



НТО ЮНИОРЫ 2026/27 · СФЕРА

Роботех

Задачник-справочник для подготовки к
отборочному этапу

5-7 класс



Автор-составитель

Артём Осинowskiй, методист и разработчик детского технопарка «Кванториум»

Детский технопарк «Кванториум», Новосибирск · 2026

Роботех

Задачник-справочник для подготовки к отборочному этапу НТО Юниоры. 5–7 класс

Автор-составитель – **Артём Осинowski**, методист и разработчик детского технопарка «Кванториум».

Нашли ошибку или неточность? Напишите на osinovsky.aa@kvantorium54.ru (рабочая почта): сфера, версия книги (1.0) и номер страницы – исправим в следующей версии. Актуальная версия и список изменений – nto.kvantorium54.ru/knigi. Личная почта для других вопросов – artem081033@gmail.com.

Важно. Это не официальное пособие олимпиады, а книга методиста «Кванториума». Каких заданий ждать на отборе 2026/27, мы не знаем: организаторы их заранее не показывают, а задания разных лет не повторяются. Книга готовит к темам из программы сферы и приёмам решения, а не к конкретным задачам.

Книга помогает школьнику самостоятельно подготовиться к отборочному этапу сферы «Роботех» Национальной технологической олимпиады «НТО Юниоры»: теория с нуля, подробные разборы задач и задачи для самопроверки с ответами.

Большинство задач составлены автором по образовательной программе сферы, часть – по мотивам открытых заданий прошлых лет; источник указан у каждой задачи. Уровни сложности – шкала этой книги, а не баллы олимпиады. Правила, сроки и формат отбора – на сайте junior.ntcontest.ru.

Версия 1.0 от 02.10.2026, сборка 02.10.2026. © Артём Осинowski, 2026.

Содержание

Как устроена эта книга	4
Для кого эта книга	4
Как устроен отбор	4
Что пригодится на отборе и где это в книге	5
Как пользоваться книгой	6
Как вводить ответы на отборе	7
Что понадобится	7
План на месяц до отбора	28
Родителям и наставникам	28
Честно об этой книге	29
Глава 1 Как устроен робот	30
1.1. Что такое робот и какие они бывают	31
1.2. Из чего состоит мобильный робот	33
1.3. Как робот едет и поворачивает	36
1.4. Зубчатая передача и передаточное число	37
1.5. Рычаг	39
1.6. Разбираем задачи	40
Глава 2 Движение: скорость, путь, время	50
2.1. Скорость, путь и время	51
2.2. Единицы скорости и их перевод	53
2.3. Колесо: путь за один оборот	54
2.4. График «путь – время»	55
2.5. Разгон, торможение и инерция	57
2.6. Трение: почему колёса буксуют	59
2.7. Разбираем задачи	60
Глава 3 Координаты и углы: навигация робота	70
3.1. Координаты на поле	71
3.2. Направление и углы поворота	72
3.3. Программа движения: где окажется робот	75
3.4. Масштаб: план и настоящий полигон	77
3.5. Расстояние и маршрут: по клеткам и по прямой	78
3.6. Поворот на колёсах: сколько крутить колесо	79
3.7. Разбираем задачи	80
Глава 4 Алгоритмы и исполнители	90
4.1. Исполнитель и его команды	91
4.2. Циклы: «повтори» и «пока»	92
4.3. Ветвление: «если – иначе»	94
4.4. Подпрограммы	95
4.5. Лабиринт и правило правой руки	96
4.6. Трассировка и поиск ошибок	98
4.7. Когда робот ходит по кругу	100

4.8. Разбираем задачи	101
Глава 5 Моторы, датчики и электричество	117
5.1. Цепь: ток, напряжение, сопротивление	118
5.2. Закон Ома и резистор для светодиода	119
5.3. Мотор и драйвер: почему мотор нельзя прямо к плате	121
5.4. ШИМ: как ехать не на полной скорости	124
5.5. Батарея: на сколько хватит заряда	125
5.6. analogRead: как контроллер измеряет напряжение	126
5.7. Датчик линии: белое и чёрное	128
5.8. Дальномёры: ультразвук и инфракрасный свет	129
5.9. Разбираем задачи	133
Глава 6 Программирование: Arduino и машины состояний	141
6.1. Переменные и монитор порта	141
6.2. Условия и циклы на C++	144
6.3. Функции: свои команды для робота	146
6.4. Машина состояний	149
6.5. Массивы: робот запоминает путь	151
6.6. Разбираем задачи	154
Глава 7 Движение по линии и регуляторы	167
7.1. Как датчик видит линию	168
7.2. Калибровка: где провести границу	169
7.3. Релейный регулятор: зигзаг по двум датчикам	172
7.4. Пропорциональный регулятор	174
7.5. Почему робот виляет	177
7.6. ПИД-регулятор: идея	178
7.7. Разбираем задачи	180
Глава 8 Сигналы и обработка данных	189
8.1. Сигнал: рассказ датчика во времени	190
8.2. Дискретизация: контроллер видит точки	192
8.3. Шум и выбросы	194
8.4. Сглаживание: скользящее среднее и медиана	195
8.5. Детектирование: по уровню и по периоду	198
8.6. Спектр и огибающая – на пальцах	201
8.7. Разбираем задачи	204
Ответы и решения	214
Справочник	222
Числа, которые используются в книге	222
Перевод единиц	222
Формулы и правила	223
Мини-шпаргалки по инструментам	231
Словарик	237
Шпаргалка: всё «Запомни» из глав	241
Где что найти	247

Как устроена эта книга

Эта книга – тренажёр для отборочного этапа сферы «Роботех» олимпиады «НТО Юниоры». В ней теория с нуля, подробные разборы задач и сразу после каждого разбора – похожие задачи «Реши сам» с ответами в конце книги. Свой робот и набор деталей не нужны. Скорость, повороты и маршруты считаются в тетради с калькулятором, а схемы и программы собираются в браузере, в том же симуляторе, что и на курсе сферы.

Для кого эта книга

- Для школьников **5–7 классов**, которые выбрали (или думают выбрать) сферу «Роботех».
- Для тех, кто **не проходил физику и ни разу не программировал**. Скорость, ток, напряжение, мотор и датчик объясняем с самого начала, а первую программу разбираем построчно.
- Для тех, у кого **нет кружка и конструктора**. В первом полугодии, которое как раз готовит к отбору и финалу, курс сферы идёт целиком в онлайн-симуляторах, а конструктор нужен только во втором. Книга устроена так же.
- Для **наставников и родителей**, которые хотят помочь. Для них есть отдельный раздел в конце введения.

Как устроен отбор

Таблица В.1

Когда	Что происходит
с 1 октября	демо-этап с пробными заданиями (по сообщению организаторов)
до 9 ноября	регистрация на олимпиаду (наш совет – успеть до 20 октября, тогда будут обе попытки)
20–30 октября	отбор, попытка 1 (3 часа)
31 октября – 10 ноября	отбор, попытка 2 (засчитывают лучшую из двух)
7–17 декабря	финалы в регионах

- Отбор проходит **дома, онлайн, с компьютера**. Задания открываются в личном кабинете на my.ntcontest.ru и решаются на платформе «Орбита» (orbita.education). Каждый решает сам за себя. Три часа идут подряд, остановить таймер нельзя.
- Задания – это задачи с ответом, который проверяет компьютер: ввести число, выбрать верные варианты, сопоставить, расставить команды по порядку, написать короткий код.

- Разработчик сферы на вебинарах августа и сентября 2026 года рассказал, из чего будет состоять отбор: задачи теоретические и задачи-практикумы в онлайн-симуляторе Tinkercad, где собирают схему и пишут программу. Практику из курса он советовал проходить руками, а не только читать. Поэтому в главах 5–8 много схем и программ, которые стоит повторить в симуляторе.
- Книга построена по программе сферы «Роботех. Основы мобильной робототехники» 2026/27. В самой программе есть отдельный модуль «Разбор задач отборочного онлайн-этапа» – значит, задачи отбора растут из тех же тем, что и курс.
- За тест бесплатного курса своей сферы дают **+5 баллов**. Курс – на academy.junior.online, вход через «Талант». Бонус нужно включить самому в личном кабинете («Мои привилегии») до конца отбора. Учти: после этого сферу уже не поменять. И бонус засчитают, только если за задания отбора ты набрал больше нуля.
- Тесты внутри курса – ещё не отбор. По словам разработчика, они несложные и нужны, чтобы ты видел, как продвигаешься по курсу и правильно ли понял модуль.
- Финал – **командный** и очный: участники приезжают на площадку в своём регионе, там их делят на команды по три человека, и команды решают задачи за компьютерами. Задачи снова теоретические и в симуляторе: спроектировать робота, ввести ему координаты, обработать и отфильтровать данные с датчиков, придумать алгоритм движения. Проходной балл в финал заранее не известен: его объявляют ближе к концу отбора.
- В некоторых регионах после финала отдельно проводят хакатон с настоящим конструктором «Роботех Юниор» на платформе Arduino. На баллы финала он не влияет, а программы там пишут те же, что в симуляторе.

Совет. Сроки и правила могут уточняться. Главный источник – сайт олимпиады junior.ntcontest.ru и памятка для Новосибирской области nto.kvantorium54.ru. Свежие новости и ответы разработчика – в публичном канале сферы «Роботех» в мессенджере MAX: он находится поиском по словам «сфера Роботех».

Что пригодится на отборе и где это в книге

На странице сферы перечислено, что пригодится на отборе, по трём предметам. Вот где это в книге:

Таблица В.2

Что пригодится на отборе	Глава
Физика: механика и механизмы	1. Как устроен робот
Математика: расстояние, время, скорость; величины и единицы измерения	2. Движение: скорость, путь, время
Физика: параметры движения, сила трения, инерция	2
Математика: координаты, углы и фигуры, пропорции и масштаб	3. Координаты и углы: навигация робота
Информатика: алгоритмы и исполнители	4. Алгоритмы и исполнители

Что пригодится на отборе	Глава
Физика: электричество и электрические цепи, звук	5. Моторы, датчики и электричество
Информатика: язык и среда программирования, переменные, условия, циклы, подпрограммы	6. Программирование: Arduino и машины состояний
Движение по линии, П- и ПИД-регулятор (из программы сферы)	7. Движение по линии и регуляторы
Математика: графики; обработка сигналов с датчиков (из программы сферы)	8. Сигналы и обработка данных

Таблицы встречаются по всей книге, а мини-шпаргалка по табличному редактору – в справочнике.

Начни с главы 1: в ней видно, из чего собран робот, и дальше книга всё время на это опирается. Главы 2 и 3 – про математику движения: их хватит тетради, карандаша и калькулятора. Третью читай после второй, потому что повороты мы считаем через путь колеса. Главу 4 можно открыть когда угодно. Главу 5 прочитай раньше главы 6: сначала понятно, как мотор и датчик подключены, потом – как ими командовать из программы. Главы 7 и 8 – после шестой: в них датчики, моторы и программа работают вместе.

Как пользоваться книгой

В каждой главе теория и разборы идут вперемешку: за каждым разбором сразу стоят две похожие задачи «Реши сам», чтобы закрепить приём, пока он свежий в голове.

- 1. Теория.** Коротко и по делу, с рисунками. Каждая новая тема начинается с большой картинкой «на пальцах», а самое нужное собрано в рамках «Запомни». Все «Запомни» из всех глав повторяются в конце книги, в шпаргалке.
- 2. Разбираем задачи.** К каждой задаче – решение по шагам: что дано, что делаем и почему, где часто ошибаются, какой ответ получился и можно ли ему верить.
- 3. Реши сам.** Сразу после разбора – две задачи на тот же приём, но с другими числами и другим сюжетом.
- 4. Ответы** – в конце книги, в разделе «Ответы и решения». Там только ответ и иногда короткая подсказка, если задача с подвохом. Полного решения нет специально: если подсказки не хватило, вернись к разбору перед задачей.

Не готов читать всю главу целиком? Есть отдельная короткая книжка «Быстрый старт»: из каждой главы самое нужное, один разбор и несколько задач «Реши сам». Её удобно пройти за неделю, а эту книгу держать под рукой.

У каждой задачи есть уровень – как в обычном задачнике: у трудных задач звёздочка рядом с номером («задача 3*»), в них нужно несколько идей сразу. Лёгкие задачи, в одно действие, подписаны словом «разминка». Это уровни нашей книги, а не баллы олимпиады. Сколько задач будет на отборе и сколько стоит каждая, заранее не объявляют.

Совет. Разбор читай с листком бумаги. Закрой решение, попробуй решить сам, а потом сверяйся по шагам. Если в разборе есть схема или программа, собери её в симуляторе и запусти. Робот, который у тебя на экране поехал не туда, научит большему, чем три страницы правильного текста.

У каждой задачи внизу написано, откуда она взялась. «Своя задача по программе сферы» – мы придумали её по теме из программы 2026/27 года; таких в книге большинство. «По мотивам задачи» – идею мы взяли из задач прошлых лет или из задач старшей олимпиады для 8–11 классов по профилю «Интеллектуальные робототехнические системы», упростили, пересказали своими словами и поменяли числа.

Как вводить ответы на отборе

Так было на отборочных этапах прошлых лет в других сферах. На «Орбите» перед каждой задачей читай инструкцию: там сказано, в каком виде нужен ответ.

- **Только число**, без единиц измерения и без знака градуса. Смотри, в чём просят ответ: расстояние – в сантиметрах или метрах, время – в секундах, миллисекундах или микросекундах, угол – в градусах.
- **Дробную часть** на сайте обычно отделяют **точкой**: `0.75`, а не `0,75`. В книге мы пишем по-русски, через запятую.
- **Несколько верных вариантов** часто записывают строкой из их номеров подряд, без пробелов и запятых: выбрал варианты 1, 3 и 4 – вводишь `134`. Точный формат – в инструкции к задаче.
- **Округляй так, как сказано в условии**. «До десятых» – один знак после точки. Промежуточные результаты не округляй – только ответ.
- **Считай, что можно сосчитать поштучно**. Перекрёстки, шаги исполнителя, показания `analogRead` – целые числа. Если робот проехал 7,4 перекрёстка, ошибка где-то раньше. А вот обороты колеса бывают и дробными: половина оборота – обычный ответ.
- **У каждого могут быть свои числа**. В некоторых задачах условие генерируется для каждого участника отдельно, так что списать ответ не выйдет.
- **Попытки ограничены**. На задачу даётся от 3 до 5 вводов, и каждая неверная попытка снимает 30 % балла за эту задачу. Сначала реши на бумаге или проверь программу в симуляторе, и только потом вводи ответ.
- **«Сохранить» – ещё не отправка**. Ответ засчитан, когда нажата кнопка «Отправить».

Что понадобится

- Тетрадь в клетку, карандаш, линейка и транспортир – маршруты, координаты и углы удобно рисовать прямо по клеткам.
- Калькулятор. Хватит обычного на телефоне, но пусть в нём будут кнопки π и корня.
- Компьютер с интернетом и мышью – для глав 5–8. Планшета мало: схему собирают, протягивая провода мышкой.
- **Tinkercad** – онлайн-симулятор, в котором работают курс, отбор и финал сферы. Раздел «Цепи» (по-английски Circuits): собираешь схему из платы Arduino Uno, моторов, датчи-

ков и проводов, пишешь программу и смотришь, как всё работает. Личная учётная запись не нужна: в курсе сферы есть ссылка на класс «Роботех» и пошаговая инструкция, как в него войти. Заходи именно по ссылке на класс: личные учётные записи на вебинаре называли ограниченными.

- **Wokwi** (wokwi.com) – запасной симулятор Arduino в браузере; если у тебя он не открывается, хватит Tinkercad. На бесплатной учётной записи запуск иногда ждёт в очереди.
- **Кибериада IDE** (Cyberiada IDE) – редактор, где программу не пишут строчками, а рисуют схемой из состояний и переходов (язык ПРИМС, глава 6). Ссылка на неё – в первых модулях курса.
- **Arduino IDE** – по желанию. Разработчик сферы сказал, что для курса она не обязательна: код можно писать прямо в симуляторе. Но в ней удобно набирать программу с подсветкой, а потом копировать в Tinkercad. Для подготовки к отбору хватит симулятора; программу для настоящей платы поможет поставить педагог кружка.
- **Таблицы** – Excel, LibreOffice Calc или любые онлайн-таблицы: для задач, где данные с датчиков нужно посчитать и построить по ним график.

Эти же инструменты – онлайн-симулятор, Кибериада IDE, Arduino IDE и таблицы – названы в официальных требованиях к площадкам финала сферы «Роботех».

Tinkercad: вход в класс и первая схема

Tinkercad (tinkercad.com) – бесплатный симулятор электронных схем и Arduino прямо в браузере: ставить ничего не нужно. В нём идёт практика курса сферы. Регистрироваться тоже не надо – курс даёт ссылку на класс и общее имя ученика. Ниже весь путь по шагам, рамка с номером на картинке – куда нажимать.

Шаг 1. Войти в класс. Открой ссылку на класс курса: tinkercad.com/joinclass/YMRYDIZP (класс «Роботех»; та же ссылка – в курсе сферы, модуль 5 «Теория и практика по мобильной робототехнике. Часть 1»). Нажми «Присоединиться, используя имя учащегося» (1 на левой половине рисунка). Впиши имя учащегося `test` (1 на правой половине) и нажми «Перейти в класс» (2).

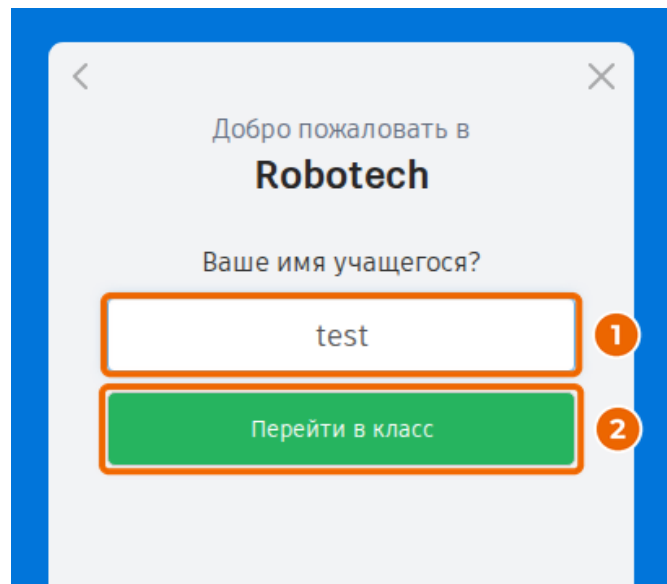
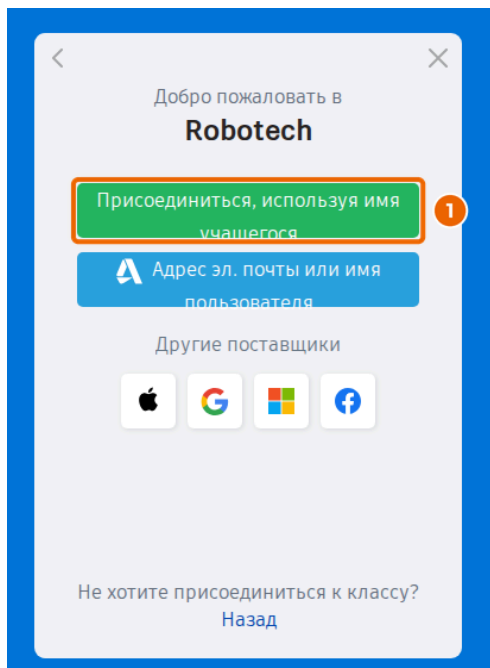


Рисунок В.1 – Вход в класс Tinkercad: слева – выбор способа входа, справа – имя учащегося

Шаг 2. Новая схема. Вверху справа нажми «Создать» (1), в меню – «Цепи» (2). Откроется редактор схем.

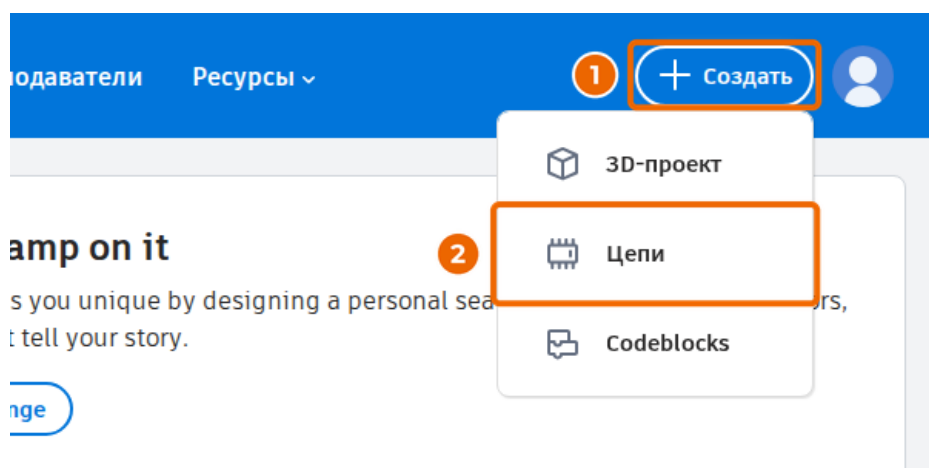


Рисунок В.2 – Новая схема: «Создать» → «Цепи»

Шаг 3. Окно редактора. Вот что в нём где.

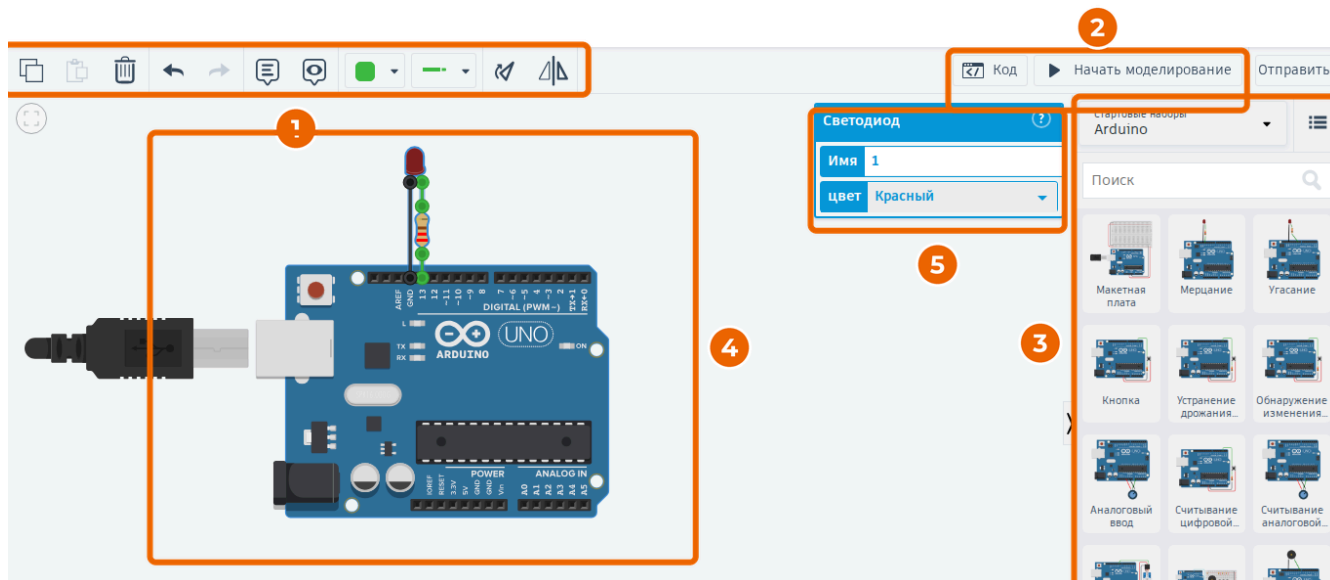


Рисунок В.3 – Редактор схем Tinkercad

1. **Инструменты:** копировать, вставить, удалить, отменить и вернуть действие, повернуть деталь.
2. **«Код»** – программа для Arduino; **«Начать моделирование»** – запуск схемы.
3. **Компоненты:** сверху выбор набора, ниже поиск и сами детали. Деталь перетаскивают мышью на поле.
4. **Рабочее поле** – здесь собирают схему. Провод тянется так: щёлкни по выводу одной детали, потом по выводу другой.
5. **Свойства выбранной детали** – например, цвет светодиода или сопротивление резистора.

Шаг 4. Готовая схема для старта. В списке компонентов выбери «Стартовые наборы» → «Arduino» (1 на левой половине) и перетащи на поле «Мерцание» (1 на правой половине): это Arduino со светодиодом, резистором и уже написанной программой. Рядом лежат и другие готовые схемы – кнопка, сервопривод, ультразвуковой дальномер.

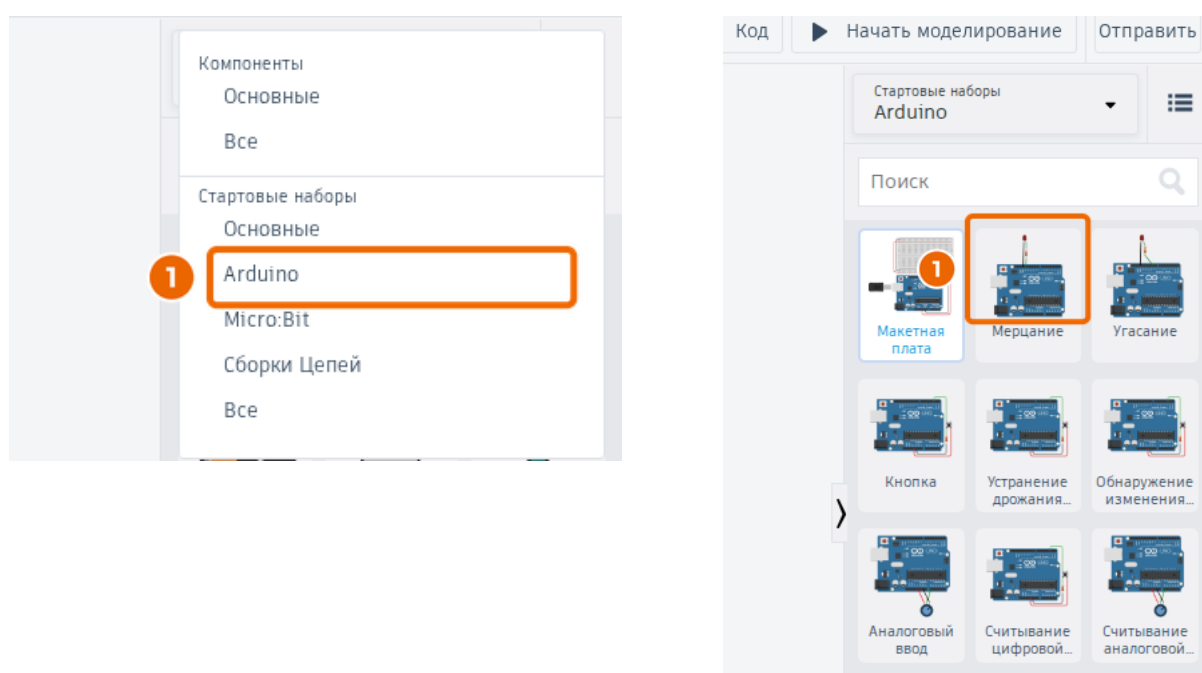


Рисунок В.4 – Стартовые наборы: слева – выбор набора «Arduino», справа – схема «Мерцание»

Шаг 5. Код и запуск. Кнопка «Код» (1) открывает программу. Её можно смотреть блоками или текстом на языке Arduino: переключатель «Блоки» вверху окна кода. Кнопка «Начать моделирование» (2) запускает схему.

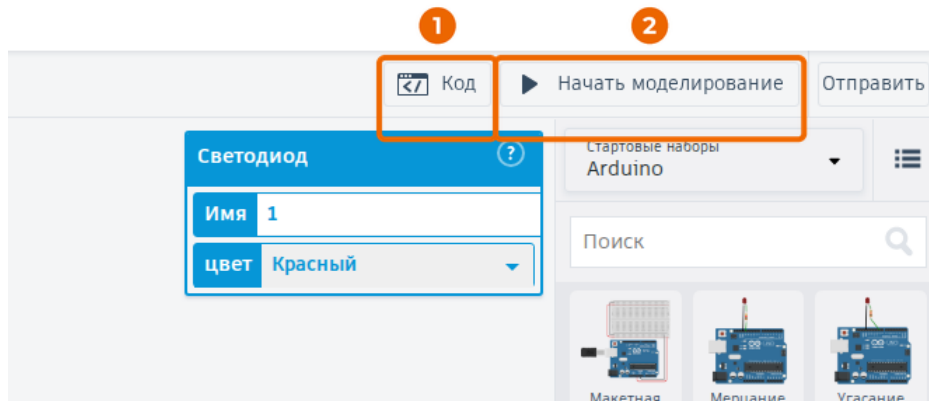


Рисунок В.5 – Кнопки «Код» и «Начать моделирование»

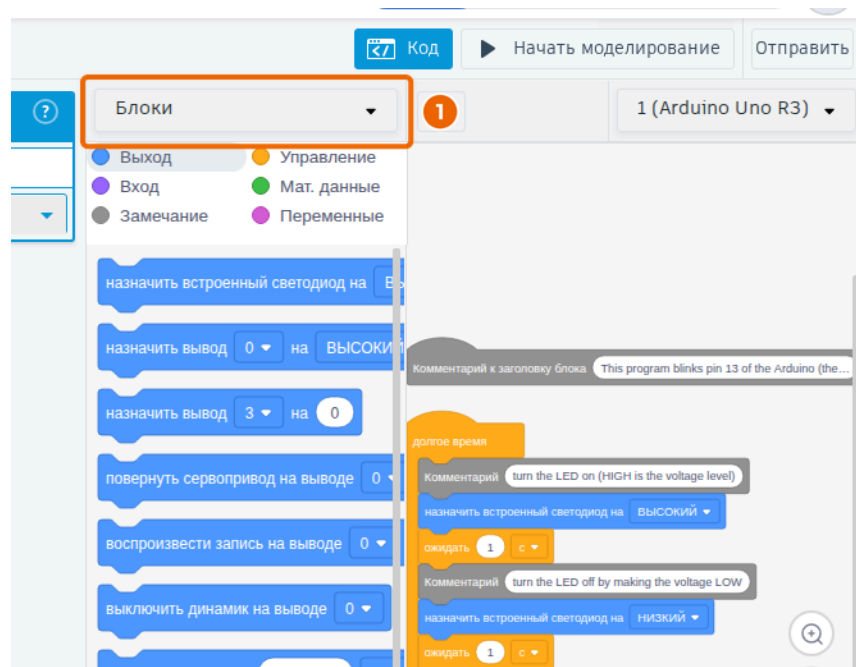


Рисунок В.6 – Программа «Мерцания» блоками

Во время моделирования кнопка называется «Остановить моделирование» (1), а светодиод на плате мигает (2): секунду горит, секунду нет. Остановить – той же кнопкой.

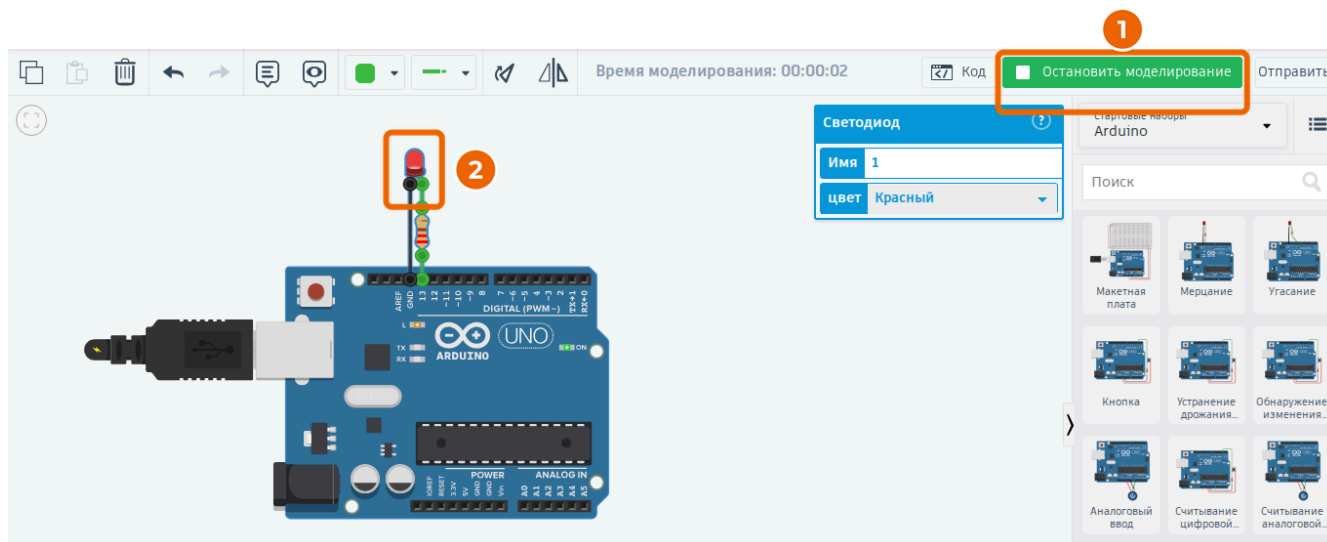


Рисунок В.7 – Схема работает

Важно про имя test . Под ним одновременно сидят все ученики курса: твою схему видят и могут поменять другие, а сохранение не гарантируется. Не пиши туда ничего личного. Если хочешь хранить свои работы, заведите вместе со взрослым свой аккаунт Autodesk – вход «Адрес эл. почты или имя пользователя» на том же экране, где выбирают способ входа.

Кибериада IDE: установка и первая программа

Кибериада IDE – бесплатная отечественная среда, в которой программу не пишут строчками, а рисуют диаграммой на языке ПРИМС: режимы работы – прямоугольники-состояния, смена режима – стрелки-переходы. Ниже весь путь от установки до первой программы «мигалка» – задача из курса сферы. Рамка с номером на картинке – куда нажимать.

Шаг 1. Скачать и поставить. Открой method.insitulab.pro/cyberiada и нажми «Скачать программу». Выбери файл под свою систему: .exe – Windows, .dmg – macOS, .deb или .AppImage – Linux. Запусти его. Установщик попросит права администратора – он ставит драйверы для плат, так и должно быть; дома нажимай «Да» вместе со взрослым. После установки программа откроется сама.

Шаг 2. Новый документ. Слева – панель «Документ», в ней «Создать...» (1).



Рисунок В.8 – Стартовый экран Кибериады: «Документ» → «Создать...»

Шаг 3. Платформа. Кибериада умеет работать с разными платами, поэтому сначала выбирают, для какой пишем. Дважды щёлкни «Arduino Uno» в списке платформ (1) – она появится в «Выбрано» (2). Нажми «Создать» (3).

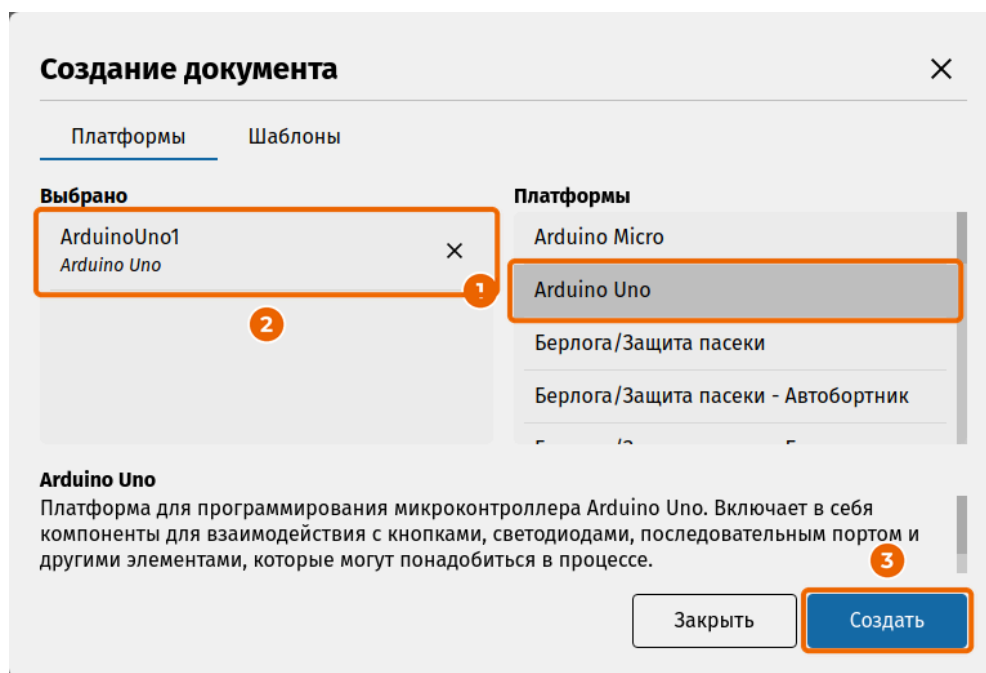


Рисунок В.9 – Выбор платформы для документа

Шаг 4. Компоненты. Компонент – это то, чем программа управляет или от чего ждёт сигнала: светодиод, кнопка, таймер. Нажми «+» у «Компонентов» (1).

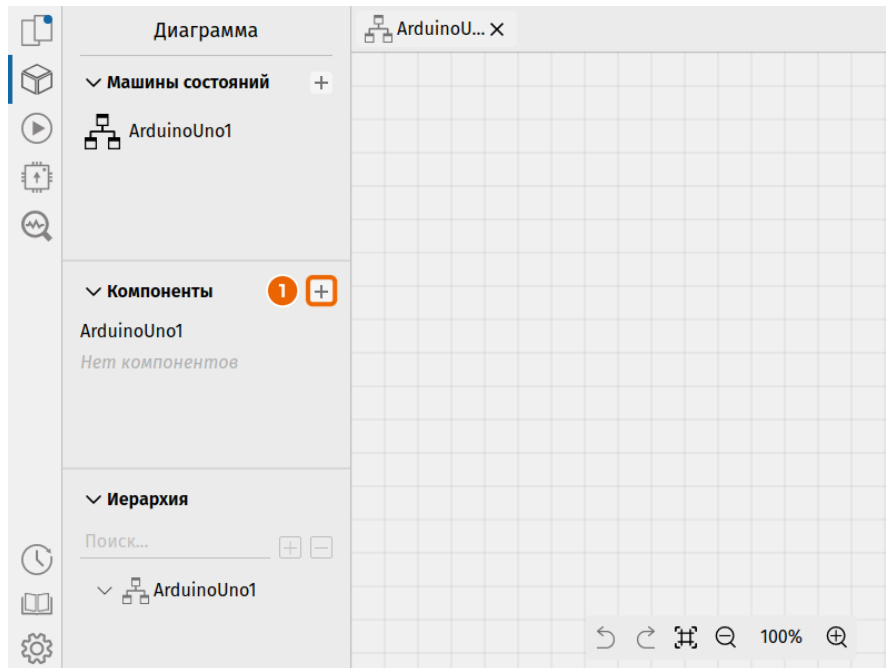


Рисунок В.10 – Окно диаграммы: плюс у «Компонентов»

Выбери «Светодиод» (1 на левой половине) и нажми «Добавить» (2). В форме впиши название «Светодиод» (1 на правой половине) и выбери пин (2) – номер вывода Arduino, к которому подключён светодиод; на занятии курса это 2. Нажми «Применить» (3). Так же добавь «Таймер» – у него пина нет, только название «Таймер».

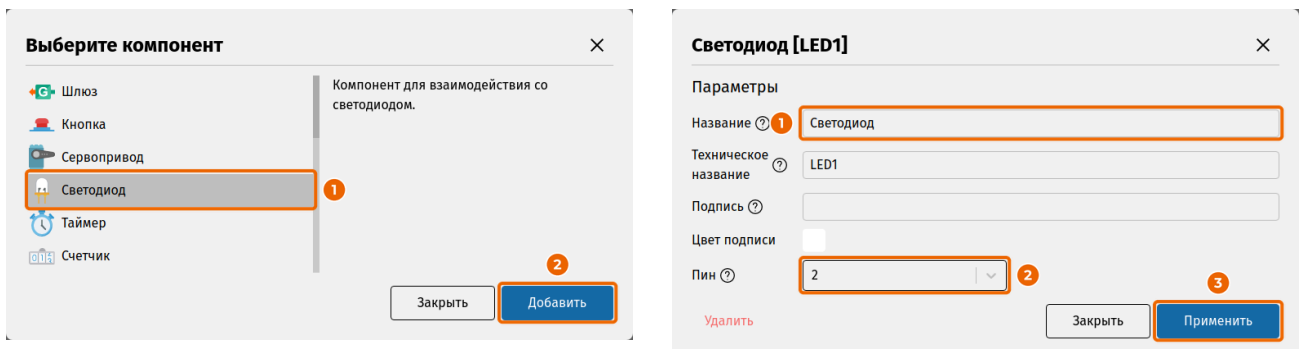


Рисунок В.11 – Новый компонент: слева – выбор «Светодиода», справа – его название и пин

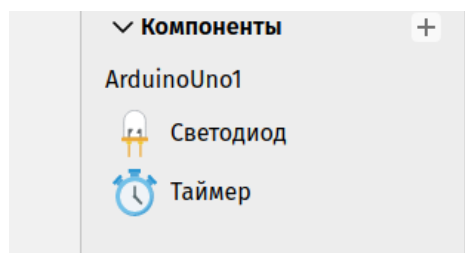


Рисунок В.12 – Оба компонента в списке

Шаг 5. Состояние «Не горит». Дважды щёлкни по пустому месту поля – появится прямоугольник «Состояние». К нему от синего кружка ведёт стрелка: отсюда программа начинает работу. Нажми карандаш (1) и впиши «Не горит», затем Enter.

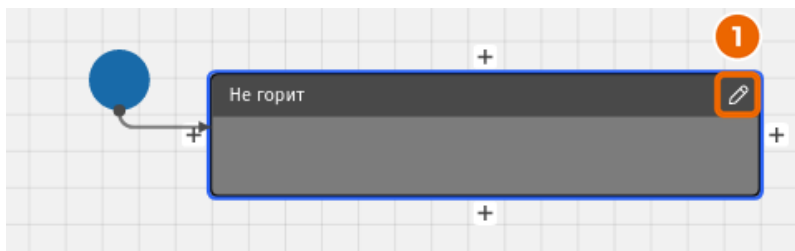


Рисунок В.13 – Новое состояние: карандаш – переименовать

Теперь скажем, что делать в этом состоянии. Дважды щёлкни по нему – откроется «Редактор состояния». Нажми «+» (1).

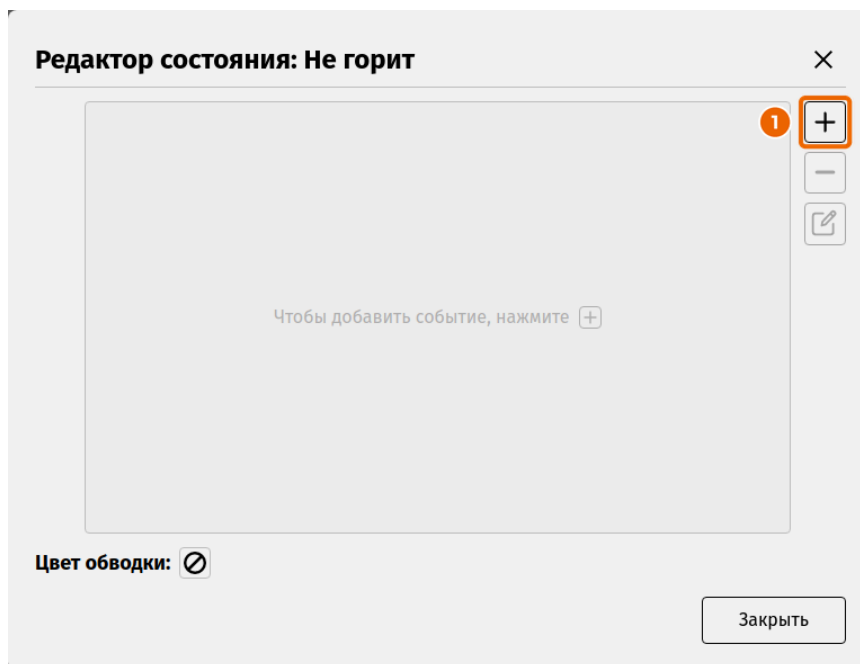


Рисунок В.14 – Редактор состояния

В «Редакторе события» в строке «Когда» уже стоит «Общие → Вход» (1): действия выполняются, как только программа войдёт в это состояние. Так и оставь. Нажми «+» у «Делай» (2).

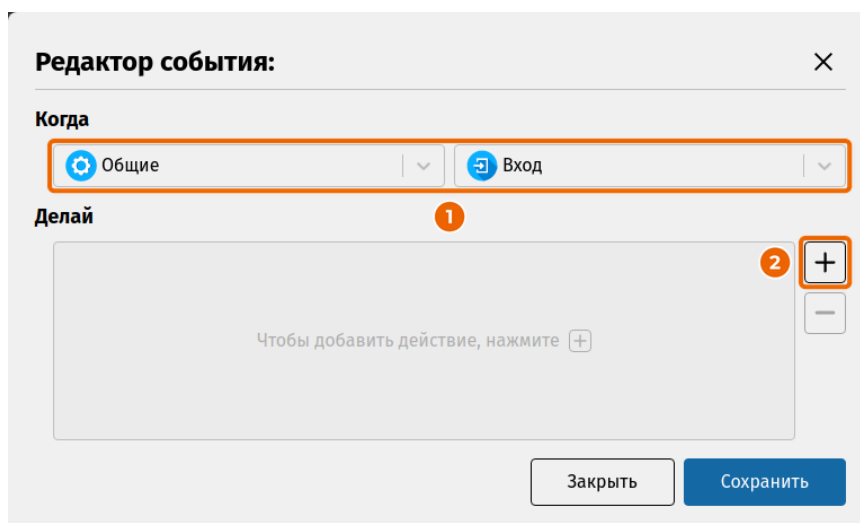


Рисунок В.15 – Редактор события: «Вход» и плюс у «Делай»

Выбери компонент «Светодиод» (1) и действие «Погасить» (2), нажми «Сохранить».

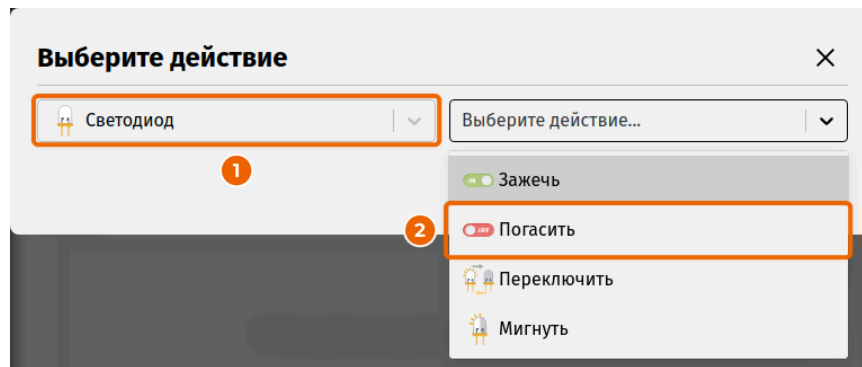


Рисунок В.16 – Действие «Светодиод – Погасить»

Ещё раз «+» у «Делай»: компонент «Таймер» (1), действие «Старт» (2), в поле интервала – 1000 (3), это миллисекунды, то есть одна секунда. Нажми «Сохранить» (4). Переключатель «Интервал» не трогай: он нужен, чтобы брать число из другого компонента, а нам хватит простого числа.



Рисунок В.17 – Действие «Таймер – Старт» на 1000 мс

В «Делай» теперь два действия (1). Нажми «Сохранить» (2), потом «Закреть» в редакторе состояния.

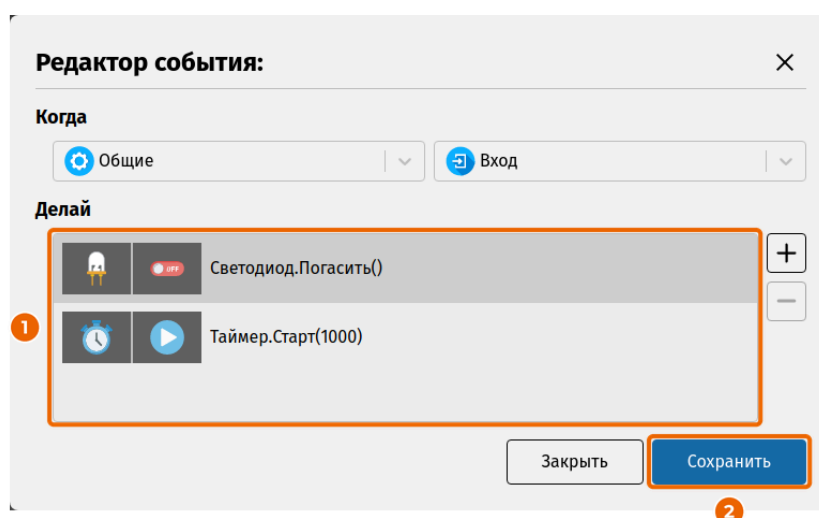


Рисунок В.18 – Событие «Вход» с двумя действиями

Шаг 6. Состояние «Горит». Так же: двойной щелчок по пустому месту, имя «Горит», одно действие – «Светодиод → Зажечь». Таймер второй раз запускать не нужно: на платформе Arduino Uno таймер повторяющийся – запустили один раз, и он срабатывает каждую секунду. На других платформах бывает иначе – смотри описание компонента.

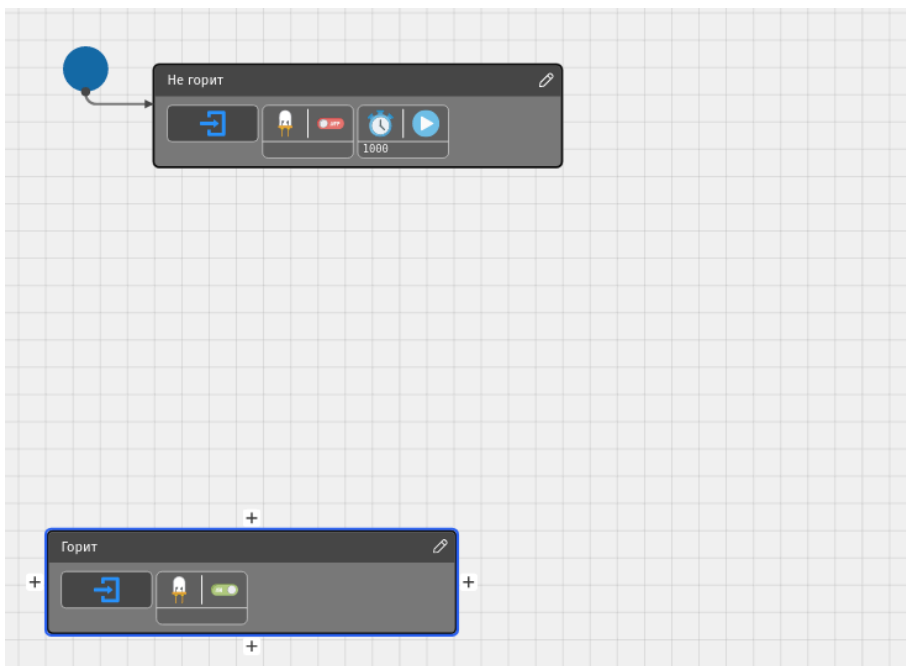


Рисунок В.19 – Два состояния готовы: на каждом видно, что оно делает при входе

Шаг 7. Переходы. Щёлкни по «Не горит» – вокруг появятся плюсики. Потяни мышью нижний плюстик (1) до состояния «Горит» (2) и отпусти.

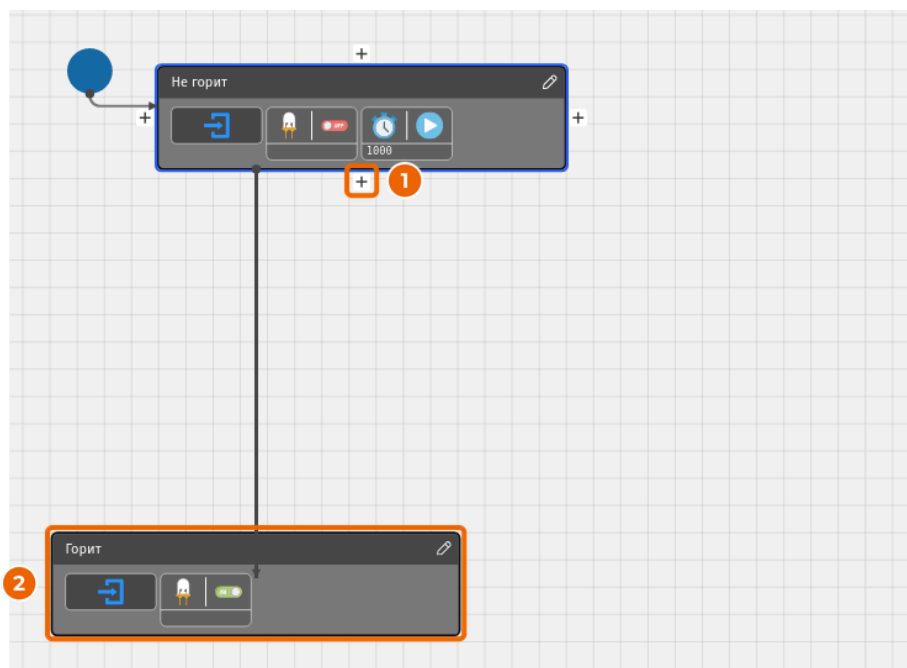


Рисунок В.20 – Тянем стрелку перехода

Откроется «Редактор соединения». В «Когда» выбери «Таймер» (1) и «Тайм-аут» (2) – переход случится, когда таймер отсчитает секунду. Условия и действия не нужны. Нажми «Сохранить» (3).

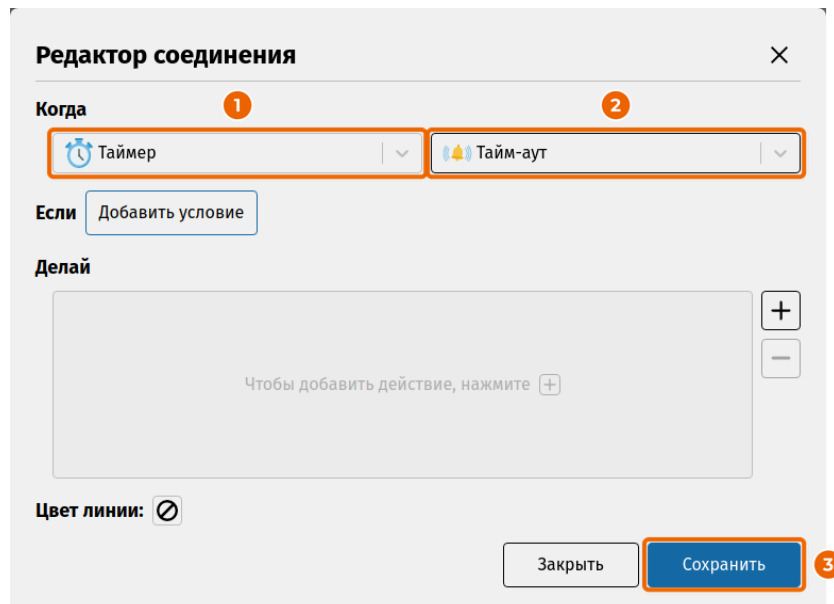


Рисунок В.21 – Переход по тайм-ауту таймера

Так же протяни стрелку обратно – из «Горит» в «Не горит», с тем же событием. Если две стрелки легли одна на другую, перетащи значок перехода мышью в сторону.

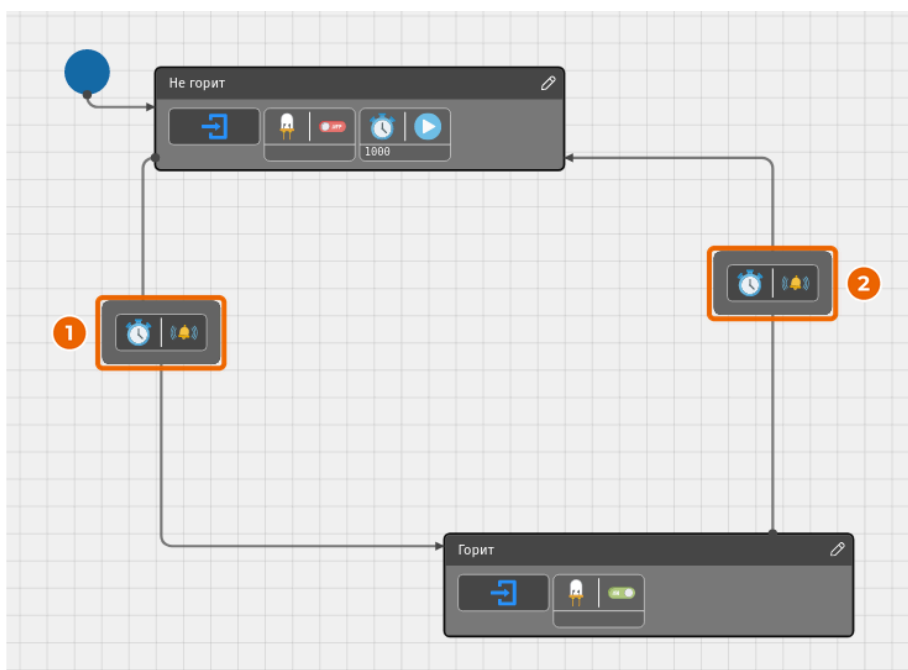


Рисунок В.22 – Готовая «мигалка»: переход «Не горит → Горит» (1) и обратный (2)

Прочитай программу по стрелкам: старт → «Не горит» (погасить, запустить таймер) → через секунду тайм-аут → «Горит» (зажечь) → тайм-аут → «Не горит» – и так по кругу. Светодиод мигает раз в секунду. Сохрани работу: «Документ → Сохранить как...», программа сохранится файлом `.graphml`.

Карта значков слева. По ним переключают панели.

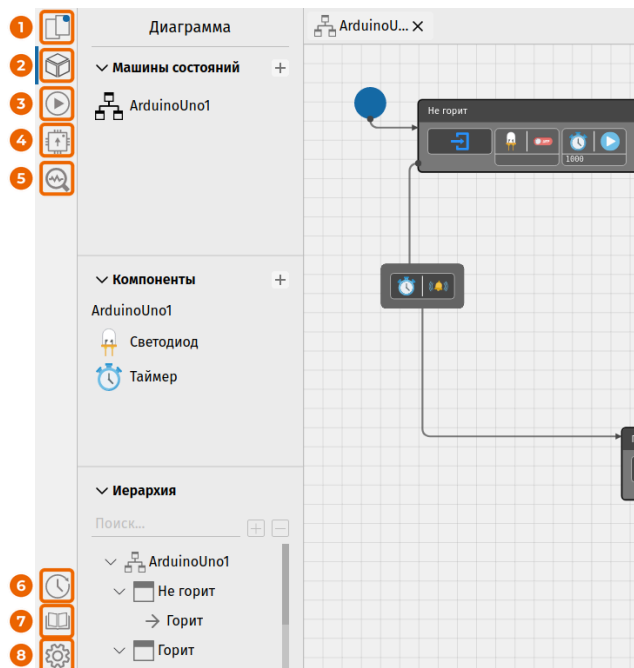


Рисунок В.23 – Значки левой панели

1. **Документ** – создать, открыть, сохранить.
2. **Диаграмма** – машины состояний, компоненты, иерархия состояний.
3. **Компилятор** – превращает диаграмму в прошивку для платы.
4. **Загрузчик** – записывает прошивку в плату через USB.
5. **Монитор порта** – показывает, что плата присылает по USB.
6. **История изменений** – что и когда менялось в документе.
7. **Документация** – справка (её же открывает F1).
8. **Настройки**.

Шаг 8. Компиляция и прошивка – когда есть плата. Открой «Компилятор» (3 на карте) и нажми большую синюю кнопку (1) – «Скомпилировать». Кружок рядом с заголовком (2) показывает связь с компилятором: если он красный, кнопка называется «Переподключиться» – нажми её и дождись, пока кружок станет зелёным.

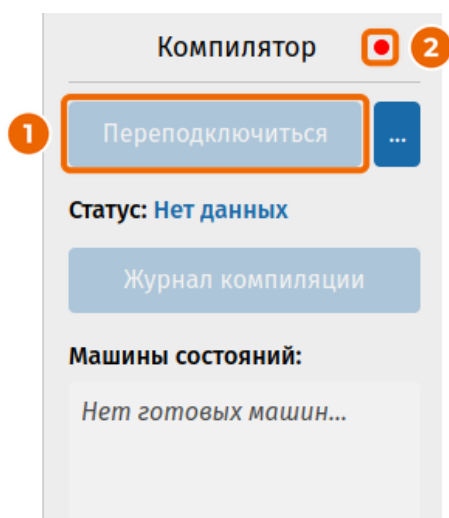


Рисунок В.24 – Панель «Компилятор»

Подключи Arduino к компьютеру кабелем и открой «Загрузчик» (4 на карте). Нажми «Подключить плату» (1), выбери свою плату в списке и нажми синюю кнопку с изображением платы – «Прошить». Прошивка готова, когда в «Журнале действий» появится подтверждение.

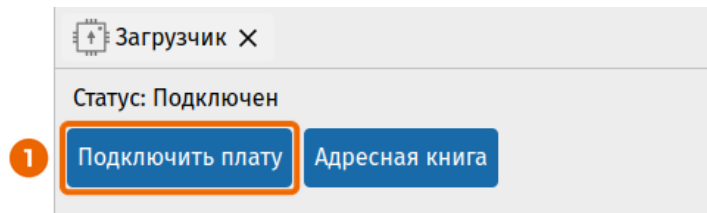


Рисунок В.25 – Панель «Загрузчик»

Платы под рукой нет – диаграмму всё равно можно проверить: пройди пальцем по стрелкам, как в разборах этой книги, и посмотри, в каком состоянии окажется программа после каждого события.

Wokwi: симулятор Arduino в браузере

Wokwi (wokwi.com) – бесплатный симулятор Arduino прямо в браузере. Ставить ничего не нужно, регистрироваться тоже: плату, светодиоды, кнопки и датчики «собирают» на экране проводами, а программу пишут тут же и запускают кнопкой. Ничего не сгорит, даже если ошибёшься. Если на твоём компьютере сайт не открывается, все схемы этой книги собираются и в Tinkercad. Интерфейс английский: названия кнопок ниже даны в кавычках по-английски, с переводом. Рамка с номером на картинке – куда нажимать. Зачем в схеме резистор – в главе об электричестве и электронике, что делают `setup()`, `loop()` и `Serial` – в главе про программирование Arduino. А здесь – как работать в самой программе и три задачи, чтобы руки привыкли.

Шаг 1. Открыть. Самый короткий путь – ссылка wokwi.com/projects/new/arduino-uno: сразу откроется чистый проект с платой Arduino Uno. То же самое кнопками: на главной странице wokwi.com выбери «Arduino (Uno, Mega, Nano)» (1 на левой половине рисунка), а на открывшейся странице в блоке «Start from Scratch» («Начать с нуля») – «Arduino Uno» (1 на правой половине).

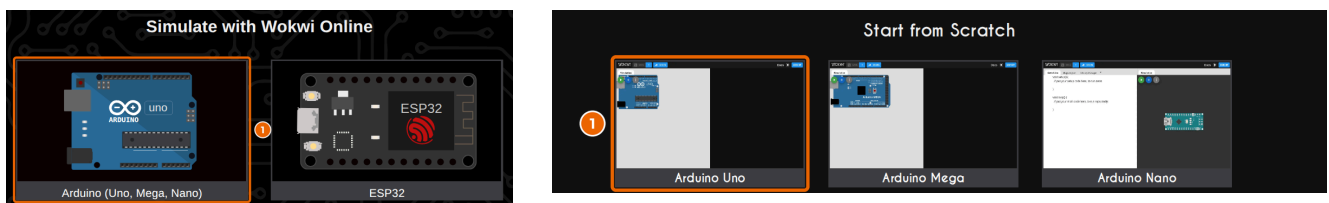


Рисунок В.26 – Как открыть чистый проект: слева – главная страница, справа – выбор платы

Входить в аккаунт не нужно: кнопки «Sign in» и «Sign up» (вход и регистрация) вверху справа не нажимай. Зато у работы без входа есть два неудобства. Первое: проект нигде не хранится. Обновишь страницу или закроешь вкладку – всё пропадёт, так что перед закрытием скачай проект (шаг 7). Второе: перед запуском Wokwi собирает программу на своём сервере. Обычно это доли секунды, иногда до минуты. Подожди, это нормально.

Шаг 2. Окно Wokwi. Слева – программа, справа – плата и схема.

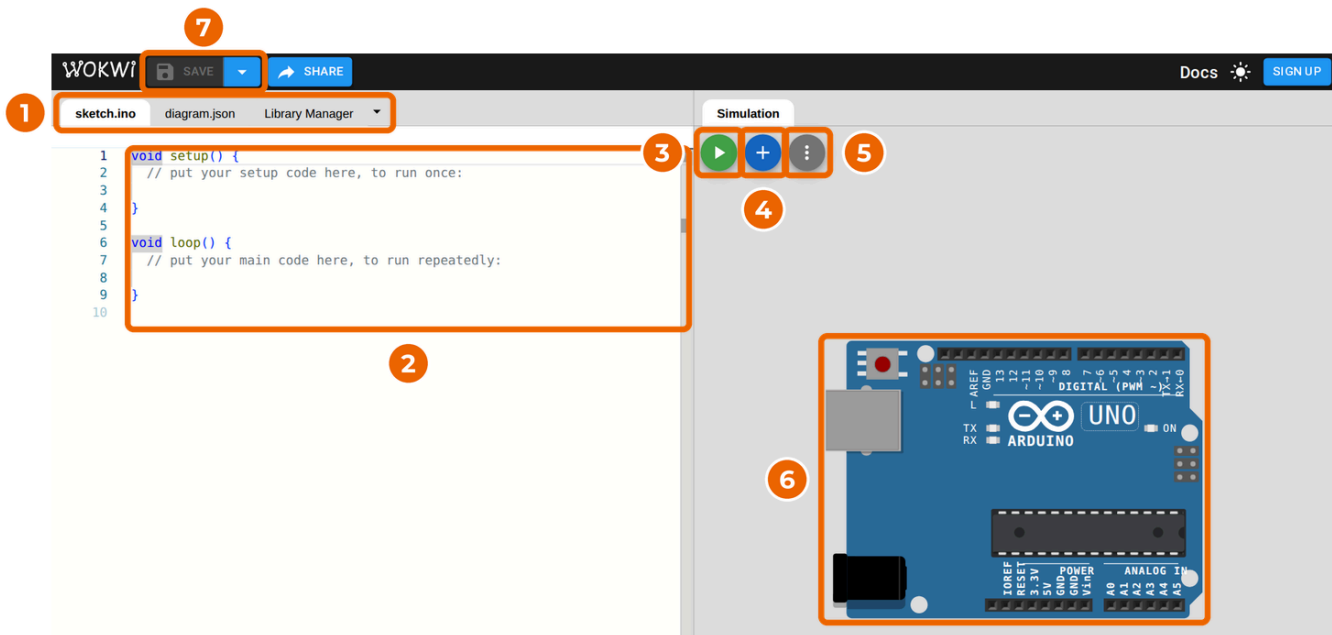


Рисунок В.27 – Окно Wokwi

1. **Вкладки.** «sketch.ino» – твоя программа. «diagram.json» – та же схема, записанная текстом: Wokwi пишет её сам, и в эту вкладку можно вставить готовую схему. «Library Manager» – подключаемые библиотеки, нам не понадобится.
2. **Окно программы.** Сразу лежат две обязательные функции, `setup()` и `loop()`.
3. **«Start the simulation»** («запустить симуляцию») – зелёная кнопка.
4. **«Add a new part»** («добавить деталь») – синяя кнопка «+».
5. **Меню «Zoom, grid, and more»** (три точки). «Undo» – отменить (Ctrl + Z), «Zoom in» и «Zoom out» – приблизить и отдалить, «Fit» – вернуть схеме размер по окну, «Toggle Grid» – показать сетку. Меню само не закрывается: закрой его клавишей Esc.
6. **Плата Arduino Uno.** Ножки подписаны: цифровые 0–13 – сверху, а питание (5V, GND – «земля») и аналоговые A0–A5 – снизу.
7. **«Save»** («сохранить») работает только со входом, поэтому нам нужна стрелка рядом с ним: в её меню лежит «Download project ZIP» (шаг 7).

Шаг 3. Добавить деталь. Нажми «+» (4 на карте окна). Откроется список деталей. В разделе «Basic» («основные») лежат LED – светодиод, Pushbutton и Pushbutton 6mm – кнопки (по-больше и поменьше), Resistor – резистор (2). Нужной детали на виду нет? Впиши её английское название в строку поиска (1), например `potentiometer`. Щёлкни по детали – она появится в левом верхнем углу поля. Нажми на неё и перетащи мышью на свободное место рядом с платой.

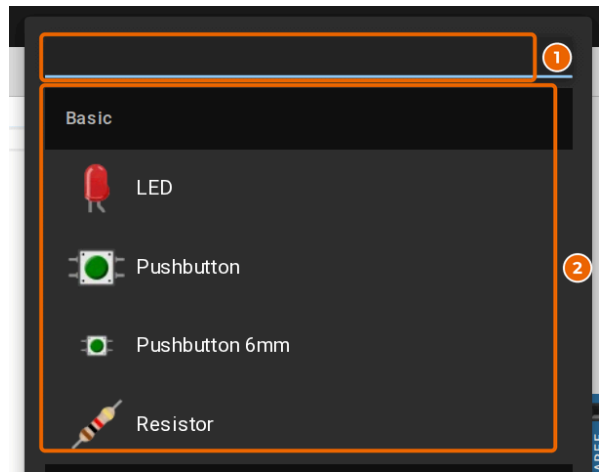


Рисунок В.28 – Список деталей: строка поиска (1) и раздел «Basic» (2)

Шаг 4. Настроить и повернуть. Щёлкни по детали – над полем появится её панель. У резистора в ней поле «Resistance» («сопротивление», 1), кнопка «Rotate» («повернуть», 2) и корзина «Delete» («удалить», 3). Впиши в поле 220 и нажми Enter: полоски на резисторе перецветятся. В поле число – это омы, а буква к – «кило»: 1k – это 1000 Ом.



Рисунок В.29 – Панель резистора: «Resistance» (1), «Rotate» (2), «Delete» (3)

Деталь поворачивают кнопкой «Rotate» или клавишей R: одно нажатие – четверть оборота. Удалить деталь можно корзиной или клавишей Delete.

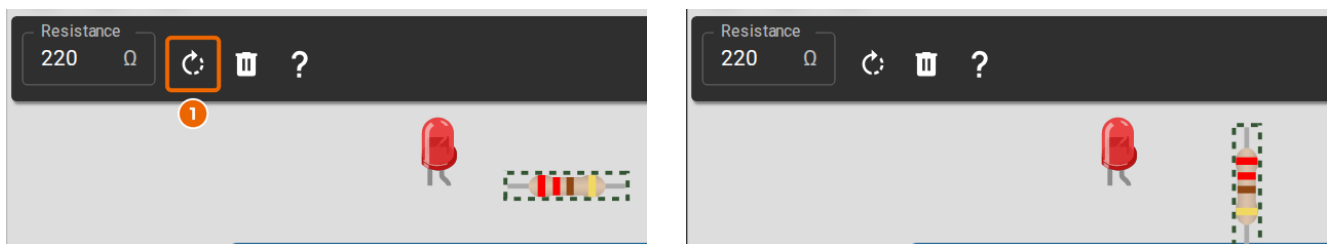


Рисунок В.30 – Поворот: слева резистор лежит, справа – после нажатия «Rotate» (1) он встал вертикально

Шаг 5. Провода. Щёлкни по ножке одной детали – от неё потянется провод. Щёлкни по ножке другой детали – провод закрепится. Передумал, пока тянешь, – нажми Esc, и провода не будет. Наведи мышь на ножку, и Wokwi её подпишет: led1:C – это деталь led1 и её ножка C, катод (у светодиода есть ещё led1:A – анод). Пока тянешь провод, цвет можно сменить цифрой: 2 – красный, 0 – чёрный, 5 – зелёный (как цвета полосок на резисторах). Лишний провод убирают так: щёлкни по нему и нажми Delete.

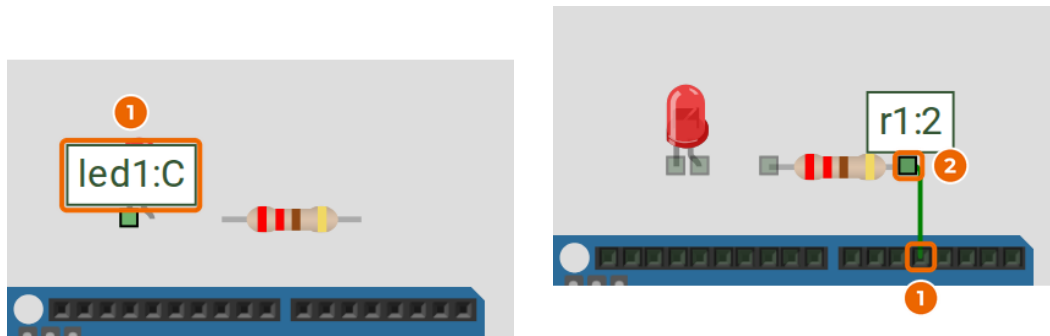


Рисунок В.31 – Слева – подпись ножки при наведении (1), справа – провод тянется от ножки 4 (1) к ножке r1:2 резистора (2)

Шаг 6. Запуск и монитор порта. Нажми «Start the simulation». Пока Wokwi собирает программу, подожди, потом схема оживёт: светодиоды горят, кнопки нажимаются мышью. На месте зелёной кнопки появятся три: «Restart the simulation» (начать заново), «Stop the simulation» (остановить, 1) и «Pause» (пауза). Кнопки «+» и меню с тремя точками при этом скрыты: чтобы поправить схему, сначала нажми «Stop». Вверху справа бежит время симуляции (2). Если в программе есть `Serial.println(...)`, внизу откроется **монитор порта** (3). Это окно, куда плата «печатает» числа и слова. В нижнем ряду у него кнопка графика («Serial plotter»), корзина (очистить окно) и стрелка (свернуть монитор).

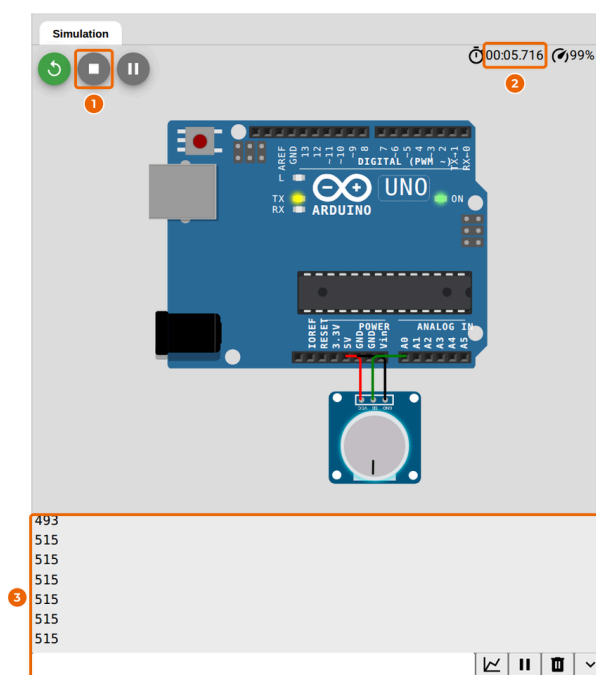


Рисунок В.32 – Идёт симуляция: «Stop the simulation» (1), время (2), монитор порта (3)

Шаг 7. Сохранить. Без входа работу сохраняют архивом: нажми стрелку рядом с «Save» (1) и выбери «Download project ZIP» (2). Скачается `project.zip`: в нём `sketch.ino` – программа, `diagram.json` – схема и ещё `wokwi-project.txt` – ссылка на сайт. Чтобы вернуться к работе, открой новый проект, во вкладку `sketch.ino` вставь текст из `sketch.ino`, а во вкладку `diagram.json` – текст из `diagram.json`.

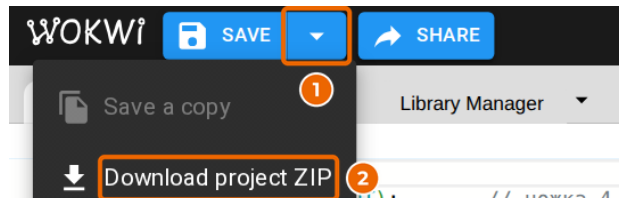


Рисунок В.33 – Меню рядом с «Save»

Задача П.1. Мигалка. Собери схему: ножка 4 → резистор 220 Ом → анод светодиода, катод светодиода → GND. На рисунке провод от ножки 4 (1) идёт к одному концу резистора (2), от другого его конца – к аноду светодиода (3), а от катода (левая ножка) – на GND (4) рядом с ножкой 13. Резистор ограничивает ток: в настоящей схеме без него светодиод бы сгорел.

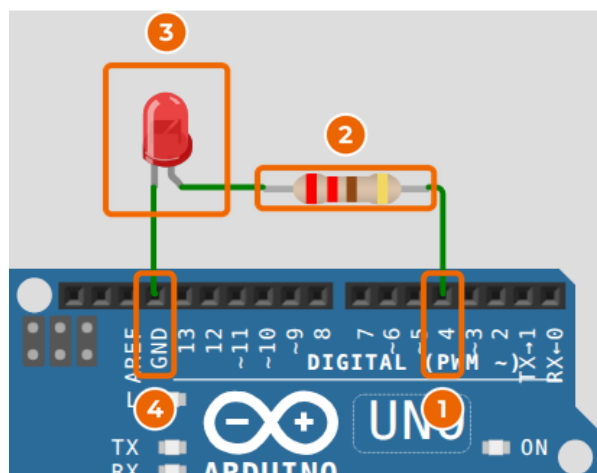


Рисунок В.34 – Схема мигалки

Вставь в окно программы текст и нажми «Start the simulation».

```
void setup() {
  pinMode(4, OUTPUT);      // ножка 4 - выход
}

void loop() {
  digitalWrite(4, HIGH);   // светодиод включить
  delay(1000);             // ждать 1000 мс = 1 с
  digitalWrite(4, LOW);    // выключить
  delay(1000);            // ждать ещё 1 с
}
```

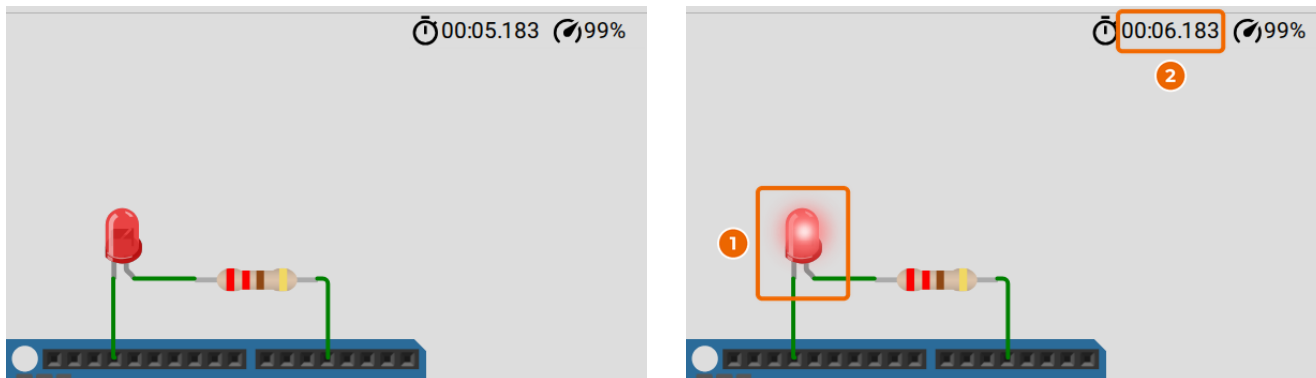


Рисунок В.35 – Слева светодиод не горит, справа горит (1); время симуляции – (2)

Что должно получиться: секунду светодиод горит, секунду нет. Теперь нажми «Stop», поменяй оба `delay(1000)` на `delay(250)` и запусти снова. Сколько раз вспыхнет светодиод за 10 секунд по времени симуляции? Ответ: 20. Одна вспышка – это 250 мс горит и 250 мс не горит, то есть 0,5 с, а $10 : 0,5 = 20$. Мы прогнали это в Wokwi: за 10 секунд ровно 20 вспышек. Светодиод не горит совсем? Проверь полярность: анод (правая ножка) должен смотреть в сторону резистора, а катод – на GND. Включённый наоборот светодиод не загорится, но и не сгорит.

Задача П.2. Кнопка. Открой новый проект (шаг 1): схема из П.1 больше не нужна. Лампочку возьмём ту, что уже стоит на плате: это светодиод L, он подключён к ножке 13. Добавь Pushbutton. У кнопки четыре вывода – два сверху и два снизу. Внутри кнопки два верхних вывода соединены всегда, два нижних тоже, а нажатие соединяет верхнюю пару с нижней. Поэтому провода берут так: один вывод верхней пары – к ножке 2, один вывод нижней пары – к GND. На рисунке это верхний правый вывод (1) и ножка 2 (3), нижний левый вывод (2) и GND (4).

