

**INFORMATIKOS FAKULTETAS**

**T125B114 Robotų programavimo technologijos**

**Projekto ataskaita**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | Darbą atliko:  Ernestas Venckus, IFF-4/3  Mantas Zambacevičius, IFF-4/1  Darbą priėmė:  Doc. Aurimas Jakštas  Lekt. Mindaugas Vasiljevas |

KAUNAS, 2017

Turinys

[1. Darbo užduotis 3](#_Toc501664041)

[2. Užduoties analizė 3](#_Toc501664042)

[3. Roboto aprašymas 3](#_Toc501664043)

[4. Roboto valdymo architektūra 4](#_Toc501664044)

[5. Roboto valdymo algoritmai 5](#_Toc501664045)

[6. Programos kodas 7](#_Toc501664046)

[7. Roboto valdymo eksperimentinis tyrimas 14](#_Toc501664047)

[8. Išvados 17](#_Toc501664048)

[9. Naudota literatūra 17](#_Toc501664049)

# Darbo užduotis

Projektinio darbo tikslas - suprogramuoti robotą, kuris gebėtų sekti liniją bei išvengti kelyje pasitaikiusių kliūčių. Robotas turi vykdyti linijos sekimą iki tol kol aptinka kliūtį, o apvažiavęs kliūtį robotas turi grįžti prie linijos sekimo algoritmo.

# Užduoties analizė

Užduočiai realizuoti bus naudojami 2 roboto jutikliai – linijos sekimo bei ultragarso. Reikia realizuoti algoritmą, kuris pagal tam tikrą susidariusią situaciją vykdytų linijos sekimą arba kliūčių išvengimą. Problemos sprendimui kiekvienos iteracijos metu bus skaitomi ultragarso sensoriaus duomenys.

# Roboto aprašymas

Projektiniam darbui atlikti buvo naudojamas Makeblock mBot v1.1 - Blue (Bluetooth version):



1 pav. Makeblock mBot v1.1 – Blue (Bluetooth version)

Roboto jutikliai:

* Šviesos jutiklis
* Infraraudonųjų spindulių jutiklis, skirtas nuotolinio valdymo pultelio veikimui
* Vienas mygtukas ant paties roboto
* Ultragarso jutiklis, skirtas atstumų matavimui
* Du linijos sekimo jutikliai
* Bluetooth sąsaja

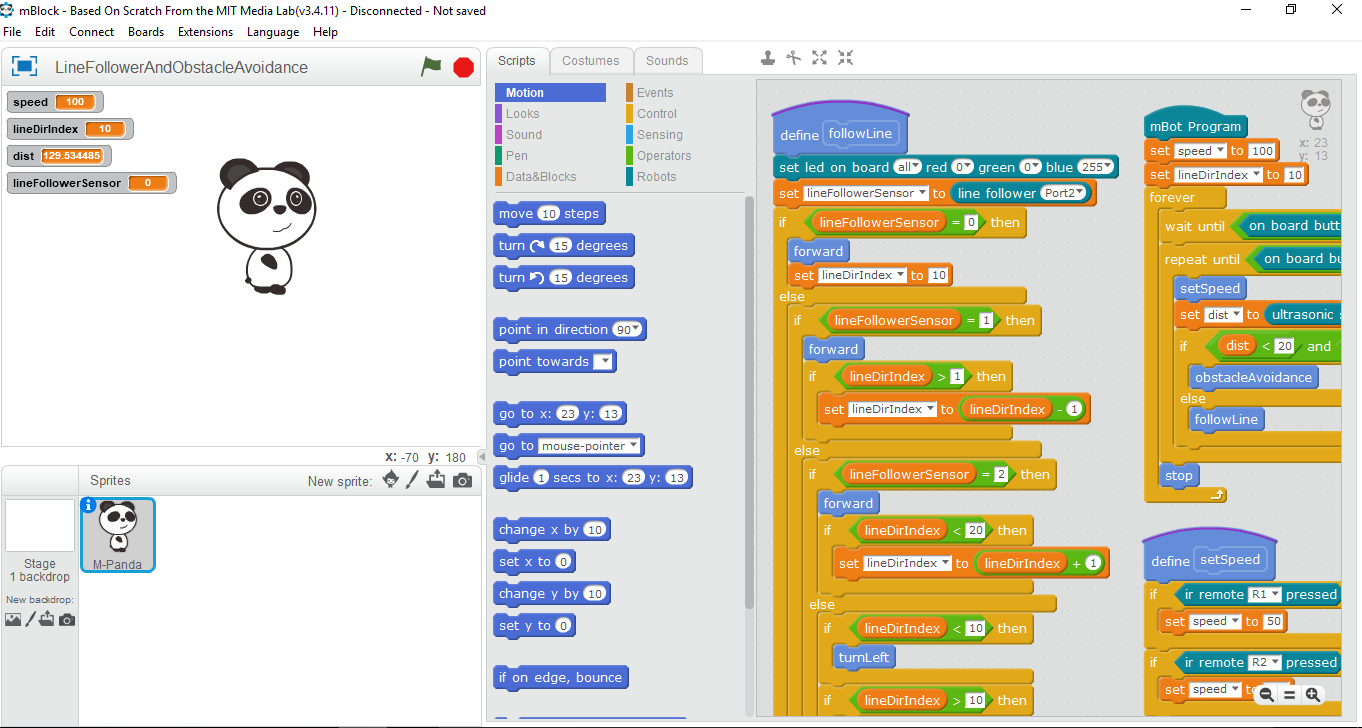
Robotas turi:

* 2 motorus, skirtus ratų judėjimui
* Garsiakalbį
* 2 RGB LED lemputes

# Roboto valdymo architektūra

Kadangi mBot mikro valdiklis paremtas Arduino Uno architektūra, robotas buvo programuojamas C kalba.

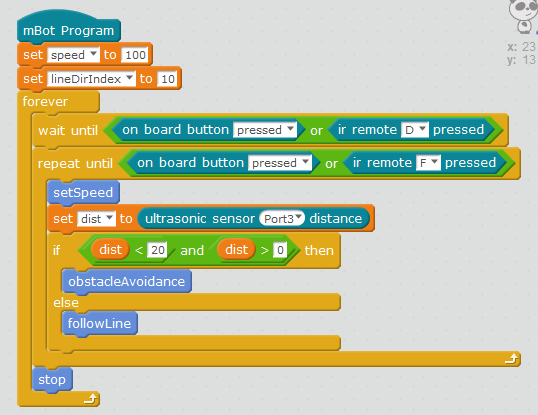
Roboto valdymui įgyvendinti taip pat naudojome „mBot“ programuoti skirtą IDE „mBlock“.



2 pav. "mBlock" IDE langas

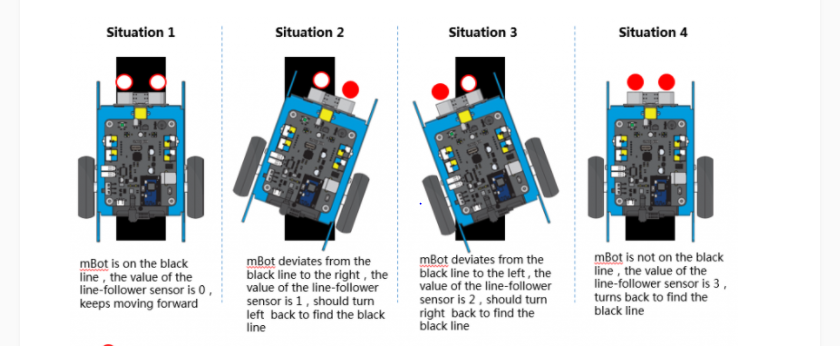
# Roboto valdymo algoritmai

Roboto valdymas susideda iš dviejų algoritmų – linijos sekimo bei kliūčių išvengimo. Jei ultragarsinio atstumo matuoklio reikšmė yra tarp 0 ir 20, laikome, kad kliūtis netoliese todėl reikia vykdyti kliūties išvengimą. Kitu atveju vykdome linijos sekimą. Žemiau pateikiamas „mBlock“ aplinkoje sudarytas pagrindinės programos pseudokodas:



3 pav. Pagrindinio algoritmo pseudokodas

Linijos sekimo algoritmas buvo sukurtas vadovaujantis „mBot“ linijos sekimo dokumentacija. Joje buvo aprašytos visos galimos situacijos bei sensoriaus grąžinama reikšmė.

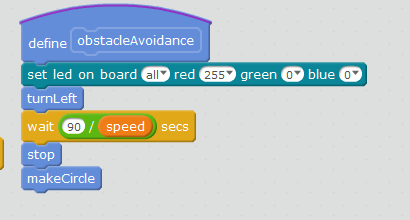


4 pav. Linijos sekimo galimos situacijos

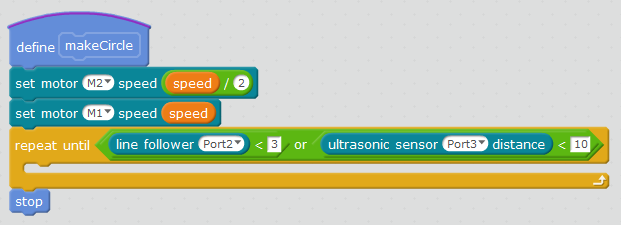


5 pav. Linijos sekimo algoritmo pseudokodas

Kliūčių išvengimo algoritmas veikia tokiu principu – robotas, pamatęs kliūtį, sukasi į kairę ir po tam tikro laiko važiuoja apskritimu, jei kliūtis vis dar yra, vėl kartojama tol, kol kliūties nebematoma ir važiuodamas apskritimu robotas grįžta į trasą.



