```
        else
              r = m;
      }
      cout << fixed << setprecision(15) << sqrt((double)r) / 2;
        return 0;
}
```

3.2. Задача командного тура

В командной части заключительного этапа участники должны спроектировать автономную систему управления роботом-погрузчиком для решения задач навигации и локализации на модели логистического центра.

Командная часть заключительного этапа имеет продолжительность 3,5 дня (всего 24 астрономических часа), которые включают работу по оснащению роботов необходимыми датчиками, программированию, пробные заезды на макете логистического центра, зачетные попытки.

3.2.1. Легенда

Работники логистического центра с предыдущей смены оставили робот-погрузчик где-то внутри логистического центра. Система управления робота имеет информацию о структуре центра, но у нее нет информации о текущем расположении робота на его территории.

По внутренней системе распределения задач, погрузчик получает задачу самостоятельно прибыть в определенные сектора логистического центра, где он пройдет сервисное обслуживание, после чего он должен самостоятельно проследовать в сектор приписки данного робота.

3.2.2. Набор заданий

Решение командной задачи было разбито на 3 этапа. Первые три этапа итеративно подводили участников к решению полной финальной задачи, осуществляемому во время последнего четвертого этапа. На каждом этапе помимо полноты решения заданий данного этапа проверялась воспроизводимость результатов - робототехническое устройство участников должно было неоднократно выполнить требуемые действия.

Первый этап

Робототехническое устройство должно проехать по модели логистического центра по правилу правой или левой руки из сектора старта, пока не увидит линию черную линию, нанесенную на пол модели. Линия обозначает сектор финиша. Путь перемещения робота будет заранее выбран так, что сектор финиша достижим из сектора старта - в нем не будет циклических структур. Устройство должно заехать в сектор финиша, остановиться и вывести на экран количество, пройденных сегментов.

Включая содержательные задачи:

- Определение оптимального расположения датчиков сенсорной системы, расчет интенсивности принимаемого сенсорами сигнала, определение пороговых значений;
- Реализация навигации
    - Перемещение вдоль стены с использованием показаний датчиков расстояния или освещенности;
    - Выравнивание;
    - Перемещение по азимуту с использованием показаний датчиков гироскопа или акселерометра;
    - Поворот на заданный угол с использованием показаний датчиков гироскопа или акселерометра;
- Реализация счисления пути.

Второй этап

Робототехническое устройство после старта из заранее неизвестного расположения определяет свою текущую позицию после некоторого времени перемещения по лабиринту. При этом проверяется два возможных режима работы:

1. Определение циклических структур полигона: устройство должно двигаться вдоль циклической структуры до тех пор, пока не определит, что двигается по той же траектории. При этом должен быть пройден, как минимум один полный обход вдоль циклической структуры. После определения циклической структуры, устройство должно начать двигаться по секциям, лежащим вне начальной траектории.
2. Локализация: устройство должно двигаться до тех пор, пока не определит свою текущую позицию в лабиринте. После остановки, на экране должны отобразиться текущие координаты устройства.

Включая содержательные задачи:

- Реализация построения карты по данным одометрии;
- Реализация локализации.

Третий этап

Робототехническое устройство после старта из заранее неизвестного расположения (сектор старта находится не далее чем в пяти секторах до сектора сервисного обслуживания) доезжает до штрих-кода, останавливается и выводит на экран десятичное значение закодированного числа.

Включая содержательные задачи:

- Реализация алгоритмов распознавания штрих-кода.

Четвертый этап

Робототехническое устройство после старта из заранее неизвестного расположения проезжает через два сектора сервисного обслуживания и завершает свое перемещение в секторе приписки робота.

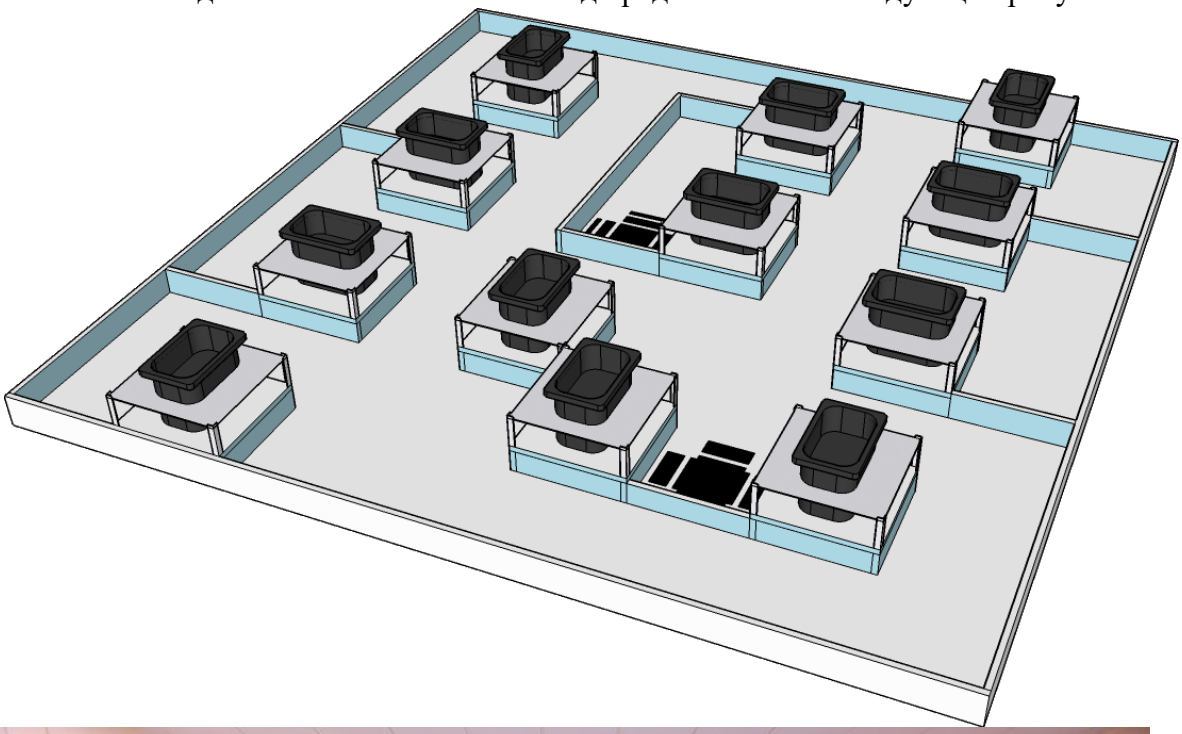
Включая содержательные задачи:

