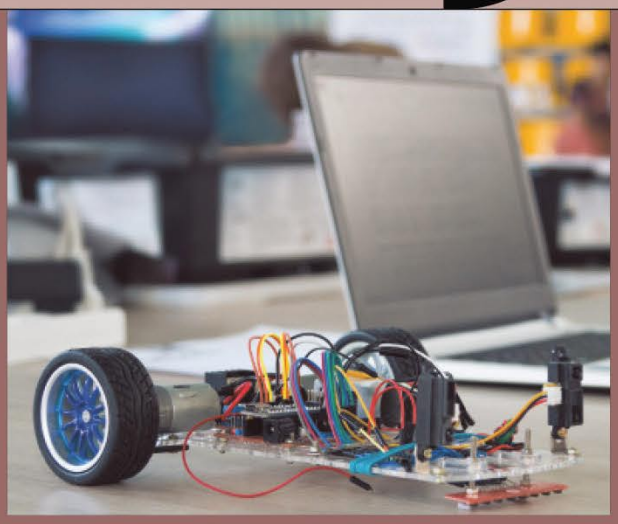
Концепция преподавания предметной области «ТЕХНОЛОГИЯ»
Модуль «РОБОТОТЕХНИКА»

9



Д. Г. Копосов

# ТЕХНОЛОГИЯ

## РОБОТОТЕХНИКА НА ПЛАТФОРМЕ ARDUINO

Программы для микроконтроллеров на языке C++

Моделирование в среде OpenSCAD

**Д. Г. Копосов**

# ТЕХНОЛОГИЯ

## РОБОТОТЕХНИКА НА ПЛАТФОРМЕ ARDUINO

9 класс

Учебник

Допущено
Министерством просвещения
Российской Федерации

*2-е издание, стереотипное*

Москва
«Просвещение»
2022

УДК 372.862
ББК 32.97
К65

*Учебник допущен к использованию при реализации имеющих государственную аккредитацию образовательных программ начального общего, основного общего, среднего общего образования организациями, осуществляющими образовательную деятельность, в соответствии с Приказом Министерства просвещения Российской Федерации № 766 от 23.12.2020.*
*Эксперты, осуществлявшие экспертизу учебника: Васильев М. В., Заславская О. Ю., Рыжова Н. И., Лукин В. В.*

Издание выходит в pdf-формате.

**Копосов, Денис Геннадьевич.**
К65 Технология. Робототехника на платформе Arduino. 9 класс : учебник : издание в pdf-формате / Д. Г. Копосов. — 2-е изд., стер. — Москва : Просвещение, 2022. — 176 с. : ил.
ISBN 978-5-09-100196-9 (электр. изд.) — Текст : электронный.
ISBN 978-5-09-092039-1 (печ. изд.).

Учебник предназначен для изучения технологии в 9 классе в рамках модуля «Робототехника» (5–6, 7–8 и 9 классы). Соответствует требованиям федерального государственного образовательного стандарта основного общего образования.

Учебные занятия с использованием данного учебника способствуют развитию всех универсальных учебных действий, помогают выстроить межпредметные связи, обеспечивают вовлечение учащихся в научно-техническое творчество.

В учебнике представлена практическая реализация П-, ПД- и ПИД-регуляторов для смоделированного, собранного и запрограммированного на языке C++ робота.

**УДК 372.862**
**ББК 32.97**

**В работе с учебником вам помогут навигационные значки:**

— важное определение или утверждение;

— задание по ссылке на интернет-ресурс;

— межпредметные связи.

---

*Учебное издание*

**Копосов** Денис Геннадьевич

**Технология**
**Робототехника на платформе Arduino**

**9 класс**

**Учебник**

Центр развития углублённого и профильного образования, функциональной грамотности

Ответственный за выпуск *Л. А. Осипова*
Редактор *Л. А. Осипова*. Оформление: *Н. А. Новак*. Технический редактор *Е. В. Денюкова*
Корректоры *Е. Н. Клитина, О. Ч. Кохановская*. Компьютерная вёрстка: *Н. И. Беляева*

В издании использованы иллюстрации, находящиеся в общественном достоянии и не являющиеся объектом авторского права, а также материалы фотобанка Adobe Stock

Подписано в печать 22.11.2021. Формат 84×108/16. Усл. печ. л. 18,48. Тираж экз. Заказ

Акционерное общество «Издательство «Просвещение»
Российская Федерация, 127473, г. Москва, ул. Краснопролетарская, д. 16, стр. 3, этаж 4, помещение I

Адрес электронной почты «Горячей линии» — **vopros@prosv.ru**.

---

**ISBN 978-5-09-100196-9 (электр. изд.)**
**ISBN 978-5-09-092039-1 (печ. изд.)**

# ОГЛАВЛЕНИЕ

# ПРЕДИСЛОВИЕ

Если вас спросят о новых направлениях в сфере информационных технологий последнего десятилетия, то, скорее всего, вы ответите, что это ноутбуки, смартфоны, планшеты и Интернет. Касаясь сенсорного экрана своего смартфона, мало кто задумывается, что касание обрабатывает специальный микроконтроллер этого экрана. Садясь в современный автомобиль, вы не говорите: «Вот это да! В нём 60 микроконтроллеров!» А между тем они контролируют датчики подушек безопасности, мониторинг и управление двигателем, управление впрыском топлива, скорость, иммобилайзер (противоугонное устройство), управление дворниками, управление фарами и светом, предупреждение столкновений, антиблокировочную систему, аудиосистему, электронный компас, дисплей и панель приборов, управление зеркалами, вентиляцию и кондиционирование, управление трансмиссией, замки дверей, мониторинг и автоподкачку колёс, активную подвеску, GPS-приёмник и т. д.

Незаметно произошла автоматизация практически всей технической среды с помощью дешёвых и мощных микроконтроллеров.

В 1971 г. сотрудники американской компании Texas Instruments Майкл Кочрен и Гарри Бун получили первый патент на однокристальную микроЭВМ. Идея разместить на одном кристалле не только процессор, но и память с устройствами ввода-вывода стала основой всех дальнейших революционных изменений потребительской электроники и техники. Такое устройство и называется *микроконтроллером*.

Контроллеры требуются практически во всех устройствах, которые нас окружают. Их используют в роботах и промышленных станках, автомобилях и самолётах, средствах связи, бытовой технике, медицинских приборах, электронных музыкальных инструментах, светофорах, автоматических воротах, шлагбаумах, интерактивных детских игрушках, смартфонах, планшетах, компьютерах и во всём оборудовании для обеспечения функционирования сети Интернет.

Как же всё так быстро изменилось? Просто незаметно для обычных пользователей произошёл качественный скачок в разработке микроэлектронных систем. Если два десятка лет назад, чтобы разрабатывать такие системы, требовалось получить высшее профессиональное образование и опыт работы, то в нынешних реалиях все возможности стали доступны школьнику. Конечно, вам будут говорить, что лёгкость раз-

работки под контроллеры — это только иллюзия. Да, микроконтроллер гораздо чувствительнее к ошибкам программиста, чем обычные компьютеры. Очень ограниченный объём памяти, требования к быстродействию и почти полное отсутствие защит от неправильного поведения пользователя требуют высокой квалификации разработчика. Но свой первый проект на микроконтроллере любой школьник может сделать за один день. И этот проект даже может обладать коммерческими перспективами. А в грядущую эпоху интернета вещей микроконтроллерные технологии — ключевой аспект развития межмашинных коммуникаций.

В этом курсе вам предстоит смоделировать робота на базе микроконтроллера Arduino и запрограммировать его на выполнение нескольких несложных действий.

Итак, *микроконтроллер* (англ. *Micro Controller Unit*, MCU) — микросхема, предназначенная для управления электронными устройствами. В его составе есть все элементы для построения системы управления: память данных (оперативная память, ОЗУ), память программ (постоянная память, ПЗУ), генератор тактовых импульсов, таймеры, счётчики, параллельные и последовательные порты. Фактически микроконтроллер — одноплатный мини-компьютер на основе одной микросхемы, подходящий для встраивания в объект управления.

Программы для микроконтроллеров пишут в специальных интегрированных средах разработки (англ. *Integrated Development Environment*, IDE) на языках ассемблера (машинных команд) или C++.

Вы будете использовать Arduino IDE и язык C++.

# ПРАВИЛА БЕЗОПАСНОСТИ И ОРГАНИЗАЦИЯ РАБОЧЕГО МЕСТА

## Безопасность превыше всего

Кабинеты робототехники, информатики, технологии являются специально оборудованными кабинетами, в которых действуют особые правила техники безопасности, поэтому к работе в них допускаются лица, прошедшие инструктаж по правилам безопасности, который проводится не реже одного раза в полугодие. Во время занятий в этих кабинетах и помещениях необходимо строго соблюдать инструкции по охране труда, требования педагога.

Ответственность за нарушение правил техники безопасности определена в Уставе вашей образовательной организации, и она может быть очень серьёзной, вплоть до исключения из учебного заведения.

### Задание

Изучите все инструкции и правила по безопасности, которые есть в вашем кабинете. По всем вопросам обращайтесь к преподавателю.

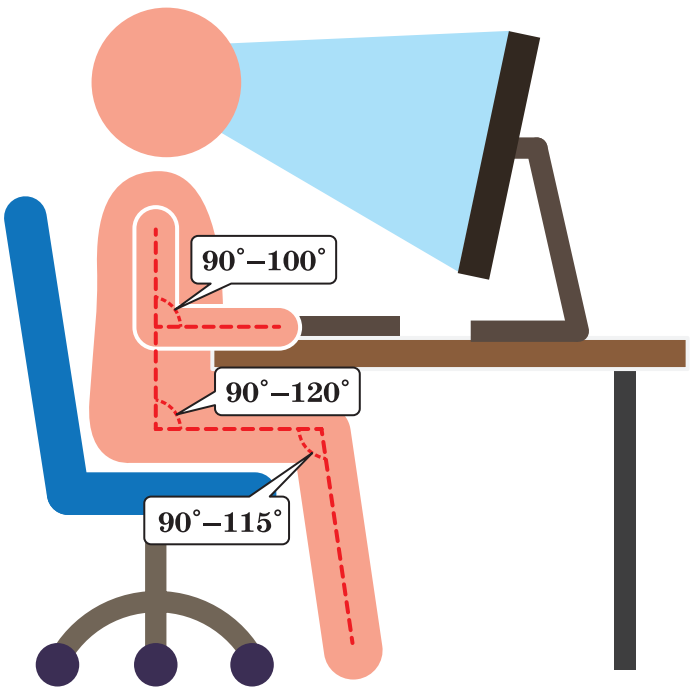
Мы же рассмотрим некоторые общие моменты работы с образовательным конструктором и необходимым оборудованием.

В кабинете присутствует достаточное количество источников опасности. Рассмотрим основные.

- Системные блоки, мониторы, зарядные устройства, сетевые фильтры, 3D-принтеры и другое оборудование, использующее напряжение 220 В, увеличивают опасность электротравматизма. *Поэтому работать с оборудованием, имеющим повреждения корпуса или изоляции проводов, производить самовольное переключение разъёмов, приносить и самовольно подключать какое-либо оборудование, вставлять в различные отверстия на корпусах посторонние предметы, выключать или включать оборудование без разрешения преподавателя ЗАПРЕЩЕНО!*
- Указанное выше оборудование увеличивает опасность возгорания, а излишняя грязь и пыль нарушают отвод тепла. *Поэтому вы должны соблюдать чистоту. Запрещено входить в кабинет в верхней одежде, головных уборах, с громоздкими предметами и едой.*
- Указанное выше оборудование является слабым источником ионизирующего излучения электромагнитных, электрических и магнитных статических полей. Электризация пыли не является положи-

тельным фактором для ваших органов дыхания. *Поэтому, прежде чем разговаривать в кабинете, проконсультируйтесь с преподавателем.*

- Мониторы, экраны ноутбуков и планшетов являются источниками повышенной опасности для вашего зрения. *Поэтому по санитарным нормам и правилам расстояние от центра экрана до глаз должно быть не менее 60 см; время интенсивной непрерывной работы перед экраном не должно превышать 25 мин, после чего обязателен перерыв. Не допускайте работы на максимальной яркости экрана.*
- Долговременная статическая поза при работе. *Помните, ваш организм ещё формируется, поэтому очень важно контролировать, как вы сидите на стуле. Обратите внимание на значения углов на рисунке!*




Правильное положение

- Оборудование и инструменты, которыми вы пользуетесь, за один учебный день в руки берут несколько школьников. Поэтому соблюдайте правила гигиены. Мойте руки перед и после работы в кабинете.
- *ВНИМАНИЕ!* Постарайтесь не использовать ABS-пластик. При нагреве выделяются ядовитые пары. Вдыхать такие испарения вредно для здоровья. Конечно, их количество не превышает предельно допустимые концентрации вредных веществ, но зачем вам повышать риски. *Используйте PLA-пластик*, лучше российского про-

изводителя, на сайте которого есть все сертификаты безопасности и информация по выбросам частиц во время печати. Сертификат гарантирует, что в нормальных условиях эксплуатации указанный пластик является безопасным. Конечно, производители оборудования планируют выпуск более безопасных 3D-печатных устройств с собственной системой угольной фильтрации, но пока в школах их нет. *Поэтому не находитесь долго в помещении, в котором происходит процесс печати. Выполняйте все рекомендации преподавателя.*

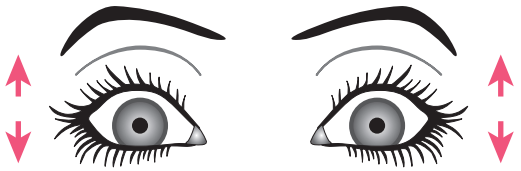
- Материалы для печати могут быть небезопасны. Критически важный фактор риска связан с наночастицами (диаметром менее 1 микрона), которые могут проникать непосредственно в альвеолы лёгких и эпидермис. Время, за которое содержание наночастиц в воздухе возвращается к обычному уровню, составляет от 10 до 30 мин после окончания процесса 3D-печати. *Не находитесь долго в помещении, в котором происходит процесс печати. Выполняйте все рекомендации преподавателя.*
- Во время печати рабочая температура экструдера превышает 200 °C. Можно получить ожог. Значение температуры указано на дисплее принтера. *Нельзя во время работы принтера прикасаться к элементам, которые нагреваются до высоких температур. Будьте внимательными и соблюдайте все правила поведения в кабинете повышенной опасности.*
- 3D-принтеры имеют очень много движущихся частей. Это двигатели, шкивы, резьбовые стержни, каретка, вентиляторы и т. д. Всё это может причинить вам много неприятностей. *Во время работы 3D-принтера необходимо избегать попадания посторонних предметов в подвижные механизмы принтера. Не допускайте контакта с движущимися частями принтера во время его работы! Следите за своей причёской.*
- При удалении поддержки образуются маленькие острые фрагменты. Они могут причинить вред вашим глазам. *Всегда используйте защитные очки при удалении поддержки с вашей модели.*
- Работа с расплавленным металлом и горячим паяльником может привести к ожогу. *Если вы пользуетесь паяльной станцией, то обязательно прочтите инструкцию по работе с ней. Используйте защитные очки и другие средства защиты, которые перечислит ваш преподаватель.*

## Упражнения для глаз

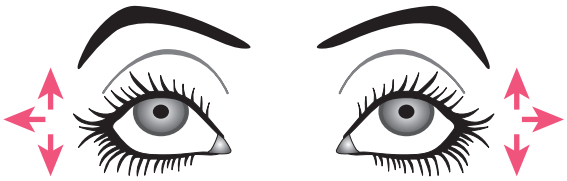
Если в ходе работы вы неожиданно почувствовали неприятные ощущения в глазах, то немедленно прекратите работу на компьютере,

ноутбуке или планшете и выполните небольшой *комплекс упражнений для глаз*.

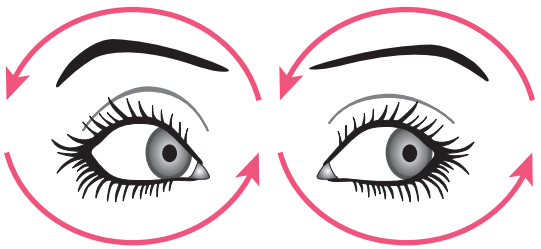
**1.** Сильно закройте глаза на 2 секунды



**2.** Быстро моргайте на протяжении 30 секунд

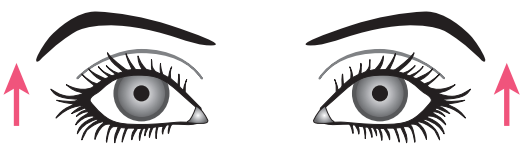


**3.** Посмотрите вверх, вниз, влево, вправо. Повторите 5 раз



**4.** Делайте глазами круговые движения в одну сторону, потом в другую. Повторите 5 раз

**5.** Закройте глаза на 5 секунд



**6.** Откройте глаза

## Рекомендации по организации рабочего места

Ваша работа состоит из трёх этапов: программирование и моделирование на компьютере, непосредственно работа с роботом, а также печать на 3D-принтере.

Во время занятий важно, чтобы рабочее место было максимально удобным. Организуйте для работы пространство рядом с компьютером и предусмотрите свободное место для сборочной площадки или тестирования робота. Размеры примерно 60 × 80 см.

При работе за компьютером обратите внимание на стандартные рекомендации по организации рабочего места. К сожалению, нет однозначного рецепта, который бы подошёл всем без исключения, и многие вопросы, конечно, индивидуальны. Главное, не игнорировать элементарные советы, иначе вас, вероятно, ждут:

- близорукость;
- искривление позвоночника;
- боли в суставах.

Соблюдайте несколько правил, чтобы этого избежать.

- Спину и шею держите ровно. Не наклоняйте шею вперёд, чтобы не провоцировать проблемы с кровообращением.
- Правильная посадка предполагает, что стопы стоят на полу ровно, а колени согнуты под углом 90°.
- Руки свободно лежат на клавиатуре, не сгибаясь в запястьях. Локти согнуты под углом 90°.
- Оптимальное расстояние от глаз до монитора — 50–70 см. Вы не должны опускать или сильно закидывать голову.

На что стоит обратить пристальное внимание при работе на 3D-принтере? 3D-принтер, как и любое устройство, может причинить вред человеку. Необходимо с вниманием отнестись к своему рабочему месту, изучить руководство по эксплуатации оборудования, соблюдать элементарные меры предосторожности. Тогда 3D-принтер принесёт только радость творчества и положительные эмоции.

Сохраняйте чистоту рабочего места — убирайте мелкие кусочки пластмассы сразу, а не в конце урока.

Обратите внимание, что все электронные компоненты робота открыты, то есть существует повышенная возможность случайного короткого замыкания. Вам оно не повредит, но роботу противопоказано однозначно.

# Глава 1
# НЕОБХОДИМОЕ ОБОРУДОВАНИЕ

## 1.1. Инвентаризация

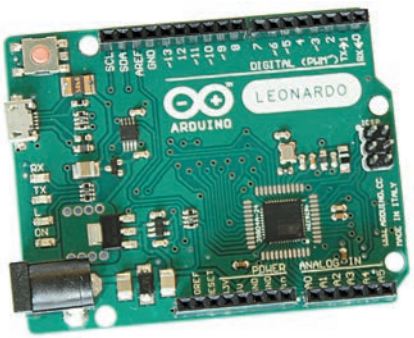
Для успешного прохождения курса по созданию робота на платформе Arduino необходимы комплектующие, представленные далее. Для более подробного изучения воспользуйтесь поиском в Интернете.

www

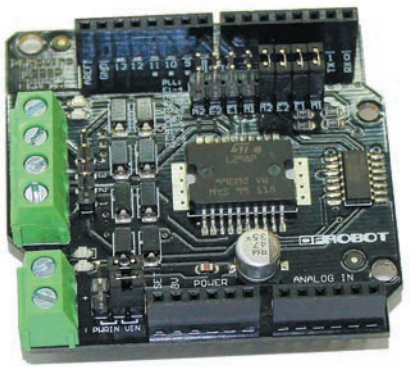
### Задание 1

Изучите необходимое оборудование и программное обеспечение.

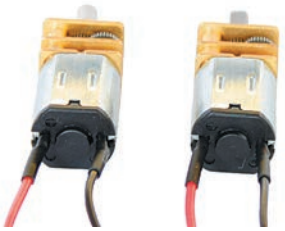
**Плата Arduino Leonardo**
Цифровые входы/выходы: 20.
Порты с поддержкой ШИМ: 7.
Порты, подключённые к АЦП: 12.
Флеш-память: 32 Кб.
ОЗУ: 2 Кб.
Тактовая частота: 16 МГц.



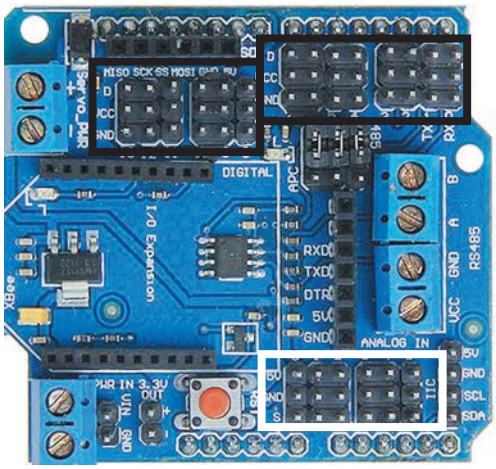

**Плата управления моторами**
Необходима для более простого управления моторами. Выходы под двигатели выполнены в виде разъёма (клеммника) с винтом. Для управления моторами на Arduino используются четыре контакта: 4, 5, 6 и 7.



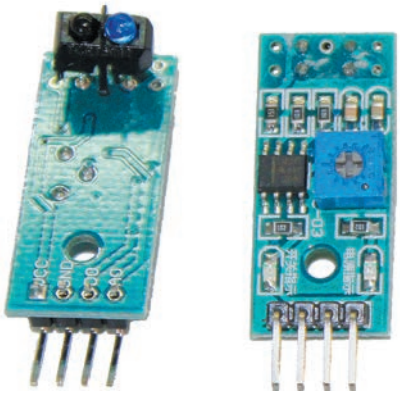

**Два мотора N20**
Особенность моторов — широкий диапазон поддерживаемого питания: 1,5–12 В. Контакт «+» соответственно обозначен. При 9 В максимальная скорость составляет 150 оборотов в минуту.



**Плата расширения ввода/вывода**
Позволяет просто и быстро подключать различные устройства к Arduino, например датчики, сервомоторы, модули для SD-карт памяти, Bluetooth-модули и т. д. Цветом выделены те элементы, которые будут задействованы: белым — аналоговые входы для датчиков, красным — цифровые входы/выходы.

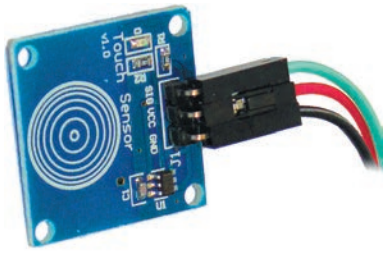



**Датчики цвета поверхности (6 шт.)**
Работают в двух режимах одновременно: определяют цвет поверхности в градации от 0 до 1023 единиц и наличие или отсутствие белого объекта.

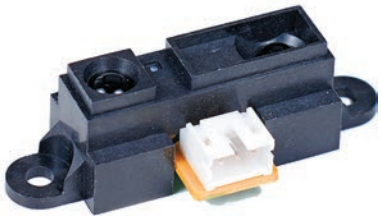


**Ультразвуковой дальномер**
Генерирует импульсы на частоте 40 кГц и слушает эхо. По времени распространения звуковой волны туда и обратно можно определить расстояние до объекта.


**Датчик касания**
Это цифровой датчик — когда нет касания сенсора, он передаёт логический нуль, когда есть касание — логическую единицу.



**Инфракрасный дальномер**
Модель Sharp GP2Y0A021. Сенсор определяет расстояние по отражённому лучу света в инфракрасном спектре.



**Источник питания**
Или NiMH-аккумулятор, или батарейка типа «Крона». Если есть зарядное устройство, то лучше использовать аккумулятор.

**Винты, шайбы, гайки М3 х 16**
Необходимое количество — по 30 шт.
Также могут подойти винты длиной 12 мм.

**Провода (40 шт.)**
Подходят для соединения, если контакты выполнены в виде штырьков.

**Канцелярские резинки (6 шт.)**
Диаметр — 50 мм; толщина — 1 мм или 2 мм. Это будут необычные шины для колёс нашего робота.

**Стальные шарики (2 шт.)**
Диаметр — 1 см. На их основе будут сделаны две шаровые опоры для третьего и четвёртого колёс робота.

**Кабель micro-USB (1 шт.)**
Стандартный USB-кабель с разъёмом типа micro-USB. У вашего смартфона, скорее всего, такой же.

**Кабель и переключатель питания**
Кабель для подачи питания и маленькая кнопка (10 × 15 мм), фиксирующая положения: включено и выключено. Переключатель также называют просто выключателем.

## 1.2. Сделай лучше

### Задание 2

Если соединительные провода (рис. 1) предоставлены в едином шлейфе, то разделите их или поштучно, или по три провода.

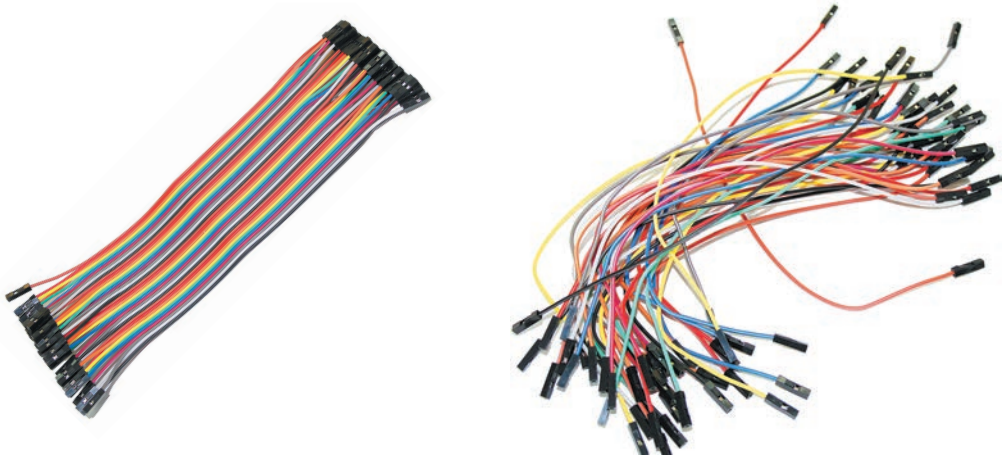


**Рис. 1.** Соединительные провода

### Задание 3

Необходимо, разрезав красный провод на кабеле для источника питания, припаять переключатель (рис. 2). Если вы хотите выполнить пай-

**Рис. 2.** Кабель для источника питания типа «Крона» с выключателем

ку самостоятельно, то это несложно. Посмотрите 13-минутное видео, как припаять, например, провода к моторам (https://youtu.be/fV-uPN4tA8w).

## 1.3. Техника безопасности

Первая смерть человека от робота была зафиксирована 25 января 1979 г. на заводе «Форд» в США. Из-за несоблюдения техники безопасности 25-летнего рабочего сборочного цеха робот-манипулятор ударил по голове.

Вопросы техники безопасности были рассмотрены в начале учебного года на первом уроке технологии и на первом уроке информатики. Помните, за свою безопасность отвечаете вы сами.

С одной стороны, все задания этой книги не несут в себе реальной опасности для вас, и батарейка с напряжением 9 В не представляет какой-либо угрозы. С другой — присутствует опасность для электронных компонентов. Из таких опасностей можно выделить:

- статическое электричество;
- короткое замыкание;
- отношение «да нормально всё».

*Статическое электричество* представляет большую угрозу для микросхем. Электростатический разряд, источником которого можете стать именно вы, способен приводить к образованию токов, которые или сразу выведут из строя чувствительную электронику, или нанесут повреждения, вызывающие неправильную работу микросхемы. Последний случай самый неприятный, так как вы будете долго искать причину того, почему у вас что-то не работает, а возможно, что уже и не должно ничего работать.

*Короткие замыкания* очень опасны! Если вы, например, закоротили автомобильный аккумулятор, то сила тока будет настолько большой, что батарея может даже взорваться, выплеснув на вас кислоту. Если вы случайно проводом соединили контакты «+» и «–» батарейки 9 В, то не стоит ждать, когда произойдёт значительное тепловыделение и расплавятся провода. Будьте внимательны!

Самый страшный по нанесённому ущербу — подход к работе «*да нормально всё*». Помните об ответственности!

### Задание 4

Проанализируйте информацию, найденную по запросу "*В Протоне перепутали датчики*" (кавычки в запросе сохраните).

## 1.4. Что должно получиться по окончании курса

После прохождения курса у вас получится робот, изображённый на рисунке 3. Вы научите его ездить прямо, определять расстояние до объекта, двигаться быстро и плавно по заданной траектории и многому другому.

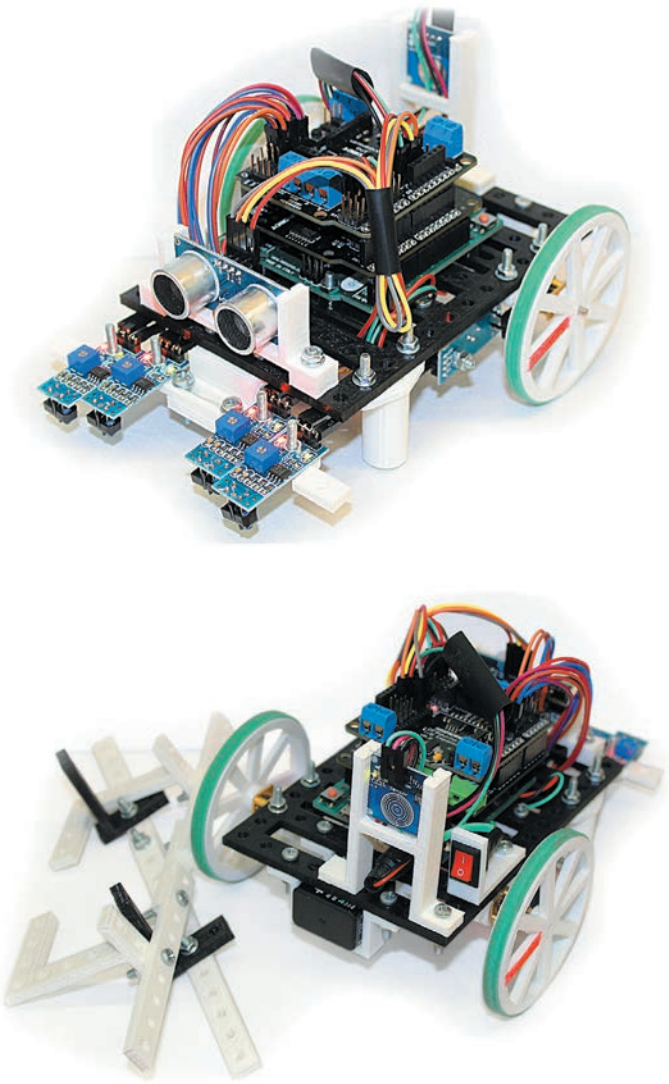


**Рис. 3.** Модель робота, которую необходимо собрать

Ваше оборудование может немного отличаться от указанного характеристиками, цветом. Но размер платы Arduino (или UNO, или Leonardo) останется неизменным (рис. 4).

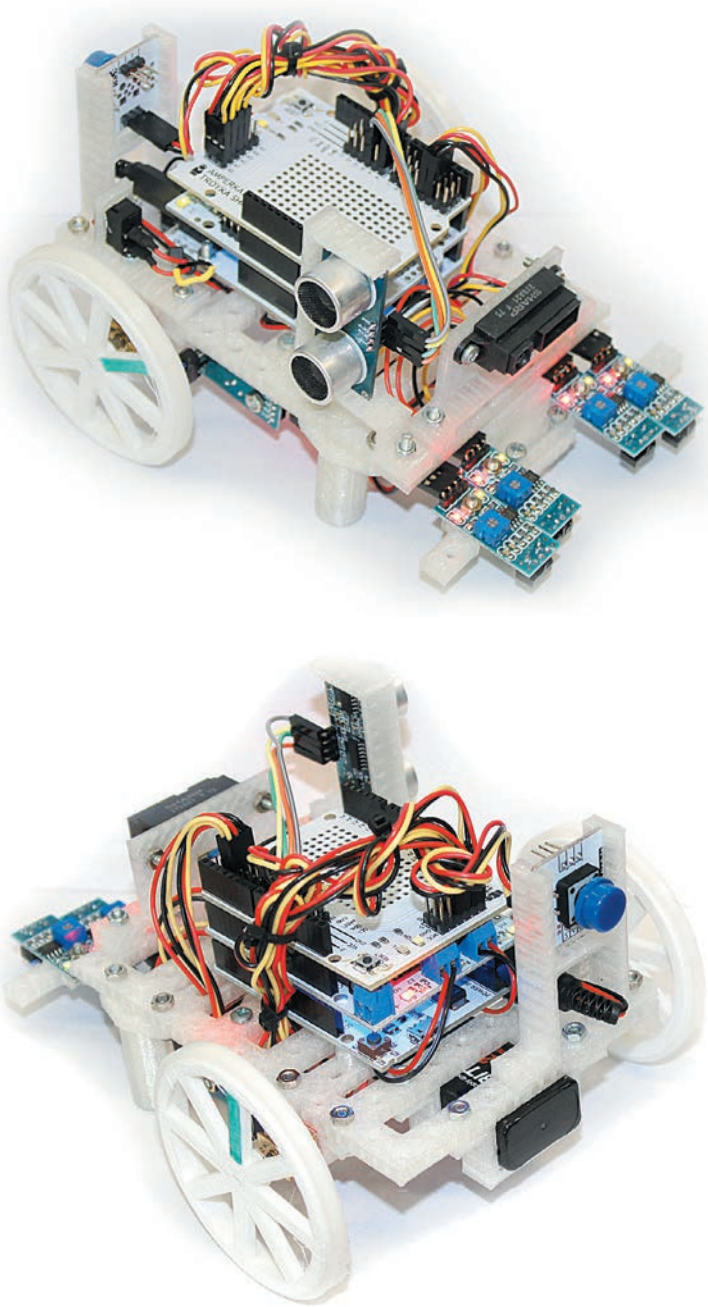


**Рис. 4.** Модель робота на основе других комплектующих

# Глава 2
# ПЛАТФОРМА ARDUINO

## 2.1. Платы Arduino

Когда говорят «платформа», имеют в виду систему, состоящую из аппаратного и программного обеспечения. *Аппаратной частью* Arduino является плата на основе микроконтроллера фирмы Atmel, подключаемая к компьютеру через USB. Эта плата может считывать входной сигнал, например свет на датчике, палец на кнопке или сообщение в Twitter, и превращать его в выходной сигнал, например включение/выключение двигателя, светодиода, публикация чего-то в Интернете. При этом разновидностей плат, объединённых торговой маркой Arduino, очень много, и с каждым годом их число увеличивается. Самая популярная — это Arduino UNO. А чуть более современная — Arduino Leonardo (рис. 5).

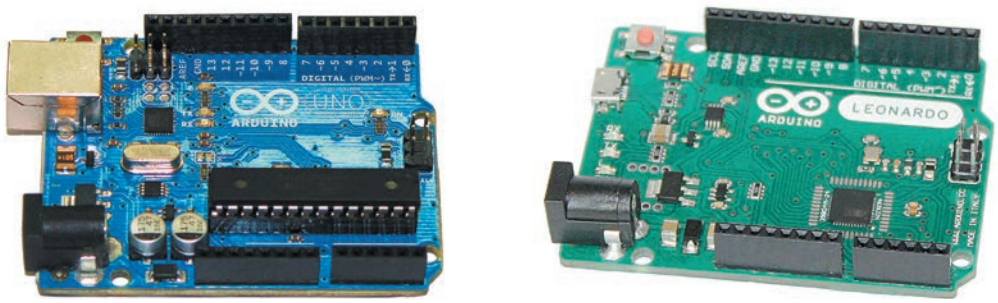



**Рис. 5.** Платы Arduino UNO и Arduino Leonardo

Также очень много совместимых плат, которые производятся по всему миру, включая, конечно, Россию. Одна из первых российских плат, Iskra Neo, по сути своей является Arduino Leonardo (рис. 6).

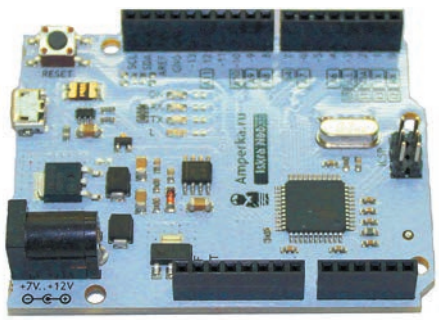


**Рис. 6.** Плата Iskra Neo

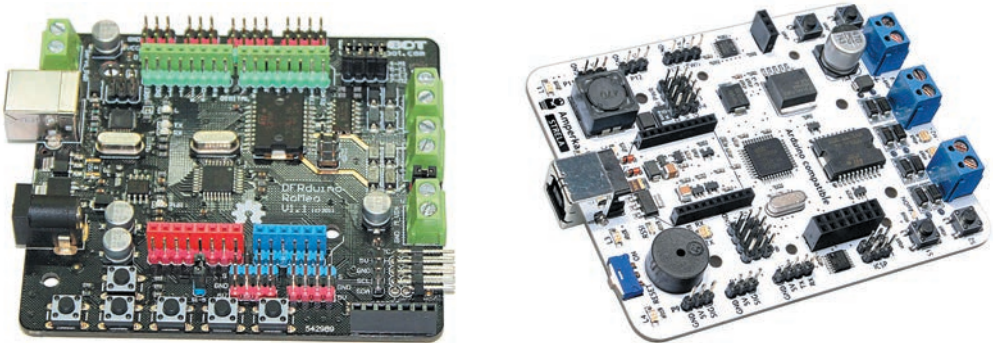


**Рис. 7.** Платы RoMeo и STRELA

Также выпускаются платы, специально предназначенные для любительской робототехники, как, например, на рисунке 7.

Вторая часть платформы — *программное обеспечение* Arduino IDE. Аббревиатура IDE (Integrated Development Environment, интегрированная среда разработки) — это комплекс программных средств, используемый для разработки программного обеспечения. Программируется Arduino на языке C++.

Arduino представляет собой открытую платформу, позволяющую собирать разнообразные электронные устройства творческим людям, программистам, дизайнерам и всем новичкам, кому нравится конструировать собственные гаджеты. Она разрабатывалась в Университете проектирования взаимодействий (г. Ивреа, Италия). *Проектирование взаимодействия* (англ. *Interaction Design*) — дисциплина дизайна, занимающаяся проектированием интерактивных, обладающих интерфейсом цифровых изделий, систем, сред, услуг. Как инструмент для обучения студентов этого университета и была создана платформа, совершившая революцию в электронных самоделках.

А самое революционное и масштабное творение на основе Arduino — это проект RepRap — создание открытых 3D-принтеров на основе Arduino.

### Задание 5

www Посмотрите 15-минутное выступление одного из создателей Arduino Массимо Банци на конференции TED в 2013 году (https://youtu.be/PA51vikUMVs).

## 2.2. Arduino Leonardo

Производителей Arduino-совместимых плат очень много. Поэтому встаёт вопрос о том, какую выбрать. Мнений на этот счёт не меньше, чем производителей. Большинство рекомендаций в сети Интернет относятся к плате Arduino UNO.

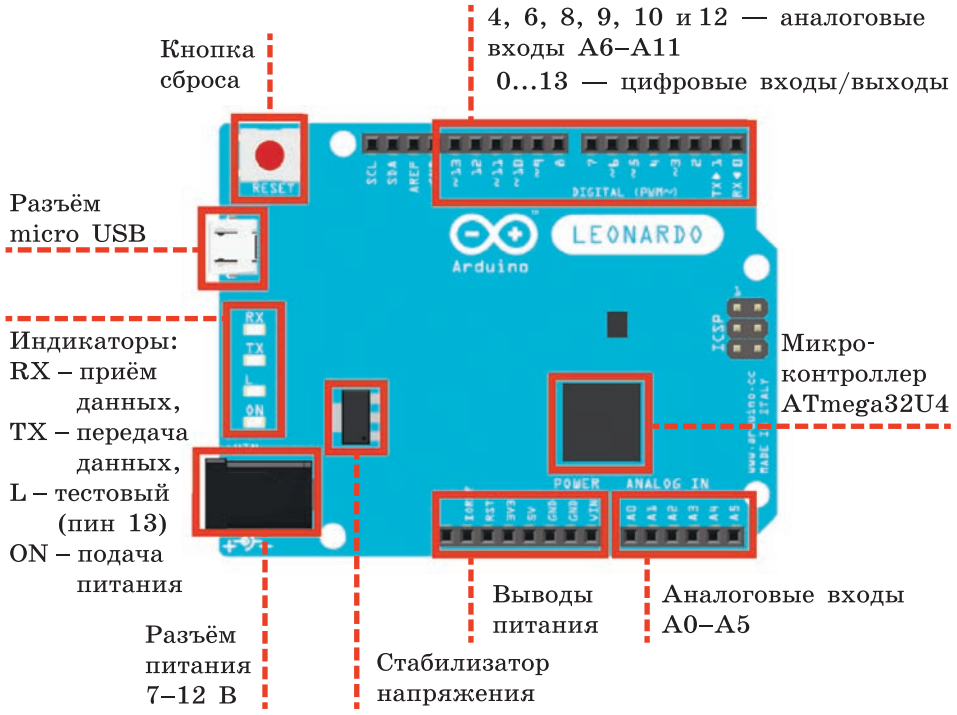



**Рис. 8.** Схема расположения основных элементов (Leonardo)

Мы же выберем плату Leonardo, основные элементы которой представлены на рисунке 8. И вот по каким причинам. На плате UNO имеется 14 цифровых контактов, но контакты 0 и 1 заняты соединением по последовательному порту, например, когда плата подключена к компьютеру или Bluetooth. То есть реально доступных только 12. У Leonardo можно задействовать все 14.

Кроме того, на UNO всего шесть аналоговых входов, в то время как у Leonardo их 12. Контакты 4, 6, 8, 9, 10 и 12 подключены к АЦП (аналого-цифровой преобразователь) и, следовательно, могут быть использованы как аналоговые входы с номерами A6–A11 в дополнение к обычным A0–A5.

### Задание 6

Изучите описание платы Arduino Leonardo по ссылке: http://wiki.amperka.ru/продукты:arduino-leonardo.

## 2.3. Размеры

Нам предстоит смоделировать шасси для робота, поэтому важной информацией является расположение отверстий для крепления пла-

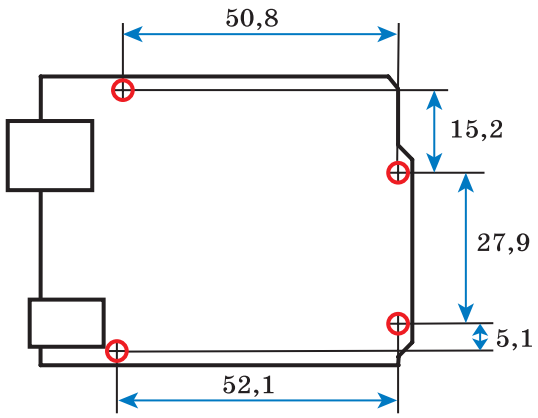



**Рис. 9.** Расположение отверстий для крепления (мм)

ты. У Leonardo и UNO они расположены в одних и тех же местах (рис. 9).

## 2.4. Платы расширений

Одной из особенностей Arduino является наличие большого количества плат расширения (англ. *shields*). Это платы, которые ставятся поверх Arduino, чтобы дать ему новые возможности. Существуют платы расширения для управления моторами, для быстрого подключения датчиков и управляемых элементов (рис. 10), для подключения сервомоторов, для подключения к локальной сети и Интернету, для получения координат со спутников и многие другие.

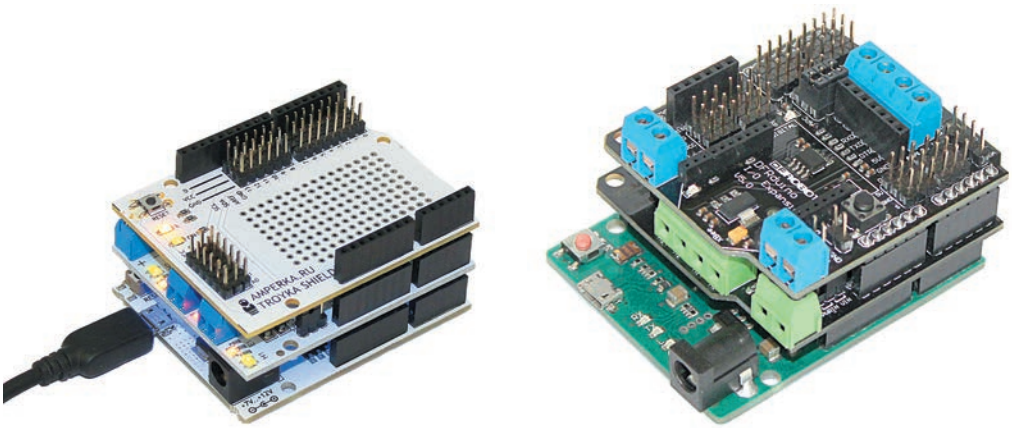


**Рис. 10.** Примеры установленных плат расширений

Нас особенно интересуют две такие платы: плата расширения ввода/вывода (рис. 11, 12) и плата управления моторами (рис. 13, 14).

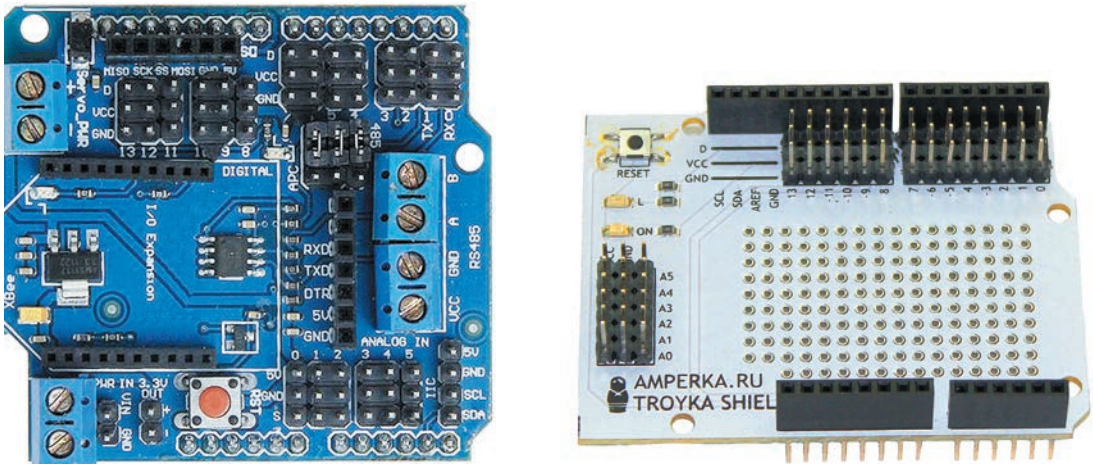



**Рис. 11.** Платы расширения ввода/вывода

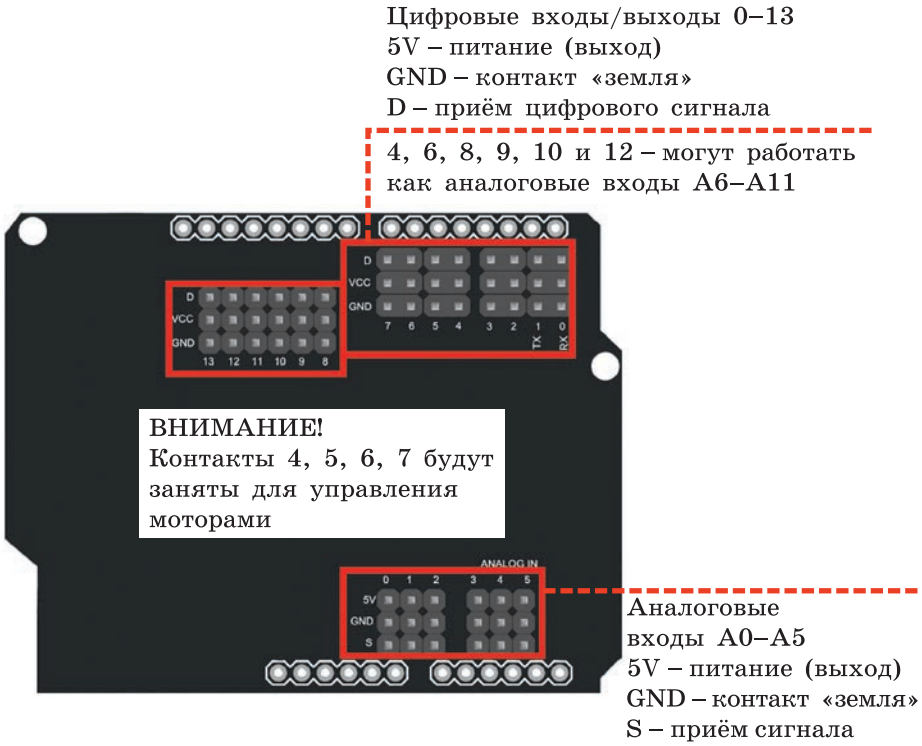



**Рис. 12.** Плата расширения ввода/вывода

## Задание 7

Изучите схему основных подключений платы расширения ввода/вывода (см. рис. 12). Обратите внимание на маркировку контактов.

