KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

T125B114

ROBOTŲ PROGRAMAMVIMO TECHNOLOGIJOS

Projektinio darbo ataskaita

**Atliko:**

**Tomas Kašelynas IFF-7/5**

**Giedrius Rastauskas IFF-7/12**

**Lukas Žaromskis IFF-7/5**

**Priėmė:**

**Doc. Brūzgienė Rasa**

**Doc. Adomkus Tomas**

KAUNAS 2020

Turinys

[Užduotis 3](#_Toc58313685)

[Užduoties analizė 3](#_Toc58313686)

[Roboto aprašymas 3](#_Toc58313687)

[Roboto valdymo architektūra 4](#_Toc58313688)

[Roboto valdymo algoritmas 4](#_Toc58313689)

[Roboto modeliavimo rezultatai 5](#_Toc58313690)

[Roboto valdymo programa 6](#_Toc58313691)

[Roboto valdymo eksperimentinis tyrimas 7](#_Toc58313692)

[Išvados 7](#_Toc58313693)

[Naudota literatūra 8](#_Toc58313694)

# Užduotis

Robotas važiuoja link labirinto pradžios. Aptikęs kliūtis, jas apvažiuoja BUG0 algoritmu. Labirintą įveikia pagal dešinės rankos taisyklę. Toliau važiuoja link pabaigos taško ir kliūtis apvažiuoja BUG2 algoritmu.

# Užduoties analizė

Roboto pradinis taškas yra toliau nuo labirinto ir jis turi pasiekti jo pradžią. Važiuoti tiesiai robotui nepavyks, nes tarp jo ir labirinto pradžios bus kliūčių. Kai robotas pasieks labirintą, jis jį turės įveikti ir pasiekti jo pabaigą.

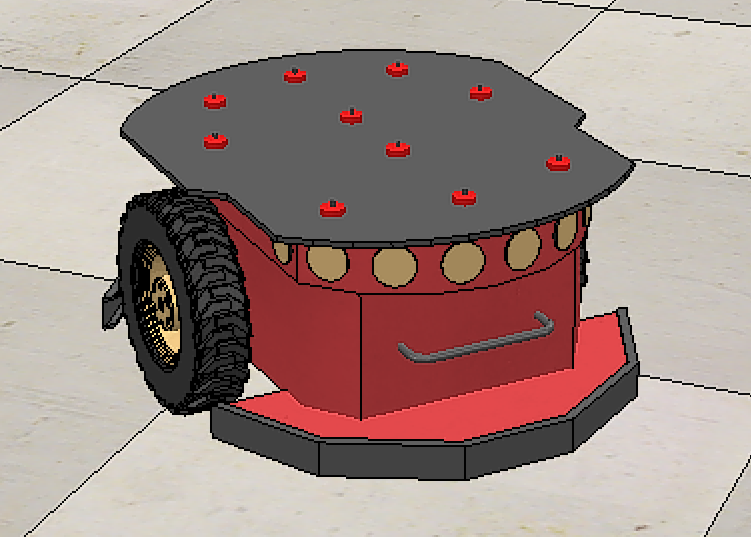
Reikalinga, kad robotas mokėtų apvažiuoti kliūtis ir įveikti labirintą pagal dešinės rankos taisyklę.

