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"

Отчёт по лабораторной работе №1

по дисциплине «Аппаратное и программное обеспечение роботизированных систем»

РУКОВОДИТЕЛЬ:

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

СТУДЕНТЫ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Йылдирим М.

Группа 18-В-1

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

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

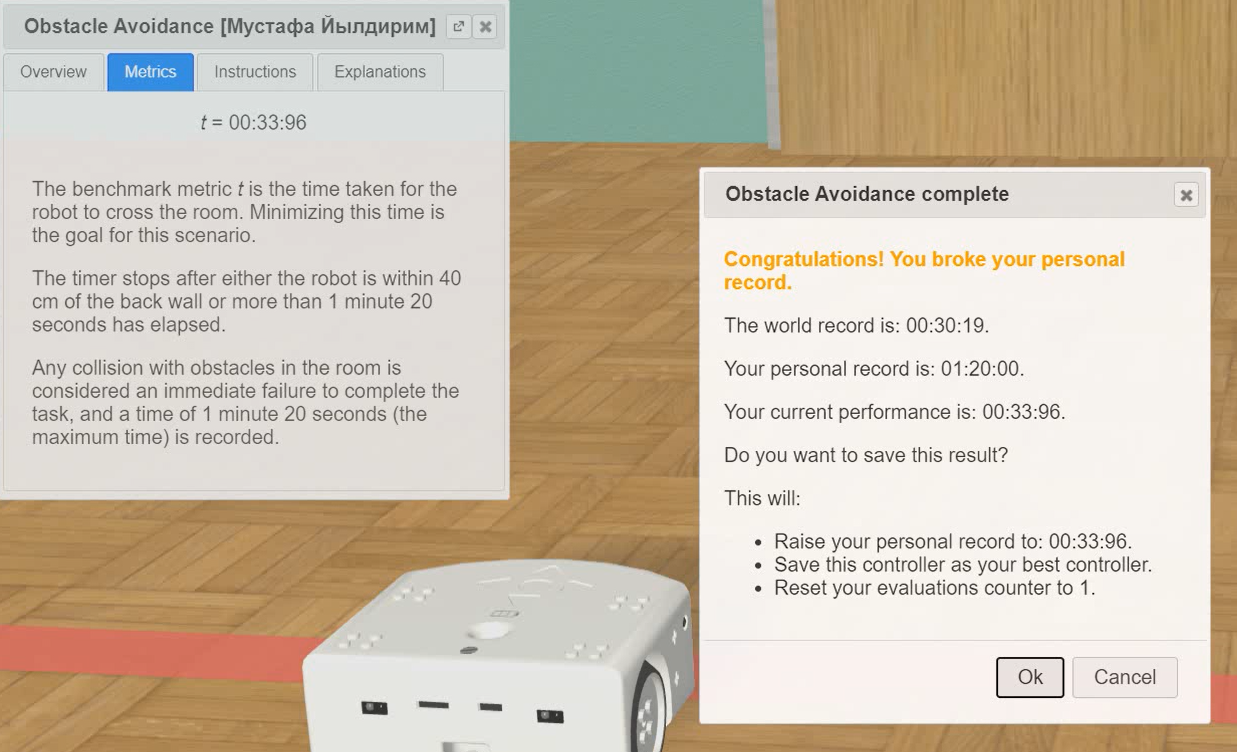
Нижний Новгород

2021 г.

**Задача 1. Обход препятствий**

Этот тест направлен на создание надежного и эффективного алгоритма обхода препятствий для робота Thymio II с использованием языка программирования Python.

Результаты:



Описание. В коде мы убрали ограничение на скорость, стала максимальной для увеличения производительности. Был подключен компас для корректировки направления робота. Если нет препятствий, а значение вектора показывает меньше -0.01, значит постепенно снижаем скорость у левого колеса пока робот не выровняется. Если нет препятствий, а значение вектора показывает больше 0.01, значит постепенно снижаем скорость у правого колеса пока робот не выровняется. Если есть препятствия, меняли скорость робота в зависимости от датчиков расстояний. Если препятствие слева или по центру, снижаем скорость у левого колеса, если препятствие справа, снижаем скорость у правого колеса.

