Работа победителя заключительного этапа

командной инженерной олимпиады школьников

Олимпиада Национальной технологической инициативы

Профиль «Интеллектуальные робототехнические системы»

Гариханов Айдар Дамирович

Класс: 10

Город: Иннополис

Школа: ГАОУ "Лицей Иннополис"

Регион: Республика Татарстан

Уникальный номер участника: 949

Команда на заключительном этапе: КиберТунец

Параллель: 10-11

Результаты заключительного этапа:

Индивидуальная часть													Командная часть					Результат
№	Математика			Информатика							Итого	Макс. балл	Решение задач этапов				Итого	
	1	2	3	1.1	1.2	1.3	2.1	2.2	3.1	3.2			№1	№2	№3	№4		
949	16	27	0	6	9	15	0	0	0	0	73	200	25	0	31	24	80	153

Индивидуальная часть

Персональный лист участника с номером 949:

Олимпиада НТИ

ФИО Гарифанов Айдар Дамирович

Город Иннополис

Школа № ГАОУ „Лицей Иннополис"

Математика

Лист 1:

Командная инженерная олимпиада «Олимпиада НТИ»

Направление Интеллектуальные робототехнические системы

Предмет математика

Номер участника 949

1	2а	2б	3а	3б	3в	Σ
16	7	20	0	0	0	43

1. Заметим, что на каждом шаге переливается 200-100=100 мл из одной кружки (в которой 200 мл) в другую (в которой 100 мл). Концентрация в кружке из которой переливали остается такой же, а в кружке в которую переливали изменяется и рассчитывается, как

$\eta = \frac{m}{M}$, где m – масса (объем) кофе в стакане
M – масса (объем) всей жидкости в стакане

$\eta = \frac{M_1\eta_1 + M_2\eta_2}{M_1 + M_2}$ $M_{1/2}$ – масса (объем) жидкости в 1/2 стакане (во первом, либо втором)
$\eta_{1/2}$ – конц. жидкости в 1/2 стакане

$M_1 = M_2 = 100$ мл (т.к. из кружки переливают 100 мл, в кружку с 100 мл)

Тогда

$\eta = \frac{M_1(\eta_1 + \eta_2)}{2M_1} = \frac{\eta_1 + \eta_2}{2}$

Рассчитаем конц. на каждом шаге

шаг	η_1	η_2	m_1	m_2
0	1	0	200	100
1	1	0,5	100	200
2	0,75	0,5	200	100
3	0,75	0,625	100	200
4	0,6875	0,625	200	100
5	0,6875	0,65625	100	200
6	0,671875	0,65625	200	100
7	0,671875	0,6640625	100	200
8	0,66796875	0,6640625	200	100

m_{xy} – масса (объем) жидкости в x-ом стакане на y-ом шаге

На 8 шаге $\eta_1 - \eta_2 < 0,01 = 1\%$ и $m_{18} = m_{10}$ и $m_{28} = m_{10}$

Ответ: 8 переливаний

+

Лист 2:

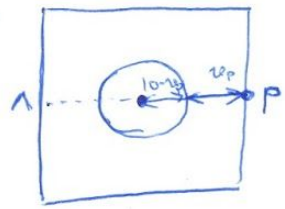
Командная инженерная олимпиада «Олимпиада НТИ»

Направление ИРС

Предмет математика

Номер участника 919

2.



Угловая скорость лазера

$\omega_л = \frac{2\pi}{T}$, где T - период; $T = 1$ мин

$\omega_л = \frac{2\pi}{1} = 2\pi \frac{рад}{мин} \approx 6{,}283$

Угловая скорость робота

$\omega_р = \frac{v_р}{R}$, где $v_р$ - скорость робота

R - радиус окружности, вдоль которой движется робот.

б) Пусть $v_р \in (\frac{20\pi}{2\pi+1} \frac{м}{мин}; 10 \frac{м}{мин})$

Тогда робот сможет добраться до центра комнаты!

1) когда лазер проходит рядом со стартом робота, робот начинает двигаться напрямую к центру.

2) немного не доезжая до точки, удаленной от центра на расстояние $(10-v_р)$, робот начинает двигаться по окружности с радиусом $(10-v_р)$ и центром в центре комнаты, „убегая" от лазера.

3) т.к. $\omega_р > \omega_л$, робот начнет догонять лазер с другой стороны.

4) как только он почти добрался до лазера, он продолжает свое движение к центру комнаты.

5) он доберется до стены за время, равное $\frac{10-v_р}{v_р} < 1$ мин, и лазер не успеет его заметить.

а) Это невозможно, т.к. скорость робота должна быть больше $\frac{20\pi}{2\pi+1}$, но $v_р < \frac{20\pi}{2\pi+1}$

Рассчитаем скорость робота, х что при движении по окружности радиусом $10-x$, его скорость (угловая была) бы больше $\omega_л$

$\omega_р > \omega_л$

$\frac{x}{10-x} > 2\pi$

$\frac{x - 2\pi(10-x)}{10-x} > 0 \quad | \cdot(-1)$

$\frac{x - 2\pi \cdot 10 + 2\pi x}{x-10} < 0$

$\frac{(2\pi+1)x - 20\pi}{x-10} < 0 \quad | : 2\pi+1$

$\frac{x - \frac{20\pi}{2\pi+1}}{x-10} < 0$ +

$x \in (\frac{20\pi}{2\pi+1}; 10)$

$\frac{20\pi}{2\pi+1} < 10$

$\frac{20\pi}{2\pi+1} = \frac{20\pi+10}{2\pi+1} - \frac{10}{2\pi+1} =$

$= 10 - \frac{10}{2\pi+1} < 10$

−

А почему он обязан двигаться до окр, а потом по ней?

