, kad nuo 2022 metų rugpjūčio 16 dienos, nebepalaikys „IOT core“ ir visi prijungti prietaisai bus atjungti [44].
Ši naujiena iškelia dilemą, kaip reikėtų tęsti naudotis „Google Cloud“ paslaugomis prietaisams neturint prieigos prie MQTT metodo. Vienas iš galimų sprendimų yra naudotis „Webhook“ metodu, bet kaip antras tyrimas parodė, siunčiant daug žinučių neefektyviu metodu prarandama resursų, ypatingai jei tai daroma tuo pačiu metu ir taip pat padidėja kaštai [41]. Tuo tarpu siunčiant daug žinučių per trumpą laiko tarpą ir neatjungiant MQTT prenumeratos, kaip parodė antras tyrimas yra daug efektyvesnis metodas nei prijungimas ir atjungimas kiekvienos žinutės naudojant „Webhook“ metodą.
Kiti pasiūlyti variantai yra naudoti kitus rinkose esančius brokerius – „AWS IoT Core“, MS Azure IoT. Bet problema iškyla randant tinkamiausią alternatyvą „Google IoT Core“ [45]. Dėl šios priežasties neradus tinkamos alternatyvos, būtų puikus variantas sukurti savo brokerį, kuris atitiks organizacijos iškeltus reikalavimus. Taip pat pastebėta, kad dėl to, kad ši problema egzistuoja mažiau nei metus, nėra dokumentacijos ar mokslinių darbų ieškančių alternatyvų. Šis eksperimentinis tyrimas bandys tai išspręsti. Bus bandoma sukurti nuosavą brokerį, kuris prisijungs prie „Google Cloud“ naudojant pub/sub metodą ir leis vartotojui toliau siųsti duomenis tarp TTN ir „Google Cloud“.

6.1.1 Duomenų siuntimo konfigūracijos

Šiame tyrime, bus kuriamas tarpinis brokeris, kuris priims duomenis iš TTN MQTT ir siųs juos „Google Cloud“ prenumeratai. Veikimo modelis galima matyti žemiau atvaizduotame paveikslėlyje (žr. 58 pav).



pav. 62 Eksperimentinės sistemos veikimo modelis

Duomenys bus siunčiami iš TTN, kurie bus priimti nuosavame MQTT serveryje. Šie duomenys tada bus siunčiami į „Google Cloud“ prenumeratos metodu. Prietaisas siųsti LoRa protokolus išmatuotus duomenims bus naudojamas „SparkFun 1 channel gateway“. O šie duomenys kaip ir antrame tyrime bus persiunčiami naudojant „Raspberry“ siųstuvą/vartai.

6.1.2 Tyrimo hipotezė

Pagal tai, kad atliekamas eksperimentinis tyrimas, ieškantis būdo pakeisti pašalinamą *IOT core*, yra formuojama ši hipotezė:
H1: Sukurtas MQTT metodas galės priimti ir siusti duomenis tarp TTN ir „Google Cloud“

6.1.3 Hipotezės patikrinimas

Hipotezės patikrinimui bus testuojama sukurta sistema ir tikrinama ar siunčiami duomenys į sukurtą MQTT serverį yra sėkmingai persiunčiami į „Google Cloud“ ir atgal jei yra duomenų ateinančių iš „Google Cloud“.

6.1.4 Eksperimentinio Tyrimo eiga

6.1.4.1 Eksperimentinio Tyrimo parametrai

Tyrimo pagrindą sudarė antrajame tyrime atlikti pasiruošimo darbai. Šiuos darbus sudarė MQTT Pub/Sub metodo sukūrimas „Google Cloud“ ir to pačio metodo sukūrimas ir TTN sistemoje.

Nuosavo brokerio veiklai bus naudojamas *Mosquito* serveris, dėl plataus palaikymo ir lengvos konfigūracijos.

6.1.4.2 Eksperimentinio tyrimo eiga

Pirmiausiai reikalingas virtualus serveris, kuriame būtų laikomas MQTT brokeris. Šiuo atveju buvo pasirinktas „Hostinger“ kompanijos VPS serveris, su Ubuntu sistema joje. Kitas žingsnis, atidaryti komunikacijai platformų yra atidaromas 1883 vartų (angl. *gate*) priėjimas naudojant komandą: *sudo ufw allow 22/tcp* komandą.

Tada yra diegiamas „Mosquito“ brokeris naudojant šias *linux* sistemos komandas (žr. 63 pav.).