# Roboto aprašymas

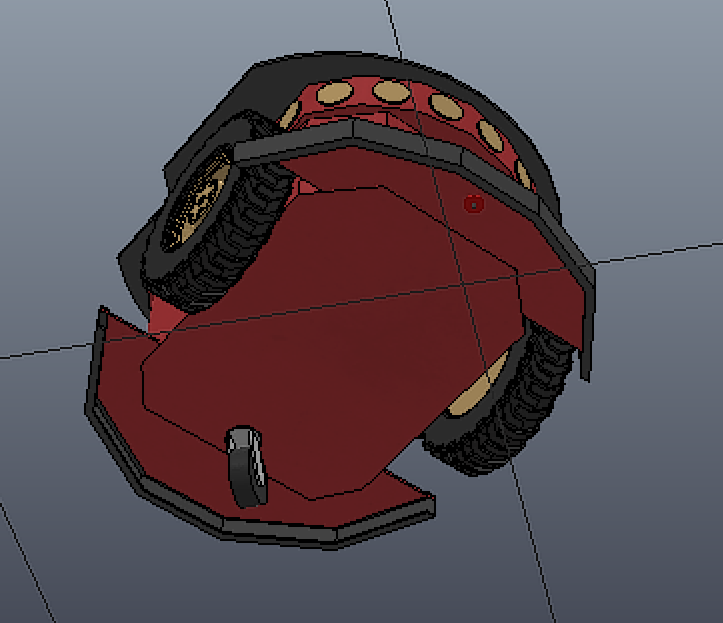
Robotas turi tris ratus: du pagrindiniai ratai su motorais ir vienas ratukas stabilizacijai. Taip pat robotas turi 16 ultragarsinių atstumo sensorių bei 11 jungčių.



1 pav. Roboto priekis



2 pav. Roboto galas



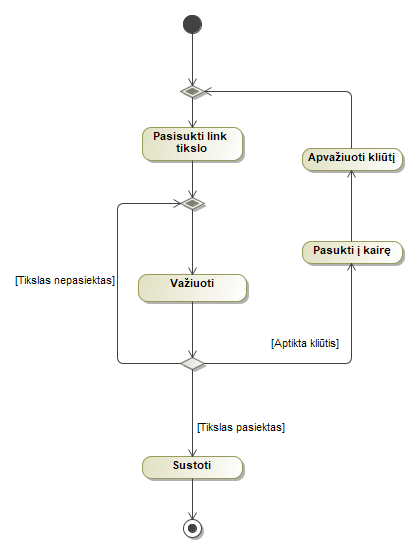
3 pav. Roboto apačia

# Roboto valdymo architektūra

Robotas valdomas siunčiant jam komandas per Python „sim“ API [1]. Galima nustatyti kairiojo arba dešiniojo rato sukimosi greičius, gauti informaciją iš sensorių.

