Командный тур

В командной части заключительного этапа участники должны спроектировать автономную систему управления групой роботов-погрузчиков для решения задач навигации на модели логистического центра с использованием алгоритмов компьютерного зрения.

Командная часть заключительного этапа проходит в течении 3.5 дней (всего 26 астрономических часов), которые включают работу по оснащению роботов необходимыми датчиками, программированию, пробные заезды на макете логистического центра, зачетные попытки.

6.1. Легенда

Недалёкоое будущее, в автоматических логистических центрах практически нет людей, всю работу выполняют роботы-погрузчики, которыми управляет интеллектуальное ПО по распределению задач.

Несмотря на отсутствие человеческого фактора, не следует думать, что в таких центрах никогда не будет проблем. Форс-мажор может произойти в любой момент и система из роботов и ПО должна уметь справляться с такими ситуациями.

Поэтому для финальной задачи профиля «Интеллектуальные робототехнические системы» предлагается рассмотреть следующий эпизод: в логистическом центре произошла перезагрузка всех систем, что привело к сбросу информации о местоположениях работающих в данный момент роботов-погрузчиков.

В начале выполнения задания считается, что робототехнические устройства активируются в каких-то секторах (каждый в своём) логистического центра. При этом 2 из них имеют информацию о своём местоположении, а у 3-го была повреждена память и по этому эта информация была потеряна. Структура логистического центра известна заранее всем устройствам. Роботы-погрузчики должны переместиться в сектор своей приписки. Координаты секторов сервисного обслуживания известны заранее. Информацию о местоположении необходимого сектора приписки можно узнать в секторе сервисного обслуживания. В нём располагается ARTag маркер, в котором закодирован номер робота и координаты сектора его приписки. При перемещении роботы не должны сталкиваться, а также повреждать логистический центр.

Задача участников Олимпиады — разработать программу управления несколькими робототехническими устройствами для выполнения задания описанного выше.

Рис. 6.1: Полигон для запуска робототехнических устройств на финале Олимпиады НТИ

6.2. Набор заданий

Решение командной задачи разбито на 5 этапов. Первые четыре этапа итеративно подводят участников к решению полной финальной задачи, осуществляемому во время последнего пятого этапа. На каждом этапе в проверку решения заданий данного этапа входят:

- способность проверить гипотезу о работоспособности алгоритма через демонстрацию решения в симуляторе;
- полнота решения задания конкретного этапа;
- воспроизводимость результатов — робототехническое устройство участников должно было неоднократно выполнить требуемые действия.

Первый этап

Задача: три робототехнических устройства располагаются в модели логистического центра. Им необходимо обменяться данными о наличии или отсутствии стен слева от робота, перед собой и справа от робота, где 1 — присутствует стена, 0 — отсутствует. Выводить показация следует на экран всех роботов непрерывно. Данные, принятые с каждого робота, необходимо вывести на разных строках при этом показания с одного робота следует выводить через пробел в порядке указаном выше.

Включая содержательные задачи:

- Нахождение порогового значения для датчиков расстояния;
- Реализация коммуникации между робототехническими устройствами.

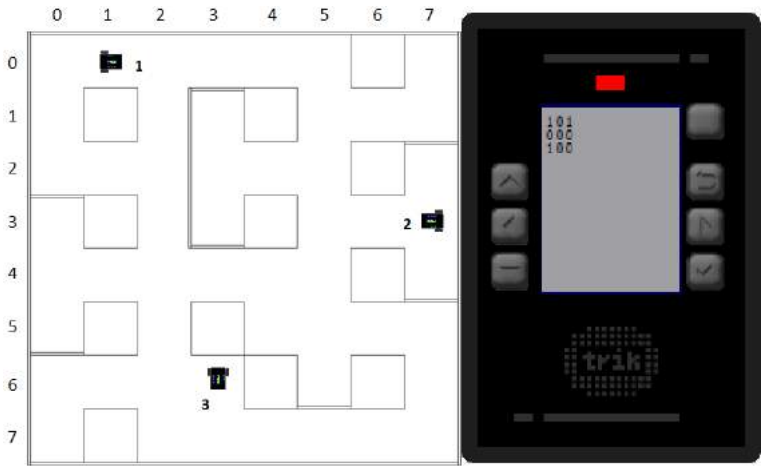

Рис. 6.2: Пример расположения роботов и показания выведенные на экран

Второй этап

Задача: три робототехнических устройства должны определить своё местоположение в модели логистического центра из случайных точек старта, одна из которых заранее не известна.

Включая содержательные задачи:

- Реализация коммуникации между робототехническими устройствами;
- Реализация алгоритмов локализации для группы робототехнических устройств;
- Реализация алгоритмов распределения группы роботов с целью достижения общей задачи;
- Реализация алгоритмов движения нескольких роботов в замкнутой среде.

Третий этап

Задача: три робототехнических устройства должны проехать по модели логистического центра из точек старта в сектора финишей, не столкнувшись.

Включая содержательные задачи:

- Реализация коммуникации между робототехническими устройствами;
- Реализация алгоритмов движения нескольких роботов в замкнутой среде.

Четвёртый этап

Задача: робототехническое устройство должно проехать по модели логистического центра из сектора старта в сектор финиша через сектор сервисного обслуживания. Координаты сектора финиша робот должен определить самостоятельно по ARTag метке, расположенной на стеллаже, прилегающем к сектору сервисного обслуживания.

Включая содержательные задачи:

- Калибровка камеры робототехнического устройства;

- Реализация алгоритмов компьютерного зрения: считывание ARTag меток без использования дополнительных библиотек;
- Декодирование бинарного кода с использованием кода Хэмминга.

Пятый этап

Задача: три робототехнических устройства должны проехать по модели логистического центра из случайных точек старта, одна из которых заранее не известна, в соответсвующие сектора финиша. Координаты секторов финиша для каждого робота указаны в ArTag маркерах, расположенных в секторах сервисного обслуживания.

Включая содержательные задачи:

- Реализация коммуникации между робототехническими устройствами;
- Реализация алгоритмов локализации для группы робототехнических устройств;
- Реализация алгоритмов распределения группы роботов с целью достижения общей задачи;
- Реализация алгоритмов компьютерного зрения: считывание ARTag меток без использования дополнительных библиотек;
- Декодирование бинарного кода, с использованием кода Хэмминга;
- Реализация алгоритмов движения нескольких роботов в замкнутой среде.

6.3. Описание модели логистического центра

Полигон - квадратное поле 3200 × 3200 мм., разделенное на квадратные сектора 400 × 400 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 с грузом (рис. 6.3) размером 200 × 300 × 100 мм (груз - 2-3 коробки (рис. 6.4) размером 180 × 80 × 85 мм).

Рис. 6.3: Внешний вид контейнера

Рис. 6.4: Внешний вид коробки

Полигон окружен бортом высотой 100 мм. Конфигурация полигона определяется в первый день финального этапа и объявляется участникам. Данная конфигурация

будет использоваться во все дни финального этапа.

На нижней полке стеллажа, прилегающего одной из четырех сторон к сектору сервисного обслуживания, на высоте от 100-150 мм от уровня поверхности поля закреплен ARTag маркер (`https://goo.gl/WaTFMB`), определяющий номер робота и координаты сектора приписки (сектор финиша для конкретного робота). Размер маркера - 30×30 мм. Маркер обращен лицевой стороной внутрь сектора сервисного обслуживания, координаты которого известны заранее. Конкретная высота расположения маркеров определяется в первый день финала и остается постоянной во все дни финального этапа. При этом допустимая погрешность установки маркеров ± 5 мм. Пример расположения маркера на стелаже представлен на рисунке 6.5

Рис. 6.5: Стеллаж с установленным ARTag маркером

Маркер состоит из 6×6 элементов одинакового размера. Элементы маркера, расположенные по его границе — всегда черные. Четыре элемента, находящиеся в углах внутреннего 4×4 квадрата определяют ориентацию маркера таким образом, что только один из них — белый. Оставшиеся 12 элементов маркера кодируют число по следующему правилу: если элемент черный, то в он обозначает 1, если белый, то 0 при этом самый первый элемент — старший бит закодированного числа. Нумерация элементов относительно ориентационных элементов обозначена на рисунке 6.6.

