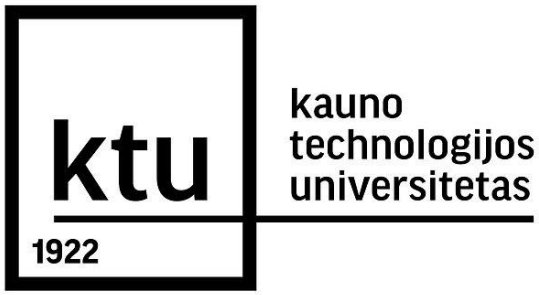


INFORMATIKOS FAKULTETAS

T125B114 Robotų programavimo technologijos

Projekto ataskaita

„Robotas šuo ir robotas šeimininkas“

Studentai: Martynas Veikutis IFF-6/14
Ernestas Milius IFF-6/14
Nerijus Dulkė IFF-6/11

Darbą priėmė: doc. Lina Narbutaitė
doc. Rasa Brūzgienė

KAUNAS 2019

Turinys

1. Darbo užduotis ir reikalavimų analizė

Projekto užduotis – panaudojant kuo įvairesnes „mBot“ roboto galimybes suprogramuoti roboto valdymo programą, kurią vykdydamas robotas – šuo gebėtų reaguoti į kito roboto – šeimininko komandas bei išsiuntus komandą robotas-šuo parneštų ant linijos padėtą kamuoliuką robotui – šeimininkui.

Užduoties įgyvendinimui keliami reikalavimai:

1) Robotas – šuo negali grįžti be kamuoliuko.
2) Robotas – šuo negali nuvažiuoti nuo linijos, išskyrus tuo atveju, kada robotas – šuo apvažiuoja kamuoliuką, jog galėtų jį „paimti“.
3) Robotas – šuo reaguoja į roboto – šeimininko komandas.
4) Robotas – šuo baigia vykdyti programą, kuomet gauna pranešimą iš roboto – šeimininko, jog gavo kamuoliuką.

Toliau pateikiama iškeltų reikalavimų analizė:

Nr.1	Šiam reikalavimui įgyvendinti planuojama panaudoti mBot roboto linijos sekimo modulį.
Nr.2	Robotas ieškos kamuoliuko naudodamas ultragarso sensorių.
Nr.3	Robotai savo programoje turės sąlygas su žinutės priėmimu, kurios padės įvertinti, ar jau reikia užbaigti programos vykdymą.
Nr.4	Apie programos vykdymo pabaigą (ir kitus vykdymo etapus) robotai praneš naudodamas mBot šviesos išvesčių modulius.

lentelė 1 Projekto užduoties reikalavimų analizė

2. Roboto veikimo scenarijus

Šiame skyriuje pateikiamas galimas roboto – šeimininko ir roboto – šuns užduočių scenarijus.

1. Robotas – šeimininkas valdomas pulteliu ir darbą pradeda paspaudus mygtuką „B“.
2. Robotas – šeimininkas savo šviesas nudažo žaliai ir išsiunčia žinutę su tekstu „fetch“ tam, kad robotas – šuo pradėtų vykdyti programą.
3. Robotas – šeimininkas laukia, kol jo matymo zonoje pasirodys robotas – šuo, kuomet robotas – šuo priartėja reikiamu atstumu, robotas – šeimininkas supranta, jog kamuoliukas atneštas, todėl išsiunčia pranešimą su žinute „good boy“, perjungia šviesas į geltoną spalvą ir baigia programos vykdymą.
4. Robotas – šuo valdomas pulteliu ir darbą pradeda paspaudus mygtuką „B“.
5. Robotas – šuo gavęs žinutę su tekstu „fetch“ iš roboto – šeimininko pradeda programos vykdymą.
6. Robotas – šuo apsisuka 180 laipsnių kampu ir pradeda važiuoti linija.

7. Robotas – šuo pastebėjęs kamuoliuką jį apvažiuoja ir apsisuka.
8. Robotas – šuo neša kamuoliuką tol, kol negauna žinutės iš šeimininko su tekstu „good boy“.
9. Robotas – šuo gavęs žinutę su tekstu „good boy“ nudažo savo spalvas žaliai ir baigia darbą.

