ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО

ОБРАЗОВАНИЯ

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ИМ. Р.Е. АЛЕКСЕЕВА

**ЛАБОРАТОРНАЯ РАБОТА №2**

**Выполнил**: Тронин А.А. 19-В-1

**Проверил**: Гай В.Е.

**Оценка**: \_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород

2021

**Тема**: Программирование алгоритмов управления роботом в Webots

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

**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).

**Задача 1. Обход препятствий**

**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 = 360

# Get left and right wheel motors.

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

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

# 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")

compass = robot.getCompass("compass")

# Enable distance sensors.

outerLeftSensor.enable(timeStep)

centralLeftSensor.enable(timeStep)

centralSensor.enable(timeStep)

centralRightSensor.enable(timeStep)

outerRightSensor.enable(timeStep)

compass.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:

# Read values from four distance sensors and calibrate.

values = compass.getValues()

outerLeftSensorValue = outerLeftSensor.getValue() / distanceSensorCalibrationConstant

centralLeftSensorValue = centralLeftSensor.getValue() / distanceSensorCalibrationConstant

centralSensorValue = centralSensor.getValue() / distanceSensorCalibrationConstant

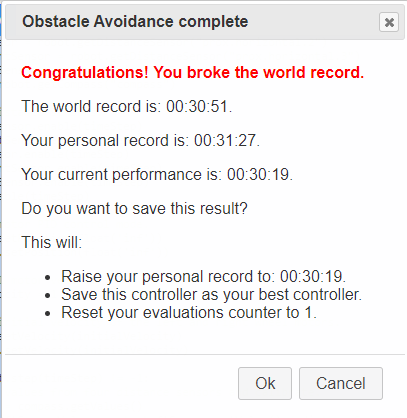
centralRightSensorValue = centralRightSensor.getValue() / distanceSensorCalibrationConstant

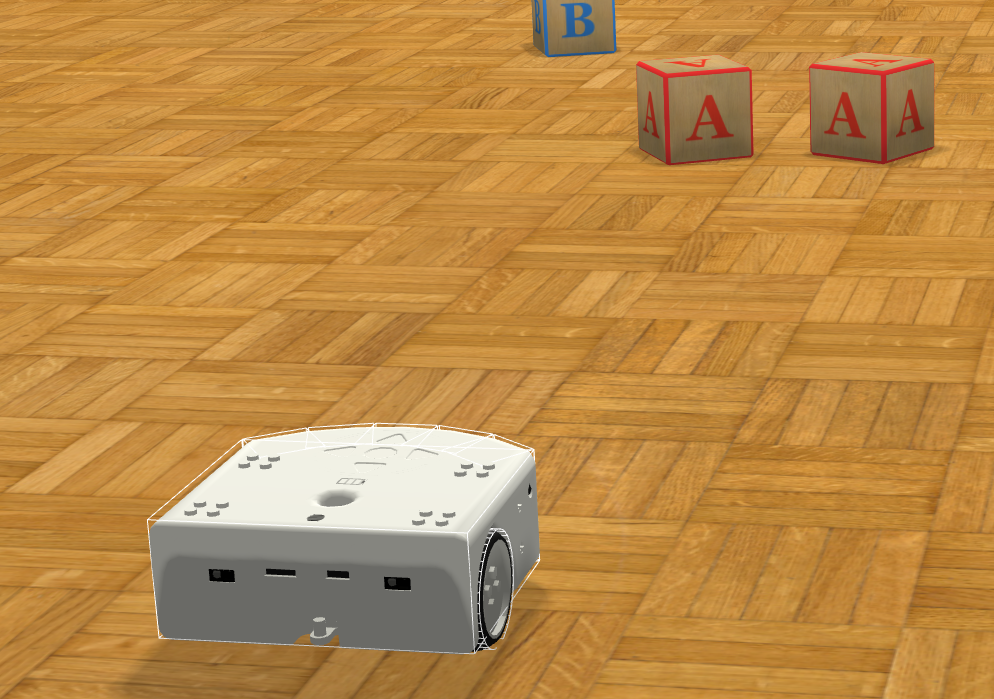
outerRightSensorValue = outerRightSensor.getValue() / distanceSensorCalibrationConstant

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

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

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





**Задача 2. Движение по квадрату**

**Код:**

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

from controller import Robot

TIME\_STEP = 16

# 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')

leftWheel.setPosition(float('inf'))

rightWheel.setPosition(float('inf'))

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

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

i2 = 0

j2 = 0

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

for i in range(0, 4):

i1 = i2

j1 = j2

while (robot.step(TIME\_STEP) != -1):

leftWheel.setVelocity(4.6)

rightWheel.setVelocity(4.6)

rightWheelSensor.enable(16)

leftWheelSensor.enable(16)

i1 = leftWheelSensor.getValue()

j1 = rightWheelSensor.getValue()

if ((i1>=i2 + 20.5) and (j1>=j2 + 20.5)):

break

i2 = i1

j2 = j1

while (robot.step(TIME\_STEP) != -1):

leftWheel.setVelocity(1.24)

rightWheel.setVelocity(-1.24)

rightWheelSensor.enable(16)

leftWheelSensor.enable(16)

i2 = leftWheelSensor.getValue()

j2 = rightWheelSensor.getValue()

if ((i2>=i1 + 2.50585) and (j2<=j1 - 2.50585)):

break

# 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)

