МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

ОТЧЕТ

по лабораторной работе №1

по дисциплине

Аппаратное и программное обеспечение роботизированных систем

РУКОВОДИТЕЛЬ:

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

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Демин Д.И.

(подпись) (фамилия, и.,о.)

18-В-1

(шифр группы)

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

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

Нижний Новгород 2021

**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.

**Задача 1. Обход препятствий**

"""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 = 150

# 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 or centralLeftSensorValue != 0 or centralSensorValue != 0 or centralRightSensorValue != 0 or outerRightSensorValue != 0:

# initialVelocity = 0.75 \* maxMotorVelocity

# else:

# initialVelocity = 1 \* maxMotorVelocity

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

# Set wheel velocities based on sensor values, prefer right turns if the central sensor is triggered.

leftMotor.setVelocity(initialVelocity - (centralRightSensorValue + outerRightSensorValue) / 2 - centralSensorValue)

rightMotor.setVelocity(initialVelocity - (centralLeftSensorValue + outerLeftSensorValue) / 2)

robot.step(timeStep)

robot.step(timeStep)

robot.step(timeStep)

else:

if values[2] > -1:

if values[0] > 0:

leftMotor.setVelocity(initialVelocity)

rightMotor.setVelocity(8.0)

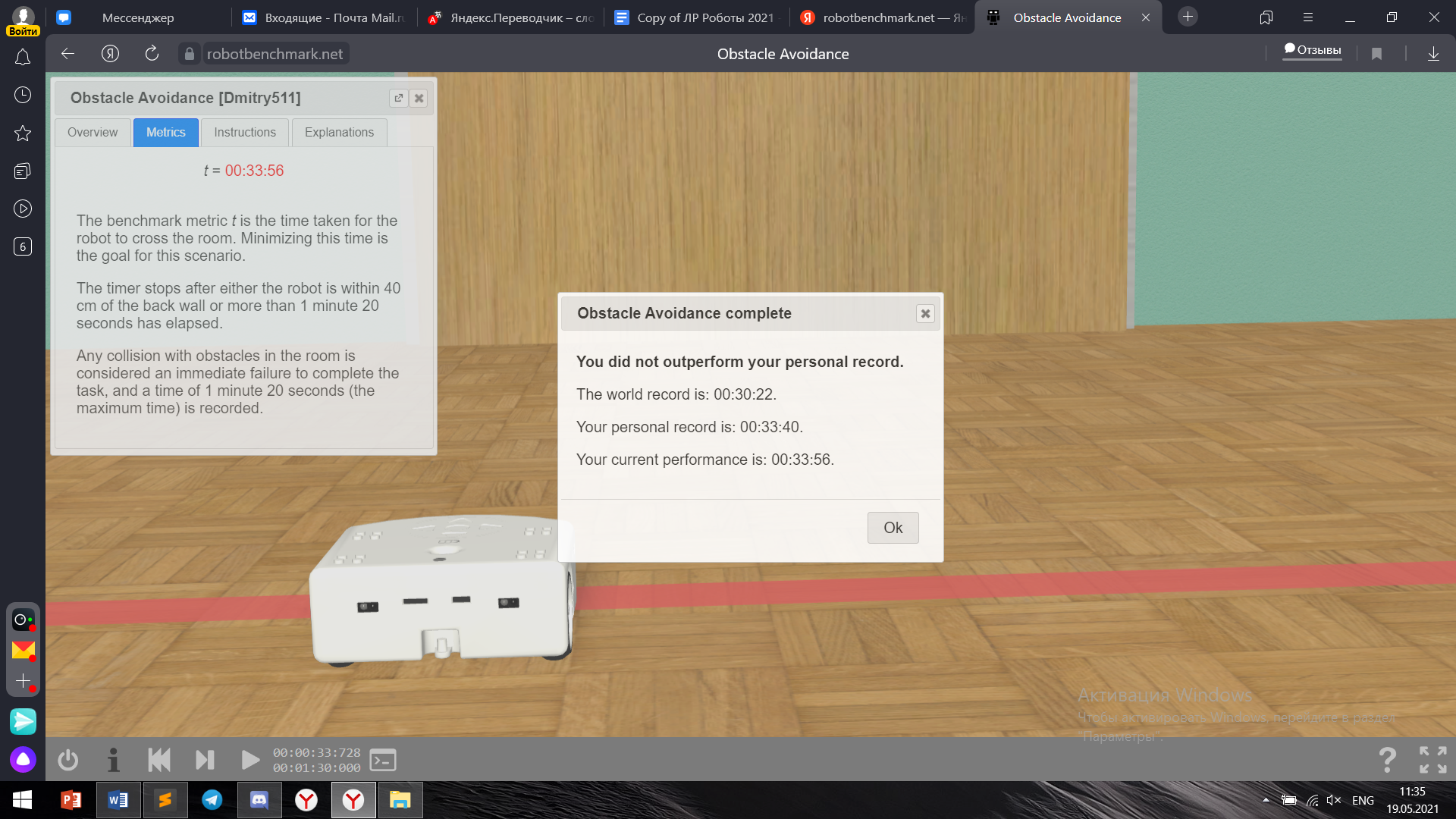
if values[0] < 0:

rightMotor.setVelocity(initialVelocity)

leftMotor.setVelocity(8.0)

Для программирования этого робота я использую сенсоры для обнаружения препятствий. Для начала параметру distanceSensorCalibrationConstant я выставил значение 150, чтобы он заранее реагировал на препятствие. Затем делаю проверку, если есть препятствия, то разворачиваться в нужную сторону, если нет, то считать данные с компаса и развернуться так, чтобы он ехал по направлению к финишу.

**Результат:** в силу того, что препятствия выпадают случайным образом, сложно представить в отчете именно лучший результат.



**Задача 2. Движение по квадрату**

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

from controller import Robot

# 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')

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

leftWheelSensor.enable(16)

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

length = 20.51

step = 0.083

turn = 2.447

count = 0

spin = length + turn

way = length

for i in range(0, 4):

leftWheel.setPosition(1000)

rightWheel.setPosition(1000)

robot.step(16)

while leftWheelSensor.getValue() < way:

if i != 0:

if count == 15:

rightWheel.setVelocity(2.62)

if i == 1:

robot.step(16)

if i == 2:

robot.step(16)

robot.step(16)

if i == 3:

rightWheel.setVelocity(1.72)

robot.step(16)

robot.step(16)

if count == 195:

if i == 3:

rightWheel.setVelocity(1.72)

robot.step(16)

robot.step(16)

print(count)

robot.step(16)

rightWheel.setVelocity(5.24)

count+=1

if i == 2:

way = way - 0.19

if i == 3:

break

leftWheel.setPosition(1000)

rightWheel.setPosition(-1000)

while leftWheelSensor.getValue() < spin:

robot.step(16)

count = 0

way = way + length + turn

spin = way + turn

if i == 0:

way = way + 0.2

spin = spin + 0.2

if i == 1:

way = way + 0.19

spin = spin + 0.13

if i == 2:

way = way + 0.4

# 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)

Для этого робота пришлось рассчитать расстояния самого пути и расстояние разворота колеса, чтобы останавливать его в нужный момент. Также при отклонении я присваиваю колесам разную скорость, чтобы он выравнивался на пути. И так как он идет не ровно, значит он проходит не ровно рассчитанное расстояние, поэтому его путь приходится корректировать.

**Результат:**

