



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS  
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS  
INFORMACINIŲ SISTEMŲ KATEDRA

Vygintas Marčiulaitis

**BALSU VALDOMAS NAVIGACINIS ROBOTAS**  
**VOICE CONTROLLED NAVIGATIONAL ROBOT**

Baigiamasis bakalauro darbas

Programų inžinerijos studijų programa, valstybinis kodas 6121BX023  
Programų sistemų studijų kryptis

Vilnius, 2023

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS  
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS  
INFORMACINIŲ SISTEMŲ KATEDRA

TVIRTINU

Katedros vedėjas

\_\_\_\_\_  
(Parašas)

\_\_\_\_\_  
(Vardas, pavardė)

\_\_\_\_\_  
(Data)

Vygintas Marčiulaitis

**BALSU VALDOMAS NAVIGACINIS ROBOTAS**  
**VOICE CONTROLLED NAVIGATIONAL ROBOT**

Baigiamasis bakalauro darbas

Programų inžinerijos studijų programa, valstybinis kodas 6121BX023  
Programų sistemų studijų kryptis

**Vadovas** Algirdas Laukaitis \_\_\_\_\_  
(Pedag. vardas, vardas, pavardė) (Parašas) (Data)

**Konsultantas** \_\_\_\_\_  
(Pedag. vardas, vardas, pavardė) (Parašas) (Data)

**Konsultantas** \_\_\_\_\_  
(Pedag. vardas, vardas, pavardė) (Parašas) (Data)

Vilnius, 2023

## **Turinys**

1. ĮVADAS .....	7
1.1. Darbo aktualumas ir tikslas .....	8
1.2. Darbo uždaviniai.....	8
1.3. Darbo naujumas ir praktinė nauda.....	8
1.4. Darbo struktūra .....	9
2. DALYKINĖS SRITIES ANALIZĖ .....	10
2.1. Panašių sistemų analizė .....	11
2.2. Sistemos vizija.....	13
3. BALSO ATPAŽINIMO ALGORITMŲ ANALIZĖ .....	14
3.1. Neuroniniai tinklai .....	14
3.2. Rekurentiniai neuroniniai tinklai .....	15
3.3. Konvulciniai neuroniniai tinklai .....	16
3.4. Išvados.....	19
3.5. Balso atpažinimo technologijos.....	20
4. NAVIGACINIS ROBOTAS .....	22
4.1. Mikrokontroleriai .....	22
4.2. Arduino.....	24
4.3. Raspberry Pi .....	31
4.4. Išvados.....	36
4.5. Kontroleris .....	37
4.6. Programavimo kalba.....	39
5. SISTEMOS ARCHITEKTŪRA.....	41
5.1. Sistemos specifikacija.....	41
5.2. Architektūra .....	48
5.3. PS struktūra .....	54
6. SISTEMOS ĮGYVENDINIMAS .....	56
6.1. Neuroninių tinklų apmokymas .....	56
6.2. Navigacinis robotas .....	60
6.3. Kontrolerio kūrimas.....	63
6.4. Sistemos vizualizacija.....	66
7. TESTAVIMAS .....	69
7.1. Balso atpažinimo testavimas .....	69
7.2. Bluetooth ryšio testavimas.....	69
7.3. Rankinis roboto testavimas.....	70
8. IŠVADOS .....	71
LITERATŪROS SARAŠAS.....	72

## LENTELIŲ SĄRAŠAS

LENTELĖ 1 RNN PRIVALUMAI IR TRŪKUMAI .....	16
LENTELĖ 2 CNN PRIVALUMAI IR TRŪKUMAI .....	19
LENTELĖ 3 NEURONINIŲ TINKLŲ PALYGINIMAS .....	20
LENTELĖ 5 API.AI PRIVALUMAI IR TRŪKUMAI .....	21
LENTELĖ 6 WIT.AI PRIVALUMAI IR TRŪKUMAI .....	21
LENTELĖ 7 BALSO ATPAŽINIMO TECHNOLOGIJŲ PALYGINIMAS.....	22
LENTELĖ 8 MIKROVALDIKLIŲ KOMPONENTAI .....	23
LENTELĖ 9 ARDUINO UNO R3 PRIVALUMAI IR TRŪKUMAI .....	26
LENTELĖ 10 ARDUINO NANO PRIVALUMAI IR TRŪKUMAI.....	27
LENTELĖ 11 ARDUINO MEGA PRIVALUMAI IR TRŪKUMAI.....	27
LENTELĖ 12 ARDUINO MODELIŲ PALYGINIMAS .....	28
LENTELĖ 13 RASPBERRY PI KOMPONENTAI .....	33
LENTELĖ 14 RASPBERRY PI MODELIŲ PALYGINIMAS.....	35
LENTELĖ 15 ARDUINO IR RASPBERRY PI PALYGINIMAS.....	36
LENTELĖ 16 ARDUINO IR RASPBERRY PI PANAUDOJAMUMAS .....	37
LENTELĖ 17 IDE PALYGINIMAS .....	38
LENTELĖ 18 PROGRAMAVIMO KALBŲ PALYGINIMAS .....	40
LENTELĖ 19 SISTEMOS NAUDOTOJŲ ISTORIJŲ PAAIŠKINIMAI.....	44
LENTELĖ 20 SEKŲ DIAGRAMOS PAAIŠKINIMAS .....	46
LENTELĖ 21 SEKŲ DIAGRAMOS PAAIŠKINIMAS .....	47
LENTELĖ 22 SEKŲ DIAGRAMOS PAAIŠKINIMAS .....	48
LENTELĖ 23 REIKALAVIMŲ MATRICA .....	49
LENTELĖ 24 REIKALAVIMŲ MATRICA KOMPONENTAMS.....	50
LENTELĖ 25 REIKALAVIMŲ RYŠIO MATRICA .....	50
LENTELĖ 26 ROBOTO KOMPONENTAI.....	60
LENTELĖ 27 KOMANDŲ PAAIŠKINIMAI.....	62
LENTELĖ 28 RANKINIO TESTAVIMO REZULTATAI.....	70

## PAVEIKSLŲ SĄRAŠAS

PAV. 1 <i>SCORPIO</i> KONTROLERIS .....	11
PAV. 2 <i>SCORPIO</i> .....	11
PAV. 3 <i>SCORPIO</i> ARCHITEKTŪRA .....	12
PAV. 4 NEURONINIO TINKLO VEIKIMAS .....	14
PAV. 5 REKURENTINIAI NEURONINIAI TINKLAI .....	15
PAV. 6 CNN VEIKIMAS .....	16
PAV. 7 CNN FILTRO VEIKIMAS .....	17
PAV. 8 KONVULCINIŲ SLUOKSNIŲ HIERARCHIJA .....	17
PAV. 9 MAKSIMALIOS REIKŠMĖS FILTRAVIMAS .....	18
PAV. 10 VIDUTINĖS REIKŠMĖS FILTRAVIMAS .....	19
PAV. 11 HARVARDO ARCHITEKTŪRA .....	23
PAV. 12 VON-NEUMANNO ARCHITEKTŪRA .....	23
PAV. 13 ARDUINO ARCHITEKTŪRA .....	25
PAV. 14 ARDUINO UNO R3 .....	26
PAV. 15 ARDUINO NANO .....	26
PAV. 16 ARDUINO MEGA .....	27
PAV. 17 SKYDO JUNGTYŚ .....	29
PAV. 18 HC-SR04 ULTRAGARSO SENSORIUS .....	30
PAV. 19 SENSORIAUS VEIKIMAS .....	30
PAV. 20 HC-05 <i>BLUETOOTH</i> MODELIS .....	31
PAV. 21 RASPBERRY PI APARATŪRA .....	32
PAV. 22 PI 3 MODEL B .....	34
PAV. 23 RASPBERRY PI ZERO W .....	34
PAV. 24 RASPBERRY PI 4 MODEL B .....	35
PAV. 25 SISTEMOS VAIZDUSIS PAVEIKSLĖLIS .....	41
PAV. 26 SISTEMOS NAUDOTOJŲ DIAGRAMA .....	43
PAV. 27 UŽDUOTIES “SIUSTI KOMANDĄ” SEKŲ DIAGRAMA .....	45
PAV. 28 UŽDUOTIES “PRISIJUNGTI PRIE ĮRENGINIO” SEKŲ DIAGRAMA .....	46
PAV. 29 UŽDUOTIES “SKENUOTI ĮRENGINIUS” SEKŲ DIAGRAMA .....	47
PAV. 30 SISTEMOS KOMPONENTŲ DIAGRAMA .....	48
PAV. 31 KOMPONENTŲ TARPUSAVIO SĄVEIKOS SEKŲ DIAGRAMA .....	52
PAV. 32 SISTEMOS ĮGYVENDINIMO DIAGRAMA .....	53
PAV. 33 SISTEMOS DUOMENŲ SRAUTO DIAGRAMA .....	54
PAV. 34 SISTEMOS KLASIŲ DIAGRAMA .....	54
PAV. 35 SISTEMOS OBJEKTŲ DIAGRAMA .....	56
PAV. 36 KOMANDOS “LEFT” SPEKTROGRAMA .....	57
PAV. 37 RELU VEIKIMAS .....	58
PAV. 38 TRENIRAVIMO TIKSLUMAS IR PRARADIMAS .....	58
PAV. 39 VALIDACIJOS TIKSLUMAS IR PRARADIMAS .....	59
PAV. 40 MODELIO PRARADIMAS PO OPTIMIZACIJOS .....	59
PAV. 41 ROBOTO ELEKTROS GRANDINĖ .....	61
PAV. 42 NAVIGACINIS ROBOTAS .....	61
PAV. 43 ROBOTO VEIKLOS DIAGRAMA .....	63
PAV. 44 BALSO ATPAŽINIMO VEIKLOS DIAGRAMA .....	64
PAV. 45 KOMANDA “DEŠINĖN” .....	65
PAV. 46 PROGRAMĖLĖS VEIKLOS DIAGRAMA .....	66
PAV. 47 PAGRINDINIS LANGAS .....	67
PAV. 48 PROGRAMOS PROGRESO MATUOKLIS .....	68
PAV. 49 KOMANDŲ SIUNTIMO LANGAS .....	68
PAV. 50 UNIT TESTAI BALSO ATPAŽINIMO KOMPONENTUI .....	69
PAV. 51 UNIT TESTAI <i>BLUETOOTH</i> RYŠIO KOMPONENTUI .....	70
PAV. 52 ARDUINO UNO KOMPONENTAI .....	75

## **PRIEDŲ SĄRAŠAS**

PRIEDAS 1 ARDUINO APARATŪROS KOMPONENTAI .....	74
2 PRIEDAS - KODASROBOTUI.CPP .....	76

## 1. ĮVADAS

Balso atpažinimas (Angl. *voice recognition*) yra plačiai besivystanti dirbtinio intelekto šaka. Jos tikslas suprogramuoti kompiuterį taip, kad šis galėtų atpažinti audio failuose tariamus žodžius ir dirbti su jais (M.Lee, 2020).

Šešiasdešimt metų mašinų atliekami automatinio kalbos atpažinimo tyrimai sulaukė didelio susidomėjimo dėl įvairių priežasčių – nuo mokslinio smalsumo apie mechaninio žmogaus kalbos gebėjimų realizavimo įrankius iki prašymo automatizuoti valdomas užduotis, kurioms reikia žmogaus ir mašinos sąveikos (N.Fadhil, 2021).

Balso atpažinimo technologija plačiai naudojama robotikos industrijoje. Šiais laikais žmonės ir robotai gali naudotis bendra darbo vieta tuo pačiu metu. Tokiais atvejais bendravimas yra pagrindinis veiksnys, užtikrinantis darbuotojo saugą ir roboto priėmimą. Norint sėkmingai atlikti užduotį, reikia robotus išmokyti suprasti garsines komandas (CanBingol, 2021).

Autonominės navigacijos sistemos pastaraisiais dešimtmečiais sukėlė didelį susidomėjimą robotikos sritimi dėl daugybės pritaikymų, kuriuose robotai sugeba naviguoti aplinkoje. Tokie robotai plačiai naudojami sveikatos apsaugoje, savaeigiuose automobiliuose, karo pramonėje. Literatūroje buvo pasiūlyti įvairūs autonominės navigacijos judesių planavimo algoritmai, tokie kaip grafiko paieška, erdvės atranka, tolstantis horizontas, optimalus valdymas ir simbolinis planavimas (J.A.Dulce-Galindo, 2022).

Šis darbas tiria galimybę sujungti balso kontrolės ir robotų navigacijos sistemą į vieną.

## **1.1. Darbo aktualumas ir tikslas**

Balso atpažinimo technologiją galima pritaikyti ir robotų navigacijoje. Susidūrimo su kliūtimis išvengimas yra pagrindinė saugios navigacijos sudedamoji dalis. Net ir naudojant pažangius matematinius algoritmus neįmanoma užtikrinti, kad robotas visada sėkmingai aptiks kliutį arba prisitaikys prie besikeičiančios aplinkos (S.Matveev, 2021). Su tokiomis problemomis dažniausiai susidūria dronai arba miškinguose reljefuose dirbančios sistemos (pvz. Pasienio patruliavimo robotai, žmonių gelbėjimo robotai). Valdymo balsu galimybė leistų atsakingiems asmenims bet kada pakeisti roboto kryptį arba ją sustabdyti.

**Šio darbo tikslas** – Sumažinti navigacinių robotų susidūrimo su kliūtimis tikimybę, pritaikant balso atpažinimą.

## **1.2. Darbo uždaviniai**

1. Išanalizuoti dalykinę sritį ir palyginti panašias funkcijas atliekančias sistemas
2. Atlikti balso atpažinimo algoritmų ir neuroninių tinklų analitinę apžvalgą
3. Atlikti techninių įrankių palyginamąją analizę
4. Apmokyti neuroninius tinklus skirtus balso atpažinimui
5. Sukurti navigacinį robotą, kontrolerį
6. Ištestuoti robotą ir neuroninius tinklus

## **1.3. Darbo naujumas ir praktinė nauda**

Balso atpažinimas šiais laikais dažniausiai naudojamas identifikacijos ar automatinių atsakiklių paslaugose. Balso kontrolei neužtenka tiesiog atpažinti audio faile esančius duomenys, bet ir susieti juos su atitinkama komanda.

Dirbtinis neuroninis tinklas yra tik pirma šio projekto dalis, galutinis tikslas yra integruoti šią sistemą į robotų navigaciją.

Jau anksčiau buvo kurti panašūs robotai – balsu valdomas egzoskeletas (YunfeiGuo, 2022) balso komandomis kontroliuojami automobiliai (Tesla).

Roboto praktinė nauda slypi jo gebėjime išvengti kliūčių ir klausyti balso komandų. Tokia technologija galėtų praversti sudėtingo reljefo tyrinėjimuose, karo pramonėje, žvalgyboje.



## 1.4. Darbo struktūra

Darbą sudaro 7 pagrindiniai skyriai:

**Pirmame skyriuje** yra darbo įvadas.

**Antrame skyriuje** yra aptariama robotikos dalykinė sritis. Yra paaiškinama, su kokiomis problemomis susiduria navigaciniai robotai. Taip pat yra aptariama sistemos vizija – kaip ji turėtų atrodyti.

**Trečiame skyriuje** yra aptariami neuroniniai tinklai, jų rūšys.

**Ketvirtame skyriuje** yra kalbama apie navigacinį robotą. Apžvelgiamos jo konstrukcinės dalys, operacinė Sistema. Taip pat apžvelgiama numatomo roboto konstrukcijos schema, jo funkcijos. Įvykdoma panašių technologijų bei mikrokontrolerių analizė.

**Penktame skyriuje** yra sistemos specifikacija, funkciniai ir nefunkciniai reikalavimai, architektūra ir funkcijų aprašymai su UML diagramomis.

**Šeštame skyriuje** apibūdinamas sistemos veikimas, yra parodomos roboto funkcionalumo vizualizacijos.

**Septintame skyriuje** yra aprašomas roboto ir neuroninių tinklu testavimas.

## 2. DALYKINĖS SRITIES ANALIZĖ

Robotika yra viena iš sparčiausiai besivystančių technologijos šakų. Jos užduotys apėmė robotų dizainą, konstrukciją, operavimą ir panaudojimą. Kuriant robotą reikia atsižvelgti į daugybę skirtingų kriterijų. Kokia bus kuriamos sistemos paskirtis? Kokias funkcijas turės minima sistema? Kokios technologijos bus panaudotos sistemos įgyvendinimui?

Norėdami išspręsti šiuos klausimus prie roboto dirba inžinieriai, elektromechanikai, dizaineriai ir programų inžinieriai. Nors robotas gali turėti begalę paskirčių, dauguma robotų susideda iš dvejų esminių dalių – Hardware (Liet. Aparatūra) ir Software (Liet. Programinė įranga). Norint tiek vieną, tiek kitą sritį atlikti teisingai ir produktyviai svarbu pasirinkti tinkamas technologijas. Robotų aparatūra remiasi mikrokontroleriais arba mikrovaldikliais. „Mikrovaldiklis yra kompaktiškas integrinis grandynas, skirtas valdyti konkrečias įterptosios sistemos operacijas. Įprastas mikrovaldiklis apima procesorių, atmintį ir įvesties/išvesties (I/O) periferinius įrenginius viename luste.“ (Lutkevich, 2022). Galima rinktis iš daugybės skirtingų mikrokontrolerių, tačiau populiariausi yra *Arduino*, *Rasbery Pi* ir *ESP8266*. Skirtingi mikrokontroleriai turi skirtingus privalumus, todėl jų pasirinkimas priklauso nuo projekto tikslo.

Nuo kruopštaus derliaus nuėmimo iki automobilių surinkimo ir vaistų pristatymo – robotų sprendimai didina produktyvumą, saugumą ir suteikia daugiau lankstumo įvairiose pramonės šakose (Intel.com). Pagrindinės robotų naudojamų pramonėje yra: humanoidai (Angl. humanoids), kobotai (Angl. Cobots), robotiškos rankos (Angl. robotic arms), automatizuotos transporto priemonės (Angl. Automated Guided Vehicles) ir autonominiai arba navigaciniai robotai (Angl. Autonomous Mobile Robots) (Lutkevich, 2022).

Robotų navigacija plačiai naudojama transporto, gynybos ir kosmoso industrijose. Šie robotai atlieka tokias funkcijas kaip patruliavimas, gelbėjimas, medžiagų transportavimas. Todėl reikalingas autonominis robotas, galintis laisvai judėti statinėje ar dinamiškoje aplinkoje. Priversti robotą nuvykti iš taško A į tašką B yra naudojami kelio paieškos algoritmai (Angl. pathfinding), tačiau dar reikia pasirūpinti, kad robotas nesusidurtų su galimomis kliūtimis. Tam naudojami sensoriai. Sensorius tai integruojama aparatūros dalis galinti išgauti duomenis iš aplinkos ir perduoti juos mikrokontroleriui (J.A.Dulce-Galindo, 2022).

Problema, kurią bandau išspręsti savo darbu – susidūrimo su kliūtimis tikimybės sumažinimas pritaikant balso kontrolę. Net ir naudojant moderniausius sensorius neįmanoma visiškai užtikrinti, kad sistema išvengs visų aplinkos kelemų grėsmių. Kliūtis gali būti judanti, nepatekti į sensoriaus radarą arba būti paslėpta. Galimybė bet kada sustabdyti arba pakeisti roboto kryptį yra kritinis aspektas saugioje navigacijoje.

## 2.1. Panašių sistemų analizė

### „Scorpio“

*SCORPIO* yra nedidelis autonominis robotas skirtas aptikti spąstus. Šią sistemą 2013 metais įgyvendino techninė bendrovė „ZTS VVU Kosice“. Jis gali tarnauti keliems tikslams, tokiems kaip stebėjimas, manevravimas ir kliučių išvengimas pavojingose ar sudėtingo reljefo vietovėse. Robotas sugeba patekti iš taško A į tašką B ir kartu su savimi gabenti reikiamą krovinį (pvz. vandens). Pagrindinė Sistema susideda iš dvejų dalių – fizinio roboto (figūra 1) ir kontrolerio (figūra 2) (Ondas, 2013).



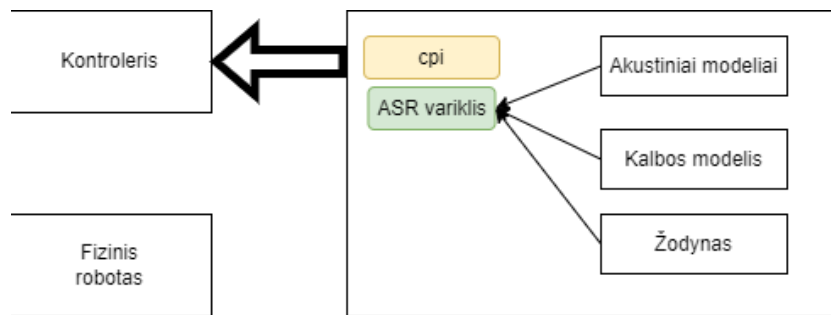
pav. 1 *Scorpio* kontroleris



pav. 2 *Scorpio*

*SCORPIO* robotas turi penkias monochromatines BW kameras (dvi priekines, dvi galines, vieną plačiakampį viršuje), vieną spalvotą kamerą krypties nustatymui (visos kameros yra analoginės), dvi lazerines krypties rodykles, tris nuotolio ieškiklius (priekyje, gale) ir septynis žibintus (du priekiniai, du galiniai, du viršutiniai, vienos krypties nustatymas). Robotas taip pat turi integruotą kompiuterį, kuris leidžia distancinę kontrolę. Belaidžiu būdu prijungtas nešiojamas valdymo įtaisas leidžia operatoriui rankiniu būdu valdyti robotą tiesiog naudojant vairasvirtę, klaviatūrą ir iš anksto įdiegtus mygtukus. Tokios sąsajos apribojimas yra tas, kad yra daug įrenginių (funkcijų), kuriuos sunku vienu metu valdyti. Operatoriui reikia rankų, kad galėtų valdyti roboto judėjimą ir kai kurias kitas funkcijas, pavyzdžiui, diržo padėtį. Antra problema yra ta, kad jis ar ji turi nuolat stebėti ekraną su roboto kamerų išvestimi ir negali sutelkti žvilgsnio į valdymo pulto mygtukus. Čia į pagalbą ateina balso atpažinimas (Ondas, 2013).

*SCORPIO* balso valdymo sistema susideda iš dvejų dalių: kontrolerio sąsajos (CPI) ir balso atpažinimo variklio (ASR).



**pav. 3** *Scorpio* architektūra

Akustinio modeliui parametrizavimui ir kurimui buvo naudoti *Markov* modeliai (angl. *Hidden Markov models*). paslėptas Markovo modelis (HMM) yra statistinis metodas, dažnai naudojamas modeliuojant biologines sekas. Ją taikant, seka modeliuojama kaip atskiro stochastinio proceso išvestis, kuri vyksta per daugybę būsenų, kurios yra „paslėptos“ nuo stebėtojo. Modelio treniravimui buvo panaudota fiksuoto telefono ryšio kalbų duomenų bazė. Pirmajame etape dėmesys buvo sutelktas į geriausią modelių suderinimą naudojant HTK pagrįstą plokščiosios paleidimo metodą (angl. *HTK-based flat-start method*). Antrasis etapas buvo sutelktas į galutinių modelių mokymą su fonemų modelių įvertinimu, pagrįstu ankstesniu Viterbi priverstiniu derinimu ir pakartotiniu įvertinimu naudojant pirmyn-atgal algoritmą (angl. *Forward-Backward*).

Sąranka buvo sukurta naudojant esamą įterptąją kompiuterio aplinką su integruotu „Intel x586“ suderinamu „Tiny886ULP8-800/128-L-X“ kompiuteriu. su CPU –1GHz (TM5800), OS: Linux Debian 6, 500 MB atminties, 1 GB atminties OS ir programoms (SD kortelė) ir USB sąsaja (Ondas, 2013).

### „Astro“

„Amazon Astro“, dažnai trumpinama į „Astro“, yra namų robotas, kurį sukūrė Amazon.com, Inc. Jis buvo skirtas namų saugumui stebėti, nuotolinei pagyvenusių giminaičių priežiūrai ir kaip virtualus asistentas, galintis sekti žmogų iš vieno kambario į kitą (Priest, 2022).

Tai mobilus robotas, galintis atlikti daugybę dalykų, pavyzdžiui, nešti gėrimus į konkrečias vietas, leisti muziką ir vaizdo įrašus bei nuotoliniu būdu stebėti įvairias jūsų namų vietas. Jis gali atpažinti jūsų namų gyventojus ir aptikti nepažįstamus žmones. Jis gali reaguoti į neįprastus įvykius, tokius kaip dūžtančio stiklo garsas. Ir jis gali sąveikauti su kitais „Amazon“ išmaniųjų namų įrenginiais, tokiais kaip „Ring“ judesio jutikliai ir kameros (Priest, 2022).

Šis robotas geba atpažinti objektus, jų išvengti arba juos sekti, taip pat *Astro* supranta balso komandas.

*Astro* kaip ir kitos robotikos sistemos turi dvi pagrindines dalis – aparatūra ir programinė įranga. Balso atpažinimui buvo panaudoti „*TinyML*“, mašininio mokymosi modeliai, kurie yra suglaudinti, kad jie veiktų ribotų išteklių turinčiuose įrenginiuose, ir krašto AI aparatinė įranga, procesoriai, kurie specializuojasi efektyviai valdyti giluminius neuroninius tinklus.

Ypač įdomi *Astro* mobilumo dalis. Robotas turi aptikti sienas, duris, objektus, žmones, gyvūnus, kliūtis ir laiptus ir sukurti jūsų namų žemėlapi, kad galėtų juo naršyti. Robotikoje tai vadinama vienalaikiu lokalizavimu ir kartografavimu (SLAM), ir tai yra ilgalaikis šios srities iššūkis. „*Astro*“ išsprendžia SLAM, naudodama kelias kameras ir jutiklius, kad nustatytų aplinką ir paleistų juos per kelis neuroninius tinklus ir planavimo sistemas (Priest, 2022).

## **2.2. Sistemos vizija**

Anksčiau aptartos sistemos naudoja skirtingus įrankius, norint pasiekti tą patį tikslą. Tokioms sistemoms įgyvendinti reikia pasirinkti tinkamas technologijas visiems darbo uždaviniams.