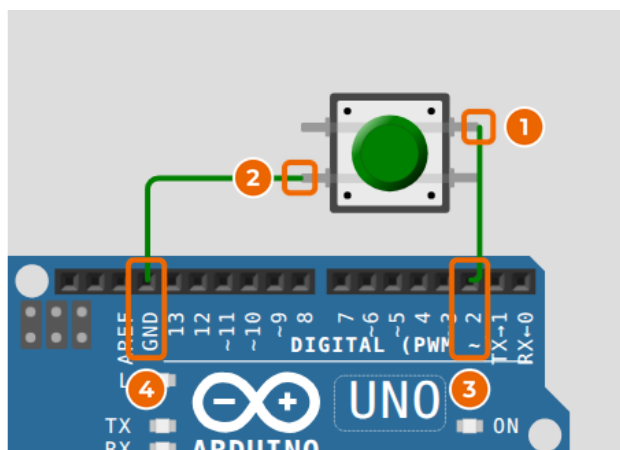


Рисунок В.36 – Схема с кнопкой

Программа. Здесь кнопка – вход с подтяжкой (`INPUT_PULLUP`): пока кнопка отпущена, на ножке 1.

```

void setup() {
  pinMode(2, INPUT_PULLUP); // кнопка: вход, подтянутый к 5 В
  pinMode(13, OUTPUT);      // светодиод L на плате
  Serial.begin(9600);
}

void loop() {
  int knopka = digitalRead(2);
  Serial.println(knopka);
  if (knopka == LOW) {      // нажата = LOW
    digitalWrite(13, HIGH);
  } else {
    digitalWrite(13, LOW);
  }
  delay(200);
}

```

Запусти, потом нажми на кнопку мышью и подержи, а затем отпусти.

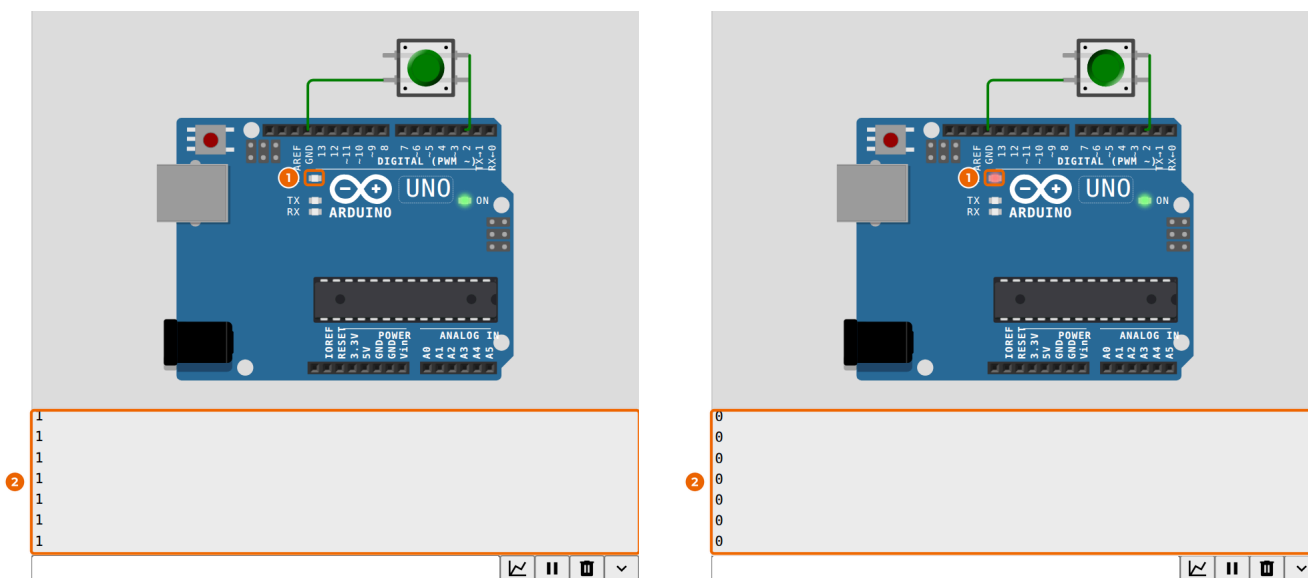


Рисунок В.37 – Слева кнопка отпущена, справа нажата: лампочка L (1) и монитор порта (2)

Что должно получиться: пока кнопка отпущена, в мониторе порта единицы, а L не горит. Нажал и держишь – пошли нули, и L загорается. Отпустил – снова единицы. Ответ: нажатой кнопке соответствует 0 (LOW), отпущенной – 1. Сплошные нули без нажатия? Значит, оба провода сели на выводы одной пары, например на два верхних: такая кнопка «нажата» всегда. Перенеси один провод на вывод другой пары.

Задача П.3. Потенциометр. Открой новый проект. Перетащи плату повыше: под ней нужно место. Добавь «Potentiometer» (строка поиска, potentiometer), поставь его под платой, щёлкни по нему и дважды нажми R: выводы окажутся сверху, ближе к плате. Они подписаны мелко: VCC, SIG, GND. Провода: красный – от VCC на 5V, чёрный – от GND на GND рядом с 5V, зелёный – от SIG на A0. На рисунке это выводы потенциометра (1), питание платы (2) и аналоговая ножка A0 (3).

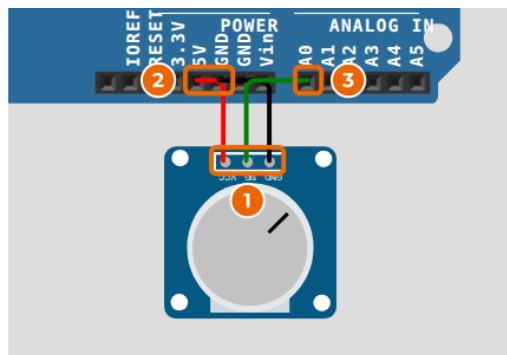


Рисунок В.38 – Схема с потенциометром

Программа печатает в монитор порта число с ножки A0.

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int n = analogRead(A0);    // число от 0 до 1023
  Serial.println(n);
  delay(300);
}
```

Запусти, нажми мышью на круглую ручку и води по кругу: ручка идёт за курсором, а числа в мониторе меняются сразу. Между двумя упорами у ручки есть пустой промежуток. Если провести через него, число перескочит с 1023 на 0 или обратно.

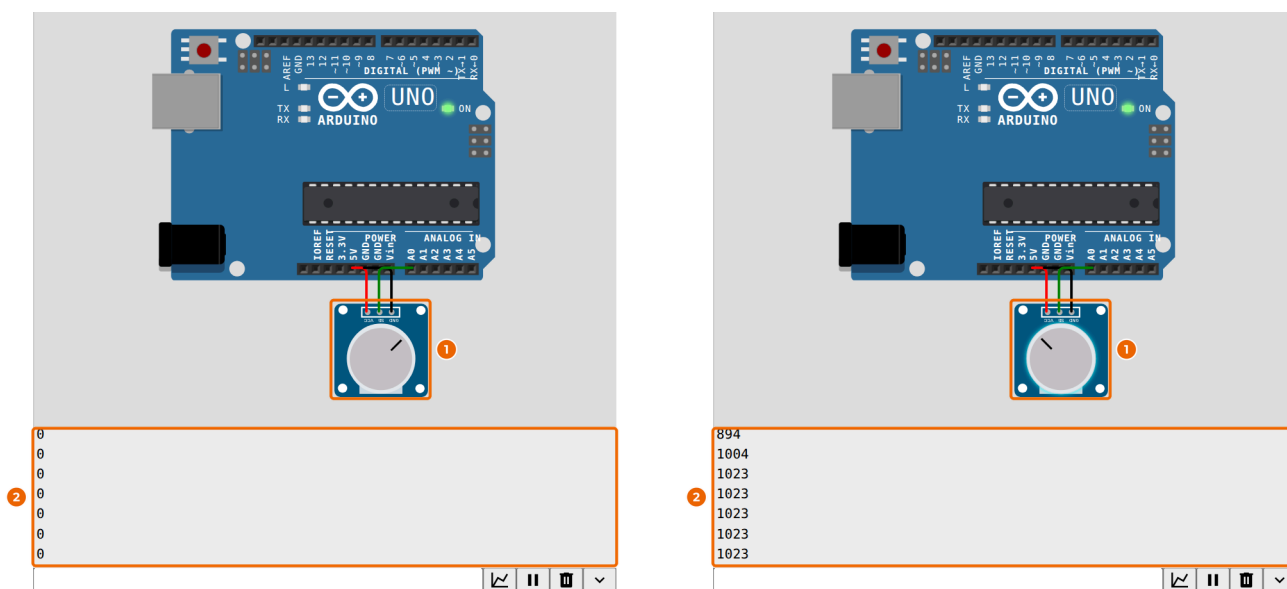


Рисунок В.39 – Слева ручка в одном крайнем положении, справа – в другом: ручка (1) и монитор порта (2)

Что должно получиться: в одном крайнем положении ручки в мониторе 0, в другом – 1023, а посередине – около 500. Сразу после добавления ручка стоит на нуле, поэтому первые числа в мониторе – нули. Ответ: 0 и 1023. Это границы `analogRead`: 0 соответствует 0 В, а 1023 – 5 В. Вопрос на подумать: какое напряжение на A0, когда в мониторе 512? Ответ: около 2,5 В, потому что $512 \cdot 5 / 1023 \approx 2,5$.

План на месяц до отбора

Если начинаешь в конце сентября, вот спокойный план: примерно по часу в день.

Таблица В.3

Неделя: что читаем	Главное дело недели
1: главы 1–3	записаться на курс сферы
2: главы 4–5	собрать схемы в Tinkercad
3: главы 6–7	запустить программы из разборов
4: глава 8 и шпаргалка	тест курса сферы (+5 баллов)

Подробнее по неделям:

- **Неделя 1.** Запишись на курс сферы и войди в класс «Роботех» в Tinkercad – просто чтобы убедиться, что всё открывается. Каждый день решай «Реши сам» к свежим разборам: путь колеса, скорость, координаты, повороты.
- **Неделя 2.** Прогони в тетради алгоритмы исполнителей из главы 4. Собери в Tinkercad схемы из главы 5: мотор через драйвер, ультразвуковой дальномер, сервомотор.
- **Неделя 3.** Повтори программы из разборов глав 6 и 7, посмотри показания датчиков в мониторе порта. Нарисуй в Кибериаде хотя бы одну машину состояний.
- **Неделя 4.** Заново реши задачи, где были ошибки, пролистай шпаргалку и пройди тест курса сферы.

С 1 октября у олимпиады по плану открывается демо-этап с пробными заданиями. Попробуй их, как только появятся: так заранее увидишь, как выглядят задачи на «Орбите» и куда вводить ответы, и не будешь тратить на это время отбора.

Отбор длится до 10 ноября, и попыток две. Если первая прошла неудачно, вернись к главам, где потерял баллы, и иди на вторую.

Родителям и наставникам

- **Помогайте разбираться, но не решайте за ребёнка.** На командном финале сразу видно, кто разобрался сам, а за кого решили. Лучше пусть ребёнок ошибётся на отборе и научится, чем попадёт в финал и растеряется там.
- **Садитесь рядом на первых задачах в симуляторе.** Детей останавливает скорее «не могу войти в класс», «схема пропала» или «сервер недоступен», чем трудная задача. Две подсказки из вебинаров сферы: в общем классе Tinkercad проект может не сохраниться до следующего захода, поэтому код стоит копировать в файл, а схему зарисовывать в тетрадь; если симулятор пишет, что сервер недоступен, помогает подождать и обновить страницу.
- **Если дома есть настоящий робот или плата Arduino,** схему меняют только при выключенном питании, мотор подключают только через драйвер, а пальцы держат подальше от колёс и шестерёнок, пока робот включён. В симуляторе ничего не сгорит, а настоящий светодиод без резистора сгорает за секунду.

- **Всё про регистрацию, бонусы и финал** для Новосибирской области – на nto.kvantorium54.ru (вкладки «Я наставник» и «Я участник или родитель»). Там же – про бонус за нормативы ТехноГТО и его сроки.
- **Региональный координатор НТО Юниоры в Новосибирской области** – детский технопарк «Кванториум», info@kvantorium54.ru.

Честно об этой книге

Главное, что нужно знать про эту книгу: заданий отбора 2026/27 мы не видели – организаторы их заранее не показывают. Угадывать их бесполезно: наставники, которые следят за олимпиадой не первый год, говорят, что задания из года в год не повторяются. Поэтому книга учит темам из программы сферы и приёмам решения, а не готовит к конкретным задачам. Когда организаторы выложат пробные задания, мы сверим с ними книгу и выпустим новую версию.

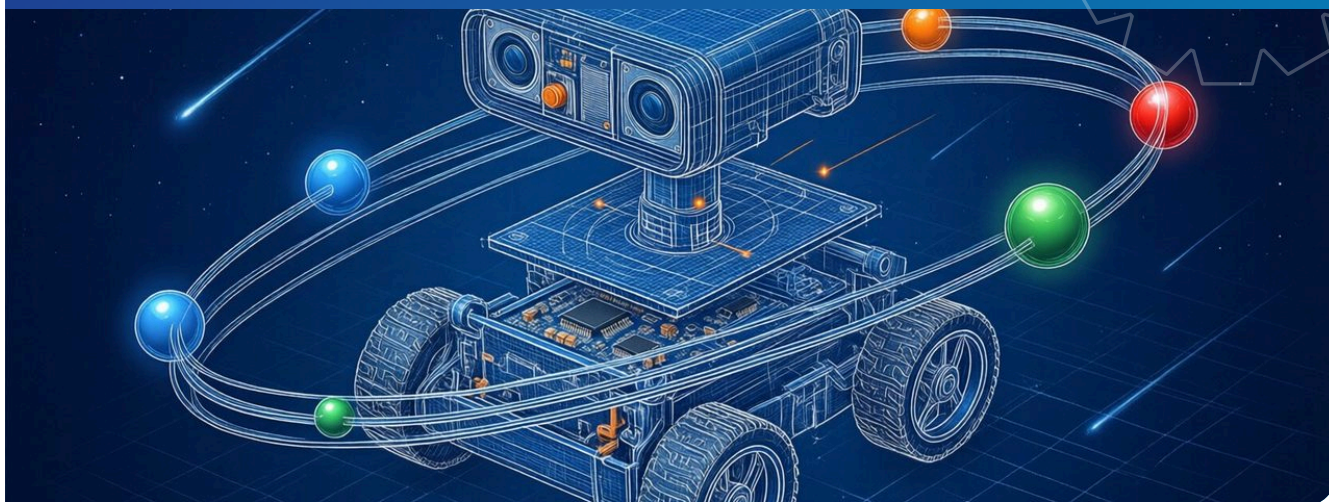
Это не официальное издание НТО. Книгу составил методист «Кванториума» по открытым материалам олимпиады: программе сферы «Роботех. Основы мобильной робототехники» 2026/27, записям вебинаров сферы для наставников и участников, официальным требованиям к площадкам финала.

У «Роботеха» были предшественницы: «Технологии для мира роботов», «Технологии и роботы», а в 2022 году – сразу две сферы, «Технологии и роботы-исследователи» и «Технологии и роботы на производстве». Но тексты их отборочных задач в открытом доступе закрыты: от курса 2023 года видны только названия уроков вроде «Математика наше всё!» и «Пора разобратся со схемами». Эти названия подсказали нам, о чём спрашивают. Ещё несколько идей мы взяли из сборника задач старшей олимпиады по профилю «Интеллектуальные робототехнические системы» 2024/25 – там задачи для 8–11 классов, поэтому мы их сильно упростили. Остальные задачи придуманы заново по программе сферы.

Книга написана до демо-этапа с пробными заданиями. Когда он откроется, мы сверим книгу с настоящим форматом задач и обновим её. Если на пробном этапе тебе попало что-то, чего в книге нет, смело пиши – это поможет следующей версии. Кнопки Tinkerscad и Кибериады описаны такими, какими их показывали на вебинарах сферы в августе и сентябре 2026 года: если интерфейс изменится, смысл останется тем же.

Каждое число в ответах мы пересчитали на компьютере, а каждую программу запустили. Если ты всё-таки нашёл ошибку или что-то осталось непонятным, напиши автору: osinovsky.aa@kvantorium54.ru. Следующие версии книги станут лучше благодаря тебе.

Как устроен робот



С этой главы начинается вся книга: прежде чем учить робота ездить по линии и искать выход из лабиринта, надо знать, из каких деталей он собран и что делает каждая. Здесь почти нет формул – зато много названий, которые дальше встретятся на каждой странице. В конце главы – две механические штуки, без которых не обходится ни один робот: зубчатая передача и рычаг. На отборе по этой теме бывают задачи «сопоставь деталь и её работу», «выбери все верные утверждения» (в ответ вводишь цифры выбранных пунктов) и короткие расчёты: сколько оборотов сделает колесо, если между ним и мотором стоят шестерёнки.

1.1. Что такое робот и какие они бывают

НА ПАЛЬЦАХ

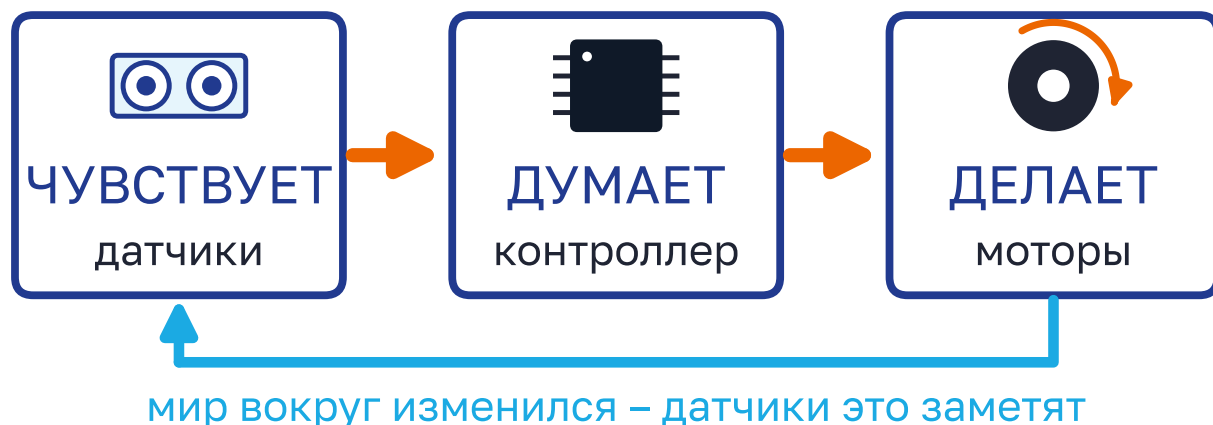


Рисунок 1.1 – Робот всё время делает три вещи по кругу

Датчики замечают, что творится вокруг. Контроллер решает, как поступить. Моторы выполняют решение – и мир вокруг робота меняется, а датчики это снова замечают.

Робот – машина, которая сама узнаёт, что происходит вокруг, сама решает, что делать, и сама это делает. Решает она не «от ума», а по **программе** – списку правил, который заранее написал человек. Например: «впереди стена ближе 20 см – остановись и повернись».

Возьмём машинку на радиоуправлении. Моторы у неё есть, колёса есть, а решает за неё мальчик с пультом. Отпусти он пульт – машинка встанет или врежется в диван. Такую машину роботом не называют. А робот-пылесос ездит по квартире один: наткнулся бампером на ножку стула – сдал назад и повернул. Про такие машины говорят – **автономные**, то есть работающие без человека.

Роботов на свете много, и они совсем разные. Первое, что бросается в глаза, – умеет робот перемещаться или нет.

- **Промышленный робот-манипулятор** – это «рука» с несколькими суставами, прикрученная к полу. Он сваривает кузова машин на заводе, перекладывает коробки, красит детали. Сам по цеху не ездит.
- **Мобильный робот** («мобильный» значит «подвижный») сам перемещается с места на место. Эта книга – о них.

Мобильные роботы различаются тем, **как** они двигаются:



Рисунок 1.2 – Какие бывают мобильные роботы. Подписи внизу – где такой способ движения выручает

- **Колёсные** – самые простые и быстрые на ровном полу: складские тележки, роботы-пылесосы, роботы-доставщики, марсоходы.
- **Гусеничные** – гусеница давит на землю широкой лентой и не проваливается там, где колесо буксует: рыхлый снег, песок, завалы. Такие роботы помогают спасателям и сапёрам.
- **Шагающие** – на ногах, как собака или человек. Перешагивают камни и поднимаются по ступенькам, но устроены сложнее всех.
- **Летающие** – дроны-квадрокоптеры. Если дрон летит по заданному маршруту сам, без пилота с пультом, он тоже робот.
- Есть ещё **плавающие и подводные** роботы: они осматривают дно, трубы и опоры мостов.

Роботу не бывает скучно, поэтому ему отдают однообразную работу, а заодно и опасную. На складе интернет-магазина тележки-роботы подвозят стеллажи к сборщикам. На поле робот-опрыскиватель едет вдоль грядок. В космосе без таких машин вообще никуда: люди побывали только на Луне, а колёсные машины без людей ездили и по Луне, и по Марсу.

Из истории. Первым в мире планетоходом, который успешно работал на другом небесном теле, стал советский «Луноход-1». Его доставила на Луну станция «Луна-17» 17 ноября 1970 года. У лунохода было восемь колёс, и у каждого – свой электромотор со своим редуктором (что такое редуктор, узнаешь в разделе 1.4). Он проработал на Луне около десяти месяцев и проехал 10 540 м. Строго говоря, роботом он не был: вёл его экипаж водителей с Земли, глядя на снимки с телекамер лунохода. Современные марсоходы часть пути уже прокладывают сами.

Запомни. Спорный случай «робот или нет» решай одним вопросом: **кто принимает решение?** Человек с пультом – значит, это управляемая машина. Программа по сигналам датчиков – робот. Мобильный робот вдобавок сам перемещается: на колёсах, гусеницах, ногах или винтах.

1.2. Из чего состоит мобильный робот

НА ПАЛЬЦАХ

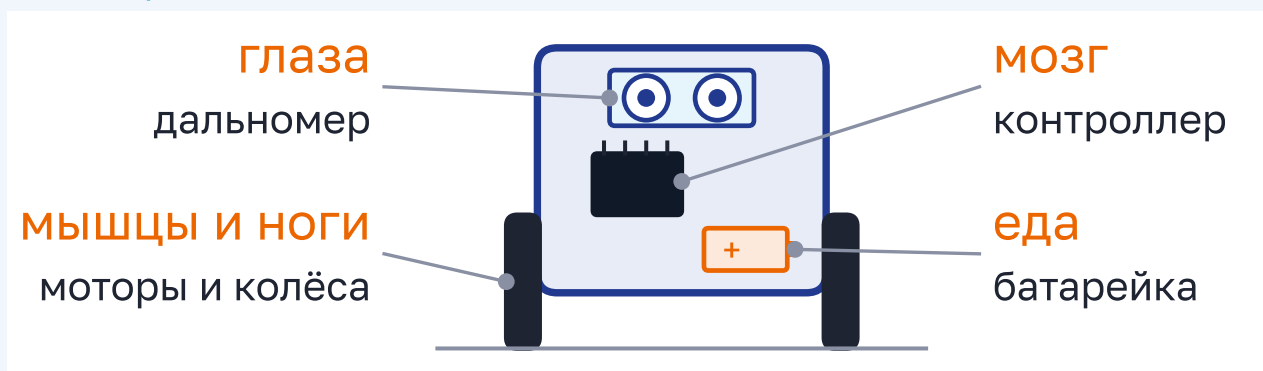


Рисунок 1.3 – Робот похож на человека: у каждой его детали есть «родственник» в нашем теле

Датчики – органы чувств, контроллер – мозг, моторы – мышцы, батарейка – еда, которая даёт силы.

Разберём по деталям робота, которого собирают в кружках сферы «Роботех». Это небольшая тележка на двух колёсах с моторами и одним маленьком опорном колёсике.

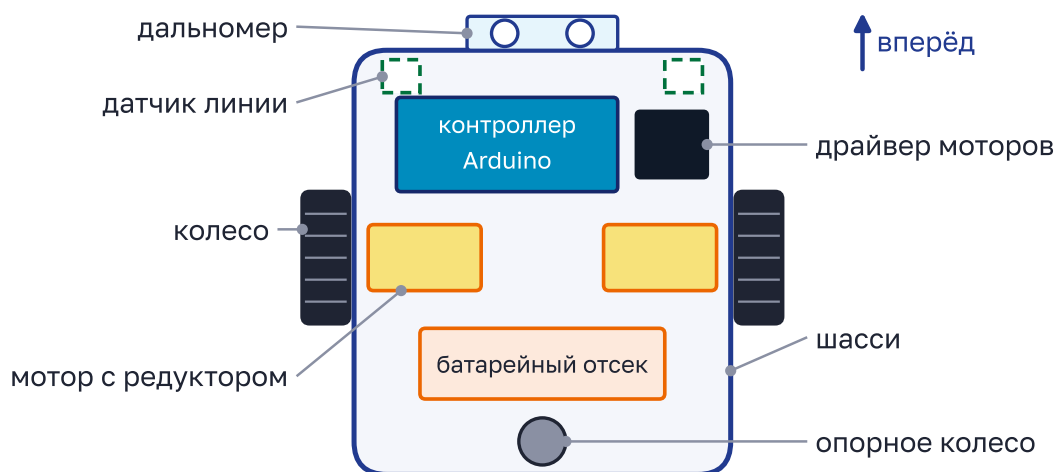


Рисунок 1.4 – Мобильный колёсный робот, вид сверху. Пунктиром показаны датчики линии: они под днищем и смотрят вниз, на поле

Контроллер – «мозг» робота. Это плата с **микроконтроллером** – маленьким компьютером, уместившимся в одной микросхеме. В сфере «Роботех» работают с платой **Arduino Uno**. Программу пишут на компьютере и загружают в плату по USB-кабелю, после этого плата работает сама, хоть от батарейки. По краям платы – **выводы** (их ещё называют пинами): к ним проводами подключают датчики и другие устройства. Контроллер питается напряжением 5 В (вольт – единица напряжения; что это такое, разберём в главе 5).

Датчики – «органы чувств». Датчик переводит то, что происходит вокруг, в электрический сигнал, понятный контроллеру.

- **Кнопка** – самый простой датчик, датчик касания: нажата или нет.

- **Ультразвуковой дальномер** (в наборах – модуль HC-SR04 с двумя «глазами») измеряет расстояние до препятствия. Он пищит неслышным для нас звуком и ждёт эха, как летучая мышь.
- **Датчик линии** светит вниз на поле и ловит отражённый свет: белое поле отражает много, чёрная линия – мало. Так робот отличает линию от фона.
- Бывают и другие: **фоторезистор** чувствует яркость света, **датчик температуры** – тепло.

Исполнительные устройства – «мышцы и голос» робота. Они выполняют то, что решил контроллер.

- **Мотор постоянного тока** (его ещё называют DC-мотором, от английского direct current – постоянный ток) крутит колесо. Сам по себе такой моторчик вертится очень быстро, тысячи оборотов в минуту, но слабо: зажми вал пальцами – и он встанет. Поэтому к нему приделывают **редуктор** – коробочку с шестерёнками, которая замедляет вращение и делает его сильнее. Популярный «жёлтый моторчик» из учебных наборов – это как раз мотор вместе с редуктором.
- **Сервомотор** (сервопривод) – мотор, который поворачивает вал **на заданный угол** и держит его. Обычный сервомотор из учебного набора поворачивается в пределах от 0 до 180°. Им поворачивают клешню захвата, «руль» или датчик.
- **Светодиод, пищалка, экранчик** тоже выполняют команды контроллера: светятся, пищат, показывают числа.

Драйвер моторов – посредник между контроллером и моторами. С выводов контроллера идёт слабый ток – его хватает, чтобы зажечь светодиод, но не чтобы крутить мотор. Драйвер похож на водопроводный кран: контроллер лишь поворачивает «ручку», а «вода» – сильный ток – течёт к мотору прямо от батарейки. В курсе сферы используют микросхему драйвера L293D, она управляет двумя моторами сразу. Почему мотор нельзя подключить прямо к плате, подробно разберём в главе 5.

Источник питания – «еда» робота: батарейный отсек (например, четыре батарейки по 1,5 В дают вместе 6 В) или аккумулятор. Без него не заработает ничего.

Шасси, колёса и опорное колесо – «скелет» и «ноги». На **шасси** (раме) крепятся все детали. Два ведущих колеса надеты на валы моторов. Третья точка опоры – **опорное колесо**: шарик или колёсико, которое свободно катится в любую сторону. Без него робот на двух колёсах клевал бы носом.

Посмотри, как детали связаны между собой. По одним проводам бегут сведения и команды, по другим – энергия.

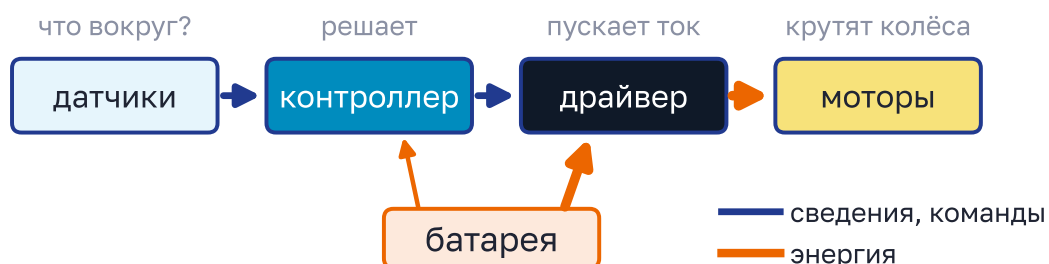


Рисунок 1.5 – Синие стрелки – сведения и команды, оранжевые – энергия от батареи

Датчик сообщает контроллеру, что видит. Контроллер решает и отдаёт команду драйверу. Драйвер берёт энергию у батареи и отдаёт её мотору столько, сколько велел контроллер. Обрати внимание на толстую оранжевую стрелку: к моторам энергия идёт мимо контроллера.

Таблица 1.1

Деталь	Что делает	«У человека»
Контроллер	выполняет программу, решает	мозг
Датчики	сообщают, что вокруг	глаза, уши, кожа
Моторы, сервомоторы	двигают колёса и захват	мышцы
Драйвер моторов	пропускает к моторам сильный ток	нервы к мышцам
Батарея	даёт энергию	еда
Шасси, колёса	держат и везут всё	скелет, ноги

Совет. Схему робота можно собрать без единой детали – в онлайн-симуляторе. В курсе сферы показывают **Tinkercad** (tinkercad.com): курс работает по ссылке на класс. Запасной вариант – **Wokwi** (wokwi.com). Шасси и колёс в симуляторе нет: ты собираешь только электрическую схему и пишешь программу. Колёса, шестерёнки и рычаги придётся держать в голове – вот для чего вторая половина главы.

Запомни. Провод от вывода Arduino прямо к мотору – ошибка, даже если в схеме на экране так «короче». Контроллер только командует, а кормит мотор батарея. Между контроллером и мотором всегда стоит драйвер.

1.3. Как робот едет и поворачивает

НА ПАЛЬЦАХ

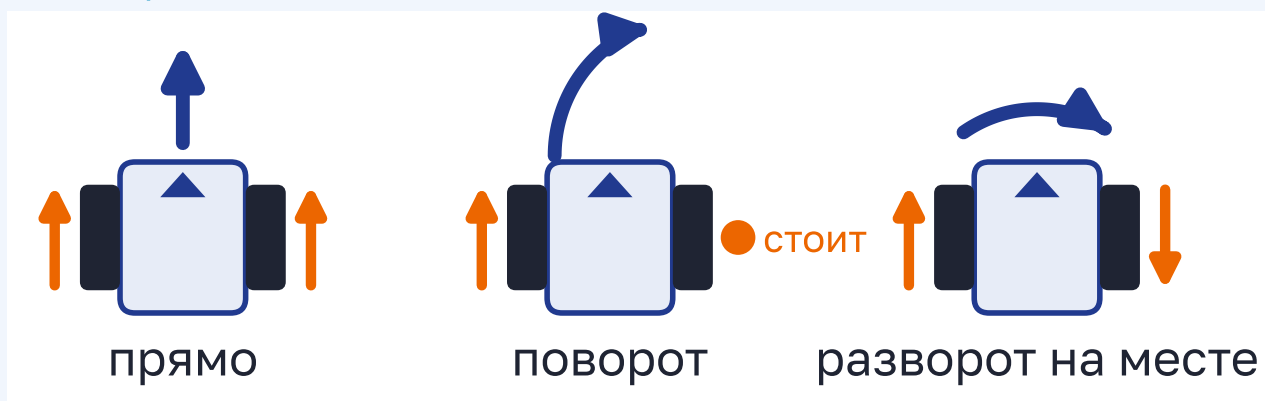


Рисунок 1.6 – Руля нет – поворачивают колёса, которые крутятся по-разному

Колёса крутятся одинаково – робот едет прямо. Одно отстаёт – робот сворачивает в его сторону. Колёса крутятся в разные стороны – робот вертится на месте.

У нашего робота нет руля. Каждое ведущее колесо крутит свой мотор, и контроллер управляет ими по отдельности. Такой способ называют **дифференциальным приводом** (от слова «разница»: робот поворачивает за счёт разницы в скоростях колёс). Так же поворачивают танк и трактор на гусеницах.

- **Оба колеса вперёд с одной скоростью** – робот едет прямо вперёд. Оба назад – прямо назад.
- **Одно колесо вперёд, другое стоит** – робот поворачивает вокруг стоящего колеса. Левое едет, правое стоит – поворот направо.
- **Одно колесо быстрее другого** – робот едет по дуге в сторону медленного колеса.
- **Колёса крутятся в разные стороны с одной скоростью** – разворот на месте вокруг середины между колёсами. Левое вперёд, правое назад – робот крутится направо, по часовой стрелке, если смотреть сверху.

Бывает и по-другому – как у автомобиля. Один мотор крутит задние колёса, а сервомотор поворачивает передние, как руль. Такого робота тоже можно собрать в Tinkercad. Но в курсе сферы главный – дифференциальный привод: он проще и умеет разворачиваться на месте.

Запомни. Робот поворачивает в сторону того колеса, которое крутится медленнее (или стоит, или едет назад). Сам себя проверяй так: встань, вытяни руки в стороны – это колёса. Шагни вперёд только левой стороной тела – и повернёшься направо.

1.4. Зубчатая передача и передаточное число

НА ПАЛЬЦАХ

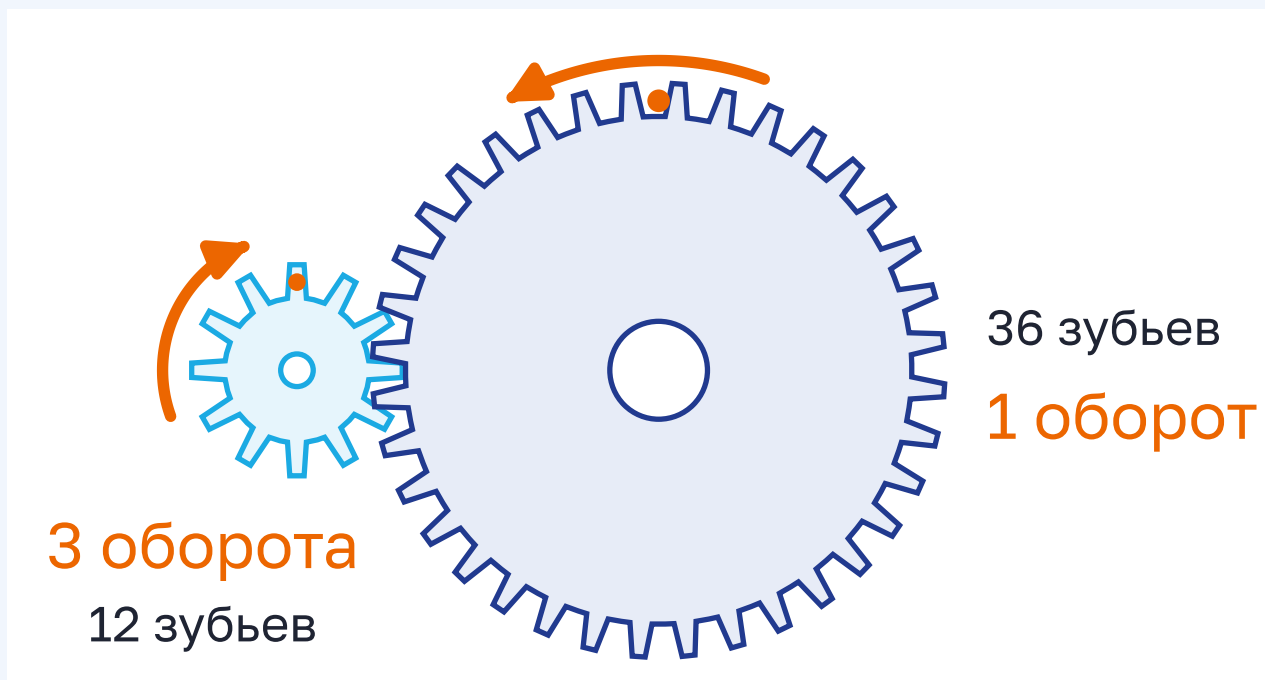


Рисунок 1.7 – Большая шестерёнка и маленькая. Маленькая успевает сделать три оборота, пока большая делает один

У маленькой 12 зубьев, у большой 36 – втрое больше. Поэтому маленькая крутится втрое быстрее. Зато большая крутится сильнее.

Зубчатое колесо, или **шестерня**, – колёсико с зубьями по краю. Две шестерни ставят так, чтобы зубья одной входили между зубьями другой. Это называется **зацеплением**. Крутишь одну – вертится и другая. Шестерню, которая сидит на валу мотора и крутит остальных, называют **ведущей**. Ту, которую крутят, – **ведомой**.

Почему у шестерён разная скорость? Посмотри на место, где они встречаются. Зубья проходят через него по одному: один зуб маленькой шестерни толкнул один зуб большой. Значит, сколько зубьев прошло у ведущей, столько же прошло и у ведомой. Ведущая с 12 зубьями за один свой оборот протолкнёт 12 зубьев. У ведомой их 36 – она повернётся только на 12 из 36, то есть на треть оборота. Чтобы большая сделала полный оборот, маленькой придётся обернуться три раза.

Передаточное число показывает, во сколько раз ведомая шестерня крутится медленнее ведущей. Словами: **передаточное число равно числу зубьев ведомой шестерни, делённому на число зубьев ведущей**. Буквами:

$$i = \frac{z_2}{z_1},$$

где i – передаточное число, z_1 – число зубьев ведущей шестерни, z_2 – ведомой. Чтобы узнать скорость ведомой, скорость ведущей делят на передаточное число:

$$n_2 = \frac{n_1}{i} = n_1 \cdot \frac{z_1}{z_2}.$$

Здесь n_1 и n_2 – скорости вращения ведущей и ведомой шестерён. Скорость вращения измеряют **оборотами в минуту** (об/мин) или **оборотами в секунду** (об/с). В минуте 60 секунд, поэтому 120 об/мин – это $120:60 = 2$ об/с.

Пример. На моторе шестерня с 12 зубьями, на колесе – с 36. $i = 36/12 = 3$. Мотор делает 90 об/мин, колесо – $90/3 = 30$ об/мин.

Что мы выигрываем. Шестерни не делают энергию из ничего. Если ведомая крутится в 3 раза медленнее, она крутит в 3 раза сильнее (если не считать трения в шестерёнках, которое немного съедает силу). Силу вращения – насколько сильно вал может провернуть то, что к нему приделано, – называют **крутящим моментом**. Правило простое: выиграл в силе – проиграл в скорости, и наоборот.

- **Передаточное число больше 1** (ведомая больше ведущей) – передача **понижающая**: колесо крутится медленнее, но сильнее. Коробку с понижающими передачами называют **редуктором**. У «жёлтого моторчика» из учебных наборов передаточное число редуктора 48: вал мотора крутится в 48 раз быстрее, чем выходная ось, на которую надевают колесо. При питании 6 В колесо такого моторчика делает около 200 об/мин.
- **Передаточное число меньше 1** (ведомая меньше ведущей) – передача **повышающая**: быстрее, но слабее. Так устроен велосипед. Спереди у педалей большая звёздочка, например на 44 зуба, сзади маленькая, на 11 зубьев. За один оборот педалей заднее колесо делает $44/11 = 4$ оборота. Звёздочки соединены цепью, а не зацеплены напрямую, но зубья считаются так же.

В какую сторону. Две шестерни в зацеплении всегда крутятся **в разные стороны**: одна по часовой стрелке, другая – против. Если между ними поставить третью, промежуточную шестерню, крайние будут крутиться в одну сторону.

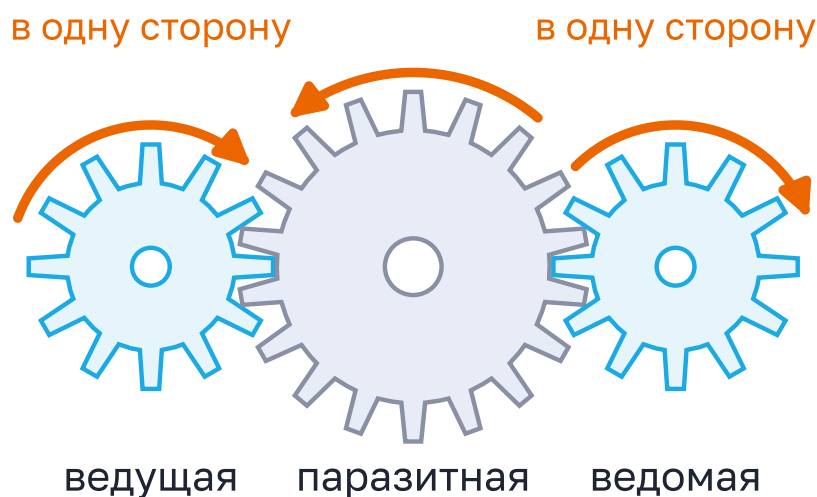


Рисунок 1.8 – Три шестерни в ряд. Средняя крутится против часовой стрелки, а крайние – обе по часовой

Промежуточная шестерня меняет только направление, а на скорость не влияет: сколько зубьев она приняла от первой, столько и отдала последней. Инженеры зовут её «паразитной» – вроде бы работает, а в счёте её нет. Цепь и ремень, в отличие от зацепления, направление сохраняют: педали и заднее колесо велосипеда крутятся в одну сторону.

Несколько ступеней. Если одной пары шестерён мало, ставят несколько пар подряд – это **ступени** редуктора. Две шестерни разного размера сидят на одном валу и крутятся вместе,

как одно целое. Большая из них принимает вращение от предыдущей ступени, маленькая передаёт следующей. Передаточное число всего редуктора равно **произведению** передаточных чисел ступеней. Две ступени по 3 дают $3 \cdot 3 = 9$, а не $3 + 3 = 6$. Так редуктор маленького моторчика и набирает свои 48.

Запомни. $i = z_{\text{ведомой}} / z_{\text{ведущей}}$. Скорость ведомой = скорость ведущей : i . Во сколько раз медленнее – во столько же раз сильнее. Ступени – перемножаем. Каждое зацепление меняет направление вращения, паразитная шестерня скорость не меняет.

Совет. Проверь себя здравым смыслом: вращение перешло с маленькой шестерни на большую – должно замедлиться. Если у тебя колесо закрутилось быстрее мотора, а шестерня на колесе больше, – ты перевернул дробь.

1.5. Рычаг

НА ПАЛЬЦАХ

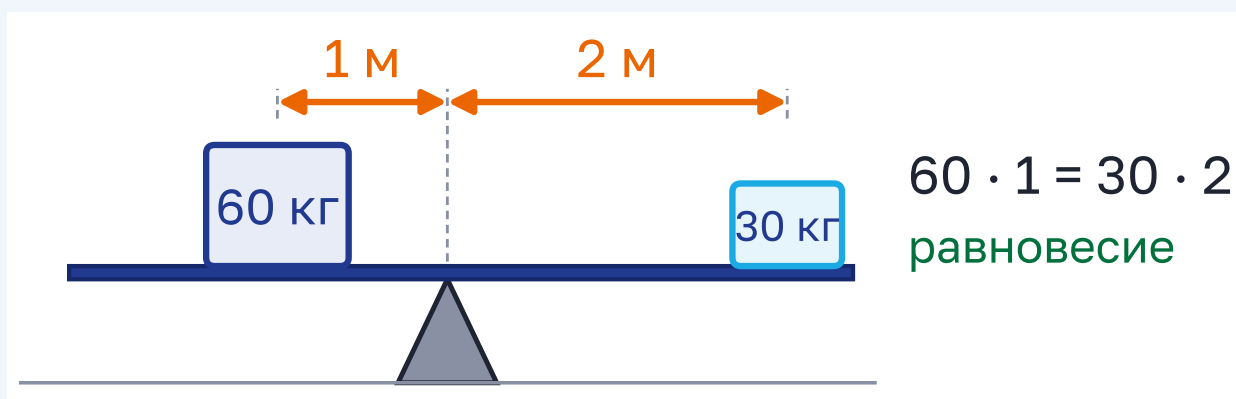


Рисунок 1.9 – Качели: взрослый сел ближе к середине, ребёнок – на самый край

Ребёнок легче взрослого в 2 раза, но сидит в 2 раза дальше от середины. Качели в равновесии.

Рычаг – твёрдая палка или планка, которая может поворачиваться вокруг одной точки – **опоры**. Качели во дворе, ножницы, открывашка, лом, которым отковыривают лёд с крыльца, – всё это рычаги.

Плечо – расстояние от опоры до места, где на рычаг давят. У качелей два плеча: от середины до взрослого и от середины до ребёнка.

Правило равновесия рычага: **сила, умноженная на плечо, с одной стороны равна силе, умноженной на плечо, с другой**. Буквами:

$$F_1 \cdot l_1 = F_2 \cdot l_2,$$

где F_1, F_2 – силы, которые давят на рычаг с двух сторон, а l_1, l_2 – их плечи. **Сила** – то, с чем одно тело толкает или тянет другое. Её измеряют в **ньютонах** (Н). Груз массой 1 кг давит на ладонь с силой около 10 Н. Если на рычаг давят только подвешенные грузы, можно не пере-

водить в ньютоны: вместо сил подставляй массы. Проверим качели с картинки:
 $60 \cdot 1 = 30 \cdot 2$, то есть $60 = 60$ – равновесие.

Раз сила умножается на плечо, давить на **длинное** плечо выгодно: маленькой силой уравниваешь большую. Поэтому лом берут за дальний конец, а дверь толкают у ручки, а не у петель.

Даром это не даётся: длинный конец проходит больший путь. Чтобы приподнять камень на 1 см ломом с плечами 10 см и 50 см, дальний конец придётся опустить на 5 см. Выиграли в силе в 5 раз – проиграли в пути в 5 раз. С шестерёнками было то же самое, только там вместо пути – скорость.

Рычаги в работе. Клешня захвата – это два рычага на общей оси. Качалка на валу сервомотора – тоже рычаг: чем она длиннее, тем дальше отводит тягу, но тем слабее давит. Стрела подъёмного крана или погрузчика – рычаг с опорой на башне. Чтобы кран не опрокинулся, на другом конце стрелы висит **противовес** – тяжёлый груз, который уравнивает поднимаемый.

Запомни. Сила $\times$ плечо = сила $\times$ плечо. Лёгкий груз уравнивает тяжёлый, если висит дальше от опоры – во столько же раз дальше, во сколько он легче. Плечо меряют от опоры, а не от края рычага.

Совет. Плечи с двух сторон подставляй в одинаковых единицах: оба в сантиметрах или оба в метрах. Тогда ответ выйдет в тех же единицах, и переводить ничего не придётся.

1.6. Разбираем задачи

В задачах с выбором ответ – строка из цифр выбранных пунктов, например «246». Удобнее записывать цифры по возрастанию, а как нужно на отборе – всегда написано в условии задачи.

Задача 1.1 Деталь и её работа

разминка · устройство мобильного робота, сопоставление

В кружке собирают робота-разведчика для макета лунной базы. На столе лежат детали:

1 – плата Arduino Uno; 2 – ультразвуковой дальномер; 3 – драйвер моторов; 4 – мотор с редуктором; 5 – сервомотор; 6 – батарейный отсек; 7 – датчик линии.

Что делает деталь:

- **А** – измеряет расстояние до стены;
- **Б** – выполняет программу и решает, что делать;
- **В** – крутит колесо;
- **Г** – даёт энергию всем устройствам;
- **Д** – поворачивает клешню захвата на заданный угол и держит её;
- **Е** – отличает чёрную линию от белого поля;

- **Ж** – получает слабую команду от контроллера и пропускает к моторам сильный ток от батареи.

Для каждой работы А, Б, В, Г, Д, Е, Ж запиши номер детали. Ответ – строка из семи цифр.

Дано: 7 деталей (номера 1–7), 7 работ (буквы А–Ж).

Найти: номера деталей в порядке А, Б, В, Г, Д, Е, Ж.

Решение

Шаг 1. Сначала то, что узнаётся с ходу. «Измеряет расстояние» – дальномер, само слово так и собрано: «даль» + «мерить». А – 2. «Крутит колесо» – мотор с редуктором. В – 4. «Даёт энергию» – батарейный отсек. Г – 6.

Шаг 2. Датчик линии. Работа Е – отличать чёрное от белого. Это датчик линии, он ловит отражённый от поля свет. Е – 7.

Шаг 3. Мотор или сервомотор. Остались два мотора. Работа Д – повернуть на **заданный угол** и держать. Это умеет сервомотор, а обычный мотор просто крутится, пока на него подают ток. Д – 5.

Шаг 4. Мозг и посредник. Остались плата и драйвер. «Выполняет программу» – плата Arduino, Б – 1. «Пропускает к моторам сильный ток по команде контроллера» – драйвер, Ж – 3.

Шаг 5. Собираем строку. А – 2, Б – 1, В – 4, Г – 6, Д – 5, Е – 7, Ж – 3. Каждая цифра от 1 до 7 встретилась по одному разу – ни одна деталь не потерялась и не досталась двум работам.

 **Типичная ошибка.** Отдать работу Ж плате Arduino: «командует же контроллер». Командует он, но сильный ток идёт мимо него – через драйвер. Вторая частая путаница – мотор и сервомотор в работах В и Д. Ищи в условии слова «на заданный угол»: это про сервомотор.

Ответ: 2146573.

Своя задача по программе сферы, модуль 2

Реши сам

1.1. Детали: 1 – ведущее колесо; 2 – шасси; 3 – опорное колесо-шарик; 4 – кнопка; 5 – светодиод.

Работы: **А** – подаёт контроллеру сигнал «старт»; **Б** – держит на себе все остальные детали; **В** – светится и показывает, в каком режиме робот; **Г** – третья точка опоры, чтобы робот не клевал носом; **Д** – толкает робота, отталкиваясь от пола; **Е** – измеряет температуру воздуха.

Для каждой работы А–Е запиши номер детали. Если ни одна деталь из списка эту работу не делает, поставь 0. Ответ – строка из шести цифр.

1.2. Выбери все **датчики**: 1 – кнопка; 2 – ультразвуковой дальномер; 3 – сервомотор; 4 – фоторезистор; 5 – пищалка; 6 – датчик линии; 7 – драйвер моторов; 8 – датчик температуры. Ответ – цифры выбранных пунктов по возрастанию.

Задача 1.2 Медленнее, но сильнее

разминка · зубчатая передача, передаточное число, направление вращения

На вал мотора лунохода-макета надета шестерня с 12 зубьями. Она сцеплена с шестерней на 36 зубьев, которая сидит на оси колеса. Вал мотора делает 180 об/мин. Сколько оборотов в минуту делает колесо? В ту же сторону, что и вал мотора, или в обратную? В ответ введи число оборотов в минуту.

Дано: $z_1 = 12$ (ведущая, на моторе); $z_2 = 36$ (ведомая, на колесе); $n_1 = 180$ об/мин.

Найти: n_2 – ?; направление.

Решение

Шаг 1. Кто кого крутит. Мотор крутит свою шестерню – она ведущая, $z_1 = 12$. Шестерня колеса – ведомая, $z_2 = 36$.

Шаг 2. Передаточное число. $i = z_2/z_1 = 36/12 = 3$. Число больше 1 – передача понижающая, колесо будет крутиться медленнее мотора.

Шаг 3. Скорость колеса. $n_2 = n_1/i = 180/3 = 60$ об/мин.

Шаг 4. Направление. Шестерни сцеплены напрямую, без промежуточной. В одном зацеплении направление меняется: колесо крутится в обратную сторону.

60 об/мин – это один оборот в секунду. Колесо «жёлтого моторчика» диаметром 6,5 см за оборот проезжает около 20 см, так что вдоль школьной парты длиной 120 см робот проползёт секунд за шесть. Для учебного робота – обычный неторопливый ход.

 **Типичная ошибка.** Перевернуть дробь и умножить: $180 \cdot 3 = 540$ об/мин. Шестерня колеса втрое больше – значит, крутится медленнее, а не быстрее. Если сомневаешься, сосчитай зубья «по штучке»: за один оборот мотора через зацепление проходит 12 зубьев, а колесу на полный оборот нужно 36.

Ответ: 60 об/мин, в обратную сторону.

Своя задача по программе сферы, модуль 6

Реши сам

1.3. На валу мотора шестерня с 10 зубьями, на оси колеса – с 50 зубьями, они сцеплены напрямую. Мотор делает 250 об/мин. Сколько оборотов в минуту делает колесо?

1.4. Ведущая шестерня с 15 зубьями делает 120 об/мин. Сколько зубьев должно быть у ведомой шестерни, чтобы она делала 40 об/мин?

Задача 1.3 Правда или нет

устройство робота, дифференциальный привод, передачи, рычаг, выбор верных утверждений

Выбери все верные утверждения.

1. Контроллер Arduino выполняет программу, но крутить мотор колеса прямо со своих выводов не может: ток с выводов для этого слишком слабый.
2. Датчик линии – исполнительное устройство: он двигает робота вдоль линии.
3. Если у робота с двумя моторами левое колесо крутится вперёд, а правое с той же скоростью назад, робот разворачивается на месте.
4. Редуктор замедляет вращение колеса, зато колесо крутится сильнее, чем вал мотора.
5. Две шестерни в зацеплении всегда крутятся в одну сторону.
6. Если ведущая шестерня меньше ведомой, ведомая крутится быстрее ведущей.
7. Сервомотор из учебного набора умеет повернуть вал на заданный угол и удерживать его.
8. Рычаг даёт выигрыш в силе, если давить на его короткое плечо.

Ответ – цифры верных утверждений по возрастанию.

Дано: 8 утверждений.

Найти: номера верных.

Решение

Шаг 1. Утверждение 1 – верно. С выводов контроллера идёт слабый ток, его хватает на светодиод. Мотору нужен гораздо больший, поэтому между ними ставят драйвер.

Шаг 2. Утверждение 2 – неверно. Датчик ничего не двигает, он только сообщает контроллеру, что видит под собой: линию или поле. Двигают робота моторы.

Шаг 3. Утверждение 3 – верно. Колёса крутятся в разные стороны с одинаковой скоростью – это разворот на месте (раздел 1.3).

Шаг 4. Утверждение 4 – верно. Редуктор – понижающая передача: во сколько раз медленнее, во столько же раз сильнее.

Шаг 5. Утверждение 5 – неверно. В зацеплении шестерни крутятся в **разные** стороны. В одну сторону крутятся крайние шестерни, если между ними стоит промежуточная, – но друг с другом они не сцеплены.

Шаг 6. Утверждение 6 – неверно. У меньшей ведущей шестерни зубьев меньше, за один её оборот большая успевает повернуться лишь на часть круга. Ведомая крутится медленнее.

Шаг 7. Утверждение 7 – верно. Это и отличает сервомотор от простого мотора.

Шаг 8. Утверждение 8 – неверно. Выигрыш в силе даёт **длинное** плечо: лом берут за дальний конец. Надавишь на короткое – придётся давить сильнее, чем весит груз.

Шаг 9. Собираем ответ. Верные: 1, 3, 4, 7.

 **Типичная ошибка.** Проскочить мимо маленьких слов. В утверждении 5 всё решает «всегда», в утверждении 8 – «короткое». Читай каждое утверждение до точки и спрашивай себя: «А бывает ли по-другому?» Ещё одна ловушка – отметить только одно-два самых очевидных пункта. В задаче «выбери все» верных может оказаться сколько угодно.

Реши сам

1.5. Выбери все верные утверждения.

1. Робот-пылесос – мобильный робот: он сам ездит по квартире и объезжает мебель по сигналам датчиков.
2. Промышленный робот-манипулятор, прикрученный к полу цеха, – не мобильный робот: сам по цеху он не ездит.
3. Машинка на радиоуправлении – автономный робот.
4. Гусеницы помогают роботу проходить по рыхлому снегу и песку, где колёса буксуют.
5. «Луноход-1» – первый в мире планетоход, успешно работавший на другом небесном теле; на Луну его доставила станция «Луна-17» в 1970 году.
6. У «Лунохода-1» было четыре колеса и один общий мотор на всех.
7. Квадрокоптер, который сам летит по заданному маршруту, тоже можно назвать мобильным роботом.

Ответ – цифры верных утверждений по возрастанию.

1.6. У робота два ведущих колеса, у каждого свой мотор. Команды моторам:

1 – левый вперёд, правый вперёд, скорости одинаковые; 2 – левый вперёд, правый стоит; 3 – левый назад, правый вперёд, скорости одинаковые; 4 – оба назад, скорости одинаковые; 5 – оба вперёд, левый быстрее правого; 6 – левый стоит, правый вперёд.

Движения робота (смотрим сверху): **А** – едет прямо назад; **Б** – разворачивается на месте против часовой стрелки; **В** – поворачивает направо вокруг правого колеса; **Г** – едет прямо вперёд; **Д** – едет вперёд по дуге вправо.

Для каждого движения А, Б, В, Г, Д запиши номер команды. Одна команда лишняя. Ответ – строка из пяти цифр.

Задача 1.4 Редуктор из двух ступеней

двухступенчатый редуктор, передаточное число, перевод единиц

Моторчик марсохода-макета делает 6000 об/мин – слишком быстро и слабо. Между мотором и колесом поставили двухступенчатый редуктор (рисунок ниже). На валу мотора шестерня с 10 зубьями. Она крутит шестерню с 40 зубьями. На одном валу с ней сидит шестерня с 12 зубьями, она крутит шестерню с 60 зубьями на оси колеса. Сколько оборотов **в секунду** делает колесо? Крутится оно в ту же сторону, что вал мотора, или в обратную?

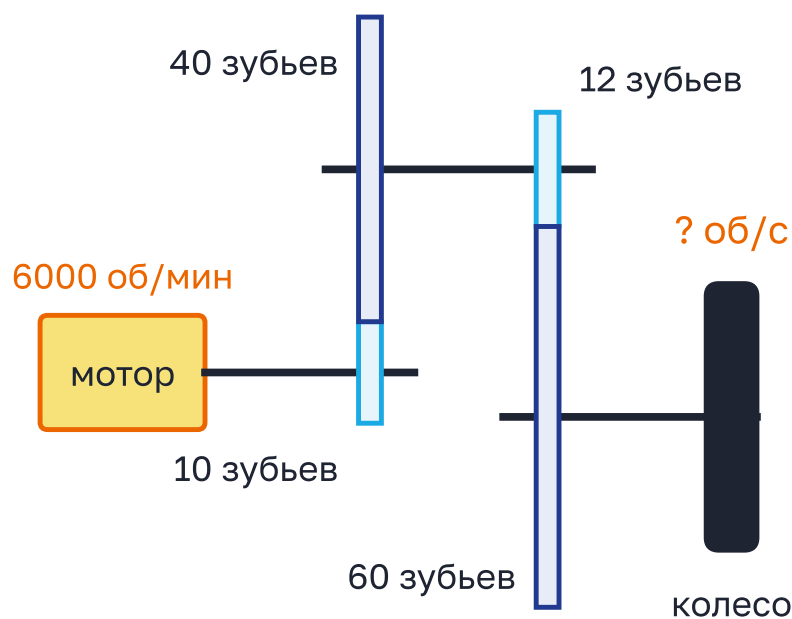


Рисунок 1.10 – Двухступенчатый редуктор, вид сбоку. Шестерни 40 и 12 сидят на одном валу и крутятся вместе

Дано: $n_m = 6000$ об/мин; ступень 1: 10 → 40 зубьев; ступень 2: 12 → 60 зубьев.

Найти: скорость колеса в об/с; направление.

Решение

Шаг 1. Разбираем редуктор на ступени. В первой ступени ведущая – шестерня мотора (10 зубьев), ведомая – 40 зубьев. Во второй ведущая – шестерня с 12 зубьями, ведомая – с 60. Шестерня с 12 зубьями сидит на одном валу с шестерней на 40 зубьев и крутится вместе с ней.

Шаг 2. Передаточное число первой ступени. $i_1 = 40/10 = 4$.

Шаг 3. Передаточное число второй ступени. $i_2 = 60/12 = 5$.

Шаг 4. Весь редуктор. Ступени перемножаем: $i = i_1 \cdot i_2 = 4 \cdot 5 = 20$.

Шаг 5. Скорость колеса в об/мин. $n_k = 6000/20 = 300$ об/мин.

Шаг 6. Переводим в обороты в секунду. В минуте 60 секунд: $300/60 = 5$ об/с.

Шаг 7. Направление. Зацеплений два: 10 с 40 и 12 с 60. Первое поворачивает направление в обратную сторону, второе – ещё раз, обратно. Колесо крутится в ту же сторону, что и вал мотора.

Можно проверить по шагам, без общего передаточного числа: вал с шестернями 40 и 12 крутится со скоростью $6000/4 = 1500$ об/мин, колесо – $1500/5 = 300$ об/мин. Пять оборотов колеса в секунду – бойкий робот, но глаз за колесом ещё успевает, в отличие от вала мотора со ста оборотами в секунду.

⚠ **Типичная ошибка.** Сложить передаточные числа: $4 + 5 = 9$, и получить $6000/9 = 667$ об/мин. Ступени работают одна за другой: первая замедлила в 4 раза, вторая замедляет уже замедленное ещё в 5 раз – это умножение. И второе – забыть перевести минуты в секунды и ввести 300: в условии просили обороты в секунду.

Ответ: 5 об/с, в ту же сторону, что и вал мотора.

Своя задача по программе сферы, модуль 6

Реши сам

1.7. Три шестерни стоят в ряд, каждая сцеплена с соседней: ведущая на 16 зубьев, промежуточная на 24 зуба и последняя на 48 зубьев. Ведущая делает 90 об/мин. Сколько оборотов в минуту делает последняя шестерня? В ту же сторону, что ведущая, или в обратную?

1.8. «Жёлтый моторчик» из учебного набора: передаточное число редуктора 48, колесо делает 200 об/мин. Сколько оборотов **в секунду** делает вал самого мотора, до редуктора?

Задача 1.5 Кран с противовесом

рычаг, правило равновесия, противовес

На макете лунной базы работает робот-кран. Его стрела – рычаг с опорой на башне. На коротком конце стрелы, в 25 см от опоры, закреплён противовес массой 12 кг. На длинном конце висит крюк, его можно передвигать вдоль стрелы. а) На каком расстоянии от опоры надо повесить на крюк ящик массой 5 кг, чтобы стрела была в равновесии? б) Потом нужно поднять ящик массой 8 кг. Куда передвинуть крюк? Массой самой стрелы и крюка пренебреги. Ответы – в сантиметрах.

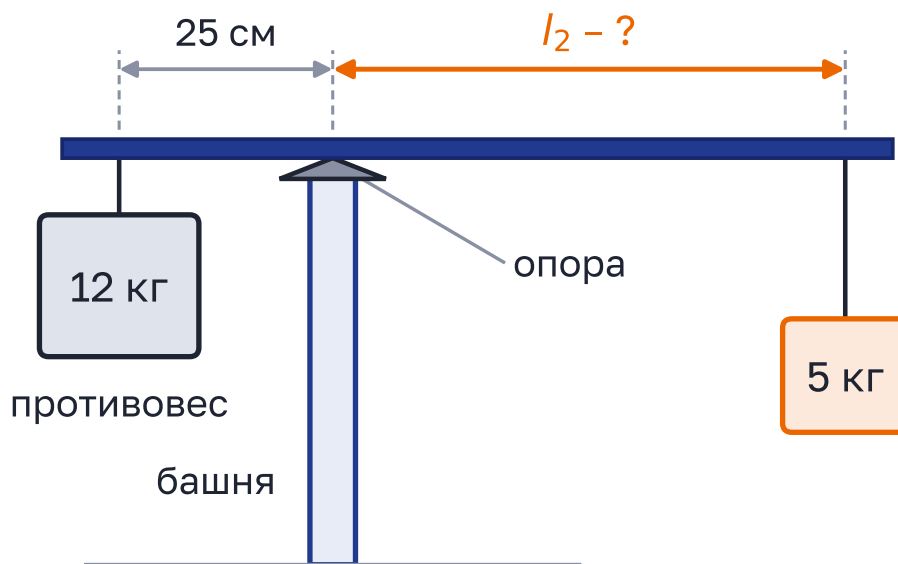


Рисунок 1.11 – Стрела крана – рычаг. Слева противовес, справа ящик на крюке

Дано: $m_1 = 12$ кг, $l_1 = 25$ см; а) $m_2 = 5$ кг; б) $m_2 = 8$ кг.

Найти: l_2 – ? в обоих случаях.

Решение

Шаг 1. Правило рычага. На стрелу давят только подвешенные грузы, поэтому вместо сил можно подставлять массы: $m_1 \cdot l_1 = m_2 \cdot l_2$.

Шаг 2. Левая сторона. Противовес и его плечо не меняются: $12 \cdot 25 = 300$ (кг·см). Это число должно получиться и справа.

Шаг 3. Ящик 5 кг. $5 \cdot l_2 = 300$, значит, $l_2 = 300/5 = 60$ см.

Шаг 4. Ящик 8 кг. $8 \cdot l_2 = 300$, $l_2 = 300/8 = 37,5$ см. Ящик тяжелее – крюк нужно придвинуть к опоре.

Проверка. а) $5 \cdot 60 = 300$; б) $8 \cdot 37,5 = 300$. Обе стороны дают по 300. Кстати, на настоящей Луне всё весит примерно в 6 раз меньше, чем на Земле. Но и противовес облегчает во столько же раз, поэтому точки равновесия останутся на тех же местах.

 **Типичная ошибка.** Решить «наоборот»: раз ящик тяжелее, отодвинуть его дальше. Но дальше от опоры груз сильнее тянет стрелу вниз – тяжёлый ящик вешают **ближе** к опоре. Ещё ошибаются, когда плечо меряют от края стрелы или от противовеса, а не от опоры.

Ответ: а) 60 см; б) 37,5 см.

Своя задача по программе сферы, модуль 9

Реши сам

1.9. Линейка-рычаг подвешена за середину. Слева от середины, в 20 см от неё, висят три грузика по 50 г. Справа нужно повесить два таких же грузика вместе, в одной точке. На каком расстоянии от середины их повесить, чтобы линейка висела горизонтально? Ответ в сантиметрах.

1.10. Каждая половинка клешни робота – рычаг с опорой на оси. Тяга сервомотора давит на короткое плечо, в 2 см от оси, с силой 15 Н. Губка клешни, которая сжимает кубик, находится в 6 см от оси. С какой силой губка давит на кубик? Ответ в ньютонах.

Задача 1.6* Настоящий размер колеса

передаточное число, путь за оборот колеса, калибровка

Робот ездит на «жёлтых моторчиках»: передаточное число редуктора 48. На валу мотора стоит счётчик оборотов – он считает обороты вала мотора, а не колеса. По паспорту диаметр колеса 65 мм. Но шина мягкая и немного проминается под весом робота, поэтому катится колесо как будто меньшего диаметра.

Чтобы узнать настоящий диаметр, робота пустили прямо по коридору. Счётчик показал 960 оборотов мотора, а по рулетке робот проехал 4,02 м. Число π считай равным 3,14.

а) Каков настоящий диаметр колеса? Округли до целых миллиметров. б) Сколько оборотов мотора надо задать в программе, чтобы робот проехал 2,5 м? Округли до целых. В ответ введи число из пункта б.

Дано: $i = 48$; $N_m = 960$ оборотов мотора; путь $S = 4,02$ м; паспортный диаметр 65 мм; $\pi = 3,14$.

Найти: а) настоящий диаметр D ; б) число оборотов мотора для пути 2,5 м.

Решение

Шаг 1. Что происходит. Мотор крутится, редуктор замедляет вращение в 48 раз, колесо катится. За один оборот колесо проходит путь, равный **длине своей окружности**, – как если бы обод размотали и положили на пол. Длина окружности равна $\pi \cdot D$ (подробнее о пути за оборот – в главе 2). Цепочка такая: обороты мотора → обороты колеса → путь за один оборот → диаметр.



Рисунок 1.12 – Колесо сделало один оборот. Метка на ободке снова внизу, а за колесом осталась «размотанная» полоска длиной $\pi \cdot D$

Шаг 2. Обороты колеса. Колесо крутится в 48 раз медленнее мотора: $960/48 = 20$ оборотов.

Шаг 3. Путь в миллиметрах. Диаметр нужен в мм, поэтому путь тоже переводим: $4,02$ м $= 4,02 \cdot 1000 = 4020$ мм.

Шаг 4. Путь за один оборот колеса. $4020/20 = 201$ мм.

Шаг 5. Диаметр. $\pi \cdot D = 201$ мм, значит, $D = 201/3,14 = 64,01$ мм ≈ 64 мм. Шина промялась на миллиметр по сравнению с паспортом – мелочь, но сейчас увидим, во что она обходится.

Шаг 6. Обороты колеса на 2,5 м. $2,5$ м $= 2500$ мм. За оборот колесо проходит $3,14 \cdot 64 = 200,96$ мм. Оборотов колеса: $2500/200,96 = 12,44$.

Шаг 7. Обороты мотора. Мотор крутится в 48 раз быстрее колеса: $12,44 \cdot 48 = 597,1 \approx 597$ оборотов.

Пункт б можно решить и без π : путь за оборот мы уже измерили в шаге 4, это 201 мм. Тогда $2500/201 \cdot 48 = 597,0$ – те же 597.

А если поверить паспорту? Для колеса 65 мм вышло бы $2500/(3,14 \cdot 65) \cdot 48 = 587,9 \approx 588$ оборотов. С такой программой робот проедет $588/48 \cdot 201 = 2462$ мм и не дотянет до цели почти 4 см. На соревнованиях этого хватит, чтобы промахнуться мимо клетки финиша. Поэтому колёса и проверяют заездом, а не по надписи на коробке.

 **Типичная ошибка.** Забыть про редуктор и считать, что колесо сделало все 960 оборотов, – тогда колесо выйдет диаметром около 1,3 мм, со спичечную головку. Такой ответ сам себя выдаёт. Вторая – поделить на π весь путь 4020 мм, забыв разделить его на 20 оборотов: выйдет колесо диаметром 1280 мм, ростом с первоклассника. Третья – подставить в пункт б паспортные 65 мм и ввести 588.

Ответ: а) 64 мм; б) 597 оборотов мотора.

По мотивам задачи первого отборочного этапа НТО 2024/25, профиль «Интеллектуальные робототехнические системы», инженерный тур, «Калибровка одометрии»

Реши сам

1.11.* У робота редуктор с передаточным числом 30, счётчик оборотов стоит на валу мотора. Мотор сделал 900 оборотов, и робот по рулетке проехал 4,15 м. Найди настоящий диаметр колеса в миллиметрах, округли до целых. Число π считай равным 3,14.

1.12.* Мотор соединён с колесом парой шестерён: на моторе 10 зубьев, на колесе 30. Диаметр колеса 60 мм. Сколько оборотов должен сделать мотор, чтобы робот проехал 1 м? Округли до целых, π считай равным 3,14.

Движение: скорость, путь, время

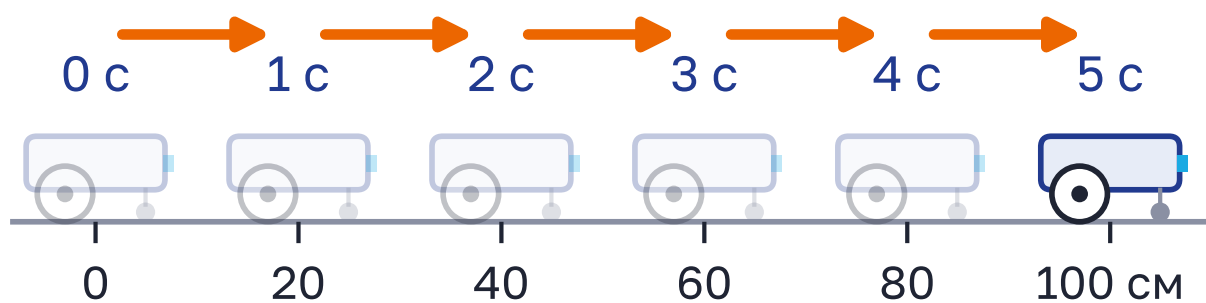


Робот на полигоне всё время куда-то едет, и задачи про него начинаются со слов «сколько» и «где»: сколько секунд ехать до стенки, сколько оборотов сделает колесо, где робот встанет после команды «стоп». Формула тут одна, и она простая. Баллы теряют на единицах: в одной задаче встречаются миллиметры, метры, миллисекунды и обороты в минуту, и стоит одну не перевести – ответ уезжает в десять или сто раз. Вторая половина главы – про то, почему робот не встаёт как вкопанный и почему на гладком полу колёса буксуют.

2.1. Скорость, путь и время

НА ПАЛЬЦАХ

каждую секунду – ещё 20 см



за 5 секунд: $5 \cdot 20 = 100$ см

Рисунок 2.1 – Робот едет с одной и той же скоростью – 20 см каждую секунду

Скорость – сколько робот проезжает за одну секунду. Прошло 5 секунд – значит, пять раз по 20 см.

Когда робот едет без рывков, каждую секунду он проезжает одно и то же расстояние. Это расстояние за одну секунду называют **скоростью**. У робота на картинке скорость 20 сантиметров в секунду, коротко – 20 см/с. Косая черта читается как «в»: «сантиметров в секунду». Движение, при котором скорость не меняется, называют **равномерным**.

Путь – сколько робот проехал, **время** – сколько длилась поездка. Для равномерного движения правило такое: **путь равен скорости, умноженной на время**. Буквами:

$$s = v \cdot t,$$

где s – путь, v – скорость, t – время. Из этой формулы получаются ещё две. Нужно время – путь делим на скорость. Нужна скорость – путь делим на время:

$$t = \frac{s}{v}, \quad v = \frac{s}{t}.$$

Где умножать, а где делить, легко проверить по единицам. Сантиметры в секунду, умноженные на секунды, дают сантиметры – это путь. Сантиметры, делённые на сантиметры в секунду, дают секунды – это время. Если после действия единицы не сложились в то, что ищешь, ты выбрал не то действие.

Три примера для робота со скоростью 20 см/с:

- за 5 с он проедет $20 \cdot 5 = 100$ см;
- на 1,5 м ему нужно: сначала переводим $1,5 \text{ м} = 150 \text{ см}$, потом $150 : 20 = 7,5$ с;
- другой робот проехал 90 см за 6 с – его скорость $90 : 6 = 15$ см/с.

Средняя скорость. Всю дорогу с одной скоростью робот едет редко: он разгоняется, сбрасывает скорость перед поворотом, ждёт на перекрёстке. Тогда считают **среднюю скорость** – весь путь делят на всё время, вместе с остановками:

$$v_{\text{ср}} = \frac{\text{весь путь}}{\text{всё время}}.$$

Пример. Робот проехал 120 см за 6 с (это 20 см/с), 2 с стоял у перекрёстка, потом проехал 80 см за 2 с (это 40 см/с). Весь путь $120 + 80 = 200$ см, всё время $6 + 2 + 2 = 10$ с, средняя скорость $200 : 10 = 20$ см/с. Рука так и тянется сложить 20 и 40 и поделить пополам. Получится 30 см/с – неверно: медленно робот ехал вдвое дольше, чем быстро, да ещё и стоял.

Два робота. Если роботы едут навстречу друг другу, каждую секунду расстояние между ними уменьшается на **сумму** их скоростей: роботы со скоростями 20 см/с и 30 см/с сближаются на 50 см за секунду. Если один догоняет другого, расстояние сокращается на **разность**: $30 - 20 = 10$ см за секунду. Стоят такие роботы в 2 м друг от друга и едут навстречу – встретятся через $200 : 50 = 4$ с. А если медленный уехал вперёд на 2 м, быстрый догонит его за $200 : 10 = 20$ с.

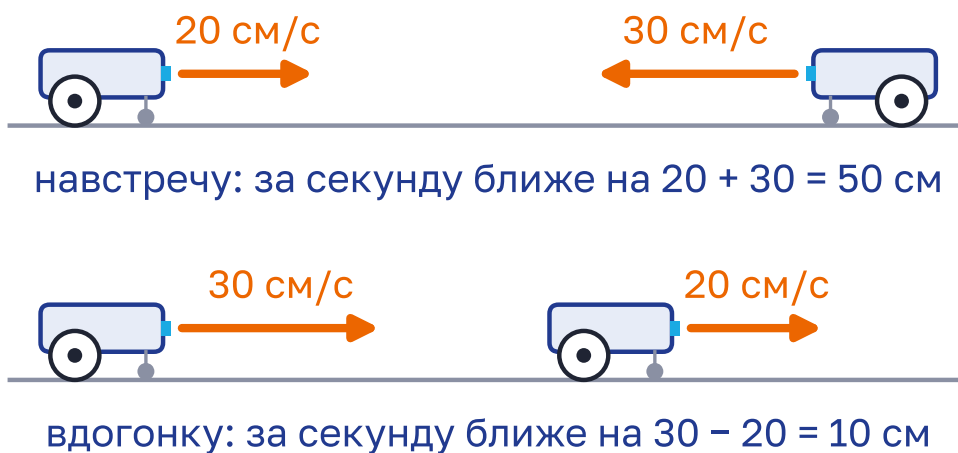


Рисунок 2.2 – Навстречу скорости складываются, вдогонку – вычитаются

Запомни. $s = v \cdot t$. Ищешь время – путь дели на скорость, ищешь скорость – путь дели на время. Средняя скорость – весь путь, делённый на всё время вместе с остановками; две скорости пополам не складывают.

Совет. Прежде чем умножать или делить, посмотри, в чём дан путь и в чём скорость. Сантиметры идут в паре с см/с, метры – с м/с. Не совпало – сначала переведи (как – в следующем разделе).

2.2. Единицы скорости и их перевод

НА ПАЛЬЦАХ

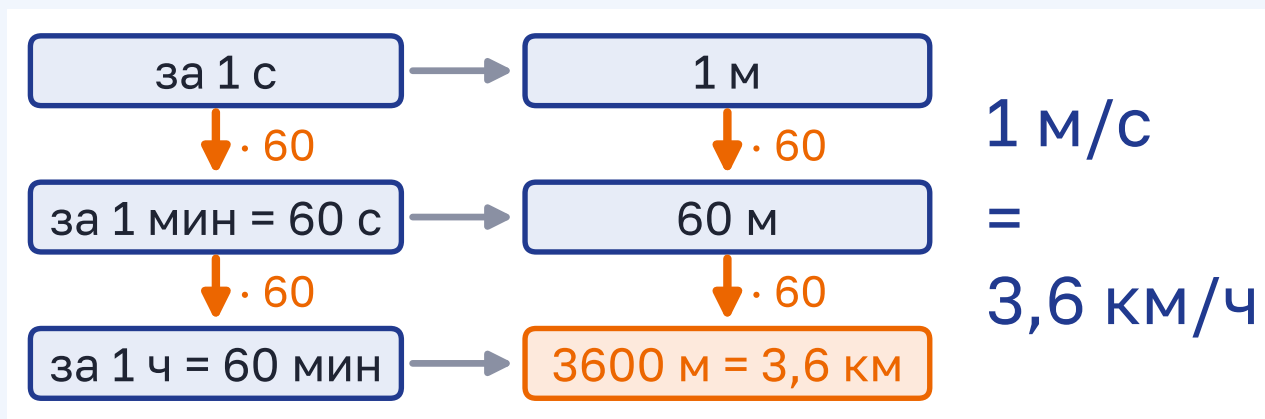


Рисунок 2.3 – Растягиваем одну секунду до целого часа

Робот, который за секунду проезжает метр, за минуту проедет 60 м, а за час – 3600 м, то есть 3,6 км. Поэтому 1 м/с – то же самое, что 3,6 км/ч.

У автобуса на спидометре скорость в километрах в час (км/ч), у робота в задаче – в сантиметрах в секунду (см/с), а у мотора в паспорте – в оборотах в минуту (об/мин). Переводить можно по шагам или готовыми множителями.

По шагам. Отдельно переводим путь, отдельно время. Возьмём 30 см/с. Путь: 30 см = 0,3 м, значит, это 0,3 м/с. А сколько это километров в час? За час, то есть за 3600 с, робот проедет $0,3 \cdot 3600 = 1080 \text{ м} = 1,08 \text{ км}$. Значит, 1,08 км/ч.

Готовыми множителями. Когда переводишь часто, удобнее запомнить несколько равенств:

- $1 \text{ м/с} = 100 \text{ см/с} = 1000 \text{ мм/с}$;
- $1 \text{ м/с} = 3,6 \text{ км/ч}$: из м/с в км/ч – умножаем на 3,6, из км/ч в м/с – делим на 3,6;
- $1 \text{ об/с} = 60 \text{ об/мин}$: из оборотов в минуту в обороты в секунду – делим на 60;
- $1 \text{ с} = 1000 \text{ мс}$ (миллисекунд): в Arduino команда `delay(1000)` ждёт одну секунду, `delay(250)` – четверть секунды.

Сравним скорости из жизни. Пешеход идёт со скоростью около 5 км/ч – это $5 : 3,6 \approx 1,4 \text{ м/с}$. Школьный автобус на трассе едет 60 км/ч, то есть $60 : 3,6 \approx 16,7 \text{ м/с}$. Учебный робот со своими 30 см/с (1,08 км/ч) медленнее пешехода больше чем в четыре раза. Ему и не надо спешить: на поле размером с комнату дорожка точность.

Со временем осторожнее: в минуте 60 секунд, а не 100. Поэтому 1,5 мин – это 90 с, и никак не 150 с. 2 мин 15 с – это $2 \cdot 60 + 15 = 135 \text{ с}$.

Запомни. $1 \text{ м/с} = 100 \text{ см/с} = 3,6 \text{ км/ч}$. $1 \text{ об/с} = 60 \text{ об/мин}$. $1 \text{ с} = 1000 \text{ мс}$. Чем мельче единица, тем больше число: $0,3 \text{ м/с} = 30 \text{ см/с} = 300 \text{ мм/с}$.

Совет. После перевода спроси себя, может ли так быть. Учебный робот не ездит 30 м/с: это 108 км/ч, быстрее машины на трассе. Такое число почти всегда значит, что сантиметры забыли перевести в метры.

2.3. Колесо: путь за один оборот

НА ПАЛЬЦАХ

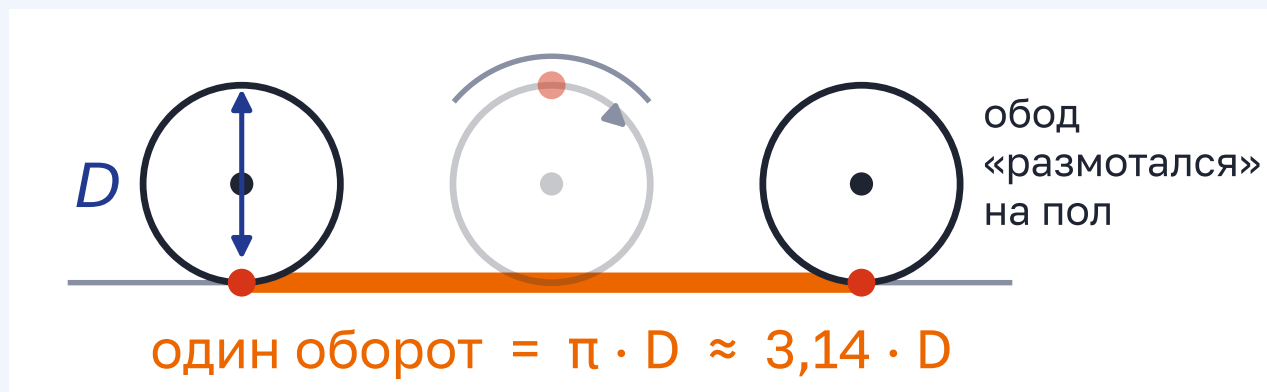


Рисунок 2.4 – Колесо с каплей краски на ободе прокатилось на один оборот

Капля снова коснулась пола, когда колесо прокатилось на длину своей окружности. Столько колесо проезжает за каждый оборот: $\pi \cdot D$.

Спидометра у робота нет. Зато он может считать обороты колёс – для этого на колесо или на вал мотора ставят датчик оборотов, **энкодер**. А программа знает, с какой скоростью крутится мотор. Остаётся связать обороты с путём.

Обод – это край колеса, та часть, что касается пола. Если колесо катится **без проскальзывания**, точки обода по очереди прикладываются к полу, и за один оборот весь обод как будто размотался и лёг на пол ленточкой. Длина этой ленточки – длина окружности колеса:

$$C = \pi \cdot D,$$

где C – путь за один оборот (длина окружности), D – **диаметр** колеса (от края до края через центр), а $\pi \approx 3,14$ – число «пи», оно одинаково для любого круга. Точнее его даёт кнопка π на калькуляторе. Если в условии дан **радиус** r – расстояние от центра до края, – то $D = 2r$ и $C = 2\pi r$.

Колесо диаметром 65 мм за оборот проезжает $3,14159 \cdot 65 = 204,2$ мм, то есть примерно 20,4 см – длину короткой школьной линейки. Дальше всё считается умножением и делением:

$$s = n \cdot \pi D, \quad n = \frac{s}{\pi D},$$

где n – число оборотов. За 5 оборотов колесо 65 мм проедет $5 \cdot 20,42 = 102,1$ см. А чтобы проехать 1 м, ему нужно $100 : 20,42 = 4,9$ оборота.

От оборотов в минуту к скорости робота. Как быстро крутится колесо, говорят в оборотах в минуту или в секунду. Колесо делает 2 оборота в секунду – значит, за секунду робот проедет путь двух оборотов:

$$v = (\text{оборотов в секунду}) \cdot \pi D.$$

Пример. Колесо 65 мм крутится со скоростью 120 об/мин. В секунду это $120:60 = 2$ об/с. Скорость робота $2 \cdot 20,42 = 40,8$ см/с.

Если дана скорость вала **мотора**, а между мотором и колесом стоит редуктор, сначала находим обороты колеса: делим обороты мотора на передаточное число (раздел 1.4). Прямо робот едет, только когда оба ведущих колеса крутятся одинаково, – тогда его скорость равна скорости любого из колёс.

Запомни. Один оборот колеса – путь πD . Оборотов нужно: путь, делённый на πD . Скорость робота – обороты в секунду, умноженные на πD ; обороты в минуту сначала делим на 60. В формулу идёт диаметр: дали радиус – удвой его.

Совет. Не заменяй π тройкой «для простоты». Если посчитать обороты с $\pi = 3$, робот на трёх метрах проедет лишние 14 см. Промежуточные числа тоже не спеша округлять: округляют ответ, а не путь за оборот.

2.4. График «путь – время»

НА ПАЛЬЦАХ

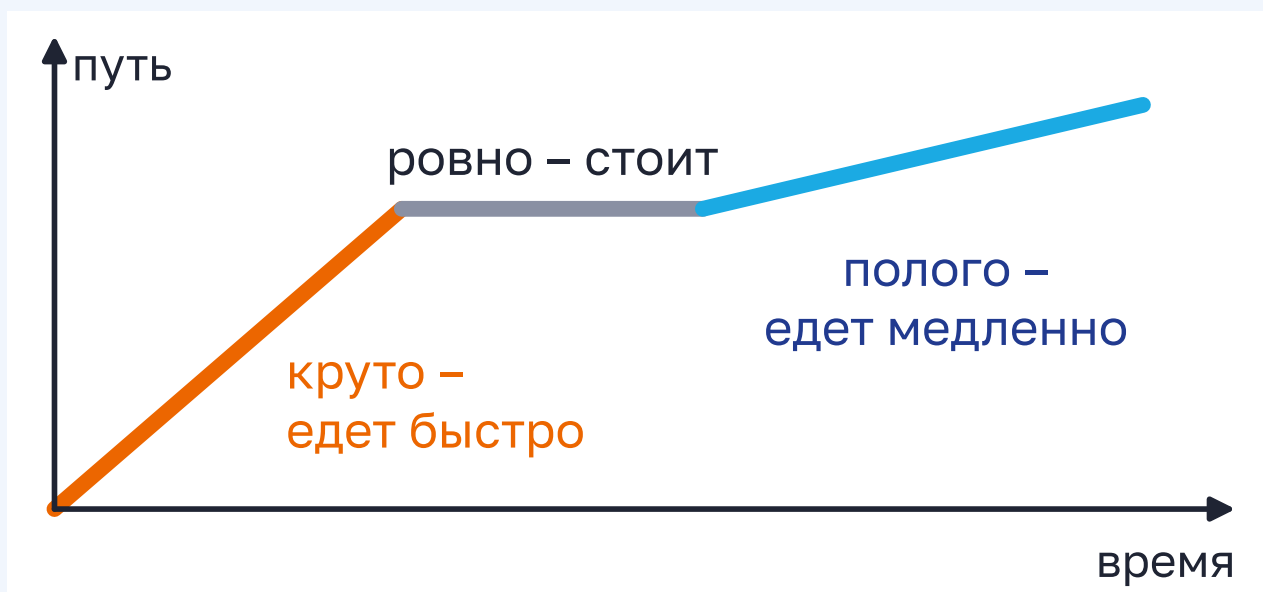


Рисунок 2.5 – График – это дневник поездки

Чем круче поднимается линия, тем быстрее ехал робот. Ровная полочка – робот стоял: время идёт, а путь не растёт.

График – рисунок, на котором видно, как одна величина меняется вместе с другой. На графике «путь – время» по горизонтальной оси откладывают время, по вертикальной – путь, который робот проехал с начала. Каждая точка линии говорит: «к такой-то секунде робот проехал столько-то сантиметров».

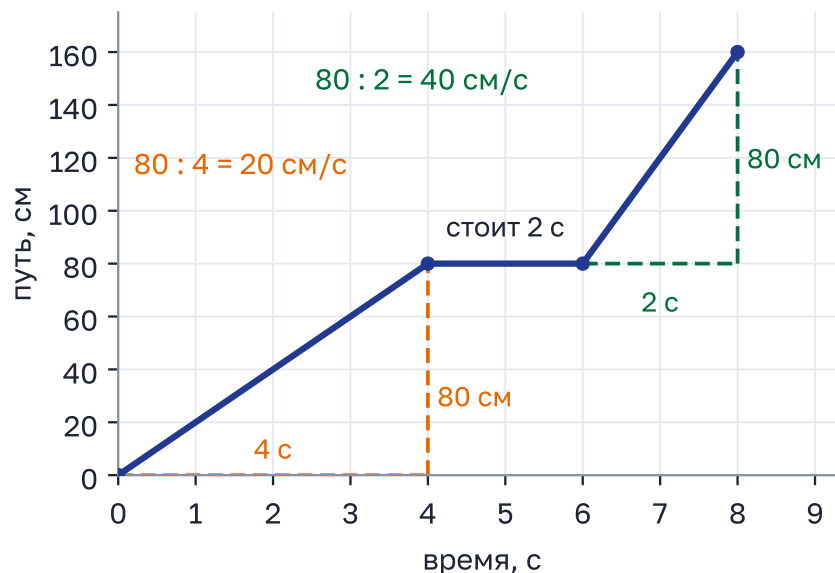


Рисунок 2.6 – Поездка робота: 4 с ехал, 2 с стоял, ещё 2 с ехал. На каждом участке скорость – подъём линии, делённый на время участка

Что можно узнать по такому графику?

- **Путь к моменту времени.** Находишь нужную секунду внизу, поднимаешься до линии и смотришь влево, на шкалу пути. К 2-й секунде робот проехал 40 см, к 7-й – 120 см.
- **Скорость на участке.** Путь, набранный на участке, делим на время этого участка. С 0 до 4 с робот проехал 80 см, его скорость $80 : 4 = 20$ см/с. С 6 до 8 с – тоже 80 см, но всего за 2 с: $80 : 2 = 40$ см/с. Линия здесь вдвое круче.
- **Стоянки.** С 4 до 6 с путь не меняется, 80 см и 80 см, – робот стоял.

Средняя скорость по графику – тот же весь путь, делённый на всё время: $160 : 8 = 20$ см/с.

Путь не может уменьшаться: даже если робот поехал назад, пройденные сантиметры не исчезают. Если на графике линия идёт вниз, значит, по вертикали отложено **расстояние от старта**: робот едет обратно. Поэтому первым делом читай подписи осей.

Запомни. На графике «путь – время» скорость видна по крутизне линии: путь участка дели на его время. Полочка – стоянка. Сначала прочитай подписи осей и узнай, сколько стоит одна клетка.

Совет. Цену клетки находи по двум соседним подписанным линиям. Если подписаны 40 и 80, а между ними ещё одна линия, то клетка по вертикали – 20 см, а вовсе не 40. Ошибка в цене клетки ломает всё решение, хотя каждое действие ты сделаешь верно.

2.5. Разгон, торможение и инерция

НА ПАЛЬЦАХ

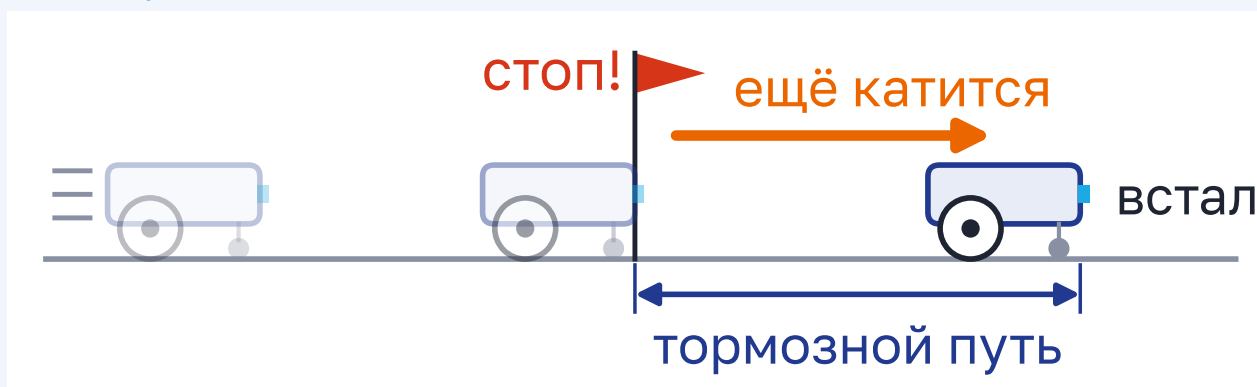


Рисунок 2.7 – Команду «стоп» робот получил у флажка, а встал заметно дальше

Моторы уже не тянут, а робот ещё катится – его несёт инерция. Расстояние от команды до остановки называют **тормозным путём**.

Вспомни школьный автобус. Водитель резко затормозил – и все пассажиры качнулись вперёд. Их никто не толкал: автобус замедлился, а пассажиры по-прежнему двигались с той скоростью, что была. Свойство тел сохранять свою скорость, пока что-нибудь её не изменит, называют **инерцией**. Из-за инерции санки, скатившись с горки, ещё долго едут по ровному, а шайба на льду Обского моря скользит дальше, чем по снегу.

С роботом то же самое. Выключили моторы – он ещё немного проедет, пока трение в колёсах и о пол его не остановит (о трении – в следующем разделе). Чем тяжелее робот и чем быстрее он ехал, тем дольше он тормозит. Разгон тоже не бывает мгновенным: мотору нужно время, чтобы сдвинуть робота с места и раскрутить колёса.

В условии про это пишут прямо. «Разгон и торможение не учитывай» – значит, считаем, что робот сразу едет с полной скоростью и сразу встаёт. Если же дано время торможения или тормозной путь, без них ответ выйдет неверным.

Как посчитать тормозной путь. Часто скорость при торможении падает **равномерно** – за каждую долю секунды на одно и то же число см/с. Такое торможение на графике скорости выглядит как прямая линия вниз.

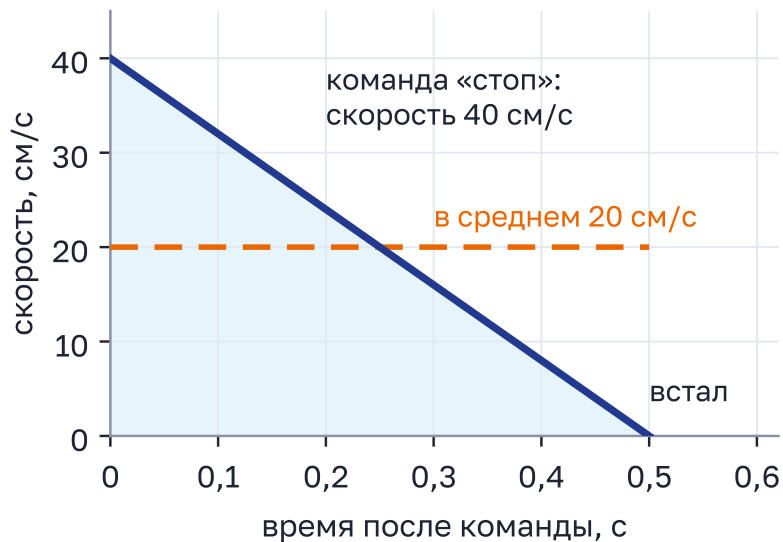


Рисунок 2.8 – Равномерное торможение: скорость падает с 40 см/с до нуля за 0,5 с. Первую половину времени робот едет быстрее 20 см/с, вторую – медленнее

Робот ехал 40 см/с и остановился за 0,5 с. В начале торможения его скорость 40 см/с, в конце – ноль, а в середине, через 0,25 с, – 20 см/с. Насколько в начале торможения робот едет быстрее 20 см/с, настолько же в конце он едет медленнее. Так что в среднем он ехал 20 см/с – вдвое медленнее, чем до команды. Тормозной путь:

$$s_{\text{торм}} = \frac{v}{2} \cdot t_{\text{торм}} = \frac{40}{2} \cdot 0,5 = 20 \cdot 0,5 = 10 \text{ см.}$$

Словами: **при равномерном торможении средняя скорость – половина начальной, а тормозной путь – эта половина, умноженная на время торможения.** С разгоном так же: если робот равномерно разгоняется от нуля до 40 см/с, в среднем он едет 20 см/с. Правило годится только для **равномерного** изменения скорости – в условии так и напишут: «скорость уменьшается равномерно» или «линейно».

Для программы вывод простой: если робот должен встать у линии, команду «стоп» даём раньше – за тормозной путь до неё.

Запомни. После «стопа» робот ещё проезжает тормозной путь, поэтому команду дают раньше на эту длину. Скорость падает (или растёт) равномерно – значит, в среднем робот едет с половиной полной скорости, и путь торможения или разгона равен $\frac{v}{2} \cdot t$.

2.6. Трение: почему колёса буксуют

НА ПАЛЬЦАХ



Рисунок 2.9 – Одно и то же колесо на ковре и на льду

На ковре колесо цепляется за ворс и отталкивает пол назад – робот едет вперёд.
На льду зацепиться не за что: колесо крутится, а робот стоит на месте.

Сначала одно новое слово. **Сила** – это толчок или тяга: то, что может сдвинуть предмет, остановить его или изменить его скорость. Силу измеряют в **ньютонaх** (Н). Чтобы почувствовать 1 Н, положи на ладонь яблоко массой 100 г – примерно с такой силой оно давит на руку.

Сила трения появляется, когда одна поверхность скользит по другой или пытается скользить, и всегда мешает этому скольжению. Попробуй сдвинуть по полу коробку с книгами. Толкаешь слабо – коробка стоит: трение её держит. Толкаешь сильнее – поехала, но трение всё время тормозит, и стоит перестать толкать, как коробка остановится. Сила трения зависит от двух вещей:

- **из чего поверхности:** резиновая шина на ковре держится крепко, пластиковая шина на гладком линолеуме или на льду – слабо;
- **как сильно они прижаты:** полную коробку сдвинуть труднее, чем пустую, а тяжёлый робот прижат к полу сильнее лёгкого.

Трение помогает ехать. Без трения робот никуда бы не уехал. Колесо, вращаясь, упирается в пол и толкает его назад. Трение не даёт колесу проскользнуть, и робот отталкивается от пола вперёд – так же, как ты отталкиваешься ногой при ходьбе. На льду нога проскальзывает, и идти трудно; колесу на скользком полу так же трудно.

Пробуксовка. Если мотор крутит колесо сильнее, чем трение может удержать, колесо проскальзывает по полу – **буксует**. Так бывает на резком старте, на гладком или пыльном полу, у лёгкого робота, у колёс с твёрдыми пластиковыми шинами. Для расчётов это значит: колесо сделало обороты, а робот за каждый оборот проехал меньше πD . Энкодер честно считает обороты, но путь по ним получается больше настоящего. Пример: по оборотам робот должен был проехать 200 см, а по рулетке проехал 190 см. Он недоехал 10 см – это $10:200 \cdot 100 = 5\%$ от расчётного пути.

Что с этим делать? Разгон делают плавным: программа поднимает скорость ступеньками, и колёса не срываются в пробуксовку. А путь по оборотам проверяют заездом с рулеткой на том полу, где робота выступать, – как в задаче 1.6.

Трение и тормозной путь. Робота после выключения моторов останавливает то же трение. На скользком полу его мало, робот тормозит дольше, и тормозной путь длиннее. На ковре тот же робот встанет почти сразу.

Для любопытных: вдвое быстрее – вчетверо дальше. Как тормозной путь зависит от скорости, подробно проходят в физике 9 класса, но понять это можно и сейчас. Возьмём робота на скользком полу: с 20 см/с он тормозит полсекунды и прокатывается 5 см. Пол и робот те же – значит, трение за каждую секунду отнимает у скорости одно и то же число см/с. С 40 см/с робот будет тормозить вдвое дольше, целую секунду, и едет он при этом в среднем вдвое быстрее: 20 см/с, а не 10. Время вдвое больше и скорость вдвое больше – путь вырастает в $2 \cdot 2 = 4$ раза: $20 \cdot 1 = 20$ см. Скорость втрое больше – тормозной путь в $3 \cdot 3 = 9$ раз длиннее. Поэтому в гололёд водители и едут медленно: лишние 10 км/ч удлиняют тормозной путь намного сильнее, чем кажется.

ехал 20 см/с  5 см

ехал 40 см/с  20 см = 4 · 5 см

тот же пол, тот же робот: скорость вдвое больше –
тормозит вдвое дольше и едет в среднем вдвое быстрее

Рисунок 2.10 – Тормозной путь на одном и том же полу: при скорости вдвое больше он вчетверо длиннее

Запомни. Колесо едет, пока трение не даёт ему проскользнуть. Мотор крутит сильнее, чем трение держит, – колесо буксует, и робот проезжает меньше, чем по оборотам. Недобор в процентах считают от пути по оборотам – от того, сколько робот должен был проехать.

Совет. Перед заездом протри шины влажной салфеткой или прокатай их по липкой стороне скотча. Пыль с поля забивается в резину, и колёса начинают проскальзывать там, где вчера держали.

2.7. Разбираем задачи

Задача 2.1 До стенки полигона

разминка · путь, скорость, время, перевод единиц

Робот стоит на старте и едет прямо к стенке полигона со скоростью 25 см/с. До стенки 1,8 м. Через сколько секунд робот коснётся стенки? Разгон не учитывай. Ответ дай в секундах.

Дано: $v = 25$ см/с; $s = 1,8$ м.

Найти: t – ?

Решение

Шаг 1. Единицы. Скорость дана в сантиметрах в секунду, а путь – в метрах. Переводим путь в сантиметры: $1,8 \text{ м} = 1,8 \cdot 100 = 180 \text{ см}$.

Шаг 2. Время. Время – это путь, делённый на скорость: $t = 180 : 25 = 7,2 \text{ с}$.

Прикинем: каждую секунду робот проезжает 25 см, за 7 с – 175 см. До 180 см не хватает совсем чуть-чуть, так что «чуть больше семи секунд» – как раз наш ответ.

 **Типичная ошибка.** Поделить 1,8 на 25 и записать 0,072 с. Метры поделили на сантиметры в секунду – и робот пересекает почти два метра быстрее, чем ты моргнёшь. Бывает и так: $25 \cdot 1,8 = 45$, но скорость, умноженная на путь, – вообще не время.

Ответ: 7,2 с.

Своя задача по программе сферы, модуль 2

Реши сам

2.1. Робот проехал по коридору 2,4 м за 16 с. С какой скоростью он ехал? Ответ дай в сантиметрах в секунду.

2.2. Два робота стоят на концах прямой дорожки длиной 3,5 м и одновременно выезжают навстречу друг другу: первый со скоростью 20 см/с, второй – 30 см/с. На каком расстоянии от места старта первого робота они встретятся? Ответ в сантиметрах.

Задача 2.2 Сколько оборотов сделает колесо

разминка · путь за оборот колеса, длина окружности

Диаметр колеса робота – 64 мм. Робот проехал прямо 2 м, колёса не проскальзывали. Сколько оборотов сделало каждое колесо? Округли до десятых.

Дано: $D = 64$ мм; $s = 2$ м.

Найти: n – ?

Решение

Шаг 1. Что происходит. За каждый оборот колесо прокатывается на длину своей окружности πD . Число оборотов – это весь путь, делённый на путь за один оборот.

Шаг 2. Путь за один оборот. $C = \pi \cdot D = 3,14159 \cdot 64 = 201,06 \text{ мм}$.

Шаг 3. Единицы. Путь за оборот у нас в миллиметрах, значит, и весь путь переводим в миллиметры: $2 \text{ м} = 2 \cdot 1000 = 2000 \text{ мм}$.

Шаг 4. Число оборотов. $n = 2000 : 201,06 = 9,947 \approx 9,9$ оборота.

За оборот колесо проходит около 20 см, а в двух метрах таких кусков как раз около десяти.

❗ **Типичная ошибка.** Принять 64 мм за радиус и посчитать $2\pi \cdot 64 = 402$ мм за оборот – выйдет 5,0 оборота, вдвое меньше верного. Ещё одна – не перевести метры: $2 : 201 = 0,0099$ оборота, колесо даже не шелохнулось. И не округляй 9,947 до 10: про- сили десятиые, значит, 9,9.

Ответ: 9,9 оборота.

Своя задача по программе сферы, модуль 6

Реши сам

2.3. Колесо диаметром 6 см сделало 15 оборотов без проскальзывания. Какой путь проехал робот? Ответ в сантиметрах, округли до целых.

2.4. В описании робота указан радиус колеса – 3,5 см. Сколько оборотов сделает колесо, пока робот проедет 1,1 м? Округли до десятых.

Задача 2.3 Скорость по оборотам мотора

обороты в минуту, редуктор, скорость робота, перевод единиц

Вал мотора робота крутится со скоростью 7200 об/мин. Между мотором и колесом стоит редуктор с передаточным числом 48. Диаметр колеса 65 мм. За сколько секунд робот проедет 3 м? Разгон не учитывай, колёса не проскальзывают. Округли до десятых.

Дано: $n_m = 7200$ об/мин; $i = 48$; $D = 65$ мм; $s = 3$ м.

Найти: t – ?

Решение

Шаг 1. Строим цепочку. Время найдём, если знаем скорость робота. Скорость – по оборотам колеса в секунду и пути за оборот. Обороты колеса – по оборотам мотора и редуктору. Получается так:

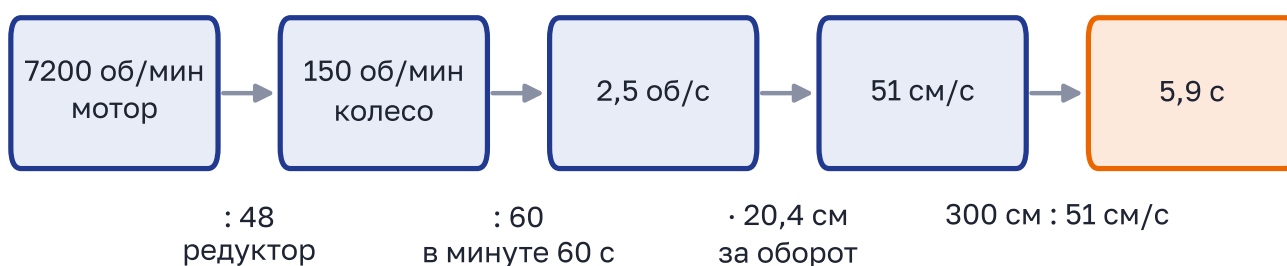


Рисунок 2.11 – Цепочка расчёта: обороты мотора → обороты колеса → обороты в секунду → скорость робота → время

Шаг 2. Обороты колеса. Редуктор замедляет вращение в 48 раз: $7200:48 = 150$ об/мин.

Шаг 3. Обороты в секунду. В минуте 60 секунд: $150:60 = 2,5$ об/с.

Шаг 4. Путь за оборот. Скорость удобно получить сразу в см/с, поэтому диаметр переводим в сантиметры: $65 \text{ мм} = 6,5 \text{ см}$. Путь за оборот $\pi D = 3,14159 \cdot 6,5 = 20,42 \text{ см}$.

Шаг 5. Скорость робота. За секунду колесо делает 2,5 оборота: $v = 2,5 \cdot 20,42 = 51,05 \text{ см/с}$.

Шаг 6. Время. $3 \text{ м} = 300 \text{ см}$. $t = 300:51,05 = 5,88 \approx 5,9 \text{ с}$.

Полметра в секунду – бодрый учебный робот, и три метра для него – дело шести секунд.

 **Типичная ошибка.** Забыть про редуктор. Тогда колесо будто бы делает $7200:60 = 120$ об/с, а робот мчится $120 \cdot 20,42 = 2450 \text{ см/с}$ – это $24,5 \text{ м/с}$, почти 90 км/ч . Так же выдаёт себя пропущенное деление на 60: $150 \cdot 20,42 = 3063 \text{ см/с}$. Если скорость робота вышла больше пары метров в секунду, ищи пропущенный шаг.

Ответ: 5,9 с.

Своя задача по программе сферы, модули 2 и 6

Реши сам

2.5. Колесо диаметром 6 см делает 90 об/мин. С какой скоростью едет робот? Ответ в сантиметрах в секунду, округли до десятых.

2.6. Робот должен ехать со скоростью 30 см/с. Диаметр колеса 5 см. Сколько оборотов в минуту должно делать колесо? Округли до целых.

Задача 2.4 Читаем график поездки

график путь–время, скорость на участке, средняя скорость

На графике показано, как робот проехал по полигону. а) Сколько секунд робот стоял? б) С какой самой большой скоростью он ехал? в) Какова средняя скорость робота за всю поездку? В ответ введи среднюю скорость в сантиметрах в секунду.

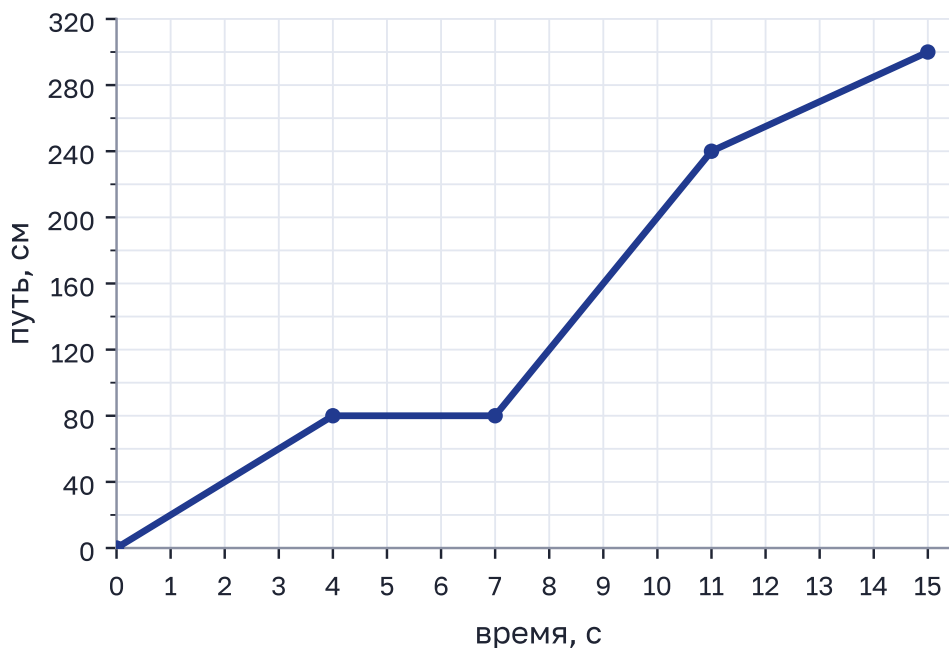


Рисунок 2.12 – График поездки робота

Дано: график «путь – время».

Найти: время стоянки; самую большую скорость; $v_{\text{ср}}$ – ?

Решение

Шаг 1. Цена клетки. По горизонтали соседние линии – через 1 с. По вертикали подписаны 0, 40, 80..., а между подписанными линиями есть ещё одна. Значит, клетка по вертикали – 20 см.

Шаг 2. Разбиваем на участки. Линия ломается в точках: (0 с; 0 см), (4 с; 80 см), (7 с; 80 см), (11 с; 240 см), (15 с; 300 см). Получилось четыре участка.

Шаг 3. Стоянка. С 4-й по 7-ю секунду путь не меняется – 80 см. Робот стоял $7 - 4 = 3$ с.

Шаг 4. Скорости на участках. На каждом участке путь участка делим на его время:

- с 0 до 4 с: $80 : 4 = 20$ см/с;
- с 7 до 11 с: $(240 - 80) : (11 - 7) = 160 : 4 = 40$ см/с;
- с 11 до 15 с: $(300 - 240) : (15 - 11) = 60 : 4 = 15$ см/с.

Самая большая – 40 см/с, на самом крутом участке.

Шаг 5. Средняя скорость. Весь путь 300 см, всё время – 15 с, вместе со стоянкой:
 $300 : 15 = 20$ см/с.

Средняя вышла вдвое меньше самой большой скорости: робот три секунды стоял, а в конце еле полз.

⚠ Типичная ошибка. Сложить скорости участков и поделить на их число:
 $(20 + 40 + 15) : 3 = 25$ см/с или, со стоянкой, $(20 + 0 + 40 + 15) : 4 = 18,75$ см/с.
 Так нельзя: участки длились разное время, а среднее «по скоростям» считается, только когда время на каждой скорости одинаковое. Вторая ловушка – принять клетку по вертикали за 40 см и получить все пути вдвое больше.

Ответ: а) 3 с; б) 40 см/с; в) 20 см/с.

Своя задача по программе сферы, модуль 6

Реши сам

2.7. Робот вёз груз, а его программа каждые 2 с записывала пройденный путь в таблицу. В какой промежуток времени робот стоял? Какова его средняя скорость за 10 с? Ответ в сантиметрах в секунду.

Таблица 2.1

Время, с	Путь, см
0	0
2	30
4	60
6	60
8	100
10	140

2.8. Робот доехал до стенки, стоящей в 1,2 м от старта, со скоростью 30 см/с, развернулся и вернулся на старт со скоростью 20 см/с. Время разворота не учитывай. Найди среднюю скорость за всю поездку туда и обратно в сантиметрах в секунду.

Задача 2.5 Где остановится робот

инерция, равномерное торможение, тормозной путь, миллисекунды

Робот стартует и сразу едет со скоростью 40 см/с (разгон не учитываем). Его программа: «включить моторы вперёд, ждать t миллисекунд, выключить моторы». После выключения моторов робот ещё 0,6 с катится, и его скорость равномерно падает до нуля. Робот должен остановиться на стоп-линии в 2 м от старта. Какое t поставить в программе? Ответ в миллисекундах.

Дано: $v = 40$ см/с; $t_{\text{торм}} = 0,6$ с; до стоп-линии 2 м.

Найти: t в мс – ?

Решение

Шаг 1. Что происходит. Путь до линии складывается из двух кусков: пока моторы работают, робот едет 40 см/с; после выключения он проезжает ещё тормозной путь. Вместе – 2 м.

Шаг 2. Тормозной путь. Скорость падает равномерно, значит, в среднем робот тормозит со скоростью $40 : 2 = 20$ см/с. За 0,6 с он прокатится $20 \cdot 0,6 = 12$ см.

Шаг 3. Путь с включёнными моторами. $2 \text{ м} = 200 \text{ см}$. На моторах робот должен проехать $200 - 12 = 188 \text{ см}$.

Шаг 4. Время работы моторов. $188 : 40 = 4,7 \text{ с}$.

Шаг 5. В миллисекунды. В секунде 1000 мс: $4,7 \cdot 1000 = 4700 \text{ мс}$.

Без торможения вышло бы 5 с. Наш ответ немного меньше: последние 12 см робот проезжает уже без моторов, по инерции.

⚠ Типичная ошибка. Забыть о торможении и поставить $200 : 40 = 5 \text{ с}$, то есть 5000 мс. Тогда моторы выключатся на отметке 200 см, робот прокатится ещё 12 см и остановится в 212 см от старта – за линией. Другая ошибка – посчитать тормозной путь с полной скоростью: $40 \cdot 0,6 = 24 \text{ см}$. Тогда выйдет 4400 мс, и робот не доедет до линии 12 см. И не забудь перевести секунды: в программу для Arduino идёт 4700, а не 4,7.

Ответ: 4700 мс.

Своя задача по программе сферы, модуль 6

Реши сам

2.9. Робот едет со скоростью 50 см/с, его тормозной путь на этом полу – 8 см. Коробка стоит в 1,5 м от старта, и робот должен остановиться за 2 см до неё. Через сколько миллисекунд после старта нужно выключить моторы? Разгон не учитывай.

2.10.* На одном и том же полу робот, ехавший со скоростью 20 см/с, после выключения моторов прокатился до остановки 3 см. Каким будет тормозной путь, если робот поедет со скоростью 60 см/с? Торможение в обоих случаях равномерное, и за каждую секунду скорость падает на одну и ту же величину. Ответ в сантиметрах.

Задача 2.6* Колёса буксуют

трение, проскальзывание колёс, проценты, путь за оборот

Робота с колёсами диаметром 60 мм проверили на двух полах. Счётчик оборотов стоит на колесе. На ковре за 20 оборотов колеса робот проехал по рулетке 3,77 м, а на гладком линолеуме за те же 20 оборотов – только 3,47 м.

- а) На сколько процентов меньше расчётного пути робот проезжает на линолеуме? Округли до целых.
- б) Программу написали по диаметру колеса, чтобы робот проехал 2,5 м. Где он остановится на линолеуме? Ответ в метрах от старта, округли до сотых.
- в) Сколько оборотов колеса нужно задать, чтобы на линолеуме робот проехал 2,5 м? Округли до десятых. В ответ введи число из пункта в.

Дано: $D = 60 \text{ мм}$; 20 оборотов: на ковре 3,77 м, на линолеуме 3,47 м; нужный путь 2,5 м.

Найти: а) недобор в %; б) где остановится; в) число оборотов для линолеума.

Решение

Шаг 1. Кто виноват – колесо или пол? Посчитаем путь по диаметру. За оборот $\pi D = 3,14159 \cdot 60 = 188,5$ мм, за 20 оборотов $20 \cdot 188,5 = 3770$ мм = 3,77 м. Это совпало с заездом по ковру до сантиметра. Значит, диаметр указан верно, и на ковре колёса не проскальзывают. На линолеуме колесо то же, а путь меньше – колёса буксуют: трения не хватает.

Шаг 2. а) Процент недобора. На линолеуме робот недоехал $3,77 - 3,47 = 0,30$ м. Процент считаем от расчётного пути 3,77 м: $0,30 : 3,77 \cdot 100 = 7,96 \approx 8 \%$.

Шаг 3. Сколько оборотов в программе. По диаметру на 2,5 м нужно $2500 : 188,5 = 13,26$ оборота.

Шаг 4. Настоящий путь за оборот на линолеуме. $3470 : 20 = 173,5$ мм.

Шаг 5. б) Где остановится. $13,26 \cdot 173,5 = 2301$ мм $\approx 2,30$ м. Робот не доедет до цели 20 см.

Шаг 6. в) Обороты для линолеума. $2500 : 173,5 = 14,41 \approx 14,4$ оборота.

Для скользкого пола оборотов понадобилось больше, чем по расчёту (14,4 против 13,3): каждый оборот там даёт меньше пути. Вышло бы меньше 13,3 – ищи, где перевернул деление.

 **Типичная ошибка.** Считать процент от пути по линолеуму: $0,30 : 3,47 \cdot 100 = 8,6 \approx 9 \%$. Недобор считают от того, сколько робот **должен был** проехать, – от 3,77 м. Ещё одна ловушка – решить, что колесо «стало меньше», и заказать новые колёса. Колёса тут ни при чём: на ковре они дают путь точно по диаметру. На линолеуме не хватает трения, и помогут плавный разгон или шины, которые лучше держат пол.

Ответ: а) 8 %; б) 2,30 м; в) 14,4 оборота.

Своя задача по программе сферы, модуль 6

Реши сам

2.11. Колесо диаметром 5 см сделало 40 оборотов, а робот по рулетке проехал 6,03 м. На сколько процентов меньше расчётного пути он проехал? Округли до целых.

2.12.* На гладком полу колёса робота проскальзывают, и он проезжает на 10 % меньше, чем выходит по оборотам. Диаметр колеса 6 см. Сколько оборотов должно сделать колесо, чтобы робот проехал 1,8 м? Округли до десятых.

Задача 2.7* Разгон по секундам

плавный разгон, пошаговая модель движения, таблица, цикл

Чтобы колёса не буксовали на старте, робот разгоняется плавно. Программа устроена так: в начале каждой секунды она прибавляет к скорости 10 см/с, пока скорость не станет 40 см/с, а дальше робот едет с 40 см/с. Считаем, что внутри секунды скорость не меняется: какой её

поставили в начале секунды, с такой робот и едет до конца этой секунды. Робот стартует с места. За сколько секунд он проедет 3 м? Ответ – целое число секунд.

Дано: прибавка скорости 10 см/с в начале каждой секунды; наибольшая скорость 40 см/с; путь 3 м.

Найти: t – ?

Решение

Шаг 1. Модель. Движение считаем по шагам, секунда за секундой. На каждом шаге сначала меняется скорость, потом к пути прибавляется то, что робот проехал за эту секунду, – столько сантиметров, сколько сейчас его скорость в см/с.

Шаг 2. Разгон по секундам. Выпишем первые секунды в таблицу:

Таблица 2.2

Секунда	Скорость, см/с	Проехал за секунду, см	Путь с начала, см
1-я	10	10	10
2-я	20	20	30
3-я	30	30	60
4-я	40	40	100

С 4-й секунды скорость больше не растёт. За 4 с робот проехал 100 см.

Шаг 3. Остаток пути. 3 м = 300 см. Осталось $300 - 100 = 200$ см со скоростью 40 см/с: $200 : 40 = 5$ с.

Шаг 4. Всё время. $4 + 5 = 9$ с.

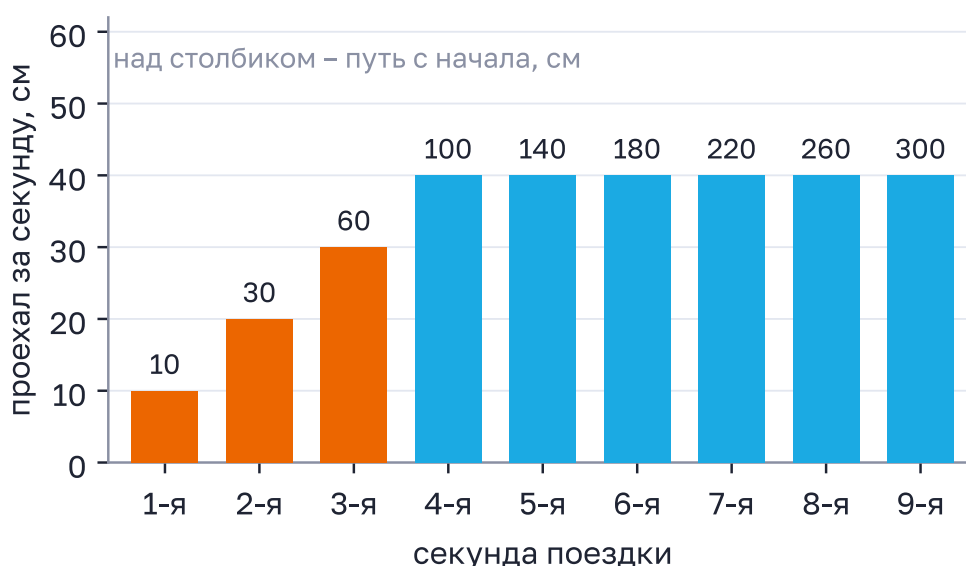


Рисунок 2.13 – Сколько робот проезжает за каждую секунду: первые три секунды – разгон. Над столбиками – путь с начала поездки

Проверка кодом. Одинаковые шаги – работа для цикла. Вот та же модель на Python:

