МИНОБРНАУКИ РОССИИ

Изображение выглядит как текст

Автоматически созданное описаниеФедеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра «Вычислительные системы и технологии»

ОТЧЁТ

по лабораторной работе №3

по дисциплине  
«Аппаратное и программное обеспечение роботизированных систем»

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гай В.Е.

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Перевалов А.Д.

Севастьянова Е.В.

Цветков Н.М.

19-В-1

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

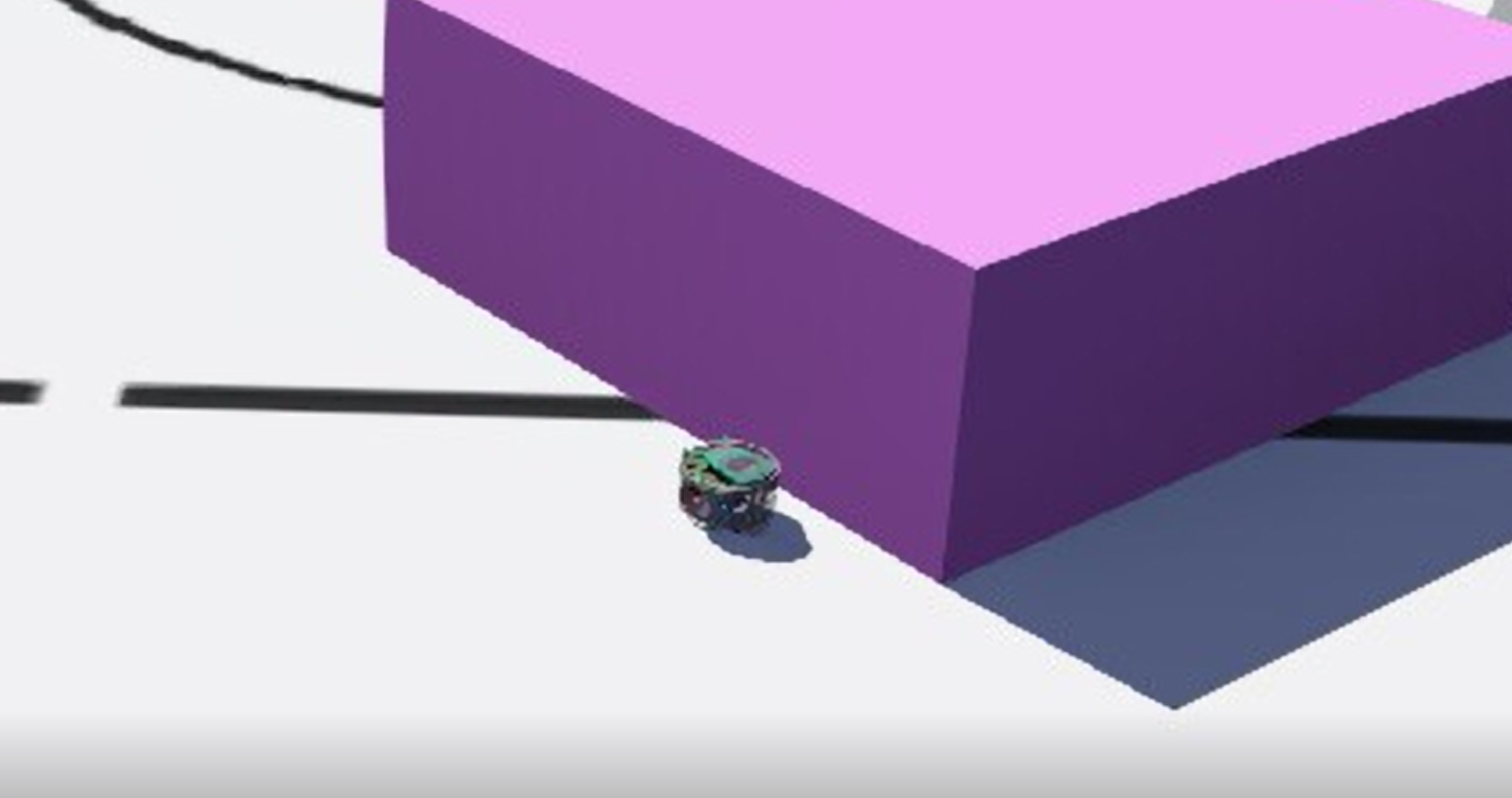
С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2022

