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

Лабораторная работа № 2

ОТЧЕТ

по лабораторной работе

по дисциплине

Аппаратное и программное обеспечение роботизированных систем

РУКОВОДИТЕЛЬ:

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

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Цветков Н.М.

19-В-1

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

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

Нижний Новгород 2022

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

**Задание**

Задание: выполнить вариант и загрузить программу на платформу для соревнований, записать видео работы, подготовить отчёт с подробным описанием результатов. В отчёт вставить результаты с соревнования. Есть примеры, можно на них посмотреть (видео на сайте).

**Задание №1**

**Код**

"""Braitenberg-based obstacle-avoiding robot controller."""

from controller import Robot

from controller import Compass

# 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 = 9.53

distanceSensorCalibrationConstant = 200

# Get left and right wheel motors.

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

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

compass = robot.getCompass("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 = 1 \* maxMotorVelocity

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

leftMotor.setVelocity(initialVelocity)

rightMotor.setVelocity(initialVelocity)

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

values = compass.getValues()

# 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 outerLeftSensorValue == 0 and centralLeftSensorValue == 0 and centralSensorValue == 0 and centralRightSensorValue == 0 and outerRightSensorValue == 0:

if values[0] < -0.001:

leftMotor.setVelocity(0.95 \* maxMotorVelocity)

rightMotor.setVelocity(maxMotorVelocity)

elif values[0] > 0.001:

leftMotor.setVelocity(maxMotorVelocity)

rightMotor.setVelocity(0.95 \* maxMotorVelocity)

else:

leftMotor.setVelocity(maxMotorVelocity)

rightMotor.setVelocity(maxMotorVelocity)

if outerLeftSensorValue != 0 or centralLeftSensorValue != 0 or centralSensorValue != 0:

leftMotor.setVelocity(maxMotorVelocity)

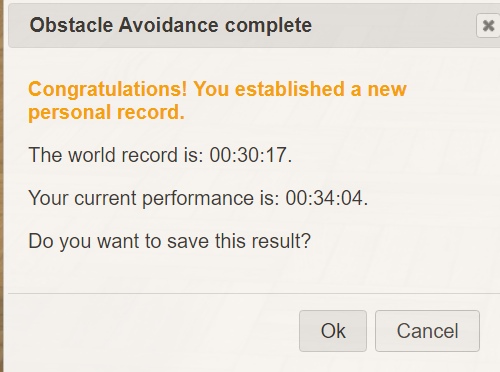
rightMotor.setVelocity(-0.4 \* maxMotorVelocity)

if outerRightSensorValue != 0 or centralRightSensorValue != 0:

leftMotor.setVelocity(-0.4 \* maxMotorVelocity)

rightMotor.setVelocity(maxMotorVelocity)

**Результат**



**Задание №2**

**Код**

"""Sample Webots controller for the square path benchmark."""

from controller import Robot

MAX\_SPEED = 5.24

# Get pointer to the robot.

robot = Robot()

# Get pointer to each wheel of our robot.

leftWheel = robot.getMotor('left wheel')

rightWheel = robot.getMotor('right wheel')

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

rightWheelSensor.enable(16)

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

leftWheelSensor.enable(16)

positionSensorValue = 0

wheel\_radius = 0.195/2

wheel\_spacing = 0.33

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

for i in range(0, 4):

# First set both wheels to go forward, so the robot goes straight.

leftWheel.setPosition(1000)

rightWheel.setPosition(1000)

# Wait for the robot to reach a corner.

robot.step(16)

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

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

leftWheel.setVelocity(0.6 \* MAX\_SPEED)

rightWheel.setVelocity(0.6 \* MAX\_SPEED)

robot.step(160)

if i == 0:

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

rightWheel.setPosition(rightWheelSensor.getValue() - 2.72)

robot.step(912)

if i == 1:

leftWheel.setPosition(leftWheelSensor.getValue() + 2.72)

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

robot.step(912)

if i == 2:

leftWheel.setPosition(leftWheelSensor.getValue() + 2.77)

rightWheel.setPosition(rightWheelSensor.getValue() - 2.66)

robot.step(912)

if i == 3:

break

leftWheel.setVelocity(MAX\_SPEED)

rightWheel.setVelocity(MAX\_SPEED)

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

# 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)

**Результат**

