

Informatikos fakultetas

**T125B114 Robotų programavimo technologijos**

**Projektinis darbas**

**Ataskaita**

|  |  |  |
| --- | --- | --- |
|  | Parengė: | Mangirdas Kazlauskas, IFF-4/1  Simonas Baltulionis IFF-4/1 |
| Dėstytojai: | Mindaugas Vasiljevas  Aurimas Jakštas |

Kaunas 2017

Turinys

[1. Projekto tikslas ir užduoties analizė 3](#_Toc501065043)

[2. Roboto aprašymas 4](#_Toc501065044)

[3. Roboto valdymo architektūra 6](#_Toc501065045)

[4. Roboto veikimo algoritmai 6](#_Toc501065046)

[4.1. Bendra roboto veikimo veiklos diagrama 6](#_Toc501065047)

[4.2. Roboto valdymo pulteliu veiklos diagrama 8](#_Toc501065048)

[4.3. Linijos sekimo algoritmas veiklos diagrama 9](#_Toc501065049)

[4.4. Autonominio judėjimo algoritmas veiklos diagrama 10](#_Toc501065050)

[4.5. Himno grojimo algoritmas veiklos diagrama 11](#_Toc501065051)

[5. Roboto valdymo programa 11](#_Toc501065052)

[6. Roboto valdymo eksperimentinis tyrimas 28](#_Toc501065053)

[7. Išvados 29](#_Toc501065054)

[8. Naudota literatūra 29](#_Toc501065055)

# Projekto tikslas ir užduoties analizė

Projektinio darbo tikslas – išanalizuoti gauto roboto architektūrą, elgseną, galimybes bei naudojantis programinėmis priemonėmis realizuoti konkretų roboto veikimo scenarijų/užduotis.

Mūsų užsibrėžta projektinio darbo užduotis – panaudoti kiekvieną gauto roboto mBot modulį konkrečioms užduotims atlikti:

1. Linijos sekimo sensorius – realizuoti linijos sekimo algoritmą, kad robotas neišklystų iš trasos;
2. Ultragarso sensorius – aptikti kliūtis ir jų išvengti;
3. Roboto garso modulis – sugroti programiškai aprašytą melodiją;
4. Roboto infraraudonųjų spindulių modulis – valdyti robotą pulteliu.

Apžvelgus „mBot“ turimas galimybes (turimus modulius) projektiniam darbui atlikti buvo sugalvoti 4 konkrečios užduotys, kurios turi būti realizuotos projektinio darbo metu:

1. Valdyti roboto judėjimą pulteliu bei kartu turint galimybę keisti roboto greitį;
2. Robotas turi sekti liniją (neišklysti iš trasos) bei apvažiuoti kelyje sutinkamas kliūtis;
3. Robotas turi autonomiškai judėti erdvėje ir išvengti priešakyje esančių kliūčių;
4. Robotas turi „mokėti“ sugroti Lietuvos himną.

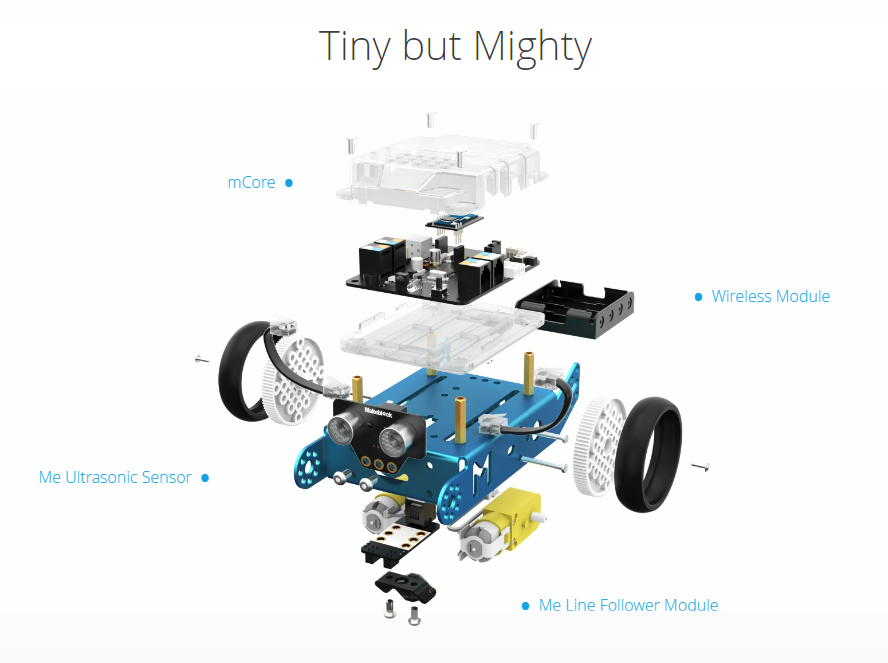
Kiekvienos iškeltos užduoties analizė pateikta pirmoje lentelėje:

1 lentelė. Iškeltų projektinio darbo užduočių analizė

|  |  |
| --- | --- |
| Užduotis | Analizė |
| Roboto valdymas pulteliu | mBot pakete pridedamas infraraudonuosius spindulius skleidžiantis pultelis su programuojamais mygtukais, todėl reikia išsiaiškinti, kaip yra aprašomas kiekvienas mygtukas ir kaip priskirti konkrečias funkcijas atskiriems mygtukams bei taip kontroliuoti roboto judėjimą. |
| Linijos sekimas | Reikia išsiaiškinti, kaip veikia linijos sekimo sensorius bei kaip aprašomi sensoriaus perduodami duomenys. Atsižvelgiant į konkrečius sensoriaus perduodamus parametrus robotas turi atlikti konkrečius veiksmus, kad tik išsilaikytų trasoje.  Taip pat reikia išsiaiškinti ultragarso sensoriaus veikimą, kad robotas, priešakyje aptikęs kliūtį (kai sensorius užfiksuoja objektą priešakyje), ją apvažiuotų. |
| Autonominis judėjimas | Taip pat, kaip ir linijos sekimo užduotyje, pagrinde bus atsižvelgiama į ultragarso sensoriaus perduodamus duomenis, kad jei priešakyje robotas aptiks kliūtį, jis jos išvengtų ir toliau judėtų erdvėje. |
| Lietuvos himno grojimas | Reikia išsiaiškinti, kaip yra aprašomi konkretūs garso modulio skleidžiami garsai (natos, jų dažnis, ilgis ir pan.). Taip pat reikia aprašyti kiekvieną Lietuvos himno natą programiškai, kad robotas galėtų sugroti Lietuvos himną. |

# Roboto aprašymas

„mBot“ – edukacinio tipo robotas, labiau orientuotas į nedidelio amžiaus auditoriją, taip skatinant ją domėtis elektronika, robotika bei programavimu (1 pav.). Pats mBot robotas nėra vientisas, jis yra modifikuojamas, o prieš pirmą kartą naudojant, jį reikia susikonstruoti pačiam (sekant instrukcijas). Tiesa, robotą galima patobulinti papildomomis dalimis, kurios į bazinį roboto paketą nėra įtrauktos (pvz., naujos mechaninės detalės ir sensoriai).



1 pav. mBot bazinis robotas (Bluetooth versija)

2 paveikslėlyje pateikiamas detalių sąrašas, kuris sudaro mBot bazinę versiją.



2 pav. mBot bazinį robotą sudarančios dalys

Kaip matome, be roboto karkasą sudarančių dalių, „mBot“ turi linijos sekimo bei ultragarso sensorius ir valdymo pultelį, pagrindinę valdymo plokštę „mCore“ – šios dalys yra svarbiausios mūsų projektiniam darbui atlikti.

„mBot“ roboto specifikacijos yra pateiktos 2 lentelėje:

2 lentelė. mBot specifikacijos

|  |  |
| --- | --- |
| Programinė įranga ir programavimas | mBlock aplinka arba Arduino IDE. |
| Įvestys | Šviesos sensorius, korpuso mygtukas, infraraudonųjų spindulių valdymo pultelis, ultragarso sensorius ir linijos sekimo sensorius. |
| Išvestys | Garso modulis (angl. buzzer), RGB šviesos, infraraudonų spindulių siųstuvas, du motorai ir portai. |
| Mikrokontroleris | Atitinkantis Arduino Uno |
| Energijos šaltinis | 3.7V ličio jonų baterija arba 4 1.5V AA baterijos. |
| Belaidė komunikacija | Bluetooth arba 2.4G |
| Išmatavimai | 17 x 13 x 9cm (sumontuotas) |
| Svoris | 1,034 kg (sumontuotas) |

# Roboto valdymo architektūra

Roboto valdymą aprašyti galima dviem būdais:

1. Naudojant grafinę aplinką (pvz., mBlock);
2. Aprašant roboto valdymą programiniu kodu;

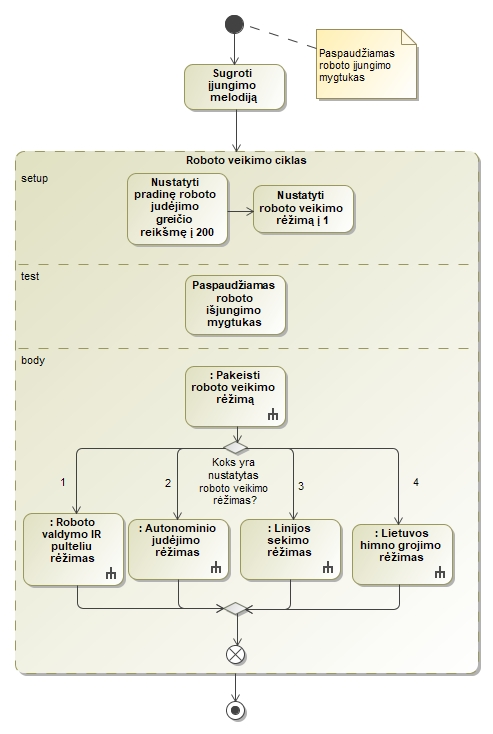
Projektiniam darbui atlikti buvo naudojamas antrasis būdas – roboto veikimo kodas buvo aprašytas naudojant C programavimo kalbą, kartu pasitelkiant atviro kodo Makeblock Arduino biblioteką „*Makeblock library*“ (versija 3.24). Biblioteka naudinga tuo, kad norint naudoti Makeblock elektroninius modulius (šiuos modulius turi robotas mBot), nereikia rašyti programinio kodo nuo pat pagrindų (pvz., aprašyti nuskaitytų signalų perdavimą), galima naudoti jau aprašytas atskirų modulių bei sensorių funkcijos, kurios paspartina programavimo procesą.

Projektinio darbo metu naudota programavimo aplinka – Arduino IDE (versija 1.6.5).

# Roboto veikimo algoritmai

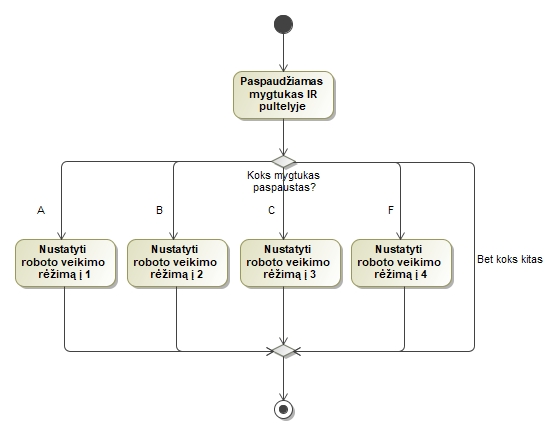
Šiame skyriuje pateikiamos roboto veikimo algoritmų (konkrečių užduočių) veiklos diagramos su jų paaiškinimais.

## Bendra roboto veikimo veiklos diagrama



3 pav. Bendra roboto veikimo veiklos diagrama

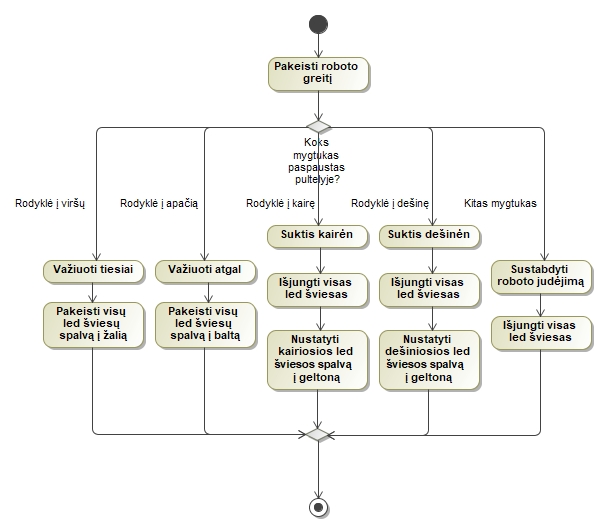
Kai robotas yra įjungiamas, pirmiausia sugroja įjungimo melodija, o toliau pradeda suktis amžinas ciklas iki tol, kol robotas yra išjungiamas („test“ dalis). Prieš amžiną ciklą atliekami pasiruošimo veiksmai („setup“ dalis): nustatoma roboto pradinė greičio reikšmė bei roboto veikimo rėžimas. Amžino ciklo metu pirmiausia yra tikrinama, koks yra aktyvus (nustatytas) roboto veikimo rėžimas, pagal kurį yra nustatoma, kokius veiksmus turi atlikti robotas. Taip pat amžino ciklo metu galima pakeisti roboto veikimo rėžimą (veiksmas prieš roboto veikimo rėžimo tikrinimą). Roboto veikimo rėžimo nustatymo veiklos diagrama pateikta 4 paveikslėlyje:



4 pav. Roboto veikimo rėžimo pakeitimo veiklos diagrama

Iš roboto veikimo rėžimo pakeitimo veiklos diagramos matome, kad rėžimui pakeisti yra skirti 4 IR pultelio mygtukai: A, B, C ir F – kiekvienas iš jų nustatom atitinkamą roboto veikimo rėžimo reikšmę, kuri yra tikrinama roboto amžino ciklo sąlygoje.