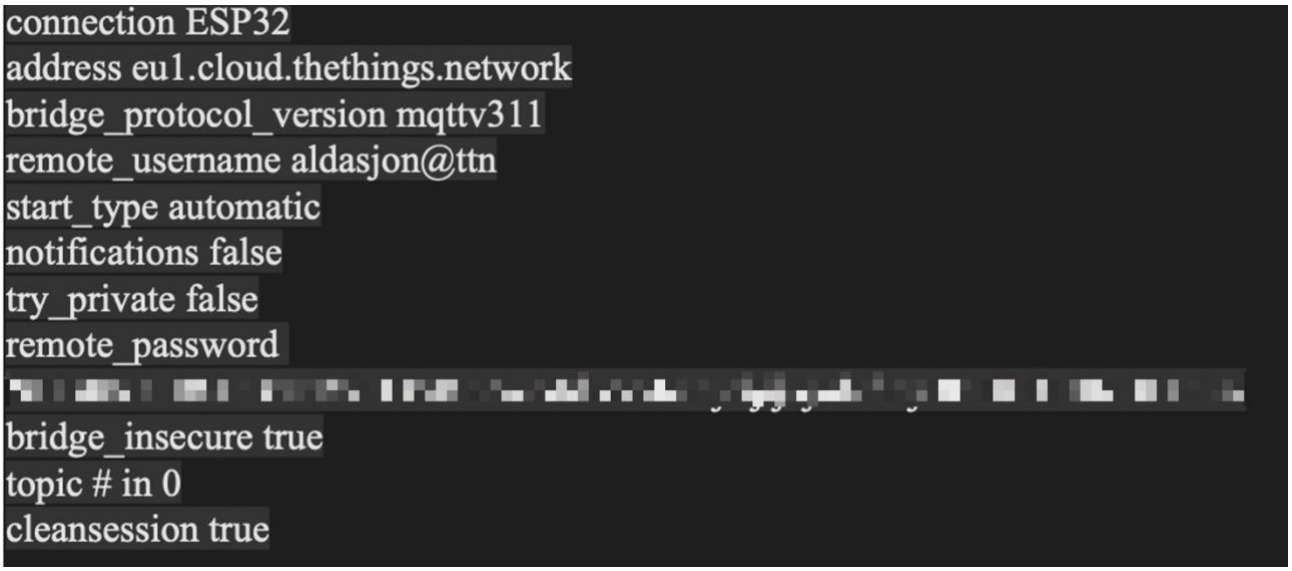
```
//Komanda įdiegti Mosquito repozitoriją:
sudo apt-add-repository ppt:mosquito-dev/mosquitto-ppa
// Komanda paleisti atnaujinimus sistemoje
sudo apt-get update
// Komanda įdiegti "Mosquito" programie įrangą
sudo apt-get install mosquito
// Komanda įdiegti klientų palaikymą:
sudo apt-get install mosquito-clients
// Išvalyti visas likusias repozitorijas
Sudo apt clean
```

pav. 63 Naudotos komandos įdiegti serverį

Šiame etape įdiegus serverį, buvo pastebėta, kad, nėra būdų prijungti “Google Cloud PUB/SUB”, nes šiai sistemai neužtenka adaptuoti “Mosquito” konfigūracinį failą. Taip pat testuojant sukurta serverį pastebėta, kad sistema neprisijungia prie “Google”, nes “Google” tikisi kliento iš kurio atliekamas prisijungimas ir nesitiki MQTT serverio žinučių . Ši problema išsprendžiama sukūrus MQTT klientą kuris priima duomenis iš “Mosquito” brokerio ir perduoda į “Pub/Sub” prenumeratos metodu. Kita bėda buvo pastebėta ir jungiantis prie TTN. Kaip antrajame tyrime buvo išsiaiškinta, TTN MQTT pats irgi yra brokeris ir nebepalaiko funkcijos būti klientu. Sprendimai tai išspręsti yra šie:

1. Sukurti tiltą tarp “TTN” ir “Mosquito” brokerio. Tai bus galima atlikti panašiu principu kaip antrame tyrime, tik vietoj lokalaus kliento, galima naudoti “Mosquito” serverį ir jame sukurti MQTT tiltą.
2. Jungimui tarp “Mosquito” brokerio ir “Google Cloud”, bus naudojamas papildomas scenarijus, kur duomenys bus konvertuojami į “Cloud Pub/Sub” suprantamus duomenis.

Tiltui tarp TTN ir “Mosquito” brokeriui sukurti yra naudojamas konfigūracinis failas iš “Mosquito” MQTT (žr. 64 Pav.).



pav. 64 „Mosquitto“ konfigūravimo failas

Su šiuo konfigūravimo failu, galima prisijungti prie TTN sistemos. Taip išsprendžiama pirma bėda kaip priimti duomenis iš TTN serverio.