## Задание 8

Изучите схему основных подключений платы управления моторами (см. рис. 14).

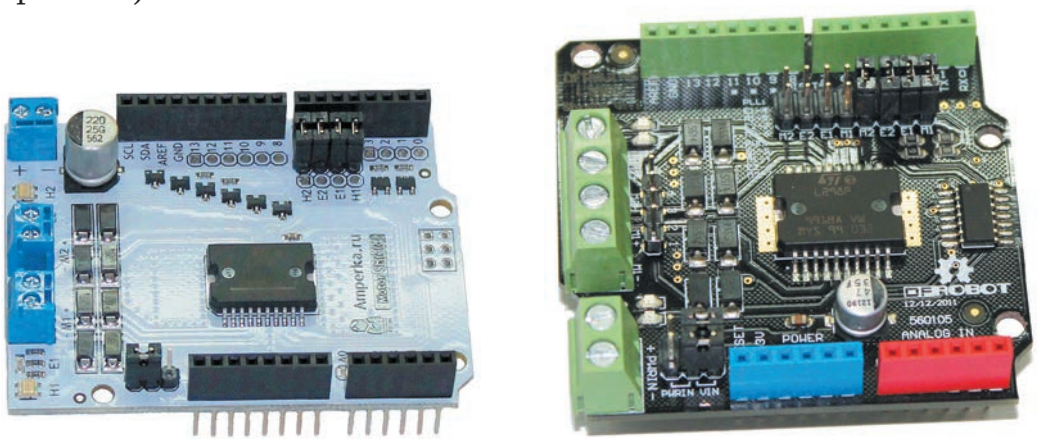


**Рис. 13.** Платы управления моторами

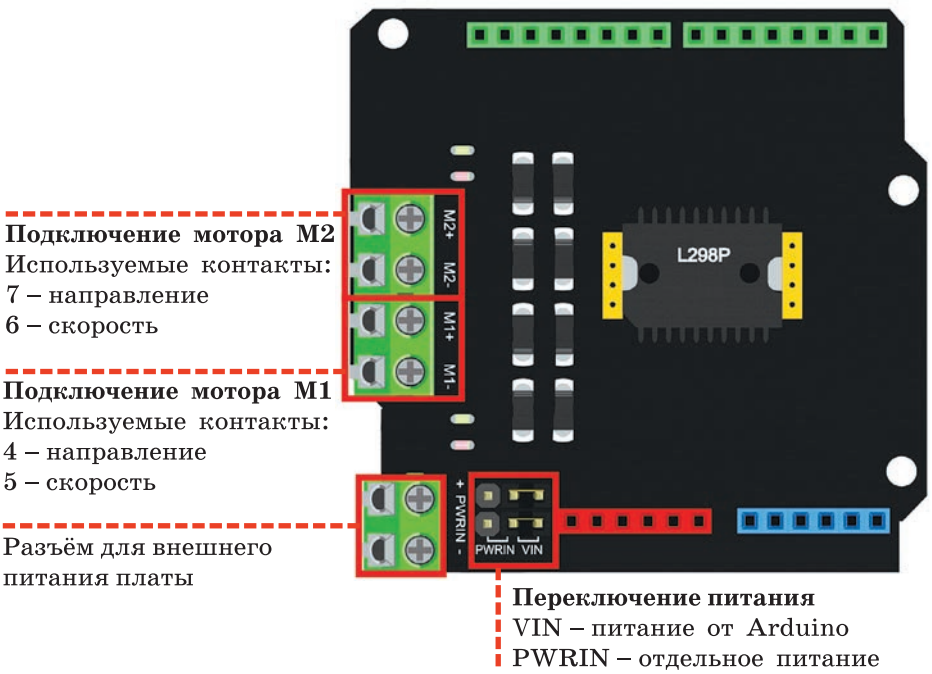



**Рис. 14.** Схема платы управления моторами

Обратите внимание на переключатель питания. В нашем случае источником питания является аккумуляторная батарея типа «Крона», и подключается она к разъёму на плате Arduino. Следовательно, переключатель (две перемычки) на плате должен быть в положении VIN.

# Глава 3
# МОДЕЛИРУЕМ ШАССИ

## 3.1. Модель колёс

Итак, вы изучили комплектующие, на основе которых предстоит собрать робота. Вначале необходимо создать модели всех частей шасси робота: рамы; крепежей моторов, датчиков, батареи, кнопки включения; двух колёс и двух шаровых опор. После чего напечатать модели на 3D-принтере.

Моделировать мы будем в среде OpenSCAD. Это не визуальный редактор, а скриптовый, т. е. модель описывается системой команд.

Заметим, что всё многообразие объектов сводится к трём графическим примитивам: кубоиду (прямоугольному параллелепипеду), шару и цилиндру, а также к их преобразованиям.

### Задание 9

Изучив примеры на рисунках 15–21, создайте заготовку для будущей модели колёс. В OpenSCAD клавиша F5 — предварительный просмотр модели, но удобнее использовать иконку *Предпросмотр* на панели инструментов.

Сначала создаём заготовку для обода колеса.

```
1  //цвет задаётся для удобства
2  color ("red")
3  cylinder (7, 32, 32, true) ;
4  color ("green")
5  cylinder (7+2, 28, 28, true);
6  //первое число – высота
7  //второе – нижний радиус
8  //третье – верхний радиус
9  //все числа – это миллиметры
10 //true – разместить объект в
     начале координат
```

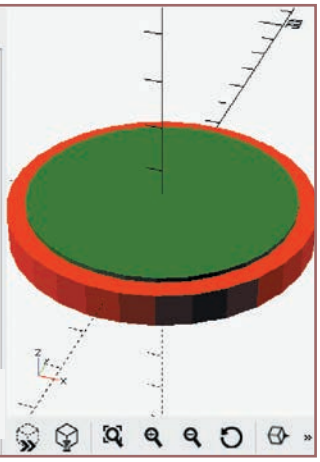


**Рис. 15.** Будущий обод колеса на основе цилиндров (`cylinder`)

В коде будут указываться комментарии к новым командам. Для получения подробной справки выберите в меню *Справка* пункт *Шпаргалка*. В ней собраны все команды и приведены примеры.

Далее создаём сам обод (рис. 16).

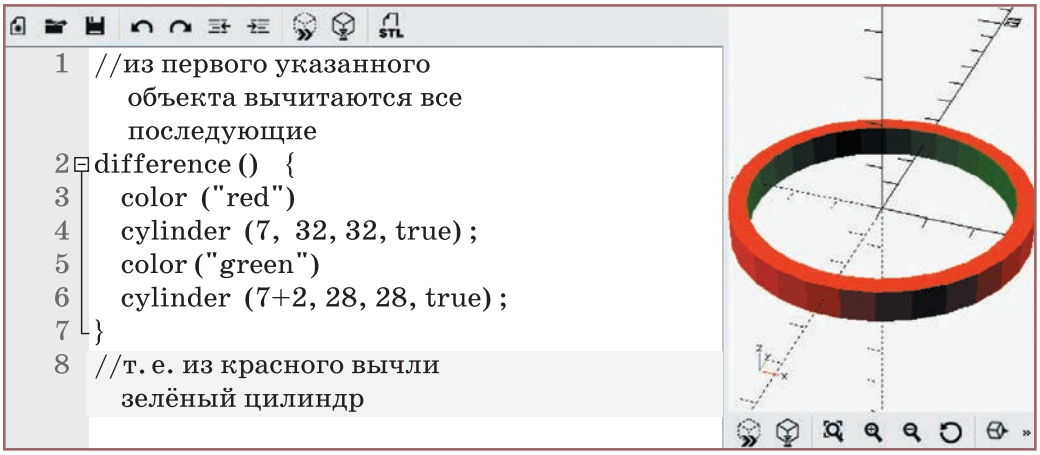



**Рис. 16.** Обод колеса на основе вычитания геометрических тел

Толщина нашего колеса будет 7 мм, а диаметр 64 мм.

Затем формируем одну спицу (рис. 17).

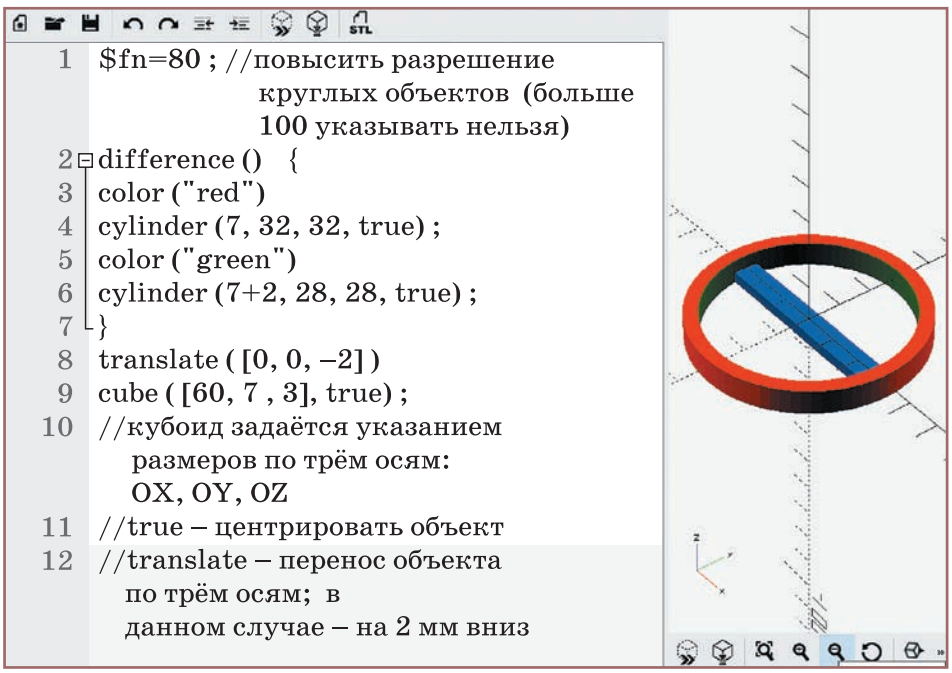



**Рис. 17.** Создание спицы командой `cube`

Спицы все одинаковые. Для их формирования нужно четыре кубоида (рис. 18).

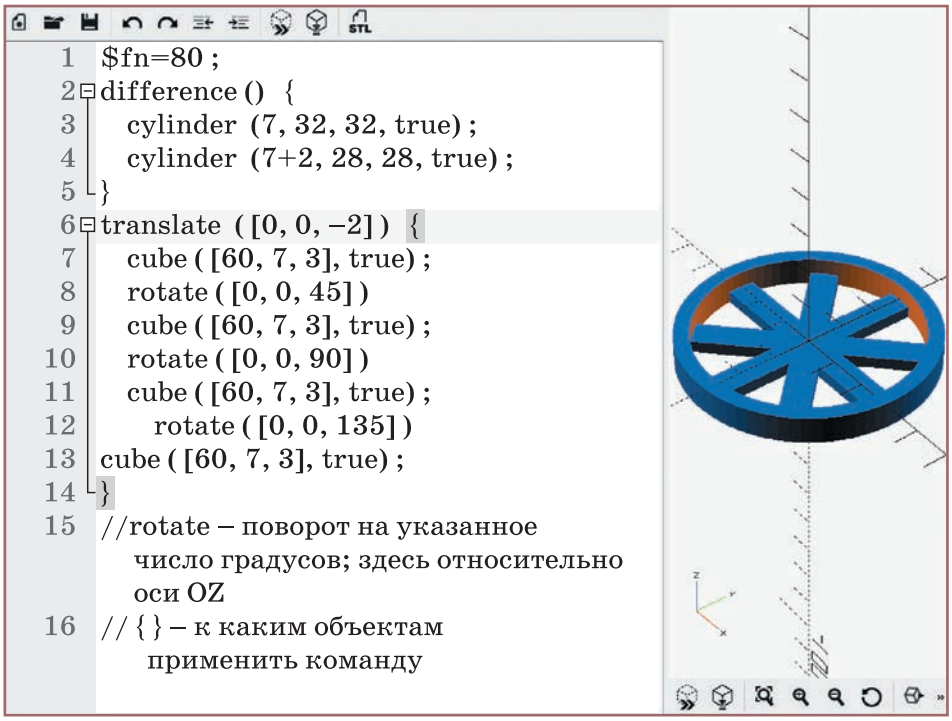



**Рис. 18.** Создание спиц командой `cube` и поворотом (`rotate`)

Моделируем ту часть, в которую будет вставляться ось мотора (рис. 19 и 20).

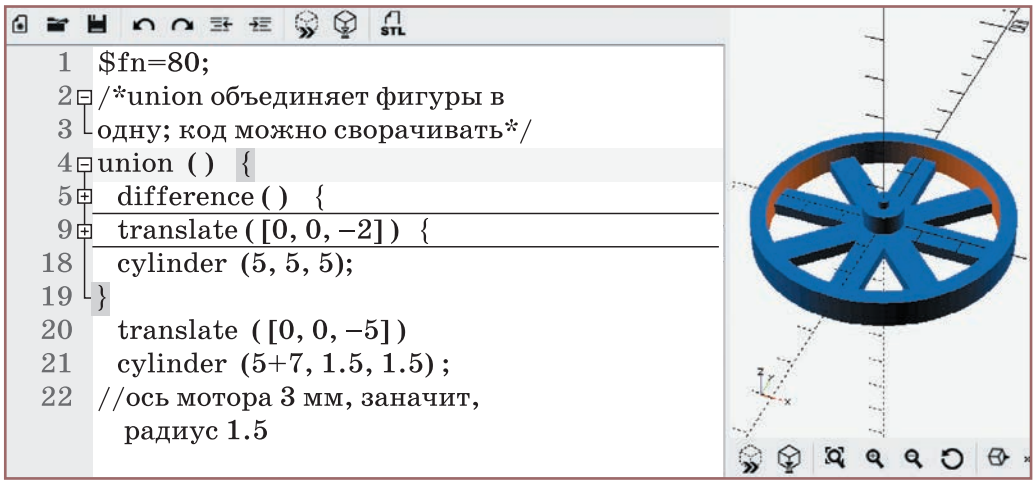



**Рис. 19.** Создание крепежа под ось мотора

```
1  $fn=80 ;
2  difference ( ) {
3    union ( ) {
4      difference ( ) {
8      translate ( [0, 0, –2] ) {
17       cylinder (5, 5, 5) ;
18     }
19     translate ( [0, 0, –5] )
20     cylinder (5+7, 1.5, 1.5) ;
21   }
22   translate ( [1.5, 0, 0.75] )
23   cube ( [1, 3, 8.5], true) ;
24   /*вычитаем из объединения и
       учитываем скос на оси мотора,
       добавляя кубоид
```

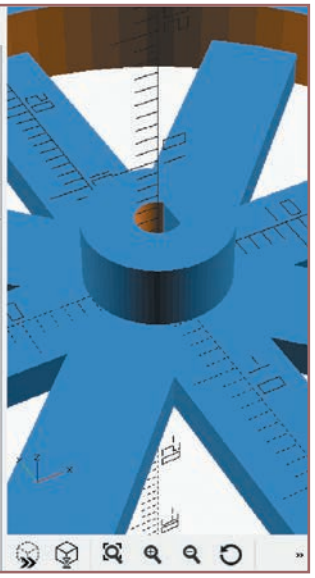


**Рис. 20.** Создание крепежа под ось мотора (со скосом)

Осталось только добавить паз для шин, и модель колеса готова (рис. 21).

```
1  $fn=80 ;
2  difference ( ) {
3    union ( ) {
19     translate ( [0, 0, –5] )
20     cylinder (5+7, 1.5, 1.5) ;
21   /*создаём паз под шины
22     формируем обод и вычитаем*/
23   difference ( ) {
24     cylinder (7–2, 34, 34, true) ;
25     cylinder (7, 31.4, 31.4, true) ;
26   }
27 }
28 translate ( [1.5, 0, 0.75] )
29 cube ( [1, 3, 8.5], true) ;
```

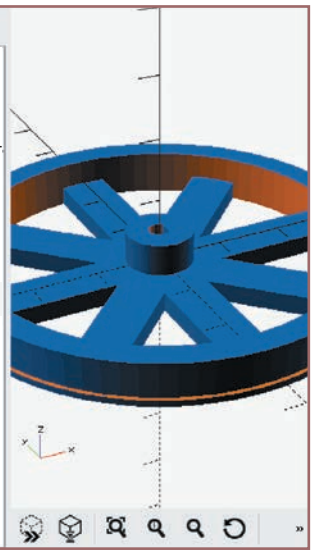


**Рис. 21.** Готовая модель колеса

## Задание 10

Проанализируйте указанный ниже код и создайте модель колеса.

```
$fn=80;
difference () {
 union () {
  difference() {
   cylinder (7,32,32,true);
   cylinder (7+2,28,28,true);
  }
  translate ([0,0,-2]){
   cube ([60,7,3],true);
   rotate([0,0,45])
   cube ([60,7,3],true);
   rotate([0,0,90])
   cube ([60,7,3],true);
   rotate([0,0,135])
   cube ([60,7,3],true);
 }
  cylinder (5,5,5);
 }
 translate ([0,0,-5])
 cylinder (5+7,1.5,1.5);
 difference () {
  cylinder (7-2,34,34,true);
  cylinder (7,31.4,31.4,true);
 }
}
translate ([1.5,0,0.75])
cube ([1,3,8.5],true);
```

В OpenSCAD можно создавать и использовать константы. Кроме того, возможно использование циклов со счётчиком. Например, в модели колеса четыре одинаковых кубоида можно запрограммировать следующим циклом:

```
for(i=[0:45:135]) {
    rotate([0,0,i])
    cube ([60,h,3],true);
   }
```

В квадратных скобках — начальное значение, шаг и конечное значение. То есть переменная *i* будет последовательно принимать значения: 0, 45, 90, 135.

В итоге код получит вид, представленный на рисунке 22.

```
$fn=80 ;  h=7 ;
difference ( )  {
  union ( )  {
   difference ( )  {
     cylinder (h, 32, 32, true) ;
     cylinder (h+2, 28, 28, true) ;
  }
  translate ( [0, 0, –2] )  {
    for (i= [0 : 45 : 135] ) {
      rotate ( [0, 0, i] )
      cube ( [60, h, 3], true) ;
    }
  }
  cylinder (5, 5, 5) ;
}
translate ( [0, 0, –5])
cylinder (5+h, 1.5, 1.5) ;
difference ( )  {
  cylinder  (h–2, 34, 34, true) ;
  cylinder  (h, 31.4, 31.4, true) ;
}
}
translate ( [1.5, 0, 0.75] )
cube  ( [1, 3, 8.5], true) ;
```

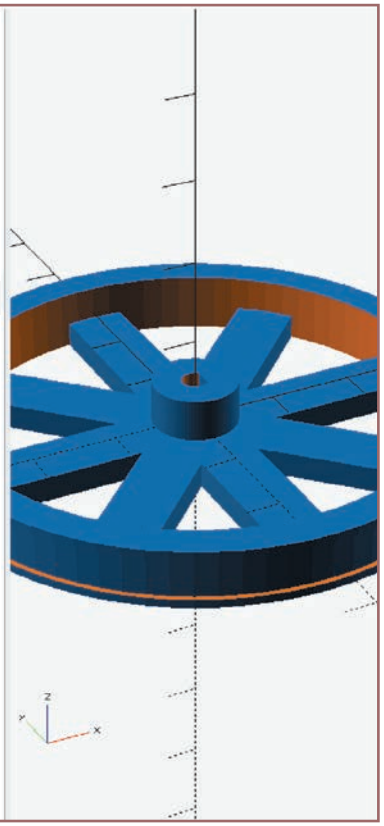


**Рис. 22.** Итоговая модель колеса для робота

## 3.2. Модель рамы

Самая сложная и интересная модель — рама. Необходимо предусмотреть возможность крепежа разных деталей к ней. Поэтому лучше взять за основу рамы балки с отверстиями (рис. 23).

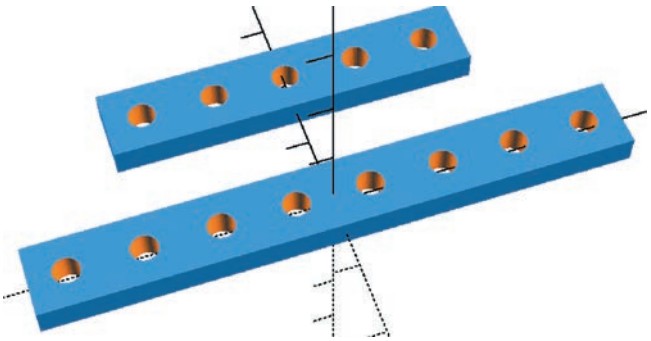
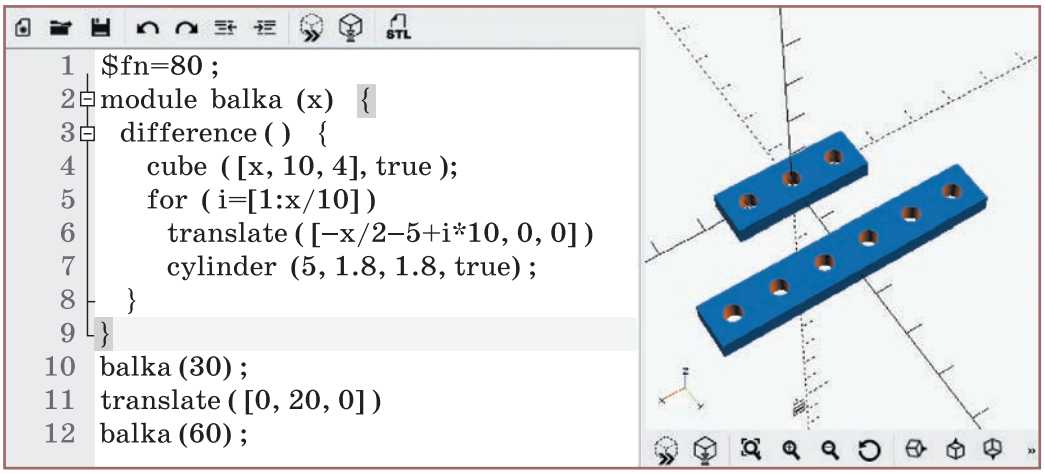


**Рис. 23.** Балка — основа рамы робота

Расстояние между отверстиями составляет 10 мм, ширина балки — 10 мм, высота — 4 мм. Кроме того, балка должна быть параметрической, т. е. мы задаём её длину, а отверстия получаются автоматически. В OpenSCAD есть возможность создавать модули, которые можно вызывать в любой части кода. Это как подпрограммы или процедуры в программировании. Описываются модули командой `module`, а в скобках указывается один или несколько параметров.

Балка состоит из одного кубоида. Пусть $x$ — длина балки или размер кубоида по оси $Ox$. Количество отверстий в 10 раз меньше, чем длина балки, поэтому в цикле получаем значения от 1 до $x/10$. Отверстия — это цилиндры, следовательно, необходимо их смещать командой `translate` по оси $Ox$, но для этого необходимо знать, на сколько миллиметров. Если длина балки кратна 10, то всё легко: перемещаем на половину длины балки ($x/2$) и на половину расстояния между отверстиями (5 мм), затем добавляем в цикле $i \cdot 10$. В итоге получаем описание, представленное на рисунке 24.



```
1  $fn=80 ;
2  module balka (x) {
3    difference ( ) {
4      cube ( [x, 10, 4], true );
5      for ( i=[1:x/10] )
6        translate ( [-x/2-5+i*10, 0, 0] )
7        cylinder (5, 1.8, 1.8, true) ;
8    }
9  }
10 balka (30) ;
11 translate ( [0, 20, 0] )
12 balka (60) ;
```

**Рис. 24.** Модель балки (длина кратна 10)

Однако если задать длину, не кратную 10, то получим результат, который показан на рисунке 25.

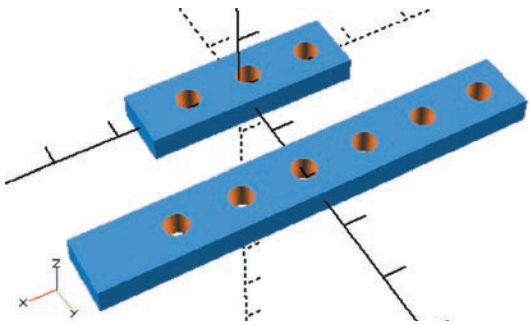


**Рис. 25.** Ошибка при произвольных значения длины балки

В соответствии с рисунком 25 мы видим, что необходимо сдвигать отверстия в положительном направлении оси *Ox*. Причём сдвигать на половину числа единиц ($x\%10$ — количество единиц в числе, $x\%10/2$ — сдвиг).

В итоге решение будет выглядеть следующим образом (рис. 26).

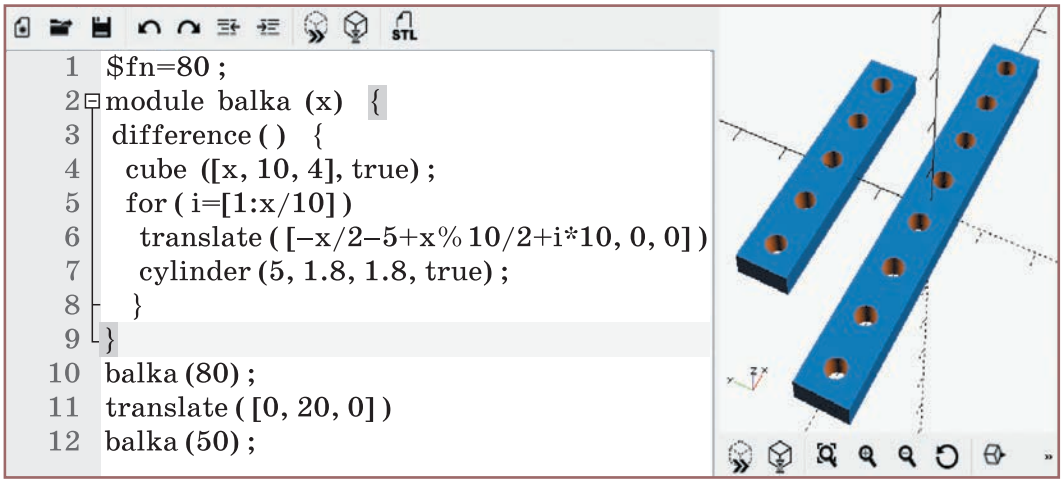



**Рис. 26.** Правильная модель параметрической балки

Нам нужна балка и другого вида (рис. 27).

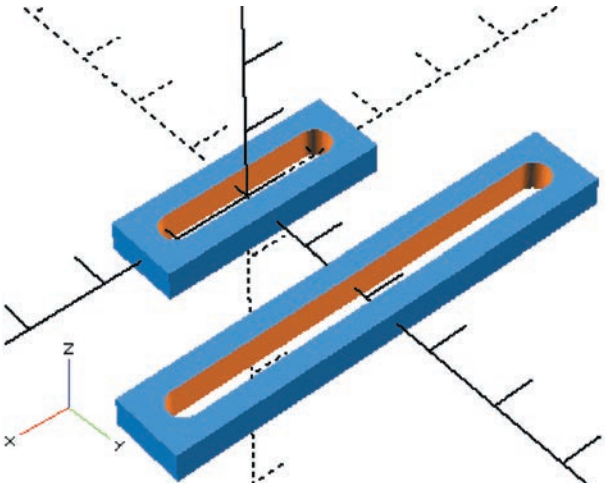
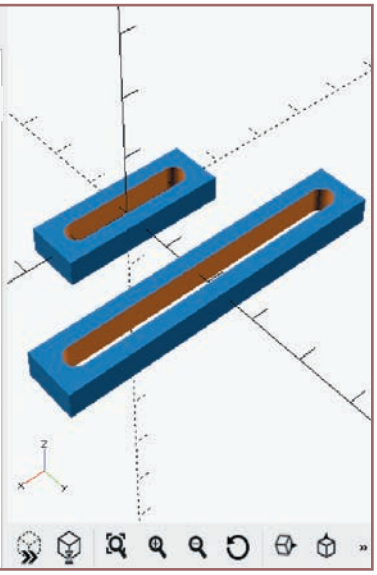



**Рис. 27.** Балка второго вида

Для создания такой балки потребуется ещё одна новая команда из OpenSCAD — `hull`. Она создаёт переход от одной фигуры к другой. Как работает эта команда, посмотрите на рисунке 28. В OpenSCAD есть специальный маркер (`%`), которым удобно подсвечивать объекты, являющиеся вычитаемой частью команды `difference`.

```
1  $fn=80 ;
2  module balka2 (x)  {
3   difference ( )  {
4    cube ( [x, 10, 4], true ) ;
5    hull ()  {
6     translate ( [–x/2–5+x% 10/2+10, 0, 0] )
7     cylinder (5, 2, 2, true) ;
8     translate ( [–x/2–5+x% 10/2+x, 0, 0] )
9     cylinder (5, 2, 2, true) ;
10   }
11  }
12 }
13 balka2 (30) ;
14 translate ( [0, 20, 0] )
15 balka2 (60) ;
```



```
1  $fn=80 ;
2  module balka2 (x)  {
3   difference ()  {
4    cube ( [x, 10, 4], true) ;
5    % hull ()  {
6    translate ( [–x/2–5+x% 10/2 + 10, 0, 0] )
7    cylinder (5, 2, 2, true) ;
8    translate ( [–x/2–5+x% 10/2+x, 0, 0] )
9    cylinder (5, 2, 2, true) ;
10   }
11  }
12 }
13 balka2 (30) ;
14 translate ( [0, 20, 0] )
15 balka2 (60) ;
```

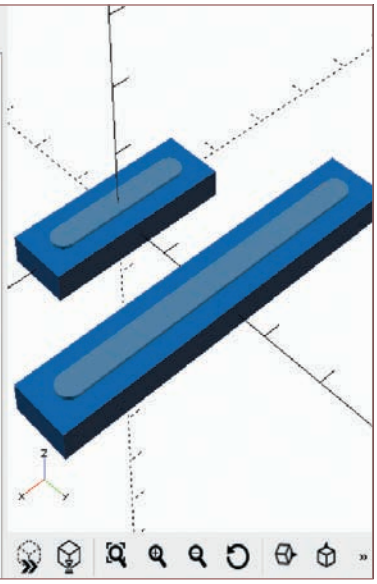
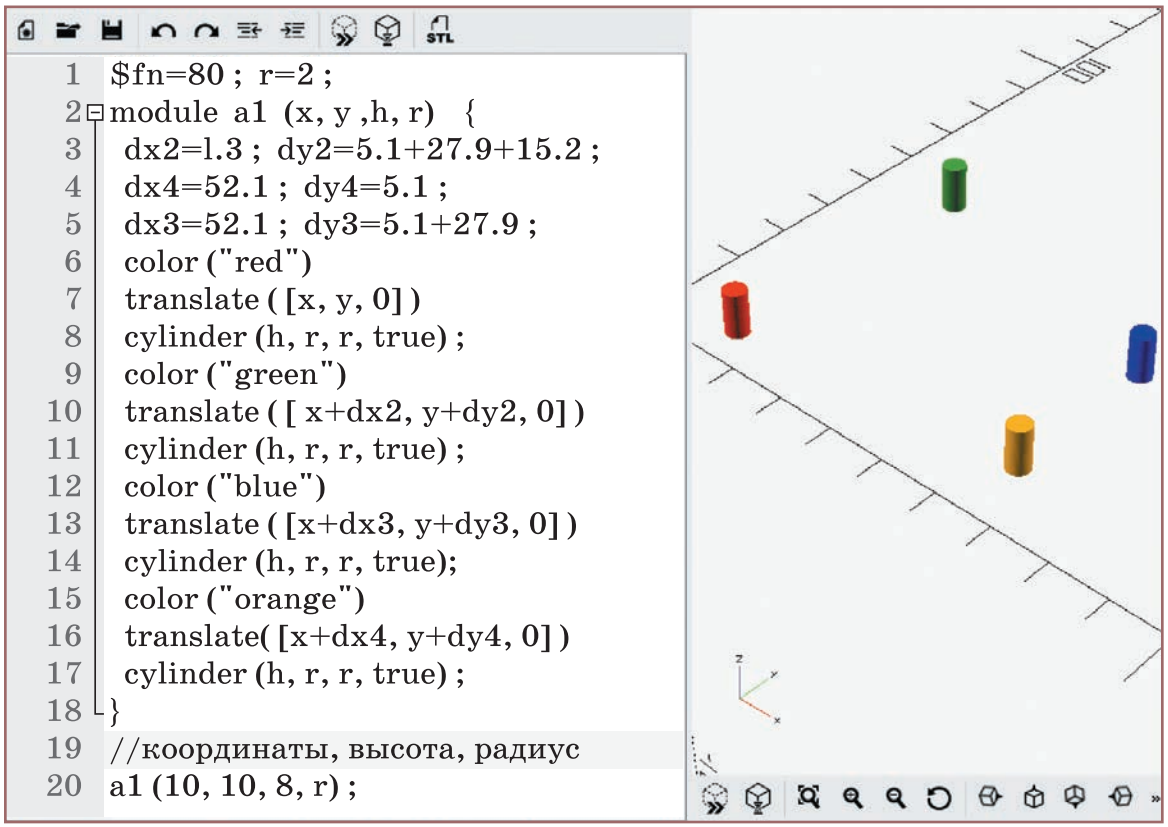


**Рис. 28.** Реализация модели балки второго вида

## Задание 11

Создайте единую модель, содержащую оба модуля с балками. Исследуйте её.

Следующий шаг — создание модуля крепежа для платы Arduino Leonardo. Сначала разберёмся с отверстиями. Это цилиндры. Зафиксируем левое нижнее и относительно его будем располагать все остальные. Где находятся отверстия, мы уже знаем. Таким образом, пишем модуль с четырьмя параметрами: две координаты основного отверстия, высота и радиус (рис. 29).



```
$fn=80 ; r=2 ;
module a1 (x, y ,h, r)  {
 dx2=l.3 ; dy2=5.1+27.9+15.2 ;
 dx4=52.1 ; dy4=5.1 ;
 dx3=52.1 ; dy3=5.1+27.9 ;
 color ("red")
 translate ( [x, y, 0] )
 cylinder (h, r, r, true) ;
 color ("green")
 translate ( [ x+dx2, y+dy2, 0] )
 cylinder (h, r, r, true) ;
 color ("blue")
 translate ( [x+dx3, y+dy3, 0] )
 cylinder (h, r, r, true);
 color ("orange")
 translate( [x+dx4, y+dy4, 0] )
 cylinder (h, r, r, true) ;
}
//координаты, высота, радиус
a1 (10, 10, 8, r) ;
```

**Рис. 29.** Моделирование отверстий для крепежа платы Arduino

Поскольку мы написали модуль (a1), то его удобно использовать и для создания непосредственно стоек платы (рис. 30).

```
$fn=80; r=2; h=8;
module a1 (x, y, h, r) {
  dx2=l.3 ; dy2=5.1+27.9 + 15.2 ;
  dx4=52.1 ; dy4=5.1 ;
  dx3=52.1 ; dy3=5.1+27.9 ;
  color ("red")
  translate ( [x, y, 0] )
  cylinder (h, r, r, true) ;
  color ("green")
  translate ( [x+dx2, y+dy2, 0] )
  cylinder (h, r, r, true) ;
  color ("blue")
  translate ( [x+dx3r, y+dy3, 0] )
  cylinder (h, r, r, true) ;
  color ("orange")
  translate ( [x+dx4, y+dy4, 0] )
  cylinder (h, r, r, true) ;
}
difference () {
  a1 (0, 0, h, r+1.5) ;
  a1 (0, 0, h+2, r) ;
}
```

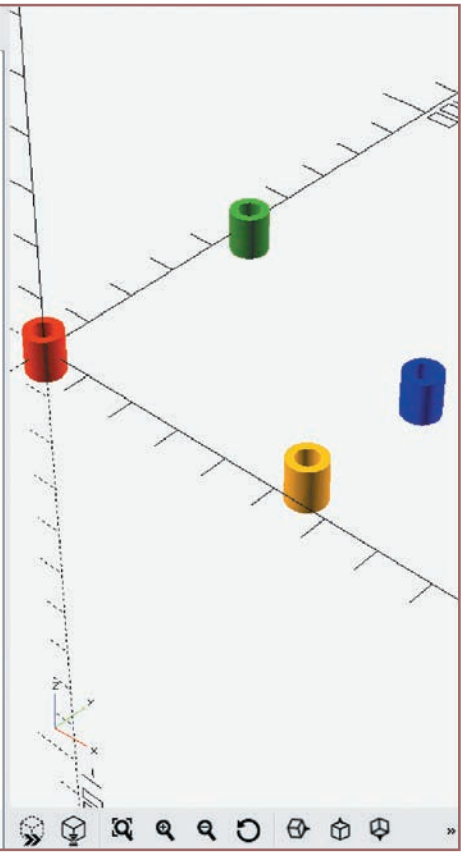


**Рис. 30.** Модель стоек для крепежа платы Arduino

Осталось только использовать модули балок для формирования рамы, изображённой на рисунке 31. Длина рамы 130 мм, ширина 80 мм.

## Задание 12

Создайте модель рамы для робота (рис. 31).

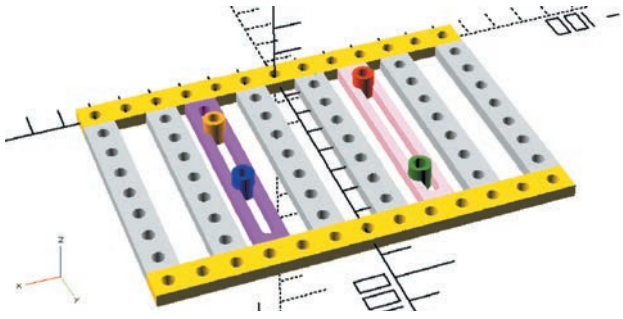


**Рис. 31.** Модель рамы для робота

```
$fn=32;
module balka (x) {
 difference() {
  cube ([x,10,4],true);
  for (i=[1:x/10])
   translate ([-x/2-5+x%10/2+i*10,0,0])
   cylinder (5,2,2,true);
 }
}
module balka2 (x) {
 difference() {
  cube ([x,10,4],true);
  hull () {
   translate ([-x/2-5+x%10/2+10,0,0])
   cylinder (5,2,2,true);
   translate ([-x/2-5+x%10/2+x,0,0])
   cylinder (5,2,2,true);
  }
 }
}
module a1 (x,y,h,r) {
 dx2=1.3;  dy2=5.1+27.9+15.2;
 dx4=52.1; dy4=5.1;
 dx3=52.1; dy3=5.1+27.9;
 translate([x,y,0])
 color("red")cylinder(h,r,r,true);
 translate([x+dx2,y+dy2,0])
 color("green") cylinder(h,r,r,true);
 translate([x+dx3,y+dy3,0])
 color("blue") cylinder(h,r,r,true);
 translate([x+dx4,y+dy4,0])
 color("orange") cylinder(h,r,r,true);
}

//длина и ширина рамы
L=130; D=80;
//координаты x для размещения поперечных балок; оформлены в
виде одномерного массива (вектора)

teleport = [-L/2+5,-40,-10,10,40,L/2-5];
//если в цикле указать через "," — это будет перебор
указанных значений

color ("gold")
for (i=[0,D])
 translate([0,i,0]) balka(L);
color ("silver")
//i будет принимать последовательно значения из указанного
массива (вектора)
for (i=teleport)
 translate([i,D/2,0]) rotate([0,0,90]) balka(D-10);
```

```
translate ([-25,15,2]) {
 difference () {
  a1(0,0,8,2+1.5);
  a1(0,0,8+2,2);
 }
 translate([0,D/2-15,-2])
 rotate([0,0,90])
 color ("pink") balka2(D-10);
 translate([52.1,D/2-15,-2])
 rotate([0,0,90])
 color ("magenta") balka2(D-10);
}
```

## 3.3. Крепежи для остальных элементов

После рамы все остальные крепежи выглядят достаточно просто. При моделировании крепежей обязательно измеряйте элементы (рис. 32).