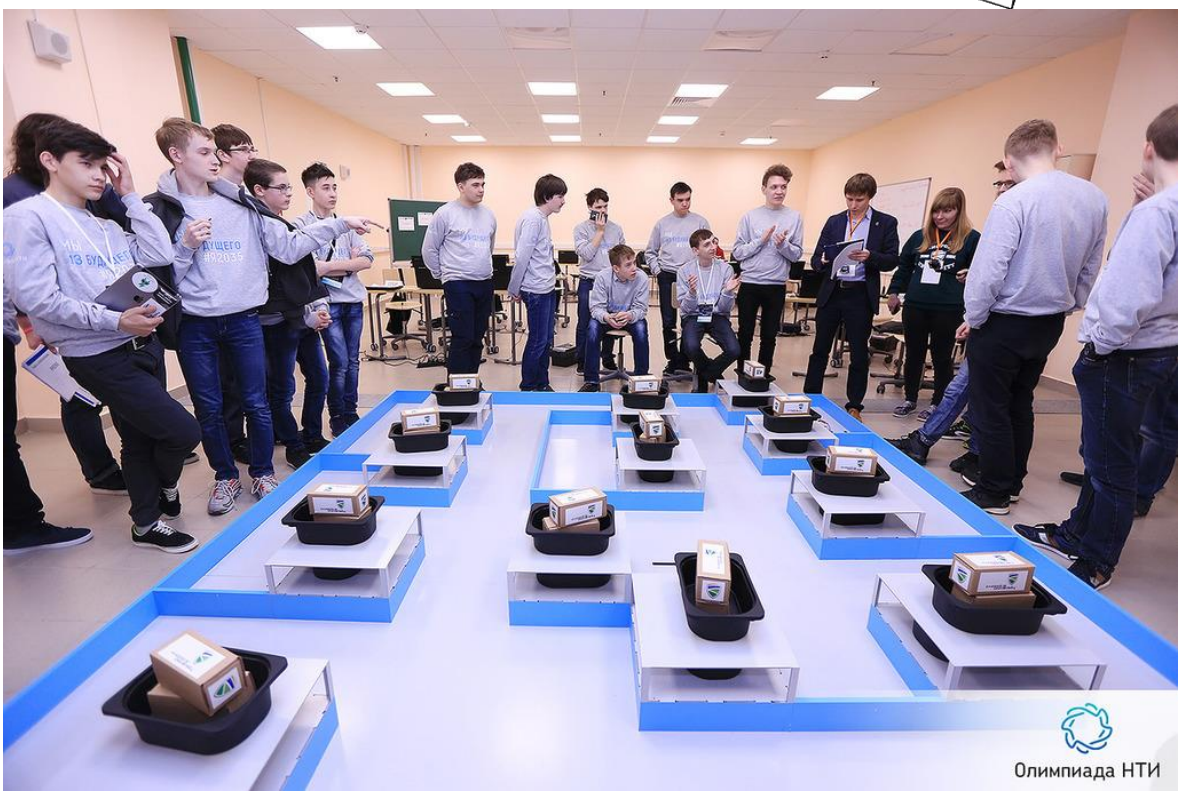
- Реализация алгоритмов построения оптимальных маршрутов до секторов сервисного обслуживания и до финального сектора;
- Реализация алгоритмов автоматического планирования перемещения.

3.2.3. Описание модели логистического центра

Полигон - квадратное поле 3200х3200 мм., разделенное на квадратные сектора 400х400 мм. Некоторые сектора отделены друг от друга перегородкой высотой 100 мм. Некоторые сектора недоступны для посещения робототехническим устройством и представляют из себя модель стеллажа высотой 210 мм., на каждой полке стеллажа располагается черный контейнер размером 200х300х100 мм.

Схема модели и ее окончательный вид представлены на следующих рисунках:





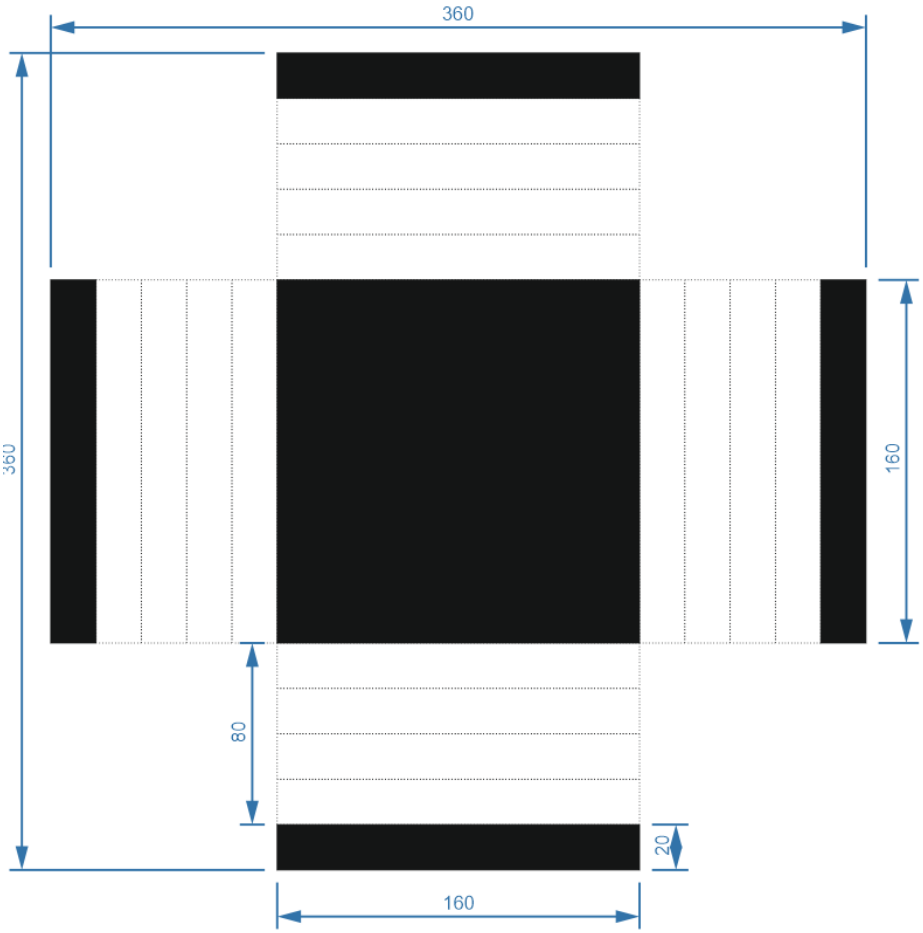
Полигон окружен бортом высотой 100 мм.