Код программы:

"""avoid\_obstacles controller."""

from controller import Robot, Compass

# Получить доступ к роботу

robot = Robot()

# Получить длину шага моделирования.

timeStep = int(robot.getBasicTimeStep())

# Константы двигателей Thymio II и датчиков расстояния.

maxMotorVelocity = 9.53

distanceSensorCalibrationConstant = 360

# Получить доступ к двигателям левого и правого колес.

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

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

# Получить доступ к фронтальным датчикам 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")

# Включить датчики расстояний

outerLeftSensor.enable(timeStep)

centralLeftSensor.enable(timeStep)

centralSensor.enable(timeStep)

centralRightSensor.enable(timeStep)

outerRightSensor.enable(timeStep)

# Отключите режим ПИД-управление двигателем

# (чтобы робот двигался бесконечно)

leftMotor.setPosition(float('inf'))

rightMotor.setPosition(float('inf'))

# Установить начальную скорость левого и правого колеса (без ограничений)

leftMotor.setVelocity(maxMotorVelocity)

rightMotor.setVelocity(maxMotorVelocity)

# Подключить компас

compass = robot.getCompass("compass")

# Включить компас

compass.enable(timeStep)

# Основной цикл с шагом симуляции

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

# Считывание значений с четырех датчиков расстояния и калибровка

outerLeftSensorValue = outerLeftSensor.getValue() / distanceSensorCalibrationConstant

centralLeftSensorValue = centralLeftSensor.getValue() / distanceSensorCalibrationConstant

centralSensorValue = centralSensor.getValue() / distanceSensorCalibrationConstant

centralRightSensorValue = centralRightSensor.getValue() / distanceSensorCalibrationConstant

outerRightSensorValue = outerRightSensor.getValue() / distanceSensorCalibrationConstant

# Считывание значений с компаса

values = compass.getValues()

# Если препятствия не обнаружены, мы считываем значение с компаса

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

# Если значение показывает < -0.01, значит немного уменьшаем скорость у левого колесса

if values[0] < -0.01:

leftMotor.setVelocity(0.85 \* maxMotorVelocity)

rightMotor.setVelocity(maxMotorVelocity)

# Если значение показывает > 0.01, значит немного уменьшаем скорость у правого колеса

elif values[0] > 0.01:

leftMotor.setVelocity(maxMotorVelocity)

rightMotor.setVelocity(0.85 \* maxMotorVelocity)

# Иначе продолжаем двигаться

else:

leftMotor.setVelocity(maxMotorVelocity)

rightMotor.setVelocity(maxMotorVelocity)

# Если препятствия обнаружены, мы считываем значения с датчиков

# Если левые или центральный датчики показывают ненулевые значения,

# значит снижаем скорость у правого колеса

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

leftMotor.setVelocity(maxMotorVelocity)

rightMotor.setVelocity(-0.7 \* maxMotorVelocity)

# Если правые датчики показывают ненулевые значения,

# значит снижаем скорость у левого колеса

if outerRightSensorValue != 0 or centralRightSensorValue != 0:

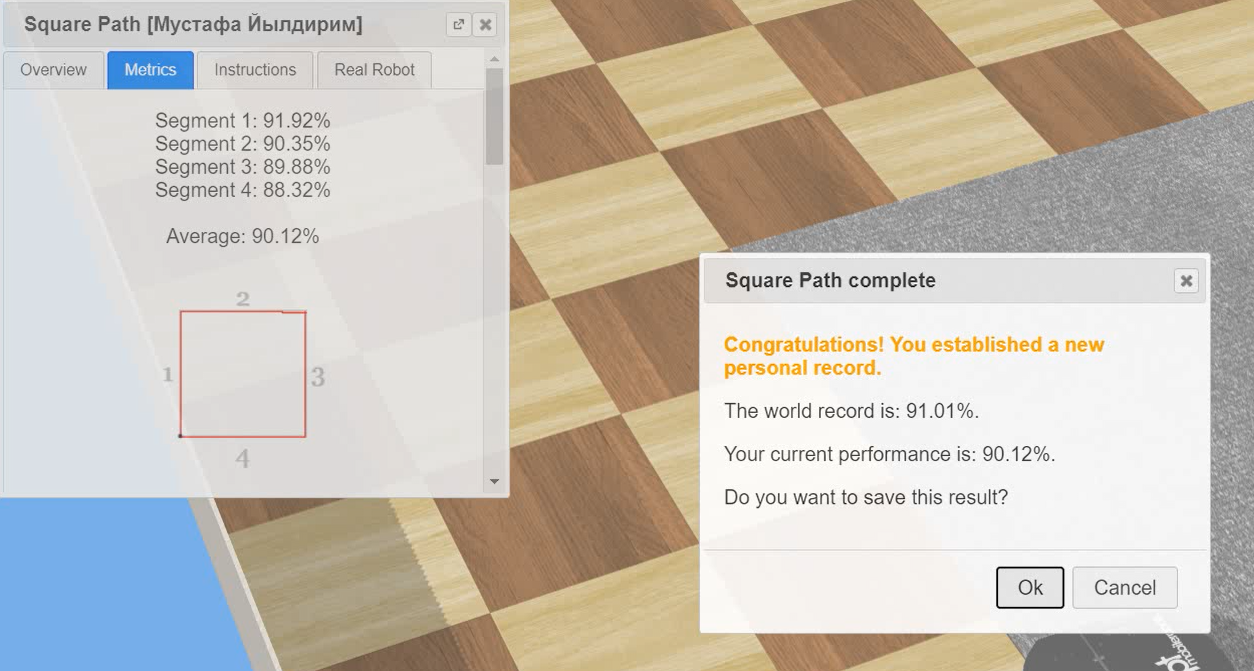
leftMotor.setVelocity(-0.7 \* maxMotorVelocity)

rightMotor.setVelocity(maxMotorVelocity)

**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.

Результаты:



Описание. Для прохождения квадрата использовали датчики положения. В алгоритме сравнивается значение текущего пройденное расстояние роботом с 2 метрами + расстояние относительно предыдущей вершины. Робот снижает свою скорость, как только почти достигнет вершины. В конце цикла возвращает. Для поворота на 90 градусов устанавливаются различные позиции колес в зависимости от вершины. Берется текущее положение колеса угол поворота. Так как датчики положения выдают свои данные периодически, выполнить идеальное движение по квадрату проблематично.

Код программы:

"""square\_path controller."""

from controller import Robot

# Максимальная скорость робота

MAX\_VEL = 5.24

# Получить доступ к роботу

robot = Robot()

# Получить доступ к двигателям левого и правого колес.

leftWheel = robot.getMotor('left wheel')

rightWheel = robot.getMotor('right wheel')

# Получить доступ к сенсорному датчику правого колеса

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

rightWheelSensor.enable(16)

# Получить доступ к сенсорному датчику левого колеса

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

leftWheelSensor.enable(16)

# Текущее значение датчика положения колес (в метрах)

positionSensorValue = 0

# Радиус колеса

wheel\_radius = 0.195/2

# Расстояние между колесами

wheel\_spacing = 0.33

# Повторение следующих действий 4 раза (по одному разу для каждой стороны).

for i in range(0, 4):

# Для первоночальных значений датчика

leftWheel.setPosition(1000)

rightWheel.setPosition(1000)

robot.step(16)

# Двигаться пока текущее положение не будет равно двум метрам + положение пред. вершины

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

# Снижаем скорость почти достигнув вершины

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

leftWheel.setVelocity(0.6 \* MAX\_VEL)

rightWheel.setVelocity(0.6 \* MAX\_VEL)

robot.step(160)

# При повороте в зависимости от вершины меняем позицию колеса

if i == 0:

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

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

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

# Возвращаем скорость

leftWheel.setVelocity(MAX\_VEL)

rightWheel.setVelocity(MAX\_VEL)

# Считываем тек. позицию на датчике

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

# Остановить робота, когда путь будет завершен, так как производительность робота

# вычисляется только тогда, когда робот остановился.

leftWheel.setVelocity(0)

rightWheel.setVelocity(0)