Tarp visų aptartų sistemų neradau nei vienos, kurios inžinerinis sprendimas sutaptų su manuoju. Galima rasti ne vieną balso atpažinimo sistemą, kuri buvo kurta naudojant *Markov* modelius ir ne vieną autonominių robotą, kurio realizavimui buvo panaudoti integruoti kompiuteriai ir mikrokontrolerių deriniai.

Savo darbui aš pasirinkau dar niekur nerastą sprendimą – *Arduino* mikrokontrolerį ir jo integraciją su *Tensorflow* balso atpažinimo modeliu.

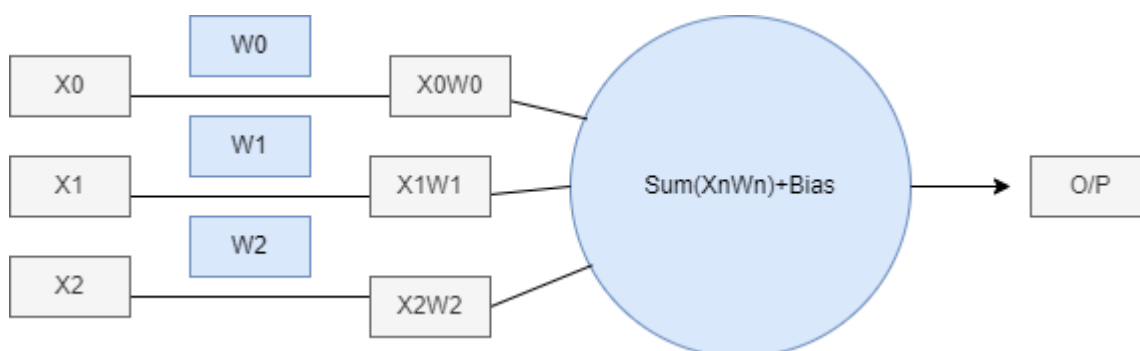
Nors šiame darbe bus bandoma įgyvendinti šią sistemą antžeminiam autonominiam robotui, bet ateityje tokią architektūrą galima pritaikyti ir dronams.

### 3. Balso atpažinimo algoritmų analizė

#### 3.1. Neuroniniai tinklai

Neuroniniai tinklai atspindi žmogaus smegenų elgseną, leidžiančią kompiuterių programoms atpažinti modelius ir išspręsti įprastas AI, mašininio mokymosi ir gilaus mokymosi problemas. Neuroniniai tinklai plačiai naudojami objektų klasifikacijos, teksto generavimo, kompiuterinės regos (Angl. *Computer Vision*) ir balso atpažinimo uždaviniuose. Neuroniniai tinklai sukurti remiantis žmogaus neuronų veikimu, neuronai žmonių kūnuose aktyvuojami tam tikru aplinkybių ir priverčia atitinkamas kūno dalis atlikti reikiamus veiksmus. Panašiai veikia ir dirbtiniai neuronai – jų pagrindas yra sluoksniai (Angl. *layers*), kurie yra įjungiami arba išjungiami aktyvavimo funkcijų (Chen, 2023).

Kiekvienas neuronas gauna padaugintą įvesties ir atsitiktinių svorių versiją, kuri vėliau pridedama su poslinkio reikšme (unikali kiekvienam neurono sluoksniui); tada tai perduodama atitinkamai aktyvinimo funkcijai, kuri nusprendžia galutinę neurono vertę (Chen, 2023).



pav. 4 neuroninio tinklo veikimas

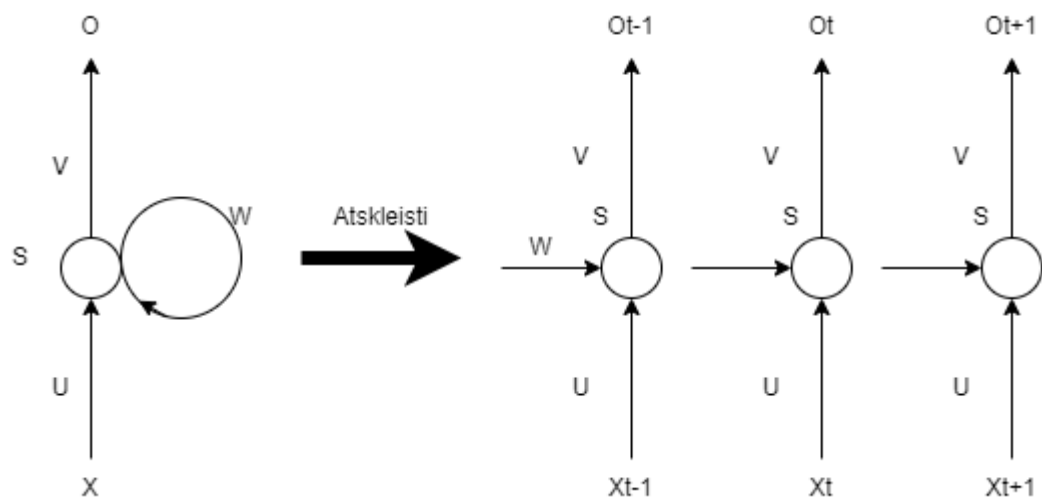
- $X_0, X_1$  ir t.t. – įvesties reikšmės paduodamos neuroniniam tinklui
- $W_0, W_1$  ir t.t – svoriai iš kurių dauginama kiekviena įvestis, yra skaitinės reikšmės, padaugintos iš įvesties. Dauginant atgal, jie modifikuojami siekiant sumažinti nuostolius.
- Aktyvavimo formulė yra matematinė funkcija, kuri įjungia arba išjungia neuroną

Neuroninis tinklas susideda iš daugelio sluoksnių, kurių kiekvienas yra sudarytas iš tarpusavyje susijusių neuronų

### 3.2. Rekurentiniai neuroniniai tinklai

FF-DNN yra nukreiptas tik į priekį, tačiau yra daugybę problemų, tokių kaip balso atpažinimas, kuriom reikia grįžtamojo ryšio (Angl. *loopback*). Neįmanoma suprasti sakinio prasmės be visų žodžių ir net juos turint reikia sudaryti seką, kad rasti teisingą reikšmę.

Rekurentiniai tinklai, skirtingai nei FF-DNN, įvestį ima ne tik dabartinę įvestį, bet ir anksčiau laike suvoktą būseną. RNN idėja yra naudoti nuoseklią informaciją. Teoriškai RNN gali panaudoti informaciją savavališkai ilgomis sekomis, tačiau praktiškai jos apsiriboja tik keliais žingsniais atgal (Brownlee, 2023).



**pav. 5** rekurentiniai neuroniniai tinklai

Kairė pusė reprezentuoja visą rekurentinį neuroninį tinklą, dešinė pusė parodo jo būsenas laike  $t-1$ ,  $t$  ir  $t+1$ . RNN (rekurencinis neuroninis tinklas) naudojasi ne tik dabartine, bet ir praeitimis būsenomis.

$x_t$  yra įvestis laiko žingsnyje  $t$ .  $s_t$  yra būsena laiko žingsnyje  $t$ .  $s_t$  yra ankstesnė būsena ir apskaičiuojama atsižvelgiant į ankstesnę paslėptą būseną kartu su dabartine įvestimi.  $s_t = f(Ux_t + Ws_{t-1})$ . Funkcija  $f$  dažniausiai yra  $\tanh$  arba  $\text{ReLU}$ .  $s_{-1}$ , kuris reikalingas pirmajai paslėptai būsenai apskaičiuoti ir paprastai inicijuojamas iki visų nulių.  $o_t$  yra  $t$  žingsnio išvestis. skirtingai nuo tradicinio giluminio neuroninio tinklo, kuris kiekviename sluoksnyje naudoja skirtingus parametrus, RNN turi tuos pačius parametrus –  $U$ ,  $V$  ir  $W$ ; per visus žingsnius (Brownlee, 2023).

Taigi, RNN naudoja tas pačias funkcijas kiekvieną kartą, tačiau su skirtingomis įvestimis.

Dėl savo galimybės dirbti su sekomis RNN yra dažniausiai naudojami neuroniniai tinklai, sprendžiant balso atpažinimo užduotis (Brownlee, 2023).

**Lentelė 1** RNN privalumai ir trūkumai

Privalumai	Trūkumai
Gebėjimas dirbti ir apdoroti sekas	Skaiciavimas gali būti labai lėtas.
Galimybė saugoti arba „įsiminti“ praeitą informaciją	Tinklas neatsižvelgia į būsimą įvestį priimant sprendimus.
Gebėjimas apdoroti įvairaus ilgio įvestį	Išnyksta gradiento problema, kai koeficientų atnaujinimui apskaičiuoti naudojami gradientai gali priartėti labai arti 0, tinklas nebegali išmokyti naujų duomenų. Kuo gilesnis tinklas, tuo ši problema ryškesnė.

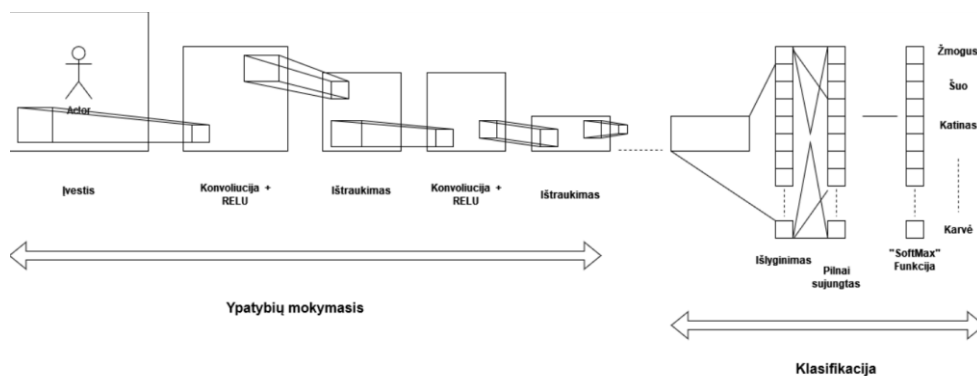
### 3.3. Konvolciniai neuroniniai tinklai

Konvoliucinis neuroninis tinklas (CNN arba *ConvNet*) yra gilaus mokymosi tinklo architektūra, kuri mokosi tiesiogiai iš duomenų, todėl nereikia rankiniu būdu išgauti funkcijų.

CNN ypač naudingi dirbant su vaizdų failais arba objektų lokalizacijoje. Jie taip pat gali būti gana veiksmingi klasifikuojant ne vaizdo duomenis, tokius kaip garsas ir signalo duomenys.

Konvoliuciniai neuroniniai tinklai išsiskiria iš kitų neuroninių tinklų puikiu našumu, naudojant vaizdo, kalbos ar garso signalų įvestis. Kaip ir kiti neuroniniai tinklai, CNN sudaro įvesties sluoksnis, išvesties sluoksnis ir daugybę paslėptų sluoksnių tarp jų (BenjaminWalter, 2023).

CNN architektūra susideda iš 3 pagrindinių dalių: konvolcinio sluoksnio, sujungimo sluoksnio ir sujungtų ryšių sluoksnio. Kadangi priklausomai nuo sprendžiamos problemos šie sluoksniai gali būti keičiami vietomis, jiems gali būti paduodama skirtinga įvestis, egzistuoja daugybę CNN architektūros variantų.



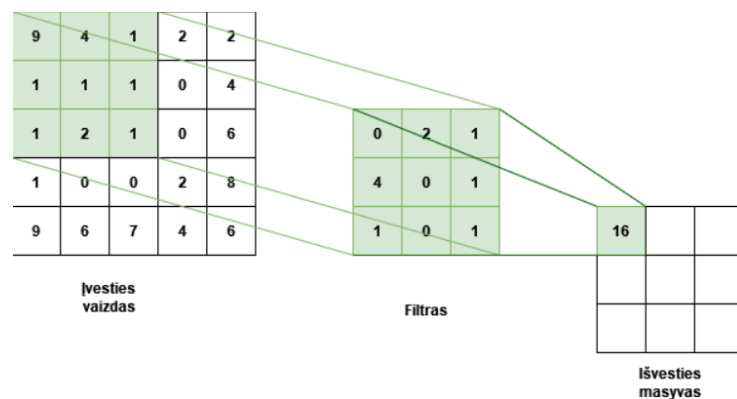
**pav. 6** CNN veikimas

- Konvolcinis sluoksnis.



Šis sluoksnis yra pagrindinė CNN dalis, kurioje vyksta dauguma skaičiavimų. Jam reikia įvesties, filtro ir ypatybių žemėlapių. Sprendžiant vaizdo klasifikavimo arba objektų atpažinimo uždavinius konvoliucinis sluoksnis kaip įvestį gauna vaizdo failą, kuriame kiekvienas pikselis sudarytas iš atitinkamo dimensijų kiekio (3D, 2D), kas atitinka RGB sistemą duotajame paveiksluke. Naudojant filtrą konvoliucinis sluoksnis tikrina įvestį ir ieško jame reikiamų ypatybių. Šis procesas vadinamas konvoliucija (BenjaminWalter, 2023).

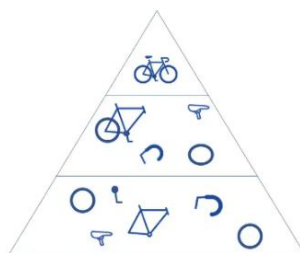
Įpatybių žemėlapis yra dvimatis (2-D) svorių masyvas, vaizduojantis dalį vaizdo. Tada filtras taikomas tam tikrai įvesties daliai ir tarp įvesties pikselių ir filtro apskaičiuojamas taškas. Tada šis taškinis produktas įvedamas į išvesties masyvą. Po to filtras pasislenka žingsniu, kartodamas procesą, kol branduolys nubrauks visą vaizdą. Galutinė taškinių produktų serijos iš įvesties ir filtro išvestis yra žinoma kaip funkcijų žemėlapis, aktyvinimo žemėlapis arba sudėtinė funkcija.



**pav. 7** CNN filtro veikimas

Kaip matoma paveiksluke, filtras neuždengia visos įvesties, tik šiuo metu tyriamą jos dalį. Kai filtras atlieka savo darbą ir randa arba neranda ieškomų savybių tiriamos įvesties dalyje, jo branduolys pasislenka ir procesas kartojasi (CNN | Introduction to Pooling Layer, 2023).

Verta paminėti, kad CNN gali turėti daugiau negu vieną konvoliucinį sluoksnį, tokiu atveju susidaro sluoksnių hierarchija, kai vis aukštesniame sluoksnyje gaunamas vis tikslesnis rezultatas.



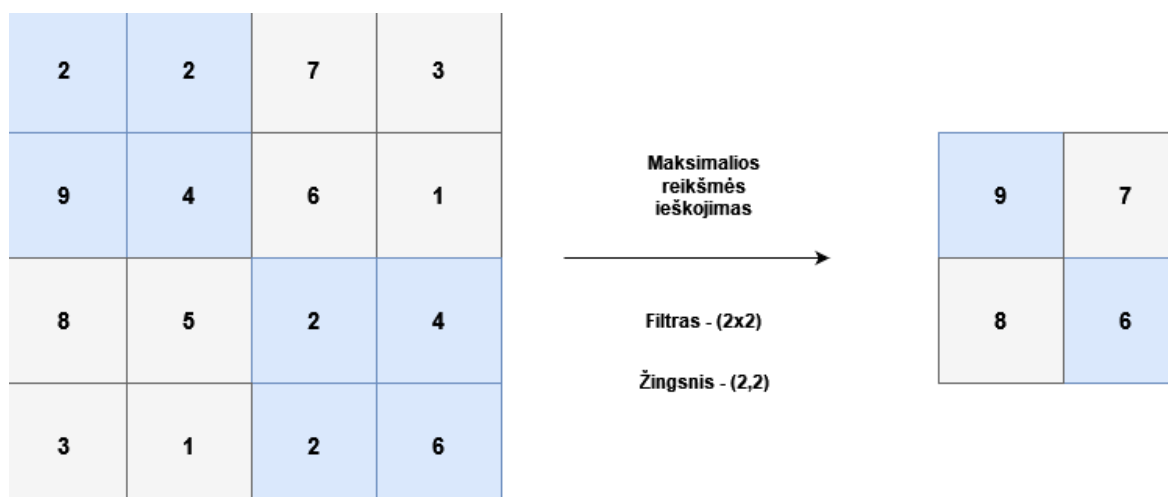
**pav. 8** konvoliucinių sluoksnių hierarchija

Aukščiau esantis paveikslėlis vaizduoja konvoliucinių sluoksnių hierarchijos pavyzdį, kai vis aukštesnis sluoksnis gauna vis tikslesnę dviračio reprezentaciją, naudodamasis praeitų sluoksnių duomenimis.

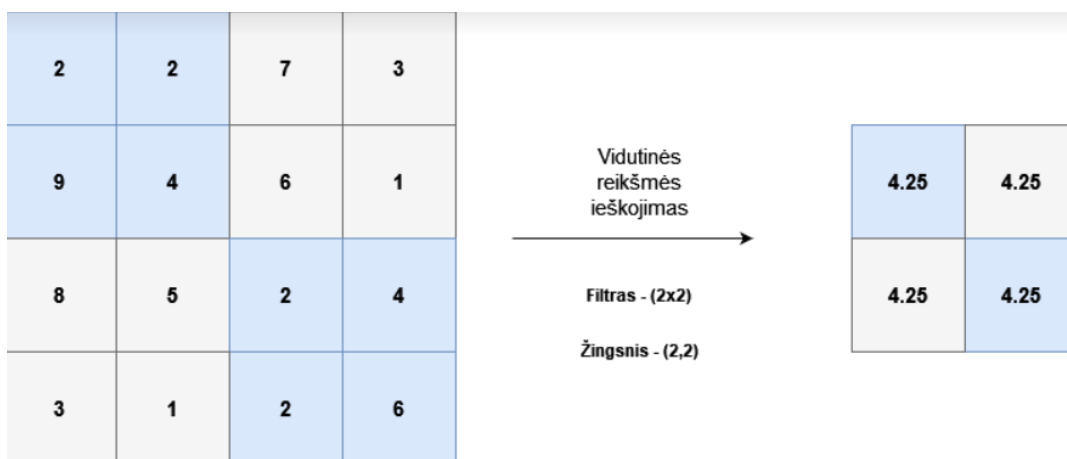
- Sujungimo sluoksnis

Šis sluoksnis atlieka dimensių sumažinimą, sumažindamas įvestyje esančių parametrų kiekį. Panašiai kaip ir konvoliucinis sluoksnis, sujungimo sluoksnis taip pat naudoja filtrą, tačiau šiuo atveju filtras yra tuščias. Filtras naudoja agregatoriaus funkciją tyriamajai įvesties daliai ir taip populiuoja išvesties masyvą.

Priklausomai nuo užduoties filtras tyriamojoje dalyje ieško maksimalios reikšmės arba skaičiuoja vidutinę tyriamos dalies reikšmę ir siunčia ją į išvesties masyvą. Tokiu būdu mes galime sumažinti įvesties sudėtingumą ir padidinti CNN našumą.



**pav. 9** maksimalios reikšmės filtravimas



**pav. 10** vidutinės reikšmės filtravimas

- Sujungtų ryšių sluoksnis

Šis sluoksnis atlieka klasifikavimo užduotį pagal savybes, išskirtas per ankstesnius sluoksnius ir skirtingus jų filtrus. Nors konvoliuciniai ir telkimo sluoksniai dažniausiai naudoja ReLu funkcijas, FC sluoksniai paprastai naudoja softmax aktyvinimo funkciją, kad tinkamai klasifikuotų įvestis, todėl tikimybė yra nuo 0 iki 1 (CNN | Introduction to Pooling Layer, 2023).

CNN gali būti naudojamas ir klasifikuojant garso failus. Tokiu atveju CNN kaip įvestį gautų garsinių failų spektrogramas ir atlikdamas klasifikaciją išmokyti atpažinti skirtingus žodžius.

**lentelė 2** CNN privalumai ir trūkumai

Privalumai	Trūkumai
Galimybė dirbti tiek su vaizdais tiek su sekomis	Lėtesnis veikimas
Didelis tikslumas	Komputacijos greitis smarkiai priklauso nuo kompiuterio įrangos
Didelis architektūrų pasirinkimas	Komputacijos greitis smarkiai priklauso nuo kompiuterio įrangos

(CNN | Introduction to Pooling Layer, 2023)

### 3.4 Išvados

Dėl savo našumo ir didesnio potencialo atliekant klasifikavimo užduotys savo darbui pasirinkau CNN arba konvoliucinį neuroninį tinklą. Kitaip negu gilieji neuroniniai tinklai (FF-DNN) ir rekurentiniai tinklai (RNN), konvoliucinis tinklas gali dirbti su sekų tipo duomenimis taip pat kaip ir vaizdais, todėl naudojant šį metodą atsiveria didesnės algoritmų pasirinkimų galimybės.

**lentelė 3** Neuroninių tinklų palyginimas

Modelis	Kriterijus	Įvestis	Našumas	Apmokymo sąnaudos	Parametrų dalijimasis	Tikslumas	Panaudojimo galimybės
<b>CNN</b>		Vaizdinė	+++	++	+	+++	+++
<b>RNN</b>		Sekų	++	+++	-	++	++

(+ funkcija yra; - funkcijos nėra; ++ funkcija atlieka savo paskirtį labai gerai; +++ funkcija atlieka savo paskirtį puikiai) (Chen, 2023)

### 3.5. Balso atpažinimo technologijos

- *Tensorflow*

*Tensorflow* yra atviro kodo mašininio mokymosi platforma, sukurta *Google*. *Tensorflow* turi platų bibliotekų, funkcijų ir išteklių pasirinkimą. Dėl savo plačių galimybių ir prieinamumo ši sistema plačiai naudojama tiek mašininio mokymosi entuziastų, tiek mokslininkų.

Šiuo metu garsiausia gilaus mokymosi biblioteka pasaulyje yra „*Google*“ *TensorFlow*. „*Google*“ visuose savo produktuose naudoja mašininį mokymąsi, kad pagerintų paieškos variklį, vertimą, vaizdų antraštes ar rekomendacijas. „*TensorFlow*“ yra „*Google Brain Team*“ sukurta biblioteka, skirta paspartinti mašinų mokymąsi ir giliųjų neuronų tinklų tyrimus. Ji buvo sukurta veikti keliuose procesoriuose arba GPU ir net mobiliosiose operacinėse sistemose, be to, jis turi keletą paketų kalbomis, pvz., *Python*, *C++* arba *Java* (Yegulalp, 2022).

Naudojant *Tensorflow* atsiranda galimybės ne tik sukurti savo ML (mašininio mokymosi) modelius, bet ir pertreniruoti jau esamus, taip sutaupant ir laiko ir resursų.

„*TensorFlow*“ programos galima paleisti bet kuriame patogiam objekte: vietiniame kompiuteryje, debesies klasteryje, „*iOS*“ ir „*Android*“ įrenginiuose, procesoriuose arba GPU. „*TensorFlow*“ sukurti modeliai gali būti naudojami bet kuriame įrenginyje, kur jie bus naudojami prognozėms atlikti (Yegulalp, 2022).

**lentelė 4** *Tensorflow* privalumai ir trūkumai

<b>Privalumai</b>	<b>Trūkumai</b>
Galimybė leisti sukurtus modelius skirtinguose įrenginiuose	Lėtas
Atviro kodo	Tensorflow palaiko tik NVIDIA GPU ir Python programavimo kalbą GPU programavimui.
Gera vizualizacija	
Suderinamas su skirtingomis kalbomis	

- *API.AI*

*API.AI* yra python kalbos biblioteka skirta automatiniam kalbos atpažinimui (ASR). Naudodami *API.AI* Python biblioteką, vartotojai gali skaityti ir rašyti programas, susijusias su natūralios kalbos apdorojimu, mašininiu mokymusi ir balso atpažinimu. *API.AI* leidžia naudoti balso komandas ir integruoti dialogo scenarijus, apibrėžtus tam tikram agentui *API.AI*. Autorius Dmitrijus Kuraginas, ištekliai apima veikimo pavyzdžius, diegimą ir dokumentaciją (apiai 1.2.3, 2017).

**lentelė 5** *API.AI* privalumai ir trūkumai

<b>Privalumai</b>	<b>Trūkumai</b>
Paprastas naudojimas	Diagnosticinių priemonių trūkumas
Leidžia tvarkyti konteksto parametrus naudojantis UI	Vizualizacijos trūkumas
Leidžia paprastai pridėti esybes	

- *Wit.ai*

Neseniai įsigijo *Facebook*. Vienas iš geresnių sprendimų su mašininiu mokymusi. Kaip ir *API.ai* atveju, *Wit.ai* gali būti naudojamas tais atvejais, kai platformos išmoksta naujų komandų, semantiškai panašių į tas, kurias kūrėjas įveda kaip įvestį. „*Wit.ai*“ taip pat teikia vartotojo sąsają, padedančią kūrėjams kurti objektus ir agentus. Pagrindinis skirtumas yra tas, kad *Wit.ai* taip pat teikia kūrėjo GUI, kuri apima vaizdinį pokalbių srautų vaizdą, verslo logikos iškvietimus, konteksto kintamuosius, šuoliais ir šakojimosi logiką (Kang, 2017).

**lentelė 6** *WIT.AI* privalumai ir trūkumai

<b>Privalumai</b>	<b>Trūkumai</b>
Lengva kurti aplikacijas naudojant UI	Nėra parametro funkcijos, todėl po kiekvienos sąveikos turite iškviešti verslo logiką
Paprastai matyti ir redaguoti pokalbių srauto medį	Visi dokumentuoti pavyzdžiai naudoja logiką tik interaktyviuoju režimu vietoje arba veikia tik su „Facebook Messenger“ programomis.
Kiekvienai esybei kuria roles	

## Išvados

Tarp aukščiau išvardintų balso atpažinimo technologijų buvo pasirinkta *Tensorflow*. Ši sistema jau turi apmokytus neuroninius tinklus, todėl norint atlikti ML uždavinį nereikia pradėti nuo nulio, galima naudotis jau sukurtais CNN modeliais. Tensorflow taip pat siūlo platų duomenų paketų rinkinį neuroninių tinklų apmokymui ir galimybę naudoti savo duomenis. Naudojantis šia Sistema galima pasirinkti vieną iš kelių palaikomų programavimo kalbų, todėl šios sistemos universalumas ir paprastumas puikiai tiks mano darbui.