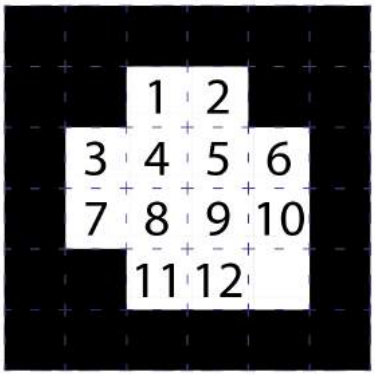

Рис. 6.6: Нумерация элементов маркера относительно ориентационных элементов

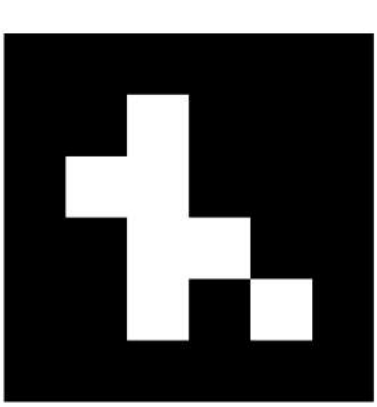

Рис. 6.7: Маркер с закодированным значением - 010011100101_2

Закодированное на маркере двоичное двенадцатибитное число закодировано с использованием кода Хэмминга (https://habr.com/ru/post/140611/), в котором 8 информационных битов и 4 контрольных бита:

1 Первый контрольный бит;

2 Второй контрольный бит;

3 Старший бит номера робота $N(1 \leqslant N \leqslant 3)$;

4 Третий контрольный бит;

5 Младший бит номера робота $N(1 \leqslant N \leqslant 3)$;

6 Первый (старший) бит координаты X робота N $(0 \leqslant X_N \leqslant 7)$;

7 Второй бит координаты X робота N $(0 \leqslant X_N \leqslant 7)$;

8 Четвёртый (последний) контрольный бит;

9 Третий (младший) бит координаты X робота N $(0 \leqslant X_N \leqslant 7)$;

10 Первый (старший) бит координаты Y робота N $(0 \leqslant Y_N \leqslant 7)$;

11 Второй бит координаты Y робота N $(0 \leqslant Y_N \leqslant 7)$;

12 Третий (младший) бит координаты Y робота N $(0 \leqslant Y_N \leqslant 7)$;

Маркер на рисунке 6.7 кодирует число 010011100101_2 — $N = 1$, $X = 6$, $Y = 5$. Таким образом, сектор приписки для робота 1 находится в позиции с координатами $(6, 5)$ — см. рисунок 6.8.

Гарантируется, что сектора сервисного обслуживания будут располагаться в таких местах полигона, где к сторонам сектора прилегает как минимум один стеллаж.

Сектора активаций роботов (стартов), сектора сервисного обслуживания и сектора приписки никак не обозначаются на поле и определяются непосредственно перед каждым заездом робота.

6.4. Описание конструктора

В первый день финального тура каждой команде выдаются три комплекта:

- Мобильная наземная платформа на базе конструктора TRIK в сборе (блок управления TRIK, аккумулятор, два мотора с энкодерами на датчиках Холла, колеса), но без установленных датчиков. Мобильная платформа построе-

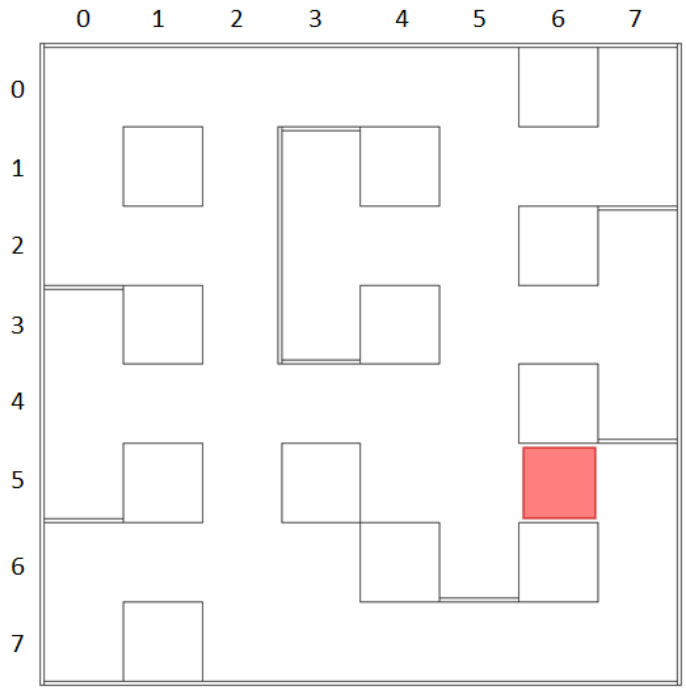

Рис. 6.8: Расположение сектора финиша для маркера из рисунка 6.7

на по принципу дифференциального управления. Физические размеры платформы позволяют совершать все маневры внутри одного сектора модели логистического центра без касания со стенками стеллажей или бортов.

- Комплект дополнительных деталей из конструктора TRIK и набор датчиков: 2 инфракрасных датчика дальности, 2 ультразвуковых датчика расстояния и 1 VGA-камера;
- комплект дополнительных деталей из конструктора TRIK;
- Ноутбуки с установленной TRIK Studio, каждой команде по одному ноутбуку. При этом участники могут пользоваться своими ноутбуками.

6.5. Условия проведения

1. Из полученного набора датчиков команды могут выбирать те, с помощью которых, по мнению участников, можно решить задачу наиболее эффективным способом.
2. Команды могут вносить любые изменения в мобильные наземные платформы.
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 выполняют решение задачи поэтапно. Критерии прохождения каждого этапа формулируются для каждого дня финального тура. За подзадачи, решенные в конкретном этапе начисляются баллы. Баллы за под-

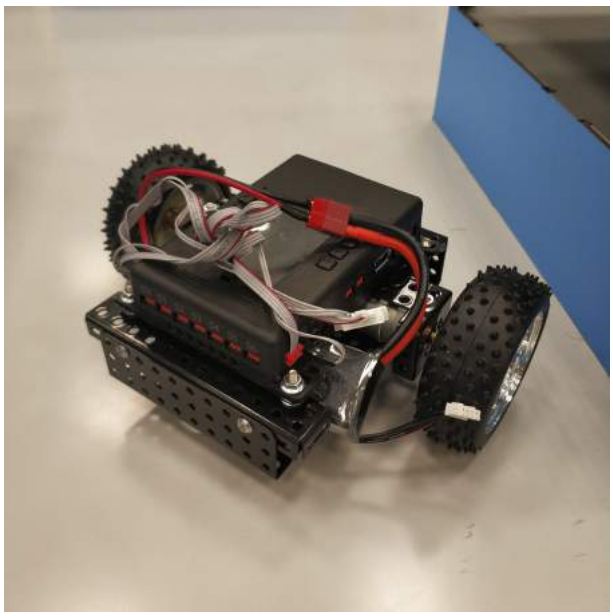

Рис. 6.9: Мобильная платформа TRIK в сборе

задачи можно получить только в день, закрепленный за конкретным этапом.