```

v = 0           # скорость, см/с: робот стоит
s = 0           # путь с начала, см
t = 0           # сколько секунд прошло
while s < 300:  # повторяем, пока не проехали 3 м
    if v < 40:  # ещё не разогнались?
        v = v + 10 # в начале секунды прибавляем скорость
        s = s + v   # за эту секунду проехали v сантиметров
        t = t + 1   # прошла ещё одна секунда
print(t, s)       # 9 300

```

Первые три строки заводят **переменные** – ячейки памяти с именами `v`, `s` и `t`, в которых лежат числа. Всё, что после знака `#`, – комментарий для человека, компьютер его пропускает. Слово `while` значит «пока»: строки с отступом под ним повторяются, пока верно условие `s < 300`. `if` значит «если»: скорость растёт, только пока она меньше 40. `print` выводит на экран число секунд и путь – 9 и 300.

Без разгона робот проехал бы 3 м за $300:40 = 7,5$ с. Плавный старт обошёлся в полторы секунды – зато без пробуксовки.

⚠ Типичная ошибка. Не заметить, **когда** прибавляется скорость. Если бы прибавка шла в конце секунды, в 1-ю секунду робот стоял бы на месте, и ответ вышел бы 10 с. Слова «в начале» и «в конце» в таких условиях перечитай дважды. Другая ошибка – забыть про разгон и ответить 7,5 или 8 с.

Ответ: 9 с.

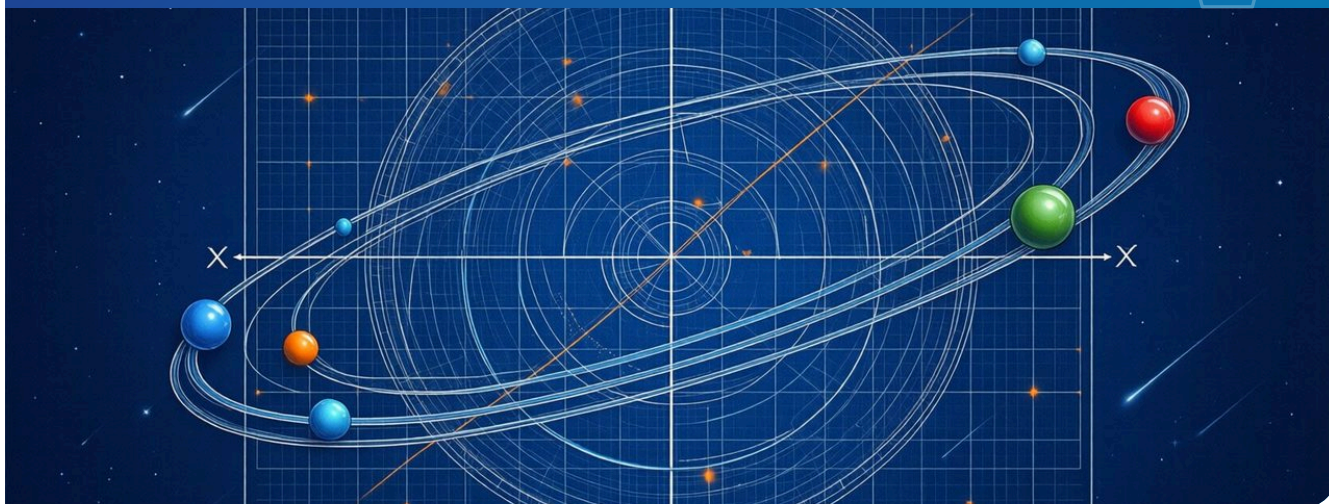
По мотивам задачи первого отборочного этапа НТО 2024/25, профиль «Интеллектуальные робототехнические системы», предметный тур по информатике, «Ускорение ускорения»

Реши сам

2.13. Модель разгона та же, что в задаче 2.7, но программа прибавляет 5 см/с в начале каждой секунды, пока скорость не станет 25 см/с. Какой путь робот проедет за первые 8 секунд? Ответ в сантиметрах.

2.14.* Робот разгоняется по модели задачи 2.7: в начале каждой секунды скорость растёт на 10 см/с, пока не станет 30 см/с. Потом робот едет с 30 см/с, а перед финишем тормозит тоже по секундам: в начале каждой секунды скорость падает на 10 см/с (30, 20, 10, 0). Когда скорость стала 0, робот стоит. От старта до остановки он проехал 2,4 м. Сколько секунд робот был в движении?

Координаты и углы: навигация робота



Робот на полигоне не понимает слов «у окна» или «возле двери». Ему нужны числа: где он стоит, куда смотрит, сколько клеток ехать и на сколько градусов повернуть. Этими числами мы и займёмся. На отборе задача может дать программу из команд «вперёд» и «поворот» и спрашивает, где робот окажется. Или показывает, куда он смотрит, и просит угол поворота. Или рисует маршрут на плане и хочет знать его длину на настоящем поле.

3.1. Координаты на поле

НА ПАЛЬЦАХ

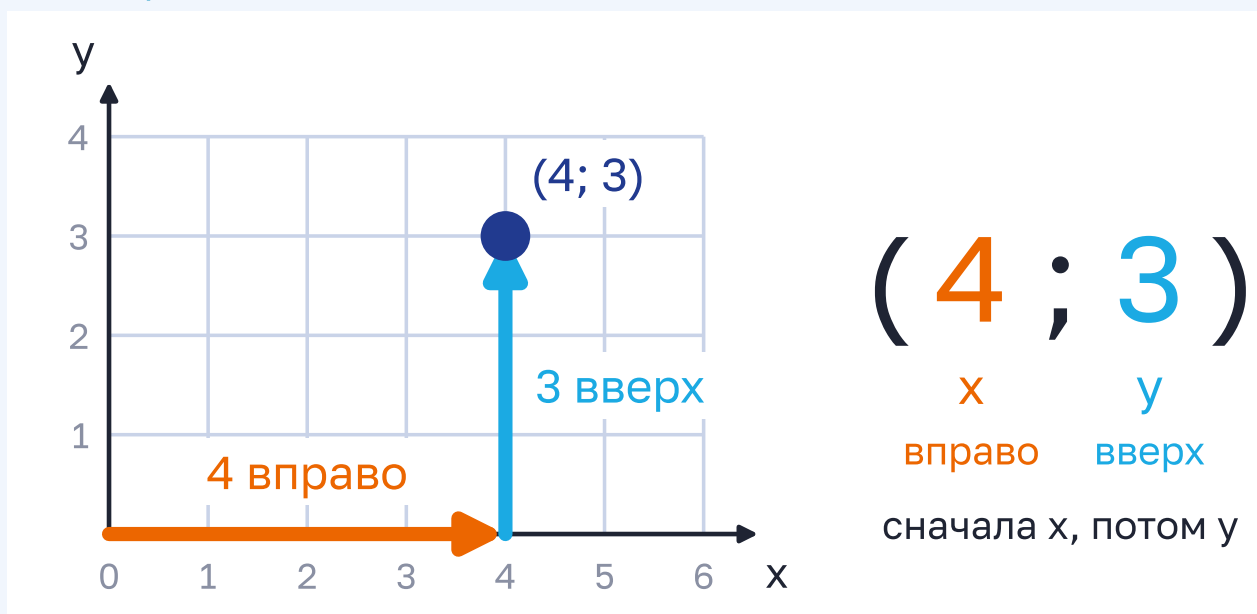


Рисунок 3.1 – Адрес точки – два числа

От угла поля сначала отсчитываем шаги вправо, потом вверх. Точка (4; 3) – четыре шага вправо и три вверх.

Полигон для роботов в сфере «Роботех» – это белое поле, по которому нарисованы чёрные линии. Линии пересекаются и образуют сетку, как в тетради в клетку. Места, где линии пересекаются, называют **перекрёстками**. Робот едет по линии, считает перекрёстки и на нужном останавливается или поворачивает.

Чтобы сказать роботу, куда ехать, перекрёстки нумеруют. В кинотеатре место задают двумя числами: ряд и кресло. В «Морском бое» – буквой и цифрой: «Б-5, ранил». На поле – двумя числами: сколько шагов от угла **вправо** и сколько **вверх**. Эти два числа называют **координатами** точки. Угол, от которого считаем, – **начало отсчёта**, у него координаты (0; 0).

- Первое число – **x** («икс»): сколько шагов вправо. Вдоль него идёт горизонтальная **ось x**.
- Второе число – **y** («игрек»): сколько шагов вверх. Вдоль него идёт вертикальная **ось y**.

Записывают так: (4; 3). Между числами ставим точку с запятой, потому что запятая у нас нужна для дробей: точка (2,5; 1) – это два с половиной шага вправо и один вверх. Порядок менять нельзя. Точки (4; 3) и (3; 4) – разные перекрёстки, и робот, перепутавший их, приедет не туда.

Шаг здесь – расстояние между соседними линиями, сторона клетки. На полигоне она может быть 20 см, 30 см, 50 см – в условии задачи это всегда сказано. Координаты при этом считают в клетках, а в сантиметры переводят в самом конце.

Осторожно с подсчётом. От линии 0 до линии 4 – четыре шага, хотя линий по дороге пять (0, 1, 2, 3, 4). С забором то же самое: у пяти столбиков четыре пролёта. Считай промежутки, а не линии.

Вернись к рисунку в начале раздела. Оранжевая стрелка кончается в точке (4; 0): четыре шага вправо и ноль вверх, точка лежит прямо на оси x . Самый верхний перекрёсток левого края – (0; 4): вправо идти не нужно, только вверх. А правый верхний угол нарисованного поля – (6; 4).

Иногда начало отсчёта ставят в середину поля. Тогда у точек левее середины x получается с минусом, у точек ниже – y с минусом: (–2; 3) – два шага влево и три вверх. А в программах, которые работают с картинками с камеры, ось y часто смотрит **вниз**, и отсчёт идёт от левого верхнего угла. Поэтому, прежде чем считать, посмотри на рисунок: где ноль и куда смотрят стрелки осей.

Запомни. Точка (x ; y): x – шаги вправо от начала отсчёта, y – шаги вверх. Сначала всегда x . Считай промежутки между линиями, а не сами линии.

3.2. Направление и углы поворота

НА ПАЛЬЦАХ



Рисунок 3.2 – «Направо» у робота – своё

Робот смотрит вниз – и его «направо» на карте показывает влево. Повороты всегда считаем от носа робота.

Угол поворота меряют в градусах. Полный оборот вокруг себя – 360°. Остальное получается из него:

- **180°** – половина оборота, разворот назад;
- **90°** – четверть оборота, **прямой угол**: так робот сворачивает на перекрёстке;
- **45°** – половина прямого угла: так смотрит диагональ клетки, из одного её угла в противоположный.

Для тренировки глаза посмотри на часы. Минутная стрелка за час делает полный оборот, 360°. За 15 минут – четверть, 90°. За одну минуту – 6°.

Слов для направлений у нас два набора. Перепутаешь их – робот уедет в другую сторону. «Вверх», «вниз», «вправо», «влево» – это направления **на карте**, как север, юг, восток и за-

пад. Они от робота не зависят. А «направо», «налево», «вперёд», «назад» – это направления **от робота**. Смотрит робот вверх – его «направо» совпадает с «вправо» на карте. Развернулся и смотрит вниз – его «направо» на карте уже слева (рисунок выше).

Куда смотрит робот, удобно записывать одним числом – **курсом**. Курс – угол от направления «вверх» по часовой стрелке, как у компаса: 0° – вверх, 90° – вправо, 180° – вниз, 270° – влево. Посередине – диагонали: 45°, 135°, 225°, 315°.

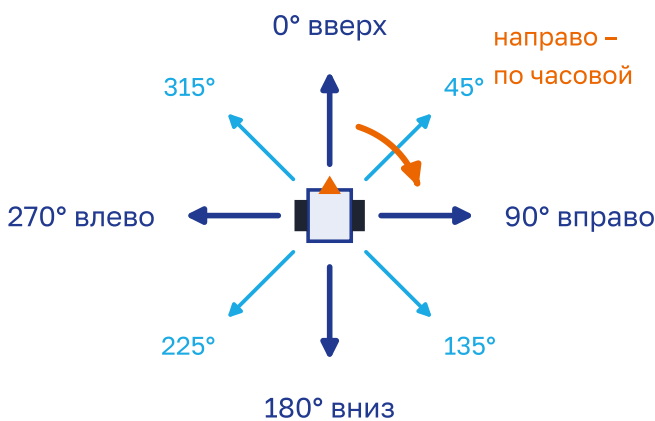


Рисунок 3.3 – Курс робота. Поворот направо – по часовой стрелке, курс растёт

Когда робот едет вперёд вдоль линии, у него меняется только одна координата:

Таблица 3.1

Курс	Куда смотрит	Шаг вперёд
0°	вверх	$y + 1$
90°	вправо	$x + 1$
180°	вниз	$y - 1$
270°	влево	$x - 1$

Как считать курс после поворотов. Поворот направо – прибавляем угол, налево – вычитаем:

- направо на α : новый курс = курс + α ;
- налево на α : новый курс = курс – α .

Буквой α («альфа») обозначаем угол поворота. Если получилось 360° или больше, вычитаем 360°: целый оборот ничего не меняет. Если получилось меньше нуля, прибавляем 360°. Пример: робот смотрит влево (270°) и поворачивает направо на 135°: $270 + 135 = 405$, $405 - 360 = 45^\circ$ – теперь он смотрит по диагонали вверх-вправо. Другой пример: курс 0°, налево на 90°: $0 - 90 = -90$, $-90 + 360 = 270^\circ$, робот смотрит влево.

На какой угол повернуть. Робот смотрит с курсом A , а ехать нужно с курсом B . Поворот направо – на $B - A$ (если вышло меньше нуля, прибавь 360°). Поворот налево – на то, что осталось до полного оборота: 360° минус угол поворота направо. Из двух выбирают меньший, он всегда не больше 180°. Пример: робот смотрит вправо (90°), а ехать надо вниз-влево (225°). Направо – на $225 - 90 = 135^\circ$, налево – на $360 - 135 = 225^\circ$. Короче направо, на 135°.

Если цель лежит на диагонали от робота – сдвиги по x и по y одинаковые, – курс на неё 45° , 135° , 225° или 315° . Например, из (1; 1) в (4; 4) сдвиг 3 вправо и 3 вверх – курс 45° . Из (5; 5) в (2; 2) – 3 влево и 3 вниз, курс 225° .

Совет. Запутался, где у робота «направо»? Поверни тетрадь так, чтобы нос робота на рисунке смотрел от тебя. Теперь его «направо» – твоё «направо». Сделай поворот карандашом и верни тетрадь обратно.

Робот рисует фигуру. Допустим, робот обходит квадрат: едет, поворачивает направо на 90° , едет, снова поворачивает... После четырёх поворотов он вернулся в начало и смотрит туда же, куда смотрел на старте. Значит, все повороты вместе дали полный оборот: $4 \cdot 90^\circ = 360^\circ$. Так с любой **правильной** фигурой – у которой все стороны равны и все углы равны. Если у неё n сторон, то в каждой вершине робот поворачивает на

$$\text{угол поворота} = \frac{360^\circ}{n}.$$

Для квадрата $360:4 = 90^\circ$, для правильного шестиугольника $360:6 = 60^\circ$, для восьмиугольника – 45° .

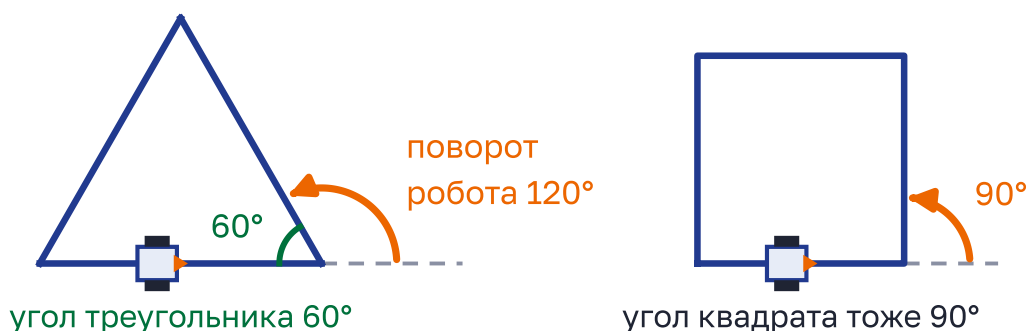


Рисунок 3.4 – У треугольника угол 60° , а робот в вершине поворачивает на 120°

С треугольником ошибаются больше всего. Угол правильного треугольника – 60° , и рука сама пишет «поворот на 60° ». Но робот поворачивает не на угол фигуры, а на угол между тем, куда он ехал, и тем, куда поедет дальше (оранжевая дуга на рисунке). Это $360:3 = 120^\circ$. Проверка: угол фигуры и угол поворота вместе дают 180° , ведь робот, не повернув, поехал бы прямо дальше по пунктиру: $60 + 120 = 180$. У квадрата оба угла по 90° , поэтому там их и не путают.

Запомни. «Вверх–вниз–вправо–влево» – по карте, «направо–налево» – от носа робота. Курс: 0° вверх, 90° вправо, 180° вниз, 270° влево. Направо – прибавь угол, налево – вычти, вышло за 360 или ниже нуля – поправь на 360. В вершине правильной фигуры робот поворачивает на $360^\circ:n$.

3.3. Программа движения: где окажется робот

НА ПАЛЬЦАХ

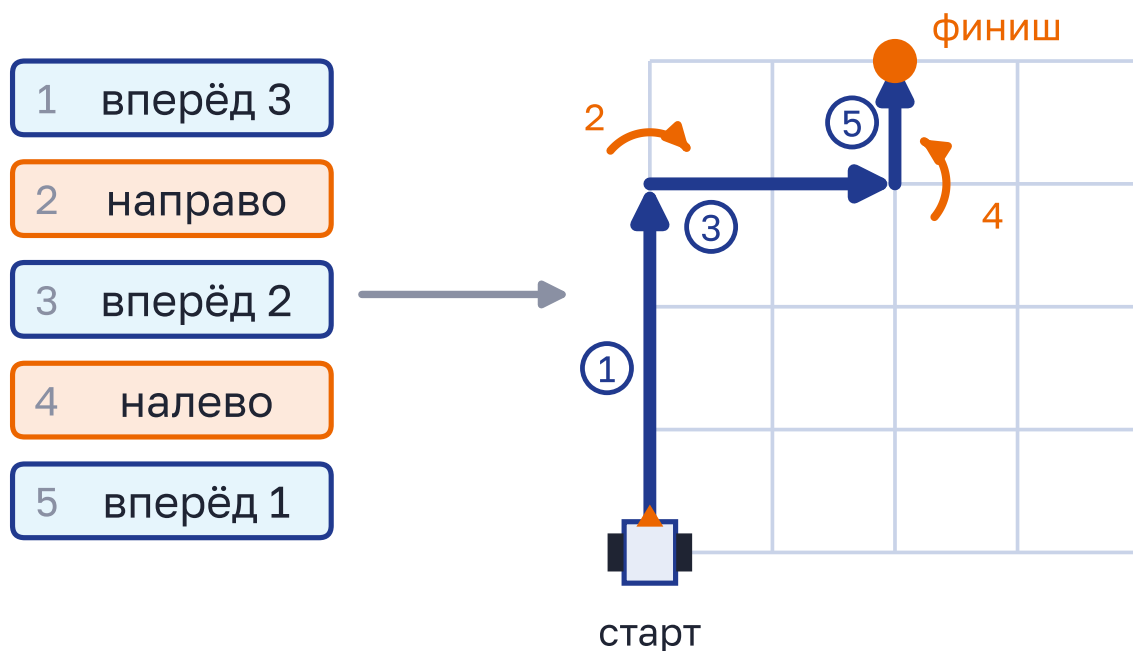


Рисунок 3.5 – Программа – список шагов и поворотов

Робот выполняет команды по одной, сверху вниз. На поворотах он крутится на месте, как фигурист, – колёсами никуда не уезжает.

Робот – **исполнитель**: он умеет выполнять несколько простых команд, и из них собирают программу. Для езды по полю хватает таких:

- **вперёд N** – проехать вперёд N клеток (или N сантиметров – скажут в условии);
- **направо** и **налево** – повернуть на месте на 90°, если не сказано другое;
- **развернуться** – повернуть на месте на 180°;
- иногда ещё **назад N** – откатиться на N клеток, не поворачиваясь.

Чтобы узнать, где робот закончит программу, заведи таблицу, как бортовой журнал. В ней четыре столбца: команда, курс, x, y. Первая строка – старт. Дальше на каждую команду – новая строка: поворот меняет только курс, «вперёд» – только одну из координат, а какую – смотри по таблице из раздела 3.2.

Так выглядит журнал для программы с рисунка. Робот стартует в точке (0; 0) и смотрит вверх.

Таблица 3.2

Команда	Курс	x	y
старт	0° (вверх)	0	0
вперёд 3	0°	0	3
направо	90° (вправо)	0	3

Команда	Курс	x	y
вперёд 2	90°	2	3
налево	0° (вверх)	2	3
вперёд 1	0°	2	4

Робот закончил в точке (2; 4) и снова смотрит вверх. Можно проверить на глаз: вверх он проехал $3 + 1 = 4$ клетки, вправо – 2 клетки. Раз начал он из (0; 0), эти числа и дают его координаты.

Журнал удобен и тем, что его легко превратить в программу для компьютера. Код ниже на языке Python делает то же, что мы сделали в таблице. Его можно пропустить – о программах подробно рассказывают главы 4 и 6. Но если хочешь проверять себя на компьютере, вот как он устроен.

```
# на сколько меняются x и y за одну клетку вперёд при каждом курсе
shag = {0: (0, 1), 90: (1, 0), 180: (0, -1), 270: (-1, 0)}

x, y, kurs = 0, 0, 0 # старт: точка (0; 0), робот смотрит вверх
programma = [("вперёд", 3), ("направо", 90), ("вперёд", 2),
             ("налево", 90), ("вперёд", 1)]

for komanda, chislo in programma:
    if komanda == "вперёд":
        dx, dy = shag[kurs]
        x = x + dx * chislo
        y = y + dy * chislo
    elif komanda == "направо":
        kurs = (kurs + chislo) % 360
    else:
        kurs = (kurs - chislo) % 360 # налево

