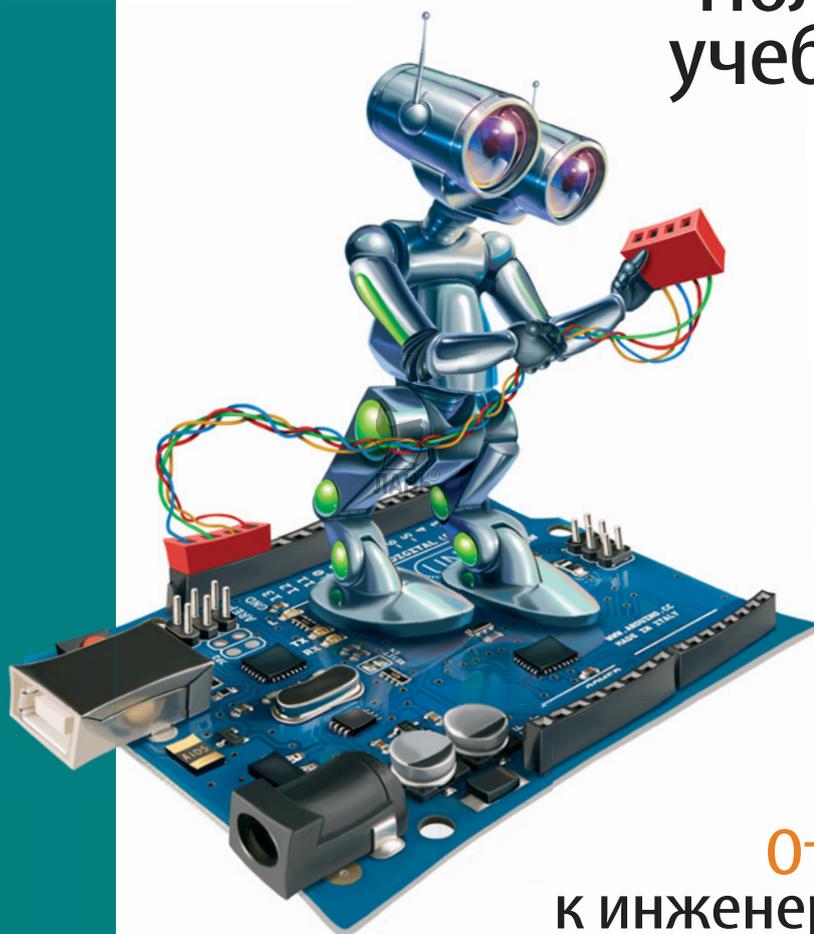


Р • О • Б • О • Ф • И • Ш • К • И

Arduino®

Полный
учебный
курс



От игры
к инженерному
проекту



Лаборатория
ЗНАНИЙ

Arduino®

Полный учебный курс



@CODELIBRARY_IT

От игры
к инженерному
проекту

Электронное издание



Лаборатория знаний
Москва
2020

УДК 373.167
ББК 32.816; 32.97
А79

Серия основана в 2016 г.

Авторский коллектив:

А. А. Салахова, О. А. Феоктистова, канд. пед. наук
Н. А. Александрова, канд. пед. наук М. В. Храмова

А79 **Arduino®**. Полный учебный курс. От игры к инженерному проекту / А. А. Салахова, О. А. Феоктистова, Н. А. Александрова, М. В. Храмова. — Электрон. изд. — М. : Лаборатория знаний, 2020. — 178 с. — (РОБОФИШКИ). — Систем. требования: Adobe Reader XI ; экран 10". — Загл. с титул. экрана. — Текст : электронный.

ISBN 978-5-00101-886-5

Предлагаемый полный курс познакомит вас с особенностями аппаратного обеспечения и программирования микроконтроллера Arduino Uno®. Рассмотрены графические языки программирования Snap! и ArduBlock, текстовый язык Wiring и производственные языки. Кроме того, вы научитесь читать, составлять и собирать действующие схемы из электронных компонентов. В заключение мы расскажем вам, как правильно оформлять инженерные проекты.

Материал излагается в формате последовательно выстроенных тем, сопровождаемых вопросами, практическими заданиями и проектами.

Проектная часть курса может быть расширена серией книг «РОБОФИШКИ. Конструируем роботов на Arduino®» издательства «Лаборатория знаний».

Для детей среднего и старшего школьного возраста для применения в урочной и внеурочной деятельности и технического творчества дома.