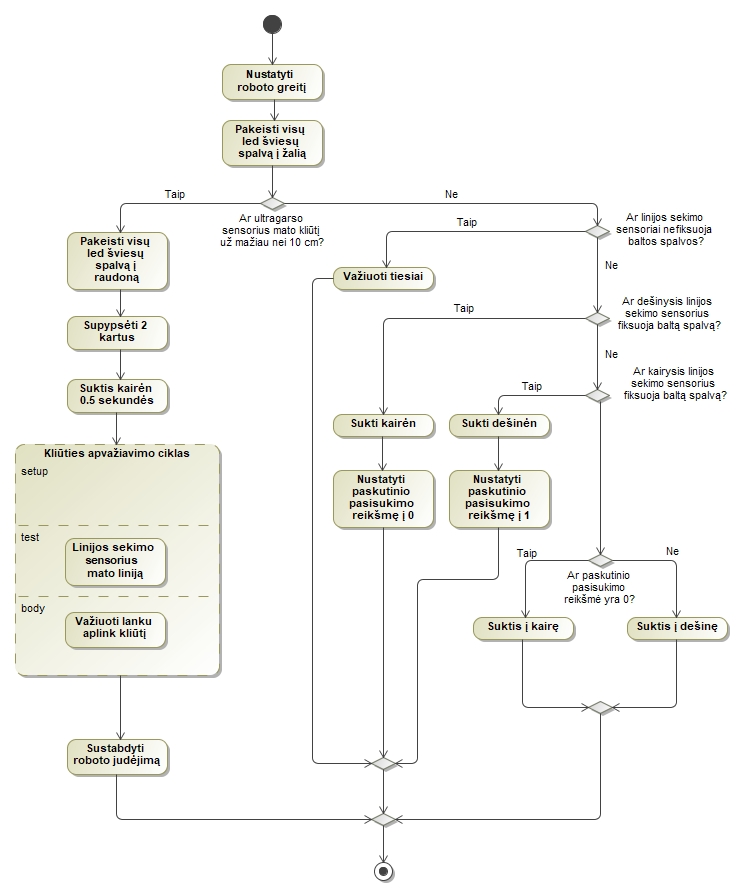
## Roboto valdymo pulteliu veiklos diagrama



5 pav. Roboto valdymo IR pulteliu veiklos diagrama

Pirmiausia yra nustatomas roboto greitis (mygtukais 1,2,3). Po to yra tikrinama, koks pultelio mygtukas yra nuspaustas – pagal jį robotas juda tiesiai, atgal arba sukasi į šonus.

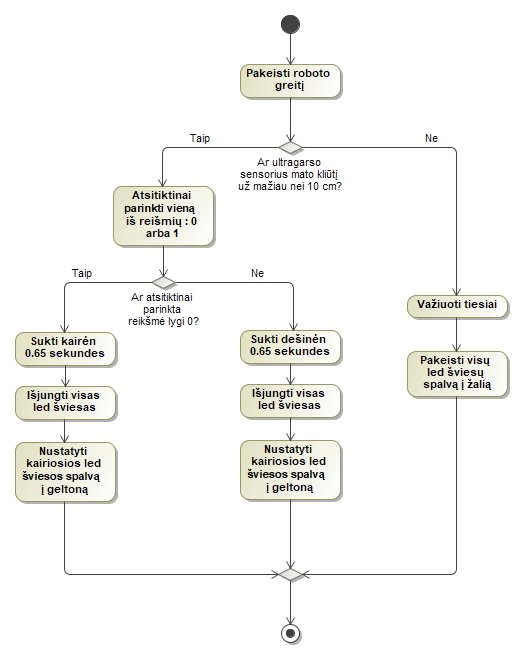
## Linijos sekimo algoritmas veiklos diagrama



