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:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Перевалов А. Д.

19-В-1

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

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

Нижний Новгород 2021

**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.

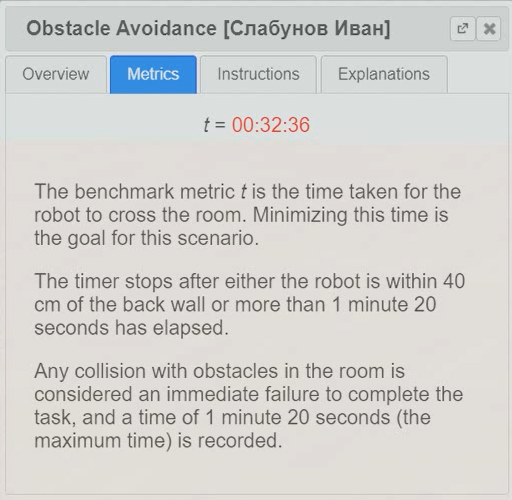
**Задача 1. Обход препятствий**

Этот тест направлен на создание надежного и эффективного алгоритма обхода препятствий для робота Thymio II с использованием языка программирования Python. Цель состоит в том, чтобы робот пересек комнату и достиг противоположной стены как можно быстрее, избегая при этом всех столкновений с препятствиями. Чтобы стимулировать устойчивое поведение избегания, препятствия располагаются случайным образом при каждом пробеге.

**Листинг:**

"""my\_avoid\_obstacles controller."""  
  
"""Braitenberg-based obstacle-avoiding robot controller."""  
  
from controller import Robot, Compass  
  
# Get reference to the robot.  
robot = Robot()  
  
# Get robot's Compass device  
compass = robot.getCompass("compass")  
  
# 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")  
  
# Enable distance sensors.  
outerLeftSensor.enable(timeStep)  
centralLeftSensor.enable(timeStep)  
centralSensor.enable(timeStep)  
centralRightSensor.enable(timeStep)  
outerRightSensor.enable(timeStep)  
  
# Enable the Compass  
compass.enable(timeStep)  
  
# Disable motor PID control mode.  
leftMotor.setPosition(float('inf'))  
rightMotor.setPosition(float('inf'))  
  
# Set the initial velocity of the left and right wheel motors.  
leftMotor.setVelocity(maxMotorVelocity)  
rightMotor.setVelocity(maxMotorVelocity)  
  
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 compass  
 compassValues = compass.getValues()  
  
 # Detected obstacles  
 if outerLeftSensorValue != 0 or centralLeftSensorValue != 0:  
 leftMotor.setVelocity(maxMotorVelocity)  
 rightMotor.setVelocity(-0.5 \* maxMotorVelocity)  
 elif centralSensorValue != 0 or outerRightSensorValue != 0 or centralRightSensorValue != 0:  
 leftMotor.setVelocity(-0.5 \* maxMotorVelocity)  
 rightMotor.setVelocity(maxMotorVelocity)  
  
 # Not detected obstacles  
 if outerLeftSensorValue == 0 and centralLeftSensorValue == 0 and centralSensorValue == 0 and centralRightSensorValue == 0 and outerRightSensorValue == 0:  
 if compassValues[0] > 0.001:  
 leftMotor.setVelocity(maxMotorVelocity)  
 rightMotor.setVelocity(0.9 \* maxMotorVelocity)  
 elif compassValues[0] < -0.001:  
 leftMotor.setVelocity(0.9 \* maxMotorVelocity)  
 rightMotor.setVelocity(maxMotorVelocity)  
 else:  
 leftMotor.setVelocity(maxMotorVelocity)  
 rightMotor.setVelocity(maxMotorVelocity)

**Результаты:**



Для обхода препятствий мы изменяли скорость колес в зависимости от значений датчиков расстояний: если датчики считывают препятствие справа (и по центру) – левое колесо резко снижает скорость, если слева – правое колесо резко снижает скорость.

Робот может дезориентироваться, так как не знает точного направления своего пути. Поэтому в нем присутствует компасное устройство, которого нет по умолчанию. Чтобы он заработал, была подключена библиотека *Compass*. Инициализация устройства и считывание значений происходит схожим образом, как и у датчиков расстояний. Для корректировки пути был написан следующий алгоритм: если на пути нет препятствий и робот движется в левую сторону (т.е. первое значение вектора > 0.001) – постепенно снижаем скорость у правого колеса до стабилизации компаса, если движется направо (т.е. первое значение вектора < -0.001) – так же постепенно снижаем скорость у левого колеса.

Общую скорость робота мы увеличили до полной, убрав в коде ограничения. Она регулируется только под воздействием датчиков расстояний и компасного устройства.