УДК 373.167
ББК 32.816; 32.97

Деривативное издание на основе печатного аналога: Arduino®. Полный учебный курс. От игры к инженерному проекту / А. А. Салахова, О. А. Феоктистова, Н. А. Александрова, М. В. Храмова. — М. : Лаборатория знаний, 2020. — 175 с. : ил. — (РОБОФИШКИ). — ISBN 978-5-00101-250-4.

12+

В соответствии со ст. 1299 и 1301 ГК РФ при устранении ограничений, установленных техническими средствами защиты авторских прав, правообладатель вправе требовать от нарушителя возмещения убытков или выплаты компенсации

ISBN 978-5-00101-886-5

© Лаборатория знаний, 2020

Оглавление

От авторов	5
Глава 1. Прототипирование в робототехнике	7
1.1. Микроконтроллер как основной компонент робота.	7
1.2. Робототехнические платформы открытого типа. Прототипирование	9
Глава 2. Знакомство с Arduino	15
2.1. Платформа Arduino	15
2.2. Контакты платы Arduino Uno	17
2.3. Макетные платы	24
Глава 3. Программное обеспечение Arduino	27
3.1. Среда разработки Snap4Arduino (S4A)	27
Установка S4A	28
3.2. Среда разработки Arduino IDE	32
3.3. Работа в Arduino IDE	33
3.4. Альтернативное программное обеспечение для Arduino	38
Глава 4. Периферия Arduino	47
4.1. Виды периферийного оборудования	47
4.2. Однокомпонентные устройства	48
4.3. Простые упражнения для Arduino и S4A	52
Мигание светодиодом	52
Маячок с убывающей яркостью	54
Светофор, срабатывающий по кнопке	56
Управление движением с помощью потенциометра. Упражнение «Краб»	59
Управление яркостью светодиода с помощью потенциометра	65
Работа с фоторезистором. Упражнение «Робот» ...	67
Терменвокс	73
Ночной светильник	74
RGB-светодиод	76
Сахарница	79
4.4. Модули и сложные датчики	84
4.5. Применение модулей и S4A	88
Сигнализатор затопления	88
Сервопривод	92
4.6. Платы расширения	94

Глава 5. Язык программирования Wiring	103
5.1. Введение в язык Wiring	103
5.2. Программы на языке Wiring: библиотеки и переменные	104
5.3. Основные функции в языке Wiring	107
Функция <i>setup()</i>	108
Функция <i>loop()</i>	108
5.4. Функции Wiring и ШИМ. Работа со звуками	113
5.5. Графические блоки и код в ArduBlock	117
Подключение датчика уровня жидкости	119
Управление потенциометром	121
Работа с LCD-дисплеем	122
5.6. Практические задания по Wiring	127
Фоторезистор	128
Дальномер	129
Шаговый двигатель	131
Датчик температуры и влажности DHT11	134
5.7. Дополнительные задания для самостоятельной работы	135
Шар с предсказаниями	135
Усложнение задачи 1	136
Автоповорот	136
Реклама «Бегущая строка»	137
5.8. Проект «Развитие моторики»	138
Дополнительные задания	150
Глава 6. Применение робототехники в различных сферах ..	151
6.1. Робототехника в современном мире	151
6.2. Arduino и производственные языки	156
6.3. Оформление робототехнических проектов	164
Этап 1. Постановка и осознание проблемы	169
Этап 2. Выбор стратегии решения	169
Этап 3. Требования и ограничения для выбранного решения	170
Этап 4. Формулирование концепции решения	171
Этап 5. Моделирование архитектуры	171
Этап 6. Ресурсная база	172
Этап 7. Техническое задание	173
Заключение	174

ОТ АВТОРОВ

Уважаемые ребята!

Вы держите в руках учебное пособие по курсу, посвященному робототехнике. Эта область технического творчества (именно творчества!) вам знакома по занятиям с робототехническими конструкторами в предыдущих классах. Но уже тогда вы наверняка задумывались: что же дальше? Возможно, вам казалось, что собранные вами роботы больше похожи на увлекательную игру? Поздравляем! Это значит, что ваше мышление — взрослое. В этом курсе мы постараемся ответить на вопросы о том, как робототехника в школе и ваши проекты связаны с робототехникой на предприятиях и инфраструктурой города.