Kitas žingsnis, yra sukurti ryšį tarp sukurto serverio ir “Google Cloud”. Tai yra atliekama sukuriant MQTT ir PUB/SUB klientą, kuris pakeistų ateinančią informaciją iš TTN į tinkamą “Google Cloud” informacijos paketą. Aprašyta komanda (žr. 64 pav.) importavo “Mosquito” kliento palaikymą į sukurtą serverį. Šis MQTT klientas bus konfigūruojamas priimti MQTT brokerio duomenis ir siųsti duomenis į “Google Cloud Pub/Sub”. Problema galima užrašyti ir taip – norint užtikrinti, kad duomenys pasieks “Google” buvo išsiaiškinta kad reikalingas dar vienas tarpininkas, kuris persiųs duomenis gaunamus iš MQTT serverio į klientą ir iš kliento į “Google cloud”. Šiam veiksmui pasinaudojama “Google” jau

suteikiamomis JAVA bibliotekomis [46] ir adaptuojamas kodas priimti ateinančius “Mosquito” MQTT duomenis ir paversti juos į “Google Pub/Sub” naudojamus duomenis (žr. 65 Pav.).

```
public MqttToCloudPubSubKelias() {
    UUID uuid = UUID.randomUUID();
    mqttFromSourceTopicClientId =
     MQTT_CLIENT_ID_PREFIX + MQTT_CLIENT_ID_FROM_SOURCE_TOPIC_PREFIX + uuid;
    mqttToCloudPubSubKelioId = MQTT_TO_CLOUD_PUB_SUB_Kelio_ID_PREFIX + uuid;
}

   String kelioPradzia =
           "Mosquito-mqtt5:" + mqttSourceTopic + "?" + "clientId=" + mqttFromSourceTopicClientId;

    String kelioPabaiga =
           "google-pubsub:" + cloudPubSubProjectId + ":" + cloudPubSubDestinationTopicName;
    LOG.infof("Jusu MQTT serveris %s", kelioPradzia);
    LOG.infof("Google Cloud pabaiga: %s", kelioPabaiga);

    from(kelioPradzia).id(mqttToCloudPubSubKelioId).to(kelioPabaiga);
}

public String getFromSourceTopicMqttClientId() {
    return mqttFromSourceTopicClientId;
}

public String getMqttToCloudPubSubRouteId() {
    return mqttToCloudPubSubRouteId;
}
```

pav. 65 Kodo iškarpa duomenų perdavimui iš MQTT serverio į “Google Cloud Pub/Sub”

Parašius kodą ir užkrovus TTN MQTT ir paleidus sukurtą MQTT klientą, siunčiama žinutė į Google “Testing TTN, first message”. Šios žinutės ateinančius duomenis galima ištraukti pasinaudojant “Google Cloud” konsolę (žr. 66 Pav.).

```
aldasjonauskis@cloudshell:~ (smart-kettle-esp32)$ gcloud pubsub subscriptions pull --auto-ack MosquitoPull
DATA: Testing TTN, first message
MESSAGE_ID: 8995434653456365
```

pav. 66 Atėjusių duomenų ištraukimas Google konsolėje

Tokiu būdu buvo sukurtas metodas perduoti duomenis iš prietaiso *LoRa* protokolu į TTN ir iš TTN į “Google Cloud”.

6.1.4.3 Eksperimentinio tyrimo apribojimai ir grėsmės

Nors eksperimentiniame tyrime buvo pasiektas pagrindinis tikslas – pakeisti duomenų siuntimą naudojant MQTT iš TTN į “IOT core”, reikia paminėti, kad sprendimas nėra pilnavertis. Šiuo sprendimu galima siuntinėti duomenis iš TTN į “Google”, bet šis siuntimas tik galimas iš TTN. Kol kas negalima siųsti duomenų iš “Google” į TTN ir nėra jokios apsaugos siuntimo metodui. Taip pat serveris nėra labai saugus, nes turi tik slaptažodį. Dėl šios priežasties sprendimą dar galima tobulinti ir didinti jo galimybės. Vienas galima variantas yra sukurti universalų keitiklį, kuris pagal iš, kur duomenys ateina juos automatiškai persiųstų į atvirkščią pusę. Bet šis sprendimas reikalauja kito

algoritmo nei buvo naudotas. Tai yra dėl to, kad naudojamas algoritmas neskiria ateinančių duomenų ir visada ieško siųsti į „Google Cloud“.

Hipotezė, kad galima pakeisti “Google IOT core” nuosavu serveriu iš dalies pasiteisino dėl to, kad galima siųsti duomenis iš TTN į “Google Pub/Sub” nenaudojant brokerio.

6.1.4.4 Eksperimentinio tyrimo išvados

Eksperimentinis tyrimas buvo dalinai sėkmingai tuo atžvilgiu, kad buvo galima sėkmingai siųsti duomenis tarp TTN ir “Google Cloud. Tokiu būdu buvo įtvirtintas tyrimo tikslas – rasti variantą išsiųsti duomenis iš vienos platformos į kitą, išvengiant “IOT core” naudojimo, kuris nuo rugpjūčio mėnesio nebebus palaikomas. Tiesa, sukurtas serveris, vis dar negali priimti duomenų iš “Google Cloud” ir autentifikacija neegzistuoja.