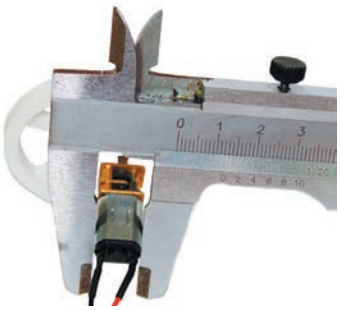
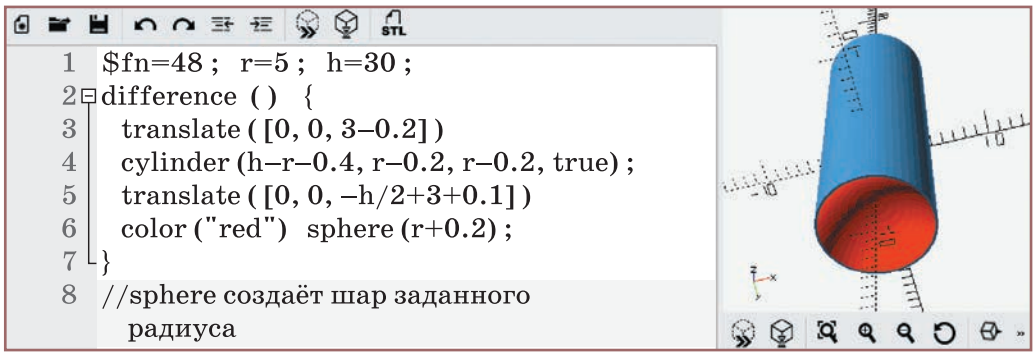


**Рис. 32.** Измерение мотора штангенциркулем

### Задание 13

Создайте модели всех необходимых крепежей (рис. 33–40). При необходимости внесите изменения. Цветовая подсветка графических примитивов облегчит вам понимание кода.



```
1  $fn=48 ;  r=5 ;  h=30 ;
2  difference ( )  {
3    translate ( [0, 0, 3–0.2] )
4    cylinder (h–r–0.4, r–0.2, r–0.2, true) ;
5    translate ( [0, 0, –h/2+3+0.1] )
6    color ("red")  sphere (r+0.2) ;
7  }
8  //sphere создаёт шар заданного
     радиуса
```

**Рис. 33.** Модель шаровой опоры (часть 1)

```
$fn=48 ; r=5 ; h=30 ;
difference () {
  union () {
    color ("red")
    cylinder (h, r+2, r+2, true) ;
    color ("gold")
    hull () {
      translate ( [0, 0, h/2–1.5] )
      cube ( [16, 3, 3], true) ;
      for (i=[–10, 10] )
        translate ( [ 0, i, h/2–1.5] )
        cylinder (3, 5, 5, true) ;
}
}
translate ( [0, 0, –h/2+2] )
color ("green") sphere (r+0.1) ;
translate ( [0, 0, r/2] )
color ("pink")
cylinder (h, r+0.2, r+0.2, true) ;
for ( i=[–10, 10] )
  translate ( [0, i, h/2–1.5] )
  color ("cyan")
  cylinder (5, 2, 2, true) ;
}
```

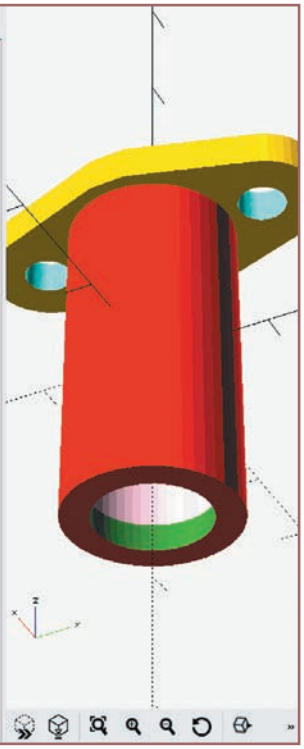


**Рис. 34.** Модель шаровой опоры (часть 2)

```
$fn=48 ;
translate ( [7, 0, 11] )
rotate ( [0, 90, 0] )
difference () {
  color ("red")
  cube ( [25, 12, 3], true) ;
  translate ( [–2, 0, 0] )
  color ("green")
  cube ( [14, 9, 4], true) ;
}
translate ( [–9, 0, 0] ) {
 difference () {
   color ("gold")
   cube ( [30, 12, 3], true) ;
   translate ( [10, 0, 0] )
   color ("cyan")
   cylinder (6, 2, 2, true) ;
   color ("blue")
   translate ( [–10, 0, 0] )
   cylinder (6, 2, 2, true) ;
  }
}
```

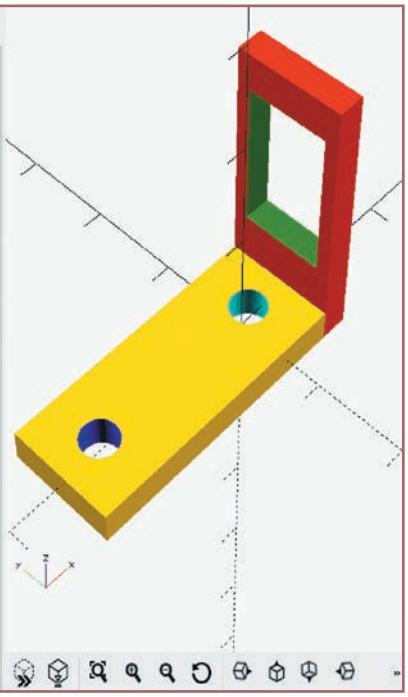
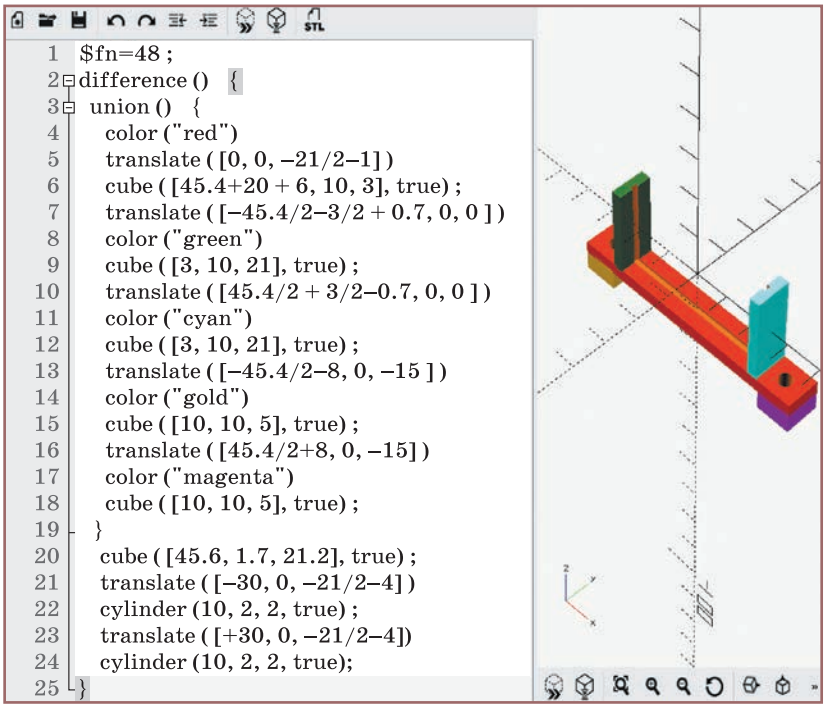
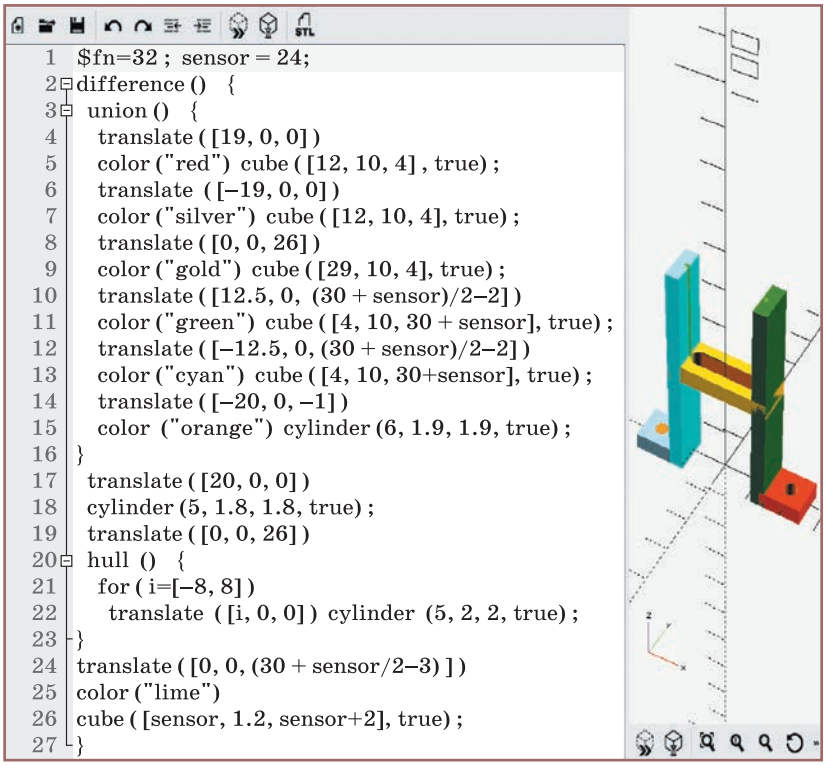


**Рис. 35.** Модель крепежа для кнопки включения/выключения



```
$fn=48 ;
difference () {
 union () {
  color ("red")
  translate ( [0, 0, –21/2–1] )
  cube ( [45.4+20 + 6, 10, 3], true) ;
  translate ( [–45.4/2–3/2 + 0.7, 0, 0 ] )
  color ("green")
  cube ( [3, 10, 21], true) ;
  translate ( [45.4/2 + 3/2–0.7, 0, 0 ] )
  color ("cyan")
  cube ( [3, 10, 21], true) ;
  translate ( [–45.4/2–8, 0, –15 ] )
  color ("gold")
  cube ( [10, 10, 5], true) ;
  translate ( [45.4/2+8, 0, –15] )
  color ("magenta")
  cube ( [10, 10, 5], true) ;
 }
  cube ( [45.6, 1.7, 21.2], true) ;
  translate ( [–30, 0, –21/2–4] )
  cylinder (10, 2, 2, true) ;
  translate ( [+30, 0, –21/2–4])
  cylinder (10, 2, 2, true);
}
```

**Рис. 36.** Модель крепежа для ультразвукового датчика расстояния



```
$fn=32 ; sensor = 24;
difference () {
 union () {
  translate ( [19, 0, 0] )
  color ("red") cube ( [12, 10, 4] , true) ;
  translate ( [–19, 0, 0] )
  color ("silver") cube ( [12, 10, 4], true) ;
  translate ( [0, 0, 26] )
  color ("gold") cube ( [29, 10, 4], true) ;
  translate ( [12.5, 0, (30 + sensor)/2–2] )
  color ("green") cube ( [4, 10, 30 + sensor], true) ;
  translate ( [–12.5, 0, (30 + sensor)/2–2] )
  color ("cyan") cube ( [4, 10, 30+sensor], true) ;
  translate ( [–20, 0, –1] )
  color ("orange") cylinder (6, 1.9, 1.9, true) ;
 }
 translate ( [20, 0, 0] )
 cylinder (5, 1.8, 1.8, true) ;
 translate ( [0, 0, 26] )
 hull () {
  for ( i=[–8, 8] )
   translate ( [i, 0, 0] ) cylinder (5, 2, 2, true) ;
 }
translate ( [0, 0, (30 + sensor/2–3) ] )
color ("lime")
cube ( [sensor, 1.2, sensor+2], true) ;
}
```

**Рис. 37.** Модель крепежа для датчика касания

```
$fn=48 ; t=3 ; d=10;
difference () {
  union () {
    color ("red")
    cube ( [26.4, 10, t], true) ;
    translate ( [26.4/2+t/2, 0, –16.7/2] )
    color ("green")
    cube ( [t, d, 16.7+3], true) ;
    translate ( [–26.4/2–t/2, 0, –16.7/2] )
    color ("magenta")
    cube ( [t, d, 16.7+t], true) ;
    translate ( [–26.4/2–6.5, 0, –16.7] )
    color ("navy")
    cube ( [13, d, t], true) ;
    translate ( [+26.4/2+6.5, 0, –16.7] )
    color ("silver")
    cube ( [13, d, t], true) ;
}
  translate ( [2*d, 0, –16.7] )
  color ("blue") cylinder (5, 2, 2, true) ;
  translate ( [–2*d, 0, –16.7] )
  color ("cyan") cylinder (5, 2, 2, true) ;
}
//размеры батареи: 26.4 x 16.7 мм
```

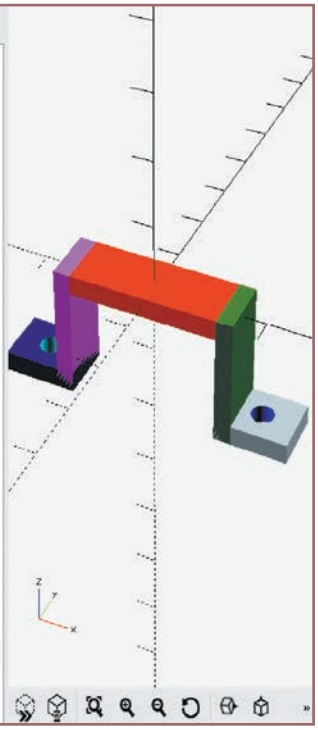


**Рис. 38.** Модель держателя для батареи типа «Крона»

```
$fn=48 ;  delta=0.1 ;
difference () {
  union () {
    intersection () {
    cylinder  (10, 8.2, 8.2, true) ;
    translate ( [0, 0, 0] )
    cube ( [15, 15, 10], true) ;
  }
  for ( i=[–10:20:10] )
    translate ( [i, 3.5, 0] )
    color  ("green")  cube ( [8, 3, 10], true) ;
}
  //под мотор
  cylinder (12, 6+delta, 6+delta, true) ;
  translate ( [0, 5, 0] )
  cube ( [10, 8, 12] , true) ;
  translate ( [0, 7, 0] )
  cube ( [14, 4, 12] , true) ;
  //под винты
  for ( i=[–10:20:10] )
    translate ( [i, 4, 0] )
    rotate ( [90, 0, 0] )
    cylinder (5, 1.7, 1.7, true) ;
}
translate ( [0, –6–delta, 0] )
color ("red")  cube ( [7, 2, 10], true) ;
```

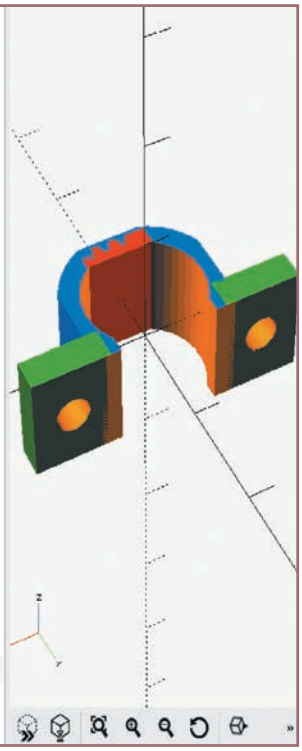
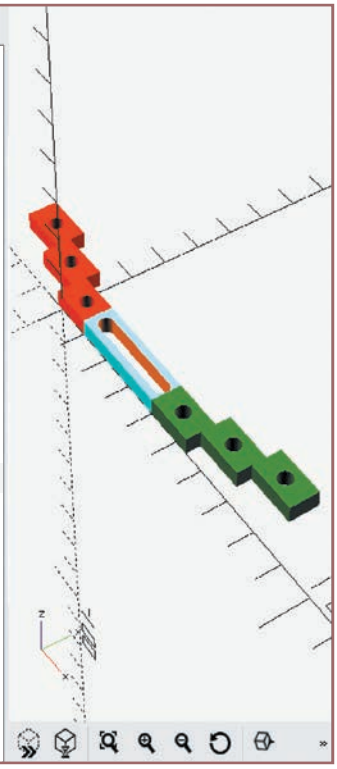


**Рис. 39.** Модель крепежа для мотора N20

```
$fn=48;
module krep ()  {
 difference ()  {
   cube ( [14, 8, 4] );
   translate ( [14/2, 8/2, –1] )
   cylinder (6, 2, 2) ;
 }
}
for ( i=[0:2] )  {
 translate ( [–14*i–2, 4*i,0] )
 color  ("red")  krep () ;
 translate ( [14*(i+2)+12,4*i, 0] )
 color ("green")  krep () ;
}
difference ()  {
 translate ( [10, 0, 0] )
 color ("cyan")  cube ( [30, 8, 4] );
 hull ()  {
   translate ( [14, 8/2, –1] )
   cylinder (6, 2, 2);
   translate ( [38, 8/2, –1] )
   cylinder (6, 2, 2);
 }
}
```



```
$fn=48 ;  h=15 ;
difference  ()  {
 union  ()  {
   color ("cyan")  cube ( [25, 10, 4], true);
   translate ( [0, 5+2, h/2–2])
   color ("pink")  cube ( [25, 4, h], true) ;
   translate ( [0, 10+4, h–4] )
   color ("red")  cube ( [25, 10, 4], true) ;
}
 color ("green")
 hull ()  {
   for ( i=[–10, 10] )
   translate ( [–1, 0, 0] )
   cylinder  (6, 2, 2, true) ;
 }
 color ("red")
 hull ()  {
   for ( i=[–10, 10] )
   translate  ( [–i,10 + 4,h–4] )
   cylinder (6, 2, 2, true) ;
 }
}
```

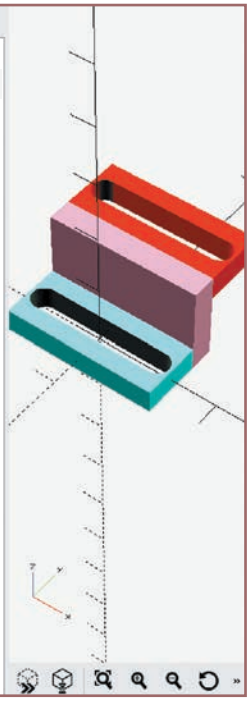


**Рис. 40.** Модель держателя для датчиков линии (две части)

## 3.4. Печать деталей

В OpenSCAD итоговое создание моделей для печати происходит только после выполнения рендеринга (соответствующая кнопка на панели инструментов) и экспорта в STL. Это формат файла, который поддерживается не только 3D-принтером, но и любым станком с программным управлением.

Некоторые детали нужно повернуть (рис. 41). Печать деталей необходимо распланировать. Например, все крепежи ориентировочно будут печататься почти 4 часа, рама для робота — около 3 часов (рис. 42) и примерно 1,5 часа — колёса. Итого запланируйте приблизитель-

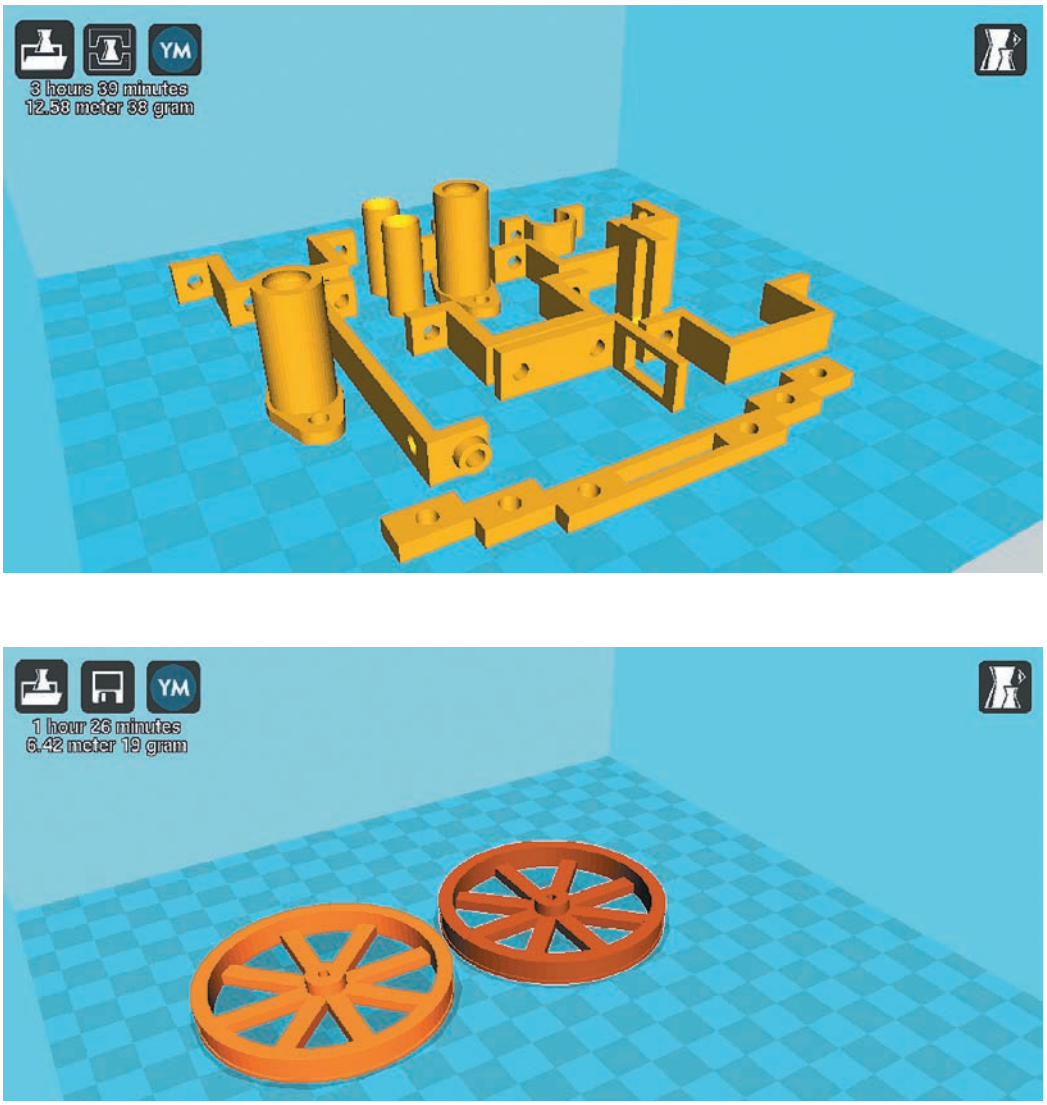



**Рис. 41.** Продолжительность печати (программа *Cura*)

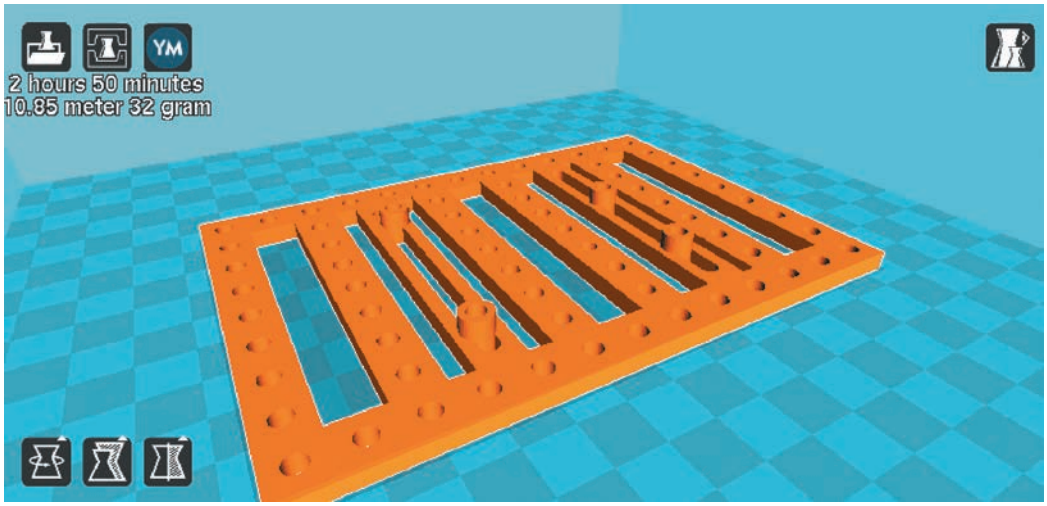



**Рис. 42.** Продолжительность печати рамы (программа *Cura*)

но 10 часов на печать. При этом принтеру, скорее всего, 10 часов практически непрерывной печати могут оказаться противопоказаны.

### Задание 14

Ещё раз проверьте все созданные модели. Проведите рендеринг каждой и экспортируйте в STL-файлы. Используя программное обеспечение, которое рекомендовано для печати на имеющемся принтере, напечатайте все необходимые детали для вашего робота.

## 3.5. Пластик PLA

Среди многообразия термопластиков отдельно стоит выделить самый популярный материал для 3D-печати — PLA-пластик. Сокращение PLA — это ***polylactic acid***, если перевести дословно — «полимолочная кислота».

Данный материал является полимером молочной кислоты, что делает PLA полностью биоразлагаемым материалом. Сырьём для его производства служат кукуруза и сахарный тростник, поэтому он экологичен и действительно абсолютно безопасен. В природных условиях срок разложения составляет от двух месяцев до двух лет, причём в итоге PLA (научно — *полилактид*) превращается в углекислый газ и воду. Нетоксичность материала позволяет осуществлять процесс печати даже в слабопроветриваемых помещениях. Не стоит печатать другими видами пластика в учебных кабинетах.

# Глава 4
# СБОРКА РОБОТА

## Задание 15

Соберите учебную модель робота, которую часто называют тележкой.

Итак, вы напечатали все смоделированные части первого робота. Сначала нужно сделать колёса. Для этого на напечатанные диски необходимо надеть три канцелярские резинки (рис. 43). Это будут своеобразные шины. По соотношению цена/качество — это самое замечательное решение. Паз для этого имеется.

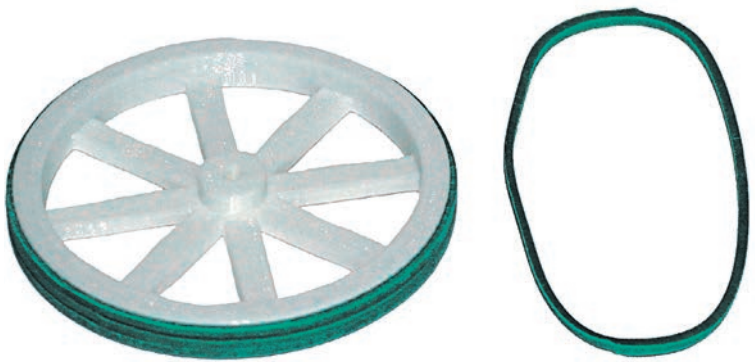


**Рис. 43.** Диск и «шина»

Далее надеваем колёса на оси моторов (рис. 44).

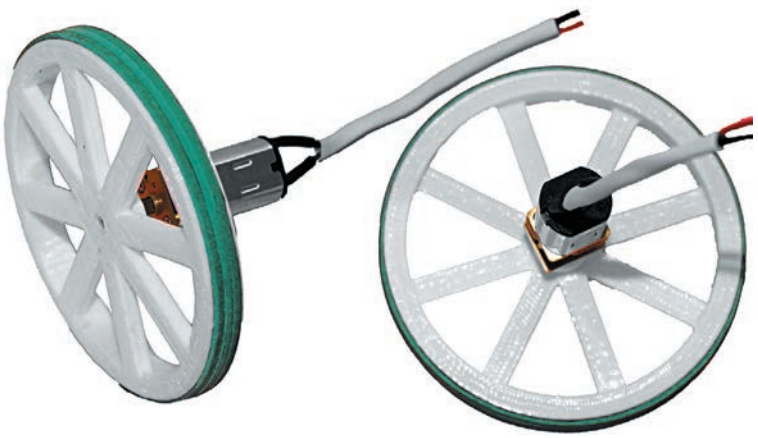


**Рис. 44.** Колёса и моторы

Затем собираем два балансира (рис. 45).

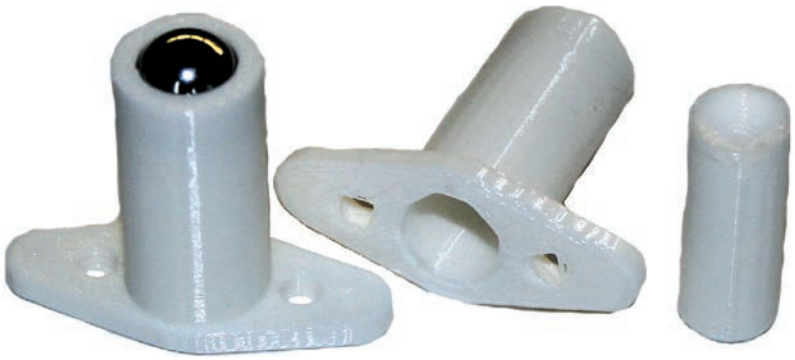


**Рис. 45.** Части балансиров

Теперь уже можно все части прикрутить к напечатанной базе. Сначала закрепим моторы с колёсами (рис. 46).

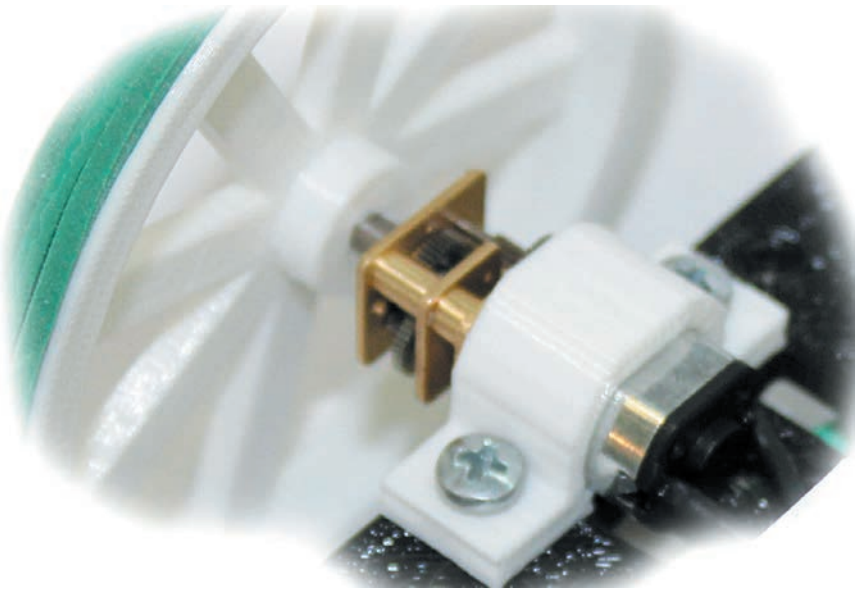


**Рис. 46.** Крепёж мотора

После этого — балансиры, держатели аккумуляторной батареи и кнопки включения (рис. 47).

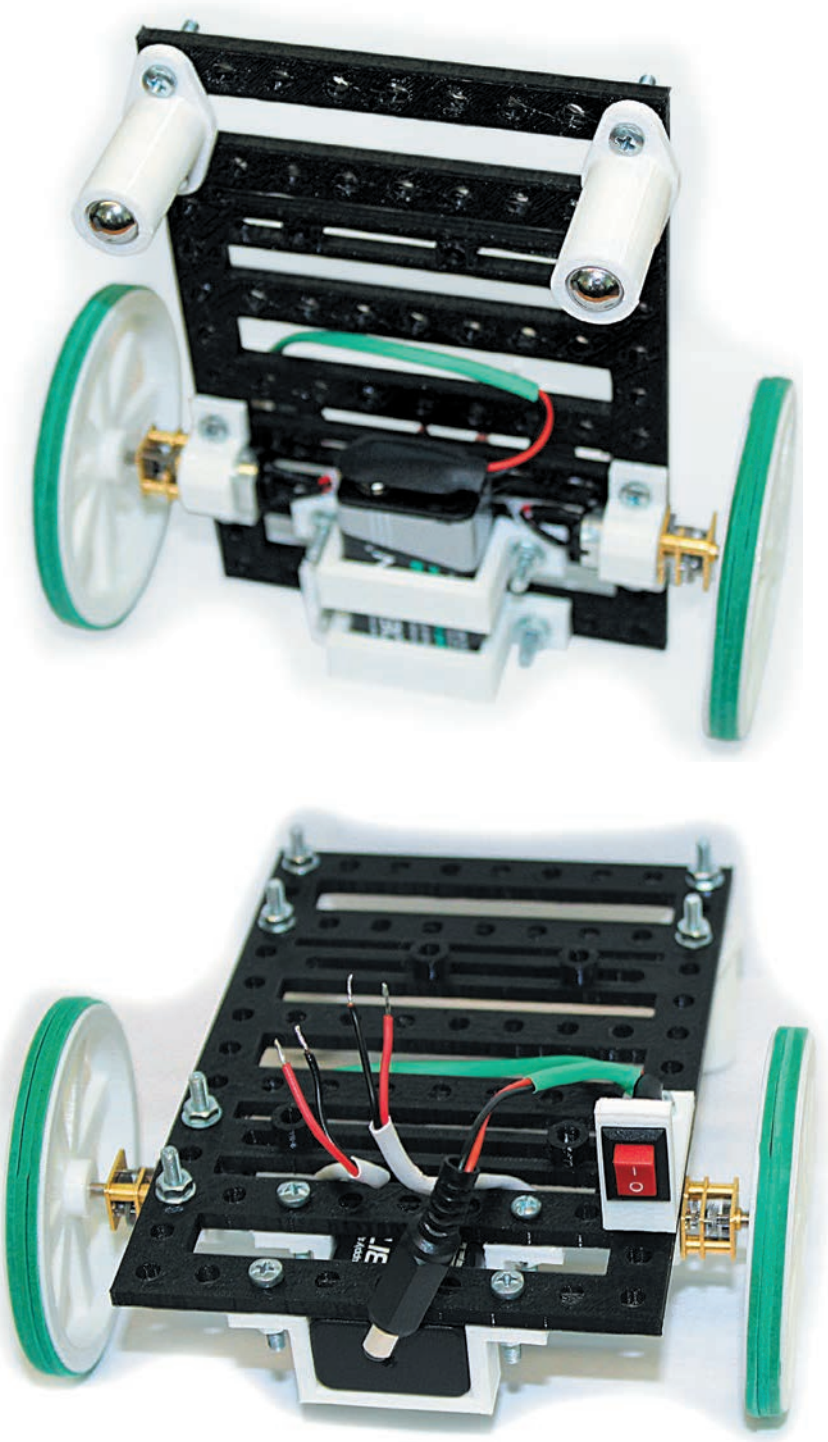


**Рис. 47.** Сборка робота

Следующий шаг — прикручивание платы Arduino. Для этого достаточно двух винтов, но можно использовать и третий (рис. 48).

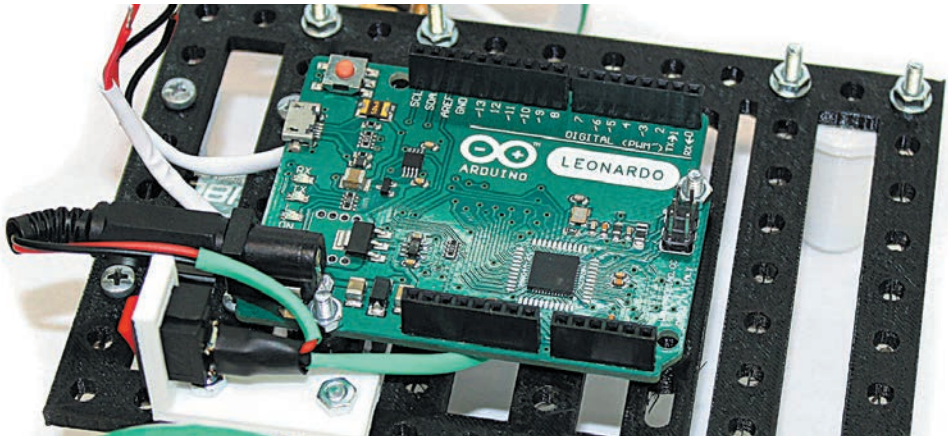



**Рис. 48.** Крепление контроллера Arduino

Установим плату управления моторами. Проверим, чтобы все её контакты были в соответствующих разъёмах на Arduino (рис. 49).

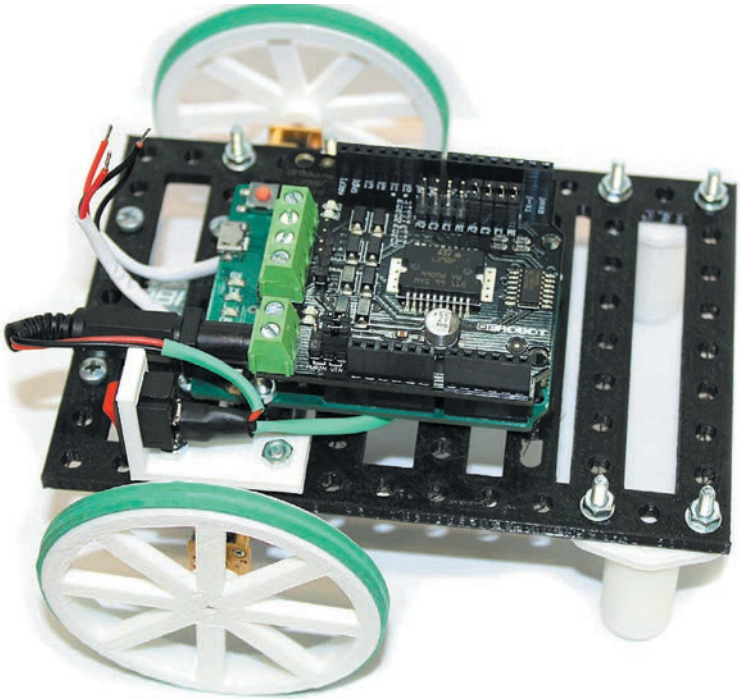


**Рис. 49.** Установка платы управления моторами

Прикрутим провода моторов к плате, как показано на рисунке 50. Левый мотор к левому разъёму, а правый — к правому. При этом правому мотору необходимо поменять полярность, так как ось должна будет вращаться в другую сторону (против часовой стрелки).

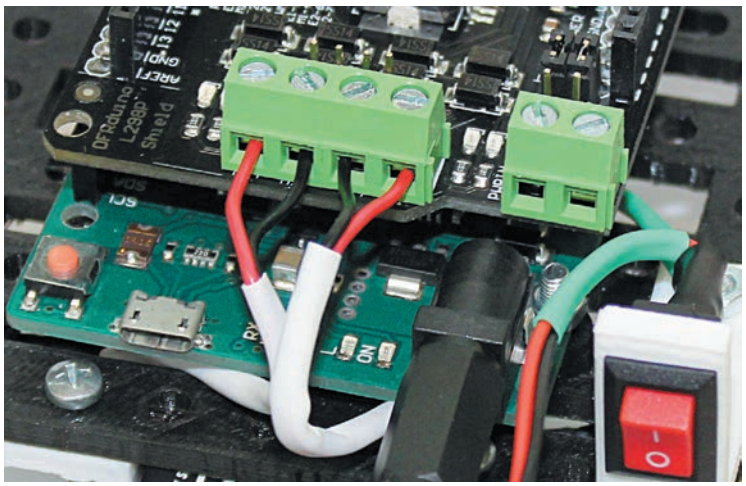



**Рис. 50.** Подключение моторов к плате

Проверим надёжность крепления, немного подёргав провода. Если какой-то провод выскочил, то нужно прикрутить более надёжно.

Последним шагом является установка платы ввода/вывода (рис. 51).

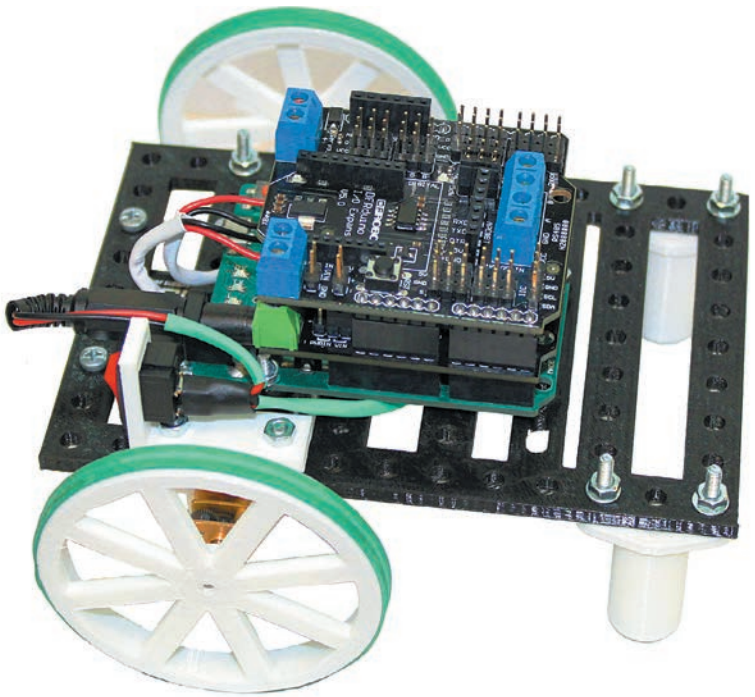


**Рис. 51.** Подключение платы ввода/вывода

Собранная модель позволяет удобно тестировать робота. Для этого поставим его вертикально балансирами вниз. А теперь пора программировать...

# Глава 5
# КРАТКОЕ ОПИСАНИЕ ЯЗЫКА ARDUINO

## 5.1. Среда Arduino IDE

Для программирования используется среда быстрой разработки программ Arduino IDE. Новейшие версии можно скачать с официального сайта www.arduino.cc.

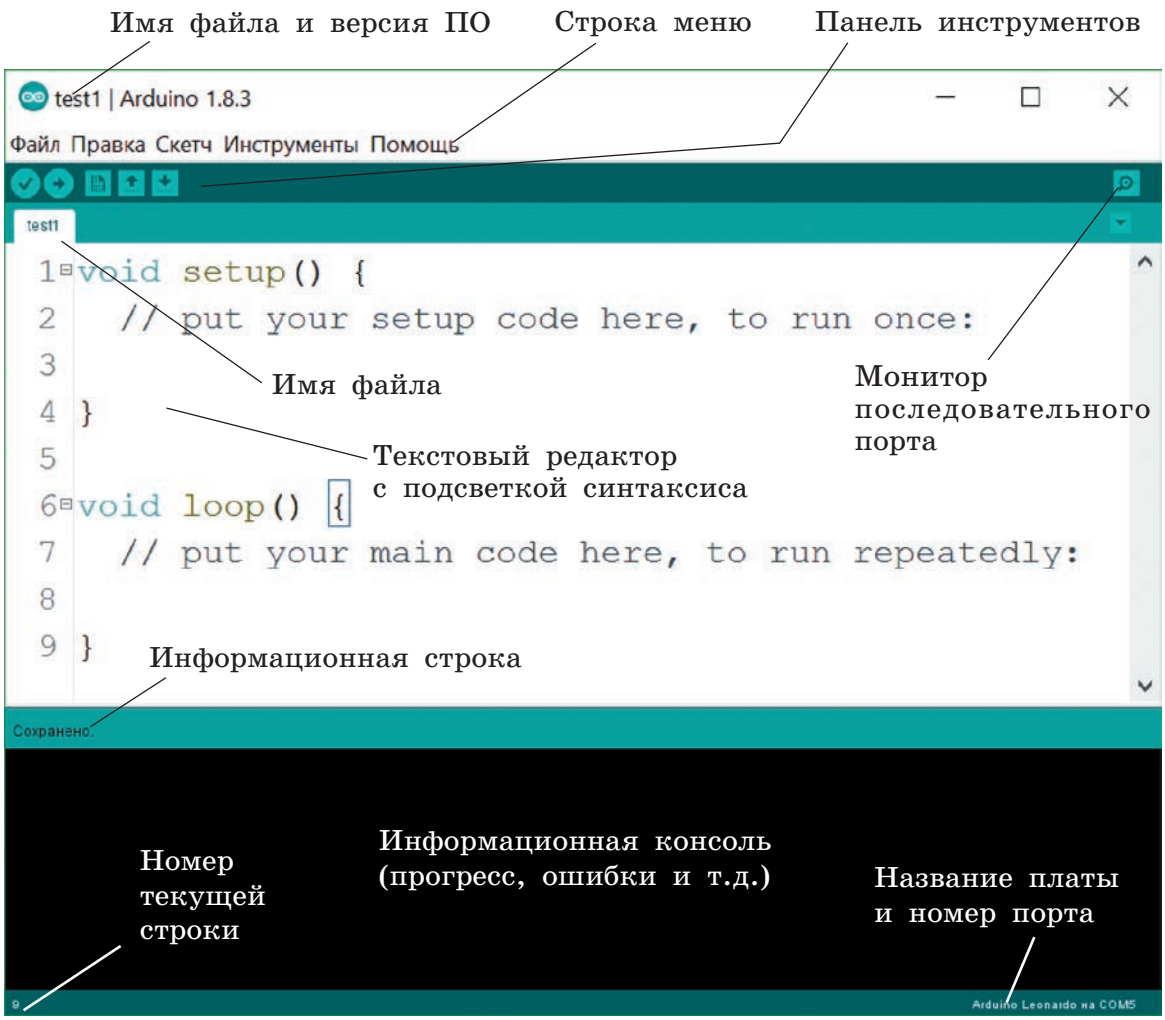



**Рис. 52.** Основные элементы интерфейса Arduino IDE

### Задание 16

Изучите основные элементы интерфейса программного обеспечения Arduino IDE (рис. 52) и основы программирования в этой среде.

## 5.2. Обязательная структура программы

Язык программирования Arduino основан на языке C++, но с небольшими адаптациями для программирования микроконтроллеров. В нём можно выделить три раздела: операторы, данные и функции.

Подробно обо всех возможностях языка вы можете посмотреть на следующих сайтах: www.arduino.cc (раздел *Reference*), www.arduino.ru (раздел *Программирование*). Мы же очень кратко рассмотрим основные операторы, структуры и синтаксис для программирования в среде Arduino IDE.

Программы называются скетчами. Каждый скетч имеет две обязательные функции (их ещё называют подпрограммами):

```
void setup() {
      Код внутри фигурных скобок
      будет выполняться один раз
}
void loop() {
      Код выполняется бесконечно,
      пока подаётся питание на плату
}
```

Каждая строка кода должна заканчиваться точкой с запятой:

```
; (точка с запятой)
```

Важным элементом языка программирования являются фигурные скобки:

```
{ } (фигурные скобки)
```

Их используют в функциях, операторах, циклах, чтобы указать, где блок кода начинается и где заканчивается.

В программах пишут комментарии — это строки, которые используются, чтобы помочь вам понять (а чаще вспомнить), как работает программа, или объяснить это другим. Они игнорируются компилятором и не занимают место в памяти микроконтроллера. Есть два способа пометить строку как комментарий:

```
//это однострочный комментарий

/*
 это многострочный комментарий
 */
```

## 5.3. Типы переменных

Основная цель любой программы состоит в обработке данных. Данные, с которыми работают машинные команды, хранятся в оперативной памяти. *Переменная* — это поименованная область памяти, имя которой можно использовать для осуществления доступа к данным. Данные, находящиеся в переменной, называются *значением* этой переменной. Переменная должна иметь: имя, значение и тип. Тип нужен, чтобы указать, сколько памяти выделить для хранения значения переменной.

Тип данных (табл. 1) однозначно определяет множество их возможных значений и допустимые действия над данными (операции и функции).

Таблица 1

**Основные типы данных в Arduino IDE**

| Тип | Описание | Пример |
|---|---|---|
| **int** | Целые числа в диапазоне от –32 768 до 32 767. Занимает 2 байта памяти | `int count = 0;` |
| **long** | Целые числа в диапазоне от –2 147 483 648 до 2 147 483 647. Занимает 4 байта (32 бита) | `long count = 0;` |
| **unsigned long** | Неотрицательные целые числа: от 0 до 4 294 967 295. Занимает 4 байта | `unsigned long z = 0;` |
| **bool** | Принимает логические значения true или false. Занимает в памяти 1 байт | `bool flag = true;` |
| **float** | Числа с плавающей запятой от $-3{,}4028235 \cdot 10^{38}$ до $3{,}4028235 \cdot 10^{38}$. Занимает 4 байта (32 бита) | `float x = 3.1415;` |
| **char** | Хранит один ASCII-символ. Занимает в памяти 1 байт. Строка — это массив символов | `char myChar = "D";`<br>`char youName[] = "Alex";` |
| **void** | Используется для указания, что функция не возвращает значения (и для определения таких функций) | |

Сведения обо всех переменных должны быть объявлены (т. е. указан тип и имя) до того, как они будут использоваться. Если кроме типа и имени указано значение, то говорят, что переменную *проинициализировали*.

Переменные, которые объявлены до блока `setup()`, называют *глобальными*, их используют в любой части программы. Также переменные можно объявлять внутри функций, циклов, такие переменные используют только внутри этих блоков программы и называют *локальными*.

## 5.4. Арифметические операции

Таблица 2

**Основные арифметические операторы**

| Оператор | Описание | Пример |
|---|---|---|
| **=** оператор присваивания | Присваивает переменной слева значение переменной или выражения, которые находятся справа | `int a = 11;`<br>`int b = 5, s = 4;`<br>`s = s + a + b;`<br>`//итог: s = 20` |
| **/** оператор деления | Деление обычное и целочисленное в зависимости от того, к какому типу переменной будет относиться | `int a = 3, b = 2;`<br>`float a1 = 3;`<br>`int c; float d, d1;`<br>`c = a/b; d = a/b;`<br>`d1=a1/b;`<br>`//c=1,d=1.0,d1=1.5` |
| ***  –  +** | Операторы умножения, вычитания и сложения работают в программе, как и в обычной математике | |
| **%** оператор остаток от деления | Остаток от деления целого числа на другое целое число. Например, x = = 7 % 5, число 5 может один раз поместиться в числе 7, поэтому остаток 2; x получит значение 2 | `int a, b, c;`<br>`a = 11;`<br>`b = 7;`<br>`c = a % b;`<br>`//в итоге c = 4` |

Также есть удобные **операции смешанного присваивания.**

`i++;` увеличивает переменную *i* на 1.
`i--;` уменьшает переменную *i* на 1.
`i+=5;` увеличивает переменную *i* на 5.
`i-=5;` уменьшает переменную *i* на 5.

## 5.5. Операторы сравнения

Операторы сравнения (табл. 3) используют для логического сравнения переменных.

Таблица 3

**Операторы сравнения**

| Обозначение | Название |
|---|---|
| == | Равно |
| != | Не равно |
| >= | Больше или равно (не меньше чем) |
| <= | Меньше или равно (не больше чем) |
| < | Меньше |
| > | Больше |

## 5.6. Логические операторы

Аргументы операции или данные, которые обрабатываются командой, в языках программирования называются *операндами*.

Таблица 4

**Логические операторы**

| Оператор | Описание |
|---|---|
| && | Логическое «И» (конъюнкция). Если оба операнда имеют значение истины (true, 1), то и результат истинный.<br>`int a = 5, b = 0, c = -5;`<br>`bool flag1, flag2, flag3;`<br>`flag1 = a>0 && c>0; //ложь`<br>`flag2 = a+c == 0 && b==0; //истина` |
| \|\| | Логическое «ИЛИ» (дизъюнкция). Если хотя бы один операнд имеет значение истины, то результат тоже истинный.<br>`int a = 10, b = 0, c = -10;`<br>`bool flag1, flag2, flag3;`<br>`flag1 = a>0 \|\| c>0; //истина`<br>`flag2 = a<0 \|\| b<0 ; //ложь` |
| ! | Логическое отрицание (инверсия). Если операнд ложь (false, 0), то результат операции — истина. И наоборот.<br>`int x = 10, y = 0, z = -10;`<br>`bool flag1, flag2, flag3;`<br>`flag1 = x>0 && z<0; //истина`<br>`flag2 = x<0 \|\| z>0 \|\| y==0; //истина`<br>`flag3 = !flag1; //ложь` |

## 5.7. Управляющие операторы

Управляющие операторы (табл. 5) предназначены для управления ходом выполнения программы. Два основных их вида — это условные операторы и циклы.

Таблица 5

**Основные управляющие операторы в Arduino IDE**

| Оператор | Описание | Пример |
|---|---|---|
| **if**<br>*если* | if (условие) {<br>//код программы, если выполняется условие<br>}<br>Код в фигурных скобках после оператора выполняется, только если условие в круглых скобках истинно | `if (value==0) {`<br>`  count++;`<br>`}` |
| **if ... else**<br>*если ... иначе* | if (условие) {<br>//код программы, если выполняется условие<br>}<br>else {<br>//код программы, если не выполняется условие<br>} | `if (value>0) {`<br>`  count1++;`<br>`}`<br>`else {`<br>`  count2++;`<br>`}` |
| **Цикл for**<br>*для* | for (начальное значение; условие выхода; шаг) {<br>//код программы }<br>Используется для того, чтобы повторять действие в скобках определённое количество раз | `int m=0;`<br>`for (int i=0;i<5;i++)`<br>`{`<br>` m=m+i;`<br>`}`<br>`//m = 0+1+2+3+4 = 10` |
| **Цикл while**<br>пока | while (условие) {<br>// код программы<br>}<br>Код в цикле будет выполняться непрерывно до тех пор, пока условие в круглых скобках не станет ложным | `int m=0, i=0;`<br>`while (m<=10) {`<br>` i=i+1;`<br>` m=m+i;`<br>`}`<br>`// i: 0 → 1 → 2 → 3 → 4`<br>`// m: 0 → 1 → 3 → 6 → 10` |
| **do while** | do {<br>// код программы<br>} while (условие)<br>Сначала выполняется код в цикле, а потом проверяется условие его завершения | `int m=0, i=0;`<br>`do {`<br>` i=i+1;`<br>` m=m+i;`<br>`} while (m<10)`<br>`// i: 0 → 1 → 2 → 3 → 4`<br>`// m: 0 → 1 → 3 → 6 → 10` |

## 5.8. Массивы

*Массив* — это область памяти, где последовательно хранятся несколько переменных. Объявляется массив так:

```
int distance [10];   //массив из 10 переменных типа int
float velocity[100]; //массив из 100 переменных типа float
```

При объявлении массивы можно инициализировать:

```
int distance [10] = {20,30,34,24,45,23,27,30,50,40};
```

Каждый элемент массива имеет свой номер, который называется *индексом*. Нумерация элементов начинается с нуля. В программе обращение к элементу массива происходит по его индексу:

```
x = distance [5];// x = 23;
```

Это означает, что переменной *x* присваивается значение из пятого элемента массива.