6 pav. Linijos sekimo algoritmo veiklos diagrama

Iš diagramos matome, kad pirmiausia yra nustatomas roboto judėjimo greitis, bei pakeičiama led šviesų spalva į žalią. Po to yra tikrinama sąlyga ar robotas savo kelyje mato kokią nors kliūtį (arčiau nei 10 cm atstumu). Jei taip – roboto šviesų spalva pasikeičia į raudoną, robotas supypsi 2 kartus ir pasisuka kairėn. Po to vykdomas amžinas ciklas, kurio metu robotas lanku apvažiuoja kliūtį ir sustoja, jei vėl aptinka liniją (apvažiavus kliūtį grįžta į trasą). Jei roboto ultragarso sensorius kliūties nemato, tada tikrinami linijos sekimo sensoriaus duomenys. Pirmiausia tikrinama ar abu sensoriai mato liniją (nemato baltos spalvos) – jei taip, robotas važiuoja į priekį, jei ne, tada yra tikrinama, ar dešinysis linijos sekimo sensorius fiksuoja baltą spalvą. Jei taip, tada robotas sukasi kairėn ir nustato paskutinio pasisukimo reikšmę į 0, jei ne – tikrinami linijos sekimo kairiojo sensoriaus duomenys. Jei kairysis sensorius fiksuoja baltą spalvą, tada robotas sukasi dešinėn ir nustato paskutinio pasisukimo reikšmę į 1. Jei paskutinė sąlyga neteisinga, tai reiškia, kad robotas linijos nebemato ir turi jos ieškoti. Tam tikslui tikrinama paskutinio pasisukimo reikšmė – jei ji yra lygi 0, tai robotas sukasi į kairę ir ieško linijos, jei ne – sukasi į dešinę.

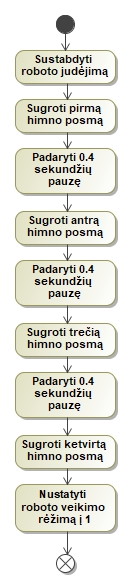
## Autonominio judėjimo algoritmas veiklos diagrama



7 pav. Autonominio judėjimo algoritmo veiklos diagrama

Pirmiausia yra pakeičiamas roboto greitis (jį keisti galima mygtukais 1, 2, 3), o tada tikrinama sąlyga, ar ultragarso sensorius priešais robotą fiksuoją kliūtį (10 cm ir mažesniu atstumu). Jei kliūtis yra, tada atsitiktinai yra sugeneruojama 0 arba 1 reikšmė, o į ją atsižvelgus, robotas pasisuka į kairę arba dešinę pusę, degant atitinkamos pusės geltonai led šviesai. Jei kliūties priešais robotą nėra, robotas tiesiog juda į priekį degant žaliai led šviesai.

## Himno grojimo algoritmas veiklos diagrama



8 pav. Lietuvos himno grojimo veiklos diagrama

8 paveikslėlyje pavaizduota diagrama tiesiog vaizduoja veiksmų seką, kuria remiantis yra grojamas Lietuvos himnas. Kiekvienas posmo grojimo veiksmas susideda iš eilės atitinkamų himno natų „sugrojimo“ roboto garso (angl. buzzer) moduliu. Kad posmai atsiskirtų vienas nuo kito, daromos 0.4 sekundžių pauzės.

# Roboto valdymo programa

Roboto valdymo programos kodas pateiktas 3 lentelėje:

|  |
| --- |
| Patriot\_mBot.ino |
| // Papildomos bibliotekos  #include <Arduino.h>  #include <Wire.h>  #include <SoftwareSerial.h>  #include <MeMCore.h>  // Kintamieji, skirti apibrežti roboto dalis/sensorius  MeDCMotor motor\_9(9);  MeDCMotor motor\_10(10);  MeRGBLed rgbled\_7(7, 7==7?2:4);  MeUltrasonicSensor ultrasonic\_3(3);  MeLineFollower linefollower\_2(2);  MeBuzzer buzzer;  MeIR ir;  // Kintamieji, naudojami algoritmuose  int Speed;  int Mode;  int Find\_Path\_Turn\_Direction;  // Pagalbines funkcijos  void Initialize();  void Set\_Mode();  void TurnOnSound();  void Change\_Speed();  // Himno grojimo funkcijos  void LT\_Anthem();  void Anthem\_part\_1();  void Anthem\_part\_2();  void Anthem\_part\_3();  void Anthem\_part\_4();  // IR valdymo funkcija  void Drive\_remote();  // Autonominio judejimo funkcija  void Object\_Avoidance(double Turn\_Speed);  // Linijos sekimo funkcija  void Line\_Tracer();  // Funkcija, kuri yra aktivuojama paleidus robota  void setup(){  ir.begin();  TurnOnSound();  Initialize();  }  // Roboto veikimo amžinojo ciklo veiksmai  void loop(){  Set\_Mode();  if((Mode == 1)){  Drive\_remote();  }  if((Mode == 2)){  Object\_Avoidance(100);  }  if((Mode == 3)){  Line\_Tracer();  }  if((Mode == 4)){  LT\_Anthem();  }  \_loop();  }  // Amžinasis ciklas, kuris sukasi ir laukiama signalu iš pultelio  void \_loop(){  ir.loop();  }  // Paruošimo veiksmai  void Initialize()  {  Speed = 200;    Mode = 1;    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();  }  // Funkcija, kuri fiksuoja, kad pulteliu buvo pakeistas veikimo režimas  void Set\_Mode()  {  if(ir.keyPressed(69)){  Mode = 1;  }    if(ir.keyPressed(70)){  Mode = 2;  }    if(ir.keyPressed(71)){  Mode = 3;  }    if(ir.keyPressed(13)){  Mode = 4;  }  }  // Isijungimo melodija  void TurnOnSound()  {  buzzer.tone(494, 250);  delay(20);    buzzer.tone(494, 250);  delay(20);    buzzer.tone(494, 250);  delay(20);    buzzer.tone(392, 1000);  delay(20);    buzzer.tone(440, 250);  delay(20);    buzzer.tone(440, 250);  delay(20);    buzzer.tone(440, 250);  delay(20);    buzzer.tone(349, 1000);  delay(20);  }  // Laukimo funkcija, kuri laukia nurodyta laiko tarpa (milisekundemis)  void \_delay(float miliseconds){  long endTime = millis() + miliseconds \* 1000;  while(millis() < endTime)\_loop();  }  // Roboto valdymo pulteliu funkcija  void Drive\_remote()  {  Change\_Speed();    if(ir.keyPressed(64)){  move(1,Speed);  rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();  }    if(ir.keyPressed(25)){  move(2,Speed);  rgbled\_7.setColor(0,255,255,255);  rgbled\_7.show();  }    if(ir.keyPressed(7)){  move(3,Speed);  rgbled\_7.setColor(0,0,0,0);  rgbled\_7.show();  rgbled\_7.setColor(2,255,255,0);  rgbled\_7.show();  }    if(ir.keyPressed(9)){  move(4,Speed);  rgbled\_7.setColor(0,0,0,0);  rgbled\_7.show();  rgbled\_7.setColor(1,255,255,0);  rgbled\_7.show();  }    if((!(ir.keyPressed(64))) || (!(ir.keyPressed(25)))){  move(1,0);  rgbled\_7.setColor(0,0,0,0);  rgbled\_7.show();  }  }  // Autonominio judejimo funkcija  void Object\_Avoidance(double Turn\_Speed)  {  Change\_Speed();  if((ultrasonic\_3.distanceCm()) < (10)){  if(((random(0,2))==1)){  move(4,Turn\_Speed);  rgbled\_7.setColor(0,0,0,0);  rgbled\_7.show();  rgbled\_7.setColor(1,255,255,0);  rgbled\_7.show();  \_delay(0.65);  }else{  move(3,Turn\_Speed);  rgbled\_7.setColor(0,0,0,0);  rgbled\_7.show();  rgbled\_7.setColor(2,255,255,0);  rgbled\_7.show();  \_delay(0.65);  }  }else{  move(1, Speed/2);  rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();  }  }  // Roboto judejimo greicio keitimo funkcija  void Change\_Speed()  {  if(ir.keyPressed(22)){  move(1,0);  Speed = 0;  buzzer.tone(131, 500);  delay(20);  }    if(ir.keyPressed(12)){  move(1,0);  Speed = 150;  buzzer.tone(175, 500);  delay(20);  }    if(ir.keyPressed(24)){  move(1,0);  Speed = 200;  buzzer.tone(262, 500);  delay(20);  }    if(ir.keyPressed(94)){  move(1,0);  Speed = 250;  buzzer.tone(349, 500);  delay(20);  }  }  // Himno grojimo funkcija  void LT\_Anthem()  {  move(1,0);    Anthem\_part\_1();  \_delay(0.4);    Anthem\_part\_2();  \_delay(0.4);    Anthem\_part\_3();  \_delay(0.4);    Anthem\_part\_4();    Mode = 1;  }  // 1 posmo natu seka  void Anthem\_part\_1()  {  rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(523, 1000);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(392, 250);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(523, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(587, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(659, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(587, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(523, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(587, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(659, 1000);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(587, 250);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(659, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(698, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(784, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(880, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(784, 500);  delay(20);    \_delay(0.3);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(880, 1000);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(784, 250);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(698, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(659, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(587, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(523, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(494, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(698, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(659, 1000);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(587, 250);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(523, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(587, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(659, 1000);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(587, 1000);  delay(20);  }  // 2 posmo natu seka  void Anthem\_part\_2()  {  rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(523, 1000);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(392, 250);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(523, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(587, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(659, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(587, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(523, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(587, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(659, 