Конфигурация полигона определяется в первый день финального этапа и объявляется участникам. Данная конфигурация будет использоваться все дни финального тура.

В местах расположения секторов сервисного обслуживания расположен штрих-код, кодирующий сектор приписки робота-погрузчика - финальный сектор, куда должен прибыть робот в конце выполнения задачи.

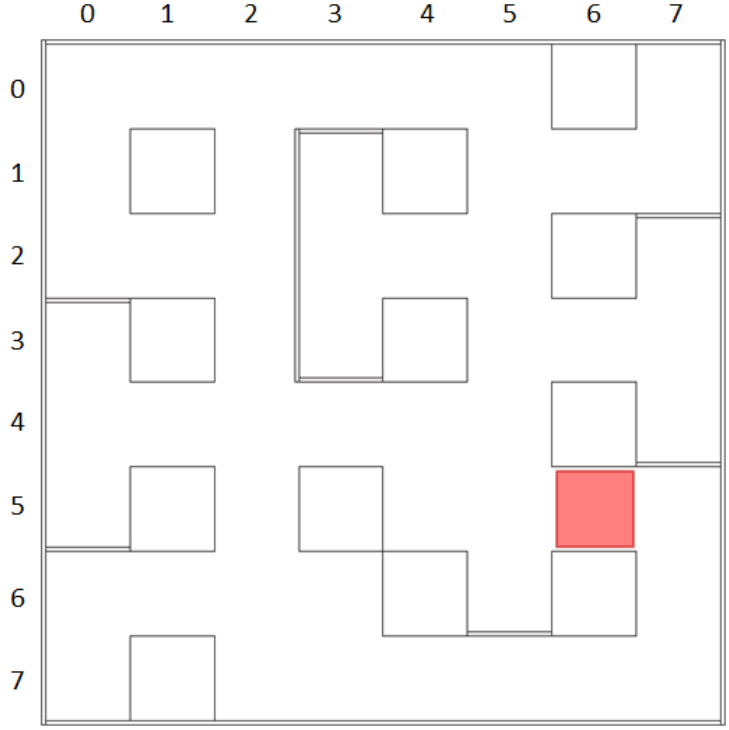
Штрих-код - симметричный, где черная линия обозначает единицу, отсутствие черной линии - 0. Старший бит, закодированного двоичного числа, располагается ближе к краю сектора. Младший бит - ближе к середине. Если закодированное двоичное число находится в диапазоне от 0_{10} до 7_{10}, то оно кодирует координату X финального сектора. Если двоичное число - в диапазоне от 8_{10} до 15_{10}, то оно кодирует Y координату финального сектора, при этом координата определяется вычитанием 8 из закодированного числа. Начало отсчета координат находится в левом верхнем углу поля.

Размеры штрих-кода:



Примеры штрих-кода изображены на рисунке:

В данном примере левый штрих-код задает число 110_2 - 6_{10}. Правый штрих-код задает число 1101_2 - 13_{10}. Финальный сектор находится в позиции с координатами (6, 5):



Финальный сектор никак не обозначается на поле.

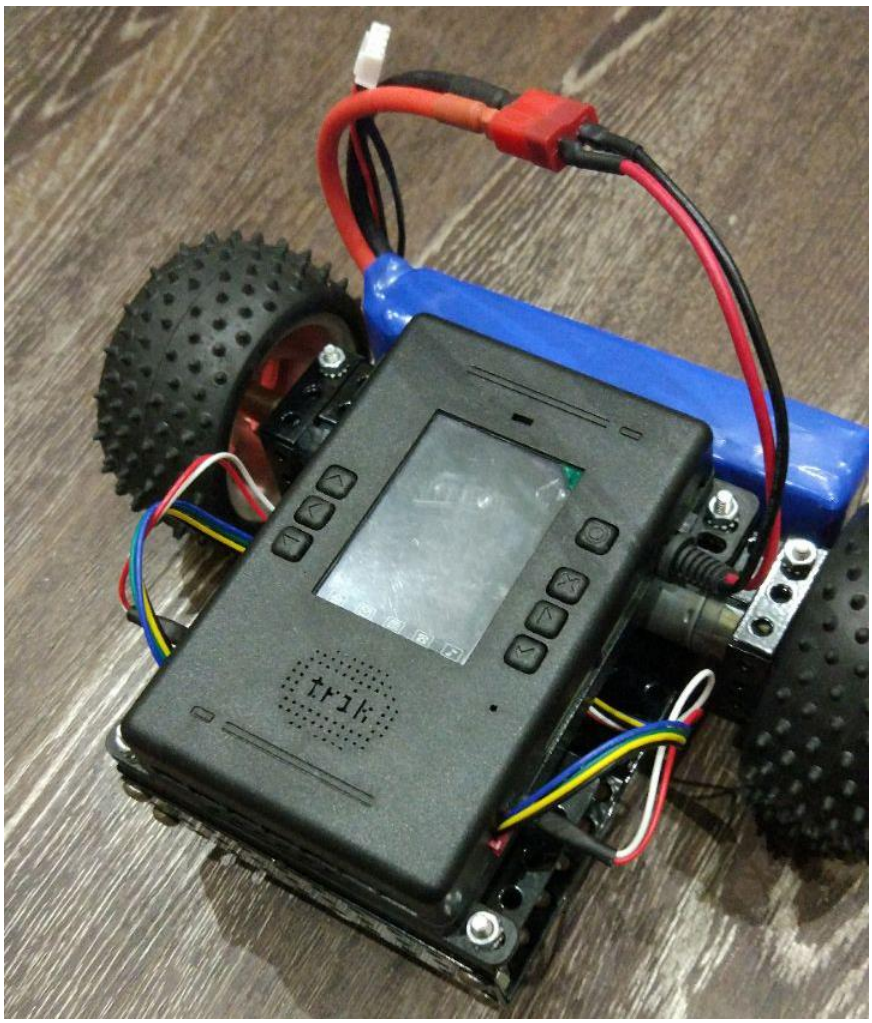
Местоположение секторов сервисного обслуживания определяется в первый день финального этапа и объявляется участникам. Оно остается постоянным все дни финального тура.