6. Некоторые подзадачи строго требуют выполнение каких-то предыдущих подзадач. Выполнение данных подзадач без выполнения предыдущих допускается, однако данная попытка будет оценена в 0 баллов.
7. При выполнении подзадачи в виртуальной среде полное количество баллов можно получить лишь при первой попытке. При второй попытке сдать подзадачу может быть начислено лишь половина баллов за данную задачу. При последующих попытках баллы не начисляются.
8. Во время рабочего времени команды могут проводить испытания на полигоне. Количество подходов, которое может сделать команда может быть ограничено в зависимости ограничений, накладываемых расписанием финального этапа Олимпиады.
9. Испытания на полигоне должны осуществляться так, чтобы не мешать другим командам, проводящим в это время свои испытания на полигоне. Для этого всем командам может быть назначено ограничение по времени, которое они могут тратить на одно испытание. После истечения этого времени, команда должна дать возможность проводить испытания следующей команде.
10. Часть подзадач необходимо будет решить в симуляторе TRIK Studio: команда получает 3 тестовых виртуальных полигона с соответствующими наборами входные данных для подготовки решения, в то время как приемка решения происходит на расширенном наборе полигонов для проверки универсальности управляющей программы. Начисление баллов за подзадачи может происходить только в тот этап, в котором данные подзадачи сформулированы.
11. У команды есть не более двух попыток для сдачи решения подзадач в симуляторе:
    (a) Решения принимаются на проверку до истечения первых 6 часов работы в соответствующий соревновательный день. Может меняться в зависимости от дня.

(b) До истечения 6 часов, команда должна загрузить свое решение на *Google Drive* в каталог, доступ к которому участники получат в начале дня. Участники команды ответственны за то, что ссылка на каталог с их решениями не попадет участникам других команд.

(c) До истечения указанного времени команды могут изменять файл с решением сколько угодно раз. Проверяться будет всегда только последняя доступная версия.

(d) Если решение отправлено на проверку в течение первых 4 часов работы в соответствующий соревновательный день, то команда имеет право на вторую попытку, если результаты проверки решения ее не устраивают.

(e) Если команда хочет воспользоваться правом проверки решения до истечения 4 часов, то она должна загрузить в каталог на *Google Drive* программу со своим решением и сообщить об этом судьям.

12. Часть подзадач для реальных роботов может быть запрещена к приемке без успешного прохождения 60% всех тестов, предназначенных для проверки решения соответствующей подзадачи в симуляторе.

13. Каждый день финального тура за 2 часа (может варьироваться в зависимости от расписания) 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.

14. Перед сдачей робота в карантин команды должны загрузить на роботов управляющие программы, подготовленные для демонстрации решения задачи, а также ее копию в *Google Drive* в каталог, доступ к которому участники получат в начале соревновательного дня. Без программы, загруженной в каталог *Google Drive*, команды не допускаются до проверки решения на реальном роботе.

15. После момента, когда все роботы сданы в карантин, судьи по одной вызывают команды для приемки решения подзадач, закрепленных за этапом конкретного дня финального тура.

16. Может быть предусмотрено до двух попыток сдачи решения одной и той же подзадачи на реальных роботах. Конкретное количество попыток определяется в конкретных подзадачах.

17. 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.

18. Роботы должны выполнять задание полностью автономно. Удаленное управление не допускается. Касание какого-либо робота участником команды после его старта во время приемочных запусков не допускается. Алгоритм, реализующий систему управления группой роботов, должен планировать свое выполнение, полагаясь только на информацию с датчиков.

19. Введение данных в программу до старта устройства (например, координат робота в начале работы) разрешается только для тех задач, где это явно прописано. Во всех других случаях введение данных в программу роботов перед запуском запрещено.

20. Для всех роботов программа должна быть одинаковой, допускается отличие лишь в разрешенных входных данных.
21. Если какая-то подзадача подразумевает считывание информации с элементов, расположенных на полигоне, запрещается при запуске роботов вводить информацию о положении этих элементов или значениях, которые данные элементы определяют.
22. Если во время приемочных запусков у судьи возникли сомнения о том, что задачи подэтапа решены корректно (роботы не выполняют задачу полностью автономно, участник вводит значения в каких-либо роботов перед запуском), то он вправе провести инспекцию кода. По результатам инспекции, судья вправе снять с команды баллы, набранные за данный этап.
23. 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.
24. Команда вправе обсуждать с судьей результаты приемочных запусков до вызова следующей команды, но финальное решение остается о начислении баллов остается за судьей.

6.6. Процедура проведения приемочных запусков и критерии оценки

Первый этап

1. Командам будет предоставлено две попытки на демонстрацию решения задачи на реальном роботе.
2. Обе попытки будут осуществляться 7 марта.
3. Необходимо сдать данную задачу в любой момент периода отладки.
4. Правила именования файлов с программой управления:
   (a) для первой попытки: `part1_1.js`.
   (b) для второй попытки: `part1_2.js`.
5. Максимальное время выполнения одной попытки - 30 секунд.
6. Баллы за решение задач этапа:
   (a) **Реальный робот:**
      i. Один из роботов вывел показания всех 9ти датчиков расстояния — 2 балла;
      ii. Все роботы вывели показания всех 9ти датчиков расстояния один раз — 2 балла;
      iii. Все роботы выводят показания всех 9ти датчиков расстояния непрерывно — 2 баллов;
7. Баллы за две попытки суммируются.
8. Выполнение всех критериев в каждой из двух попыток всех двух подзадач дает дополнительные 2 балла.
9. Максимальное количество баллов за этап — 14.

Второй этап

1. Командам необходимо подготовить две задачи для симулятора:
   (a) В качестве первой задачи участникам необходимо выполнить следующее:
      i. Робот устанавливается в модели логистического центра в заранее неизвестном секторе. Необоходимо в процессе движения опредлить своё местоположение, остановиться и вывести на экран координаты робота в формате "(X, Y)" без пробелов и без кавычек.

      Ожидаемый результат: После запуска программы робот определил своё местоположение, остановился и вывел на экран координаты своего местоложожения в формате "(X, Y)" без пробелов и без кавычек.

   (b) В качестве второй задачи:
      i. Робот устанавливается в модели логистического центра в заранее неизвестном секторе. Известно, что два сектора данного логистического центра заблокированы другими роботами. Координаты этих секторов передаются через входной файл. Необоходимо в процессе движения опредлить своё местоположение, остановиться и вывести на экран координаты робота в формате "(X, Y)" без пробелов и без кавычек.

      Входные данные: через файл `input.txt` управляющей программе передаются:
         A. В первой строке через пробел — координаты одного заблокированного сектора X_1, Y_1, $0 \leqslant X_1, Y_1 \leqslant 7$;
         B. Во второй строке через пробел — координаты другого заблокированного сектора X_2, Y_2, $0 \leqslant X_2, Y_2 \leqslant 7$;

      Ожидаемый результат: После запуска программы робот определил своё местоположение, остановился и вывел на экран координаты своего местоложожения в формате "(X,Y)" без пробелов и без кавычек.

   (c) Правила именования файлов с управляющей программой для проверки решений в симуляторе:
      i. Для первой задачи: `sim_part2_1.qrs`;
      ii. Для второй задачи: `sim_part2_2.qrs`.
2. Команде необходимо будет подготовить решения для двух разных подзадач для реальных роботов. На демонстрацию каждого решения предоставляется 2 попытки.
3. Все попытки осуществляются 7 марта.
4. После сдачи в карантин для 1ой подзадачи судья определяет сектор старта и направление робота в секторе старта.
5. За 5 мин до сдачи в карантин для 2ой подзадачи судья определяет сектора старта и направления в секторах старта двух роботов. Данные значения команда должна внести в программу перед тем, как сдать робота в карантин.
6. После сдачи в карантин для 2ой подзадачи судья определяет сектор старта

и направление в секторе старта для оставшегося робота.

