МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра вычислительные системы и технологии

ОТЧЕТ

по лабораторной работе №2

по дисциплине

Программное обеспечение вычислительных сетей

РУКОВОДИТЕЛЬ:

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

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сапожников В.О.

19-В-1

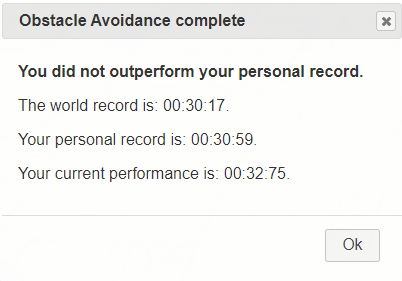
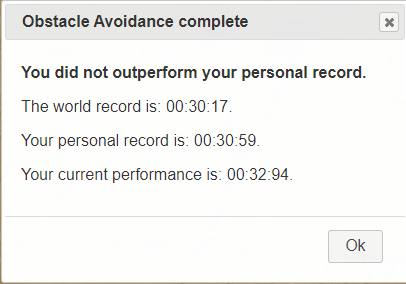
Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

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

Нижний Новгород 2022

1. **Обход препятствий**

**Цель:** разработка надёжного и эффективного алгоритма обхода препятствий робота Thymio II. Робот должен перечь комнату и как можно быстрее достичь противоположной стены, избегая при этом столкновений с препятствиями.



Для обнаружения препятствий данный робот имеет пять датчиков расстояния, которые возвращают значения обратно пропорциональные расстоянию до препятствия. Для обхода препятствии в зависимости от значений датчиков меняются скорости двигателей для совершения поворота в противоположную от препятствия сторону.

Для движения в нужном направлении после обхода препятствий, при помощи компаса получаем положение в пространстве робота. Если данный вектор отгоняется, то совершается поворот за счёт изменения скорости двигателей робота.

**Листинг**

from controller import Robot, Compass

# Получение ссылки на робота

robot = Robot()

# Получение ссылки на компас робота

compass = robot.getCompass("compass")

# Get simulation step length.

timeStep = int(robot.getBasicTimeStep())

# Константы максимальной скорости и датчиков расстояния робота

maxMotorVelocity = 9.53

distanceSensorCalibrationConstant = 360

# Получение ссылок на левый и правый моторы

leftMotor = robot.getMotor("motor.left")

rightMotor = robot.getMotor("motor.right")

# Получение ссылок на датчики расстояния

outerLeftSensor = robot.getDistanceSensor("prox.horizontal.0")

centralLeftSensor = robot.getDistanceSensor("prox.horizontal.1")

centralSensor = robot.getDistanceSensor("prox.horizontal.2")

centralRightSensor = robot.getDistanceSensor("prox.horizontal.3")

outerRightSensor = robot.getDistanceSensor("prox.horizontal.4")

# Включение датчиков расстояния

outerLeftSensor.enable(timeStep)

centralLeftSensor.enable(timeStep)

centralSensor.enable(timeStep)

centralRightSensor.enable(timeStep)

outerRightSensor.enable(timeStep)

# Включение компаса

compass.enable(timeStep)

# Disable motor PID control mode.

leftMotor.setPosition(float('inf'))

rightMotor.setPosition(float('inf'))

# Цикл работы робота

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

# Получаем данные с компаса

values = compass.getValues()

# Получение данных с четырёх датчиков (кроме центрального) и

# их калибровка

outerLeftSensorValue = outerLeftSensor.getValue() / distanceSensorCalibrationConstant

centralLeftSensorValue = centralLeftSensor.getValue() / distanceSensorCalibrationConstant

centralSensorValue = centralSensor.getValue() / distanceSensorCalibrationConstant

centralRightSensorValue = centralRightSensor.getValue() / distanceSensorCalibrationConstant

outerRightSensorValue = outerRightSensor.getValue() / distanceSensorCalibrationConstant

# Если встречено препятствие

if centralLeftSensorValue != 0.0 or centralSensorValue != 0.0 or centralRightSensorValue != 0.0 or outerRightSensorValue != 0.0 or outerLeftSensorValue != 0.0:

#Если препятствие слева

if centralLeftSensorValue != 0.0 or centralSensorValue != 0.0 or outerLeftSensorValue != 0.0:

rightMotor.setVelocity(-maxMotorVelocity)

leftMotor.setVelocity(maxMotorVelocity)

#Если препятствие справа

elif centralSensorValue != 0.0 or centralRightSensorValue != 0.0 or outerRightSensorValue != 0.0:

leftMotor.setVelocity(-maxMotorVelocity)

rightMotor.setVelocity(maxMotorVelocity)

#Если препятствий нет, но робот отклонен от курса

else:

#displacement of the vision towards the line

if (round(values[0],3) > 0.001) and (centralLeftSensorValue == 0.0 and centralSensorValue == 0.0 and outerLeftSensorValue == 0.0 and centralRightSensorValue == 0.0 and outerRightSensorValue == 0.0):

rightMotor.setVelocity(maxMotorVelocity \* 0.8)

leftMotor.setVelocity(maxMotorVelocity)

elif (round(values[0],3) < 0.001) and (centralLeftSensorValue == 0.0 and centralSensorValue == 0.0 and outerLeftSensorValue == 0.0 and centralRightSensorValue == 0.0 and outerRightSensorValue == 0.0):

leftMotor.setVelocity(maxMotorVelocity \* 0.8)

rightMotor.setVelocity(maxMotorVelocity)

#Если робот движется в нужном направлении

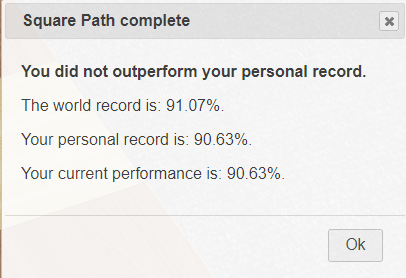
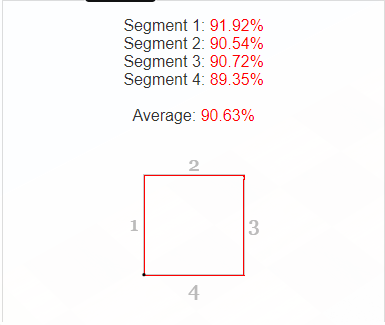
else:

leftMotor.setVelocity(maxMotorVelocity)

rightMotor.setVelocity(maxMotorVelocity)

1. **Движение по квадрату**

**Цель:** разработка программы, управляющей движением робота Pionner при движении по квадрату размером 2 на 2.



По условию соревнования результат определяется как среднее значение из результатов для каждой стороны. Результат для одной стороны равен 100%, когда сторона пройдена идеально за 0 секунд. Поэтому результат в 100% является недостижимым, но к нему можно пытаться максимально приблизиться, в чем и заключается соревнование.

**Листинг**

from controller import Robot

import math

# Получение ссылки на робота

robot = Robot()

# Получение ссылок на левый и правый моторы работа

leftWheel = robot.getMotor('left wheel')

rightWheel = robot.getMotor('right wheel')

# Получение ссылок на позиционные датчики

rightWheelSensor = robot.getPositionSensor('right wheel sensor')

rightWheelSensor.enable(16)

leftWheelSensor = robot.getPositionSensor('left wheel sensor')

leftWheelSensor.enable(16)

# диаметр колеса

diamWheel = 0.195

# расстояние между колесами

distWheels = 0.33

sensorValue = 0

# Длина окружности колеса

l = math.pi \* diamWheel

# Максимальная скорость

MAX\_SPEED = 5.24

# Repeat the following 4 times (once for each side).

for i in range(0, 4):

leftWheel.setPosition(1000)

rightWheel.setPosition(1000)

robot.step(16)

while rightWheelSensor.getValue() \* diamWheel/2.0 <

sensorValue + 2.0:

if rightWheelSensor.getValue() \* diamWheel/2.0 >

sensorValue + 1.9:

leftWheel.setVelocity(0.6 \* MAX\_SPEED)

rightWheel.setVelocity(0.6 \* MAX\_SPEED)

robot.step(160)

if i == 0:

leftWheel.setPosition(leftWheelSensor.getValue() + (math.pi/2.0 \* distWheels/2.0)/l \* 2.0 \* math.pi + 0.09)

rightWheel.setPosition(rightWheelSensor.getValue() - (math.pi/2.0 \* distWheels/2.0)/l \* 2.0 \* math.pi - 0.05)

elif i == 1:

leftWheel.setPosition(leftWheelSensor.getValue() + (math.pi/2.0 \* distWheels/2.0)/l \* 2.0 \* math.pi + 0.06)

rightWheel.setPosition(rightWheelSensor.getValue() - (math.pi/2.0 \* distWheels/2.0)/l \* 2.0 \* math.pi - 0.05)

elif i == 2:

leftWheel.setPosition(leftWheelSensor.getValue() + (math.pi/2.0 \* distWheels/2.0)/l \* 2.0 \* math.pi + 0.11)

rightWheel.setPosition(rightWheelSensor.getValue() - (math.pi/2.0 \* distWheels/2.0)/l \* 2.0 \* math.pi)

elif i == 3:

continue

robot.step(912)

sensorValue = rightWheelSensor.getValue() \* diamWheel/2

leftWheel.setVelocity(MAX\_SPEED)

rightWheel.setVelocity(MAX\_SPEED)

# Stop the robot when path is completed, as the robot performance

# is only computed when the robot has stopped.

leftWheel.setVelocity(0)

rightWheel.setVelocity(0)