Оценка за задачу №1: 16

Комментарий к решению: Задача решена верно.

Оценка за задачу №2: 27

Комментарий к решению: Задача решена не полностью.

Оценка за задачу №3: 0

Комментарий к решению: Решение не предложено.

Информатика

Задача №1

Задача решена для всех трех групп тестов (30 баллов).

Код программы на языке C++, написанный участником и решающий задачу:

```
#include <iostream>
#include <vector>
#include <set>
#include <algorithm>

struct container {
	int num, size;
};

bool operator <(const container& a, const container& b) {
	return (a.size < b.size);
}

using namespace std;

int main()
{
	int n, k, m;
	cin >> n >> k >> m;
	vector <int> h;
	h.resize(n + 1);
	for (int i = 1; i <= n; i++)
		cin >> h[i];
	multiset <container> left;
	multiset <container> right;
	int max_left = min(1 + k, n);
	int min_right = max(n + 1 - k, 2);
	for (int i = 2; i <= max_left; i++) {
		container t;
		t.num = i;
		t.size = h[i];
		left.insert(t);
	}
	for (int i = min_right; i <= n; i++) {
		container t;
		t.num = i;
```

```
            t.size = h[i];
            right.insert(t);
        }
        int time_left = 0;
        int time_right = 0;
        while (max_left < m) {
            time_left += (*left.begin()).size;
            left.erase(left.begin());
            max_left++;
            container t;
            t.num = max_left;
            t.size = h[max_left];
            left.insert(t);
        }
        while (min_right > m) {
            time_right += (*right.begin()).size;
            cout << endl;
            right.erase(right.begin());
            min_right--;
            container t;
            t.num = min_right;
            t.size = h[min_right];
            right.insert(t);
        }
        cout << min(time_left, time_right) + h[m];
    return 0;
}
```

Задача №2

Решение не было предложено (0 баллов).

Задача №3

Решение не было предложено (0 баллов).

Командная часть

Результаты были получены в рамках выступления команды: КиберТунец

Личный состав команды:

- Гариханов Айдар Дамирович
- Усманов Данил Рустемович
- Евграфов Максим Александрович

Фотография команды и робототехнического устройства, оснащенного датчиками, перед проверочными испытаниями последнего этапа:

Протокол выполнения заданий:

Первый этап:

№ задания	Критерий	Возможные баллы	Полученные баллы
1	Робот покинул сектор старта	6x2	12
2	Робот проехал половину от всего количества секторов от сектора старта до сектора финиша	13x2	13

3	Робот проехал от сектора старта до черной линии (в ходе движения робот пересек черную линию или остановился таким образом, что черная линия пересекает проекцию робота на поле)	6x2	0
4	Робот проехал от сектора старта, заехал в сектор финиша (проекция робота на поле внутри сектора) и остановился	6x2	0
5	Робот проехал от сектора старта, остановился в секции финиша (проекция робота внутри сектора) и вывел на экран верное количество пройденных секций	12x2	0
Дополнительные баллы		19	0
Всего за этап		105	25

Второй этап:

№ задания	**Критерий**	**Возможные баллы**	**Полученные баллы**
6	Робот проехал по периметру вокруг циклической структуры и остановился после определения цикла (допускается проезд по второму кругу до 3х секторов)	7x2	0
7	Робот после верного определения циклической структуры, остановился на 3 секунды, и продолжил движение по другим секторам полигона, не принадлежащим циклической структуре (движение не менее, чем по 5 не принадлежащим циклической структуре)	12x2	0
8	Робот проехал не меньше 7 секторов по полигону и остановился; на экране отображена отображены координаты смещения относительно сектора старта: первое число – смещение по оси X, второе число – смещение по оси Y. При этом ось X совпадает с направлением робота в секторе старта, а ось Y направлена вправо. Данный критерий не будет оцениваться, если робот способен выполнять сразу следующий пункт списка критериев.	6x2	0
9	После остановки и отображения карты на экране устройства отображается 2 числа, определяющее позицию, где произошла остановка: первое число – сектор по оси X, второе число – сектор по оси Y (см. рис. 5)	31x2	0
Дополнительные баллы		19	0
Всего за этап		119	0

Третий этап:

№ задания	**Критерий**	**Возможные баллы**	**Полученные баллы**

10	Робот доехал до сектора сервисного обслуживания (проекция робота внутри сектора) и остановился	6x2	12
11	После остановки робота на экран выведено закодированное значение.	19x2	19
Дополнительные баллы		12	0
Всего за этап		62	31

Четвертый этап:

№ задания	**Критерий**	**Возможные баллы**	**Полученные баллы**
12	После запуска программы, роботе стоит на месте в течение на 10 секунд. На экран выведен путь от сектора старта до сектора финиша. При этом путь выведен в виде координат секций, через которые должен пройти робот. Данный критерий не будет оцениваться, если робот способен выполнять сразу следующий пункт списка критериев.	6x2	0
13	После запуска программы, роботе стоит на месте в течение на 10 секунд. На экран выведен путь от сектора старта до сектора финиша. При этом путь выведен в виде виде алгоритма перемещения (F - одна секция прямо, R- поворот направо, L - поворот налево) с учетом необходимых поворотов в секторе старта.	9x2	18
14	Путь, выведенный после запуска робота, является оптимальным по количеству секторов, необходимых для посещения.	3x2	6
15	Путь перемещения робота из сектора старта до сектора финиша является оптимальным по количеству секторов.	3x2	0
16	Робот доехал до финального сектора и остановился. И издал звуковой сигнал.	6x2	0
Дополнительные баллы		12	0
17	Робот выехал из сектора старта и остановился на 10 секунд в первом (по ходу движения робота) секторе сервисного обслуживания. Перед началом отсчета таймера робот подает звуковой сигнал.	6	0
18	После остановки робота в секторе сервисного обслуживания выведена одна компонента координаты финального сектора вместе с идентификатором оси (буква X или Y). Выведенное число совпадает со штрих-кодом нанесенным в данном секторе сервисного обслуживания.	5	0
19	Робот продолжил движение и остановился на 10 секунд во втором секторе сервисного обслуживания. Перед началом отсчета таймера робот подает звуковой сигнал.	11	0