print(x, y, kurs) # 2 4 0
```

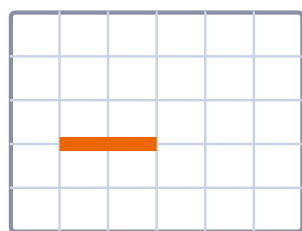
- Всё, что после знака `#`, – пояснения для человека, компьютер их пропускает.
- `shag` – это таблица из раздела 3.2: для каждого курса записано, на сколько меняются `x` и `y` за одну клетку. Курс 90° (вправо) даёт (1, 0): `x` растёт на 1, `y` не меняется.
- `x, y, kurs = 0, 0, 0` – заводим три **переменные** (ячейки памяти с именами) и кладём в каждую ноль.
- `programma` – список команд, по порядку. Такой список в программировании называют **массивом**. Кстати, в курсе сферы робот так и «запоминает путь»: складывает свои команды в массив.
- `for ... in programma:` – «для каждой команды из списка сделай то, что ниже с отступом».
- `if / elif / else` – «если / иначе если / иначе»: смотрим, какая пришла команда.
- `%` – остаток от деления. `(kurs + chislo) % 360` убирает лишние полные обороты: $405\%360 = 45$. А $-90\%360$ Python считает как 270 – это то же правило «прибавь 360» из раздела 3.2.
- `print` выводит ответ: 2 4 0, то есть точка (2; 4), курс 0°.

Запомни. В журнале на каждую команду – своя строка. У строки с поворотом х и у переписываются без изменений, у строки «вперёд» меняется одно число из двух, а курс остаётся прежним.

3.4. Масштаб: план и настоящий полигон

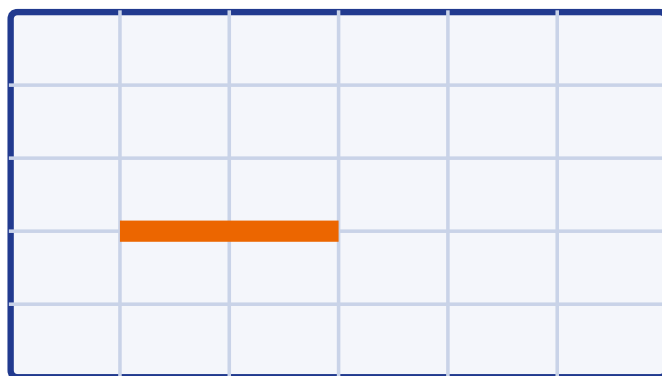
НА ПАЛЬЦАХ

масштаб 1 : 50



план: 1 см

× 50



поле: 50 см

Рисунок 3.6 – Масштаб 1 : 50: сантиметр на плане – это 50 см на поле

План – уменьшенная копия поля. Масштаб говорит, во сколько раз уменьшили.

Полигон размером 3 м × 2 м на тетрадный лист не поместится, поэтому рисуют **план** – уменьшенную копию, у которой все размеры уменьшены в одно и то же число раз. Это число записывают в **масштаб**. Масштаб 1 : 50 («один к пятидесяти») значит: 1 см на плане – это 50 см на настоящем поле.

- С плана на поле – **умножаем** на 50: отрезок 7 см на плане – это $7 \cdot 50 = 350$ см = 3,5 м на поле.
- С поля на план – **делим** на 50: край поля 2,4 м = 240 см, на плане это $240 : 50 = 4,8$ см.

Умножив сантиметры плана на 50, получаешь тоже сантиметры – только уже настоящие. Нужен ответ в метрах – переведи отдельным шагом: 350 см = $350 : 100 = 3,5$ м.

Масштаб бывает записан словами: «в 1 см – 30 см». Это то же самое, что 1 : 30. А иногда его надо найти самому. Скажем, на плане одна клетка тетради (5 мм) – одна клетка поля (30 см). Переводим в одинаковые единицы: 30 см = 300 мм. Во сколько раз клетка поля больше: $300 : 5 = 60$. Масштаб 1 : 60.

Чем меньше второе число, тем **крупнее** рисунок. На плане 1 : 20 поле выйдет крупнее, чем на плане 1 : 50: каждый метр поля займёт 5 см, а не 2 см.

Совет. Проверь себя здравым смыслом. Настоящее поле всегда больше плана. Если с плана на поле у тебя получилось меньше, чем было, – ты поделил вместо того, чтобы умножить.

Запомни. Масштаб $1 : k$: с плана на поле – умножь на k , с поля на план – раздели на k . Если масштаб не дан, а даны размеры одного и того же отрезка на плане и на поле, переведи их в одни единицы и раздели большее на меньшее.

3.5. Расстояние и маршрут: по клеткам и по прямой

НА ПАЛЬЦАХ

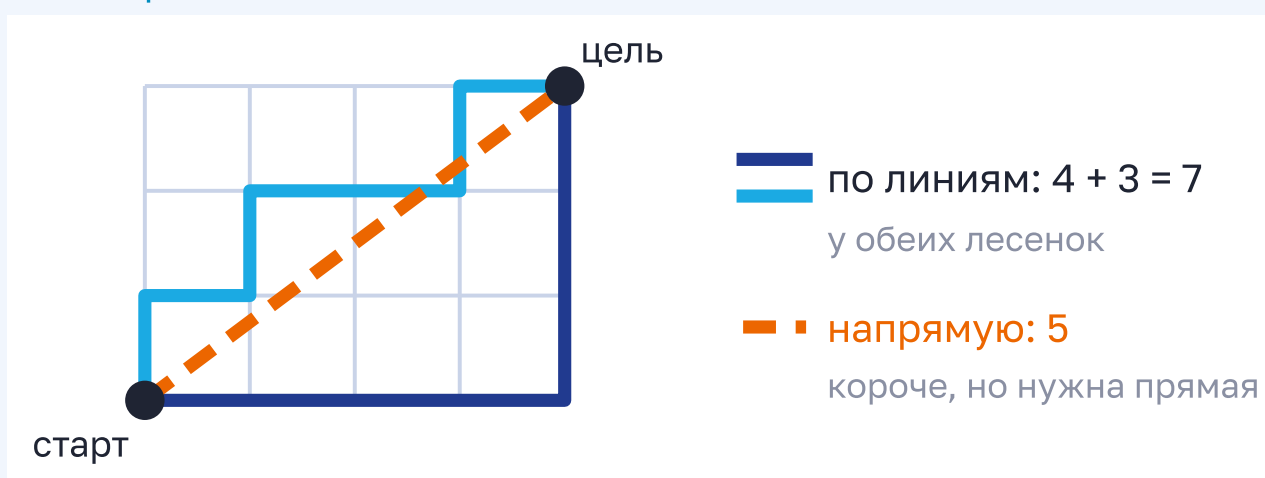


Рисунок 3.7 – Путь по линиям поля и путь напрямую

Как ни петляй лесенкой вправо и вверх, по линиям выйдет те же $4 + 3 = 7$ клеток. Напрямую короче, но так можно ехать только там, где нет стен и не нужны линии.

Робот, который ездит по линиям, не может срезать угол. Как пешеход в городе: между кварталами дома, и по диагонали через двор не пройдёшь. Сколько он проедет от точки $(x_1; y_1)$ до точки $(x_2; y_2)$ самым коротким путём? Нужно проехать сдвиг по x и сдвиг по y :

$$\text{путь по линиям} = |x_2 - x_1| + |y_2 - y_1|.$$

Маленькие цифры 1 и 2 внизу – номера точек: x_1 – x первой точки, x_2 – x второй. Палочки $||$ значат «без минуса»: из большего числа вычитаем меньшее. Из (1; 2) в (6; 8): по x $6 - 1 = 5$, по y $8 - 2 = 6$, путь $5 + 6 = 11$ клеток. Из (6; 8) обратно в (1; 2) – те же 11: едем влево и вниз, но расстояние от этого не меняется.

Почему у всех лесенок на рисунке длина одна и та же? Все горизонтальные кусочки вместе дают сдвиг вправо на 4, все вертикальные – сдвиг вверх на 3, как их ни перемешивай. Длиннее путь станет, только если где-то поехать назад: влево или вниз.

Маршрут через несколько точек считают по кусочкам: от старта до первой точки, от первой до второй и так далее, а потом складывают. Каждый кусок – от **предыдущей** точки, а не от

старта. Потом клетки переводят в сантиметры: умножают на сторону клетки. 11 клеток по 20 см – это $11 \cdot 20 = 220$ см.

Для любопытных: путь напрямую. Если линий нет и робот умеет ехать куда угодно, короче всего – по прямой. Длину прямой находят по **теореме Пифагора** – её проходят в 8 классе, а пока хватит кнопки корня на калькуляторе:

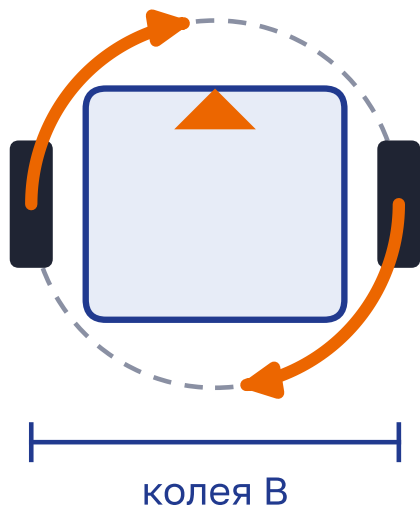
$$\text{путь напрямую} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

Сдвиг по x возводим в квадрат (умножаем сам на себя), сдвиг по y – тоже, складываем и берём **квадратный корень** – число, которое при умножении само на себя даёт то, что под корнем. На рисунке сдвиги 4 и 3: $\sqrt{16 + 9} = \sqrt{25} = 5$ клеток, ведь $5 \cdot 5 = 25$. По линиям было 7. Прямая никогда не бывает длиннее пути по линиям, а равны они, только когда точки на одной линии поля – горизонтальной или вертикальной. Одна клетка по диагонали – это $\sqrt{1 + 1} \approx 1,41$ клетки, почти в полтора раза длиннее стороны.

Запомни. По линиям: сдвиг по x плюс сдвиг по y, оба без минуса. Маршрут через точки – сумма кусков, каждый от предыдущей точки. Напрямую – корень из суммы квадратов сдвигов, и это всегда не длиннее, чем по линиям.

3.6. Поворот на колёсах: сколько крутить колесо

НА ПАЛЬЦАХ



колёса бегут
по одной окружности
её диаметр – колёса В

поворот на 90° –
четверть этой окружности

Рисунок 3.8 – При развороте на месте колёса едут по одной окружности

Колёса – как два бегуна на круглой дорожке стадиона: робот крутится на месте, а они бегут друг за другом по кругу. Повернулся робот на 180° – каждый бегун пробежал полкруга.

На поле с линиями робот поворачивает, пока датчик снова не увидит линию. А если линий нет – при навигации по координатам, – робот поворачивает вслепую: считает, на сколько повернулись колёса. Для этого на моторе ставят **энкодер** – датчик, который считает обороты

(похожий счётчик стоял у робота в задаче 1.6). Значит, надо уметь переводить поворот робота в поворот колеса.

Вспомни главу 1: при развороте на месте одно колесо крутится вперёд, другое назад, а робот вертится вокруг точки посередине между ними. Каждое колесо при этом едет по окружности. Её диаметр – **колея** B , расстояние между серединами левого и правого колёс.

Шаг 1. Путь колеса. Длина окружности – $\pi \cdot$ диаметр, где $\pi \approx 3,14$ (подробнее – в главе 2). Полный оборот робота – колесо проходит $\pi \cdot B$. Поворот на угол α – такую же долю окружности, какую α составляет от 360° :

$$\text{путь колеса} = \pi \cdot B \cdot \frac{\alpha}{360}.$$

Шаг 2. Обороты колеса. За один свой оборот колесо диаметром D проезжает $\pi \cdot D$. Делим путь на это число. Число π стоит и сверху, и снизу, поэтому сокращается:

$$\text{обороты колеса} = \frac{\pi \cdot B \cdot \alpha / 360}{\pi \cdot D} = \frac{B}{D} \cdot \frac{\alpha}{360}.$$

А если энкодер считает не обороты, а градусы, умножаем обороты на 360 и получаем совсем короткую формулу:

$$\text{угол колеса} = \alpha \cdot \frac{B}{D}.$$

Пример. Колея 12 см, колесо 6 см, поворот на 90° . Путь колеса $3,14 \cdot 12 \cdot 90 : 360 \approx 9,4$ см. Колесо за оборот проезжает $3,14 \cdot 6 \approx 18,8$ см, значит, нужно $9,4 : 18,8 = 0,5$ оборота – это 180° . По короткой формуле то же самое: $90 \cdot 12 : 6 = 180^\circ$. Колея здесь вдвое больше колеса, и колесо поворачивается вдвое больше, чем робот.

Есть и второй способ повернуть – вокруг стоящего колеса: одно колесо стоит, другое едет. Тогда едущее колесо идёт по окружности вокруг стоящего, её диаметр уже $2B$, и путь выходит вдвое длиннее: для того же робота на 90° – около 18,8 см, целый оборот колеса.

Зачем такая точность? Робот, который при каждом «направо» поворачивается на 88° вместо 90° , после десяти поворотов смотрит мимо цели на 20° . Проедет он после этого метр – и уйдёт вбок примерно на 34 см, больше чем на клетку. Не веришь – начерти в тетради прямую, а от неё транспортиром луч под углом 20° . Отмерь на луче 10 см и измерь угольником, как далеко эта точка от прямой: около 3,4 см. Увеличь в 10 раз – получишь те же 34 см. Поэтому колею и колёса меряют линейкой, а потом проверяют заездом.

Запомни. Разворот на месте: колёса едут по окружности диаметром колея B . Путь колеса – $\pi B \cdot \alpha / 360$, угол колеса – $\alpha \cdot B / D$. Вокруг стоящего колеса путь вдвое длиннее.

3.7. Разбираем задачи

Координаты в ответах книги записаны как $(x; y)$. Как вводить их на отборе – двумя числами через пробел, одной строкой или по отдельности, – всегда сказано в условии задачи.

Задача 3.1 Где окажется робот

разминка · координаты, команды «вперёд» и «поворот», курс

Робот-курьер стоит на полигоне в точке (1; 1) и смотрит вверх, вдоль оси у. Он выполняет программу:

вперёд 3; направо; вперёд 4; направо; вперёд 1; налево; вперёд 2.

«Вперёд N» – проехать N клеток, «направо» и «налево» – повернуть на месте на 90°. В какой точке робот остановится и куда будет смотреть?

Дано: старт (1; 1), курс 0° (вверх); семь команд.

Найти: точку финиша и курс.

Решение

Шаг 1. Заводим журнал. Столбцы: команда, курс, x, y. Первая строка – старт: курс 0°, x = 1, y = 1.

Шаг 2. Идём по командам. Курс после поворота считаем по правилу из раздела 3.2: направо – плюс 90°, налево – минус 90°. «Вперёд» меняет x или y по таблице из того же раздела.

Таблица 3.3

Команда	Курс	x	y
старт	0° (вверх)	1	1
вперёд 3	0°	1	4
направо	90° (вправо)	1	4
вперёд 4	90°	5	4
направо	180° (вниз)	5	4
вперёд 1	180°	5	3
налево	90° (вправо)	5	3
вперёд 2	90°	7	3

Задержимся на двух строках. Второе «направо» робот делает, глядя вправо: его «направо» на карте – вниз, курс $90 + 90 = 180^\circ$. Потом «налево», глядя вниз: $180 - 90 = 90^\circ$, и робот снова смотрит вправо, а не влево.

Шаг 3. Проверяем по сдвигам. Вправо робот проехал $4 + 2 = 6$ клеток: $1 + 6 = 7$. Вверх 3 и вниз 1, итого вверх на 2: $1 + 2 = 3$. Точка (7; 3) – та же, что в журнале.

⚠ Типичная ошибка. Путать «налево» робота с «влево» на карте. После «вперёд 1» робот смотрит вниз, и его «налево» показывает вправо по карте. Кто поехал влево, получил $x = 5 - 2 = 3$ и ответ (3; 3). Бывает и так: на повороте робота мысленно сдвигают на клетку, хотя он крутится на месте и x с y не трогает.

Ответ: (7; 3), робот смотрит вправо.

Своя задача по программе сферы, модуль 10

Реши сам

3.1. Робот стоит в точке (0; 0) и смотрит вверх. Программа: вперёд 2; направо; вперёд 3; налево; вперёд 2; налево; вперёд 1. Все повороты – на 90° на месте. В какой точке остановится робот?

3.2. Робот стоит в точке (4; 5) и смотрит вправо. Программа: вперёд 2; направо; вперёд 3; развернуться; вперёд 5; налево; вперёд 1. «Развернуться» – поворот на месте на 180° , остальные повороты – на 90° . Где остановится робот и куда он будет смотреть?

Задача 3.2 Куда смотрит робот

разминка · угол поворота, курс, наименьший поворот

Робот стоит на месте и смотрит вверх. Он поворачивается четыре раза: направо на 90° , направо на 45° , налево на 180° , налево на 90° . а) Куда он теперь смотрит? Ответ – курс в градусах. б) На какой наименьший угол и в какую сторону ему нужно повернуться, чтобы снова смотреть вверх?

Дано: начальный курс 0° ; повороты: направо 90° , направо 45° , налево 180° , налево 90° .

Найти: а) конечный курс; б) наименьший поворот до курса 0° .

Решение

Шаг 1. Поворачиваем по очереди. Направо – прибавляем, налево – вычитаем.

- направо на 90° : $0 + 90 = 90^\circ$ (смотрит вправо);
- направо на 45° : $90 + 45 = 135^\circ$ (вправо-вниз);
- налево на 180° : $135 - 180 = -45$. Меньше нуля – прибавляем 360° :
 $-45 + 360 = 315^\circ$ (влево-вверх);
- налево на 90° : $315 - 90 = 225^\circ$.

Курс 225° – по диагонали вниз-влево (посмотри на розу направлений в разделе 3.2).

Шаг 2. То же одним действием. Сложим все повороты сразу, направо со знаком плюс, налево с минусом: $90 + 45 - 180 - 90 = -135$. Робот в сумме повернулся на 135° налево: $0 - 135 = -135$, $-135 + 360 = 225^\circ$. Результат тот же.

Шаг 3. Как вернуться к курсу 0° . «Вверх» – это 0° , но для подсчёта удобнее считать его 360° . Направо нужно повернуться на $360 - 225 = 135^\circ$. Налево – на оставшиеся $360 - 135 = 225^\circ$. Меньший поворот – направо на 135° . Это видно и без счёта: в сумме робот повернулся на 135° налево, и чтобы это отменить, достаточно тех же 135° направо.

⚠ Типичная ошибка. Получить $135 - 180 = -45$ и написать «курс -45° » или вовсе забыть про минус и записать 45° . Курс 45° смотрит вверх-вправо, а -45° – это 315° , вверх-влево: совсем другая сторона. А в пункте б часто пишут «налево на 225° »: повернуться так можно, но это не наименьший поворот.

Ответ: а) 225° ; б) направо на 135° .

Своя задача по программе сферы, модуль 10

Реши сам

3.3. Робот стоит в точке (2; 2) и смотрит вправо. Ему нужно ехать напрямую к точке (5; 5). На какой наименьший угол и в какую сторону ему повернуться?

3.4. Робот проехал 30 см, повернул направо на 40° , снова проехал 30 см, снова повернул направо на 40° – и так далее, пока не вернулся в точку старта. Сколько отрезков по 30 см он проехал?

Задача 3.3 Маршрут на плане

масштаб, перевод единиц, клетки поля

Полигон нарисован на плане в масштабе 1 : 20. Маша провела на плане маршрут робота из трёх отрезков и измерила их линейкой: 5 см, 7,5 см и 2,5 см. Сторона клетки на настоящем полигоне – 25 см. а) Какой длины маршрут на полигоне? Ответ дай в метрах. б) Сколько клеток проедет робот?

Дано: масштаб 1 : 20; на плане отрезки 5 см, 7,5 см, 2,5 см; клетка поля 25 см.

Найти: а) длину маршрута на поле, м; б) число клеток.

Решение

Шаг 1. Длина маршрута на плане. $5 + 7,5 + 2,5 = 15$ см.

Шаг 2. С плана на поле. Масштаб 1 : 20 – каждый сантиметр плана на поле в 20 раз длиннее. Умножаем: $15 \cdot 20 = 300$ см.

Шаг 3. В метры. В метре 100 см: $300 : 100 = 3$ м.

Шаг 4. Клетки. Одна клетка – 25 см: $300 : 25 = 12$ клеток.

Шаг 5. Тот же ответ через клетки плана. Клетка поля на плане – $25 : 20 = 1,25$ см. Тогда отрезки на плане – это $5 : 1,25 = 4$, $7,5 : 1,25 = 6$ и $2,5 : 1,25 = 2$ клетки. Каждый отрезок уложился в целое число клеток, как и положено маршруту по линиям. $4 + 6 + 2 = 12$ – столько же. Три метра – меньше двух ростов взрослого человека: для полигона в кабинете кружка размер обычный.

 **Типичная ошибка.** Поделить на масштаб: $15 : 20 = 0,75$ см. Настоящий маршрут вышел бы короче нарисованного, а так не бывает. И обидная мелочь: написать в ответ «300», когда просили метры. Автоматическая проверка не знает, что ты имел в виду сантиметры, и снимет балл.

Ответ: а) 3 м; б) 12 клеток.

Своя задача по программе сферы, модуль 10

Реши сам

3.5. План полигона нарисован в масштабе 1 : 50. На плане линия от старта до финиша – 7,2 см. Какой длины эта линия на настоящем полигоне? Ответ дай в метрах.

3.6. Полигон 3,6 м × 2,4 м нужно нарисовать на листе, где под рисунок оставлено место 20 см × 15 см. Масштабы на выбор: 1 : 10, 1 : 15, 1 : 20, 1 : 25. С каким из них рисунок получится самым крупным и при этом поместится?

Задача 3.4 Три груза по порядку

маршрут по точкам, путь по линиям, повороты

Робот стоит в точке S (1; 1) и смотрит вверх. Ему нужно забрать грузы в точках A (4; 3), B (2; 5) и C (5; 1) – строго в этом порядке. Ездит он только по линиям поля и к каждой следующей точке едет так: сначала вдоль оси x, потом вдоль оси y. Поворачивает на месте на 90°. Сторона клетки – 25 см.

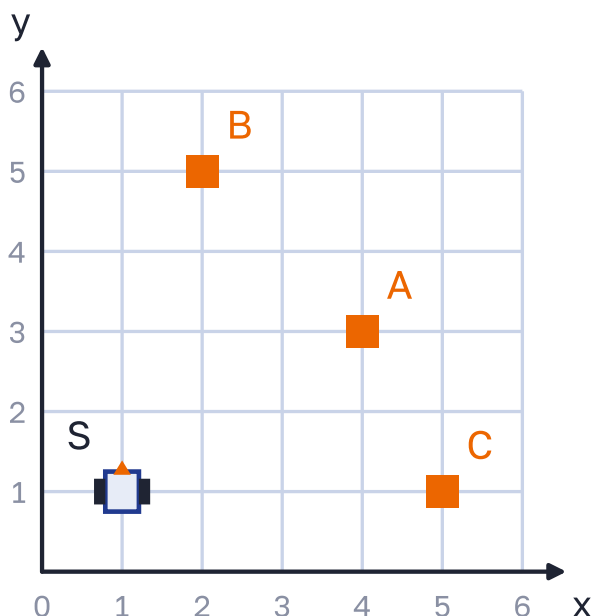


Рисунок 3.9 – План полигона к задаче 3.4

а) Сколько сантиметров проедет робот от S до C? б) Сколько раз он повернёт направо и сколько налево?

Дано: S (1; 1), курс 0°; A (4; 3), B (2; 5), C (5; 1); сначала по x, потом по y; клетка 25 см.

Найти: а) длину пути, см; б) число поворотов направо и налево.

Решение

Шаг 1. Длина каждого куска. Считаем по линиям: сдвиг по x плюс сдвиг по y, каждый кусок – от предыдущей точки.

- S → A: по x $4 - 1 = 3$, по y $3 - 1 = 2$, всего 5 клеток;
- A → B: по x $4 - 2 = 2$ (влево), по y $5 - 3 = 2$, всего 4 клетки;
- B → C: по x $5 - 2 = 3$, по y $5 - 1 = 4$ (вниз), всего 7 клеток.

Шаг 2. Весь путь. $5 + 4 + 7 = 16$ клеток. В сантиметрах: $16 \cdot 25 = 400$ см.

Шаг 3. Повороты. Проедем маршрут по командам и будем следить за курсом.

- Из S к A: надо вправо (90°), а робот смотрит вверх (0°) – **направо**. Вперёд 3 – он в (4; 1). Дальше вверх (0°): с 90° на 0° – **налево**. Вперёд 2 – он в A (4; 3).
- Из A к B: надо влево (270°), робот смотрит вверх – **налево**. Вперёд 2 – в B (2; 3). Дальше вверх: с 270° на 0° – **направо**. Вперёд 2 – в B (2; 5).
- Из B к C: надо вправо, робот смотрит вверх – **направо**. Вперёд 3 – в (5; 5). Дальше вниз (180°): с 90° на 180° – **направо**. Вперёд 4 – в C (5; 1).

Считаем: направо – 4 раза, налево – 2 раза.

Шаг 4. Сверяем с концом маршрута. Четыре поворота направо и два налево – в сумме $4 \cdot 90 - 2 \cdot 90 = 180^\circ$ направо. Начиная с курса 0° , значит, закончить должен с курсом 180° , то есть глядя вниз. Последний кусок он и правда ехал вниз.

 **Типичная ошибка.** Считать каждый кусок от старта или от начала отсчёта: например, решить, что от A до B надо проехать 5 клеток вверх, потому что у B $y = 5$. Кусок считается от предыдущей точки: от $y = 3$ до $y = 5$ – две клетки. В пункте б путаница другая: «направо» определяют по карте. Из A к B робот едет влево по карте и поворачивает для этого налево, потому что смотрит вверх. А у точки (2; 3), глядя влево, он поворачивает к «вверх» уже направо.

Ответ: а) 400 см (4 м); б) направо 4 раза, налево 2 раза.

По мотивам задачи второго отборочного этапа НТО 2024/25, профиль «Интеллектуальные робототехнические системы», инженерный тур, «Планирование движения робота»

Реши сам

3.7. Сторона клетки полигона – 30 см. Робот стартует в точке (0; 0) и по линиям объезжает точки A (3; 2), B (1; 4) и C (5; 4) в этом порядке. Какой самый короткий путь он проедет от старта до точки C? Ответ дай в сантиметрах.

3.8. Робот стоит в точке (0; 0). Ему нужно забрать три груза: груз 1 лежит в точке (6; 1), груз 2 – в (1; 5), груз 3 – в (5; 5). Порядок любой, ездит робот только по линиям, возвращаться на старт не нужно. В каком порядке объехать грузы, чтобы путь был самым коротким? Ответ – номера грузов подряд, например 312.

Задача 3.5 На сколько повернуть колесо

разворот на месте, колея, путь колеса, угол поворота колеса

У робота колея 15 см, диаметр колёс 6 см. На моторах стоят энкодеры, и в программе поворот задают углом поворота колеса в градусах. Робот должен развернуться на месте направо на 90° (левое колесо крутится вперёд, правое назад). На какой угол должно повернуться каждое колесо?

Дано: $B = 15$ см; $D = 6$ см; поворот робота $\alpha = 90^\circ$ на месте.

Найти: угол поворота колеса, в градусах.

Решение

Шаг 1. Что происходит. При развороте на месте колёса едут по одной окружности с центром посередине робота. Её диаметр – колея, 15 см (рисунок в разделе 3.6).

Шаг 2. Длина этой окружности. $\pi \cdot 15 = 3,14159 \cdot 15 \approx 47,12$ см.

Шаг 3. Путь колеса. 90° – это четверть полного оборота ($90:360 = 0,25$). Колесо проедет четверть окружности: $47,12 \cdot 0,25 \approx 11,78$ см.

Шаг 4. Путь за один оборот колеса. $\pi \cdot 6 = 3,14159 \cdot 6 \approx 18,85$ см.

Шаг 5. Сколько оборотов. $11,78:18,85 = 0,625$ оборота.

Шаг 6. В градусы. Один оборот колеса – 360° : $0,625 \cdot 360 = 225^\circ$.

Шаг 7. То же по короткой формуле. Угол колеса = $\alpha \cdot B:D = 90 \cdot 15:6 = 225^\circ$, как и в шаге 6. Колея в $15:6 = 2,5$ раза больше колеса, поэтому колесо поворачивается в 2,5 раза больше, чем робот: $90 \cdot 2,5 = 225$.

⚠ Типичная ошибка. Решить, что колесо повернётся на те же 90° , что и робот. Тогда робот повернётся всего на $90 \cdot 6:15 = 36^\circ$ – даже до диагонали не дотянет. Бывает и наоборот: колею принимают за радиус окружности, а не за диаметр. Выйдет 450° , и робот повернётся на 180° вместо 90° . Колея – это расстояние от колеса до колеса, то есть весь поперечник круга.

Ответ: 225° .

Своя задача по программе сферы, модуль 6

Реши сам

3.9. У робота колея 16 см и колёса диаметром 8 см. Он разворачивается на месте на 180° . Сколько оборотов сделает каждое колесо?

3.10. У робота колея 10 см и колёса диаметром 5 см. Во время разворота на месте энкодеры показали, что каждое колесо повернулось на 300° . На какой угол повернулся робот?

Задача 3.6* Напрямую или по линиям

расстояние по линиям, расстояние по прямой, теорема Пифагора

Робот стоит в точке (2; 1), и ему нужно попасть в точку (7; 13). Сторона клетки – 20 см. Старая программа водит его только по линиям поля. Новая умеет повернуть робота на нужный угол и ехать напрямую. На сколько сантиметров путь напрямую короче пути по линиям?

Дано: старт (2; 1), цель (7; 13); клетка 20 см.

Найти: разницу путей по линиям и напрямую, см.

Решение

Шаг 1. Сдвиги. По x : $7 - 2 = 5$ клеток. По y : $13 - 1 = 12$ клеток.

Шаг 2. Путь по линиям. $5 + 12 = 17$ клеток, в сантиметрах $17 \cdot 20 = 340$ см.

Шаг 3. Путь напрямую – по теореме Пифагора (раздел 3.5, «Для любопытных»). Квадраты сдвигов: $5^2 = 25$, $12^2 = 144$. Сумма $25 + 144 = 169$. Корень: $\sqrt{169} = 13$ клеток, потому что $13 \cdot 13 = 169$. В сантиметрах $13 \cdot 20 = 260$ см.

Шаг 4. Разница. $340 - 260 = 80$ см.

Шаг 5. А если сразу в сантиметрах? Сдвиги: $5 \cdot 20 = 100$ см и $12 \cdot 20 = 240$ см.

$\sqrt{100^2 + 240^2} = \sqrt{10\,000 + 57\,600} = \sqrt{67\,600} = 260$ см – то же самое. Прямая вышла короче, но не в разы: цель почти прямо над роботом, и срезать тут особенно нечего. Будь сдвиги равными, выигрыш был бы заметнее.

⚠ Типичная ошибка. Сложить сдвиги и взять корень из суммы: $\sqrt{5 + 12} \approx 4,1$ клетки. Такой путь короче даже сдвига по y , а робот никак не может подняться на 12 клеток, проехав 4. Квадраты нужно взять **до** сложения. И не спутай вопрос: спрашивают разницу путей, а не длину одного из них – ответ «260» или «340» будет неверным.

Ответ: на 80 см.

Своя задача по программе сферы, модуль 10

Реши сам

3.11.* Сторона клетки – 25 см. Робот едет из точки (0; 0) в точку (4; 7). Найди длину пути по линиям и длину пути напрямую. Путь напрямую округли до целых сантиметров.

3.12.* Робот стоит в точке (0; 0). Груз 1 лежит в точке (6; 1), груз 2 – в точке (4; 4). Какой груз ближе, если ехать по линиям? А если напрямую? Ответ – два номера подряд: сначала для пути по линиям, потом для пути напрямую.

Задача 3.7* Сколько есть коротких дорог

кратчайшие маршруты по сетке, подсчёт вариантов, обязательная точка

Робот-разведчик едет по линиям поля из точки (0; 0) в точку (4; 3) самым коротким путём. По дороге он обязан проехать через контрольную точку К (2; 1) – там его фотографирует камера. Сколько существует разных самых коротких маршрутов через К?

Дано: старт (0; 0), финиш (4; 3), обязательная точка К (2; 1); путь – самый короткий, по линиям.

Найти: число маршрутов.

Решение

Шаг 1. Какие маршруты самые короткие. По линиям от (0; 0) до (4; 3) самый короткий путь – $4 + 3 = 7$ клеток. Такой путь получается, только если робот едет **вправо и вверх** и ни разу не сворачивает влево или вниз. Каждый шаг влево или вниз потом пришлось бы отыгрывать, и путь бы удлинился.

Шаг 2. Сколькими способами попасть в перекрёсток. В каждый перекрёсток робот въезжает последним шагом либо слева, либо снизу. Значит, число способов попасть в перекрёсток = способы попасть в соседа слева + способы попасть в соседа снизу. На нижней линии и на левой – по одному способу: туда можно только ехать прямо, не сворачивая.

Шаг 3. Считаем до К. До К (2; 1) робот не может заехать выше $y = 1$ и правее $x = 2$ – иначе в К ему пришлось бы ехать вниз или влево. Нижняя линия: 1, 1, 1. Перекрёсток (1; 1): $1 + 1 = 2$. Перекрёсток К (2; 1): слева 2, снизу 1, итого $2 + 1 = 3$ способа.

Шаг 4. Считаем от К до финиша. Дальше робот едет только правее К и выше К. Все маршруты проходят через К, поэтому в К ставим 3 и считаем тем же правилом:

- правее К по линии $y = 1$: 3 и 3; выше К по линии $x = 2$: 3 и 3;
- (3; 2): $3 + 3 = 6$; (4; 2): $6 + 3 = 9$;
- (3; 3): $3 + 6 = 9$; (4; 3): $9 + 9 = 18$.

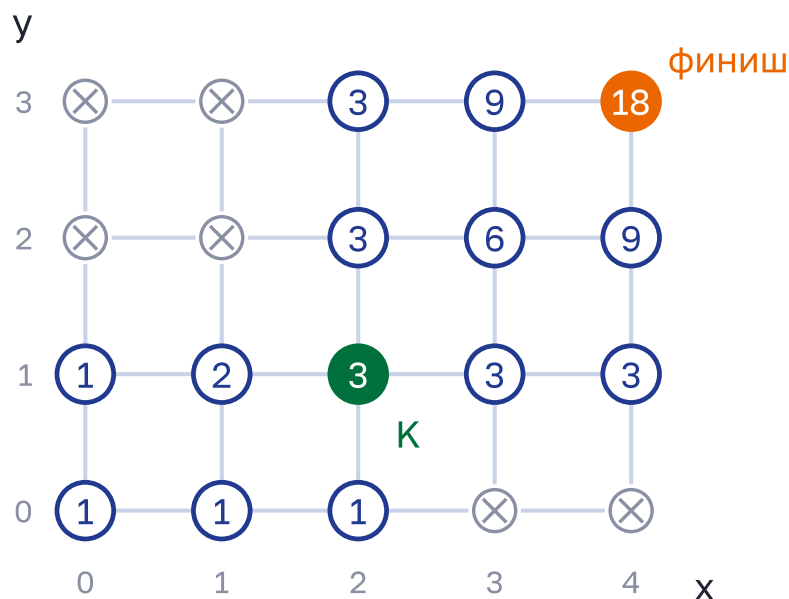


Рисунок 3.10 – Числа у перекрёстков – сколькими самыми короткими путями через К туда можно попасть. Крестики – перекрёстки, где такой путь не проходит

Шаг 5. Короче – умножением. От старта до К – 3 маршрута. От К до финиша (2 вправо и 2 вверх) – тем же правилом выходит 6. Любой из трёх первых кусков можно продолжить любым из шести вторых: $3 \cdot 6 = 18$. Если убрать камеру и посчитать тем же правилом всю сетку, в финише выйдет 35 – обязательная точка отсекала почти половину путей.

⚠ Типичная ошибка. Сложить маршруты двух половин, $3 + 6 = 9$, вместо того чтобы умножить. Каждый маршрут до К соединяется с **каждым** маршрутом после К, поэтому их 3 раза по 6. Другая ловушка – в таблице прибавить к перекрёсткам выше К те маршруты, что обходят К стороной (например, через (1; 2)). Их брать нельзя: такие пути мимо камеры.

Ответ: 18 маршрутов.

По мотивам задачи первого отборочного этапа НТО 2024/25, профиль «Интеллектуальные робототехнические системы», предметный тур по информатике, «Ладья и обязательная клетка»

Реши сам

3.13.* Робот едет по линиям из точки $(0; 0)$ в точку $(3; 3)$ самым коротким путём. Перекрёсток $(1; 1)$ закрыт: на нём стоит банка-препятствие. Сколько существует самых коротких маршрутов, которые его объезжают?

3.14.* Робот едет по линиям из точки $(0; 0)$ в точку $(5; 3)$ самым коротким путём и обязательно через точку $(2; 2)$. Сколько существует таких маршрутов?

Алгоритмы и исполнители



В этой главе нет ни формул, ни единиц измерения – только робот на клетчатом поле и короткие программы для него. Зато здесь нужна внимательность: одна перепутанная команда «налево» уводит робота совсем в другой угол поля. Мы научимся проходить программу шаг за шагом, сокращать её циклом, собирать из подпрограмм, выводить робота из лабиринта по правилу правой руки и замечать, когда он попал в круг, из которого не выйдет. На отборе такая задача – поле с роботом и несколько строк программы, а спрашивают, где робот окажется, какую команду вставить или сколько команд он выполнит. В ответ вводится число, клетка или номер варианта.

4.1. Исполнитель и его команды

НА ПАЛЬЦАХ

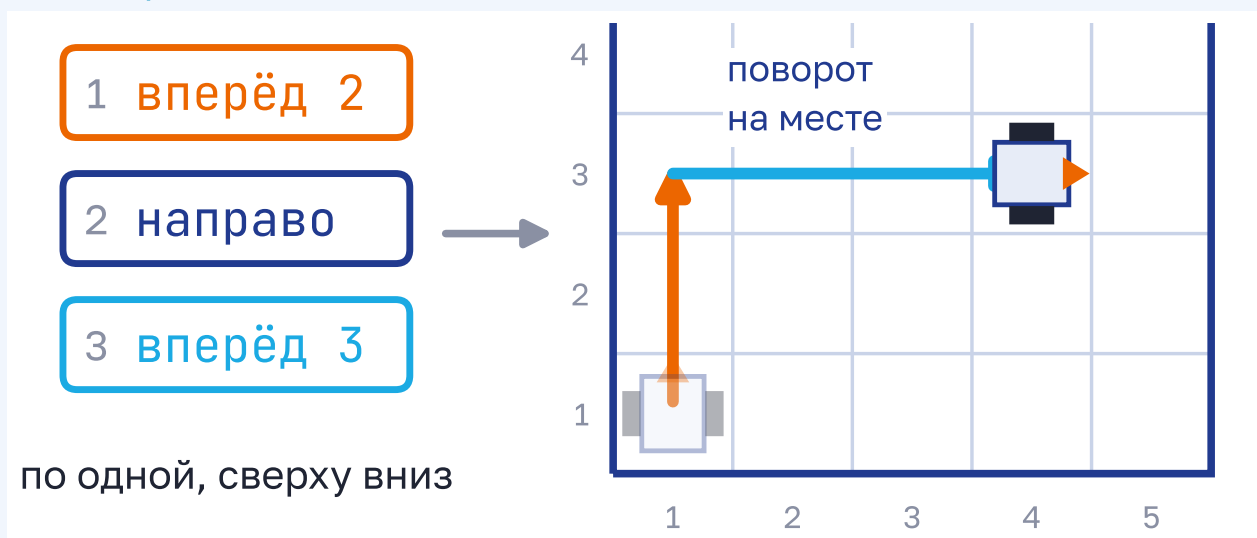


Рисунок 4.1 – Три карточки с командами – и три участка пути робота

Робот берёт карточки по одной, сверху вниз, и делает то, что на них написано. Сам он ничего не придумывает.

Исполнитель – тот, кто выполняет команды. Это может быть робот на полигоне, черепашка на экране компьютера или твой младший брат, которому ты объясняешь дорогу до остановки. У каждого исполнителя свой набор команд, который он понимает. Этот набор называют **системой команд исполнителя** (коротко СКИ). Команду не из своего списка исполнитель выполнить не может, даже если человеку она кажется простой.

Алгоритм – точный порядок действий, который приводит к цели. Записанный командами исполнителя, он становится **программой**. Главное слово тут – «точный». «Иди куда-нибудь направо» – пожелание, исполнитель с ним ничего не сделает. «Вперёд 3, направо, вперёд 2» – алгоритм.

Наш исполнитель на всю главу – робот на клетчатом поле. Поле расчерчено на клетки, робот всегда стоит в центре какой-то клетки и смотрит в одну из четырёх сторон: вверх, вправо, вниз или влево (по листу). Клетку записываем двумя числами: сначала номер столбца (считаем слева направо), потом номер строки (снизу вверх). Клетка (4; 3) – четвёртый столбец, третья строка. В главе 3 точки отмечались на пересечениях линий, здесь робот стоит внутри клеток, но порядок чисел тот же: сначала «вправо», потом «вверх».

Система команд робота:

- вперёд N – проехать N клеток в ту сторону, куда робот смотрит; вперёд 1 – на одну клетку;
- направо – повернуться на месте на четверть оборота по часовой стрелке (на 90°);
- налево – повернуться на месте на четверть оборота против часовой стрелки.

Поворот не сдвигает робота из клетки, он меняет только направление взгляда. Край поля и стены – препятствие. Команду вперёд N робот выполняет клетка за клеткой. Если по дороге

встретилась стена или край поля, он доезжает до препятствия, врезается – и программа останавливается с ошибкой. Команды после этой уже не выполняются.

Направо – с точки зрения робота, а не листа. Робот смотрит вверх – его «направо» совпадает с правой стороной листа. Но если робот смотрит вниз, его «направо» – это левая сторона листа, а «налево» – правая. Подробно про это в главе 3. Если запутался, поверни тетрадь так, чтобы робот смотрел от тебя, или встань и повернись вместе с ним.

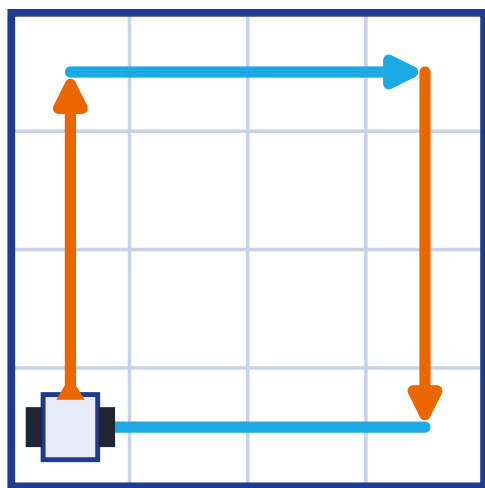
Линейный алгоритм – команды выполняются одна за другой, по одному разу, без пропусков и повторов. Так устроена программа на картинке выше: `вперёд 2`, `направо`, `вперёд 3`. Робот из клетки (1; 1), смотревший вверх, приедет в клетку (4; 3) и будет смотреть вправо.

Сколько команд нужно, чтобы доехать. Если нужна самая короткая программа, считай так: каждый прямой отрезок пути – одна команда `вперёд N`, каждый поворот на 90° – одна команда, разворот на месте – две (`направо`, `направо`). Чтобы попасть из (1; 1) в (4; 3), надо сдвинуться и вправо, и вверх. Значит, прямых отрезков нужно хотя бы два, и между ними – поворот. Получается 3 команды – меньше нельзя.

Запомни. Робот делает только то, что есть в его системе команд, и строго по порядку. Повороты меняют взгляд, а не клетку, и «направо» считаем от носа робота. Клетка (x; y): сначала столбец, потом строка.

4.2. Циклы: «повтори» и «пока»

НА ПАЛЬЦАХ



повтори 4:
вперёд 3
направо

4 раза по 2 команды –
8 команд робот выполнит,
а в программе 3 строки

Рисунок 4.2 – Четыре одинаковых куска пути – одна запись «повтори 4»

Учитель физкультуры не говорит «присядь, присядь, присядь...» двадцать раз. Он говорит «двадцать приседаний». Цикл – то же самое для робота.

Чтобы робот объехал квадрат со стороной в 3 шага, можно написать восемь команд подряд: `вперёд 3`, `направо`, `вперёд 3`, `направо` и так ещё два раза. Длинно, и легко сбиться со счёта.

Удобнее сказать роботу: «повтори четыре раза: вперёд 3, направо». Такая конструкция называется **циклом**.

```
повтори 4:  
  вперёд 3  
  направо
```

Команды, сдвинутые вправо под строкой `повтори`, – **тело цикла**: они повторяются. Число после слова `повтори` говорит, сколько раз. Это **цикл со счётчиком**: робот сам считает, сколько кругов уже сделал. Как только тело выполнено нужное число раз, робот переходит к строке, которая стоит после цикла без сдвига.

Сколько команд в программе и сколько робот выполнит – разные вещи. В записи выше 3 строки: сам `повтори 4` и две команды тела. А выполнит робот $4 \cdot 2 = 8$ команд движения и поворота. В задачах про это спрашивают по-разному, поэтому договоримся для всей главы:

- **длина программы** – число строк-команд в записи; `повтори N:` – тоже команда, тело считаем один раз;
- **выполненные команды** – сколько раз робот сделал `вперёд`, `направо` или `налево`; `вперёд 3` – одна команда, хотя клеток три. Сама строка `повтори` роботом «не делается», её не считаем.

Как считать, пишут в условии задачи. Прочитай это место до того, как начнёшь считать, а не после.

Как сократить программу циклом. Ищи кусок, который повторяется подряд несколько раз без изменений. Например, `вперёд 1, направо, вперёд 1, налево, вперёд 1, направо, вперёд 1, налево` – это дважды один и тот же кусок из четырёх команд, то есть `повтори 2:` и четыре команды в теле. Если после повторов остался хвостик – часть куска, – его пишут отдельно после цикла. Число повторов – сколько раз кусок встретился **целиком**.

Цикл с условием. Иногда заранее неизвестно, сколько раз повторять. Робот стоит где-то в коридоре и должен доехать до стены. Тогда пишут:

```
пока впереди свободно:  
  вперёд 1
```

Перед каждым повтором робот смотрит датчиком вперёд. Свободно – делает шаг и снова проверяет. Стена – цикл заканчивается, и робот идёт к следующей строке. Если впереди стена с самого начала, тело не выполнится ни разу. Условия, которые понимает наш робот: «впереди свободно», «справа свободно», «слева свободно» и обратные к ним «впереди стена», «справа стена», «слева стена».

Запомни. `повтори N:` – тело выполняется N раз. `пока условие:` – тело выполняется, пока условие верно; проверка перед каждым повтором. Длина программы и число выполненных команд – разные числа: у `повтори 5:` с тремя командами в теле длина 4, а выполнено 15.

Совет. Сокращаешь программу – сначала подчеркни в ней одинаковые куски разными цветами. Сразу видно, сколько их и что осталось в хвосте.

4.3. Ветвление: «если – иначе»

НА ПАЛЬЦАХ

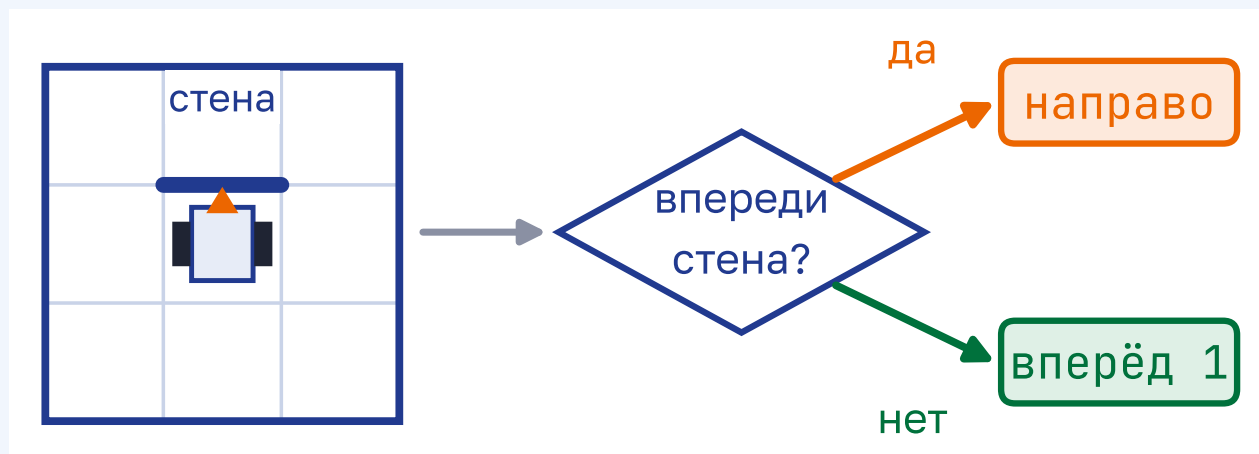


Рисунок 4.3 – Робот упёрся в стену и спрашивает датчик

Как ты утром смотришь в окно: идёт дождь – берёшь зонт, нет – не берёшь. Робот тоже выбирает одну из двух дорог по ответу датчика.

Ветвление – место в программе, где робот выбирает, что делать, в зависимости от **условия**. Условие – вопрос, на который ответ «да» или «нет»: «впереди стена?», «справа свободно?». Запись такая:

```
если впереди стена:  
    направо  
иначе:  
    вперёд 1
```

Если условие верно, выполняются команды под строкой **если**, а команды под **иначе** пропускаются. Если неверно – наоборот. Одна из двух веток выполняется всегда, обе сразу – никогда. Ветку **иначе** можно не писать вовсе: тогда при неверном условии робот просто ничего не делает и идёт дальше.

Бывает, что вариантов больше двух. Тогда проверки ставят цепочкой: **если ...**, **иначе если ...**, **иначе если ...**, **иначе**. Робот проверяет условия сверху вниз и выполняет ветку **первого** верного, остальные даже не смотрит. Поэтому порядок проверок меняет поведение робота – на этом держится правило правой руки из раздела 4.5.

Ветвление почти всегда живёт внутри цикла. Одна проверка – это один выбор на одном перекрёстке. А чтобы робот принимал решение на каждом шаге, проверку кладут в тело цикла **пока**.

Запомни. В ветвлении выполняется одна ветка: под `если` – когда условие верно, под `иначе` – когда нет. В цепочке `если – иначе если` побеждает первое верное условие.

4.4. Подпрограммы

НА ПАЛЬЦАХ

подпрограмма

ступенька:

вперёд 1

направо

вперёд 1

налево

программа

ступенька

ступенька

ступенька

три вызова –
три ступеньки

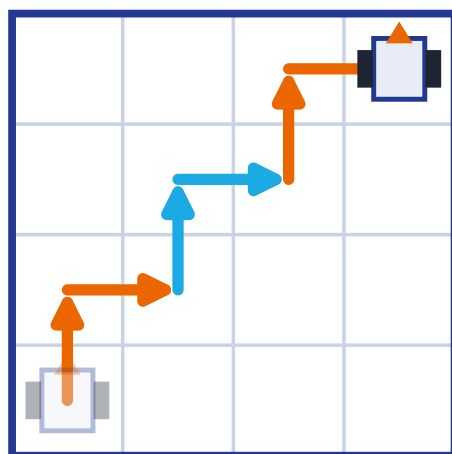


Рисунок 4.4 – Кусок программы получил имя – и вызывается одним словом

В рецепте пишут «приготовь тесто», а как его готовить – описано один раз на другой странице. Подпрограмма – такая же страница для робота.

Подпрограмма – кусок программы, которому дали имя. Её описывают один раз, а потом **вызывают** – пишут имя в том месте, где нужны эти команды. Встретив имя, робот выполняет все команды подпрограммы по порядку и возвращается к строке, которая идёт после вызова.

подпрограмма ступенька:

вперёд 1

направо

вперёд 1

налево

ступенька

ступенька

ступенька

Описание подпрограммы (первые пять строк) робот сам по себе не выполняет – это как рецепт на полке. Работа начинается с первого вызова. Здесь их три, и робот из (1; 1), смотревший вверх, поднимется по трём ступенькам в (4; 4) – как на картинке.

Зачем это нужно? Программу проще читать: «ступенька, ступенька, ступенька» понятнее двенадцати строк. Ошибку исправляешь в одном месте, а не в двенадцати. И подпрограмму можно вызывать внутри цикла и даже внутри другой подпрограммы. В языке C++ для Arduino подпрограммы называют **функциями** – с ними ты встретишься в главе 6.

Сколько команд выполнит робот? Каждый вызов – это все команды подпрограммы. Ступенька – 4 команды, три вызова – $3 \cdot 4 = 12$. Вызов внутри цикла умножается на число повторов.

Запомни. Подпрограмму описывают один раз и вызывают по имени сколько угодно раз. Само описание ничего не двигает. Команд выполнено = (команд в подпрограмме) · (число вызовов).

4.5. Лабиринт и правило правой руки

НА ПАЛЬЦАХ

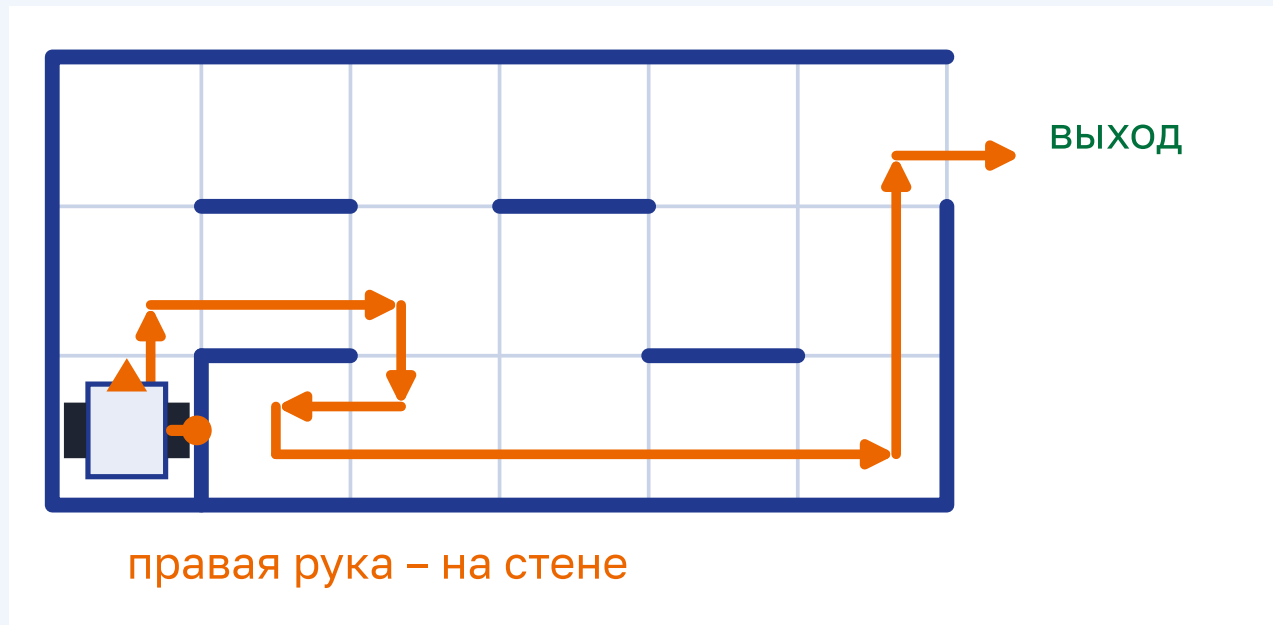


Рисунок 4.5 – Робот ведёт правой рукой по стене – и выходит

В тёмной комнате положи правую руку на стену и иди, не отрывая её. Рано или поздно рука приведёт тебя к двери.

Лабиринт на поле – это клетки со стенами между некоторыми из них. Карты у робота нет, он видит только стены рядом с собой: датчики смотрят вперёд, вправо и влево. Нужен алгоритм, который выводит к выходу, даже если лабиринт видишь впервые. Самый известный – **правило правой руки**: двигаться так, будто правая рука всё время касается стены.

В каждой клетке робот проверяет стороны строго по порядку:

1. справа свободно – повернуть направо и шагнуть;
2. иначе впереди свободно – шагнуть прямо;
3. иначе слева свободно – повернуть налево и шагнуть;
4. иначе (кругом стены, это тупик) – развернуться и шагнуть обратно.

На языке нашего робота:

```
пока не вышел:
  если справа свободно:
    направо
    вперёд 1
  иначе если впереди свободно:
    вперёд 1
  иначе если слева свободно:
    налево
    вперёд 1
  иначе:
    направо
    направо
    вперёд 1
```

Заметь: после каждого поворота стоит **вперёд 1**. Без него робот, повернув направо, на следующей проверке снова найдёт справа свободное место – хотя бы то, откуда пришёл, – и повернётся ещё раз. Вместо того чтобы идти вдоль стены, он закружится на месте или поедет назад. Забытый шаг после поворота ломает всё правило.

Правило левой руки – то же самое зеркально: сначала проверяем слева, потом впереди, потом справа. Оба правила выводят из лабиринта, если все его стены соединены с внешней стеной. Короче ли путь по одной руке, чем по другой, заранее не скажешь – зависит от лабиринта. Кратчайший путь они находят редко: рука честно заводит робота в каждый тупик, который попадётся ей по дороге.

Если внутри лабиринта есть «остров» – кусок стены, не соединённый с краями, – робот, начавший у острова, может ходить вокруг него по кругу бесконечно. Вспомни об этом, если в задаче спрашивают, всегда ли правило срабатывает.

Путь обратно. Допустим, робот прошёл от входа до выхода без заездов в тупики и записал, что делал на каждой развилке: «направо», «прямо» или «налево». Чтобы вернуться, он разворачивается и проходит записанный список **с конца**, заменяя каждое «направо» на «налево» и наоборот. «Прямо» остаётся «прямо». Почему замена? Ты шёл на север и свернул направо, на восток. Обратно ты придёшь на ту же развилку с востока, лицом на запад, а тебе нужно на юг – это поворот налево.

Запомни. Правая рука: справа → прямо → слева → разворот, в таком порядке, и после каждого поворота – шаг. Левая рука – зеркально. Путь обратно: список развилки с конца, «направо» ↔ «налево».

4.6. Трассировка и поиск ошибок

НА ПАЛЬЦАХ

- 1 вперёд 2
 - 2 направо
 - ▶ 3 вперёд 3
 - 4 налево
 - 5 вперёд 1
- после строки 3



Рисунок 4.6 – Палец на строке программы – робот на поле

Ведёшь пальцем по программе и после каждой строки отмечаешь на поле, где робот и куда смотрит. Это и называется трассировкой.

Трассировка – выполнение программы «вручную», на бумаге, строка за строкой. Удобнее всего таблица из четырёх колонок: номер шага, команда, клетка робота, куда он смотрит. После каждой команды – новая строка таблицы. Даже если «и так понятно», не перепрыгивай через шаги: ошибки прячутся на поворотах.

Цикл в трассировке разворачивай полностью: при **повтори 3** тело пишешь три раза, строка за строкой. Подпрограмму тоже раскрывай – вместо имени вписывай её команды. Позже, когда набьёшь руку, заметишь закономерность («за один круг робот сдвигается на 2 клетки вправо») и сможешь считать сразу. Но первые задачи честно трассируй до конца.

Поиск ошибки в программе (программисты говорят «отладка») – та же трассировка, только ты сравниваешь, что **должно** быть после каждой строки и что **получилось**. Первая строка, где разошлось, – подозреваемая. Самые частые ошибки в программах для робота на поле:

- **перепутаны направо и налево** – чаще когда робот смотрит вниз или влево и хочется повернуть «по листу»;
- **ошибка заборчика** – в ряду из 6 клеток между первой и последней всего 5 шагов, а не 6: столбиков забора на один больше, чем пролётов;
- **не то число повторов** – на один больше или меньше, чем нужно;
- **команда не на своём месте** – поворот попал в тело цикла, хотя должен стоять после него, или наоборот;
- **забыт шаг после поворота** в правиле руки.

Для любопытных: трассировку может сделать компьютер. Этот кусок можно пропустить – все задачи главы решаются карандашом. В главе 3 была программа на Python, которая ведёт журнал робота по таблице курсов. Здесь – её родственница: поворот она делает без таблицы

и вдобавок следит за краем поля. Если Python ты видишь впервые, читай медленно, со строкой кода перед глазами, – разбор ниже идёт по порядку. Зато потом подставляешь свою программу и проверяешь ответ.

```
W, H = 6, 5          # поле: 6 столбцов, 5 строк
x, y = 2, 1          # клетка робота: столбец 2, строка 1
dx, dy = 0, 1         # куда смотрит: вверх
program = ["вперёд 3", "направо", "вперёд 3", "направо",
           "вперёд 2", "налево", "вперёд 2"]
crash = False
for i, cmd in enumerate(program, start=1):
    if cmd == "направо":
        dx, dy = dy, -dx
    elif cmd == "налево":
        dx, dy = -dy, dx
    else:
        n = int(cmd.split()[1])          # число после слова «вперёд»
        for step in range(n):            # едем по одной клетке
            if not (1 ≤ x + dx ≤ W and 1 ≤ y + dy ≤ H):
                crash = True              # следующая клетка – за краем поля
                break
            x, y = x + dx, y + dy
print(i, cmd, "→", x, y)
if crash:
    print("врезался в край поля")
    break
```

Разберём по строкам. Всё, что после #, – комментарий для человека, компьютер его пропускает. W, H = 6, 5 и x, y = 2, 1 кладут числа в **переменные** – ячейки памяти с именами: размер поля и клетку робота (2; 1). Направление хранится двумя числами: dx – на сколько меняется столбец за один шаг, dy – на сколько меняется строка. Вверх – это (0; 1), вправо – (1; 0), вниз – (0; -1), влево – (-1; 0). program – список команд в квадратных скобках. for i, cmd in enumerate(program, start=1): – цикл «для каждой команды из списка»: в cmd по очереди попадает очередная команда, а в i – её номер, начиная с 1. Дальше ветвление: if – «если», elif – «иначе если», else – «иначе».

Самая хитрая строка – поворот направо, dx, dy = dy, -dx. Компьютер сначала вычисляет оба числа справа от = по **старым** значениям и только потом раскладывает их: первое – в dx, второе – в dy. Проверь на примере: робот смотрел вверх, (0; 1). Справа получается 1, 0 – теперь он смотрит вправо. Ещё раз: из (1; 0) выйдет 0, -1, то есть вниз. А если записать то же двумя строками, dx = dy и потом dy = -dx, вторая строка возьмёт уже новое dx. Из «вверх» получится (1; -1) – робот смотрит наискосок, такого направления у нас нет. Налево – та же строка с минусом у другого числа: dx, dy = -dy, dx.

Команду вперёд N программа выполняет по одной клетке. Сначала cmd.split() режет текст «вперёд 3» по пробелу на два куса, [1] берёт второй кусок – «3», а int превращает текст «3» в число 3. Перед каждым шагом программа проверяет, не за краем ли следующая клетка: 1 ≤ x + dx ≤ W значит «новый столбец от 1 до W», и так же для строки. Если за краем – записывает аварию в crash и командой break прерывает внутренний цикл. print выводит номер команды, её саму и клетку робота. Второй break, после сообщения об аварии, останавливает и внешний цикл: врезавшийся робот дальше не едет. Для этой программы последние

строки вывода – 7 вперёд 2 → 6 2 и врезался в край поля. Это задача 4.1 – сейчас решим её руками.

Запомни. Трассировка – таблица «шаг – команда – клетка – куда смотрит», по строке на каждую команду. Циклы и подпрограммы при этом раскрываем. Ошибку ищи в первой строке, где «должно» и «получилось» разошлись.

Совет. Держи рядом карандаш и черновик в клетку. Нарисуй поле, поставь робота стрелочкой и веди его по программе. Минута на рисунок сэкономит тебе неверную попытку ввода ответа.

4.7. Когда робот ходит по кругу

НА ПАЛЬЦАХ



Рисунок 4.7 – Сначала робот подъезжает к кругу, потом крутится по нему без конца

Кольцевой автобус выезжает из парка, пару остановок едет к кольцу, а дальше весь день ходит по одному и тому же кругу. Где он будет вечером, можно узнать, не выписывая каждый рейс: хватит знать, сколько остановок в круге.

Бывает исполнитель совсем простой. Робот-погрузчик ездит по дорожке из пронумерованных клеток, а на каждой клетке стоит табличка с номером: «→ 5» значит «переезжай на клетку 5». Робот читает табличку, переезжает, читает следующую. Своей головы у него нет, только таблички.

Допустим, робот вернулся на клетку, где уже был. Таблички не поменялись, значит, дальше он поедет точно так же, как в прошлый раз, – и снова вернётся сюда же. Робот попал в бесконечный круг. Это называют **зацикливанием**. Для трассировки вывод простой: как только клетка повторилась, дальше писать таблицу бессмысленно, всё уже известно.

Возьмём дорожку с рисунка: робот начинает на клетке 1 и едет $1 \rightarrow 5 \rightarrow 2 \rightarrow 7 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow \dots$. Клетка 2 повторилась. До неё робот ехал 2 переезда – это **подъезд**. Круг $2 \rightarrow 7 \rightarrow 4 \rightarrow 3 \rightarrow 2$ занимает 4 переезда, это **длина круга**.

Где робот будет через много переездов. Спросим: где он окажется после 31 переезда? Выписывать 31 строку долго, и легко ошибиться. Считаем иначе.

- Отнимаем подъезд: $31 - 2 = 29$. Столько переездов робот сделал уже по кругу, начав с клетки 2.
- Делим на длину круга с остатком: $29 = 4 \cdot 7 + 1$. Семь полных кругов вернули робота на клетку 2.
- Остаток 1 – ещё один переезд по кругу от клетки 2: робот на клетке 7.

Прежде чем верить такому счёту, проверь его на маленьком числе, которое легко протрассировать руками. Три переезда: $3 - 2 = 1$, кругов ноль, остаток 1 – клетка 7. По дорожке: $1 \rightarrow 5 \rightarrow 2 \rightarrow 7$. Совпало, значит, подъезд и длину круга ты нашёл правильно.

Запомни. Клетка повторилась – робот зациклился и до финиша не доедет. Чтобы узнать, где он будет через N переездов: отними подъезд, то, что осталось, раздели на длину круга с остатком и от первой клетки круга отсчитай вперёд столько переездов, сколько вышло в остатке.

Совет. Подъезд легко посчитать на один переезд больше или меньше. Отмечай номер переезда прямо над стрелкой: $1 \rightarrow 5$ – первый, $5 \rightarrow 2$ – второй. Подъезд заканчивается там, где начинается круг.

4.8. Разбираем задачи

Задача 4.1 Доедет или врежется

разминка · линейный алгоритм, трассировка, повороты относительно робота, край поля

Поле 6×5 клеток, стен внутри нет. Робот стоит в клетке (2; 1) и смотрит вверх. Он выполняет программу:

```
вперёд 3
направо
вперёд 3
направо
вперёд 2
налево
вперёд 2
```

Выполнит ли робот программу до конца? Если да – в какой клетке он окажется. Если нет – на какой по счёту команде он врежется и в какой клетке будет стоять в этот момент.

Дано: поле 6 × 5; старт (2; 1), смотрит вверх; 7 команд.
Найти: доедет ли; клетку в конце или в момент аварии.

Решение

Шаг 1. Готовим таблицу. Записываем начальное положение, а потом по строке на каждую команду. **вперёд** меняет клетку, повороты меняют только направление. Держим в голове размер поля: столбцы от 1 до 6, строки от 1 до 5.

Шаг 2. Трассировка.

Таблица 4.1

Шаг	Команда	Клетка	Смотрит
старт	–	(2; 1)	вверх
1	вперёд 3	(2; 4)	вверх
2	направо	(2; 4)	вправо
3	вперёд 3	(5; 4)	вправо
4	направо	(5; 4)	вниз
5	вперёд 2	(5; 2)	вниз
6	налево	(5; 2)	вправо
7	вперёд 2	(6; 2), дальше край	вправо

Шаг 3. Самое опасное место – шаг 6. Робот смотрит вниз. Поверни тетрадь вверх ногами: теперь робот смотрит от тебя, и его «налево» совпадает с твоим левым. Верни тетрадь обратно – эта сторона оказалась справа по листу. Налево от робота, который смотрит вниз, – это вправо по листу.

Шаг 4. Последняя команда. Робот в (5; 2) смотрит вправо, ему нужно проехать 2 клетки. Первая – в (6; 2), это последний столбец поля. Вторая клетка была бы в седьмом столбце, а его нет: робот врежется в край. Он стоит в (6; 2), программа остановилась на 7-й команде.

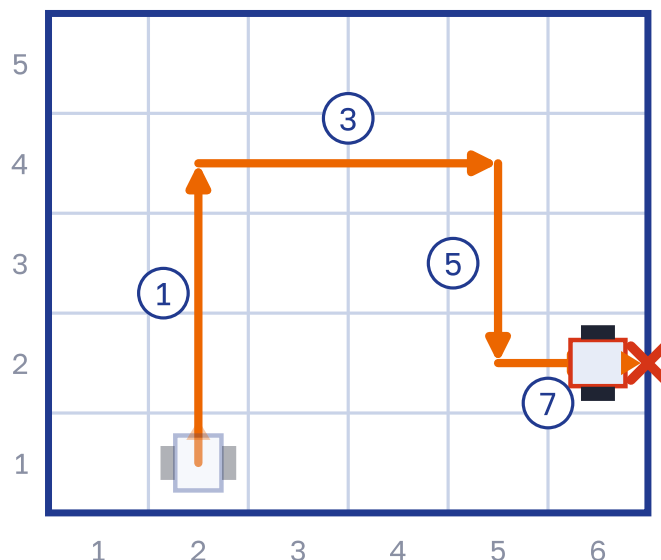


Рисунок 4.8 – Путь робота в задаче 4.1. Цифры – номера команд `вперёд`, красный крест – место аварии

Проверить себя можно программой на Python из раздела 4.6 – это та же задача, и она печатает то же самое.

❗ Типичная ошибка. На шаге 6 повернуть «налево по листу» – влево. Тогда робот спокойно проедет 2 клетки влево, в (3; 2), и покажется, что программа выполнена до конца. Одна неверно понятая команда поменяла не только клетку, но и сам ответ «доедет – не доедет». Вторая ошибка поменьше: решить, что робот при аварии остаётся в (5; 2). По нашему правилу он проезжает, сколько может, и останавливается у края.

Ответ: не доедет: робот врежется на 7-й команде, стоя в клетке (6; 2).

Своя задача по программе сферы, модуль 10

Реши сам

4.1. Поле 5×5 без стен. Робот стоит в клетке (1; 1) и смотрит вправо. Программа: `вперёд 3`, `налево`, `вперёд 2`, `налево`, `вперёд 4`. Выполнит ли робот её до конца? Если нет – на какой команде он врежется и в какой клетке будет стоять?

4.2. Поле 6×6 без стен. Робот стоит в клетке (1; 1) и смотрит вверх. Ему нужно попасть в клетку (5; 4) и в конце смотреть вниз. Какое наименьшее число команд для этого нужно? `вперёд N` с любым N считается одной командой.

Задача 4.2 Сократи программу циклом

цикл со счётчиком, сокращение программы, длина программы

Робот стоит в клетке (1; 1) поля 6×6 и смотрит вверх. Ему дали программу из 14 команд:

```
вперёд 1
направо
вперёд 1
налево
вперёд 1
направо
вперёд 1
налево
вперёд 1
направо
вперёд 1
налево
вперёд 1
направо
```

Перепиши её как можно короче, используя один цикл `повтори`. Робот должен сделать то же самое. Какая длина у короткой программы и где окажется робот? Длина программы – число строк-команд; `повтори N`: считается одной командой, команды тела – каждая по одной.

Дано: 14 команд; старт (1; 1), смотрит вверх; правило счёта длины.

Найти: длину короткой программы; конечную клетку.

Решение

Шаг 1. Ищем повторяющийся кусок. Подчеркнём четвёрки команд: `вперёд 1, направо, вперёд 1, налево` – одна, вторая, третья. После них остались две команды: `вперёд 1, направо`. Четвёртой полной четвёрки нет – не хватает двух последних команд.

Шаг 2. Записываем коротко.

```
повтори 3:
  вперёд 1
  направо
  вперёд 1
  налево
вперёд 1
направо
```

Хвостик из двух команд стоит после цикла без сдвига: он выполняется один раз.

Шаг 3. Считаем длину. `повтори 3`: – 1 команда, тело – 4, хвост – 2. Всего $1 + 4 + 2 = 7$. Вдвое короче исходной.

Шаг 4. Где окажется робот. Одна четвёрка – это ступенька: вверх на клетку, вправо на клетку, и робот снова смотрит вверх. Три ступеньки из (1; 1): (2; 2), (3; 3), (4; 4). Хвост: `вперёд 1` – вверх в (4; 5), `направо` – теперь смотрит вправо. Поле 6×6 , клетка (4; 5) на нём есть.

Проверим счёт выполненных команд. В исходной программе 14 команд, в новой робот выполняет $3 \cdot 4 + 2 = 14$ – столько же. При сокращении меняется запись, а работа робота остаётся прежней.

❗ **Типичная ошибка.** Написать `повтори 4:` – «ну там же почти четыре раза». Четвёртый круг добавит лишние `вперёд 1, налево`: робот уедет в (5; 5) и будет смотреть вверх. Второй промах – посчитать `повтори` за 0 команд и ответить 6, хотя в условии сказано, как считать.

Ответ: длина 7 команд; робот в клетке (4; 5), смотрит вправо.

Своя задача по программе сферы, модуль 1

Реши сам

4.3. Программа: `вперёд 2, направо, вперёд 2, направо, вперёд 2, направо, вперёд 2, направо`. Перепиши её с одним циклом как можно короче (длину считаем по правилу задачи 4.2). Какой длины получилась программа? Где окажется робот, если он начал в (1; 1) и смотрел вверх?

4.4. Робот стоит в (1; 1) поля 6 × 6 и смотрит вверх. Программа:

```
повтори 5:
  вперёд 1
  направо
  вперёд 1
  налево
```

Сколько команд движения и поворота выполнит робот? Где он окажется и куда будет смотреть?

Задача 4.3 Какую команду вставить

цикл, вставка пропущенной команды, перебор вариантов

Поле 8 × 5 без стен. Робот стоит в клетке (1; 1) и смотрит вправо. В клетке (7; 4) лежит флажок. В программе одна команда стёрлась:

```
повтори 3:
  вперёд 2
  налево
  вперёд 1
  ???
```

Какую команду нужно вставить вместо `???`, чтобы робот закончил программу на флажке? Варианты: 1) `направо`; 2) `налево`; 3) `вперёд 1`; 4) `вперёд 2`. В ответ запиши номер варианта.

Дано: поле 8 × 5; старт (1; 1), смотрит вправо; флажок (7; 4); 4 варианта.

Найти: номер варианта.

Решение

Шаг 1. Прикидываем по цели. От (1; 1) до (7; 4) нужно 6 клеток вправо и 3 вверх. Кругов цикла три. Если бы каждый круг сдвигал робота одинаково, это было бы 2 вправо и 1 вверх за круг. В теле уже есть **вперёд 2** и **вперёд 1** – числа подходят. Значит, в начале каждого круга робот должен снова смотреть вправо.

Шаг 2. Проследим первый круг. Робот смотрит вправо: **вперёд 2** – в (3; 1); **налево** – смотрит вверх; **вперёд 1** – в (3; 2). Чтобы снова смотреть вправо, из «вверх» нужно повернуть **направо**. Это вариант 1.

Шаг 3. Проверим вариант 1 до конца. Каждый круг: +2 столбца, +1 строка, робот снова смотрит вправо. После первого круга он в (3; 2), после второго – в (5; 3), после третьего – в (7; 4), на флажке.

Шаг 4. Проверим остальные варианты. Когда вариантов четыре, лучше пройти все: вдруг подходят два.

- Вариант 2, **налево** : после первого круга робот смотрит влево. Во втором круге он возвращается влево и вниз, в клетку старта, а третий круг повторяет первый. Робот закончит в (3; 2), до флажка не доедет.
- Вариант 3, **вперёд 1** : робот продолжает ехать вверх, во втором круге доезжает до верхнего края и поворачивает влево, а в третьем врежется в левый край поля.
- Вариант 4, **вперёд 2** : во втором круге робот упрётся в верхний край поля и врежется.

Подходит только вариант 1.

 **Типичная ошибка.** Выбрать **налево** : «флажок же выше, надо поворачивать туда». Но поворот вверх в программе уже есть: это строка **налево** в середине тела. Пропущенная команда нужна для другого: вернуть робота к началу следующего круга, лицом вправо.

Ответ: 1 (направо).

Своя задача по программе сферы, модуль 1

Реши сам

4.5. Поле 6 × 6 без стен. Робот стоит в левом нижнем углу, в клетке (1; 1), и смотрит вверх. Программа:

```
повтори 4:
  вперёд 3
  ???
```

Робот должен объехать квадрат и вернуться в (1; 1). Какую команду вставить: **направо** или **налево** ? Подумай, почему на бумаге подходят обе, а на этом поле – только одна.

4.6. Поле 8×6 без стен. Робот стоит в клетке (3; 2) и смотрит вверх. Программа:

```
пока впереди свободно:
    вперёд 1
???
пока впереди свободно:
    вперёд 1
```

Какую команду вставить вместо ???, чтобы робот оказался в правом верхнем углу поля: 1) направо ; 2) налево ; 3) вперёд 1 ? Сколько раз при этом выполнится команда вперёд 1 ?

Задача 4.4 Гребёнка из подпрограмм

подпрограмма, цикл с вызовом, счёт выполненных команд

Робот рисует на полу маркером «гребёнку». Он стоит в клетке (1; 1) поля 8×3 и смотрит вверх. Программа:

```
подпрограмма зубец:
    вперёд 2
    направо
    вперёд 1
    направо
    вперёд 2
    налево
    вперёд 1
    налево

повтори 3:
    зубец
```

Сколько команд движения и поворота выполнит робот? В какой клетке он окажется и куда будет смотреть?

Дано: подпрограмма из 8 команд; вызов в цикле 3 раза; старт (1; 1), смотрит вверх.

Найти: число выполненных команд; конечную клетку и направление.

Решение

Шаг 1. Что делает один зубец. Трассируем подпрограмму один раз, от (1; 1) лицом вверх:

Таблица 4.2

Команда	Клетка	Смотрит
вперёд 2	(1; 3)	вверх
направо	(1; 3)	вправо
вперёд 1	(2; 3)	вправо
направо	(2; 3)	вниз
вперёд 2	(2; 1)	вниз

Команда	Клетка	Смотрит
налево	(2; 1)	вправо
вперёд 1	(3; 1)	вправо
налево	(3; 1)	вверх

За один зубец робот сдвинулся на 2 столбца вправо, вернулся в нижнюю строку и снова смотрит вверх. Это удобно: каждый следующий зубец начинается так же, как первый, только на 2 клетки правее.

Шаг 2. Три зубца. Столбец растёт на 2 за каждый вызов: $1 + 3 \cdot 2 = 7$. Строка в конце зубца снова первая. Робот в (7; 1) и смотрит вверх.

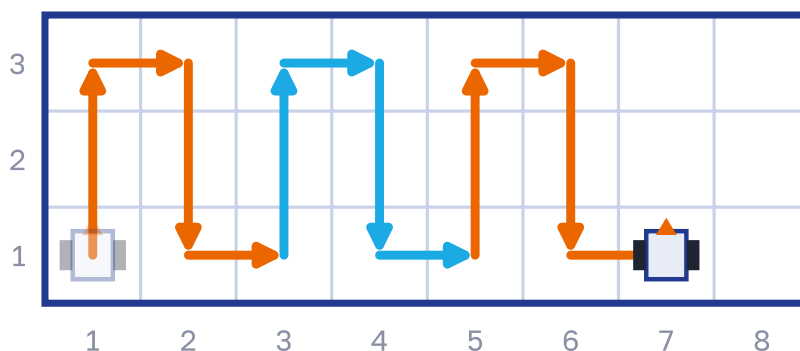


Рисунок 4.9 – Гребёнка из трёх зубцов: каждый вызов подпрограммы сдвигает робота на 2 клетки вправо

Шаг 3. Считаем команды. В подпрограмме 8 команд, вызывается она 3 раза: $8 \cdot 3 = 24$. Сами строки `повтори 3:` и `зубец` мы не считаем – по ним робот не двигается, они только говорят, что и сколько раз делать.

Проверка по полю. Поле 8×3 : строки от 1 до 3, зубец поднимается до третьей и в край не упирается. Последний зубец заканчивается в столбце 7, восьмой остаётся свободным.

⚠ Типичная ошибка. Посчитать строчки в записи: 8 строк подпрограммы + 2 строки основной программы = 10. Это длина программы, а спрашивают, сколько робот выполнил. Другая ошибка – решить, что каждый вызов начинается из клетки (1; 1). Подпрограмма продолжает с того места, где робот сейчас стоит.

Ответ: 24 команды; клетка (7; 1), смотрит вверх.

Своя задача по программе сферы, модуль 1

Реши сам

4.7. Подпрограмма шаг : `вперёд 1` , `направо` , `вперёд 1` , `налево` . Робот стоит в (1; 1) и смотрит вверх, поле 8×8 . Основная программа: `шаг` , `шаг` , `вперёд 3` , `шаг` . Где окажется робот? Сколько команд движения и поворота он выполнит?

4.8.* Поле 8×8 , робот в (1; 1) смотрит вверх. Программа:

```
подпрограмма уголок:  
  вперёд 1  
  направо  
  
подпрограмма квадрат:  
  повтори 4:  
    уголок  
  
повтори 3:  
  квадрат  
  вперёд 2
```

Сколько команд движения и поворота выполнит робот? В какой клетке он окажется?

Задача 4.5 Выход из лабиринта

лабиринт, правило правой руки, ветвление в цикле, трассировка

Робот стоит в клетке (1; 1) лабиринта 5×5 и смотрит вверх. Вход за ним закрылся. Выход – в верхней стене над клеткой (5; 5). Робот идёт по правилу правой руки (программа – в разделе 4.5). Сколько раз он выполнит команду **вперёд 1**, пока не выйдет из лабиринта? Шаг через выход наружу тоже считается.

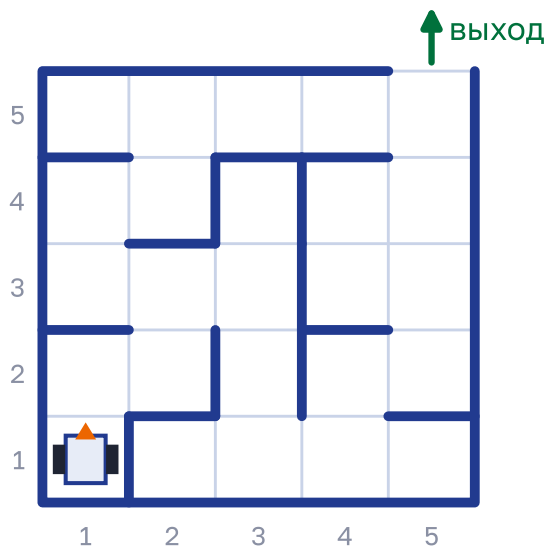


Рисунок 4.10 – Лабиринт к задаче 4.5. Толстые линии – стены

Дано: лабиринт на рисунке; старт (1; 1), смотрит вверх; правило правой руки.

Найти: число команд **вперёд 1**.

Решение

Шаг 1. Порядок проверок. В каждой клетке спрашиваем: справа свободно? Если да – поворот направо и шаг. Если нет: впереди свободно? – шаг прямо. Если нет: слева? – поворот налево и

шаг. Если и слева стена – разворот и шаг. Шаг в каждой клетке один, поэтому число команд вперёд 1 – это число строк в таблице.

Шаг 2. Трассировка. «Справа / впереди / слева» – что видит робот: «с» – свободно, «ст» – стена.

Таблица 4.3

Шаг	Робот стоит, смотрит	Справа / впереди / слева	Что делает	Куда пришёл
1	(1; 1), вверх	ст / с / ст	прямо	(1; 2)
2	(1; 2), вверх	с / ст / ст	направо	(2; 2)
3	(2; 2), вправо	ст / ст / с	налево	(2; 3)
4	(2; 3), вверх	с / ст / с	направо	(3; 3)
5	(3; 3), вправо	с / ст / с	направо	(3; 2)
6	(3; 2), вниз	ст / с / ст	прямо	(3; 1)
7	(3; 1), вниз	с / ст / с	направо	(2; 1)
8	(2; 1), влево	ст / ст / ст	разворот	(3; 1)
9	(3; 1), вправо	ст / с / с	прямо	(4; 1)
10	(4; 1), вправо	ст / с / с	прямо	(5; 1)
11	(5; 1), вправо	ст / ст / ст	разворот	(4; 1)
12	(4; 1), влево	с / с / ст	направо	(4; 2)
13	(4; 2), вверх	с / ст / ст	направо	(5; 2)
14	(5; 2), вправо	ст / ст / с	налево	(5; 3)
15	(5; 3), вверх	ст / с / с	прямо	(5; 4)
16	(5; 4), вверх	ст / с / с	прямо	(5; 5)
17	(5; 5), вверх	ст / с / с	прямо	наружу

Посмотри на шаги 7 и 10. Из клетки (3; 1) справа от робота открыт проход в тупик (2; 1) – и робот туда послушно сворачивает, хотя выход совсем в другой стороне. На шаге 10 рука заводит его во второй тупик, (5; 1). В тупиках (шаги 8 и 11) робот разворачивается и выходит обратно.

Шаг 3. Любопытная деталь – шаг 12. Робот стоит в (4; 1), смотрит влево, и впереди свободно. Но справа – проход вверх, а правое по правилу проверяется **первым**. Поэтому он поворачивает направо и уходит вверх, в (4; 2).

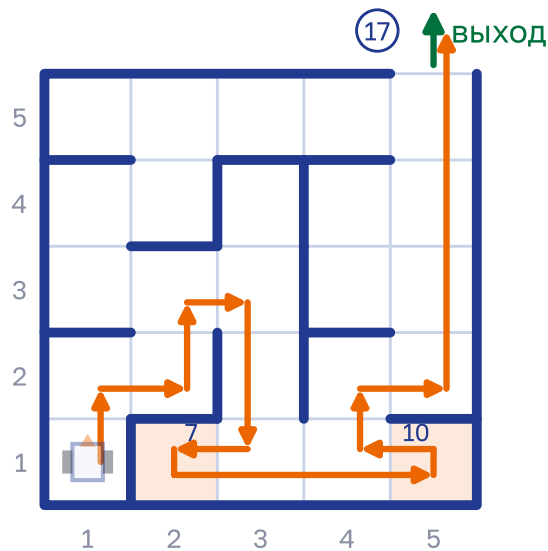


Рисунок 4.11 – Путь по правой руке. Туда и обратно робот едет по разным сторонам коридора, поэтому тупики видны петлями

Прикинем. Кратчайший путь от (1; 1) до выхода – 11 шагов. У нас 17: лишние шесть ушли на два тупика, в каждом – заезд и выезд. Так платит за дорогу любой, кто идёт без карты. Зато стены здесь соединены с внешней, и к выходу рука приведёт наверняка.

❗ **Типичная ошибка.** Посчитать кратчайший путь (11 шагов) – ведь «робот же умный». Робот без карты кратчайшего пути не знает, он честно следует правилу. Вторая ошибка – поменять порядок проверок и сначала смотреть вперёд: на шаге 12 робот поедет прямо, в (3; 1), и путь станет совсем другим.

Ответ: 17.

Своя задача по программе сферы, модуль 8

Реши сам

4.9. Тот же лабиринт (рисунок к задаче 4.5), тот же старт: (1; 1), лицом вверх. Теперь робот идёт по правилу **левой** руки: сначала слева, потом впереди, потом справа, в тупике разворот. Сколько раз он выполнит **вперёд 1**, пока не выйдет? Какое правило в этом лабиринте короче?

4.10. Робот прошёл другой лабиринт от входа до выхода без заездов в тупики и записал свои решения на развилках по порядку: «направо, налево, прямо, налево, налево». Теперь ему нужно вернуться от выхода ко входу той же дорогой. Какой список решений он должен выполнить на развилках обратного пути? Запиши буквами: П – направо, Л – налево, С – прямо (например, «ЛПС»).

Задача 4.6 Найди ошибку

поиск ошибки, трассировка, цикл

На поле 8 × 6 клетки (2; 2) и (3; 2) заняты клумбой. Робот стоит в клетке (1; 1) и смотрит вверх. Он должен объехать клумбу по соседним клеткам по часовой стрелке – (1; 1) → (1; 3) → (4; 3) → (4; 1) → (1; 1) – и вернуться в (1; 1) лицом вверх. Программист написал:

```
1  повтори 2:
2      вперёд 2
3      направо
4      вперёд 3
5      налево
```

Робот уехал куда-то не туда. В какой строке ошибка и что там должно стоять? В ответ запиши номер строки.

Дано: программа из 5 строк; задуманный маршрут; старт (1; 1), смотрит вверх.

Найти: номер строки с ошибкой и исправление.

Решение

Шаг 1. Что должно быть. Задуманный путь – прямоугольник из четырёх отрезков: вверх 2, вправо 3, вниз 2, влево 3. Он складывается из двух одинаковых половинок: «вверх 2, вправо 3» и «вниз 2, влево 3». Поэтому цикл **повтори 2** здесь к месту. После каждого отрезка робот поворачивает по часовой стрелке – то есть направо.

Шаг 2. Трассируем, что получилось.

Таблица 4.4

Строка	Команда	Клетка	Смотрит	Должно быть
2	вперёд 2	(1; 3)	вверх	(1; 3) – да
3	направо	(1; 3)	вправо	вправо – да
4	вперёд 3	(4; 3)	вправо	(4; 3) – да
5	налево	(4; 3)	вверх	смотреть вниз – нет

Первое расхождение – строка 5. Робот должен повернуть вниз, к клетке (4; 1), а повернул вверх.

Шаг 3. Что будет дальше с ошибкой. Второй круг: **вперёд 2** – вверх в (4; 5), **направо** – вправо, **вперёд 3** – в (7; 5), **налево** – вверх. Робот закончит в (7; 5), далеко от клумбы. Красная линия на рисунке.

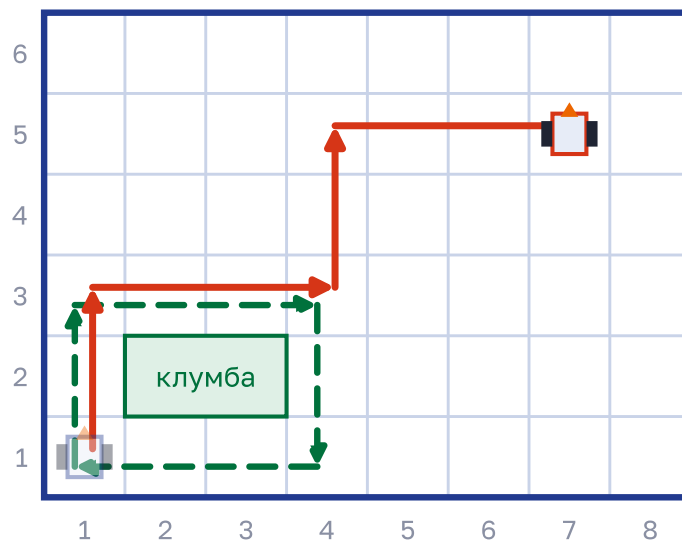


Рисунок 4.12 – Зелёный пунктир – задуманный объезд клумбы, красная линия – путь по программе с ошибкой в строке 5

Шаг 4. Исправляем и проверяем. В строке 5 ставим `направо`. Первый круг: (1; 3), вправо, (4; 3), вниз. Второй: `вперёд 2` – (4; 1), `направо` – смотрит влево, `вперёд 3` – (1; 1), `направо` – смотрит вверх. Робот на месте и смотрит как в начале.

А нельзя ли исправить иначе, например в строке 3 поставить `налево` и объехать клумбу против часовой? Нельзя: робот стоит в левом нижнем углу поля, и первый же `вперёд 3` влево врежет его в край. Значит, ошибка только в строке 5.

⚠ Типичная ошибка. Начать «исправлять» числа: заменить `вперёд 3` на `вперёд 4` или `вперёд 2` на `вперёд 3`, потому что клумба «вроде бы шире». Числа в программе верные – это видно по таблице: до строки 5 робот идёт по задуманному пути. Сначала найди первую строку, где разошлось, и только потом думай, что менять.

Ответ: строка 5, вместо `налево` должно быть `направо`.

Своя задача по программе сферы, модуль 1

Реши сам

4.11. Коридор – одна колонка из 8 клеток, снизу вверх. Робот стоит в нижней клетке и смотрит вверх. Программа `повтори 8`: `вперёд 1` должна довести его до верхней клетки, но робот врежется в стену. Какое число нужно поставить вместо 8?

4.12. Робот стоит в (1; 1) поля 6 × 6 и смотрит вверх. Программа должна поднять его по лестнице в клетку (5; 5), и там он должен смотреть вверх:

```

1 подпрограмма ступенька:
2   вперёд 1
3   направо
4   вперёд 1
5   направо
6 повтори 4:
7   ступенька

```

Вместо этого робот кружит по маленькому квадратику возле старта и никуда не поднимается. Где он окажется после программы? В какой строке ошибка?

Задача 4.7* Таблички на клетках

трассировка, заикливание, таблица переходов

На складе вдоль стены – дорожка из 9 клеток с номерами от 1 до 9. На каждой клетке, кроме девятой, стоит табличка с номером другой клетки. Робот-погрузчик начинает на клетке 1. Он читает табличку, переезжает на клетку с этим номером, там снова читает табличку, и так далее. Клетка 9 – финиш: доехав до неё, робот останавливается.

- Таблички такие, как на рисунке. Сколько переездов сделает робот до финиша?
- Таблички переставили: 1 → 3, 2 → 6, 3 → 5, 4 → 9, 5 → 8, 6 → 4, 7 → 2, 8 → 3. Сколько переездов теперь? Если робот никогда не доедет до финиша, в ответ запиши -1.
- Робота из пункта б) никто не выключил, и он продолжает ездить по табличкам. На какой клетке он будет после 20 переездов?



Рисунок 4.13 – Дорожка к пункту а): на каждой клетке – номер клетки, куда ехать дальше

Дано: 9 клеток; таблички для а) – на рисунке, для б) и в) – в условии; старт – клетка 1, финиш – клетка 9.

Найти: а), б) число переездов или -1; в) клетку после 20 переездов.

Решение

Шаг 1. Пункт а) – трассировка. Табличка на клетке – это команда «переехать на клетку №...». Ведём робота:

Таблица 4.5

Переезд	Откуда	Табличка	Куда
1	1	→ 4	4
2	4	→ 7	7
3	7	→ 3	3
4	3	→ 6	6
5	6	→ 5	5
6	5	→ 2	2
7	2	→ 9	9 – финиш

Робот на финише после 7 переездов. На клетке 8 он так и не побывал: на неё не указывает ни одна табличка.

Шаг 2. Пункт б) – та же трассировка. $1 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 3$. Стоп: на клетке 3 робот уже был – после первого переезда.

Шаг 3. Почему дальше можно не считать. Это заикливание из раздела 4.7. С клетки 3 робот в прошлый раз поехал на 5, потом на 8, потом снова на 3 – и сейчас поедет так же. Круг $3 \rightarrow 5 \rightarrow 8 \rightarrow 3$ будет крутиться вечно, финиша робот не увидит.

А клетка 4, с которой есть переезд на финиш? До неё можно добраться только с клетки 6, до 6 – только с 2, до 2 – только с 7, а на 7 не указывает ни одна табличка. Робот с клетки 1 в эту цепочку не попадёт никогда.

Шаг 4. Сколько строк таблицы может понадобиться. Клеток с табличками – восемь. Пусть робот сделал 8 переездов и не доехал до финиша. Считая старт, это 9 остановок, а разных клеток с табличками всего 8 – значит, одна повторилась, и робот уже на круге. Больше 8 строк в такой таблице не бывает никогда.

Шаг 5. Пункт в) – где робот после 20 переездов. Считаем, как в разделе 4.7. Подъезд: $1 \rightarrow 3$ – один переезд, дальше робот на круге. Длина круга $3 \rightarrow 5 \rightarrow 8 \rightarrow 3$ – три переезда. Отнимаем подъезд: $20 - 1 = 19$. Делим на длину круга с остатком: $19 = 3 \cdot 6 + 1$. Шесть полных кругов возвращают робота на клетку 3, остаток 1 – ещё один переезд, на клетку 5.

Сверим на малом числе. После 4 переездов счёт даёт $4 - 1 = 3 = 3 \cdot 1 + 0$, остаток 0 – робот на клетке 3. По трассировке из шага 2 после четвёртого переезда он как раз вернулся на 3.

Для любопытных: как это считает компьютер. В задаче старшей олимпиады, откуда взята идея, клеток было до ста тысяч, и трассировку вела программа. Вот она для пункта б):