**lentelė 7** balso atpažinimo technologijų palyginimas

Technologija	Kriterijus	Nemokamas	Kalbos apdorojimas	ML modelių kūrimas	Patogus GUI	Lengva naudotis	Aukštas našumas
Tensorflow		+	+++	+	+++	++	+++
Wit.ai		+	++	+	+	++	++
Api.ai		+	++	+	-	+	++

(+ funkcija yra; - funkcijos nėra; ++ funkcija atlieka savo paskirtį labai gerai; +++ funkcija atlieka savo paskirtį puikiai) (Kang, 2017)

## 4. Navigacinis robotas

### 4.1. Mikrokontroleriai

Mikrovaldiklis (Mikrokontroleris) yra mažas ir nebrangus mikrokompiuteris, skirtas atlikti specifines užduotis, tokias kaip mikrobangų informacijos rodymas, nuotolinių signalų priėmimas ir kt. Mikrovaldikliai randami transporto priemonėse, robotuose, biuro mašinose, medicinos prietaisuose, mobiliuosiuose radijo imtuvuose, pardavimo automatuose ir buitinėje technikoje. Iš esmės tai yra paprasti miniatiūriniai asmeniniai kompiuteriai, skirti valdyti mažas didesnio komponento funkcijas, be sudėtingos priekinės operacinės sistemos (OS) (Lutkevich, microcontroller (MCU), 2021).

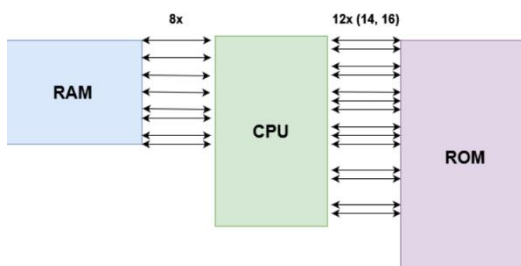
Bendrajį mikrovaldiklį sudaro procesorius, atmintis (RAM, ROM, EPROM), nuoseklieji prievada, periferiniai įrenginiai (laikmačiai, skaitikliai) ir kt.

**lentelė 8** Mikrovaldiklių komponentai

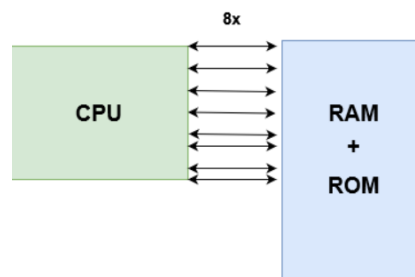
Komponentas	Paskirtis
<b>CPU</b>	Mikrovaldiklio smegenys. Atsakingas už įvesties ir išvesties, aritmetikos ir pagrindinių operacijų valdymą. Taip pat atlieka duomenų persiuntimus į kitas sistemas, didesnioje sistemoje.
<b>Atmintis</b>	Programos atmintis, kurioje saugoma ilgalaikė informacija apie procesoriaus vykdomas instrukcijas.
<b>Periferiniai įrenginiai</b>	Periferiniai įrenginiai. Įvesties ir išvesties įrenginiai yra procesoriaus sąsaja su išoriniu pasauliu. Įvesties prievada priima informaciją ir siunčia ją procesoriui dvejetainių duomenų pavidalu.

### Architektūra

Mikrovaldiklių architektūra gali būti pagrįsta Harvardo architektūra arba von Neumann architektūra, kurios abi siūlo skirtingus duomenų mainus tarp procesoriaus ir atminties



**pav. 11** Harvardo architektūra



**pav. 12** von-Neumanno architektūra

metodus. Naudojant Harvardo architektūrą, duomenų ir instrukcijų magistralės yra atskiros. Naudojant Von Neumann architektūrą, viena magistralė naudojama ir duomenims, ir instrukcijoms.

- Harvardo architektūra yra naujesnė koncepcija nei von-Neumanno.
- Harvardo architektūroje duomenų magistralė ir adresų magistralė yra atskiros. Taigi per centrinį procesorių galimas didesnis duomenų srautas ir, žinoma, didesnis darbo greitis.
- Harvardo architektūrai taip pat būdinga mažiau instrukcijų nei von-Neumanno (Shi, 2021).

### Operacinė Sistema

Mikrokontroleriai naudoja realaus laiko operacinę sistemą (RTOS). Tai prioritetinis prevencinis realaus laiko *kernel* branduolys mikroprocesoriams, parašytas daugiausia C programavimo kalba. Jis skirtas naudoti įterptosiose sistemose (Lutkevich, microcontroller (MCU), 2021).

„*MicroC/OS*“ leidžia apibrėžti keletą C funkcijų, kurių kiekviena gali būti vykdoma kaip nepriklausoma gija arba užduotis. Kiekviena užduotis vykdoma skirtingu prioritetu ir vykdoma taip, tarsi jai priklausytų centrinis procesorius (CPU). Žemesnio prioriteto užduotis bet kuriuo metu gali būti pertrauktos aukštesnio prioriteto užduotimis. Didesnio prioriteto užduotys naudoja operacinės sistemos (OS) paslaugas (pvz., delsą ar įvykį), kad būtų galima vykdyti žemesnio prioriteto užduotis (Lutkevich, microcontroller (MCU), 2021).

Mikrokontrolerių OS skiriasi nuo mum pažįstamų kompiuteriuose naudojamų operacinių sistemų. Vietoj to Mikrovaldikliai turi bibliotekų rinkinį, kuris leidžia jiems paprastai ir greitai atlikti veiksmus su jiems prijungta įranga (*Hardware*).

Mikrokontroleriai plačiai naudojami robotikos industrijoje. Jie leidžia užprogramuoti robotus ar jų dalis tam tikrai užduočiai atlikti.

## 4.2. Arduino

*Arduino* yra atvirojo kodo platforma, naudojama elektronikos projektams kurti. „*Arduino*“ sudaro tiek fizinę programuojama plokštė (mikrovaldiklis), tiek programinė įranga arba IDE (integruota kūrimo aplinka), kuri veikia jūsų kompiuteryje, naudojama kompiuterio kodui rašyti ir įkelti į fizinę plokštę (What is an Arduino?, 2022).

Arduino platforma tapo gana populiari tarp žmonių, kurie tik pradeda su elektronika, ir dėl geros priežasties. Skirtingai nuo daugelio ankstesnių programuojamų plokščių, „*Arduino*“ nereikia atskiros aparatinės įrangos, kad į plokštę būtų galima įkelti naują kodą – galite tiesiog naudoti USB kabelį. Be to, Arduino IDE naudoja supaprastintą C++ versiją, todėl lengviau išmokyti programuoti. Galiausiai „*Arduino*“ pateikia standartinę formos faktorių, kuris mikrovaldiklio funkcijas išskaido į labiau prieinamą paketą (What is an Arduino?, 2022).

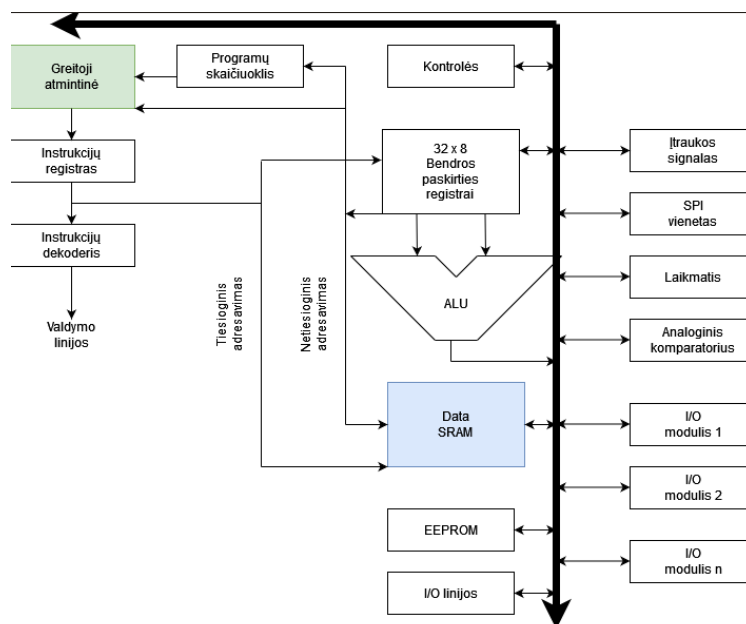
Nors yra daugybę skirtingų *Arduino* variacijų (aptartos žemiau), dauguma iš jų turi panašius komponentus.

Aparatūros paaiškinimą žiūrėkite. pirmajame priede.



## Architektūra

„Arduino“ procesorius naudoja Harvardo architektūrą, kurioje programos kodas ir



pav. 13 Arduino architektūra

programos duomenys saugomos atskirai. *Arduino* sudaro dvi atmintinės – programos ir duomenų atmintis. Kodas saugomas sparčiojoje (*Flash*) atmintyje, o duomenys – duomenų atmintyje (SRAM). Šiuos duomenis apjungia duomenų magistralė (Data bus 8-bit).

Svarbiausias „*Arduino*“ pranašumas yra tas, kad programas galima tiesiogiai įkelti į įrenginį. Tai įmanoma dėl 0,5 KB *Bootloader*, kuris leidžia įrašyti programą į grandinę. Viskas ką turi padaryti naudotojas, tai atsisiųsti „*Arduino*“ programinę įrangą ir parašyti kodą.

Arduino IDE ir 5KB *Bootloader* atstoja *Arduino* operacinę sistemą (KishanKondaveeti, 2021).

## Arduino rūšys

- *Arduino UNO R3*

Arduino Uno yra atvirojo kodo mikrovaldiklio plokštė, pagrįsta Microchip ATmega328P mikrovaldikliu ir sukurta Arduino.cc. Išleista 2010 m. Be visų ankstesnės plokštės funkcijų, Uno R3 naudoja ATmega16U2, o ne 8U2, rastą ankstesnėse kartose. Tai leidžia padidinti perdavimo spartą ir suteikia daugiau atminties. „Uno R3“ taip pat prideda SDA ir SCL kaiščius šalia AREF. Be to, šalia RESET kaiščio yra du nauji kaiščiai. Vienas iš jų yra IOREF, kuris leidžia ekranams prisitaikyti prie įtampos, tiekiamos iš plokštės (What is an Arduino?, 2022).



pav. 14 Arduino UNO R3

lentelė 9 Arduino Uno R3 privalumai ir trūkumai

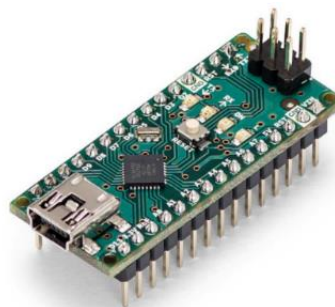
Privalumai	Trūkumai
Lengva programuoti	Kitų programavimo kalbų nepalaikymas (JAVA, PYTHON)
Didelis bibliotekų pasirinkimas	Pakankamai primityvi elektronika
Suderinama su praktiškai bet kokiais skydais ir sensoriais	Nėra OS
Lengva Bluetooth komunikacija	
Pakankamai pigi technologija	

(KishanKondaveeti, 2021)

- *Arduino Nano*

„Arduino Nano“ yra nedidelė „Arduino UNO“ versija. Jis turi daugiau ar mažiau „Arduino UNO“ funkcijų, tačiau yra nedidelės formos. Vieninteliai pagrindiniai skirtumai nuo UNO yra nuolatinės srovės maitinimo lizdo nebuvimas, Mini USB prievado naudojimas vietoj USB B prievado ir USB-TTL keitiklio lusto. Nano naudoja FT232, skirtą USB-UART tilto lustą iš FTDI, o ne ATmega16U2. Tai taip pat labai populiarus pasirinkimas tarp kūrėjų, kaip ir UNO dėl savo mažo dydžio ir pigios kainos

(What is an Arduino?, 2022).



pav. 15 Arduino Nano

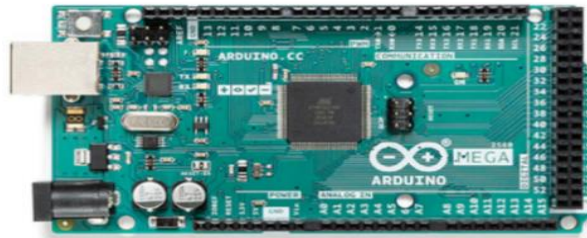
**lentelė 10** Arduino Nano privalumai ir trūkumai

<b>Privalumai</b>	<b>Trūkumai</b>
Kompaktiškas	Limituota atmintis
Lengvesnis prototipavimas	Mažiau įvesties išvesties kaiščių
Funkcionuoja taip pat kaip ir didesnės versijos	

- *Arduino Mega*

Priešingai negu *Arduino Uno*, Ši versija turi net 54 įvesties, išvesties kaiščius. Sugeba palaikyti iki 12V įtampą ir yra naudojama dirbant prie didelės elektroninės apimties projektų. Šiai plokštei reikalinga standartinė 7–12 V įvesties įtampa ir ji jungiama prie kompiuterio per tą patį B tipo USB laidą kaip ir Uno. Tai puikus pasirinkimas sudėtingiems projektams.

Jį maitina didesnis ir pajėgesnis procesorius ATmega2560. Labiausiai mėgstamas CNC ir 3D printerių bendruomenėje (What is an Arduino?, 2022).



**pav. 16** Arduino Mega

**lentelė 11** Arduino Mega privalumai ir trūkumai

<b>Privalumai</b>	<b>Trūkumai</b>
Daugiau atminties	Turi tik 8 bitų architektūrą
Daugiau kaiščių	Kodo vykdymo dažnis apsiriboja 20MHz
Turi saugiklius, apsaugančius nuo perkaitimo	Užėma daug vietos
Turi 2 įtampos reguliatorius	

**lentelė 12** Arduino modelių palyginimas

	<b>Arduino Uno</b>	<b>Arduino Mega</b>	<b>Arduino Nano</b>
Procesorius	Atmega328P	Atmega2560	Atmega32U4
Dažnis	16MHz	16MHz	16MHz
Greitoji atmintinė	32KB	256KB	32KB
EEPROM	1KB	4KB	1KB
SRAM	2KB	8KB	2,5KB
Palaikoma įtampa	5V	12V	5V
Kaiščių skaičius	14	54	20
Reikalaujamas USB tipas	standartinis	standartinis	mikro
Suderinamumas su skydais/sensoriais	Aukštas	Vidutinis	Vidutinis
Wifi/bluetooth	Naudojant skydus/sensorius	Naudojant skydus/sensorius	Ne

### *Skydai, moduliai ir sensoriai*

- Skydai

Skydai yra aparatinės įrangos dalys (*Hardware*), kurios yra užmaunamos ant jūsų Arduino tam, kad suteiktų jam papildomą funkcionalumą. Pavyzdžiui, galite naudoti skydą, kad būtų lengviau prijungti ir valdyti variklius arba net paversti *Arduino* į tokį sudėtingą dalyką kaip mobilusis telefonas (Nussey, 2022).

Skydai leidžia naudoti „*Arduino*“ daugiau nei vienam tikslui. Skydai gali būti naudojami po vieną, taip suteikiant grandinei tam tikrą funkciją arba užmaunami vienas ant kito, kuriant sudėtingesnę funkcionalumą. Kiekvienam skydui reikalinga GND jungtis, todėl naudojant kelis skydus reikėtų pagalvoti apie įtampos dydį ir galimų kaiščių kiekį. (Nussey, 2022)

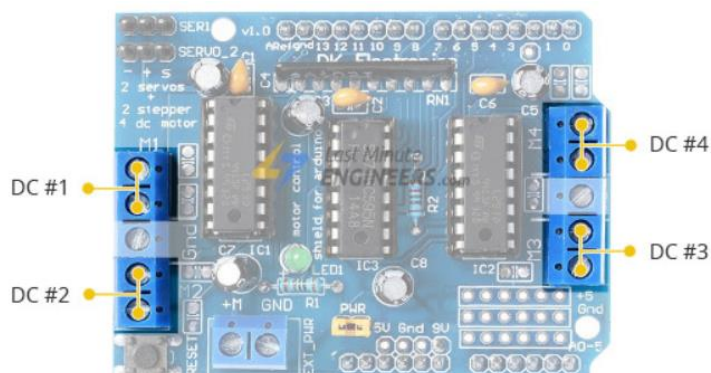
Vieni iš labiausiai naudojamų *Arduino* skydų yra L2932D motorų valdiklis, *Adafruit GPS*, *Relay shield* ir *Arducam mini camera shield*.

Kadangi kuriamas robotas savo judėjimui naudos motorus, jų integravimui ir valdymui reikia išsirinkti tinkamą skydą. Tam pasirinkau *Adafruit L2932D* motorų valdiklį.

Skydo smegenys yra dvi L293D variklių tvarkyklės ir 74HC595 pamainų registras. L293D yra dviejų kanalų H tipo tiltų variklio tvarkyklė, galinti valdyti du nuolatinės srovės variklius arba vieną žingsninį variklį. Kadangi ekrane yra dvi tokios variklio tvarkyklės, jis gali valdyti iki keturių nuolatinės srovės variklių arba du žingsninius variklius. (Control DC, Stepper & Servo with L293D Motor Driver Shield & Arduino, n.d.)

74HC595 poslinkio registras išplečia keturis *Arduino* skaitmeninius kaiščius iki aštuonių L293D lustinių kryptų valdymo kaiščių.

Abiejų L2932D IC išvesties kanalai yra išvesti 5 kontaktų varžtų gnybtais, pažymėtais M1, M2, M3 ir M4. Iš viso prie šių gnybtų galima prijungti keturis nuolatinės srovės variklius, veikiančius 4,5 – 25 V įtampa.



**pav. 17** Skydo jungtys

- Sensoriai

Sensoriai apibrėžiami kaip mašina, modulis arba įrenginys, aptinkantis aplinkos pokyčius. Sensoriai perduoda šiuos pokyčius į elektroninius prietaisus signalo forma. Sensorius ir elektroniniai prietaisai visada veikia kartu.

Jis sudarytas iš vieno silicio kristalo. Šis metodas plačiai naudojamas puslaidininkių pramonėje. Jis pasižymi puikiu mechaniniu stabilumu, apdirbamumu ir kt (What are the Arduino sensors?, n.d.).

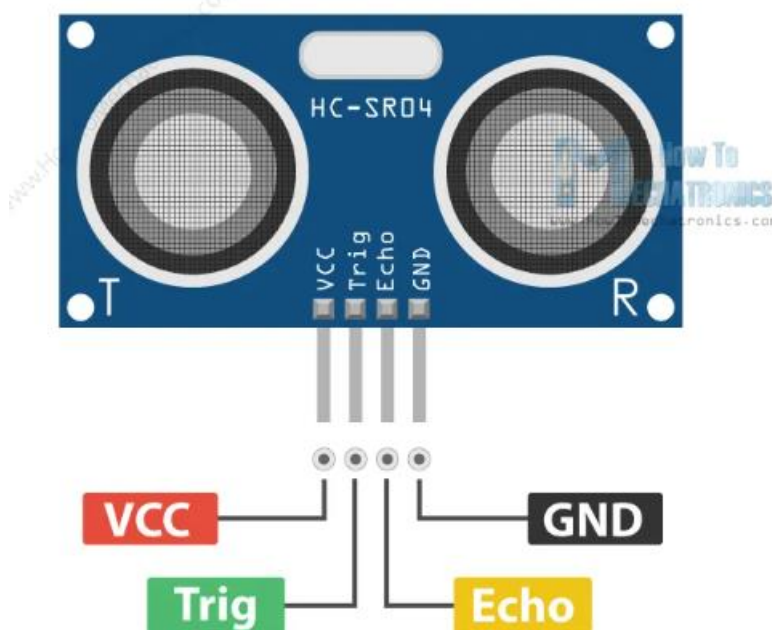
Integravus norimus sensorius su *Arduino*, šie gali perskaityti aplinkos pokyčius (pasikeitęs atstumas, šviesa ir t.t.) ir perduoti juos *Arduino* plokštei, kad ši priklausomai nuo gautos informacijos atliktų reikiamus veiksmus.

Dažniausiai naudojami sensoriai yra ultragarsinis sensorius, šviesos sensorius ir temperatūros sensorius.

Kadangi viena iš kuriamo roboto užduočių bus aptikti ir išvengti aplinkoje sutiktas kliūtis, savo darbui pasirinkau ultragarsinį sensorių. Naudojant šį jutiklį *Arduino* galės matuoti atstumą iki artimiausios kliūtis ir priklausomai nuo to priimti sprendimą.

HC-SR04 yra prieinamas ir lengvai naudojamas atstumo matavimo jutiklis, kurio diapazonas yra nuo 2 cm iki 400 cm. Jutiklis sudarytas iš dviejų ultragarsinių keitiklių. Vienas

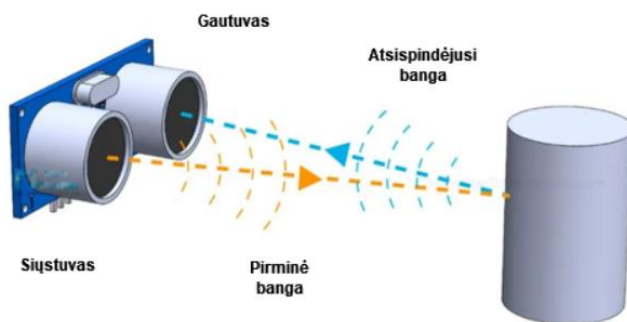
yra siųstuvas, kuris išveda ultragarso impulsus, o kitas yra imtuvas, kuris klauso atsispindėjusių bangų. Ši architektūra yra naudojama povandeniniuose sonaruose.



**pav. 18** HC-SR04 ultragarso sensorius

Sensorius turi 4 kontaktus. VCC ir GND eina į 5V ir GND kaiščius ant *Arduino*, o Trig ir Echo eina į bet kurį skaitmeninį *Arduino* kaištį. Trig kaiščiu siunčiame ultragarso bangą iš siųstuvo, o su Echo kaiščiu klausomės atsispindėjusio signalo (Dejan, n.d.).

Ultragarsinis sensorius skleidžia 40 000 Hz ultragarsą, kuris sklinda oru ir, jei jo kelyje yra objektas ar kliūtis, jis atšoks atgal į modulį. Atsižvelgdami į kelionės laiką ir garso greitį, galite apskaičiuoti atstumą. Norėdami generuoti ultragarsą, Trig kaiščiui turime suteikti aukštą



**pav. 19** Sensoriaus veikimas

10  $\mu$ s dažnį. Tai išsiųs 8 ciklų ultragarso pliūpsnį, kuris sklis garso greičiu. Echo kaištis laukia ir klausosi atsispindėjusio signalo (Dejan, n.d.).

Kadangi garso banga keliauja tą patį atstumą du kartus, kiekvieną kartą kai gauname apskaičiuotą atstumą, turime jį padalinti iš dviejų.

- Moduliai

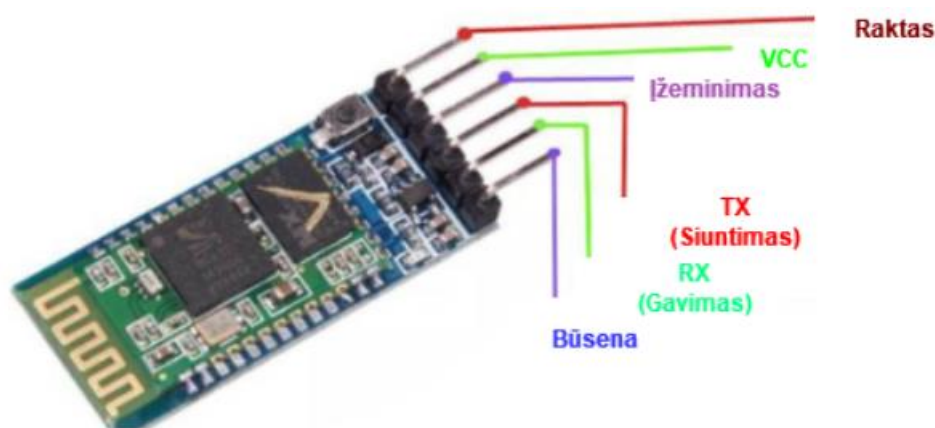
Modulis yra naudojimui paruošta plokštė su sensoriumi arba įvesties/išvesties įrenginiais savyje. Modulis sujungia visus reikalingus komponentus ir jungtis, kurių reikia

jutikliui (ar įrenginiui), kad veiktų, ir pristato jums paruoštą naudoti gaminį. Moduliai taip pat naudojami norint sukurti belaidžius ryšius tokius kaip *Wifi* ar *Bluetooth* (Teja, 2021).

Populiariausi *Arduino* moduliai yra mygtuko modulis, LED modulis, *Wifi* ir *Bluetooth* moduliai

Kadangi kuriamam robotui bus integruojama balso kontrolė, reikia rasti būdą, *Arduino* priimti komandas iš išorės. Tam pasirinkau HC-05 *Bluetooth* ryšio modulį.

HC-05 yra „*Bluetooth*“ įrenginys, naudojamas belaidžiam ryšiui su „*Bluetooth*“ palaikančiais įrenginiais (pvz., išmaniuoju telefonu). Jis bendrauja su mikrovaldikliais naudodamas nuoseklųjį ryšį (USART) (HC-05 Bluetooth Module Interfacing with Arduino UNO, 2022).



**pav. 20** HC-05 *Bluetooth* modelis

1. Įjungimas (Angl. *Enable*) – šis kaištis naudojamas duomenų režimui arba AT komandų režimui nustatyti.
2. VCC –jungtis prie +5V maitinimo šaltinio.
3. Ižeminimas (Angl. *Ground*) – prijungtas prie maitinimo sistemos įžeminimo.
4. Tx (siųstuvas) – šis kaištis nuosekliai perduoda gautus duomenis.
5. Rx (imtuvas) – naudojamas duomenims nuosekliai transliuoti per „*Bluetooth*“.
6. Būsena (Angl. *state*) – naudojama patikrinti, ar „*Bluetooth*“ veikia tinkamai.