20	После остановки робота в секторе сервисного обслуживания выведены обе компоненты координаты финального сектора вместе с идентификаторами осей (буква X или Y). Координата совпадает со штрих-кодом нанесенным в данном секторе сервисного обслуживания.	5	0
21	Путь перемещения робота из сектора, где робот локализовался до очередного сектора сервисного обслуживания или финального сектора является оптимальным по количеству секторов.	5	0
22	Робот доехал до финального сектора и остановился. И издал звуковой сигнал.	28	0
Всего за этап		114	24

Программа команды для решения задачи первого этапа:

```
pi = 3.141592653589793
wait = script.wait
sign = function(n) {
  return n > 0 ? 1 : n = 0 ? 0 : -1
}
sqr = function(n) {
  return n * n
}
sqrt = Math.sqrt
min = function(a, b) {
  return a < b ? a : b
}
max = function(a, b) {
  return a > b ? a : b
}

abs = Math.abs
sin = Math.sin
cos = Math.cos
round = Math.round

isFoundLine = false
count = 0

// Параметры робота
d = 5.6 // Диаметр колеса
w = 20.6 // База (расстояние от колеса до
 колеса)
cpr = 240 // Показания энкодера за один
```

```
оборот
x = 0 // Начальные координаты робота
y = 0
a = 0 // Начальный угол робота

// Моторы
mLeft = brick.motor(M4).setPower
mRight = brick.motor(M3).setPower

// Энкодеры
eLeft = function() {
  return -brick.encoder(E4).readRawData()
}

eRight = function() {
  return brick.encoder(E3).readRawData()
}

// Датчики освещения
cLeft = brick.sensor(A5)
cRight = brick.sensor(A6)

// Ультразвуковые датчики
uRight = brick.sensor(D1)

// Ифракрасный датчик
iForward = brick.sensor(A1)

// Гироскоп
gyro = brick.gyroscope()

// Остановка робота
function stop() {
  brick.motor(M3).forceBreak()
  brick.motor(M4).forceBreak()
}

function stop2() {
  mLeft(0)
  mRight(0)
}

// Сброс показаний энкодера
```

```
function resetEncoders() {
  brick.encoder(E3).reset()
  brick.encoder(E4).reset()
}

// Движения робота на расстояние dist
function move(dist) {
  resetEncoders()
  target = cpr * dist / pi / d
  kp = 0.2
  while (eLeft() + eRight() < 2 * target) {
    u = (eLeft() - eRight()) * kp
    v = 30
    mLeft(v - u)
    mRight(v + u)
    wait(10)
  }
  stop()
}

function turn(angle) {
  resetEncoders()
  x = w * angle * cpr / 360 / d
  v = 30
  mLeft(v * sign(angle))
  mRight(-v * sign(angle))
  while (abs(eLeft() - eRight() - 2 * x) > 10)
    wait(10)
  stop()
}

// Проехать вперед на один сегмент
function moveSegment() {
  print("move_forward")
  move(36)
  count++
}

// Поворот налево
function turnLeft() {
  turn(-85.5)
}
```

```
// Поворот направо
function turnRight() {
  turn(85.5)
}

function calibrate() {
  for (i = 0; i < 120; i++) {
    mRight(-40)
    mLeft(-30)
    wait(10)
  }
}

// Свободно ли справа
function isFreeRight() {
  return (uRight.read() > 25)
}

function isFreeForward() {
  print(iForward.read())
  return (iForward.read() > 25)
}

function check() {
  if (cLeft.read() > 60 || cRight.read() > 60) {
    isFoundLine = true
    timer.stop()
    print("STOPPED!")
  }
}

// Проезд по лабиринту по правилу правой руки
function RightHand() {
  while (!isFoundLine) { // Пока не нашли черную линию
    if (isFreeRight()) { // Если справа свободно
      print('FreeRight')
      turnRight() // Поворот направо
      moveSegment() // Проезд на 1 сегмент вперед
    } else if (isFreeForward()) { // Если спереди свободно
      print('FreeForward')
```

```
      moveSegment() // Проезд на 1 сегмент вперед
    } else { // Если тупик
      print("Deadend")
      turnLeft() // Поворот налево
      calibrate()
      move(10)
    }
    wait(500)
  }
  stop()
}

var main = function() {
  print("START")
  timer = script.timer(50)
  timer.timeout.connect(check)
  RightHand()
  brick.display().addLabel("count = " + count, 1, 20);
  brick.display().redraw()
  for (i = 0; i < 300; i++)
    wait(10)
  print("END")
  return;
}
```

Программы для решения задачи “определение циклической структуры” второго этапа нет.