7. Секторы старта могут быть различными в разных попытках. Данные значения команда должна внести в программу перед тем, как сдать робота в карантин.
8. Правила именования файлов с программой управления:
   (a) для первой попытки первой подзадачи: `part2_1_1.js`;
   (b) для второй попытки первой подзадачи: `part2_1_2.js`;
   (c) для первой попытки второй подзадачи: `part2_2_1.js`;
   (d) для второй попытки второй подзадачи: `part2_2_2.js`.
9. Максимальное время выполнения одной попытки - 5 минут.
10. Баллы за решение задач этапа:
    (a) **Первая задача в симуляторе**: робот смог определить своё местоположение на всех проверочных полигонах — 12 баллов.
    (b) **Вторая задача в симуляторе**: робот смог определить своё местоположение на всех проверочных полигонах — 14 баллов.
    (c) **Первая подзадача на реальном роботе:** Робот располагается с случайном секторе робототехнического полигона. Робот смог определить своё местоположение на поле, остановился, издал звуковой сигнал, вывел на экран свои координаты в формате "(X, Y)" — 14 баллов.
    (d) **Вторая подзадача на реальном роботе:** Роботы располагается с случайных секторах робототехнического полигона. Роботы смогли определить своё местоположение на поле, остановились, издали звуковой сигнал, вывел на экран свои координаты в формате "(X, Y)", а также вывели `finish` — 18 баллов.
11. Баллы за первую подзадачу не начисляются, если не было частично засчитано решение первой задачи в симуляторе
12. Баллы за вторую подзадачу не начисляются, если не были частично засчитаны решения за все задачи в симуляторе.
13. Баллы за все попытки в каждой подзадаче суммируются.
14. Выполнение всех критериев в каждой из двух попыток всех двух подзадач дает дополнительные 4 балла.
15. Максимальное количество баллов за этап — 94.

Третий этап

1. Командам необходимо подготовить две задачи для симулятора:
   (a) В качестве первой задачи участникам необходимо выполнить следующее:
      i. Робот устанавливается в модели логистического центра в заранее неизвестном секторе. Задача робота проехать из точки старта в точку финиша.

      Входные данные: через файл `input.txt` управляющей программе передаются:

A. В первой строке через пробел — координаты сектора старта X_s, Y_s и направление старта D_s (направление робота в секторе старта от 0 до 3 начиная с направления вверх и дальше по часовой стрелке), $0 \leqslant X_s, Y_s \leqslant 7$;

B. В второй строке через пробел — координаты сектора финиша X_f, Y_f, $0 \leqslant X_f, Y_f \leqslant 7$.

Ожидаемый результат: После запуска программы робот перемещается в сектор финиша по оптимальному пути. Оптимальным является путь, длина строки маршрута, которого наименьшая. Для маршрута используется следующая нотация действий без пробелов:

A. F — проезд в следующий сектор по ходу движения;

B. L — поворот налево в данном секторе;

C. R — поворот направо в данном секторе.

После остановки на экран робота выведен путь в описанной нотации выше.

(b) В качестве второй задачи:

i. Роботы устанавливаются в модели логистического центра в заранее неизвестном секторе. Задача роботов проехать из точки старта в точки финиша.

Входные данные: через файл `input.txt` управляющей программе передаются:

A. В первой строке через пробел — координаты сектора старта первого робота X_{1s}, Y_{1s}, направление старта первого робота D_{1s} (направление робота в секторе старта от 0 до 3 начиная с направления вверх и дальше по часовой стрелке) и координаты сектора финиша первого робота X_{1f}, Y_{1f}, $0 \leqslant X_{1s}, Y_{1s}, X_{1f}, Y_{1f} \leqslant 7$;

B. Во второй строке через пробел — координаты сектора старта второго робота X_{2s}, Y_{2s}, направление старта второго робота D_{2s} (направление робота в секторе старта от 0 до 3 начиная с направления вверх и дальше по часовой стрелке) и координаты сектора финиша вторго робота X_{2f}, Y_{2f}, $0 \leqslant X_{2s}, Y_{2s}, X_{2f}, Y_{2f} \leqslant 7$;

C. В третьей строке через пробел — координаты сектора старта третьего робота X_{3s}, Y_{3s}, направление старта третьего робота D_{3s} (направление робота в секторе старта от 0 до 3 начиная с направления вверх и дальше по часовой стрелке) и координаты сектора финиша третьего робота X_{3f}, Y_{3f}, $0 \leqslant X_{3s}, Y_{3s}, X_{3f}, Y_{3f} \leqslant 7$;

Ожидаемый результат: После запуска программы происходит вычисление планируемых маршрутов для каждого робота. После вычисления в файл “output.txt” записана одна строка следующего вида: $1P_1 2P_2 3P_3$, без пробелов, где $1P_1$, P_2 и P_3 – маршруты соответсвующих роботов. Для вывода маршрута используется следующая нотация без пробелов:

- F — проезд в следующий сектор по ходу движения;
- L — поворот налево в данном секторе;
- R — поворот направо в данном секторе;
- S — остановка в данном секторе.

Роботы движутся без столкновений и так, что команды выполняются паралельно и любое действие занимает одинаковое количество времени.

(c) Правила именования файлов с управляющей программой для проверки решений в симуляторе:

i. Для первой подзадачи: `sim_part3_1.js`;

ii. Для второй подзадачи: `sim_part3_2.js`.

2. Команде необходимо будет подготовить решения для двух разных подзадач для реальных роботов. На демонстрацию каждого решения предоставляется 2 попытки.
3. Все попытки осуществляются 8 марта.
4. За 15 минут до времени сдачи роботов в карантин для данных подзадач судья определяет сектора старта и финиша, а также направление робота в секторе старта. Данные значения команда должна внести в программу перед тем, как сдать робота в карантин.
5. Секторы старта и финиша могут быть различными в разных попытках.
6. Правила именования файлов с программой управления:
   (a) для первой попытки первой подзадачи: `part3_1_1.js`;
   (b) для второй попытки первой подзадачи: `part3_1_2.js`;
   (c) для первой попытки второй подзадачи: `part3_2_1.js`;
   (d) для второй попытки второй подзадачи: `part3_2_2.js`.
7. Максимальное время выполнения одной попытки - 5 минут.
8. Баллы за решение задач этапа:
   (a) **Первая задача в симуляторе**: робот проехал из точки старта в точку финиша и вывел на экран маршрут робота на всех проверочных полигонах — 6 баллов.
   (b) **Вторая задача в симуляторе**: В консоль выведены верные маршруты перемещения из точки старта в точку финиша для всех роботов на всех проверочных полигонах.
      i. Маршруты позволяют роботам доехать из начальных в конечные координаты – 10 баллов.
      ii. В маршруте кажного робота было использовано не более 3х команд S – 4 балла.
   (c) **Первая подзадача на реальном роботе:** Робот доехал до сектора финиша, остановился и вывел `finish` — 8 баллов.
   (d) **Вторая подзадача на реальном роботе:** Роботы располагается с случайных секторах робототехнического полигона.
      i. Первый робот доехал до сектора финиша не задев остальных роботов, остановился и вывел `finish` — 10 баллов.

ii. Второй робот доехал до сектора финиша не задев остальных роботов, остановился и вывел `finish` — 12 баллов.

iii. Третий робот доехал до сектора финиша не задев остальных роботов, остановился и вывел `finish` — 12 баллов.

9. За первую подзадачу начисляется только половина возможных баллов, если не было засчитано решение первой задачи в симуляторе
10. За вторую подзадачу начисляется только половина возможных баллов, если не были засчитаны решения за все задачи в симуляторе.
11. Баллы за все попытки в каждой подзадаче суммируются.
12. Выполнение всех критериев в каждой из двух попыток всех трех подзадач дает дополнительные 4 балла.
13. Максимальное количество баллов за этап — 108.

Четвёртый этап

1. В качестве задачи для симулятора участникам необходимо выполнить следующее:
   (a) Робот устанавливается в модели логистического центра в случайных секторе. При этом структура логистического центра известна заранее. Задача робота проехать из точки старта в точку финиша, чьи координаты заданы изображением ARTag маркера, заранее считанным с камеры реального устройства. На финише необходимо вывести на экран слово `finish`.

   Входные данные: через файл `input.txt` управляющей программе передаются:

   - В первой строке через пробел — координаты сектора старта X_s, Y_s и направление старта D_s (направление робота в секторе старта от 0 до 3 начиная с направления вверх и дальше по часовой стрелке), $0 \leqslant X_s, Y_s \leqslant 7$;
   - Во второй строке 19200, разделенных пробелом, целых чисел $P_{1,i}$ $(0 \leqslant P_{1,i} \leqslant 2^{24})$ — изображение ARTag маркера;

   Каждое число в маркере — точка, закодированная в формате RGB, т.е. строка с изображением маркера эквивалентна снимку разрешением 160×120 точек. Один маркер кодирует координаты для первого робота, в данной ситуации являющегося единственным.