Или, например, девятому элементу массива задаётся значение 32:

```
distance [9] = 32;
```

Пример программы, находящей сумму элементов массива:

```
int distance[10] = {20,30,34,24,45,23,27,30,50,40};
int sum = 0;
for(int i=0; i<=10; i++) {
   sum = sum + distance[i];
}
```

## 5.9. Директива #define

Есть команда, указывающая компилятору, что во всём тексте программы нужно указанное имя заменить указанным значением:

```
#define имя значение
#define BUTTON_PIN    8
```

Директива #define используется для именования значений, которые не будут изменяться. Обратите внимание, что #define не завершается точкой с запятой, а *определённые этой директивой константы не занимают программной памяти*.

## 5.10. Функции

Функции позволяют выполнять одни и те же действия с разными данными. У функции есть:

- *имя*, по которому её можно вызвать;
- *аргументы*, т. е. данные, необходимые для вычисления;
- *тип* данных, которые возвращает функция.

Функции бывают встроенные и пользовательские. Пользовательские описываются перед setup () и loop ().

```
int hypotenuse (int  leg1, int leg2)
{
  return( sqrt (leg1*leg1 + leg2*leg2) );
}
```

Вызов функции происходит так:

```
c = hypotenuse (3,4);
```

Обратите внимание — то, что именно вычисляет эта функция, понятно сразу.

## 5.11. Несколько правил

- Переменные должны быть записаны именами в смешанном регистре, первая буква строчная (нижний регистр).

  ```
  int encoderCount; bool buttonState;
  ```

- Все константы должны иметь имена, написанные заглавными буквами с символом нижнего прочерка _ на месте пробелов.

  ```
  #define BUTTON_PIN    8
  ```

- Подпрограммы и функции должны быть названы глаголами, записанными в смешанном регистре, первая буква — в нижнем регистре.

  ```
  void turnRight (int velocity, int duration){…}
  ```

## 5.12. ЦАП, АЦП, ШИМ

Двоичное число в технике представлено в виде последовательности прямоугольных импульсов (рис. 53) благодаря простоте кодирования:

- 1, HIGH — контакт замкнут, есть импульс, ток течёт;
- 0, LOW — контакт разомкнут, импульса нет, тока нет.

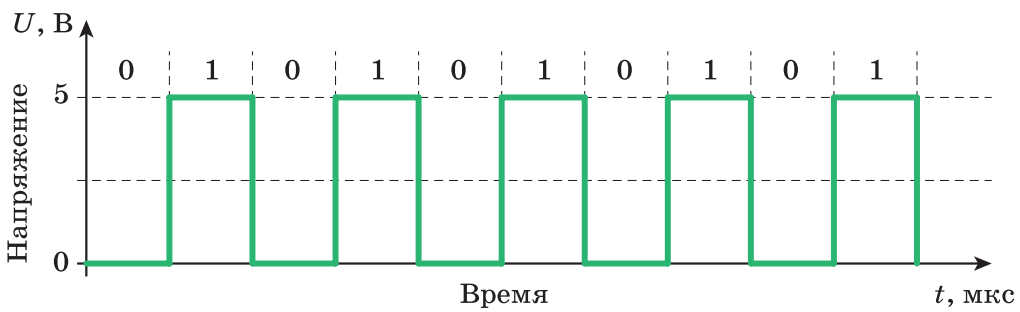



**Рис. 53.** Прямоугольные импульсы

Однако уровни логического нуля и единицы в реальном оборудовании — это не 0 и 5 В. Микросхемы бывают с разным напряжением питания, и уровни логического нуля и единицы могут принимать разные значения. Уровень логической единицы на плате Arduino находится в интервале от 3 до 5 В, а логический нуль — от 0 до 2 В (рис. 54).

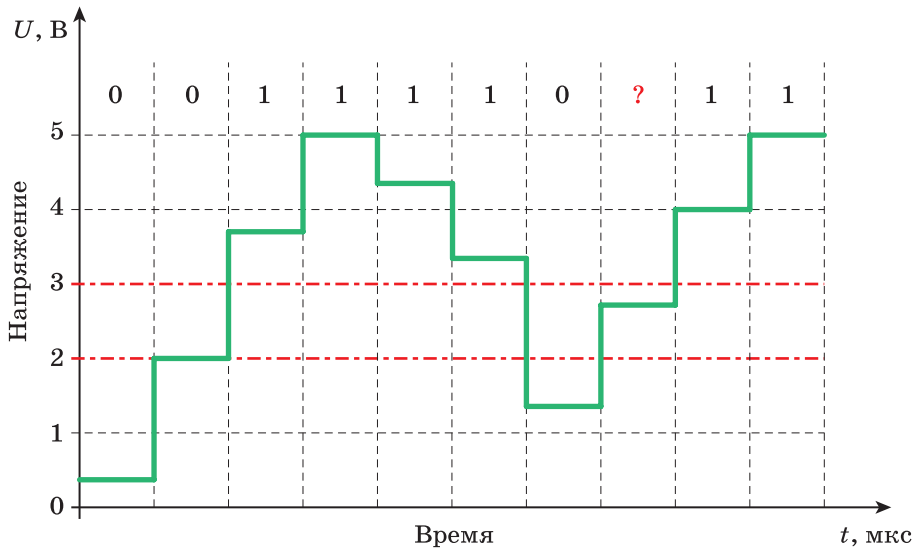



**Рис. 54.** Пример работы функции чтения цифровых данных

На рисунке 54 показана выпадающая точка. Это воздействия на датчик помех, внешних факторов. Такое часто встречается. Точка эта будет проигнорирована.

Из курса информатики вам уже знаком основной принцип хранения информации в памяти компьютера — *принцип дискретности*: любые данные в памяти компьютера хранятся в виде цепочек битов, т. е. последовательностей нулей и единиц (дискретная форма). Таким образом, любая программа в компьютере имеет форму двоичного кода. Чтобы она превратилась в физическое воздействие на управляемый объект, необходимо преобразовать код в электрический сигнал. Такое преобразование называют цифроаналоговым преобразованием, а устройство, его выполняющее, называется *цифроаналоговым преобразователем* (ЦАП).

Микроконтроллер на Arduino выдаёт не произвольное напряжение, а только напряжение питания (5 В) либо земли (0 В). Однако уровнем напряжения управляется скорость вращения мотора нашего робота.

Для симуляции разных уровней напряжения используется *широтно-импульсная модуляция* (ШИМ, англ. Pulse Width Modulation, PWM). Выход микроконтроллера переключается между 0 и 5 В тысячи раз в секунду. Частота неизменна, но может быть разной для разных контактов (например, на Leonardo на контактах 3 и 11 частота 980 Гц,

а на остальных контактах с ШИМ — 490 Гц), а меняется длительность времени включения 5 В относительно включения 0 В (рис. 55).

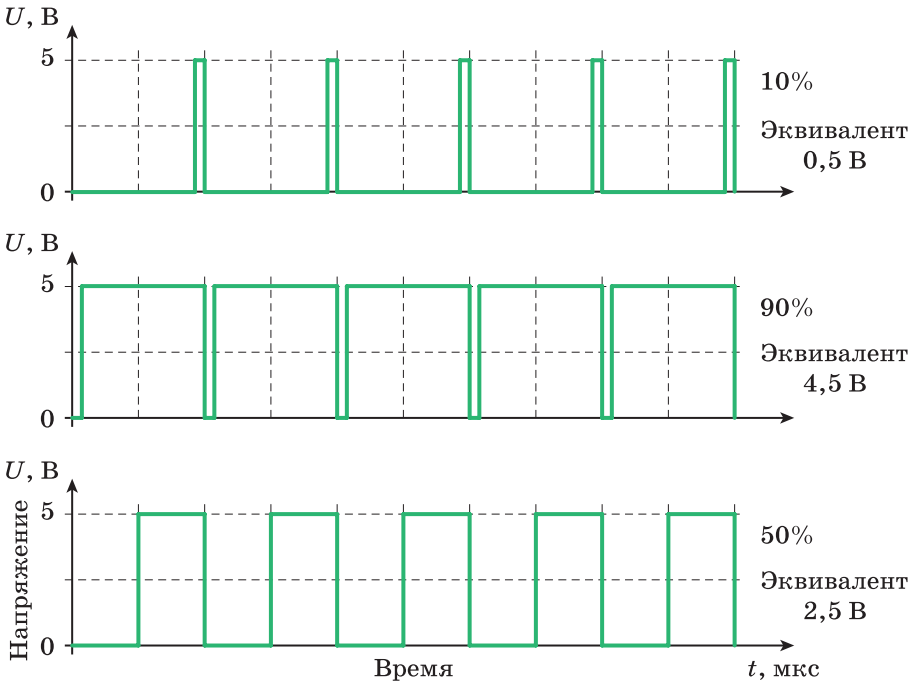



**Рис. 55.** Принцип работы ШИМ

Длительность управляющих импульсов выражена в процентах. Это коэффициент заполнения (англ. *Duty cycle*).

В программе эта длительность задаётся не в процентах, а числом от 0 (это 0%) до 255 (это 100%). Промежуточные значения пропорциональны. Получается, что можно сымитировать аналоговый сигнал на цифровом выходе. Благодаря этому мы можем управлять скоростью вращения двигателя, яркостью светодиодов и т. д.

Приборы, которые дают информацию о состоянии объекта управления, о внешней среде, называются датчиками. Они представляют информацию, например, о температуре, влажности, расстоянии, давлении, деформации, наличии магнитного поля и т. д. в виде электрических сигналов. Если такие данные датчики передают в аналоговой форме (электрический ток или напряжение), то они должны быть преобразованы в двоичную цифровую форму. Такое преобразование называется аналого-цифровым, а прибор, его выполняющий, — *аналого-цифровым преобразователем* (АЦП).

При преобразовании аналогового сигнала в цифровом устройстве АЦП много раз в секунду производит измерение амплитуды аналогового сигнала и выдаёт результаты этих измерений в виде чисел. Представление сигнала рядом своих отсчётов (англ. *samples*), взятых через равные промежутки времени, называется *дискретизацией*.

Частота, с которой производятся цифровые значения, получила название *частота дискретизации* АЦП.

На плате Arduino Leonardo максимальное значение этой частоты равно 10 кГц, т. е. *10 000 значений за 1 секунду*. Это максимальная частота, с которой будет происходить опрос датчиков.

Разрядность АЦП характеризует количество дискретных значений, которые преобразователь может выдать на выходе, и измеряется в битах.

Микроконтроллеры Atmega, используемые в Arduino Leonardo, содержат двенадцатиканальный аналого-цифровой преобразователь разрядностью 10 бит. $2^{10} = 1024$, следовательно, всего 1024 различных значения, их называют *уровнями квантования*.

*Поэтому при подключении к аналоговым входам A0–A11 плата Arduino Leonardo может выдавать значения в диапазоне от 0 до 1023.*

## 5.13. Функции для работы с цифровыми сигналами

Таблица 6

**Функции для работы с цифровыми сигналами**

| Функция | Описание | Пример |
|---|---|---|
| **pinMode** | pinMode (№ контакта, режим работы);<br>Задаёт режим работы контакта: вход (INPUT), выход (OUTPUT) | `pinMode(3,OUTPUT);`<br>`pinMode(2,INPUT);` |
| **digitalWrite** | digitalWrite (№ контакта, значение);<br>Когда вывод настроен на выход (OUTPUT), он может выдавать значение HIGH (подать 5 В) либо LOW (0 В) | `digitalWrite(13,HIGH);`<br>`digitalWrite(12,LOW);` |
| **digitalRead** | digitalRead (№ контакта);<br>В режиме входа (INPUT) эта функция возвращает значение HIGH (если напряжение — уровня логической единицы) или LOW (при напряжении уровня логического нуля) | `bool encoderState =`<br>`digitalRead(2);` |

## 5.14. Функции для работы с аналоговыми сигналами

Таблица 7

**Функции для работы с аналоговыми сигналами**

| Функция | Описание | Пример |
|---|---|---|
| **analogWrite** | Это как раз ШИМ на указанном контакте выхода (пины с ШИМ: 3, 5, 6, 9, 10, 11, 13) | `analogWrite(5, 255);`<br>`analogWrite(6, 0);` |
| **analogRead** | Читаем, что показывает АЦП на указанном входе | `int lightValueLeft = analogRead(A0);`<br>`int lightValueRight = analogRead(A1);` |

## 5.15. Функции для работы со временем

Таблица 8

**Функции для работы со временем**

| Функция | Описание | Пример |
|---|---|---|
| **delay** | Функция останавливает выполнение программы на заданное количество миллисекунд | `delay (3000);`<br>`delay (1500);`<br>`delay (10);` |
| **millis()** | Эта функция возвращает количество миллисекунд с момента начала выполнения текущей программы на плате | `//узнать количество секунд`<br>`int t = millis()/1000;`<br><br>`//секунды от 0 до 59`<br>`int clockSeconds = millis()/1000%60;` |
| **delayMicroseconds** | Останавливает выполнение программы на заданное в параметре количество микросекунд (в 1 секунде 1 000 000 микросекунд) | `delayMicroseconds (10);` |

## 5.16. Монитор последовательного порта

При написании программы, чтобы понять, какое действие она выполняет в данный момент, какие значения у переменных и что «показывают» датчики, используется монитор последовательного порта (табл. 9).

Таблица 9

**Основные команды при работе с Serial-monitor**

| Функция | Описание |
|---|---|
| **Serial.begin** | `Serial.begin(9600);`<br>Задаём скорость обмена данными с портом 9600 бит/с.<br>Пишется команда обязательно в блоке setup() |
| **Serial.print** | `Serial.print("A0 = ");`<br>`Serial.print(a);`<br>Выводит на экран строку (текст, значение переменной) |
| **Serial.println** | `Serial.println(a);`<br>Выводит на экран значение и переходит на новую строку |

## 5.17. Некоторые математические функции

```
map(value, fromLow, fromHigh, toLow, toHigh)
```

Функция пропорционального управления возвращает целочисленное значение из интервала [toLow, toHigh]. Значение является пропорциональным отображением содержимого в переменной *value* из интервала [fromLow, fromHigh].

Например, мы получаем с аналогового порта значения от 0 до 1023, а хотим через ШИМ управлять скоростью мотора. При этом задавать можем только числа от 0 до 255. Для этого используем *map*:

```
int lightSensorValue = analogRead(A0);
int motorSpeedLeft =
                map(lightSensorValue,0,1023,0,255);
```

Возможно использование обратнопропорционального преобразования:

```
int lightSensorValue = analogRead(A0);
int motorSpeedLeft =
                map(lightSensorValue,0,1023,255,0);
```

Функция *map* не отбрасывает значения за пределами указанных диапазонов и масштабирует их по заданному правилу.

Если есть необходимость ограничить множество допустимых значений, то используется функция

```
constrain(переменная, min_значение, max_значение)
```

Всё, что превышает максимальное значение, будет заменено максимальным значением. И аналогично для минимального.

### Задание 17

Ознакомьтесь со встроенными математическими функциями (www.arduino.cc или www.arduino.ru).

## 5.18. Тернарный оператор

*Тернарная условная операция* ( ? : ) — операция, возвращающая свой второй или третий операнд в зависимости от значения логического выражения, заданного первым операндом.

В Интернете есть очень много примеров программ, обеспечивающих мигание светодиодом, большинство из них используют функцию *delay*. Для робота это катастрофа. Он же не может «всё бросить» и ждать. Как сделать, чтобы, например, светодиод включался на 2 секунды, а выключался на одну?

```
#define LED_PIN 13
int seconds = 0;
bool ledStatus = false;
void setup() {
  pinMode(LED_PIN, OUTPUT);
}
void loop() {
  seconds = millis ()/1000;
  ledStatus = seconds%3!=0 ? HIGH : LOW;
  digitalWrite (LED_PIN, ledStatus);
}
```

Будем считать количество секунд и записывать в переменную. Так как через 3 секунды всё повторяется, то используем деление с остатком на три. Остатки будут: 0, 1, 2, 0, 1, 2 и т. д. Если остаток не равен 0, то в переменную ledStatus записываем HIGH, в противном случае — LOW. Эта переменная и будет задавать режим работы светодиода. Переменные в примере были созданы для более лёгкого понимания. Можно, конечно, записать и без них.

```
#define LED_PIN 13
void setup() {
  pinMode(LED_PIN, OUTPUT);
}
void loop() {
 digitalWrite (LED_PIN, millis()/1000%3!=0 ? HIGH : LOW);
}
```

### Задание 18

Найдите в Интернете примеры мигания светодиодом без *delay*.

# Глава 6
# ПРОГРАММИРУЕМ РОБОТА

## 6.1. Подключение платы

Используя кабель micro-USB, подключите плату Arduino Leonardo к вашему компьютеру. Если плата подключается в первый раз, то в операционной системе будут установлены драйверы, и через несколько минут (максимум) плата появится в диспетчере устройств. Обязательно проверьте это и запомните номер последовательного порта, к которому плата будет подключена (рис. 56).

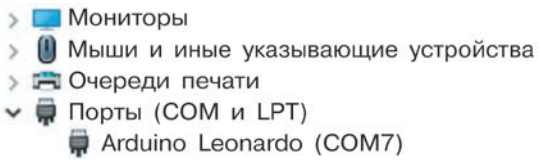



**Рис. 56.** Плата Arduino в диспетчере устройств (порт № 7)

Запустите среду быстрой разработки Arduino IDE через соответствующий ярлык на рабочем столе. В меню *Инструменты* обязательно выберите плату Leonardo и порт, к которому она подключена (рис. 57). **Обязательно проверяйте имя и порт платы перед загрузкой программы!**

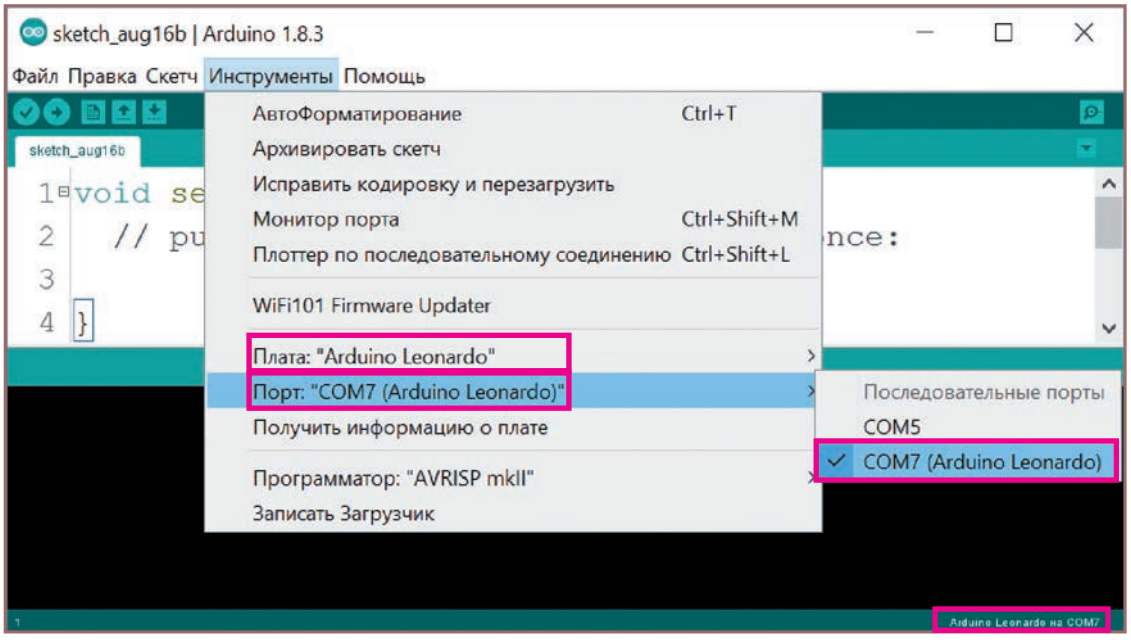



**Рис. 57.** Выбор платы и порта в Arduino IDE

Для быстрой проверки того, что плата работает, обычно загружают тестовый пример. Найдите в примерах программу *Blink* (в меню), откройте её и загрузите в микроконтроллер (рис. 58).

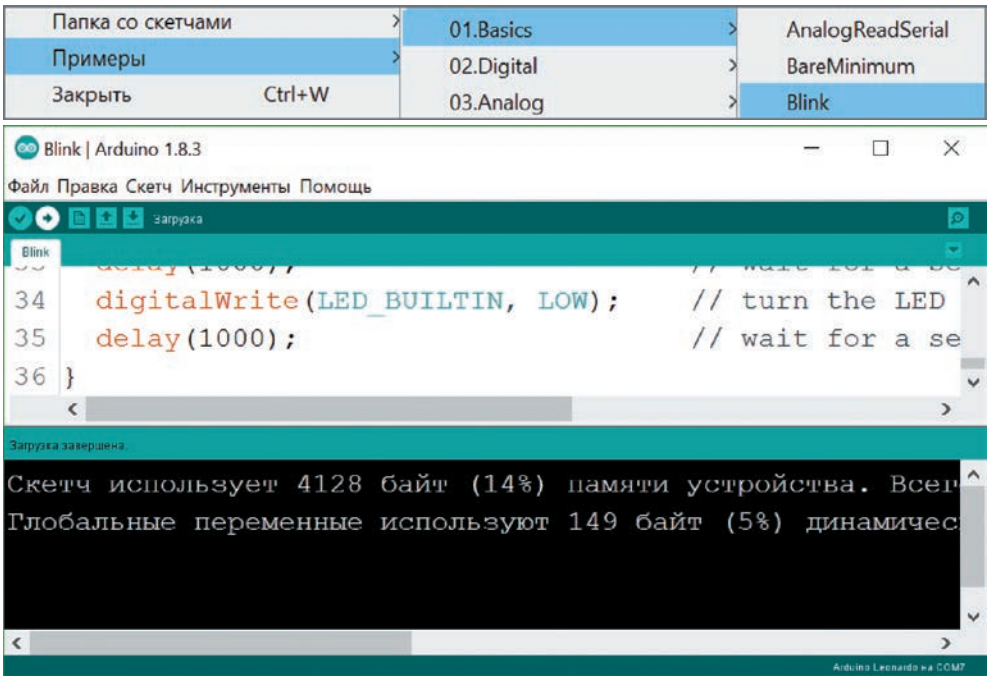



**Рис. 58.** Пример успешной загрузки программы

При программировании в коде возможны ошибки, в этом случае при попытке загрузки будет выведено предупреждение об их наличии и указан их характер (рис. 59). Тогда необходимо их найти и исправить.

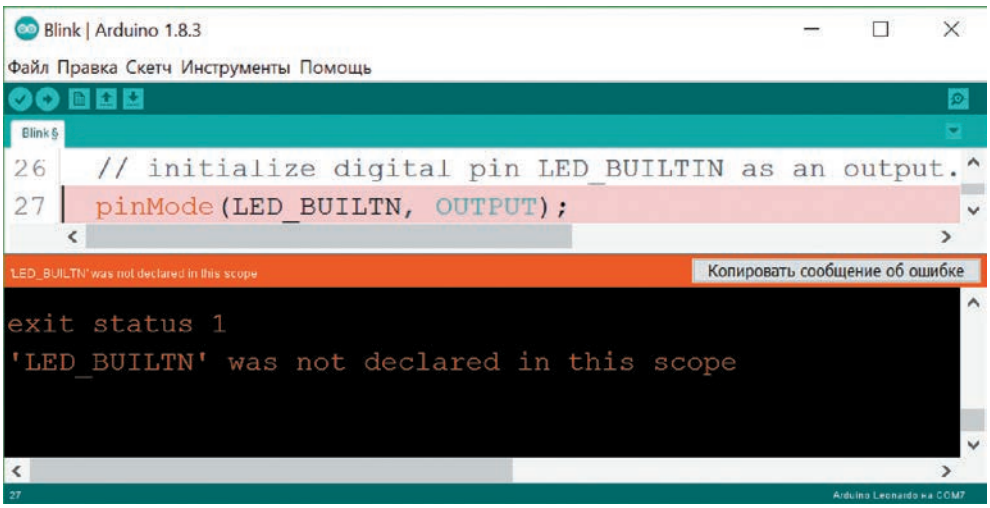



**Рис. 59.** Ошибка в программе

## 6.2. Поехали!

Настала пора включить робота и начать его программировать.

### Задание 19

Необходимо, чтобы робот вращал одно колесо в течение 2 секунд, а потом его остановил. Напишите программу.

```
#define SPEED_M1       5
#define DIR_M1         4
void setup() {
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
}
void loop() {
  digitalWrite(DIR_M1, HIGH);
  analogWrite (SPEED_M1,100);
  delay(2000);
  analogWrite (SPEED_M1,0);
  while(true) {
     delay(1000);
  }
}
```

Давайте немного разберём код. Первые две строчки аналогичны функции *Найти и заменить* в текстовом редакторе. То есть слова SPEED_M1 и DIR_M1 будут заменены на «5» и «4» соответственно, что очень удобно, так как это не переменные и память микроконтроллера не занимают. Используется команда `#define`.

Поскольку мы хотим управлять мотором M1, нам необходимо указать, что контакты (пины) 4 и 5 будут работать в режиме выхода, т. е. OUTPUT. Это следует сделать всего один раз, поэтому пишем эти команды в `setup()`. Блок `loop()` будет выполняться бесконечно, как бесконечный цикл. Значит, это нужно предотвратить — пишем `while(true) {delay(1000);}`. Практически перед нами «заглушка», бесконечный цикл, из которого нет выхода. Пока этого достаточно. Такой способ удобно использовать при поиске ошибок в программе.

Также обратите внимание, что направление вращения задаётся командой `digitalWrite(DIR_M1, HIGH)`, при этом возможны всего два случая: по часовой и против часовой стрелки. А скоростью можно управлять командой

```
analogWrite (SPEED_M1,100)
```

Если просто описать, что делает указанная программа, то получится: программа включает мотор M1 на 2 секунды, а потом его выключает.

## Задание 20

Напишите программу, чтобы робот ехал вперёд в течение 2 секунд, а потом останавливался.

```
#define DIR_M1         4
#define SPEED_M1       5
#define SPEED_M2       6
#define DIR_M2         7
void setup() {
  pinMode(4,OUTPUT); pinMode(5,OUTPUT);
  pinMode(6,OUTPUT); pinMode(7,OUTPUT);
  digitalWrite(DIR_M1, HIGH);
  digitalWrite(DIR_M2, HIGH);
  analogWrite (SPEED_M1,100);
  analogWrite (SPEED_M2,100);
  delay(2000);
  analogWrite (SPEED_M1,0);
  analogWrite (SPEED_M2,0);
}
void loop() {
}
```

В этом случае мы не стали «изобретать велосипед» и написали все команды управления движением робота в `setup()`, так как выполняться они должны всего один раз, а не постоянно.

Стоит заметить, что задание режима работы для четырёх контактов можно реализовать с помощью цикла. В следующем задании так и сделаем.

## Задание 21

Напишите программу, чтобы робот ехал вперёд в течение 2 секунд, остановился на 2 секунды и вернулся обратно.

```
#define DIR_M1         4
#define SPEED_M1       5
#define SPEED_M2       6
#define DIR_M2         7
void setup() {
```

```
  for (int i=4;i<=7;i=i+1) pinMode(i,OUTPUT);

  digitalWrite(DIR_M1, HIGH);
  digitalWrite(DIR_M2, HIGH);
  analogWrite (SPEED_M1,100);
  analogWrite (SPEED_M2,100);
  delay(2000);
  analogWrite (SPEED_M1,0);
  analogWrite (SPEED_M2,0);
  delay(2000);
  digitalWrite(DIR_M1, LOW);
  digitalWrite(DIR_M2, LOW);
  analogWrite (SPEED_M1,100);
  analogWrite (SPEED_M2,100);
  delay(2000);
  analogWrite (SPEED_M1,0);
  analogWrite (SPEED_M2,0);
}
void loop() {
}
```

Так как в теле цикла всего одна команда, фигурные скобки можно не писать.

**Задание 22**

Проведите испытания вашего робота. Обратите особое внимание на его «поведение», как он перемещается: прямо ли, вернулся ли точно в исходную точку, а если всё повторить, то результат будет таким же или чуть отличаться и т. д.

## 6.3. Управляем касанием

Не очень радует, что робот едет сразу. Было бы лучше, чтобы он начинал движение после какой-нибудь нашей команды. Реализовать это можно с помощью *датчика касания*. Это цифровой датчик, который передаёт роботу два значения: есть касание и нет касания. Поэтому его часто называют сенсорной кнопкой.

## Задание 23

Прикрепите модуль с датчиком касания, как показано на рисунке 60, и подключите к разъёму 8 на плате расширения. Красный провод обеспечивает питание (VCC), чёрный провод соединяет контакты GND (земля), а зелёный — сигнальный провод (на датчике пин SIG или D).

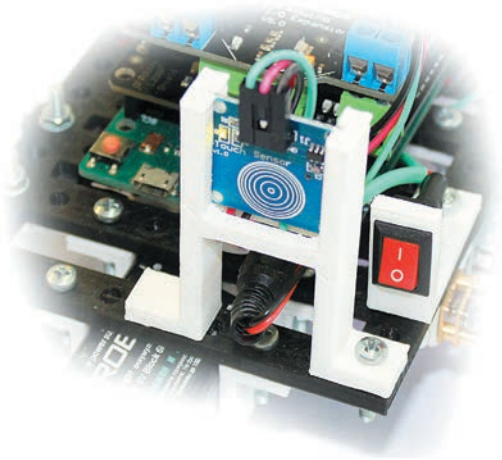


**Рис. 60.** Крепление модуля с датчиком касания

## Задание 24

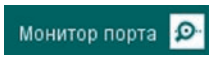
Напишите программу, которая выводит на монитор последовательного порта информацию о текущем состоянии датчика касания (сенсорной кнопки).

```
#define BUTTON_PIN    8
bool buttonState;
void setup() {
  Serial.begin(9600);
  pinMode (BUTTON_PIN,INPUT);
}
void loop() {
  buttonState = digitalRead(BUTTON_PIN);
  Serial.print ("Есть ли касание?");
  Serial.println (buttonState);
  delay(100);
}
```

Из первой строки видно, что сенсор будет подключён к контакту 8. Введём логическую переменную, указав соответствующий тип (bool),

которая будет хранить значение текущего состояния сенсора, т. е. или «ложь» (0, false), или «истина» (1, true).

Далее включим обмен данными через виртуальный последовательный порт на скорости 9600 бит/с, зададим режим работы контакта 8, и в бесконечном цикле (loop) будем читать данные с датчика касания, выводя данные на монитор последовательного порта.




После успешной загрузки программы в Arduino необходимо открыть монитор последовательного порта (Serial Monitor), нажав иконку в верхнем правом углу окна.

Небольшое замечание: у сенсоров касания может быть своя специфика. Посмотрите на значения, которые отображаются на мониторе. Некоторые модели таких сенсоров могут быть инверсными. У них всё наоборот: когда нет касания — 0, а когда есть — 1. В принципе нам ведь важно только знать, какое значение будет при касании, а 0 или 1 — не принципиально.

Через монитор последовательного порта (рис. 61) можно выводить значения переменных, показания датчиков и т. п. Это мощнейший инструмент при тестировании ваших программ.

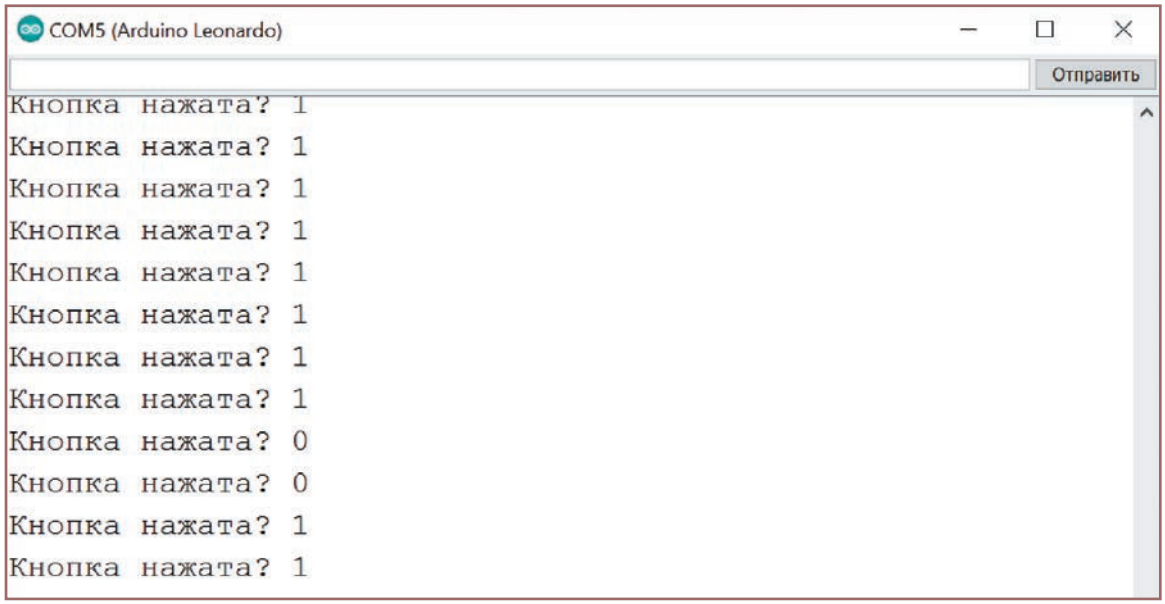



**Рис. 61.** Монитор последовательного порта

## Задание 25

По нажатию на датчик касания робот должен проехать вперёд, затем развернуться и вернуться в исходную точку.

```
#define DIR_M1         4
#define SPEED_M1       5
#define SPEED_M2       6
#define DIR_M2         7
#define BUTTON_PIN     8
void setup() {
  for (int i=4;i<=7;i=i+1) pinMode(i,OUTPUT);
  pinMode (BUTTON_PIN,INPUT);
}
void loop() {
  //ждём нажатия кнопки
  while(digitalRead(BUTTON_PIN)==false);
  //вперёд
  digitalWrite(DIR_M1, HIGH);
  digitalWrite(DIR_M2, HIGH);
  analogWrite (SPEED_M1,100);
  analogWrite (SPEED_M2,100);
  delay(2000);
  //остановка
  analogWrite (SPEED_M1,0);
  analogWrite (SPEED_M2,0);
  delay(2000);
  //разворот
  digitalWrite(DIR_M1, LOW);
  digitalWrite(DIR_M2, HIGH);
  analogWrite (SPEED_M1,100);
  analogWrite (SPEED_M2,100);
  delay(650);
  //остановка
  analogWrite (SPEED_M1,0);
  analogWrite (SPEED_M2,0);
  delay(1000);
  //вперёд
  digitalWrite(DIR_M1, HIGH);
  digitalWrite(DIR_M2, HIGH);
  analogWrite (SPEED_M1,100);
  analogWrite (SPEED_M2,100);
  delay(1000);
  //остановка
  analogWrite (SPEED_M1,0);
  analogWrite (SPEED_M2,0);
}
```

Необходимо подобрать задержку и скорость мотора, чтобы робот повернул примерно на 180°.

Мы видим, что программа очень разрослась, и её нужно скомпоновать более разумно. Для этого, например, можно добавить функции, которые будут отвечать за разные движения робота.

## Задание 26

По нажатию на датчик касания робот должен проехать вперёд, затем развернуться и проследовать в исходную точку. Используйте процедуры.

```
#define DIR_M1         4
#define SPEED_M1       5
#define SPEED_M2       6
#define DIR_M2         7
#define BUTTON_PIN     8
void forward (int velocity, int duration) {
  digitalWrite(DIR_M1, HIGH);
  digitalWrite(DIR_M2, HIGH);
  analogWrite (SPEED_M1,velocity);
  analogWrite (SPEED_M2,velocity);
  delay(duration);
  analogWrite (SPEED_M1,0);
  analogWrite (SPEED_M2,0);
}
void turnRight (int velocity, int duration){
  digitalWrite(DIR_M1, LOW);
  digitalWrite(DIR_M2, HIGH);
  analogWrite (SPEED_M1,velocity);
  analogWrite (SPEED_M2,velocity);
  delay(duration);
  analogWrite (SPEED_M1,0);
  analogWrite (SPEED_M2,0);
}
void setup() {
  for (int i=4;i<=7;i=i+1) pinMode(i,OUTPUT);
  pinMode (BUTTON_PIN,INPUT);
}
```

```
void loop() {
  //ждём нажатия кнопки
  while(digitalRead(BUTTON_PIN)==false);
  forward(100,2000);
  delay (1000);
  turnRight (100,650);
  delay (1000);
  forward(100,2000);
}
```

Ключевое слово `void` используется при объявлении функций, которые не возвращают никакого значения при их вызове. Такие функции ещё называют *процедурами*. Как можно заметить, в таком виде значительно удобнее понимать код программы.

## 6.4. Включаем/выключаем светодиод

Самый первый проект на Arduino — это написание программы для мигания (включения/выключения) светодиода. Мы её тоже напишем, но с другими целями.

Итак, наш робот ждёт касания сенсора и только после этого выполняет нужные нам действия. Хорошо бы мигал светодиод, когда робот был готов к действиям. Поэтому нам нельзя использовать мигание по задержке, применяя команду *delay*. Сделаем это без неё, задействуя счётчик прошедших от запуска программы миллисекунд (команда *millis*).

### Задание 27

Напишите программу мигания встроенными светодиодами (рис. 62) без использования задержек.

```
#define LED_PIN    13
void setup() {
  pinMode (LED_PIN,OUTPUT);
}
void loop() {
  if (millis()/1000%2 == 0) {
    digitalWrite(LED_PIN, HIGH);
  }
  else {
    digitalWrite(LED_PIN, LOW);
  }
}
```

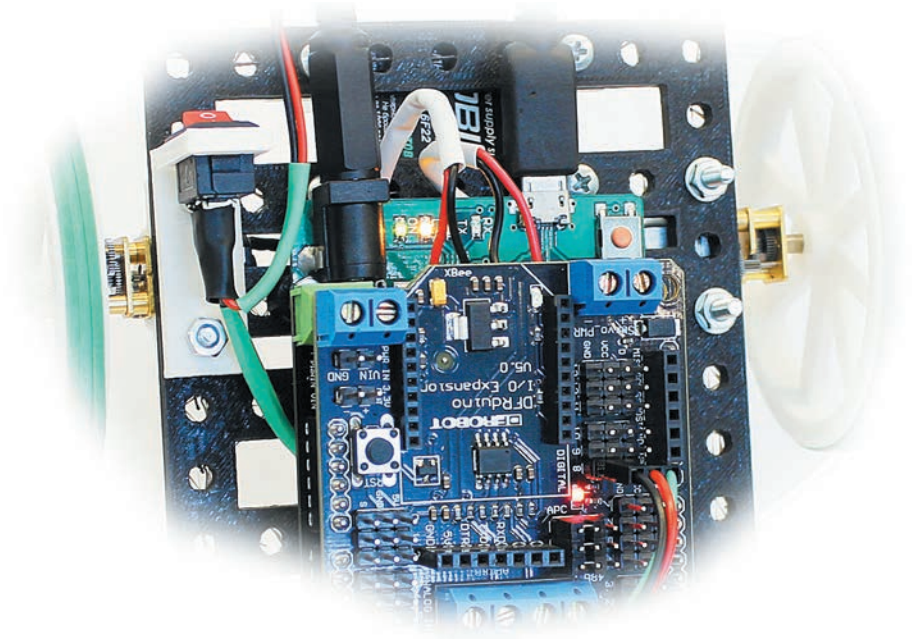


**Рис. 62.** Встроенные в платы светодиоды (контакт 13)

Мы проверяем, прошло чётное количество секунд или нет, и соответственно включаем или выключаем светодиод. А теперь давайте построим выражение для логической переменной, которая тоже может принимать два значения.

## Задание 28

Напишите программу мигания встроенным светодиодом без использования задержек и условного оператора *if*.

```
#define LED_PIN    13
bool ledStatus;
void setup() {
  pinMode (LED_PIN,OUTPUT);
}
void loop() {
  ledStatus = millis()/1000%2 ? true: false;
  digitalWrite(LED_PIN, ledStatus);
}
```

Если остаток от деления количества прошедших секунд с момента запуска программы равен нулю, то переменная принимает значение *true*. В противном случае — *false*.

## 6.5. Экономь память

Запустите скетч ещё раз и обратите внимание, что будет написано в консоли.

*Скетч использует 4154 байт.*
*Глобальные переменные используют 150 байт.*

А теперь измените код следующим образом:

```
#define LED_PIN    13
void setup() {
  pinMode (LED_PIN,OUTPUT);
}
void loop() {
 digitalWrite(LED_PIN,millis()/1000%2 ? true:false);
}
```

И запустите.

*Скетч использует 4136 байт.*
*Глобальные переменные используют 149 байт.*

Помните, что на микроконтроллерах к памяти нужно относиться очень бережно.

### Задание 29

Робот ждёт нажатия на сенсор касания и о своей готовности сообщает миганием светодиода. После касания робот должен проехать вперёд, затем развернуться и проследовать в исходную точку. Используйте процедуры.

```
#define DIR_M1         4
#define SPEED_M1       5
#define SPEED_M2       6
#define DIR_M2         7
#define BUTTON_PIN     8
#define LED_PIN       13
void forward (int velocity, int duration) {
  digitalWrite(DIR_M1, HIGH);
  digitalWrite(DIR_M2, HIGH);
  analogWrite (SPEED_M1,velocity);
  analogWrite (SPEED_M2,velocity);
  delay(duration);
```

```
  analogWrite (SPEED_M1,0);
  analogWrite (SPEED_M2,0);
}
void turn_right (int velocity, int duration){
  digitalWrite(DIR_M1, LOW);
  digitalWrite(DIR_M2, HIGH);
  analogWrite (SPEED_M1,velocity);
  analogWrite (SPEED_M2,velocity);
  delay(duration);
  analogWrite (SPEED_M1,0);
  analogWrite (SPEED_M2,0);
}
void setup() {
  for (int i=4;i<=7;i=i+1) pinMode(i,OUTPUT);
  pinMode (BUTTON_PIN,INPUT);
  pinMode (LED_PIN,OUTPUT);
}
void loop() {
  //ждём нажатия кнопки и мигаем светодиодом
  while(digitalRead(BUTTON_PIN)==false)
  digitalWrite(LED_PIN,millis()/1000%2 ?true:false);
  digitalWrite(LED_PIN, LOW);
  forward(100,2000);     delay (1000);
  turn_right (100,650);  delay (1000);
  forward(100,2000); }
```

# Глава 7
# КАК ЕХАТЬ ПРЯМО

## 7.1. Придумаем энкодер

### Задание 30

Проанализируйте ваши эксперименты с роботом. Сформулируйте, что вас не устраивает в том, как он выполняет задания.

Мы столкнулись с первой действительно большой проблемой — робот не двигается прямо. Главное — понимать, что он этого и не должен делать. Двух одинаковых предметов и объектов не бывает. Моторы разные, колёса разные, питание на моторах тоже полностью одинаковым быть не может. Хоть немного, но отличия всегда есть. Возникает вопрос: как в таких условиях заставить робота ехать прямо?

Для выполнения этой задачи сначала необходимо сделать устройство, которое будет как-то подсчитывать, на какую часть полного оборота повернулось колесо. Другими словами, необходимо сделать два энкодера, подсчитывающих количество оборотов колёс.

Поможет нам в этом датчик линии TCRT5000 (рис. 63) — это собранные в одном корпусе светодиод (он синий) и фототранзистор (тёмный). Светодиод излучает в инфракрасном диапазоне, свет отражается от поверхности и попадает на фототранзистор. У датчика два выхода: AO и DO. Логично, что это *analog output* — аналоговый выход и *digital output* — цифровой. Если поверхность перед датчиком светлая, то на выходе DO будет 1, а если тёмная — 0.

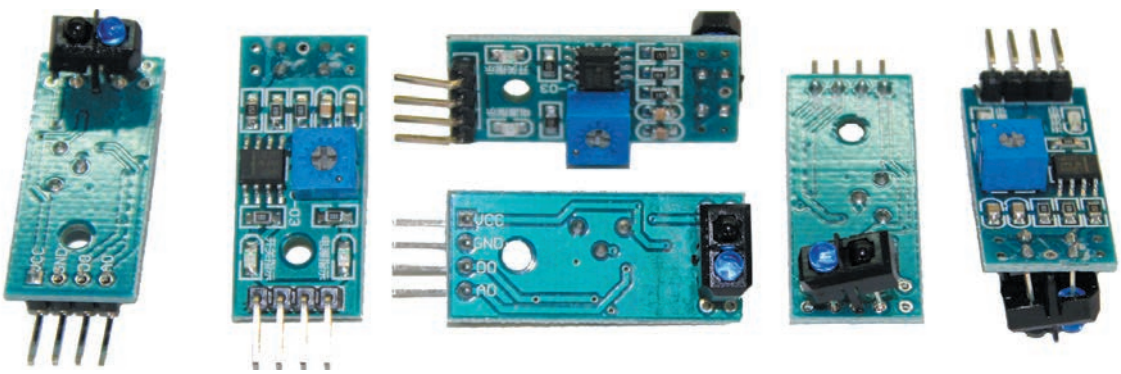



**Рис. 63.** Датчик на основе TCRT5000

## Задание 31

Соберите и прикрепите к раме робота модуль с двумя датчиками линии (рис. 64). Подключите датчики к цифровым контактам 2 и 3 на плате: красный — VCC, чёрный — GND и зелёный — D0–D.

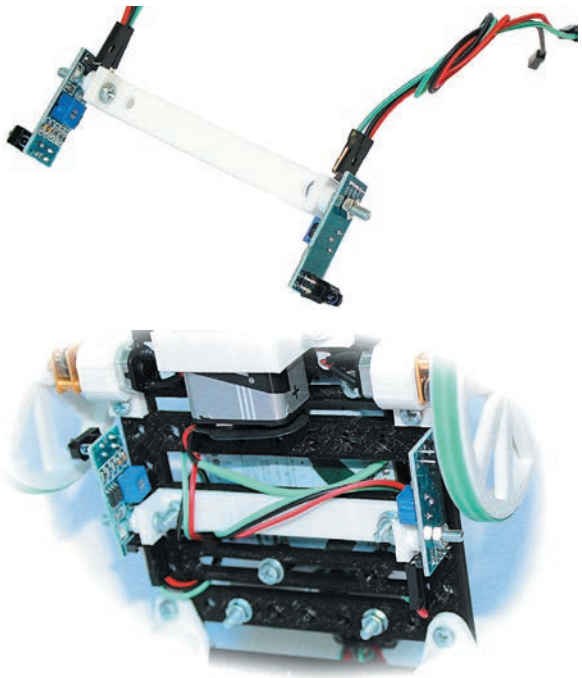


**Рис. 64.** Модуль для крепления датчиков

После подсоединения включите подачу питания. На обоих датчиках должны засветиться красные светодиоды (рис. 65), это означает, что на датчик подаётся питание. Если красный светодиод не светится, то проверьте подключение. В крайнем случае замените датчик.

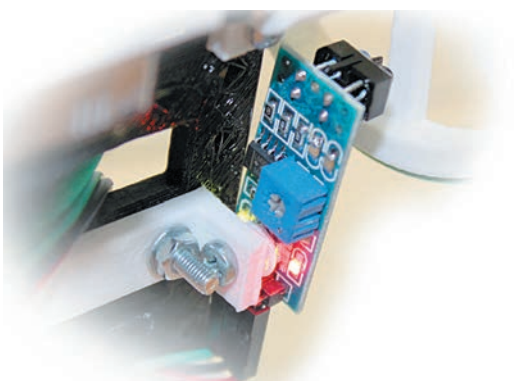


**Рис. 65.** Работающие светодиоды на датчике линии

После того как вы убедитесь, что датчики получают питание, немного покрутите моторы, при этом на каждом датчике должен загораться и гаснуть зелёный светодиод. Это говорит о том, что датчики видят (опознают) белые спицы колёс.

Далее, выключив питание, пометьте маркером по одной спице на каждом колесе — будет удобнее следить, сколько оборотов сделали колёса (рис. 66).