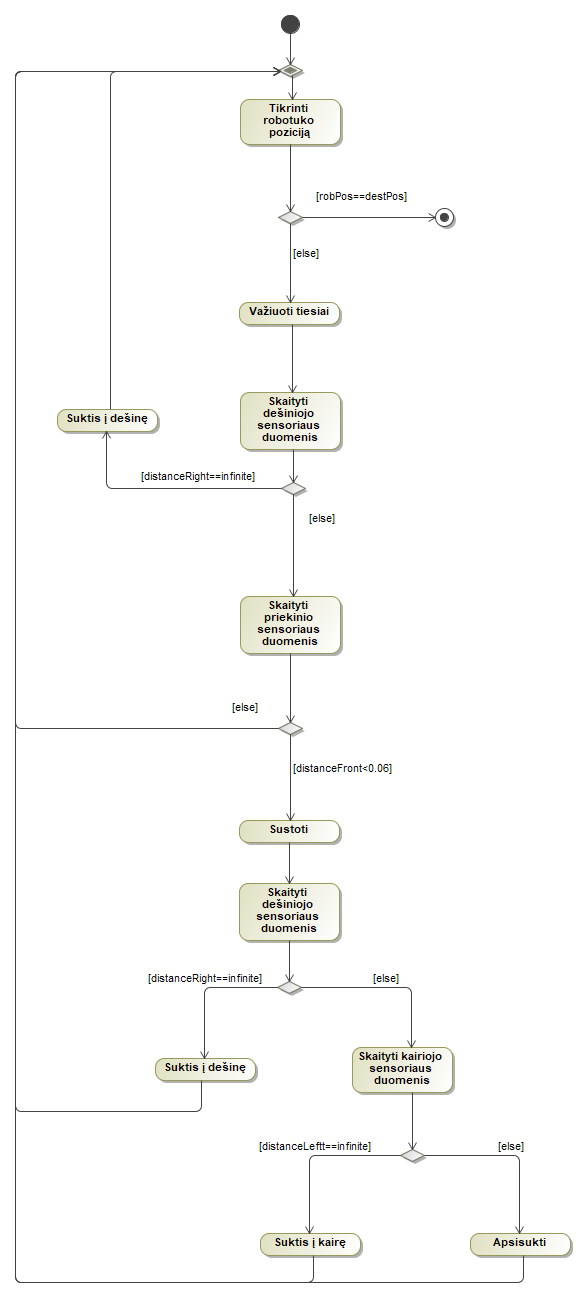
# Roboto valdymo algoritmas

Pirmąsias kliūtis robotas įveikia naudojant BUG0 algoritmą. Jis važiuoja tiesiai link tikslo pozicijos, o kai yra aptinkama kliūtis yra sukama į kairę ir kliūtis apvažiuojama. Toliau vėl yra važiuojama tiesiai link kliūties ir jei yra aptinkama kliūtis, yra kartojamas apvažiavimas.



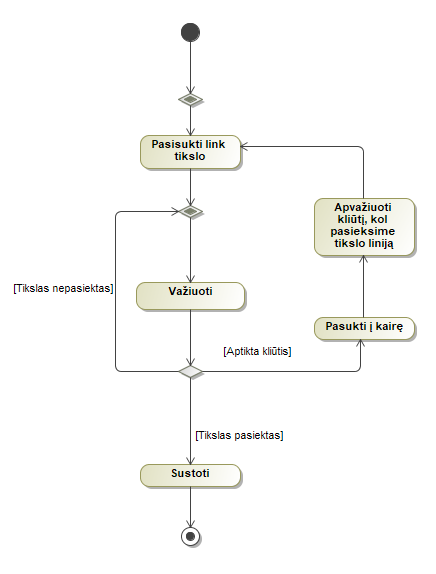
4 pav. BUG0 veiklos diagrama

Labirintą robotas įveikia naudojant dešinės rankos taisyklės algoritmą. Robotas važiuoja tiesiai tol, kol išvažiuoja iš labirinto. Aptikęs priešais sieną, pasižiūri, ar gali važiuoti į dešinę pusę. Jei gali, pasisuka į dešinę ir važiuoja tiesiai. Jei negali, tikrina kairę pusę. Jei ten nėra sienos, sukasi į kairę ir važiuoja tiesiai. Jei negali važiuoti nei į dešinę, nei į kairę, apsisuka ir važiuoja tiesiai. Jeigu važiuojant tiesiai aptinka, kad gali važiuoti į dešinę, sukasi į dešinę ir važiuoja tiesiai.