IŠVADOS

- Literatūros analizėje, buvo nustatyta, kad nėra žinomas išmanusis virdulys, kuris dirbtų naudojantis *LoRA* protokolu, galėtų prisijungti prie TTN tinklo ir iš ten perduotų informaciją į "Google Cloud". Taip pat atrasta, kad šis gebėjimas prisijungti prie "Google Cloud" suteikia galimybę itin lengvai integruoti sistemą į didesnę IOT ekosistemą. Taip pat nustatyta, kad nėra virdulio, kuris turėtų tris vienu metu palaikomus komunikacijos metodus – *WiFi*, *LoRa* ir *Bluetooth*.
- Buvo sukurta programinė ir aparatinė įranga leidžianti valdyti išmanųjį virdulį iš mobilios programėlės naudojant tris komunikacijos protokolus – *LoRa*, *Bluetooth* ir *WiFi*. Taip pat sistema leidžia matuoti virdulio temperatūrą ir *Bluetooth*, arba *WiFi* protokolais, perduoti temperatūrą tiesiai į mobilią programą, o *LoRa* protokolu persiųsti šiuos duomenis į TTN tinklą ir iš ten šiuos duomenis persiųsti į "Google Cloud". Iš 'Google Cloud" duomenys yra perduodami į "Firebase" duomenų bazę, prie, kurios yra prisijungusi mobili programėlė. Tokiu pačiu principu yra perduodami duomenys iš programėlės į prietaisą naudojant *LoRa* protokolą. Taip pat išsiaiškinta, kad *LoRa* protokolas nėra pats efektyviausias metodas siųsti duomenis iš mobilios programėlės į serverį, nes *LORAWAN* riboja siuntimų skaičių iš TTN į prietaisą (angl. *node*). Taip pat protokolas nėra idealus, nes galima iš prietaiso siųsti duomenis tik kas kelias minutes. Dažnesnis siuntimas rizikuoja prietaiso blokavimu TTN tinkle.
- Rasti efektyviausią *LoRa* duomenų perdavimo įrangą ir metodą, buvo atlikti du tyrimai. Tyrimo metu patikrintos hipotezės dėl geriausios įrangos perduoti duomenis LoRa metodu į TTN ir geriausias duomenų perdavimo metodas iš TTN tinklo į "Google Cloud" tinklą. Pirmasis tyrimas patvirtino hipotezes, kad patikimiausias siųstuvas/vartai yra nuosava *Raspberry Pi* įranga jei duomenys yra siunčiami yra dažniau nei kas 2 min ir mažiausiai patikimas yra 1 kanalo siųstuvas, kurio TTN tinklas nebepalaiko. Antrojo tyrimo metu patvirtinta hipotezė, kad MQTT duomenų perdavimo metodas yra pats efektyviausias norint perduoti duomenis iš TTN į "Google Cloud". Šis metodas nuo antro pranešimo perdavė duomenis iš TTN į "Google Cloud" trečdaliu mažesniu duomenų dydžiu negu alternatyvus "Webhook" metodas.
- Taip pat užtikrinti, kad kai bus atjungtas "Google IOT core" brokeris, kuriuo rėmėsi antrasis tyrimas, egzistuotų alternatyvus sprendimas, buvo atliktas eksperimentinis tyrimas. Šiame tyrime, naudojantis antro tyrimo metu įgytomis žiniomis buvo sukurtas nuosavas MQTT serveris. Šio serverio pagalba, tapo įmanoma išvengti naudoti "Google IOT core" ir visą informaciją ateinančia iš TTN į MQTT tapo įmanoma persiųsti į "Google PUB/SUB".

LITERATŪROS SĄRAŠAS

[1] SOFI M. A., „Bluetooth Protocol in Internet of Things (IoT), Security Challenges and a Comparison with Wi-Fi Protocol: A Review“

[2] RAZAA S., MISRAB P., HEA Z., VOIGT T., 2017 „Building the Internet of Things with bluetooth smart“

[3] ELKHODR M., SHAHRESTANI S. CHEUNG H., 2016 „Emerging Wireless Technologies In The Internet Of Tghings: A Comparative Study“

[4] KHUTSOANE O., ABU-MAHFOUZ A. M., 2017 „IoT Devices and Applications based on LoRa/LoRaWAN“

[5]. WIXTED A. J., KINNAIRD P., LARIJANI H., TAIT A., AHMADINIA A., STRACHAN N., “Evaluation of LoRa and LoRaWAN for wireless sensor networks,”

[6]. ABSAR N., ALAM M. AHMED J., T., „Performance Study of Star Topology in Small Internetworks 2014“

[7] RUSEK F., PERSSON D., LAU B. K., LARSSON E. G., 2013 „Scaling up MIMO: Opportunities and challenges with very large arrays,“ IEEE.

[8] PERAHIA E. 2013 „Next Generation Wireless LANs: 802.11 n and 802.11 ac: Cambridge university press“.

[9] AKYILDIZ. F., SU W., Sankarasubramaniam, and E. Cayirci, „Wireless Sensor Networks: A Survey,“ Computer Networks, vol. 38, pp. 393-422, 2002.

