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 №3

по дисциплине «Аппаратное и программное обеспечение   
роботизированных систем»

«Программное обеспечение  
роботизированных систем»

ПРОВЕРИЛ:

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

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Круглов С.В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Малинин М.Ю.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Попов В.А.

\_\_\_\_\_\_\_\_\_\_\_\_ Дятлов Д.

**17-В-1**

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

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

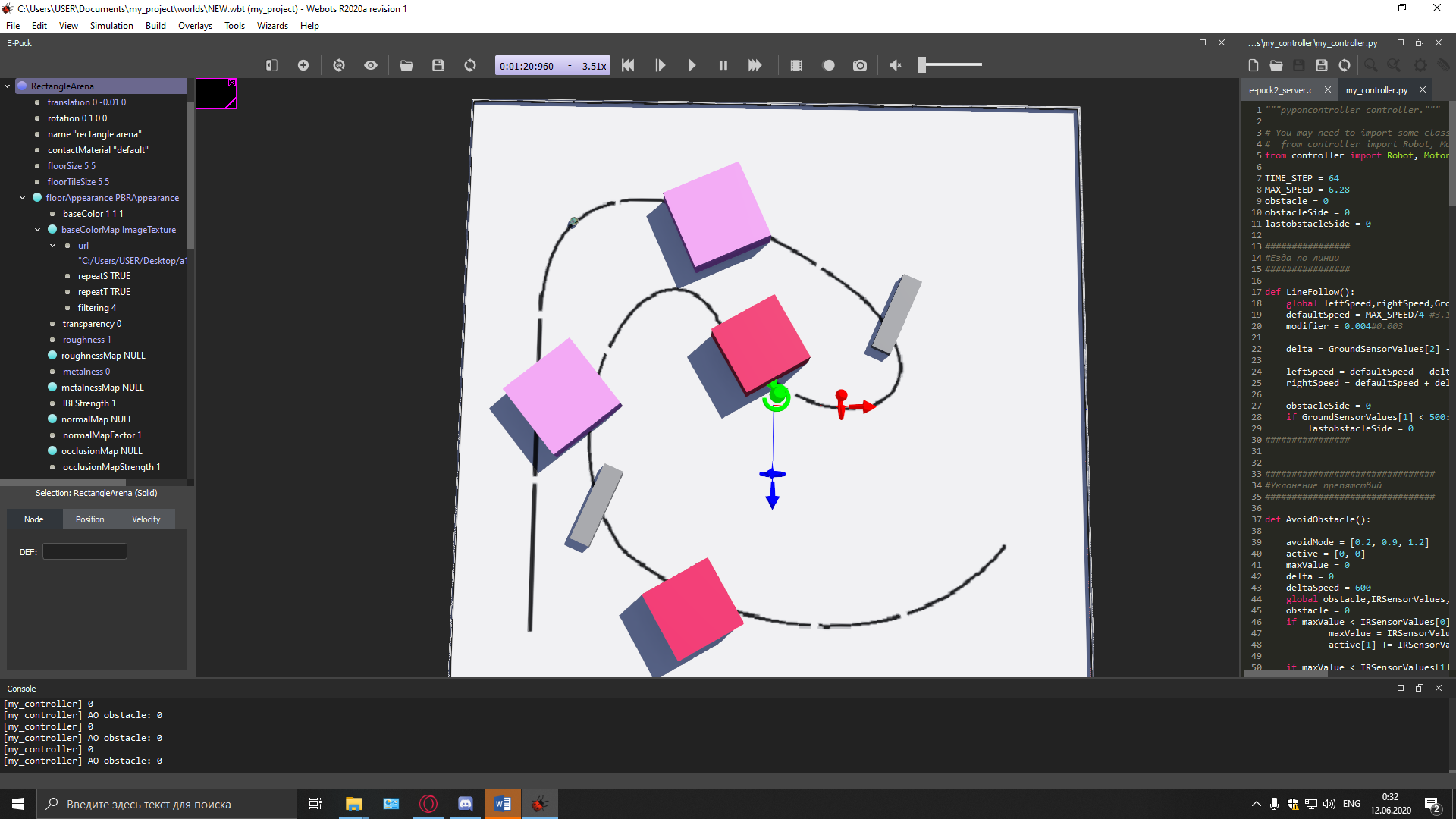
Нижний Новгород

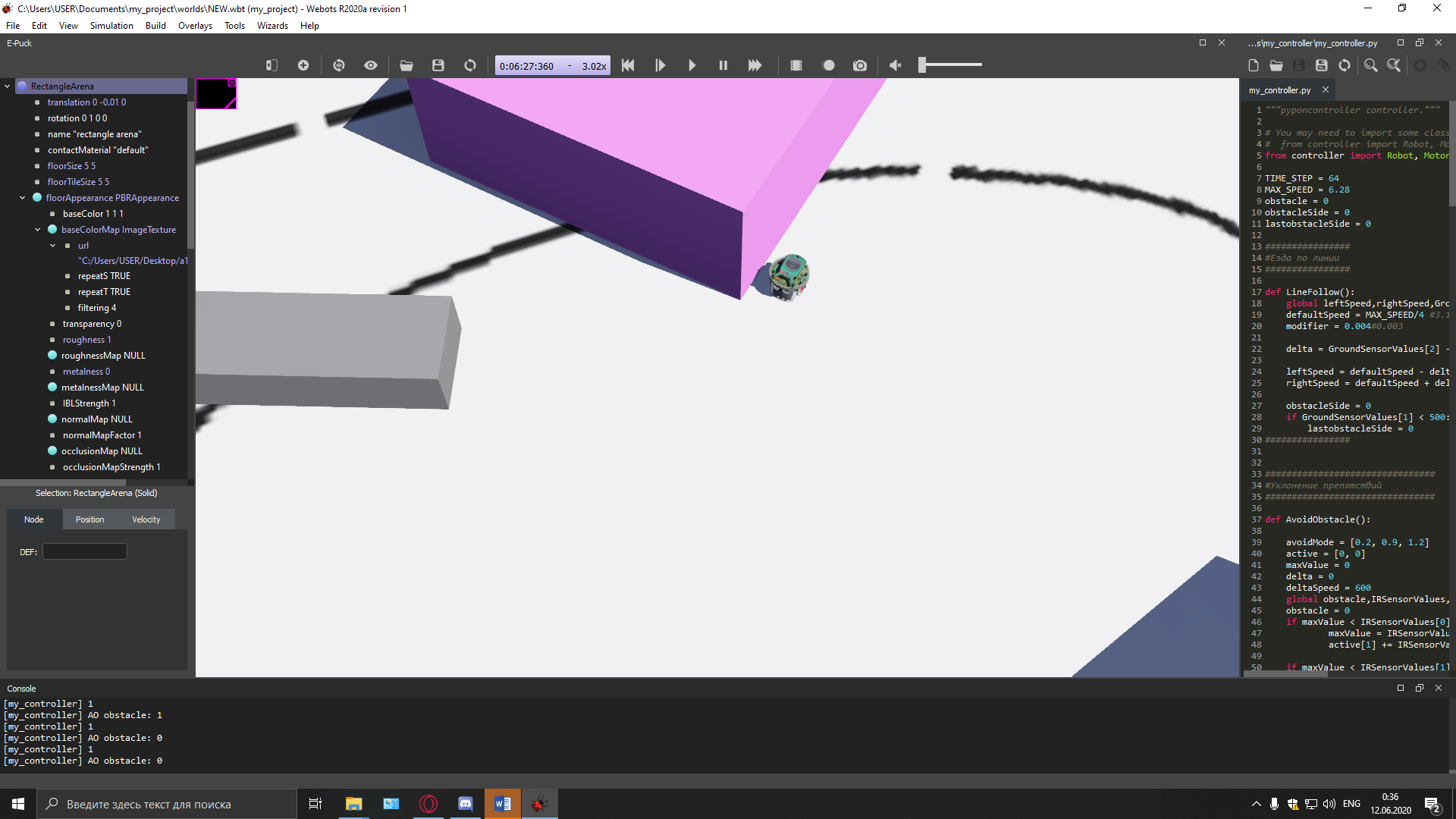
2020

**Цель:** написать программу для робота по вариантам в среде симуляции Webots.

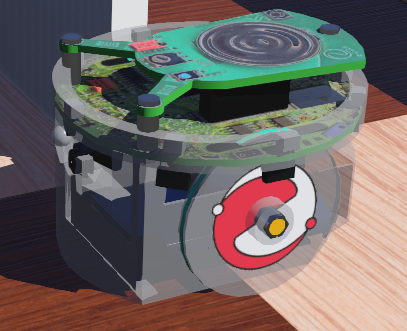
**Вариант:** **обход лабиринта:** робот должен ездить по линии и объезжать препятствия.