```

tabl = {1: 3, 2: 6, 3: 5, 4: 9, 5: 8, 6: 4, 7: 2, 8: 3} # пункт б)
pos = 1 # где робот
moves = 0 # сколько переездов сделано
visited = set() # на каких клетках уже был
while pos != 9 and pos not in visited:
    visited.add(pos)
    pos = tabl[pos]
    moves = moves + 1
print(moves if pos == 9 else -1) # -1

```

`tabl` – словарь: по номеру клетки он выдаёт номер с таблички. `visited` – множество клеток, где робот уже побывал. Цикл `while` («пока») идёт, пока робот не на финише и не вернулся на старую клетку. `moves if pos == 9 else -1` значит «напечатай число переездов, если робот на финише, а иначе -1». Подставь таблички из пункта а) – и программа напечатает 7.

 **Типичная ошибка.** В пункте б) увидеть табличку «4 → 9» и решить, что робот через клетку 4 доедет до финиша, – и начать подгонять ответ. Смотреть надо на путь **этого** робота с клетки 1, а не на все таблички подряд. Вторая ошибка – долго считать переезды по кругу 3 → 5 → 8, надеясь, что «когда-нибудь выйдет». Повтор клетки – это уже ответ. В пункте в) легко забыть про подъезд и разделить на 3 сразу все 20 переездов: $20 = 3 \cdot 6 + 2$, два шага по кругу от клетки 3 – и выходит неверная клетка 8.

Ответ: а) 7; б) -1; в) клетка 5.

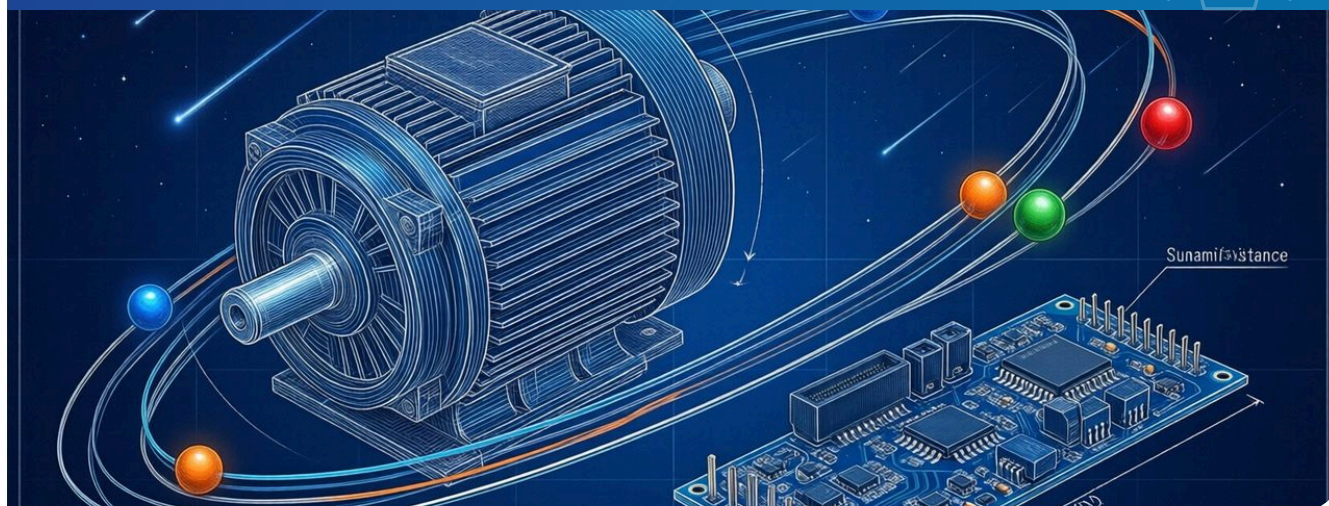
По мотивам задачи заключительного этапа НТО 2024/25, профиль «Интеллектуальные робототехнические системы», предметный тур по информатике, «Робот-манипулятор»

Реши сам

4.13. Дорожка из 7 клеток, финиш – клетка 7, робот начинает на клетке 1. Таблички: 1 → 5, 2 → 7, 3 → 2, 4 → 6, 5 → 3, 6 → 1. а) Сколько переездов сделает робот до финиша? б) Табличку на клетке 5 заменили на «→ 4». Сколько переездов теперь (если не доедет никогда – ответ -1)?

4.14.* Дорожка из 8 клеток, финиша нет – робот ездит, пока его не выключат. Старт на клетке 1. Таблички: 1 → 6, 2 → 8, 3 → 1, 4 → 2, 5 → 7, 6 → 2, 7 → 3, 8 → 4. На какой клетке окажется робот после 100 переездов?

Моторы, датчики и электричество



Всё, что робот делает, он делает электричеством. Батарея толкает ток, драйвер пропускает его к моторам, датчики превращают свет и звук в напряжение, а контроллер – напряжение в числа. Физики здесь больше, чем в любой другой главе книги, но начинаем с нуля: что такое ток и вольты, узнаешь из первого же раздела. На отборе такие задачи короткие: посчитать ток или подобрать резистор, найти значение ШИМ для нужной скорости, перевести число с датчика в вольты, время эха – в сантиметры, заряд аккумулятора – в минуты работы. Баллы здесь теряют в основном на единицах: миллиамперы вместо ампер, микросекунды вместо миллисекунд.

5.1. Цепь: ток, напряжение, сопротивление

НА ПАЛЬЦАХ



Рисунок 5.1 – Ток бежит по проводам, как вода по трубам, замкнутым в кольцо

Батарейка – насос, который создаёт напор. Мотор – колесо, которое крутит поток. Узкая труба мешает воде – это сопротивление.

Электрический ток – движение крошечных заряженных частиц, электронов, по проводам. Чтобы ток пошёл, нужны три вещи: **источник** (батарейка, аккумулятор, USB-порт компьютера), **нагрузка** (светодиод, мотор, пищалка) и провода, которые соединяют их **в замкнутый круг**. Такой круг называют **электрической цепью**. Разорви его в одном месте – и ток пропадёт сразу во всей цепи. Так работает выключатель на стене: он разрывает провод к люстре.

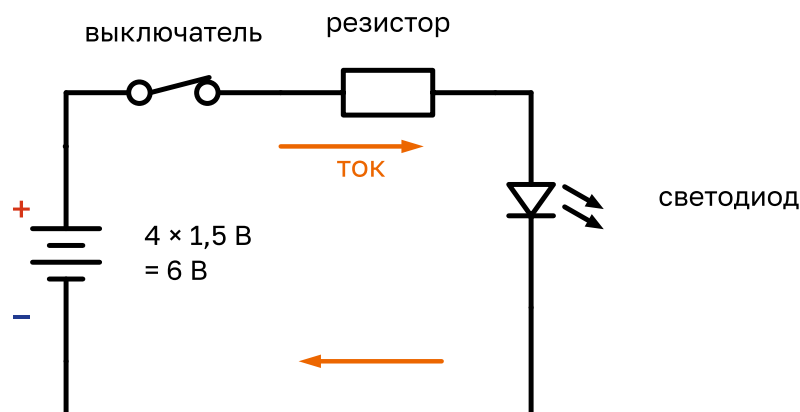


Рисунок 5.2 – Простая цепь: батарея, выключатель, резистор и светодиод. Стрелки – направление тока, от «+» батареи через светодиод к «-»

У любого источника два вывода: **плюс (+)** и **минус (-)**. Условились считать, что снаружи источника ток идёт от плюса к минусу. На плате Arduino плюс подписан **5V**, минус – **GND** (от английского ground – «земля»). Все выводы GND на схеме соединены между собой, это общий минус.

Теперь три величины, без которых не обходится ни одна задача главы. Держи в голове картинку с водопроводом.

- **Напряжение U** – напор, с которым насос гонит воду. Чем больше напряжение, тем сильнее оно толкает ток. Измеряют в **вольтах** (В). Пальчиковая батарейка даёт 1,5 В, плата Arduino – 5 В, батарейка «Крона» – 9 В. В розетке 220 В – это смертельно опасно, с розеткой не экспериментируем никогда.
- **Сила тока I** , или просто ток, – сколько воды протекает по трубе за секунду. Измеряют в **амперах** (А). Токи в роботе маленькие, поэтому чаще пишут **миллиамперы**: 1 А = 1000 мА. Через светодиод пускают до 20 мА, моторчик робота берёт сотни миллиампер.
- **Сопротивление R** – насколько узкая труба. Измеряют в **омах** (Ом), 1 кОм (килоом) = 1000 Ом. Деталь, вся работа которой – мешать току, называется **резистор**.

Колесо в водопроводе воду не пьёт: сколько её втекло, столько и вытекло дальше по кольцу. С током так же. В светодиоде он **не расходуется** – сколько миллиампер вошло в одну ножку, столько вышло из другой. Тратится энергия батарейки: она уходит в свет, тепло и вращение мотора.

Батарейки в отсеке робота стоят **последовательно**: плюс одной упирается в минус следующей. Напряжения при этом складываются. Четыре батарейки по 1,5 В дают $4 \cdot 1,5 = 6$ В – самое частое питание учебного робота. Поставь одну батарейку задом наперёд, и она «вычтется»: вместо 6 В останется $1,5 + 1,5 + 1,5 - 1,5 = 3$ В.

Запомни. Нет замкнутого круга – нет тока. Вольты толкают, амперы текут, омы мешают. 1 А = 1000 мА, 1 кОм = 1000 Ом. На Arduino 5V – плюс, GND – минус.

5.2. Закон Ома и резистор для светодиода

НА ПАЛЬЦАХ

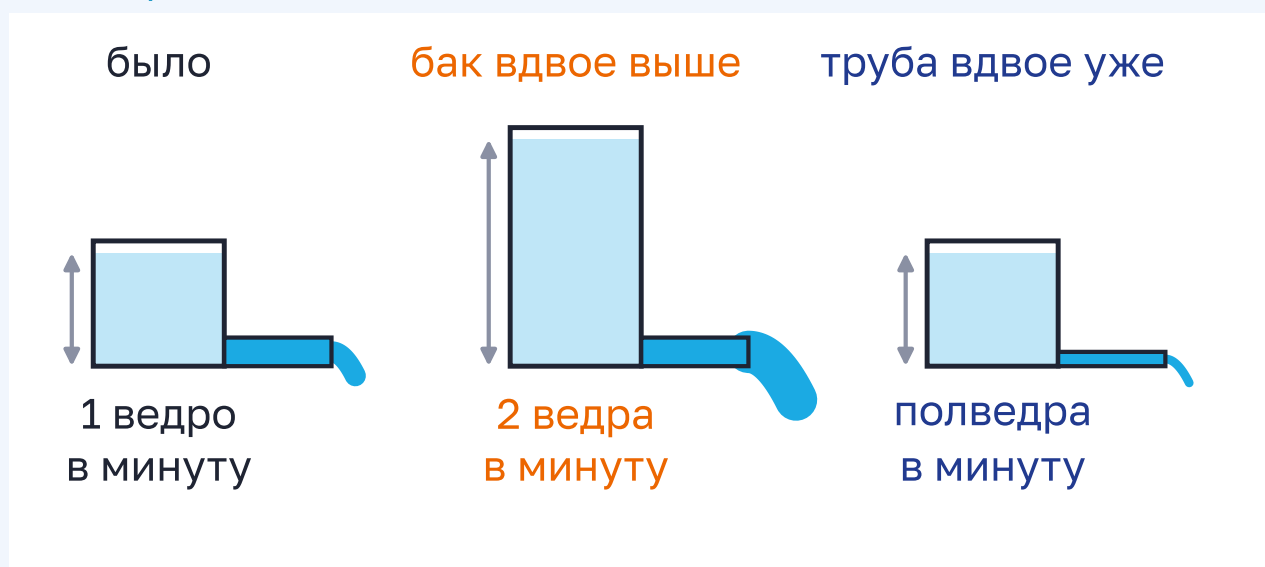


Рисунок 5.3 – Что меняет поток воды

Подняли бак вдвое выше – воды течёт вдвое больше. Сузили трубу вдвое – вдвое меньше. С током то же самое: он растёт вместе с напряжением и падает, когда растёт сопротивление.

Эту связь в 1827 году описал немецкий учитель физики Георг Ом, и с тех пор её называют **законом Ома**. Словами: **сила тока равна напряжению, делённому на сопротивление**. Буквами:

$$I = \frac{U}{R},$$

где I – ток в амперах, U – напряжение в вольтах, R – сопротивление в омах. Из этой формулы получаются ещё две: напряжение равно току, умноженному на сопротивление, а сопротивление – напряжению, делённому на ток:

$$U = I \cdot R, \quad R = \frac{U}{I}.$$

Пример. Резистор 250 Ом подключили к 5 В. Ток $I = 5 : 250 = 0,02$ А. В миллиамперах это $0,02 \cdot 1000 = 20$ мА.

Единицы в этой формуле строгие: вольты, амперы, омы. Дан ток 15 мА – сначала переведи: $15 : 1000 = 0,015$ А. Если подставить в формулу число 15, сопротивление выйдет в 1000 раз меньше настоящего. Есть и короткий путь для тех, кто любит килоомы: **вольты, делённые на килоомы, дают сразу миллиамперы**. 5 В на резисторе 2 кОм – это $5 : 2 = 2,5$ мА.

Если на пути тока стоят два резистора подряд (последовательно), их сопротивления складываются: 100 Ом и 220 Ом вместе мешают току, как один резистор на 320 Ом. Это как труба с двумя узкими местами.

Светодиод и резистор к нему. Светодиод – маленькая лампочка, которая светится, когда через неё идёт ток. Ток он пропускает только в одну сторону: длинная ножка (**анод**) – к плюсу, короткая (**катод**) – к минусу. Без резистора светодиод почти не мешает току, ток получается огромным, и светодиод сгорает – в Tinkercad это видно прямо на экране. Поэтому последовательно с ним всегда ставят резистор.

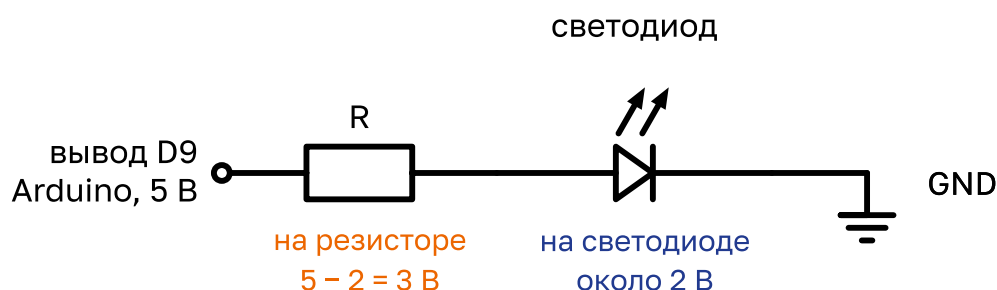


Рисунок 5.4 – Светодиод с резистором на выводе Arduino. 5 В делятся между ними: около 2 В «оседает» на светодиоде, остальное – на резисторе

На горящем светодиоде «оседает» почти постоянное напряжение: у красного, жёлтого и зелёного – около 2 В, у синего и белого – около 3 В. Остальное напряжение приходится на резистор. Поэтому сопротивление считают **по оставшемуся напряжению**:

$$R = \frac{U_{\text{пит}} - U_{\text{св}}}{I},$$

где $U_{\text{пит}}$ – напряжение питания, $U_{\text{св}}$ – напряжение на светодиоде, I – ток, который мы хотим пустить. Пример: красный светодиод от вывода Arduino, ток 20 мА. $R = (5 - 2) : 0,02 = 150$ Ом.

Резисторы не продают любого номинала: есть стандартный ряд, в учебных наборах чаще всего 100, 150, 220, 330, 470 Ом, 1 кОм, 10 кОм. Если нужного нет, берут **ближайший боль-**

ший. Ток станет чуть меньше, светодиод – чуть тусклее, зато он не перегреется. Меньший резистор пустит ток больше допустимого.

Запомни. $I = U/R$, и ток в эту формулу идёт в амперах. Резистор для светодиода считают по напряжению, которое осталось: $(U_{\text{пит}} - U_{\text{св}})/I$. Из ряда берут ближайший **большой** номинал.

Совет. Собери цепь со светодиодом в Tinkercad (tinkercad.com) или в запасном Wokwi (wokwi.com) и поставь резисторы разных номиналов. Сравни, как меняется яркость. Числа из задачи сразу станут понятнее, если один раз увидеть, что 100 Ом – ярко, а 10 кОм – еле тлеет.

5.3. Мотор и драйвер: почему мотор нельзя прямо к плате

НА ПАЛЬЦАХ

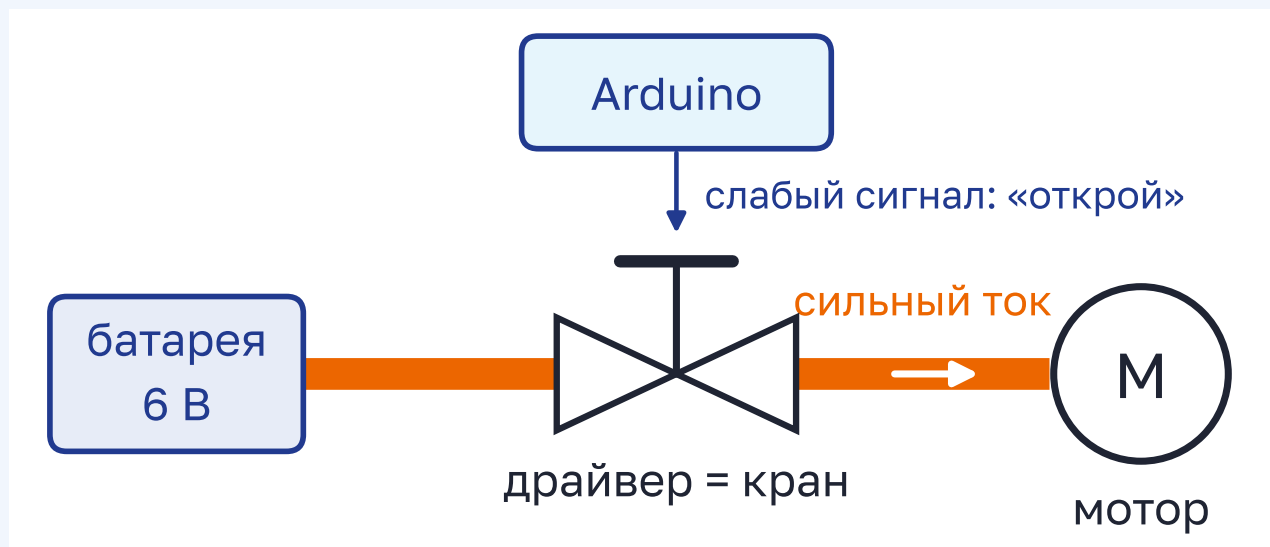


Рисунок 5.5 – Драйвер моторов работает как кран

Контроллер не носит ток к мотору сам. Он только поворачивает ручку крана, а сильный ток бежит к мотору от батареи.

Внутри **мотора постоянного тока** (DC-мотора, ты знаешь его по главе 1) – катушка из тонкой проволоки и магниты. Пошёл ток через катушку – вал закрутился. Поменяй провода местами – вал закрутится в обратную сторону. Подними напряжение – закрутится быстрее.

Почему же мотор не подключают прямо к выводу Arduino? Вывод платы отдаёт самое большее около **40 мА** – это предел, на все выводы вместе до 200 мА. Безопасно держаться около 20 мА: этого хватает на светодиод, и только. Катушка мотора – это просто кусок проволоки, сопротивление у неё маленькое, несколько омов или десятков. По закону Ома при 5 В и 10 Ом получится $5:10 = 0,5$ А, то есть 500 мА – в 25 раз больше безопасных 20 мА (и в 12,5 раза больше предельных 40 мА). Пока вал крутится, ток немного меньше, а вот в момент трогания

с места он как раз такой. Вывод сгорит, а может, и вся плата. Вдобавок мотор при включении и выключении даёт скачки напряжения, и плата от них сбоит.

Поэтому между контроллером и мотором ставят **драйвер моторов**. В курсе сферы это микросхема **L293D**. Она управляет двумя моторами и отдаёт каждому до 600 мА. Для каждого мотора у неё три входа:

- **IN1 и IN2** – направление вращения;
- **EN** (от английского enable – «разрешить») – можно ли мотору крутиться и насколько быстро.

Сами моторы питаются от батареи через ножку 8 микросхемы (VS). Сигналы с выводов Arduino к мотору не идут – они только «открывают кран». А ещё минус батареи и GND платы обязательно соединяют: без общего минуса драйвер не поймёт сигналов контроллера.

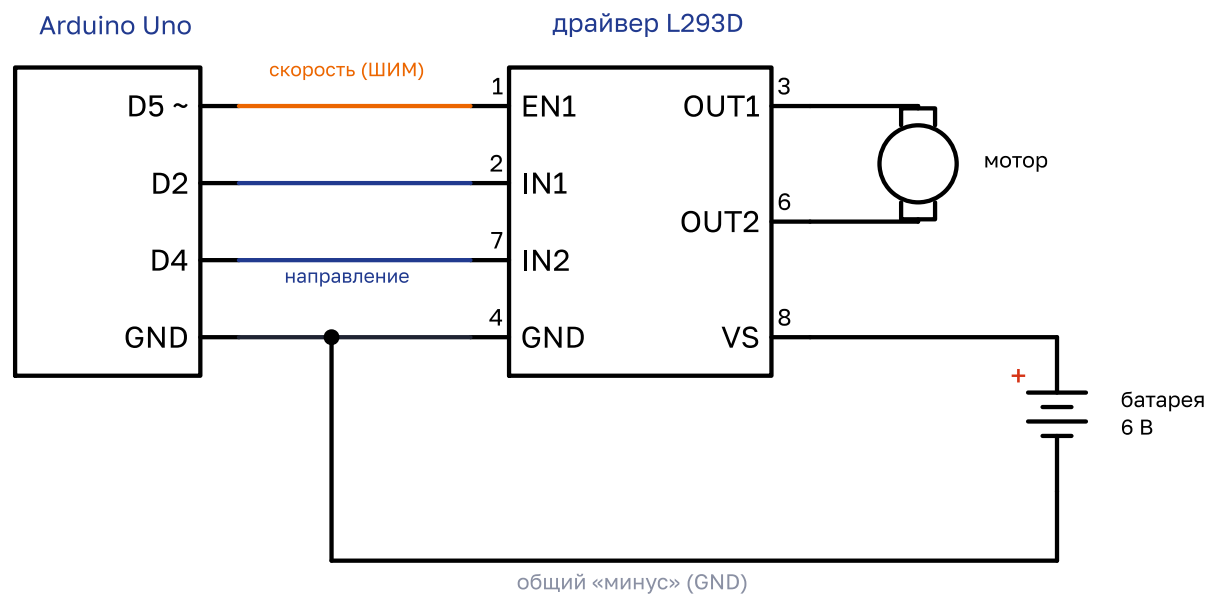


Рисунок 5.6 – Один мотор через драйвер L293D. Не показано питание самой микросхемы: ножка 16 подключается к 5V платы

Сигнал на выводе бывает двух видов: **HIGH** («высокий», 5 В) и **LOW** («низкий», 0 В). Что делает мотор при разных сигналах на IN1 и IN2, если на EN подано HIGH, – в таблице:

Таблица 5.1

IN1 и IN2	Мотор
HIGH и LOW	крутится в одну сторону
LOW и HIGH	крутится в другую сторону
LOW и LOW	стоит, катится свободно
HIGH и HIGH	стоит, тормозит

Какую из двух сторон считать «вперёд», зависит от того, как припаяны провода мотора; в задаче это всегда сказано. Если на EN подано LOW, мотор стоит при любых IN.

Вот программа, которая две секунды крутит мотор со схемы выше, потом секунду стоит, и так без конца. Это первая программа для Arduino в книге, разберём её построчно.

```

const int EN1 = 5;    // вывод 5 – скорость мотора
const int IN1 = 2;    // выводы 2 и 4 – направление
const int IN2 = 4;

void setup() {
  pinMode(EN1, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
}

void loop() {
  digitalWrite(IN1, HIGH); // едем вперёд
  digitalWrite(IN2, LOW);
  analogWrite(EN1, 200);    // скорость 200 из 255
  delay(2000);              // 2000 мс = 2 с
  digitalWrite(IN1, LOW);   // стоп
  digitalWrite(IN2, LOW);
  delay(1000);
}

```

- `const int EN1 = 5;` – даём выводу 5 имя EN1, чтобы в программе не путаться в номерах. Всё, что после `//`, – комментарий для человека, плата его не читает.
- `setup` выполняется один раз при включении. `pinMode(..., OUTPUT)` говорит: «этот вывод будет выдавать сигнал».
- `loop` повторяется по кругу, пока есть питание.
- `digitalWrite(IN1, HIGH)` подаёт на вывод 5 В, `LOW` – 0 В.
- `analogWrite(EN1, 200)` задаёт скорость – как это работает, расскажет следующий раздел.
- `delay(2000)` – подождать 2000 миллисекунд. В одной секунде 1000 мс, значит, это 2 с.

Совет. В Tinkercad есть и мотор постоянного тока, и микросхема L293D, так что схему можно собрать один в один. Запусти программу: вал на экране закрутится, и видно, с какой скоростью. Поменяй местами `HIGH` и `LOW` у `IN1` и `IN2` – мотор закрутится в другую сторону.

Запомни. Мотор – только через драйвер: вывод платы даёт до 40 мА (безопасно – около 20), мотору нужны сотни. `IN1` и `IN2` задают направление (разные сигналы – крутится, одинаковые – стоит), `EN` – можно ли ехать и как быстро. Минус батареи и `GND` платы соединяют.

5.4. ШИМ: как ехать не на полной скорости

НА ПАЛЬЦАХ



Рисунок 5.7 – ШИМ: вывод быстро-быстро щёлкает между 5 В и 0 В

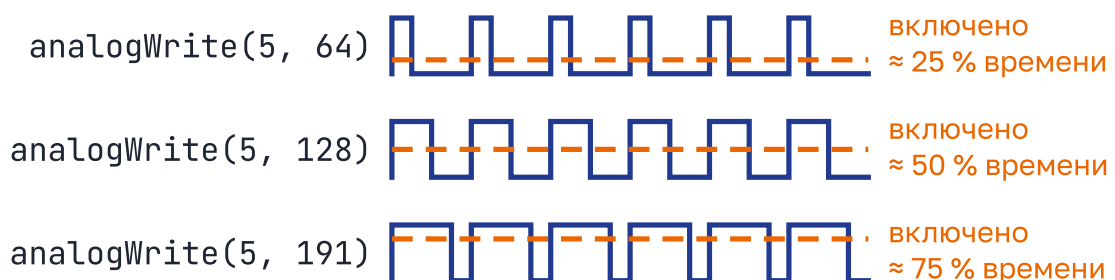
Выводу Arduino доступны только «5 В» и «0 В». Если щёлкать между ними очень быстро, мотор почувствует среднее и закрутится медленнее.

Цифровой вывод знает два состояния: HIGH и LOW. Для мотора это «полный ход» и «стоп». А как же ехать вполсилы? Похоже на велосипед: крутишь педали полоборота, полоборота отдыхаешь – и едешь медленнее, но не останавливаешься, потому что велосипед катится по инерции.

Контроллер делает так же, только намного быстрее. Вывод включается и выключается около 500 раз в секунду (на выводах 5 и 6 – около 1000 раз). Вал мотора тяжёлый и за такое короткое время остановиться не успевает, поэтому крутится с какой-то средней скоростью. Этот приём называют **ШИМ** – широтно-импульсная модуляция. «Импульс» – короткое включение, «широотно» – потому что меняют его ширину, то есть сколько времени вывод включён.

Значение ШИМ задаёт команда `analogWrite(вывод, значение)`, где значение – целое число от 0 до 255:

- 0 – вывод всё время выключен, мотор стоит;
- 255 – всё время включён, полная скорость;
- 128 – включён примерно половину времени.



оранжевый пунктир – среднее напряжение, его и «чувствует» мотор

Рисунок 5.8 – Три значения ШИМ. Чем больше значение, тем дольше вывод включён и тем выше среднее напряжение

Доля времени, когда вывод включён, равна значению, делённому на 255. Отсюда две формулы:

$$\text{доля} = \frac{\text{значение}}{255}, \quad \text{значение} = \text{доля} \cdot 255.$$

Чтобы вывод был включён 40 % времени, нужно $0,4 \cdot 255 = 102$. Значение `analogWrite` бывает только целым, поэтому дробный результат округляют до целого. Среднее напряжение на моторе – та же доля от напряжения питания моторов. От батареи 6 В при значении 128 мотор получит в среднем $6 \cdot 128 : 255 \approx 3$ В.

ШИМ умеют не все выводы Arduino Uno, а только помеченные значком ~ : 3, 5, 6, 9, 10, 11. Поэтому вход EN драйвера подключают к одному из них – на схеме в разделе 5.3 это вывод 5.

В задачах почти всегда считают, что **скорость работа пропорциональна значению ШИМ**: значение вдвое меньше – робот едет вдвое медленнее. В жизни это не совсем так. При маленьком значении мотор гудит, но не трогается – не хватает силы сдвинуть робота с места. И чем сильнее разряжена батарея, тем медленнее едет робот при том же значении. Если такое учтено в задаче, об этом скажут в условии.

Запомни. `analogWrite` : от 0 до 255, только выводы с ~ . Значение = доля · 255. Не путай с `analogRead` , у него шкала от 0 до 1023 – о нём раздел 5.6.

5.5. Батарея: на сколько хватит заряда

НА ПАЛЬЦАХ

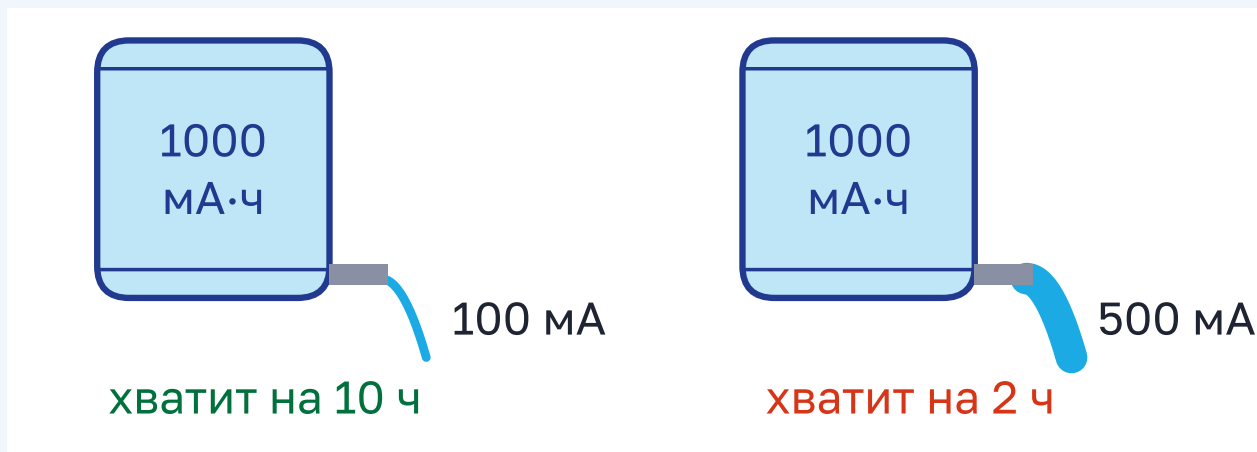


Рисунок 5.9 – Одна и та же батарея при маленьком и большом токе

Ёмкость – сколько «воды» в бочке. Чем сильнее открыт кран, тем быстрее бочка опустеет.

На аккумуляторе пишут его **ёмкость** – в **миллиампер-часах** (МА·ч). Надпись 2000 МА·ч значит: аккумулятор может отдавать ток 2000 мА в течение часа. Или 1000 мА два часа. Или 200 мА десять часов. Произведение тока на время всегда одно и то же – это запас заряда. У пауэр-банка для телефона бывает 10 000 МА·ч, у пальчикового аккумулятора – около 2000 МА·ч.

Отсюда формула: **время работы равно ёмкости, делённой на ток**:

$$t = \frac{\text{ёмкость}}{I}.$$

Если ёмкость в мА·ч, а ток в мА, время получится в часах. Минуты – умножаем на 60. Осторожно с дробями часов: 1,6 ч – это 1 ч 36 мин ($0,6 \cdot 60 = 36$), а вовсе не 1 ч 60 мин.

Ток всего робота. От одной батареи питаются сразу моторы, плата и датчики. Все они подключены к батарее рядом, как ветки к стволу, – такое соединение называют **параллельным**. Каждая ветка берёт свой ток, а батарея отдаёт их **сумму**. Два мотора по 250 мА, плата 50 мА, датчики 20 мА: $2 \cdot 250 + 50 + 20 = 570$ мА. Больше всего, конечно, берут моторы, поэтому робот, который стоит на месте, почти не тратит заряд.

Запас. Аккумулятор не разряжают до нуля: под конец напряжение проседает, робот начинает ехать медленнее и сбиваться с линии. Поэтому в задачах часто пишут, сколько ёмкости можно использовать, например 80 %. Тогда сначала находим полезную ёмкость: $0,8 \cdot 2000 = 1600$ мА·ч, и уже её делим на ток.

Несколько батареек. Последовательно (как в отсеке робота) – напряжения складываются, а ёмкость остаётся как у одной батарейки: через них всех идёт один и тот же ток, и садятся они одновременно. Четыре батарейки по 1,5 В и 2000 мА·ч дают 6 В и те же 2000 мА·ч. А если соединить батарейки параллельно, плюс к плюсу и минус к минусу, напряжение останется как у одной, зато ёмкости сложатся.

Запомни. Время работы = ёмкость : ток; мА·ч делим на мА – получаем часы. Токи всех устройств складываем. Последовательно соединённые батарейки прибавляют вольты, но не мА·ч.

5.6. analogRead: как контроллер измеряет напряжение

НА ПАЛЬЦАХ

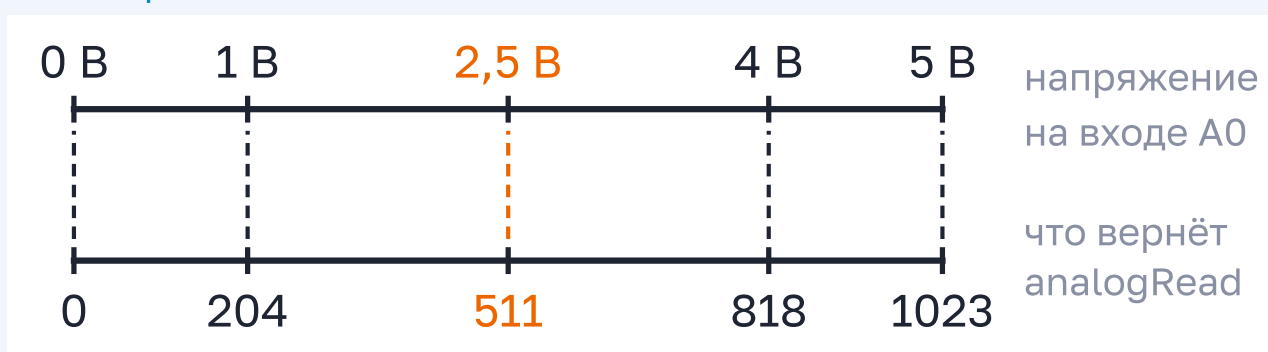


Рисунок 5.10 – Одна и та же шкала – в вольтах и в числах

Контроллер меряет напряжение линейкой с делениями от 0 до 1023. 0 В – это 0, 5 В – это 1023, одно деление – около 5 милливольт, то есть 5 тысячных вольт.

Кнопка отвечает «да» или «нет». Но многим датчикам этого мало: датчик линии, датчик освещённости, ИК-дальномер, крутилка-потенциометр выдают **напряжение от 0 до 5 В**, и оно

меняется вместе с тем, что датчик измеряет: светом, расстоянием, поворотом ручки. Такие датчики подключают к **аналоговым входам** платы – A0, A1, ..., A5.

Команда `analogRead(A0)` измеряет напряжение на входе A0 и возвращает **целое число от 0 до 1023**. Это номер деления на воображаемой линейке, где 0 В – деление 0, а 5 В – деление 1023. Почему 1023? Делений вместе с нулевым 1024, это число удобно для компьютера ($2 \cdot 2 \cdot \dots \cdot 2$, десять двоек).

Перевод туда и обратно – обычная пропорция. Какую долю от 5 В составляет напряжение, такую же долю от 1023 составляет число:

$$U = N \cdot \frac{5}{1023}, \quad N = \frac{U}{5} \cdot 1023,$$

где N – число, которое вернул `analogRead`, U – напряжение в вольтах. Число N всегда целое: контроллер **отбрасывает** дробную часть (в этой книге считаем так).

- $U = 2,5$ В: $N = 2,5 : 5 \cdot 1023 = 511,5$, дробную часть отбрасываем – 511.
- $N = 614$: $U = 614 \cdot 5 : 1023 = 3,0$ В.

В некоторых задачах делят не на 1023, а на 1024 – разница крошечная, но ответ может не совпасть в последней цифре. Считай так, как сказано в условии.

Как посмотреть показания своими глазами? Плата умеет посылать числа на компьютер, а Arduino IDE показывает их в окне «Монитор порта»:

```
void setup() {  
  Serial.begin(9600);           // включаем связь с компьютером  
}  
  
void loop() {  
  int n = analogRead(A0);       // число от 0 до 1023  
  float u = n * 5.0 / 1023;     // переводим в вольты  
  Serial.println(u);            // печатаем в «Монитор порта»  
  delay(500);                  // и так два раза в секунду  
}
```

`int` – целое число, `float` – число с дробной частью. Зачем `5.0`, а не просто `5`? Если написать `n * 5 / 1023`, программа поделит целое на целое и выбросит дробную часть ещё до того, как число попадёт в `u`. Вместо 2.49 в мониторе появится 2.00.

Запомни. `analogRead` – от 0 до 1023, это 0...5 В. Число в вольтах: $N \cdot 5 : 1023$. Вольты в число: $U : 5 \cdot 1023$, дробную часть отбрасываем.

5.7. Датчик линии: белое и чёрное

НА ПАЛЬЦАХ



Рисунок 5.11 – Датчик линии над белым полем и над чёрной линией

Датчик светит вниз и ловит отражённый свет. Белое поле возвращает много света, чёрная линия почти весь свет «съедает».

Датчик линии висит под днищем робота в нескольких миллиметрах от поля. В нём две детали. **ИК-светодиод** светит вниз инфракрасным светом – невидимым для глаза, как у пульта от телевизора. **Фототранзистор** рядом с ним – приёмник: чем больше света на него падает, тем больше тока он пропускает. От этого меняется напряжение на выходе датчика, и `analogRead` превращает его в число.

Белая бумага отражает почти весь свет, поэтому над полем приёмнику достаётся много света. Чёрная краска свет поглощает, и над линией на приёмник почти ничего не попадает. В каких числах это выразится, зависит от схемы датчика: у одних над белым число большое, а над чёрным маленькое, у других – наоборот. Поэтому первое, что делают с новым датчиком, – **смотрят его показания** в мониторе порта над полем и над линией.

Чтобы робот решал «линия или не линия», в программу записывают **порог** – число посередине между показаниями на белом и на чёрном:

$$\text{порог} = \frac{\text{показание на белом} + \text{показание на чёрном}}{2}.$$

Над белым 850, над чёрным 120: порог $(850 + 120) : 2 = 485$. Показание меньше порога – датчик ближе к «чёрному» числу, значит, под ним линия. Больше – поле.

```
int n = analogRead(A0);
if (n < 485) {
    // под датчиком линия
} else {
    // под датчиком белое поле
}
```

`if` значит «если»: если условие в скобках верно, выполняется то, что в первых фигурных скобках, а иначе (`else`) – во вторых. Подробно об этом – в главе 6, а как по двум датчикам удерживать робота на линии – в главе 7.

Показания для порога меряют на том самом поле и на той высоте, где поедет робот. Подними датчик на пару сантиметров – и разница между белым и чёрным станет гораздо меньше. Солнце из окна тоже подсвечивает поле невидимыми лучами, и числа уплывают. Бывают и модули попроще: на плате стоит маленький винтик-регулятор, им заранее задают порог, и модуль выдаёт только «0» или «1». В Tinkercad готового датчика линии нет, его изображают **фоторезистором** – деталью, сопротивление которой меняется от света.

Запомни. Белое отражает много света, чёрное – мало. Порог – среднее между показаниями над белым и над чёрным. Какое число у линии больше, а какое меньше, проверяй по показаниям своего датчика.

5.8. Дальномёры: ультразвук и инфракрасный свет

НА ПАЛЬЦАХ



Рисунок 5.12 – Дальномёр «крикнул» и ждёт эха

Звук успел долететь до стены и вернуться. Значит, до стены – половина пути, который он пролетел.

Звук – это дрожание воздуха, которое бежит от источника во все стороны. Сколько раз в секунду воздух дрожит, называют **частотой** и измеряют в **герцах** (Гц): 1 Гц – одно колебание в секунду, 1 кГц = 1000 Гц. Человек слышит звуки примерно от 20 до 20 000 Гц. Всё, что выше, – **ультразвук**: его слышат летучие мыши и дельфины, а мы – нет.

Звук летит не мгновенно. **Скорость звука в воздухе – около 340 м/с**: за секунду он пролетает треть километра. Поэтому гром доходит позже молнии, а крик в пустом спортзале возвраща-

ется эхом. От температуры скорость немного зависит: в мороз звук летит медленнее, в жару – быстрее. В задачах этой книги берём 340 м/с, если в условии не сказано другое.

Ультразвуковой дальномер HC-SR04

У дальномера HC-SR04 два «глаза»: один – крошечный динамик, другой – микрофон. И четыре вывода: VCC (питание 5 В), **Trig**, **Echo** и GND. Контроллер подаёт на Trig сигнал HIGH на 10 микросекунд – это команда «крикни». Дальномер восемь раз подряд щёлкает ультразвуком на частоте 40 кГц, и сразу после этого на выводе Echo появляется HIGH. Держится он, пока к микрофону не вернётся эхо.



Рисунок 5.13 – Сигналы дальномера во времени. Длительность сигнала на Echo – время полёта звука туда и обратно

Контроллер меряет, сколько длился сигнал на Echo, командой `pulseIn`. Результат – в **микросекундах** (мкс): $1 \text{ с} = 1000 \text{ мс} = 1\,000\,000 \text{ мкс}$. За это время звук долетел до препятствия и вернулся, то есть пролетел расстояние дважды. Поэтому **расстояние равно скорости звука, умноженной на время и делённой на два**:

$$s = \frac{v \cdot t}{2}.$$

Скорость звука удобно заранее перевести в сантиметры за микросекунду. 340 м/с – это 34 000 см/с. В секунде миллион микросекунд: $34\,000 : 1\,000\,000 = 0,034 \text{ см/мкс}$. Звук за одну микросекунду пролетает треть миллиметра.

Пример. Эхо пришло через 1160 мкс. $s = 1160 \cdot 0,034 : 2 = 19,7 \text{ см}$. В программах вместо этого часто делят время на 58: $1160 : 58 = 20 \text{ см}$. Откуда 58? Чтобы слетать на 1 см и обратно, звуку нужно $2 : 0,034 \approx 58,8 \text{ мкс}$. Число округлили для простоты, поэтому ответы двух способов чуть расходятся.

```

const int TRIG = 9;
const int ECHO = 10;

void setup() {
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);           // с этого вывода читаем
  Serial.begin(9600);
}

void loop() {
  digitalWrite(TRIG, LOW);        // сначала Trig точно в LOW
  delayMicroseconds(2);
  digitalWrite(TRIG, HIGH);       // «крикни»: импульс 10 мкс
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);
  long t = pulseIn(ECHO, HIGH);   // сколько мкс ждали эха
  float s = t * 0.034 / 2;        // расстояние в сантиметрах
  Serial.println(s);
  delay(100);
}

```

INPUT – с этого вывода сигнал принимают. Первые две строки `loop` опускают Trig в LOW на пару микросекунд. После прошлого круга он и так в LOW, но так пишут почти во всех примерах: импульс «крикни» тогда начинается с чистого листа, даже если до этого что-то оставило Trig включённым. `long` – тоже целое число, но с большим запасом: `pulseIn` может вернуть сотни тысяч микросекунд, а в `int` на Arduino помещается только до 32 767.

HC-SR04 меряет примерно от 2 до 400 см. Мягкие вещи – штору, свитер, пушистого кота – он видит плохо: они глушат звук. Стену под острым углом тоже: звук отскакивает от неё в сторону, как мяч, и к микрофону не возвращается.

ИК-дальномер

НА ПАЛЬЦАХ

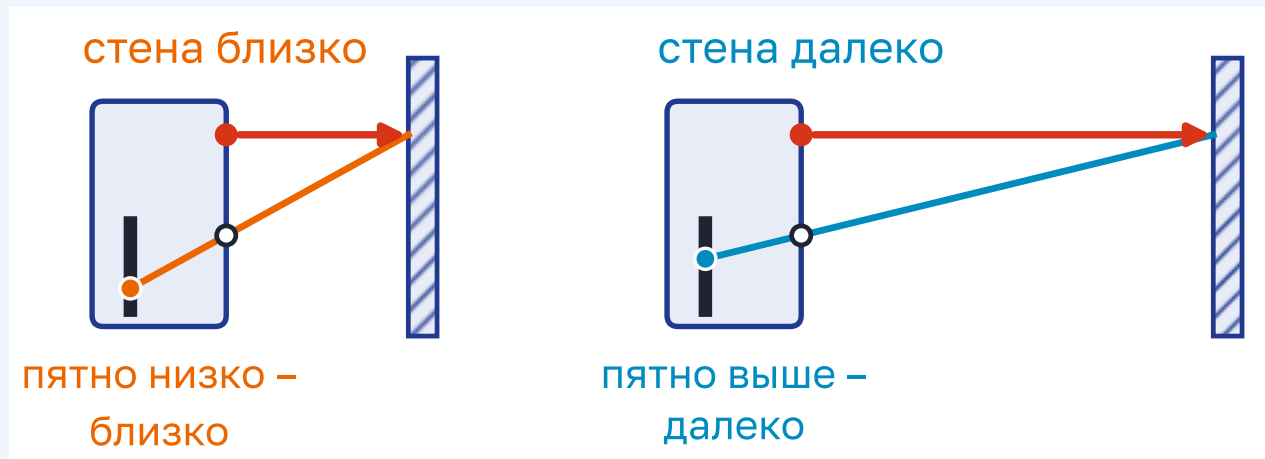


Рисунок 5.14 – ИК-дальномер: луч туда, отражённый свет – обратно, на полоску-приёмник

Стена близко – свет возвращается круто и попадает в самый низ приёмника.
Стена далеко – обратный луч идёт почти рядом с прямым, и пятно ложится выше.

ИК-дальномер светит узким невидимым лучом и смотрит, куда на его приёмнике попало отражённое пятнышко света: чем ближе препятствие, тем сильнее сдвинуто пятно. Выдаёт он напряжение, и контроллер читает его через `analogRead`, как датчик линии. Такие датчики меряют примерно от 10 до 80 см.

Главная трудность: **напряжение меняется с расстоянием неравномерно**. Вблизи каждый сантиметр заметно меняет показание, вдали – еле-еле. Посмотри на график условного датчика ниже: с 10 до 20 см напряжение падает на 1,1 В, а с 60 до 80 см – всего на 0,1 В. Поэтому простой пропорцией расстояние тут не посчитать – в сантиметры переводят по таблице или формуле, которую дают в паспорте датчика или в условии задачи.

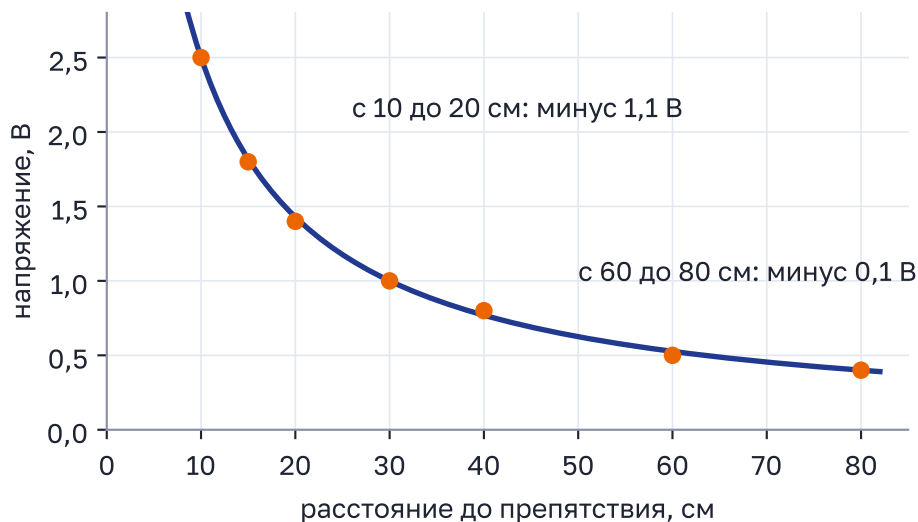


Рисунок 5.15 – Условный ИК-дальномер: так напряжение на выходе зависит от расстояния. Точки – значения из таблицы в задаче 5.12

И ещё одна ловушка. Если препятствие ближе нижней границы (у таких датчиков это около 10 см), показание снова падает, и датчик может «спутать» 5 см с 25 см. Поэтому его ставят поглубже в корпус, чтобы стена никогда не оказалась так близко.

Таблица 5.2

	Ультразвуковой	ИК-дальномер
Чем меряет	звуком	светом
Что получает контроллер	время эха, мкс	напряжение, <code>analogRead</code>
Расстояния	примерно 2–400 см	примерно 10–80 см
Чего не любит	мягкого, наклонной стены	стекла, яркого солнца

Запомни. $s = v \cdot t / 2$: звук летит туда и обратно. 340 м/с = 0,034 см/мкс; время в микросекундах даёт расстояние в сантиметрах. ИК-дальномер переводят в сантиметры только по таблице или формуле.

5.9. Разбираем задачи

Задача 5.1 Мотор прямо к плате

разминка · закон Ома, миллиамперы, почему мотор подключают через драйвер

Серёжа хочет подключить мотор робота прямо к выводу Arduino, без драйвера. На выводе 5 В, сопротивление обмотки мотора 12,5 Ом. Вывод безопасно отдаёт 20 мА. Какой ток потечёт через мотор в момент включения? Во сколько раз он больше допустимого? Ток дай в миллиамперах.

Дано: $U = 5$ В; $R = 12,5$ Ом; $I_{\text{доп}} = 20$ мА.

Найти: I – ?, во сколько раз I больше $I_{\text{доп}}$.

Решение

Шаг 1. Закон Ома. Знаем напряжение и сопротивление, ищем ток: $I = U/R = 5:12,5 = 0,4$ А.

Шаг 2. Переводим в миллиамперы. В одном ампере 1000 мА: $0,4 \cdot 1000 = 400$ мА.

Шаг 3. Сравниваем. Оба тока теперь в одних единицах: $400:20 = 20$. Ток в 20 раз больше того, что может отдать вывод.

Для мотора 400 мА – вполне обычный ток при трогании с места. А вывод платы рассчитан на светодиод. В условии принято безопасное значение 20 мА; если сравнивать с пределом 40 мА из курса, получится в 10 раз. Вывод Серёжиной платы, скорее всего, сгорел бы в первую же секунду.

 **Типичная ошибка.** Сравнить 0,4 и 20 без перевода единиц и решить, что ток «в 50 раз меньше допустимого – можно подключать». Амперы и миллиамперы отличаются в 1000 раз, сравнивать можно только числа в одинаковых единицах.

Ответ: 400 мА; в 20 раз больше допустимого.

Своя задача по программе сферы, модуль 2

Реши сам

5.1. Через резистор 1,5 кОм течёт ток 4 мА. Какое напряжение на резисторе? Ответ в вольтах.

5.2. При трогании с места мотор берёт от батареи 6 В ток 0,75 А. Каково сопротивление его обмотки? Во сколько раз этот ток больше тех 20 мА, которые безопасно отдаёт вывод Arduino?

Задача 5.2 Синяя фара

закон Ома, резистор для светодиода, стандартный ряд

На нос робота ставят синий светодиод-фару и подключают к выводу Arduino (5 В) через резистор, как на рисунке в разделе 5.2. На синем светодиоде оседает 3,1 В, ток через него дол-

жен быть не больше 15 мА. В наборе есть резисторы 100, 150, 220 и 330 Ом. а) Какой резистор взять? б) Какой ток пойдёт через светодиод с этим резистором? Ток дай в миллиамперах, округли до десятых.

Дано: $U_{\text{пит}} = 5 \text{ В}$; $U_{\text{св}} = 3,1 \text{ В}$; $I \leq 15 \text{ мА}$; ряд: 100, 150, 220, 330 Ом.

Найти: R из ряда; I с этим резистором.

Решение

Шаг 1. Сколько вольт достаётся резистору. Светодиод забирает 3,1 В, остальное – резистору: $5 - 3,1 = 1,9 \text{ В}$.

Шаг 2. Ток в амперах. $15 : 1000 = 0,015 \text{ А}$.

Шаг 3. Нужно сопротивление. $R = 1,9 : 0,015 = 126,7 \text{ Ом}$. Меньше этого брать нельзя: ток станет больше 15 мА.

Шаг 4. Выбираем из ряда. Ближайший больший номинал – 150 Ом. Резистор 100 Ом меньше 126,7 Ом и не подходит.

Шаг 5. Настоящий ток. На резисторе всё те же 1,9 В: $I = 1,9 : 150 = 0,01267 \text{ А} \approx 12,7 \text{ мА}$.

С фарой всё в порядке: 12,7 мА меньше 15 мА, она не перегреется и будет светить чуть мягче, чем могла бы. Будь в наборе резистор 130 Ом, ток вышел бы 14,6 мА – у самой границы.

⚠ Типичная ошибка. Забыть вычесть напряжение светодиода: $5 : 0,015 = 333 \text{ Ом}$. С таким расчётом возьмут резистор 330 Ом, и фара еле засветится – ток будет всего $1,9 : 330 \approx 5,8 \text{ мА}$. Вторая ошибка – округлить 126,7 Ом «по правилам» до 100 Ом: тогда пойдёт $1,9 : 100 = 19 \text{ мА}$, больше разрешённого.

Ответ: а) 150 Ом; б) 12,7 мА.

Своя задача по программе сферы, модуль 2

Реши сам

5.3. Красный светодиод (на нём оседает 2 В, ток не больше 20 мА) хотят зажечь от батарейки «Крона» на 9 В. Какой резистор нужно взять из ряда 220, 330, 390, 470, 560 Ом? Ответ в омах.

5.4. Два красных светодиода соединили последовательно, один за другим, и через резистор подключили к выводу Arduino (5 В). На каждом светодиоде оседает 2 В, ток через них должен быть 10 мА. Какое сопротивление нужно? Ответ в омах.

Задача 5.3 Куда поедет робот

драйвер L293D, направление вращения, дифференциальный привод, сопоставление

У робота два мотора, оба подключены через драйвер L293D. Левым мотором управляют входы IN1 и IN2, правым – IN3 и IN4. Сигнал HIGH на IN1 (и LOW на IN2) крутит левое колесо вперёд,

сигнал HIGH на IN3 (и LOW на IN4) – правое колесо вперёд. На входы скорости обоих моторов подано HIGH. Сопоставь наборы сигналов и то, что делает робот.

Таблица 5.3

Набор	IN1	IN2	IN3	IN4
А	HIGH	LOW	LOW	HIGH
Б	LOW	HIGH	LOW	HIGH
В	LOW	LOW	HIGH	LOW
Г	HIGH	LOW	HIGH	LOW

Что делает робот: 1 – едет вперёд; 2 – едет назад; 3 – поворачивает налево вокруг левого колеса; 4 – поворачивает направо вокруг правого колеса; 5 – разворачивается на месте направо; 6 – разворачивается на месте налево.

В ответ введи четыре цифры подряд – номера действий для наборов А, Б, В, Г.

Дано: таблица сигналов; вперёд – HIGH на первом входе пары и LOW на втором.

Найти: номера действий для А, Б, В, Г.

Решение

Шаг 1. Правило для одного мотора. Разные сигналы – колесо крутится: HIGH, LOW – вперёд; LOW, HIGH – назад. Одинаковые сигналы – колесо стоит.

Шаг 2. Расшифровываем каждый набор по колёсам.

- А: левое HIGH, LOW – вперёд; правое LOW, HIGH – назад.
- Б: левое LOW, HIGH – назад; правое LOW, HIGH – назад.
- В: левое LOW, LOW – стоит; правое HIGH, LOW – вперёд.
- Г: левое HIGH, LOW – вперёд; правое HIGH, LOW – вперёд.

Шаг 3. Что делает робот (правила дифференциального привода – в главе 1).

- А: колёса крутятся в разные стороны – разворот на месте. Левое вперёд, правое назад – робот вертится направо: это 5.
- Б: оба назад – едет назад: 2.
- В: левое стоит, правое толкает вперёд – робот поворачивает вокруг левого колеса налево: 3.
- Г: оба вперёд – едет вперёд: 1.

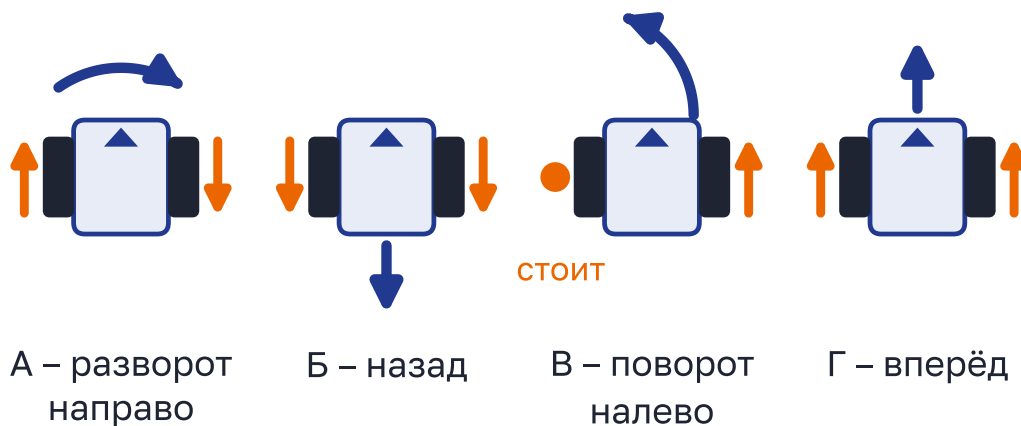


Рисунок 5.16 – Наборы из задачи 5.3, вид сверху, нос робота смотрит вверх. Оранжевые стрелки – куда крутится каждое колесо, синие – куда движется весь робот

Проверь А руками: вытяни руки в стороны, левую двигай вперёд, правую назад – повернёшься вправо.

❗ **Типичная ошибка.** В наборе В решить, что робот поворачивает направо: «ведь едет правое колесо». Робот поворачивает в сторону **медленного** колеса, а здесь стоит левое. Ещё путают пары входов и смотрят на IN2 и IN3 как на один мотор.

Ответ: 5231.

Своя задача по программе сферы, модули 2 и 6

Реши сам

5.5. Робот и подключение те же, что в задаче 5.3, действия те же, от 1 до 6. Наборы сигналов (IN1, IN2, IN3, IN4): А – LOW, HIGH, HIGH, LOW; Б – HIGH, LOW, LOW, LOW; В – HIGH, HIGH, HIGH, LOW. В ответ введи три цифры подряд для А, Б, В.

5.6. Робот тот же. Какие сигналы подать на IN1, IN2, IN3, IN4, чтобы он развернулся на месте налево? Запиши их четырьмя цифрами (1 – HIGH, 0 – LOW). И какое значение `analogWrite` подать на входы скорости, чтобы моторы крутились на 60 % от полной скорости?

Задача 5.4 Помедленнее, пожалуйста

ШИМ, `analogWrite`, пропорциональность, скорость

При `analogWrite(5, 255)` робот едет со скоростью 51 см/с. Робот должен проехать 1,2 м за 6 с с постоянной скоростью. Какое значение нужно подать командой `analogWrite`? Считай, что скорость робота пропорциональна значению ШИМ.

Дано: $v_{\max} = 51$ см/с при значении 255; $s = 1,2$ м; $t = 6$ с.

Найти: значение ШИМ – ?

Решение

Шаг 1. Нужная скорость. Переводим путь в сантиметры: $1,2 \text{ м} = 120 \text{ см}$. Скорость $v = s/t = 120:6 = 20 \text{ см/с}$ (как считать скорость, мы разбирали в главе 2).

Шаг 2. Какая это доля от полной скорости. $20:51 = 0,392$.

Шаг 3. Значение ШИМ. Скорость пропорциональна значению, значит, значение – та же доля от 255: $0,392 \cdot 255 = 100$.

Можно короче: на каждую единицу ШИМ приходится $51:255 = 0,2 \text{ см/с}$ скорости. Для 20 см/с нужно $20:0,2 = 100$ единиц.

Значение 100 – меньше половины от 255, и скорость 20 см/с – тоже меньше половины от 51: при пропорциональности доли совпадают.

 **Типичная ошибка.** Взять шкалу `analogRead` вместо `analogWrite`: $20:51 \cdot 1023 = 401$. Такое значение `analogWrite` не примет – у него предел 255. Другая ошибка – подставить путь в метрах, а скорость в см/с: $1,2:6 = 0,2$ – и дальше считать, будто нужно 0,2 см/с.

Ответ: 100.

Своя задача по программе сферы, модуль 6

Реши сам

5.7. При значении ШИМ 200 робот едет со скоростью 32 см/с. С какой скоростью он поедет при значении 150? Скорость пропорциональна значению ШИМ. Ответ в сантиметрах в секунду.

5.8. Мотор другого робота трогается с места только при значении ШИМ 60. От 60 до 255 скорость растёт равномерно: при 60 она равна 0, при 255 – 39 см/с. Какое значение нужно для скорости 26 см/с?

Задача 5.5 Сколько заездов выдержит аккумулятор

ёмкость аккумулятора, сумма токов, проценты, перевод часов в минуты

Команда готовит робота к тренировке. Аккумулятор ёмкостью 1800 мА·ч. Каждый из двух моторов на ходу берёт 300 мА, плата вместе с датчиками – ещё 120 мА. Разряжать аккумулятор можно только до 20 %, то есть использовать 80 % ёмкости. Один заезд длится 6 минут, и всё это время моторы работают. Сколько полных заездов проедет робот до подзарядки?

Дано: ёмкость 1800 мА·ч; моторы $2 \times 300 \text{ мА}$; плата и датчики 120 мА; используем 80 %; заезд 6 мин.

Найти: число полных заездов – ?

Решение

Шаг 1. Ток всего робота. Токи всех устройств складываем: $2 \cdot 300 + 120 = 720 \text{ мА}$.

Шаг 2. Полезная ёмкость. 80 % от 1800 мА·ч: $0,8 \cdot 1800 = 1440 \text{ мА·ч}$.

Шаг 3. Время работы. $t = 1440 : 720 = 2$ ч.

Шаг 4. В минуты. $2 \cdot 60 = 120$ мин.

Шаг 5. Заезды. $120 : 6 = 20$ заездов. Разделилось без остатка, округлять не нужно. Будь там 20,7 – ответ был бы 20: двадцать первый заезд робот не доедет до конца.

На тренировке час-полтора пролетают быстро, и двух часов езды команде как раз хватит, если не держать робота включённым между заездами.

⚠ Типичная ошибка. Забыть про 20 % запаса и получить $1800 : 720 \cdot 60 : 6 = 25$ заездов. Или посчитать только один мотор: $1440 : 420 = 3,43$ ч. Ещё бывает, что 1440 делят на 720 и сразу на 6 – но 2 здесь часы, а заезд в минутах.

Ответ: 20 заездов.

Своя задача по программе сферы, модуль 6

Реши сам

5.9. Робот берёт от аккумулятора 800 мА. Ёмкость аккумулятора 1200 мА·ч, использовать можно всю. Сколько минут проработает робот?

5.10. Роботу нужно 45 минут ездить по полигону, он берёт ток 1,2 А. Аккумулятор можно разряжать только до 20 %. Какой самый маленький аккумулятор подойдёт из трёх: 1000, 1200 или 1500 мА·ч? Ответ в мА·ч.

Задача 5.6 Порог для датчика линии

analogRead, напряжение и число, порог датчика линии

Датчик линии подключён ко входу А0. Мультиметром измерили напряжение на его выходе: над белым полем 4,0 В, над чёрной линией 1,0 В. а) Какие числа покажет `analogRead` над полем и над линией? б) Какой порог записать в программу, если его берут посередине? в) Датчик показал 700. Что под ним – поле или линия?

Дано: $U_{\text{бел}} = 4,0$ В; $U_{\text{чёрн}} = 1,0$ В; шкала `analogRead` 0...1023 на 0...5 В.

Найти: $N_{\text{бел}}$, $N_{\text{чёрн}}$, порог; что значит 700.

Решение

Шаг 1. Белое поле в число. $N = U : 5 \cdot 1023 = 4,0 : 5 \cdot 1023 = 0,8 \cdot 1023 = 818,4$. Дробную часть отбрасываем: 818.

Шаг 2. Чёрная линия в число. $1,0 : 5 \cdot 1023 = 0,2 \cdot 1023 = 204,6$, получаем 204. Не 205: контроллер дробную часть отбрасывает, а не округляет.

Шаг 3. Порог. $(818 + 204) : 2 = 1022 : 2 = 511$.

Шаг 4. Что значит 700. У этого датчика белое – большое число, чёрное – маленькое. 700 больше порога 511, значит, ближе к «белому» числу: под датчиком поле.

Порог 511 – почти середина всей шкалы 0...1023. Не случайно: 4 В и 1 В отстоят на одинаковые 1,5 В от 2,5 В, середины шкалы в вольтах.

 **Типичная ошибка.** Сосчитать порог в вольтах, $(4 + 1) : 2 = 2,5$, и записать в программу число 2,5. Программа сравнивает `analogRead` с порогом, а `analogRead` возвращает числа от 0 до 1023 – порог 2,5 почти всегда будет меньше показаний. Порог должен быть в тех же «делениях», что и показания.

Ответ: а) 818 и 204; б) порог 511; в) поле.

Своя задача по программе сферы, модуль 7

Реши сам

5.11. Над белым полем датчик линии показывает 760, над чёрной линией – 180. Порог берут посередине. Какие из показаний означают линию: 1) 175; 2) 420; 3) 505; 4) 790? В ответ введи номера подряд, в порядке возрастания.

5.12. ИК-дальномер робота подключён ко входу A0. По паспорту его напряжение на разных расстояниях такое:

Таблица 5.4

Расстояние, см	Напряжение, В
10	2,5
15	1,8
20	1,4
30	1,0
40	0,8
60	0,5
80	0,4

`analogRead(A0)` вернул 286. На каком расстоянии препятствие? Ответ в сантиметрах.

Задача 5.7*. Робот и сугроб

ультразвуковой дальномер, эхо, скорость звука, пропорция

Зимой ребята из кружка в Академгородке испытывают робота во дворе. Робот едет к сугробу и останавливается, когда программа посчитает по ультразвуковому дальномеру, что до сугроба 30 см или меньше. Программа считает расстояние по скорости звука 340 м/с. Но на улице -20°C , и звук летит медленнее – 319 м/с. На каком настоящем расстоянии от сугроба остановится робот? Ответ в сантиметрах, округли до десятых.

Дано: $s_{\text{прогр}} = 30$ см; $v_{\text{прогр}} = 340$ м/с; $v_{\text{наст}} = 319$ м/с.

Найти: $s_{\text{наст}} - ?$

Решение

Шаг 1. Что меряет дальномер. Дальномер ничего не знает про сантиметры. Он меряет только время эха t . Сантиметры из этого времени получает программа, умножая на скорость звука. Значит, в момент остановки время эха было таким, что по формуле программы выходит 30 см.

Шаг 2. Скорости в сантиметрах за микросекунду. 340 м/с $= 0,034$ см/мкс. 319 м/с $= 31\,900$ см/с, $31\,900 : 1\,000\,000 = 0,0319$ см/мкс.

Шаг 3. Время эха в момент остановки. Из $s = v \cdot t / 2$ получаем $t = 2 \cdot s / v$:
 $t = 2 \cdot 30 : 0,034 = 1764,7$ мкс.

Шаг 4. Сколько пролетел звук. За те же 1764,7 мкс звук летел с настоящей скоростью:
 $s_{\text{наст}} = 0,0319 \cdot 1764,7 : 2 = 28,15$ см $\approx 28,1$ см.

Короткий путь – пропорция. Время одно и то же, поэтому расстояние пропорционально скорости: $s_{\text{наст}} = 30 \cdot 319 : 340 = 28,15$ см. Настоящий звук медленнее в $340 : 319$ раза, и настоящее расстояние во столько же раз меньше.

Ошибка получилась около 2 см, это примерно 6 %. Возле сугроба это мелочь. Но ошибка растёт вместе с расстоянием: если стена в 5 м, программа на морозе промахнётся уже на $500 - 500 \cdot 319 : 340 \approx 31$ см.

 **Типичная ошибка.** Перепутать, куда сдвигается ответ: «звук медленнее – значит, сугроб дальше» и умножить на $340 : 319$, получив 32 см. Проверка здравым смыслом: медленный звук возвращается позже, программа видит «долгое» эхо и думает, что сугроб дальше, чем он есть. Значит, настоящее расстояние **меньше** 30 см. Если идёшь длинным путём, двойку ставь и когда ищешь время, и когда ищешь расстояние: забудешь в одном месте – ответ уедет вдвое.

Ответ: 28,1 см.

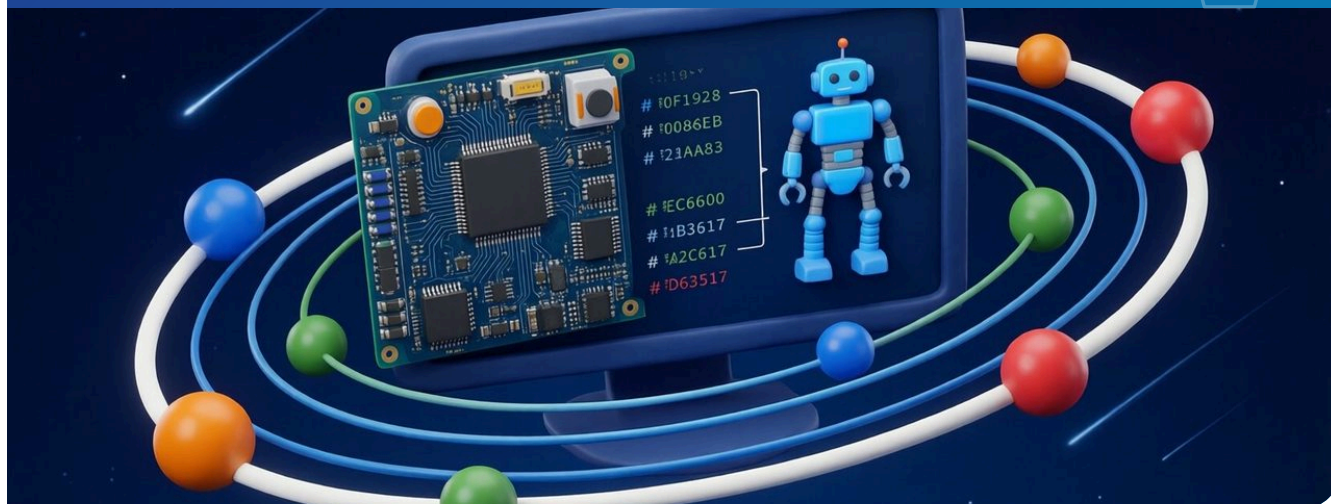
Своя задача по программе сферы, модуль 8

Реши сам

5.13. Эхо вернулось к дальномеру через 3500 мкс (скорость звука 340 м/с). Робот едет прямо к стене со скоростью 25 см/с и должен остановиться в 12 см от неё. Через сколько секунд после этого измерения нужно выключить моторы? Разгон и торможение не учитывай. Ответ в секундах, округли до десятых.

5.14.* Летом на солнечной площадке +30 °С, и звук летит со скоростью 349 м/с. Программа робота по-прежнему считает по 340 м/с и показывает, что до стенки 40 см. Каково настоящее расстояние? Ответ в сантиметрах, округли до десятых.

Программирование: Arduino и машины состояний



В главе 4 робот понимал команды «вперёд», «направо» и «пока впереди свободно». Теперь те же мысли запишем на языке C++, на котором программируют плату Arduino. Робот научится помнить числа, получит свои команды из нескольких строк, схему поведения из состояний и память о пройденном пути. Задача отбора по этой теме – короткий скетч и вопрос к нему: что напечатает монитор порта, куда поедет робот, в какой строке ошибка, в каком порядке поставить перемешанные строки. Ответ – число или строка из цифр.

6.1. Переменные и монитор порта

НА ПАЛЬЦАХ

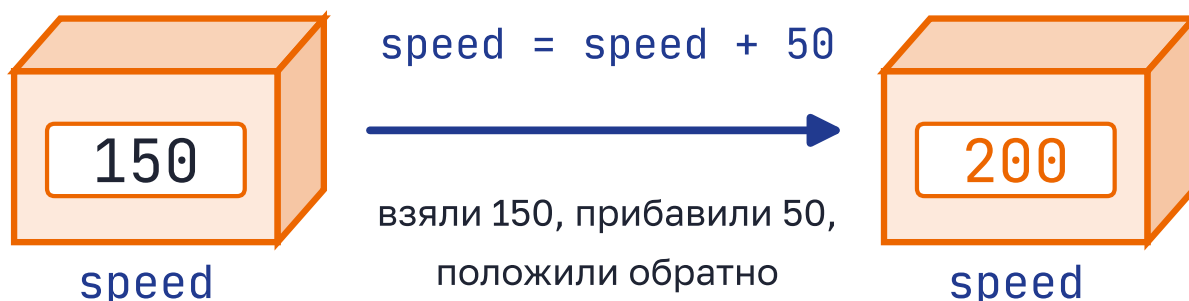


Рисунок 6.1 – Переменная – коробка с именем на боку

В коробке всегда лежит одно число. Кладёшь новое – старое пропадает.

Программу для Arduino называют **скетчем**. Ты уже видел его в главе 5: `setup()` выполняется один раз при включении, `loop()` – по кругу, пока есть питание. Каждая команда кончается точкой с запятой, а всё, что после `//`, – комментарий для человека.

Переменная – ячейка памяти с именем, в которой лежит число. Чтобы завести переменную, пишут её тип, имя и начальное значение:

```
int speed = 150;           // переменная speed, в ней 150

void setup() {
    Serial.begin(9600);
    Serial.println(speed);  // 150
    speed = speed + 50;     // взяли 150, прибавили 50, положили обратно
    Serial.println(speed);  // 200
}

void loop() {
}
```

Строка `speed = speed + 50;` сбивает с толку тех, кто помнит математику: как число может равняться себе плюс 50? В программе знак `=` означает **«положи»**. Компьютер сначала считает правую часть (берёт из коробки 150, прибавляет 50), а потом кладёт результат в коробку слева. Прежнее 150 при этом пропадает.

Тип говорит, какое число поместится в коробку:

- `int` – целое, на Arduino Uno от –32 768 до 32 767;
- `long` – тоже целое, но с огромным запасом (больше двух миллиардов); в нём хранят, например, результат `pulseIn`;
- `float` – дробное; дробную часть в программе пишут через **точку**: `0.5`, а не `0,5`;
- `bool` – «да или нет»: `true` или `false`.

Если перед типом стоит слово `const`, это постоянная: её значение задают один раз и больше не меняют. Так дают имена номерам выводов: `const int ENA = 5;`.

НА ПАЛЬЦАХ



Рисунок 6.2 – Семь конфет на двоих: $7 / 2$ – сколько досталось каждому, $7 \% 2$ – сколько осталось

Резать конфету пополам C++ не умеет. Каждому по три, а одна остаётся в вазе.

Деление целых – с подвохом. Если оба числа целые, C++ отбрасывает дробную часть: $7 / 2$ даёт 3. Дробь даже не округляется, её просто выбрасывают: $99 / 10$ – это 9, хотя 9,9 почти

10. Знак `%` даёт **остаток** от деления – ту самую конфету, которая не поделилась: `7 % 2` – это 1, `17 % 5` – это 2. Чтобы получить дробный ответ, одно из чисел пишут с точкой: `7.0 / 2` даёт 3.5. Эту ловушку ты уже встречал в главе 5, когда переводил показания `analogRead` в вольты.

Монитор порта – окно на компьютере, куда плата присылает текст и числа. Робот не умеет говорить, но через это окно видно, что у него «в голове». В `setup()` пишут `Serial.begin(9600);` – это включает связь, 9600 – её скорость. Дальше:

- `Serial.print(x);` – напечатать `x` и остаться на той же строке;
- `Serial.println(x);` – напечатать `x` и перейти на новую строку.