5 pav. MazeSolver veiklos diagrama

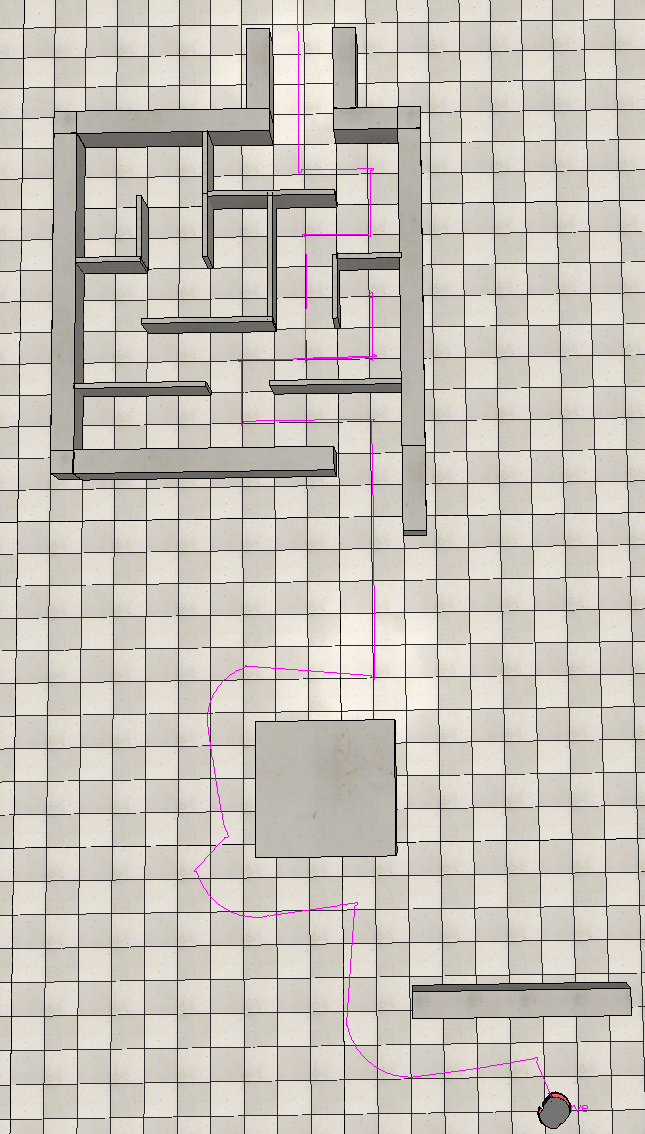
Įveikęs labirintą robotas toliau kliūtis apvažiuoja BUG2 algoritmu. Panašiai kaip ir BUG0 algoritmu pirmiausiai robotas važiuoja tiesiai link tikslo pozicijos. Aptikęs kliūtį ją apvažiuoja sukdamas į kairę ir važiuoja tol kol priartėja prie tikslo linijos (tikslo linija - nuo pradžios taško iki tikslo taško įsivaizduojama linija). Toliau vėl kartojamas algoritmas.



pav. 6 BUG2 veiklos diagrama

# Roboto modeliavimo rezultatai

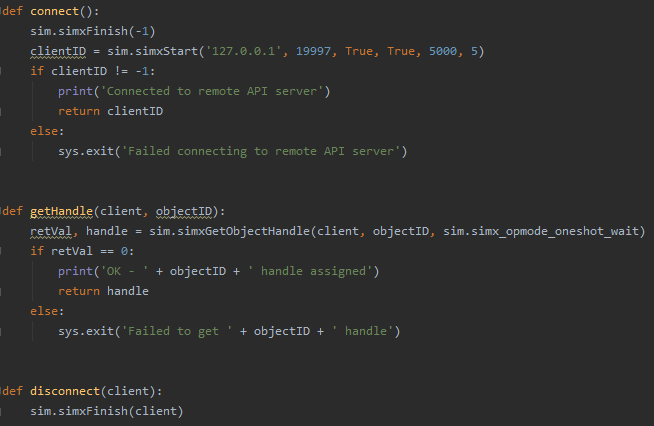
Simuliacijos rezultatas pateiktas 7 pav. Robotukas yra pradinėje pozicijoje. Rožinė linija rodo jo judėjimo trajektoriją. Jo trajektorija rodo, kad jis juda link tikslo tol, kol pasiekia kliūtį. Tada ją apvažiuoja ir bando toliau judėti link tikslo. Kartoja tol, kol pasiekia tikslą. Tada važiuoja link labirinto, kurį įveikia pagal dešinės rankos taisyklę – seka dešinę sieną. Įveikęs labirintą juda link galutinio tikslo, apvažiuoja likusias kliūtis ir sustoja.