   Ожидаемый результат: После запуска программы робот перемещается в сектор финиша. После остановки на экран робота выведено `finish`.

   (b) Имя файла с управляющей программой для проверки решения в симуляторе: `sim_part4.js`.
2. Команде необходимо будет подготовить решения для двух разных подзадач для реального робота. На демонстрацию каждого решения предоставляется 2 попытки.
3. Все попытки осуществляются 9 марта.
4. За 15 минут до времени сдачи роботов в карантин для 2ой подзадачи судья

определяет сектор старта и направление робота в секторе старта. Данные значения команда должна внести в программу перед тем, как сдать робота в карантин.

5. За 15 минут до времени сдачи роботов в каранин судья определяет сектор сервисного обслуживания (сектор, из которого необходимо сканировать ARTag метку) и направление расположения ARTag метки (0 - маркеры находятся на “верхнем” стеллаже, 1 - на “правом” стеллаже и т.д. по часовой стрелке). Данные значения команда должна внести в программу перед тем, как сдать робота в карантин.
6. Сектор старта и сектор сервисного обслуживания может быть разным для каждой попытки.
7. Правила именования файлов с программой управления:
   (a) для первой подзадачи: `part4_1.js`;
   (b) для первой попытки второй подзадачи: `part4_2_1.js`;
   (c) для второй попытки второй подзадачи: `part4_2_2.js`.
8. Максимальное время выполнения одной попытки - 3 минуты.
9. Баллы за решение задач этапа:
   (a) **Симулятор**: робот проехал из точки старта в точку финиша на всех проверочных полигонах — 6 баллов.
   (b) **Первая подзадача на реальном роботе:** Робот верно распознал ARTag метку, которая установлена прямо перед камерой, и вывел на экран верное значение, закодированное на метке — 2 балла.
   (c) **Вторая подзадача на реальном роботе:** Робот располагается за два сектора до сектора сервисного обслуживания:
      i. Робот доехал до сектора сервисного обслуживания, распределения задач, остановился, издал звуковой сигнал, вывел на экран верное ARTag метки — 4 балла.
      ii. Робот доехал до указаных в метке координат, остановился, издал сигнал и вывел на экран `finish`— 8 баллов.
10. За вторую подзадачу начисляется только половина возможных баллов, если не было засчитано решение в симуляторе, а также если не сдана первая подзадача.
11. Выводить значение метки и `finish` следует не менее 10 секунд.
12. Баллы за все попытки в каждой подзадаче суммируются.
13. Выполнение всех критериев в каждой из двух попыток всех трех подзадач дает дополнительные 4 балла.
14. Максимальное количество баллов за этап — 38.

Пятый этап

1. В качестве задачи для симулятора участникам необходимо выполнить следующее:
   (a) Робот устанавливается в модели логистического центра в заранее неизвестном секторе. При этом структура логистического центра из-

вестна заранее. Задача робота локализоваться и доехать в точку финиша, чьи координаты заданы изображением ARTag маркера, заранее считанным с камеры реального устройства. На финише необходимо вывести на экран слово `finish`.

Входные данные: через файл `input.txt` управляющей программе передаются:

- В первой строке 19200, разделенных пробелом, целых чисел $P_{1,i}$ ($0 \leqslant P_{1,i} \leqslant 2^{24}$) — изображение ARTag маркера;

Каждое число в маркере — точка, закодированная в формате RGB, т.е. строка с изображением маркера эквивалентна снимку разрешением 160×120 точек. Один маркер кодирует координаты для первого робота, в данной ситуации являющегося единственным.

Ожидаемый результат: После запуска программы робот перемещается в сектор финиша. После остановки на экран робота выведено `finish`.

(b) Имя файла с управляющей программой для проверки решения в симуляторе: `sim_part5.qrs`.

2. Командам будет предоставлено две попытки на демонстрацию решения задачи на реальном роботе.
3. Все попытки осуществляются 10 марта.
4. За 15 мин до сдачи в карантин судья определяет сектора старта и направления в секторах старта двух роботов. Данные значения команда должна внести в программу перед тем, как сдать робота в карантин.
5. После сдачи в карантин судья определяет сектор старта и направление в секторе старта для оставшегося робота.
6. В начале соревновалельного дня определяются сектора сервисного обслуживания (сектора, из которых необходимо сканировать ARTag метки) и направление расположения ARTag меток (0 - маркеры находятся на “верхнем” стеллаже, 1 - на “правом” стеллаже и т.д. по часовой стрелке). Данные значения команда должна внести в программу самостоятельно.
7. Сектора сервисного обслуживания являются одинаковыми для всех попыток.
8. Сектора сервисного обслуживания: первый — “$(2, 1)$ 3”, второй — “$(2, 5)$ 3”, третий — “$(5, 2)$ 1”.
9. Сектора старта могут быть различными в разных попытках.
10. Правила именования файлов с программой управления:
    (a) для первой попытки: `part5_1.js`.
    (b) для второй попытки: `part5_2.js`.
11. Максимальное время выполнения одной попытки - 5 минут.
12. Баллы за решение задач этапа:
    (a) **Симулятор:**
        i. Робот проехал из точки старта в точку финиша и вывел `finish` на всех проверочных полигонах — 8 баллов.
    (b) **Реальный робот:** Роботы располагаются в секторах старта, задача определить своё местоположение, распознать arTag метки располо-

женные с секторах сервисного обслуживания, и доехать до финиша.

i. Все роботы определили своё местоположение, остановившись, издали звуковой сигнал и вывели на экран свои координаты в формате $(X;Y)$ и через 10 секунд продолжили движение — 14 баллов.

ii. Один из роботов смог распознать один из arTag маркеров, издал звуковой сигнал, находясь в секторе сервисного обслуживания, вывел на экран номер робота для которого предназначется данный сектор финиша и его координаты в формате $N\ (X, Y)$, где N — номер робота, и через 10 секунд продолжил движение — 10 баллов.

iii. Все arTag маркеры были распознаны согласно предыдущему пункту — 14 баллов.

iv. Один из роботов доехал до сектора финиша и вывел `finish` — 10 баллов.

v. Все роботы достигли соотвествующх секторов финиша и вывели `finish` — 12 баллов.

13. За подзадачу начисляется только половина возможных баллов, если не было засчитано решение подзадачи в симуляторе.
14. Баллы за все попытки в каждой подзадаче суммируются.
15. Выполнение всех критериев в каждой из двух попыток всех трех подзадач дает дополнительные 4 балла.
16. Максимальное количество баллов за этап — 132.