### 4.3. Raspberry Pi

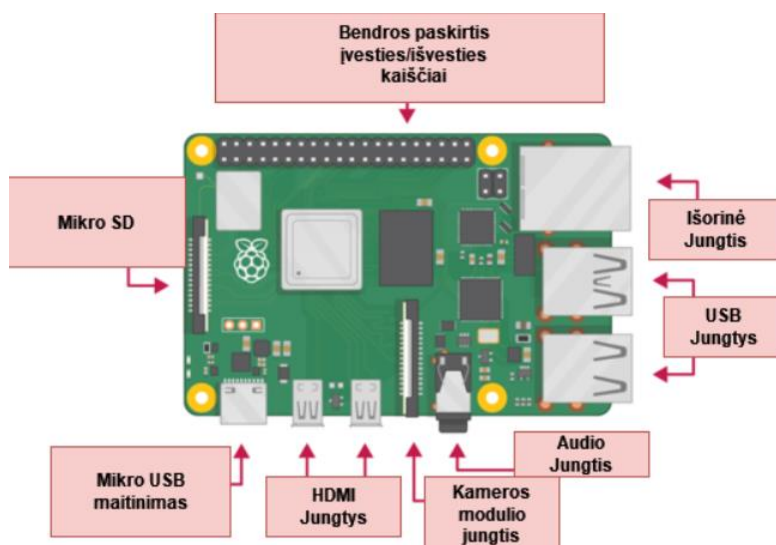
*Raspberry Pi* yra mikrokontrolerių serijos pavadinimas, kurį sukūrė JK labdaros organizacija *Raspberry Pi Foundation*, kurios tikslas – šviesti žmones kompiuterijos srityje ir sudaryti lengvesnę prieigą prie mokymosi kompiuteriu (What is a Raspberry Pi?, 2020).

*Raspberry Pi* buvo paleistas 2012 m., o nuo to laiko buvo išleista keletas iteracijų ir variantų. Originalus *Pi* turėjo vieno branduolio 700 MHz procesorių ir tik 256 MB RAM, o naujausiame modelyje keturių branduolių procesorius veikia daugiau nei 1,5 GHz, ir 4 GB RAM (What is a Raspberry Pi?, 2020).



„*Raspberry Pi*“ yra labai pigus kompiuteris, kuriame veikia „*Linux*“, tačiau jame taip pat yra GPIO (bendrosios paskirties įvesties/išvesties) kaiščių rinkinys, leidžiantis valdyti elektroninius fizinio skaičiavimo komponentus ir dirbti su daiktų internetu (IoT).

*Raspberry Pi* yra programuojamas įrenginys. Jame yra visos pagrindinės plokštės funkcijos vidutiniame kompiuteryje, tačiau be išorinių įrenginių ar vidinės atminties. Norint nustatyti *Raspberry* kompiuterį, į tam skirtą vietą reikės įdėti SD kortelę. SD kortelėje turi būti įdiegta operacinė sistema. *Raspberry* kompiuteriai yra suderinami su *Linux OS*. Tai sumažina reikalingos atminties kiekį ir sukuria įvairovę (What is a Raspberry Pi?, 2020).



pav. 21 Raspberry Pi aparatūra



**lentelė 13** Raspberry Pi komponentai

<b>Komponentas</b>	<b>Paskirtis</b>
USB prievadai	Jie naudojami pelei ir klaviatūrai prijungti.
Micro SD	Čia saugoma operacinės sistemos programinė įranga ir failai.
Eterneto prievadas	Naudojamas <i>Raspberry Pi</i> prijungti prie tinklo kabeliu. <i>Raspberry Pi</i> taip pat gali prisijungti prie tinklo per belaidį LAN.
Garso lizdas	Čia galite prijungti ausines arba garsiakalbius.
HDMI prievadas	Skirtas monitoriaus prijungimui, tokiu būdu galima gauti <i>Raspberry Pi</i> išvestį.
Micro USB	Čia prijungiate maitinimo šaltinį.
GPIO prievadai	Leidžia prie <i>Raspberry Pi</i> prijungti elektroninius komponentus, tokius kaip šviesos diodai ir mygtukai.

- *Architektūra*

„*Raspberry Pi*“ architektūra pagrįsta aparatinės ir programinės įrangos komponentų deriniu, kurie veikia kartu, kad pateiktų išsamų skaičiavimo sprendimą.

Aparatūros pusėje *Raspberry Pi* sudaro centrinis procesorius (CPU), atmintis ir įvairūs įvesties / išvesties (I / O) periferiniai įrenginiai. CPU yra *Raspberry Pi* „smegenys“, atsakingas už instrukcijų vykdymą ir skaičiavimų atlikimą. Atmintis, paprastai RAM pavidalu, naudojama duomenims ir instrukcijoms, kurias CPU turi greitai pasiekti, saugoti. Įvesties / išvesties įrenginiai leidžia *Raspberry Pi* sąveikauti su išoriniu pasauliu ir gali apimti tokius dalykus kaip USB prievadai, HDMI išvestis ir GPIO kaiščiai (Sivamurugesan, 2018).

„*Raspberry Pi*“ architektūros programinės įrangos komponentas paprastai yra pagrįstas „*Linux*“ operacine sistema, tokia kaip „*Raspbian*“. Tai suteikia patogią sąsają ir įrankių bei programų rinkinį, kurį galima naudoti norint bendrauti su *Raspberry Pi* ir atlikti įvairias užduotis. „*Linux*“ operacinė sistema taip pat valdo „*Raspberry Pi*“ aparatūros išteklius, įskaitant centrinį procesorių, atmintį ir I/O periferinius įrenginius, ir užtikrina, kad jie būtų naudojami efektyviai ir efektyviai (Sivamurugesan, 2018).

Apskritai *Raspberry Pi* architektūra yra aparatinės ir programinės įrangos derinys, suteikiantis išsamų skaičiavimo sprendimą mažame, prieinamame pakete. *Raspberry Pi OS* (anksčiau *Raspbian*) yra į *Unix* panaši operacinė sistema, pagrįsta *Debian Linux* distribucija, skirta *Raspberry Pi* kompiuterių šeimai. Sukurta ir platinama *Raspberry Pi Foundation* nuo 2013 metų (*Raspberry Pi OS*, n.d.).

*Raspberry Pi OS* turi darbalaukio aplinką *PIXEL*, pagrįstą *LXDE*, kuri atrodo panaši į daugelį įprastų stalinių kompiuterių, pvz., „*MacOS*“ ir „*Microsoft Windows*“. Darbalaukyje yra meniu juosta, spartieji klavišai, galimybė valdyti *bluetooth*, *wifi* ryšius (*Raspberry Pi OS*, n.d.).

Prie *raspberry Pi* taip pat galima prijungti klaviatūra su pele.

- *Raspbery Pi rūšys*
- *Pi 3 Model B*



**pav. 22** *Pi 3 model B*

Vienas iš populiariausių *Raspberry Pi* modelių yra *Raspberry Pi 3 Model B*. Šis modelis turi 1,2 GHz 64 bitų keturių branduolių ARM Cortex-A53 centrinį procesorių, 1 GB RAM ir integruotą Wi-Fi bei Bluetooth ryšį. Jame taip pat yra 40 GPIO kaiščių, kuriais galima prijungti įvairius jutiklius, variklius ir kitus įrenginius (Emet, 2022).

- *Raspberry Pi Zero W*



**pav. 23** *Raspberry Pi Zero W*

Kitas populiarus *Raspberry Pi* modelis yra *Raspberry Pi Zero W*. Šis modelis yra dar mažesnis ir pigesnis nei *Raspberry Pi 3*, bet vis tiek turi daug tų pačių funkcijų. Jame yra 1GHz vieno branduolio procesorius, 512 MB RAM ir integruotas „*Wi-Fi*“ bei „*Bluetooth*“. Jis taip pat turi 40 GPIO kaiščių, tačiau, palyginti su *Raspberry Pi 3*, jo galimybės yra šiek tiek ribotos (Emet, 2022).

- Raspberry Pi 4 Model B



**pav. 24** Raspberry Pi 4 model B

*Raspberry Pi 4 Model B* yra naujausias ir galingiausias *Raspberry Pi* modelis. Jame yra 1,5 GHz 64 bitų keturių branduolių procesorius, iki 8 GB RAM ir integruotas „Wi-Fi“ bei „Bluetooth“. Jis taip pat turi 40 GPIO kaiščių, bet taip pat prideda du USB 3.0 prievadus ir du mikro HDMI prievadus, kad palaikytų du ekranus (Emet, 2022).

Vienas pagrindinių šių *Raspberry Pi* modelių skirtumų yra jų apdorojimo galia ir atmintis. *Raspberry Pi 3* ir *Raspberry Pi 4* turi greitesnius procesorius ir daugiau RAM nei *Raspberry Pi Zero*, todėl jie geriau tinka sudėtingesnėms užduotims atlikti. „*Raspberry Pi 4*“ taip pat turi papildomų prijungimo galimybių, tokių kaip USB 3.0 ir dvigubo ekrano palaikymas, todėl jis yra universalesnis nei kiti modeliai.

**lentelė 14** Raspberry Pi modelių palyginimas

Modelis	Pliusai	Minusai
Raspberry Pi 3 Model B	Greitas procesorius ir daug RAM, integruotas Wi-Fi ir Bluetooth, 40 GPIO kontaktų	Brangesnis nei kiti modeliai
Raspberry Pi Zero W	Mažas ir prieinamas, integruotas Wi-Fi ir Bluetooth, 40 GPIO kaiščių	Ribota apdorojimo galia ir atmintis
Raspberry Pi 4 Model B	Greitas procesorius ir daug RAM, integruotas Wi-Fi ir Bluetooth, 40 GPIO kontaktų, USB 3.0 ir dviejų ekranų palaikymas	Brangesnis nei kiti modeliai

(Emet, 2022)

- *Papildoma įranga*

Yra daug įvairių dalykų, kuriuos galite prijungti prie *Raspberry Pi* vienos plokštės kompiuterio (What is a Raspberry Pi?, 2020).

- Jutikliai ir pavaros: *Raspberry Pi* turi GPIO (bendrosios paskirties įvesties/išvesties) kaiščių rinkinį, kurį galima naudoti įvairiems jutikliams ir pavaroms prijungti.

- Ekranai: *Raspberry Pi* galima prijungti prie ekrano, pvz., monitoriaus ar televizoriaus, naudojant HDMI išvestį. Tai leidžia *Raspberry Pi* rodyti vaizdus ir vaizdo įrašus ir gali būti naudinga projektams, tokiems kaip žiniasklaidos centrai ar namų automatikos sistemos.
- Klaviatūra ir pelė: *Raspberry Pi* galima prijungti prie klaviatūros ir pelės naudojant USB prievadus. Tai leidžia bendrauti su *Raspberry Pi* ir gali būti naudinga atliekant tokias užduotis kaip programavimas ar prieiga prie interneto.
- Tinklo įrenginiai: *Raspberry Pi* galima prijungti prie maršrutizatoriaus ar kitų tinklo įrenginių naudojant *Ethernet* prievadą arba integruotas *Wi-Fi* ir *Bluetooth* galimybes. Tai leidžia „*Raspberry Pi*“ pasiekti internetą ir susisiekti su kitais įrenginiais vietiniame tinkle.
- Išorinė atmintis: *Raspberry Pi* galima prijungti prie išorinės atminties, pvz., išorinio standžiojo disko arba USB atmintinės, naudojant USB prievadus.

#### 4.4. Išvados

Tiek *Raspberry Pi*, tiek *Arduino* turi savus privalumus ir savus trūkumus. Dėl *Arduino* suderinamumo su įvairios paskirties skydais ir moduliais ir naudojimosi paprastumo, savo projektui pasirinkau *Arduino Uno R3*. Šis modelis puikiai tinka kuriant vienos paskirties sistemas ir yra pakankamai lengvai programuojamas. Kadangi tai yra vienas iš naujesnių *Arduino*, galima būti užtikrintu, kad nekils rūpesčių ieškant aparatūros norimam funkcionalumui (Emet, 2022).

**lentelė 15** Arduino ir Raspberry Pi palyginimas

<b>Arduino</b>	
<b>Privalumai</b>	<b>Trūkumai</b>
Lengva naudoti	Maža atmintis
Pigu	Nėra operacinės sistemos
Paprastumas, Arduino yra pritaikyti vienos užduoties sistemoms	Ribotas kiekis kaiščių
Plačios panaudojamumo galimybės	
Platus skydų ir modulių pasirinkimas	
<b>Raspberry Pi</b>	
<b>Privalumai</b>	<b>Trūkumai</b>
Operacinė sistema	Sudėtingumas, Raspberry Pi yra labiau kompleksiškas įrenginys negu Arduino
Daug atminties	Labiau pritaikyti kuriant dideles, didelio funkcionalumo sistemas

**lentelė 16** Arduino ir Raspberry Pi panaudojamumas

<b>Panaudojamumas</b>	
<b>Arduino</b>	<b>Raspberry Pi</b>
Šviesoforuose	Namų apsaugos sistemos
Automobilių stovėjimo aikštelės skaitikliuose	Autonominės sistemos
Įterptinėse sistemose	Elektronikos projektai
Namų automatikoje	Retro žaidimų kūrimas
Pramoninėje automatikoje	Media serveriai
Medicinoje	
Robotikoje	

## 4.5. Kontroleris

Apmokius neuroninį tinklą ir sukonstravus robotą, iškyla užduotis apjungti šiuos darbus į vieną sistemą. Šiam tikslui pasirinkau sukurti mobiliąją programėlę, kuri *Bluetooth* jungties pagalba valdys navigacinį robotą.

- *Android Studio*

*Android studio* yra integruota programavimo aplinka (IDE) skirta išmaniųjų įrenginių programavimui. Ji paremta *Intelij Idea* programavimo aplinka ir naudoja *JAVA*, bei *Kotlin* programavimo kalbas (Android Studio, 2022).

Kad palaikytų programų kūrimą „*Android*“ operacinėje sistemoje, „*Android Studio*“ naudoja „*Gradle*“ pagrįstą kūrimo sistemą, emuliatorių, kodo šablonus ir „*Github*“ integraciją. Kiekvienas „*Android Studio*“ projektas turi vieną ar daugiau būdų su šaltinio kodu ir išteklių failais. Šie būdai apima „*Android*“ programų modulius, bibliotekos modulius ir „*Google App Engine*“ modulius (Android Studio, 2022).

*Android studio* naudoja „*Gradle*“. Tai yra pažangus kūrimo įrankių rinkinys, skirtas automatizuoti ir valdyti kūrimo procesą, leidžiant apibrėžti lanksčias, pasirinktines kūrimo konfigūracijas. Kiekviena kūrimo konfigūracija gali apibrėžti savo kodų ir išteklių rinkinį, pakartotinai naudojant dalis, bendras visoms jūsų programos versijoms. Naudojant „*Gradle*“ programuotojas gali atsiųsti reikiamus failus į projektą, jam būnant kompiliuojamu (What is Gradle?, 2022).

Tai atveria galimybę integruoti jau apmokytą neuroninį tinklą į kūriamą aplikaciją. *Android studio* priima *Tensorflow Lite* modelius.

- *Visual studio*

*Visual studio* yra integruota programavimo aplinka skirta įvairios programinės įrangos kūrimui. Naudojantis *Visual studio* galima sukurti tiek vaizduoklines, tiek išmaniųjų įrenginių

aplikacijas. *Visual studio* paliko kelias programavimo kalbas: *C#, Python, CPP* (incredibuild, 2020).

*Visual Studio IDE* (integruota kūrimo aplinka) yra programinė įranga, skirta kūrėjams rašyti ir redaguoti savo kodą. Jo vartotojo sąsaja naudojama programinės įrangos kūrimui, norint redaguoti, derinti ir kurti kodą. Integruoti įrankiai yra kodo profiliavimo priemonė, GUI programų kūrimo dizaineris, žiniatinklio dizaineris, klasės dizaineris ir duomenų bazės schemas kūrimo įrankis. Dėl aktyvios bendruomenės ir nuolatinio palaikymo iš *Microsoft*, *Visual studio* yra viena iš populiariausių IDE (incredibuild, 2020).

- *Xcode*

*Xcode* yra *Apple* integruota kūrimo aplinka (IDE), skirta visoms *Apple* platformoms, ir ji yra nemokama visiems *Apple* vartotojams. „*Xcode*“ teikia visus įrankius programoms kurti visoms „*Apple*“ platformoms: „*iOS*“, „*iPadOS*“, „*tvOS*“, „*watchOS*“ ir „*MacOS*“. Be to, *Xcode* palaiko daugelio populiarių programavimo kalbų, įskaitant *Swift*, *Objective-C*, *Objective-C++*, *C*, *C++*, *Java*, *Python* ir kt., šaltinio kodą (Ekren, 2022).

„*Xcode*“ yra vienintelis oficialus įrankis, skirtas programėlėms kurti ir publikuoti *Apple App Store*.

Naujoji *Xcode* versija (*Xcode* 14.0.1) sukurta taip, kad būtų lengvesnė ir greitesnė nei ankstesnės versijos ir leidžia kūrėjams kurti kelių platformų programas naudojant *Swift* ir *SwiftUI* (Ekren, 2022).

## Išvados

Savo darbui pasirinkau *Android studio*. Ši IDE turi aktyvią bendruomenę ir patogią vartotojo sąsają. *Android studio* lengva naudotis ir prireikus integruoti į ją *tensorflow lite* modelius.

lentelė 17 IDE palyginimas

IDE	Kriterijus	Palaikomų kalbų skaičius	Palaikomos platformos	Komandinė integracija	Patogi grafinė sąsaja	Testavimas	Bendruomenės aktyvumas
Android studio		2	1	++	+++	+++	+++
Visual studio		4	3	++	++	++	+++
Xcode		4	1	++	++	++	++

(+ funkcija yra; - funkcijos nėra; ++ funkcija atlieka savo paskirtį labai gerai; +++ funkcija atlieka savo paskirtį puikiai) (Ekren, 2022)

## 4.6. Programavimo kalba

- *Java*

„Java“ yra bendros paskirties, klasėmis pagrįsta, objektiškai orientuota programavimo kalba, skirta mažesnėms diegimo priklausomybėms. Tai kompiuterinė platforma taikomųjų programų kūrimui. Todėl „Java“ yra greita, saugi ir patikima. Ji plačiai naudojama kuriant programas nešiojamuosiuose kompiuteriuose, duomenų centruose, žaidimų pultuose, moksliniuose superkompiuteriuose, mobiliuosiuose telefonuose ir kt (Hartman, 2022). „Java“ savybės:

- Tai viena iš lengvai naudojamų programavimo kalbų, kurią galima išmokti.
- „Java“ yra nepriklausoma nuo platformos. Kai kurios programos, sukurtos vienoje mašinoje, gali būti vykdomos kitoje mašinoje
- Jis skirtas kurti objektiškai orientuotas programas.

- *Kotlin*

*Kotlin* yra bendros paskirties, nemokama, atvirojo kodo, statiškai įvedama „pragmatiška“ programavimo kalba, iš pradžių sukurta *JVM* („Java Virtual Machine“) ir „Android“, jungianti objektinio ir funkcinio programavimo funkcijas. Joje pagrindinis dėmesys skiriamas sąveikai, saugai, aiškumui ir įrankių palaikymui. Taip pat gaminamos „Kotlin“ versijos, skirtos „JavaScript ES5.1“ ir vietiniam kodui (naudojant LLVM), skirtiems daugeliui procesorių (Heller, 2022).

„Kotlin“ atsirado 2010 m. „JetBrains“, „IntelliJ IDEA“ kompanijoje, o nuo 2012 m. yra atvirojo kodo. *Kotlin* projektas „GitHub“ turi daugiau nei 770 bendradarbių; Nors dauguma komandos dirba „JetBrains“, *Kotlin* projekte dalyvavo beveik 100 išorinių bendradarbių. „JetBrains“ naudoja „Kotlin“ daugelyje savo produktų, įskaitant pavyzdinį „IntelliJ IDEA“ (Heller, 2022).

### *Išvados*

Kontrolierio kūrimui pasirinkau *Kotlin* programavimo kalbą. Ši kalba turi geresnį saugumo užtikrinimą ir ja sukurtos aplikacijos pasižymi didesniu našumu. Kadangi mano projekte svarbu greitai duomenų mainai ir apdorojimas tarp kontrolierio ir roboto, šis *Kotlin* bruožas yra kritinis.

**lentelė 18** programavimo kalbų palyginimas

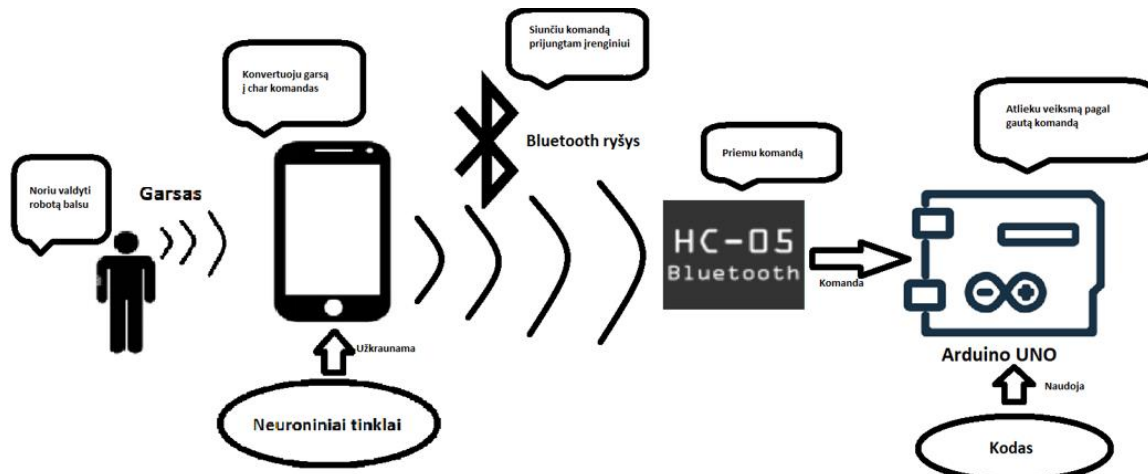
<b>Kal ba</b>	<b>Kriter ijus</b>	<b>Papildom o funkciona lumo pridėjima s</b>	<b>“nulin is” saugu mas</b>	<b>Lengv iau skaity ti</b>	<b>Bibliote kų pasirink imas</b>	<b>Bendruo menės aktyvuma s</b>	<b>našu mas</b>	<b>Kod o kok ybė</b>	<b>Saugu mas</b>
Java		++	-	+	++	+++	++	+	+
Kotlin		+++	+	++	++	++	+++	+++	+++

(+ funkcija yra; - funkcijos nėra; ++ funkcija atlieka savo paskirtį labai gerai; +++ funkcija atlieka savo paskirtį puikiai) (Heller, 2022)



## 5. Sistemos architektūra

### 5.1. Sistemos specifikacija



pav. 25 Sistemos vaizdusis paveikslėlis

Vaizdusis paveikslėlis atskleidžia planuojamos sistemos veikimą realioje aplinkoje. Vartotojas nori valdyti robotą balsu, kad padėtų jam išvengti kliučių. Vartotojas paduoda garsines komandas *android* programėlei, kuri naudodama neuroninius tinklus garsą konvertuoja į komandą. Programėlė turi *Bluetooth* ryšį, kurio pagalba siunčia komandas prijungtam įrenginiui. Robotas sugeba naviguoti erdvėje sensorių pagalba, tačiau taip pat naudoja hc-05 modulį *bluetooth* ryšiui. Šio modulio pagalba Arduino gauna komandas iš programėlės. Užkrautas kodas pasako robotui kokį veiksmą atlikti nuo gautos komandos.