```
long d = 23;
Serial.print("d = ");
Serial.println(d);      // в мониторе: d = 23
```

Текст в кавычках печатается как есть, а имя без кавычек заменяется числом из коробки. В Tinkercad монитор порта открывается внизу окна с кодом.

Где живёт переменная. Переменная, объявленная в самом начале скетча, до `setup()`, живёт всё время, пока работает плата. А переменная, объявленная внутри `loop()`, создаётся заново на каждом круге – и каждый раз с начальным значением. Если в ней что-то копить, она будет забывать накопленное. На этом споткнёмся в задаче 6.3.

Запомни. `=` в программе – «положи», а не «равно». Целое делённое на целое – дробь отбрасывается: `7 / 2 = 3`, `7 % 2 = 1`. `println` переводит строку, `print` – нет. Счётчик, который должен копить, объявляй до `setup()`.

6.2. Условия и циклы на C++

НА ПАЛЬЦАХ

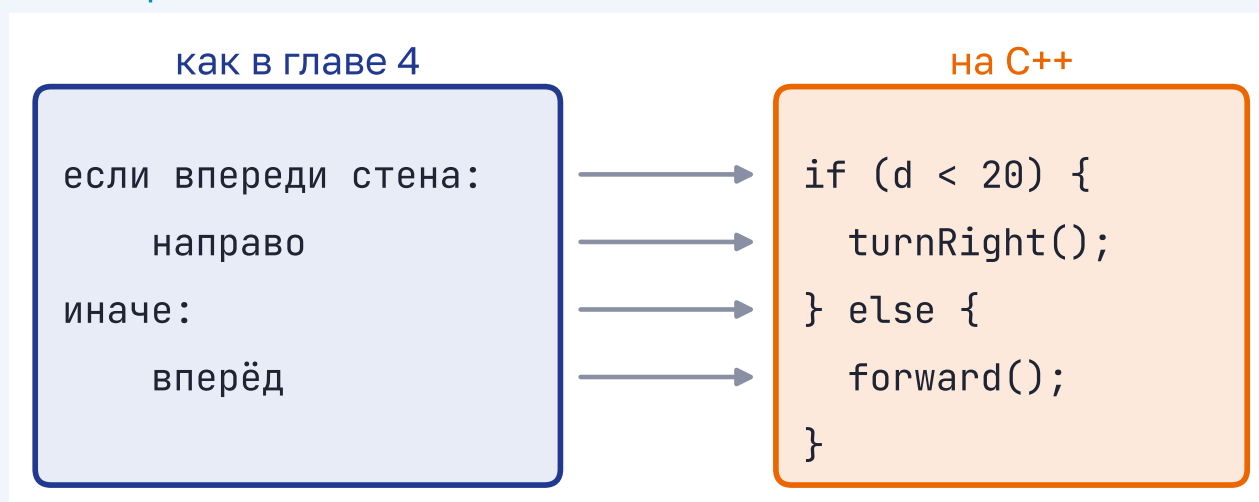


Рисунок 6.3 – Та же программа, что в главе 4, только словами C++

Слова другие, мысль та же. «Если» стало `if`, «иначе» – `else`, а вместо отступов команды собирают фигурные скобки.

Ветвление на C++ пишут так:

```
long d = distanceCm(); // расстояние до препятствия, см  
if (d < 20) {  
    stopMotors();  
} else {  
    forward();  
}
```

Условие – в круглых скобках, команды каждой ветки – в фигурных. Функции `distanceCm()`, `forward()` и `stopMotors()` мы напишем в разделе 6.3, пока просто читай их имена. Сравнивать можно так:

- `<` меньше, `>` больше, `≤` меньше или равно, `≥` больше или равно;
- `=` равно (два знака!), `≠` не равно.

Одиночный `=` внутри `if` – самая коварная ошибка этой главы. `if (d = 20)` не сравнивает, а **кладёт** 20 в `d`, и условие становится верным всегда. Плата не ругается, просто робот ведёт себя странно.

Два условия соединяют словами-значками: `&&` – «и», `||` – «или». Например, `if (d > 0 && d < 20)` – «расстояние больше нуля и меньше двадцати». Зачем здесь `d > 0`? Если эхо не вернулось совсем (штора, пушистый кот), `pulseIn` выдаёт 0, и без этой проверки робот испугается пустоты.

Цикл со счётчиком на C++ – это `for`. Вот светодиод на выводе 13 мигает три раза:

```
for (int i = 0; i < 3; i++) {
    digitalWrite(13, HIGH);
    delay(200);
    digitalWrite(13, LOW);
    delay(200);
}
```

В заголовке `for` три части через точку с запятой. `i++` – короткая запись для `i = i + 1`.

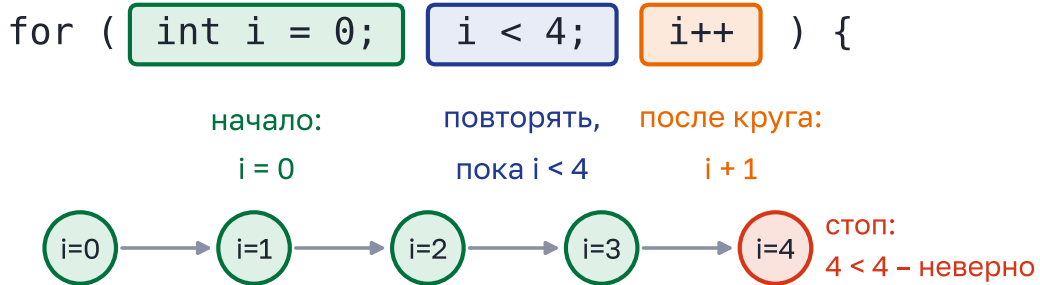


Рисунок 6.4 – Три части заголовка цикла `for`. Тело выполняется, пока условие верно: при `i = 0, 1, 2, 3` – четыре раза

Переменная `i` сначала равна 0. Перед каждым кругом проверяется условие: верно – выполняется тело, потом `i` увеличивается на 1, и снова проверка. Как только условие стало неверным, цикл закончился. Сколько раз выполнится тело, считай по началу и условию:

- `for (int i = 0; i < 4; i++)` – `i = 0, 1, 2, 3`: **4 раза**;
- `for (int i = 1; i ≤ 4; i++)` – `i = 1, 2, 3, 4`: тоже 4 раза;
- `for (int i = 0; i ≤ 4; i++)` – `i = 0, 1, 2, 3, 4`: **5 раз**. На этой строчке теряют больше всего баллов.

Цикл с условием – `while` («пока»). Проверка стоит перед каждым кругом, как в главе 4:

```
while (distanceCm() > 20) { // пока до стены больше 20 см
    forward();
}
stopMotors();
```

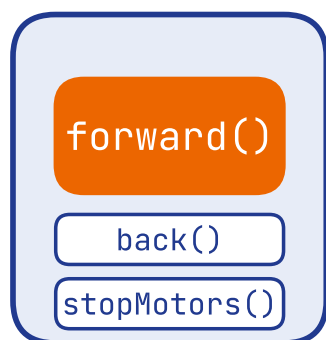
Робот едет, пока расстояние больше 20 см. Стало 20 или меньше – цикл закончился, робот остановился. Если стена была ближе с самого начала, тело не выполнится ни разу. А если условие не станет неверным никогда, программа застрянет в цикле навсегда. Иногда так делают нарочно: `while (true) {}` – «стоять и ничего не делать до выключения».

Если этот кусок стоит внутри `loop()`, получается цикл в цикле. Робот доехал до стены и встал, `loop()` начался сначала, `while` снова спросил датчик: до стены уже 20 см или меньше, тело пропускается, и робот стоит, пока стену не уберут.

Запомни. Сравнение – `=`, а `=` – «положи». `for (int i = 0; i < N; i++)` – тело выполнится `N` раз, с `≤` – на один раз больше. `while` проверяет условие перед каждым кругом.

6.3. Функции: свои команды для работа

НА ПАЛЬЦАХ



```
digitalWrite(IN1, HIGH);  
digitalWrite(IN2, LOW);  
digitalWrite(IN3, HIGH);  
digitalWrite(IN4, LOW);
```

одна кнопка – четыре команды

Рисунок 6.5 – Функция – кнопка на пульте

Нажимаешь одну кнопку «вперёд» – внутри срабатывают четыре команды для драйвера. Функция прячет их под одним именем.

Чтобы робот поехал вперёд, драйверу L293D нужно подать четыре сигнала: два на левый мотор, два на правый (глава 5). Писать эти четыре строки каждый раз долго, и легко перепутать HIGH с LOW. Поэтому их собирают в **функцию** – это подпрограмма из главы 4, только на C++. Описывают один раз, а потом **вызывают** по имени.

Вот скетч, с которого начинается почти любой робот курса. Моторы подключены через L293D, как в главе 5: левый – к выводам 5, 7, 8, правый – к выводам 6, 2, 4.

```

const int ENA = 5, IN1 = 7, IN2 = 8; // левый мотор: скорость и направление
const int ENB = 6, IN3 = 2, IN4 = 4; // правый мотор

void forward() { // оба колеса вперёд
    digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);
}

void back() { // оба колеса назад
    digitalWrite(IN1, LOW); digitalWrite(IN2, HIGH);
    digitalWrite(IN3, LOW); digitalWrite(IN4, HIGH);
}

void turnLeft() { // левое назад, правое вперёд
    digitalWrite(IN1, LOW); digitalWrite(IN2, HIGH);
    digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);
}

void turnRight() { // левое вперёд, правое назад
    digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW); digitalWrite(IN4, HIGH);
}

void stopMotors() { // все входы LOW - стоим
    digitalWrite(IN1, LOW); digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW); digitalWrite(IN4, LOW);
}

void setup() {
    pinMode(ENA, OUTPUT); pinMode(IN1, OUTPUT); pinMode(IN2, OUTPUT);
    pinMode(ENB, OUTPUT); pinMode(IN3, OUTPUT); pinMode(IN4, OUTPUT);
    analogWrite(ENA, 200); // скорость левого мотора
    analogWrite(ENB, 200); // скорость правого
}

void loop() {
    forward(); // 2 с вперёд
    delay(2000);
    turnRight(); // полсекунды разворота направо
    delay(500);
}

```

Посмотрим на `forward()` по частям.

- `void forward() { ... }` – описание. `void` значит «ничего не возвращает»: функция просто делает дело. Пустые скобки после имени – ей ничего не нужно сообщать.
- Внутри фигурных скобок – тело, те самые четыре сигнала. Две команды на одной строке – это можно, лишь бы у каждой была своя точка с запятой.
- `forward();` в `loop()` – **вызов**. Встретив его, плата выполняет тело функции и возвращается к следующей строке.
- `turnLeft()` и `turnRight()` – развороты на месте: колёса крутятся в разные стороны. Почему так робот вертится, разобрано в главе 1.

Функция моторов только **переключает** сигналы и сразу заканчивается. Робот поедет вперёд и будет ехать, пока ему не дадут другую команду, а сколько он проедет, решает `delay` после вызова. Поэтому в `loop()` пара «команда моторам – `delay`» описывает один кусок пути.

Функция с параметром. Иногда функции нужно что-то сообщить. Число в скобках называют **параметром**:

```
void goForward(int ms) {    // ехать вперёд ms миллисекунд и встать
    forward();
    delay(ms);
    stopMotors();
}
```

Вызов `goForward(1500);` кладёт 1500 в коробку `ms`, и робот едет полторы секунды.
`goForward(300);` – 0,3 с. Одна функция на все случаи.

Функция, которая возвращает число. Дальномер из главы 5 тоже удобно спрятать в функцию. Вместо `void` перед именем стоит тип ответа, а сам ответ отдаёт команда `return`:

```
const int TRIG = 9, ECHO = 10;

long distanceCm() {
    digitalWrite(TRIG, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG, HIGH);    // импульс 10 мкс – «щёлк»
    delayMicroseconds(10);
    digitalWrite(TRIG, LOW);
    long t = pulseIn(ECHO, HIGH); // время туда и обратно, мкс
    return t / 58;               // в сантиметры
}
```

Теперь `long d = distanceCm();` кладёт в `d` расстояние в сантиметрах. Откуда деление на 58 – в главе 5. Выводы `Trig` и `Echo` тоже надо настроить в `setup()`: `pinMode(TRIG, OUTPUT);`
`pinMode(ECHO, INPUT);`.

Дальше в главе мы пользуемся этими функциями, не переписывая их: `forward()`, `back()`, `turnLeft()`, `turnRight()`, `stopMotors()` и `distanceCm()`. В справочнике в конце книги есть вариант, где скорость каждого колеса задаётся параметром.

Запомни. Функцию описывают один раз: `void имя() { ... }`, вызывают по имени со скобками: `имя();`. Параметр передаёт в функцию число, `return` отдаёт число обратно. Функция моторов только переключает сигналы – как долго ехать, решает `delay` после неё.

Совет. Проверь функции моторов, прежде чем писать большую программу. Вызови по очереди `forward()`, `back()`, `turnLeft()`, `turnRight()` с `delay(1000)` между ними и посмотри в симуляторе, как крутятся валы. Перепутанный провод лучше найти сейчас, чем на третьем часу отладки.

6.4. Машина состояний

НА ПАЛЬЦАХ

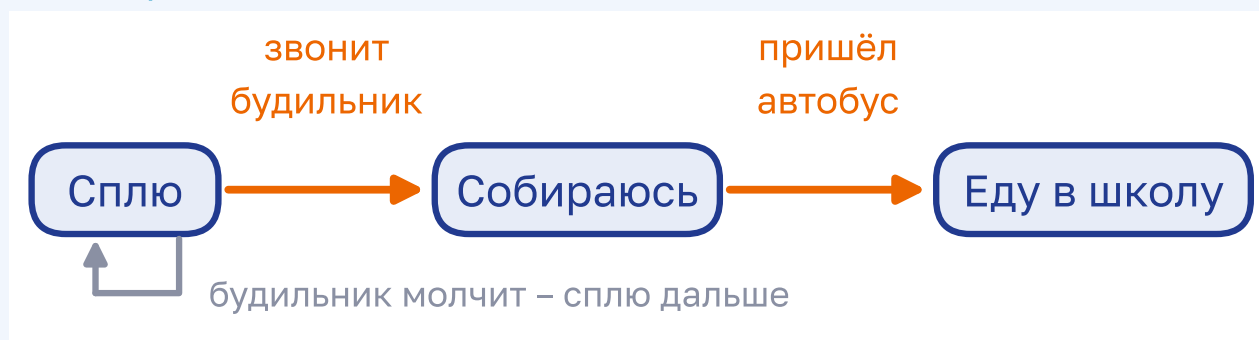


Рисунок 6.6 – Утро школьника из трёх состояний

В каждый момент ты в одном состоянии: спишь, собираешься или едешь. В другое переходишь, когда что-то случилось, а пока ничего не случилось – остаёшься, где был.

Робот, который объезжает препятствия, в разные моменты занят разным. Едет прямо и поглядывает на дальномер. Увидел стену – отъезжает назад. Отъехал – поворачивает. Повернул – снова едет прямо. Каждое такое занятие называют **состоянием**, а всю схему – **машиной состояний** (ещё говорят «конечный автомат»).

В языке ПРИМС три ключевых элемента: состояния, события и действия. К ним добавляются ограждающие условия. Те же слова – в справочнике в конце книги:

- **состояние** – режим работы: «еду», «назад», «поворот»;
- **событие** – то, что случилось: пришло новое расстояние, прошло время;
- **действие** – что робот делает: включить моторы вперёд, засечь время;
- **ограждающее условие** (далее в книге – просто «условие») – проверка в квадратных скобках, без которой переход не сработает: «расстояние меньше 20 см»;
- **переход** – смена состояния. Отдельным элементом он не считается: на схеме это стрелка, подписанная событием.

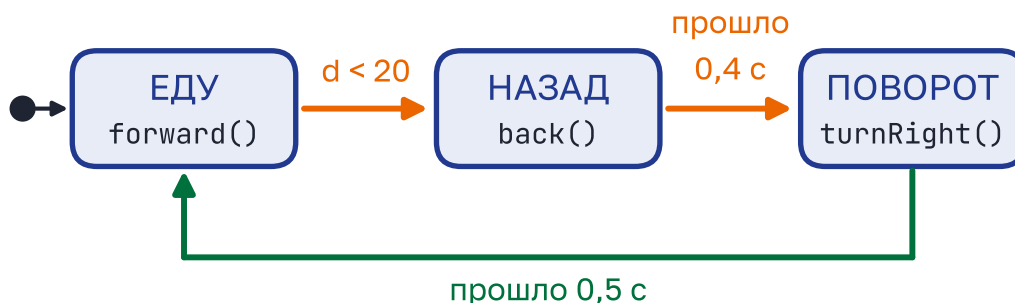


Рисунок 6.7 – Машина состояний для объезда препятствия. Точка слева – старт, внутри прямоугольников – что робот делает в этом состоянии, на стрелках – когда он переходит в следующее

Прочитаем схему. Робот включается в состоянии ЕДУ и едет вперёд. Пока расстояние больше или равно 20 см, ничего не происходит – робот остаётся, где был. Стало меньше 20 – переход

в НАЗАД: робот отъезжает. В НАЗАД на дальномер уже не смотрят, переход отсюда – по времени: через 0,4 с робот переходит в ПОВОРОТ. Ещё через 0,5 с – снова в ЕДУ.

Такие схемы рисуют в **Кибериаде IDE** на языке **ПРИМС** – с ним знакомит третий модуль курса сферы. Программу там не пишут строчками: ставят состояния, соединяют их стрелками и подписывают стрелки по правилу «событие [условие] / действие». По готовой схеме редактор сам генерирует код на C++. Но даже без редактора полезно сначала нарисовать схему на бумаге: когда видны все стрелки, сразу заметно, что робот может застрять в каком-то состоянии навсегда.

Как это выглядит на C++. Номер текущего состояния хранится в переменной. На каждом круге `loop()` программа смотрит, в каком состоянии робот, делает то, что положено, и проверяет стрелки, которые выходят из этого состояния.

```
const int G0 = 0, BACK = 1, TURN = 2;    // номера состояний
int state = G0;                           // в каком состоянии робот сейчас
unsigned long t0 = 0;                     // когда попали в это состояние, мс

void setup() {
  Serial.begin(9600);
}

void loop() {
  long d = distanceCm();                  // расстояние до препятствия, см
  if (state == G0) {
    forward();
    if (d < 20) {                         // препятствие ближе 20 см
      state = BACK;
      t0 = millis();                     // засекли время
    }
  } else if (state == BACK) {
    back();
    if (millis() - t0 ≥ 400) {            // отъезжали 0,4 с
      state = TURN;
      t0 = millis();
    }
  } else if (state == TURN) {
    turnRight();
    if (millis() - t0 ≥ 500) {           // поворачивали 0,5 с
      state = G0;
    }
  }
  Serial.println(state);
  delay(100);
}
```

Что здесь нового:

- `G0`, `BACK`, `TURN` – просто имена для чисел 0, 1, 2. `state == BACK` читать легче, чем `state == 1`.
- `millis()` – сколько миллисекунд прошло с включения платы. Это секундомер, который запустили при включении и никогда не останавливают. Число быстро растёт (за час – 3 600 000), поэтому время хранят в `unsigned long` – целом без минуса с огромным запасом.

- `t0 = millis();` – «засечь время» в момент перехода. Это **действие на входе** в новое состояние: выполняется один раз, когда в него попали. А `millis() - t0` – сколько мы уже в этом состоянии.
- `delay(100)` в конце – один круг `loop()` длится примерно 0,1 с. Так проще считать шаги при трассировке.
- `Serial.println(state)` печатает номер состояния на каждом круге. Монитор порта покажет, как робот «думает».

Почему не написать проще: `back(); delay(400); turnRight(); delay(500);` ? Для одного объезда так можно. Но пока идёт `delay`, плата ничего другого не делает: не смотрит на датномер, не проверяет кнопку. Робот со скоростью 25 см/с за `delay(2000)` проедет вслепую $25 \cdot 2 = 50$ см. В машине состояний `loop()` крутится часто, и робот успевает заметить, что случилось.

Трассировка машины состояний. Заведи таблицу: номер круга, время, показание датчика, состояние в начале круга, выполнено ли условие перехода, состояние в конце. Заполняй строку за строкой, как в главе 4. Если в условии перехода время, считай его от момента, когда засекли `t0`, а не от включения.

Запомни. В каждый момент робот в одном состоянии. Переход срабатывает, только когда верно его условие; иначе робот остаётся, где был. Условия переходов проверяются только у стрелок, выходящих из текущего состояния. Время в состоянии – `millis() - t0`, где `t0` засекли при входе.

Совет. Прежде чем писать код, нарисуй схему: прямоугольники, стрелки, подписи. Потом проверь две вещи: из каждого состояния есть выход, и в каждое, кроме стартового, есть вход. Нашёл состояние без выхода – робот в нём застрянет.

6.5. Массивы: робот запоминает путь

НА ПАЛЬЦАХ



Рисунок 6.8 – Массив – ряд пронумерованных ящиков под одним именем

У всех ящиков одно имя `d`, а различают их по номеру. Номера начинаются с нуля.

Пусть робот пять раз измерил расстояние. Можно завести пять переменных `d0`, `d1`, `d2`, `d3`, `d4`. А если измерений сто? Для этого есть **массив** – ряд ячеек под одним именем. Номер ячейки называют **индексом** и пишут в квадратных скобках.

```
int d[5] = {34, 28, 41, 17, 23}; // массив из 5 чисел
Serial.println(d[0]);           // 34 – первый ящик
Serial.println(d[3]);           // 17 – четвёртый ящик
d[1] = 30;                      // положили 30 во второй ящик
```

- `int d[5]` – массив из пяти целых. Размер пишут один раз, при объявлении, потом он не меняется.
- Индексы идут **с нуля**: у массива из пяти ячеек они 0, 1, 2, 3, 4. Ячейки `d[5]` нет.
- `d[3]` – ячейка с индексом 3, то есть **четвёртая** по счёту. Путать эти два слова – ошибка номер один с массивами.

Если обратиться к `d[5]`, плата не остановит программу и не предупредит. Она прочитает или испортит чужую память, и робот начнёт вести себя непонятно. Следи, чтобы индекс не выходил за пределы.

С массивом почти всегда работает цикл `for`: индекс бежит от 0 до последнего ящика. Сложим все измерения. Помни: во втором ящике теперь 30, мы его только что поменяли.

```
int s = 0;
for (int i = 0; i < 5; i++) {
    s = s + d[i];
}
Serial.println(s);           // 145 = 34 + 30 + 41 + 17 + 23
Serial.println(s / 5);       // 29 – среднее, дробь отброшена
```

Запоминание пути. В модуле 10 курса робот едет по полигону и записывает свои команды в массив, а потом по записи возвращается. Договоримся о кодах: 1 – проехать клетку вперёд, 2 – повернуть налево на 90°, 3 – повернуть направо на 90°. Для клеток пригодятся ещё три функции, собранные из тех, что в разделе 6.3: `cell()` – проехать одну клетку и встать, `left90()` и `right90()` – повернуть на месте на четверть оборота и встать.

```

int path[20];           // сюда записываем команды: 1 - клетка вперёд, 2 - налево, 3 - на-
                        // право
int n = 0;              // сколько команд уже записано

void remember(int cmd) {
    path[n] = cmd;      // кладём команду в ящик номер n
    n = n + 1;          // следующая ляжет в соседний ящик
}

void play(int cmd) {
    if (cmd == 1) {
        cell();
    } else if (cmd == 2) {
        left90();
    } else if (cmd == 3) {
        right90();
    }
}

```

Переменная `n` работает как палец, который показывает на первый пустой ящик. В начале `n = 0`. Записали команду в `path[0]` – `n` стало 1, следующая ляжет в `path[1]`. После восьми записей `n = 8`, и занятые ящики – с 0 по 7.

Как вернуться по записи. Сначала робот разворачивается на месте (два раза `right90()`). Потом проходит записанные команды **с конца к началу** и каждый поворот меняет на противоположный. Почему так? Путь назад – это путь туда, пройденный задом наперёд: последний отрезок становится первым. А поворот, который туда был налево, на обратной дороге превращается в поворот направо – проверь на своей дороге до школы.

```

right90();              // разворот на месте
right90();
for (int i = n - 1; i ≥ 0; i--) { // обратно: с конца массива к началу
    if (path[i] == 1) {
        cell();
    } else if (path[i] == 2) {
        right90();          // туда был поворот налево - обратно направо
    } else if (path[i] == 3) {
        left90();
    }
}

```

Цикл идёт в обратную сторону: `i` начинается с `n - 1` (последний занятый ящик), `i--` уменьшает его на 1, а условие `i ≥ 0` не даёт уйти левее нулевого ящика.

Запомни. Индексы массива – с нуля: у массива из `N` ячеек последняя – `[N - 1]`. `d[3]` – четвёртая ячейка. Обратный путь по записи: развернуться, пройти команды с конца и поменять местами «налево» и «направо».

6.6. Разбираем задачи

В задачах этой главы проверить себя можно без робота. Скопируй скетч в Tinkercad или Wokwi, запусти и посмотри в монитор порта. Но на отборе времени на это может не хватить, поэтому сначала реши руками, а симулятор оставь для проверки.

Задача 6.1 Что покажет монитор порта

разминка · переменные, цикл for, целочисленное деление, Serial

Робот каждый круг цикла проезжает 25 см, а программа складывает путь в переменную. Что появится во второй строке монитора порта?

```
void setup() {  
  Serial.begin(9600);  
  int path = 0;  
  for (int i = 0; i < 4; i++) {  
    path = path + 25;  
  }  
  Serial.println(path);  
  Serial.println(path / 3);  
}  
  
void loop() {  
}
```

В ответ введи одно число.

Дано: начальное значение `path = 0`; в теле цикла `path = path + 25`; цикл `for` с `i = 0` и условием `i < 4`.

Найти: второе напечатанное число.

Решение

Шаг 1. Сколько раз выполнится тело цикла. `i` принимает значения 0, 1, 2, 3. При `i = 4` условие `4 < 4` неверно, цикл заканчивается. Значит, тело выполнится 4 раза.

Шаг 2. Следим за коробкой `path`. Было 0. После первого круга $0 + 25 = 25$, после второго 50, после третьего 75, после четвёртого 100. Первая строка монитора: 100.

Шаг 3. Вторая строка. `path / 3` – оба числа целые, поэтому дробная часть отбрасывается. $100 : 3 = 33,33\dots$, остаётся 33. `loop()` пустой, больше ничего не печатается.

Четыре круга по 25 см – это метр, а треть метра – 33 см с хвостиком. Хвостик C++ и выбросил.

❗ **Типичная ошибка.** Ввести 33,3 – забыть, что целое делится на целое без дробной части. Вторая частая ошибка – насчитать 5 кругов («от 0 до 4») и получить 125 и 41.

Ответ: 33.

Своя задача по программе сферы, модуль 1

Реши сам

6.1. Какие два числа напечатает монитор порта? Запиши их через пробел.

```
void setup() {
  Serial.begin(9600);
  int path = 0;
  for (int i = 0; i ≤ 3; i++) {
    path = path + 15;
  }
  Serial.println(path);
  Serial.println(path / 7);
}

void loop() {
}
```

6.2. Программа повышает скорость робота, а потом печатает её в процентах от полной (255). Какие два числа появятся в мониторе порта? Запиши через пробел.

```
void setup() {
  Serial.begin(9600);
  int speed = 90;
  for (int i = 0; i < 3; i++) {
    speed = speed + 40;
  }
  Serial.println(speed);
  Serial.println(speed * 100 / 255);
}

void loop() {
}
```

Задача 6.2 Что нарисует робот

функции моторов, цикл for, delay, путь и время

Робот собран по разделу 6.3, функции моторов оттуда же. При скорости 200 он едет со скоростью 24 см/с, а при развороте на месте за 0,5 с поворачивается на 90°. К корпусу привязан фломастер, который оставляет след на листе. В `setup()` после настройки выводов и скорости стоит такой код:

```
for (int i = 0; i < 4; i++) {
  forward();
  delay(1500);
  turnRight();
  delay(500);
}
stopMotors();
```

`loop()` пустой. Какую фигуру нарисует робот и какой длины будет след? Повороты на месте в длину следа не входят. Ответ в сантиметрах.

Дано: $v = 24$ см/с; вперёд – 1500 мс; поворот направо – 500 мс, это 90° ; цикл на 4 круга.
Найти: фигуру и длину следа L , см.

Решение

Шаг 1. Переводим миллисекунды в секунды. В одной секунде 1000 мс: $1500 \text{ мс} = 1,5 \text{ с}$, $500 \text{ мс} = 0,5 \text{ с}$.

Шаг 2. Один круг цикла. `forward()` включает моторы вперёд, `delay(1500)` даёт роботу ехать 1,5 с. Путь равен скорости, умноженной на время: $24 \cdot 1,5 = 36$ см. Потом `turnRight()` и `delay(500)` – разворот на месте на 90° . За один круг: отрезок 36 см и четверть оборота направо.

Шаг 3. Четыре круга. Четыре одинаковых отрезка, после каждого – поворот на 90° в одну и ту же сторону. Это квадрат со стороной 36 см, и робот возвращается в точку старта. После четвёртого поворота он смотрит туда же, куда в начале: $4 \cdot 90^\circ = 360^\circ$.

Шаг 4. Длина следа. $L = 4 \cdot 36 = 144$ см.

А сколько времени это заняло? Круг длится $1,5 + 0,5 = 2$ с, четыре круга – 8 с. Потом `stopMotors()`, и больше ничего: `setup()` выполняется один раз, а `loop()` пустой.

24 см/с – раз в пять медленнее, чем ты идёшь по улице, так что квадрат со стороной в треть метра за восемь секунд выглядит правдоподобно. Такой поместится на листе ватмана.

⚠ Типичная ошибка. Посчитать путь за всё время: $24 \cdot 8 = 192$ см. Но две секунды из восьми робот крутится на месте, и след в это время не удлиняется. Ещё бывает, что код из `setup()` мысленно переносят в `loop()` и отвечают «рисует квадраты без конца».

Ответ: квадрат со стороной 36 см, длина следа 144 см.

Своя задача по программе сферы, модуль 2

Реши сам

6.3. Тот же робот, но при скорости 150 он едет со скоростью 15 см/с, а `turnLeft()` за 600 мс разворачивает его на месте на 120° . В `setup()`:

```
for (int i = 0; i < 3; i++) {  
  forward();  
  delay(2000);  
  turnLeft();  
  delay(600);  
}  
stopMotors();
```

Какую фигуру нарисует робот? Какой длины будет след в сантиметрах?

6.4. Робот едет со скоростью 20 см/с, `turnRight()` за 500 мс поворачивает его на 90°. На этот раз команды стоят в `loop()` :

```
void loop() {  
  forward();  
  delay(2000);  
  turnRight();  
  delay(500);  
}
```

Сколько сантиметров проедет робот за первую минуту после включения? Где он окажется через 60 с после включения?

Задача 6.3 Найди ошибку: робот не считает перекрёстки

поиск ошибки, переменная внутри loop, счётчик, датчик линии

Робот едет прямо по белому полю. Поперёк его пути нарисованы чёрные полосы – перекрёстки. Датчик линии подключён к А0: над чёрным он даёт меньше 400, над белым – больше. Робот должен остановиться на третьей полосе. Функции `forward()` и `stopMotors()` – из раздела 6.3, настройка выводов для краткости не показана.

```
1 void setup() {  
2   Serial.begin(9600);  
3 }  
4  
5 void loop() {  
6   int count = 0;  
7   forward();  
8   if (analogRead(A0) < 400) {  
9     count = count + 1;  
10    Serial.println(count);  
11    delay(300);  
12  }  
13  if (count == 3) {  
14    stopMotors();  
15    while (true) {}  
16  }  
17 }
```

`delay(300)` нужен, чтобы робот успел съехать с полосы и не посчитал её дважды. Робот проезжает все полосы и не останавливается, а монитор порта показывает: 1, 1, 1, 1. В какой строке ошибка? В ответ введи номер строки.

Дано: скетч из 17 строк; в мониторе порта одни единицы.

Найти: номер строки с ошибкой.

Решение

Шаг 1. Что должно быть. Счётчик `count` на каждой полосе увеличивается на 1: 1, 2, 3. На третьей полосе условие `count == 3` верно, робот останавливается, а `while (true) {}` держит его

на месте до выключения.

Шаг 2. Прослеживаем, что получилось. Первый круг `loop()` : строка 6 создаёт `count` и кладёт 0. Робот едет, под ним белое – в `if` не заходим, и круг кончается. Второй круг: строка 6 **снова** создаёт `count` и снова кладёт 0. Так на каждом круге. Вот робот на полосе: `count` становится $0 + 1 = 1$, печатается 1. Но на следующем круге строка 6 опять делает 0. Когда робот доедет до второй полосы, `count` снова станет только 1.

Шаг 3. Первое расхождение. Единицы в мониторе подтверждают: к строке 9 `count` каждый раз приходит с нулём. Виновата строка 6: переменная объявлена внутри `loop()` и живёт один круг (раздел 6.1). До 3 она не дорастёт никогда.

Шаг 4. Исправляем. Переносим `int count = 0;` в начало скетча, выше `void setup()` . Такая переменная создаётся один раз при включении и помнит своё значение между кругами. Теперь монитор покажет 1, 2, 3, и на третьей полосе робот встанет.

 **Типичная ошибка.** Искать ошибку в строке 13 и «чинить» её на `count ≥ 3` или `count == 1` . Условие здесь ни при чём: оно верное, просто `count` никогда не бывает больше единицы. Подсказку дал монитор порта – всегда смотри, что печатает программа, прежде чем менять условия.

Ответ: 6.

Своя задача по программе сферы, модуль 2

Реши сам

6.5. Что напечатает монитор порта третьей строкой? А что – если перенести строку `int k = 5;` выше `void setup()` ? Запиши два числа через пробел.

```
void setup() {  
  Serial.begin(9600);  
}  
  
void loop() {  
  int k = 5;  
  k = k * 2;  
  Serial.println(k);  
}
```

6.6. Робот должен остановиться, когда до стены меньше 20 см. Он едет к стене, датчик каждые 0,1 с показывает 31, 26, 21, 17, 12, 8, 4 см – и робот врезается. В какой строке ошибка и как её исправить?

```
1 void loop() {
2   long d = distanceCm();
3   if (d == 20) {
4     stopMotors();
5   } else {
6     forward();
7   }
8   delay(100);
9 }
```

Задача 6.4 Расставь строки

порядок строк, if – else, analogRead, Serial

Нужно заполнить `loop()`. Каждые 0,2 с робот измеряет датчик линии на входе A0 и пишет в монитор порта «линия», если число меньше 400, и «поле» во всех остальных случаях. Сначала измеряет, потом пишет, потом ждёт. Строки перемешаны:

```
1 Serial.println("поле");
2 if (n < 400) {
3   int n = analogRead(A0);
4   Serial.println("линия");
5 } else {
6 }
7 delay(200);
```

В каком порядке их поставить? Ответ – семь цифр подряд.

Дано: семь строк; порядок действий «измерить – написать – подождать».

Найти: порядок номеров.

Решение

Шаг 1. Первая строка – измерение. Сначала робот измеряет, и строка 2 проверяет `n` – значит, `n` уже должна существовать. Первой идёт строка 3: она создаёт `n` и кладёт в неё показание датчика.

Шаг 2. Ветвление. Проверка – строка 2: `if (n < 400) {`. Сразу после неё – то, что делаем при верном условии: число меньше 400 – это линия, строка 4.

Шаг 3. Ветка «иначе». Строка 5 `} else {` закрывает первую ветку и открывает вторую. В ней – строка 1, «поле». Строка 6 `}` закрывает вторую ветку.

Шаг 4. Ожидание. Последней – строка 7, `delay(200)`: 200 мс – это 0,2 с.

Получается 3, 2, 4, 5, 1, 6, 7. Проверим фигурные скобки: открываются в строках 2 и 5, закрываются в строках 5 и 6 – каждой открывающей нашлась своя закрывающая. Если перепутать 1 и 4, программа тоже соберётся, но будет писать «поле» над линией.

❗ **Типичная ошибка.** Поставить строку 3 внутрь ветвления или после `delay`: «сначала пождём, потом измерим». В первом случае `if` проверяет переменную, которой ещё нет, – программа не соберётся. Во втором соберётся, но нарушен порядок из условия. Ещё путают ветки: над чёрной линией число **меньше** порога (глава 5).

Ответ: 3245167.

Своя задача по программе сферы, модуль 2

Реши сам

6.7. Светодиод на выводе 13 должен мигнуть три раза: гореть 0,5 с, потом гаснуть на 0,2 с. Расставь строки внутри `setup()` и запиши номера подряд.

```
1 delay(200);
2 digitalWrite(13, HIGH);
3 }
4 for (int i = 0; i < 3; i++) {
5   digitalWrite(13, LOW);
6   delay(500);
```

6.8. Робот должен ехать вперёд, пока до стены больше 25 см, потом остановиться и напечатать, сколько сантиметров осталось до стены. Расставь строки и запиши номера подряд.

```
1 stopMotors();
2 while (distanceCm() > 25) {
3   Serial.println(distanceCm());
4   forward();
5 }
```

Задача 6.5 Проследи переходы

машина состояний, трассировка, *millis*, датномер

Робот объезжает препятствия по схеме на рисунке. Состояния пронумерованы: 1 – ВПЕРЁД, 2 – НАЗАД, 3 – ПОВОРОТ.

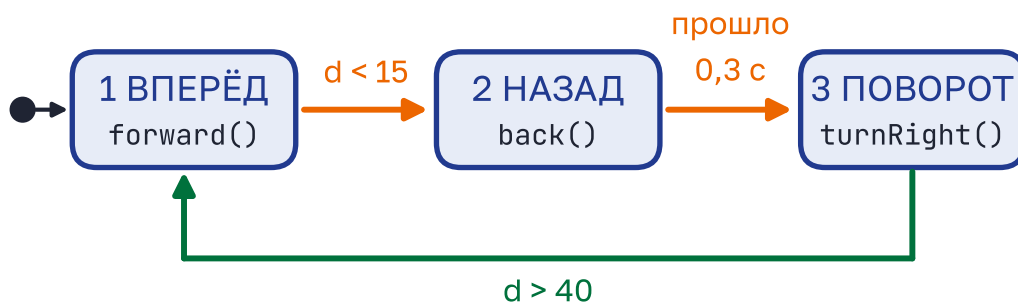


Рисунок 6.9 – Машина состояний к задаче 6.5

Вот её код. Круг `loop()` длится 100 мс, первый круг начинается в момент 0.

```
const int GO = 1, BACK = 2, TURN = 3;
int state = GO;
unsigned long t0 = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  long d = distanceCm();
  if (state == GO) {
    forward();
    if (d < 15) {
      state = BACK;
      t0 = millis();
    }
  } else if (state == BACK) {
    back();
    if (millis() - t0 ≥ 300) {
      state = TURN;
    }
  } else if (state == TURN) {
    turnRight();
    if (d > 40) {
      state = GO;
    }
  }
  Serial.print(state);
  delay(100);
}
```

На первых десяти кругах дальномер показал по порядку: 50, 32, 18, 12, 16, 22, 27, 35, 48, 44 см. Что напечатает монитор порта за эти десять кругов? `Serial.print` не переводит строку, поэтому цифры идут подряд. Ответ – десять цифр.

Дано: схема и код; 10 показаний дальномера по одному на круг; круг длится 100 мс.

Найти: десять цифр в мониторе порта.

Решение

Шаг 1. Что печатается. `Serial.print(state)` стоит в конце круга, после всех проверок. Значит, на каждом круге печатается состояние, в котором робот **закончил** этот круг.

Шаг 2. Когда засекают время. Круг номер k начинается в момент $(k - 1) \cdot 100$ мс: первый – в 0, четвёртый – в 300 мс. Поэтому `millis()` на круге 4 равно 300.

Шаг 3. Трассируем. Условия проверяем только у стрелок, выходящих из текущего состояния.

Таблица 6.1

Круг	Время, мс	d, см	Было	Стало
1	0	50	1	1
2	100	32	1	1

Круг	Время, мс	d, см	Было	Стало
3	200	18	1	1
4	300	12	1	2
5	400	16	2	2
6	500	22	2	2
7	600	27	2	3
8	700	35	3	3
9	800	48	3	1
10	900	44	1	1

Пояснения к строкам, где что-то происходит:

- Круг 3: 18 не меньше 15, робот остаётся в 1.
- Круг 4: $12 < 15$ – переход в 2, $t_0 = 300$.
- Круги 5 и 6: в состоянии 2 на расстояние не смотрят. Прошло $400 - 300 = 100$ и $500 - 300 = 200$ мс – меньше 300, остаёмся в 2.
- Круг 7: $600 - 300 = 300$ мс, условие ≥ 300 верно – переход в 3.
- Круг 8: 35 не больше 40, продолжаем поворачивать.
- Круг 9: $48 > 40$ – переход в 1.
- Круг 10: 44 не меньше 15, едем дальше.

Шаг 4. Собираем ответ из колонки «Стало»: 1, 1, 1, 2, 2, 2, 3, 3, 1, 1.

Робот отъезжал три круга – это 0,3 с, как и нарисовано на схеме, а повернул, как только впереди стало свободно.

 **Типичная ошибка.** На круге 5 увидеть 16 и решить, что робот вернулся в ВПЕРЁД: «16 уже не меньше 15». Но из состояния НАЗАД стрелка только одна – по времени, дальномер здесь не проверяется. Вторая ошибка – отсчитывать 300 мс от включения и переходить в ПОВОРОТ уже на круге 4.

Ответ: 1112223311.

Своя задача по программе сферы, модуль 1

Реши сам

6.9. Тот же робот и тот же код. За двенадцать кругов дальномер показал: 36, 22, 14, 9, 11, 17, 26, 33, 39, 52, 30, 12 см. Что напечатает монитор порта? Ответ – двенадцать цифр.

6.10. Светофор – машина из трёх состояний, переходы только по времени: красный горит 3 с, потом зелёный 4 с, потом жёлтый 1 с, потом снова красный, и так по кругу. В момент включения загорается красный. Какой свет будет гореть через 50 с после включения? Сколько раз за первую минуту загорится зелёный?

Задача 6.6 Дорога домой

массивы, обратный порядок, повороты, поле из клеток

Робот стоит на базе в клетке (1; 1) и смотрит вверх. Клетку записываем как в главе 4: сначала столбец, потом строка. Он проехал по полю и записал свои команды в массив `path`, как в разделе 6.5: 1 – клетка вперёд, 2 – налево на 90°, 3 – направо на 90°.



Рисунок 6.10 – Путь робота от базы и записанный массив

```
int path[8] = {1, 1, 3, 1, 1, 1, 2, 1};
```

Теперь робот разворачивается на месте (`right90()` два раза) и едет на базу по записи. Какие восемь команд он выполнит после разворота? Запиши их коды подряд, восемь цифр. Куда он будет смотреть, когда приедет?

Дано: `path = {1, 1, 3, 1, 1, 1, 2, 1}`; старт (1; 1), смотрит вверх.

Найти: коды обратного пути; направление в конце.

Решение

Шаг 1. Где робот сейчас. Проследим запись. Вверх две клетки: (1; 2), (1; 3). Направо – смотрит вправо. Три клетки: (2; 3), (3; 3), (4; 3). Налево – смотрит вверх. Клетка вперёд: (4; 4). Робот в (4; 4), смотрит вверх, как на рисунке.

Шаг 2. Разворот. Два поворота направо по 90° – это 180°: теперь робот смотрит вниз.

Шаг 3. Команды с конца массива. Читаем `path` справа налево, от `path[7]` до `path[0]`: 1, 2, 1, 1, 1, 3, 1, 1.

Шаг 4. Меняем повороты. 2 (налево) становится 3 (направо), а 3 становится 2. Клетки остаются клетками: 1, 3, 1, 1, 1, 2, 1, 1.

Шаг 5. Проверяем на поле. Из (4; 4) вниз: клетка – (4; 3). Направо – смотрящий вниз робот поворачивается к левой стороне листа. Три клетки: (3; 3), (2; 3), (1; 3). Налево – смотрит вниз. Две клетки: (1; 2), (1; 1). Приехали на базу, робот смотрит вниз.

Робот вернулся тем же коридором, каким уехал. Уезжал он носом вверх, а на базе стоит носом вниз: в начале обратного пути он развернулся, и больше его никто не разворачивал.

❗ **Типичная ошибка.** Прочитать массив с конца, но забыть поменять повороты: 1, 2, 1, 1, 1, 3, 1, 1. Робот в (4; 3) повернёт налево – к правому краю поля – и уедет от базы. Другая ошибка – поменять повороты, но ехать по массиву с начала.

Ответ: 13111211; робот смотрит вниз.

Своя задача по программе сферы, модуль 10

Реши сам

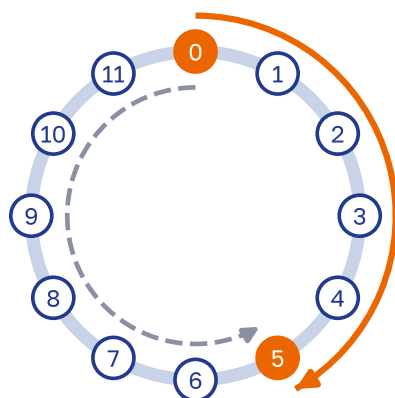
6.11. Массив `int d[5] = {34, 28, 41, 17, 23};`. Что напечатают строки `Serial.println(d[3]);` и `Serial.println(d[1] + d[4]);`? Запиши два числа через пробел.

6.12. Робот стартовал из клетки (2; 1), смотря вверх, и записал `path = {1, 1, 2, 1, 3, 1, 1, 3, 1}` (коды те же). В какой клетке он оказался? После разворота на месте он едет домой по записи. Запиши девять цифр обратного пути.

Задача 6.7 *. Кольцевая трасса

массивы, остаток от деления, цикл, кратчайшее направление по кругу

Робот-почтальон ездит по кольцевой дорожке вокруг клумбы. На дорожке 12 остановок, как цифры на циферблате, с номерами от 0 до 11 по часовой стрелке. Переезд до соседней остановки называют перегонем. Робот может ехать и по часовой стрелке, и против. Список остановок программа считает по формуле, а между двумя остановками выбирает направление, где перегонов меньше.



по часовой стрелке:
5 перегонов

против часовой:
 $12 - 5 = 7$ перегонов

Рисунок 6.11 – Кольцевая дорожка на 12 остановок. От 0 до 5 можно доехать двумя путями

```

int stops[7]; // номера остановок по порядку

void setup() {
  Serial.begin(9600);
  stops[0] = 0; // старт - остановка 0
  for (int i = 1; i < 7; i++) {
    stops[i] = (stops[i - 1] * 7 + 5) % 12;
  }
  int total = 0; // всего перегонов
  for (int i = 1; i < 7; i++) {
    int f = (stops[i] - stops[i - 1] + 12) % 12; // по часовой стрелке
    int b = 12 - f; // против часовой
    if (f < b) {
      total = total + f;
    } else {
      total = total + b;
    }
  }
  Serial.println(total);
}

void loop() {
}

```

Что напечатает монитор порта? В ответ введи одно число.

Дано: 12 остановок по кругу; `stops[0] = 0`; каждая следующая – $(\text{предыдущая} \cdot 7 + 5) \% 12$; 6 переездов.

Найти: `total` – сколько всего перегонов.

Решение

Шаг 1. Заполняем массив. Первый цикл считает `stops[1] ... stops[6]`. Знак `%` – остаток от деления на 12.

- `stops[1] = (0 · 7 + 5) % 12 = 5 % 12 = 5;`
- `stops[2] = (5 · 7 + 5) % 12 = 40 % 12 = 4`, потому что $40 = 3 \cdot 12 + 4$;
- `stops[3] = (4 · 7 + 5) % 12 = 33 % 12 = 9;`
- `stops[4] = (9 · 7 + 5) % 12 = 68 % 12 = 8;`
- `stops[5] = (8 · 7 + 5) % 12 = 61 % 12 = 1;`
- `stops[6] = (1 · 7 + 5) % 12 = 12 % 12 = 0.`

Маршрут: $0 \rightarrow 5 \rightarrow 4 \rightarrow 9 \rightarrow 8 \rightarrow 1 \rightarrow 0$. Робот вернулся туда, откуда начал.

Шаг 2. Что такое `f` и `b`. `f` – сколько перегонов по часовой стрелке: из номера «куда» вычитаем номер «откуда». Посмотри на обычные часы. От 2 часов до 5 часов стрелка проходит $5 - 2 = 3$ часа. А от 11 до 2? Вычитание даёт $2 - 11 = -9$, но стрелка прошла 3 часа: через 12 перешла. Отрицательное число значит, что по пути проехали через 0, и к нему надо прибавить 12: $-9 + 12 = 3$. Это и делает `+ 12`. А `% 12` убирает лишние 12, если разность и так была не меньше нуля: от 2 до 5 выйдет $(3 + 12) \% 12 = 15 \% 12 = 3$. Против часовой – остаток круга: `b = 12 - f`. Проверим на рисунке: от 0 до 5 по часовой 5 перегонов, против – 7.

Шаг 3. Считаем каждый переезд. Выбираем меньшее из `f` и `b`.

Таблица 6.2

Откуда	Куда	f	b	Взяли
0	5	5	7	5
5	4	11	1	1
4	9	5	7	5
9	8	11	1	1
8	1	5	7	5
1	0	11	1	1

Посчитаем для примера $5 \rightarrow 4$: $(4 - 5 + 12) \% 12 = 11 \% 12 = 11$. По часовой – почти целый круг, а против – всего один перегон, соседняя остановка.

Шаг 4. Сумма. $5 + 1 + 5 + 1 + 5 + 1 = 18$.

Если бы робот всегда ехал по часовой, вышло бы $5 + 11 + 5 + 11 + 5 + 11 = 48$ перегонов – четыре круга. Выбор направления экономит больше половины пути.

 **Типичная ошибка.** Выкинуть $+ 12$ из формулы «чтобы проще». В C++ остаток от отрицательного числа – тоже отрицательный: $(4 - 5) \% 12$ даёт -1 , а не 11 . Тогда $f = -1$, $b = 13$, условие $f < b$ верно, и к `total` прибавится -1 . Ещё ошибаются в остатке от больших чисел: $68 \% 12$ – это 8 ($68 = 5 \cdot 12 + 8$), а не 5 или 6 .

Ответ: 18.

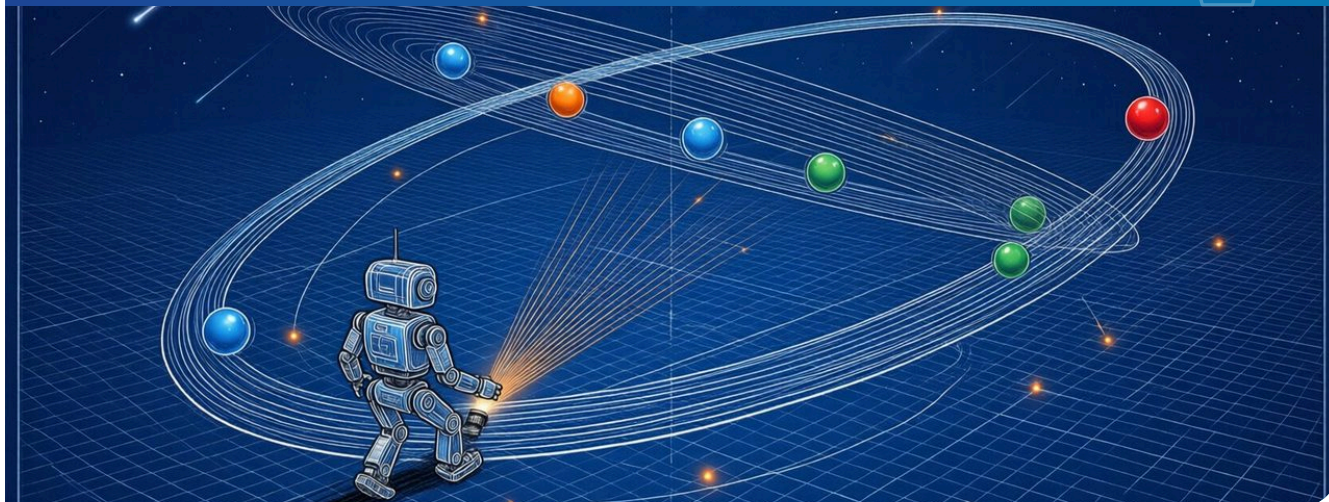
По мотивам задачи первого отборочного этапа НТО 2024/25, профиль «Интеллектуальные робототехнические системы», предметный тур по математике, задача 2.3.2.5 о гиперпространственной навигации

Реши сам

6.13. Кольцо на 10 остановок (от 0 до 9), код тот же, только вместо 12 везде стоит 10, а маршрут записан сразу: `int stops[6] = {0, 3, 8, 2, 7, 0};` (5 переездов). Сколько перегонов считает программа? Обрати внимание, что будет, когда `f` и `b` равны.

6.14.* Кольцо на 12 остановок. Маршрут начинается с остановки 1, каждая следующая – (предыдущая $\cdot 5 + 3) \% 12$, всего 4 переезда (массив `stops[5]`). Какое число окажется в `stops[4]`? Сколько перегонов проедет робот, если каждый раз выбирает короткое направление? Запиши два числа через пробел.

Движение по линии и регуляторы



Чёрная линия на белом поле – самая старая задача робототехники и одна из самых честных: робот либо едет по ней, либо нет. Видит линию робот двумя датчиками, а куда рулить, решает **регулятор** – короткое правило в программе. Их здесь два, «зиг-заг» и плавный пропорциональный, а в конце – зачем к нему приписывают ещё две буквы и получают ПИД. На отборе робота на столе не будет: по замерам ищут порог, по показаниям датчиков – куда он повернёт, по формуле – поправку и скорости моторов. Ответ – одно число или строка из цифр.

7.1. Как датчик видит линию

НА ПАЛЬЦАХ



Рисунок 7.1 – Пятно света датчика съезжает с белого поля на чёрную линию

Датчик смотрит на пятнышко поля размером в несколько миллиметров. Чем больше в этом пятне чёрного, тем меньше число. Половина пятна на линии – половина пути от «белого» числа к «чёрному».

Как устроен датчик линии, мы разобрали в разделе 5.7: светодиод светит вниз невидимым инфракрасным светом, приёмник ловит отражение, а `analogRead` превращает его в число от 0 до 1023. Белая бумага свет возвращает, чёрная изолянта его поглощает. Над белым полем у нашего датчика получается около 870, над чёрной линией – около 110.

В этой главе, если в задаче не сказано другое, **белое – большое число, чёрное – маленькое**. Так выходит у фоторезистора в Tinkercad, которым там изображают датчик линии, и у многих настоящих датчиков. Бывают датчики и наоборот – тогда все правила ниже работают так же, только «больше» и «меньше» меняются местами. Какой у тебя случай, покажут первые же замеры.

В пятой главе мы это проскочили, а здесь без этого никуда: датчик не видит точку. Он видит пятно, и число зависит от того, какая доля пятна лежит на чёрном. Когда датчик съезжает с белого на линию, показание не прыгает с 870 на 110 в один миг, а сползает плавно: 870, 800, 650, 490, 300, 110. Когда половина пятна на линии, датчик показывает середину: $(870 + 110) : 2 = 490$. Значит, по числу можно понять не только «линия или нет», но и **насколько** датчик заехал на линию. На этом держится весь пропорциональный регулятор из раздела 7.4.

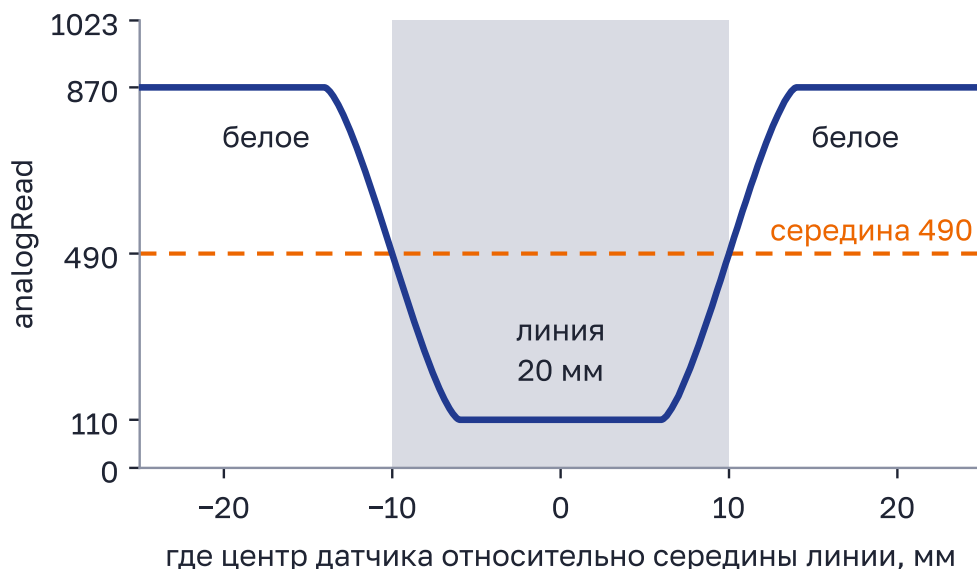


Рисунок 7.2 – Показание датчика, который медленно везут поперёк линии шириной 20 мм. На краях линии число меняется плавно. Пунктир – середина между белым и чёрным

Сколько датчиков ставить? С одним можно ехать вдоль **края** линии, но тогда робот не знает, с какой она стороны, если вдруг съехал целиком на белое. Поэтому в этой главе у робота два датчика: левый и правый, по бокам от линии. Расстояние между ними чуть больше ширины линии. Полоска чёрной изолянт — это около 1,5–2 см, значит, датчики ставят сантиметрах в 2–3 друг от друга. Пока робот едет точно над линией, она проходит между датчиками, и оба видят белое. Уехал робот вправо – линия оказалась под левым датчиком, и его число упало. Уехал влево – упало у правого.

Запомни. Датчик видит пятно, а не точку, поэтому его число меняется плавно. Два датчика стоят по бокам от линии, на прямой оба видят белое. Какой датчик «потемнел», с той стороны и линия.

7.2. Калибровка: где провести границу

НА ПАЛЬЦАХ

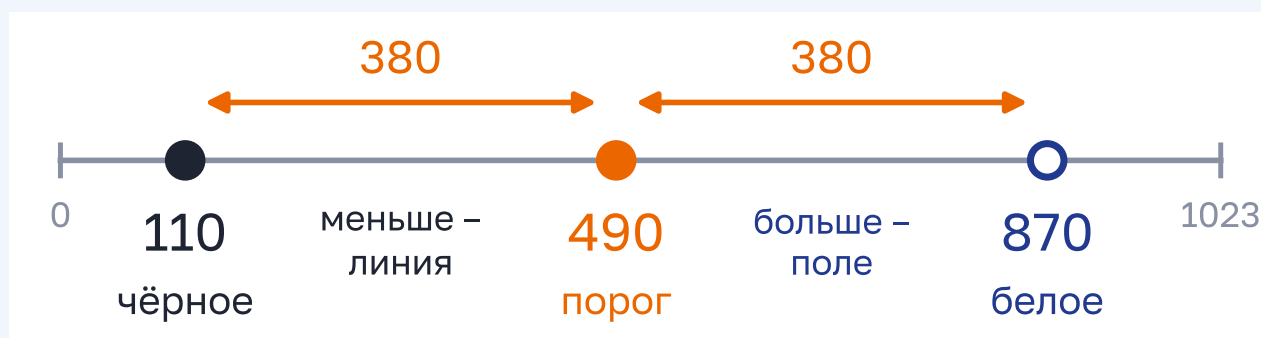


Рисунок 7.3 – Порог – посередине между белым и чёрным

От порога до «белого» и до «чёрного» числа одинаково далеко. Поэтому датчик ошибётся, только если число уплывёт больше чем на половину разницы.

Калибровка – настройка датчика под то место, где он будет работать: замеряем показания над белым и над чёрным и по ним считаем порог. **Порог** – число-граница: меньше порога – под датчиком линия, больше – поле. Считается он как в главе 5, средним:

$$\text{порог} = \frac{\text{белое} + \text{чёрное}}{2}.$$

Почему середина? Показания никогда не стоят на месте. Чуть подпрыгнул робот на стыке листов, солнце выглянуло из-за тучи, ребята включили свет в классе – и число сдвинулось на 20, 50, 100. Порог посередине даёт одинаковый запас в обе стороны. Поставишь его близко к «белому» числу – робот начнёт видеть линию на чуть сероватом поле, близко к «чёрному» – перестанет замечать линию, когда она чуть блестит.

Замеров – несколько. Одно показание может случайно оказаться неудачным. Поэтому над белым меряют несколько раз в разных местах поля, над чёрным – несколько раз на разных участках линии, а потом берут среднее. Пример: над белым левый датчик показал 906, 914 и 910. Среднее: $(906 + 914 + 910) : 3 = 2730 : 3 = 910$. Над линией – 128, 132 и 130, среднее $390 : 3 = 130$. Порог левого датчика $(910 + 130) : 2 = 520$.

Каждому датчику – свой порог. Два датчика из одной коробки всё равно немного разные, да и светит на них по-разному. У правого датчика того же робота над белым вышло 766, 774, 770 (среднее 770), над чёрным 246, 254, 250 (среднее 250). Его порог $(770 + 250) : 2 = 510$. Если взять для обоих один порог «на глаз», один из датчиков начнёт ошибаться у самого края линии.

Калибровка на месте. Замеры делают на том поле, где поедет робот, и при том свете, что будет на заезде. Дома на белом ламинате и на соревнованиях на баннере числа будут разные. На соревнованиях команды калибруют робота прямо перед попыткой. Часто это делает сама программа: первые секунды после включения робота возят руками над линией туда-сюда, а он запоминает самое большое и самое маленькое число.

```
int minL = 1023;    // самое маленькое показание – сначала «хуже некуда»
int maxL = 0;       // самое большое показание

void setup() {
  Serial.begin(9600);
  for (int i = 0; i < 300; i++) {    // 300 замеров по 10 мс – это 3 секунды
    int n = analogRead(A0);          // левый датчик
    if (n < minL) minL = n;           // нашли число меньше – запомнили
    if (n > maxL) maxL = n;           // нашли число больше – запомнили
    delay(10);
  }
  int porogL = (minL + maxL) / 2;    // порог – середина
  Serial.println(porogL);
}

void loop() {
}
```

Разберём по строкам. `int minL = 1023;` – заводим переменную для самого маленького числа и сразу кладём в неё самое большое, какое может выдать `analogRead`. Любой первый же замер будет меньше, и `minL` его запомнит. С `maxL` наоборот: начинаем с нуля. Цикл `for` повторяет замер 300 раз с паузой 10 мс – 3 секунды, за которые робота успевают провести над ли-

нией. В конце $(\min L + \max L) / 2$ – середина. Дробную часть при делении целых чисел Arduino отбрасывает: из $1025 / 2$ получится 512.

Когда датчики совсем разные. Бывает, что у одного датчика белое 900 и чёрное 100, а у другого 700 и 300. Решать «линия или нет» это не мешает: у каждого свой порог. А вот сравнивать их числа между собой (а это понадобится в разделе 7.4) уже нельзя. Для первого датчика 700 – ещё не совсем белое, для второго – самое белое, какое у него бывает. Тогда показание переводят в проценты: где оно находится на пути от чёрного к белому.

$$\text{белизна} = \frac{\text{показание} - \text{чёрное}}{\text{белое} - \text{чёрное}} \cdot 100 \, \%$$

Словами: сколько число поднялось над «чёрным», делим на весь путь от «чёрного» до «белого». Над линией выходит 0 %, над полем 100 %, у края – около 50 %. Первый датчик показал 700: $(700 - 100) : (900 - 100) \cdot 100 = 600 : 800 \cdot 100 = 75 \, \%$. Второй показал 600: $(600 - 300) : (700 - 300) \cdot 100 = 300 : 400 \cdot 100 = 75 \, \%$. Числа разные, а видят датчики одно и то же: у обоих пятно на четверть заехало на линию.

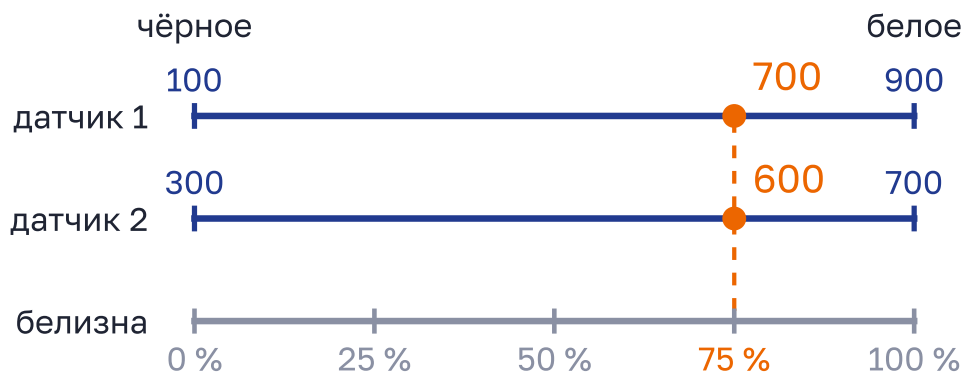


Рисунок 7.4 – Шкалы двух разных датчиков, пересчитанные в проценты. Числа 700 и 600 оказываются на одной отметке – 75 %

Запомни. Порог = $(\text{белое} + \text{чёрное}) : 2$, и у каждого датчика он свой. Белое и чёрное – средние по нескольким замерам, на том поле и при том свете, где поедет робот. Разные датчики сравнивают в процентах белизны.

Совет. Перед тем как считать порог, проверь, какое число у линии: маленькое или большое. Если у твоего датчика чёрное – большое число, то и условие в программе переворачивается: линия – это «больше порога».

7.3. Релейный регулятор: зигзаг по двум датчикам

НА ПАЛЬЦАХ



Рисунок 7.5 – Четыре случая: оба датчика на белом, линия под левым, линия под правым, оба на чёрном

Робот поворачивает в ту сторону, где датчик увидел линию, – ловит её обратно между датчиками. А если потемнели оба, под ним поперечная полоса.

Регулятор – правило, по которому робот сам поправляет свой курс, глядя на датчики. Самый простой – **релейный**. Название от реле: это выключатель, у которого два положения – «включено» и «выключено», промежуточных нет. Так и наш регулятор: он либо едет прямо, либо поворачивает на одну и ту же величину. Повернуть «чуть-чуть» он не умеет.

Поворачивает робот разницей скоростей колёс (глава 1): **робот уходит в сторону медленного колеса**. Хочешь налево – левое колесо медленнее правого. Теперь правило для двух датчиков. Сначала каждый датчик сравниваем со своим порогом и получаем «чёрное» или «белое», а затем смотрим, какой из случаев вышел.

- **Оба на белом**. Линия между датчиками – едем прямо, колёса одинаково.
- **Левый на чёрном, правый на белом**. Робот уехал вправо, линия ушла под левый датчик. Поворачиваем налево: левое колесо медленнее.
- **Левый на белом, правый на чёрном**. Робот уехал влево. Поворачиваем направо: правое колесо медленнее.

Есть и четвёртый случай – **оба на чёрном**. На обычном участке так не бывает: линия уже, чем расстояние между датчиками. Значит, под роботом поперечная линия – **перекрёсток**. Что делать на перекрёстке, решает задача: остановиться, посчитать его, повернуть.

Если робот не встаёт на перекрёстке, а едет дальше, проезжает он его не за один миг. Полосу шириной 2 см на скорости 30 см/с он переезжает за $2 : 30 \approx 0,07$ с, а программа за это время успевает прочитать датчики раз шесть. Шесть раз подряд «оба на чёрном» – это один перекрёсток, а не шесть. Считать надо тот миг, когда «оба на чёрном» **появилось** после любого другого случая.

Таблица 7.1

Левый датчик	Правый датчик	Что делает робот	Скорости: левый и правый
белое	белое	едет прямо	150 и 150
чёрное	белое	поворачивает налево	60 и 180
белое	чёрное	поворачивает направо	180 и 60
чёрное	чёрное	перекрёсток – стоп	0 и 0

В программе это цепочка «если – иначе если». `drive(левый, правый)` – подпрограмма из справочника в конце книги, она задаёт скорости двух моторов (от –255 до 255, минус – назад).

```
const int POROG_L = 520; // порог левого датчика (из калибровки)
const int POROG_R = 510; // порог правого датчика

void loop() {
  int left = analogRead(A0); // левый датчик
  int right = analogRead(A1); // правый датчик
  bool lineL = left < POROG_L; // true, если под левым датчиком линия
  bool lineR = right < POROG_R; // true, если под правым линия
  if (lineL && lineR) {
    drive(0, 0); // оба на чёрном: перекрёсток – стоп
  } else if (lineL) {
    drive(60, 180); // линия слева – поворачиваем налево
  } else if (lineR) {
    drive(180, 60); // линия справа – поворачиваем направо
  } else {
    drive(150, 150); // оба на белом – прямо
  }
  delay(10);
}
```

Напомним из главы 6: в переменной `bool` лежит «да» (`true`) или «нет» (`false`), а `&&` значит «и». Условие `lineL && lineR` верно, только когда верны обе части. Порядок проверок важен: перекрёсток проверяем первым. Иначе на перекрёстке сработает строчка `else if (lineL)` – ведь левый датчик тоже на чёрном, – и робот свернёт налево.

Как едет такой робот? Он всё время чуть-чуть промахивается: датчик задел линию – резкий поворот, линия ушла между датчиками – снова прямо, но нос уже смотрит вбок, и вскоре линию задевает второй датчик. Получается **зигзаг**. На прямой это терпимо, на скорости – робот вылетает.

С **одним** датчиком тоже ездят, только вдоль края линии: видит чёрное – подворачивает в одну сторону, видит белое – в другую. Робот идёт по границе чёрного и белого, как по канату, и всё время перескакивает её то туда, то обратно.

Запомни. Робот поворачивает в сторону медленного колеса и в сторону датчика, который увидел линию. Оба на белом – прямо, оба на чёрном – перекрёсток. Перекрёсток в программе проверяют первым, а считают по началу серии «оба на чёрном».

Совет. В задаче «куда повернёт робот» сначала выпиши для каждого датчика одно слово – «чёрное» или «белое», – и только потом смотри в таблицу случаев. Когда в голове сразу и числа, и пороги, и стороны, путаница почти гарантирована.

7.4. Пропорциональный регулятор

НА ПАЛЬЦАХ

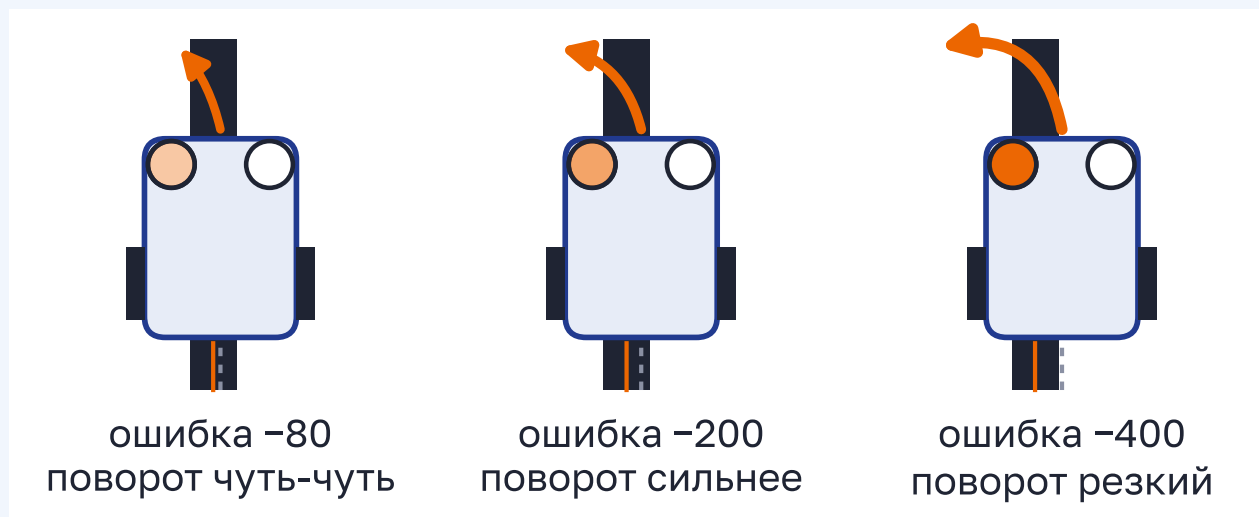


Рисунок 7.6 – Чем дальше робот ушёл от линии, тем сильнее он поворачивает обратно

Как ты сам ведёшь велосипед по тропинке: съехал чуть-чуть – чуть повернул руль, съехал далеко – повернул сильнее.

Релейный регулятор знает только «налево» и «направо». Хочется, чтобы робот поворачивал **тем сильнее, чем дальше он ушёл от линии**. Такой регулятор называют **пропорциональным**, или П-регулятором: поправка пропорциональна ошибке, то есть во сколько раз больше ошибка, во столько раз сильнее поворот.

- **Ошибка** – насколько робот ушёл от линии. У двух датчиков её считают как разницу: ошибка = левый – правый. Робот стоит посередине – оба датчика видят одинаково, ошибка 0. Левый заехал на линию и потемнел – ошибка меньше нуля. Потемнел правый – больше нуля.
- **Коэффициент k** – число, которое говорит, насколько резко робот отвечает на ошибку. Его подбирают опытом.
- **Поправка** – на сколько изменить скорости колёс: поправка = $k \cdot$ ошибка.

Поправку одному колесу прибавляем, у другого вычитаем. Скорость, с которой робот едет по прямой, назовём **базой**.

$$\text{левый мотор} = \text{база} + \text{поправка}, \quad \text{правый мотор} = \text{база} - \text{поправка}.$$

Проверим, что робот поворачивает куда надо. Линия под левым датчиком – ошибка отрицательная, поправка тоже. Левому мотору прибавляем отрицательное число, то есть он замедляется, а правый ускоряется. Робот уходит в сторону медленного левого колеса – налево, к линии. Потемнел правый – всё зеркально.

Вот как это выглядит в числах при базе 150 и $k = 0,25$:

Таблица 7.2

Ошибка	Поправка	Левый мотор	Правый мотор	Что делает робот
-400	-100	50	250	резко налево
-200	-50	100	200	налево
-80	-20	130	170	чуть налево
0	0	150	150	прямо
80	20	170	130	чуть направо
200	50	200	100	направо
400	100	250	50	резко направо

Ошибка выросла вдвое – вдвое выросла и поправка. Посередине таблицы, у линии, робот почти не крутит руль, поэтому и не виляет, как релейный.

Границы. Значение ШИМ не может быть больше 255 (глава 5). А если поправка получилась больше базы, у медленного мотора выйдет отрицательное число. В самой простой программе такие числа **обрезают**: всё, что больше 255, становится 255, всё, что меньше 0, – нулём. В Arduino это делает `constrain(x, 0, 255)` – «зажми x между 0 и 255».

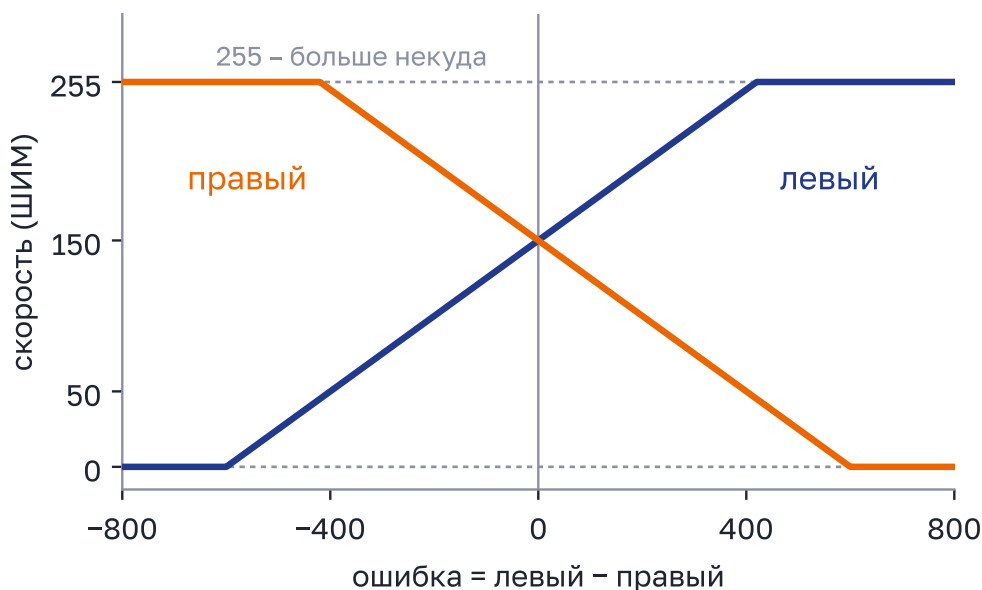


Рисунок 7.7 – Скорости моторов в зависимости от ошибки: база 150, $k = 0,25$. Где линия упирается в 0 или 255, её обрезает constrain

```

const int BASE = 150;      // база – скорость на прямой
const float K = 0.25;      // коэффициент

void loop() {
    int left = analogRead(A0);           // левый датчик
    int right = analogRead(A1);          // правый датчик
    int error = left - right;             // ошибка
    int u = K * error;                   // поправка (дробная часть отбрасывается)
    drive(constrain(BASE + u, 0, 255),    // левый мотор
           constrain(BASE - u, 0, 255));  // правый мотор
    delay(10);
}

```

`float` – переменная для дробного числа (у `int` дробей не бывает). В программе нет ни одного `if`: робот не выбирает из случаев, он каждые 10 мс заново считает поправку по свежей ошибке.

Ошибка «левый – правый» честная, только если датчики одинаковые. Поставь слева датчик с белым 900, а справа – с белым 700 (из раздела 7.2), и на прямой, когда оба над белым, ошибка выйдет $900 - 700 = 200$. Робот будет всё время уходить направо, хотя линия у него точно посередине. Лечится это переводом в проценты из раздела 7.2: ошибку считают как разницу белизны. Разберём в задаче 7.6.

Запомни. Ошибка = левый – правый; поправка = $k \cdot$ ошибка; левый мотор = база + поправка, правый = база – поправка. Отрицательная ошибка – робот поворачивает налево, положительная – направо. Скорости зажимают между 0 и 255.

Совет. Если на полигоне робот не возвращается к линии, а убегает от неё, перепутан знак: поменяй местами плюс и минус у моторов (или датчики местами в ошибке). Подбирать k до этого бесполезно.

7.5. Почему робот виляет

НА ПАЛЬЦАХ



Рисунок 7.8 – Три робота с разным коэффициентом на одном и том же повороте

Маленький k – робот не успевает довернуть и уезжает с поворота. Подходящий – проходит его. Слишком большой – виляет уже на прямой, а на повороте теряет линию.

П-регулятор тоже умеет вилять, хоть и не так отчаянно, как релейный. Причин три, и на полигоне они действуют заодно.

Робот узнаёт об ошибке с опозданием. Программа читает датчики раз в 10 мс, моторам нужно время разогнаться и притормозить, а корпус робота по инерции продолжает поворачивать. **Инерция** – свойство тела продолжать двигаться так же, как оно двигалось: ты это чувствуешь в автобусе, когда он резко тормозит. Пока ошибка дошла до нуля, нос робота уже смотрит поперёк линии, и робот её проскакивает. С душем то же самое: вода нужной температуры доходит до тебя не сразу. Крутанёшь кран сильно – через секунду кипяток, крутанёшь обратно – ледяная.

Коэффициент слишком большой. Поправка огромная даже на маленькой ошибке, и робот поворачивает так резко, что каждый раз перелетает линию. У релейного регулятора эта беда всегда: поправка у него одна и та же, большая, поэтому он и едет зигзагом.

Скорость слишком большая. За те же 10 мс опоздания быстрый робот уезжает дальше. На 30 см/с за 10 мс это $30 \cdot 0,01 = 0,3$ см = 3 мм, на 60 см/с – уже 6 мм, а линия всего 15–20 мм шириной.

А слишком **маленький** k плох по-другому: на прямой робот едет спокойно, но на крутом повороте не успевает довернуть, и линия уходит из-под обоих датчиков. Тогда ошибка снова 0, и робот уверенно едет прямо – мимо поворота.

Как подбирают k на практике? Начинают с маленького, и пока робот не справляется с поворотами, увеличивают, например, вдвое. Как только робот начал вилять на прямой, возвращаются на шаг назад. Если ни один k не подходит, уменьшают базу.

Как это посчитать. Упростим робота: пусть за один шаг программы он сдвигается к линии на k умноженное на смещение. Смещение 10 мм, $k = 0,5$: сдвинулся на 5 мм, осталось 5 мм. Следующий шаг: сдвиг 2,5 мм, осталось 2,5 мм. Робот подходит к линии и не перескакивает

её. А при $k = 1,5$ робот сдвигается на 15 мм – на 5 мм дальше линии, на другую сторону. Такие цепочки мы посчитаем в задаче 7.7.

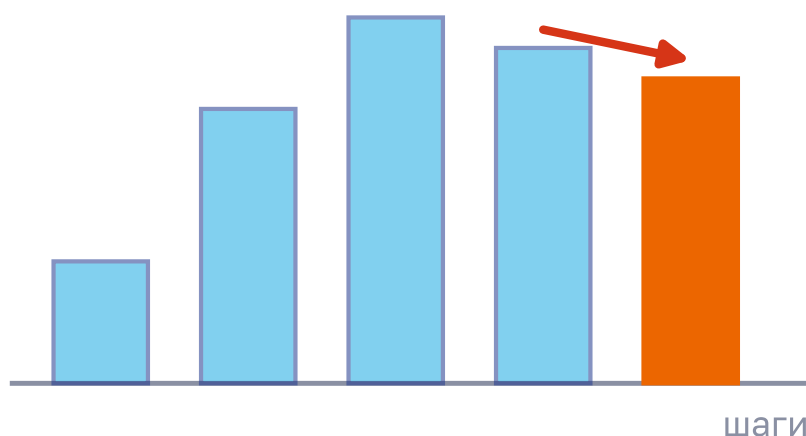
Для любопытных: один множитель на всё. В этой модели новое смещение = старое $\cdot (1 - k)$. Смещения по разные стороны от линии запишем с разными знаками: справа – плюс, слева – минус. Если k меньше 1, множитель положительный и меньше единицы: робот подходит к линии с одной стороны. Если k от 1 до 2, множитель отрицательный: смещение меняет знак, робот перескакивает линию, но каждый раз всё меньше. При $k = 2$ множитель -1 : робот качается туда-сюда на одном и том же расстоянии, как качели. Больше 2 – каждый раз всё дальше, пока не слетит. У настоящего робота из-за опоздания влияние начинается раньше, при меньшем k .

Запомни. Влияние – это перелёт через линию. Его дают большой k , большая скорость и опоздание реакции. Маленький k не влияет, но на повороте вылетает. Подбирают k так: увеличивают, пока не начнётся влияние, и чуть уменьшают.

7.6. ПИД-регулятор: идея

НА ПАЛЬЦАХ

И – все столбики вместе, «копилка»



Д – как быстро
меняется ошибка:
уменьшается

П – ошибка сейчас

Рисунок 7.9 – Ошибка за последние шаги: что видит каждая буква ПИД

П смотрит, какая ошибка сейчас. И – сколько ошибок набралось за всё время. Д – как быстро ошибка меняется.

Для большинства полигонов П-регулятора хватает. Подводит он, когда один мотор слабее другого: робот всё время чуть тянет вбок и едет немного сбоку от линии. Ошибка маленькая, поправка тоже маленькая – сил вытащить робота на середину у неё нет. Да и влияние из раздела 7.5 никуда не делось. Против обеих бед к П приписывают ещё две части и получают **ПИД-регулятор**: его поправка – сумма трёх слагаемых.

- **П – пропорциональная часть:** $k_P \cdot \text{ошибка}$. Та же, что в разделе 7.4.
- **И – интегральная часть** («интеграл» у математиков – сумма многих маленьких кусочков): $k_I \cdot \text{сумма ошибок}$. Каждый шаг программа прибавляет свежую ошибку в копилку. Робот долго едет чуть сбоку – копилка растёт, и поправка растёт вместе с ней, пока не вытянет робота на линию.
- **Д – дифференциальная часть** («дифференциал» – разница): $k_D \cdot (\text{ошибка} - \text{прошлая ошибка})$. Смотрит, как быстро меняется ошибка. Если робот стремительно возвращается к линии, ошибка быстро уменьшается, эта разница отрицательная – и Д-часть заранее притормаживает поворот, чтобы робот не проскочил. Так водитель отпускает газ и тормозит перед стоп-линией заранее, а не на ней.

$$\text{поправка} = k_P \cdot \text{ошибка} + k_I \cdot \text{сумма ошибок} + k_D \cdot (\text{ошибка} - \text{прошлая ошибка}).$$

Пример. $k_P = 0,4$, $k_I = 0,02$, $k_D = 3$. Ошибка сейчас 50, на прошлом шаге была 60, в копилке вместе с сегодняшней набралось 300.

- П: $0,4 \cdot 50 = 20$.
- И: $0,02 \cdot 300 = 6$.
- Д: $3 \cdot (50 - 60) = 3 \cdot (-10) = -30$.
- Поправка: $20 + 6 - 30 = -4$.

П и И просят повернуть направо, а Д говорит: «робот и так быстро возвращается, хватит». В сумме поправка почти ноль, и робот плавно доезжает до линии.