6.7. Решение

```
// Параметры робота
var pi = 3.141592653589793;
var d = 5.6 // Диаметр колеса, см
var l = 17.5 // База, см
var x = 0 // Начальные координаты робота
var y = 0
var a = 0

// Моторы
var mLeft = brick.motor(M4).setPower;
var mRight = brick.motor(M3).setPower;
var cpr = 360 // Показания энкодера за оборот

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

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

// Датчики расстояния
var svFront = brick.sensor(D1).read;
var svLeft = brick.sensor(A1).read;
var svRigth = brick.sensor(A2).read;
```

```
25
26  var readGyro = brick.gyroscope().read
27
28  function readYaw() {
29    return -readGyro()[6];
30  }
31
32  var direction = 0; // absolute angle of direction movement
33  var directionOld = 0;
34  var azimut = 0; // we should go on azimut or turn to it
35  print("-----------------------------------------");
36
37  eLeft.reset();
38  eRight.reset();
39
40  var el = eLeft.readRawData();
41  var er = eRight.readRawData();
42  mLeft(0);
43  mRight(0);
44
45  //инициализация и калибровка гироскопа
46  brick.gyroscope().calibrate(12000);
47  script.wait(13000);
48  print("gyro inited");
49  brick.display().addLabel(30, 10, "Start!");
50  brick.display().redraw();
51  script.wait(1000);
52
53  var v = 50; // velocity
54
55  var ex = 0;
56  var ey = 0;
57  var encLeftOld = 0;
58  var encRightOld = 0;
59  var encLeft = 0;
60  var encRight = 0;
61  var n = 0;
62
63  // вычисление абсолютного угла относительно начального положения
64  function angle() {
65    var sgn = 0;
66    var _direction = readYaw(); // mgrad
67    var dtDirection = _direction - directionOld;
68
69    sgn = directionOld == 0 ? 0 : directionOld / Math.abs(directionOld);
70    n += sgn * Math.floor(Math.abs(dtDirection / 320000));
71    direction = _direction + n * 360000;
72    directionOld = _direction;
73  }
74
75  // делаем прерывание основной программы с частотой 200Гц,
76  // чтобы посчитать абсолютный угол с гироскопа
77  var mtimer = script.timer(50);
78  mtimer.timeout.connect(angle);
79
80  //поворот на угол по гироскопу _angle - относительный угол на который необходимо повернуться
81  function turnDirection(_angle, _v) {
82    _angle = azimut + _angle;
83    azimut = _angle;
84    _angle = _angle * 1000; //toMGrad
```

```
85
86    eLeft.reset();
87    eRight.reset();
88
89    var _vel = _v == undefined & 40: _v; // скорость по умолчанию
90    var angleOfRotate = _angle - direction;
91    var sgn = angleOfRotate == 0 ? 0 : angleOfRotate / Math.abs(angleOfRotate);
92    mLeft(-_vel * sgn);
93    mRight(_vel * sgn);
94    eLeftOld = eLeft.read();
95    eRightOld = eRight.read();
96    target = 211;
97
98    while (Math.abs(eRight.read() - eLeft.read()) / 2 < target) {
99      if ((eLeft.read() - eLeftOld) == 0) && (eRight.read() - eRightOld == 0) {
100       _vel += 5;
101       mLeft(-_vel * sgn);
102       mRight(_vel * sgn);
103     }
104     eLeftOld = eLeft.read();
105     eRightOld = eRight.read();
106     script.wait(20);
107   }
108
109   brick.motor(M3).powerOff(500);
110   brick.motor(M4).powerOff(500);
111   script.wait(500);
112 }
113
114 // проезд в перед на количество ячеек _kceel со скоростью _v
115 // выравниваясь по гироскопу на угол azimut
116 function forward(_v, _kcell) {
117   print("azimut = " + azimut);
118   _alpha = azimut;
119   var _vel = _v == undefined ? 40 : _v; // скорость по умолчанию
120   var u = 0;
121   eLeft.reset();
122   eRight.reset();
123   var el = Math.abs(eLeft.readRawData());
124   while (Math.abs(eLeft.readRawData()) < (el + (_kcell * cellLength))) {
125     u = 1.5 * (_alpha - direction / 1000);
126     mLeft(_vel - u);
127     mRight(_vel + u);
128     script.wait(5);
129   }
130   brick.motor(M3).powerOff();
131   brick.motor(M4).powerOff();
132   script.wait(300);
133 }
134
135 min = function(a, b) {
136   return a < b ? a : b;
137 }
138 max = function(a, b) {
139   return a > b ? a : b;
140 }
141
142 //=============================================================================
143 // массив для изображения
144 var pic = [];
```

```
145
146 var marker_size = 5;
147 var total_width = 320 / 2;
148 var total_height = 240 / 2;
149 var prob = 25 * 5 * 5;
150
151 function getColor(pic, x, y) {
152   return pic[y * total_width + x];
153 }
154
155 function squareAverage(pic, x, y, diam) {
156   var sum = 0;
157   var start_row = y * total_width;
158   var end_row = start_row + diam * total_width;
159
160   for (var index = start_row + x; index < end_row; index += total_width)
161     for (var j = 0; j < diam; j += 1)
162       sum += pic[index + j];
163
164   return sum;
165 }
166
167 // lu - left-up corner. Coordinates: (x, y)
168 // ld - left-down corner
169 // ru - right-up corner
170 // rd - right-down corner
171 function getCenterColor(pic, lu, ld, ru, rd, diam) {
172   var x = (lu[0] + ld[0] + ru[0] + rd[0]) >> 2;
173   var y = (lu[1] + ld[1] + ru[1] + rd[1]) >> 2;
174   var color = squareAverage(pic, x - diam, y - diam, diam << 1);
175   return color;
176 }
177
178 function findGridCorners(corners, marker_size) {
179   var grid_corners = [];
180
181   var vertical_lines = [];
182   var upper_line_x1 = corners[0][0];
183   var upper_line_y1 = corners[0][1];
184   var upper_line_x2 = corners[2][0];
185   var upper_line_y2 = corners[2][1];
186   var down_line_x1 = corners[1][0];
187   var down_line_y1 = corners[1][1];
188   var down_line_x2 = corners[3][0];
189   var down_line_y2 = corners[3][1];
190
191   var mks = 1.0 / marker_size;
192   var k_ux = (upper_line_x2 - upper_line_x1) * mks;
193   var k_uy = (upper_line_y2 - upper_line_y1) * mks;
194   var k_dx = (down_line_x2 - down_line_x1) * mks;
195   var k_dy = (down_line_y2 - down_line_y1) * mks;
196
197   for (var i = 0; i < marker_size + 1; i += 1) {
198
199     var up_x = upper_line_x1 + k_ux * i;
200     var up_y = upper_line_y1 + k_uy * i;
201
202     var down_x = down_line_x1 + k_dx * i;
203     var down_y = down_line_y1 + k_dy * i;
204
```

```
    var k_x = (down_x - up_x) * mks;
    var k_y = (down_y - up_y) * mks;

    for (j = 0; j <= marker_size; j += 1) {

      var point_x = up_x + k_x * j;
      var point_y = up_y + k_y * j;

      grid_corners.push([Math.floor(point_x), Math.floor(point_y)]);
    }
  }
  return grid_corners;
}

function detectCode(pic, grid_corners, diam) {
  var calculated_colors = []
  var markerSizePlusOne = marker_size + 1;
  var shiftedDiam = diam << 8;

  for (var i = 0; i < marker_size; i += 1) {
    for (var j = 0; j < marker_size; j += 1) {
      lu_index = i * (markerSizePlusOne) + j;
      ld_index = i * (markerSizePlusOne) + j + 1;
      ru_index = (i + 1) * (markerSizePlusOne) + j;
      rd_index = (i + 1) * (markerSizePlusOne) + j + 1;

      var lu = grid_corners[lu_index];
      var ld = grid_corners[ld_index];
      var ru = grid_corners[ru_index];
      var rd = grid_corners[rd_index];

      grid_color = getCenterColor(pic, lu, ld, ru, rd, diam);
      if (grid_color < shiftedDiam) {
        calculated_colors.push(0);
      } else {
        calculated_colors.push(1);
      }
    }
  }
  return calculated_colors;
}

function findULCorner(pic, diam) {
  var color = 1;
  for (var i = 0; i < total_height; i += 1) {
    for (var j = 0; j <= i; j += 1) {
      var x = j;
      var y = i - j;
      if (getColor(pic, x, y) == 0) {
        color = squareAverage(pic, x, y, diam);
        if (color < prob) {
          return [x, y];
        }
      }
    }
  }
}

function findDLCorner(pic, diam) {
  var color = 1;
```

```
  for (var i = 0; i < total_height; i += 1) {
    for (var j = 0; j <= i; j += 1) {
      var x = j;
      var y = total_height - (i - j);
      if (getColor(pic, x, y) == 0) {
        color = squareAverage(pic, x, y - diam + 1, diam);
        if (color < prob) {
          return [x, y];
        }
      }
    }
  }
}

function findURCorner(pic, diam) {
  for (var i = 0; i < total_height; i += 1) {
    for (var j = 0; j <= i; j += 1) {
      var x = total_width - j;
      var y = i - j;
      if (getColor(pic, x, y) == 0) {
        var color = squareAverage(pic, x - diam + 1, y, diam);
        if (color < prob) {
          return [x, y];
        }
      }
    }
  }
}

function findDRCorner(pic, diam) {
  for (var i = 0; i < total_height; i += 1) {
    for (var j = 0; j <= i; j += 1) {
      var x = total_width - j;
      var y = total_height - (i - j);
      if (getColor(pic, x, y) == 0) {
        var color = squareAverage(pic, x - diam + 1, y - diam + 1, diam);
        if (color < prob) {
          return [x, y];
        }
      }
    }
  }
}

function findCorners(pic, diam) {
  return [findULCorner(pic, diam), findDLCorner(pic, diam), findURCorner(pic,
    diam), findDRCorner(pic, diam)];
}

function threshold2(level, pic, height, width) {
  var length = pic.length;
  for (var i = 0; i < length; i += 1) {
    var color = pic[i];
    if (color < level) {
      pic[i] = 0;
    } else {
      pic[i] = 255;
    }
  }