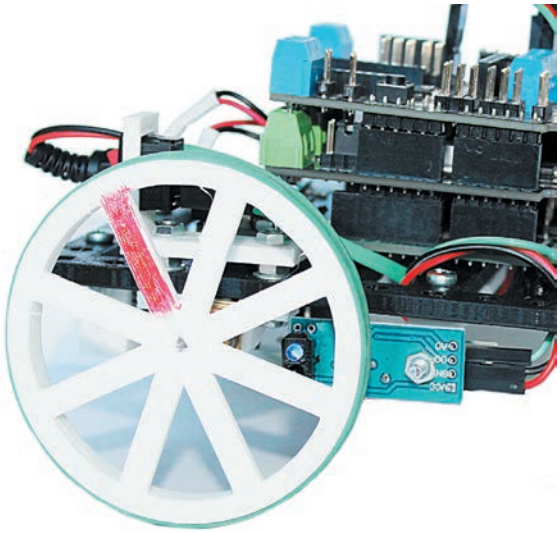


**Рис. 66.** Крепление датчиков (будущих энкодеров)

## Задание 32

Напишите программу, чтобы контроллер отображал на мониторе последовательного порта текущее состояния датчиков.

```
#define DIR_M1         4
#define SPEED_M1       5
#define SPEED_M2       6
#define DIR_M2         7
#define BUTTON_PIN     8
#define ENCODER_M1     2
#define ENCODER_M2     3
#define LED_PIN    13
bool encM1, encM2;
void setup() {
  for (int i=4;i<=7;i=i+1) pinMode(i,OUTPUT);
  pinMode (LED_PIN,OUTPUT);
  pinMode (BUTTON_PIN,INPUT);
  pinMode (ENCODER_M1,INPUT); pinMode (ENCODER_M2,INPUT);
```

```
  Serial.begin (9600);
  while(digitalRead(BUTTON_PIN)==false)
    digitalWrite(LED_PIN, millis()/1000%2 ? true: false);
}
void loop() {
  digitalWrite(DIR_M1, HIGH);
  digitalWrite(DIR_M2, HIGH);
  analogWrite (SPEED_M1,100);
  analogWrite (SPEED_M2,100);
  encM1 = digitalRead (ENCODER_M1);
  encM2 = digitalRead (ENCODER_M2);
  Serial.print ("M1: "); Serial.print (encM1);
  Serial.print ("   ");
  Serial.print ("M2: "); Serial.print (encM2);
  Serial.println ("   ");
  delay(10);
}
```

Мы сразу используем все удобства, которые были рассмотрены ранее. Запуск только по касанию, индикация готовности мигающими светодиодами. Увидели, что робот сообщил о готовности, открыли монитор последовательного порта, коснулись сенсора и спокойно смотрите показания датчиков.

Во время тестирования программы на мониторе можно заметить, что показания датчиков различаются.

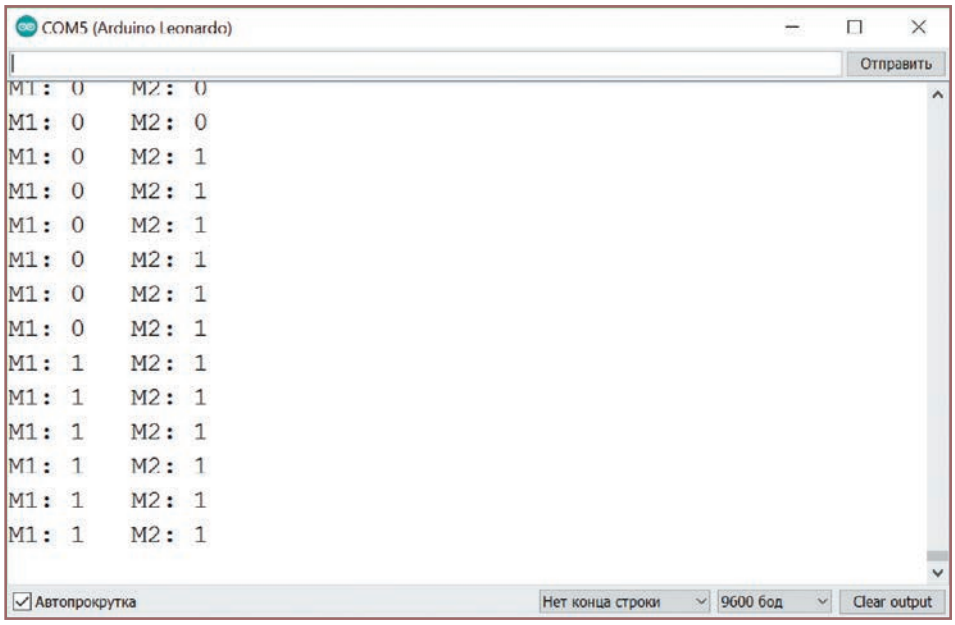



**Рис. 67.** Информация в окне монитора последовательного порта

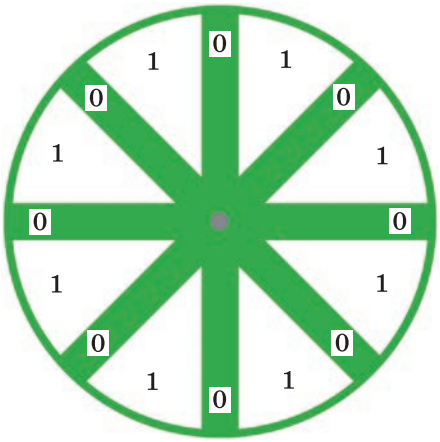



**Рис. 68.** Показания датчика на разных частях колёсного диска

Датчик «видит» белую спицу как 0, а её отсутствие — как 1 (рис. 67 и 68). Таким образом можно подсчитать количество спиц. Их количество равно числу переходов от 1 к 0. Следовательно, нам необходимо помнить, что показывал датчик мгновение назад, т. е. в предыдущий его опрос.

Можно подойти и немного по-другому. У нас датчик расположен таким образом, что участки белого и не белого цвета примерно одинаковые. Значит, мы можем считать, не только сколько белых сегментов, но и сколько чёрных. В этом случае точность удвоится и мы насчитаем не 7 сегментов на один оборот, а уже 14. Будет меньше ошибок.

## Задание 33

Напишите программу, чтобы микроконтроллер считал спицы и выводил их количество на монитор последовательного порта (рис. 69).

```
#define DIR_M1         4 //правый мотор
#define SPEED_M1       5
#define SPEED_M2       6
#define DIR_M2         7 //левый мотор
#define BUTTON_PIN     8
#define ENCODER_M1     2
#define ENCODER_M2     3
#define LED_PIN    13
bool encM1, encM2, encOldM1,encOldM2;
int countM1=0, countM2=0;
void setup() {
  for (int i=4;i<=7;i=i+1) pinMode(i,OUTPUT);
```

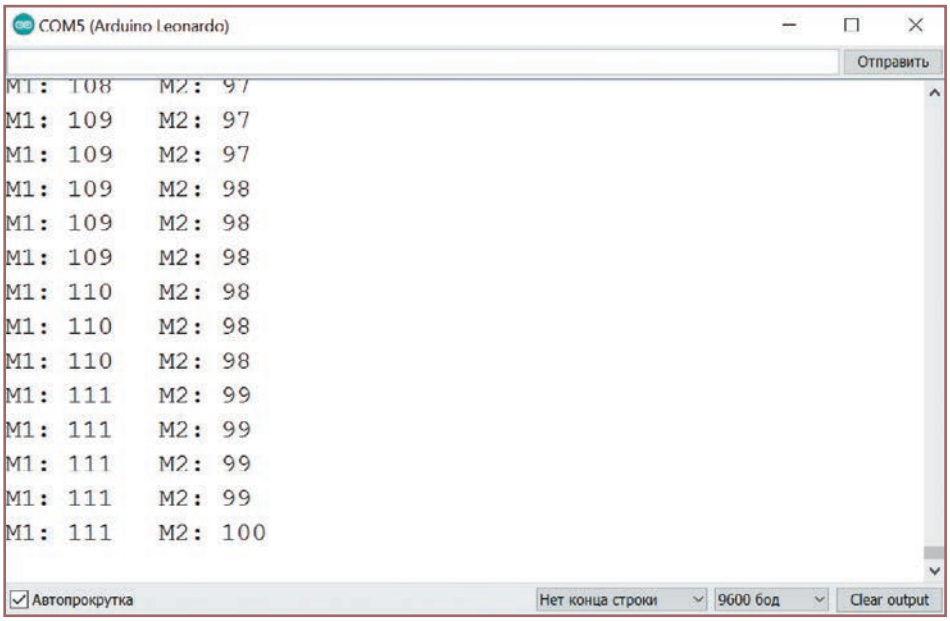



**Рис. 69.** Вывод значений счётчиков

```
  pinMode (LED_PIN,OUTPUT);
  pinMode (BUTTON_PIN,INPUT);
  pinMode (ENCODER_M1,INPUT); pinMode (ENCODER_M2,INPUT);
  Serial.begin (9600);
  while(digitalRead(BUTTON_PIN)==false)
    digitalWrite(LED_PIN, millis()/1000%2 ? true: false);
  encOldM1 = digitalRead (ENCODER_M1);
  encOldM2 = digitalRead (ENCODER_M2);
}
void loop() {
  while (countM1<100 || countM2 <100) {
   digitalWrite(DIR_M1, HIGH);
   digitalWrite(DIR_M2, HIGH);
   analogWrite (SPEED_M1,100);
   analogWrite (SPEED_M2,100);
   encM1 = digitalRead (ENCODER_M1);
   encM2 = digitalRead (ENCODER_M2);
   if (encM1 != encOldM1) countM1++;
   if (encM2 != encOldM2) countM2++;
   Serial.print ("M1: "); Serial.print (countM1);
   Serial.print ("   ");
   Serial.print ("M2: "); Serial.print (countM2);
   Serial.println ("   ");
   delay(10);
```

```
    encM1_old = encM1;
    encM2_old = encM2;
  }
  analogWrite (SPEED_M1,0);
  analogWrite (SPEED_M2,0);
```

### Задание 34

Проведите серию экспериментов. Обсудите идеи, как можно выровнять движение робота.

*Не забывайте использовать вывод данных на монитор последовательного порта для поиска опечаток, ошибок, для отладки вашей программы.*

## 7.2. Управление по ошибке

Получившиеся счётчики спиц можно смело называть счётчиками оборотов, или энкодерами. К сожалению, они позволяют делать 14 измерений на 360°, т. е. отслеживают угол в 25,7°. Но, сделав энкодер, мы получили важные данные, которые можно использовать для корректировки движения робота по прямой. Если один счётчик оборотов станет показывать больше, чем другой, то можно скорость одного колеса уменьшить, а другого — увеличить.

Пусть `countM1`, `countM2` — наши счётчики, а робот должен двигаться со скоростью `motorsPower` = 100. Когда показания равны — робот едет прямо, т. е. каждый из моторов вращается с этой скоростью. Как только показания стали разными — ошибка, и надо менять мощность моторов (назовём их `powerM1` и `powerM2`). Итак, наша ошибка (назовём её `error`) — это разность в показаниях счётчиков:

```
error = countM1 – countM2;.
```

Тогда мощности моторов будут:

```
powerM1 = motorsPower – K·error;
powerM2 = motorsPower + K·error;
```

При этом, как вы видите, необходим ещё коэффициент K > 1 (даже больше 10). Почему? Скорость и ошибка отличаются на два порядка. Чтобы выровнять робота, нам надо пропорционально ошибке изменять мощность моторов сравнимой величиной.

## Задание 35

**Исследуйте работу приведённой ниже программы. Запрограммируйте робота, используя разные коэффициенты K.**

```
#define DIR_M1         4 //правый мотор
#define SPEED_M1       5
#define SPEED_M2       6
#define DIR_M2         7 //левый мотор
#define BUTTON_PIN     8
#define ENCODER_M1     2
#define ENCODER_M2     3
#define LED_PIN       13
bool encM1, encM2, encOldM1,encOldM2;;
int countM1=0, countM2=0;
int K=20, motorsPower=100, powerM1, powerM2,
error=0;
void setup() {
  for (int i=4;i<=7;i=i+1) pinMode(i,OUTPUT);
  pinMode (LED_PIN,OUTPUT);
  pinMode (BUTTON_PIN,INPUT);
  pinMode (ENCODER_M1,INPUT);
  pinMode (ENCODER_M2,INPUT);
  //Serial.begin (9600);
  while(digitalRead(BUTTON_PIN)==false)
   digitalWrite(LED_PIN,millis()/1000%2?true:false);
  encOldM1 = digitalRead (ENCODER_M1);
  encOldM2 = digitalRead (ENCODER_M2);
}
void loop() {
  while (countM1<100 || countM2 <100) {
   encM1 = digitalRead (ENCODER_M1);
   encM2 = digitalRead (ENCODER_M2);
   if (encM1!=encOldM1) countM1++;
   if (encM2!=encOldM2) countM2++;
   error = countM1 – countM2;
   powerM1= motorsPower – K*error;
   powerM2= motorsPower + K*error;
   digitalWrite(DIR_M1, HIGH);
   digitalWrite(DIR_M2, HIGH);
   analogWrite (SPEED_M1,powerM1);
   analogWrite (SPEED_M2,powerM2);
```

```
    /*Serial.print ("M1: "); Serial.print (countM1);
      Serial.print ("   ");
      Serial.print ("M2: "); Serial.print (countM2);
      Serial.println ("   ");*/
    delay(10);
    encOldM1 = encM1;
    encOldM2 = encM2;
   }
   analogWrite (SPEED_M1,0);
   analogWrite (SPEED_M2,0);
   while(digitalRead(BUTTON_PIN)==false)
     digitalWrite(LED_PIN, millis()/1000%2 ? true: false);
   countM1=0; countM2=0; error=0;
}
```

Итак, благодаря управлению, пропорциональному ошибке, траектория движения робота стала больше похожа на прямую.

Давайте далее улучшать наш алгоритм. Мы учитывали ошибку в данный момент времени, т. е. текущую ошибку. Если коэффициент K взять малым, то траектория по-прежнему будет иметь крен либо вправо, либо влево (уж у кого какие моторы). Если коэффициент взять очень большой — начинаются колебания (траектория похожа на волну).

Можно попробовать суммировать ошибки, а так как у нас ошибка очень маленькая, то будем суммировать усиленные коэффициентом ошибки:

```
controlSummErrors=controlSummErrors  +  K2*error;
```

Также необходимо обеспечить плавный старт робота и плавное торможение.

## Задание 36

Исследуйте работу приведённой ниже программы. Запрограммируйте робота, используя разные коэффициенты K1 и K2. Подберите для вашего робота значения этих коэффициентов, чтобы движение было лучше, чем у робота на видео (https://youtu.be/yX10Sijshqg).

```
#define DIR_M1          4 //правый мотор
#define SPEED_M1        5
#define SPEED_M2        6
#define DIR_M2          7 //левый мотор
#define BUTTON_PIN      8
#define ENCODER_M1      2
```

```
#define ENCODER_M2    3
#define LED_PIN      13
#define START_SPEED  50 //стартовая скорость
//целевая скорость
const int needSpeed=100;
bool encM1, encM2, encOldM1,encOldM2;;
int countM1=0, countM2=0;
int K1=26, motorsPower=START_SPEED;
int powerM1, powerM2, error=0;
float K2=0.2, controlSummErrors=0.0;
void setup() {
 for (int i=4;i<=7;i=i+1) pinMode(i,OUTPUT);
 pinMode (LED_PIN,OUTPUT);
 pinMode (BUTTON_PIN,INPUT);
 pinMode (ENCODER_M1,INPUT);
 pinMode (ENCODER_M2,INPUT);
 //Serial.begin (9600);
 while(digitalRead(BUTTON_PIN)==false)
  digitalWrite(LED_PIN, millis()/1000%2 ? true: false);
 encOldM1 = digitalRead (ENCODER_M1);
 encOldM2 = digitalRead (ENCODER_M2);
}
void loop() {
 while (countM1 < 100 || countM2 < 100) {
  /* плавное увеличение скорости,
   * тернарный оператор в цикле
   * обеспечивает увеличение до целевой скорости
   */
  motorsPower = motorsPower<100 ? motorsPower+=1 :
needSpeed;
  /*работаем с энкодерами и счётчиками*/
  encM1 = digitalRead (ENCODER_M1);
  encM2 = digitalRead (ENCODER_M2);
  if (encM1 != encOldM1) countM1++;
  if (encM2 != encOldM2) countM2++;
  /* вычисляем ошибку и управляющие
   * воздействия на моторы */
  error = countM1 — countM2;
  controlSummErrors=controlSummErrors + K2*error;
  int controlAction = K1*error + controlSummErrors;
```

```
  /* также ограничиваем допустимыми
   * значениями управления через ШИМ */
  powerM1=constrain (motorsPower – controlAction,0,255);
  powerM2=constrain (motorsPower + controlAction,0,255);
  digitalWrite(DIR_M1, HIGH);
  digitalWrite(DIR_M2, HIGH);
  analogWrite (SPEED_M1,powerM1);
  analogWrite (SPEED_M2,powerM2);
  //Serial.print (powerM1);
  //Serial.print ("  ");Serial.println (powerM2);
  delay(5);
  encOldM1 = encM1;
  encOldM2 = encM2;
 }
 /* плавное увеличение скорости
  * находим среднее мощности моторов
  * и быстро уменьшаем скорость
  */
 motorsPower = (powerM1+powerM2)/2;
 while (motorsPower>0) {
  analogWrite (SPEED_M1,motorsPower);
  analogWrite (SPEED_M2,motorsPower);
  motorsPower-=1;
  //Serial.println (motorsPower);
  delay(1);
 }
 analogWrite (SPEED_M1,0);
 analogWrite (SPEED_M2,0);
 //ждём касания, чтобы можно было повторить всё
 while(digitalRead(BUTTON_PIN)==false)
  digitalWrite(LED_PIN, millis()/1000%2 ? true: false);
 countM1=0; countM2=0; error=0;
}
```

**Робот поехал прямо. Конечно, совсем не идеально, но для наших энкодеров, можно сказать, маленькое «чудо» мы сделали.**

**Если использовать моторы N20 со встроенным энкодером (рис. 70), то движение робота по данному алгоритму будет практически идеальным.**

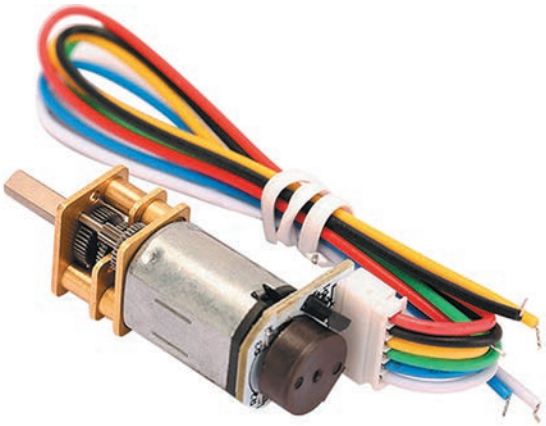


**Рис. 70.** Мотор N20 со встроенным энкодером

Что касается программы, то она постепенно разрослась, и с этим нужно что-то делать.

# Глава 8
# НЕСКОЛЬКО ИСХОДНЫХ ФАЙЛОВ

## 8.1. Работаем с несколькими файлами

Сначала научимся работать с несколькими файлами одновременно. Это удобнее, чем всё писать в одном файле.

### Задание 37

Создайте один основной и два вспомогательных файла для работы.

Запустите Arduino IDE, создайте новый файл и сохраните, например, под именем *robotMoveTest* (автоматически создадутся каталог с этим именем и одноимённый файл с расширением ino). Далее чуть ниже иконки монитора последовательного порта вызовите меню и выберите создание новой вкладки (рис. 71).

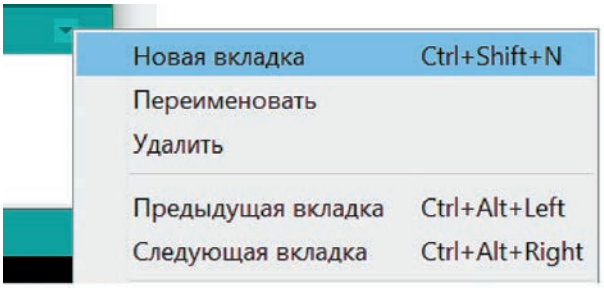



**Рис. 71.** Меню работы с вкладками

В появившейся оранжевой строчке наберите имя файла robotMoveSetup.h. Нажмите CTRL + S, чтобы сохранить файл (рис. 72).

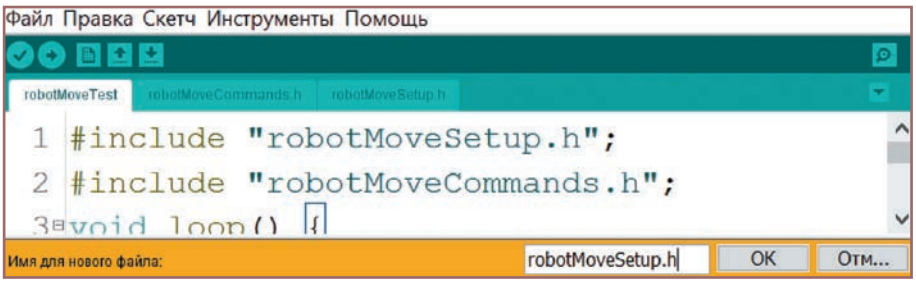



**Рис. 72.** Создание нового вспомогательного файла

Создайте ещё одну вкладку с именем robotMoveCommands.h и сохраните файл. В вашем каталоге с программой теперь должно быть три файла (рис. 73).

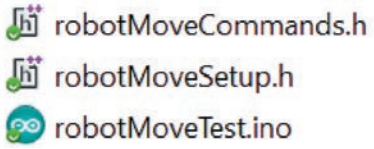



**Рис. 73.** Созданные файлы

Чтобы подключить два вспомогательных файла к основному, необходимо набрать команду #include, как показано на рисунке 74. Переключаться по вкладкам просто и удобно.

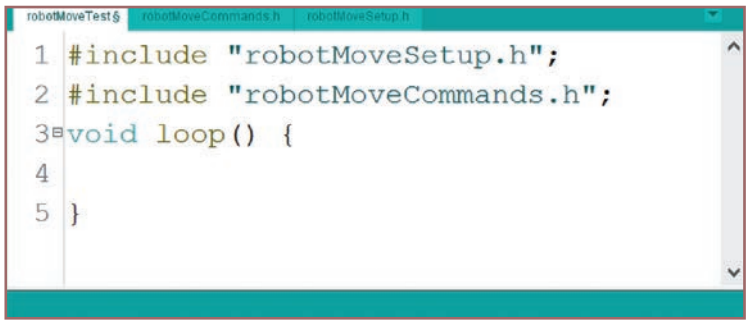



**Рис. 74.** Подключение файлов к основному командой #include

## Задание 38

В первом файле пропишите все директивы, переменные, настройки, а во втором создайте функции движения робота вперёд, назад, вправо и влево. Таким образом, в основном файле можно будет указывать только непосредственно алгоритм поведения робота.

**Файл robotMoveTest.ino**

```
#include "robotMoveSetup.h";
#include "robotMoveCommands.h";
void loop() {
 //указать скорость и счётчик на энкодере
  moveForward(100,30);  delay (1000);
  turnLeft (80,6);      delay (1000);
  moveBack(100,50);     delay (1000);
  turnLeft (80,6);      delay (1000);
  moveForward(100,30);  delay (1000);
```

```
  turnLeft (80,6);      delay (1000);
  moveBack(100,50);
  while (true);
}
```

**Файл robotMoveSetup.h**

```
#define DIR_M1         4
#define SPEED_M1       5
#define SPEED_M2       6
#define DIR_M2         7
#define BUTTON_PIN     8
#define ENCODER_M1     2
#define ENCODER_M2     3
#define LED_PIN       13
#define START_SPEED   50
bool encM1, encM2, encOldM1,encOldM2;;
int countM1=0, countM2=0;
int K1=26, motorsPower=START_SPEED;
int powerM1, powerM2, error=0;
float K2=0.2, controlSummErrors=0.0;
void setup() {
 for (int i=4;i<=7;i=i+1) pinMode(i,OUTPUT);
 pinMode (LED_PIN,OUTPUT);
 pinMode (BUTTON_PIN,INPUT);
 pinMode (ENCODER_M1,INPUT);
 pinMode (ENCODER_M2,INPUT);
 //Serial.begin (9600);
 while(digitalRead(BUTTON_PIN)==false)
  digitalWrite(LED_PIN, millis()/1000%2 ? true: false);
}
```

**Файл robotMoveCommands.h**

```
void moveForward(int needSpeed, int countLimit) {
 encOldM1 = digitalRead (ENCODER_M1);
 encOldM2 = digitalRead (ENCODER_M2);
 while (countM1 < countLimit || countM2 < countLimit) {
  motorsPower = motorsPower<100 ? motorsPower+=1 :
needSpeed;
  encM1 = digitalRead (ENCODER_M1);
  encM2 = digitalRead (ENCODER_M2);
  if (encM1 != encOldM1) countM1++;
  if (encM2 != encOldM2) countM2++;
```

```
  error = countM1 – countM2;
  controlSummErrors=controlSummErrors + K2*error;
  int controlAction = K1*error + controlSummErrors;
  powerM1=constrain (motorsPower – controlAction,0,255);
  powerM2=constrain (motorsPower + controlAction,0,255);
  digitalWrite(DIR_M1, HIGH);
  digitalWrite(DIR_M2, HIGH);
  analogWrite (SPEED_M1,powerM1);
  analogWrite (SPEED_M2,powerM2);
  delay(5);
  encOldM1 = encM1;
  encOldM2 = encM2;
 }
 motorsPower = (powerM1+powerM2)/2;
 while (motorsPower>0) {
  analogWrite (SPEED_M1,motorsPower);
  analogWrite (SPEED_M2,motorsPower);
  motorsPower-=1;
  delay(1);
 }
 analogWrite (SPEED_M1,0);
 analogWrite (SPEED_M2,0);
 countM1=0; countM2=0;
}
//только HIGH на LOW поменять
void moveBack(int needSpeed, int countLimit) {
//всё аналогично, только HIGH на LOW поменять
...
}
void turnLeft(int needSpeed, int countLimit) {
 encOldM1 = digitalRead (ENCODER_M1);
 encOldM2 = digitalRead (ENCODER_M2);
 while (countM1 < countLimit || countM2 < countLimit) {
  encM1 = digitalRead (ENCODER_M1);
  encM2 = digitalRead (ENCODER_M2);
  if (encM1 != encOldM1) countM1++;
  if (encM2 != encOldM2) countM2++;
  digitalWrite(DIR_M1, HIGH);
  digitalWrite(DIR_M2, LOW);
  analogWrite (SPEED_M1,needSpeed);
  analogWrite (SPEED_M2,needSpeed);
  delay(5);
  encOldM1 = encM1;
```

```
    encOldM2 = encM2;
   }
   analogWrite (SPEED_M1,0);
   analogWrite (SPEED_M2,0);
   countM1=0; countM2=0;
}
void turnRight(int needSpeed, int countLimit) {
//только HIGH на LOW поменять и наоборот
...
}
```

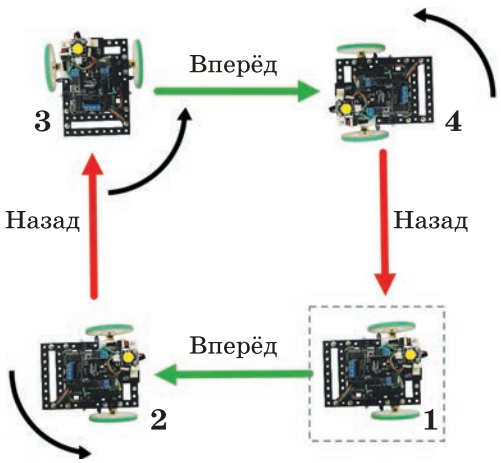



**Рис. 75.** Схема движения робота

### Задание 39

Внесите изменения в программы, чтобы ваш робот проехал по квадрату или прямоугольнику (рис. 75). Пример выполненного задания можно посмотреть по ссылке: https://youtu.be/XbC-oc3nC1k.

## 8.2. Создание своей библиотеки

Очень часто одни и те же части кода копируются из программы в программу. Чтобы не писать этот код каждый раз заново, его выносят в отдельные файлы — библиотеки. Мы рассмотрим самый простой способ создания библиотеки. Правильный способ рассмотрим в главе 13.

## Задание 40

Создайте свою библиотеку *robotMove* с командами движения робота вперёд, назад, вправо, влево и инициализацией подключаемых датчиков.

Обычно при создании библиотеки необходимы два файла: заголовочный файл и файл с кодом библиотеки, а также каталог с примерами.

Откройте проводник и последовательно перейдите по папкам:

*Документы* → *Arduino* → *libraries*.

В папке *libraries* создайте каталог с названием библиотеки (название может состоять только из латинских букв). Перейдите в эту папку и создайте в ней два текстовых файла и каталог (рис. 76).

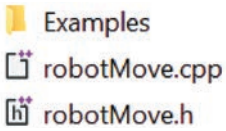



**Рис. 76.** Необходимые файлы и папка

Используя программу *Блокнот*, откройте оба файла. Запустите Arduino IDE и создайте новый файл. Сохраните его в папку *Examples* под именем *defaultRobot*.

Затем тот код, который у вас написан для движения робота по квадрату, необходимо распределить по этим трём файлам.

В файле *robotMove.h* (рис. 77) прописываем все неизменные директивы (контакты изменяться уже не будут) и список функций. Обратите внимание: нам нужно написать ещё несколько функций и подправить уже имеющиеся.

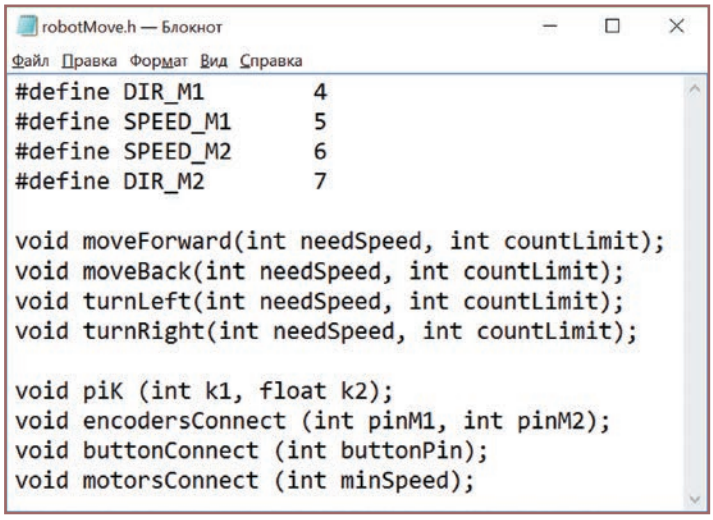



**Рис. 77.** Заголовочный файл

Из названия дополнительных функций вы видите, что они необходимы, чтобы указывать значения коэффициентов, номера контактов для подключения энкодеров и минимальную скорость.

Далее оформляем файл с кодом библиотеки.

**Файл robotMove.cpp**

```
#include <Arduino.h>
#include <robotMove.h>

bool encM1, encM2, encOldM1,encOldM2;
int countM1=0, countM2=0, error=0, K1;
int motorsPower, powerM1, powerM2;
float controlSummErrors=0.0, K2;
byte encoderPinM1, encoderPinM2, buttonPin, startSpeed;

void piK (int k1, float k2) {
   K1=k1; K2=k2;
}
void encodersConnect (int pinM1, int pinM2) {
  pinMode (pinM1,INPUT); pinMode (pinM2,INPUT);
  encoderPinM1 = pinM1; encoderPinM2 = pinM2;
}
void buttonConnect (int buttonPin) {
  pinMode (buttonPin,INPUT);
}
void motorsConnect (int minSpeed) {
 for (int i=4;i<=7;i=i+1) pinMode(i,OUTPUT);
 startSpeed = minSpeed;
}
void moveForward(int needSpeed, int countLimit) {
 motorsPower = startSpeed;
 encOldM1 = digitalRead(encoderPinM1);
 encOldM2 = digitalRead(encoderPinM2);
 while (countM1<countLimit || countM2<countLimit) {
  motorsPower = motorsPower<100 ? motorsPower+=1 :
needSpeed;
  encM1 = digitalRead(encoderPinM1);
  encM2 = digitalRead(encoderPinM2);
  if (encM1 != encOldM1) countM1++;
  if (encM2 != encOldM2) countM2++;
  error = countM1 — countM2;
  controlSummErrors = controlSummErrors + K2*error;
  int controlAction = K1*error + controlSummErrors;
  powerM1=constrain(motorsPower-controlAction,0,255);
  powerM2=constrain(motorsPower+controlAction,0,255);
```

```
  digitalWrite(DIR_M1, HIGH);
  digitalWrite(DIR_M2, HIGH);
  analogWrite(SPEED_M1,powerM1);
  analogWrite(SPEED_M2,powerM2);
  delay(5);
  encOldM1 = encM1; encOldM2 = encM2;
 }
 motorsPower = (powerM1+powerM2)/2;
 while (motorsPower>0) {
  analogWrite(SPEED_M1,motorsPower);
  analogWrite(SPEED_M2,motorsPower);
  motorsPower-=1;
  delay(1);
 }
 analogWrite(SPEED_M1,0); analogWrite(SPEED_M2,0);
 countM1=0; countM2=0;
}
void moveBack(int needSpeed, int countLimit) {
 ...//всё аналогично (поменять только HIGH на LOW)
}
void turnLeft(int needSpeed, int countLimit) {
 encOldM1 = digitalRead (encoderPinM1);
 encOldM2 = digitalRead (encoderPinM2);
 while (countM1<countLimit || countM2<countLimit) {
  encM1 = digitalRead(encoderPinM1);
  encM2 = digitalRead(encoderPinM2);
  if (encM1 != encOldM1) countM1++;
  if (encM2 != encOldM2) countM2++;
  digitalWrite(DIR_M1, HIGH);
  digitalWrite(DIR_M2, LOW);
  analogWrite(SPEED_M1,needSpeed);
  analogWrite(SPEED_M2,needSpeed);
  delay(5);
  encOldM1 = encM1;
  encOldM2 = encM2;
 }
 analogWrite(SPEED_M1,0); analogWrite(SPEED_M2,0);
 countM1=0; countM2=0;
}
void turnRight(int needSpeed, int countLimit) {
...// всё аналогично, только поменять LOW на HIGH
   // и наоборот
}
```

Осталось заполнить пример. Чтобы через некоторое время ничего не забыть, необходимо написать подробные комментарии.

**Файл defaultRobot.ino**

```
#include <robotMove.h>
//подключить библиотеку
/*указать контакты: светодиода,
 * сенсора касания и двух энкодеров */
#define LED_PIN      13
#define BUTTON_PIN    8
#define ENCODER_M1    2
#define ENCODER_M2    3
/*указать самую медленную скорость,
 при которой робот ещё может ехать */
#define START_SPEED  50
void setup() {
 /*провести инициализацию подключения
  * энкодеров, сенсора касания и моторов  */
 encodersConnect (ENCODER_M1,ENCODER_M2);
 buttonConnect (BUTTON_PIN);
 motorsConnect (START_SPEED);
 /*указать коэффициенты управления
  * по ошибке и накоплению ошибки */
 piK (26,0.2);
 //Serial.begin (9600);
 /*настройте запуск робота по касанию и мигание
  * светодиодом: 2 секунды включён, одну выключен)*/
 while(digitalRead(BUTTON_PIN)==false)
  digitalWrite(LED_PIN,millis()/1000%3 ?true:false);
}
void loop() {
  /*в командах указывается скорость
   * и количество делений, посчитанных энкодером */
  int delayTime =1000;
  moveForward(100,30);   delay (delayTime);
  turnLeft (80,6);       delay (delayTime);
  moveBack(100,50);      delay (delayTime);
  turnLeft (80,6);       delay (delayTime);
  moveForward(100,30);   delay (delayTime);
  turnLeft (80,6);       delay (delayTime);
  moveBack(100,50);
 /*поставьте "заглушку",
```

```
    * чтобы цикл не выполнялся бесконечно */
    while (true);
}
```

В таком виде файл уже можно сохранить. Если компьютер не позволяет сохранить, предупреждая, что файл только для чтения, — сохраните в любое место, а потом переименуйте и перепишите файл. После перезагрузки Arduino IDE ваша библиотека станет доступна.

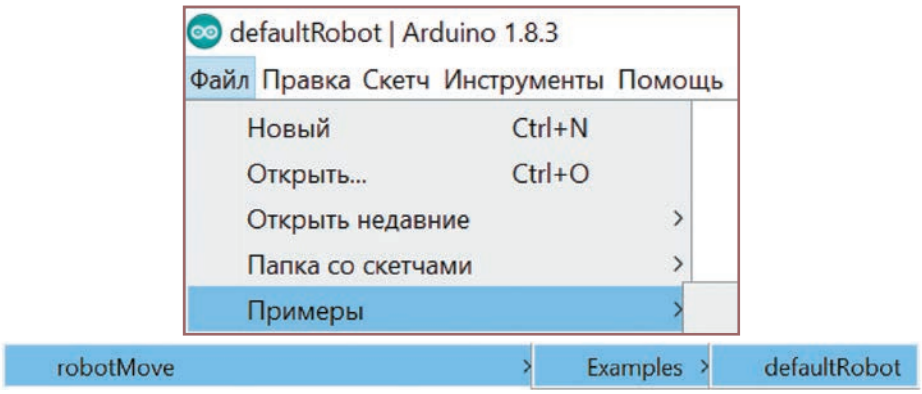



**Рис. 78.** Наличие примеров в меню Arduino IDE

При выполнении заданий вы действительно сможете оценить, как это удобно, когда у вас создана своя библиотека.

# Глава 9
# КЕГЕЛЬРИНГ

## 9.1. Самое первое соревнование

Настала пора немножечко посоревноваться и проверить, как усвоен вами материал. Задание классическое (рис. 79), очень много алгоритмов представлено в Интернете, однако на платформе Arduino изобилия реализаций нет. Вам необходимо подготовить автономного робота, способного вытолкнуть восемь кеглей за пределы внутреннего белого круга ринга. Робот находится в центре.

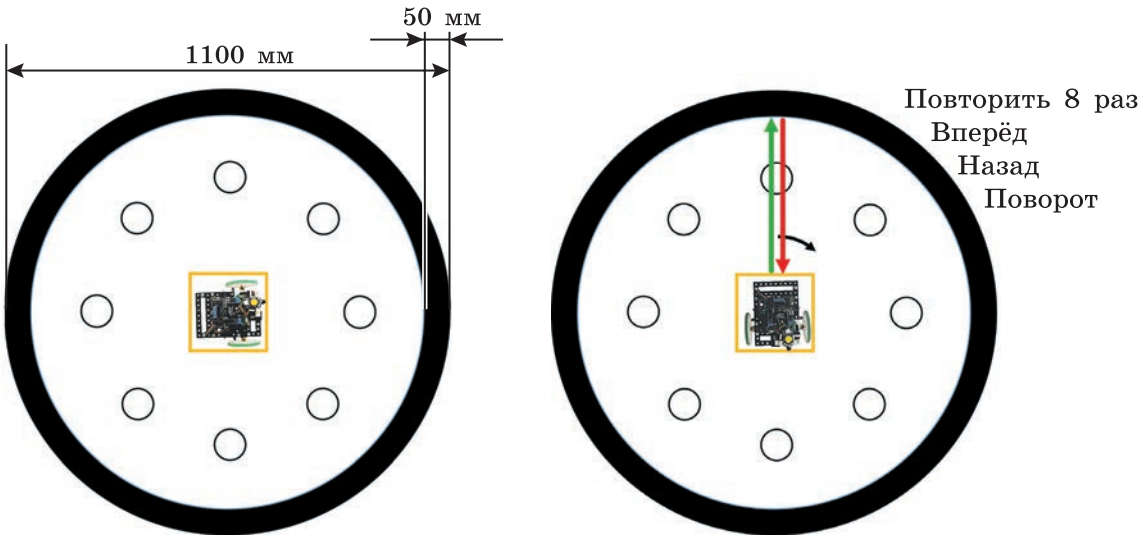



**Рис. 79.** Поле «Кегельринг» и первый алгоритм

### Задание 41

Обдумайте способы выполнения задания, кроме представленного на рисунке 79.

## 9.2. Сделаем бампер

В большинстве правил по кегельрингу робот может иметь размеры 20 × 20 см. Поэтому есть смысл на основе разработанных балок (рис. 80) создать модели переднего и заднего бамперов для робота.

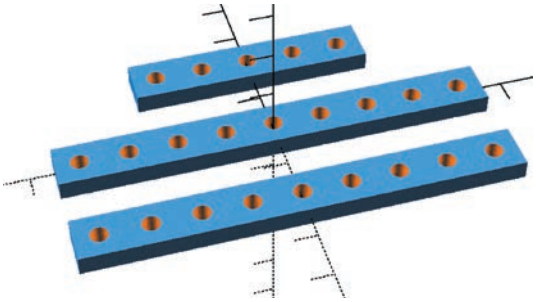


**Рис. 80.** Балки для бампера

## Задание 42

Создайте модель заднего бампера. Крепление для балок придумайте или используйте модель на рисунке 81. Распечатайте и установите бампер (рис. 82).

```
1  $fn=80 ;
2  module balka  (x)  {
3    difference ()  {
4      cube ( [x, 10, 4], true) ;
5      for ( i=[1:x/10] )
6        translate ( [-x/2-5+x%10/2+i*10, 0, 0] )
7        cylinder (5, 1.8, 1.8, true) ;
8    }
9  }
10 lenght=4 0 ;  lenghtSmall=30 ;
11 balka  (lenght) ;
12 translate ( [lenght/2, 0, lenghtSmall/2-5] )
13 rotate ( [0, 85, 0])
14 balka (lenghtSmall) ;
```

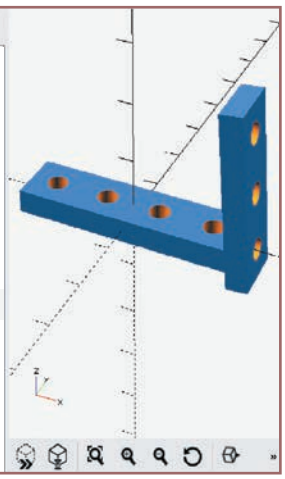


**Рис. 81.** Модель крепежа заднего бампера

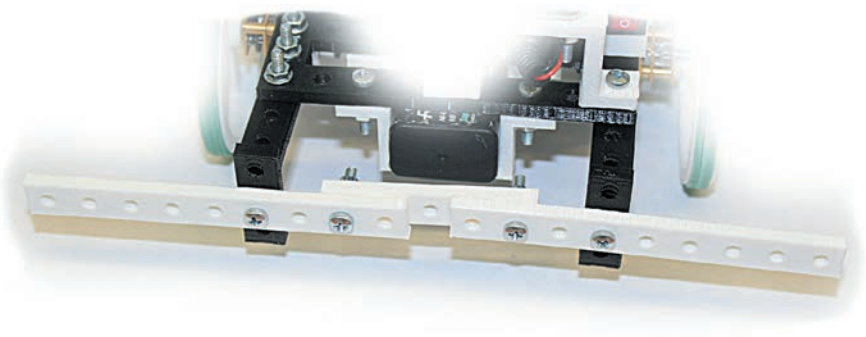


**Рис. 82.** Установленный задний бампер

## Задание 43

Создайте модель переднего бампера. Крепление для балок придумайте или используйте модель на рисунке 83. Распечатайте и установите бампер (рис. 84).

Обратите внимание, что необходимо сделать симметричные детали для переднего бампера, а не одинаковые.

```
1  $fn=48;
2  module balka (x)  {
3    difference  ()  {
4      cube ( [x, 10, 4], true) ;
5      for ( i=[1:x/10] ) t
6        ranslate ( [–x/2–5+x% 10/2+i*10 ,0 , 0] )
7        cylinder (5, 1.8, 1.8, true) ;
8    }
9  }
10 rotate ( [–90, 0, 0] )  {
11   lenght=50 ;  lenghtSmall=30 ;
12   color ("red")  balka (lenght) ;
13   translate ( [lenght/2, 0, 0] )
14   rotate ( [0, 85, 0] )
15   color("green") balka (lenghtSmall);
16   translate ( [–lenght/2+5, –lenghtSmall/2–5, 0] )
17   rotate ( [0, 0, 90 ])
18   color ("gold")  balka (lenghtSmall) ;
19 }
```

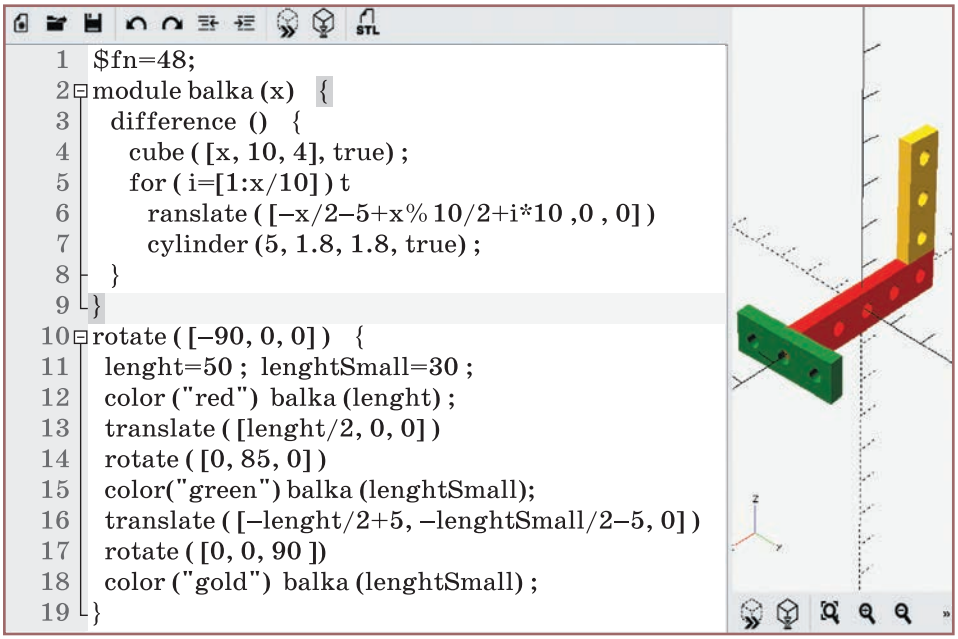


**Рис. 83.** Модель крепежа переднего бампера

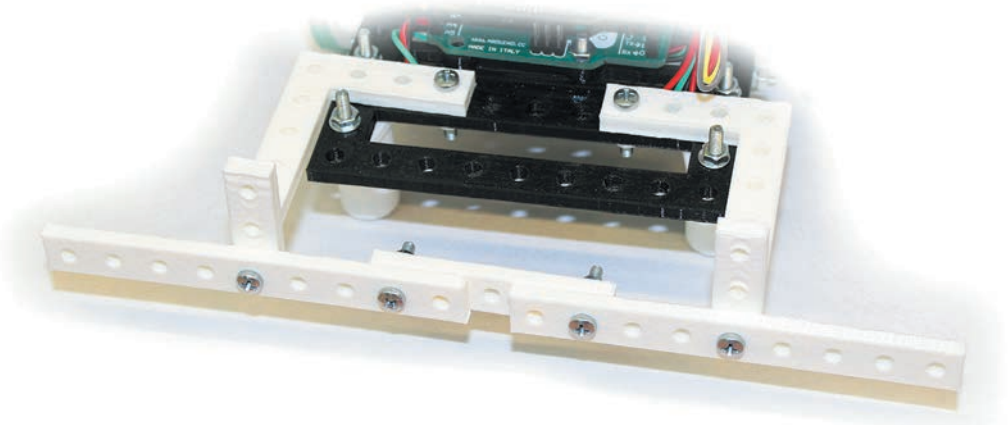
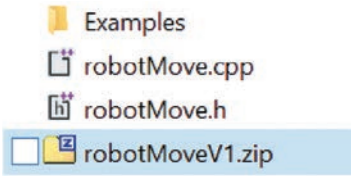


**Рис. 84.** Установленный передний бампер

## 9.3. Укажем сантиметры

Согласитесь, указывать непонятные числа в командах для робота, мягко говоря, как-то странно. Хорошо бы изменить код так, чтобы указывать число сантиметров, которые должен проехать робот с заданной скоростью.