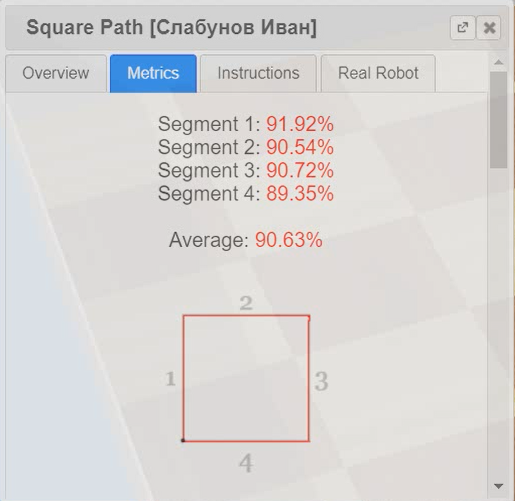
**Задача 2. Движение по квадрату**

Этот бенчмарк направлен на разработку программы с разомкнутым контуром, которая управляет роботом-пионером, чтобы он следовал квадратной траектории размером 2 на 2 метра.

**Листинг:**

"""my\_square\_path controller."""  
  
"""Sample Webots controller for the square path benchmark."""  
  
from controller import Robot  
import math  
  
# 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')  
  
# Получаем доступ к датчику положения правого колеса  
rightWheelSensor = robot.getPositionSensor('right wheel sensor')  
rightWheelSensor.enable(16)  
# Получаем доступ к датчику положения левого колеса  
leftWheelSensor = robot.getPositionSensor('left wheel sensor')  
leftWheelSensor.enable(16)  
  
# Диаметр колеса (в метрах)  
diamWheel = 0.195  
# Расстояние между колесами (в метрах)  
distWheels = 0.33  
# Текущее значение датчика положения колес (в метрах)  
sensorValue = 0  
# Расстояние, пройденное роботом за один оборот  
l = math.pi \* diamWheel  
# Max velocity  
MAX\_SPEED = 5.24  
  
# Repeat the following 4 times (once for each side).  
for i in range(0, 4):  
  
 leftWheel.setPosition(1000)  
 rightWheel.setPosition(1000)  
 robot.step(16)  
  
 while rightWheelSensor.getValue() \* diamWheel/2.0 < sensorValue + 2.0:  
 if rightWheelSensor.getValue() \* diamWheel/2.0 > sensorValue + 1.9:  
 leftWheel.setVelocity(0.6 \* MAX\_SPEED)  
 rightWheel.setVelocity(0.6 \* MAX\_SPEED)  
 robot.step(160)  
  
 if i == 0:  
 leftWheel.setPosition(leftWheelSensor.getValue() + (math.pi/2.0 \* distWheels/2.0)/l \* 2.0 \* math.pi + 0.09)  
 rightWheel.setPosition(rightWheelSensor.getValue() - (math.pi/2.0 \* distWheels/2.0)/l \* 2.0 \* math.pi - 0.05)  
 elif i == 1:  
 leftWheel.setPosition(leftWheelSensor.getValue() + (math.pi/2.0 \* distWheels/2.0)/l \* 2.0 \* math.pi + 0.06)  
 rightWheel.setPosition(rightWheelSensor.getValue() - (math.pi/2.0 \* distWheels/2.0)/l \* 2.0 \* math.pi - 0.05)  
 elif i == 2:  
 leftWheel.setPosition(leftWheelSensor.getValue() + (math.pi/2.0 \* distWheels/2.0)/l \* 2.0 \* math.pi + 0.11)  
 rightWheel.setPosition(rightWheelSensor.getValue() - (math.pi/2.0 \* distWheels/2.0)/l \* 2.0 \* math.pi)  
 elif i == 3:  
 continue  
 robot.step(912)  
   
 sensorValue = rightWheelSensor.getValue() \* diamWheel/2  
  
 leftWheel.setVelocity(MAX\_SPEED)  
 rightWheel.setVelocity(MAX\_SPEED)  
  
# 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)

**Результаты:**



Для прохождения нужного нам расстояния (2 на 2 метра) были подключены датчики положения, считывающие каждые 16 мс текущее положение колес робота (в радианах). В самом алгоритме выполняется цикл «пока робот не проедет 2 метра», где сравнивается значение текущего положения правого колеса, помноженное на его радиус, с положением правого колеса относительно предыдущей вершины + 2 метра (т.е. сравниваются метры, а не радианы). При достижении вершины, робот снижает свою скорость на 40%, после поворота возвращает.

Для того чтобы повернуть робота на 90 градусов, мы так же воспользовались датчиками положения. В зависимости от вершины, устанавливаются различные позиции колес: текущее положение колеса угол поворота погрешность. Угол поворота рассчитывается исходя из расстояния между колесами.

Из-за того, что значения датчиков возвращаются периодически, достигнуть идеального результата довольно трудно, возникают погрешности.