3. Roboto aprašymas

„mBot“ – edukacinės paskirties robotas, kuris skirtas pradedantiesiems robotų programuotojams. Robotas yra sudarytas iš atskirų modulių ir dalių, todėl yra lengvai modifikuojamas. „mBot“ programuotojas gali pats pasirinkti, kokias detales nori naudoti pagal tai, kokios funkcijos, įvestys ir išvestys reikalingos konkrečiai užduočiai.

Šiame projekte naudojamas mbot yra sudarytas iš tokių pagrindinių modulių:

pav 1 Projekte naudojamo mBot roboto struktūra

1 – linijos sekimo/atpažinimo modulis, kuris dviem sensoriais atpažįsta tamsias linijas.

2 – ultragarso sensoriaus modulis, kuris leidžia robotui „matyti“ priekyje esančias kliūtis, modulis susideda iš ultragarso signalo siųstuvo ir imtuvo.

3 – „Servo“ tipo motorai, kurie leidžia kontroliuoti roboto ratų judėjimą.

4 - „mCore“ pagrindinis modulis, valdymo plokštė, kuri yra programuojama ir sujungia visus sensorius ir įrenginius robote. Plokštelėje taip pat yra šviesos ir IR sensoriai, garso ir LED išvesčių moduliai.

Į mBot roboto komplektaciją taip pat įeina 4xAA baterijų laikiklis, USB kabelis ir valdymo pultelis.

Apibendrinta mBot roboto specifikacija pateikta lentelėje:

Sensoriai	Šviesos sensorius, mygtukas, IR imtuvas, ultragarso sensorius, linijos sekimo modulis.
Kiti programuojami moduliai	Garso modulis, RGB LED šviesų modulis, IR siųstuvas, motorai.
Valdymo plokštės modelis	ATmega328
Energijos šaltinis	3.7V ličio baterija arba 4 1.5V AA baterijos
Išmatavimai	17 x 13 x 9 cm (Ilgis x Plotis x Aukštis)
Svoris	500g
Programavimo įrankiai	Arduino IDE arba mBlock
Belaidė komunikacija	Bluetooth arba 2.4G

lentelė 2 mBot roboto specifikacija

4. Roboto valdymo architektūra

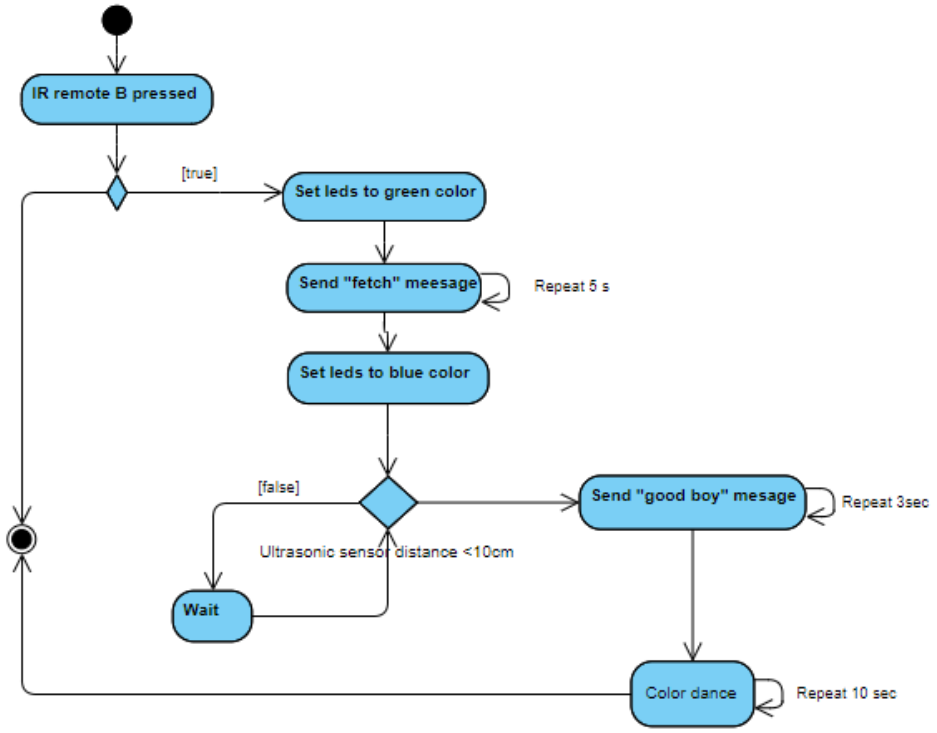
mBot Roboto valdymą galima aprašyti dviem būdais:

1) Naudojant mBlock grafinę aplinką.
2) Aprašant roboto valdymą C programiniu kodu.