1000);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(587, 250);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(659, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(698, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(784, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(880, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(784, 500);  delay(20);    \_delay(0.3);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(880, 1000);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(784, 250);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(698, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(659, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(587, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(523, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(494, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(698, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(659, 1000);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(587, 250);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(523, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(494, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(523, 1000);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(523, 1000);  delay(20);  }  // 3 posmo natu seka  void Anthem\_part\_3()  {  rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(587, 1000);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(523, 250);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(494, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(523, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(587, 1000);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(659, 250);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(587, 500);  delay(20);    \_delay(0.3);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(698, 1000);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(659, 250);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(587, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(659, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(698, 1000);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(784, 250);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(698, 500);  delay(20);    \_delay(0.3);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(659, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(880, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(784, 500);  delay(20);    \_delay(0.3);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(523, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(698, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(659, 500);  delay(20);    \_delay(0.3);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(494, 1000);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(698, 250);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(659, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(523, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(494, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(440, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(494, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(523, 500);  delay(20);  }  // 4 posmo natu seka  void Anthem\_part\_4()  {  rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(587, 1000);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(523, 250);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(494, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(523, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(587, 1000);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(659, 250);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(587, 500);  delay(20);    \_delay(0.3);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(698, 1000);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(659, 250);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(587, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(659, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(698, 1000);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(784, 250);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(698, 500);  delay(20);    \_delay(0.3);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(659, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(880, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(784, 500);  delay(20);    \_delay(0.3);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(523, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(698, 500);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(659, 500);  delay(20);    \_delay(0.3);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(494, 1000);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(698, 250);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(659, 500);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(523, 500);  delay(20);    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    buzzer.tone(659, 1000);  delay(20);    rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();    buzzer.tone(587, 1000);  delay(20);    rgbled\_7.setColor(0,255,255,0);  rgbled\_7.show();    buzzer.tone(523, 2000);  delay(20);  }  // Linijos sekimo algoritmas  void Line\_Tracer()  {  Speed = 200;    rgbled\_7.setColor(0,0,255,0);  rgbled\_7.show();    if((ultrasonic\_3.distanceCm()) < (10)){  move(1,0);  rgbled\_7.setColor(0,255,0,0);  rgbled\_7.show();  buzzer.tone(988, 250);  delay(20);  buzzer.tone(988, 250);  delay(20);  move(3,100);  \_delay(0.5);  while(!((linefollower\_2.readSensors()) < (3)))  {  \_loop();  motor\_9.run((9)==M1?-Speed:Speed);  motor\_10.run((10)==M1?-(Speed-80):(Speed-80));  }  }else{  if(((linefollower\_2.readSensors())==0)){  move(1,Speed/2);  }else{  if(((linefollower\_2.readSensors())==1)){  move(3,100);  Find\_Path\_Turn\_Direction = 0;  }else{  if(((linefollower\_2.readSensors())==2)){  move(4,100);  Find\_Path\_Turn\_Direction = 1;  }else{  if(((Find\_Path\_Turn\_Direction)==0)){  move(3,100);  }else{  move(4,100);  }  }  }  }  }    }  // Judejimo funkcija  // Kryptys:  // 1 - i prieki  // 2 - atgal  // 3 - i kaire  // 4 - i dešine  void move(int direction, int speed)  {  int leftSpeed = 0;  int rightSpeed = 0;  if(direction == 1){  leftSpeed = speed;  rightSpeed = speed;  }else if(direction == 2){  leftSpeed = -speed;  rightSpeed = -speed;  }else if(direction == 3){  leftSpeed = -speed;  rightSpeed = speed;  }else if(direction == 4){  leftSpeed = speed;  rightSpeed = -speed;  }  motor\_9.run((9)==M1?-(leftSpeed):(leftSpeed));  motor\_10.run((10)==M1?-(rightSpeed):(rightSpeed));  } |