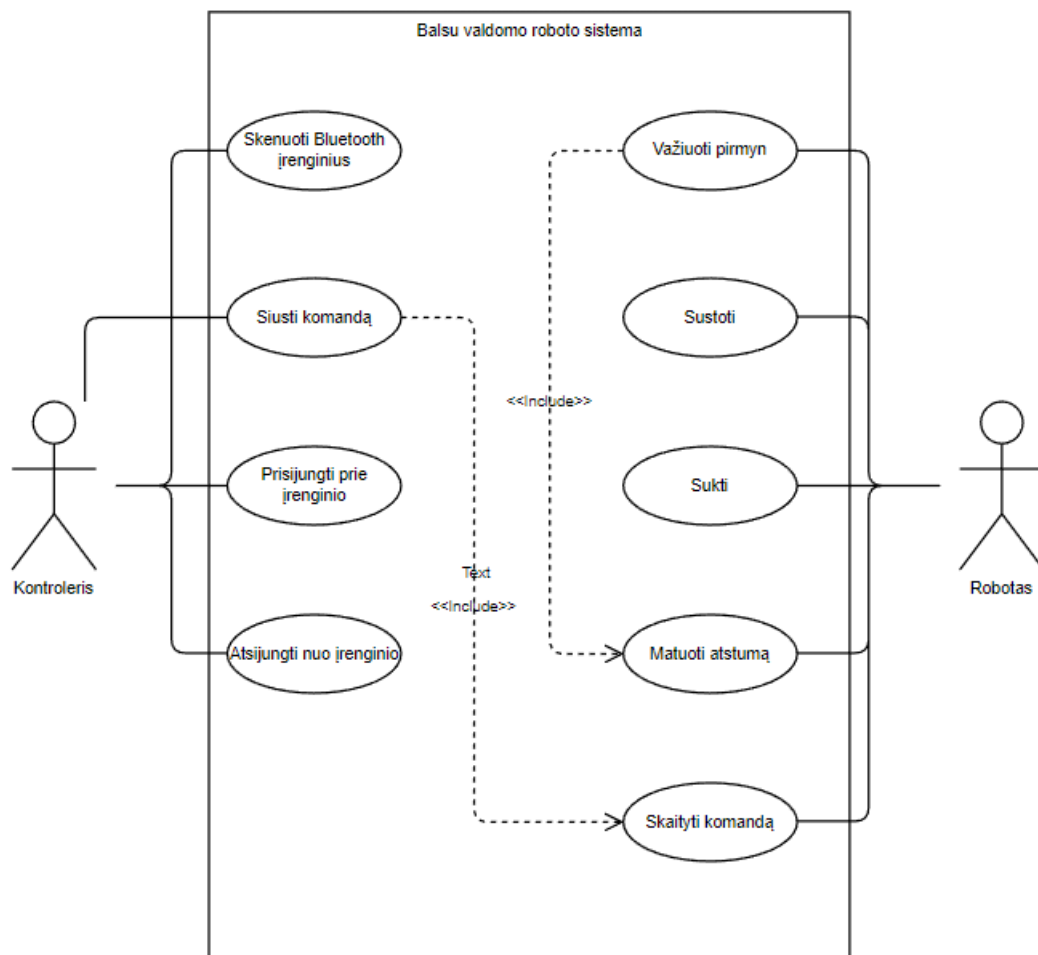
#### Funkciniai reikalavimai

- Sistema turi nuskaityti galimus Bluetooth įrenginius
- Sistema turi prisijungti prie įrenginio
- Sistema turi atsijungti nuo įrenginio
- Sistema turi gražinti prisijungimo ir atsijungimo patvirtinimus
- Sistema turi gražinti galimų įrenginių sąrašą
- Sistema turi priimti garsinę įvestį
- Sistema turi konvertuoti garsinę įvestį į baitų komandą
- Sistema turi siusti komandą Bluetooth ryšiu
- Sistema turi gražinti patvirtinimą apie sėkmingą arba nesėkmingą operaciją
- Robotas turi priimti prisijungimo prašymą
- Robotas turi turėti Bluetooth ryšį
- Robotas turi priimti komandą Bluetooth ryšiu
- Robotas turi atlikti komandą
- Robotas turi matuoti atstumą
- Robotas turi atlikti posūkius

- Robotas turi sustoti
- Robotas turi atsijungti nuo Bluetooth

#### **Nefunkciniai reikalavimai**

- Galimų įrenginių nuskaitymas turi užimti ne daugiau kaip 2 sekundes
- Atsijungimas ir prisijungimas turi būti saugus
- Garsiniu komandų atpažinimas turi vykti realiu laiku
- Komandų siuntimas turi vykti realiu laiku
- Robotas turi vykdyti komandas realiu laiku
- Komandos priėmimas ir apdorojimas negali užtrukti ilgiau negu 1 sekundę
- Robotas turi nuolat skaičiuoti atstumą



**pav. 26** Sistemos naudotojų diagrama

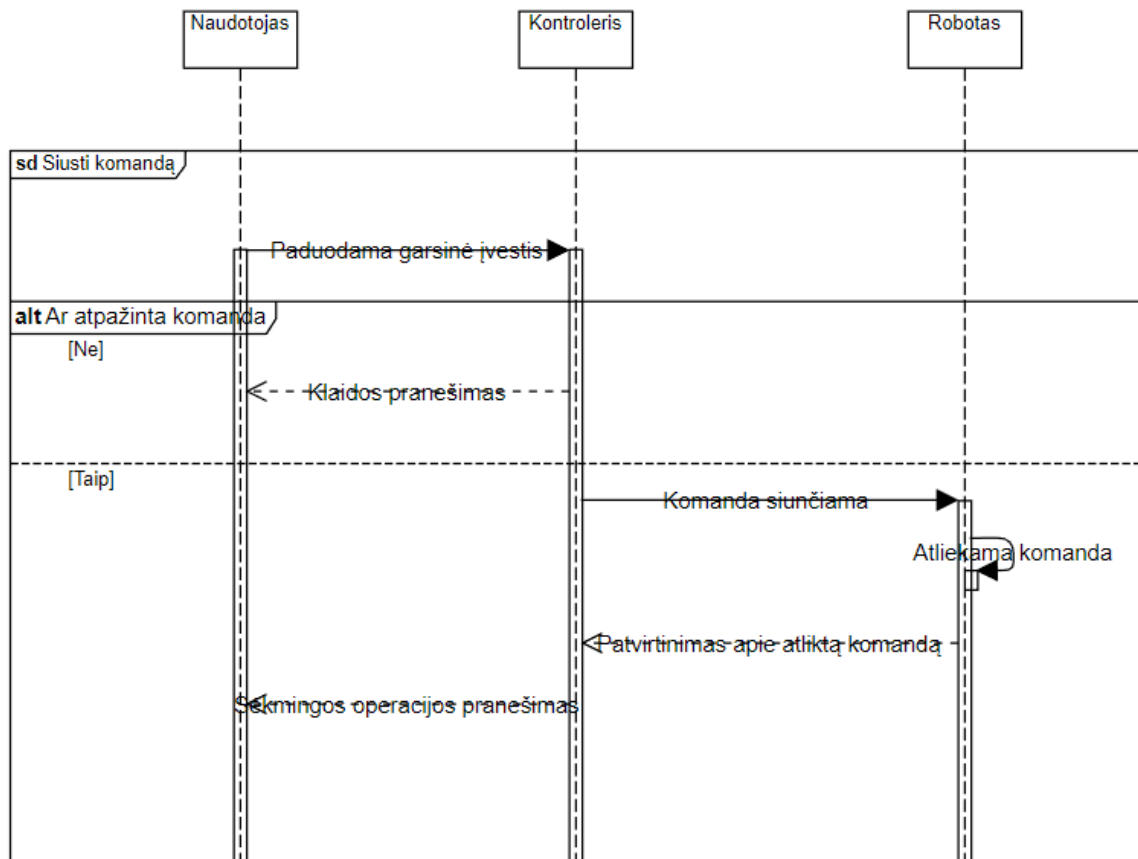
Naudotojo diagrama atskleidžia sistemos aktorius ir ryšį tarp jų. Balsu valdomo navigacinio roboto sistema turi dvi dalis – fizinį robotą, bei programėlę jo valdymui.

Kontrolierio funkcijos yra matomos vartotojui ir galimos pasirinkimui. Naudodamasis programėle vartotojas gali prisijungti arba atsijungti nuo įrenginio, skenuoti galimus įrenginius bei siusti komandą prijungtam įrenginiui.

Šiuo veiksmu atsiskleidžia programėlės ryšys su robotu. Robotas veikia tik gaves komandą iš kontrolierio, roboto galimi veiksmai apėma važiavimą pirmyn, sustojimą, atstumo matavimą ir posūkio atlikimą. Šie veiksmai priklauso nuo gautos komandos. Veiksmas “matuoti atstuma” yra iškviečiamas tik robotui važiuojant į priekį, norint naviguoti tarp kliučių.

**lentelė 19** Sistemos naudotojų istorijų paaiškinimai

<b>ID</b>	<b>Pavadinimas</b>	<b>Aprašymas</b>
<b>1.</b>	Prisijungti prie įrenginio	Kaip sistemos naudotojas, noriu prisijungti prie įrenginio, kad galėčiau siusti jam komandas.
<b>2.</b>	Atsijungti nuo įrenginio	Kaip sistemos naudotojas, noriu atsijungti nuo įrenginio, kad galėčiau pabaigti darbą.
<b>3.</b>	Skenuoti Bluetooth įrenginius	Kaip sistemos naudotojas, peržiūrėti galimus įrenginius, kad galėčiau pasirinkti norimą.
<b>4</b>	Siusti komandą	Kaip sistemos naudotojas, noriu siusti komandą, kad gavėjo sistema galėtų ją atlikti.
<b>5.</b>	Sustoti	Kaip sistemos naudotojas, noriu kad robotas sustotų, kad išvengtų galimos kliūties.
<b>6.</b>	Važiuoti pirmyn	Kaip sistemos naudotojas, noriu kad robotas važiuotų pirmyn, kad pasiektų siekiamą vietovę.
<b>7.</b>	Sukti	Kaip sistemos naudotojas, noriu, kad robotas atliktų posūkį, kad pakeistų judėjimo kryptį į kairę.
<b>8.</b>	Skaityti komandą	Kaip sistemos naudotojas, noriu, kad robotas skaitytų per Bluetooth gautą komandą, kad atliktų nurodomą veiksmą.
<b>9.</b>	Matuoti atstumą	Kaip sistemos naudotojas, noriu, kad robotas matuotų atstumą, kad žinoti kiek centimetrų yra iki kliūties.



**pav. 27** Užduoties “Siusti komandą” sekų diagrama

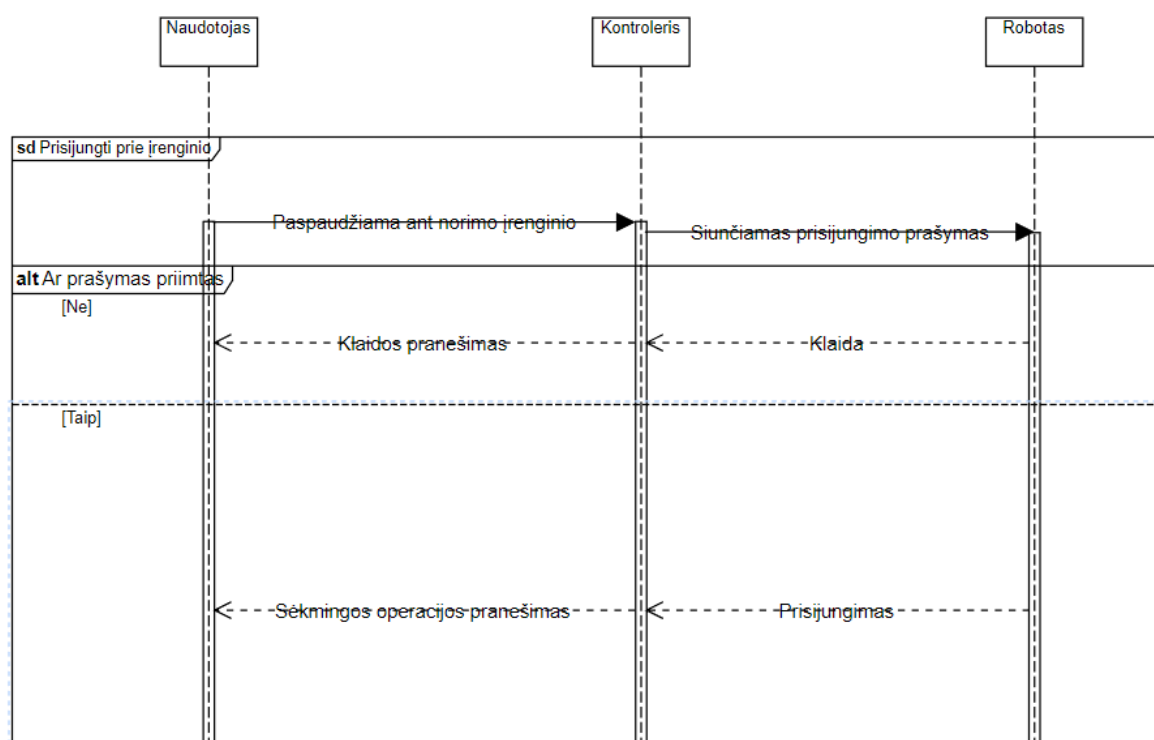
Komandos siuntimo veiksmo dalyvauja naudotojas, kontroleris ir robotas. Veiksmas yra inicijuojamas kai vartotojas paduoda sistemai garsinę įvestį. Kontroleris priklausomai nuo įvesties gali atpažinti joje tariamą komandą arba ne, pastaruoju atveju vartotojas gauna klaidos pranešimą.

Sėkmingai atpažinus įvestį kontroleris vkonvertuoja spėjimą į *char* tipo kintamąjį ir siunčia jį *Bluetooth* ryšiu robotui.

Gautai komandai robotas priskyria atitinkamą veiksmą ir jį atlieka. Atlikus veiksmą naudotojui gražinamas patvirtinimas apie sėkmingą operaciją.

**lentelė 20** Sekų diagramos paaiškinimas

užduoties numeris	1
užduoties pavadinimas	Siusti komandą
siekiamas tikslas	Perduoti garsinę komandą navigaciniam robotui
užduoties vykdymą inicijuojantis įvykis (trigeris)	Pasakoma komanda, esant prisijungus prie roboto
užduoties prioritetas	1-as (aukščiausias)
užduoties vykdymo sritis	Balsu valdomas navigacinis robotas
užduoties lygmuo	Sumarinis tikslas
pirminis aktorius	Naudotojas
antriniai aktoriai	Kontrolieris, robotas
"prieš" sąlygos	Sėkmingai prisijungta prie roboto
sėkmingos baigties "po" sąlygos	Robotas atlieka komandą
nesėkmingos baigties "po" sąlygos	Robotas neatlieka komandos arba atlieka ją neteisingai



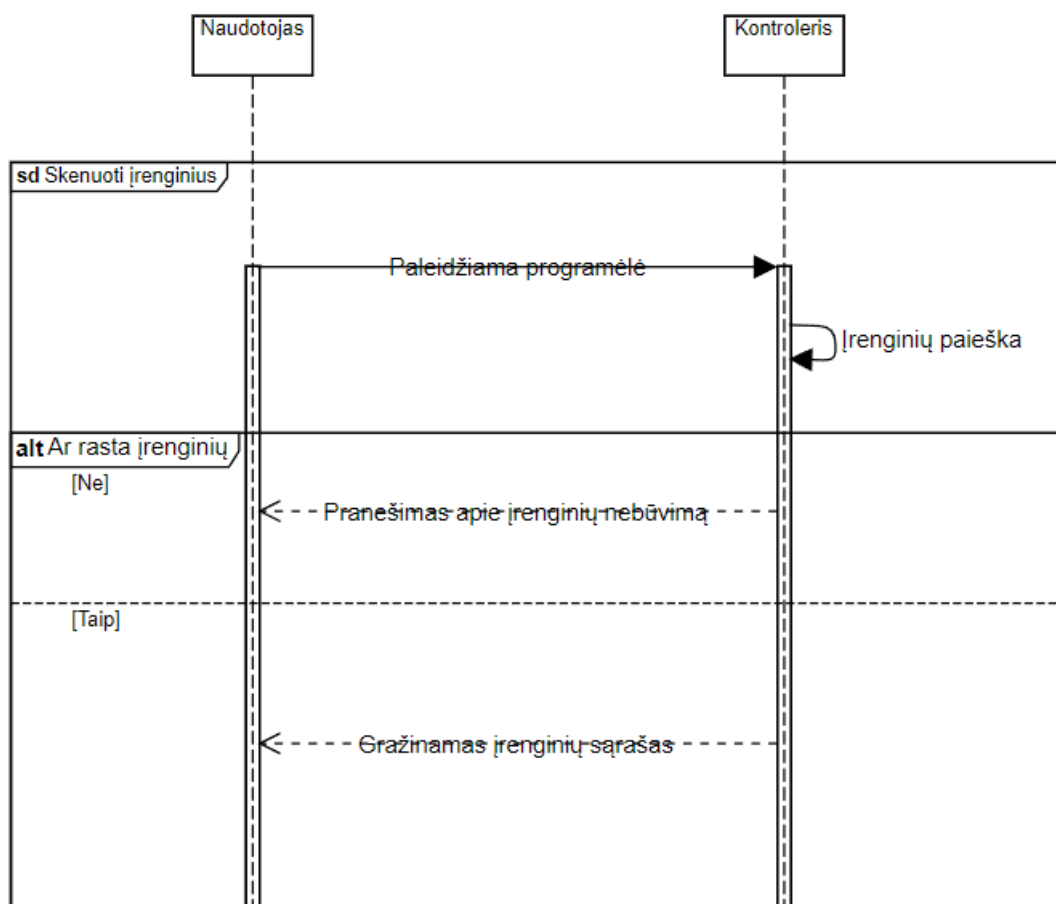
**pav. 28** Užduoties "Prisijungti prie įrenginio" sekų diagrama

Norint prisijungti prie įrenginio vartotojas turi programos pagrindiniame lange paspausti ant norimo įrenginio pavadinimo. Pasirinkus įrenginį programėlė inicijuoja *Bluetooth* prisijungimo prašymą norimam įrenginiui.

Priklausomai nuo to ar įrenginys priėmė ar atmetė prisijungimo prašymą, vartotojui grįžta patvirtinimas apie sėkmingą operaciją arba klaidos pranešimas.

**lentelė 21** Sekų diagramos paaiškinimas

užduoties numeris	2
užduoties pavadinimas	Prisijungti prie įrenginio
siekiamas tikslas	Prisijungti prie norimo įrenginio Bluetooth būdu
užduoties vykdymą inicijuojantis įvykis (triggeris)	Paspaudžiama iš norimo įrenginio iš galimų įrenginių sąrašo
užduoties prioritetas	1-as (aukščiausias)
užduoties vykdymo sritis	Balsu valdomas navigacinis robotas
užduoties lygmuo	Sumarinis tikslas
pirminis aktorius	Naudotojas
antriniai aktoriai	Kontrolieris, robotas
"prieš" sąlygos	Gautas sąrašas galimų įrenginių
sėkmingos baigties "po" sąlygos	Sėkmingai prisijungiama prie įrenginio
nesėkmingos baigties "po" sąlygos	Neįvyksta prisijungimas



**pav. 29** Užduoties "Skenuoti įrenginius" sekų diagrama

Bluetooth įrenginių skenavimas vyksta automatiškai kiekvieną kartą įjungus programėlę. Į pagrindiniame ekrane esantį sąrašą pirmiausiai yra įkeliami jau anksčiau suporuoti įrenginiai, tada atliekama naujų įrenginių paieška. Šis veiksmas gali būti kartojamas tiek kartų kiek nori vartotojas, jam paslinkus ekraną į viršų.

Priklausomai nuo to ar kontroleris randa aktyvių *Bluetooth* įrenginių ar ne vartotojui grįžta atitinkamas sąrašas.

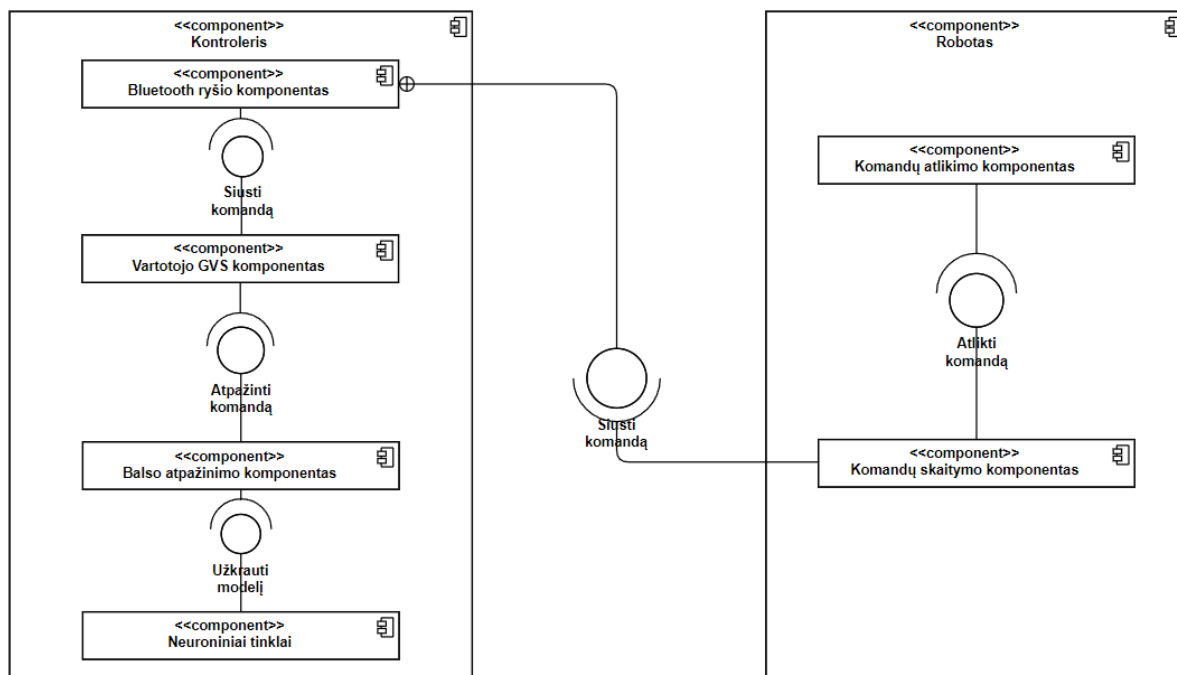
**lentelė 22** Sekų diagramos paaiškinimas

užduoties numeris	3
užduoties pavadinimas	Skenuoti Bluetooth įrenginius
siekiamas tikslas	Gauti galimų Bluetooth įrenginių sąrašas
užduoties vykdymą inicijuojantis įvykis (triggeris)	Ijungiamo programėlė
užduoties prioritetas	1-as (aukščiausias)
užduoties vykdymo sritis	Balsu valdomas navigacinis robotas
užduoties lygmuo	Sumarinis tikslas
pirminis aktorius	Naudotojas
antriniai aktoriai	Kontrolieris
"prieš" sąlygos	Paleidžiama programėlė
sėkmingos baigties "po" sąlygos	Gautas sąrašas Bluetooth įrenginių
nesėkmingos baigties "po" sąlygos	Nerasta jokių įrenginių

## 5.2. Architektūra

Sistema susideda iš keturių pagrindinių dalių: kontrolierio, roboto, balso atpažinimo sistemos ir Bluetooth ryšio komponento.

Kontrolieris naudoja balso atpažinimo sistemą su Tensorflow Lite modeliu, kad išgautų komandas iš jam paduotos garsinės įvesties. Vėliau tas pats kontrolieris perduoda šias komandas robotui naudodamasis Bluetooth ryšio komponentu.



**pav. 30** Sistemos komponentų diagrama

Toks architektūrinis sprendimas leidžia atskirti savarankiškus sistemos komponentus taip įgalinant juos ateities patobulinimams.



Kadangi neuroniniai tinklai yra atskiras sistemos komponentas, bet kada galima modelį pakeisti kitu. Modelis į kontrolerį yra užkraunamas Gradle pagalba ir gaunamas iš internetinės repozitorijos. Užkrautą modelį vėliau naudoja balso atpažinimo komponentas. Ši sistemos dalis yra sudaryta iš klasės, atsakingos už garsinės įvesties priėmimą ir jos konvertavimą į reikiamą formatą, bei klasių rinkinio atsakingo už darbą su *TFLite* modeliu. Šio komponento tikslas yra leisti vartotojui nuolat padavinėti garsinę įvestį ir gauti spėjimus.

Šie spėjimai yra atvaizduojami vartotojo grafinėje sąsajoje taip pat kaip ir pagrindinės sistemos funkcijos. Naudodamasis grafinės sąsajos komponentu vartotojas prisijungia prie norimo įrenginio ir paduoda jam komandas. Grafinė vartotojo sąsaja taip pat atsakinga už klaidų, bei sėkmingų operacijų pranešimų atvaizdavimą vartotojui.

Naudodamasis GVS naudotojas gali paduoti garsines komandas *Bluetooth* ryšiu.

Šis sistemos komponentas jungia kontrolerio bei roboto komponentus, jis atsakingas už prisijungimą ir komandų siuntimą. Jam paduodama komanda turi būti tinkamo format, kad galima būtų sėkmingai ją persiusti į *Arduino Bluetooth* modulį.

Roboto komponentas sudarytas iš dvejų mažesnių komponentų – komandų skaitymo ir komandų atlikimo.

Komandų skaitymo komponentas atsakingas už komandų iš kontrolerio gavimą ir atitinkamo veiksmo joms priskyrimą. Ši roboto dalis sudaryta iš funkcijos, skirtos kontroliuoti HC-05 *Bluetooth* ryšio modulį, gauti jam siunčiamus duomenis ir pagal juos nkviesti atitinkamą funkciją.

Komandų atlikimo komponentas dirba su likusia roboto aparatūros dalimi. Jis gauna funkciją iš komandų skaitymo komponento ir priklausomai nuo jos siunčia instrukcijas reikiams roboto aparatūros dalims.

**lentelė 23** Reikalavimų matrica

Nr.	Aprašymas
<b>SR-01</b>	Sistema turi leisti prisijungti prie pasirinkto įrenginio.
<b>SR-02</b>	Sistema turi leisti atsijungti nuo įrenginio.
<b>SR-03</b>	Sistema turi suteikti galimybę nuskaityti galimus įrenginius.
<b>SR-04</b>	Sistema turi leisti paduoti garsinę įvestį.
<b>SR-05</b>	Sistema turi leisti peržiūrėti prijungto įrenginio pavadinimą.
<b>SR-06</b>	Sistema turi leisti siusti komandas prijungtam įrenginiui.
<b>SR-07</b>	Sistema turi leisti priimti komandas.
<b>SR-08</b>	Sistema turi leisti dekoduoti komandas.
<b>SR-09</b>	Sistema turi leisti atlikti komandas.
<b>SR-10</b>	Sistema turi leisti peržiūrėti komandų statusą.
<b>SR-11</b>	Sistema turi leisti peržiūrėti atsakymus iš prijungto įrenginio.