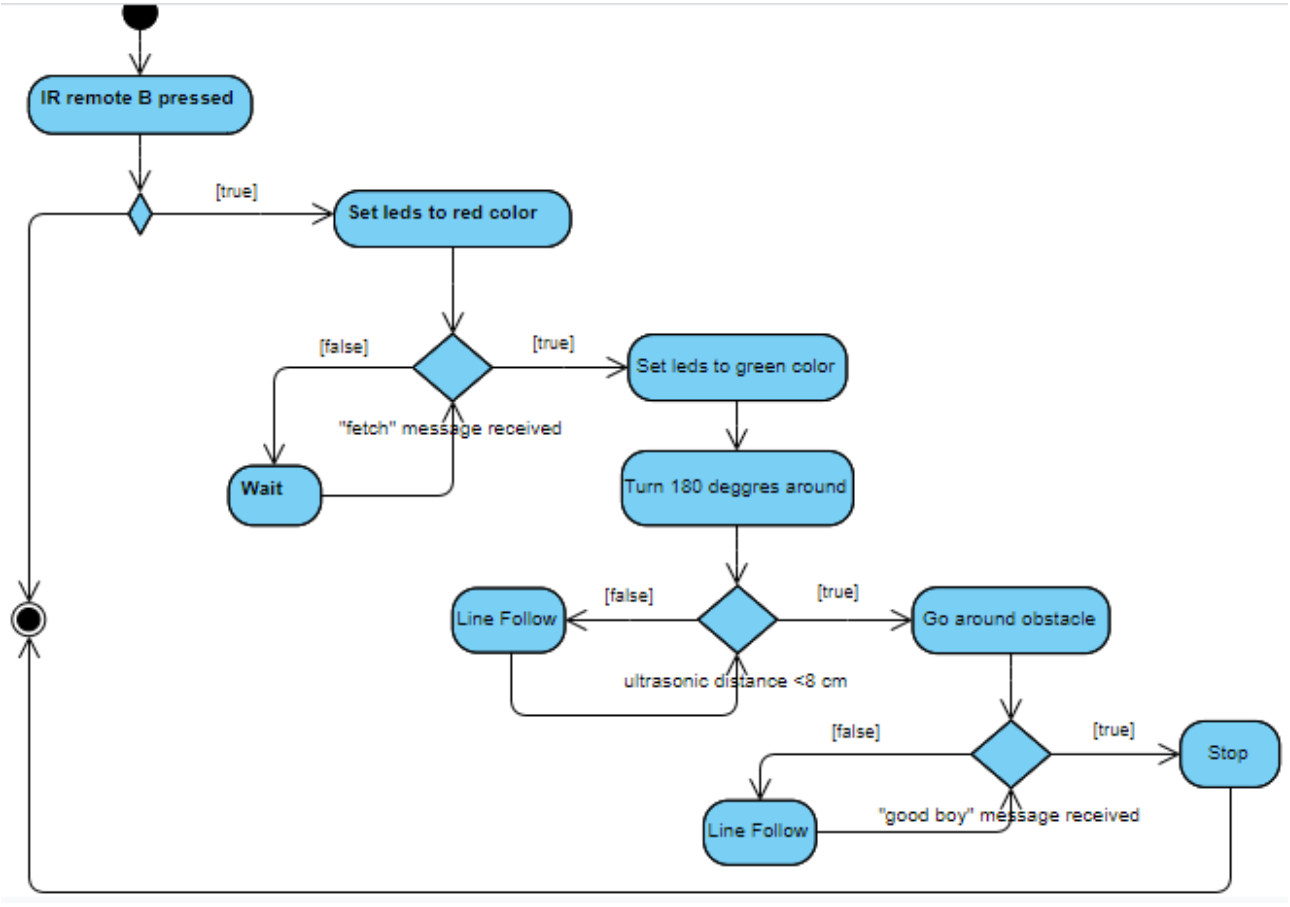
Šiame projektiniame darbe buvo panaudotas pirmasis būdas. Roboto valdymo programos konstravimas iš blokų grafinėje aplinkoje leidžia aiškiau matyti atskirus roboto veiksmus, būsenų perėjimus, identifikuoti klaidas ir įgyvendinti pakeitimus. Grafinė aplinka taip pat palengvina ir iš dalies automatizuoja roboto įvesčių sensorių ir išvesčių modulių valdymą.

