Минобрнауки России

Федеральное государственное бюджетное образовательное

учреждение высшего образования

**НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**ИМ. Р.Е. АЛЕКСЕЕВА**

ИНСТИТУТ РАДИОЭЛЕКТРОНИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

|  |  |  |
| --- | --- | --- |
|  |  |  |

Курс “Аппаратное и программное обеспечение роботизированных систем”

**Отчет по лабораторной работе №1**

Выполнил: Балашов М.А.

Проверил: Гай В.Е.

Нижний Новгород 2021

**Задача 1. Обход препятствий**

Этот тест направлен на создание надежного и эффективного алгоритма обхода препятствий для робота Thymio II с использованием языка программирования Python. Цель состоит в том, чтобы робот пересек комнату и достиг противоположной стены как можно быстрее, избегая при этом всех столкновений с препятствиями. Чтобы стимулировать устойчивое поведение избегания, препятствия располагаются случайным образом при каждом пробеге. """Braitenberg-based obstacle-avoiding robot controller."""

# import Compass module

from controller import Compass

from controller import Robot

# Get reference to the robot.

robot = Robot()

# Get simulation step length.

timeStep = int(robot.getBasicTimeStep())

# Constants of the Thymio II motors and distance sensors.

maxMotorVelocity = 15

distanceSensorCalibrationConstant = 200

# Get left and right wheel motors.

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

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

# get robot's Compass device

compass = robot.getCompass("compass")

# enable the Compass

compass.enable(timeStep)

# Get frontal distance sensors.

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")

# Enable distance sensors.

outerLeftSensor.enable(timeStep)

centralLeftSensor.enable(timeStep)

centralSensor.enable(timeStep)

centralRightSensor.enable(timeStep)

outerRightSensor.enable(timeStep)

# Disable motor PID control mode.

leftMotor.setPosition(float('inf'))

rightMotor.setPosition(float('inf'))

# Set ideal motor velocity.

initialVelocity = 0.7 \* maxMotorVelocity

# Set the initial velocity of the left and right wheel motors.

leftMotor.setVelocity(initialVelocity)

rightMotor.setVelocity(initialVelocity)

#flag=False

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

# to read values

values = compass.getValues()

leftMotor.setVelocity(initialVelocity)

rightMotor.setVelocity(initialVelocity)

# Read values from four distance sensors and calibrate.

outerLeftSensorValue = outerLeftSensor.getValue() / distanceSensorCalibrationConstant

centralLeftSensorValue = centralLeftSensor.getValue() / distanceSensorCalibrationConstant

centralSensorValue = centralSensor.getValue() / distanceSensorCalibrationConstant

centralRightSensorValue = centralRightSensor.getValue() / distanceSensorCalibrationConstant

outerRightSensorValue = outerRightSensor.getValue() / distanceSensorCalibrationConstant

#if there is an obstacle

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

#if obstacle in left

if ((centralLeftSensorValue != 0.0) or (outerLeftSensorValue != 0.0)):

rightMotor.setVelocity(-initialVelocity)

leftMotor.setVelocity(initialVelocity)

#if obstacle in right

elif ((centralRightSensorValue != 0.0) or (outerRightSensorValue != 0.0)):

rightMotor.setVelocity(initialVelocity)

leftMotor.setVelocity(-initialVelocity)

#if obstacle in everywhere

elif ((outerLeftSensorValue != 0.0) and (centralLeftSensorValue != 0.0) and (centralSensorValue != 0.0) and (centralRightSensorValue != 0.0) and (outerRightSensorValue != 0.0)):

rightMotor.setVelocity(-initialVelocity)

leftMotor.setVelocity(-initialVelocity)

#if abstacle in between

elif ((outerLeftSensorValue != 0.0) and (outerRightSensorValue != 0.0)):

rightMotor.setVelocity(-initialVelocity)

leftMotor.setVelocity(-initialVelocity/1.6)

#if there no obstacle

else:

#if driving to the left

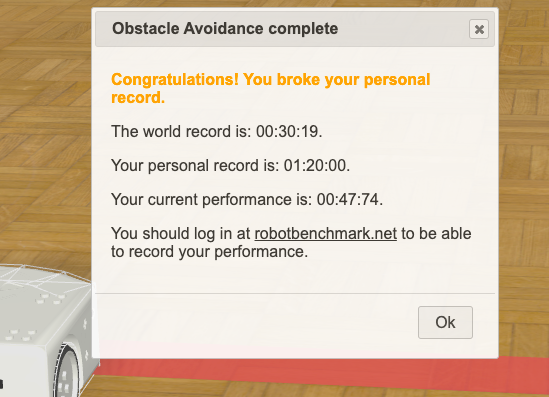
if (values[0] > 0):

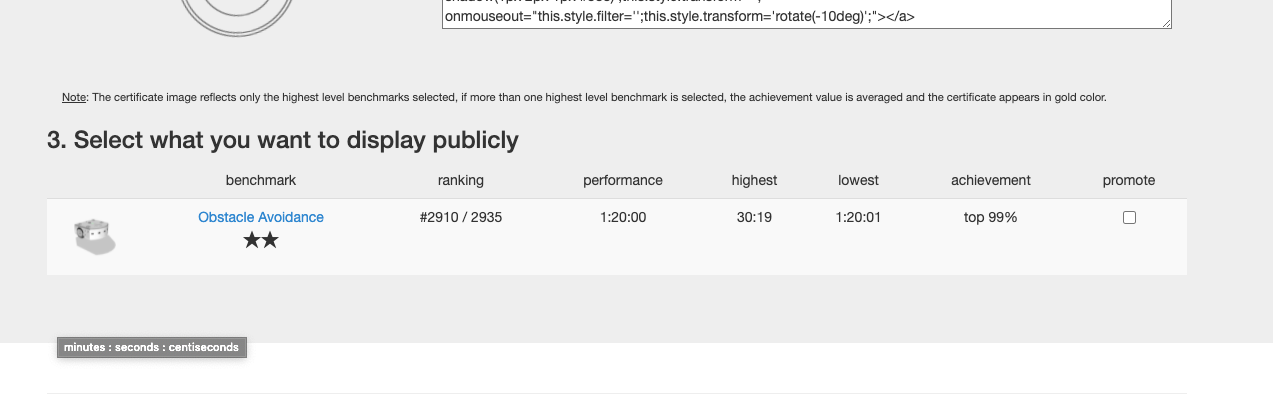
rightMotor.setVelocity(initialVelocity/1.1)

#if driving to the right

elif (values[0] < 0):

leftMotor.setVelocity(initialVelocity/1.1)

**Результат:**   


.

**Задача 2. Движение по квадрату**

Этот бенчмарк направлен на разработку программы с разомкнутым контуром, которая управляет роботом-пионером, чтобы он следовал квадратной траектории размером 2 на 2 метра.

"""square\_path controller."""

from controller import Robot

# Get pointer to the robot.

robot = Robot()

leftWheel = robot.getMotor('left wheel')

rightWheel = robot.getMotor('right wheel')

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

rightWheelSensor.enable(10)

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

leftWheelSensor.enable(10)

sensorValue = -0.04

wheel\_radius = 0.190/1.99

MAX\_SPEED = 5.20

for i in range(0, 4):

leftWheel.setPosition(1000)

rightWheel.setPosition(1000)

robot.step(10)

while ((rightWheelSensor.getValue() \* wheel\_radius) < (2.0 + sensorValue)):

if ((rightWheelSensor.getValue() \* wheel\_radius) > (1.9 + sensorValue)):

leftWheel.setVelocity(0.7 \* MAX\_SPEED)

rightWheel.setVelocity(0.7 \* MAX\_SPEED)

robot.step(10)

if i == 0:

leftWheel.setPosition(leftWheelSensor.getValue() + 2.75)

rightWheel.setPosition(rightWheelSensor.getValue() - 2.71)

robot.step(930)

if i == 1:

leftWheel.setPosition(leftWheelSensor.getValue() + 2.74)

rightWheel.setPosition(rightWheelSensor.getValue() - 2.709)

robot.step(930)

if i == 2:

leftWheel.setPosition(leftWheelSensor.getValue() + 2.75)

rightWheel.setPosition(rightWheelSensor.getValue() - 2.70)