[10] KUSHALNAGAR N, „Ipv6 over LowPower Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919, August 2007“

[11] CHICH-Yung C. A, KUEI S., 2007 “A location-aware multicasting protocol for Bluetooth Location Networks”, Information Sciences 177 3161–3177

[12] Straipsnis „Bluetooth SIG, Specification of the Bluetooth system – wireless connections made easy, Bluetooth Core Specification v1.1, 2001“: http://www.bluetooth.org/spec

[13] SOURON, E., OUSSALAH, M., & ALAKHRAS, M. (2012). Bluetooth -model analysis and simulation using NS2. Proceedings of the 11th IEEE International Conference on Cybernetic Intelligent Systems 2012, CIS 2012, 178–183. https://doi.org/10.1109/CIS.2013.6782153

[14] GOLDIS V., 2016 „Mobile Computing Using Android With An Emphasizes On Economic Application“.

[15] FOJTIK R., 2020 „Swift a new programming language for development and education“

[16] UDAI A. D., 2015 „A Beginners’ Guide to ATMEL AVR Development“

[17] BANZI M., 2015 „Getting Started with Arduino. Maker Media Inc“

[18] SHAH M. B. N., ZAILAN N., ABIDIN A. F. Z., 2019 „PID-based temperature control device for electric kettle“

[19] JO H., 2018, „Intelligent smart home energy efficiency model using artificial TensorFlow engine“

[20] BENNET S., 1984 “Nicholas Minorsky and the Automatic Steering of Ships,” IEEE Control Systems Magazine, vol. 4, no. 4, pp. 10–15, November.

[21] ASTR K.˚ 1995 ¨ PID Controllers: Theory, Design and Tuning. Instrument Society of America,.

[22] K. Astrom, 1995 “PID controllers: theory, design and tuning”, USA: Instrument Society of America;.

[23] MALWATKAR, 2011 „PID controllers for higher order systems based on maximum sensitivity function“,
[24] Straipsnis 2016 „7th International Conference on Communication, Computing and Virtualization“
[25] AGARWAL T. 2017 „How Bluetooth Module Interfacing with Microcontrollers“
[26] MESQUITA J 2018 „Assessing the ESP8266 WiFi module for the Internet of Things“
[27] FAHAD, E. 2019 „Sensor Monitoring using LoRa by Reyax rylr890/rylr896 and Arduino“
[28] ALAM M. R. 2012 , „A Review of Smart Homes – Past, Present, and Future“,
[29] STERLING G., 2020 „More than 200 million smart speakers have been sold, why aren't they a marketing channel“, [žiūrėta 2021 1 15].
https://marketingland.com/more-than-200-million-smart-speakers-have-been-sold-why-arent-they-a-marketing-channel-276012
[30] O'DEA. S., 2020 „Number of smartphone users worldwide from 2016 to 2021“, December 10, [žiūrėta 2021 1 15]
https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/
[31] Straipsnyje „Mobile Operating System Market Share Worldwide – December 2020“ December 10, 2020 [žiūrėta 2021 1 15]
https://gs.statcounter.com/os-market-share/mobile/worldwide
[32] SHAH M. B. 2019 „PID-based temperature control device for electric kettle“
[33] ASTROM K., HAGGLUND T., 1995 "PID controllers: theory, design and tuning", USA: Instrument Society of America;.
[34] MALWATKAR, 2011"PID controllers for higher order systems based on maximum sensitivity function“, 2011 3rd International Conference on Electronics Computer Technology,.
[35] 7th International Conference on Communication, Computing and Virtualization 2016
[36] GOLDIS. V., 2016 „Mobile Computing Using Android With An Emphasizes On Economic Application“.
[37] FOJTIK. R., 2020 „Swift a new programming language for development and education“
[38] UDAI. D., 2015 „A Beginners' Guide to ATMEL AVR Development“
[39] PAUL, B. 2021 „An overview of LoRaWan“
[40] HAXHIBEQIRI, J., KARAAGAC, A., ABEELE, F., V. 2017. LoRa indoor coverage and performance in an industrial environment: Case study, doi: 10.1109/ETFA.2017.8247601
[41] MALAWSKI, M. 2017. Serverless execution of scientific workflows: Experiments with HyperFlow, AWS Lambda and Google Cloud Functions, doi: 502-514.
[42] KAVATS, E. 2019. Analysis of connection methods of telegram robots with server part, Vol. 3 No. 122 (2019): System technologies doi: https://doi.org/10.34185/1562-9945-3-122-2019-03
[43] HUMFREY N. 2020. RSMB: Really Small Message Broker,
https://github.com/eclipse/mosquitto.rsmb
[44] Straipsnyje *IOT core release notes.* Puslapio nuoroda
[https://cloud.google.com/iot/docs/release-notes]
[45] ILIEVA G. 2022 IoT System Selection as a Fuzzy Multi-Criteria Problem doi:
https://doi.org/10.3390/s22114110
[46] Straipsnyje Google Cloud IOT core documentation. Puslapio nuoroda:
https://cloud.google.com/iot/docs/how-tos/mqtt-bridge