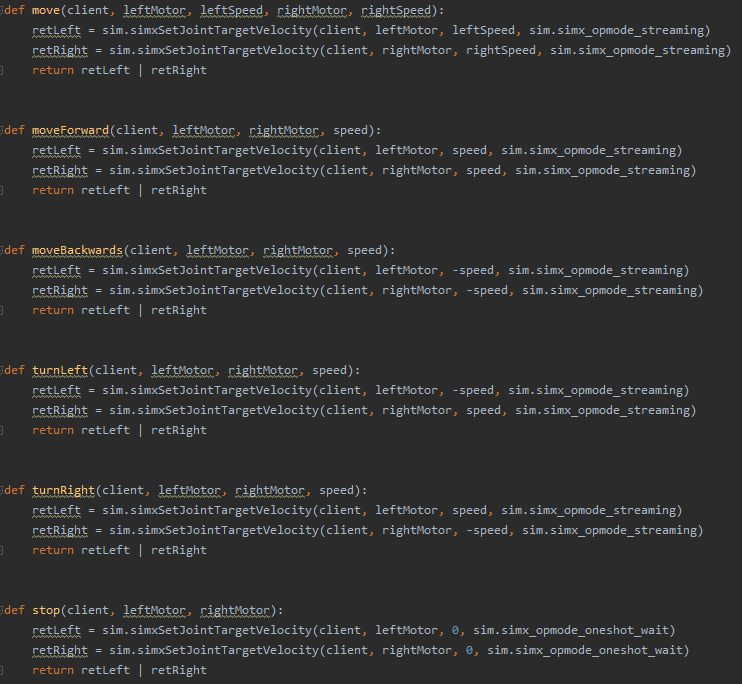
7 pav. Simuliacijos rezultatai

# Roboto valdymo programa

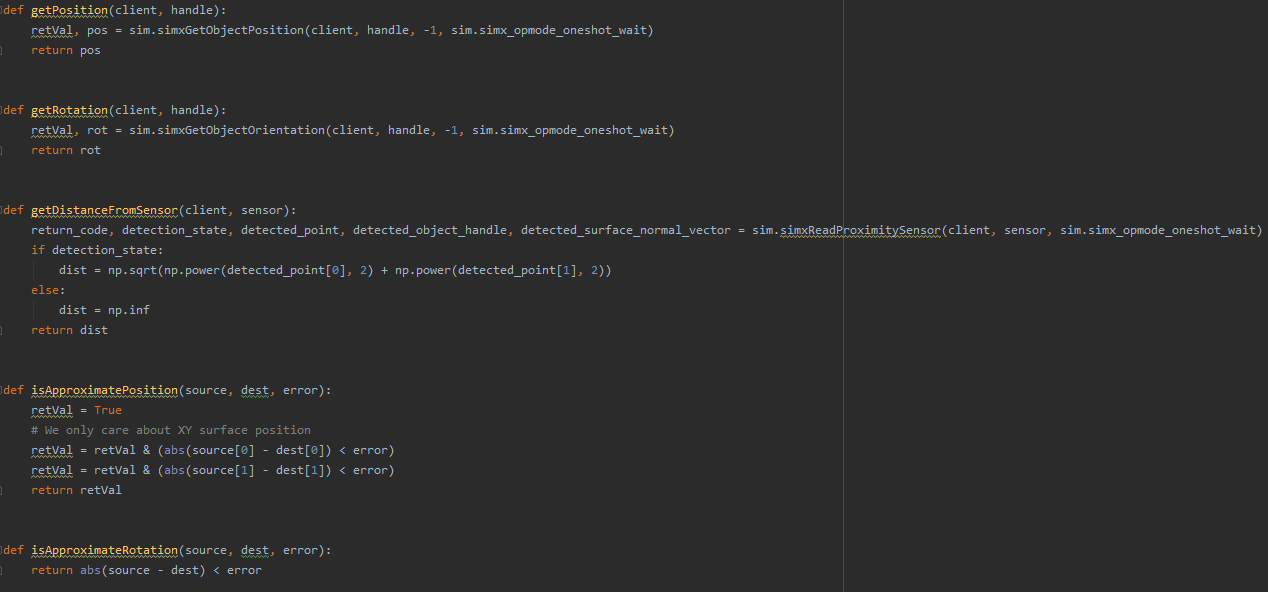
Vienos svarbiausių funkcijų (8 pav.) yra *connect*, kuri prisijungia prie serverio, *disconnect*, kuri atsijungia, bei *getHandle*, kuri grąžina valdiklio objektą. Dažniausiai naudojamos funkcijos yra susijusios su judėjimu (9 pav.). *moveForward* funkcija abiem motorams duoda teigiamo greičio, kad važiuotų į priekį, *moveBackwards* duoda neigiamo greičio, kad važiuotų atgal, *turnLeft* ir *turnRight* duoda priešingus greičius, kad robotukas suktųsi į atitinkamą pusę. *Stop* funkcija sustabdo robotuko judėjimą. Orientavimosi funkcijos (10 pav.) skirtos darbui su robotuko pozicija aplinkoje. *getPosition* gauna robotuko poziciją aplinkoje, *getRotation* gauna robotuko pasisukimą, *getDistanceFromSensor* grąžina atstumą iki kliūties, *isApproximatePosition* tikrina, ar robotukas yra tam tikroje vietoje su nustatyta paklaida, *isApproximateRotation* grąžina, ar robotukas yra pasisukęs tam tikru kampu pagal nustatytą paklaidą. *normalizeAngle* (11 pav.) normalizuoja duotą kampą. *rotateUntilAngle* (12 pav.) verčia robotuką suktis tol, kol pasiekiamas nurodytas kampas su tam tikra paklaida, o *getDesiredRotation* (12 pav.) gauna kampą, iki kurio reikia suktis. Šios funkcijos yra naudojamos *rotateTowards* funkcijoje (13 pav.), kuri suka robotuką link nurodyto taško. *moveForwardFor* funkcija (14 pav.) verčia robotuką judėti nurodytu greičiu nurodytą laiko tarpą. *wallFollowRHS* algoritmas (15 pav.) seka kliūtį, laikydamas sau iš dešinės. Ji naudojama *bug0* algoritme, kuris parodytas 16 pav. *turn90Degrees* funkcija (17 pav.) naudojama labirinte, kad robotukas suktųsi pagal ašis ir nenukryptų nuo sienų. Labirinto įveikimo algoritmas pagal dešinės rankos taisyklę pateiktas 18 pav. Pagrindinis kodas, kuris yra atsakingas už algoritmų valdymą, pateiktas 19 pav.



