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 №2

по дисциплине «Аппаратное и программное обеспечение   
роботизированных систем»

«Знакомство с симулятором Webots»

ПРОВЕРИЛ:

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

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Максимова О.В.

**17-В-1**

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

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

Нижний Новгород

2020

**Цель:** познакомиться с симулятором Webots. Реализовать для конкретного робота (по вариантам) программу движения, а также создать новый мир и добавить в него объекты.

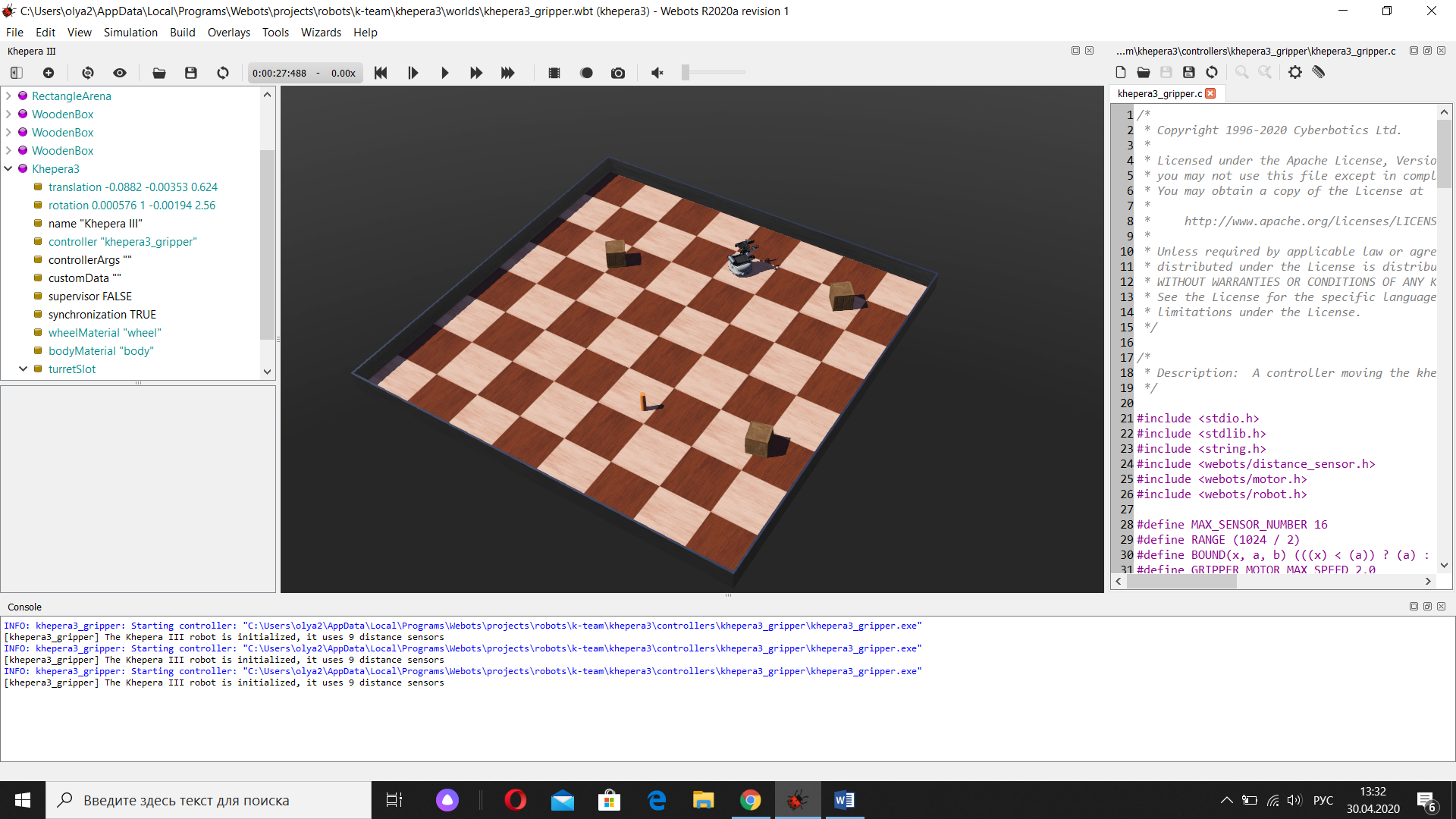
**Вариант:** робот Khepera 3 gripper.

**О роботе:**

Модель Khepera 3 gripper – 2-х колесный робот, предоставлена ​​корпорацией K-Team. Ему установлены несколько датчиков, включая 8 датчиков расстояния.



**Мир.** Реализованный мир представлен ниже, а также к данному отчету приложен: видео – пример моделирования, а также сам созданный симулятор - мир в формате данного симулятора.