То есть наша задача: перевести указанное число сантиметров в двойной счётчик спиц (у нас счётчик количества переходов белое/чёрное и чёрное/белое).




Предстоит внести изменения в библиотеку, поэтому предварительно создайте архив с прошлой версией и сохраните её на ваш носитель.

У нашего робота можно измерить диаметр, и тогда узнаем длину окружности — пройденное расстояние за один оборот колеса:

$$L = \pi \cdot D.$$

Следовательно, расстояние, пройденное при повороте на 1 часть:

$$L_1 = \pi \cdot D / 14.$$

Тогда для $n$ частей:

$$L_n = n \cdot \pi \cdot D / 14.$$

Выражаем $n$:

$$n = 14 \cdot L_n / (\pi \cdot D).$$

Измерим диаметр колеса. Это значение будем хранить в константе WHEEL_DIAMETER (в нашем случае 6,67 см) и, используя полученную формулу, напишем функцию.

### Задание 44

На основе созданного примера для библиотеки напишите тестовую программу, чтобы в результате робот точно проезжал указанное расстояние. Проведите серию экспериментов. При необходимости внесите коррективы в вычисления. Сделайте поправку на торможение.

```
#include <robotMove.h>
#define LED_PIN      13
#define BUTTON_PIN    8
#define ENCODER_M1    2
#define ENCODER_M2    3
```

```
#define START_SPEED  50
const float WHEEL_DIAMETER =  6.67;
float spokes (float distance) {
  return 14*distance/(PI*WHEEL_DIAMETER);
}
void setup() {
 encodersConnect (ENCODER_M1,ENCODER_M2);
 buttonConnect (BUTTON_PIN);
 motorsConnect (START_SPEED);
 piK (26,0.2);
 while(digitalRead(BUTTON_PIN)==false)
  digitalWrite(LED_PIN, millis()/1000%3 ? true: false);
}
void loop() {
  int delayTime =3000;
  for (int i=1;i<=3;i++) {
   moveForward(100,spokes(30)); delay (delayTime);
  }
  while (true);
}
```

## Задание 45

Реализуйте алгоритм, представленный на рисунке 75. Подберите параметры, чтобы ваш робот выполнял задание лучше, чем в видеофрагменте по ссылке: https://youtu.be/iK4rIorx8_A.

www

```
void loop() {
  int delayTime=300, delayTime2=500;
  moveForward(120,spokes(50)); delay (delayTime);
  moveBack(120,spokes(45)); delay (delayTime);
  turnLeft (90,3); delay (delayTime2);
  for (int i=1;i<=7;i++) {
   moveForward(120,spokes(42)); delay (delayTime);
   moveBack(120,spokes(43)); delay (delayTime);
   turnLeft (90,3);delay (delayTime2);
  }
  while (true);
}
```

## Задание 46

Реализуйте алгоритм, представленный на рисунке 85.

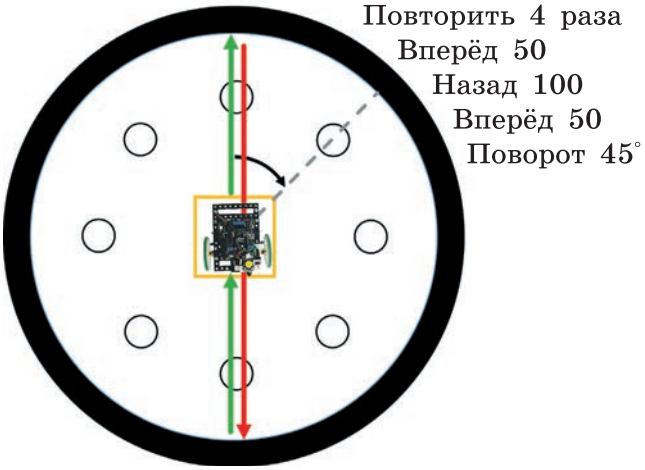



**Рис. 85.** Схема выполнения задания

## Задание 47

Реализуйте алгоритм «Треугольник», представленный на рисунке 86. Видео с выполнением задания роботом: https://youtu.be/mkGL6Jb2w1c.

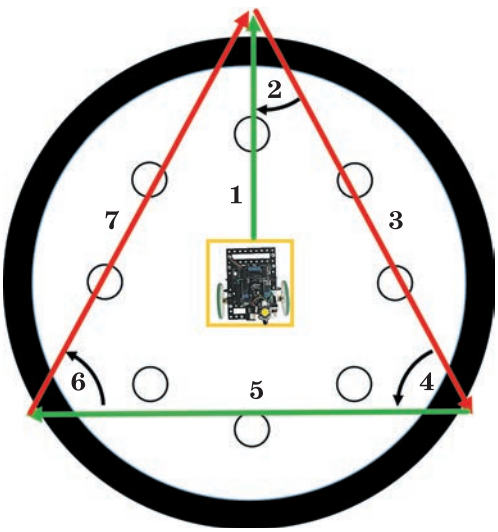



**Рис. 86.** Более быстрый алгоритм «Треугольник»

Обратите внимание, что робот должен не один раз выполнить указанные задания, а иметь хорошую повторяемость действий.

## 9.4. Движение по спирали

Есть ещё один алгоритм выполнения задания (рис. 87). Движение по спирали кажется таким сложным. По какой формуле всё это происходит? Может, нужны сложные математические расчёты? На самом деле это самый простой алгоритм из всех приведённых. В нём даже не нужно направлять робота, чтобы он ехал прямо, нет необходимости писать библиотеку. Здесь и программа небольшая.

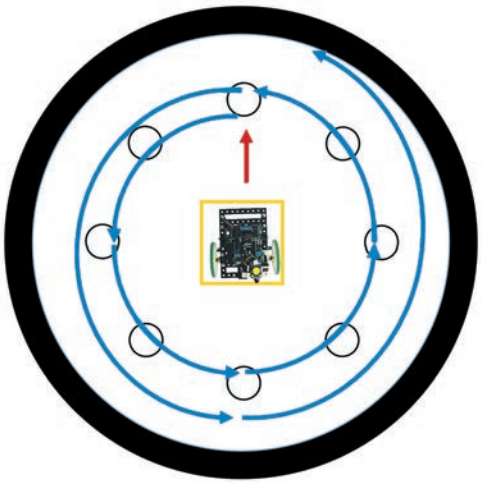


**Рис. 87.** Алгоритм «Спираль»

Если у первого мотора будет скорость вращения оси постоянна, а у второго мотора скорость увеличивается, то траектория движения робота будет похожа на спираль.

Прежде всего для конкретного робота нужно узнать, какое соотношение скоростей обеспечивает движение по кругу, диаметр которого равен 1 метру. Чтобы не ходить много раз от компьютера к кегельрингу, можно использовать цикл, в котором скорость второго мотора изменяется, причём по нажатию на датчик касания.

```
#define LED_PIN      13
#define BUTTON_PIN    8
#define DIR_M1        4 //правый мотор
#define SPEED_M1      5
#define SPEED_M2      6
#define DIR_M2        7 //левый мотор
void setup() {
 for (int i=4;i<8;i++) pinMode(i,OUTPUT);
```

```
 while(digitalRead(BUTTON_PIN)==false)
  digitalWrite(LED_PIN,millis()/1000%3 ? true:false);
}
void loop() {
  digitalWrite (DIR_M1,HIGH);
  digitalWrite (DIR_M2,HIGH);
  for (int i = 0;i<=4;i++) {
   analogWrite(SPEED_M1,200);
   analogWrite(SPEED_M2,80+i*10);   delay(7000);
   analogWrite(SPEED_M1,0); analogWrite(SPEED_M2,0);
   while(digitalRead(BUTTON_PIN)==false)
   digitalWrite(LED_PIN,millis()/1000%3 ? true:false);
  }
  while(true);  }
```

## Задание 48

**Экспериментально определите скорость второго мотора вашего робота при фиксированной скорости первого, равной 200 оборотам в минуту (или 250).**

**Самая простейшая спираль легко реализуется обычным циклом, мы даже для небольшого её усложнения добавим зависимость задержки от текущей скорости.**

```
 digitalWrite (DIR_M1,HIGH);
 digitalWrite (DIR_M2,HIGH);
  for (int i = 1;i<=20;i++) {
   analogWrite(SPEED_M1,200);
   analogWrite(SPEED_M2,20+i*4);
   delay(100*i);
  }
  analogWrite(SPEED_M1,0); analogWrite(SPEED_M2,0);
```

## Задание 49

**Исследуйте траекторию движения вашего робота по приведённой ранее программе.**

Спираль получается, конечно, красивая, но кегли робот не сбивает. Ещё раз внимательно посмотрите на схему алгоритма «Спираль». Хоть его так и называют, но это не совсем спираль. Там два участка (рис. 88). На первом робот должен быстро повернуться, а вот потом траектория становится похожа на спираль. Кроме того, необходимо запланировать контрольный проезд по кругу, чтобы довыбивать кегли, а также, используя крепёж для заднего бампера, закрепить его спереди под углом.

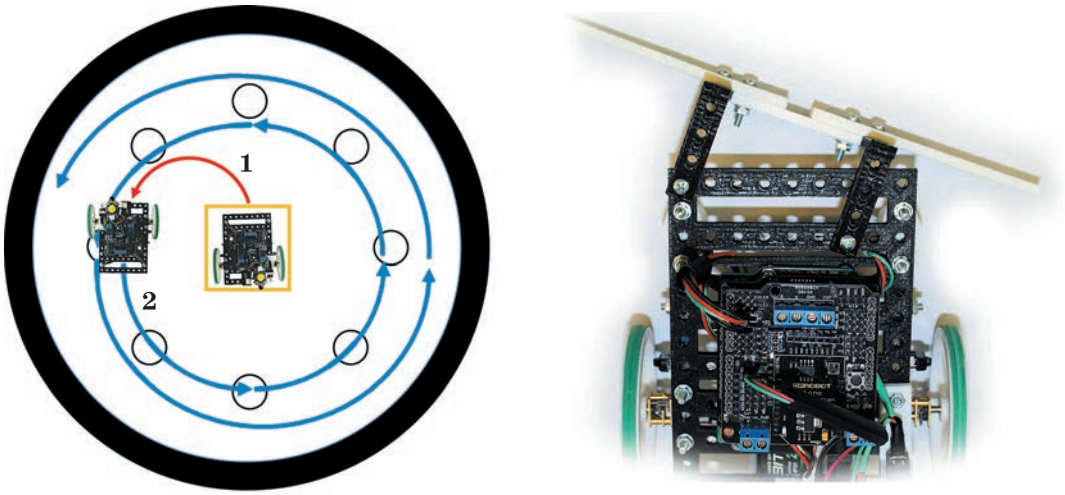



**Рис. 88.** Уточнённая схема алгоритма «Спираль»

## Задание 50

Реализуйте алгоритм «Спираль», представленный на рисунке 88. Видео с примером выполнения задания можно посмотреть по ссылке: https://youtu.be/ya8_1zzTjfQ.

```
#define LED_PIN      13
#define BUTTON_PIN    8
#define DIR_M1        4 //правый мотор
#define SPEED_M1      5
#define SPEED_M2      6
#define DIR_M2        7 //левый мотор
//мощности моторов для движения
//по кругу радиуса 1 м
const int POWER_M2 = 83;
const int POWER_M1 = 250;
void setup() {
```

```
 for (int i=4;i<8;i++) pinMode(i,OUTPUT);
 while(digitalRead(BUTTON_PIN)==false)
  digitalWrite(LED_PIN, millis()/1000%3 ? true: false);
}
void loop() {
 //первый участок: изменяем мощность
 //второго мотора от 20 до 55
 float timeDelta = 10;
 digitalWrite (DIR_M1,HIGH);  digitalWrite (DIR_M2,HIGH);
 for (int motorPowerM2 = 20; motorPowerM2 < 55 ;
motorPowerM2++) {
  timeDelta=timeDelta+motorPowerM2*0.5;
  analogWrite(SPEED_M1,POWER_M1);
  analogWrite(SPEED_M2,motorPowerM2);
  delay(60);
 }
 //второй участок: медленно изменяем мощность
 //мотора от 67 до POWER_M2
 analogWrite(SPEED_M1,250);
 for (int i=0; i<=16; i+=5) {
  analogWrite(SPEED_M2,67+i); delay(1500);
 }
 //контрольный проезд, довыбиваем кегли
 analogWrite(SPEED_M2,POWER_M2); delay (13000);
 //остановка, ждём нового заезда
 analogWrite(SPEED_M1,0); analogWrite(SPEED_M2,0);
 while(digitalRead(BUTTON_PIN)==false)
 digitalWrite(LED_PIN, millis()/1000%3 ? true: false);
}
```

### Задание 51

**Замените первый участок простым поворотом с разными мощностями моторов. Оптимизируйте код программы.**

**Конечно, в данном случае алгоритм «Спираль» получился не самым быстрым. Самый быстрый алгоритм — «Треугольник». Однако если поставить моторы со скоростью 500 оборотов в минуту и более, то именно этот алгоритм становится самым быстрым. По крайней мере, за 2,5 секунды, используя алгоритм движения по спирали, робот может выбить все восемь кеглей.**

# Глава 10
# ОБНАРУЖЕНИЕ ОБЪЕКТА

## 10.1. Ультразвуковой дальномер

### Задание 52

Установите ультразвуковой датчик расстояния и подключите его к плате расширения (рис. 89). Питание и землю подключите к соответствующим контактам, *Trig* к контакту 10 (с маркировкой *D*), *Echo* к контакту 11 (также с маркировкой *D*).

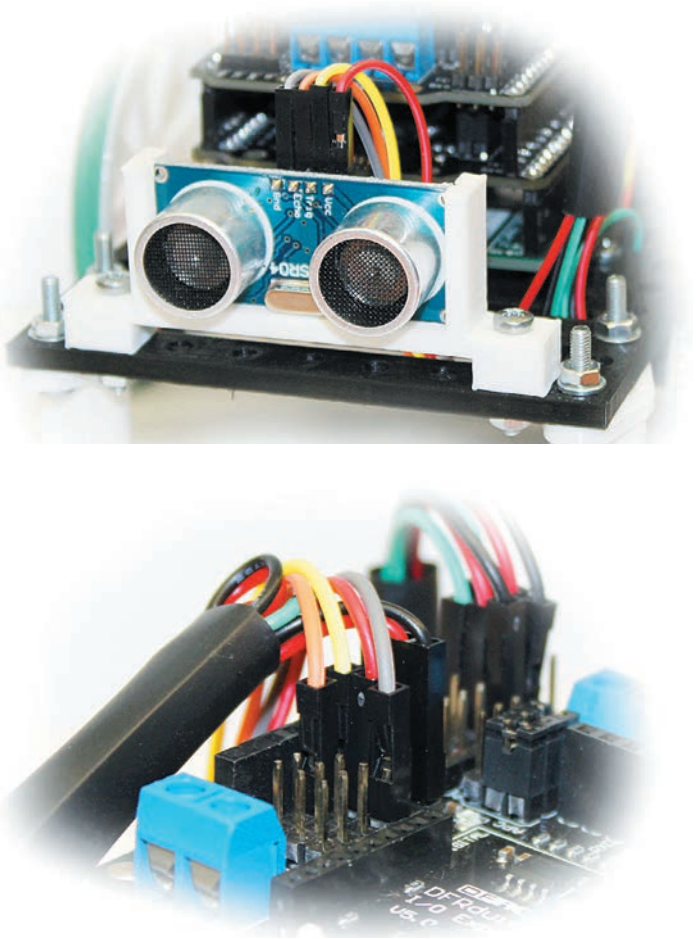


**Рис. 89.** Крепление датчика расстояния и его подключение

Дальномер — это устройство для измерения расстояния до некоторого предмета.

Звуковыми волнами или просто звуком принято называть волны, воспринимаемые человеческим ухом. Волны с частотой более 20 кГц называют ультразвуком. На плате модуля HC-SR04 размещены пьезоизлучатель ультразвука, работающий на частоте 40 кГц, и микрофон, воспринимающий отражённую звуковую волну. То есть датчик генерирует звуковые импульсы и слушает эхо. По времени распространения звуковой волны до объекта и обратно можно однозначно определить расстояние до него.

Ток потребления в рабочем режиме 15 мА, следовательно, датчик можно подключать непосредственно к плате Arduino, у которой каждый контакт поддерживает ток в 40 мА. Диапазон измеряемых расстояний: 2–400 см.


**Рис. 90.** Датчик расстояния HC-SR04

Датчик HC-SR04 имеет 4 вывода (рис. 90):

- *Vcc* — на него необходимо подать питание от платы;
- *Trig* — контакт для триггера, чтобы послать звуковые импульсы — быстро включить/выключить;
- *Echo* — выходной сигнал, по которому определяют, получен ли обратный импульс;
- *Gnd* — соединяется с соответствующим контактом платы (земля).

## Задание 53

Изучите технический лист на датчик HC-SR04 по указанной ссылке: https://lib.chipdip.ru/092/DOC001092302.pdf. Найдите информацию, сколько импульсов посылает передатчик за один раз и какова продолжительность подачи сигнала HIGH на триггер.

В справочниках можно найти значение скорости звука в воздухе при +20 °C — 343,1 м/с, следовательно, чтобы узнать расстояние до объекта, необходимо знать время, за которое звуковая волна до него доходит (или вернётся после излучения, разделить это время на 2).

В Arduino IDE есть возможность использовать специальную функцию *pulseIn*.

```
pulseIn(контакт, или HIGH, или LOW);
```

Этой функции нужно указать номер пина и один из двух уровней (HIGH/LOW), который должен появиться на указанном контакте в течение 1 секунды. Если такой сигнал появится, то функция *pulseIn* выдаст время в микросекундах, если не появится — 0.

То есть, отправив сигнал на ультразвуковом передатчике, мы вызываем эту функцию, указывая ей номер контакта *Echo* и уровень HIGH, и в результате получим время, за которое сигнал доходит до объекта и, отразившись от него, возвращается.

## 10.2. Определение расстояния

### Задание 54

Составьте программу, чтобы робот передавал на монитор последовательного порта расстояние в сантиметрах до объекта.

```
#define ECHO_PIN  11
#define TRIG_PIN  10
const long speedOfSound = 34310;
// функция определения расстояния до объекта в см
float distance(){
  digitalWrite(TRIG_PIN, LOW);//выключаем триггер
  delayMicroseconds(1); //ждём 1 мкс
  digitalWrite(TRIG_PIN, HIGH); //включаем триггер
  delayMicroseconds(10); //ждём 10 мкс
  digitalWrite(TRIG_PIN, LOW); //выключаем триггер
  //ждём, через сколько мкс сигнал вернётся обратно
  unsigned long timeBack = pulseIn(ECHO_PIN, HIGH);
  Serial.println(timeBack);   //временно, для тестирования
  float seconds = (timeBack+10.0)/1000/1000/2;
                                //переводим в секунды
  return seconds*speedOfSound; //находим расстояние
}
void setup() {
  Serial.begin(9600);
  pinMode(ECHO_PIN, INPUT);
  pinMode(TRIG_PIN, OUTPUT);
}
void loop(){
```

```
    Serial.print(distance());
    Serial.println("cm");
    //delay(10);
  }
```

### Задание 55

Проведите серию измерений. Запишите результаты в электронную таблицу. Найдите величину ошибки при разных измерениях. Подумайте, как можно скорректировать ошибку.

## 10.3. Скорость звука зависит от температуры

Справочник — это, конечно, хороший источник информации, но скорость звука в воздухе (и в газах вообще) зависит от температуры. Открываем любой курс термодинамики...

Скорость звука в воздухе находится по следующей формуле:

$$v = \sqrt{\frac{\gamma \cdot R \cdot T}{M}},$$

где $\gamma$ — показатель адиабаты: $\frac{5}{3}$ для одноатомных газов, $\frac{7}{5}$ для двухатомных (и для воздуха), $\frac{4}{3}$ для многоатомных; $T$ — абсолютная температура; $M$ — молярная масса (для воздуха $29 \cdot 10^{-3}$ кг/моль); $R$ — универсальная газовая постоянная (8,3144598 Дж/моль).

Также при написании программы учтём, что в рекомендациях производителя дальномера не советуется делать паузы хотя бы в 50 миллисекунд между измерениями.

### Задание 56

Составьте программу, чтобы робот передавал на монитор последовательного порта расстояние в сантиметрах до объекта, учитывая температуру воздуха.

```
#define ECHO_PIN  11
#define TRIG_PIN  10
#define TEMP      20
float oldDistance;//для хранения предыдущего измерения
float distance(float t){
  digitalWrite(TRIG_PIN, LOW);//выключаем триггер
  delayMicroseconds(1); //ждём 1 мкс
  digitalWrite(TRIG_PIN, HIGH); //включаем триггер
```

```
  delayMicroseconds(10); //ждём 10 мкс
  digitalWrite(TRIG_PIN, LOW); //выключаем триггер
  //ждём, через сколько мкс сигнал вернётся обратно
  unsigned long timeBack = pulseIn(ECHO_PIN, HIGH);
  //вычисляем в секундах
  float seconds = (timeBack+10.0)/1000/1000/2;
  float speedOfSound =
     sqrt(8.3145*1.4*(273+ TEMP)/0.029)*100;
  return seconds*speedOfSound; //находим расстояние
}
void setup() {
  Serial.begin(9600);
  pinMode(ECHO_PIN, INPUT);
  pinMode(TRIG_PIN, OUTPUT);
  oldDistance = distance(TEMP);
}
void loop(){
//используем тернарный условный оператор, чтобы вызывать
функцию не чаще 1 раза в 100 мс
  float nowDistance =
    millis()%100==0 ? distance(TEMP) : oldDistance;
  Serial.print(nowDistance); Serial.println("cm");
  //delay(100);
  oldDistance = nowDistance;
}
```

### Задание 57

Проведите серию измерений. Запишите результаты в электронную таблицу. Найдите величину ошибки при разных измерениях. Сравните точность измерений обоих алгоритмов и выберите лучший, чтобы использовать в дальнейшем.

## 10.4. И снова кегельринг

### Задание 58

Робот медленно вращается на одном месте. При обнаружении объекта в радиусе 25 см останавливается и «смотрит» на объект. Если объект «уходит», то продолжает вращение. Видео выполнения задания роботом находится по ссылке: https://youtu.be/RNb7oghQqjc.

```
#include <robotMove.h>
#define ECHO_PIN     11
#define TRIG_PIN     10
#define TEMP         20
#define LED_PIN      13
#define BUTTON_PIN    8
#define ENCODER_M1    2
#define ENCODER_M2    3
#define START_SPEED  50
float oldDistance;

float distance(float t){
  digitalWrite(TRIG_PIN, LOW);//выключаем триггер
  delayMicroseconds(1); //ждём 1 мкс
  digitalWrite(TRIG_PIN, HIGH); //включаем триггер
  delayMicroseconds(10); //ждём 10 мкс
  digitalWrite(TRIG_PIN, LOW); //выключаем триггер
  //ждём, через сколько мкс сигнал вернётся обратно
  unsigned long timeBack = pulseIn(ECHO_PIN, HIGH);
  float seconds = (timeBack+10.0)/1000/1000/2;
  float speedOfSound =sqrt(8.314*1.4*(273+t)/0.029)*100;
  return seconds*speedOfSound; //находим расстояние
}

void turn (int turnSpeed) {
  digitalWrite(4,HIGH);digitalWrite(7,LOW);
  analogWrite(5,turnSpeed);
  analogWrite(6,turnSpeed);
}

void stopMove () {
  digitalWrite(4,HIGH);digitalWrite(7,LOW);
  analogWrite(5,0); analogWrite(6,0);
}

void setup() {
  pinMode(ECHO_PIN,INPUT); pinMode(TRIG_PIN,OUTPUT);
  oldDistance = distance(TEMP);
  encodersConnect (ENCODER_M1,ENCODER_M2);
  buttonConnect (BUTTON_PIN);
```

```
  motorsConnect (START_SPEED);
  piK (26,0.2);
  while(digitalRead(BUTTON_PIN)==false)
  digitalWrite(LED_PIN, millis()/1000%3 ? true: false);
}

void loop(){
  float nowDistance =
    millis()%50==0 ? distance(TEMP) : oldDistance;
  if (nowDistance > 26)  turn (100);
  else stopMove();
  oldDistance = nowDistance;
}
```

## Задание 59

Робот в центре. При этом кеглей не восемь, а всего четыре, но они могут располагаться в любом из восьми указанных мест (например, как на рисунке 92). Он должен все кегли убрать за пределы белого круга. Установите передний бампер (рис. 91) и, используя созданную ранее библиотеку с точными перемещениями робота, выполните задание.

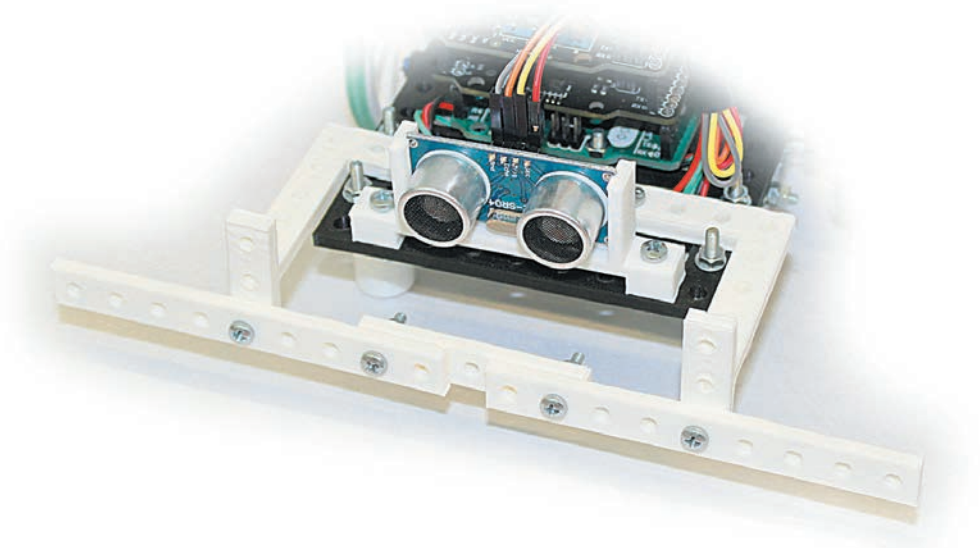


**Рис. 91.** Датчик расстояния HC-SR04 и бампер для выбивания кеглей

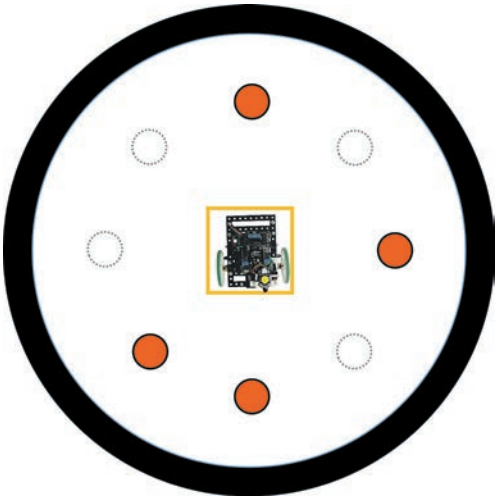


**Рис. 92.** Схема задания

После того как робот уберёт все четыре кегли, он должен остановиться и быть готовым выполнить задание ещё раз.

## Задание 60

Робот в центре. Кеглей может быть любое количество, хоть 100. Задача робота очистить территорию, ограждённую чёрной линией. После очистки необходимо остановиться.

# Глава 11
# ДВИЖЕНИЕ ПО ЛИНИИ

## 11.1. Революция в автоматизации логистики

На Всемирном экономическом форуме в Давосе в 2016 году был представлен доклад о четвёртой промышленной революции — массовом внедрении киберфизических систем в производство, обслуживание человеческих потребностей, включая быт, труд и досуг. Важнейшие области: искусственный интеллект, робототехника, интернет вещей, автоматические транспортные средства, трёхмерная печать, квантовые компьютеры и нанотехнологии.

Использование киберфизических систем (роботов) стало достаточно привычным. Например, компания Foxconn, производящая электронику для Apple, Hewlett-Packard, Dell, Sony и т. д., ещё в 2011 году начала процесс замены 1,2 млн работников на 1 млн роботов.

Перемещение грузов — одно из постоянных занятий человечества. Современное складирование и распределение, также называемое логистикой, переживает сложные процессы автоматизации. Например, известная компания Amazon уже постоянно использует 45 тыс. мобильных роботов, которые максимально оптимизируют складские операции (рис. 93), притом что постоянных сотрудников в компании около 120 тыс.


**Рис. 93.** Роботы компании Amazon

### Задание 61

Посмотрите видео работы современных склада и почты по указанным ссылкам: https://youtu.be/3eQAFVetNGI, https://youtu.be/_QndP_PCRSw. Есть также и российские разработки резидентов Сколково (https://youtu.be/dMZp4G3X_2Q).

## 11.2. Движение вдоль линии

Конечно, роботы в виде автоматических тележек в логистике используются достаточно давно, и если вы посмотрите подборку примеров их работы, то обратите внимание, что на полу множество меток. Это или контрастные линии (рис. 94), или радиометки (см. рис. 93).

### Задание 62

Просмотрите подборку работы логистических мобильных роботов (https://youtu.be/tBN0B1MylPM).

В основе — не очень сложное движение по линии. Эта задача уже стала считаться типовой, причём в различной литературе роботов часто так и называют — «тележка».


**Рис. 94.** Движение роботов вдоль линии

Чтобы наш робот мог двигаться вдоль линии, необходимо понять, как работают датчики, и изучить алгоритмы регулирования.

## 11.3. Оптопара TCRT5000

Мы тоже научимся программировать нашего робота так, чтобы он максимально быстро передвигался по маршруту, указанному чёрной линией на светлом поле. Для этого будем использовать уже знакомые вам инфракрасные датчики, которые определяют разницу в контрасте линии и окружающего фона, — TCRT5000.

Основой датчика является оптопара TCRT5000 — электронный прибор, состоящий из излучателя света и фотоприёмника, связанных оптическим каналом и объединённых в общем корпусе. Принцип работы заключается в преобразовании электрического сигнала в свет, его передаче по оптическому каналу и последующем преобразовании обратно в электрический сигнал.

В нашем случае (рис. 95) излучатель — инфракрасный светодиод, который закрыт синим колпаком, а фотоприёмник — инфракрасный фототранзистор, закрытый чёрным колпаком. Когда инфракрасное излучение, испускаемое светодиодом, отражается от поверхности и возвращается на чёрное окно фототранзистора, возникает ток проводимости. Чем больше излучения попадает на базу фототранзистора, тем больше будет его ток проводимости.

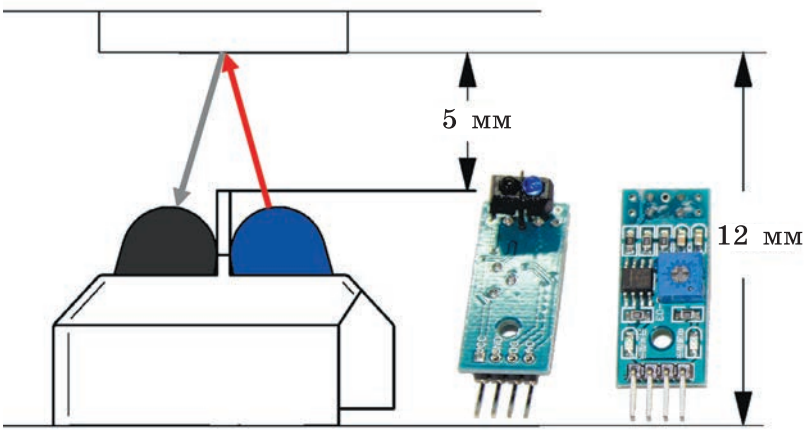



**Рис. 95.** Датчик на основе оптопары

Передаёт датчик уровень напряжения (от 0 до 5 В), при этом имеет 1024 градации этого уровня. Таким образом, мы можем получать с датчика информацию о 1024 градациях отражённого сигнала, для этого имеется выход с маркировкой «AO».

## 11.4. Трасса и установка датчиков линии

### Задание 63

Установите четыре датчика линии на основе TCRT5000, как показано на рисунке 96, и подключите к аналоговым входам A0–A3 на плате расширения вводов/выводов. У каждого датчика контакт «A0» соедините с контактом «S» платы, GND с GND (земля к земле) и VCC с V (VCC). Если подключили всё правильно, то при подаче питания на плату Arduino будет светиться красный светодиод, а если датчик находится над белым полем, то ещё и зелёный.

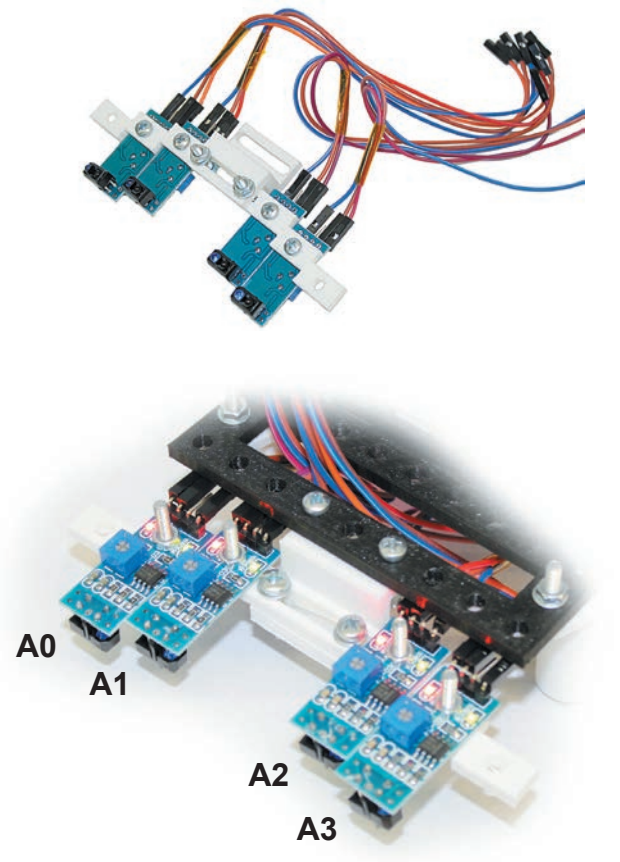



**Рис. 96.** Крепление датчиков линии

Пусть у нас имеется линия шириной 2 см, изображённая на рисунке 97. Общая идея движения вдоль линии выглядит примерно так: робот едет с какой-то заданной скоростью, левый датчик управляет изменением скорости левого мотора, а правый — правого. Если датчик приближается к чёрной линии, то скорость мотора уменьшается, если отдаляется от чёрной линии — увеличивается.

Датчики линии будут выдавать значения от 0 до 1023, однако мы не знаем реальный диапазон после установки датчиков. Кроме того, на белом поле датчики выдают близкие к 0 значения, а на чёрном — близкие к 1023. Прежде всего необходимо узнать реальный диапазон каждого датчика и преобразовать его в интервал 0–255, так как это соответствует мощности моторов (при управлении питанием через ШИМ). Как вы поняли, нужно использовать для движения по линии информацию со всех четырёх датчиков, поэтому откалибруем показания каждого.

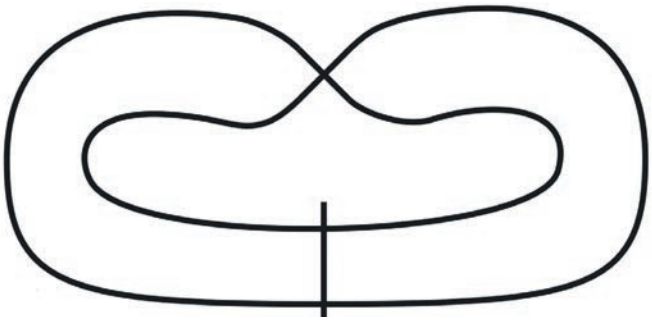



**Рис. 97.** Траектория движения робота вдоль чёрной линии

## Задание 64

Напишите программу вывода информации со всех датчиков линии TCRT5000 (рис. 98) на монитор последовательного порта. Отметьте, какие показания выдаёт каждый датчик на чёрной линии и на белом поле. Запишите значения в комментариях, чтобы не потерять их. Также важно понимать, что при изменении освещённости в помещении показания датчиков также изменятся.

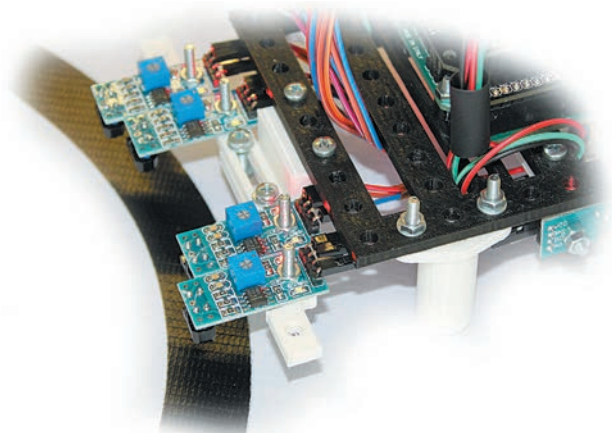


**Рис. 98.** Расположение робота при измерении показаний

```
#define LINE_SENSOR_M1_1   A0
#define LINE_SENSOR_M1_2   A1
#define LINE_SENSOR_M2_1   A2
#define LINE_SENSOR_M2_2   A3
#define LED_PIN            13
#define BUTTON_PIN          8
int sensorValueM1_1, sensorValueM1_2;
int sensorValueM2_1, sensorValueM2_2;
void sensorRead() {
 sensorValueM1_1 = analogRead(LINE_SENSOR_M1_1);
 sensorValueM1_2 = analogRead(LINE_SENSOR_M1_2);
 sensorValueM2_1 = analogRead(LINE_SENSOR_M2_1);
 sensorValueM2_2 = analogRead(LINE_SENSOR_M2_2);
}
void setup() {
 pinMode(BUTTON_PIN, INPUT); pinMode(LED_PIN,OUTPUT);
 Serial.begin (9600);
 while(digitalRead(BUTTON_PIN)==false)
  digitalWrite(LED_PIN, millis()/1000%3 ? true: false);
}
void loop() {
 sensorRead();
 Serial.print ("A0=");
 Serial.print (sensorValueM1_1);Serial.print ("; ");
 Serial.print ("A1=");
 Serial.print (sensorValueM1_2);Serial.print ("; ");
 Serial.print ("A2=");
 Serial.print (sensorValueM2_1);Serial.print ("; ");
 Serial.print ("A3=");
 Serial.print (sensorValueM2_2);Serial.println ("; ");
 delay(300);
}
```

**Например, после тестирования выяснилось, что на чёрной линии и белом поле датчики выдают следующие значения: A0 — 34 и 622, A1 — 33 и 694, A2 — 42 и 654, A3 — 38 и 761 (рис. 99).**

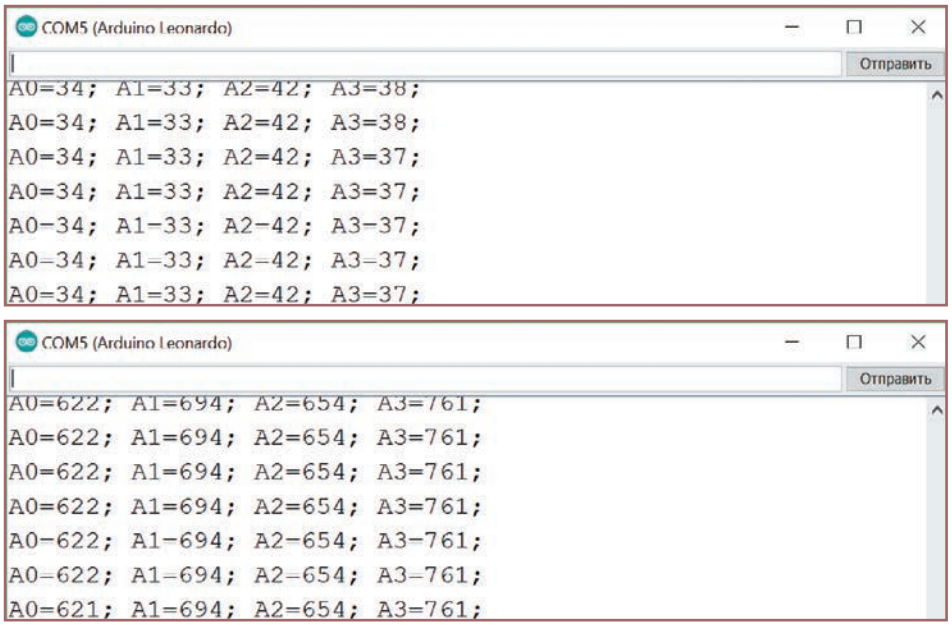



**Рис. 99.** Показания датчиков на белом поле и чёрной линии

Тогда функцию чтения с датчиков мы преобразуем командами *map* и *constrain*, и на выходе будут значения от 0 до 255, соответствующие возможной мощности моторов.

```
void sensorRead() {
 sensorValueM1_1 = constrain
(map(analogRead(LINE_SENSOR_M1_1),34,623,255,0),0,255);
 sensorValueM1_2 = constrain
(map(analogRead(LINE_SENSOR_M1_2),33,694,255,0),0,255);
 sensorValueM2_1 = constrain
(map(analogRead(LINE_SENSOR_M2_1),42,654,255,0),0,255);
 sensorValueM2_2 = constrain
(map(analogRead(LINE_SENSOR_M2_2),37,761,255,0),0,255);
}
```

## Задание 65

Напишите программу, преобразующую показания датчиков линии в строгий диапазон от 0 до 255 (рис. 100). Протестируйте. При необходимости внесите изменения.

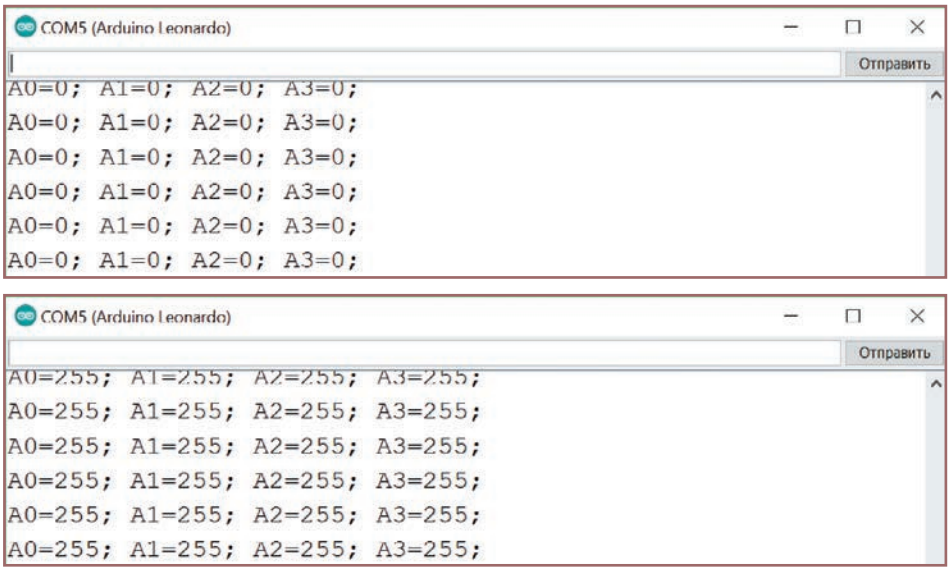



**Рис. 100.** Обработанные показания датчиков

## 11.5. Регуляторы

*Регулятор* — устройство, которое следит за состоянием некоторых параметров объекта управления (как системы) и, применяя определённые алгоритмы, вырабатывает для него управляющие сигналы.

Связь входа системы с её выходом называется *обратной связью*. Если изменение выходного сигнала призваны компенсировать внешние возмущения, то такая связь называется *отрицательной*. Регуляторы в подавляющем большинстве работают по принципу отрицательной обратной связи. Система регулирования может быть представлена рядом элементов, выполняющих определённые функции (рис. 101).

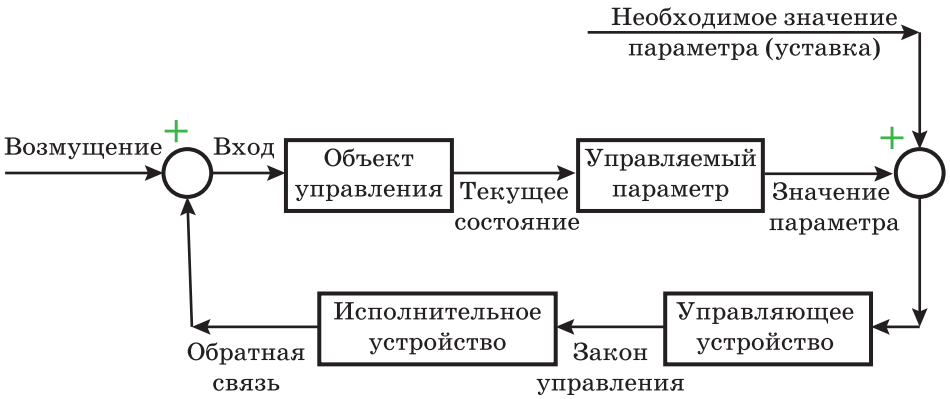



**Рис. 101.** Схема работы регулятора с обратной связью

Программно-аппаратные средства, обеспечивающие поддержание заданного значения параметра системы или изменение его по определённому закону, называются *автоматическими регуляторами*. *Закон регулирования* — это математическая зависимость, с помощью которой определяется регулирующее воздействие по сигналу рассогласования (ошибке). Автоматический регулятор состоит из трёх основных частей: измерительной, преобразовательной и исполнительной.

Современные управляющие устройства основываются на микроконтроллерах. Принцип работы управляющего микроконтроллера состоит в выполнении следующего непрерывного цикла операций:

1) сбор сигналов с датчиков (в нашем случае с TCRT5000);
2) обработка сигналов согласно алгоритму (закону) управления;
3) выдача управляющих воздействий на исполнительные устройства (в нашем случае на моторы посредством ШИМ).

## 11.6. ПИД-регулятор

Итак, закон регулирования — это зависимость, с помощью которой определяется по сигналу рассогласования регулирующее воздействие как функция от времени (т. е. по ошибке как функции от времени). Около 90% всех автоматических регуляторов используют закон регулирования, который называется пропорционально-интегрально-дифференциальным законом, а сам регулятор, его использующий, — *пропорционально-интегрально-дифференциальным (ПИД) регулятором*. Давайте разберёмся, что это за закон. Из названия уже видно, что используются три составляющие, сумма которых образует управляющее воздействие (управляющий сигнал).

*Пропорциональная составляющая* вырабатывает выходной сигнал, который противодействует отклонению (наблюдаемому в данный момент времени) регулируемой величины от заданного значения и пропорционален этому отклонению. Сигнал тем больше, чем больше ошибка. Если входной сигнал равен уставке, то выходной равен нулю.

С точки зрения математической формулировки пропорциональная составляющая выглядит следующим образом:

$$U_p = K_p \cdot e,$$

где $U_p$ — управляющий сигнал (воздействие); $e$ — ошибка в данный момент времени; $K_p$ — усиливающий пропорциональный коэффициент ($K_p > 1$).

Как выглядит реализация этого пропорционального закона средствами языка C++

```
//инициализация необходимых переменных
int setPoint; //начальное значение (уставка)
int actualValue; //текущее значение
```

```
int error = 0;//ошибка или рассогласование
//коэффициент пропорциональной составляющей
float Kp = ...;//указать коэффициент
//интегральная составляющая
float proportionalTerm = 0;
...
void loop() {
   ...
   error = setPoint – actualValue;
   proportionalTerm = Ki*error;
   delay(10);
}
```

Вспомните, как вы настраивали программу управления роботом, чтобы он ехал прямо. Знакомый подход?

*Интегральная составляющая* учитывает сумму всех предыдущих ошибок, как будто регулятор «учится» на своём опыте, путём постоянного суммирования ошибок и формирования сигнала управления, пропорционального полученной сумме:

$$U_i = K_i \cdot S_e,$$

где $U_i$ — управляющий сигнал (воздействие); $S_e$ — сумма ошибок к данному моменту времени; $K_i$ — интегральный коэффициент, при этом или $K_i > 1$, или $0 < K_i < 1$.