Взросление — это ответственность. Настало время изучить робототехнику как серьезную область, а не только как инструмент моделирования. Вы узнаете, чем отличается открытая архитектура приложений и аппаратного обеспечения от закрытой. Вы познакомитесь с принятыми стандартными обозначениями в электрических схемах и других документах, сможете представлять свои робототехнические проекты в формате, понятном любому инженеру мира, — в соответствии с международными стандартами. Полезным на вашем творческом пути будет и ознакомление с жизненным циклом проекта. Вы научитесь мыслить так, как мыслят инженеры.

Еще один важный навык взрослого человека — умение адаптироваться и выделять основную информацию. Именно он помогает профессиональным программистам создавать программы высокого качества на различных языках программирования. Вы научитесь не писать на определенных языках, а мыслить алгоритмами и условиями. Алгоритмическое мышление позволит впоследствии быстро выучить любой язык программирования, не важно, будет он графическим или текстовым.

Робототехника — важная составляющая жизни современного высокотехнологичного общества. Даже если вы не свяжете свою будущую профессию непосредственно с робототехникой, вы будете сталкиваться с ней и ее принципами повсеместно. Пособие, которое вы держите в руках, призвано помочь вам чувствовать себя комфортно и легко адаптироваться к быстро развивающемуся высокотехнологичному миру.

Для того чтобы вам было удобнее, мы создали систему сигналов-подсказок. На страницах вам встретятся следующие условные обозначения.

 **Запомните** — самые важные понятия, формулировки которых следует выучить наизусть. Они будут использоваться в дальнейшем и являются фундаментальными.

 **Вопросы** — теоретические задания для проверки знаний после прочтения каждого раздела. В них спрашивается только о том, что было изложено в этом разделе.

 **Практические задания** — практические вопросы и простые упражнения для самостоятельной работы по изученной теме. Они требуют дополнительного поиска информации или работы с электронными компонентами.

Это интересно! — любопытные факты, на которые стоит обратить внимание. Они могут пригодиться вам в повседневной жизни и расширить вашу эрудицию.

В книге вы также найдете проект для самостоятельного выполнения. Работа над проектом потребует обобщения навыков и знаний, полученных на основе нескольких тем. В большинстве случаев проекты похожи на лабораторные работы, однако в конце имеют открытую проблему (вопрос), которую необходимо разрешить самостоятельно, применяя творческие решения. Работа над проектом займет больше времени, чем выполнение практического задания.

Мы надеемся, что курс будет не только полезным, но и увлекательным. Если вам потребуется какая-либо помощь, не стесняйтесь спрашивать своих педагогов или писать нам по адресу arduino@pilotlz.ru

И, главное, никогда не сомневайтесь в себе и своих силах! Знаете ли вы, что главная задача инженера — решить поставленную задачу в условиях ограниченных ресурсов. Ваши ресурсы — это в первую очередь ваши знания и время, а остальное легко приобретается в магазине или уже имеется под рукой. Иногда не мешает заглянуть в заветный уголок дома, где хранится всякий хлам — старая электроника, деревянные или пластиковые ящики и многое другое. Остается только включить инженерную фантазию!

Успехов, дорогие друзья!

Глава 1. Прототипирование в робототехнике

1.1. Микроконтроллер как основной компонент робота

Основным компонентом робота является его «мозг» — *микроконтроллер*. Микроконтроллеры применяются всюду для решения совершенно непохожих задач: от автоматизации рутинных процессов в сельском хозяйстве, обслуживании и эксплуатации зданий, электрификации до сложных автономных систем жизнеобеспечения бункеров. Везде, где автоматизация процесса не требует огромных вычислительных мощностей, применяют компьютеры на базе микроконтроллеров. Основными причинами являются их малое энергопотребление, компактные размеры и простота подключения датчиков для взаимодействия с окружающей средой и обработки показаний. Когда необходимо спроектировать систему, решающую многокомпонентную, но все же *единственную* задачу, микроконтроллер становится эффективным и одновременно недорогим решением. Ранее вы изучали робототехнические конструкторы, блок управления которых содержал микроконтроллер и некоторые периферийные устройства.

Существуют две большие группы плат с микроконтроллерами. К первой группе относятся *отдельные платы* с распаянными контактами для подключения периферийных устройств и подачи питания, например Arduino Mega (рис. 1.1). На такой плате в ряде случаев дополнительно устанавливают минимальный интерфейс для передачи и приема информации от внешних устройств, например разъем microUSB. Кроме того, он может одновременно быть и питающим. Платы Arduino используют напряжение 5 В, аналогичное компьютерным стандартам. Именно такое напряжение подается через USB-разъем от компьютерного блока питания.