**Текст контроллера робота:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <webots/distance\_sensor.h>

#include <webots/motor.h>

#include <webots/robot.h>

#define MAX\_SENSOR\_NUMBER 16

#define RANGE (1024 / 2)

#define BOUND(x, a, b) (((x) < (a)) ? (a) : ((x) > (b)) ? (b) : (x))

#define GRIPPER\_MOTOR\_MAX\_SPEED 2.0

static WbDeviceTag sensors[MAX\_SENSOR\_NUMBER];

static WbDeviceTag gripper\_motors[3];

static WbDeviceTag left\_motor, right\_motor;

static const double matrix[9][2] = {{-2.67, -2.67}, {-10.86, 21.37}, {-16.03, 26.71}, {-37.4, 37.4}, {37.4, -32.06},

{26.71, -21.37}, {21.37, -10.86}, {-2.67, -2.67}, {-5.34, -5.34}};

static const int num\_sensors = 9;

static const double range = 2000.0;

static int time\_step = 0;

static const double max\_speed = 19.1;

static void initialize() {

/\* necessary to initialize Webots \*/

wb\_robot\_init();

time\_step = wb\_robot\_get\_basic\_time\_step();

char sensors\_name[5];

sprintf(sensors\_name, "%s", "ds0");

int i;

for (i = 0; i < num\_sensors; i++) {

sensors[i] = wb\_robot\_get\_device(sensors\_name);

wb\_distance\_sensor\_enable(sensors[i], time\_step);

if ((i + 1) >= 10) {

sensors\_name[2] = '1';

sensors\_name[3]++;

if ((i + 1) == 10) {

sensors\_name[3] = '0';

sensors\_name[4] = '\0';

}

} else {

sensors\_name[2]++;

}

}

gripper\_motors[0] = wb\_robot\_get\_device("horizontal\_motor");

gripper\_motors[1] = wb\_robot\_get\_device("left\_finger\_motor");

gripper\_motors[2] = wb\_robot\_get\_device("right\_finger\_motor");

left\_motor = wb\_robot\_get\_device("left wheel motor");

right\_motor = wb\_robot\_get\_device("right wheel motor");

wb\_motor\_set\_position(left\_motor, INFINITY);

wb\_motor\_set\_position(right\_motor, INFINITY);

wb\_motor\_set\_velocity(left\_motor, 0.0);

wb\_motor\_set\_velocity(right\_motor, 0.0);

const char \*robot\_name = wb\_robot\_get\_name();

printf("The %s robot is initialized, it uses %d distance sensors\n", robot\_name, num\_sensors);

}

void step(double seconds) {

const double ms = seconds \* 1000.0;

int elapsed\_time = 0;

while (elapsed\_time < ms) {

wb\_robot\_step(time\_step);

elapsed\_time += time\_step;

}

}

void braitenberg() {

while (wb\_robot\_step(time\_step) != -1) { // Run simulation

int i, j;

double speed[2];

double sensors\_value[num\_sensors];

for (i = 0; i < num\_sensors; i++)

sensors\_value[i] = wb\_distance\_sensor\_get\_value(sensors[i]);

/\*

\* The Braitenberg algorithm is really simple, it simply computes the

\* speed of each wheel by summing the value of each sensor multiplied by

\* its corresponding weight. That is why each sensor must have a weight

\* for each wheel.

\*/

for (i = 0; i < 2; i++) {

speed[i] = 0.0;

for (j = 0; j < num\_sensors; j++) {

/\*

\* We need to recenter the value of the sensor to be able to get

\* negative values too. This will allow the wheels to go

\* backward too.

\*/

speed[i] += matrix[j][i] \* (1.0 - (sensors\_value[j] / range));

}

speed[i] = BOUND(speed[i], -max\_speed, max\_speed);

}

/\* Set the motor speeds \*/

wb\_motor\_set\_velocity(left\_motor, speed[0]);

wb\_motor\_set\_velocity(right\_motor, speed[1]);

}

}

void moveArms(double position) {

wb\_motor\_set\_velocity(gripper\_motors[0], GRIPPER\_MOTOR\_MAX\_SPEED);

wb\_motor\_set\_position(gripper\_motors[0], position);

}

void moveFingers(double position) {

wb\_motor\_set\_velocity(gripper\_motors[1], GRIPPER\_MOTOR\_MAX\_SPEED);

wb\_motor\_set\_velocity(gripper\_motors[2], GRIPPER\_MOTOR\_MAX\_SPEED);

wb\_motor\_set\_position(gripper\_motors[1], position);

wb\_motor\_set\_position(gripper\_motors[2], -position);

}

void moveForwards(double speed) {

wb\_motor\_set\_velocity(left\_motor, speed);

wb\_motor\_set\_velocity(right\_motor, speed);

}

void turn(double speed) {

wb\_motor\_set\_velocity(left\_motor, speed);

wb\_motor\_set\_velocity(right\_motor, -speed);

}

void stop(double seconds) {

wb\_motor\_set\_velocity(left\_motor, 0.0);

wb\_motor\_set\_velocity(right\_motor, 0.0);

step(seconds);

}

int main() {

initialize();

stop(1.0);

moveForwards(12.8);

moveArms(-3.0);

step(1.0);

stop(1.0);

moveFingers(0.42);

step(1.0);

moveArms(0.0);

moveForwards(-12.8);

moveArms(0.0);

step(1.0);

stop(1.0);

turn(-2.4);

step(0.5);

moveForwards(12.8);

step(0.6);

stop(0.25);

moveArms(-1.6);

step(1.5);

moveFingers(0.0);

step(1.0);

moveArms(0.0);

moveFingers(0.0);

step(1.0);

moveForwards(-12.8);

step(1.0);

turn(-10.7);

step(1.0);

braitenberg();

wb\_robot\_cleanup();

return 0;

}

**Выводы:** в рамках выполнения данной лабораторной работы был опробован симулятор Webots, написан контроллер для робота, а также изучены некоторые главы документации и гайды.