Финальный сектор может изменяться перед каждым заездом робота.

3.2.4. Описание конструктора

В первый день финального тура каждой команде выдается мобильная наземная платформа на базе конструктора ТРИК в сборе (блок управления TRIK, аккумулятор, два мотора с энкодерами на датчиках Холла, колеса), но без установленных датчиков. Мобильная платформа построена по принципу дифференциального управления. Физические размеры платформы позволяют совершать все маневры внутри одного сектора модели логистического центра без касания со стенками стеллажей или бортов.

Фотография собранной мобильной платформы:



В первый день финального тура каждой команде выдается набор датчиков:

- 1 датчик касания;
- 2 инфракрасных датчика дальности;
- 2 ультразвуковых датчика расстояния;
- 2 датчика освещенности;

Также команде выдается

- комплект дополнительных деталей из конструктора ТРИК;
- ноутбуки с установленной TRIK Studio (версия с поддержкой интерпретации QtScript в режиме отладки), на каждого члена команды по одному ноутбуку.

3.2.5. Условия проведения

1. Из полученного набора датчиков команды могут выбирать те, с помощью которых, на их взгляд, можно решить задачу наиболее эффективным способом.
2. Команды могут вносить любые изменения в мобильную наземную платформу.
3. Участники во время командного этапа финального тура могут использовать интернет и заранее подготовленные библиотеки для решения задачи.
4. Участники не могут использовать помощь тренера, сопровождающего лица или привлекать третьих лиц для решения задачи.
5. Финальная задача формулируется участникам в первый день финального тура, но участники должны выполнять решение задачи поэтапно. Критерии прохождения каждого этапа формулируются для каждого дня финального тура. За подзадачи,

решенные в конкретном этапе начисляются, баллы. Баллы за подзадачи можно получить только день, закрепленный за конкретным этапом.

6. Во время рабочего времени команды могут совершать неограниченное количество подходов к полигону для проведения испытаний.
7. Испытания на полигоне должны осуществляться так, чтобы не мешать другим командам, проводящим в это время испытания на полигоне.
8. Каждый день финального тура за 1,5 часа до конца выделенного рабочего времени команды должны сдать роботов в зону карантина. Время сдачи роботов в карантин может изменяться и зависит от количества команд и сложности подзадач, принимаемых в конкретный этап.
9. После момента, когда все роботы сданы в карантин, судьи вызывают команды по одной для определения решения подзадач, закрепленных за этапом конкретного дня финального тура. После прохождения приемочных запусков, баллы набранные командой заносятся судьями в протокол. Один из участников команды расписывается за набранный результат, подтверждая согласие команды с оценкой проведенных запусков.
10. Робот должен выполнять задание полностью автономно. Удаленное управление не допускается. Касание робота участником команды после его старта во время приемочных запусков не допускается. Алгоритм, реализующий систему управления робота должен планировать свое выполнение, полагаясь только на информацию с датчиков. Если какая-то подзадача подразумевает считывание информации с элементов, расположенных на полигоне, запрещается при запуске робота вводить информацию о положении этих элементов или значениях, которые данные элементы определяют.
11. Если во время приемочных запусков у судьи возникли сомнения о том, что задачи подэтапа решены корректно (робот не выполняет задачу полностью автономно, участник вводит значения в робота перед запуском), то он вправе провести инспекцию кода. По результатам инспекции, судья вправе снять с команды баллы, набранные за данный этап.
12. Если во время приемочных запусков у судьи возникает ситуация, когда он не может однозначно решить выполняются ли критерии решения подзадачи, он вправе принять решение не в пользу команды.
13. Команда вправе обсуждать с судьей результаты приемочных запусков до вызова следующей команды, но финальное решение остается о начислении баллов остается за судьей.

3.2.6. Процедура проведения приемочных запусков и критерии оценки

Первый этап

1. Судья спрашивает команду об алгоритме обхода секторов макета логистического центра (по правилу правой руки или по правилу левой руки) и определяет 2 сектора запуска робота. Для всех команд, с одинаковым алгоритмом обхода, сектора запуска робота будут одинаковым. Место запуска для робота неизвестно до начала приемочных запусков. Первый сектор запуска используется для первой попытки. Второй сектор запуска используется для второй попытки.
2. Максимальное время выполнения одной попытки - 2 минуты. Время на доработку решения между попытками не выдается.
3. Критерии оценки:

№ п.п.	Критерий	Баллы
1	Робот покинул сектор старта	6
2	Робот проехал половину от всего количества секторов от сектора старта до сектора финиша	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)	6
4	Робот проехал от сектора старта, заехал в сектор финиша (проекция робота на поле внутри сектора) и остановился	6
5	Робот проехал от сектора старта, остановился в секции финиша (проекция робота внутри сектора) и вывел на экран верное количество пройденных секций	12

4. Баллы за две попытки суммируются.
5. Выполнение всех критериев в каждой из двух попыток дает дополнительные 19 баллов.
6. Максимальное количество баллов за этап - 105 баллов.

Второй этап

1. Судья спрашивает команду об алгоритме обхода секторов макета логистического центра (по правилу правой руки или по правилу левой руки) и определяет 4 сектора запуска робота. Для всех команд, с одинаковым алгоритмом обхода, сектора запуска робота будут одинаковым. Место запуска для робота неизвестно до начала приемочных запусков. Первый, второй и четвертый сектора для запуска выбираются так, чтобы робот точно двигался вдоль циклической структуры на макете. Третий сектор выбирается так, чтобы на пути робота не было циклических структур (не считая цикла обхода роботом макета по периметру). Первый сектор запуска используется для первой попытки, второй сектор запуска используется для второй попытки и т.д. Первые две попытки для проверки решения задачи на определение циклических структур макета. Оставшиеся попытки для проверки решения задачи локализации.
2. Максимальное время выполнения первой или второй попытки - 2 минуты. Время на доработку решения между попытками не выдается.
3. Максимальное время выполнения третьей или четвертой попытки - 3 минуты. Время на доработку решения между попытками не выдается.
4. Критерии оценки:
   Определение циклических структур макета логистического центра:

№ п.п.	Критерий	Баллы
1	Робот проехал по периметру вокруг циклической структуры и остановился после определения цикла (допускается проезд по второму кругу до 3х секторов)	7
2	Робот после верного определения циклической структуры, остановился на 3 секунды, и продолжил движение по другим секторам макета, не принадлежащим циклической структуре	12

	(движение не менее, чем по 5 не принадлежащим циклической структуре)	

Локализация:

№ п.п.	Критерий	Баллы
1	Робот проехал не меньше 7 секторов по макету логистического центра 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.	6
2	После остановки отображается 2 числа, определяющее позицию, где произошла остановка: первое число – сектор по оси X, второе число – сектор по оси Y	31

5. Баллы за четыре попытки суммируются.
6. Выполнение всех критериев в каждой из попыток для проверки задачи локализации дает дополнительные 19 баллов.
7. Максимальное количество баллов за этап - 119 баллов.