Программа команды для решения задачи “локализация” второго этапа:

```
pi = 3.141592653589793
wait = script.wait
sign = function(n) {
  return n > 0 ? 1 : n = 0 ? 0 : -1
}
sqr = function(n) {
  return n * n
}
sqrt = Math.sqrt
min = function(a, b) {
  return a < b ? a : b
}
```

```
max = function(a, b) {
  return a > b ? a : b
}
abs = Math.abs
sin = Math.sin
cos = Math.cos
round = Math.round

//adj[i][j] содержит номер помещения, который
//находится по направлению 0 <= j <= 3 от
//i-ого помещения (0 если такого нет)
adj = []
adj[0] = [-1, 1, 7, -1]
adj[1] = [-1, 2, -1, 0]
adj[2] = [-1, 3, 8, 1]
adj[3] = [-1, 4, -1, 2]
adj[4] = [-1, 5, -1, 3]
adj[5] = [-1, -1, 10, 4]
adj[6] = [-1, -1, 12, -1]
adj[7] = [0, -1, 13, -1]
adj[8] = [2, -1, 15, -1]
adj[9] = [-1, -1, 16, -1]
adj[10] = [5, 11, 18, -1]
adj[11] = [-1, 12, -1, 10]
adj[12] = [6, -1, -1, 11]
adj[13] = [7, 13, -1, -1]
adj[14] = [-1, 15, -1, 13]
adj[15] = [8, -1, 21, 14]
adj[16] = [9, 17, 22, -1]
adj[17] = [-1, 18, -1, 16]
adj[18] = [10, -1, 23, 17]
adj[19] = [-1, -1, 25, -1]
adj[20] = [-1, -1, 26, -1]
adj[21] = [15, -1, 28, -1]
adj[22] = [16, -1, -1, -1]
adj[23] = [18, 24, 31, -1]
adj[24] = [-1, 25, -1, 23]
adj[25] = [19, -1, 32, 24]
adj[26] = [20, 27, 33, -1]
adj[27] = [-1, 28, -1, 26]
adj[28] = [21, 29, 34, 27]
adj[29] = [-1, 30, -1, 28]
adj[30] = [-1, 31, 35, 29]
```

```
adj[31] = [23, -1, 36, 30]
adj[32] = [25, -1, -1, -1]
adj[33] = [26, -1, -1, -1]
adj[34] = [28, -1, 41, -1]
adj[35] = [30, 36, -1, -1]
adj[36] = [31, 37, 43, 35]
adj[37] = [-1, 38, -1, 36]
adj[38] = [-1, -1, 44, 37]
adj[39] = [-1, 40, 45, -1]
adj[40] = [-1, 41, -1, 39]
adj[41] = [34, 42, 46, 40]
adj[42] = [-1, -1, 47, 41]
adj[43] = [36, -1, -1, -1]
adj[44] = [38, -1, 51, -1]
adj[45] = [39, -1, -1, -1]
adj[46] = [41, 47, -1, -1]
adj[47] = [42, 48, -1, 46]
adj[48] = [-1, 49, -1, 47]
adj[49] = [-1, 50, -1, 48]
adj[50] = [-1, 51, -1, 49]
adj[51] = [44, -1, -1, 50]

x = []
x[0] = 0
x[1] = 1
x[2] = 2
x[3] = 3
x[4] = 4
x[5] = 5
x[6] = 7
x[7] = 0
x[8] = 2
x[9] = 3
x[10] = 5
x[11] = 6
x[12] = 7
x[13] = 0
x[14] = 1
x[15] = 2
x[16] = 3
x[17] = 4
x[18] = 5
x[19] = 7
```

```
x[20] = 0
x[21] = 2
x[22] = 3
x[23] = 5
x[24] = 6
x[25] = 7
x[26] = 0
x[27] = 1
x[28] = 2
x[29] = 3
x[30] = 4
x[31] = 5
x[32] = 7
x[33] = 0
x[34] = 2
x[35] = 4
x[36] = 5
x[37] = 6
x[38] = 7
x[39] = 0
x[40] = 1
x[41] = 2
x[42] = 3
x[43] = 5
x[44] = 7
x[45] = 0
x[46] = 2
x[47] = 3
x[48] = 4
x[49] = 5
x[50] = 6
x[51] = 7

y = []
y[0] = 0
y[1] = 0
y[2] = 0
y[3] = 0
y[4] = 0
y[5] = 0
y[6] = 0
y[7] = 1
y[8] = 1
```

```
y[9] = 1
y[10] = 1
y[11] = 1
y[12] = 1
y[13] = 2
y[14] = 2
y[15] = 2
y[16] = 2
y[17] = 2
y[18] = 2
y[19] = 2
y[20] = 3
y[21] = 3
y[22] = 3
y[23] = 3
y[24] = 3
y[25] = 3
y[26] = 4
y[27] = 4
y[28] = 4
y[29] = 4
y[30] = 4
y[31] = 4
y[32] = 4
y[33] = 5
y[34] = 5
y[35] = 5
y[36] = 5
y[37] = 5
y[38] = 5
y[39] = 6
y[40] = 6
y[41] = 6
y[42] = 6
y[43] = 6
y[44] = 6
y[45] = 7
y[46] = 7
y[47] = 7
y[48] = 7
y[49] = 7
y[50] = 7
y[51] = 7
```

```
//dir[i][j] содержит направление, в котором должен находится
//робот, если он стартовал в комнате i с направлением j
dir = []
//room[i][j] содержит номер помещения, в котором должен
//находится робот, если он стартовал в комнате i с
//направлением j (-1 если такого нет)
room = []
locFlag = true

startX = 0
startY = 0
finishX = 0
finishY = 0

for (i = 0; i < 52; i++) {
  dir[i] = [0, 1, 2, 3]
  room[i] = [i, i, i, i]
}

// Параметры робота
d = 5.6 // Диаметр колеса
w = 19 // База (расстояние от колеса до колеса)
cpr = 374 // Показания энкодера за один оборот
robotRoom = 0 // Начальное положение робота
robotDir = 0 // Начальный азимут робота


// Моторы
mLeft = brick.motor(M4).setPower
mRight = brick.motor(M3).setPower

// Энкодеры
eLeft = function() {
  return -brick.encoder(E4).readRawData()
}
eRight = function() {
```

```
  return brick.encoder(E3).readRawData()
}

// Датчики освещения
cLeft = brick.sensor(A5)
cRight = brick.sensor(A6)

// Ультразвуковые датчики
uRight = brick.sensor(D1)
uLeft = brick.sensor(D2)

// Ифракрасный датчик
iForward = brick.sensor(A1)

// Остановка робота
function stop() {
  brick.motor(M3).forceBreak()