# Roboto valdymo eksperimentinis tyrimas

Kadangi robotui buvo iškeltos 4 skirtingos užduotys, todėl jų išpildymas buvo tikrinamas atskirai:

1. Roboto valdymas pultelio – robotas padedamas ant lygaus paviršiaus (grindų) ir valdymo pultelio buvo spaudžiamas kiekvienas pultelio mygtukas ir tikrinama roboto reakciją į tai. Žiūrint į veiklos diagramą, buvo tikrinama, ar robotas reaguoja tik į tuos mygtukus, kurie buvo aprašyti kode. Po bandymo nustatyta, kad robotas reaguoja į visus aprašytus pultelio mygtukus taip, kaip ir buvo numatyta.
2. Roboto autonominis judėjimas – robotas padedamas ant lygaus paviršiaus ir pulteliu parenkamas autonominio judėjimo rėžimas (B mygtukas). Robotas pradeda judėti tiesiai ir, priešaky pamatęs kliūti, 90 laipsnių kampu pasisuka į dešinę arba kairę (atsitiktinai, kaip ir buvo numatyta bei aprašyta kode). Eksperimento metu buvo pastebėta netikslumų – kai kliūtis yra pakrypusi kampu roboto sensoriaus atžvilgiu, robotas dažnai tos kliūties nefiksuoja ir į ją atsitrenkia arba kliūtį kabina korpuso dalimi.
3. Linijos sekimas – pirmiausia robotas buvo padėtas ant mBot pakete pridėtos trasos – ją robotas įveikė be vargo. Vėliau buvo sukurta nauja trasa, su smailesniais kampais – ją robotas taip pat įveikė (tiek judant į vieną pusę, tiek į kitą). Tiesa, buvo pastebėta, kad trasos plotis yra kiek per mažas, nes kartais robotas tiesiame kelyje kiek mėtosi iš kairės trasos kelios pusės į dešinę, o kai reikia įveikti smailų kampą – sukasi ne į tą pusę ir grįžta trasa atgal posūkio neįveikęs. Taip atsitinka dėl to, kad kode yra saugoma paskutinė roboto sensoriaus sukimosi reikšmė (ar robotas paskutinį posūkį matė kairėje, ar dešinėje pusėje). Kai robotas juda siauresne trasa, ši reikšmė kartais kinta neprognozuojamai ir robotą suklaidina, kai jis neberanda trasos ir sukasi ne į tą pusę. Taip pat trasoje buvo padėta kliūtis (pažymėta X), ją robotas apvažiavo lanku ir grįžo į trasą kaip numatyta. Tiesa, buvo pastebėta, kad jei kliūtis yra kiek per plati, robotas, bandydamas ją apvažiuoti, kliudo.
4. Himno grojimas – tiesiog buvo paleistas himno grojimo rėžimas ir išklausyta visa melodija. Eksperimento metu paaiškėjo, kad muzika skamba melodingai, gana tiksliai, o šviesos keičiasi taip, kaip ir reikia (pastoviai keičiasi geltona – žalia – raudona).

# Išvados

**Mangirdas:** projektinio darbo metu buvo išsiaiškinta „mBot“ roboto struktūrą, detalės, buvo įdomu sukonstruoti patį robotą bei sužinoti, kaip veikia konkretūs sensoriai, kaip jiems aprašomas programinis kodas. Projekto tikslas, remiantis eksperimentiniu rezultatu yra įgyvendintas, robotas veikia tinkamai. Pats buvau atsakingas už himno melodijos aprašymą programiniu kodu bei linijos sekimo algoritmą – manau, kad užduotis įgyvendinti pavyko. Sunkiausia projekto dalis man – išsiaiškinti Makeblock bibliotekos veikimą bei susirasti tinkamas šios bibliotekos funkcijas, kurios buvo naudojamos mūsų projekte.

**Simonas:** kaip matome iš rezultatų bei patiems išmėginus robotą, robotas veikai tinkamai, reaguoja į IR pultelio siunčiamus signalus, veikia taip, kaip aprašyta veiklos diagramomis. Mano indėlis į projektą – roboto valdymas pulteliu bei autonominio judėjimo algoritmas, aišku, kartu ir šių algoritmų veiklos diagramų braižymas bei atitinkamos ataskaitos dalys. Nors šie algoritmai ir yra aprašyti pagal nutylėjimą pačiame „mBot Core“ modulyje, tačiau pats juos realizavau iš naujo, taip išsiaiškinant, kaip kodu aprašomi roboto sensoriai, IR pultelio siunčiami signalai bei roboto judėjimas.

# Naudota literatūra

1. <http://forum.makeblock.com/t/how-can-i-program-mbot-using-arduino-ide-instead-of-mblock/5401/2>
2. <https://github.com/Makeblock-official/Makeblock-Libraries>
3. <http://store.makeblock.com/product/mbot-robot-kit>
4. <http://learn.makeblock.com/en/robot-kits/mbot/the-line-follower>
5. <https://www.arduino.cc/en/Guide/HomePage>