Управлять процессом будет микроконтроллер, а следовательно, закон управления мы пишем конкретно для него в форме вычислительного алгоритма:

```
//инициализация необходимых переменных
int setPoint; //начальное значение (уставка)
int actualValue; //текущее значение
int error = 0, errorSum = 0;//ошибка и сумма
//коэффициент интегральной составляющей
float Ki = ...;//указать коэффициент
//интегральная составляющая
float integralTerm = 0;
...
void loop() {
    ...
    error = setPoint – actualValue;
    errorSum = errorSum + error;
    integralTerm = Ki*errorSum;
    ...
    delay(10);
}
```

А что вообще может происходить с ошибкой? Ошибка — это функция от времени, т. е. $e(t)$. Вспомните протекание различных процессов, которые изучали в физике, и графики, знакомые вам из уроков алгебры. График функции $e(t)$ может иметь любой вид, например такой, как на рисунке 102. Значит, в момент времени $t_1$ что-то произошло и причём очень быстро. Ни пропорциональная, ни интегральная составляющая не смогут «спасти» ситуацию. Следовательно, нам нужно отслеживать не только текущую ошибку, но и прошлое её значение, чтобы знать, как она изменяется. При этом чем больше изменение ошибки, тем больше управляющее воздействие.

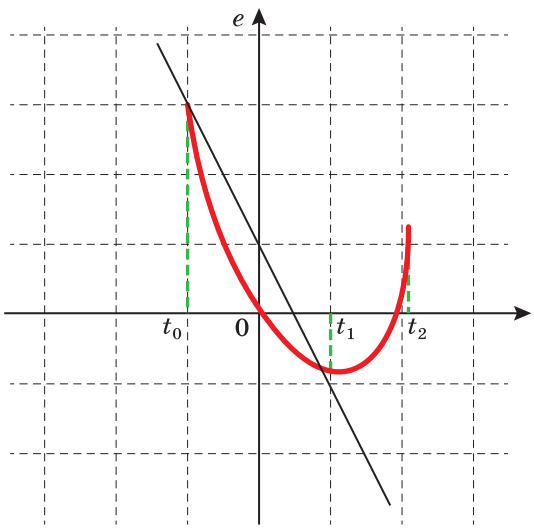



**Рис. 102.** Возможный график функции e(t)

Итак, чем быстрее регулируемая величина отклоняется от уставки, тем сильнее противодействие, создаваемое дифференциальной составляющей $U_d$:

$$U_d = K_d \cdot (e - e_{\text{old}}),$$

где $U_d$ — управляющий сигнал (воздействие); $e$ — текущая ошибка; $e_{\text{old}}$ — ошибка в предыдущий момент времени; $K_d$ — дифференциальный коэффициент, при этом $K_d > 1$,

Опишем это на языке C++:

```
//инициализация необходимых переменных
int setPoint; //начальное значение (уставка)
int actualValue; //текущее значение
int error = 0; //ошибка (рассогласование)
int errorOld = 0;//предыдущее значение ошибки
//коэффициент дифференциальной составляющей
float Kd = ...;//указать коэффициент
//дифференциальная составляющая
float derivativeTerm = 0;
```

```
...
void loop() {
  ...
  error = setPoint – actualValue;
  derivativeTerm = Kd*(error – errorOld);
  ...
  delay(10);
  errorOld = error;
}
```

**Осталось только объединить все три составляющие в единый алгоритм и получить шаблон для реализации ПИД-регулятора.**

```
int setPoint; //начальное значение (уставка)
int actualValue; //текущее значение
int error = 0; //ошибка (рассогласование)
int errorOld = 0;//предыдущее значение ошибки
int errorSum = 0;//сумма ошибок
//коэффициент пропорциональной составляющей
float Kp = ...;//указать коэффициент
//коэффициент интегральной составляющей
float Ki = ...;//указать коэффициент
//коэффициент дифференциальной составляющей
float Kd = ...;//указать коэффициент
//пропорциональная составляющая
float proportionalTerm = 0;
//интегральная составляющая
float integralTerm = 0;
//дифференциальная составляющая
float derivativeTerm = 0;
float resultTerm = 0;
...
void loop() {
  ...
  error = setPoint – actualValue;
  errorSum = errorSum + error;
  proportionalTerm = Kp*error;
  integralTerm     = Ki*errorSum;
  derivativeTerm   = Kd*(error – errorOld);
  resultTerm =
   proportionalTerm + integralTerm + derivativeTerm;
  ...
  delay(10);
  errorOld = error;
}
```

### Задание 66

Изучите реализацию ПИД-регулятора на языке C++.

### Задание 67

Для более глубокого понимания вопроса ПИД-регулирования изучите брошюру В. Э. Карпова «ПИД-управление в нестрогом изложении» (http://robofob.ru/materials/articles/pages/Karpov_mobline1.pdf), в которой в краткой форме излагаются основы построения ПИД-регулятора — одного из основных компонентов автоматики и робототехники.

## 11.7. П-, ПИ-, ПД-регуляторы

Если в ПИД-регуляторе не используется какая-то составляющая, то её убирают из названия. Раньше мы заставили робота ехать прямо, использовав для этого пропорционально-интегральный регулятор (ПИ-регулятор).

## 11.8. Как подбирать коэффициенты

Не спешите сразу ставить робота на трассу. Приподняв его колёса и передвигая вручную на шаровых опорах, вы можете посмотреть, как изменяется мощность моторов.

Нужно очень внимательно следить за тем, что происходит с вашим роботом, когда вы его запускаете, и помнить, за что отвечает каждая составляющая ПИД-регулятора.

Если, например, робота качает из стороны в сторону — это автоколебания, следовательно, пропорциональный коэффициент очень большой. Ведь именно он отвечает за мгновенную реакцию на ошибку.

Если по прямым участкам робот едет плавно, а на повороте слетает, то есть смысл увеличить дифференциальный коэффициент.

Если он закрутился на одном месте — интегральный коэффициент сверхбольшой.

Пусть $K_p$, $K_i$, $K_d$ — коэффициенты соответствующих составляющих, тогда примерный алгоритм действий по подбору коэффициентов выглядит следующим образом.

1. $K_p = 1$; $K_i = 0$; $K_d = 0$. Запускаем робота.
2. Увеличиваем $K_p$ и запускаем робота. Увеличиваем $K_p$ до появления ярко выраженных колебаний робота.
3. Уменьшаем $K_p$ на 50–60% и фиксируем.

4. Увеличиваем $K_d$, пока робот не станет проходить повороты.
5. Устанавливаем $K_i$ = 0,01. Увеличиваем с шагом 0,01, выбирая более плавные движения робота.

Оценив диапазон значений регулируемой величины и диапазон значений ошибки, легко представить диапазон значений пропорционального коэффициента. Таким образом, пункты 1 и 2 вы выполните очень быстро.

## 11.9. Пишем программу и тестируем

Итак, как работают датчики, вы знаете, самый популярный закон регулирования тоже. Понятно, что управляющее воздействие направлено на скорость вращения моторов. Возникает вопрос: какой параметр мы будем контролировать?

Если два датчика (справа от линии и слева от линии) показывают одинаковые значения, то робот должен ехать прямо. Следовательно, и изменять ничего не надо. Значит, в этом случае ошибка (рассогласование) равна нулю. Когда робот должен начать поворачивать в ту или другую сторону? Тогда, когда разность этих показаний стала отличной от нуля. Итак, в качестве контролируемого параметра выберем разность показаний двух датчиков. А уставка — это значение разности в момент старта робота, именно тогда оба датчика находятся на белом поле.

Также вспомним, что конкретный робот может иметь проблемы при движении по прямой. Написав небольшую программу для движения робота по прямой, можно подобрать значения у моторов, при которых робот движется более прямо. Это необходимые нам константы.

```
#define DIR_M1          4
#define SPEED_M1        5
#define SPEED_M2        6
#define DIR_M2          7
#define BUTTON_PIN      8
const int START_POWER_M1 = 220;
const int START_POWER_M2 = 200;
void forward (int velocityM1,int velocityM2, int duration) {
 digitalWrite(DIR_M1,HIGH); digitalWrite(DIR_M2,HIGH);
 analogWrite (SPEED_M1,velocityM1);
 analogWrite (SPEED_M2,velocityM2); delay(duration);
 analogWrite (SPEED_M1,0); analogWrite (SPEED_M2,0);
}
void setup() {
  for (int i=4;i<=7;i=i+1) pinMode(i,OUTPUT);
  pinMode (BUTTON_PIN,INPUT);
}
```

```
void loop() {
  while(digitalRead(BUTTON_PIN)==false);
  forward(START_POWER_M1, START_POWER_M2,2000);
  delay (1000);
}
```

Мы, конечно, универсально настроили чтение данных с датчиков, но в данном случае нам нет необходимости использовать все 256 градаций. Вполне достаточно шкалы от 0 до 100.

## Задание 68

Реализуйте ПИД-управление движением робота по чёрной линии, используя два датчика TCRT5000. Расположив робота, как показано на рисунке 103, исследуйте показания мощности моторов на мониторе последовательного порта, двигая робота вручную. После подтверждения правдоподобности результатов включайте команду управления моторами. Исследуйте влияние величины коэффициентов трёх составляющих регулятора. Пример движения робота по линии, согласно этой программе, приведён по ссылке: https://youtu.be/ZUhijQTDYnw.

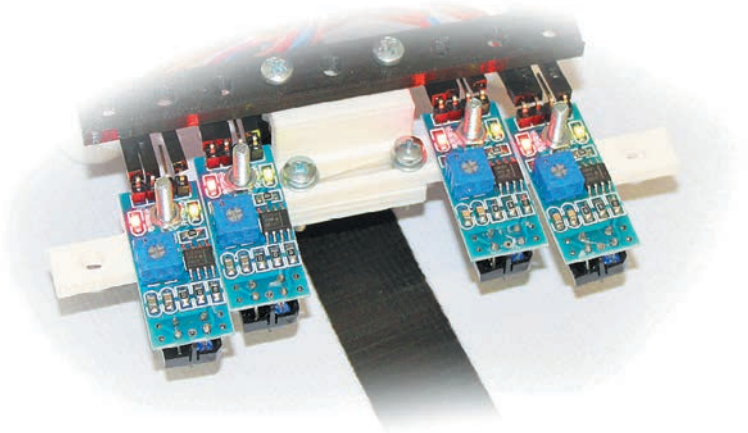


**Рис. 103.** Расположение датчиков при старте

**Программа ПИД-регулирования**

```
#define LED_PIN          13
#define BUTTON_PIN        8
#define DIR_M1            4 //правый мотор
#define SPEED_M1          5
#define SPEED_M2          6
#define DIR_M2            7 //левый мотор
```

```
#define LINE_SENSOR_M1   A1
#define LINE_SENSOR_M2   A2
const int START_POWER_M1 = 220;
const int START_POWER_M2 = 200;
int setPoint; //начальное значение (уставка)
int actualValue; //текущее значение
int error = 0; //ошибка (рассогласование)
int errorOld = 0;//предыдущее значение ошибки
int errorSum = 0;//сумма ошибок
//коэффициент пропорциональной составляющей
float Kp = 2.4;//подобрать коэффициент
//коэффициент интегральной составляющей
float Ki = 0.01;//подобрать коэффициент
//коэффициент дифференциальной составляющей
float Kd = 2.0;//подобрать коэффициент
//результирующее управляющее воздействие
int resultTerm = 0;
int sensorValueM1, sensorValueM2;
int powerM1, powerM2;
bool reverseM1 = false, reverseM2 = false;
//функция управления вращением моторов
//имеет 4 параметра: 2 скорости и 2 реверса
//функция чтения показаний с датчиков
//с преобразованием в диапазоне от 0 до 100
void sensorRead() {
 sensorValueM1 = constrain
(map(analogRead(LINE_SENSOR_M1),33,694,100,0),0,100);
 sensorValueM2 = constrain
(map(analogRead(LINE_SENSOR_M2),42,654,100,0),0,100);
}
void robotGo(int speedM1, int speedM2, bool reverseM1,
             bool reverseM2) {
  analogWrite(SPEED_M1, speedM1);
  analogWrite(SPEED_M2, speedM2);
  digitalWrite(DIR_M2, reverseM2 ? LOW : HIGH);
  digitalWrite(DIR_M1, reverseM1 ? LOW : HIGH);
}
void setup() {
 for (int i=4;i<=7;i++) pinMode(i,OUTPUT);
 pinMode(BUTTON_PIN, INPUT);
 pinMode(LED_PIN,OUTPUT);
 //Serial.begin(9600);
 while(digitalRead(BUTTON_PIN)==false)
```

```
  digitalWrite(LED_PIN, millis()/1000%3 ? true: false);
 sensorRead();
 //уставка — разность в показаниях датчиков
 setPoint = sensorValueM1 — sensorValueM2;
}
void loop() {
 sensorRead();
 actualValue = sensorValueM1 — sensorValueM2;
 error = setPoint — actualValue;
 errorSum = errorSum + error;
 resultTerm =
     Kp*error + Ki*errorSum + Kd*(error — errorOld);
 powerM1 = START_POWER_M1 — resultTerm;
 powerM2 = START_POWER_M2 + resultTerm;
 //проверка, в какую сторону вращать моторы
 //если число отрицательное, то включить реверс
 reverseM1 = powerM1 < 0 ? true : false;
 reverseM2 = powerM2 < 0 ? true : false;
 //проверка превышения максимального значения
 // и возможных отрицательных значений
 powerM1 = abs(powerM1) > 255 ? 255: abs(powerM1);
 powerM2 = abs(powerM2) > 255 ? 255: abs(powerM2);
 // управляем моторами
 robotGo (powerM1, powerM2, reverseM1, reverseM2);

//Serial.print ("powerM1=");
 //Serial.print (powerM1);Serial.print ("; ");
 //Serial.print ("powerM2=");
 //Serial.print (powerM2);Serial.println ("; ");

 delay(10);
 errorOld = error;
}
```

### Задание 69

www

Просмотрите видео с более быстрым движением робота по этой же линии (https://youtu.be/cPLKKBvqqVo). Обратите внимание на быстрые развороты робота в узких местах трассы. Объясните те или иные отличия в конструктиве роботов.

## 11.10. Альфа-бета фильтр

Пока мы использовали только два датчика линии. При медленных моторах этого вполне достаточно, однако если моторы имеют большее количество оборотов, то два датчика не смогут стабилизировать робота на линии.

Рассмотрим, как написать алгоритм ПИД-управления при использовании четырёх датчиков.

Есть соблазн просто объединить пары датчиков и находить суммы показаний, а за ошибку принять разность двух сумм. Этот подход подойдёт, если линия достаточно широкая.

Вы обратили внимание, что крайние датчики чуть выдвинуты? Сделано это для того, чтобы на крутых поворотах сразу два датчика оказывались на чёрной линии.

Когда робот только начинает вылетать с линии, именно тогда крайние датчики оказываются на чёрной линии. А когда робот едет вдоль линии стабильно, крайние датчики всегда на белом.

Тогда есть смысл сделать так, чтобы вклад крайних датчиков в общее управляющее воздействие был большим, чем у средних. В итоге общая ошибка (рассогласование сигналов) будет иметь вид

$$\text{error} = \alpha \cdot \text{error}_{14} + \beta \cdot \text{error}_{23},$$

где $\text{error}_{14}$ — ошибка для крайних датчиков; $\text{error}_{23}$ — ошибка для средних датчиков; $\alpha$ и $\beta$ — коэффициенты, определяющие вклад в общую ошибку. Такой фильтр называется *альфа-бета фильтром*.

Можно попробовать его упрощённый вариант, когда $\alpha + \beta = 1$. Тогда выражение примет вид

$$\text{error} = \alpha \cdot \text{error}_{14} + (1 - \alpha) \cdot \text{error}_{23}.$$

Такой фильтр называется *комплементарным*, а $\alpha$ — коэффициентом комплементарного фильтра.

Коэффициенты $\alpha$ и $\beta$, по сути, являются весовыми коэффициентами, т. е. определяют значимость (вес) вклада каждой из ошибок.

В итоге алгоритм управления роботом получается следующим.

**ПИД-регулятор с применением альфа-бета фильтра**

```
#define LED_PIN           13
#define BUTTON_PIN         8
#define DIR_M1             4 //правый мотор
#define SPEED_M1           5
#define SPEED_M2           6
#define DIR_M2             7 //левый мотор
#define LINE_SENSOR_M1_1   A0
#define LINE_SENSOR_M1_2   A1
#define LINE_SENSOR_M2_1   A2
```

```
#define LINE_SENSOR_M2_2   A3
const int START_POWER_M1 = 240;
const int START_POWER_M2 = 230;
int brake = 3;
int setPoint1, setPoint2;
int actualValue1, actualValue2;
float Kp = 1.1, Ki = 0.01, Kd = 1.4;
float alfa = 0.9; float beta = 2.2;
float resultTerm = 0;
float error, errorSum, errorOld;
int sensorValueM1_1, sensorValueM1_2, powerM1, powerM2;
int sensorValueM2_1, sensorValueM2_2;
bool reverseM1 = false, reverseM2 = false;
void sensorRead() {
 sensorValueM1_1 = constrain
(map(analogRead(LINE_SENSOR_M1_1),34,623,100,0),0,100);
 sensorValueM1_2 = constrain
(map(analogRead(LINE_SENSOR_M1_2),33,694,100,0),0,100);
 sensorValueM2_1 = constrain
(map(analogRead(LINE_SENSOR_M2_1),42,654,100,0),0,100);
 sensorValueM2_2 = constrain
(map(analogRead(LINE_SENSOR_M2_2),37,761,100,0),0,100);
}
void robotGo(int speedM1, int speedM2, bool reverseM1,
             bool reverseM2) {
  analogWrite(SPEED_M1, speedM1);
  analogWrite(SPEED_M2, speedM2);
  digitalWrite(DIR_M2, reverseM2 ? LOW : HIGH);
  digitalWrite(DIR_M1, reverseM1 ? LOW : HIGH);
}
void setup() {
 for (int i=4;i<=7;i++) pinMode(i,OUTPUT);
 pinMode(BUTTON_PIN, INPUT);
 pinMode(LED_PIN,OUTPUT);
 while(digitalRead(BUTTON_PIN)==false)
  digitalWrite(LED_PIN, millis()/1000%3 ? true: false);
 sensorRead();
 setPoint1 = sensorValueM1_1 – sensorValueM2_2;
 setPoint2 = sensorValueM1_2 – sensorValueM2_1;
}
void loop() {
 sensorRead();
 actualValue1 = sensorValueM1_1 – sensorValueM2_2;
```

```
  actualValue2 = sensorValueM1_2 – sensorValueM2_1;
  error = (setPoint1 – actualValue1)*alfa +
          (setPoint2 – actualValue2)*beta;
  errorSum = errorSum + error;
  resultTerm = Kp*error + Ki*errorSum +
               Kd*(error – errorOld);
  powerM1 = (START_POWER_M1 – resultTerm)/brake;
  powerM2 = (START_POWER_M2 + resultTerm)/brake;
  reverseM1 = powerM1 < 0 ? true : false;
  reverseM2 = powerM2 < 0 ? true : false;
  powerM1 = abs(powerM1) > 255 ? 255: abs(powerM1);
  powerM2 = abs(powerM2) > 255 ? 255: abs(powerM2);
  robotGo (powerM1, powerM2, reverseM1, reverseM2);
  delay(10);
  errorOld = error;
}
```

Обратите внимание, что для отладки программы была использована переменная *brake* (тормоз), чтобы искусственно снизить скорость робота и понаблюдать за его поведением. Как проходит трассу робот, можно посмотреть по ссылке: https://youtu.be/uRk2EpymoHY.

www

### Задание 70

Исследуйте работу алгоритма на вашем роботе. Начинайте эксперименты с малыми скоростями робота (большой коэффициент *brake*).

### Задание 71

Измените программу, чтобы в ней использовался комплементарный фильтр. Исследуйте работу нового алгоритма на вашем роботе.

### Задание 72

Подумайте, как можно улучшить плавность прохождения трассы, а также скорость её прохождения.

## 11.11. Снижение скорости на поворотах

Представьте, что вместо робота — автомобиль, и он сейчас войдёт в крутой поворот. Логично, что водитель должен снизить скорость перед поворотом. Вот и идея, как можно улучшить автоматический регулятор.

```
...
float brakeGain = 0.3;
...
 powerM1 =
START_POWER_M1 – brakeGain*resultTerm – resultTerm;
 powerM2 =
START_POWER_M2 – brakeGain*resultTerm + resultTerm;
```

Если робот едет по прямому участку, то скорость не уменьшается, а как только начинаются повороты, скорость немного сбрасывается. Причём чем дольше датчики находятся на чёрной линии, тем медленнее движется робот.

### Задание 73

www

Реализуйте управление роботом с применением снижения скорости на поворотах. Пример прохождения трассы роботом можно посмотреть по ссылке: https://youtu.be/CZz0icsellA.

### Задание 74

Реализуйте движение робота вдоль линии, используя имеющиеся шесть датчиков.

# Глава 12
# ОСНОВЫ ООП

## 12.1. Классы, свойства, методы

*Объектно-ориентированное программирование* (ООП) — это совокупность понятий и идей программирования, которые основаны на работе с объектами, а не просто с данными, функциями и переменными. Каждый объект может включать в себя все необходимые функции, процедуры и параметры. Такой подход облегчает написание, усовершенствование, поиск ошибок и правку программ.

*Класс* — описание ещё не созданного объекта (шаблон). Класс содержит данные о строении объекта и его возможностях, методах работы с ним. *Объект* — экземпляр класса, т. е. он создан по этому шаблону.

Классы состоят из свойств и методов и позволяют создавать новые типы объектов. *Свойства — это данные*, которыми можно характеризовать объект класса. *Методы — функции*, выполняющие действия над свойствами класса.

Если коротко: *свойства класса — это его переменные, а методы класса — это его функции*.

Определение класса выглядит так:

```
class имя_класса { члены класса };
```

где члены класса — это переменные, функции, другие классы.

Все свойства и методы класса имеют права доступа. Для этого существуют модификаторы `private` и `public`.

Функции и переменные, которые находятся после модификатора `public`, доступны из любого места программы и называются *публичными*, а после модификатора `private` — только внутри этого класса и называются *частными*.

Давайте создадим пример, основанный на объектно-ориентированном программировании. У нашего робота есть светодиод, который может использоваться для индикации того или иного состояния робота. Напишем программу, задающую время включения светодиода в секундах, с использованием класса.

```
class LED {//создаём класс LED
 public://общие свойства и методы
  LED(int connectPin);//это конструктор объекта
   int pin;//свойство pin — номер контакта
   void on (); //метод on включает светодиод
   void off ();//метод off выключает светодиод
   //метод blink — мигание светодиодом
```

```
    //параметр задаёт время включения в секундах,
   void blink(int timeOn);
  private: //используются только внутри
   //на какой контакт подключён светодиод
   byte pinNumber;
};
```

**Мы создали класс объектов LED. Каждый объект этого класса будет хранить номер контакта, к которому подключён, и ещё он может выполнять три действия: включать, выключать и мигать.**

**Далее необходимо расписать все методы класса, показывающие, как объект будет выполнять все действия.**

```
class LED {//создаём класс LED
 public://общие свойства и методы
  LED(byte connectPin);//это конструктор класса
   byte pin;//свойство pin — номер контакта
   void on (); //метод on включает светодиод
   void off ();//метод off выключает светодиод
   //метод blink — мигание светодиодом
   //параметр задаёт время включения в секундах,
  void blink(int timeOn);
 private: //используются только внутри
  //на какой контакт подключён
  byte pinNumber;
};

//что произойдёт, когда создадут
//объект класса LED
LED::LED(byte connectPin){
 //настраиваем порт на работу в режиме вывода
 pinMode(pin, OUTPUT);
 //Сохраним номер порта во внутреннюю переменную класса,
 //чтобы потом использовать в методах
 pinNumber = connectPin;
 //заполним свойство "pin"
 pin =  connectPin;
}
//описываем метод on
void LED::on(){
 digitalWrite(pinNumber,HIGH);
}
//описываем метод on
void LED::off(){
```

```
 digitalWrite(pinNumber,LOW);
}
//описываем метод blink
void LED::blink(int timeOn ){
 unsigned int t = millis()/1000%(timeOn+1);
 digitalWrite(pinNumber, t != 0 ? true : false);
}
```

А теперь посмотрите, как работать с классами в программе (рис. 104).

```
classLED §
  1 class LED  { //создаём класс LED
 15 LED::LED (int connectPin)  {
 23 void LED::on ()  {
 26 void LED::off ()  {
 29 void LED::blink (int timeOn ) {
 33  //создаём объект led1 класса LED
 34  //он наследует все свойства и методы своего класса
 35  LED ledl (13);
 36  void setup ()  {
 37    Serial.begin (9600);
 38    ledl.on (); delay (3000); led1.off ();
 39    Serial.print ("LED1: pin N" ); Serial.println (ledl.pin);
 40   }
 41  void loop ()  {
 42    ledl.blink (2);
 43  }
```

**Рис. 104.** Пример создания класса и объекта этого класса

### Задание 75

Напишите программу мигания светодиодом, используя создание класса.

## 12.2. Четыре основных принципа

В объектно-ориентированном программировании выделяют четыре основных принципа: *абстракция*, *инкапсуляция*, *наследование* и *полиморфизм*.

*Абстракция* — способ выделения самых значимых характеристик объекта, при этом менее значимые отбрасываются.

*Инкапсуляция* — это принцип, согласно которому любой класс должен рассматриваться как некий чёрный ящик, а пользователь класса должен видеть и использовать только список декларируемых свойств и методов класса. Данные принято инкапсулировать в классе таким

образом, чтобы доступ к ним по чтению или записи осуществлялся не напрямую, а с помощью методов.

*Наследование* — способность построить новый класс на основе уже заданного. В результате можно создавать прототипы — объекты-образцы, на основе которых «рождаются» другие объекты, полностью копирующие или изменяющиеся в процессе. При изменении в прототипе в копиях также происходят соответствующие изменения. Класс, от которого производится наследование, называется базовым или *родительским*, новый класс — *потомком*, наследником или производным классом.

*Полиморфизм* — способность объектов самим определять, какие методы они должны применить в зависимости от того, где именно в коде они находятся.

Эти принципы понадобятся, когда возникнет необходимость написать действительно сложную программу.

Получается хороший критерий: если эти принципы не используются при написании программы, то программа очень простая.

### Задание 76

www Ознакомьтесь по указанной ссылке (https://code-live.ru/post/cpp-classes/) с краткими примерами, как эти принципы применяют при создании программ на C++.

## 12.3. Создаём библиотеку правильно

У вас появился достаточный опыт работы с платформой Arduino, пора его переводить на более высокий уровень.

Давайте наконец-то рассмотрим пример правильной библиотеки, которую можно показать и единомышленникам.

Библиотека в программировании — это сборник подпрограмм, объектов (набор элементов программного кода), используемых для разработки программного обеспечения.

Библиотеки нужны для упрощения кода, для работы с различными аппаратными модулями и датчиками. В одной команде из библиотеки скрывается несколько строчек кода, написанных создателем библиотеки.

Итак, для создания библиотеки нам необходимо: создать папку с именем библиотеки; в этой папке создать три файла: заголовочный (.h), файл исходного кода (.cpp) и keywords.txt — файл для организации подсветки синтаксиса. Кроме того, в созданной папке должна быть подпапка Example с примерами использования библиотеки (рис. 105).

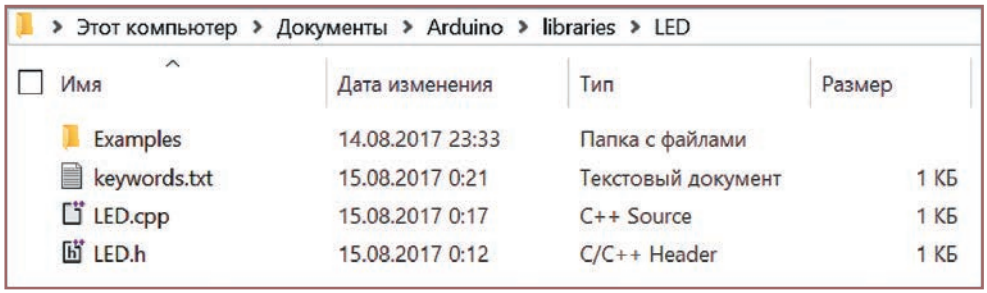



**Рис. 105.** Необходимые файлы и папки для создания библиотеки

В заголовочном файле описываются все типы/функции/классы/константы, а в файле исходного кода всё это реализуется.

Правилом «хорошего тона» является оформление кода в виде класса.

Начнём с заголовочного файла LED.h. В нём необходимо указать все определения классов, все директивы #define. Ещё в начале файла пишут комментарии, описывающие содержимое библиотеки.

Конструкция #ifndef / #define / #endif необходима для защиты от случайного повторного включения заголовка (рис. 106).

LED.h — Блокнот

Файл Правка Формат Вид Справка

```
/*
 * LED.h - библиотека для включения/выключения светодиода
 * без использования задержки (delay)
 * Автор (указать), Дата (указать).
 * Выпущено для публичного использования.
*/
#ifndef LED_h
#define LED_h
class LED {
 public:
  LED (byte pinConnect);
   byte pin;
   void on ();
   void off ();
   void blink(int timeOn);
 private:
  byte pinNumber;
};
#endif
```

**Рис. 106.** Заголовочный файл LED.h

Переходим к исходному коду библиотеки.

```
/*
 * LED.h — библиотека для включения/выключения светодиода
 * без использования задержки (delay)
 * Автор (указать), Дата (указать).
 * Выпущено для публичного использования.
*/
#include "Arduino.h"
#include "LED.h"

//описываем, что необходимо сделать при создании объекта
класса LED
LED::LED (byte connectPin) {
 //настраиваем порт на работу в режиме вывода
 pinMode(pin, OUTPUT);
 //Сохраним номер порта во внутреннюю переменную класса,
 //чтобы потом использовать в методах
 pinNumber = connectPin;
 //заполним свойство "pin"
 pin =  connectPin;
}


//описываем все используемые методы (функции)
//имя объекта :: название метода
void LED :: on() {
 digitalWrite(pinNumber,HIGH);
}
void LED :: off() {
 digitalWrite(pinNumber,LOW);
}
//включение/выключение через остатки от деления
void LED :: blink (int timeOn ) {
 unsigned int t = millis()/1000%(timeOn+1);
 digitalWrite(pinNumber, t != 0 ? true : false);
}
```

В начале файла пишется аналогичный информационный блок в виде комментариев.

Поскольку в исходном коде используются команды для работы с Arduino, подключаем библиотеку Arduino.h командой #include. Далее подключаем заголовочный файл нашей библиотеки и описываем все классы, функции и т. д.

Префикс «::» указывает, что метод входит в состав класса.

В файле keywords.txt нужно организовать подсветку синтаксиса (рис. 107).

В каждой строке должно быть имя ключевого слова, за которым через табуляцию указан тип ключевого слова. Имя класса должно быть обозначено как KEYWORD1 (оранжевая подсветка); функции (методы) обозначаются как KEYWORD2 (коричневая подсветка). Ещё есть подсветка констант (LITERAL1).

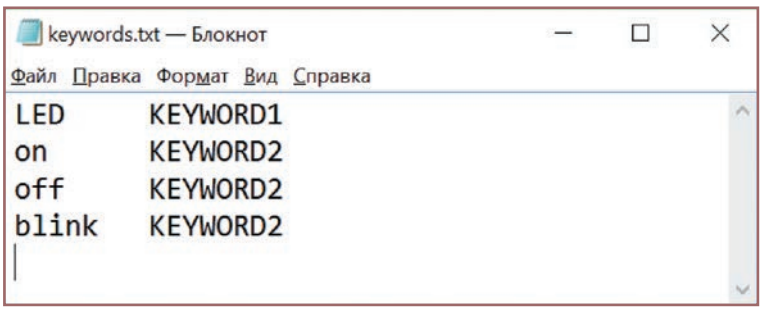



**Рис. 107.** Организация подсветки синтаксиса

Осталось написать пример, показывающий, как использовать библиотеку (например, файл с именем example1.ino).

```
#include <LED.h>
//создаём объект led1 класса LED
//все свойства и методы своего класса
LED led1(13);
void setup() {
  Serial.begin (9600);
  //используем методы on и off (включение и выключение)
  led1.on(); delay(3000); led1.off();
  //используем свойство pin — номер контакта подключения
  Serial.print ("LED1: pin N");Serial.println (led1.pin);
}
void loop() {
  //используем метод blink — включение на
  //указанное число секунд (выключение на 1 с)
  led1.blink (2);
}
```

## Задание 77

Создайте библиотеку, описанную выше.

# Глава 13
# ДВИЖЕНИЕ ПО ТРАЕКТОРИИ

## 13.1. Описание задания

Вспомним складских роботов. Предположим, что у нас стоит следующая задача: робот получает команду привезти товар из места хранения, назовём его условно стеллажом для товара, к терминалу упаковщика. Текущее месторасположение робота и маршрут движения показаны на рисунке 108. В основе всё то же движение по линии, но появились перекрёстки.

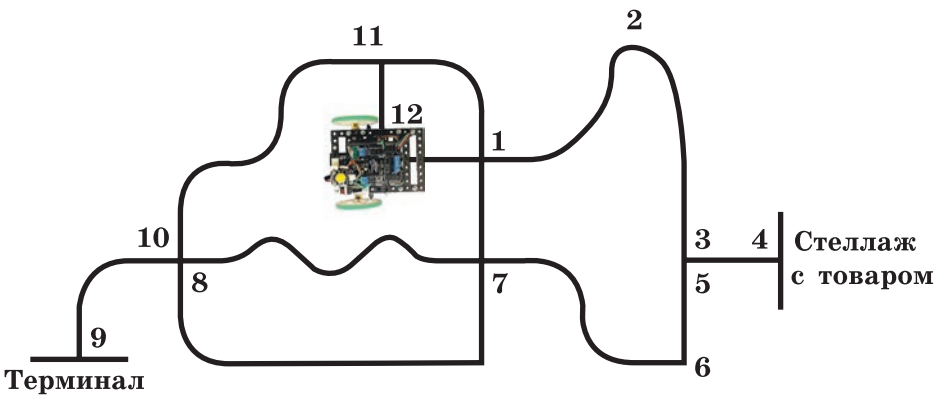



**Рис. 108.** Схема движения робота

Задача не очень сложная. Два датчика и ПИД-регулятор обеспечат плавное движение по линии, а два крайних датчика — обработку перекрёстков и углов (рис. 109).

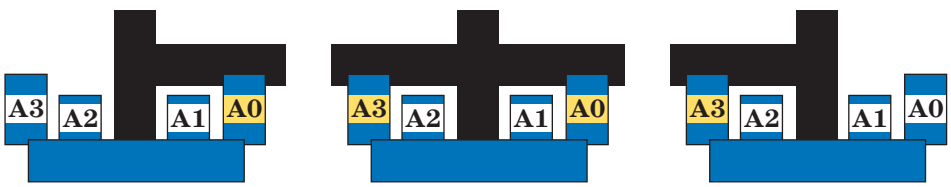



**Рис. 109.** Расположение датчиков на перекрёстках и углах

Какие могут быть сложности на пути? На участке 1–2–3 очень крутой поворот, участок 7–8 представляет собой «змейку», резкие переходы вправо/влево. Следовательно, придётся использовать разные коэффициенты ПИД-регулятора для разных участков, т. е. написать соответствующую функцию. В точках 3, 5, 6, 10 и 11 необходимо поворачивать на 90° (вправо и влево), а в точках 4 и 9 разворачиваться на 180°, значит, нужно использовать энкодеры (у вас уже написана библиотека). На схеме двенадцать точек — это много, следовательно,

есть смысл написать библиотеку, в которой будут команды движения вперёд/назад, повороты, движение по линии. Все части у вас уже сделаны, осталось объединить всё вместе и чуть оптимизировать.

## 13.2. Шаблон программы

Давайте обобщим, что конкретно мы хотим от робота, и создадим класс объектов. В рамках этого задания робот должен:

- ждать нажатия на сенсорную кнопку;
- двигаться вперёд и назад (используя ПИ-регулятор);
- поворачивать вправо и влево;
- останавливаться;
- двигаться по линии (ПИД-регулятор);
- предоставить нам информацию с крайних сенсоров (чтобы останавливаться на углах и перекрёстках).

Кроме того, он должен получать от нас информацию, с какой скоростью ему двигаться.

Первый шаг — создание папки и файлов (рис. 110).

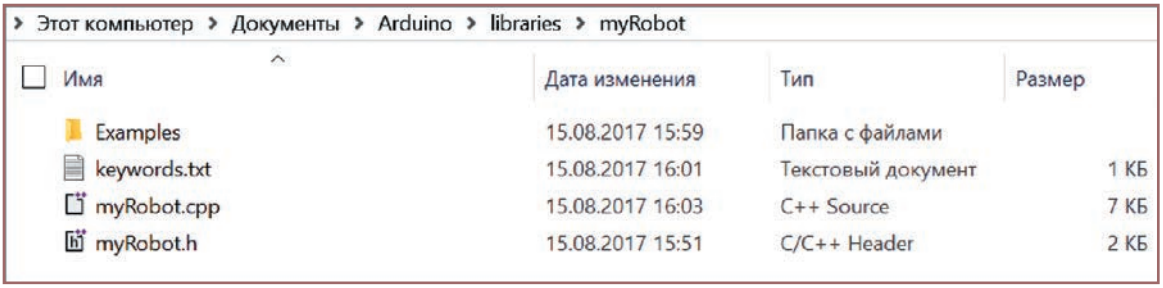



**Рис. 110.** Файлы и папки создаваемой библиотеки myRobot

Мы составили список действий, следовательно, мы перечислили методы, которые будут у класса *myRobot*.

Какую информацию о нашем роботе мы должны указать? Диаметр колёс, к какому порту подключены энкодеры и кнопка. Всё это — различные необходимые параметры методов нашего класса.

Итак, создаём класс *Robot* и прописываем его в заголовочном файле библиотеки. Указываем в нём директивы #define. После модификатора `private` перечисляем все необходимые переменные, которые мы использовали ранее (при поворотах, при движении по прямой и при движении по линии).

В итоге наш заголовочный файл myRobot.h будет выглядеть следующим образом.

**Файл myRobot.h**

```
/*
 * myRobot.h — библиотека для робота
 * Автор (указать), Дата (указать).
 * Выпущено для публичного использования.
*/
#ifndef myRobot_h
#define myRobot_h

#define DIR_M1              4
#define SPEED_M1            5
#define SPEED_M2            6
#define DIR_M2              7
#define LINE_SENSOR_M1_1   A0
#define LINE_SENSOR_M1_2   A1
#define LINE_SENSOR_M2_1   A2
#define LINE_SENSOR_M2_2   A3
#define LED_PIN            13

class myRobot {
 public:
  myRobot(float wheelDiameter, byte encoderPinM1,
          byte encoderPinM2, byte buttonPin);
   void waitButton ();
   void motors(byte minSpeed, byte motorM1, byte motorM2);
   void stopMove();
   void moveLine(float Kp, float Ki, float Kd, float alfa,
                 float beta, float brakeGain, int brake);
   void forward(int needSpeed, float distance, float Kp,
                float Ki);
   void backward(int needSpeed, float distance, float Kp,
                 float Ki);
   void turnLeft(int needSpeed, int countLimit);
   void turnRight(int needSpeed, int countLimit);
   int  sensorRead(byte port);

private:
   byte ledPin, BUTTON_PIN, ENCODER_M1, ENCODER_M2;
   bool encM1, encM2, encOldM1,encOldM2;
   bool reverseM1 = false, reverseM2 = false;
   int countM1=0, countM2=0;
   int motorsPower, powerM1, powerM2;
```

```
    int sensorValueM1_1, sensorValueM1_2;
    int sensorValueM2_1, sensorValueM2_2;
    int setPoint1, setPoint2;
    int actualValue1, actualValue2;
    byte START_POWER_M1, START_POWER_M2, START_SPEED;
    float resultTerm, error, errorSum, errorOld;
    float controlSummErrors=0;
    float WHEEL_DIAMETER;
    float spokes (float distance);
    void robotGo(int speedM1, int speedM2, bool reverseM1,
                 bool reverseM2);
    void sensorsSetPoint();
    void sensorsRead();
};
#endif
```

Так как все методы мы прописали, то можно их указать в файле keywords.txt для подсветки синтаксиса (рис. 111).

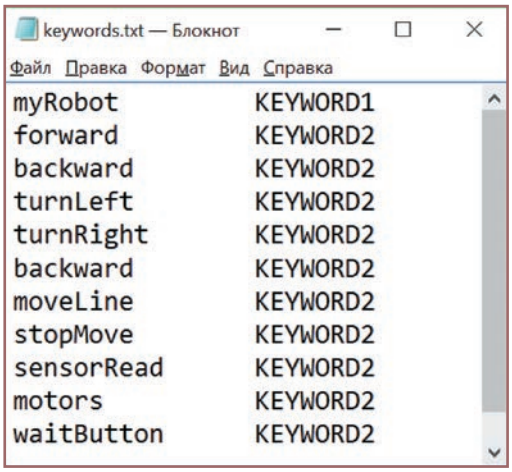



**Рис. 111.** Организация подсветки синтаксиса библиотеки myRobot

Следующий шаг — написание исходного кода. Он у нас практически весь уже написан в прошлых заданиях. Необходимо только аккуратно проверить, чтобы не было ни лишних переменных, ни их дублирований.

**Файл myRobot.cpp**

```
/*
 * myRobot.h – библиотека для робота
 * Автор (указать), Дата (указать).
 * Выпущено для публичного использования.
*/
```

```
#include "Arduino.h"
#include "myRobot.h"

myRobot::myRobot (float wheelDiameter, byte encoderPinM1,
                  byte encoderPinM2, byte buttonPin) {
   Serial.begin(9600);
   pinMode (buttonPin,INPUT);
   pinMode (encoderPinM1,INPUT);
   pinMode (encoderPinM2,INPUT);
   for (int i=4;i<=7;i=i+1) pinMode(i,OUTPUT);
   WHEEL_DIAMETER = wheelDiameter;
   BUTTON_PIN = buttonPin;
   ENCODER_M1 = encoderPinM1;
   ENCODER_M2 = encoderPinM2;
}

void myRobot :: waitButton () {
  while(digitalRead(BUTTON_PIN)==false)
  digitalWrite(LED_PIN, millis()/1000%3 ? true: false);
  sensorsSetPoint ();
}

void myRobot :: motors (byte minSpeed, byte motorM1,
                        byte motorM2) {
 START_SPEED = minSpeed;
 START_POWER_M1 = motorM1;
 START_POWER_M2 = motorM2;
}

int myRobot :: sensorRead(byte port) {
 if (port == 0) {return constrain
    (map(analogRead(LINE_SENSOR_M1_1),34,623,100,0),0,100);}
 if (port == 1) {return constrain
    (map(analogRead(LINE_SENSOR_M1_2),33,694,100,0),0,100);}
 if (port == 2) {return constrain
    (map(analogRead(LINE_SENSOR_M2_1),42,654,100,0),0,100);}
 if (port == 3) {return constrain
    (map(analogRead(LINE_SENSOR_M2_2),37,761,100,0),0,100);}
}

void myRobot :: forward(int needSpeed, float distance, f
                        loat Kp, float Ki) {
 motorsPower = START_SPEED;
```

```
 encOldM1 = digitalRead(ENCODER_M1);
 encOldM2 = digitalRead(ENCODER_M2);
 while (countM1 < spokes(distance) || countM2
     < spokes(distance)) {
  Serial.println(countM1);
  motorsPower = motorsPower<100 ? motorsPower+=1 :
                needSpeed;
  encM1 = digitalRead(ENCODER_M1);
  encM2 = digitalRead(ENCODER_M2);
  if (encM1 != encOldM1) countM1++;
  if (encM2 != encOldM2) countM2++;
  error = countM1 — countM2;
  controlSummErrors=controlSummErrors + Ki*error;
  int controlAction = Kp*error + controlSummErrors;
  powerM1=constrain(motorsPower-controlAction,0,255);
  powerM2=constrain(motorsPower+controlAction,0,255);
  digitalWrite(DIR_M1, HIGH);
  digitalWrite(DIR_M2, HIGH);
  analogWrite(SPEED_M1,powerM1);
  analogWrite(SPEED_M2,powerM2);
  delay(5);
  encOldM1 = encM1; encOldM2 = encM2;
 }
 motorsPower = (powerM1+powerM2)/2;
 while (motorsPower>0) {
  analogWrite(SPEED_M1,motorsPower);
  analogWrite(SPEED_M2,motorsPower);
  motorsPower-=1; delay(1);
 }
 stopMove();
}

void myRobot :: backward(int needSpeed, float distance,
                         float Kp, float Ki) {
 motorsPower = START_SPEED;
 encOldM1 = digitalRead(ENCODER_M1);
 encOldM2 = digitalRead(ENCODER_M2);
 while (countM1 < spokes(distance) || countM2
       < spokes(distance)) {
  motorsPower = motorsPower<100 ? motorsPower+=1 :
                needSpeed;
  encM1 = digitalRead(ENCODER_M1);
  encM2 = digitalRead(ENCODER_M2);
```

```
  if (encM1 != encOldM1) countM1++;
    if  (encM2  !=  encOldM2)  countM2++;
  error = countM1 – countM2;
  controlSummErrors=controlSummErrors + Ki*error;
  int controlAction=Kp*error+controlSummErrors;
  powerM1=constrain(motorsPower-controlAction,0,255);
  powerM2=constrain(motorsPower+controlAction,0,255);
  digitalWrite(DIR_M1, LOW);
  digitalWrite(DIR_M2, LOW);
  analogWrite(SPEED_M1,powerM1);
  analogWrite(SPEED_M2,powerM2);
  delay(5);
  encOldM1 = encM1;  encOldM2 = encM2;
 }
 motorsPower = (powerM1+powerM2)/2;
 while (motorsPower>0) {
  analogWrite(SPEED_M1,motorsPower);
  analogWrite(SPEED_M2,motorsPower);
  motorsPower-=1;  delay(1);
 }
 stopMove();
}

void myRobot :: turnLeft(int needSpeed, int countLimit) {
 encOldM1 = digitalRead (ENCODER_M1);
 encOldM2 = digitalRead (ENCODER_M2);
 while (countM1 < countLimit || countM2 < countLimit) {
  encM1 = digitalRead(ENCODER_M1);
  encM2 = digitalRead(ENCODER_M2);
  if (encM1 != encOldM1) countM1++;
  if (encM2 != encOldM2) countM2++;
  digitalWrite(DIR_M1, HIGH);
  digitalWrite(DIR_M2, LOW);
  analogWrite(SPEED_M1,needSpeed);
  analogWrite(SPEED_M2,needSpeed);
  delay(5);
  encOldM1 = encM1; encOldM2 = encM2;
 }
 stopMove();
}

void myRobot :: stopMove() {
 analogWrite(SPEED_M1,0); analogWrite(SPEED_M2,0);
 countM1=0; countM2=0;
}
```

```
void myRobot :: turnRight(int needSpeed, int countLimit) {
 encOldM1 = digitalRead (ENCODER_M1);
 encOldM2 = digitalRead (ENCODER_M2);
 while (countM1 < countLimit || countM2 < countLimit) {
  encM1 = digitalRead(ENCODER_M1);
  encM2 = digitalRead(ENCODER_M2);
  if (encM1 != encOldM1) countM1++;
  if (encM2 != encOldM2) countM2++;
  digitalWrite(DIR_M1, LOW);
  digitalWrite(DIR_M2, HIGH);
  analogWrite(SPEED_M1,needSpeed);
  analogWrite(SPEED_M2,needSpeed);
  delay(5);
  encOldM1 = encM1;
  encOldM2 = encM2;
 }
 stopMove();
}

void myRobot :: sensorsRead() {
 sensorValueM1_1 = constrain
    (map(analogRead(LINE_SENSOR_M1_1),34,623,100,0),0,100);
 sensorValueM1_2 = constrain
    (map(analogRead(LINE_SENSOR_M1_2),33,694,100,0),0,100);
 sensorValueM2_1 = constrain
    (map(analogRead(LINE_SENSOR_M2_1),42,654,100,0),0,100);
 sensorValueM2_2 = constrain
    (map(analogRead(LINE_SENSOR_M2_2),37,761,100,0),0,100);
}

void myRobot :: sensorsSetPoint () {
 sensorsRead();
 setPoint1 = sensorValueM1_1 – sensorValueM2_2;
 setPoint2 = sensorValueM1_2 – sensorValueM2_1;
}

float myRobot :: spokes (float distance) {
  return 14*distance/(PI*WHEEL_DIAMETER);
}

void myRobot :: robotGo(int speedM1, int speedM2,
                bool reverseM1, bool reverseM2) {
  analogWrite(SPEED_M1, speedM1);
  analogWrite(SPEED_M2, speedM2);
  digitalWrite(DIR_M2, reverseM2 ? LOW : HIGH);
```

```
  digitalWrite(DIR_M1, reverseM1 ? LOW : HIGH);
}

void myRobot :: moveLine(float Kp, float Ki, float Kd,
   float alfa, float beta, float brakeGain, int brake) {
 sensorsRead();
 actualValue1 = sensorValueM1_1 – sensorValueM2_2;
 actualValue2 = sensorValueM1_2 – sensorValueM2_1;
 error = (setPoint1 – actualValue1)*alfa +
         (setPoint2 – actualValue2)*beta;
 errorSum = errorSum + error;
 resultTerm = Kp*error + Ki*errorSum +
              Kd*(error – errorOld);
 powerM1 = (START_POWER_M1 – brakeGain*resultTerm –
           (resultTerm))/brake ;
 powerM2 = (START_POWER_M2 – brakeGain*resultTerm +
           (resultTerm))/brake ;
 reverseM1 = powerM1 < 0 ? true : false;
 reverseM2 = powerM2 < 0 ? true : false;
 powerM1 = abs(powerM1) > 255 ? 255: abs(powerM1);
 powerM2 = abs(powerM2) > 255 ? 255: abs(powerM2);
 robotGo (powerM1, powerM2, reverseM1, reverseM2);
 delay(10);
 errorOld = error;
}
```

**В завершение оформим файл с примером использования библиотеки myRobot.**

```
#include <myRobot.h>;
/* загрузить библиотеку;
 * создать объект класса myRobot;
 * указать диаметр колеса, контакты
 * двух энкодеров и датчика касания
 */
myRobot Leonardi (6.67, 2, 3, 8);

void setup() {
 /* в параметрах метода motors
  * указать min скорость, при которой
  * робот начинает двигаться; и скорости
  * моторов, при которых робот едет прямо  */
 Leonardi.motors (50,220,200);
 /* ожидание роботом нажатия кнопки 4
  * также будет задана уставка   */
 Leonardi.waitButton ();
```

```
}
void loop() {
  /*указать задержку между действиями */
  int delayTime = 1000;
  /*в команде sensorRead указать аналоговый порт от 0
   * до 3; результат — число от 0 ("вижу чёрный")
   * до 100 ("вижу белый") */
  while (Leonardi.sensorRead(0)>10 && Leonardi.
sensorRead(3)>10 ){
   /*в команде moveLine указать 3 коэффициента
    * ПИД-регулятора и во сколько раз снизить скорость
    * moveLine (Kp,Ki,Kd,alfa,beta,brakeGain,brake)*/
    Leonardi.moveLine (1.1,0.005,1.3,0.8,2.5,0.3,1);
  }
  Leonardi.stopMove(); delay (delayTime);
  /* в командах вперёд/назад указывается скорость
   * и дистанция в см
   * forward (speed,distance,Kp, Ki)
   * backward (speed,distance,Kp, Ki)     */
  Leonardi.forward(150,30,16,1.2);   delay (delayTime);
  Leonardi.backward(150,30,16,1.2);  delay (delayTime);
  /* в командах поворотов turnRight и turnLeft
   *  указывается скорость
   *  и количество делений, посчитанных энкодером   */
  Leonardi.turnRight(120,6);  delay (delayTime);
  Leonardi.turnLeft (120,6); delay (delayTime);
  /*поставьте "заглушку",
   * чтобы цикл не выполнялся бесконечно */
  while (true);
}
```

### Задание 78

Создайте библиотеку myRobot и реализуйте движение робота по схеме, указанной на рисунке 104.