  brick.motor(M4).forceBreak()
}

// Сброс показаний энкодера
function resetEncoders() {
  brick.encoder(E3).reset()
  brick.encoder(E4).reset()
}

// Проехать вперед на один сегмент
function moveSegment() {
  resetEncoders()
  while (eLeft() + eRight() < 1030) {
    u = 0.05 * (eLeft() - eRight())
    v = 30
    mLeft(v - u)
    mRight(v + u)
    wait(10)
  }
  stop()
}

// Поворот налево
function turnLeft() {
  resetEncoders()
  xx = 200
```

```
  v = 30
  mLeft(-v)
  mRight(v)
  while (abs(eRight() - eLeft() - 2 * xx) > 10)
    wait(10)
  mLeft(0)
  mRight(0)
}

// Поворот направо
function turnRight() {
  resetEncoders()
  xx = 200
  v = 30
  mLeft(v)
  mRight(-v)
  while (abs(eLeft() - eRight() - 2 * xx) > 10)
    wait(10)
  mLeft(0)
  mRight(0)
}

function movehalfSegment() {
  resetEncoders()
  while (eLeft() + eRight() < 20 * 360 / pi / d) {
    u = 0.1 * (eLeft() - eRight())
    v = 20
    mLeft(v - u)
    mRight(v + u)
    wait(10)
  }
  stop()
}

function calibrate() {
  mLeft(-25)
  mRight(-25)
  wait(3000)
  movehalfSegment()
}

// Свободно ли справа
function isFreeRight() {
```

```
  return (uRight.read() > 25)
}

// Свободно ли спереди
function isFreeForward() {
  return (iForward.read() > 25)
}

// Обновление матриц состояний при
повороте налево
function turnLeftUpdate() {
  for (i = 0; i < 52; i++)
    for (j = 0; j < 4; j++)
      dir[i][j] = (dir[i][j] + 3) % 4
}

// Обновление матриц состояний при
повороте направо
function turnRightUpdate() {
  for (i = 0; i < 52; i++)
    for (j = 0; j < 4; j++)
      dir[i][j] = (dir[i][j] + 1) % 4
}

// Обновление матриц состояний при
прохождении одного сегмента
function moveSegmentUpdate() {
  for (i = 0; i < 52; i++)
    for (j = 0; j < 4; j++)
      if (room[i][j] != -1)
        room[i][j] = adj[room[i][j]][dir[i][j]]; // Переходим
в другую комнату
}

// Обновление матриц состояний при
получении показаний с датчиков
function valuesUpdate() {
  valueLeft = (uLeft.read() < 25 ? 1 : 0)
  valueRight = (uRight.read() < 25 ? 1 : 0)
  valueForward = (iForward.read() < 25 ? 1 : 0)
  //print("values:", valueLeft, valueForward, valueRight)
  cnt = 0
  for (i = 0; i < 52; i++)
```

```
    for (j = 0; j < 4; j++)
      if (room[i][j] != -1) {
        if (valueLeft != (adj[room[i][j]][(dir[i][j] + 3) % 4] == -1
?
            1 : 0) || valueForward != (adj[room[i][j]][dir[i][j]] ==
            -1 ? 1 : 0) || valueRight != (adj[room[i][j]][(dir[i][j]
+
            1) % 4] == -1 ? 1 : 0))
          room[i][j] = -1; // Если показания
датчиков не совпадают
        else {
          cnt++
          robotRoom = i
          robotDir = j
          //print(i, " ", j, " left")
        }
      }
  if (cnt == 1) locFlag = false;
}

// Движение по правилу правой руки
function RightHand() {
  while (true)
    if (isFreeRight()) { // Если справа свободно
      turnRight() // Поворот направо
      moveSegment() // Проезд на 1 сегмент вперед
    } else if (isFreeForward()) { // Если спереди
свободно
    moveSegment() // Проезд на 1 сегмент вперед
  } else { // Если тупик
    turnLeft() // Поворот налево
  }
}

// Локализация
function localization() {
  valuesUpdate()
  wait(1000)
  while (locFlag)
    if (isFreeRight()) { // Если справа свободно
      turnRight() // Поворот направо
      turnRightUpdate()
      wait(1000)
```

```
      valuesUpdate()
      wait(1000)
      moveSegment() // Проезд на 1 сегмент вперед
      moveSegmentUpdate()
      wait(1000)
    } else if (isFreeForward()) { // Если спереди
свободно
    moveSegment() // Проезд на 1 сегмент вперед
    moveSegmentUpdate()
    wait(1000)
  } else { // Если тупик
    turnLeft() // Поворот налево
    turnLeftUpdate()
    wait(1000)
  }
}

var main = function() {
  //print("START")
  localization()
  startX = x[robotRoom]
  startY = y[robotRoom]
  finishX = x[room[robotRoom][robotDir]]
  finishY = y[room[robotRoom][robotDir]]
  print(robotRoom)
  print(robotDir)
  print(startX)
  print(startY)
  print(finishX)
  print(finishY)
  print(s)
  brick.display().addLabel(s, 1, 20)
  brick.display().redraw()
  wait(30000)
  //print("FINISH")
}
```