Платы имеют небольшие размеры, но их можно сделать еще меньше с помощью *внешнего программатора* — устройства для интерпретации микропроцессором данных от компьютера и обратной трансляции команд внешним устройствам. Например, в программаторе нуждается Arduino Pro Micro. Датчики и дополнительные устройства подключаются к платам проводами.

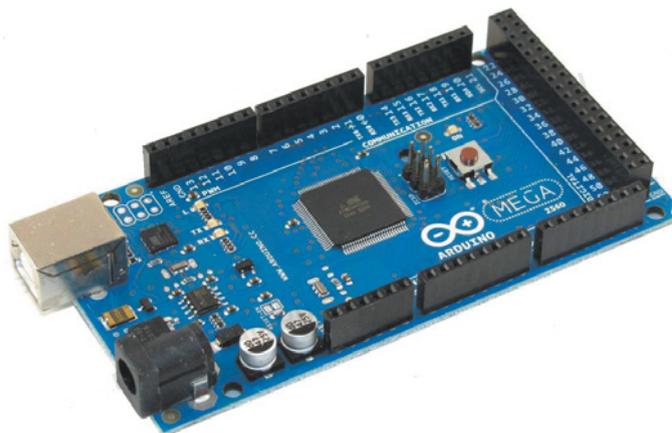


Рис. 1.1. Плата Arduino Mega

Вторая группа плат — это *одноплатные компьютеры*, например Raspberry Pi (рис. 1.2). Как ясно из названия, в этом случае на одной плате располагается не только микроконтроллер, но и часть периферийного оборудования: модули связи, светодиоды и т. д. Благодаря этому одноплатные компьютеры позволяют реализовывать множество разных проектов без каких-либо дополнительных соединений. В результате экономится место и по-



Рис. 1.2. Одноплатный компьютер Raspberry Pi 2 Model B

вышается скорость обмена информацией между компонентами устройства.

Плата микроконтроллера — это основа робототехнической платформы. Платы обеих групп не имеют закрытого корпуса в отличие от готовых программируемых блоков многокомпонентных робототехнических конструкторов, например LEGO NXT, LEGO Mindstorms EV3, VEX.

Вопросы

1. Перечислите плюсы применения микроконтроллеров на производстве или, например, при обслуживании зданий.
2. Назовите основные группы плат микроконтроллеров и их особенности.
3. Чем отличается робот, автоматизирующий процессы обслуживания или производства, от производственного автомата? Имеют ли значение для каждого из них постоянно изменяющиеся показатели окружающей среды? Найдите и выпишите определения робота и автомата.

Практическое задание

Зайдите на сайты с проектами на робототехнических платформах Arduino и Raspberry Pi. Познакомьтесь с проектами и выпишите пять идей, которые вам особенно понравились. Чем они полезны?

Сайты с проектами:

1. Каталог устройств и поделок на Arduino (<http://arduino-projects.ru/>).
2. Проекты для Arduino (<http://arduino-diy.com/>).
3. Робототехнические проекты (<https://diyhacking.com/>).
4. Сообщество Hackster.io (<https://www.hackster.io/>).

1.2. Робототехнические платформы открытого типа. Прототипирование

Что же характерно для *робототехнических платформ открытого типа* (например, Arduino Uno, Raspberry Pi или STM32)? Во-первых, в уже знакомых вам робототехнических конструкциях датчики были заключены в закрытые корпуса (как и сам блок управления) и имели ограниченные (количеством деталей

в наборе) вариации сборки. Доступа непосредственно к электронным схемам не было. В практической робототехнике, применяемой на производстве и в других сферах деятельности человека, количество подключаемых готовых компонентов к микроконтроллеру почти не ограничивается. Основная плата, на которой он расположен, должна иметь возможность бесконечного наращивания компонентов без необходимости полного перемонтажа.

Вторая особенность платформ, которые будут рассмотрены в этой главе, вынесена в заголовок и напрямую связана с возможностью быстрого наращивания компонентов. **Прототипирование** (от англ. *prototyping*) — это процесс быстрой «черновой» сборки устройства.

Устройство, собранное в процессе прототипирования, называется **прототипом**. Прототипы изготавливаются штучно и нужны для тестирования и совершенствования идеи перед запуском в массовое производство.

Процесс создания прототипа состоит из четырех шагов.