```

```
325   return pic;
326 }
327
328 // ------------------------------------------------------------------------------
329 var scale = 1;
330 var histogram = [];
331 var histSize = 256;
332
333 function calculateHistogram() {
334   for (var i = 0; i < histSize; i += 1)
335     histogram[i] = 0;
336
337   var curPixelLine = 0;
338   for (var i = 0; i < total_height; i += 1) {
339     curPixelLine = i * total_width;
340     for (var j = 0; j < total_width; j += 1)
341       histogram[Math.floor(pic[curPixelLine + j])] += 1;
342   }
343 }
344
345 // binarization using 2 elems in grayscale
346 var grayscale = "@#ao|-. ";
347 var numOfBins = grayscale.length;
348 var rangeBins = [];
349 var binCapacity = total_height * total_width / numOfBins;
350
351 function getRange() {
352   for (var i = 0; i < numOfBins; i += 1)
353     rangeBins[i] = 0;
354
355   var curBin = 0;
356   var curSum = 0;
357   var i = 0;
358   var lastIndexBin = numOfBins - 1;
359
360   for (;
361     (i < histSize) && (curBin < lastIndexBin); i += 1) {
362     var diff = binCapacity - curSum;
363
364     if (Math.abs(diff) < Math.abs(diff - histogram[i])) {
365       curBin++;
366       curSum = 0;
367     }
368
369     curSum += histogram[i];
370     rangeBins[curBin] = i;
371   }
372
373   for (; curBin <= lastIndexBin; curBin += 1)
374     rangeBins[curBin] = histSize;
375 }
376
377 var mapColorToLetter = [];
378
379 function initMapColorToLetter() {
380   var curBin = 0;
381   for (var i = 0; i < histSize; i += 1) {
382     if (rangeBins[curBin] <= i) {
383       curBin += 1;
384     }
```

```
385       mapColorToLetter[i] = grayscale[curBin];
386     }
387   }
388
389   // Возвращает значение ARTag
390   function getARTagValue() {
391     source_pic = getPhoto();
392
393     // init pic, grayscale mode
394     for (var i = 0; i < total_height; i += 1) {
395       for (var j = 0; j < total_width; j += 1) {
396         var x = (j + i * scale * total_width) * scale;
397         var p = source_pic[x];
398         p = (((p & 0xff0000) >> 18) + ((p & 0xff00) >> 10) + ((p & 0xff) >> 2));
399         pic[j + i * total_width] = p;
400       }
401     }
402     calculateHistogram();
403     getRange();
404     initMapColorToLetter();
405
406     var thresh = threshold2(rangeBins[1], pic, total_height, total_width);
407     var corners = findCorners(thresh, 9);
408     var grid_corners = findGridCorners(corners, marker_size)
409     var values = detectCode(thresh, grid_corners, 3);
410
411     var ans = 0;
412     if (values[1][1] == 0)
413       ans = 8 * values[3][2] + 4 * values[2][3] + 2 * values[2][1] + values[1][2];
414     else if (values[1][3] == 0)
415       ans = 8 * values[2][1] + 4 * values[3][2] + 2 * values[1][2] + values[2][3];
416     else if (values[3][3] == 0)
417       ans = 8 * values[1][2] + 4 * values[2][1] + 2 * values[2][3] + values[3][2];
418     else if (values[3][1] == 0)
419       ans = 8 * values[2][3] + 4 * values[2][3] + 2 * values[3][2] + values[2][1];
420     else
421       print("Error: Incorrect ARTag");
422     return ans;
423   };
424   //=====================================================================
425
426   // Местонахождение робота
427   var positionOfRobot = 0;
428   var directionOfRobot = 0;
429
430   // карта
431   lab = [
432     [-1, 1, 8, -1],
433     [-1, 2, -1, 0],
434     [-1, 3, 10, 1],
435     [-1, 4, -1, 2],
436     [-1, 5, -1, 3],
437     [-1, -1, 13, 4],
438     [-1, -1, -1, -1],
439     [-1, -1, 15, -1],
440     [0, -1, 16, -1],
441     [-1, -1, -1, -1],
442     [2, -1, 18, -1],
443     [-1, -1, 19, -1],
444     [-1, -1, -1, -1],
```

```
  [5, 14, 21, -1],
  [-1, 15, -1, 13],
  [7, -1, -1, 14],
  [8, 17, -1, -1],
  [-1, 18, -1, 16],
  [10, -1, 26, 17],
  [11, 20, 27, -1],
  [-1, 21, -1, 19],
  [13, -1, 29, 20],
  [-1, -1, -1, -1],
  [-1, -1, 31, -1],
  [-1, -1, 32, -1],
  [-1, -1, -1, -1],
  [18, -1, 34, -1],
  [19, -1, -1, -1],
  [-1, -1, -1, -1],
  [21, 30, 37, -1],
  [-1, 31, -1, 29],
  [23, -1, 39, 30],
  [24, 33, 40, -1],
  [-1, 34, -1, 32],
  [26, 35, 42, 33],
  [-1, 36, -1, 34],
  [-1, 37, 44, 35],
  [29, -1, 45, 36],
  [-1, -1, -1, -1],
  [31, -1, -1, -1],
  [32, -1, -1, -1],
  [-1, -1, -1, -1],
  [34, -1, 50, -1],
  [-1, -1, -1, -1],
  [36, 45, -1, -1],
  [37, 46, 53, 44],
  [-1, 47, -1, 45],
  [-1, -1, 55, 46],
  [-1, 49, 56, -1],
  [-1, 50, -1, 48],
  [42, 51, 58, 49],
  [-1, -1, 59, 50],
  [-1, -1, -1, -1],
  [45, -1, -1, -1],
  [-1, -1, -1, -1],
  [47, -1, 63, -1],
  [48, -1, -1, -1],
  [-1, -1, -1, -1],
  [50, 59, -1, -1],
  [51, 60, -1, 58],
  [-1, 61, -1, 59],
  [-1, 62, -1, 60],
  [-1, 63, -1, 61],
  [55, -1, -1, 62]
];

// double cycle for traveling in maze
cycle = [0, 2, 4, 13, 15, 21, 19, 29, 31, 37, 45, 46, 55, 62, 60, 50, 48, 34,
  32, 18, 16, 0, 2, 4, 13, 15, 21, 19, 29, 31, 37, 45, 46, 55, 62, 60, 50, 48,
  34, 32, 18, 16
];

var foundedDestroyedSectors = 0;
```

```
505 var destroyedSectors = 0;
506
507 // получаем маршрут перемещения до необходимой точки из текущей
508 // в нотации F, L, R
509 function getPath(finishSector) {
510   from = [];
511   for (var i = 0; i < 64; i++) {
512     from[i] = [];
513     for (var j = 0; j < 4; j++)
514       from[i][j] = "N";
515
516   }
517
518   var queue = [];
519   queue.push([positionOfRobot, directionOfRobot]);
520   var finishDir = 0;
521   while (queue.length > 0) {
522     temp = queue.shift();
523     currentSector = temp[0], currentDir = temp[1];
524     if (currentSector == finishSector) {
525       finishDir = currentDir;
526       break;
527     }
528     // Вперед
529     if (lab[currentSector][currentDir] > -1) {
530       adjSector = lab[currentSector][currentDir];
531       adjDir = currentDir;
532       if (from[adjSector][adjDir] == "N") {
533         from[adjSector][adjDir] = "F";
534         queue.push([adjSector, adjDir]);
535       }
536     }
537     // Направо
538     if (from[currentSector][(currentDir + 1) % 4] == "N") {
539       adjSector = currentSector;
540       adjDir = (currentDir + 1) % 4;
541       from[adjSector][adjDir] = "R";
542       queue.push([adjSector, adjDir]);
543     }
544     // Налево
545     if (from[currentSector][(currentDir + 3) % 4] == "N") {
546       adjSector = currentSector;
547       adjDir = (currentDir + 3) % 4;
548       from[adjSector][adjDir] = "L";