Projektinio darbo metu naudota 3.4.12 mBlock versija.

5. Robotų valdymo algoritmas



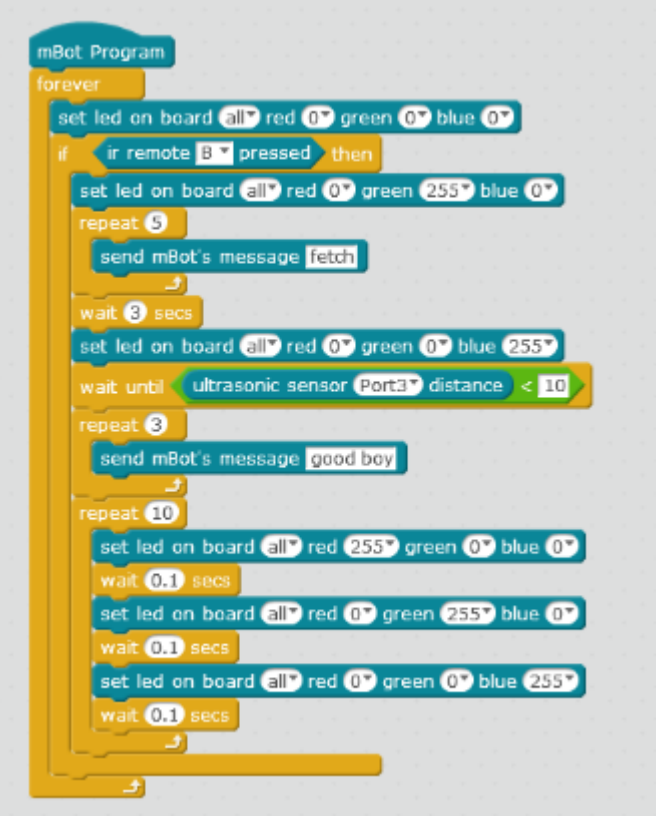
1 pav. Roboto – šeimininko algoritmas.



2 pav. roboto – šuniuko algoritmas.

6. Robotų valdymo algoritmai grafinėje aplinkoje

Šiame skyriuje pateikiamas robotų valdymo programos mBlock aplinkoje vaizdas:

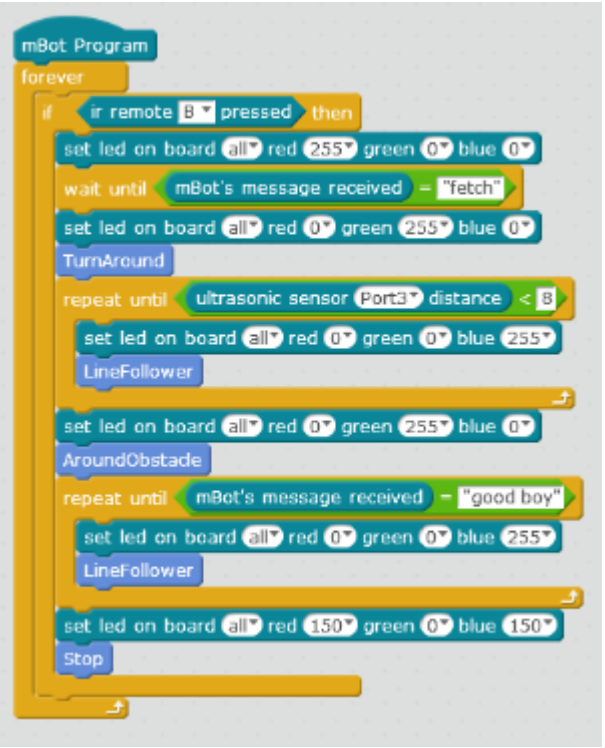


Programos vykdymas pradedamas paspaudus mygtuką „B“.