Программа команды для решения задачи третьего этапа:

```
pi = 3.141592653589793
wait = script.wait
sign = function(n) {
```

```
  return n > 0 ? 1 : n = 0 ? 0 : -1
}
sqr = function(n) {
  return n * n
}
sqrt = Math.sqrt
min = function(a, b) {
  return a < b ? a : b
}
max = function(a, b) {
  return a > b ? a : b
}
abs = Math.abs
sin = Math.sin
cos = Math.cos
round = Math.round

isFoundLine = false
count = 0

// Параметры робота
d = 5.6 // Диаметр колеса
w = 20.6 // База (расстояние от колеса до колеса)
cpr = 240 // Показания энкодера за один оборот
x = 0 // Начальные координаты робота
y = 0
a = 0 // Начальный угол робота

// Моторы
mLeft = brick.motor(M4).setPower
mRight = brick.motor(M3).setPower

// Энкодеры
eLeft = function() {
  return -brick.encoder(E4).readRawData()
}
eRight = function() {
  return brick.encoder(E3).readRawData()
}

// Датчики освещения
```

```
cLeft = brick.sensor(A5)
cRight = brick.sensor(A6)

// Остановка робота
function stop() {
  brick.motor(M3).forceBreak()
  brick.motor(M4).forceBreak()
}

// Сброс показаний энкодера
function resetEncoders() {
  brick.encoder(E3).reset()
  brick.encoder(E4).reset()
}

pop = []
yop = []
var main = function() {
    resetEncoders()
    print('start')
    /*while(cLeft.read()<5){
      u = (eLeft()-eRight())*0.02
      mLeft(25-u)
      mRight(25+u)

    }*/
    while (eLeft() < 80) {
      u = (eLeft() - eRight()) * 0.02
      mLeft(30 - u)
      mRight(33 + u)
    }
    stop()
    script.wait(1000)

    for (i = 0; i < 4; i++) {
      resetEncoders()
      u = 0
      while (eLeft() + eRight() < 30) {
        u = (eLeft() - eRight()) * 0

        mLeft(30 - u)
        mRight(33 + u)
      }
```

```
      stop()
      script.wait(500)
      if (cLeft.read() > 50) {
        pop[i] = 1

      } else {
        pop[i] = 0

      }
      if (cRight.read() > 50) {
        yop[i] = 1

      } else {
        yop[i] = 0

      }
      script.wait(500)

      ter = pop[0] * 8 + pop[1] * 4 + pop[2] * 2 + pop[3] * 1
      print(ter)
      resetEncoders()
      while (eLeft() + eRight() < 350) {
        u = (eLeft() - eRight()) * 0

        mLeft(30 - u)
        mRight(32 + u)
      }
      stop()
      brick.display().addLabel(ter + "", 1, 20)
      brick.display().redraw()
      script.wait(30000)

      print("END")
      return;
    }
```