INFORMACIJOS ŠALTINIŲ SĄRAŠAS

{1} Internetinis puslapis „Smarter Ikettle“, nuoroda:
https://smarter.am/collections/smarter-ikettle
{2} Internetinis puslapis „Fellow products“, nuoroda:
https://fellowproducts.com/products/staggekg
{3} Internetinis puslapis „My app Kettle“, nuoroda:
https://www.myappkettle.com/features
{4} Internetinis puslapis „TTN mapper“, nuoroda:
https://ttnmapper.org/heatmap/
{5} Internetinis puslapis „The Things Network“, nuoroda: https://www.thethingsnetwork.org/

PRIEDAI

1 Priedas. „Webhook“ metodo siunčiamo iš TTN į „Google Cloud“ standartinė forma {5}

```
{
  "end_device_ids" : {
    "device_id" : "dev1",                    // Device ID
    "application_ids" : {
      "application_id" : "app1"              // Application ID
    },
    "dev_eui" : "0004A30B001C0530",          // DevEUI of the end device
    "join_eui" : "800000000000000C",         // JoinEUI of the end device (also known as AppEUI in LoRaW
    "dev_addr" : "00BCB929"                  // Device address known by the Network Server
  },
  "correlation_ids" : [ "as:up:01...", ... ],// Correlation identifiers of the message
  "received_at" : "2020-02-12T15:15..."      // ISO 8601 UTC timestamp at which the message has been rec
  "uplink_message" : {
    "session_key_id": "AXA50...",            // Join Server issued identifier for the session keys used
    "f_cnt": 1,                              // Frame counter
    "f_port": 1,                             // Frame port
    "frm_payload": "gkHe",                   // Frame payload (Base64)
    "decoded_payload" : {                    // Decoded payload object, decoded by the device payload fo
      "temperature": 1.0,
      "luminosity": 0.64
    },
    "rx_metadata": [{                        // A list of metadata for each antenna of each gateway that
      "gateway_ids": {
        "gateway_id": "gtw1",                // Gateway ID
        "eui": "9C5C8E00001A05C4"            // Gateway EUI
      },
      "time": "2020-02-12T15:15:45.787Z",    // ISO 8601 UTC timestamp at which the uplink has been rece
      "timestamp": 2463457000,               // Timestamp of the gateway concentrator when the message h
      "rssi": -35,                           // Received signal strength indicator (dBm)
      "channel_rssi": -35,                   // Received signal strength indicator of the channel (dBm)
      "snr": 5.2,                            // Signal-to-noise ratio (dB)
      "uplink_token": "ChIKEA...",           // Uplink token injected by gateway, Gateway Server or fNS
      "channel_index": 2                     // Index of the gateway channel that received the message
      "location": {                          // Gateway location metadata (only for gateways with locati
        "latitude": 37.97155556731436,       // Location latitude
        "longitude": 23.72678801175413,      // Location longitude
        "altitude": 2,                       // Location altitude
        "source": "SOURCE_REGISTRY"          // Location source. SOURCE_REGISTRY is the location from th
      }
    }],
    "settings": {                            // Settings for the transmission
      "data_rate": {                         // Data rate settings
        "lora": {                            // LoRa modulation settings
          "bandwidth": 125000,               // Bandwidth (Hz)
          "spreading_factor": 7              // Spreading factor
        }
      },
      "coding_rate": "4/6",                  // LoRa coding rate
      "frequency": "868300000",              // Frequency (Hz)
    },
    "received_at": "2020-02-12T15:15..."     // ISO 8601 UTC timestamp at which the uplink has been rece
    "consumed_airtime": "0.056576s",         // Time-on-air, calculated by the Network Server using payl
    "locations": {                           // End device location metadata
      "user": {
        "latitude": 37.97155556731436,       // Location latitude
        "longitude": 23.72678801175413,      // Location longitude
        "altitude": 10,                      // Location altitude
        "source": "SOURCE_REGISTRY"          // Location source. SOURCE_REGISTRY is the location from th
      }
    },
    "version_ids": {                          // End device version information
        "brand_id": "the-things-products",    // Device brand
        "model_id": "the-things-uno",         // Device model
        "hardware_version": "1.0",            // Device hardware version
        "firmware_version": "quickstart",     // Device firmware version
        "band_id": "EU_863_870"               // Supported band ID
    },
    "network_ids": {                          // Network information
      "net_id": "000013",                     // Network ID
      "tenant_id": "tenant1",                 // Tenant ID
      "cluster_id": "eu1"                     // Cluster ID
    }
  },
```