6 pav. Kliūčių išvengimo algoritmas



7 pav. Važiavimo apskritimu algoritmas

# Programos kodas

Baigus darbą „mBlock“ aplinkoje, sugeneruojamas C kalbos kodas, kuris įkeliamas į „mBot“ robotą jo veikimo tyrimui. Žemiau pateikiamas gautas 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 obstacleAvoidance();

double speed;

MeRGBLed rgbled\_7(7, 7==7?2:4);

void followLine();

double lineFollowerSensor;

double lineDirIndex;

MeLineFollower linefollower\_2(2);

void turnRight();

void setSpeed();

MeIR ir;

void backward();

void stop();

void makeCircle();

MeUltrasonicSensor ultrasonic\_3(3);

void forward();

void turnLeft();

double dist;

void obstacleAvoidance()

{

rgbled\_7.setColor(0,255,0,0);

rgbled\_7.show();

turnLeft();

\_delay((90) / (speed));

stop();

makeCircle();

}

void followLine()

{

rgbled\_7.setColor(0,0,0,255);

rgbled\_7.show();

lineFollowerSensor = linefollower\_2.readSensors();

if(((lineFollowerSensor)==(0))){

forward();

lineDirIndex = 10;

}else{

if(((lineFollowerSensor)==(1))){

forward();

if((lineDirIndex) > (1)){

lineDirIndex = (lineDirIndex) - (1);

}

}else{

if(((lineFollowerSensor)==(2))){

forward();

if((lineDirIndex) < (20)){

lineDirIndex = (lineDirIndex) + (1);

}

}else{

if((lineDirIndex) < (10)){

turnLeft();

}

if((lineDirIndex) > (10)){

turnRight();

}

if(((lineDirIndex)==(10))){

backward();

}

}

}

}

}

void turnRight()

{

motor\_9.run((9)==M1?-(speed):(speed));

motor\_10.run((10)==M1?-((speed) / (10)):((speed) / (10)));

}

void setSpeed()

{

if(ir.keyPressed(12)){

speed = 50;

}

if(ir.keyPressed(24)){

speed = 100;

}

if(ir.keyPressed(94)){

speed = 150;

}

}

void backward()

{

motor\_9.run((9)==M1?-((0) - (speed)):((0) - (speed)));

motor\_10.run((10)==M1?-((0) - (speed)):((0) - (speed)));

}

void stop()

{

move(1,0);

rgbled\_7.setColor(0,0,0,0);

rgbled\_7.show();

}

void makeCircle()

{

motor\_10.run((10)==M1?-((speed) / (2)):((speed) / (2)));

motor\_9.run((9)==M1?-(speed):(speed));

while(!(((linefollower\_2.readSensors()) < (3)) || ((ultrasonic\_3.distanceCm()) < (10))))

{

\_loop();

}

stop();

}

void forward()

{

motor\_9.run((9)==M1?-(speed):(speed));

motor\_10.run((10)==M1?-(speed):(speed));

}

void turnLeft()

{

motor\_9.run((9)==M1?-((speed) / (10)):((speed) / (10)));

motor\_10.run((10)==M1?-(speed):(speed));

}

void setup(){

ir.begin();

speed = 100;

lineDirIndex = 10;

pinMode(A7,INPUT);

}

void loop(){

while(!(((0^(analogRead(A7)>10?0:1))) || (ir.keyPressed(68))))

{

\_loop();

}

while(!(((0^(analogRead(A7)>10?0:1))) || (ir.keyPressed(13))))

{

\_loop();

setSpeed();

dist = ultrasonic\_3.distanceCm();

if(((dist) < (10)) && ((dist) > (0))){

obstacleAvoidance();

}else{

followLine();

}

}

stop();

\_loop();

}

void \_delay(float seconds){

long endTime = millis() + seconds \* 1000;

while(millis() < endTime)\_loop();

}

void \_loop(){

ir.loop();