**Цель:** получение навыков работы с алгоритмами управления роботами

**Задание**

**Движение по линии**: робот должен выполнить движение по линии, от начала до конца.

Изображение выглядит как текст

Автоматически созданное описание

**Описание алгоритма работы робота:**

Наша программа состоит из 3 функции и одной главной части, в которой будет присваиваться скорость и вызывать функции

**Список функций:**

1) **LineFollow()** – заставляет бота двигаться по линии таким образом:

1) Впереди под ботом есть датчик, определяющий черную линию. Он состоит из 3 микро датчиков (левый – [0], центральный –[1], правый-[2]). Эти датчики имеют своё значение (в зависимости от цвета пола под датчиком)

2) С помощью разности правого с левым датчика мы регулируем движение вперед с регулировкой нахождения на линии

3) Так как уменьшаемое это правый датчик, то это разность (умноженное на коэффициент, подобранный нами) будет суммировать к значению скорости правого мотора и будет вычитаться из значения скорости левого мотора, чтобы робот сместился к той стороне где значение черного цвета меньше (поэтому при неровной дороге бота шатает)

2) **AvoidObstacle()** – Заставляет робота при встрече препятствия поворачивает в сторону с наибольшим значением датчиков для нахождения препятствия. Работает она таким образом:

1) Смотрит значения 2 передних левых датчика, после значения 2 передних правых датчиков и находит, что из них максимально.

2) От максимального значения зависит в какую сторону будет поворачивать бот

3) При нахождении максимума определяется, какая сторона имеет большое значение датчиков

4) После у той стороны, у которой значение больше, берутся значения 2 передних и одного бокового датчика и суммируются, взятые с определенными коэффициентами

5) После эти суммы прибавляются к той стороне мотора, с какой стороны взяты суммы, а также вычитаются из противоположного мотора

3) **ObstacleTrack()** – При встрече с препятствием и после поворота обеспечивает движение почти параллельно грани препятствия, компенсируется самой функцией нахождения препятствия, а именно:

1) Поворачивает бота в сторону препятствия

2) Если препятствие заканчивается, то поворачивает в ту сторону, где было препятствие

3) После поворот этой функции к препятствию компенсируется функцией нахождения препятствия (поэтому при объезде препятствия бота немножечко шатает)

Также в нашей программе присутствует главная часть, где вызываются эти функции и задается значения скоростей моторов, которые берутся из результата выполнения 3 функции). Так же проходит первичная инициализация моторов и датчиков.

**Код:**

"""controller"""

from controller import Robot, Motor, DistanceSensor

TIME\_STEP = 64

MAX\_SPEED = 6.28

obstacle = 0

obstacleSide = 0

lastobstacleSide = 0

def LineFollow():

global leftSpeed,rightSpeed,GroundSensorValues,obstacleSide,lastobstacleSide

defaultSpeed = MAX\_SPEED/4 #3.14

modifier = 0.004#0.003

delta = GroundSensorValues[2] - GroundSensorValues[0]

leftSpeed = defaultSpeed - delta \* modifier

rightSpeed = defaultSpeed + delta \* modifier

obstacleSide = 0

if GroundSensorValues[1] < 500:

lastobstacleSide = 0

def AvoidObstacle():

avoidMode = [0.2, 0.9, 1.2]

active = [0, 0]

maxValue = 0

delta = 0

deltaSpeed = 600

global obstacle,IRSensorValues,obstacleSide,leftSpeed,rightSpeed,lastobstacleSide

obstacle = 0

if maxValue < IRSensorValues[0]:

maxValue = IRSensorValues[0]

active[1] += IRSensorValues[0]

if maxValue < IRSensorValues[1]:

maxValue = IRSensorValues[1]

active[1] += IRSensorValues[1]

# if maxValue < IRSensorValues[2]:

# maxValue = IRSensorValues[2]

# active[1] += IRSensorValues[2]

# if maxValue < IRSensorValues[5]:

# maxValue = IRSensorValues[5]

# active[0] += IRSensorValues[5]

if maxValue < IRSensorValues[6]:

maxValue = IRSensorValues[6]

active[0] += IRSensorValues[6]

if maxValue < IRSensorValues[7]:

maxValue = IRSensorValues[7]

active[0] += IRSensorValues[7]

if maxValue > 100:

obstacle = 1

# else:

# obstacle = 0

if obstacleSide == 0 and obstacle == 1:

if active[1] > active[0]:

obstacleSide = 1

lastobstacleSide = 1

else:

lastobstacleSide = -1

print (lastobstacleSide)