Третий этап

1. Судья спрашивает команду об алгоритме обхода секторов макета логистического центра (по правилу правой руки или по правилу левой руки) и определяет 2 сектора запуска робота. Для всех команд, с одинаковым алгоритмом обхода, сектора запуска робота будут одинаковым. Место запуска для робота неизвестно до начала приемочных запусков. Первый сектор запуска используется для первой попытки, второй сектор запуска используется для второй попытки.
2. Перед каждой попыткой определяется вид штрих-кода. Для всех команд в одной и той же попытки штрих-код будет одинаковым.
3. Максимальное время выполнения первой или второй попытки - 2 минуты. Время на доработку решения между попытками не выдается.
4. Критерии оценки:

№ п.п.	Критерий	Баллы
1	Робот доехал до сектора сервисного обслуживания (проекция робота внутри сектора) и остановился	6
2	После остановки робота на экран выведено закодированное значение.	19

5. Баллы за две попытки суммируются.
6. Выполнение всех критериев в каждой из попыток дает дополнительные 12 баллов.
7. Максимальное количество баллов за этап - 62 балла.

Четвертый этап

1. Перед первой и второй попыткой судья определяет сектора старта и направление робота в секторе страта, также определяется сектор финиша. Команда должна в

присутствии судьи изменить программу, внеся данную информацию в программу. Допускается изменить только 5 переменных: 2 переменных (тип - целое), отвечающих за кодирование сектора старта, 1 переменная (одно целое число, либо один символ), отвечающая за кодирование направление в секторе старта, и две переменных (тип - целое), отвечающих за кодирование сектора финиша. После изменения программы, робот устанавливается в сектор старта, программа загружается на робота и должна запуститься. Никаких других изменений в программе не допускается.
Перед третьей попыткой судья определяет сектор старта робота, его направление в секторе старта и финальный сектор. Для всех команд эти условия будут одинаковыми. Значения штрих-кодов в секторах сервисного обслуживания выставляются в соответствии с позицией финального сектора.

2. Максимальное время выполнения первой или второй попытки - 2 минуты. Время на доработку решения между попытками не выдается.
3. Максимальное время выполнения третьей попытки - 5 минут.
4. Критерии оценки:

Определение пути перемещения:

№ п.п.	Критерий	Баллы
1	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.	6
2	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.	9
3	Путь, выведенный после запуска робота, является оптимальным по количеству секторов, необходимых для посещения.	3
4	Путь перемещения робота из сектора старта до сектора финиша является оптимальным по количеству секторов.	3
5	Робот доехал до финального сектора и остановился. И издал звуковой сигнал.	6

Полная задача:

№ п.п.	Критерий	Баллы
1	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
2	После остановки робота в секторе сервисного обслуживания	5

	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.	
3	Робот продолжил движение и остановился на 10 секунд во втором секторе сервисного обслуживания. Перед началом отсчета таймера робот подает звуковой сигнал.	11
4	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
5	Путь перемещения робота из сектора, где робот локализовался до очередного сектора сервисного обслуживания или финального сектора является оптимальным по количеству секторов.	5
6	Робот доехал до финального сектора и остановился. И издал звуковой сигнал.	28

5. Баллы за три попытки суммируются.
6. Выполнение всех критериев в каждой из попыток для проверки задачи определение пути перемещения дает дополнительные 19 баллов.
7. Максимальное количество баллов за этап - 114 баллов.
8.