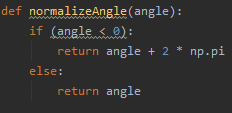
8 pav. Prisijungimo, atsijungimo ir valdiklio paėmimas



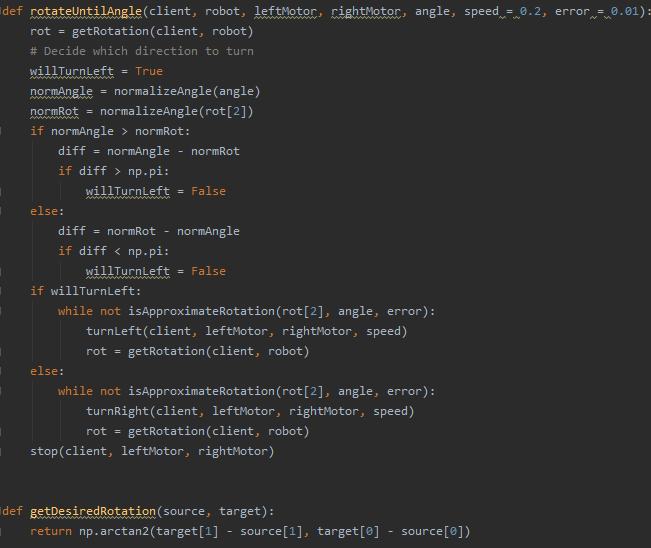
9 pav. Judėjimo funkcijos



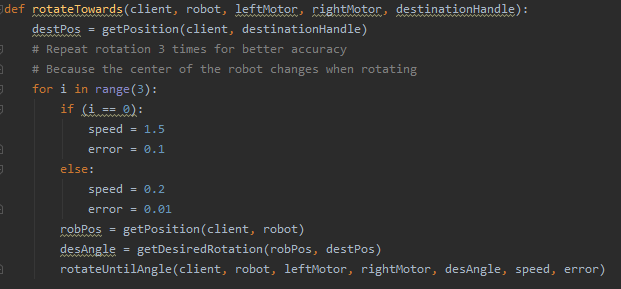
10 pav. Orientavimosi funkcijos



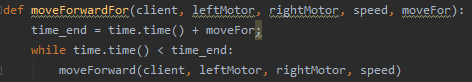
11 pav. Kampo normalizacija



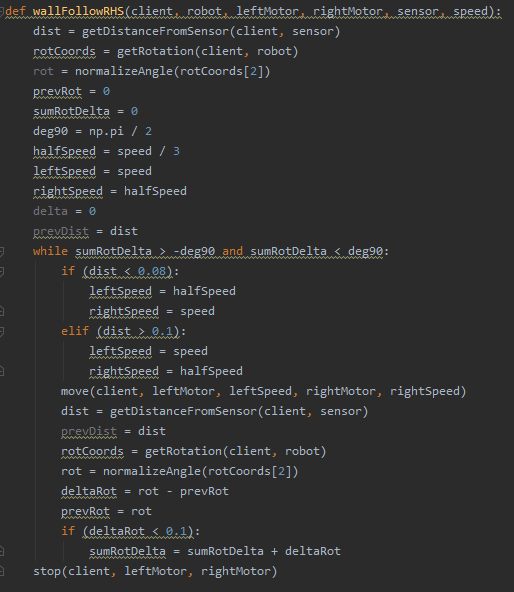
12 pav. Sukimosi iki tam tikro kampobei norimo kampo gavimo funkcijos