```
const float KP = 0.4, KI = 0.02, KD = 3;
long sum = 0;           // копилка ошибок
int prevError = 0;      // ошибка на прошлом шаге

int pid(int error) {
    sum = sum + error;    // положили ошибку в копилку
    float u = KP * error + KI * sum + KD * (error - prevError);
    prevError = error;    // запомнили для следующего шага
    return u;
}
```

Настраивать три коэффициента сразу трудно даже инженерам, у которых робот стоит на столе, – поэтому начинают с одной П, а И и Д добавляют понемногу. На отборе от тебя ждут другого: понимать, за что отвечает каждая буква, и уметь по готовым числам сложить три слагаемых, не потеряв знак у Д.

Запомни. ПИД = П (ошибка сейчас) + И (сумма ошибок) + Д (ошибка минус прошлая ошибка), каждая часть со своим коэффициентом. Д отрицательная, когда ошибка уменьшается, – она гасит влияние. И помогает, когда робот долго едет чуть сбоку.

7.7. Разбираем задачи

Задача 7.1 Три замера на каждом цвете

разминка · калибровка, среднее, порог датчика линии

Перед заездом Вика калибрует датчик линии. Над белым полем она сделала три замера: 904, 896, 912. Над чёрной линией – тоже три: 118, 126, 122. Порог считают посередине между средним на белом и средним на чёрном. а) Какой получится порог? б) Какие из показаний означают, что под датчиком линия: 1) 480; 2) 530; 3) 212; 4) 760? В ответ введи порог, а затем номера показаний подряд.

Дано: белое – 904, 896, 912; чёрное – 118, 126, 122.

Найти: порог; какие показания – линия.

Решение

Шаг 1. Среднее на белом. Складываем три замера и делим на 3: $904 + 896 + 912 = 2712$, $2712 : 3 = 904$.

Шаг 2. Среднее на чёрном. $118 + 126 + 122 = 366$, $366 : 3 = 122$.

Шаг 3. Порог. $(904 + 122) : 2 = 1026 : 2 = 513$.

Шаг 4. Где линия. У этого датчика чёрное – маленькое число, значит, линия – это показания меньше 513. 480 – меньше, линия. 530 – больше, поле: робот у самого края, но ещё над белым. 212 – меньше, линия. 760 – больше, поле. Линию означают показания 1 и 3.

 **Типичная ошибка.** Перепутать направление: «больше порога – линия». Тогда выйдут показания 2 и 4 – всё наоборот. Перед сравнением посмотри, какое число у линии в замерах: здесь это 118–126, маленькие.

Ответ: а) 513; б) 13.

Своя задача по программе сферы, модуль 7

Реши сам

7.1. Датчик линии над белым полем показал 820, 812 и 816, над чёрной линией – 204, 196 и 200. Найди порог.

7.2. В Tinkercad датчик линии изобразили фоторезистором, но подключили так, что над белым полем он показывает 150, а над чёрной линией – 870. Порог берут посередине. Какие из показаний означают линию: 1) 495; 2) 620; 3) 880; 4) 140? В ответ введи номера подряд.

Задача 7.2 Куда повернёт робот

релейный регулятор, два датчика, свой порог у каждого датчика, перекрёсток

У робота два датчика линии. При калибровке левый показал над белым 880, над чёрным 160; правый – 760 и 200. У каждого датчика свой порог посередине. Робот едет с релейным регулятором из раздела 7.3: оба на белом – 1 (прямо); линия только под левым – 2 (налево); линия только под правым – 3 (направо); оба на чёрном – 4 (перекрёсток, стоп). Вот показания датчиков в шесть моментов:

Таблица 7.3

Момент	Левый	Правый
А	812	740
Б	230	705
В	505	690
Г	790	490
Д	760	300
Е	140	170

Что делает робот в каждый момент? Ответ – шесть цифр подряд, для А, Б, В, Г, Д, Е.

Дано: левый – белое 880, чёрное 160; правый – белое 760, чёрное 200; таблица показаний.

Найти: действие в каждый момент.

Решение

Шаг 1. Пороги. Левый: $(880 + 160) : 2 = 1040 : 2 = 520$. Правый: $(760 + 200) : 2 = 960 : 2 = 480$.

Шаг 2. Каждое показание – в слово. Меньше своего порога – «чёрное», иначе «белое». Левый сравниваем только с 520, правый – только с 480.

- А: $812 > 520$ – белое; $740 > 480$ – белое.
- Б: $230 < 520$ – чёрное; 705 – белое.
- В: $505 < 520$ – чёрное; 690 – белое.
- Г: 790 – белое; $490 > 480$ – белое.
- Д: 760 – белое; $300 < 480$ – чёрное.
- Е: 140 – чёрное; 170 – чёрное.

Шаг 3. Слова – в действия.

- А: оба белые – прямо, 1.
- Б: линия под левым – налево, 2.
- В: опять линия под левым – налево, 2.
- Г: оба белые – прямо, 1.
- Д: линия под правым – направо, 3.
- Е: оба чёрные – перекрёсток, 4.

Моменты В и Г – самые коварные: 505 и 490 почти одинаковые, но у левого датчика 505 – уже линия, а у правого 490 – ещё поле. Каждый датчик сравнивается только со своим порогом.

⚠ Типичная ошибка. Взять один порог на оба датчика, например 500 – «круглое, посередине шкалы». Тогда в момент В робот поедет прямо ($505 > 500$), а в момент Г свернёт направо ($490 < 500$), и ответ выйдет 121334. Вторая частая ошибка – поворачивать **от** линии: «левый увидел чёрное – уходим направо». Робот должен поворачивать к линии, иначе он с неё просто уедет.

Ответ: 122134.

Своя задача по программе сферы, модуль 7

Реши сам

7.3. Датчики робота откалибровали: левый – белое 900, чёрное 140; правый – белое 820, чёрное 180. Пороги посередине, действия те же, что в задаче 7.2 (1 – прямо, 2 – налево, 3 – направо, 4 – перекрёсток). Показания (левый, правый): А – 870 и 410; Б – 515 и 800; В – 530 и 505; Г – 150 и 160; Д – 880 и 810. Ответ – пять цифр для А, Б, В, Г, Д.

7.4. Робот едет по линии и записывает показания двух датчиков каждые 20 мс. Порог у обоих датчиков 500, линия – меньше порога. Перекрёсток – это когда оба датчика одновременно на чёрном. Поперечную полосу робот переезжает не мгновенно и успевает записать на ней от одной до трёх строк. Вот записи подряд (левый, правый): 850 и 820; 830 и 300; 820 и 790; 210 и 180; 190 и 160; 800 и 780; 240 и 830; 820 и 810; 150 и 170; 810 и 790; 790 и 260; 830 и 800; 170 и 190; 160 и 150; 180 и 200; 840 и 800. Сколько перекрёстков проехал робот?

Задача 7.3 Скорости моторов по П-регулятору

П-регулятор, ошибка, поправка, скорости моторов, направление поворота

Робот едет по линии с П-регулятором: ошибка = левый – правый, поправка = $k \cdot$ ошибка, левый мотор = база + поправка, правый мотор = база – поправка. База 150, $k = 0,4$. Датчики одинаковые, белое у них – большое число. Сейчас левый датчик показывает 380, правый – 630. Найди скорости левого и правого моторов. В какую сторону поворачивает робот? В ответ введи скорость левого мотора, скорость правого и цифру: 1 – налево, 2 – направо.

Дано: база 150; $k = 0,4$; левый 380; правый 630.

Найти: скорости моторов; направление поворота.

Решение

Шаг 1. Ошибка. Левый минус правый: $380 - 630 = -250$. Минус означает, что левый датчик темнее – линия ближе к нему.

Шаг 2. Поправка. $0,4 \cdot (-250) = -100$.

Шаг 3. Левый мотор. $150 + (-100) = 150 - 100 = 50$.

Шаг 4. Правый мотор. $150 - (-100) = 150 + 100 = 250$. Минус на минус дал плюс: правый ускорился.

Шаг 5. Куда поворачивает. Левое колесо медленнее, робот уходит в сторону медленного колеса – налево. Линия как раз слева (шаг 1), значит, робот возвращается к ней. Оба числа в пределах 0...255, обрезать ничего не надо.

 **Типичная ошибка.** Потерять знак: взять ошибку $630 - 380 = 250$ «потому что так положительно». Тогда поправка 100, левый мотор 250, правый 50 – и робот повернёт направо, от линии. Порядок «левый – правый» задан в условии, менять его нельзя. И вторая: вычитая -100 у правого мотора, получить $150 - 100 = 50$ вместо 250.

Ответ: 50; 250; 1 (налево).

Своя задача по программе сферы, модуль 7

Реши сам

7.5. Регулятор тот же, что в задаче 7.3, но база 120 и $k = 0,5$. Левый датчик показывает 700, правый – 460. Найди скорости левого и правого моторов и направление поворота (1 – налево, 2 – направо).

7.6. Регулятор тот же, база 200, $k = 0,35$. Левый датчик показывает 300, правый – 800. Скорости обрезаются, как `constrain(x, 0, 255)`. Какие значения получают левый и правый моторы?

Задача 7.4 Подбираем коэффициент

П-регулятор, подбор коэффициента, самая большая ошибка, пропорция

У робота два одинаковых датчика линии: над белым они показывают 900, над чёрным – 100. Регулятор как в задаче 7.3, база 160. Команда хочет, чтобы при самой большой ошибке медленное колесо как раз останавливалось (скорость 0), но не уходило в минус. а) Какой нужен коэффициент k ? б) Какими с этим k будут скорости левого и правого моторов, если ошибка равна -400 ? Ответ: k , затем две скорости.

Дано: белое 900; чёрное 100; база 160; при самой большой ошибке медленный мотор 0.

Найти: k ; скорости при ошибке -400 .

Решение

Шаг 1. Какая ошибка самая большая. Ошибка = левый – правый. Больше всего она по размеру, когда один датчик целиком на чёрном, а другой на белом: $100 - 900 = -800$ или $900 - 100 = 800$. По размеру это 800.

Шаг 2. Какая нужна поправка. Медленный мотор = база – размер поправки. Чтобы он стал 0, поправка по размеру должна быть равна базе: 160.

Шаг 3. Коэффициент. $k \cdot 800 = 160$, значит, $k = 160 : 800 = 0,2$.

Шаг 4. Ошибка -400 . Поправка $0,2 \cdot (-400) = -80$. Левый мотор $160 - 80 = 80$, правый $160 + 80 = 240$. Робот поворачивает налево.

Ошибка -400 – это половина самой большой, и поправка вышла половиной базы: 80 из 160. На то регулятор и пропорциональный.

❗ Типичная ошибка. Взять за самую большую ошибку 900 – «белое число» – и получить $k = 160:900 \approx 0,18$. Ошибка – это разница двух показаний, и больше чем на $900 - 100 = 800$ показания отличаться не могут. Ещё ошибаются, решая, что медленный мотор при ошибке -400 – правый: при отрицательной ошибке медленный – левый.

Ответ: а) 0,2; б) 80 и 240.

Своя задача по программе сферы, модуль 7

Реши сам

7.7. Датчики показывают над белым 850, над чёрным 150. База 175. Каким должен быть коэффициент k , чтобы при самой большой ошибке медленное колесо как раз останавливалось?

7.8. Датчики показывают над белым 820, над чёрным 220, коэффициент $k = 0,2$. Какую базу выбрать, чтобы при самой большой ошибке **быстрый** мотор как раз дошёл до 255? Сколько тогда будет у медленного мотора?

Задача 7.5 Три слагаемых ПИД

ПИД-регулятор, сумма ошибок, разница ошибок, знак

Робот едет по линии с ПИД-регулятором: $k_{\Pi} = 0,5$, $k_{\text{И}} = 0,1$, $k_{\text{Д}} = 2$. С момента старта программа сосчитала ошибку четыре раза: 30, 50, 60, 50 (последнее число – ошибка сейчас). Сумма ошибок – сумма всех четырёх. Поправка = $k_{\Pi} \cdot \text{ошибка} + k_{\text{И}} \cdot \text{сумма ошибок} + k_{\text{Д}} \cdot (\text{ошибка} - \text{прошлая ошибка})$. База 140, левый мотор = база + поправка, правый = база – поправка. Найди поправку и скорости моторов. Ответ: три числа.

Дано: $k_{\Pi} = 0,5$; $k_{\text{И}} = 0,1$; $k_{\text{Д}} = 2$; ошибки 30, 50, 60, 50; база 140.

Найти: поправку; левый и правый моторы.

Решение

Шаг 1. Что есть что. Ошибка сейчас – последняя, 50. Прошлая ошибка – предпоследняя, 60. Сумма ошибок – $30 + 50 + 60 + 50 = 190$.

Шаг 2. П-часть. $0,5 \cdot 50 = 25$.

Шаг 3. И-часть. $0,1 \cdot 190 = 19$.

Шаг 4. Д-часть. $2 \cdot (50 - 60) = 2 \cdot (-10) = -20$. Ошибка уменьшилась – Д-часть отрицательная.

Шаг 5. Поправка. $25 + 19 - 20 = 24$.

Шаг 6. Моторы. Левый $140 + 24 = 164$, правый $140 - 24 = 116$. Робот поворачивает направо: ошибка положительная, линия под правым датчиком.

Без Д поправка была бы $25 + 19 = 44$. Д-часть срезала её почти вдвое, потому что робот уже возвращается к линии: на прошлом шаге ошибка была 60, а теперь 50.

❗ Типичная ошибка. Считать разницу наоборот, «прошлая минус сейчас»: $60 - 50 = 10$, Д-часть +20 и поправка 64. Порядок в формуле – «ошибка сейчас минус прошлая». Ещё забывают положить в сумму последнюю ошибку и получают 140 вместо 190.

Ответ: поправка 24; левый мотор 164, правый 116.

Своя задача по программе сферы, модуль 7

Реши сам

7.9. ПИД-регулятор: $k_P = 0,6$, $k_I = 0,1$, $k_D = 1,5$. Программа сосчитала ошибку три раза: 20, 40, 70 (последняя – сейчас), сумма ошибок – сумма всех трёх. Найди П-часть, И-часть, Д-часть и поправку.

7.10. Один мотор робота слабее, и робот всё время едет чуть сбоку от линии: ошибка на каждом шаге одна и та же, 10. $k_P = 0,5$, $k_I = 0,05$, копилка перед первым шагом пустая. Чему равна И-часть после 5 шагов и после 20 шагов? Во сколько раз И-часть после 20 шагов больше П-части?

Задача 7.6^{*}. Датчики из разных наборов

разные датчики, проценты белизны, П-регулятор

Левый датчик робота показывает над белым 920, над чёрным 140. Правый взяли из другого набора: над белым 760, над чёрным 200. Регулятор как в задаче 7.3, база 150.

а) Сначала ошибку считали прямо по показаниям, с $k = 0,25$. Робот стоит посередине, линия между датчиками, оба над белым. Какие скорости получают моторы?

б) Потом показания стали переводить в проценты белизны (раздел 7.2), ошибку – считать как «белизна левого – белизна правого», а k взяли 2. Сейчас левый показывает 530, правый 620. Найди ошибку и скорости моторов.

Ответ: две скорости для пункта а), ошибка и две скорости для пункта б).

Дано: левый 920/140; правый 760/200; база 150; а) $k = 0,25$, оба над белым; б) $k = 2$, показания 530 и 620.

Найти: а) скорости; б) ошибку в процентах и скорости.

Решение

Шаг 1. Пункт а), ошибка. Оба над белым: левый 920, правый 760. Ошибка $920 - 760 = 160$.

Шаг 2. Пункт а), моторы. Поправка $0,25 \cdot 160 = 40$. Левый $150 + 40 = 190$, правый $150 - 40 = 110$. Робот стоит точно над линией, но сворачивает направо – его всё время будет стаскивать с прямой линии, и регулятору придётся бороться с собственной ошибкой.

Шаг 3. Пункт б), белизна левого. $(530 - 140) : (920 - 140) \cdot 100 = 390 : 780 \cdot 100 = 50 \%$.

Шаг 4. Белизна правого. $(620 - 200) : (760 - 200) \cdot 100 = 420 : 560 \cdot 100 = 75 \%$.

Шаг 5. Ошибка. $50 - 75 = -25$ (процентов). Левый датчик темнее – линия ближе к нему.

Шаг 6. Моторы. Поправка $2 \cdot (-25) = -50$. Левый $150 - 50 = 100$, правый $150 + 50 = 200$. Робот поворачивает налево, к линии.

А что теперь на прямой, где оба над белым? Белизна у обоих 100 %, ошибка $100 - 100 = 0$, робот едет прямо. Болезнь из пункта а) прошла: силу поворота теперь задаёт то, где линия, а не то, у какого датчика числа крупнее.

 **Типичная ошибка.** В формуле белизны делить на само «белое» число: $530 : 920$. Делить нужно на весь путь от чёрного до белого, $920 - 140 = 780$. И сверху стоит то, на сколько показание поднялось над чёрным: $530 - 140$. Иначе над линией белизна выйдет не 0 %, а 15 %.

Ответ: а) 190 и 110; б) ошибка -25, моторы 100 и 200.

Своя задача по программе сферы, модуль 7

Реши сам

7.11. Датчик над белым показывает 880, над чёрным – 180. Сейчас он показывает 390. Какова белизна в процентах?

7.12.* Левый датчик: белое 860, чёрное 160. Правый: белое 780, чёрное 280. Ошибка – белизна левого минус белизна правого (в процентах), $k = 1,5$, база 130, регулятор как в задаче 7.3. Левый показывает 720, правый – 530. Найди ошибку и скорости левого и правого моторов.

Задача 7.7* *. Влияет или нет

П-регулятор, влияние, пошаговое моделирование, знак смещения

Проверим коэффициенты на упрощённой модели из раздела 7.5. Робот стоит в 10 мм справа от линии. На каждом шаге программы он сдвигается к линии на $k \cdot$ смещение (смещение – расстояние от линии в миллиметрах, справа – плюс, слева – минус). Четыре робота отличаются только коэффициентом: А – $k = 2$; Б – $k = 0,6$; В – $k = 2,4$; Г – $k = 1,6$. Посчитай для каждого смещение после четырёх шагов и определи, как он ведёт себя: 1 – подходит к линии с одной стороны, не перескакивая её; 2 – перескакивает линию, но размах уменьшается; 3 – качается на одном и том же расстоянии; 4 – размах растёт, робот слетает. Ответ – четыре цифры для А, Б, В, Г.

Дано: смещение в начале 10 мм; сдвиг за шаг = $k \cdot$ смещение; $k = 2; 0,6; 2,4; 1,6$.

Найти: поведение каждого робота.

Решение

Шаг 1. Правило одного шага. Новое смещение = смещение – $k \cdot$ смещение. Если вышло меньше нуля, робот оказался слева от линии – он её перескочил.

Шаг 2. Робот Б, $k = 0,6$. $10 - 0,6 \cdot 10 = 10 - 6 = 4$. Дальше $4 - 0,6 \cdot 4 = 4 - 2,4 = 1,6$. Потом $1,6 - 0,96 = 0,64$, и $0,64 - 0,384 = 0,256$ мм. Все числа положительные и тают: робот подходит к линии справа и ни разу её не перескакивает – 1.

Шаг 3. Робот Г, $k = 1,6$. $10 - 1,6 \cdot 10 = 10 - 16 = -6$: перескочил, теперь 6 мм слева. Дальше $-6 - 1,6 \cdot (-6) = -6 + 9,6 = 3,6$. Потом $3,6 - 5,76 = -2,16$, и $-2,16 + 3,456 = 1,296$ мм. Знак скачет, а размах $10 \rightarrow 6 \rightarrow 3,6 \rightarrow 2,16 \rightarrow 1,3$ уменьшается – 2.

Шаг 4. Робот А, $k = 2$. $10 - 20 = -10$, затем $-10 + 20 = 10$, снова -10 , снова 10. Робот качается между 10 мм справа и 10 мм слева и никогда не успокоится – 3.

Шаг 5. Робот В, $k = 2,4$. $10 - 24 = -14$, затем $-14 + 33,6 = 19,6$, потом $19,6 - 47,04 = -27,44$, потом $-27,44 + 65,856 = 38,416$ мм. Каждый раз дальше. Датчики стоят в 1–1,5 см от середины робота, так что на втором-третьем шаге линия уйдёт из-под них – 4.

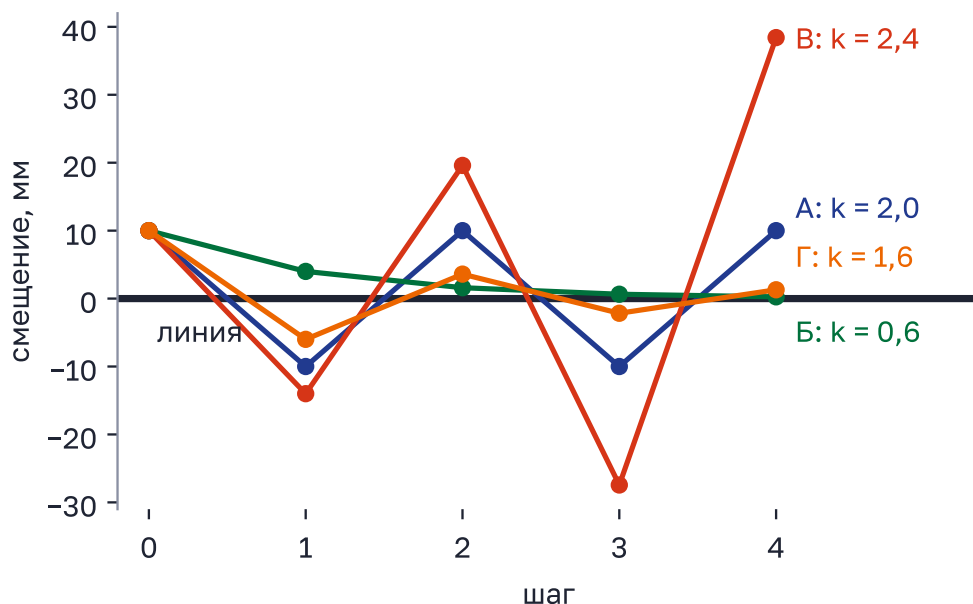


Рисунок 7.10 – Смещение от линии по шагам у четырёх роботов. Выше нуля – справа от линии, ниже – слева

Можно проверить на Python:

```
k = 1.6           # коэффициент робота Г
e = 10            # смещение, мм
for shag in range(1, 5):
    e = e - k * e  # робот сдвинулся к линии
    print(shag, round(e, 2))
```

Программа напечатает 1 -6.0, 2 3.6, 3 -2.16, 4 1.3.

⚠ **Типичная ошибка.** Забыть про знак: «смещение 6, потом 9,6 – растёт, значит, 4». Если смещение стало отрицательным, то $k \cdot$ смещение тоже отрицательное, и вычитать его – значит прибавлять. Второй промах – решить, что раз робот Г перескакивает линию, он «плохой», как В. Разница в размахе: у Г он тает, у В растёт.

Ответ: 3142.

По мотивам задачи первого отборочного этапа НТО 2024/25, профиль «Интеллектуальные робототехнические системы», «Устойчивость динамических систем»

Реши сам

7.13. В той же модели робот стоит в 12 мм справа от линии, $k = 1,5$. Найди смещение после трёх шагов (со знаком: справа – плюс, слева – минус).

7.14.* В той же модели у пяти роботов коэффициенты: 1) 0,4; 2) 1,3; 3) 2,0; 4) 0,9; 5) 2,6. Какие роботы ни разу не перескочат линию? А у какого размах будет расти? Ответ: номера первых подряд, затем номер второго.

Сигналы и обработка данных



Пока у робота есть питание, датчик выдаёт число за числом. Среди них попадаются странные: дальномер посреди комнаты вдруг показывает 0 см, датчик линии «видит» линию на чистом листе. Одному числу верить нельзя – смотреть надо на ряд целиком. Здесь ты узнаешь, как часто снимать показания, чтобы не проскочить полосу, и как сгладить ряд, чтобы соринка на поле не превратилась в лишний перекрёсток. На отборе такие задачи приходят рядом чисел или графиком: сгладь, найди выброс, посчитай события, найди период. Ответ – число или строка из цифр.

8.1. Сигнал: рассказ датчика во времени

НА ПАЛЬЦАХ

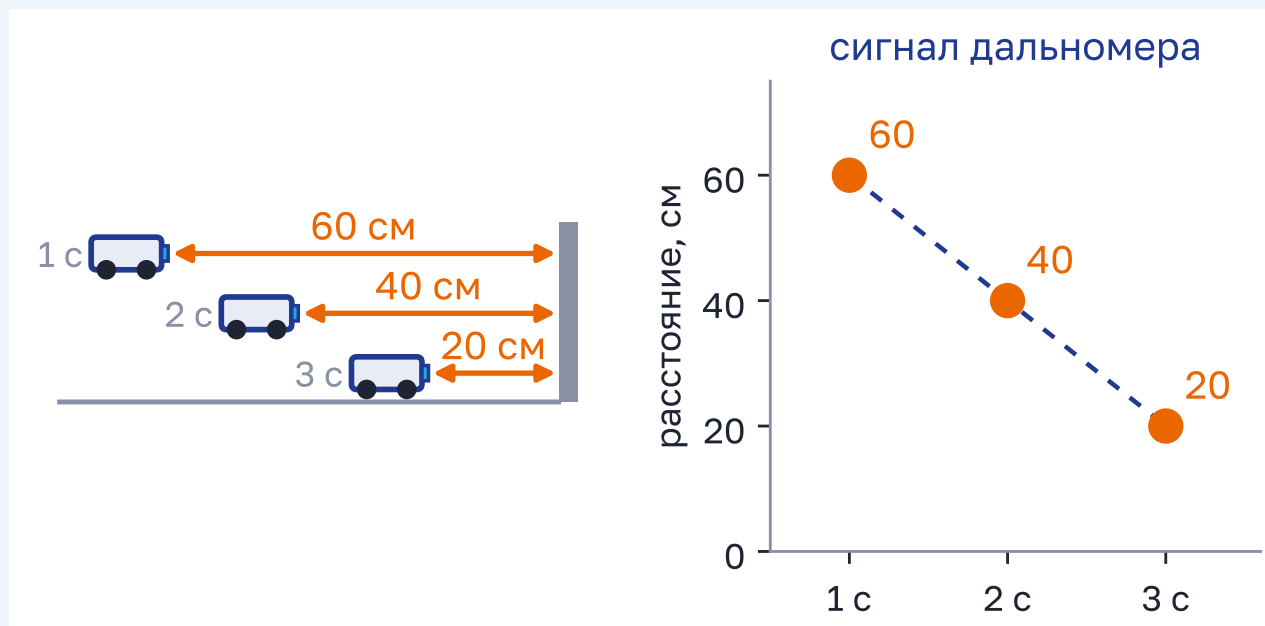


Рисунок 8.1 – Робот подъезжает к стене и раз в секунду меряет расстояние

Каждое измерение – точка на графике. Точки подряд складываются в сигнал, и по нему видно: робот приближается к стене, по 20 см в секунду.

Одно показание почти ничего не говорит. Дальномер показал 40 см – робот стоит? Едет к стене? Отъезжает? Ясно станет, только когда помотришь на несколько показаний подряд: 60, 40, 20 – приближается, и довольно быстро. **Сигнал** – это величина, которая меняется со временем и рассказывает, что происходит вокруг: расстояние до стены, яркость поля под датчиком линии, напряжение на входе платы. Сигнал рисуют графиком: по горизонтали время, по вертикали значение. Не всякое меняющееся явление – сигнал: сигналом оно становится, когда по нему что-то передают или узнают. Дрожание листьев за окном – просто явление, а мигание маяка, по которому робот находит базу, – сигнал.

Сигналы бывают двух видов. **Аналоговый** меняется плавно и может принимать любые промежуточные значения – так ведёт себя напряжение с датчика линии, от 0 до 5 В. **Цифровой** состоит из отдельных значений, у Arduino их два: есть напряжение или нет, 1 или 0. Так работают кнопка и вывод Echo у дальномера. Оба ты уже видел в главе 5.



Рисунок 8.2 – Светодиод секунду горит, секунду не горит – и так по кругу

Кусок «горит + не горит» повторяется снова и снова. Его длина – период, здесь 2 с. За 6 секунд светодиод успел мигнуть три раза.

Многие сигналы повторяются. Мигает светодиод, белая метка на колесе раз за оборот пробегает мимо датчика, дрожит воздух, когда звучит нота. Кусок, который повторяется, называют **колебанием**. Время одного колебания – **период** T , в секундах. Сколько колебаний проходит за секунду – **частота** f . Её меряют в герцах (Гц): 1 Гц – одно колебание в секунду, 50 Гц – пятьдесят. Частота – это единица, делённая на период, и наоборот:

$$f = \frac{1}{T}, \quad T = \frac{1}{f}.$$

У светодиода с картинки $T = 2$ с, а $f = 1:2 = 0,5$ Гц: полмигания в секунду.

Качели и маятник качаются плавно, и на графике у них выходит ровная волна – синусоида. Про такую волну говорят ещё два слова. **Размах** – от самого маленького значения до самого большого. **Амплитуда** – на сколько сигнал отходит от середины в каждую сторону, половина размаха.

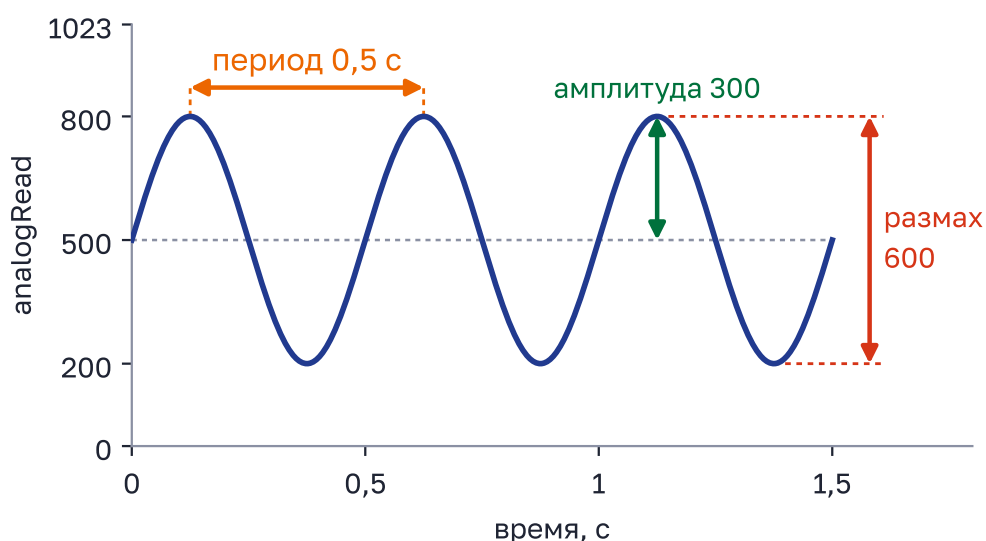


Рисунок 8.3 – Волна на входе платы: от 200 до 800, одно колебание за 0,5 с

По рисунку: размах $800 - 200 = 600$, амплитуда $600 : 2 = 300$, середина $200 + 300 = 500$. От одной верхушки до соседней $0,5$ с – это период, частота $1 : 0,5 = 2$ Гц. Период удобнее мерить по верхушкам или по низам: там их легко найти глазом.

Зачем роботу колебания? По периоду он узнаёт маяк и меряет скорость колеса (раздел 8.5). А шум от полезного сигнала отличается тем, насколько быстро он меняется (раздел 8.6).

Запомни. Сигнал – величина, которая меняется во времени. Период T – время одного колебания, частота f – сколько колебаний в секунду, $f = 1 : T$. Размах – от минимума до максимума, амплитуда – его половина.

8.2. Дискретизация: контроллер видит точки

НА ПАЛЬЦАХ

смотрит каждые 6 мм



смотрит каждые 30 мм



→ робот едет

Рисунок 8.4 – Датчик смотрит на поле короткими взглядами

Между двумя взглядами робот успевает проехать какой-то путь. Если этот путь больше ширины полосы, полоса может проскочить незамеченной.

Датчик линии выдаёт напряжение всё время. А программа читает его только тогда, когда доходит до строчки `analogRead`. В остальное время она занята: считает, командует моторами, ждёт в `delay`. Поэтому контроллер видит сигнал точками. Одно такое измерение называют **отсчётом**. Превращение плавного сигнала в ряд отсчётов называют **дискретизацией**: «дискретный» значит «прерывистый, из отдельных кусочков».

Сколько отсчётов программа снимает за секунду, называют **частотой опроса**. Промежуток между отсчётами обозначают Δt («дельта тэ»), и он связан с частотой так же, как период:

$$\Delta t = \frac{1}{f_{\text{опр}}}.$$

Если в `loop` стоит `delay(20)`, а остальные строки выполняются за доли миллисекунды, отсчёты идут каждые 20 мс, то есть $1000 : 20 = 50$ раз в секунду.

Робот при этом едет. Путь между двумя отсчётами – это скорость, умноженная на промежуток (глава 2): $s = v \cdot \Delta t$. Пусть робот едет со скоростью $0,3$ м/с, это 300 мм/с.

- `delay(20)` : $\Delta t = 0,02$ с, $s = 300 \cdot 0,02 = 6$ мм. На поперечную полосу шириной 18 мм придётся $18 : 6 = 3$ отсчёта.

- `delay(100)` : $\Delta t = 0,1$ с, $s = 300 \cdot 0,1 = 30$ мм. Шаг больше ширины полосы, и полоса может целиком уместиться между двумя отсчётами – как в нижней строке рисунка.

Сколько отсчётов попадёт на полосу **наверняка**, узнаём делением ширины на шаг с отбрасыванием дробной части: $18 : 6 = 3$ – три наверняка, а если первый отсчёт придётся на самый край, то и четыре. $18 : 10 = 1,8$ – наверняка один. $18 : 30 = 0,6$ – ни одного: может повезти, а может и нет.

С колебаниями то же самое. Чтобы заметить одно колебание, на него нужно больше двух отсчётов, а лучше пять-десять: два отсчёта, попавшие в нули синусоиды, ничего не покажут. Светодиод мигает 5 раз в секунду, а программа смотрит на него тоже 5 раз в секунду – каждый раз она попадает в одну и ту же фазу мигания и решает, что светодиод горит всё время (или не горит вовсе). Похожее ты видел в кино: колёса машины на экране иногда будто крутятся назад. Камера снимает кадры с промежутками и не успевает за спицами.

Нарезан сигнал не только по времени. `analogRead` меряет напряжение ступеньками, 1024 деления на 5 В, одно деление – около 5 мВ (глава 5). Так что до программы доходит «лесенка»: по времени – отсчёты, по величине – деления.

Запомни. Отсчёт – одно измерение. Промежуток между отсчётами $\Delta t = 1 : f_{\text{опр}}$, путь робота за это время $s = v \cdot \Delta t$. Наверняка на полосу попадёт столько отсчётов, сколько целых шагов в её ширине.

Совет. Удобный фокус с единицами: метры в секунду, умноженные на миллисекунды, дают миллиметры. $0,3 \text{ м/с} \cdot 20 \text{ мс} = 6 \text{ мм}$ – без перевода в секунды. Если сомневаешься, переводи честно: $0,3 \text{ м/с} = 300 \text{ мм/с}$, $20 \text{ мс} = 0,02 \text{ с}$.

8.3. Шум и выбросы

НА ПАЛЬЦАХ

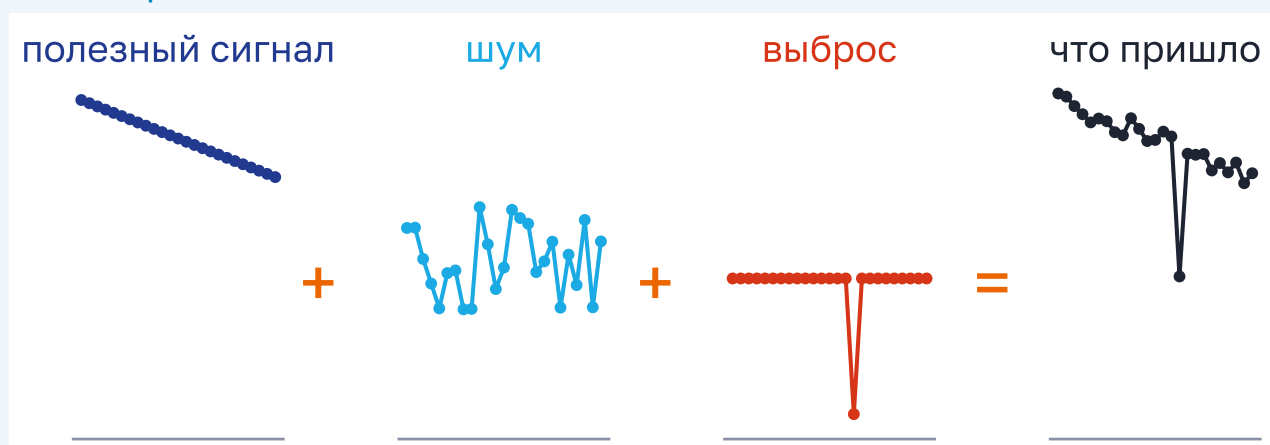


Рисунок 8.5 – Что приходит с датчика: полезный сигнал, шум и выброс вместе

Полезный сигнал – то, что мы хотим узнать. **Шум** – мелкая дрожь поверх него. **Выброс** – одиночное число, которое выскочило далеко от остальных.

Поставь робота перед стеной и не трогай. Дальномер будет показывать не «40, 40, 40», а что-то вроде «40, 41, 39, 40, 42, 40». Стена стоит – дрожат показания. Эта мелкая случайная дрожь называется **шумом**. Он есть у любого датчика. Датчик линии на ходу трясётся вместе с роботом, на него падает свет из окна и тень от руки. Моторы наводят помехи на провода питания. Даже сама «лесенка» `analogRead` добавляет своё: число может прыгать на деление туда-сюда, когда напряжение стоит почти на границе ступеньки.

Выброс – совсем другая история. Это одно, реже два показания, которые выскочили далеко от соседей: 40, 41, 0, 40. У ультразвукового дальномера так бывает, когда эхо не вернулось: звук утонул в шторе или ушёл от угла в сторону. Тогда `pulseIn` не дожждётся эха и вернёт 0. Бывает и наоборот – звук отразился от дальней стены, и вместо 40 см выходит 230. Или рука товарища на миг попала под луч. Датчик линии ловит выброс на соринке или на шве между листьями поля.

Как понять, что число – выброс? Спроси себя, мог ли робот так быстро переместиться. Пусть он едет со скоростью 0,4 м/с и меряет расстояние раз в 0,1 с. За это время он проезжает $0,4 \cdot 0,1 = 0,04$ м = 4 см. Значит, соседние показания должны отличаться сантиметра на четыре. Если одно отличается на 30 см, оно врёт. Отсюда правило, которым пользуемся в задачах главы:

отсчёт – выброс, если он отличается от обоих соседей больше, чем на допуск. Допуск дают в условии. Если не дали – бери чуть больше, чем робот может проехать между отсчётами, с запасом на шум.

Пример. Показания 40, 41, 0, 40, 42, допуск 10 см. Третий отсчёт отличается от левого соседа на 41, от правого – на 40: выброс. Четвёртый (40) отличается от третьего тоже на 40, зато от пятого – всего на 2. Одной стороной он касается выброса, другой – нормального числа, поэтому сам он в порядке. Слово «обоих» в правиле для этого и стоит.

Что делать с найденным выбросом? Можно просто не использовать его. Можно заменить средним соседей: $(41 + 40) : 2 = 40,5$. А можно поставить фильтр, который сам с ним разберётся, – про это следующий раздел.

Шум страшен не везде. У датчика линии над белым около 870, над чёрным около 110, шум – плюс-минус 20. Вдали от порога такая дрожь ни на что не влияет: 850 и 890 – одинаково «белое». Опасна она у самого порога, когда датчик съезжает на край линии, – там каждое вздрагивание может перекинуть число на другую сторону (раздел 8.5).

Запомни. Шум – мелкая дрожь во всех отсчётах. Выброс – одиночный скачок далеко от соседей. Выброс ищем сравнением с обоими соседями и допуском: сколько робот мог проехать между отсчётами, плюс запас.

8.4. Сглаживание: скользящее среднее и медиана

НА ПАЛЬЦАХ

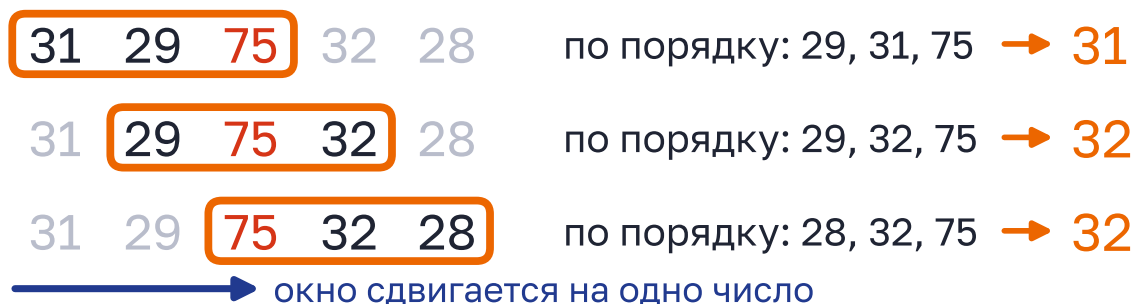


Рисунок 8.6 – Окно из трёх чисел ползёт по ряду показаний

В каждом положении окна получаем одно число. У медианы это середина трёх чисел, выстроенных по порядку. Выброс 75 – всегда самый большой, посередине он не окажется.

Фильтр – правило, которое из ряда показаний делает новый, более спокойный ряд. Два самых ходовых фильтра смотрят в **окно** – на несколько соседних отсчётов.

Скользящее среднее по трём: складываем три отсчёта подряд и делим на 3. Окно ставим так, чтобы нужная точка оказалась в его середине (это **окно с центром**, так считают в таблице и в курсе), а результат пишем под средним отсчётом тройки. Потом окно сдвигается на одно число: самый старый отсчёт выпадает, новый входит, снова считаем среднее. Окно скользит по ряду, отсюда и название. Возьмём показания дальномера 31, 29, 75, 32, 28. Результаты стоят под вторым, третьим и четвёртым отсчётами:

- $(31 + 29 + 75) : 3 = 135 : 3 = 45$;
- $(29 + 75 + 32) : 3 = 136 : 3 \approx 45,3$;
- $(75 + 32 + 28) : 3 = 135 : 3 = 45$.

Первому и последнему отсчёту окно с центром не хватает: у первого нет соседа слева, у последнего – справа. Поэтому значений получается на два меньше, чем отсчётов. Мелкую дрожь среднее гасит хорошо: плюсы и минусы шума в сумме почти сокращаются. С выбросом хуже. 75 никуда не делся – он размазался по трём значениям, и вместо честных 30 см мы получили три раза по 45.

Медианный фильтр по трём: три отсчёта подряд выстраиваем по возрастанию и берём средний по порядку. Такое число называют **медианой**. Из 31, 29, 75 по порядку выходит 29, 31, 75, медиана – 31. Дальше – 32 и 32. Одиночный выброс в окне всегда оказывается с края, самым большим или самым маленьким, поэтому медиана его просто не замечает. Если же выбросов в окне из трёх два, медиана уже не поможет: тогда берут окно пошире, из пяти.

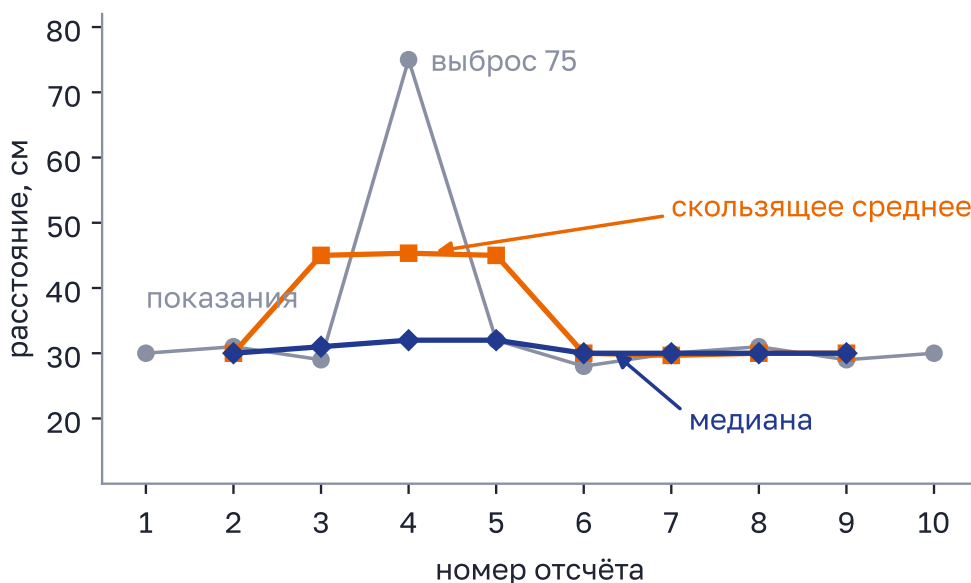


Рисунок 8.7 – Показания дальномера у стены (серые), скользящее среднее по трём (оранжевые) и медиана по трём (синие). Сглаженное значение стоит под средним отсчётом своего окна

Сколько получится сглаженных значений? Крайние отсчёты окну из трёх не по зубам – у них нет соседа с одной стороны, поэтому из n отсчётов выходит $n - 2$ значения. Окно из пяти даёт $n - 4$, окно из k – $n - k + 1$.

Теперь о цене. Окну с центром нужны отсчёты «из будущего», а робот на ходу их ещё не знает. Программа, которая работает в реальном времени, берёт окно «назад»: последние три отсчёта, а результат записывает под самым последним. Такой фильтр **опаздывает**. Робот подъезжает к стене: 50, 45, 40, 35, 30. Среднее трёх последних на третьем отсчёте – $(50 + 45 + 40) : 3 = 45$. Это расстояние, каким оно было отсчёт назад. Окно «назад» из трёх отстаёт на один отсчёт, из пяти – на два. Медиана трёх на ровном подъезде тоже выдаёт средний из трёх – то есть тоже прошлый отсчёт. Окно с центром не опаздывает: то же среднее $(50 + 45 + 40) : 3 = 45$ в таблице стоит под вторым отсчётом, где расстояние и правда 45. Чем шире окно, тем спокойнее сигнал и тем сильнее он запаздывает у фильтра «назад». Для робота, который тормозит у стены, это лишние сантиметры пути. Поэтому в задачах всегда смотри, какое окно названо: с центром (как в таблице) или «назад» (как в программе робота).

Вот как оба фильтра выглядят на Python:

```
d = [31, 29, 75, 32, 28]           # показания дальногомера, см

srednee = []
mediana = []
for i in range(1, len(d) - 1):    # окно с центром: отсчёт и по соседю с каждой стороны
    okno = d[i - 1 : i + 2]
    srednee.append(round(sum(okno) / 3, 1))
    mediana.append(sorted(okno)[1]) # середина после сортировки

print(srednee)                    # [45.0, 45.3, 45.0]
print(mediana)                   # [31, 32, 32]
```

- `d[i - 1 : i + 2]` – кусок списка с номера `i - 1` по номер `i + 1`. Правая граница в Python в кусок не входит, поэтому пишем `i + 2`.
- `sorted(okno)` расставляет три числа по возрастанию, `[1]` берёт второе из них: нумерация в Python начинается с нуля.
- `append` дописывает число в конец списка.

А так медиана трёх живёт в скетче Arduino. Здесь работает окно «назад»: будущих показаний у платы нет.

```
int a = 0, b = 0, c = 0;           // три последних показания: a – самое старое

int median3(int x, int y, int z) {
    if ((x ≤ y && y ≤ z) || (z ≤ y && y ≤ x)) return y; // y посередине
    if ((y ≤ x && x ≤ z) || (z ≤ x && x ≤ y)) return x; // x посередине
    return z; // иначе – z
}

void setup() {
    Serial.begin(9600);
}

void loop() {
    a = b; // окно сдвигается: самое старое выпадает
    b = c;
    c = analogRead(A0); // новое показание
    Serial.println(median3(a, b, c));
    delay(20);
}
```

`&&` значит «и», `||` – «или». Первая проверка читается так: «у не меньше x и не больше z – или наоборот». Первые два числа в мониторе порта получатся странными: окно ещё хранит нули, с которых начинались `a`, `b` и `c`. С третьего отсчёта всё честно.

Скользящее среднее – самый простой **фильтр низких частот**. «Низкие частоты» – это медленные изменения: робот подъезжает к стене, съезжает с линии. Их фильтр пропускает. Быстрые – дрожь от отсчёта к отсчёту – гасит. Почему это так называется, станет понятно в разделе 8.6.

Запомни. Скользящее среднее гасит дрожь, но размазывает выброс. Медиана убирает одиночный выброс целиком. Окно из k отсчётов даёт из n отсчётов $n - k + 1$ значений. Окно с центром стоит под средним отсчётом и не опаздывает, но ему нужны соседи с обеих сторон. Окно «назад» (так считает робот на ходу) опаздывает: на один отсчёт при окне из трёх, на два при окне из пяти.

Совет. Если в задаче спрашивают номер отсчёта, подпиши номера над рядом чисел, а сглаженное значение ставь под средним отсчётом его окна (у окна «назад» – под последним). Снять на единицу здесь проще всего.

8.5. Детектирование: по уровню и по периоду

НА ПАЛЬЦАХ



Рисунок 8.8 – Считаем моменты, когда сигнал нырнул под порог

Пока сигнал под порогом, это всё тот же перекрёсток. Следующий засчитываем, только когда сигнал сначала вынырнул, а потом снова нырнул.

Детектирование – обнаружение в сигнале нужного события: перекрёстка, препятствия, вспышки маяка. Можно смотреть, насколько высок сигнал, – это детектирование по уровню. А можно следить за его ритмом – это по периоду.

По уровню. Выбираем **порог** и следим, с какой он стороны. Для датчика линии порог считают по калибровке (глава 7), для дальномера это расстояние «пора тормозить», например 20 см. Событие – это **пересечение** порога, причём в какую сторону, не важно: сигнал стал сильнее порога или слабее. Какое пересечение считать в задаче, говорит условие. А пребывание за порогом – всё ещё то же событие. Датчик стоит над перекрёстком 5–6 отсчётов подряд. Если прибавлять к счётчику 1 на каждом тёмном отсчёте, один перекрёсток превратится в шесть. Поэтому прибавляем только тогда, когда предыдущий отсчёт был светлым, а этот – тёмный. Такой переход называют **фронтом**.

Ложное срабатывание – детектор сказал «событие!», а его не было. Чаще всего виноват выброс: соринка на белом поле даёт один тёмный отсчёт. Хитрее шум у порога. Когда датчик медленно съезжает на край линии, показание подходит к порогу и дрожит возле него: 470, 430, 480, 440. Каждый нырок под порог счётчик примет за новый перекрёсток.

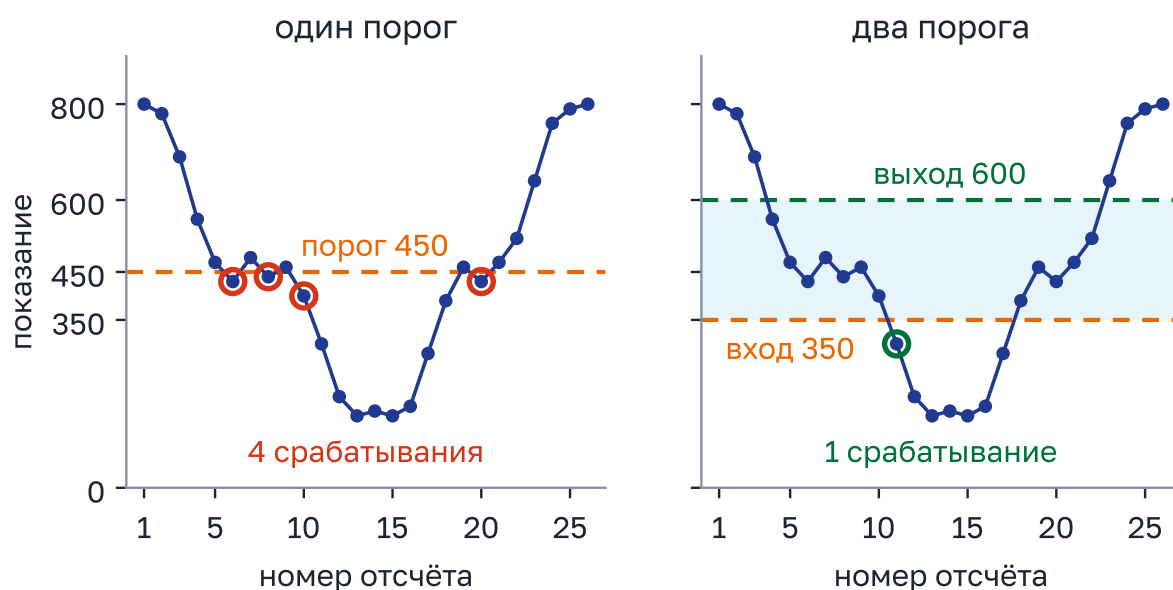


Рисунок 8.9 – Датчик медленно въезжает на линию и съезжает с неё. Слева порог один – четыре срабатывания вместо одного. Справа два порога: «вход» ниже 350, «выход» выше 600 – срабатывание одно

Против ложных срабатываний есть три средства.

- **Выдержка.** Событие засчитываем, только если сигнал за порогом не меньше N отсчётов подряд. N выбирают по размерам. Робот едет 0,3 м/с, отсчёты каждые 10 мс – шаг 3 мм. Перекрёсток шириной 18 мм даёт 6 тёмных отсчётов, соринка в 3 мм – один. $N = 3$ надёжно их разделяет.
- **Два порога.** «Вошли в перекрёсток» – когда показание ниже нижнего порога, «вышли» – когда выше верхнего. Между порогами ничего не меняем, просто помним, где мы сейчас. Дрожь около середины не дотягивается ни до одного порога. Инженеры называют это **гистерезисом**. Так работает термостат в холодильнике: мотор включается, когда внутри стало теплее одной отметки, а выключается, когда стало холоднее другой. Был бы порог один – мотор щёлкал бы каждую минуту.
- **Сгладить до порога.** Медиана из раздела 8.4 выбросит одиночную соринку ещё до того, как число сравнят с порогом.

Выдержка в скетче – это второй счётчик, который считает тёмные отсчёты подряд:

```

int dark = 0; // сколько отсчётов подряд под порогом
int count = 0; // сколько перекрёстков насчитали