1. Определение начальных требований.
2. Разработка первого варианта прототипа, который содержит только пользовательский интерфейс системы.
3. Изучение прототипа заказчиком и конечным пользователем, получение обратной связи о необходимых изменениях и дополнениях.
4. Переработка прототипа с учетом полученных замечаний и предложений.

Зачастую при прототипировании соединения выполняются проводами без пайки и отсутствует внешний корпус. На такую сборку требуется минимальное количество времени и усилий при полной работе итоговой системы. Платформы, поддерживающие прототипирование, позволяют собрать достаточно сложные и многокомпонентные схемы, которые в то же время будут обладать возможностью интеграции и (или) взаимодействия с внешним оборудованием.

Третьей важной особенностью является процесс моделирования. Наборы с ограниченной элементной базой, основанные на собственном стандарте физического интерфейса (например, крепления LEGO), не позволяют переносить и применять собранное устройство в реальных условиях.

Конструкторы, которые были рассмотрены ранее в курсе технологии или на уроках робототехники, предполагали моделирование в искусственной среде. Проекты, реализуемые на

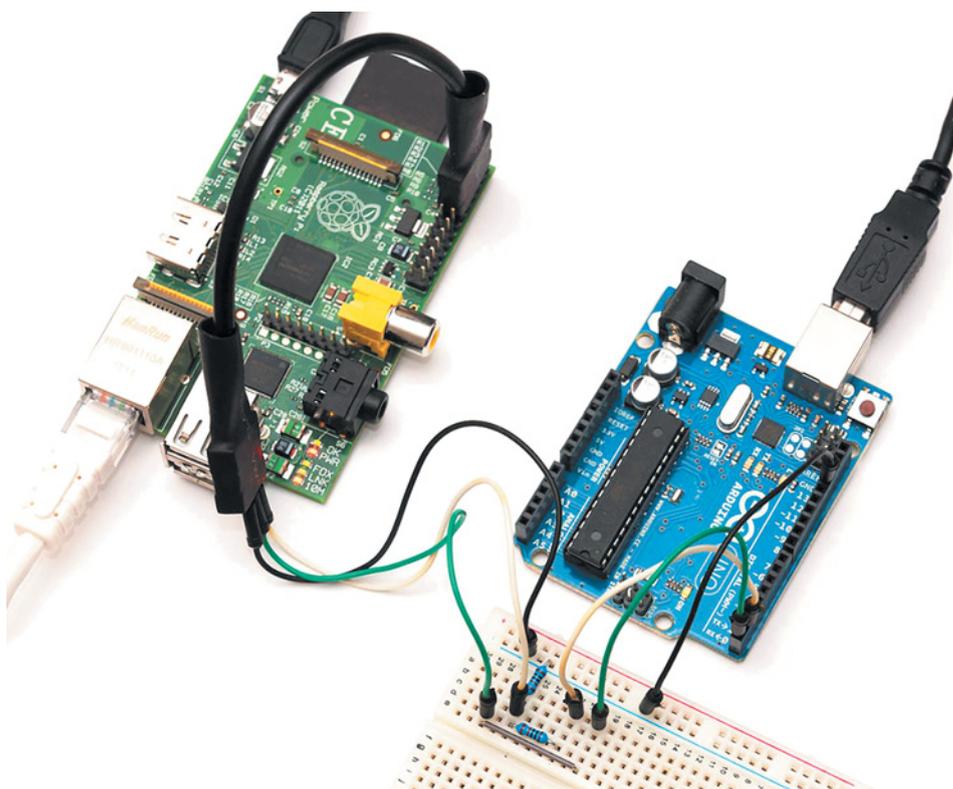


Рис. 1.3. Соединение Raspberry Pi с Arduino Uno через протокол I²C

платформах с прототипированием, носят *прикладной характер* и затрагивают неигровые ситуации.

Применение универсальных интерфейсов управления, питания (например, стандартных для компьютерной техники напряжений 5 В и 3,3 В), передачи данных и крепления позволяет соединять в проектах различные дополнительные компоненты из реального, а не моделируемого мира (рис. 1.3).

Например, установка на прототипе реле (рис. 1.4) позволяет управлять светильниками, электрическими чайниками, кофеварками, вентиляторами, подключенными к обычной бытовой сети. Поэтому платформы, поддерживающие прототипирование, применяют в качестве управляющих устройств в проектах «умного дома» и даже на производстве. Кроме того, универсальность интерфейсов дает возможность приобретать *детали разных производителей*, что практически невозможно для многокомпонентных наборов с собственным интерфейсом («ТРИК», LEGO Mindstorms



Рис. 1.4. Реле

EV3 и др.). Возможность применения дополнительных компонентов позволяет выбирать для достижения целей проекта самые точные и специфические датчики, например датчик количества газа бутана в воздухе или сканер NFC- и RFID-меток. На основе платформ Raspberry Pi или Arduino можно самостоятельно собрать даже мобильный телефон.