pav. 67 „Webhook“ metodo šablonas

4 Priedas. „MQTT“ metodo siunčiamo iš TTN į „Google Cloud“ prisijungimo žinutės standartinė forma:

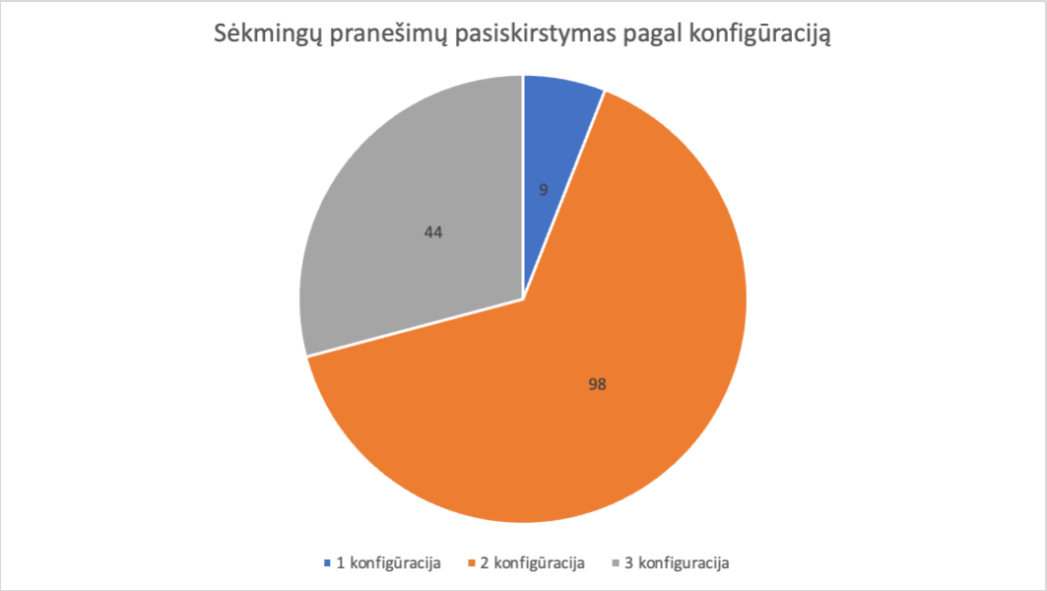

```
{
  "end_device_ids": {
    "device_id": "dev1",
    "application_ids": {
      "application_id": "app1"
    },
    "dev_eui": "4200000000000000",
    "join_eui": "4200000000000000",
    "dev_addr": "01DA1F15"
  },
  "correlation_ids": [
    "gs:conn:01D2CSNX7FJVKQPCVG612QF1TX",
    "gs:uplink:01D2CT834K2YD17ZWZ6357HC0Z",
    "ns:uplink:01D2CT834KNYD7BT2NHK5R1WVA",
    "rpc:/ttn.lorawan.v3.GsNs/HandleUplink:01D2CT834KJ4AVSD1SJ637NAV6",
    "as:up:01D2CT83AXQFQYQ35SR74CTWKH"
  ],
  "join_accept": {
    "session_key_id": "AWiZpAyXrAfEkUNkBljRoA=="
  }
}
```


5 Priedas. „MQTT“ metodo siunčiamo iš TTN į „Google Cloud“ turinio standartinė forma {5}

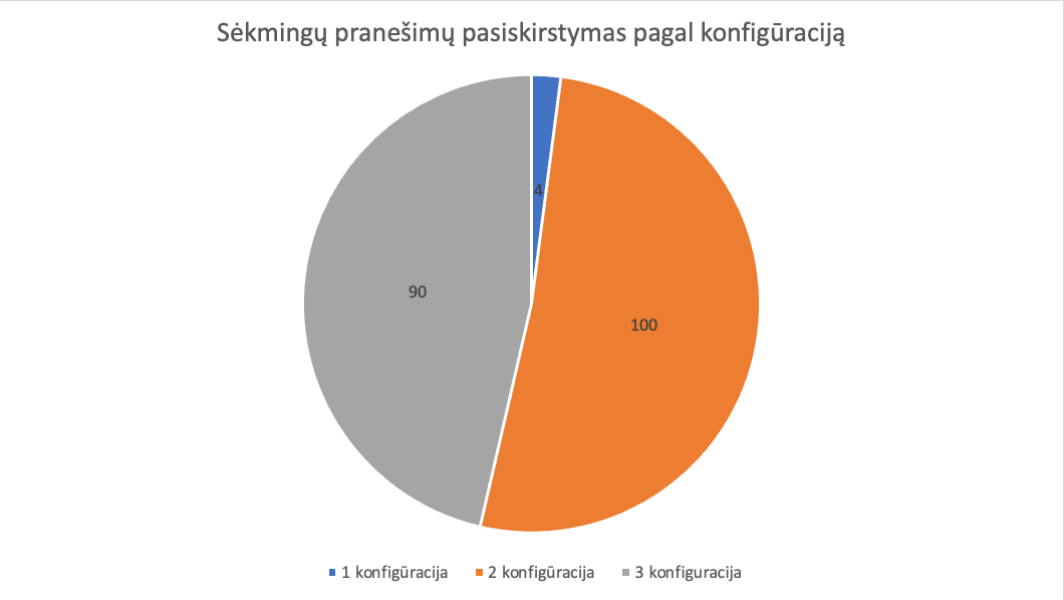
```
{
  "end_device_ids": {
    "device_id": "dev1",
    "application_ids": {
      "application_id": "app1"
    },
    "dev_eui": "4200000000000000",
    "join_eui": "4200000000000000",
    "dev_addr": "01DA1F15"
  },
  "correlation_ids": [
    "gs:conn:01D2CSNX7FJVKQPCVG612QF1TX",
    "gs:uplink:01D2CV8HF62ME0D7MZWE38HHH8",
    "ns:uplink:01D2CV8HF6FYJHKZ45YY1DB3MR",
    "rpc:/ttn.lorawan.v3.GsNs/HandleUplink:01D2CV8HF6XR7ZFVK768PDG3J4",
    "as:up:01D2CV8HNGJ57G25BW0FCZNY07"
  ],
  "uplink_message": {
    "session_key_id": "AWiZpAyXrAfEkUNkBljRoA==",
    "f_port": 15,
    "frm_payload": "VGVtcGVyYXR1cmUgPSAwLjA=",
    "rx_metadata": [{
      "gateway_ids": {
        "gateway_id": "eui-0242020000247803",
        "eui": "0242020000247803"
      },
      "time": "2019-01-29T13:02:34.981Z",
      "timestamp": 1283325000,
      "rssi": -35,
      "snr": 5,
      "uplink_token": "CiIKIAoUZXVpLTAyNDIwMjAwMDAyNDc4MDMSCAJCAgAAJHgDEMj49+ME"
    }],
    "settings": {
      "data_rate": {
        "lora": {
          "bandwidth": 125000,
          "spreading_factor": 7
        }
      },
      "coding_rate": "4/6",
      "frequency": "868500000",
      "gateway_channel_index": 2,
      "device_channel_index": 2
    }
  }
}
```