3.2.7. Решение

Пример программы на языке QtScript, обеспечивающий решение задание задачи последнего этапа (без части считывания штрих-кода - см. следующий пример программы):

```
var pi = 3.1415926535897931;
var d = 5.6;//diameter of wheel, cm
var l = 17.5; // base of robot
var degInRad = 180 / pi;
var alpha = 0;

var readGyro = brick.gyroscope().read
function readYaw() { return  -readGyro()[6]; }

//кнопки
var stopKey = KeysEnum.Esc;
var startKey = KeysEnum.Left;

//диод
var led = brick.led();

//моторы
var mLeft = brick.motor(M3).setPower;
var mRight = brick.motor(M4).setPower;
```

```
//сенсоры ИК
sA1 = brick.sensor(A1).readRawData;
sA2 = brick.sensor(A2).readRawData;
sA3 = brick.sensor(A3).readRawData;

//энкодеры
var eLeft = brick.encoder(E3);
var eRight = brick.encoder(E4);

// количество сигналов на оборот
// Подбирается для каждого вида энкодеров
var cpr = 628;

//длина клетки
var cellLength = 40 * cpr/(pi*d);

var direction = 0; // absolute angle of direction movement
var directionOld = 0;
var azimut = 0; // we should go on azimut or turn to it

led.red();
eLeft.reset();
eRight.reset();

var el = eLeft.readRawData();
var er = eRight.readRawData();
mLeft(0);
mRight(0);
print("Start");

//инициализация и калибровка гироскопа
print("gyro initialaize...");
brick.gyroscope().calibrate(12000);
script.wait(13000);
print("gyro inited");
brick.display().addLabel(30, 10, "Start!");
brick.display().redraw();
script.wait(1000);

var v = 50; // velocity

var ex = 0;
var ey = 0;
var encLeftOld = 0;
var encRightOld = 0;
var encLeft = 0;
var encRight = 0;
var n = 0;

// вычисление абсолютного угла относительно
// начального положения
function angle()
{
  var sgn = 0;
  var _direction = readYaw(); // mgrad
  var dtDirection= _direction - directionOld;
```

```
  sgn = directionOld == 0 ? 0 :
directionOld/Math.abs(directionOld);
  n += sgn*Math.floor(Math.abs(dtDirection/320000));
  direction = _direction + n * 360000;
  directionOld = _direction;
}

var t = 0;

print("Run, robot, run!");


// делаем препывание основной программы с частотой 200Гц,
// чтобы посчитать абсолютный угол с гироскопа
var mtimer = script.timer(50);
mtimer.timeout.connect(angle);

// вывод в консоль показаний гироскопа
function printGyro()
{
  print("direction=" + direction + " sA1 " + sA1() +
        " sA2 " + sA2() + " sA3 " + sA3());
}

var ptimer = script.timer(300);
ptimer.timeout.connect(printGyro);

//поворот на угол по гироскопу _angle - относительный
// угол на который необходимо повернуться
function turnDirection(_angle, _v){
  _angle = azimut + _angle;
  azimut = _angle;
  _angle = _angle * 1000; //toMGrad
  var _vel = _v == undefined ? 10 : _v;
  var angleOfRotate = _angle - direction;
  var sgn =  angleOfRotate == 0 ? 0 :
angleOfRotate/Math.abs(angleOfRotate);
  mLeft(_vel * sgn);
  mRight(-_vel * sgn);
  while (Math.abs(angleOfRotate = _angle - direction) > 8000){
    script.wait(50);
  }
  brick.motor(M3).forceBreak(500);
  brick.motor(M4).forceBreak(500);
  script.wait(500);
}

// проезд в перед на количество ячеек _kceel со
// скоростью _v выравниваясь по гироскопу на угол azimut
function forward1( _v, _kcell)
{
  _alpha = azimut;
  var _vel = _v == undefined ? 10 : _v;
  var u = 0;
  var el = Math.abs(eLeft.readRawData());
  while(Math.abs(eLeft.readRawData()) < el + (_kcell *
```

```
cellLength)){
    u = -1.5*(_alpha-direction/1000);
    mLeft(_vel - u);
    mRight(_vel + u);
    script.wait(5);
  }
  brick.motor(M3).forceBreak();
  brick.motor(M4).forceBreak();
  script.wait(300);
}


// функция считывания сенсора в массив, при этом в массиве
// {левый датчик, передний датчик,правый датчик, задний датчик}
function readSensors(b){
  var sensorRead = [0,0,0,b];
  left = sA3();
  front = sA1();
  right = sA2();
  if(left>70)    sensorRead[0]  = 1;
  else       sensorRead[0]  = 0;
  if(front>70)  sensorRead[1]  = 1;
  else      sensorRead[1]  = 0;
  if(right>70)  sensorRead[2]  = 1;
  else      sensorRead[2]  = 0;
  return sensorRead;
}

// первая проверка гипотез
function firstCheck(sensor, cell)
{
  var s = 0;
for ( i = 0; i < 4; i++)
  s += sensor[i];
for ( i = 0; i < 4; i++)
  s -= cell[2+i];
return Math.abs(s) < 2;
}

// движение вперёд с учётом переноса гипотез
function forward(preHypotheses)
{
  forward1(v,1);
  var newHypotheses = go(preHypotheses);
  sensor[3]=1;
  return newHypotheses;
}

//поворот влево с учётом переноса гипотез
function turnLeft(preHypotheses)
{
  turnDirection(90,v);
  var newHypotheses = [];
  while(preHypotheses.length > 0)
  {
    var hypotisis = preHypotheses.pop();
    var k = hypotisis[4];
```

```
    hypotisis[2] = hypotisis[5];
    hypotisis[3] = hypotisis[2];
    hypotisis[4] = hypotisis[3];
    hypotisis[5] = k;
    hypotisis[6] += 3;
    hypotisis[6] %= 4;
    newHypotheses.push(hypotisis);
  }
  sensor[3]=sensor[2]
  return newHypotheses;
}

//поворот право с учётом переноса гипотез
function turnRight( preHypotheses)
{
  turnDirection(-90,v);
  var newHypotheses = [];

while(preHypotheses.length > 0)
  {
  var hypotisis = preHypotheses.pop();
  var k = hypotisis[2];
  hypotisis[2] = hypotisis[3];
  hypotisis[3] = hypotisis[4];
  hypotisis[4] = hypotisis[5];
  hypotisis[5] = k;
  hypotisis[6] += 1;
  hypotisis[6] %= 4;
  newHypotheses.push(hypotisis);
}
sensor[3]=sensor[0]
return newHypotheses;
}

//проверка на возможность гипотезы
function check( sensors, cell)
{
  return   (sensor[0]==cell[2] && sensor[1]==cell[3] &&
            sensor[2]==cell[4] && sensor[3]==cell[5] ) ||
           (sensor[0]==cell[3] && sensor[1]==cell[4] &&
            sensor[2]==cell[5] && sensor[3]==cell[2] ) ||
           (sensor[0]==cell[4] && sensor[1]==cell[5] &&
            sensor[2]==cell[2] && sensor[3]==cell[3] ) ||
           (sensor[0]==cell[5] && sensor[1]==cell[2] &&
            sensor[2]==cell[3] && sensor[3]==cell[4] );
}
//перенос гипотезы вперёд
function go(hypotheses){
  var newHypotheses = []

  while(hypotheses.length>0){
    var cell = hypotheses.pop();
    switch(cell[6]){
      case 0: cell[1]--;break;
      case 1: cell[0]++;break;
      case 2: cell[1]++;break;
      case 3: cell[0]--;break;
```

```
      }
  if(cell[0]>=0 && cell[1]>=0 && cell[0]<=7 && cell[1]<=7)
    if(lab[2*cell[1]+1][2*cell[0]+1]!=0)
    {
      cell = generateCell(cell[6],cell[1],cell[0])
      newHypotheses.push(cell);
    }
  }
  return newHypotheses
}

//создание описания клетки
function generateCell(k , i , j ){
  switch(k){
    case 0:
          var cell = [];
          cell[0] = j;
          cell[1] = i;
          cell[2+0] = lab[2 * i + 1][2 * j];
          cell[2+1] = lab[2 * i ][2 * j +1];
          cell[2+2] = lab[2 * i  + 1][2 * j + 2];
          cell[2+3] = lab[2 * i + 2][2 * j + 1];
          cell[6] = 0;
          return cell;
    case 1:
          var cell = [];
          cell[0] = j;
          cell[1] = i;
          cell[2+3] = lab[2 * i + 1][2 * j];
          cell[2+0] = lab[2 * i ][2 * j +1];
          cell[2+1] = lab[2 * i  + 1][2 * j + 2];
          cell[2+2] = lab[2 * i + 2][2 * j + 1];
          cell[6] = 1;
          return cell;
    case 2:
          var cell = [];
          cell[0] = j;
          cell[1] = i;
          cell[2+2] = lab[2 * i + 1][2 * j];
          cell[2+3] = lab[2 * i ][2 * j +1];
          cell[2+0] = lab[2 * i  + 1][2 * j + 2];
          cell[2+1] = lab[2 * i + 2][2 * j + 1];
          cell[6] = 2;
          return cell;
    case 3:
          var cell = [];
          cell[0] = j;
          cell[1] = i;