}

# Roboto valdymo eksperimentinis tyrimas

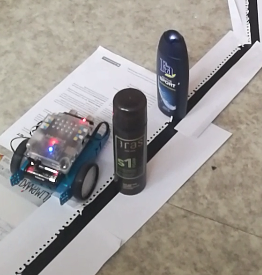
Kad būtų lengviau suprasti, ką robotas atlieka kiekvienu metu, nurodėme, kad roboto LED lemputės degtų mėlynai (8pav.), kai jis atlieka linijos sekimo algoritmą, o aptikus kliūtį, lemputės užsidegtų raudonai (9 pav.). Atlikdamas važiavimo apskritimo algoritmą, robotas išjungia LED lemputes (10 pav). Žemiau pateikiami realaus bandymo vaizdai:



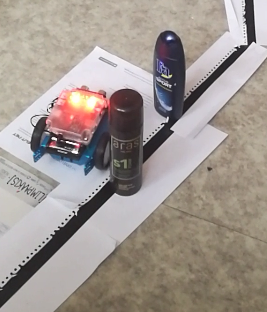
8 pav. Robotas atlieka linijos sekimą



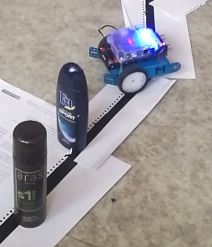
9 pav. Robotas aptinka kliūtį ir atlieka posūkį kairėn



10 pav. Robotas važiuoja apskritimu



11 pav. Robotas aptinka antrą kliūtį ir kartoja kliūties išvengimo veiksmus



12 pav. Robotas, išvengęs kliūčių, grįžta į trasą ir toliau seka liniją

Pagal mūsų naudotą kliūties vengimo algoritmą, robotas gali apvažiuoti kliūtis kurių plotis neviršija 7 centimetrų, o ilgis – 10 centimetrų. Norint sumažinti šį apribojimą, reikėtų padidinti važiuojamo apskritimo spindulį, tačiau tai padidintų nukrypimą nuo trasos ir sukeltų kitų problemų. Roboto veikimas buvo išbandytas įvairiais greičiais. Geriausi rezultatai buvo gauti, kai roboto greitis buvo nustatytas ties 100. Robotui nustačius greitį į 50, jis ties posūkiais visiškai sustoja. Tam yra dvi galimos priežastys:

* Robotas, atlikdamas posūkį, sulėtina vieną iš ratų ir kai jam reikia vėl išlyginti ratų greitį, jam galimai neužtenka jėgos tam padaryti ir galiausiai jo abu ratai sustoja;
* Roboto baterijos pasiekė per žemą lygį tokio mažo greičio palaikymui.

Išbandžius robotą su greičiais nuo 200, jis pajudėdamas iš vietos dažnai pakeldavo savo priekį aukštyn, taip pamesdamas liniją arba nepamatydamas kliūties, todėl šie greičiai buvo per dideli.

Taip pat geri rezultatai gauti ties greičiu, lygiu 150, tačiau tada robotas kartais pastumdavo kai kurias nepritvirtintas trasos dalis, taip minimaliai iškraipydamas trasą, bet nevertinant to, ją vis tiek įveikia.

# Išvados

Mantas: Projektinio darbo metu buvo sėkmingai realizuotas patobulintas linijos sekimo algoritmas. Suprogramuotas robotas, sekdamas liniją, geba išvengti nedideles kliūtis. Algoritmo veikimas buvo ištestuotas skirtingais roboto greičiais, tokiu būdu buvo rastas optimalus roboto greitis. Prie projekto dirbome kartu, bet aš buvau atsakingas už kliūčių vengimo algoritmą bei pristatymo medžiagą.

Ernestas: Projekto metu buvo pirmą kartą dirbama su realiu robotu, kas sudarė tikrai gerą įspūdį. Darbui atlikti labai padėjo pirmo ir antro laboratorinių darbų medžiaga bei patirtis juos atliekant. Buvo susipažinta su „mBlock“ programavimo grafine aplinka, kuri supaprastina visą programavimo dalį, nes nereikalauja mokėti jokios programavimo kalbos, todėl tai yra labai geras įrankis mokyti vaikus robotų programavimo pagrindų. Turbūt didžiausia problema buvo roboto veikimo tyrimas, nes tam buvo reikalinga trasa, tačiau sugalvojome, kaip nebrangiai šią problemą išspręsti, todėl projektinis darbas buvo atliktas sėkmingai. Visą projektą darėme kartu toje pačioje patalpoje, bet mano pareiga buvo pasirūpinti linijos sekimo algoritmo kūrimu, trasa bei ataskaitos tvarkingumu ir turiniu.

# Naudota literatūra

<http://www.mblock.cc/example/primary-line-patroling-program/>

<https://www.allaboutcircuits.com/projects/how-to-build-a-robot-line-follower/>

<http://www.mblock.cc/example/avoid-barriers/>

Laboratorinių darbų 1 ir 2 medžiaga