pav. 68 „Webhook“ metodo šablonas

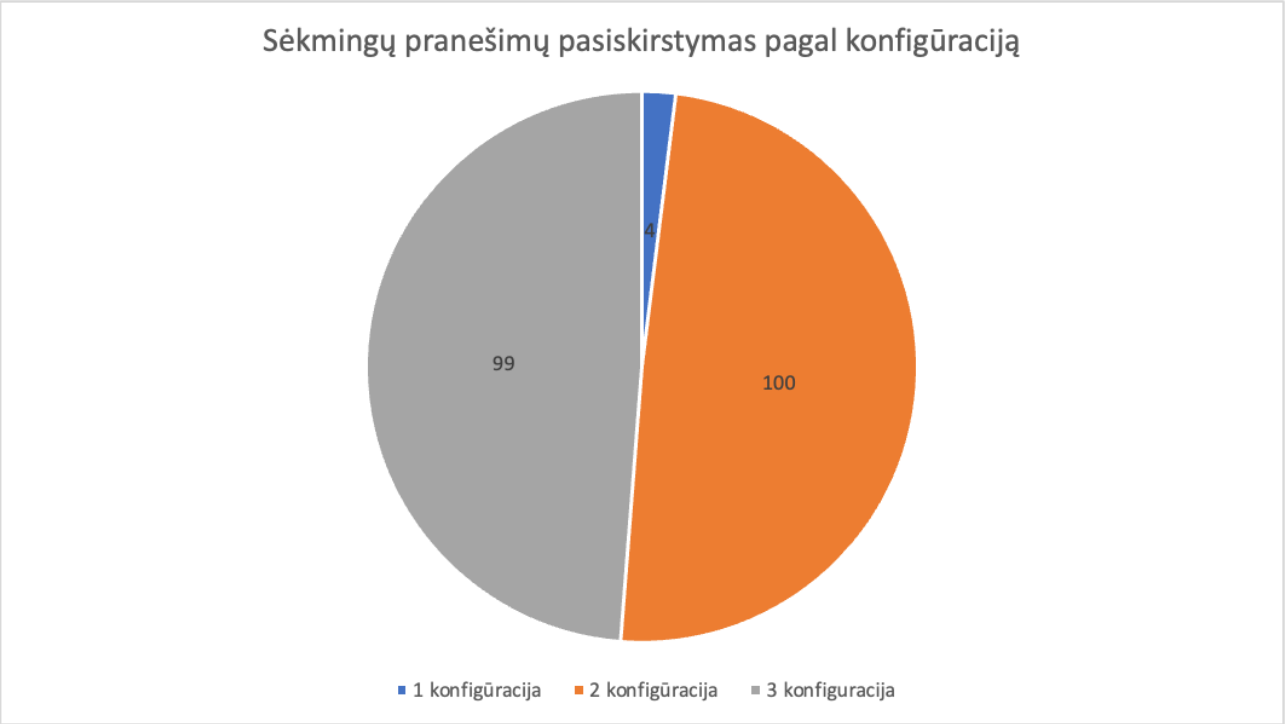
4 Priedas. Papildomos diagramos sėkmingų pranešimų atvaizdavimui



pav. 69 Pranešimų pasiskirstymas siunčiant duomenis kas 1 minutę.



pav. 70 Pranešimų pasiskirstymas siunčiant duomenis kas 2 minutes.



pav. 71 Pranešimų pasiskirstymas siunčiant duomenis kas 5 minutes.