          cell[2+1] = lab[2 * i + 1][2 * j];
          cell[2+2] = lab[2 * i ][2 * j +1];
          cell[2+3] = lab[2 * i  + 1][2 * j + 2];
          cell[2+0] = lab[2 * i + 2][2 * j + 1];
          cell[6] = 3;
          return cell;
  }
}
```

```
// карта
  var lab = [
    [0,0, 0,0, 0,0, 0,0, 0,0, 0,0, 0,0, 0,0,0],
    [0,1, 1,1, 1,1, 1,1, 1,1, 1,1, 0,0, 0,1,0],

    [0,1, 0,0, 1,1, 0,0, 0,0, 1,1, 1,0, 1,1,0],
    [0,1, 0,0, 0,1, 0,1, 0,0, 0,1, 1,1, 1,1,0],

    [0,1, 0,0, 1,1, 0,1, 0,0, 1,1, 0,0, 0,0,0],
    [0,1, 1,1, 1,1, 0,1, 1,1, 1,1, 0,0, 0,1,0],

    [0,0, 0,0, 1,1, 0,1, 0,0, 1,1, 1,0, 1,1,0],
    [0,1, 0,0, 0,1, 0,1, 0,0, 0,1, 1,1, 1,1,0],

    [0,1, 1,0, 1,1, 1,0, 1,0, 1,1, 0,0, 0,1,0],
    [0,1, 1,1, 1,1, 1,1, 1,1, 1,1, 0,0, 0,1,0],

    [0,1, 0,0, 1,1, 0,0, 1,1, 1,1, 1,0, 1,0,0],
    [0,1, 0,0, 0,1, 0,0, 0,1, 1,1, 1,1, 1,1,0],

    [0,0, 1,0, 1,1, 1,0, 0,0, 1,1, 0,0, 1,1,0],
    [0,1, 1,1, 1,1, 1,1, 0,0, 0,1, 0,0, 0,1,0],

    [0,1, 0,0, 1,1, 1,1, 1,0, 1,0, 1,0, 1,1,0],
    [0,1, 0,0, 0,1, 1,1, 1,1, 1,1, 1,1, 1,1,0],

    [0,0, 0,0, 0,0, 0,0, 0,0, 0,0, 0,0, 0,0,0]
    ]

function go_to_cell(x_begin,y_begin,directionOfRobot, x_end,
y_end){
  // creating array for path
  var path = [
  [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],
   [-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]],
  [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],
   [-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]],
  [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],
   [-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]],
  [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],
   [-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]],
  [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],
   [-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]],
  [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],
   [-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]],
  [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],
   [-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]],
  [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1],
   [-1,-1,-1],[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]],
  ];


  path[x_begin][y_begin][0] = 0;
  var x_temp = x_begin;
  var y_temp = y_begin;
  var queue = new Array();
```

```
// алгоритм a*
while(path[x_end][y_end][0] == -1){
  if((x_temp-1) >= 0){
    var coord_temp = x_temp-1;
    if(((path[coord_temp][y_temp][0]) == -1) &&
        (lab[y_temp*2+1][x_temp*2+1 -1] == 1)){
      path[coord_temp][y_temp][0] = path[x_temp][y_temp][0]+1;
      path[coord_temp][y_temp][1] = x_temp;
      path[coord_temp][y_temp][2] = y_temp;
      queue.push([coord_temp,y_temp]);
    }
  }


  if((x_temp+1) < 8){
    var coord_temp = x_temp+1;
    if(((path[coord_temp][y_temp][0]) == -1) &&
        (lab[y_temp*2+1][x_temp*2+1 +1] == 1)){
      path[coord_temp][y_temp][0] = path[x_temp][y_temp][0]+1;
      path[coord_temp][y_temp][1] = x_temp;
      path[coord_temp][y_temp][2] = y_temp;
      queue.push([coord_temp,y_temp]);
    }
  }

  if((y_temp-1) >= 0){
    var coord_temp = y_temp - 1;
    if(((path[x_temp][coord_temp][0]) == -1)  &&
        (lab[y_temp*2+1-1][x_temp*2+1] == 1)){
      path[x_temp][coord_temp][0] = path[x_temp][y_temp][0]+1;
      path[x_temp][coord_temp][1] = x_temp;
      path[x_temp][coord_temp][2] = y_temp;
      queue.push([x_temp,coord_temp]);
    }
  }

  if((y_temp+1) < 8){
    var coord_temp = y_temp + 1;
    if (((path[x_temp][coord_temp][0]) == -1) &&
         (lab[y_temp*2+1+1][x_temp*2+1] == 1)){
      path[x_temp][coord_temp][0] = path[x_temp][y_temp][0]+1;
      path[x_temp][coord_temp][1] = x_temp;
      path[x_temp][coord_temp][2] = y_temp;
      queue.push([x_temp,coord_temp]);
    }
  }
  var firstElementOfQueue = queue.shift();
  x_temp = firstElementOfQueue[0];
  y_temp = firstElementOfQueue[1];
  script.wait(3);
}


// зная предков каждой посещенной ячейки создаем
// стек необходимого перемещения робота
var stack = new Array();
x_temp = x_end;
```

```
y_temp = y_end;
while((x_temp != x_begin) || (y_temp !=y_begin)){
  stack.push([x_temp,y_temp]);
  var temp_cell = path[x_temp][y_temp];
  x_temp = temp_cell[1];
  y_temp = temp_cell[2];
}



// Проходим по стеку и перемещаем робота в
// соответствии с ним.
// В стеке у нас находятся последовательно
// ячейки в которые нам необходимо попасть.
// Последний элемент стека - ячейка в которую нам
// необходимо попасть используб данную функцию go_to_cell.
var len = stack.length
var prev_cell = [x_begin,y_begin];
for (var i = 0; i < len;i++){
  var temp_cell = stack.pop();
  var dx = temp_cell[0] - prev_cell[0];
  var dy = temp_cell[1] - prev_cell[1];

  if(dy == 0){
    if(dx > 0){
      if(directionOfRobot == 2){
        turnDirection(90,v);
        directionOfRobot --;

      }else{
        while(directionOfRobot != 1){
          turnDirection(-90,v);
          directionOfRobot ++;
          directionOfRobot %= 4;
        }
      }
    }else{
      if(directionOfRobot == 0){
        turnDirection(90,v);
        directionOfRobot = 3;

      }else{
        while(directionOfRobot != 3){
          turnDirection(-90,v);
          directionOfRobot ++;
          directionOfRobot %= 4;
        }
      }
    }
  }else{
    if(dy > 0){
      if(directionOfRobot == 3){
        turnDirection(90,v);
        directionOfRobot --;

      }else{
        while(directionOfRobot != 2){
```

```
            turnDirection(-90,v);
            directionOfRobot ++;
            directionOfRobot %= 4;
          }
        }
      }else{
        if(directionOfRobot == 1){
          turnDirection(90,v);
          directionOfRobot --;

        }else{
          while(directionOfRobot != 0){
            turnDirection(-90,v); // turn left
            directionOfRobot ++;
            directionOfRobot %= 4;
          }
        }
      }
    }
    forward1(v,1);


    prev_cell = temp_cell;
  }
  return directionOfRobot;
}

// ----------------- main part

// локализация
while(true){

  var sensor = [0,0,0,0];


  var hypotheses1 = [];
  var hypotheses = [];

  //создание гипотез
  for (i = 0; i < 8; i++)
  {
    for (j = 0; j < 8; j++)
    {
      if (lab[2 * i + 1][2 * j + 1]  = 1)
      {
         for(q = 0 ; q<4;q++)
        hypotheses.push(generateCell(q,i,j));
      }
    }
  }

  //чтение сенсоров
  sensor= readSensors(0);

  //первая проверка
  while (hypotheses.length>0)
  {
```

```
    var hypotisis = hypotheses.pop();
    if(firstCheck(sensor, hypotisis)){
      hypotheses1.push(hypotisis);
    }
  }

  //полная локализация
  while (hypotheses1.length>1)
  {
    if(sensor[0]==1){
      hypotheses = turnLeft(hypotheses1);
      hypotheses = forward(hypotheses);
      } else if(sensor[1] == 1)
      hypotheses = forward(hypotheses1);
      else {
      hypotheses = turnRight(hypotheses1);
        }

    sensor = readSensors(sensor[3]);

    while (hypotheses.length>0)
    {
      var hypotisis = hypotheses.pop();
      if (check(sensor, hypotisis))
        hypotheses1.push(hypotisis)
    }

    script.wait(1000);

  }
  if(hypotheses1.length == 1)break;
}

var x = hypotheses1[0][0];
var y = hypotheses1[0][1];
var dir = hypotheses1[0][6];
print(x + " " + y + " " + dir);
dir = go_to_cell(x,y,dir,7,1);
script.wait(5000);
var xOfTheEnd = 5;
dir = go_to_cell(7,1,dir,4,5);
script.wait(5000);
var yOfTheEnd = 7;
dir = go_to_cell(4,5,dir,xOfTheEnd,yOfTheEnd);
print("Prodram end!");
led.orange();
brick.stop();
script.quit();
```