549       queue.push([adjSector, adjDir]);
550     }
551   }
552
553   path = "";
554   // Восстанавливаем путь
555   if (from[finishSector][finishDir] == "N")
556     print("No way");
557   else {
558     currentSector = finishSector;
559     currentDir = finishDir;
560     while (currentSector != positionOfRobot || currentDir != directionOfRobot) {
561       action = from[currentSector][currentDir];
562       if (action == "F") {
563         path = "F" + path;
564         currentSector = lab[currentSector][(currentDir + 2) % 4];
```

```
565       } else if (action == "R") {
566         path = "R" + path;
567         currentDir = (currentDir + 3) % 4;
568       } else if (action == "L") {
569         path = "L" + path;
570         currentDir = (currentDir + 1) % 4;
571       }
572     }
573   }
574   return path;
575 }
576
577 // Внести информацию о недоступности сектора
578 function isolateSector(sector) {
579   foundedDestroyedSectors++;
580   for (dir = 0; dir < 4; dir++)
581     if (lab[sector][dir] > -1) {
582       lab[lab[sector][dir]][(dir + 2) % 4] = -2;
583       lab[sector][dir] = -2;
584     }
585   calcAvailable();
586   print("countUnavailable=", countUnavailable);
587 }
588
589 // Проверка окружающих секторов
590 function checkAdjacentSectors() {
591   if (!(svFront() > 25) &&
592         lab[positionOfRobot][directionOfRobot] > -1)
593     isolateSector(lab[positionOfRobot][directionOfRobot]);
594   if (!(svLeft() > 25) &&
595         lab[positionOfRobot][(directionOfRobot + 3) % 4] > -1)
596     isolateSector(lab[positionOfRobot][(directionOfRobot + 3) % 4]);
597   if (!(svRigth() > 25) &&
598         lab[positionOfRobot][(directionOfRobot + 1) % 4] > - 1)
599     isolateSector(lab[positionOfRobot][(directionOfRobot + 1) % 4]);
600 }
601
602 // Перемещение по кратчайшему пути до finishSector
603 function follow_path(finishSector) {
604   path = getPath(finishSector);
605   while (positionOfRobot != finishSector && path != "") {
606     if (foundedDestroyedSectors < destroyedSectors)
607       checkAdjacentSectors();
608     for (var i = 0; i < path.length; i++) {
609       if (path[i] == "R") {
610         turnDirection(-90, v);
611         directionOfRobot = (directionOfRobot + 1) % 4;
612       } else if (path[i] == "L") {
613         turnDirection(90, v);
614         directionOfRobot = (directionOfRobot + 3) % 4;
615       } else if (path[i] == "F") {
616         if (lab[positionOfRobot][directionOfRobot] < 0) {
617           print("The path is blocked. No way!");
618           break;
619         }
620         forward(v, 1);
621         positionOfRobot = lab[positionOfRobot][directionOfRobot];
622       }
623       if (foundedDestroyedSectors < destroyedSectors)
624         checkAdjacentSectors();
```

```
625         else if (foundedDestroyedSectors == destroyedSectors)
626           break;
627       }
628       if (foundedDestroyedSectors == destroyedSectors) {
629         foundedDestroyedSectors++;
630         break;
631       }
632       path = getPath(finishSector);
633     }
634   }
635
636   // Поворот робота на заданное направление
637   function turnTo(targetDir) {
638     var dDir = (targetDir - directionOfRobot + 4) % 4;
639     if (dDir == 1) {
640       turnDirection(-90, v);
641     } else if (dDir == 3) {
642       turnDirection(90, v);
643     } else if (dDir == 2) {
644       turnDirection(-180, v);
645     }
646     directionOfRobot = targetDir;
647   }
648
649   // Проход по лабиринту
650   function travelThroughoutMaze() {
651     var firstSector = 0;
652     for (firstSector = 0; firstSector < cycle.length / 2; ++firstSector) {
653       if (positionOfRobot == cycle[firstSector] || lab[positionOfRobot][0] ==
654         cycle[firstSector] || lab[positionOfRobot][1] == cycle[firstSector] ||
655         lab[positionOfRobot][2] == cycle[firstSector] ||
656         lab[positionOfRobot][3] == cycle[firstSector])
657         break;
658     }
659     for (i = firstSector; i < firstSector + cycle.length / 2; i++) {
660       follow_path(cycle[i]);
661       if (foundedDestroyedSectors >= destroyedSectors) {
662         break;
663       }
664     }
665   }
666
667   function dfs(sector) {
668     used[sector] = true;
669     countAvailable++;
670     for (var dir = 0; dir < 4; ++dir) {
671       nextSector = lab[sector][dir];
672       if (nextSector > -1 && !used[nextSector])
673         dfs(nextSector);
674     }
675   }
676
677   // Вычисление доступных/недоступных секторов
678   function calcAvailable() {
679     used = [];
680     for (var i = 0; i < 64; i++)
681       used = [];
682     // Сколько доступных секторов
683     countAvailable = 0;
684     dfs(positionOfRobot);
```

```
685   // Сколько недоступных секторов
686   countUnavailable = 64 - countAvailable - 12;
687 }
688
689 var value1, value2;
690 // Считывание двух ARTag маркеров
691 function readARTag(dist) {
692   forward(-v, dist);
693   value1 = getARTagValue();
694
695   forward(v, dist);
696   value2 = getARTagValue();
697   print(value1 + " " + value2);
698 }
699
700 //=============================================================================
701 //=================================M A I N =====================================
702 var main = function() {
703
704   var xStart = 7;
705   var yStart = 1;
706   directionOfRobot = 3;
707   var xARTag = 5;
708   var yARTag = 6;
709   // С какой стороны от сектора распредления решений располагается ARTag
710   var directionOfARTag = 3;
711
712   // Количество обрушенных секций
713   destroyedSectors = 2;
714   positionOfRobot = xStart + yStart * 8;
715
716   travelThroughoutMaze();
717   follow_path(xARTag + yARTag * 8);
718   turnTo((directionOfARTag + 3) % 4);
719
720   //detecting ARTag markers until it isn't successful
721   value1 = 0;
722   value2 = 0;
723   dist = 0.3;
724   while (value1 < 8 && value2 < 8 || value1 > 7 && value2 > 7) {
725     readARTag(dist);
726     dist += 0.05;
727   }
728
729   // Координаты финиша
730   xFinish = 0;
731   yFinish = 0;
732   if (value1 < 8) {
733     xFinish = value1;
734     yFinish = value2 - 8;
735   } else {
736     xFinish = value2;
737     yFinish = value1 - 8;
738   }
739
740   print(xFinish + " " + yFinish);
741   finishSector = xFinish + yFinish * 8;
742   follow_path(finishSector);
743   print(countUnavailable);
744   return;
```