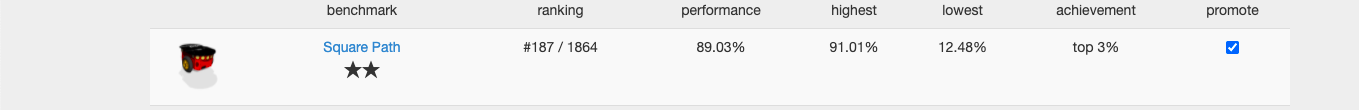
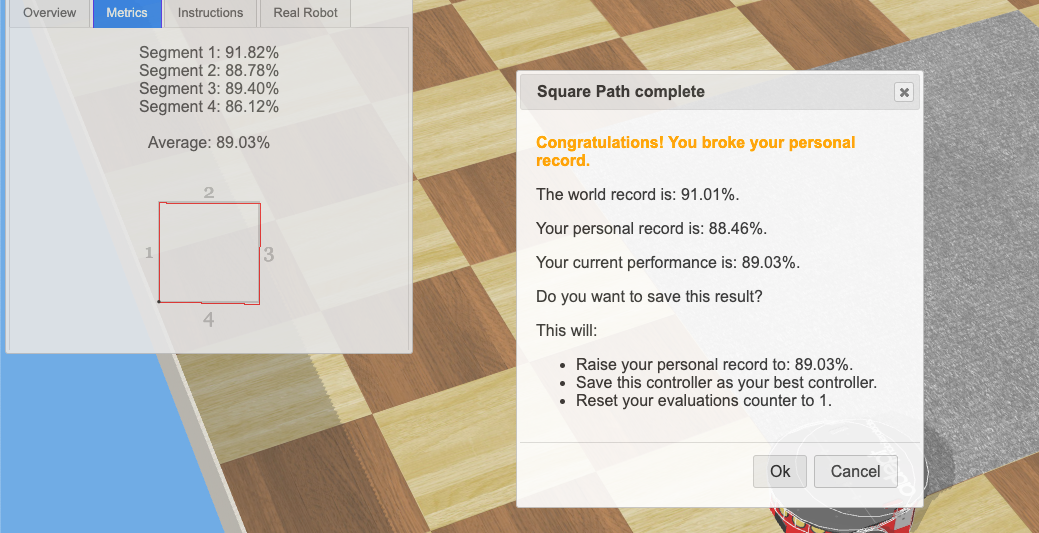
robot.step(930)

leftWheel.setVelocity(MAX\_SPEED)

rightWheel.setVelocity(MAX\_SPEED)

sensorValue = rightWheelSensor.getValue() \* wheel\_radius

leftWheel.setVelocity(0)

rightWheel.setVelocity(0)  
**Результат:**

Для прохождения данного бенчмарка, были подключены датчики положения колес, считывающие положение каждые 10мс положение робота. Цикл выполняется, пока робот не проедет 2 метра, где сравнивается значение текущего положения правого колеса, помноженное на его радиус, с положением правого колеса относительно предыдущей вершины + 2 метра. При приближении к вершине (1.9м) робот снижает скорость, чтобы успешнее повернуть, после поворота – возвращает ее.

Для того, чтобы повернуть робота на 90 градусов, пользуемся датчиками положения колес.

В зависимости от вершины, устанавливаются различные позиции колес: текущее положение колеса +- угол поворота. Угол поворота рассчитывается исходя из расстояния между колесами (считаем заранее).