# if obstacle == 0:

# obstacleSide = 0

if obstacle == 1:

leftSpeed = 6.28 / 8

rightSpeed = 6.28 / 8

if obstacleSide == -1:

for i in range(3):

delta -= avoidMode[i] \* IRSensorValues[i]

else:

for i in range(3):

delta += avoidMode[i] \* IRSensorValues[i + 5]

if delta > deltaSpeed:

delta = deltaSpeed

if delta < -deltaSpeed:

delta = -deltaSpeed

leftSpeed -= delta/100

rightSpeed += delta/100

print('AO obstacle: ' + str(obstacleSide))

def ObstacleTrack():

global leftSpeed,rightSpeed, obstacleSide, lastobstacleSide

defaultSpeed = MAX\_SPEED/8

if lastobstacleSide == -1: #препятствие слева

leftSpeed -= defaultSpeed

rightSpeed += defaultSpeed

if lastobstacleSide == 1: #препятствие справа

leftSpeed += defaultSpeed

rightSpeed -= defaultSpeed

prevValue = 900

lastobstacleside = 0

BTLworking = 0

def BackToLine():

global leftSpeed,rightSpeed,GroundSensorValues,obstacleSide, prevValue, lastobstacleside, BTLworking

turnValue = 50

if obstacleSide != 0:

defaultBTLSpeed = MAX\_SPEED/16

curValue = GroundSensorValues[1]

print('CV: ' + str(curValue) + ' PV:' + str(prevValue))

if (curValue > 500) and (prevValue < 500): #сошел с линии

lastobstacleside = obstacleSide

if (curValue < 500) and (prevValue > 500): #нашел линию

print('found')

BTLworking = 1

if (BTLworking == 1) :

delta = GroundSensorValues[2] - GroundSensorValues[0]

print(delta)

print(BTLworking)

if lastobstacleside == -1 :

leftSpeed = defaultBTLSpeed

rightSpeed = -defaultBTLSpeed

elif lastobstacleside == 1 :

leftSpeed = -defaultBTLSpeed

rightSpeed = defaultBTLSpeed

if (delta < turnValue) and (delta > -turnValue) and (curValue <500):

BLTWorking = 0

lastobstacleside = 0

prevValue = curValue

print(str(obstacleSide) + ' ' + str(lastobstacleside))

robot = Robot()

timestep = int(robot.getBasicTimeStep())

# motor = robot.getMotor('motorname')

# ds = robot.getDistanceSensor('dsname')

# ds.enable(timestep)

leftMotor = robot.getMotor('left wheel motor')

rightMotor = robot.getMotor('right wheel motor')

IRSensorNames = ['ps0','ps1','ps2','ps3',

'ps4','ps5','ps6','ps7']

IRSensors = []

for sensor in IRSensorNames:

IRSensors.append(robot.getDistanceSensor(sensor))

GroundSensorNames = ['gs0','gs1','gs2']

GroundSensors = []

for sensor in GroundSensorNames:

GroundSensors.append(robot.getDistanceSensor(sensor))

for sensor in GroundSensors:

sensor.enable(TIME\_STEP)

for sensor in IRSensors:

sensor.enable(TIME\_STEP)

leftMotor.setPosition(float('inf'))

rightMotor.setPosition(float('inf'))

leftMotor.setVelocity(0.0)

rightMotor.setVelocity(0.0)

leftSpeed = 0

rightSpeed = 0

while robot.step(timestep) != -1:

GroundSensorValues = []

IRSensorValues = []

for sensor in GroundSensors:

GroundSensorValues.append(sensor.getValue())

for sensor in IRSensors:

IRSensorValues.append(sensor.getValue())

# for sensor in GroundSensorValues:

# print(str(sensor))

LineFollow()

AvoidObstacle()

ObstacleTrack()

# BackToLine()

# if leftSpeed>6.28: leftSpeed = 6.28

# if rightSpeed>6.28: rightSpeed = 6.28

leftMotor.setVelocity(leftSpeed)

rightMotor.setVelocity(rightSpeed)

# print(str(leftSpeed) + " " + str(rightSpeed))

# Enter here functions to send actuator commands, like:

# motor.setPosition(10.0)

pass