**Ход работы:** для начала была создана поле 10х10 в среде Webots и нанесена разметка так же были установлены объекты преграждающие путь, линия в некоторых местах прерывается.





Для выполнения задания был взят робот «e-puck». <https://cyberbotics.com/doc/guide/epuck>



**Описание алгоритма работы робота:**

Рассматривает 4 передних датчика смотрит на наличие препятствий, суммирует 3 левых датчика, суммируем 3 правых датчика и находим их разность на основе этого перенимаем решение попорота с помощью некоторых коэффициентов и суммируем эту разность к скорости левого мотора и вычитаем эту разность из правого мотора.

**Код контроллера:**

"""pyponcontroller controller."""

# You may need to import some classes of the controller module. Ex:

# from controller import Robot, Motor, DistanceSensor

from controller import Robot, Motor, DistanceSensor

TIME\_STEP = 64

MAX\_SPEED = 6.28

obstacle = 0

obstacleSide = 0

lastobstacleSide = 0

################

#Езда по линии

################

def LineFollow():

global leftSpeed,rightSpeed,GroundSensorValues,obstacleSide,lastobstacleSide

defaultSpeed = MAX\_SPEED/4 #3.14

modifier = 0.004#0.003

delta = GroundSensorValues[2] - GroundSensorValues[0]

leftSpeed = defaultSpeed - delta \* modifier

rightSpeed = defaultSpeed + delta \* modifier

obstacleSide = 0

if GroundSensorValues[1] < 500:

lastobstacleSide = 0

################

################################

#Уклонение препятствий

################################

def AvoidObstacle():

avoidMode = [0.2, 0.9, 1.2]

active = [0, 0]

maxValue = 0

delta = 0

deltaSpeed = 600

global obstacle,IRSensorValues,obstacleSide,leftSpeed,rightSpeed,lastobstacleSide

obstacle = 0

if maxValue < IRSensorValues[0]:

maxValue = IRSensorValues[0]

active[1] += IRSensorValues[0]

if maxValue < IRSensorValues[1]:

maxValue = IRSensorValues[1]

active[1] += IRSensorValues[1]

# if maxValue < IRSensorValues[2]:

# maxValue = IRSensorValues[2]

# active[1] += IRSensorValues[2]

# if maxValue < IRSensorValues[5]:

# maxValue = IRSensorValues[5]

# active[0] += IRSensorValues[5]

if maxValue < IRSensorValues[6]:

maxValue = IRSensorValues[6]

active[0] += IRSensorValues[6]

if maxValue < IRSensorValues[7]:

maxValue = IRSensorValues[7]

active[0] += IRSensorValues[7]

if maxValue > 100:

obstacle = 1

# else:

# obstacle = 0

if obstacleSide == 0 and obstacle == 1:

if active[1] > active[0]:

obstacleSide = 1

lastobstacleSide = 1

else:

lastobstacleSide = -1

print (lastobstacleSide)

# if obstacle == 0:

# obstacleSide = 0

if obstacle == 1:

leftSpeed = 6.28 / 8

rightSpeed = 6.28 / 8

if obstacleSide == -1:

for i in range(3):

delta -= avoidMode[i] \* IRSensorValues[i]

else:

for i in range(3):

delta += avoidMode[i] \* IRSensorValues[i + 5]

if delta > deltaSpeed:

delta = deltaSpeed

if delta < -deltaSpeed:

delta = -deltaSpeed

leftSpeed -= delta/100

rightSpeed += delta/100

print('AO obstacle: ' + str(obstacleSide))