Siunčiama žinutė robotui – šuniui.

Jei robotas – robotas arčiau nei 10 cm, siunčiama žinutė robotui – šuniui, jog kamuoliukas gautas.

Darbo pabaigoje šviesų „šokis“.



Programos vykdymas pradedamas paspaudus mygtuką „B“.

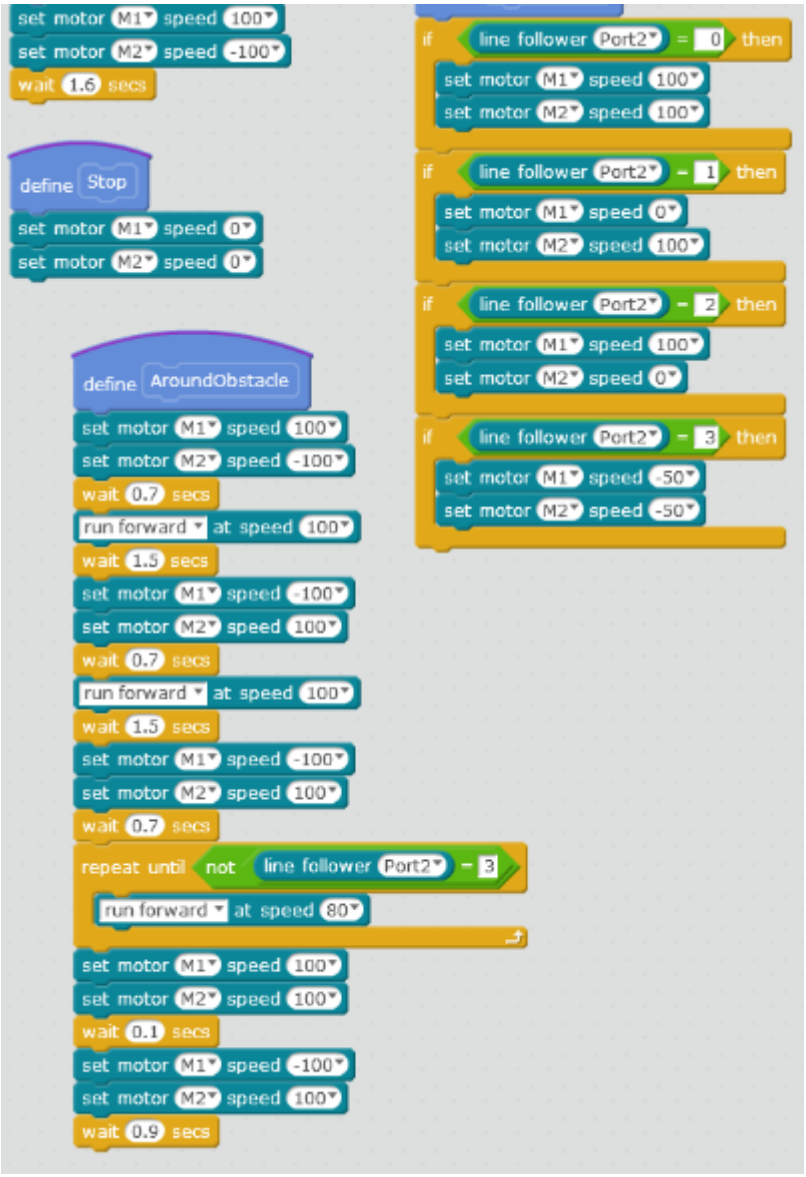
Robotas – šuo laukia, kol iš roboto – šeimininko gaus žinutę ir galės pradėti vykdyti komandą.

Robotas apsisuka, randa liniją ir važiuoja linija tol, kol kamuoliukas yra toliau nei 8cm.

Apvažiuoja kamuoliuką.

Stumia kamuoliuką tol, kol negauna žinutės iš roboto – šeimininko.

Baigus darbą sustoja ir nudažo spalvas geltonai.