**lentelė 24** Reikalavimų matrica komponentams

<b>Nr.</b>	<b>Aprašymas</b>
BAK-01	Balso atpažinimo komponentas turi pakrauti neuroninius tinklus TF Lite modelio pavidalu.
BAK-02	Balso atpažinimo komponentas turi integruoti modelį į sistemą.
BAK-03	Balso atpažinimo komponentas turi priimti garsinę įvestį.
BAK-04	Balso atpažinimo komponentas turi išgauti įvestyje tariamą komandą.
BAK-05	Balso atpažinimo komponentas turi konvertuoti komandą į Char tipo kintamąjį.
GVS-01	Grafinė vartotojo sąsaja turi atvaizduoti galimų Bluetooth įrenginių sąrašą.
GVS-02	Grafinė vartotojo sąsaja turi leisti prisijungti prie norimo įrenginio.
GVS-03	Grafinė vartotojo sąsaja turi gražinti pavykusios operacijos arba klaidos pranešimą.
GVS-04	Grafinė vartotojo sąsaja turi leisti vartotojui atnaujinti galimų įrenginių sąrašą.
GVS-05	Grafinė vartotojo sąsaja turi gražinti vartotojo tariamos įvesties komandą.
GVS-06	Grafinė vartotojo sąsaja turi gražinti vartotojui siunčiamas komandas.
GVS-07	Grafinė vartotojo sąsaja turi gražinti vartotojui roboto atliktus veiksmus.
GVS-08	Grafinė vartotojo sąsaja turi leisti vartotojui peržiūrėti prijungto įrenginio informaciją.
GVS-09	Grafinė vartotojo sąsaja turi leisti atsijungti nuo įrenginio.
BRK-01	Bluetooth ryšio komponentas turi nuskaityti galimus įrenginius.
BRK-02	Bluetooth ryšio komponentas turi leisti prisijungti arba atsijungti nuo įrenginio.
BRK-03	Bluetooth ryšio komponentas turi leisti siusti duomenis.
KSK-01	Komandų skaitymo komponentas turi priimti siunčiamą komandą.
KSK-02	Komandų skaitymo komponentas turi konvertuoti gautą komandą į reikiamą tipą.
KSK-03	Komandų skaitymo komponentas turi perduoti komandą tolimesniam jos atlikimui.
KAK-01	Komandų atlikimo komponentas turi priimti reikiamo tipo komandą.
KAK-02	Komandų atlikimo komponentas turi atlikti komandoje nusakytą veiksmą.
NT-01	Neuroniniai tinklai turi priimti garsinę įvestį.
NT-02	Neuroniniai tinklai turi konvertuoti garsinę įvestį į spektrogramą.
NT-03	Neuroniniai tinklai turi atlikti klasifikaciją.
NT-04	Neuroniniai tinklai turi gražinti garsinės įvesties komandą String formatu

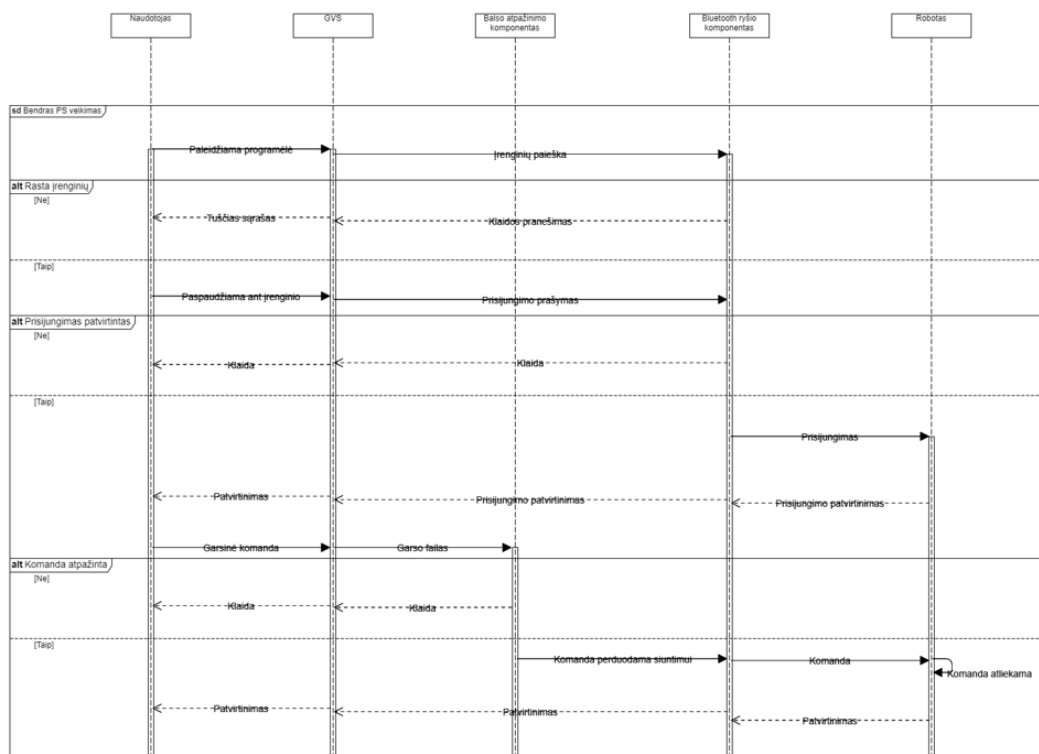
**lentelė 25** reikalavimų ryšio matrica

<b>Reikalavimas</b>	<b>Grafinė vartotojo sąsaja</b>	<b>Balso atpažinimo komponentas</b>	<b>Bluetooth ryšio komponentas</b>	<b>Komandų skaitymo komponentas</b>	<b>Komandų atlikimo komponentas</b>	<b>Neuroniniai tinklai</b>
<b>SR-01</b>	X		X			
<b>SR-02</b>	X		X			
<b>SR-03</b>	X		X			

<b>SR-04</b>	X	X				X
<b>SR-05</b>	X		X			
<b>SR-06</b>	X	X	X			
<b>SR-07</b>			X	X		
<b>SR-08</b>				X	X	
<b>SR-09</b>					X	
<b>SR-10</b>	X		X		X	
<b>SR-11</b>	X		X		X	
<b>BAK-01</b>		X				X
<b>BAK-02</b>	X	X				X
<b>BAK-03</b>	X	X				X
<b>BAK-04</b>	X	X				X
<b>BAK-05</b>	X	X				
<b>GVS-01</b>	X		X			
<b>GVS -02</b>	X		X			
<b>GVS -03</b>	X	X	X	X	X	
<b>GVS -04</b>	X		X			
<b>GVS -05</b>	X	X				X
<b>GVS -06</b>	X	X	X			X
<b>GVS -07</b>	X		X	X	X	
<b>GVS -08</b>	X		X			
<b>GVS-09</b>	X		X			
<b>BRK-01</b>	X		X			
<b>BRK-02</b>	X		X			
<b>BRK-03</b>	X	X	X			
<b>KSK-01</b>	X		X	X		
<b>KSK-02</b>				X		

<b>KSK-03</b>				X	X	
<b>KAK-01</b>				X	X	
<b>KAK-02</b>	X		X		X	
<b>NT-01</b>	X	X				X
<b>NT-02</b>						X
<b>NT-03</b>						X
<b>NT-04</b>	X	X				X

Pagal reikalavimų ryšio matricą galime pastebėti kokie komponentai reikalingi patenkinti specifinį reikalavimą. Pačių komponentų tikslingas veikimas taip pat priklauso ir nuo kitų sistemos dalių, tarkim teisingai nuspėti balso įvesčiai reikalingas ne tik balso atpažinimo komponentas, bet ir neuroniniai tinklai. Bluetooth ryšys taip pat reikalingas norint atpažintą komandą nusiusti prijungtam įrenginiui. Grafinė vartotojo sąsaja skirta vartotojo bendravimui su sistema. Pagal matricą galime pastebėti, kad reikalavimų įgyvendinimui dažniausiai figuruoja balso atpažinimo, bei Bluetooth ryšio komponentai.



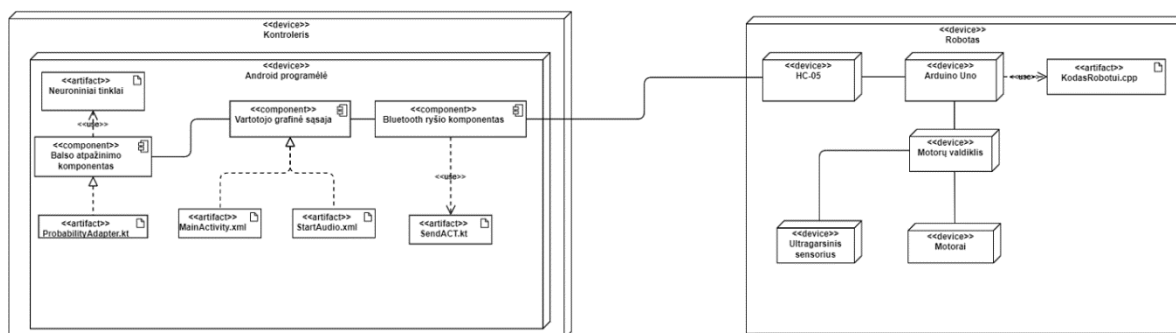
**pav. 31** Komponentų tarpusavio sąveikos sekų diagrama

Ši diagrama atskleidžia sistemos komponentų bendravimą tarpusavyje. Naudotojas inicijuoja sistemos veikimą paleisdamas programėlę. Tokiu būdu *Bluetooth* ryšio komponentas pradeda suporuotų bei aktyvių įrenginių paiešką ir atvaizduoja rezultatus grafinėje vartotojo sąsajoje. Ši leidžia vartotojui kartoti paiešką arba pasirinkti norimą įrenginį paspaudžiant ant jo iš sąrašo.

Paspaudžius ant norimo įrenginio *Bluetooth* ryšio komponentas siunčia pasirinktam įrenginiui prisijungimo prašymą, jeigu prašymas yra priimtas, grafinė vartotojo sąsaja gražina patvirtinimą ir atidaro naują langą.

Naujame lange vartotojas gali paduoti garsinę įvestį balso atpažinimo komponentui per grafinę vartotojo sąsają. Balso atpažinimo komponentas, naudodamas *TfLite* modelį gauna įvesties spėjimą ir jei komanda yra atpažinta konvertuoja jį į reikiamą formatą. Tada perduoda *Bluetooth* ryšio komponentui tolimesniam komandos siuntimui.

*Bluetooth* ryšio komponentas siunčia komandą prijungtam įrenginiui, kuriame jį gauna komandų skaitymo komponentas. Jis priklausomai nuo gautos komandos verčia ją į reikiamą formatą ir perduoda komandų atlikimo komponentui. Šis perdaves instrukcijas roboto aparatūrai gražina vartotojui patvirtinimą apie sėkmingą operaciją arba klaidą.

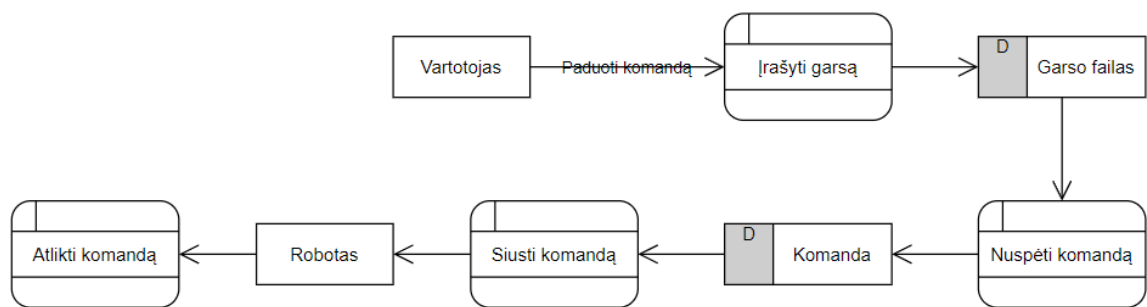


**pav. 32** Sistemos įgyvendinimo diagrama

PS įgyvendinimo diagrama rodo kokios technologijos veikia sistemos veikimo metu ir kokias funkcijas jos atlieka. Kaip ir Sistema, taip ir diagrama yra sudaryta iš dvejų dalių – kontrolierio ir roboto.

Kontrolierio architektūrinių komponentų veikimas yra realizuotas pasitelkus *TFLite*, XML ir *Socket* technologijas. Balso atpažinimo komponentas realiu laiku naudoja paduotą *TFLite* modelį ir atvaizduoja rezultatus vartotojo grafinėje sąsajoje. Čia vartotojas gali siusti komandas naudojantis *Bluetooth Socket* technologija.

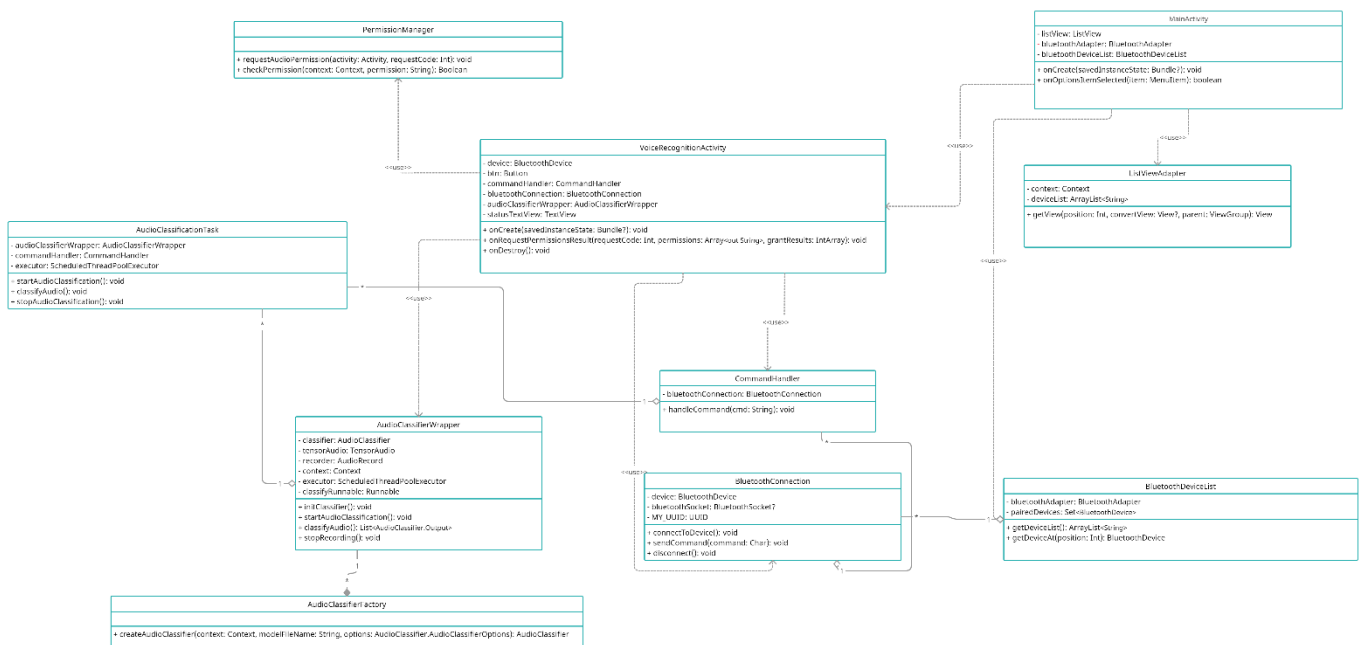
Robotas yra sudarytas už fizinių komponentų, bei programinio kodo. Kiekvienas fizinis komponentas yra atsakingas tik už vieną paskirtį, tarkim ultragarsinis sensorius atsako už atstumo matavimą. Šiuos komponentus valdo roboto “smegenys” – *Arduino UNO*. Instrukcijas *Arduino* gauna iš *Arduino IDE*, per USB kabelį.



**pav. 33** Sistemos duomenų srauto diagrama

Duomenų srauto diagrama vaizduoja komandos judėjimą tarp skirtingų sistemos komponentų. Komanda yra pagrindinis šios sistemos kintamasis nuo kurio priklauso sėkmingas roboto darbas. Vartotojas sukuria komandą garsiniu pavidalu (vaw formatu) ir sistema išsaugo ją kintamajame. Vėliau ši komanda perduodama funkcijai, kuri neuroninių tinklų pagalba nuspėja tariamą žodį. Prieš tai garsinis failas yra transformuojamas į reikiamą spektrogramos formatą. Ši komanda yra išsaugoma char tipo kintamajame ir siunčiama robotui. Šis ją pasigavęs bluetooth ryšiu atlieka komandą. Visi kontroleryje naudojami kintamieji yra globalūs.

### 5.3. PS struktūra



**pav. 34** Sistemos klasių diagrama

MainActivity yra pagrindinė veikla, kuri naudoja kitas klases norėdama surasti susijungusius Bluetooth įrenginius ir juos rodyti sąrašė. Ji naudoja:

- BluetoothDeviceList klasę, kad gautų sąrašą susijungusių Bluetooth įrenginių.

- ListViewAdapter klasę, kad šiuos įrenginius grafiškai atvaizduotų sąraše.
- VoiceRecognitionActivity klasę, kai vartotojas pasirenka vieną iš sąrašo elementų.

BluetoothDeviceList klasė yra atsakinga už susijungusių Bluetooth įrenginių sąrašo gavimą naudojant BluetoothAdapter klasę.

ListViewAdapter klasė naudojama kaip specializuota sąrašo tvarkyklė, kuri pateikia tinkamą vaizdą naudotojo sąsajai su Bluetooth įrenginiais.

VoiceRecognitionActivity yra veikla, kuri naudoja kitas klases norėdama atpažinti balso komandas ir perduoti jas per Bluetooth:

- BluetoothDevice klasė leidžia susieti su pasirinktu Bluetooth įrenginiu.
- CommandHandler klasė atsakinga už balso komandų apdorojimą ir jų perdavimą įrenginiui per BluetoothConnection.
- BluetoothConnection klasė atsakinga už ryšio su Bluetooth įrenginiu valdymą.
- AudioClassifierWrapper klasė naudojama balso klasifikavimo modelio valdymui ir klasifikavimo rezultatų gavimui.
- PermissionManager klasė naudojama norint gauti leidimus, reikalingus įrašyti garsą.

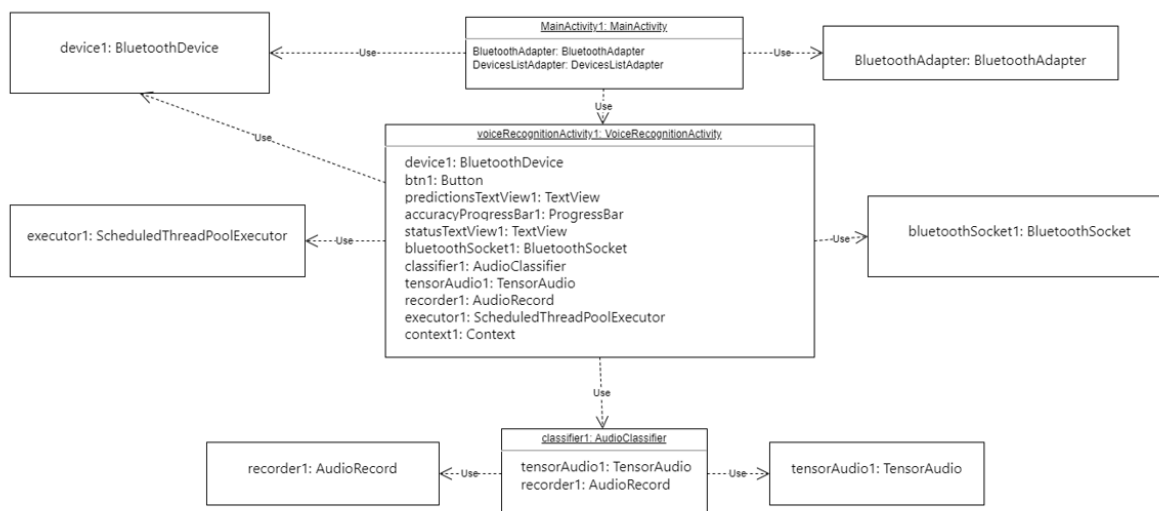
CommandHandler klasė naudoja BluetoothConnection klasę, kad perduotų komandas per Bluetooth. BluetoothConnection klasė valdo ryšį su Bluetooth įrenginiu ir atsakinga už komandų siuntimą. AudioClassifierWrapper klasė naudoja AudioClassifier klasę, kad gautų klasifikavimo rezultatus, ir AudioClassificationTask klasę, kad valdytų klasifikavimo užduotis.

AudioClassifierFactory klasė yra gamykla, kuri sukuria AudioClassifier objektą, reikalingą balso klasifikavimui. PermissionManager klasė yra pagalbinė klasė, skirta gauti reikiamus leidimus iš vartotojo.

AudioClassificationTask klasė atsakinga už klasifikavimo užduočių vykdymą, naudodama AudioClassifierWrapper ir CommandHandler klases. Tai padeda atpažinti balso komandas ir perduoti jas į Bluetooth įrenginį.

Ši sistema veikia taip: MainActivity parodo sąrašą susijungusių Bluetooth įrenginių naudodama BluetoothDeviceList ir ListViewAdapter klases. Kai vartotojas pasirenka įrenginį sąraše, pradedama VoiceRecognitionActivity veikla. Ji naudoja AudioClassifierWrapper klasę balso komandų atpažinimui ir CommandHandler klasę, kad perduotų atpažintas komandas per BluetoothConnection klasę į pasirinktą įrenginį. Taip pat PermissionManager klasė naudojama, norint gauti leidimus, reikalingus įrašyti garsą.

Visos šios klases bendrauja tarpusavyje ir padeda sukurti efektyvų balso atpažinimo ir Bluetooth įrenginio valdymo sistemą.



**pav. 35** Sistemos objektų diagrama

Sistemos objektų diagrama atvaizduoja sistemoje gyvujančius objektus bei jų ryšius fiksuotu sistemos veikimo momentu. Iš diagramos matome, kad MainActivity objektas priklausomas su VoiceRecognitionActivity objektu, pastarasis objektas yra sukuriamas sistemos, kai vartotojas pasirenka vieną iš jam atvaizduotų Bluetooth įrenginių. Šie įrenginiai buvo atvaizduoti MainActivity lange, per šio objekto atributus.

VoiceRecognitionActivity objektas skirtas atlikti balso atpažinimą, jo laukai savyje turi norimo įrenginio informaciją, bei kitus objektus, reikalingus balso atpažinimui. Objektas naudoja bluetoothSocket, kad atlikti prisijungimą prie norimo įrenginio. Tada yra sukuriamas AudioClassifier tipo objektas, kuris naudodamas TensorAudio ir AudioRecord atlieka nuolatinį balso įrašymą ir spėjimą.

Norint užtikrinti, kad garsas būtų įrašomas nuolat reikia pasirūpinti nauja gija. Kadangi pagrindiniai VoiceRecognitionActivity objekto laukai yra pasiekiami jo vidinėms funkcijoms, gali kilti “Data race”. Tam sustabdyti VoiceRecognitionActivity naudoja SceduledThreadPoolExecutor objektą, kuris pristabdo gijos darbą tam tikram laikui.

Ši diagrama reprezentuoja sistemos būseną komandų spėjimo ir siuntimo metu.

## 6. Sistemos įgyvendinimas

### 6.1. Neuroninių tinklų apmokymas

Balso atpažinimo užduočiai įgyvendinti buvo pasirinktas konvoliucinis neuroninis tinklas ir *Tensorflow* mašininio mokymosi biblioteka. Kuriama Sistema turės atpažinti keturias skirtingas komandas (pirmyn “go”, sustoti “stop”, kairėn “left”, dešinėn “right”), norint



ištreniruoti neuroninius tinklus šių žodžių klasifikacijai, reikia turėti atitinkamą duomenų rinkinį.

Šiam darbui buvo pasirinktas Google surinktas audio failų paketas “*mini-speech-commands-dataset*”. Jį sudaro aštuoni tūkstančiai garso įrašų suskirstytų į aštuonias kategorijas, kadangi kuriamai sistemai užtenka keturių aukščiau išvardintų kategorijų, like failai buvo pašalinti taip sutaupant resursų ir laiko neuroninių tinklų apmokymui.

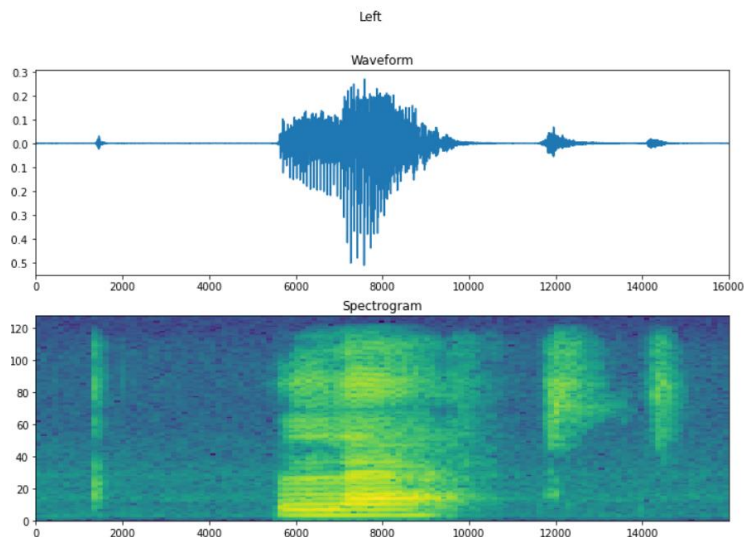
Naudodami *keras* bibliotekos funkciją `tf.keras.utils.get_file` užkrauname ir išskleidžiame duomenų paketą į *Goggle colab* aplinką. Gauta paketą išskaidome į du rinkinius – vieną skirtą apmokymui, kitą rezultatų validavimui santykiu 4:3.

Kadangi CNN gali atlikti klasifikaciją tik su vaizdinę įvestimi, reikia transformuoti audio failus į spektrogramas, tam sukuriame funkciją `def get_spectrogram(waveform)`. *Waveform* kintamasis yra iš audio failo gautas garso bangų atvaizdavimas dažnio atžvilgiu.

```
spectrogram = tf.signal.stft(  
    waveform, frame_length=255, frame_step=128)
```

Norint gauti spektrogramą naudojame STFT funkciją, kuri atlikdama *furje* transformaciją keičia *Waveform* kintamojo signalus ir prideda papildomą kanalų ašį, kad spektrogramą būtų galima naudoti kaip vaizdinį failą.

Naudodami *for* ciklą iteruojam pro kiekvieną audio failą duomenų pakete ir verčiam jį į spektrogramas, kurias išskirstom į rinkinius apmokymui, bei validacijai.



**pav. 36** Komandos “Left” spektrograma

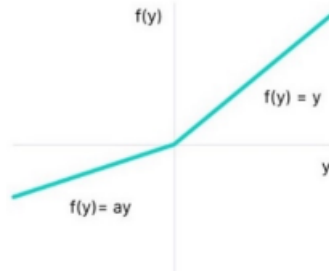
Turint duomenų rinkinius, telieka sukurti ir ištreniruoti modelį. Šiam darbui buvo pasirinktas CNN tinklas. Vienas iš pagrindinių Tensorflow privalumų yra galimybė naudoti jau paruoštus ML modelius. Šiam darbui naudojame `keras.Sequential` modelį, kuriam pridėsime papildomus išankstinio apdorojimo sluoksnius.

- `Resizing` sumažins įvesties imtį ir užtikrins greitą modelio treniravimą

- **Normalization** normalizuos kiekvieną vaizdo pikselį pagal jo standartinę nuokrypį ir vidurkį.

Prieš pridedant sluoksnius reikia išgauti įvesties parametrus, tokius kaip vidurkis ir standartinis nuokrypis, tam naudojam *Adam* funkciją.