Программа команды для решения задачи “построение маршрута” четвертого этапа:

```
pi = 3.141592653589793
wait = script.wait
sign = function(n) {
```

```
  return n > 0 ? 1 : n = 0 ? 0 : -1
}
sqr = function(n) {
  return n * n
}
sqrt = Math.sqrt
min = function(a, b) {
  return a < b ? a : b
}
max = function(a, b) {
  return a > b ? a : b
}
abs = Math.abs
sin = Math.sin
cos = Math.cos
round = Math.round

adj = [] //adj[i][j] содержит номер помещения,
который находится по направлению 0 <= j <= 3
от i-ого помещения (0 если такого нет)
adj[0] = [-1, 1, 7, -1]
adj[1] = [-1, 2, -1, 0]
adj[2] = [-1, 3, 8, 1]
adj[3] = [-1, 4, -1, 2]
adj[4] = [-1, 5, -1, 3]
adj[5] = [-1, -1, 10, 4]
adj[6] = [-1, -1, 12, -1]
adj[7] = [0, -1, 13, -1]
adj[8] = [2, -1, 15, -1]
adj[9] = [-1, -1, 16, -1]
adj[10] = [5, 11, 18, -1]
adj[11] = [-1, 12, -1, 10]
adj[12] = [6, -1, -1, 11]
adj[13] = [7, 14, -1, -1]
adj[14] = [-1, 15, -1, 13]
adj[15] = [8, -1, 21, 14]
adj[16] = [9, 17, 22, -1]
adj[17] = [-1, 18, -1, 16]
adj[18] = [10, -1, 23, 17]
adj[19] = [-1, -1, 25, -1]
adj[20] = [-1, -1, 26, -1]
adj[21] = [15, -1, 28, -1]
adj[22] = [16, -1, -1, -1]
```

```
adj[23] = [18, 24, 31, -1]
adj[24] = [-1, 25, -1, 23]
adj[25] = [19, -1, 32, 24]
adj[26] = [20, 27, 33, -1]
adj[27] = [-1, 28, -1, 26]
adj[28] = [21, 29, 34, 27]
adj[29] = [-1, 30, -1, 28]
adj[30] = [-1, 31, 35, 29]
adj[31] = [23, -1, 36, 30]
adj[32] = [25, -1, -1, -1]
adj[33] = [26, -1, -1, -1]
adj[34] = [28, -1, 41, -1]
adj[35] = [30, 36, -1, -1]
adj[36] = [31, 37, 43, 35]
adj[37] = [-1, 38, -1, 36]
adj[38] = [-1, -1, 44, 37]
adj[39] = [-1, 40, 45, -1]
adj[40] = [-1, 41, -1, 39]
adj[41] = [34, 42, 46, 40]
adj[42] = [-1, -1, 47, 41]
adj[43] = [36, -1, -1, -1]
adj[44] = [38, -1, 51, -1]
adj[45] = [39, -1, -1, -1]
adj[46] = [41, 47, -1, -1]
adj[47] = [42, 48, -1, 46]
adj[48] = [-1, 49, -1, 47]
adj[49] = [-1, 50, -1, 48]
adj[50] = [-1, 51, -1, 49]
adj[51] = [44, -1, -1, 50]

x = []
x[0] = 0
x[1] = 1
x[2] = 2
x[3] = 3
x[4] = 4
x[5] = 5
x[6] = 7
x[7] = 0
x[8] = 2
x[9] = 3
x[10] = 5
x[11] = 6
```

```
x[12] = 7
x[13] = 0
x[14] = 1
x[15] = 2
x[16] = 3
x[17] = 4
x[18] = 5
x[19] = 7
x[20] = 0
x[21] = 2
x[22] = 3
x[23] = 5
x[24] = 6
x[25] = 7
x[26] = 0
x[27] = 1
x[28] = 2
x[29] = 3
x[30] = 4
x[31] = 5
x[32] = 7
x[33] = 0
x[34] = 2
x[35] = 4
x[36] = 5
x[37] = 6
x[38] = 7
x[39] = 0
x[40] = 1
x[41] = 2
x[42] = 3
x[43] = 5
x[44] = 7
x[45] = 0
x[46] = 2
x[47] = 3
x[48] = 4
x[49] = 5
x[50] = 6
x[51] = 7

y = []
y[0] = 0
```

```
y[1] = 0
y[2] = 0
y[3] = 0
y[4] = 0
y[5] = 0
y[6] = 0
y[7] = 1
y[8] = 1
y[9] = 1
y[10] = 1
y[11] = 1
y[12] = 1
y[13] = 2
y[14] = 2
y[15] = 2
y[16] = 2
y[17] = 2
y[18] = 2
y[19] = 2
y[20] = 3
y[21] = 3
y[22] = 3
y[23] = 3
y[24] = 3
y[25] = 3
y[26] = 4
y[27] = 4
y[28] = 4
y[29] = 4
y[30] = 4
y[31] = 4
y[32] = 4
y[33] = 5
y[34] = 5
y[35] = 5
y[36] = 5
y[37] = 5
y[38] = 5
y[39] = 6
y[40] = 6
y[41] = 6
y[42] = 6
y[43] = 6
```

```
y[44] = 6
y[45] = 7
y[46] = 7
y[47] = 7
y[48] = 7
y[49] = 7
y[50] = 7
y[51] = 7

dir = [] // dir[i][j] содержит направление, в котором должен находится робот, если он стартовал в комнате i с направлением j
room = [] // room[i][j] содержит номер помещения, в котором должен находится робот, если он стартовал в комнате i с направлением j (-1 если такого нет)
locFlag = true // false когда робот локализовался
from = [] // from[i] содержит направление из которого пришли в i-ую комнату ячейку (0 - сверху, 1 - справа, 2 - снизу, 3 - слева)
used = [] // used[i] определяет, посещена ли i-ая комната обходом в ширину
codeRoom = 3 // где находится штрихкод

startX = 0
startY = 0
finishX = 0
finishY = 0

for (i = 0; i < 52; i++) {
  dir[i] = [0, 1, 2, 3]
  room[i] = [i, i, i, i]
  from[i] = -1
  used[i] = false
}

// Параметры робота
d = 5.6 // Диаметр колеса
w = 19 // База (расстояние от колеса до колеса)
cpr = 374 // Показания энкодера за один оборот
robotRoom = 0 // Начальное положение робота
```

```
robotDir = 0 // Начальный азимут робота
finishRoom = 0 // Целевое положение робота

// Моторы
mLeft = brick.motor(M4).setPower
mRight = brick.motor(M3).setPower

// Энкодеры
eLeft = function() {
  return -brick.encoder(E4).readRawData()
}
eRight = function() {
  return brick.encoder(E3).readRawData()
}

// Датчики освещения
cLeft = brick.sensor(A5)
cRight = brick.sensor(A6)

// Ультразвуковые датчики
uRight = brick.sensor(D1)
uLeft = brick.sensor(D2)

// Ифракрасный датчик
iForward = brick.sensor(A1)



// Остановка робота
function stop() {
  brick.motor(M3).forceBreak()
  brick.motor(M4).forceBreak()
}

// Сброс показаний энкодера
function resetEncoders() {
  brick.encoder(E3).reset()
  brick.encoder(E4).reset()
}

// Проехать вперед на один сегмент
function moveSegment() {
  resetEncoders()
```

```
  while (eLeft() + eRight() < 1030) {
    u = 0.05 * (eLeft() - eRight())
    v = 30
    mLeft(v - u)
    mRight(v + u)
    wait(10)
  }
  stop()
}

// Поворот налево
function turnLeft() {
  resetEncoders()
  xx = 200
  v = 30
  mLeft(-v)
  mRight(v)
  while (abs(eRight() - eLeft() - 2 * xx) > 10)
    wait(10)
  mLeft(0)
  mRight(0)
}

// Поворот направо
function turnRight() {