Linijos sekimo algoritmas.

Kamuoliuko apvažiavimas.

Roboto sustojimas.

7. Roboto valdymo programos kodas

Šiame skyriuje pateikiamas automatiškai sugeneruotas robotų valdymo programos kodas.

- Pirmojo roboto siunčiančio signalus programinis kodas:

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

#include <MeMCore.h>

MeDCMotor motor_9(9);
MeDCMotor motor_10(10);

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));
}

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
MeRGBLed rgbled_7(7, 7==7?2:4);
MeIR ir;
MeUltrasonicSensor ultrasonic_3(3);



void setup(){
    ir.begin();

}

void loop(){

    rgbled_7.setColor(0,0,0,0);
    rgbled_7.show();
    if(ir.keyPressed(70)){
        rgbled_7.setColor(0,0,255,0);
        rgbled_7.show();
        for(int __i__=0;__i__<5;++__i__)
        {
```

```
            ir.sendString("fetch");
        }
        _delay(3);
        rgbled_7.setColor(0,0,0,255);
        rgbled_7.show();
        while(!((ultrasonic_3.distanceCm()) < (10)))
        {
            _loop();
        }
        for(int __i__=0;__i__<3;++__i__)
        {
            ir.sendString("good boy");
        }
        for(int __i__=0;__i__<10;++__i__)
        {
            rgbled_7.setColor(0,255,0,0);
            rgbled_7.show();
            _delay(0.1);
            rgbled_7.setColor(0,0,255,0);
            rgbled_7.show();
            _delay(0.1);
            rgbled_7.setColor(0,0,0,255);
            rgbled_7.show();
            _delay(0.1);
        }
    }

    _loop();
}

void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)_loop();
}

void _loop(){
    ir.loop();


}
```

Antrojo roboto „šuniuko“ programinis kodas:

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

#include <MeMCore.h>

MeDCMotor motor_9(9);
MeDCMotor motor_10(10);

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));
}

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
void LineFollower();
MeLineFollower linefollower_2(2);
void TurnAround();
void Stop();
void AroundObstacle();
MeIR ir;
MeRGBLed rgbled_7(7, 7==7?2:4);
MeUltrasonicSensor ultrasonic_3(3);


void LineFollower()
{
    if(((linefollower_2.readSensors())==(  0))){
        motor_9.run((9)==M1?-(100):(100));
        motor_10.run((10)==M1?-(100):(100));
    }

    if(((linefollower_2.readSensors())==( 1))){
        motor_9.run((9)==M1?-(0):(0));
        motor_10.run((10)==M1?-(100):(100));
    }

    if(((linefollower_2.readSensors())==( 2))){
        motor_9.run((9)==M1?-(100):(100));
        motor_10.run((10)==M1?-(0):(0));
    }

    if(((linefollower_2.readSensors())==( 3))){
        motor_9.run((9)==M1?-(-50):(-50));
        motor_10.run((10)==M1?-(-50):(-50));
    }

}

void TurnAround()
{
    motor_9.run((9)==M1?-(100):(100));

    motor_10.run((10)==M1?-(-100):(-100));

    _delay(1.6);

}

void Stop()
{
```

```
    motor_9.run((9)==M1?-(0):(0));

    motor_10.run((10)==M1?-(0):(0));

}

void AroundObstacle()
{
    motor_9.run((9)==M1?-(100):(100));

    motor_10.run((10)==M1?-(-100):(-100));

    _delay(0.7);

    move(1,100);

    _delay(1.5);

    motor_9.run((9)==M1?-(-100):(-100));

    motor_10.run((10)==M1?-(100):(100));

    _delay(0.7);

    move(1,100);

    _delay(3);

    motor_9.run((9)==M1?-(-100):(-100));

    motor_10.run((10)==M1?-(100):(100));

    _delay(0.7);

    while(!(!(((linefollower_2.readSensors())==(3)))))
    {
        _loop();
        move(1,80);
    }

    motor_9.run((9)==M1?-(100):(100));

    motor_10.run((10)==M1?-(100):(100));

    _delay(0.1);

    motor_9.run((9)==M1?-(-100):(-100));

    motor_10.run((10)==M1?-(100):(100));

    _delay(0.9);

    while(!(!(((linefollower_2.readSensors())==(3)))))
    {
        _loop();
        move(1,80);
    }

}
```

```
void setup(){
    ir.begin();

}

void loop(){

    if(ir.keyPressed(70)){
        rgbled_7.setColor(0,255,0,0);
        rgbled_7.show();
        while(!(((ir.getString())==("fetch"))))
        {
            _loop();
        }
        rgbled_7.setColor(0,0,255,0);
        rgbled_7.show();
        TurnAround();
        while(!((ultrasonic_3.distanceCm()) < (13)))
        {
            _loop();
            rgbled_7.setColor(0,0,0,255);
            rgbled_7.show();
            LineFollower();
        }
        rgbled_7.setColor(0,0,255,0);
        rgbled_7.show();
        AroundObstacle();
        while(!(((ir.getString())==("good boy"))))
        {
            _loop();
            rgbled_7.setColor(0,0,0,255);
            rgbled_7.show();
            LineFollower();
        }
        rgbled_7.setColor(0,150,0,150);
        rgbled_7.show();
        Stop();
    }

    _loop();
}

void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)_loop();
}

void _loop(){
    ir.loop();
}
```