### Задание 79

Составьте свою схему движения робота и запрограммируйте робота на выполнение задания.

# Глава 14
# ОСТАНОВКА У ПРЕПЯТСТВИЯ

## 14.1. ИК-дальномер

Ещё одной классической задачей в робототехнике является объезд роботом препятствий. Мы пока должны реализовать остановку у объекта. Вы уже знаете, что для этих целей можно использовать ультразвуковой дальномер. Кроме него в вашем наборе есть сенсор, работающий по другому принципу, — инфракрасный (ИК) дальномер.

### Задание 80

www

Изучите технический лист датчика SHARP GP2Y0A021 (http://files.amperka.ru/datasheets/gp2y0a21yk_e.pdf). Узнайте:

- максимальный потребляемый ток, т. е. можно ли подключить датчик к контактам на плате Arduino напрямую;
- как подключить его к нашему роботу;
- диапазон расстояний, на которых он может определять объект;
- что представляет собой выходной сигнал и как он меняется с изменением расстояния до объекта;
- принцип работы датчика;
- условия и ограничения при работе датчика;
- как вычислить расстояние до объекта;
- размеры датчика, чтобы смоделировать крепёж.

На какие проблемы вы должны обратить внимание при изучении (и переводе) технического листа? Во-первых, датчик имеет нелинейный выход, т. е. при линейном увеличении расстояния сигнал на аналоговом выходе уменьшается нелинейно. Также по графику видно, что у этого ИК-дальномера возникнут проблемы при обнаружении объектов, находящихся ближе 10 см. Самой же главной сложностью является отсутствие прямой информации о том, как же перевести числа от 0 до 1023, видимые платой Arduino, в сантиметры. Формулы нет.

Конечно, вы можете постараться найти в Интернете примеры таких формул. Но всё-таки лучше научиться методу, который позволит находить зависимости вообще для любых датчиков. Даже для тех, техническое описание которых отсутствует.

## Задание 81

Используя созданный ранее держатель для ультразвукового дальномера, смоделируйте и напечатайте крепёж для ИК-дальномера (рис. 112). Прикрепите дальномер к роботу (рис. 113).

```
$fn=48 ;
difference () {
  union() {
    translate ( [0, 0, –21/2–1] )
    color ("red") cube ( [45.4+20+6, 10, 3], true ) ;
    translate ( [–45.4/2–8, 0, –15] )
    color ("gold") cube ( [10, 10, 5], true) ;
    translate ( [45.4/2+8, 0, –15] )
    color ("magenta") cube ( [10, 10, 5], true ) ;
    translate ( [0, 3.5, 1] )
    color ("blue") cube ( [48, 3, 25], true) ;
  }
  translate ( [–30, 0, –21/2–4] )
  cylinder (10, 2, 2, true);
  translate ( [30, 0, –21/2–4] )
  cylinder (10, 2, 2, true) ;
  translate ( [0, 2, –5] )
  cube ( [10, 10, 10], true) ;
  translate ( [–37/2, 2, 7])
  rotate ( [90, 0, 0] ) cylinder (10, 2, 2, true) ;
  translate ( [37/2, 2, 7] )
  rotate ( [90, 0, 0] ) cylinder (10, 2, 2, true) ;
}
```

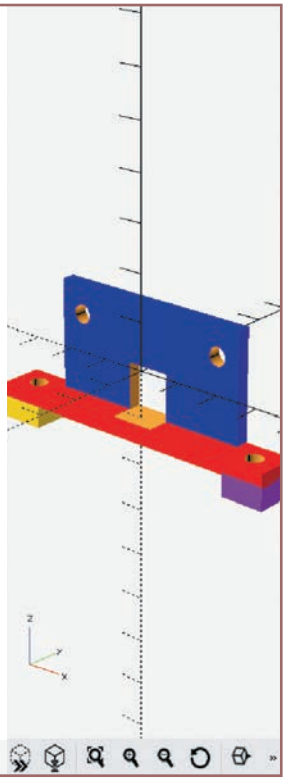


**Рис. 112.** Модель крепежа ИК-дальномера

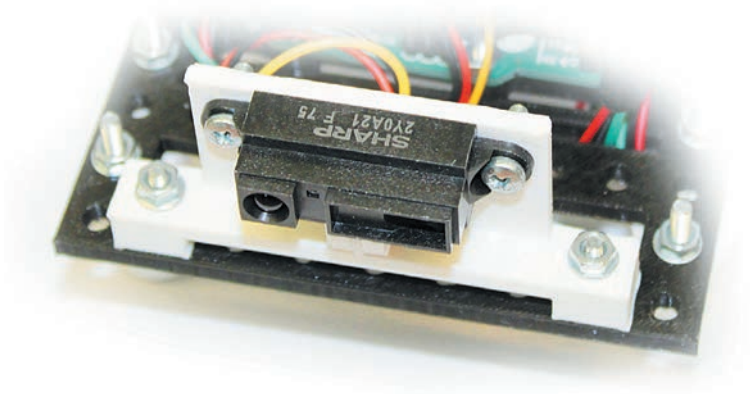



**Рис. 113.** Установленный на роботе дальномер

## 14.2. Аппроксимация и фильтр

Итак, необходимо организовать небольшой стенд (рис. 114). Робот подключён к компьютеру и передаёт на монитор данные с датчика, подключённого к входу A5, по простой программе, показанной ниже.

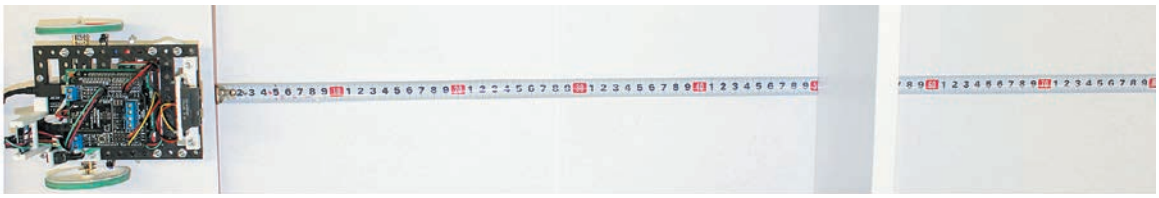


**Рис. 114.** Установленный на роботе дальномер

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  int IRsensorValue = analogRead(A5);
  Serial.println(IRsensorValue);
  delay(100);
}
```

Ставим любой объект перед роботом и записываем в электронную таблицу показания: данные с порта A5 и расстояние от 10 до 80 см. Затем строим точечную диаграмму (рис. 115).

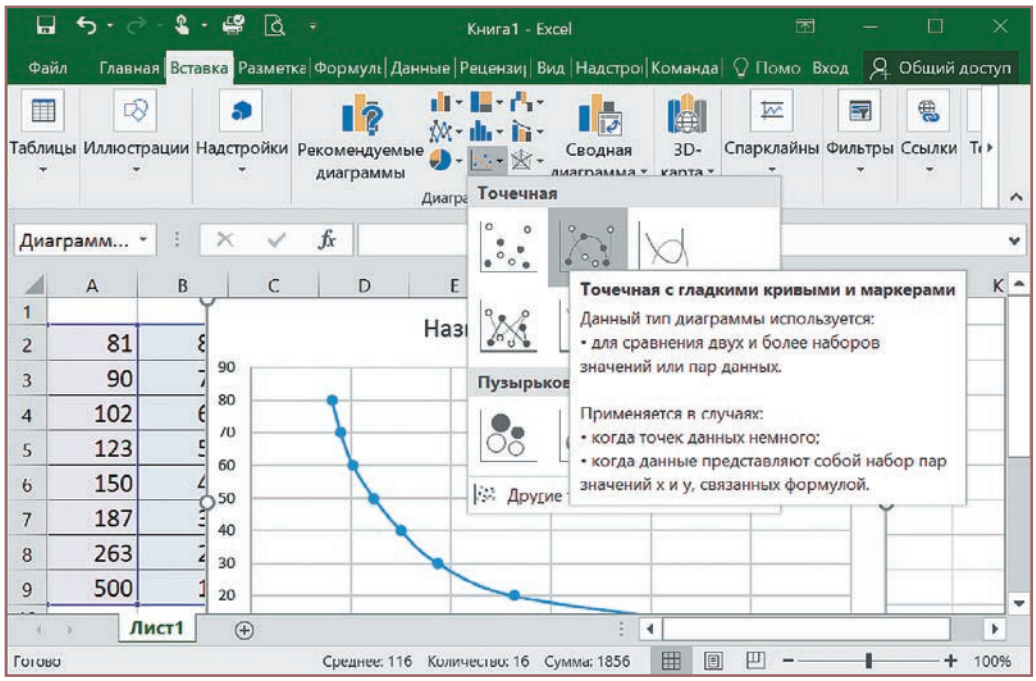



**Рис. 115.** Создание графика по точкам

После этого добавляем на диаграмму линию тренда (рис. 116). Выбираем такую, чтобы линии графиков совпали, и обязательно уточняем, что необходимо показать уравнение.

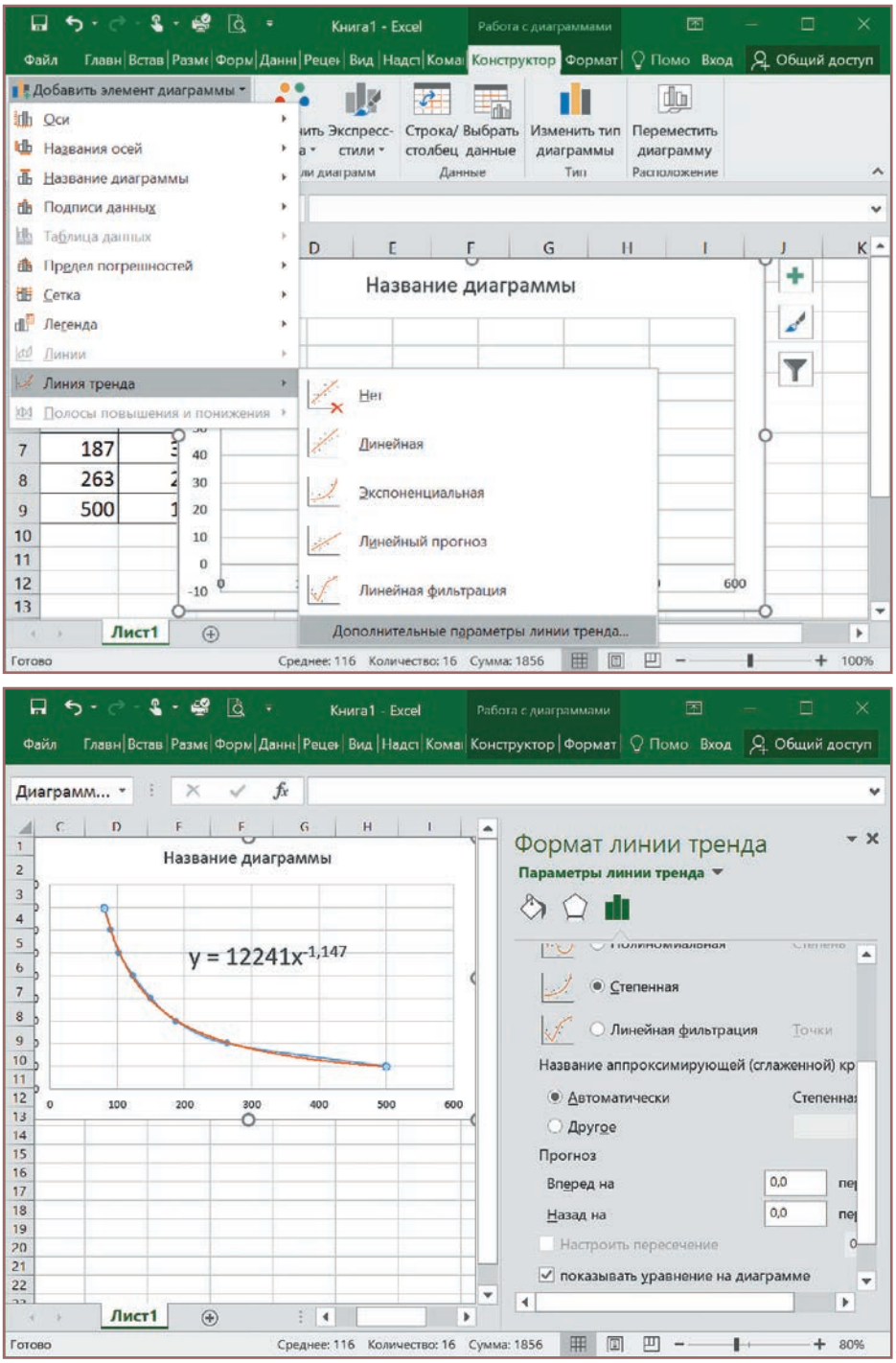



**Рис. 116.** Создание линии тренда

Для нашего датчика очень хорошо подошла степенная функция

$$y = 12\,241 \cdot x^{-1,147}.$$

Что на самом деле мы только что сделали? Мы применили мощнейший научный метод исследования объектов, процессов и явлений — *аппроксимацию*. Аппроксимация состоит в замене одних объектов другими, в каком-то смысле близкими к исходным, но более простыми. Метод позволяет исследовать числовые характеристики и качественные свойства объекта, сводя задачу к изучению более простых или более удобных объектов.

А линия тренда — один из основных инструментов анализа данных. Благодаря этому инструменту электронных таблиц мы по соседним значениям восстановили значения, находящиеся между ними. Способ определения промежуточных значений величины по имеющемуся дискретному набору известных значений называется *интерполяцией*.

Итак, аппроксимацию провели. Следующее действие — внесение корректив в программу. Создаём функцию для пересчёта данных, для возведения числа в степень используем функцию `pow`.

```
float IRdistance (int inputValue) {
 return 12241*pow(inputValue,-1.147);
}
void setup() {
  Serial.begin(9600);
}
void loop() {
  int IRsensorValue = analogRead(A5);
  Serial.println(IRdistance (IRsensorValue));
  delay(100);
}
```


**Рис. 117.** Расстояние до объекта в сантиметрах

Смотрим и анализируем информацию на мониторе последовательного порта (рис. 117). Как вы видите, регулярно появляются ошибки — 10 см!

Надо уменьшать. Что можно сделать? Воспользоваться таким понятием, как среднее арифметическое. Сделать несколько измерений, сложить данные и вычислить среднее значение. Пробуем.

Используем для этих целей массив (пронумерованный набор элементов).

```
#define IR_SENSOR_PIN   A5
float IRdistance () {
  int summValue=0;
  for (int i=1;i<6;i++) {int IRsensorValue =
                          analogRead(IR_SENSOR_PIN);
    summValue = summValue + IRsensorValue;
    delay(1);
  }
  summValue = summValue/5;
  return 12241*pow(summValue,-1.147);
}
void setup() {
  Serial.begin(9600);
}
void loop() {
    Serial.println(IRdistance ());
  delay(100);
}
```

Анализируя данные (рис. 118), мы видим, что стало значительно лучше.

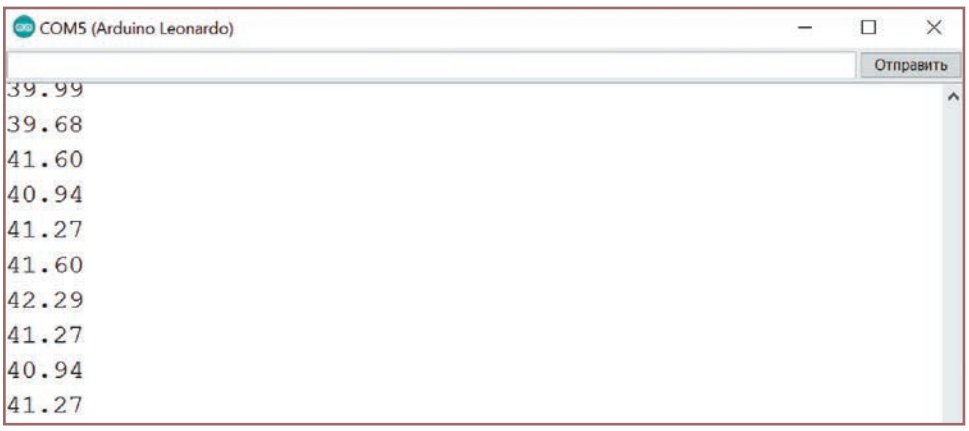



**Рис. 118.** Усреднённые данные о расстоянии до объекта в сантиметрах

Однако хотелось бы, чтобы ошибка была не более 1 см. Для этого нужно отфильтровать значения. Сделаем это следующим образом: проведём 20 измерений, запишем в массив из 20 элементов, отсортируем массив по возрастанию, 5 самых маленьких и 5 самых больших элементов проигнорируем, найдём среднее значение оставшихся десяти и будем именно его использовать.

Попробуем сортировать элементы массива методом пузырька. Алгоритм состоит из повторяющихся проходов по массиву. За каждый проход элементы массива последовательно сравниваются попарно. Если порядок в паре неверный, меняем элементы местами. Для обмена используется вспомогательная переменная *tmp*.

При каждом проходе по внутреннему циклу (со счётчиком *i*) наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива. Говорят, что элемент «всплывает» до нужной позиции, как пузырёк в воде, отсюда и название алгоритма.

```
#define IR_SENSOR_PIN   A5
int IRdistance () {
  int summValue=0; int filter [20];
  for (int i=0;i<=19;i++) {
    filter[i] = analogRead(IR_SENSOR_PIN);
 }
  for (int j=19;j<=1;j--) {
   for (int i=1;i<=j;i++) {
     if (filter[i-1] > filter [i]) {
       int tmp = filter[i-1];
       filter[i-1] = filter[i]; filter[i]=tmp;
     }
   }
  }
  for (int i=5;i<=14;i++) summValue=summValue+filter[i];
  summValue = summValue/10;
  return int (12241*pow(summValue,-1.147));
}
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.println(IRdistance ());   delay(50);
}
```

Также изменим функцию *distance*, чтобы получать натуральные числа, а не дробные. В результате мы добьёмся ошибки в 1 см (рис. 119).


**Рис. 119.** Новые усреднённые данные о расстоянии до объекта в сантиметрах

## Задание 82

Проделайте вышеописанный алгоритм для вашего датчика (ИК-дальномера). Добейтесь приемлемой точности результатов измерений (ошибка не более 1 см).

## Задание 83

Напишите программу, по которой робот может останавливаться у объекта на заданном расстоянии.

## Задание 84

Пусть робот едет прямо. Если ближе чем, например, 10 см появляется объект, то робот останавливается и ждёт. Как только путь становится свободным, робот продолжает движение. Пример выполнения роботом задания представлен по ссылке: https://youtu.be/xlvw2T2E2KI.

# Глава 15
# ДВИЖЕНИЕ ВДОЛЬ СТЕНЫ

## 15.1. Есть проблемы

Ещё одной классической задачей в робототехнике является движение робота вдоль стены и её обобщённый вариант — выход из лабиринта. С учётом ваших знаний о ПИД-регуляторе задача кажется совсем простой. Правда, это не совсем так (рис. 120).

Во-первых, стоит задуматься над следующими вопросами:

- Сколько дальномеров необходимо использовать?
- Какие использовать дальномеры: ультразвуковые или инфракрасные?
- Какое расположение датчиков предпочтительнее: вертикальное или горизонтальное?
- Что делать, когда робот подъедет к внешнему углу 90° и будет резкий (на порядок) скачок ошибки? Ведь ПИД-регулятор с этим не справится — дифференциальная и интегральные составляющие точно заставят его крутиться.
- Что делать, если робот подъезжает к внутреннему углу (например, тоже 90°)?
- На каком расстоянии от стены нужно двигаться роботу?

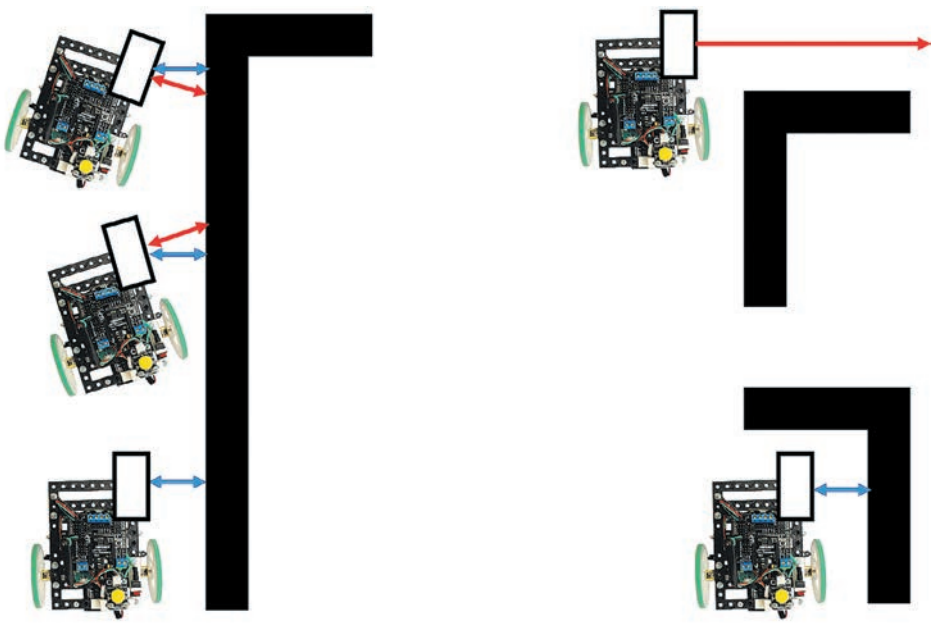


**Рис. 120.** Возможные проблемные ситуации

Сначала попробуем использовать всего один ультразвуковой дальномер. И попытаемся ответить на вопрос: если нет внутренних углов, то робот с одним дальномером сможет пройти вдоль стены?

Чтобы не делать лишних бесперспективных экспериментов, вспомним, что диапазон расстояний датчика составляет от 0 до 450 см (даже если опустить мёртвые зоны). При этом роботу необходимо быть в 20–40 см от стены. Разница получается больше чем на порядок (т. е. больше чем в 10 раз).

А что у нас с «сырыми» данными, если взять за основу число микросекунд, которое возвращает команда *pulseIn*? В таком случае диапазон значений будет от 0 до 10 000.

Как сжать этот диапазон, причём неравномерно, чтобы сжатие увеличивалось и увеличивалось с каждым приближением верхней границы? В идеале получить график, как на рисунке 121, и тогда ошибка по внешнему углу будет небольшой, и ПИД-регулятор справится с такой ситуацией, а робот спокойно повернёт.

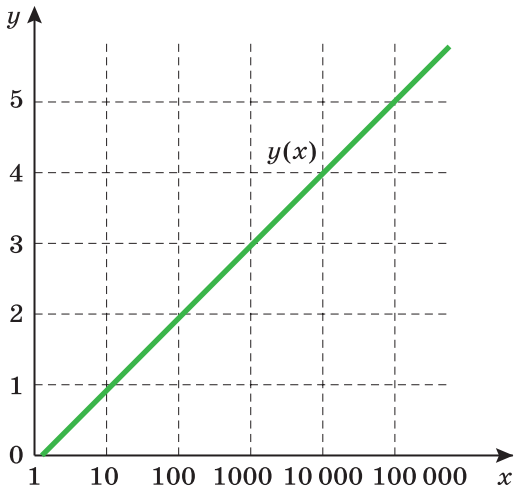



**Рис. 121.** По оси $x$ — показания дальномера, по оси $y$ — ошибка

## 15.2. Десятичный логарифм

И опять к нам на помощь придут математические функции, в частности логарифм. *Логарифм* по основанию $a$ числа $b$ — это показатель степени, в которую нужно возвести число $a$, чтобы получить число $b$. Запись выглядит следующим образом: $\log_a b = c$, что означает: чтобы получить число $b$, нужно число $a$ возвести в степень $c$, при этом $a > 0$, $a \neq 1$ и $b > 0$.

Логарифмическая функция — это функция вида $y = \log_a x$ при $a > 1$ или $0 < a < 1$.

Наиболее широкое применение нашли следующие виды логарифмов:

- десятичные: $\log_{10} a$ (основание — число 10);
- натуральные: $\ln a$ (основание — $e$ (число Эйлера));
- двоичные: $\log_2 a$ (основание — число 2).

Нас в данный момент очень интересует десятичный логарифм — логарифм по основанию 10 (рис. 122), так как неравномерная шкала десятичных логарифмов используется во многих областях науки. При этом график, построенный в логарифмическом масштабе по одной или двум осям, принимает вид прямой, более удобной для исследования.

Таким образом, на рисунке 122 изображён график логарифмической функции $y = \log_{10} x$ с использованием логарифмической шкалы.

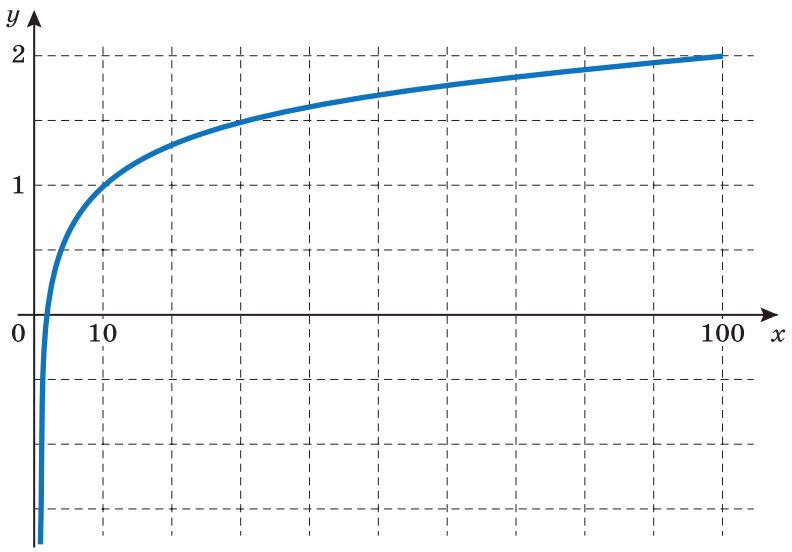



**Рис. 122.** График функции $y = \log_{10} x$

## 15.3. Вертикальное крепление дальномера

### Задание 85

Смоделируйте и распечатайте вертикальное крепление для ультразвукового дальномера. Присоедините его к роботу (рис. 123 и 124).

```
difference () {
  union () {
    color ("red")
    translate ( [5, 0, –21/2–1] )
    cube ( [45.4+16, 10, 3], true) ;
    color ("pink")
    translate ( [30, 0, –21/2–1+30] )
    cube ( [12.5, 10, 3], true) ;
    translate ( [–45.4/2–3/2+0.7, 0, 0] )
    color ("green")
    cube ( [3, 10, 21], true) ;
    translate ( [45.4/2+3/2–0.7, 0, 5] )
    color ("cyan")
    cube ( [3, 10, 30], true) ;
    translate ( [45.4/2+11, 0, –16+40] )
    color ("gold")
    cube ( [5, 10, 10], true) ;
    translate ( [45.4/2 + 11, 0, –16] )
    color ("magenta")
    cube ( [5, 10, 12] , true) ;
    color ("blue")
    translate ( [36, 0, –21/2–6] )
    rotate ( [0, 90, 0] ) cylinder (10, 2, 2, true) ;
  }
  translate ( [0, 0, 5] ) cube ( [45.6, 1.7, 34], true) ;
  translate ( [34, 0, –21/2–6+40] )
  rotate ( [0, 90, 0] ) cylinder (10, 1.8, 1.8, true) ;
}
```

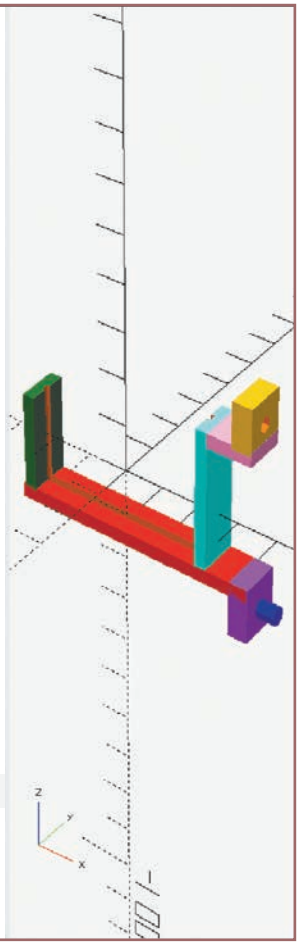


**Рис. 123.** Модель вертикального крепления дальномера

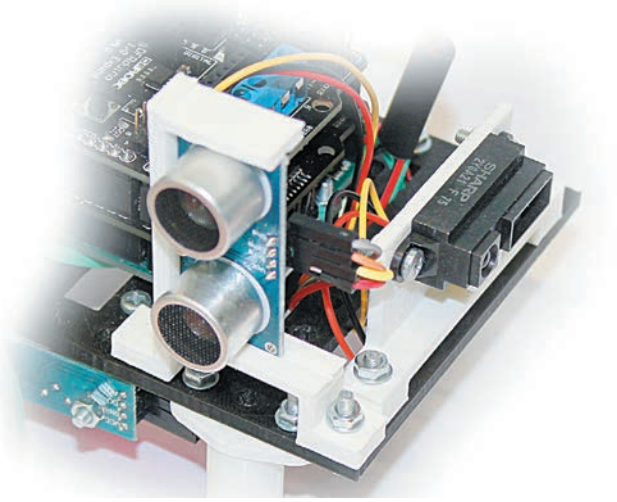


**Рис. 124.** Два дальномера, присоединённые к роботу

## 15.4. Примеры программ

### Задание 86

Изучите пример программы, при выполнении которой робот двигается вдоль стенки, обходя внешние углы. Видео с движением робота вокруг круглого и прямоугольного объектов можно посмотреть по ссылкам: https://youtu.be/jGFsT7Qqae8 и https://youtu.be/evusQxgnLJk.

```
#define DIR_M1              4 //правый мотор
#define SPEED_M1            5
#define SPEED_M2            6
#define DIR_M2              7 //левый мотор
#define LED_PIN            13
class Walker {
 public:
  Walker (byte triggerPin, byte echoPin, byte buttonPin);
   void  startDistance ();
   void  waitButton ();
   float distance ();
   void  motors (byte motorM1, byte motorM2);
   void  moveWall( float Kp, float Ki, float Kd, float
brakeGain, int brake);
 private:
   byte BUTTON_PIN, ECHO_PIN, TRIG_PIN;
   byte START_POWER_M1, START_POWER_M2;
   int setPoint;
   float oldDistance, errorSum=0, errorOld=0;
};
Walker :: Walker(byte triggerPin, byte echoPin, byte
buttonPin)   {
  Serial.begin(9600);
   pinMode (LED_PIN,OUTPUT);
   pinMode (buttonPin,INPUT);
   pinMode (triggerPin,OUTPUT);
   pinMode (echoPin,INPUT);
   for (int i=4;i<=7;i=i+1) pinMode(i,OUTPUT);
   BUTTON_PIN = buttonPin;
   ECHO_PIN = echoPin;
   TRIG_PIN = triggerPin ;
}
```

```
void Walker :: waitButton () {
  while(digitalRead(BUTTON_PIN)==false)
   digitalWrite(LED_PIN, millis()/1000%3 ? true: false);
}
void Walker :: startDistance () {
  setPoint = distance();
  oldDistance = setPoint;
  //Serial.println(setPoint);
}
float Walker :: distance(){
   digitalWrite(TRIG_PIN, LOW);
   delayMicroseconds(1);
   digitalWrite(TRIG_PIN, HIGH);
   delayMicroseconds(10);
   digitalWrite(TRIG_PIN, LOW);
   unsigned long timeBack = pulseIn(ECHO_PIN, HIGH);
   float parrots = log10(timeBack)*20;
   delay(1);
   return int (parrots);
}
void Walker :: motors (byte motorM1, byte motorM2) {
 START_POWER_M1 = motorM1;
 START_POWER_M2 = motorM2;
}
void Walker :: moveWall(float Kp, float Ki, float Kd,
                        float brakeGain, int brake) {
 float actualValue = distance();
 Serial.println (actualValue);
 int error = actualValue — setPoint;
 errorSum = errorSum + error;
 float resultTerm = Kp*error + Ki*errorSum +
                    Kd*(error — errorOld);
 int powerM1 = (START_POWER_M1 — brakeGain*resultTerm —
                resultTerm)/brake ;
 int powerM2 = (START_POWER_M2 — brakeGain*resultTerm +
                resultTerm)/brake ;
 bool reverseM1 = powerM1 < 0 ? true : false;
 bool reverseM2 = powerM2 < 0 ? true : false;
 powerM1 = abs(powerM1) > 255 ? 255: abs(powerM1);
 powerM2 = abs(powerM2) > 255 ? 255: abs(powerM2);
 digitalWrite(DIR_M2, reverseM2 ? LOW : HIGH);
 digitalWrite(DIR_M1, reverseM1 ? LOW : HIGH);
 analogWrite(SPEED_M1, powerM1);
```

```
 analogWrite(SPEED_M2, powerM2);
 delay(10);
 errorOld = error;
}
```

**Walker robotArdi (10,11,8);**

```
void setup() {
 robotArdi.motors (240,180);
 robotArdi.waitButton();
 robotArdi.startDistance();
}
void loop() {
 robotArdi.moveWall(14,0.01,2,-0.5,2);
 Serial.println (robotArdi.distance());
}
```

Обратите внимание, что вместо снижения скорости при повороте мы её увеличиваем, поэтому соответствующий коэффициент отрицательный (`robotArdi.moveWall(14,0.01,2,-0.5,2);`).

## Задание 87

Внутренний угол обойти не так сложно, если использовать ИК-дальномер. Вначале необходимо немного изменить программу, добавив возможность работы с этим дальномером.

Изучите пример программы, реализующей движение робота вдоль стенки с обходом внешних углов. При этом во внутреннем углу робот останавливается. Видео с движением робота, который останавливается во внутреннем углу, можно посмотреть по ссылке: https://youtu.be/UV-hC9CVh8o.

```
#define DIR_M1              4 //правый мотор
#define SPEED_M1            5
#define SPEED_M2            6
#define DIR_M2              7 //левый мотор
#define LED_PIN            13
#define IR_SENSOR_PIN      A5
class Walker {
 public:
  Walker (byte triggerPin, byte echoPin, byte buttonPin);
   void  startDistance ();
   void  stopMove();
   void  waitButton ();
```

```
    float distance ();
    int   IRdistance ();
    void  motors (byte motorM1, byte motorM2);
    void  moveWall(float Kp, float Ki, float Kd,
                   float brakeGain, int brake);
 private:
    byte BUTTON_PIN, ECHO_PIN, TRIG_PIN;
    byte START_POWER_M1, START_POWER_M2;
    int setPoint;
    float oldDistance, errorSum=0, errorOld=0;
};
Walker :: Walker(byte triggerPin, byte echoPin,
                 byte buttonPin)   {...}
void Walker :: stopMove () {...}
void Walker :: waitButton () {...}
void Walker :: startDistance () {...}
float Walker :: distance(){...}
void Walker :: motors (byte motorM1, byte motorM2) {...}
void Walker :: moveWall(float Kp, float Ki, float Kd,
                        float brakeGain, int brake) {...}

int Walker :: IRdistance () {
  int summValue=0; int filter [20];
  for (int i=0;i<=19;i++)
    filter[i] = analogRead(IR_SENSOR_PIN);
  for (int j=19;j<=1;j--) {
   for (int i=1;i<=j;i++) {
    if (filter[i-1] > filter [i]) {
      int tmp = filter[i-1];
      filter[i-1] = filter[i];
      filter[i]=tmp;
    }
   }
  }
  for (int i=5;i<=14;i++)
     summValue = summValue + filter[i];
  summValue = summValue/10;
  return int (12241*pow(summValue,-1.147));
}

Walker robotArdi (10,11,8);
void setup() {
 robotArdi.motors (240,180);
```

```
 robotArdi.waitButton(); robotArdi.startDistance();
}
void loop() {
  while (robotArdi.IRdistance() > 10) {robotArdi.
moveWall(14,0.01,2,-0.5,2);}
  robotArdi.stopMove();
  while (true);
}
```

## Задание 88

Напишите программу, по которой робот сможет двигаться вдоль любой стенки (с внутренними и внешними углами). Например, создайте трассу, как на рисунке 125.

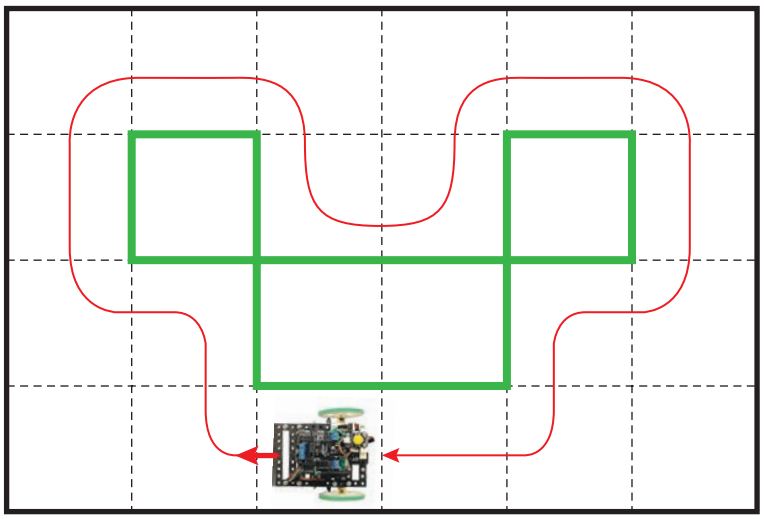



**Рис. 125.** Схема движения робота

## Задание 89

Составьте для робота лабиринт. Напишите программу, по которой робот сможет пройти этот лабиринт.

# Глава 16
# МИР ПРОФЕССИЙ

## 16.1. Робототехника. Мир новых профессий

Если вы хотя бы иногда смотрите новостные каналы, то могли обратить внимание на то, что в стране из года в год усиливается кадровый голод. Очень не хватает высококлассных мотивированных специалистов во всех областях. Причин много, одна из которых — постоянно растущие требования к этим специалистам.

Последние годы лидеры на рынке труда по-прежнему — ИТ-специалисты и инженеры. В 2019 году, например, нехватку программистов испытывали почти 40% компаний, при этом на одного разработчика часто приходится от двух до семи предложений о работе.

За очень хороших специалистов идёт серьёзная борьба. Гибкий график работы — это уже почти стандарт, а работодатели готовы создавать идеальные условия для эффективной работы.

Например, в одном очень крупном финансовом учреждении ИТ-специалисты, предпочитающие свободную форму одежды, обедая рядом с менеджерами в костюмах, чувствовали себя некомфортно. Что сделал банк? Построил отдельную столовую. Ещё пример: в другом банке программистам сделали отдельные шумоизоляционные кабинки, чтобы они... спокойно совершали личные звонки. Таких историй тысячи по всей стране. Приведём пример и из сферы образования в регионах: когда в одном из северных городов компании малого бизнеса узнали, чем со школьниками занимается преподаватель робототехники, то построили ему отдельное небольшое здание на территории школы, чтобы работал ещё эффективнее.

### Задание 90

Посмотрите небольшой видеоролик «Профессии будущего: Робототехника» (https://youtu.be/swS4j-n37IQ). Обсудите его в классе.

www

### Задание 91

Найдите в Интернете «Атлас новых профессий». Посмотрите, в каких профессиях потребуются знания и умения по робототехнике и смежным дисциплинам. Обсудите в классе.

www

Стоит отметить, что среди достаточно большого списка требований к высокооплачиваемым инженерам и ИТ-специалистам — знание английского языка: и технического, и разговорного.

Итак, давайте посмотрим на три основных тренда на мировом рынке труда:

- *автоматизация* (до 50% всех рабочих мест могут быть автоматизированы в течение ближайших десятилетий);
- *развитие искусственного интеллекта* (например, около 250 тыс. госслужащих в Великобритании к 2030 году могут быть заменены искусственным интеллектом);
- *цифровизация* (почти 60% финансовых организаций в мире включили цифровую трансформацию в основу стратегии своего бизнеса).

Благодаря автоматизации ежегодно происходят события, которые полностью изменяют представление о той или иной области. Например, в 2018 году произошла революция в сфере розничных продаж. Вы же слышали о сети магазинов, которые работают без касс, без продавцов и без банковских карт? Amazon Go! Именно так называются магазины. Однако вы же понимаете, что оборудование, установленное в таких магазинах, кто-то должен обслуживать и настраивать. Представляете требования к таким специалистам? А их уровень вознаграждения за труд?

### Задание 92

Посмотрите небольшой видеоролик о магазинах Amazon Go (https://youtu.be/NrmMk1Myrxc). Обсудите увиденное в классе.

### Задание 93

Найдите в «Атласе новых профессий» раздел «Робототехника и машиностроение» и ознакомьтесь с интереснейшими новыми профессиями в этой области.

Да, устаревшие профессии постепенно ликвидируются, людей заменяют роботы и автоматы. Или вы всё ещё пользуетесь услугами касс по продаже железнодорожных или авиабилетов? Можно даже предположить, что когда-то останутся лишь профессии, связанные с роботами. Конечно, понадобятся люди, которые будут изобретать, программировать и ремонтировать роботов. Будут нужны и операторы робототехнических систем. Выбрав профессию, связанную с робототехникой, можно быть уверенным, что со временем она не исчезнет.

Никто не знает, каким будет будущее, но все понимают, что XXI век — это про робототехнику, искусственный интеллект и интернет-технологии. Поэтому есть смысл получать фундаментальное техническое образование, развивать интерес к программированию. А актуальные технологии вы будете осваивать уже ближе к моменту непосредственной работы.

Высшие учебные заведения предлагают широкий выбор вариантов освоения профессий будущего. Множество вузов России готовят студентов по направлению «Мехатроника и робототехника» и, конечно, во всех федеральных университетах такую специальность получить можно. Кроме того, ежегодно в вузах появляются новые профили обучения, связанные с робототехникой.

Сейчас в вузах есть много направлений, профилей и программ обучения, напрямую связанных с программированием систем управления. Ниже представлены некоторые из них.

- Мехатронные модули робототехнических комплексов.
- Конструирование и технология электронных средств.
- Нанотехнологии и микросистемная техника.
- Электроника и робототехника.
- Программирование и электроника информационных систем.
- Встраиваемые системы управления.
- Электроника и наноэлектроника.
- Проектирование и технология электронно-вычислительных средств.
- Встраиваемые системы реального времени.
- Информационные технологии в проектировании встраиваемых систем управления технологическими процессами.
- Информатика и вычислительная техника.
- Информационные системы и технологии.
- Прикладная информатика.
- Программная инженерия.
- Информационная безопасность.
- Информационная безопасность телекоммуникационных систем.
- Инфокоммуникационные технологии и системы связи.
- Биотехнические системы и технологии.
- Медицинская кибернетика.
- Автоматизация технологических процессов и производств.
- Управление в технических системах.

Список постепенно расширяется. Выбирая направления, связанные с инженерией, вы уверенно будете смотреть в завтрашний день.

- Проектирование роботов и робототехнических комплексов под различные нужды:
  - для медицины;
  - для промышленности;
  - для домашнего хозяйства;
  - для детей.
- Разработка и подбор материалов для элементов робототехнических устройств.
- Разработка дизайна роботизированных систем с учётом необходимости их взаимодействия с людьми и окружающей средой.

- Разработка систем обучения роботов.
- Проектирование и управление сложными робототехническими комплексами.

Вырабатывайте в себе любопытство, настраивайтесь, что придётся учиться чему-то новому всю жизнь, а после получения диплома всё только начнётся. Какую бы профессию вы ни выбрали, нужно радостно адаптироваться к изменениям.

Надеемся, что вы выберете профессию, которая будет крайне востребована в связи с бурным развитием робототехники, приборостроения, информационной безопасности.

Удачи вам на этом пути!