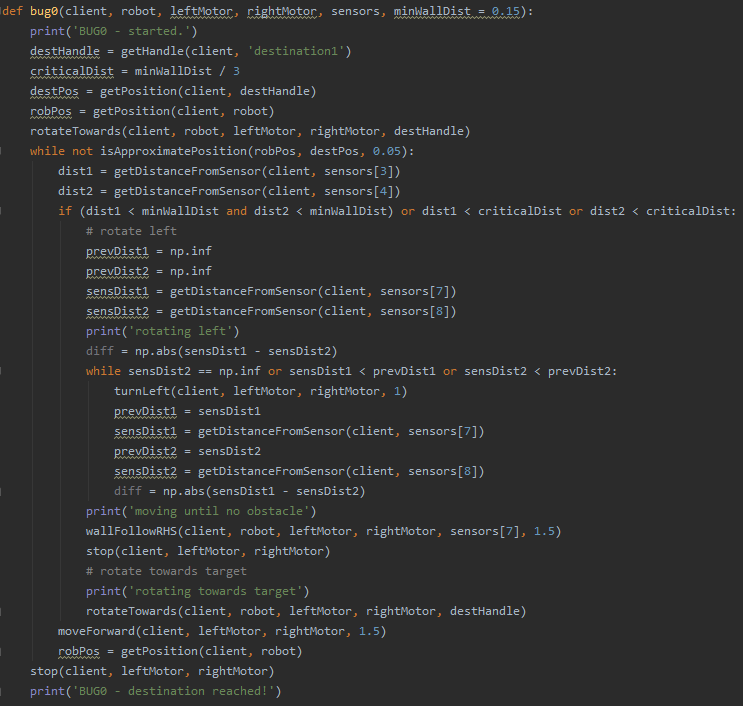
13 pav. Sukimosi link taško funkcija



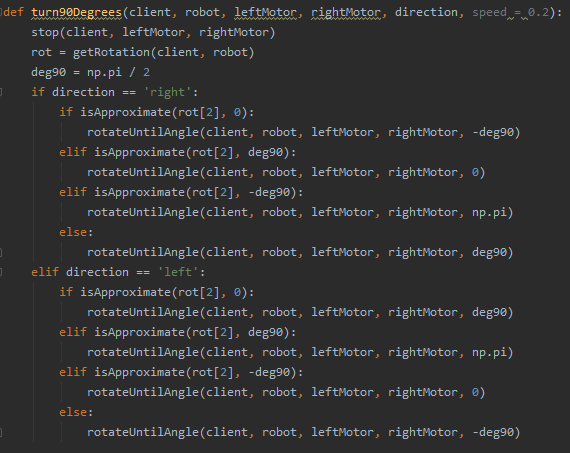
14 pav. Judėjimo nurodytą laiką funkcija



15 pav. Sienos sekimo algoritmas



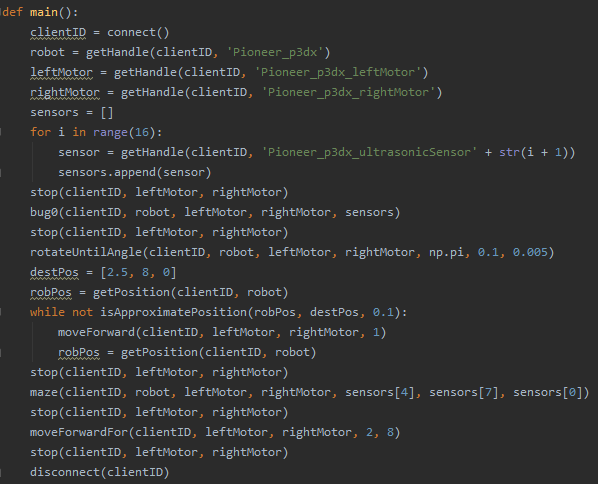
16 pav. bug0 algoritmas



17 pav. Pasisukimas 90 laipsnių kampu



18 pav. Labirinto algoritmas



19 pav. Pagrindinis kodas

# Roboto valdymo eksperimentinis tyrimas

Robotas gali įveikti bet kokį įveikiamą labirintą pagal dešinės rankos taisyklę, tačiau labirinto sienos turi būti lygiagrečios x ir y ašims, kad robotukas tiksliai pasisuktų 90 laipsnių kampu. Užduotis yra įveikiama šiek tiek greičiau nei per 8 minutes. Daugiausiai laiko užtrunka 90 laipsnių pasisukimai dėl duoto mažo sukimosi greičio, kad pasisukimai butų kuo tikslesni.

# Išvados

1. Tomo Kašelyno (labirinto įveikimas pagal dešinės rankos taisyklę, labirinto pastatymas simuliacijoje, testavimas, ataskaitos pildymas) išvados:
   1. Ne visus labirintus galima įveikti pagal dešinės rankos taisyklę.
   2. Svarbu užtikrinti, kad robotukas judėtų kuo lygiagrečiau labirinto sienoms, nes kitaip gali aptikti sieną ir pradėti suktis netinkamu laiku.
   3. Norint įveikti labirintą pagal dešinės rankos taisyklę, robotukas turi teikti prioritetą pasisukimams į dešinę.
   4. Aptikęs galimą pasisukimą į dešinę, robotukas turi iš karto suktis į dešinę.
2. Luko Žaromskio (bug0 algoritmo realizacija, 90 laipsnių pasisukimo realizacija, kliūčių pridėjimas iki labirinto, testavimas, ataskaitos pildymas) išvados:
3. Giedriaus Rastausko (bug2 algoritmo realizacija, kliūčių pridėjimas iki labirinto, testavimas, ataskaitos pildymas) išvados:
   1. Testuojant ant didesnio simuliacijos greičio buvo gaunami grubūs įverčiai, dėl ko buvo manoma, kad algoritmas veikia netinkamai.

# Naudota literatūra

|  |  |
| --- | --- |
| [1] | „Coppelia Robotics Remote API functions (Python),“ [Tinkle]. Available: https://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsPython.htm. [Kreiptasi 8 December 2020]. |