void setup() {
  Serial.begin(9600);
}

void loop() {
  if (analogRead(A0) < 450) {
    dark = dark + 1;
    if (dark == 3) { // третий тёмный отсчёт подряд – это перекрёсток
      count = count + 1;
      Serial.println(count);
    }
  } else {
    dark = 0; // светлый отсчёт – считаем заново
  }
  delay(10);
}

```

Присмотрись к `dark == 3`. На перекрёстке `dark` дорастёт до 6, но условие верно лишь в тот единственный момент, когда тёмных отсчётов стало три. Напиши `dark ≥ 3` – и один перекрёсток даст четыре прибавки: на третьем, четвёртом, пятом и шестом отсчёте. А соринка сбросит `dark` в ноль следующим же светлым отсчётом и до трёх не доберётся.

По периоду. Иногда важнее высоты сигнала его ритм.

НА ПАЛЬЦАХ

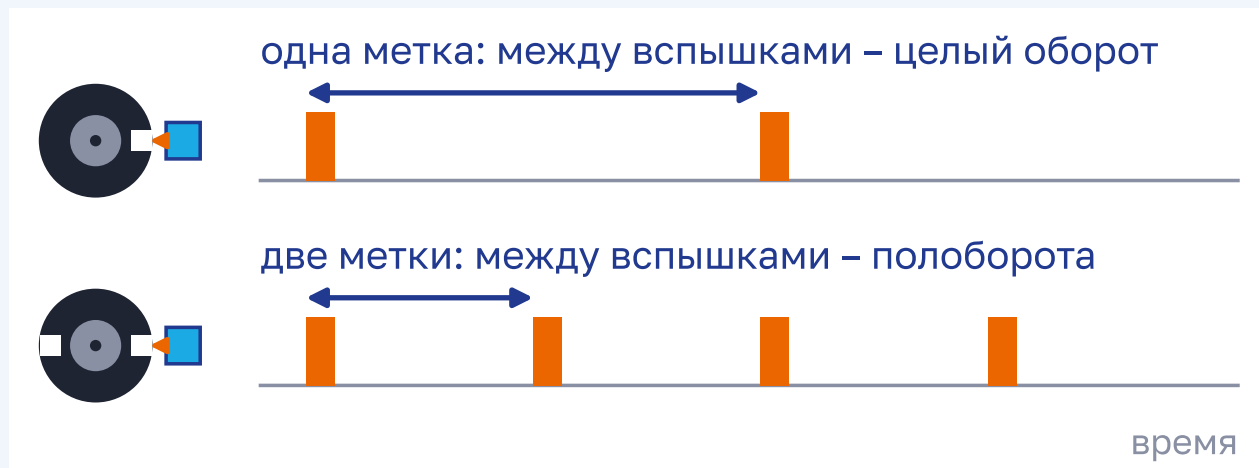


Рисунок 8.10 – Датчик смотрит на шину колеса с белыми метками

Каждая метка, пробегая мимо датчика, даёт вспышку. Одна метка – одна вспышка за оборот. Две метки – две вспышки, и между ними только полоборота.

На чёрную шину колеса клеят белую метку, рядом ставят датчик линии. Раз за оборот метка пробежит мимо датчика, и в сигнале появляется короткая вспышка. Время от начала одной вспышки до начала следующей – период, оно же время одного оборота. Частота $f = 1:T$ – сколько оборотов колесо делает за секунду. За оборот колесо проезжает длину своей окружности $\pi \cdot D$, поэтому скорость работы

$$v = \pi \cdot D \cdot f.$$

Пример. Вспышки идут каждые 0,5 с, колесо диаметром 65 мм. Частота $1 : 0,5 = 2$ оборота в секунду, за оборот $3,14 \cdot 65 \approx 204$ мм, скорость $204 \cdot 2 \approx 408$ мм/с, то есть около 0,41 м/с.

Меток может быть и несколько, расставленных по шине на равных расстояниях. Тогда вспышек за оборот столько же, сколько меток, и промежутков между соседними вспышками – только часть оборота. Две метки друг напротив друга: оборот длится два промежутка. Четыре метки – четыре промежутка. Сначала найди время целого оборота, потом считай скорость.

Так же робот отличает маяки. На поле два маяка одинаковой яркости, один мигает 2 раза в секунду, другой 5. По уровню их не различить. По периоду – легко: у первого 0,5 с, у второго 0,2 с.

Запомни. По уровню считаем пересечения порога, а не отсчёты за ним. От ложных срабатываний спасают выдержка, два порога и сглаживание. По периоду меряем время между началами соседних событий: $f = 1 : T$, скорость колеса $v = \pi \cdot D \cdot f$. Если меток на колесе несколько, оборот – это столько промежутков, сколько меток.

Совет. Три вспышки – это два периода, пять столбов забора – четыре пролёта. Считай промежутки между событиями, а не сами события.

8.6. Спектр и огибающая – на пальцах

НА ПАЛЬЦАХ

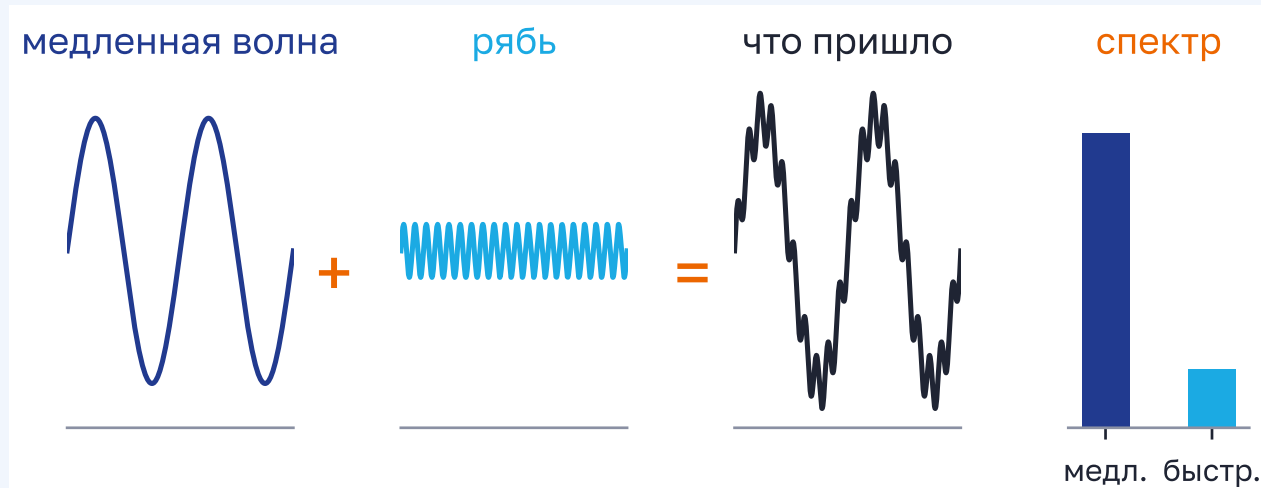


Рисунок 8.11 – Медленная волна и мелкая рябь складываются в один сигнал

Сложи медленную волну и мелкую рябь – получится то, что пришло с датчика. Спектр разбирает сигнал обратно: на каждое колебание по столбику, и чем колебание сильнее, тем столбик выше.

Постой на берегу Обского моря в ветреный день. По воде катятся большие медленные волны, а по ним бежит мелкая рябь. Поплавок слушается обеих: плавно ходит вверх-вниз вместе с

волной и мелко дрожит от ряби. Запиши его высоту – получится график, как «что пришло» на рисунке. **Спектр** сигнала отвечает на вопрос, из каких колебаний этот сигнал сложен и насколько сильно каждое. Его рисуют столбиками: слева медленные колебания (низкие частоты), справа быстрые (высокие), высота столбика – амплитуда колебания (на сколько оно отходит от середины). Прыгающие столбики эквалайзера в музыкальном плеере – это спектр песни: слева басы, справа высокие звуки.

У сигнала датчика тоже есть спектр. Полезное меняется медленно, за десятые доли секунды и секунды: робот подъезжает к стене, съезжает с линии. Шум и выбросы скачут от отсчёта к отсчёту, они быстрые. Значит, полезное сидит в левой части спектра, шум – в правой. **Фильтр низких частот** пропускает левую часть и гасит правую. Скользящее среднее из раздела 8.4 так и работает. Рябь, которая за время окна успевает сходить вверх и вниз, в сумме даёт почти ноль. А если окно по длине совпадает с одним качанием ряби, она гасится целиком.

НА ПАЛЬЦАХ

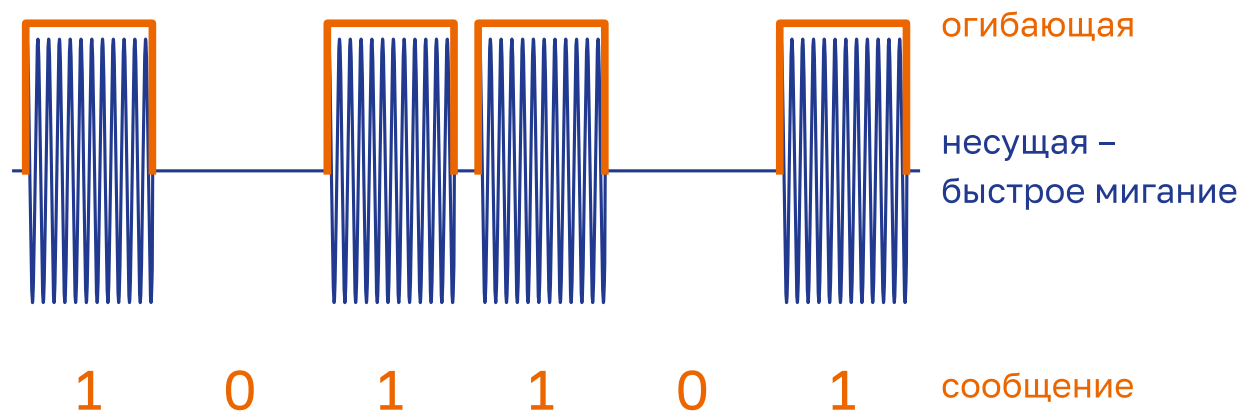


Рисунок 8.12 – Пульт мигает очень быстро, короткими пачками

Быстрое мигание – несущая. Где пачка есть, пишем 1, где её нет – 0. Обводка пачек – огибающая, в ней и едет сообщение.

Теперь обратная задача: передать по сигналу сообщение. Проще всего мигать, как фонариком по азбуке Морзе. Но вокруг ИК-приёмника работа полно чужого инфракрасного света: солнце, лампы, обогреватель. Чтобы свой сигнал не спутать с чужим, пульт мигает очень быстро – у многих пультов около 38 000 раз в секунду, 38 кГц (килогерц: 1 кГц = 1000 Гц). Приёмник настроен откликаться только на такое мигание.

Это быстрое ровное мигание называют **несущей**: оно несёт сообщение, как грузовик везёт груз. Само сообщение спрятано в том, где несущая есть, а где её нет. Линию, которая обводит верхушки несущей, называют **оглибающей**. У пульта она похожа на забор из ступенек: пачка – тишина – пачка. Менять несущую так, чтобы в ней поехало сообщение, называют **модуляцией**.

Ультразвуковой дальномер из главы 5 устроен похоже: восемь щелчков по 40 кГц – это пачка несущей, а её огибающая – простая ступенька «есть звук – нет звука».

Для любопытных: как радио прячет в волне музыку. Для задач главы это не понадобится. Радиостанция тоже шлёт несущую, только меняет её плавно. Если менять амплитуду (размах), модуляция называется амплитудной, если частоту – частотной; буквы FM на радиоприёмнике означают частотную. При амплитудной огибающая похожа на волну:

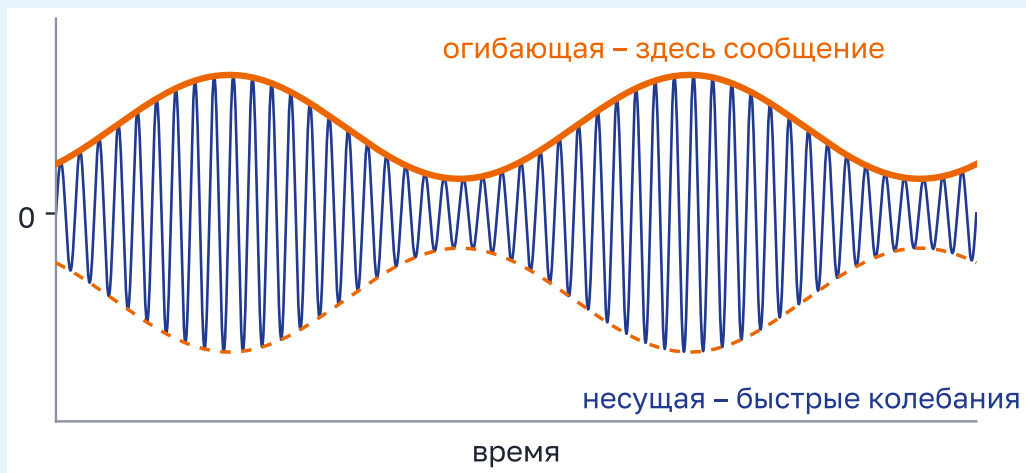


Рисунок 8.13 – Амплитудная модуляция. Синее – быстрая несущая, оранжевое – огибающая, по которой приёмник читает сообщение

Чтобы вытащить огибающую, приёмник сначала берёт модуль сигнала: все отрицательные значения переворачивает вверх. В таблице это формула $y = \text{ABS}(x)$. Потом он сглаживает результат фильтром низких частот, например скользящим средним, у которого окно равно ровно одному периоду несущей. Быстрая несущая исчезает, медленная огибающая остаётся. Высота огибающей после усреднения получается ниже исходных вершушек: форма восстанавливается точно, а высота меньше.

Для любопытных: число из пачек. Это нужно для задач со звёздочкой про пульт (8.6, 8.11, 8.12). Нули и единицы из окошек складываются в **двоичное** число – так компьютеры записывают любые числа. У каждого места в записи – **разряда** – есть свой «вес»: справа налево 1, 2, 4, 8, 16 и так далее, каждый вдвое больше соседа. Чтобы перевести число в обычное, складываем веса там, где стоят единицы. Запись 0110: единицы стоят на местах с весами 4 и 2, получается $4 + 2 = 6$. Самый тяжёлый разряд называют старшим, самый лёгкий – младшим. С какого конца идёт запись, говорит условие. Перепутаешь порядок – получишь другое число.

Запомни. Спектр показывает, каких колебаний в сигнале больше – медленных или быстрых. Полезное у датчика медленное, шум быстрый, поэтому фильтр низких частот чистит сигнал. Несущая – быстрое ровное мигание, огибающая – линия по её вершушкам; сообщение едет в огибающей.

8.7. Разбираем задачи

Задача 8.1 Сколько миллиметров между взглядами

разминка · дискретизация, частота опроса, путь между отсчётами

Робот едет по полю со скоростью 0,25 м/с. В `loop` программа читает датчик линии и ждёт `delay(40)`; остальные строки выполняются так быстро, что их время не считаем. а) Сколько отсчётов в секунду снимает программа? б) Сколько миллиметров проезжает робот между двумя отсчётами? Ответ – два числа.

Дано: $v = 0,25$ м/с; $\Delta t = 40$ мс.

Найти: $f_{\text{опр}}$ – ?; s – ?

Решение

Шаг 1. Промежуток в секундах. $1 \text{ с} = 1000 \text{ мс}$, поэтому $40 \text{ мс} = 40 : 1000 = 0,04 \text{ с}$.

Шаг 2. Частота опроса. $f_{\text{опр}} = 1 : 0,04 = 25$ отсчётов в секунду. Проверка без дробей: в секунде 1000 мс, по 40 мс их помещается $1000 : 40 = 25$.

Шаг 3. Скорость в миллиметрах в секунду. $1 \text{ м} = 1000 \text{ мм}$, поэтому $0,25 \text{ м/с} = 0,25 \cdot 1000 = 250 \text{ мм/с}$.

Шаг 4. Путь между отсчётами. $s = v \cdot \Delta t = 250 \cdot 0,04 = 10 \text{ мм}$.

Сантиметр между взглядами – для полосы шириной 18 мм это впритык: $18 : 10 = 1,8$, наверняка на неё попадёт только один отсчёт. Если программа ждёт три тёмных отсчёта подряд (раздел 8.5), такой робот перекрёсток может не засчитать.

⚠ Типичная ошибка. Посчитать $0,25 \cdot 0,04 = 0,01$ и записать «0,01 мм». Скорость была в метрах в секунду, значит, и путь вышел в метрах: 0,01 м – это 10 мм. Вторая ошибка – решить, что `delay(40)` означает «40 раз в секунду». В скобках стоит пауза в миллисекундах, а не число отсчётов.

Ответ: а) 25; б) 10.

Своя задача по программе сферы, модуль 3

Реши сам

8.1. Робот едет со скоростью 0,32 м/с, датчик читают каждые 20 мс. а) Сколько отсчётов в секунду? б) Сколько миллиметров робот проезжает между двумя отсчётами?

8.2. Робот едет по полю со скоростью 0,3 м/с. Поперёк пути нарисованы полосы шириной 15 мм. Программа засчитывает полосу, если увидела её не меньше трёх отсчётов подряд. С какой наименьшей частотой опроса (отсчётов в секунду) нужно читать датчик, чтобы на любую полосу наверняка пришлось хотя бы три отсчёта?

Задача 8.2 Найди выбросы

разминка · выброс, сравнение с соседями, допуск

Робот едет к коробке со скоростью 0,4 м/с, дальномер меряет расстояние каждые 0,1 с. Вот двенадцать показаний подряд, в сантиметрах: 96, 92, 88, 85, 0, 77, 73, 31, 65, 62, 57, 53. Отсчёт считаем выбросом, если он отличается от обоих соседей больше чем на 10 см. Какие отсчёты – выбросы? В ответ введи их номера подряд, по возрастанию.

Дано: $v = 0,4$ м/с; $\Delta t = 0,1$ с; 12 показаний; допуск 10 см.

Найти: номера выбросов.

Решение

Шаг 1. Разумен ли допуск. Между отсчётами робот проезжает $0,4 \cdot 0,1 = 0,04$ м = 4 см. Честные соседи должны отличаться сантиметра на четыре, допуск 10 см берёт это с запасом на шум.

Шаг 2. Подпишем номера и разности соседей. Номер – показание: 1 – 96, 2 – 92, 3 – 88, 4 – 85, 5 – 0, 6 – 77, 7 – 73, 8 – 31, 9 – 65, 10 – 62, 11 – 57, 12 – 53.

Разности соседних пар по порядку: 4, 4, 3, 85, 77, 4, 42, 34, 3, 5, 4. Большие скачки два: вокруг пятого отсчёта (85 и 77) и вокруг восьмого (42 и 34).

Шаг 3. Проверим подозреваемых по правилу «от обоих соседей».

- Пятый (0): от четвёртого отличается на 85, от шестого – на 77. Оба больше 10 – выброс.
- Четвёртый (85): от пятого – на 85, но от третьего – всего на 3. Не выброс.
- Шестой (77): от пятого – на 77, но от седьмого – на 4. Не выброс.
- Восьмой (31): от седьмого – на 42, от девятого – на 34. Выброс.
- Седьмой (73) и девятый (65) с другой стороны соседствуют с нормальными числами (77 и 62). Не выбросы.

Шаг 4. Что это было. Ноль – эхо не вернулось, pulseIn отдал 0. 31 см – что-то мелькнуло под лучом, например пробежал кот. Настоящие расстояния в эти моменты были около $(85 + 77) : 2 = 81$ и $(73 + 65) : 2 = 69$ см: ряд падает на 4 см за отсчёт.

 **Типичная ошибка.** Записать в выбросы всех, кто стоит рядом со скачком: 4, 5, 6, 7, 8, 9. У каждого скачка два конца, но выброс – только тот, кто прыгает сразу в обе стороны. Его соседи с другой стороны касаются нормальных чисел.

Ответ: 58 (отсчёты 5 и 8).

Своя задача по программе сферы, модуль 3

Реши сам

8.3. Робот медленно подъезжает к стене, дальномер меряет каждые 0,1 с. Показания в сантиметрах: 52, 50, 49, 47, 46, 260, 43, 41, 40, 0, 37, 35. Выброс – отсчёт, который отличается от обоих соседей больше чем на 10 см. Назови номера выбросов через пробел.

8.4. Робот отъезжает от стены, дальномер показывает (см): 20, 25, 30, 36, 18, 45, 50, 55, 97, 65. Допуск тот же, 10 см. Найди номера выбросов и замени каждый выброс средним его соседей. Ответ: два номера и два новых значения.

Задача 8.3 Сгладь показания дальномера

скользящее среднее, окно из трёх, размах, сглаживание

Робот стоит перед коробкой. Дальномер восемь раз подряд показал (см): 42, 37, 41, 42, 40, 38, 42, 43. Программа сглаживает показания скользящим средним по трём отсчётам (окно с центром: значение стоит под средним отсчётом). а) Сколько сглаженных значений получится и какие они? б) Каким был размах до сглаживания и после? Во сколько раз он уменьшился?

Дано: 8 показаний; окно из 3 отсчётов.

Найти: сглаженные значения; размах до и после.

Решение

Шаг 1. Сколько значений. Окно из трёх: $n - 2 = 8 - 2 = 6$ значений. Первое стоит под вторым отсчётом, последнее – под седьмым.

Шаг 2. Считаем окна. Каждое следующее окно сдвинуто на одно число.

Таблица 8.1

Окно (номера)	Числа	Сумма	Среднее
1–3 (под 2)	42, 37, 41	120	40
2–4 (под 3)	37, 41, 42	120	40
3–5 (под 4)	41, 42, 40	123	41
4–6 (под 5)	42, 40, 38	120	40
5–7 (под 6)	40, 38, 42	120	40
6–8 (под 7)	38, 42, 43	123	41

Сумму можно не складывать каждый раз заново: из прошлой суммы вычитаем выпавшее число и прибавляем вошедшее. Второе окно: $120 - 42 + 42 = 120$. Третье: $120 - 37 + 40 = 123$.

Шаг 3. Размах до. Самое большое показание 43, самое маленькое 37: $43 - 37 = 6$ см.

Шаг 4. Размах после. Сглаженные значения от 40 до 41: $41 - 40 = 1$ см.

Шаг 5. Во сколько раз. $6: 1 = 6$.

Робот стоит примерно в 40 см от коробки, и фильтр выдаёт 40–41. Дрожь в 6 см ужалась до сантиметра.

 **Типичная ошибка.** Разрезать ряд на отдельные тройки: (42, 37, 41), (42, 40, 38), (42, 43) – и получить два значения с огрызком. Окно скользит, сдвигаясь каждый раз на одно число, и тройки перекрываются.

Ответ: а) 6 значений: 40, 40, 41, 40, 40, 41; б) 6 см и 1 см, в 6 раз.

Своя задача по программе сферы, модуль 3

Реши сам

8.5. ИК-дальномер робота, который стоит на месте, девять раз показал (см): 29, 21, 28, 26, 27, 22, 29, 27, 25. Показания сглаживают скользящим средним по трём отсчётам (окно с центром). Сколько получится сглаженных значений? Каков размах до сглаживания и после? Во сколько раз он уменьшился?

8.6. Дальномер стоит на тонкой мачте, и на ходу мачта качается: каждые три отсчёта одно качание. Робот едет к стене, показания (см): 63, 55, 56, 57, 49, 50, 51, 43, 44. Найди скользящее среднее по трём отсчётам (окно с центром). На сколько сантиметров робот приближается к стене за один отсчёт?

Задача 8.4 Сколько перекрёстков на графике

детектирование по уровню, выдержка, ложные срабатывания, перекрёстки

Робот едет по линии. Датчик стоит сбоку от линии и смотрит на белое поле; на перекрёстке поперечная линия проходит под ним. Показания, снятые каждые 10 мс, – на рисунке. Ниже порога 450 – чёрное. На поле попадают соринки и швы между листами: они дают 1–2 тёмных отсчёта подряд. Поэтому программа засчитывает перекрёсток, только если под порогом **не меньше трёх отсчётов подряд**. а) Сколько перекрёстков проехал робот? б) Сколько насчитал бы счётчик без выдержки – тот, что прибавляет 1 при каждом переходе через порог сверху вниз? Ответ – два числа.

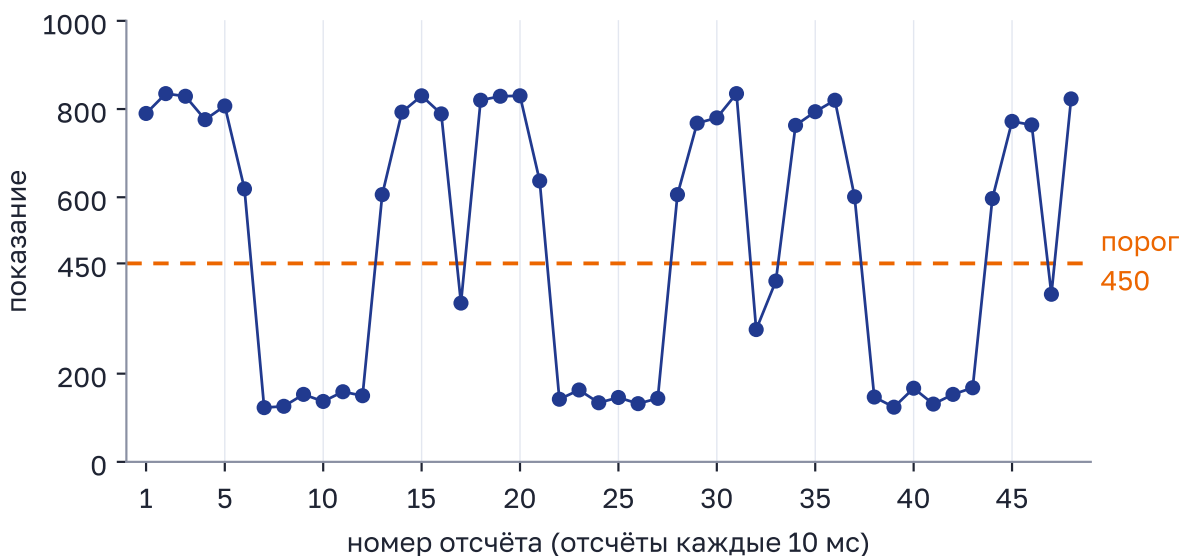


Рисунок 8.14 – Показания датчика к задаче 8.4. Отсчёты каждые 10 мс, пунктир – порог 450

Дано: 48 отсчётов на графике; порог 450; выдержка – 3 отсчёта подряд.

Найти: а) число перекрёстков; б) число срабатываний без выдержки.

Решение

Шаг 1. Находим точки под порогом. Идём по графику слева направо и выписываем номера отсчётов ниже пунктира. Удобно держать линейку вдоль порога: всё, что под ней, – «чёрное».

Шаг 2. Собираем их в серии – куски подряд.

- 7–12: шесть отсчётов;
- 17: один;
- 22–27: шесть;
- 32–33: два (32-й глубоко, около 300, 33-й около 410);
- 38–43: шесть;
- 47: один.

Шаг 3. Применяем выдержку. Серии из трёх и больше отсчётов – 7–12, 22–27 и 38–43. Это три перекрёстка. Одиночки и пара 32–33 – соринки и шов.

Шаг 4. Считаем без выдержки. Такой счётчик срабатывает на каждом нырке под порог, то есть на каждой серии, какой бы короткой она ни была. Серий шесть, значит, и срабатываний шесть.

Все три настоящих перекрёстка одинаковые, по шесть отсчётов: скорость у робота постоянная, поперечные линии одной ширины. У соринки длина разная и маленькая. По этой разнице выдержка и работает.

 **Типичная ошибка.** Засчитать пару 32–33 как перекрёсток, потому что 32-й отсчёт очень тёмный, около 300 – намного ниже порога. Правило говорит о длине серии, глубина тут ни при чём: сколько бы ни показала соринка, два отсчёта – это два отсчёта. Вторая ошибка – посчитать все тёмные точки (22 штуки) вместо серий.

Ответ: а) 3; б) 6.

По мотивам задачи первого отборочного этапа НТО 2024/25, профиль «Интеллектуальные робототехнические системы», инженерный тур, «Песок и камни»

Реши сам

8.7. Порог 500, ниже – чёрное. Перекрёсток засчитывается, если под порогом не меньше трёх отсчётов подряд. Показания подряд: 810, 790, 230, 180, 200, 170, 820, 800, 760, 310, 780, 800, 200, 190, 160, 810, 790, 350, 330, 780, 800, 150, 170, 140, 160, 820. Сколько перекрёстков проехал робот и сколько насчитал бы счётчик без выдержки? Ответ – два числа.

8.8.* Датчик линии шумит у края линии. Программа с одним порогом 450 прибавляет 1 при каждом переходе сверху вниз. Программа с двумя порогами считает «вошли в перекрёсток», когда показание стало меньше 350, и «вышли», когда оно стало больше 600; между порогами она ничего не меняет. Показания: 810, 470, 430, 490, 410, 200, 190, 440, 480, 420, 800, 460, 380, 160, 170, 520, 430, 810, 430, 470, 200, 180, 470, 420, 800. Сколько перекрёстков насчитает первая программа и сколько вторая? Ответ – два числа.

Задача 8.5 Метка на колесе

детектирование по периоду, частота, длина окружности, скорость

На чёрную шину колеса диаметром 64 мм наклеили белую метку. Рядом поставили датчик линии: над шиной он показывает около 130, над меткой – около 850. Датчик читают каждые 10 мс, на рисунке 100 отсчётов подряд. Робот едет с постоянной скоростью, колёса не проскальзывают. а) Сколько оборотов в секунду делает колесо? б) С какой скоростью едет робот? Ответ в м/с, округли до сотых.



Рисунок 8.15 – Показания датчика у колеса к задаче 8.5. Подписаны номера отсчётов, где видна метка

Дано: $D = 64$ мм; отсчёты каждые 10 мс; метка видна в отсчётах 13–14, 53–54, 93–94.

Найти: f – ?; v – ?

Решение

Шаг 1. Начала вспышек. Метка появляется в отсчётах 13, 53 и 93.

Шаг 2. Период в отсчётах. $53 - 13 = 40$ и $93 - 53 = 40$. Промежутки одинаковые – колесо крутится с постоянной скоростью.

Шаг 3. Период в секундах. $40 \cdot 10 = 400$ мс = 0,4 с. Это время одного оборота.

Шаг 4. Обороты в секунду. $f = 1 : 0,4 = 2,5$ оборота в секунду.

Шаг 5. Путь за оборот. Длина окружности $\pi \cdot D = 3,14159 \cdot 64 \approx 201,1$ мм.

Шаг 6. Скорость. $v = 201,1 \cdot 2,5 \approx 502,7$ мм/с. В метрах: $502,7 : 1000 \approx 0,50$ м/с.

Полметра в секунду – обычная скорость школьного робота на моторах с редуктором: поле в 2 м он проедет за 4 с.

⚠ Типичная ошибка. «На графике секунда (100 отсчётов по 10 мс) и три вспышки – значит, 3 оборота в секунду». Три вспышки отделяют только два полных оборота, а первая вспышка пришлась не на начало записи. Считай промежутки между вспышками. Ещё одна – подставить в $\pi \cdot D$ радиус 32 мм вместо диаметра: скорость выйдет вдвое меньше, 0,25 м/с.

Ответ: а) 2,5; б) 0,50 м/с.

Своя задача по программе сферы, модуль 3

Реши сам

8.9. На поле три маяка одинаковой яркости: первый мигает 2 раза в секунду, второй – 4 раза, третий – 5 раз. Датчик робота записал, когда начинались вспышки: 120 мс, 370 мс, 620 мс, 870 мс. Какой маяк видит робот? Введи его номер.

8.10. На шину колеса диаметром 66 мм наклеили **две** белые метки друг напротив друга. Датчик читает каждые 20 мс, вспышки идут через каждые 15 отсчётов. С какой скоростью едет робот? Ответ в м/с, округли до сотых.

Задача 8.6^{*}. Что передал пульт

несущая, огибающая, частота, двоичный код

Робот принимает команды с инфракрасного пульта. Пульт посылает стартовую пачку длиной 2 мс, потом паузу 1 мс, потом восемь окошек по 1 мс. В окошке есть пачка несущей – это 1, нет – 0. Первым идёт старший разряд, с весом 128. На рисунке – сигнал на входе приёмника; колебания несущей нарисованы реже, чем бывает у настоящего пульта. **Осциллограф** – прибор, который рисует сигнал на экране, – насчитал в одной пачке длиной 1 мс 38 колебаний. а) Какая частота несущей в кГц? б) Какое число передал пульт? Ответ – два числа.

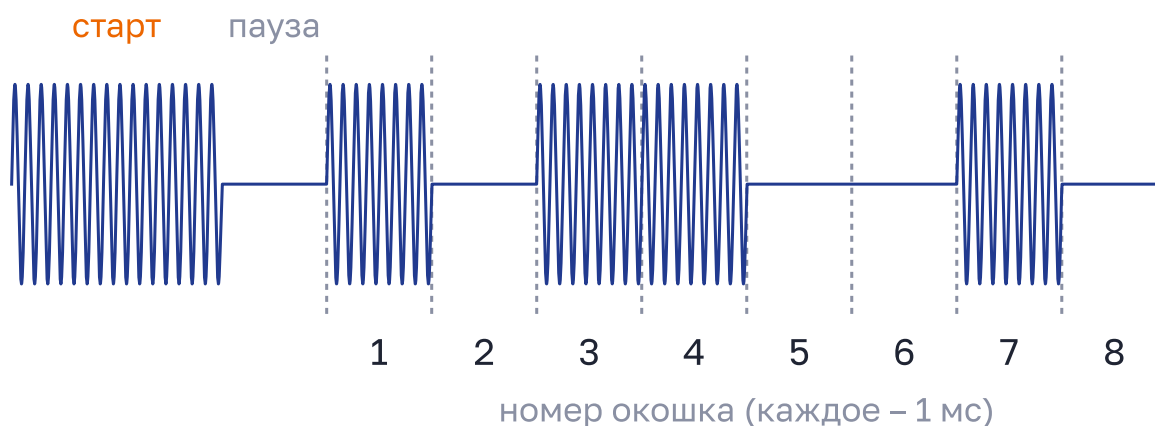


Рисунок 8.16 – Сигнал ИК-пульта к задаче 8.6: старт, пауза и восемь окошек по 1 мс

Дано: 38 колебаний за 1 мс; 8 окошек; старший разряд первым.

Найти: частоту несущей; число.

Решение

Шаг 1. Частота несущей. Частота – сколько колебаний за секунду. 1 мс = 0,001 с, поэтому $f = 38 : 0,001 = 38\,000$ Гц. В кГц: $38\,000 : 1000 = 38$ кГц. Одно колебание длится $1 : 38\,000 \approx 0,000026$ с – около 26 микросекунд.

Шаг 2. Читаем огибающую по окошкам. Старт и паузу пропускаем: это «внимание, начинаю», а не данные. Дальше смотрим на каждое окошко: есть колебания – 1, ровная линия – 0.

Таблица 8.2

Окошко	Пачка	Цифра	Вес
1	есть	1	128
2	нет	0	64
3	есть	1	32
4	есть	1	16
5	нет	0	8
6	нет	0	4
7	есть	1	2
8	нет	0	1

Шаг 3. Складываем веса единиц. $128 + 32 + 16 + 2 = 178$.

Восемь окошек вмещают числа от 0 до 255, и 178 в этот промежуток попадает. А 38 кГц – обычная частота для бытовых ИК-пультов, так что прибор посчитал правдоподобно.

 **Типичная ошибка.** Прочитать окошки справа налево, как будто первым шёл младший разряд: 01001101 – это 77. Порядок разрядов всегда смотри в условии. Другая ошибка – принять стартовую пачку за первую единицу: тогда все цифры съедут на одно окошко. И третья, с единицами: $38:1 = 38$ «Гц» – забыли, что колебания считали за миллисекунду, а не за секунду.

Ответ: а) 38; б) 178.

Своя задача по программе сферы, модуль 3

Реши сам

8.11.* Пульт работает по той же схеме, что в задаче 8.6: старший разряд первым. По окошкам пачки такие: нет, есть, есть, нет, есть, нет, нет, есть. Какое число передал пульт?

8.12.* Другой пульт в каждой пачке длиной 0,5 мс даёт 20 колебаний, а разряды посылает **младшим вперёд**: первое окошко имеет вес 1, второе – 2 и так далее до 128. Пачки по окошкам: есть, есть, нет, нет, есть, нет, нет, нет. Найди частоту несущей в килогерцах и переданное число.

Задача 8.7* *. Три программы тормозят у стены

скользящее среднее, медиана, выброс, запаздывание фильтра, порог

Робот едет к стене со скоростью 0,5 м/с, дальномер меряет раз в 0,1 с. Робот должен остановиться, как только расстояние станет **меньше** 30 см; остановку считаем мгновенной. Показания по порядку (см): 80, 75, 70, 65, 60, 55, 50, 0, 40, 35, 30, 25, 20, 15 – на восьмом отсчёте эхо не вернулось. Три программы сравнивают с порогом разное: 1) сами показания; 2) скользящее среднее трёх последних отсчётов; 3) медиану трёх последних отсчётов. Программа работает на ходу и будущих показаний не знает, поэтому окно «назад»: результат стоит под последним отсчётом. Сглаженные значения появляются с третьего отсчёта. На каком отсчёте остановится каждая программа? Ответ – три номера.

Дано: $v = 0,5$ м/с; $\Delta t = 0,1$ с; 14 показаний, восьмое – выброс; стоп, когда значение меньше 30 см.

Найти: номер отсчёта остановки для каждой программы.

Решение

Шаг 1. Настоящие расстояния. Между отсчётами робот проезжает $0,5 \cdot 0,1 = 0,05$ м = 5 см. Ряд падает на 5 см за отсчёт, поэтому на восьмом отсчёте до стены было $50 - 5 = 45$ см.

Шаг 2. Программа 1, сырые показания. Первое число меньше 30 – это 0 на восьмом отсчёте. Робот встанет в 45 см от стены – ложная остановка из-за выброса.

Шаг 3. Программа 2, скользящее среднее. Считаем с третьего отсчёта:

Таблица 8.3

Отсчёт	Три последних	Сумма	Среднее	Меньше 30?
3	80, 75, 70	225	75	нет
4	75, 70, 65	210	70	нет
5	70, 65, 60	195	65	нет
6	65, 60, 55	180	60	нет
7	60, 55, 50	165	55	нет
8	55, 50, 0	105	35	нет
9	50, 0, 40	90	30	нет
10	0, 40, 35	75	25	да

На девятом отсчёте среднее равно 30 – это не меньше 30, робот едет дальше. На десятом 25 – стоп. А настоящее расстояние там 35 см: выброс размазался по трём окнам и утащил среднее вниз раньше времени.

Шаг 4. Программа 3, медиана. Выстраиваем каждую тройку по порядку и берём середину:

Таблица 8.4

Отсчёт	Три последних	По порядку	Медиана
8	55, 50, 0	0, 50, 55	50
9	50, 0, 40	0, 40, 50	40
10	0, 40, 35	0, 35, 40	35

Отсчёт	Три последних	По порядку	Медиана
11	40, 35, 30	30, 35, 40	35
12	35, 30, 25	25, 30, 35	30
13	30, 25, 20	20, 25, 30	25

До восьмого отсчёта медиана идёт вместе с рядом: 75, 70, 65, 60, 55. Первое значение меньше 30 – на тринадцатом отсчёте. Настоящее расстояние там 20 см.

Шаг 5. Кто молодец. Без выброса сырые показания остановили бы робота на двенадцатом отсчёте, в 25 см. Медиана остановила на отсчёт позже – это её запаздывание, робот прокатился лишние 5 см. Зато она единственная не испугалась нуля. Если запаздывание мешает, порог ставят с запасом: при пороге 35 см медиана остановила бы робота на двенадцатом отсчёте.

 **Типичная ошибка.** Остановить вторую программу на девятом отсчёте, где среднее равно 30. В условии сказано «меньше 30», а 30 не меньше 30. Разница между «меньше» и «не больше» на отборе стоит целого ответа – читай знак внимательно.

Ответ: 8, 10, 13.

Своя задача по программе сферы, модуль 3

Реши сам

8.13.* Робот едет к стене со скоростью 0,3 м/с, дальномер меряет раз в 0,1 с. Стоп – когда значение меньше 25 см. Показания (см): 47, 44, 41, 38, 35, 3, 29, 26, 23, 20, 17, 14. Программы те же (окно «назад», как на ходу): сами показания, скользящее среднее трёх последних, медиана трёх последних. На каком отсчёте остановится каждая? Ответ – три номера.

8.14. Робот едет к стене со скоростью 0,4 м/с, дальномер меряет каждые 50 мс, шума и выбросов нет. Программа сравнивает с порогом скользящее среднее пяти последних отсчётов (окно «назад») вместо самих показаний. На сколько сантиметров дальше проедет робот, чем если бы программа смотрела на сами показания?

Ответы и решения

Сначала реши задачу сам и только потом сверяйся. Не сошлось – найди в решении шаг, на котором твой путь разошёлся с нашим.

К главе 1. Как устроен робот

1.1. Ответ: 425310.

Подсказка: температуру не измеряет ни одна деталь из списка – поэтому у работы E стоит 0.

1.2. Ответ: 12468.

Подсказка: датчик сообщает контроллеру, что происходит вокруг; сервомотор, пищалка и драйвер сами ничего не измеряют.

1.3. Ответ: 50 об/мин.

1.4. Ответ: 45 зубьев.

Подсказка: ведомая должна крутиться в $120:40 = 3$ раза медленнее, значит, зубьев у неё в 3 раза больше.

1.5. Ответ: 12457.

1.6. Ответ: 43215.

Подсказка: лишняя команда 6 – робот повернул бы налево вокруг левого колеса, а такого движения в списке нет.

1.7. Ответ: 30 об/мин, в ту же сторону, что ведущая.

Подсказка: промежуточная шестерня на скорость не влияет – считай, будто 16 зубьев сцеплены прямо с 48; зато зацеплений два.

1.8. Ответ: 160 об/с.

Подсказка: здесь идём от колеса к мотору, поэтому умножаем: $200 \cdot 48 = 9600$ об/мин, потом делим на 60.

1.9. Ответ: 30 см.

1.10. Ответ: 5 Н.

Подсказка: сервомотор давит на короткое плечо, а губка сидит на длинном – здесь рычаг проигрывает в силе, зато губка ходит шире.

1.11. Ответ: 44 мм.

1.12. Ответ: 16 оборотов.

Подсказка: вместо готового передаточного числа сначала найди его по зубьям: $30:10 = 3$. Колесо сделает 5,31 оборота, мотор – 15,9.

К главе 2. Движение: скорость, путь, время

2.1. Ответ: 15 см/с.

2.2. Ответ: 140 см.

Подсказка: роботы сближаются на $20 + 30 = 50$ см за секунду; сначала найди, через сколько секунд они встретятся.

2.3. Ответ: 283 см.

2.4. Ответ: 5,0 оборота.

Подсказка: дан радиус, поэтому диаметр колеса – 7 см.

2.5. Ответ: 28,3 см/с.

2.6. Ответ: 115 об/мин.

Подсказка: цепочка идёт в обратную сторону: см/с делим на путь за оборот – получаем обороты в секунду, потом умножаем на 60.

2.7. Ответ: стоял с 4-й по 6-ю секунду; средняя скорость 14 см/с.

2.8. Ответ: 24 см/с.

Подсказка: не 25 см/с – обратно робот ехал дольше (6 с против 4 с), поэтому средняя ближе к меньшей скорости.

2.9. Ответ: 2800 мс.

Подсказка: тормозной путь уже дан: на моторах робот должен проехать $150 - 2 - 8 = 140$ см.

2.10. Ответ: 27 см.

Подсказка: скорость втрое больше – тормозит втрое дольше и в среднем втрое быстрее, путь в $3 \cdot 3 = 9$ раз длиннее.

2.11. Ответ: на 4 %.

2.12. Ответ: 10,6 оборота.

Подсказка: за один оборот робот проезжает 90 % от πD , то есть $0,9 \cdot \pi \cdot 6 \approx 16,96$ см.

2.13. Ответ: 150 см.

2.14. Ответ: 10 с.

Подсказка: секунды разгона и торможения вместе дают $10 + 20 + 30 + 20 + 10 = 90$ см; остальной путь робот едет по 30 см за секунду.

К главе 3. Координаты и углы: навигация робота

3.1. Ответ: (2; 4).

3.2. Ответ: (5; 7), робот смотрит влево.

Подсказка: после разворота робот смотрит вверх, и «вперёд 5» увеличивает у, а не уменьшает.

3.3. Ответ: налево на 45°.

Подсказка: сдвиги по х и по у одинаковые (по 3), значит, цель на диагонали вверх-вправо – курс 45°, а робот смотрит с курсом 90°.

3.4. Ответ: 9 отрезков.

Подсказка: задача обратная: за весь объезд робот повернётся на 360°, по 40° в каждой вершине – 360:40.

3.5. Ответ: 3,6 м.

3.6. Ответ: 1 : 20 (рисунок 18 см × 12 см).

Подсказка: переведи метры в сантиметры и проверь каждый масштаб; при 1 : 15 рисунок выходит 24 см в ширину – шире листа.

3.7. Ответ: 390 см.

3.8. Ответ: 231.

Подсказка: порядков всего шесть – посчитай путь по линиям для каждого; самый короткий – 15 клеток.

3.9. Ответ: 1 оборот.

3.10. Ответ: 150°.

Подсказка: задача обратная: угол робота = угол колеса · $D : B$.

3.11. Ответ: по линиям 275 см, напрямую 202 см.

3.12. Ответ: 12.

Подсказка: по линиям 7 и 8 клеток, а напрямую $\sqrt{37} \approx 6,08$ и $\sqrt{32} \approx 5,66$ – ближний груз меняется.

3.13. Ответ: 8 маршрутов.

Подсказка: в таблице поставь у закрытого перекрёстка 0 и считай дальше тем же правилом; или из всех 20 маршрутов вычти 12, проходящих через (1; 1).

3.14. Ответ: 24 маршрута.

К главе 4. Алгоритмы и исполнители

4.1. Ответ: не выполнит: робот врежется на 5-й команде, стоя в клетке (1; 3).

4.2. Ответ: 4 команды (например, вперёд 3, направо, вперёд 4, направо).

Подсказка: в конце робот должен смотреть вниз, поэтому кроме поворота между двумя отрезками нужен ещё один.

4.3. Ответ: длина 3 (повтори 4: и в теле вперёд 2 , направо); робот вернётся в (1; 1) и будет смотреть вверх.

4.4. Ответ: 20 команд; клетка (6; 6), смотрит вверх.

4.5. Ответ: направо .

Подсказка: с налево робот после первого отрезка повернётся влево и врежется в левый край поля.

4.6. Ответ: 1 (направо); вперёд 1 выполнится 9 раз.

Подсказка: первый цикл ведёт робота вверх до края (4 шага), второй – вправо до края (5 шагов).

4.7. Ответ: клетка (4; 7); 13 команд.

4.8. Ответ: 27 команд; клетка (1; 7).

Подсказка: «квадрат» возвращает робота в ту же клетку с тем же направлением, двигает его только вперёд 2 .

4.9. Ответ: 13; в этом лабиринте короче правило левой руки (13 шагов против 17).

4.10. Ответ: ППСЛ.

4.11. Ответ: 7.

4.12. Ответ: робот вернётся в (1; 1) и будет смотреть вверх; ошибка в строке 5 – там должно быть налево .

4.13. Ответ: а) 4; б) –1.

4.14. Ответ: на клетке 4.

Подсказка: подъезд здесь длиннее, чем в задаче 4.7: $1 \rightarrow 6 \rightarrow 2$, а круг начинается с клетки 2.

К главе 5. Моторы, датчики и электричество

5.1. Ответ: 6 В.

Подсказка: здесь ищем напряжение: $U = I \cdot R$, ток – в амперах, сопротивление – в омах.

5.2. Ответ: 8 Ом; в 37,5 раза.

5.3. Ответ: 390 Ом.

Подсказка: нужно не меньше 350 Ом, а такого в ряду нет.

5.4. Ответ: 100 Ом.

Подсказка: на двух светодиодах вместе оседает 4 В, резистору остаётся 1 В.

5.5. Ответ: 643.

Подсказка: HIGH и HIGH – колесо стоит, как и LOW и LOW.

5.6. Ответ: 0110; значение 153.

5.7. Ответ: 24 см/с.

5.8. Ответ: 190.

Подсказка: отсчитывай не от нуля, а от 60: на скорость работают $255 - 60 = 195$ единиц.

5.9. Ответ: 90 мин.

5.10. Ответ: 1200 мА·ч.

Подсказка: задача обратная: сначала найди заряд на 45 минут (0,75 ч), потом – какая ёмкость даёт его, если можно взять только 80 %.

5.11. Ответ: 12.

Подсказка: порог 470; линия – показания меньше порога.

5.12. Ответ: 20 см.

Подсказка: сначала число в вольты: $286 \cdot 5 : 1023 \approx 1,4$ В, потом – по таблице.

5.13. Ответ: 1,9 с.

Подсказка: до стены 59,5 см, а проехать до остановки нужно на 12 см меньше.

5.14. Ответ: 41,1 см.

Подсказка: теперь звук быстрее, чем думает программа, – настоящее расстояние больше показанного.

К главе 6. Программирование: Arduino и машины состояний

6.1. Ответ: 60 8.

Подсказка: с условием `i ≤ 3` цикл делает 4 круга: $i = 0, 1, 2, 3$.

6.2. Ответ: 210 82.

Подсказка: $210 \cdot 100 = 21\,000$, а $21\,000 / 255 = 82,35\dots$, дробь отброшена.

6.3. Ответ: равносторонний треугольник со стороной 30 см, след 90 см.

Подсказка: три поворота по 120° – это полный оборот, 360° , поэтому фигура замкнётся.

6.4. Ответ: 960 см; робот в точке старта и смотрит туда же, куда в начале.

Подсказка: один круг `loop()` длится 2,5 с, за 60 с их 24 – это 6 полных квадратов.

6.5. Ответ: 10 40.

6.6. Ответ: строка 3; должно быть `if (d < 20)`.

Подсказка: показания перескакивают через 20 (21, потом 17), и условие `d == 20` не выполняется ни разу.

6.7. Ответ: 426513.

6.8. Ответ: 24513.

6.9. Ответ: 112223333112.

Подсказка: на девятом круге 39 не больше 40, робот ещё поворачивает.

6.10. Ответ: красный; 8 раз.

Подсказка: полный круг светофора – 8 с; $50 = 6 \cdot 8 + 2$, а красный горит первые 3 с каждого круга. Зелёный загорается на 3-й, 11-й, 19-й, ..., 59-й секунде.

6.11. Ответ: 17 51.

6.12. Ответ: клетка (2; 5), робот смотрит вправо; обратный путь 121121311.

6.13. Ответ: 20.

Подсказка: от 3 до 8 в обе стороны по 5 перегонов; программа берёт **b**, но число то же.

6.14. Ответ: 1 12.

Подсказка: маршрут $1 \rightarrow 8 \rightarrow 7 \rightarrow 2 \rightarrow 1$, переезды по 5, 1, 5 и 1 перегону.

К главе 7. Движение по линии и регуляторы

7.1. Ответ: 508.

7.2. Ответ: 23.

Подсказка: у этого датчика линия – большое число, поэтому линия – показания больше порога 510.

7.3. Ответ: 32141.

7.4. Ответ: 3.

Подсказка: записей «оба на чёрном» шесть, но идут они тремя сериями; каждая серия – один перекрёсток.

7.5. Ответ: левый 240, правый 0; 2 (направо).

7.6. Ответ: левый 25, правый 255.

Подсказка: у правого выходит 375, больше 255 он получить не может.

7.7. Ответ: 0,25.

7.8. Ответ: база 135; медленный мотор 15.

Подсказка: самая большая поправка $0,2 \cdot 600 = 120$, а быстрый мотор – это база плюс поправка.

7.9. Ответ: П 42; И 13; Д 45; поправка 100.

7.10. Ответ: 2,5 и 10; в 2 раза.

Подсказка: копилка каждый шаг растёт на 10: после 5 шагов в ней 50, после 20 – 200.

7.11. Ответ: 30 %.

7.12. Ответ: ошибка 30; левый мотор 175, правый 85.

7.13. Ответ: -1,5 мм.

7.14. Ответ: 14; 5.

Подсказка: каждому роботу хватит двух-трёх шагов, как в задаче 7.7; быстрее – по множителю $1 - k$ из рамки «Для любопытных» в разделе 7.5.

К главе 8. Сигналы и обработка данных

8.1. Ответ: а) 50; б) 6,4 мм.

8.2. Ответ: 60.

Подсказка: здесь задача обратная – сначала найди шаг: чтобы на 15 мм наверняка пришлось три отсчёта, шаг должен быть не больше $15 : 3 = 5$ мм.

8.3. Ответ: 6 10.

8.4. Ответ: отсчёты 5 и 9; новые значения 40,5 и 60.

8.5. Ответ: 7 значений; размах 8 см и 2 см; в 4 раза.

8.6. Ответ: 58, 56, 54, 52, 50, 48, 46; на 2 см.

Подсказка: качание повторяется каждые три отсчёта, поэтому окно из трёх каждый раз захватывает его целиком, и оно гасится полностью.

8.7. Ответ: 3 и 5.

8.8. Ответ: 8 и 3.

Подсказка: вторая программа после «вошли» не замечает ничего, пока показание не станет больше 600; нырки к 430–440 между порогами она пропускает.

8.9. Ответ: 2.

8.10. Ответ: 0,35 м/с.

Подсказка: меток две, поэтому один оборот – это два промежутка между вспышками: $2 \cdot 15 \cdot 20 \text{ мс} = 600 \text{ мс}$.

8.11. Ответ: 105.

8.12. Ответ: 40 кГц; 19.

Подсказка: веса идут слева направо 1, 2, 4, 8...: единицы в первом, втором и пятом окошках дают $1 + 2 + 16$.

8.13. Ответ: 6, 7, 10.

8.14. Ответ: на 4 см.

Подсказка: на ровном подъезде среднее пяти последних отсчётов равно показанию двухотсчётной давности, а за отсчёт робот проезжает $0,4 \cdot 0,05 = 0,02$ м.

Справочник

Всё, что может понадобиться под рукой во время решения: числа, которые используются в книге, перевод единиц, формулы и правила из глав, мини-шпаргалки по Arduino, Tinkercad, ПРИМС и таблицам, словарь и все «Запомни» из глав на нескольких страницах.

Числа, которые используются в книге

Если в условии задачи на отборе дано другое значение, бери его из условия.

Таблица С.1

Величина	Значение
Arduino Uno: напряжение питания датчиков	5 В
<code>analogRead</code> – что возвращает	целое число от 0 (0 В) до 1023 (5 В)
<code>analogWrite</code> (ШИМ) – что принимает	целое число от 0 (выключено) до 255 (полная мощность)
Выводы Arduino Uno с ШИМ (значок ~)	3, 5, 6, 9, 10, 11
Ток одного вывода Arduino	до 40 мА – это предел, на все выводы вместе до 200 мА; безопасно держаться около 20 мА. Мотору мало, нужен драйвер
Сервомотор: угол поворота	от 0 до 180°
Обычный светодиод: напряжение / допустимый ток	около 2 В / 20 мА
Одна пальчиковая батарейка (AA)	1,5 В
Скорость звука в воздухе	около 340 м/с
Ультразвуковой дальномер HC-SR04: частота / что меряет	40 кГц / примерно от 2 до 400 см
Звук, который слышит человек	примерно от 20 до 20 000 Гц; выше – ультразвук
Число π	3,14 (кнопка π на калькуляторе)
Полный оборот	360°
Тип <code>int</code> на Arduino Uno	целые числа от -32 768 до 32 767

Перевод единиц

Таблица С.2

Что	Как переводить
-----	----------------

Что	Как переводить
Длина	1 км = 1000 м · 1 м = 100 см = 1000 мм · 1 см = 10 мм
Время	1 мин = 60 с · 1 с = 1000 мс · 1 мс = 1000 мкс
Скорость	1 м/с = 100 см/с = 3,6 км/ч
Обороты	1 об/с = 60 об/мин
Частота	1 Гц – одно колебание в секунду · 1 кГц = 1000 Гц
Напряжение	1 В = 1000 мВ
Ток	1 А = 1000 мА
Сопротивление	1 кОм = 1000 Ом
Ёмкость аккумулятора	1 А·ч = 1000 мА·ч
Мощность	1 кВт = 1000 Вт
Информация	1 байт = 8 бит

Совет. Перед тем как считать, выпиши все числа из условия и переведи их в одни единицы. Колесо в миллиметрах, поле в метрах, скорость в сантиметрах в секунду – верный путь к ответу, который отличается в 10 или в 100 раз. Дальномер меряет время в **микросекундах**, а delay ждёт в **миллисекундах**: это тоже разница в тысячу раз.

Формулы и правила

Если в условии задачи правило записано по-своему, решай по условию.

Путь, скорость, время

Путь s равен скорости v , умноженной на время t . Из этой фразы получаются все три формулы:

$$s = v \cdot t, \quad v = \frac{s}{t}, \quad t = \frac{s}{v}.$$

Робот едет 25 см/с. За 8 с он проедет $25 \cdot 8 = 200$ см = 2 м. На 150 см ему нужно $150 : 25 = 6$ с.

Средняя скорость – весь путь, делённый на всё время, включая остановки:

$$v_{\text{ср}} = \frac{s_1 + s_2 + \dots}{t_1 + t_2 + \dots}.$$

Робот проехал 2 м со скоростью 0,5 м/с (4 с), а потом ещё 2 м со скоростью 0,25 м/с (8 с). Средняя скорость – $4 : 12 \approx 0,33$ м/с, а не $(0,5 + 0,25) : 2 = 0,375$ м/с. Складывать скорости и делить пополам можно, только если на каждой скорости робот ехал одинаковое **время**.

Два робота едут навстречу – расстояние между ними сокращается на сумму их скоростей за каждую секунду. Один догоняет другого – на разность скоростей.

Инерция и трение. Робот не останавливается мгновенно: после выключения моторов он ещё немного катится. Поэтому «стоп» дают чуть раньше нужной точки, а в симуляторе и на настоящем полигоне путь торможения получается разным. Сила трения мешает скольжению: на гладком полу колёса пробуксовывают, робот проезжает меньше, чем посчитано по оборотам.

Колесо, обороты, передачи

За один оборот колесо без проскальзывания прокатывается на длину своей окружности:

Таблица С.3

Что найти	Формула	Пример, колесо $D = 6,5$ см
Путь за один оборот	$C = \pi \cdot D$	$3,14 \cdot 6,5 \approx 20,4$ см
Путь за n оборотов	$s = n \cdot \pi D$	5 оборотов: около 102 см
Сколько оборотов на путь s	$n = \frac{s}{\pi D}$	1 м = 100 см: $100 : 20,4 \approx 4,9$ оборота
Скорость робота по оборотам колеса	$v = (\text{обороты в секунду}) \cdot \pi D$	120 об/мин = 2 об/с: около 40,8 см/с

D – диаметр, расстояние через центр колеса от края до края. Радиус – половина диаметра. Если в условии дан радиус r , то путь за оборот $C = 2\pi r$.

Редуктор – набор шестерёнок между мотором и колесом. Моторчик сам по себе крутится очень быстро, но слабо. Редуктор уменьшает скорость вращения и во столько же раз (если не считать потерь на трение) увеличивает силу, с которой колесо толкает робота. **Передаточное число** показывает, во сколько раз уменьшаются обороты:

$$i = \frac{\text{зубьев у ведомой шестерни}}{\text{зубьев у ведущей шестерни}}, \quad \text{обороты колеса} = \frac{\text{обороты мотора}}{i}.$$

Ведущая шестерня – та, что крутит, ведомая – та, которую крутят. Мотор 9600 об/мин с редуктором 1:48 крутит колесо со скоростью $9600 : 48 = 200$ об/мин. Если ступеней несколько, их передаточные числа **перемножают**: 4 и 5 дают $4 \cdot 5 = 20$. Каждое зацепление двух шестерён меняет направление вращения на обратное.

Рычаг в равновесии, когда сила, умноженная на своё плечо, с одной стороны равна силе, умноженной на плечо, с другой:

$$F_1 \cdot l_1 = F_2 \cdot l_2.$$

Плечо – расстояние от опоры до места, где приложена сила. Груз 20 г на плече 30 см уравновешивает груз 60 г на плече 10 см: $20 \cdot 30 = 60 \cdot 10$. Так устроены захват робота, кнопка и даже гаечный ключ.

Координаты и масштаб

Положение точки на поле задают двумя числами $(x; y)$: x – сколько шагать вправо от начала отсчёта, y – сколько вверх. Числа пишем через точку с запятой, потому что запятая у нас – для дробей.

Таблица С.4

Что найти	Формула	Пример: из (1; 2) в (6; 8)
Путь, если ехать только вдоль осей (по клеткам, по линиям полигона)	$ x_2 - x_1 + y_2 - y_1 $	$5 + 6 = 11$ клеток
Сдвиг по x и по y	$x_2 - x_1$ и $y_2 - y_1$	5 вправо и 6 вверх
Путь напрямую	$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$	$\sqrt{25 + 36} \approx 7,8$ клетки

Знаки $||$ означают «без минуса»: если $x_2 - x_1 = -3$, берём 3. Путь напрямую считают по теореме Пифагора – в школе она будет в 8 классе, а пока хватит кнопки корня на калькуляторе. Проверь себя на удобной паре: сдвиг 3 и 4 даёт ровный путь $\sqrt{9 + 16} = 5$.

Масштаб 1 : 50 значит, что 1 см на плане – это 50 см на настоящем поле. Отрезок 7 см на плане – это $7 \cdot 50 = 350$ см = 3,5 м. Если сказано «сторона клетки поля – 20 см», то 11 клеток – это $11 \cdot 20 = 220$ см.

Углы и повороты

Таблица С.5

Угол	Сколько градусов
Полный оборот	360°
Разворот назад	180°
Прямой угол, поворот «налево» или «направо» на перекрёстке	90°
Сумма углов треугольника	180°
Сумма углов четырёхугольника	360°

Робот обходит правильный многоугольник с n сторонами и в каждой вершине поворачивает на один и тот же угол. Вернувшись в начало, он повернулся на полный оборот, поэтому

$$\text{угол поворота в вершине} = \frac{360^\circ}{n}.$$

Квадрат: $360:4 = 90^\circ$. Правильный треугольник: $360:3 = 120^\circ$ – робот поворачивает на 120° , хотя угол у треугольника 60° . Путать эти два угла – ошибка номер один в задачах про черепашку и робота.

Поворот на колёсах. B – колея: расстояние между серединами левого и правого колёс.

Таблица С.6

Как поворачивает	Путь каждого колеса на угол α	Пример: $B = 12$ см, 90°
На месте: колёса крутятся в разные стороны	$\pi \cdot B \cdot \frac{\alpha}{360}$	$3,14 \cdot 12 \cdot 90:360 \approx 9,4$ см
Вокруг стоящего колеса: едет только одно	$2\pi \cdot B \cdot \frac{\alpha}{360}$	$\approx 18,8$ см

Путь колеса делим на путь за один оборот – получаем, сколько оборотов сделать. Колесо $D = 6$ см проходит за оборот $3,14 \cdot 6 \approx 18,8$ см, значит, для разворота на месте на 90° при более 12 см каждому колесу нужно $9,4 : 18,8 = 0,5$ оборота.

Откуда считают углы направления и в какую сторону они растут (по часовой стрелке или против), в разных задачах по-разному. Это всегда сказано в условии или видно на рисунке.

Формулы фигур

Таблица С.7

Фигура	Формула
Периметр прямоугольника $a \times b$	$P = 2 \cdot (a + b)$
Площадь прямоугольника	$S = a \cdot b$
Длина окружности	$C = \pi D = 2\pi r$
Площадь круга	$S = \pi r^2$
Длина дуги в α градусов	$\pi D \cdot \frac{\alpha}{360}$

Электричество

Напряжение U – в вольтах (В), ток I – в амперах (А), сопротивление R – в омах (Ом). Ток похож на воду в трубе: напряжение – это напор, сопротивление – узость трубы.

Таблица С.8

Правило	Формула	Пример
Закон Ома	$I = \frac{U}{R}$	5 В на 1000 Ом: 0,005 А = 5 мА
Резистор для светодиода	$R = \frac{U_{\text{пит}} - U_{\text{св}}}{I}$	$(5 - 2) : 0,02 = 150$ Ом, ставят ближайший больший из продающихся, например 220 Ом
Мощность	$P = U \cdot I$	6 В и 0,5 А: 3 Вт
Два резистора подряд (последовательно)	$R = R_1 + R_2$	100 Ом и 300 Ом: 400 Ом
Два резистора рядом (параллельно)	$R = \frac{R_1 \cdot R_2}{R_1 + R_2}$	100 Ом и 300 Ом: 75 Ом
Делитель напряжения	$U_{\text{вых}} = U \cdot \frac{R_2}{R_1 + R_2}$	5 В, 3 кОм и 2 кОм: 2 В
Время работы от аккумулятора	$t = \frac{\text{ёмкость}}{\text{ток}}$	2000 мА·ч при 500 мА: 4 ч

- **Последовательно:** ток везде один и тот же, напряжения складываются. Четыре батарейки по 1,5 В подряд дают 6 В.
- **Параллельно:** напряжение на всех ветках одно и то же, токи складываются. Общее сопротивление меньше самого маленького из резисторов; два одинаковых резистора рядом – половина одного.
- Ток в формулы подставляй в амперах: 20 мА = 0,02 А.

Моторы и драйвер

Мотор постоянного тока (DC-мотор) напрямую к Arduino не подключают: вывод платы даёт самое большее 40 мА (безопасно – около 20), а мотору нужны сотни миллиампер. Между ними ставят **драйвер моторов**, например микросхему L293D. Она управляет двумя моторами: для каждого есть вход скорости EN и два входа направления IN.

Таблица С.9

IN1 и IN2	Мотор
HIGH и LOW	крутится в одну сторону
LOW и HIGH	крутится в другую сторону
LOW и LOW	стоит

На вход EN подают ШИМ: `analogWrite` от 0 до 255 задаёт скорость. У L293D 16 ножек: 1 и 9 – входы скорости EN первого и второго мотора; 2 и 7 – входы IN первого мотора, 3 и 6 – его выходы на мотор; 10 и 15 – входы второго мотора, 11 и 14 – его выходы; 4, 5, 12, 13 – «минус» (GND); 16 – питание самой микросхемы 5 В; 8 – питание моторов.

Дифференциальный привод – у робота два ведущих колеса, у каждого свой мотор, а поворачивает он разницей скоростей:

Таблица С.10

Левый мотор	Правый мотор	Робот
вперёд	вперёд, так же быстро	едет прямо
вперёд быстрее	вперёд медленнее	плавно поворачивает направо
вперёд	стоит	поворачивает направо вокруг правого колеса
вперёд	назад	разворачивается на месте направо

Рулевое управление – как у машины: один мотор везёт, а сервомотор поворачивает передние колёса. В Tinkercad можно собрать и такую схему.

ШИМ в процентах мощности:

$$\text{мощность в процентах} = \frac{\text{значение}}{255} \cdot 100, \quad \text{значение} = \frac{\text{мощность в процентах}}{100} \cdot 255.$$

`analogWrite(5, 191)` – около 75 %. Для 40 % нужно $0,4 \cdot 255 = 102$. В задачах часто считают, что скорость робота пропорциональна значению ШИМ. В жизни при маленьком значении мотор вообще не трогается с места.

Датчики: число, напряжение, расстояние

Датчик превращает свет, расстояние или поворот ручки в напряжение. Arduino измеряет напряжение на аналоговом входе и выдаёт целое число N от 0 до 1023. Это ещё не сантиметры и не градусы, а номер ступеньки.

Таблица С.11

Что	Формула	Пример
-----	---------	--------

Что	Формула	Пример
Число → напряжение	$U = N \cdot \frac{5}{1023}$	$N = 614: U \approx 3,0 \text{ В}$
Напряжение → число	$N = \frac{U}{5} \cdot 1023$, дробную часть отбросить	$U = 2,5 \text{ В}: 511$
Пересчёт в другой диапазон	$\text{map}(x, a, b, c, d) = \frac{(x - a) \cdot (d - c)}{b - a} + c$	$\text{map}(512, 0, 1023, 0, 180) = 90$

map считает в целых числах и **отбрасывает** дробную часть, а не округляет: $\text{map}(300, 0, 1023, 0, 255)$ даёт $300 \cdot 255 : 1023 = 74,8$, то есть 74. В некоторых задачах напряжение считают делением не на 1023, а на 1024 – решай так, как в условии.

Ультразвуковой дальномер HC-SR04 щёлкает ультразвуком и ждёт эха. Звук идёт до препятствия и обратно, поэтому время делим пополам:

$$s = \frac{v \cdot t}{2}.$$

Скорость звука 340 м/с – это 0,034 см за микросекунду. Эхо пришло через 1160 мкс: $1160 \cdot 0,034 : 2 \approx 19,7 \text{ см}$. В программах часто делят время в микросекундах на 58 – получают почти те же сантиметры: $1160 : 58 = 20$.

Длина волны $\lambda = \frac{v}{f}$: у ультразвука 40 кГц в воздухе $340 : 40\,000 = 0,0085 \text{ м} = 8,5 \text{ мм}$.

ИК-дальномер светит невидимым инфракрасным светом и ловит отражение. Его показание меняется с расстоянием неравномерно: вблизи – сильно, вдали – еле-еле. Переводят его в сантиметры по таблице или формуле из условия.

Датчик линии светит на поле и меряет, сколько света отразилось: белое отражает много, чёрное – мало. У одних датчиков над чёрным число больше, у других меньше – проверь по показаниям. **Порог** для решения «линия или нет» берут посередине:

$$\text{порог} = \frac{\text{показание на белом} + \text{показание на чёрном}}{2}.$$

Белое 900, чёрное 100: порог $(900 + 100) : 2 = 500$. В Tinkercad готового датчика линии нет, его изображают фоторезистором: ползунком меняют, сколько света на него падает.

Движение по линии и регуляторы

Регулятор – правило, по которому робот поправляет курс, глядя на датчики. **Ошибка** – насколько робот ушёл от линии: например, разница показаний левого и правого датчиков или разница между показанием и порогом.

Релейный регулятор («зигзаг») знает только «налево» и «направо». С одним датчиком робот едет по краю линии: видит чёрное – поворачивает в одну сторону, белое – в другую. С двумя датчиками по бокам линии:

Таблица С.12

Левый датчик	Правый датчик	Что делать
белое	белое	ехать прямо
чёрное	белое	повернуть налево
белое	чёрное	повернуть направо

Левый датчик	Правый датчик	Что делать
чёрное	чёрное	перекрёсток

П-регулятор (пропорциональный) поворачивает тем сильнее, чем больше ошибка:

$$\text{поправка} = k \cdot \text{ошибка}, \quad \text{левый} = \text{база} + \text{поправка}, \quad \text{правый} = \text{база} - \text{поправка}.$$

База 150, $k = 0,3$, левый датчик 620, правый 380. Ошибка $620 - 380 = 240$, поправка $0,3 \cdot 240 = 72$. Левый мотор $150 + 72 = 222$, правый $150 - 72 = 78$. Скорость не может быть больше 255 и (в этой схеме) меньше 0, поэтому результат обрезают – в Arduino это делает `constrain`. Если робот от линии не возвращается, а убегает, поменяй знаки поправки у моторов.

Коэффициент k подбирают опытом. Слишком маленький – робот не успевает за поворотом и слетает с линии. Слишком большой – робот виляет из стороны в сторону, даже на прямой.

ПИД-регулятор складывает три поправки:

- **П** смотрит, насколько робот ушёл от линии **сейчас**: $k_P \cdot \text{ошибка}$;
- **И** копит ошибки за всё время и помогает, если робот долго держится чуть сбоку от линии: $k_I \cdot \text{сумма ошибок}$;
- **Д** смотрит, как **быстро** меняется ошибка, и гасит влияние: $k_D \cdot (\text{ошибка} - \text{прошлая ошибка})$.

$k_P = 0,5$, $k_I = 0,01$, $k_D = 2$; ошибка сейчас 40, в прошлый раз 30, сумма ошибок 200. Поправка $0,5 \cdot 40 + 0,01 \cdot 200 + 2 \cdot (40 - 30) = 20 + 2 + 20 = 42$.

Перекрёстки считают так же, как удары в сигнале: счётчик растёт на 1, когда оба датчика **впервые** увидели чёрное. Пока робот стоит на перекрёстке, новые не считаем – иначе один перекрёсток превратится в двадцать.

Лабиринт

Правило правой руки: идём так, будто правая рука всё время касается стены. На каждой развилке проверяем по порядку:

1. справа свободно – поворачиваем направо;
2. иначе впереди свободно – едем прямо;
3. иначе слева свободно – поворачиваем налево;
4. иначе тупик – разворачиваемся.

Правило левой руки – то же самое зеркально. Оно выводит из лабиринта, если все его стены соединены с внешней стеной. Если в середине стоит «остров» из стен, не связанный с краем, робот может ходить вокруг него по кругу.

Путь обратно. Если робот записал повороты на перекрёстках в массив, домой он вернётся так: развернуться, пройти список с конца и заменить каждое «налево» на «направо» и наоборот. «Прямо» остаётся «прямо».

Алгоритмы и исполнители

Исполнитель – тот, кто выполняет команды: робот, черепашка на экране, чертёжник. **Система команд исполнителя** (СКИ) – все команды, которые он понимает. Команды не из этого списка исполнитель выполнить не может.

Таблица С.13

Конструкция	Что значит	Пример
Следование	команды друг за другом	вперёд 2, направо, вперёд 1
Ветвление	«если – то – иначе»	если впереди стена – направо, иначе вперёд
Цикл со счётчиком	повторить заданное число раз	повтори 4 раза: вперёд 3, направо
Цикл с условием	повторять, пока условие верно	пока впереди свободно – вперёд 1
Подпрограмма	кусоч программы со своим именем	«квадрат» – вызываем, где нужен квадрат

Как проследить работу алгоритма: заведи таблицу «шаг – команда – где исполнитель – куда смотрит» и заполняй строка за строкой. Не прыгай через шаги даже там, где «и так понятно»: ошибки прячутся на поворотах.

Сигналы

Таблица С.14

Что	Формула или правило	Пример
Период T – время одного полного колебания	$T = \frac{1}{f}$	$f = 50$ Гц: $T = 0,02$ с = 20 мс
Частота f – сколько колебаний в секунду, в герцах	$f = \frac{1}{T} = \frac{N}{t}$	10 колебаний за 2 с: 5 Гц
Размах	наибольшее минус наименьшее	от 200 до 800: 600
Амплитуда ровного сигнала	половина размаха	300
Промежуток между измерениями (отсчётами)	$\Delta t = \frac{1}{f_d}$	100 раз в секунду: каждые 10 мс
Сколько разных значений у n битов	2^n	10 бит – 1024 (analogRead), 8 бит – 256 (ШИМ)

Детектирование по уровню – событие засчитываем, когда сигнал пересёк порог; в какую сторону, не важно: стал сильнее порога или слабее (какое пересечение считать, говорит условие). **По периоду** (в курсе – «по частоте») – когда смотрят, через сколько времени сигнал повторяется.

Фильтры убирают из сигнала лишнее. Показания датчика: 20, 21, 95, 22, 23 – число 95 явно случайный выброс.

- **Скользящее среднее** по трём точкам, окно с центром (результат стоит под средним отсчётом): $(20 + 21 + 95) : 3 \approx 45,3$ под вторым отсчётом; $(21 + 95 + 22) : 3 = 46$ под третьим; $(95 + 22 + 23) : 3 \approx 46,7$ под четвёртым. Выброс размазался по трём соседям. Робот на ходу берёт окно «назад» – три последних отсчёта, результат под последним – и поэтому опаздывает.
- **Медианный фильтр** по трём точкам берёт среднее **по величине** из трёх чисел: из 20, 21, 95 – 21; из 21, 95, 22 – 22; из 95, 22, 23 – 23. Выброс исчез совсем.

Из n чисел по окну в три получается $n - 2$ новых значения. Скользящее среднее – самый простой **фильтр низких частот**: медленные изменения пропускает, быстрые скачки гасит.

Спектр – из каких частот, медленных и быстрых, складывается сигнал. **Несущая** – быстрое ровное колебание, на которое «сажают» сообщение. **Модуляция** – как сажают: меняют амплитуду несущей (амплитудная) или её частоту (частотная). **Огибающая** – плавная линия по верхушкам быстрых колебаний: по ней сообщение и читают. Её выделяют так: берут модуль сигнала, потом усредняют; после усреднения высота получается ниже исходных верхушек.

Кодирование – запись сообщения условными знаками. Двоичное число переводят в обычное, складывая «веса» разрядов с единицами. Веса справа налево: 1, 2, 4, 8, 16... Например, $1011_2 = 8 + 0 + 2 + 1 = 11$.

Мини-шпаргалки по инструментам

Arduino: язык C++

Программа для Arduino (скетч) состоит из двух частей: `setup()` выполняется один раз при включении, `loop()` – повторяется по кругу, пока есть питание. После `//` идёт комментарий: плата его не читает. Каждая команда кончается точкой с запятой.

Таблица С.15

Команда	Что делает
<code>pinMode(7, OUTPUT);</code>	вывод 7 будет выходом; <code>INPUT</code> – входом
<code>digitalWrite(7, HIGH);</code>	подать на вывод 7 напряжение 5 В; <code>LOW</code> – снять
<code>digitalRead(2)</code>	прочитать кнопку на выводе 2: <code>HIGH</code> или <code>LOW</code>
<code>analogRead(A0)</code>	измерить напряжение на A0: число от 0 до 1023
<code>analogWrite(5, 128)</code>	ШИМ на выводе 5: от 0 до 255
<code>delay(500);</code>	ждать 500 мс, ничего не делая
<code>millis()</code>	сколько миллисекунд прошло с включения
<code>pulseIn(10, HIGH)</code>	сколько микросекунд на выводе 10 держался <code>HIGH</code>
<code>map(x, 0, 1023, 0, 180)</code>	пересчитать x в другой диапазон, дробную часть отбрасывает
<code>constrain(x, 0, 255)</code>	не выпустить x за пределы от 0 до 255

Команда	Что делает
<code>Serial.begin(9600);</code>	включить связь с компьютером (пишется в <code>setup</code>)
<code>Serial.println(d);</code>	вывести значение в монитор порта и перейти на новую строку

Кнопка. Самый простой способ подключить кнопку – `pinMode(2, INPUT_PULLUP);` и кнопка между выводом 2 и GND. Тогда всё наоборот: не нажата – HIGH, нажата – LOW. Контакты кнопки при нажатии дрожат: несколько тысячных долей секунды они то замыкаются, то размыкаются, и программа может принять одно нажатие за несколько (**дребезг**). Лечат паузой `delay(50)` после обнаружения нажатия.

Цифровой вход Uno. Напряжение ниже 1,5 В плата читает как LOW, выше 3 В – как HIGH. Между 1,5 и 3 В результат неопределённый: на такое напряжение цифровой вход не рассчитывает.

Переменные: `int` – целое число, `long` – целое побольше, `float` – дробное (пишется через точку: `0.3`), `bool` – «да/нет» (`true` / `false`). Сравнения: `=` равно, `≠` не равно, `<`, `>`, `≤`, `≥`. Связки: `&&` – «и», `||` – «или», `!` – «не».

Моторы через драйвер L293D: подпрограммы движения. Подпрограмма (функция) – кусок кода со своим именем. Один раз написали `forward`, дальше вызываем его одной строкой.

```

const int ENA = 5, IN1 = 7, IN2 = 8; // левый мотор: скорость и направление
const int ENB = 6, IN3 = 2, IN4 = 4; // правый мотор

// один мотор: скорость от -255 (полный назад) до 255 (полный вперёд)
void setMotor(int en, int inA, int inB, int speed) {
    if (speed > 0) { // вперёд
        digitalWrite(inA, HIGH);
        digitalWrite(inB, LOW);
    } else if (speed < 0) { // назад
        digitalWrite(inA, LOW);
        digitalWrite(inB, HIGH);
    } else { // стоп
        digitalWrite(inA, LOW);
        digitalWrite(inB, LOW);
    }
    analogWrite(en, abs(speed)); // ШИМ – всегда без минуса
}

void drive(int left, int right) { // оба мотора сразу
    setMotor(ENA, IN1, IN2, left);
    setMotor(ENB, IN3, IN4, right);
}

void forward(int s) { drive(s, s); }
void back(int s) { drive(-s, -s); }
void turnLeft(int s) { drive(-s, s); } // разворот на месте налево
void turnRight(int s) { drive(s, -s); } // разворот на месте направо
void stopMotors() { drive(0, 0); }

void setup() {
    int pins[] = {ENA, IN1, IN2, ENB, IN3, IN4};
    for (int i = 0; i < 6; i++) {
        pinMode(pins[i], OUTPUT);
    }
}

void loop() {
    forward(200); // 2 секунды вперёд
    delay(2000);
    turnRight(150); // полсекунды разворота
    delay(500);
    stopMotors(); // стоп на секунду
    delay(1000);
}

```

Ультразвуковой дальномер. Короткий импульс на Trig – дальномер щёлкает, `pulseIn` меряет, сколько микросекунд ждали эха.

```

const int TRIG = 9, ECHO = 10;

long distanceCm() {
    digitalWrite(TRIG, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG, HIGH);    // импульс 10 мкс - «щёлк»
    delayMicroseconds(10);
    digitalWrite(TRIG, LOW);
    long t = pulseIn(ECHO, HIGH); // время туда и обратно, мкс
    return t / 58;               // в сантиметры
}

void setup() {
    pinMode(TRIG, OUTPUT);
    pinMode(ECHO, INPUT);
    Serial.begin(9600);
}

void loop() {
    long d = distanceCm();
    Serial.println(d);
    delay(100);
}

```

П-регулятор на двух датчиках линии – только сердце программы, `drive` взят из примера с моторами выше.

```

const int BASE = 150;    // скорость на прямой
const float K = 0.3;     // коэффициент, подбирается опытом

void loop() {
    int left = analogRead(A0);    // левый датчик
    int right = analogRead(A1);   // правый датчик
    int error = left - right;     // ошибка
    int u = K * error;            // поправка
    drive(constrain(BASE + u, 0, 255), // левый мотор
          constrain(BASE - u, 0, 255)); // правый мотор
    delay(10);
}

```

Сервомотор и захват. Библиотека `Servo` поворачивает вал на угол от 0 до 180°.

```
#include <Servo.h>
Servo grip;           // сервомотор захвата

void setup() {
  grip.attach(3);      // сигнальный провод – на вывод 3
}

void loop() {
  grip.write(90);      // открыть захват
  delay(1000);
  grip.write(20);      // закрыть
  delay(1000);
}
```

Массив – несколько значений под одним именем. Номера (индексы) начинаются с нуля: у массива из пяти элементов последний – `route[4]`. Так робот хранит маршрут:

```
char route[] = {'F', 'L', 'F', 'R', 'F'}; // F – прямо, L – налево, R – направо
const int N = 5;                          // сколько элементов

void runRoute() {                          // проехать весь маршрут
  for (int i = 0; i < N; i++) {             // i = 0, 1, 2, 3, 4
    if (route[i] == 'F') forward(200);
    if (route[i] == 'L') turnLeft(150);
    if (route[i] == 'R') turnRight(150);
    delay(700);
  }
  stopMotors();
}
```

Совет. Ошибки, на которых спотыкаются чаще всего: забыли точку с запятой; открыли фигурную скобку и не закрыли; в сравнении написали `=` вместо `==`; дробное число записали через запятую (`0,3` вместо `0.3`); забыли `pinMode(..., OUTPUT)`; в `for (int i = 0; i < 5; i++)` последнее `i – 4`, а не `5`. И ещё одна, коварная: `int` на Arduino Uno вмещает только до `32 767`, поэтому `1000 * 60` в `int` даёт мусор – для миллисекунд бери `long`.

Tinkercad: что где

Так было в августе и сентябре 2026 года, на вебинарах сферы. Интерфейс может обновиться – смысл останется.

- **Вход.** Ссылка на класс «Роботех» и инструкция – в курсе сферы: зелёная кнопка входа по имени учащегося, само имя (логин) указано там же, потом «Перейти в класс». Личная учётная запись не нужна. Проекты в классе **могут не сохраниться** до следующего захода: нужный код копируй в файл, схему зарисуй в тетрадь.
- **Новая схема:** «Создать» → «Цепи» (Circuits). В панели компонентов справа выбери показ **всех** компонентов – они разложены по разделам: общие (резисторы), входы (датчики, кнопки, потенциометры), выходы (светодиоды, моторы, дисплеи), питание, макетные платы, микроконтроллеры, микросхемы (среди них драйвер L293D), приборы (мультиметр).

- **Провода** тянутся мышкой от контакта к контакту. Минус принято делать чёрным, плюс – красным: цвет выбирается в панели сверху.
- **Макетная плата:** наведи мышь на отверстие – подсветятся все отверстия, соединённые с ним. Длинные ряды по краям – шины питания, короткие столбики по пять отверстий – для деталей.
- **Код:** кнопка «Код» открывает редактор. Сначала он показывает блоки – переключи на текст, чтобы писать на C++. Внизу окна кода – монитор последовательного порта и кнопка графика.
- **Запуск:** «Начать моделирование». Щёлкни во время моделирования по фоторезистору или датчику – появится ползунок или предмет, который можно двигать: так меняют свет и расстояние.
- **Кнопка** остаётся нажатой, если щёлкнуть по ней с зажатой клавишей Shift.
- **Дальномер** бери стандартный, с четырьмя выводами: VCC (5 В), Trig, Echo, GND.
- **Датчика линии нет** – его изображают фоторезистором в делителе напряжения.
- **Моторы:** мотор постоянного тока подключают через L293D, а не прямо к плате. Колёс и корпуса в симуляторе нет – проверяется схема и то, как она работает.
- **Предупреждение о токе** значит, что в жизни деталь сгорела бы: чаще всего не хватает резистора.
- Если симулятор пишет, что сервер недоступен, или моделирование не запускается после ошибки в коде, подожди и обнови страницу.

Wokwi (wokwi.com) – запасной вариант: выбираешь Arduino Uno, добавляешь детали, пишешь код и запускаешь. Ничего ставить не нужно, но на бесплатной учётной записи запуск иногда ждёт в очереди.

Машина состояний и ПРИМС

ПРИМС – «программирование расширенных иерархических машин состояний». Программу не пишут строчками, а рисуют схемой в Кибериаде IDE: в каких режимах бывает робот и что заставляет его переходить из одного в другой. По схеме редактор умеет сгенерировать код на C++ для Arduino.

Таблица С.16

Элемент	Что это	Пример для робота
Состояние	режим работы	«едем прямо», «объезжаем», «стоим»
Событие	то, что случилось	нажата кнопка, прошло 2 с, пришло расстояние
Действие	что робот делает	повернуть направо, включить красный
Ограждающее условие (в книге – просто «условие»)	в квадратных скобках после события: переход сработает, только если оно верно	расстояние меньше 20 см

Состояния, события и действия – три ключевых элемента ПРИМС. Ограждающие условия к ним добавляются. **Переход** – не отдельный элемент, а стрелка между состояниями: она под-

писана событием, а при необходимости и условием.

Подпись на стрелке читай так: **событие [условие] / действие**. Например: «пришло расстояние [d < 20] / повернуть направо».

- В каждый момент робот находится в одном состоянии, не в двух сразу.
- Действие на **входе** в состояние выполняется один раз, когда в него попали; на **выходе** – один раз, когда уходят.
- Если условие на стрелке ложно, перехода нет: робот остаётся, где был.
- Самый первый пример курса – мигание светодиодом: два состояния, «горит» и «не горит», переходы по времени. Светофор – несколько состояний по кругу.

Как проследить работу машины: заведи таблицу «шаг – текущее состояние – событие – условие верно? – новое состояние – действие» и заполняй строку за строкой.

Таблицы

Подойдёт любой редактор: LibreOffice Calc, Excel, онлайн-таблицы. Формула начинается со знака = . В русских версиях аргументы разделяют точкой с запятой.

Таблица С.17

Формула	Что считает
=СРЗНАЧ(В2:В31)	среднее по ячейкам от В2 до В31 (AVERAGE)
=МАКС(В2:В31) , =МИН(В2:В31)	наибольшее и наименьшее (MAX , MIN)
=СУММ(В2:В31)	сумму (SUM)
=МЕДИАНА(В2:В4)	медиану (MEDIAN) – так делают медианный фильтр
=СЧЁТЕСЛИ(В2:В31;"<20")	сколько значений меньше 20 (COUNTIF)
=ОКРУГЛ(С2;1)	округлить до одного знака после запятой (ROUND)

Формулу пишут один раз, а потом тянут за правый нижний уголок ячейки вниз – она сама пересчитывается для каждой строки. Скользящее среднее (окно с центром): в С3 пишешь =СРЗНАЧ(В2:В4) – результат стоит под средним отсчётом – и тянешь вниз. Модуль числа – =ABS(В2) . График: выдели столбцы с данными и вставь диаграмму «линия» – для показаний датчика во времени.

Словарик

Таблица С.18

Слово	Что значит
Автономный робот	робот, который сам решает, что делать, по своим датчикам и программе, без пульта

Слово	Что значит
Актуатор (исполнительное устройство)	деталь, которая что-то делает: мотор, сервомотор, светодиод, пищалка
Аналоговый сигнал	сигнал, который меняется плавно: напряжение с датчика линии
АЦП	часть Arduino, которая превращает напряжение в число от 0 до 1023
Датчик (сенсор)	деталь, которая превращает свет, расстояние, нажатие в электрический сигнал
Датчик линии	датчик, который по отражённому свету отличает чёрную линию от белого поля
Детектирование	обнаружение нужного события в сигнале, например перехода через порог
Дифференциальный привод	два ведущих колеса со своими моторами; поворот – разницей их скоростей
Драйвер моторов	микросхема между Arduino и мотором: даёт мотору большой ток и меняет направление
Захват	«рука» робота, которая берёт и переносит груз; её двигает сервомотор
Инерция	свойство тела сохранять движение: робот не останавливается мгновенно
Исполнитель	тот, кто выполняет команды алгоритма: робот, черепашка, чертёжник
Калибровка	настройка датчика по известным точкам: «вот так он видит белое, а вот так – чёрное»
Колея	расстояние между серединами левого и правого колёс
Кодирование	запись сообщения условными знаками: азбукой Морзе, нулями и единицами
Коэффициент регулятора	число, на которое умножают ошибку; чем больше, тем резче поправка
Крутящий момент	«сила» вращения: насколько сильно вал мотора может крутить колесо
Макетная плата	плата с соединёнными внутри отверстиями, чтобы собирать схему без пайки
Массив	несколько значений под одним именем; номера начинаются с нуля
Машина состояний	описание поведения устройства через режимы (состояния) и переходы между ними

Слово	Что значит
Медианный фильтр	фильтр, который из нескольких соседних чисел берёт среднее по величине; убирает выбросы
Микроконтроллер	маленький компьютер на одной микросхеме; Arduino Uno – плата с микроконтроллером
Мобильный робот	робот, который сам перемещается: на колёсах, гусеницах, ногах
Модуляция	способ «посадить» сообщение на сигнал-переносчик (несущую), меняя его амплитуду или частоту
Монитор порта	окно, куда Arduino выводит числа и текст; рядом – график этих чисел
Мотор постоянного тока	мотор, который крутится, пока на него подано напряжение; от знака напряжения зависит сторона
Мотор-редуктор	мотор вместе с редуктором в одном корпусе; в учебных роботах часто жёлтый
Огибающая	плавная линия, которая обводит верхушки быстро меняющегося сигнала
Ошибка регулятора	насколько робот сейчас отклонился от того, что нужно (от линии, от стены)
Перекрёсток	место, где линии полигона пересекаются; оба датчика видят чёрное
Подпрограмма (функция)	кусочек программы со своим именем, который вызывают одной строкой
Полигон	поле для заданий робота: линии, перекрёстки, стенки лабиринта, грузы
Порог	число, при переходе через которое программа принимает решение
Потенциометр	резистор с ручкой: крутишь – меняется напряжение; им задают координаты и пороги
П-регулятор	регулятор, у которого поправка пропорциональна ошибке
ПИД-регулятор	регулятор из трёх частей: по ошибке сейчас, по сумме ошибок и по скорости изменения ошибки
ПРИМС	визуальный язык программирования машинами состояний; редактор – Кибериада IDE
Редуктор	набор шестерёнок: уменьшает обороты и увеличивает силу вращения

Слово	Что значит
Релейный регулятор	регулятор, который знает только «налево» и «направо»; робот едет зигзагом
Сервомотор	мотор, который поворачивается на заданный угол от 0 до 180°
Сигнал	то, как со временем меняется величина, несущая сведения: напряжение с датчика, звук
Сила трения	сила, которая мешает телам скользить друг по другу; без неё колёса буксуют
Система команд исполнителя (СКИ)	все команды, которые понимает исполнитель
Скетч	программа для Arduino
Спектр сигнала	из каких частот, медленных и быстрых, состоит сигнал
Ультразвук	звук с частотой выше 20 000 Гц: человек его не слышит
Фильтр низких частот	обработка, которая пропускает медленные изменения и гасит быстрые скачки
Фоторезистор	резистор, сопротивление которого зависит от освещённости
Цифровой сигнал	сигнал только с двумя значениями: есть напряжение или нет, 1 или 0
ШИМ	быстрое включение и выключение вывода, чтобы плавно менять яркость или скорость мотора
HC-SR04	ультразвуковой дальномер с четырьмя выводами: VCC, Trig, Echo, GND
I2C	способ подключить дисплей или датчик всего по двум проводам данных
L293D	микросхема-драйвер для двух моторов постоянного тока
LCD 1602	дисплей на 2 строки по 16 символов
RGB-светодиод	три светодиода в одном корпусе – красный, зелёный, синий; смешивая их, получают любой цвет

Шпаргалка: всё «Запомни» из глав

Глава 1. Как устроен робот

Запомни. Спорный случай «робот или нет» решай одним вопросом: **кто принимает решение?** Человек с пультом – значит, это управляемая машина. Программа по сигналам датчиков – робот. Мобильный робот вдобавок сам перемещается: на колёсах, гусеницах, ногах или винтах.

Запомни. Провод от вывода Arduino прямо к мотору – ошибка, даже если в схеме на экране так «короче». Контроллер только командует, а кормит мотор батарея. Между контроллером и мотором всегда стоит драйвер.

Запомни. Робот поворачивает в сторону того колеса, которое крутится медленнее (или стоит, или едет назад). Сам себя проверяй так: встань, вытяни руки в стороны – это колёса. Шагни вперёд только левой стороной тела – и повернёшься направо.

Запомни. $i = z_{\text{ведомой}} / z_{\text{ведущей}}$. Скорость ведомой = скорость ведущей : i . Во сколько раз медленнее – во столько же раз сильнее. Ступени – перемножаем. Каждое зацепление меняет направление вращения, паразитная шестерня скорость не меняет.

Запомни. Сила × плечо = сила × плечо. Лёгкий груз уравновесит тяжёлый, если висит дальше от опоры – во столько же раз дальше, во сколько он легче. Плечо меряют от опоры, а не от края рычага.

Глава 2. Движение: скорость, путь, время

Запомни. $s = v \cdot t$. Ищешь время – путь дели на скорость, ищешь скорость – путь дели на время. Средняя скорость – весь путь, делённый на всё время вместе с остановками; две скорости пополам не складывают.

Запомни. 1 м/с = 100 см/с = 3,6 км/ч. 1 об/с = 60 об/мин. 1 с = 1000 мс. Чем мельче единица, тем больше число: 0,3 м/с = 30 см/с = 300 мм/с.

Запомни. Один оборот колеса – путь πD . Оборотов нужно: путь, делённый на πD . Скорость робота – обороты в секунду, умноженные на πD ; обороты в минуту сначала делим на 60. В формулу идёт диаметр: дали радиус – удвой его.

Запомни. На графике «путь – время» скорость видна по крутизне линии: путь участка дели на его время. Полочка – стоянка. Сначала прочитай подписи осей и узнай, сколько стоит одна клетка.

Запомни. После «стопа» робот ещё проезжает тормозной путь, поэтому команду дают раньше на эту длину. Скорость падает (или растёт) равномерно – значит, в среднем робот едет с половиной полной скорости, и путь торможения или разгона равен $\frac{v}{2} \cdot t$.

Запомни. Колесо едет, пока трение не даёт ему проскользнуть. Мотор крутит сильнее, чем трение держит, – колесо буксует, и робот проезжает меньше, чем по оборотам. Недобор в процентах считают от пути по оборотам – от того, сколько робот должен был проехать.

Глава 3. Координаты и углы: навигация робота

Запомни. Точка (x; y): x – шаги вправо от начала отсчёта, y – шаги вверх. Сначала всегда x. Считай промежутки между линиями, а не сами линии.

Запомни. «Вверх–вниз–вправо–влево» – по карте, «направо–налево» – от носа робота. Курс: 0° вверх, 90° вправо, 180° вниз, 270° влево. Направо – прибавь угол, налево – вычти, вышло за 360 или ниже нуля – поправь на 360. В вершине правильной фигуры робот поворачивает на $360^\circ : n$.

Запомни. В журнале на каждую команду – своя строка. У строки с поворотом x и y переписываются без изменений, у строки «вперёд» меняется одно число из двух, а курс остаётся прежним.

Запомни. Масштаб 1 : k: с плана на поле – умножь на k, с поля на план – раздели на k. Если масштаб не дан, а даны размеры одного и того же отрезка на плане и на поле, переведи их в одни единицы и раздели большее на меньшее.

Запомни. По линиям: сдвиг по x плюс сдвиг по y, оба без минуса. Маршрут через точки – сумма кусков, каждый от предыдущей точки. Напрямую – корень из суммы квадратов сдвигов, и это всегда не длиннее, чем по линиям.

Запомни. Разворот на месте: колёса едут по окружности диаметром колея B . Путь колеса – $\pi B \cdot \alpha / 360$, угол колеса – $\alpha \cdot B / D$. Вокруг стоящего колеса путь вдвое длиннее.

Глава 4. Алгоритмы и исполнители

Запомни. Робот делает только то, что есть в его системе команд, и строго по порядку. Повороты меняют взгляд, а не клетку, и «направо» считаем от носа робота. Клетка $(x; y)$: сначала столбец, потом строка.

Запомни. повтори N : – тело выполняется N раз. пока условие: – тело выполняется, пока условие верно; проверка перед каждым повтором. Длина программы и число выполненных команд – разные числа: у повтори 5: с тремя командами в теле длина 4, а выполнено 15.

Запомни. В ветвлении выполняется одна ветка: под если – когда условие верно, под иначе – когда нет. В цепочке если – иначе если побеждает первое верное условие.

Запомни. Подпрограмму описывают один раз и вызывают по имени сколько угодно раз. Само описание ничего не двигает. Команд выполнено = (команд в подпрограмме) · (число вызовов).

Запомни. Правая рука: справа → прямо → слева → разворот, в таком порядке, и после каждого поворота – шаг. Левая рука – зеркально. Путь обратно: список развилок с конца, «направо» ↔ «налево».

Запомни. Трассировка – таблица «шаг – команда – клетка – куда смотрит», по строке на каждую команду. Циклы и подпрограммы при этом раскрываем. Ошибку ищи в первой строке, где «должно» и «получилось» разошлись.

Запомни. Клетка повторилась – робот зациклился и до финиша не доедет. Чтобы узнать, где он будет через N переездов: отними подъезд, то, что осталось, раздели на длину круга с остатком и от первой клетки круга отсчитай вперёд столько переездов, сколько вышло в остатке.

Глава 5. Моторы, датчики и электричество

Запомни. Нет замкнутого круга – нет тока. Вольты толкают, амперы текут, омы мешают. $1\text{ A} = 1000\text{ mA}$, $1\text{ k}\Omega = 1000\text{ }\Omega$. На Arduino 5V – плюс, GND – минус.

Запомни. $I = U/R$, и ток в эту формулу идёт в амперах. Резистор для светодиода считают по напряжению, которое осталось: $(U_{\text{пит}} - U_{\text{св}})/I$. Из ряда берут ближайший **больший** номинал.

Запомни. Мотор – только через драйвер: вывод платы даёт до 40 мА (безопасно – около 20), мотору нужны сотни. IN1 и IN2 задают направление (разные сигналы – крутится, одинаковые – стоит), EN – можно ли ехать и как быстро. Минус батареи и GND платы соединяют.

Запомни. `analogWrite` : от 0 до 255, только выводы с ~. Значение = доля · 255. Не путай с `analogRead`, у него шкала от 0 до 1023 – о нём раздел 5.6.

Запомни. Время работы = ёмкость : ток; мА·ч делим на мА – получаем часы. Токи всех устройств складываем. Последовательно соединённые батарейки прибавляют вольты, но не мА·ч.

Запомни. `analogRead` – от 0 до 1023, это 0...5 В. Число в вольты: $N \cdot 5 : 1023$. Вольты в число: $U : 5 \cdot 1023$, дробную часть отбрасываем.

Запомни. Белое отражает много света, чёрное – мало. Порог – среднее между показаниями над белым и над чёрным. Какое число у линии больше, а какое меньше, проверяй по показаниям своего датчика.

Запомни. $s = v \cdot t/2$: звук летит туда и обратно. $340\text{ м/с} = 0,034\text{ см/мкс}$; время в микросекундах даёт расстояние в сантиметрах. ИК-дальномер переводят в сантиметры только по таблице или формуле.

Глава 6. Программирование: Arduino и машины состояний

Запомни. `=` в программе – «положи», а не «равно». Целое делённое на целое – дробь отбрасывается: $7 / 2 = 3$, $7 \% 2 = 1$. `println` переводит строку, `print` – нет. Счётчик, который должен копить, объявляй до `setup()`.

Запомни. Сравнение – `==`, а не `=` – «положи». `for (int i = 0; i < N; i++)` – тело выполнится N раз, `<=` – на один раз больше. `while` проверяет условие перед каждым кругом.

Запомни. Функцию описывают один раз: `void имя() { ... }`, вызывают по имени со скобками: `имя();`. Параметр передаёт в функцию число, `return` отдаёт число обратно. Функция моторов только переключает сигналы – как долго ехать, решает `delay` после неё.

Запомни. В каждый момент робот в одном состоянии. Переход срабатывает, только когда верно его условие; иначе робот остаётся, где был. Условия переходов проверяются только у стрелок, выходящих из текущего состояния. Время в состоянии – `millis() - t0`, где `t0` засекли при входе.

Запомни. Индексы массива – с нуля: у массива из N ячеек последняя – `[N - 1]`. `d[3]` – четвёртая ячейка. Обратный путь по записи: развернуться, пройти команды с конца и поменять местами «налево» и «направо».

Глава 7. Движение по линии и регуляторы

Запомни. Датчик видит пятно, а не точку, поэтому его число меняется плавно. Два датчика стоят по бокам от линии, на прямой оба видят белое. Какой датчик «потемнел», с той стороны и линия.

Запомни. Порог = (белое + чёрное) : 2, и у каждого датчика он свой. Белое и чёрное – средние по нескольким замерам, на том поле и при том свете, где поедет робот. Разные датчики сравнивают в процентах белизны.

Запомни. Робот поворачивает в сторону медленного колеса и в сторону датчика, который увидел линию. Оба на белом – прямо, оба на чёрном – перекрёсток. Перекрёсток в программе проверяют первым, а считают по началу серии «оба на чёрном».

Запомни. Ошибка = левый – правый; поправка = $k \cdot$ ошибка; левый мотор = база + поправка, правый = база – поправка. Отрицательная ошибка – робот поворачивает налево, положительная – направо. Скорости зажимают между 0 и 255.

Запомни. Влияние – это перелёт через линию. Его дают большой k , большая скорость и опоздание реакции. Маленький k не влияет, но на повороте вылетает. Подбирают k так: увеличивают, пока не начнётся влияние, и чуть уменьшают.

Запомни. ПИД = П (ошибка сейчас) + И (сумма ошибок) + Д (ошибка минус прошлая ошибка), каждая часть со своим коэффициентом. Д отрицательная, когда ошибка уменьшается, – она гасит влияние. И помогает, когда робот долго едет чуть сбоку.

Глава 8. Сигналы и обработка данных

Запомни. Сигнал – величина, которая меняется во времени. Период T – время одного колебания, частота f – сколько колебаний в секунду, $f = 1 : T$. Размах – от минимума до максимума, амплитуда – его половина.

Запомни. Отсчёт – одно измерение. Промежуток между отсчётами $\Delta t = 1 : f_{\text{опр}}$, путь робота за это время $s = v \cdot \Delta t$. Наверняка на полосу попадёт столько отсчётов, сколько целых шагов в её ширине.

Запомни. Шум – мелкая дрожь во всех отсчётах. Выброс – одиночный скачок далеко от соседей. Выброс ищем сравнением с обоими соседями и допуском: сколько робот мог проехать между отсчётами, плюс запас.

Запомни. Скользящее среднее гасит дрожь, но размывает выброс. Медиана убирает одиночный выброс целиком. Окно из k отсчётов даёт из n отсчётов $n - k + 1$ значений. Окно с центром стоит под средним отсчётом и не опаздывает, но ему нужны соседи с обеих сторон. Окно «назад» (так считает робот на ходу) опаздывает: на один отсчёт при окне из трёх, на два при окне из пяти.

Запомни. По уровню считаем пересечения порога, а не отсчёты за ним. От ложных срабатываний спасают выдержка, два порога и сглаживание. По периоду меряем время между началами соседних событий: $f = 1 : T$, скорость колеса $v = \pi \cdot D \cdot f$. Если меток на колесе несколько, оборот – это столько промежутков, сколько меток.

Запомни. Спектр показывает, каких колебаний в сигнале больше – медленных или быстрых. Полезное у датчика медленное, шум быстрый, поэтому фильтр низких частот чистит сигнал. Несущая – быстрое ровное мигание, огибающая – линия по её вершушкам; сообщение едет в огибающей.

Где что найти

Таблица С.19

Что	Где
Правила олимпиады, сферы, сроки	junior.ntcontest.ru
Личный кабинет участника	my.ntcontest.ru
Платформа отбора	orbita.education (заходить кнопкой из личного кабинета)
Курс сферы «Роботех» (за тест курса +5 баллов к отбору)	academy.junior.online/event/robotech_junior , вход через «Талант»
Класс «Роботех» в Tinkercad	ссылка и инструкция – в курсе сферы
Кибериада IDE (ПРИМС)	ссылка – в первых модулях курса сферы
Tinkercad	tinkercad.com
Запасной симулятор Arduino	wokwi.com
Arduino IDE	по желанию, для настоящей платы; ставит педагог кружка
Программа сферы и кружка	junior.online/robotech → «Образовательная программа»
Новости и вопросы по сфере	канал сферы «Роботех» в мессенджере MAX (находится поиском по словам «сфера Роботех»)
Памятки для Новосибирской области	nto.kvantorium54.ru

Об авторе-составителе



Артём Осиновский

Методист и разработчик детского технопарка «Кванториум» (Новосибирск). Придумывает образовательные программы и сам делает для них цифровые инструменты: платформу смены «Миссия на Луну» для фестиваля «Технопром-2026», сайт с памятками НТО Юниоры для Новосибирской области. Эту книгу собрал для школьников области, чтобы к отбору

можно было подготовиться самостоятельно – даже без кружка рядом с домом.

Нашли ошибку – пишите на osinovsky.aa@kvantorium54.ru (рабочая почта): сфера, версия книги и страница. По другим вопросам и проектам – личная: artem081033@gmail.com.