Po išankstinio apdorojimo pridedame du konvolcinius sluoksnius su *Relu* aktyvavimo funkcijomis ir maksimalios reikšmės filtravimu. Šis derinys leidžia sumažinti konvolcinių sluoksnių kiekį neprarandant modelio tikslumo.



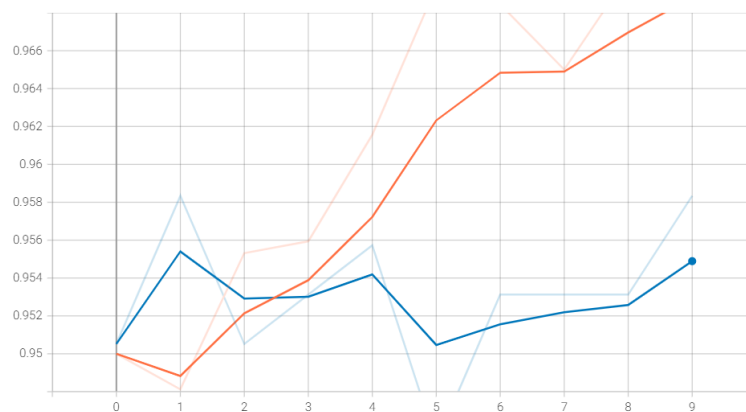
**pav. 37** Relu veikimas

```
layers.Conv2D(32, 3, activation='relu'),
layers.Conv2D(64, 3, activation='relu'),
layers.MaxPooling2D()
```

Norėdami apsidrausti nuo didelių skaičių ir sudėtingų skaičiavimų naudosime `layers.Dropout`, kuris sumažins įvesties reikšmes iki  $[0 - 1]$ , taip sumažindamas treniravimo apkrovą modeliui.

Deklaravus sluoksnius ir paruošus duomenų paketus, telieka atlikti modelio treniravimą. `model.compile`, funkcija kompiliuoja paruoštą modelį su paduotais optimizatoriais. Šiam darbui naudojamas kryžminės entropijos optimizatorius, kuris padidins modelio tikslumą.

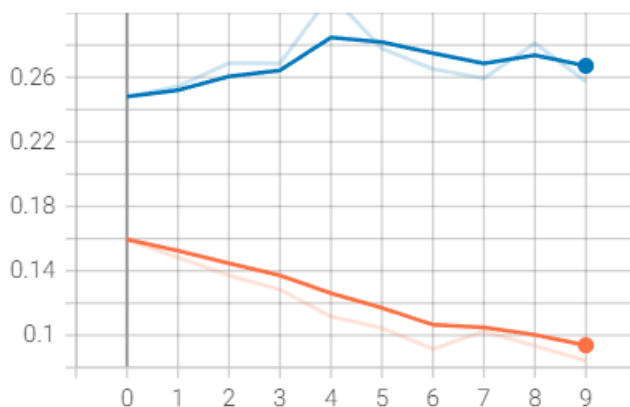
Modelio treniravimui naudojame `model.fit`, kuriam paduodame dešimt treniravimo epochu per kurias modelis pereina per visus įvesties failus ir jiems priskirtas etiketes. Modelio tikslumas didėja, didėjant epochų skaičiui.



**pav. 38** Treniravimo tikslumas ir praradimas

Diagrama vaizduoja modelio tikslumą, kiekvienos epochos metu. Melyna kreivė vaizduoja modelio tikslumą validacijos metu, oranžinė – treniravimo. Treniravimo kreivė atspindi modelio gebėjimą mokytis ir jo tikslumą treniravimo metu, tačiau tikrąjį modelio tikslumą atspindi validacijos kreivė. Validacijai parinkti duomenys skyriasi nuo treniravimo masyve buvusių, todėl taip patikriname ar modelis gali atpažinti duomenis, kurių nematė anksčiau.

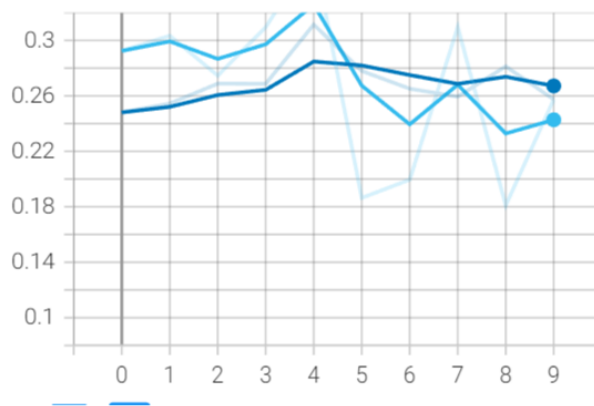
Kitas svarbus faktorius įtakojantis modelio našumą yra praradimo reikšmė. Tai skaičius kuris parodo kiek vertės modelis parranda su kiekviena epocha, taiga treniruojant modelį svarbu užtikrint tinkamą optimizavimą ir sumažinti modelio praradimą. Šiam modeliui buvo pasirinkta adam optimizacijos funkcija, kaip matoma grafike treniravimo epochoms einant į priekį praradimo vertė mažėja. To negalima pasakyti apie validacijos kreivę, matome, kad modelio validacijos pabaigoje praradimas yra didesnis negu pradžioje. Tai galėjo nutikti dėl per didelio mokymosi greičio arba per didelios duomenų apkrovos.



**pav. 39** Validacijos tikslumas ir praradimas

Norint sumažinti modelio praradimą buvo pakeistas optimizatorius ir mokymosi greitis.

Sukompiliavus modelį su kitu optimizatoriumi buvo gauti geresni rezultatai. Tamsiai



**pav. 40** Modelio praradimas po optimizacijos

mėlyna linija vaizduoja ankstesnį modelio praradimą, šviesiai mėlyna – praradimą pakeitus optimizavimo funkciją. Norint leisti ištreniruota modelį ant *android* programėles, reikia jį

konvertuoti į *Tensorflow Lite* modelį. Tokio tipo modeliai gali būti leidžiami tiek ant išmaniųjų įrenginių, tiek ant įterptųjų sistemų.

Modelio konvertavimui naudosime

`converter=tf.lite.TFLiteConverter.from_keras_model(model)` Funkciją.

Atlikus šiuos veiksmus gauname garso klasifikacijai atlikti skirtą *TFLite* modelį, kurį vėliau integruosime į *Android Studio*.

## 6.2. Navigacinis robotas

Paskutinė sistemos dalis – navigacinis robotas, kuris klausosi komandų per *Bluetooth* modulį ir atlieka atitinkamus veiksmus.

**lentelė 26** Roboto komponentai

Komponentas	Paskirtis
Arduino Uno	Roboto smegenys, pagrindinis, programuojamas komponentas atsakingas už kitų sistemos komponentų elgesį.
L293D motorų valdiklis	Skirtas prijungti motorus ir suteikti jiems energijos
<i>Bluetooth</i> modulis	Suteikia <i>Bluetooth</i> ryšio galimybę
Pavarų motorai ir ratai	Roboto judėjimui
Servo motoras	Skirtas sukti sensoriumi į reikiamą pusę
Ultragarso sensorius	Atstumo matavimas ir kliučių aptikimas
Li-On baterijos ir jungiklis	Suteikia pakankamai energijos elektronikai

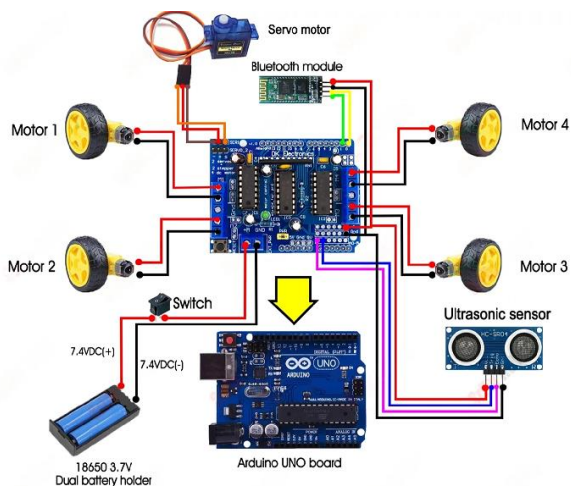
Kiekvienas robotas susideda iš dvejų pagrindinių dalių – aparatūros ir programos. Viršuje esančioje lentelėje surašytos elektronikos dalys fizinio roboto konstravimui ir jų paskirtys. Šiam darbui buvo pasirinktas Arduino Uno mikrokontroleris, kadangi robotas turės judėti aplinkoje ratų pagalba, ant arduino užmauname variklio valdymo skydą.

Skydas bus atsakingas už variklių ir servo motoro valdymą, taip pat kaip ir energijos padavimą jiems.

Pasitelkus lituoklį ir magistralinius laidus prie variklio valdymo skydo prijungiami motorai bei servas.

Atsižvelgus į poreikį kontroliuoti robotą balsu, bei priversti jį išvengti kliučių prijungiame ultragarso sensoriumi ir *Bluetooth* modulį. Kiekvieną iš jų reikia prijungti prie atitinkamų įžeminimo, energijos, siuntimo, bei gavimo kaiščių, kurių numerius vėliau identifikuosime programinėje dalyje.

Pritvirtinus elektroniką prie standžios plokštumos ir uždėjus ratus gauname išbaigtą roboto aparatūrinę dalį.



pav. 41 Roboto elektros grandinė



pav. 42 Navigacinis robotas

Sukonstravus robotą, reikia jį užprogramuoti. Arduino naudoja *Bootloader*, kuris leidžia greitai ir paprastai įkelti programos kodą į elektros grandinę. Tam tikslui įgyvendinti pasitelksime integruotą *Arduino* programavimo aplinką *Arduino IDE*.

Norint priversti *Arduino* bendrauti su aparatūros dalimis, reikia atsisiusti reikiamas bibliotekas.

```
#include <Servo.h>
#include <AFMotor.h>
#include <SoftwareSerial.h>
```

Šios bibliotekos atsako už variklio valdymo skydo, servo motoro, *Bluetooth* modulio ir ultragarsinio sensoriaus veikimą. Atsisiuntus bibliotekas ir deklaravus kaiščių numerius galima pradėti programos rašymą.

Arduino IDE turi dvi pagrindines funkcijas kodo rašymui: `void setup()` ir `void loop()`. Pirmoji funkcija yra vykdoma tik vieną kartą, paleidžiant programą, joje yra inicijuojami moduliai `bluetoothSerial.begin(9600)`, deklaruojamos reikšmės globaliems kintamiesiems, pridedami greičiai motorams, užtikrinama, kad visos judančios aparatūros dalys yra sustojusios.

```
servo.attach(motor);
M1.setSpeed(Speed);
```

`void loop()` funkcija laiko pagrindinį programos kodą, kuris atlieka roboto, jo veikimo metu. Kadangi robotas bus valdomas balso komandomis siunčiamomis *Bluetooth* ryšiu, pirma užduotis yra pasigauti tą komandą. Naudojant *Bluetooth* modulį, robotas nuolatos klausosi ir laukia ateinančių komandų, jeigu komanda nėra nulis, tai yra gauta kažkokia informacija, robotas ją įrašo į laikiną kintamąjį ir perduoda tolimesniam tikrinimui.

```
if (bluetoothSerial.available() > 0) {
  command = bluetoothSerial.read();
```

Priklausomai nuo gautos komandos yra kviečiama atitinkama funkcija.

**lentelė 27** Komandų paaiškinimai

Komanda	Funkcija	Veiksmas
“F”	“forward()”	Važiuoti pirmyn
“S”	“stop()”	Sustoti
“L”	“left()”	Sukti kairėn
“R”	“right()”	Sukti dešinėn

Kiekviena funkcija prijungia atitinkamas kryptis atitinkamiems motorams ir paleidžia juos sukti deklaruotu greičiu.

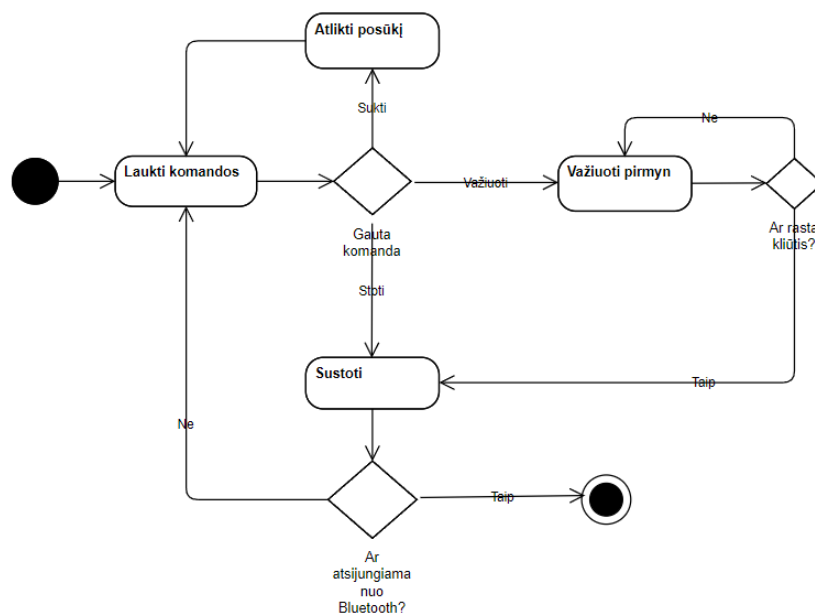
```
motor1.setSpeed(SPEED);
motor1.run(FORWARD);
```

Stabdant motorus arba sukant, atitinkamai keistusi *run* funkcijai paduodamas argumentas.

Viena iš siekiamų roboto funkcijų yra savaiminis navigavimas tarp kliučių, todėl funkcijoje *forward()* yra inicijuojamas dar vienas ciklas nuolatiniam atstumo matavimui. Jame robotas klausosi ultragarso duomenų ir jei atstumas iki kliuties yra mažesnis už nustatytą konstantą, jis atlieką posūkį, iškviesdamas sukimo funkciją. Tada robotas kartoja važiavimo į priekį funkciją tol kol negauna naujos komandos iš vartotojo.

```
if (distance <= 12)
  L = leftsee();
  servo.write(spoint);
  R = rightsee();
  servo.write(spoint);
if (L < R) {
  right();
```

forward();



pav. 43 Roboto veiklos diagrama

Veiklos diagram atvaizduoja roboto veikimą jam gavus komandą. Jeigu gauta komanda yra sustoti, robotas stabdo visus veikiančius motorus ir baigia judėjimą. Priešingu atveju robotas atlieka kitą komandoje nusakytą veiksmą, jeigu tai posūkis – robotas atitinkamai nustato ratų judėjimo kryptis, jeigu važiavimas pirmyn, robotas lygegrečiai komandos vykdymui įjungia ultragarsinį sensorių atstumo matavimui. Komanda vykdoma tol kol robotas negauna naujos komandos arba nėra aptinkama kliūtis, tokiu atveju robotas sustoja. Navigaciniam robotui atleidus visus motorus vartotojui atsiranda galimybė atsijungti nuo autonominės sistemos. Tokiu atveju robotas baigia savo darbą.

### 6.3. Kontrolerio kūrimas

Turint paruoštą *TFLite* modelį ir navigacinį robotą, reikia sukurti android programėlę, kuri naudosis mašinio mokymosi modelį balso atpažinimui ir siųs komandas robotui *Bluetooth* ryšiu. Šiam darbui atlikti buvo pasirinktas *android studio* ir *kotlin* programavimo kalba.

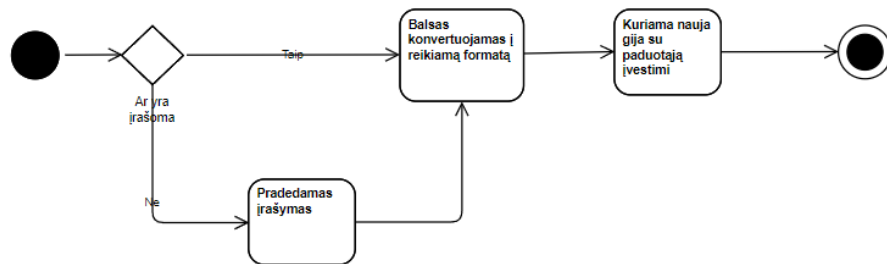
Kontrolerio įgyvendinimui buvo išskirtos pagrindinės užduotys:

1. Modelio užkrovimas
2. Balso įrašymo funkcijos sukūrimas
3. Balso atpažinimo funkcijos sukūrimas
4. Vartotojo grafinės sąsajos sukūrimas
5. *Bluetooth* ryšio programavimas

Pirmas žingsnis norint atlikti automatinį balso atpažinimą yra užkrauti *TFLite* modelį. Nors tai galima padaryti rankiniu būdu, šiam tikslui įgyvendinti pasirinkau naudoti *Gradle* konfigūracijos įrankį, kuris leidžia programos kompilacijos metu parsiusiti ir užkrauti reikalingus failus. Tokiu būdu ateityje bus galima pakeisti naudojamą modelį, nekeičiant pačios programos.

Tam sukuriamas failas *download\_model.gradle*, kuris atsiunčia modelį naudodamas funkciją *Task*, iš nurodytos repozitorijos.

```
task downloadSpeechClassifierModel(type: Download)
```



**pav. 44** Balso atpažinimo veiklos diagrama

Veliau šis atsiustas failas išsaugomas projekto direktorijoje, nurodytu pavadinimu. Kadangi robotas bus valdomas balsu, reikia sukurti funkciją, kuri klausysis garsinės įvesties ir konvertuos ją į reikiamą formatą. Tam sukurta funkcija `fun startAudioClassification()`, kuri naudoja *AudioClassifier* tipo kintamąjį, atskiroje gijoje.

Audio įrašui sukurti ir transformuoti į reikiamą formatą (neuroninis tinklas priima tik tam tikro ilgio failus) naudojamas *TensorAudio* tipo kintamasis. Naudojant šį kintamąjį, mes atidarome audio kanalą įvesčiai ir konvertuojame į reikiamą ilgį.

```
lengthInMilliseconds = ((classifier.requiredInputBufferSize * 1.0f) /
    classifier.requiredTensorAudioFormat.sampleRate) * 1000
```

Kintamasis *classifier* savyje turi kintamojo *TensorAudio* įvesties kanalą.

Turint atitinkamo ilgio audio įrašą, programa kuria naują giją klasifikacijai.

```
executor.scheduleAtFixedRate(
    classifyRunnable
```

Iškviesta funkcija atlieka pagrindinę balso atpažinimo dalį. Ši funkcija visada veikia atskiroje gijoje, taip užtikrinant realaus laiko garso įrašymą ir spėjimą. Funkcija gauna audio įrašą kaip argumentą ir naudodama *Tensorflow Lite* modelį gauna spėjimą, kurį vėliau atvaizduoja vartotojui, per vartotojo grafinės sąsajos komponentą ir perduoda Bluetooth ryšio komponentui.

Prieš iškviečiant šią funkciją svarbu deklaruoti naudojamą modelį. Modelis buvo užkrautas naudojant *Gradle* sistemą ir patalpintas į projekto direktoriją.



```
classifier = AudioClassifier.createFromFileAndOptions(context,
currentModel, options)
```

Kintamasis `currentModel` savyje laiko nuorodą į naudojamą mašininio mokymosi modelį.

Iškvietus funkciją pirmiausia paduotas `recorder` tipo kintamasis yra užkraunamas į `tensorAudio` kintamąjį ir paduodamas klasifikacijai.

```
val output = classifier.classify(tensorAudio)
```

Gautas rezultatas paduodamas *Bluetooth* ryšio komponentui.

```
SendData(output)
```

*Bluetooth* ryšio įgyvendinimui buvo pasirinkta *Socket* biblioteka. Kadangi Arduino gali priimti tik *char* tipo kintamuosius, pirmas žingsnis yra pakeisti gautą spėjimą į atitinkamą *char* komandą. Norint sugeneruotą komandą išsiusti, reikia paruošti *Bluetooth* ryšį.

Įjungus programėlę, pradedama aktyvių *Bluetooth* įrenginių paieška, kurie vėliau atvaizduojami *ListView* komponente. Kiekvienas elementas saraše sudarytas iš adreso ir vardo, adresas naudojamas kaip argumentas, prisijungimo ir duomenų siuntimo funkcijom, tačiau yra paslėptas nuo vartotojo. Įrenginio vardas atvaizduojamas vartotojui, kad šis galėtų pasirinkti ir paspausti ant norimo įrenginio.

```
BluetoothDevice.ACTION_FOUND
```

Lambda funkcija nuolat tikrina galimus įrenginius, jeigu randa, prided jų informaciją į sarašą.

```
val device: BluetoothDevice? =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE)
m_discoveredDevices.add(device)
```

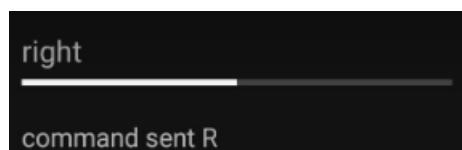
Tokiu būdu turime sarašą, su aktyviais *Bluetooth* įrenginiais, norėdami vartotojui atvaizduoti jų pavadinimus, pakartojam funkciją, su *indexOf* plėtinio.

Vienas iš sistemos reikalavimų yra galimybė vartotojui atnaujinti įrenginių sarašą, tam sukurtas specialus mygtukas, kuris su kiekvienu paspaudimu iškviečia anksčiau aprašytą funkciją.

Kitas žingsnis yra leisti vartotojui prisijungti prie pasirinkto įrenginio. Tam naudosime integruotą funkciją, kuri klausosi ar vartotojas paspaudžia ant vieno iš *ListView* elementų.

```
main_select_user_list.setOnItemClickListener =
AdapterView.OnItemClickListener{ _, _, position, _ ->
```

Kiekvienas paspaudimas kviečia naują klasę ir jai perduoda pasirinkto įrenginio duomenis, ši klasė toliau užkrauna vartotojui naują langą ir fone įvykdo prisijungimą.



**pav. 45** Komanda “Dešinėn“

Užkrautoje klasėje veikia balso atpažinimo komponentas, kuris naudodamas dvi gijas vienu metu skaito vartotojo garsinę įvestį ir perduoda ją įvesties spėjimui ir tolimesniam

siuntimui. Užkrautas langas vartotojui atvaizduoja jo įvesties spėjimus, su spėjimo tikslumu ir informuoja apie sėkmingas arba nesėkmingas siuntimo operacijas.

Taip pat šiame lange vartotojas gali atsijungti nuo pasirinkto įrenginio, taip grįždamas į pradinį langą.

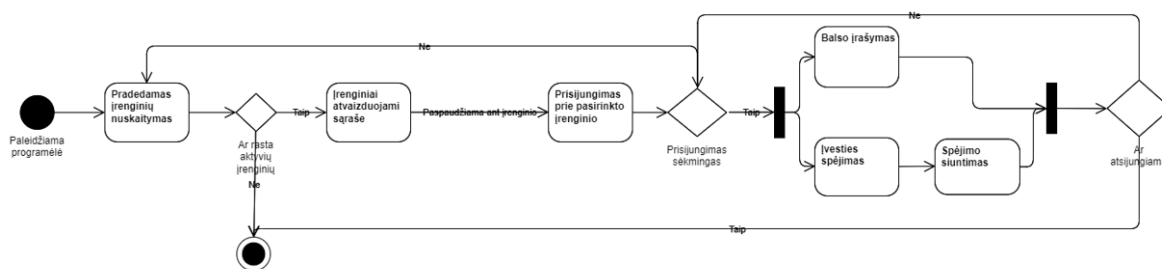
Už įvesties siuntimą atsako klasė *SendActivity*. Klasė gauna *char* tipo komandas kaip argumentus ir siunčia jas prijungtam įrenginiui.

```
val device: BluetoothDevice = intent.getParcelableExtra(EXTRA_DEVICE)
val command: String = intent.getStringExtra(EXTRA_MESSAGE)
outputStream.write(command)
```

Paskutinis etapas kontrolerio kūrimo yra grafinės vartotojo sąsajos programavimas. Šis sistemos komponentas atsako už vartotojo bendravimą su sistema, rezultatų, bei pranešimų atvaizdavimą, bei vartotojo pasirinkimų realizavimą.

Šiai daliai įgyvendinti buvo sukurti du XML langai. Vienas iš jų yra atvaizduojamas vartotojui, jam įsijungus programėlę, šis langas atsako už *Bluetooth* įrenginių atvaizdavimą, jų sąrašo atnaujinimą. Tam naudojami du elementai – *Listview* ir *Button*. Paspaudus pasirinktą sąrašo elementą atidaromas antras programos langas atsakingas už komandų spėjimo bei jų siuntimo atvaizdavimą, bei atsijungimą nuo įrenginio. Šiame lange fone veikia balso įrašymas, o vartotojas spėjamą įvestį mato *TextView* lauke, šalia kurio *Progressbar* atvaizduoja spėjamos įvesties tikslumą procentais. Apačioje yra antras *TextView*, kuris atsako už vartotojo informavimą apie sėkmingas arba neįvykusias operacijas.