  resetEncoders()
  xx = 200
  v = 30
  mLeft(v)
  mRight(-v)
  while (abs(eLeft() - eRight() - 2 * xx) > 10)
    wait(10)
  mLeft(0)
  mRight(0)
}

// Свободно ли справа
function isFreeRight() {
  return (uRight.read() > 25)
}

// Свободно ли спереди
function isFreeForward() {
```

```
  return (iForward.read() > 25)
}

// Обновление матриц состояний при
повороте налево
function turnLeftUpdate() {
  for (i = 0; i < 52; i++)
    for (j = 0; j < 4; j++)
      dir[i][j] = (dir[i][j] + 3) % 4
}

// Обновление матриц состояний при
повороте направо
function turnRightUpdate() {
  for (i = 0; i < 52; i++)
    for (j = 0; j < 4; j++)
      dir[i][j] = (dir[i][j] + 1) % 4
}

// Обновление матриц состояний при
прохождении одного сегмента
function moveSegmentUpdate() {
  for (i = 0; i < 52; i++)
    for (j = 0; j < 4; j++)
      if (room[i][j] != -1)
        room[i][j] = adj[room[i][j]][dir[i][j]]; // Переходим
в другую комнату
}

// Обновление матриц состояний при
получении показаний с датчиков
function valuesUpdate() {
  valueLeft = (uLeft.read() < 25 ? 1 : 0)
  valueRight = (uRight.read() < 25 ? 1 : 0)
  valueForward = (iForward.read() < 25 ? 1 : 0)
  //print("values:", valueLeft, valueForward, valueRight)
  cnt = 0
  for (i = 0; i < 52; i++)
    for (j = 0; j < 4; j++)
      if (room[i][j] != -1) {
        if (valueLeft != (adj[room[i][j]][(dir[i][j] + 3) % 4] == -1
?
            1 : 0) || valueForward != (adj[room[i][j]][dir[i][j]] ==
```

```
            -1 ? 1 : 0) || valueRight != (adj[room[i][j]][(dir[i][j]
+
            1) % 4] == -1 ? 1 : 0))
          room[i][j] = -1; // Если показания
датчиков не совпадают
        else {
          cnt++
          robotRoom = i
          robotDir = j
          //print(i, " ", j, " left")
        }
      }
  if (cnt == 1) locFlag = false;
}

// Движение по правилу правой руки
function RightHand() {
  while (true)
    if (isFreeRight()) { // Если справа свободно
      turnRight() // Поворот направо
      moveSegment() // Проезд на 1 сегмент вперед
    } else if (isFreeForward()) { // Если спереди
свободно
    moveSegment() // Проезд на 1 сегмент вперед
  } else { // Если тупик
    turnLeft() // Поворот налево
  }
}

// Локализация
function localization() {
  valuesUpdate()
  delay = 1000
  wait(delay)
  while (locFlag)
    if (isFreeRight()) { // Если справа свободно
      turnRight() // Поворот направо
      turnRightUpdate()
      wait(delay)
      valuesUpdate()
      wait(delay)
      moveSegment() // Проезд на 1 сегмент вперед
      moveSegmentUpdate()
```

```
      wait(delay)
    } else if (isFreeForward()) { // Если спереди свободно
      moveSegment() // Проезд на 1 сегмент вперед
      moveSegmentUpdate()
      wait(delay)
    } else { // Если тупик
      turnLeft() // Поворот налево
      turnLeftUpdate()
      wait(delay)
    }
}

// Поиск в ширину для построения маршрута
function bfs(finishRoom) {
  queue = []
  queue.push(robotRoom)
  used[robotRoom] = true
  while (queue.length > 0) { // Пока не перебрали все комнаты
    curRoom = queue.shift()
    if (curRoom == finishRoom) break // Если нашли конечную комнату
    for (i = 0; i < 4; i++) { // Перебираем все соседей
      adjRoom = adj[curRoom][i]
      if (adjRoom == -1) continue // Если стена, то выходим
      if (!used[adjRoom]) { // Если еще не посещали комнату
        queue.push(adjRoom)
        from[adjRoom] = i
        used[adjRoom] = true
      }
    }
  }
}

// Построение маршрута
function buildPath(finishRoom) {
  bfs(finishRoom)
  path = []
  curRoom = finishRoom
```

```
  while (from[curRoom] != -1) {
    path.unshift(from[curRoom])
    curRoom = adj[curRoom][(from[curRoom] + 2) % 4]
  }
  return path
}

// Получить маршрут в виде F/L/R
function getPath(finishRoom) {
  path = buildPath(finishRoom)
  strPath = ""
  curDir = robotDir
  for (i = 0; i < path.length; i++) {
    relDir = (path[i] - curDir + 4) % 4
    if (relDir == 1) strPath += "R"
    else if (relDir == 2) strPath += "RR"
    else if (relDir == 3) strPath += "L"
    curDir = path[i]
    strPath += "F"
  }
  return strPath
}

// Следование робота по заданному
маршруту
function followPath(path) {
  for (i = 0; i < path.length; i++) {
    if (path[i] == "L")
      turnLeft()
    else if (path[i] == "R")
      turnRight()
    else
      moveSegment()
    wait(1000)
  }
}

// Получить начальные координаты робота
function getStartFinish() {
  robotDir = startDir
  for (i = 0; i < 52; i++) {
    if (x[i] == startX && y[i] == startY)
      robotRoom = i
```

```
    if (x[i] == finishX && y[i] == finishY)
      finishRoom = i
  }
}

function beep() {
  brick.playSound("media/beep.wav")
}

var main = function() {
  print("START")

  startX = 6
  startY = 7
  startDir = 1
  finishX = 2
  finishY = 3
  getStartFinish()

  print("robotDir: ", robotDir)
  print("robotRoom: ", robotRoom)
  print("finishRoom: ", finishRoom)

  path = getPath(finishRoom)
  print("path: ", path)

  brick.display().addLabel(path, 1, 20)
  brick.display().redraw()
  wait(10000)

  followPath(path)
  beep()

  print("FINISH")
}
```

Программы для решения полной финальной задачи нет.