################################

################################

#ОбЪезд препятствий - подправляет робота в сторону препятствия, компенсируя поворот от уклонения

################################

def ObstacleTrack():

global leftSpeed,rightSpeed, obstacleSide, lastobstacleSide

defaultSpeed = MAX\_SPEED/8

if lastobstacleSide == -1: #препятствие слева

leftSpeed -= defaultSpeed

rightSpeed += defaultSpeed

if lastobstacleSide == 1: #препятствие справа

leftSpeed += defaultSpeed

rightSpeed -= defaultSpeed

################################

################################

#Возвращение на линию

################################

prevValue = 900

lastobstacleside = 0

BTLworking = 0

def BackToLine():

global leftSpeed,rightSpeed,GroundSensorValues,obstacleSide, prevValue, lastobstacleside, BTLworking

turnValue = 50

if obstacleSide != 0:

defaultBTLSpeed = MAX\_SPEED/16

curValue = GroundSensorValues[1]

print('CV: ' + str(curValue) + ' PV:' + str(prevValue))

if (curValue > 500) and (prevValue < 500): #сошел с линии

lastobstacleside = obstacleSide

if (curValue < 500) and (prevValue > 500): #нашел линию

print('found')

BTLworking = 1

if (BTLworking == 1) :

delta = GroundSensorValues[2] - GroundSensorValues[0]

print(delta)

print(BTLworking)

if lastobstacleside == -1 :

leftSpeed = defaultBTLSpeed

rightSpeed = -defaultBTLSpeed

elif lastobstacleside == 1 :

leftSpeed = -defaultBTLSpeed

rightSpeed = defaultBTLSpeed

if (delta < turnValue) and (delta > -turnValue) and (curValue <500):

BLTWorking = 0

lastobstacleside = 0

obstacleSide = 0

prevValue = curValue

print(str(obstacleSide) + ' ' + str(lastobstacleside))

################################

robot = Robot()

# get the time step of the current world.

timestep = int(robot.getBasicTimeStep())

# You should insert a getDevice-like function in order to get the

# instance of a device of the robot. Something like:

# motor = robot.getMotor('motorname')

# ds = robot.getDistanceSensor('dsname')

# ds.enable(timestep)

leftMotor = robot.getMotor('left wheel motor')

rightMotor = robot.getMotor('right wheel motor')

#Инициализация ИК датчиков

IRSensorNames = ['ps0','ps1','ps2','ps3',

'ps4','ps5','ps6','ps7']

IRSensors = []

for sensor in IRSensorNames:

IRSensors.append(robot.getDistanceSensor(sensor))

#----------

#Инициализация датчиков линии

GroundSensorNames = ['gs0','gs1','gs2']

GroundSensors = []

for sensor in GroundSensorNames:

GroundSensors.append(robot.getDistanceSensor(sensor))

#----------

#включаем все сенсоры

for sensor in GroundSensors:

sensor.enable(TIME\_STEP)

for sensor in IRSensors:

sensor.enable(TIME\_STEP)

#----------

#Инициализация моторов

leftMotor.setPosition(float('inf'))

rightMotor.setPosition(float('inf'))

leftMotor.setVelocity(0.0)

rightMotor.setVelocity(0.0)

leftSpeed = 0

rightSpeed = 0

# Main loop:

# - perform simulation steps until Webots is stopping the controller

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

GroundSensorValues = []

IRSensorValues = []

# Process sensor data here.

for sensor in GroundSensors:

GroundSensorValues.append(sensor.getValue())

for sensor in IRSensors:

IRSensorValues.append(sensor.getValue())

# for sensor in GroundSensorValues:

# print(str(sensor))

LineFollow()

AvoidObstacle()

ObstacleTrack()

# BackToLine()

# if leftSpeed>6.28: leftSpeed = 6.28

# if rightSpeed>6.28: rightSpeed = 6.28

leftMotor.setVelocity(leftSpeed)

rightMotor.setVelocity(rightSpeed)

# print(str(leftSpeed) + " " + str(rightSpeed))

# Enter here functions to send actuator commands, like:

# motor.setPosition(10.0)

pass

# Enter here exit cleanup code.