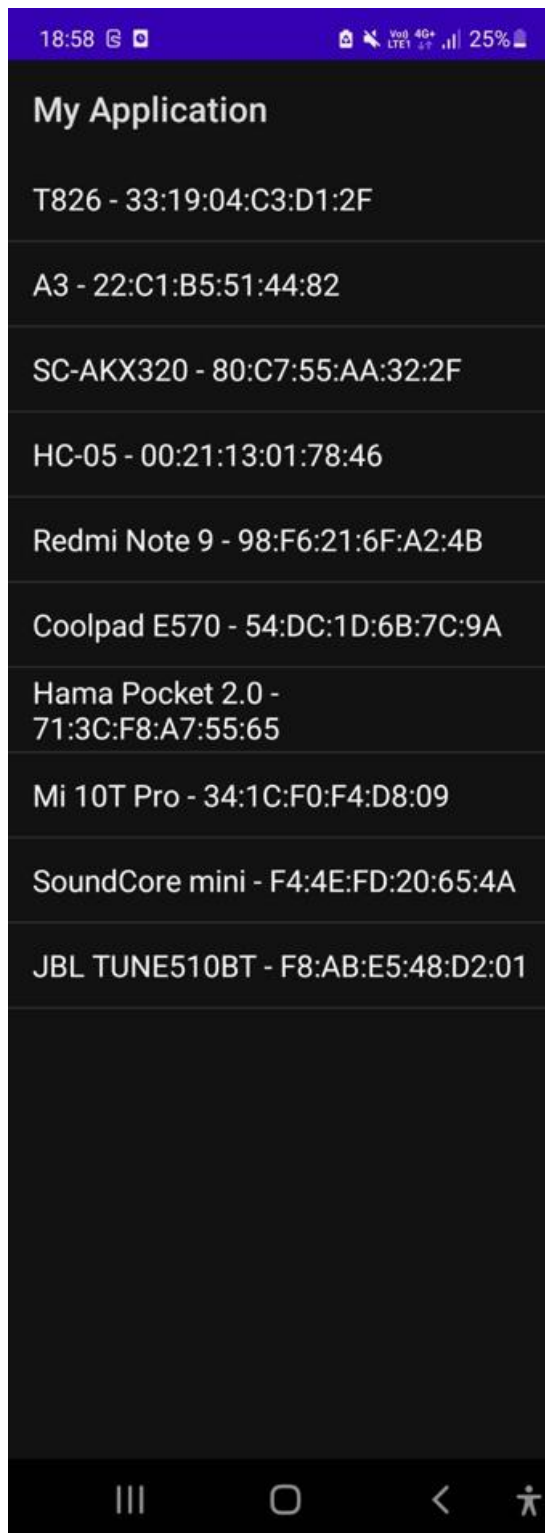
Lango apačioje yra mygtukas, kuris leidžia vartotojui atsijungti nuo įrenginio ir grįžti į pradinį langą.



pav. 46 Programėlės veiklos diagrama

## 6.4. Sistemos vizualizacija

Paleidus programėlę matomas pradinis langas, kuriame automatiškai pakraunami galimi Bluetooth įrenginiai. Šie įrenginiai vartotojui atvaizduojami sąraše (*Listview*). Kiekvienam sąrašo elementui yra priskirtas paspaudimas, taigi vartotojui norint prisijungti prie pasirinkto įrenginio, užtenka spustelėti ant jo pavadinimo.

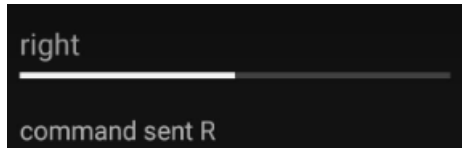


**pav. 47** Pagrindinis langas

Paspaudus ant įrenginio fone ivykdomas prisijungimas Bluetooth ryšiu ir pakraunamas naujas langas. Šiame lange vartotojas gali siusti komandas prijungtam įrenginiui. Lango viršuje yra progreso matuoklis, kuris atvaizduoja kiekvieno spėjimo tikslumą, šalia jo teksto lauke yra atvaizduojamas pats spėjimas. Vartotojo sakoma įvestis yra nuolatos įrašoma, paduodama balso atpažinimo komponentui ir atvaizduojama vartotojui. Naudotojui nereikia imtis jokių

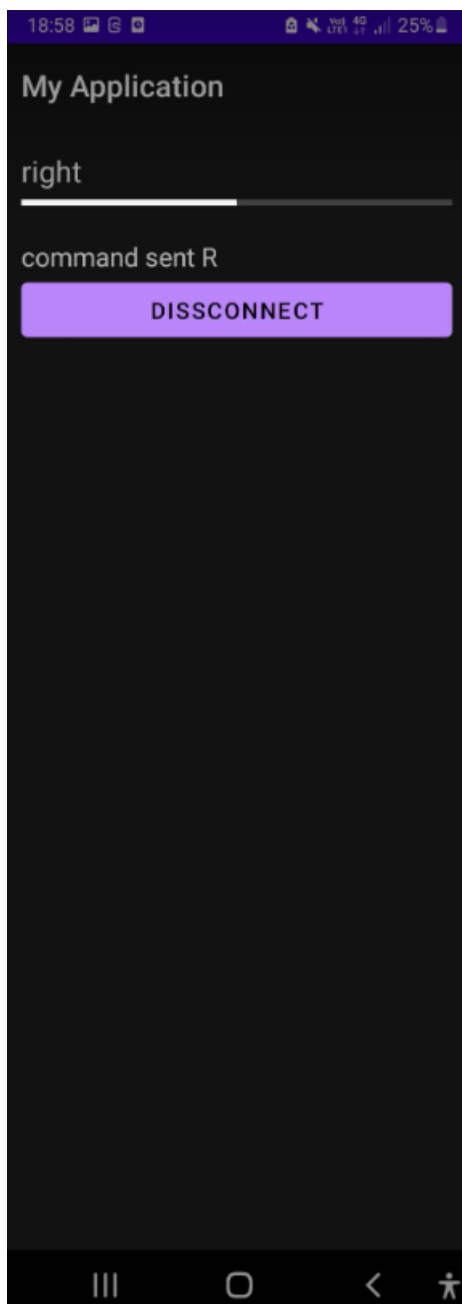
papildomų veiksmų norint išsiusti komandą prijungtam įrenginiui, sistema užtikrina, jog įrašomi žodžiai būtų nuolatos perduodami spėjimams o šie nuolatos siunčiami įrenginiui.

Visa sistemos logika veikia fone, vartotojas gauna tik tarpinius spėjimo rezultatus, bei patvirtinimus, kad komanda buvo nusiusta sėkmingai.



**pav. 48** Programos progreso matuoklis

Šiame lange vartotojas taip pat gali atsijungti nuo įrenginio, Tam yra sukurtas mygtukas “Disconnect”. Paspaudus šį mygtuką, pirmiausia prijungtam įrenginiui siunčiama sustojimo



**pav. 49** Komandų siuntimo langas

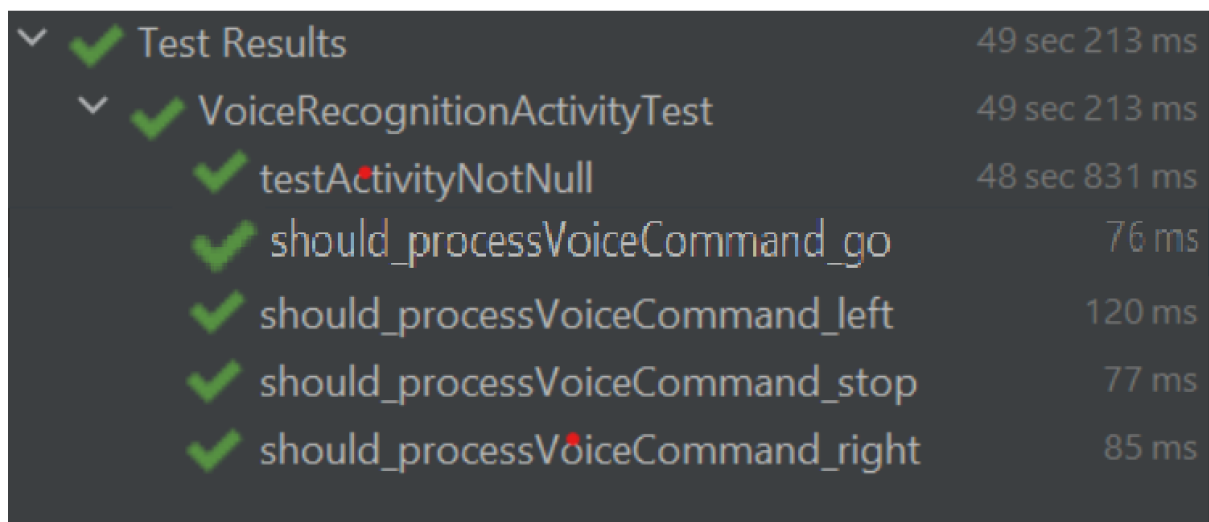
komanda, taip užtikrinant, kad robotas nejudės toliau išjungus kontrolerį. Jeigu komanda buvo nusiusta sėkmingai, atliekamas atsijungimas ir vartotojui pakraunamas pradinis langas.

## 7. Testavimas

### 7.1. Balso atpažinimo testavimas

Norint atlikti testavimą balso atpažinimo komponentui buvo pasirinkta testuoti VoicerecognitionActivity klasę. Testavimui buvo pasitelkta Junit ir RoboElectric bibliotekos, kuriomis naudojantis buvo parašyti Unit testai.

Testų tikslas buvo užtikrinti teisingą neuroninio tinkle veikimą ir komandų spėjimą. Tam įgyvendinti buvo sukurtos AudioRecord ir Classifier objektų kopijos (angl. *Mock*) ir padavus joms garsinius failus buvo stebima kokia komanda siunčiama robotui.



✓	Test Results	49 sec 213 ms
✓	VoiceRecognitionActivityTest	49 sec 213 ms
✓	testActivityNotNull	48 sec 831 ms
✓	should_processVoiceCommand_go	76 ms
✓	should_processVoiceCommand_left	120 ms
✓	should_processVoiceCommand_stop	77 ms
✓	should_processVoiceCommand_right	85 ms

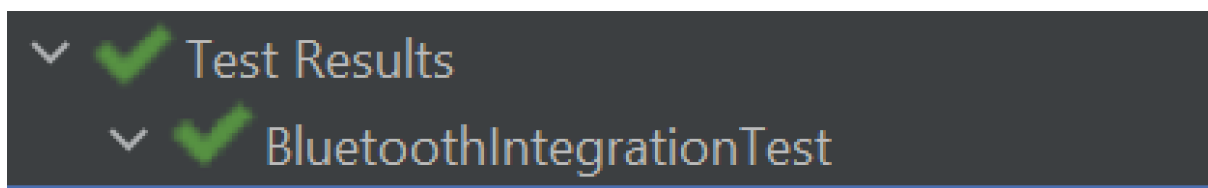
pav. 50 Unit testai balso atpažinimo komponentui

Iš testo rezultatų matome, kad visi garsiniai failai buvo atpažinti teisingai ir visiems spėjimams buvo priskirtos teisingos komandos

### 7.2. Bluetooth ryšio testavimas

Bluetooth ryšio testavimui buvo parašyti Unit testai naudojant Junit4 ir RoboElectric bibliotekas. Testavimas buvo leidžiamas ant fizinio Android įrenginio. Pagrindinis testų tikslas buvo patikrinti ar programėlė gali prisijungti prie pasirinkto įrenginio ir siusti jam duomenis. Tam buvo sukurtos atitinkamų objektų kopijos (Angl. *Mock*), buvo kopijuoti BluetoothDevice ir AudioRecord objektai. Pirmasis objektas buvo kviečiamas su prisijungimo funkcija, kuriai buvo paduotas sąrašo elemento paspaudimas. AudioRecord kopija buvo reikalinga užfiksuoti duomenis siuntimui ir paduoti juos funkcijai.

Kadangi norint ištestuoti ar duomenis bus siunčiami prijungtam įrenginiui, reikia prie to įrenginio prisijungti, šie du testai buvo įgyvendinti vienu metu.



pav. 51 Unit testai Bluetooth ryšio komponentui

### 7.3. Rankinis roboto testavimas

Paskutinis sistemos testavimo etapas buvo ištestuoti roboto ir programėlės veikimą rankiniu būdu, realioje aplinkoje. Šis testavimo etapas skirtas patikrinti sistemos atitikimą funkciniais reikalavimams ir užtikrinti, kad skirtingi sistemos komponentai dirba kartu, kaip buvo numatyta.

Testavimas apėmė tiek pilną programėlės funkcionalumą, tiek roboto. Testavimo metu buvo įjungta programėlė, su kuria buvo bandoma prisijungti prie roboto Bluetooth ryšiu. Atlikus prisijungimą programėlė įrašinėjo komandas ir siuntė jas robotui, šis atlikdavo atitinkamą veiksmą.

lentelė 28 Rankinio testavimo rezultatai

Testo nr.	Nustatymai	Laukiamas rezultatas	Gautas rezultatas	Ar testas praėjo?
1	Komanda: “go”	Robotas pradeda važiuoti pirmyn	Robotas pradėjo važiuoti <b>pirmyn</b>	Taip
2	Komanda: “left”	Robotas atlieka posūkį į kairę	Robotas pasuko <b>kairėn</b>	Taip
3	Komanda: “right”	Robotas atlieka posūkį į dešinę	Robotas pasuko <b>dešinėn</b>	Taip
4	Komanda: “stop”	Robotas sustoja	Robotas <b>sustojo</b>	Taip
5	Roboto kelyje padėta kliutis	Robotas sustoja savaime	Robotas <b>sustojo</b>	Taip
6	Atsijungiama nuo roboto	Robotas sustoja, programėlė grįžta į pradinę langą	Robotas <b>sustojo</b> , vartotojui atidarytas <b>pradinis</b> langas	Taip

Atlikus testavimą, matome, kad Sistema tenkina visus esminius funkcinis reikalavimus ir veikia taip, kaip buvo numatyta.

## 8. Išvados

1. Išanalizavus panašias funkcijas atliekančias sistemas, tokias kaip *Amazon Astro* ir *Scorpio*, buvo prieita išvada, kad tokios sistemos turi sudėtingą specifinę architektūrą ir sudėtingą naudojimą. Savo darbe pritaikiau universalesnį inžinerinį sprendimą ir sukuriau lengvesnę vartotojo sąsają, kuria lengva naudotis.
2. Atlikus neuroninių tinklų ir balso atpažinimo technologijų palyginamąją analizę buvo nuspręsta naudoti konvoliucinį neuroninį tinklą su *ReLu* aktyvavimo funkcija, kadangi tai padidina tinklo tikslumą ir leidžia greitesnį treniravimą. Šiam tinklui sukurti ir apmokyti buvo pasirinkta *Tensorflow*, dėl paprasto naudojimo ir geros duomenų vizualizacijos.
3. Atlikus mikrokontrolerių palyginamąją analizę, buvo pasirinktas *Arduino Uno R3*, dėl aukšto suderinamumo su papildoma įranga ir galimybės užkrauti programą tiesiai į grandinę.
4. Atlikus integruotųjų programavimo aplinkų ir programavimo kalbų palyginamąją analizę, kontrolerio kūrimui buvo pasirinkta naudoti *Android studio* ir *Kotlin*, dėl aukštesnio našumo ir galimybės leisti programas *Android* aplinkoje.
5. Sukurius planuojamos sistemos architektūrą buvo numatytos esminės sistemos funkcijos ir jos sandara. Sistema buvo išskaidyta į komponentų rinkinį, iš kurių kiekvienas atsako už tam tikrą sistemos veikimo dalį. Šiems komponentams vėliau buvo nubrėžta įgyvendinimo diagrama, kuri detalizuoja technologijas ir failus veikiančius sistemoje. Taip pat buvo atvaizduota preliminarinė sistemos struktūra, parodant kokios programuojamos esybės bendrauja tarpusavyje.
6. Apmokius neuroninius tinklus buvo gautas 86 procentų tikslumas atpažinant garsinę įvestį. Tinklas buvo apmokytas atpažinti pagrindinę robotų judėjimui reikalingas komandas ir atsikirti jas nuo fono garsų. Sukurtas mašininio mokymosi modelis buvo eksportuotas *TFLite* pavidalu, tolimesniam naudojimui.
7. Suprogramavus mobiliąją programėlę, buvo matyti, kad ji atlieka visas nustatytas funkcijas, skaito ir spėja garsinę įvestį, konvertuoja ją į reikiamą formatą ir kuria *Bluetooth* ryšį. Nuadodamasis ja vartotojas gauna patvirtinimus apie sėkmingas arba nepavykusias operacijas ir gali siusti komandas, bet kokiam prijungtam įrenginiui.
8. Sukonstravus ir suprogramavus robotą paaiškėjo, kad visi jo komponentas atlieka savo funkciją – robotas priima *Bluetooth* ryšiu gaunamus duomenis ir teisingai

priskiria jiems veiksmus. Robotas taip pat pajėgus orientuotis aplinkoje savarankiškai, įdiegtų sensoriu pagalba robotas ieško galimos kliuties ir priima atitinkamą sprendimą. Roboto veiksmas, bet kada gali būti pertrauktas vartotojo, nauja komanda.

9. Atlikus testavimą paaiškėjo, kad sistema atitinka 100 procentų funkcinių reikalavimų ir, kad visi sistemos komponentai veikia taip kaip iš jų tikimasi. Testavimas taip pat atskleidė, kad galima padidinti neuroninių tinklų tikslumą. Roboto komponentui buvo pritaikytas rankinis testavimas realioje aplinkoje, atlikus testavimą paaiškėjo, kad buvo pasiektas darbo pradžioje užsibrėžtas tikslas – balso atpažinimo pagalba autonominė sistema išvengia ne tik kliučių, kurias aptinka sensoriaus pagalba, bet ir į jo radarą nepatenkančių ar dėl besikeičiančių aplinkos sąlygų atsiradusių kliučių.
10. Sistema galima tobulinti ateityje didinant balso atpažinimo tikslumą ir roboto reakcijos į komandą laiką.

#### Literatūros sąrašas

(2022). *Android Studio*. techtarget.

*apiai 1.2.3*. (2017). Retrieved from pypi.org: <https://pypi.org/project/apiai/>

Begam, L. &. (2010). *oice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques*.

BenjaminWalter. (2023). Analysis of convolutional neural network image classifiers in a hierarchical max-pooling model with additional local pooling. *Journal of Statistical Planning and Inference*.

Brownlee, J. (2023). *An Introduction to Recurrent Neural Networks and the Math That Powers Them*. Retrieved from machinelearningmastery: <https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/>

CanBingol, M. (2021). Performing predefined tasks using the human–robot interaction on speech recognition for an industrial robot. *Engineering Applications of Artificial Intelligence*.

Chen, Y. (2023). A survey of recent advances on stability analysis, state estimation and synchronization control for neural networks. *Neurocomputing*.

*CNN / Introduction to Pooling Layer*. (2023). Retrieved from geeksforgeeks: <https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/>

*Control DC, Stepper & Servo with L293D Motor Driver Shield & Arduino*. (n.d.). Retrieved from engineers: <https://lastminuteengineers.com/l293d-motor-driver-shield-arduino-tutorial/>

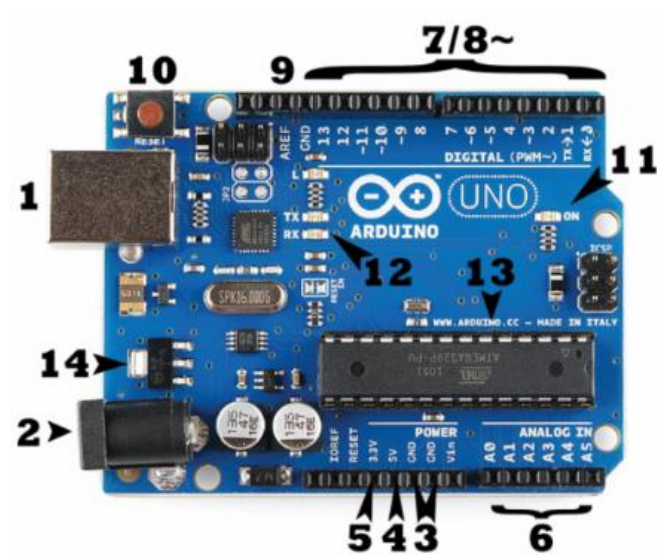


- Dejan. (n.d.). *Ultrasonic Sensor HC-SR04 and Arduino – Complete Guide*. Retrieved from howmechatronics: <https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/>
- Ekren, E. K. (2022). *What Is Xcode and How to Use It?* Retrieved from netguru: <https://www.netguru.com/blog/what-is-xcode-and-how-to-use-it>
- Emet. (2022). *The Different Versions of the Raspberry Pi*. Retrieved from pimylifeup: <https://pimylifeup.com/raspberry-pi-versions/>
- Hartman, J. (2022). *What is Java? Definition, Meaning & Features of Java Platforms*. Retrieved from guru99: <https://www.guru99.com/java-platform.html>
- HC-05 Bluetooth Module Interfacing with Arduino UNO. (2022). Retrieved from electronicwings: <https://www.electronicwings.com/arduino/hc-05-bluetooth-module-interfacing-with-arduino-uno>
- Heller, M. (2022). *What is Kotlin? The Java alternative explained*. Retrieved from infoworld: <https://www.infoworld.com/article/3224868/what-is-kotlin-the-java-alternative-explained.html>
- Yegulalp, S. (2022). *What is TensorFlow? The machine learning library explained*. Retrieved from infoWorld.
- incredibuild. (2020). Retrieved from What is Visual Studio?: <https://www.incredibuild.com/integrations/visual-studio>
- YunfeiGuo. (2022). Personalized voice activated grasping system for a robotic exoskeleton glove. *Mechatronics*.
- J.A.Dulce-Galindo. (2022). Distributed supervisory control for multiple robot autonomous navigation performing single-robot tasks. *Mechatronics*.
- Kang, A. (2017). *Understanding the Differences Between Alexa, APL.ai, WIT.ai, and LUIS/Cortana*.
- KishanKondaveeti, H. (2021). A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations. *Computer Science Review*.
- Lutkevich, B. (2021). *microcontroller (MCU)*. Retrieved from techtarget: <https://www.techtarget.com/iotagenda/definition/microcontroller>
- Lutkevich, B. (2022). *Robotics*. Retrieved from techtarget: <https://www.techtarget.com/whatis/definition/bot-robot>
- M.Lee, S. (2020). Voice recognition: An examination of an evolving technology and its use in organizations. *Computers & Operations Research*.
- N.Fadhil, M. (2021). Voice recognition system using machine learning techniques. *materialstoday:proceedings*.
- Nussey, J. (2022). *An Overview of Arduino Shields*.
- Ondas, S. (2013). *Service Robot SCORPIO with Robust Speech Interface*. Sage.
- Priest, D. (2022). *Amazon Astro Review*. Retrieved from CNET: <https://www.cnet.com/home/smart-home/amazon-astro-review/>
- Raspberry Pi OS. (n.d.). Retrieved from raspberrypi: <https://www.raspberrypi.com/software/>
- S.Matveev, A. (2021). Real-time navigation of mobile robots in problems of border patrolling and avoiding collisions with moving and deforming obstacles. *Robotics and Autonomous Systems*.
- Shi, Z. (2021). Von Neumann Architecture. *Intelligence Science*.
- Sivamurugesan, K. &. (2018). Design and Development of Laplacian Pyramid Combined with Bilateral Filtering Based Image Denoising.
- Teja, R. (2021). *Arduino Modules for DIY Projects*. Retrieved from electronicshub: <https://www.electronicshub.org/arduino-modules-list/>
- What are the Arduino sensors?* (n.d.). Retrieved from javapoint: <https://www.javatpoint.com/arduino-sensors>
- What is a Raspberry Pi?* (2020). Retrieved from opensource: <https://opensource.com/resources/raspberry-pi>

What is an Arduino? (2022). Retrieved from sparkfun:  
<https://learn.sparkfun.com/tutorials/what-is-an-arduino/all>  
 (2022). What is Gradle?

## Priedas 1 Arduino aparatūros komponentai

Komponentas	Paskirtis
Maitinimas	Kiekvieną Arduino plokštę reikia prijungti prie maitinimo šaltinio. „Arduino UNO“ gali būti maitinamas iš USB kabelio, gaunamo iš jūsų kompiuterio arba iš bet kokio kito maitinimo šaltinio. Viršuje esančiame paveikslėlyje USB jungtis pažymėta (1), o cilindro lizdas – (2). USB jungtis taip pat yra būdas įkelti kodą į savo Arduino plokštę.
Kaiščiai	Skirti prijungti periferinius įrenginius, maitinimo šaltinius ar įžeminti grandinę.
Paleidimo iš naujo mygtukas	Paleidimo iš naujo mygtukas. (Angl. <i>restart</i> ) (10), šis mygtukas panaikina visą kodą Arduino plokštėje
Maitinimo LED	(11), indikuoja kai Arduino yra prijungtas prie maitinimo šaltinio.
TX ir RX šviesos diodai	TX reiškia siuntimą, RX – priėmimą. Šie ženklai elektronikoje pasirodo gana dažnai, nurodant kaiščius, atsakingus už nuoseklių ryši. (12). Šie šviesos diodai suteikia indikacijų, kai mūsų Arduino gauna arba perduoda duomenis (pvz., kai įkeliamo naują programą į plokštę).
Pagrindinis IC	(13). IC arba integruotas elektros grandynas tai Arduino smegenys, priklausomai nuo Arduino tipo IC gali veikti kaip stiprintuvas, generatorius, laikmatis, skaitiklis, loginiai vartai, kompiuterio atmintis, mikrovaldiklis arba mikroprocesorius.
Įtampos reguliatorius	(14), atsako už įtampos leidžiamos per Arduino dydį.



pav. 50 Arduino uno komponentai

## 2 priedas - kodaRobotui.cpp

```
#include <AFMotor.h>
#include <NewPing.h>
#include<Servo.h>
#include <SoftwareSerial.h>
#define TRIGGER_PIN A1
#define ECHO_PIN A0
#define MAX_DISTANCE 300
#define IR A5

AF_DCMotor motor1(1, MOTOR12_1KHZ);
AF_DCMotor motor2(2, MOTOR12_1KHZ);
AF_DCMotor motor3(3, MOTOR34_1KHZ);
AF_DCMotor motor4(4, MOTOR34_1KHZ);

SoftwareSerial bluetoothSerial(9, 10);

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

//Servo myservo;

char voice;

void setup() {
  bluetoothSerial.begin(9600);
  //myservo.attach(10);
  //myservo.write(90);
  pinMode(IR, INPUT);
  pinMode(TRIGGER_PIN,OUTPUT); //Assign ultrasonic
  sensor pin modes
  pinMode(ECHO_PIN,INPUT);
}

void loop() {
  int distance = sonar.ping_cm();
  int IR1 = digitalRead(IR);
  //Serial.println(IR1);

  if(Serial.available()>0) {
    voice="";
    delay(2);
    voice = bluetoothSerial.read();
    delay(2);
    if (voice == "L") {
      left();
    }
    else if(voice == "R") {
      right();
    }
  }
}
```

```

while(voice == "F") {
    forward();
}
if(voice == "B") {
    Stop();
}
}

void forward() {
    int distance = sonar.ping_cm();

    if(distance < 10){
        Stop();
        voice="";
    }else {
        motor1.setSpeed(700);
        motor1.run(FORWARD);
        motor2.setSpeed(700);
        motor2.run(FORWARD);
        motor3.setSpeed(700);
        motor3.run(FORWARD);
        motor4.setSpeed(700);
        motor4.run(FORWARD);
    }
}
}

```