8. Roboto veikimo rezultatai ir eksperimentinis tyrimas

Šiame skyrelyje pateikiamos įvairios dalinės išvados ir pastebėjimai apie robotų veikimą, sudaryti eksperimentuojant su robotų veikimu:

1) Robotas – šeimininkas, pasitelkdamas ultragarso sensorių, turi robotą – šunį aptikti nemažesniu nei 10 cm atstumu, priešingu atveju šeimininkas per vėlai išsiųs pranešimą robotui – šuniui ir robotas – šuo sustos gerokai vėliau negu reikia.

2) Norint, jog robotas suktųsi reikiamu kampu ir tiksliai reikalingos labai geros sąlygos: roboto baterijos naujos, ant ratų nėra purvo ar danga kuria važiuoja nėra slidi ar grublėta, priešingu atveju reikalingas didelis kiekis testavimui suderinti roboto pasisukimus, jog viskas vyktų tikslingai.

3) Papildomai prie roboto – šuns priekinės dalies pritaisėme atsikišimus, kurie padeda valdyti kamuoliuką.

4) Tam, kad linijos sekimo modulis veiktų užtikrintai, svarbu po robotu patiesti visiškai baltą dangą, pavyzdžiui popieriaus lapą. Važiuodamas ant grindų robotas kartais grindis įvertina kaip tamsią dangą ir ima veikti neprognozuojamai.

Apibendrinant robotų veikimo rezultatus galima teigti, kad robotai veikia sklandžiai ir sudarius tinkamas sąlygas sėkmingai vykdo jiems skirtas užduotis.

9. Išvados

Martynas: Projekto metu detaliau susipažinau su roboto modulių veikimais ir bendra roboto struktūra. Po daugelio testavimų ir programos kalibravimo pavyko įgyvendinti išsikeltą užduotį ir reikalavimus.

Sudėtingiausia projekto dalis buvo – suderinti visus modulius, jog užtikrintų tikslų apsisukimo kampą.

Projekto metu prisidėjau kurdamas roboto veikimo algoritmą. Taip pat nufilmavau vaizdo įrašą, kuriame pademonstruojamas robotų veikimas.

Video galima rasti: https://youtu.be/7_sSAWVTvwo

Nerijus: Projekto metu detaliau susipažinta su roboto sensoriais, t.y. kaip jie veikia, kaip juos panaudoti.

Mano indėlis – pagalba su roboto veikimo algoritmo sudarymu, pristatymo skaidrių paruošimas.

Ernestas: Projekto metu detaliau susipažinta su roboto sensoriais, kaip jie veikia, kaip juos panaudoti.

Mano indėlis į projektą – ataskaitos dalių užpildymas, pagalba sudarant veikimo algoritmą.

10. Literatūra

1) https://www.makeblock.com/steam-kits/mbot

2) https://github.com/makerobot/mBot/wiki/Line-following-and-obstacle-avoidance-with-mBot

3) https://www.mblock.cc/example/primary-line-patroling-program/