Пример программы на языке QtScript, обеспечивающий решение задание третьего этапа:

```
var pi = 3.1415926535897931;

var readGyro = brick.gyroscope().read
function readYaw() { return  -readGyro()[6]; }
```

```
//моторы
var mLeft = brick.motor(M3).setPower;
var mRight = brick.motor(M4).setPower;

//сенсоры освещенности
sA5 = brick.sensor(A5);
sA6 = brick.sensor(A6);

//энкодеры
var eLeft = brick.encoder(E3);
var eRight = brick.encoder(E4);

eLeft.reset();
eRight.reset();

var el = eLeft.readRawData();
var er = eRight.readRawData();
mLeft(0);
mRight(0);
print("Start");

//инициализация и калибровка гироскопа
print("gyro initialaize...");
brick.gyroscope().calibrate(12000);
script.wait(13000);
print("gyro inited");
brick.display().addLabel(30, 10, "Start!");
brick.display().redraw();
script.wait(1000);

var encLeftOld = 0;
var encRightOld = 0;
var encLeft = 0;
var encRight = 0;

var main = function(){
  mLeft(0);
  mRight(0);

  var _alpha = 0;
  var _vel = 25;
  var u = 0;

  // едем до калибровочной линии
  while((sA5.read() < 90) && (sA6.read() < 90)){
    u = -0.6*(_alpha-readYaw()/1000);
    mLeft(_vel - u);
    mRight(_vel + u);
    script.wait(5);
  }

  _vel = 15;
  // количество тиков энкодера в одной полосе кода
  var _encOne = 27; // выяснено эмпирически
  encLeft = eLeft.readRawData();
  encLeftOld = encLeft;
  var twoInPower = 8;
```

```
  var numb = 0;
  var count = 1;
  var sv;

  // считываем значений 4х битов
  while(count < 5){
    u = -0.5*(_alpha-readYaw()/1000);
    mLeft(_vel - u);
    mRight(_vel + u);
    encLeft = eLeft.readRawData();
    sv = sA5.read();
    // мы считываем значение белая/черная линия
    // с частотой дискретизации _encOne
    if(Math.abs(encLeft - encLeftOld) >= count * _encOne){
      count ++;
      if(sv > 90){
        numb += twoInPower;
      }
      twoInPower = twoInPower / 2;
    }
    script.wait(3);
  }

  print(numb);

  brick.stop();
  script.quit();
}
```

3.3. Критерий определения победителей и призеров заключительного этапа

В заключительном этапе олимпиады баллы участника складываются из двух частей: он получает баллы за индивидуальное решение задач по предметам (математика и информатика) и за командное решение практической задачи.

$$S = 0{,}36*S_1 + 0{,}64* S_2$$

где S_1 — количество баллов, набранное в рамках индивидуальной части заключительного этапа (максимум 200 баллов); S_2 — количество баллов, набранное в рамках командной части заключительного этапа (максимум 400 баллов).

Критерий определения победителей и призеров:

	Призеры	**Победители**
Набранные баллы	от 120 до 150 баллов	от 150 баллов и выше