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”

ОТЧЕТ

по лабораторной работе №2

Программирование алгоритмов управления роботом в Webots

по дисциплине

Аппаратное и программное обеспечение роботизированных систем

РУКОВОДИТЕЛЬ:

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

(подпись)

СТУДЕНТ:

группы 19-В-1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Жеглов Г. В.

(подпись)

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

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

Нижний Новгород 2022

**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**

Thymio II в базовом контроллере считывает значения с датчиков переднего расстояния и использует их для непосредственного управления скоростью своих колес. Чтобы успешно избегать препятствий с помощью механизма реагирования датчика, реализованного здесь, робот должен двигаться на полной скорости. Чтобы заставить робота двигаться на полной скорости, необходимо изменить силу воздействия датчиков расстояния на управление колесом.

Не зная абсолютного направления движения, робот может дезориентироваться и не достичь другой стороны комнаты. Для борьбы с этим Thymio II оснащен устройством компаса, хотя контроллер по умолчанию не использует его.

Код контроллера

from controller import Robot

# import Compass module

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 = 5.53

distanceSensorCalibrationConstant = 660

# 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")

# 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 = 1.7 \* maxMotorVelocity

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

leftMotor.setVelocity(initialVelocity)

rightMotor.setVelocity(initialVelocity)

# get robot's Compass device

compass = robot.getCompass("compass")

# enable the Compass

compass.enable(timeStep)

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

# 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

# to read values

values = compass.getValues()

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

if outerLeftSensorValue + centralLeftSensorValue + centralSensorValue + centralRightSensorValue + outerRightSensorValue < 0.1:

if values[0] > 0:

leftMotor.setVelocity(initialVelocity + values[0] \* 2)

else:

rightMotor.setVelocity(initialVelocity + abs(values[0]) \* 2)

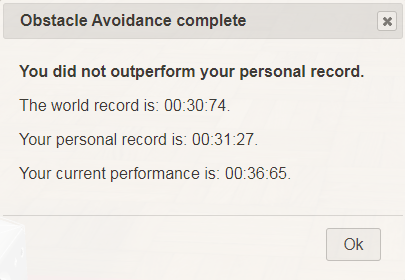
else:

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

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

print(values[0])

Результат выполнения



**Задача 2. Движение по квадрату**

Чтобы улучшить производительность надо настроить продолжительность шагов, чтобы повысить точность пути.

Также можно использовать датчики колес робота. Pioneer 3-DX имеет датчики в колесе, которые позволяют роботу знать пройденное расстояние. Используя это значение, можно заставить робота остановиться ровно через 2 метра.

Код контроллера

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')

#Get right wheel sensor

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

rightWheelSensor.enable(16)

#Get left wheel sensor

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

leftWheelSensor.enable(16)

# Current value of the wheel position sensor

sensorValue = 0

# Wheel radius

wheel\_radius = 0.0975

MAX\_SPEED = 5.24

# 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)

robot.step(16)

# Move until the current position is equal to two meters + the position before. vertexes

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

# We reduce the speed almost reaching the top

if ((rightWheelSensor.getValue() \* wheel\_radius) > (1.7 + sensorValue)):

leftWheel.setVelocity(0.7 \* MAX\_SPEED)

rightWheel.setVelocity(0.7 \* MAX\_SPEED)

robot.step(16)

# When turning, depending on the vertex, we change the position of the wheel

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() + 2.75)

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

robot.step(930)

if i == 2:

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

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

robot.step(930)

#Setting the initial speed

leftWheel.setVelocity(MAX\_SPEED)

rightWheel.setVelocity(MAX\_SPEED)

#Again we consider the current position of the sensor

sensorValue = 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 выполнения