Четвертая особенность этих платформ — в используемых языках программирования. Наборы, нацеленные на решение задач, моделируемых в искусственной среде (игровых), в основном в стандартных средах («из коробки») используют языки программирования, носящие сугубо учебный характер. К ним относится графическое программирование на Scratch, LabView в среде LME и т. д. Робототехнические платформы, поддерживающие прототипирование, о которых пойдет речь далее, предусматривают *использование языков объектно ориентированного или функционального программирования*, применяемых при написании реальных программных оболочек устройств и для управления компонентами сложных систем. Такими языками являются C и его модификации, например C++ и Wiring, JavaScript, Java, Python и Assembler. Однако Arduino, являясь платформой открытого типа, также поддерживает графические языки программирования, и именно с них мы вскоре начнем знакомство с платой Arduino.

Таким образом, переход на робототехнические платформы открытого типа позволяет значительно расширить возможности конструируемых роботов, использовать их для решения прак-

тических задач и приблизиться к изучению реальных производственных робототехнических комплексов.

Вопросы

1. Что такое прототипирование и прототип?
2. Перечислите характерные особенности робототехнических платформ открытого типа.

Запомните

- ◆ Микроконтроллер ◆ Внешний программатор
- ◆ Прототипирование ◆ Прототип

Практическое задание

Найдите информацию о различных платформах открытого типа. Заполните таблицу.

Платформа	Стандартный язык программирования	Чем интересна?
1.		
2.		
3.		
4.		
5.		

Это интересно!

Название «Arduino» согласно официальной документации от разработчиков должно использоваться лишь в США. В остальном мире платформа обязана называться «Genuino». Меняются и логотипы. Несмотря на пожелания разработчиков, по всему миру плата все равно известна как Arduino.



Платформа Raspberry Pi, получившая широкое применение, поддерживает установку операционных систем Linux, Windows 10 и Android. На ее основе можно собрать настоящий смартфон с дополнительными функциями, которых нет даже у самых передовых гаджетов. Например, такой смартфон сможет одновременно служить мультиметром и режущим фанеру лазером. На основе платы Raspberry Pi можно собрать настольный компьютер, подключив к ней монитор, мышь и клавиатуру (рис. 1.5).

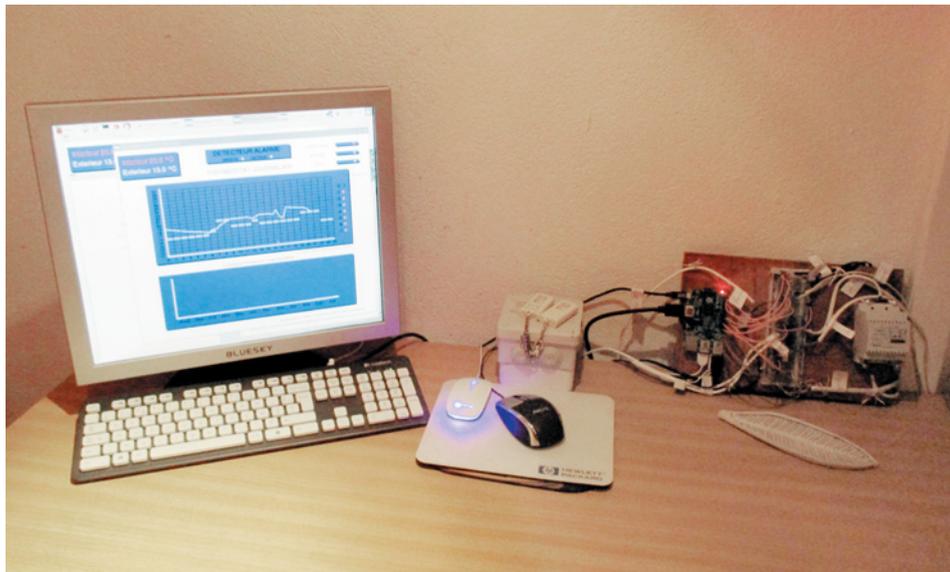


Рис. 1.5. Настольный компьютер на основе платы Raspberry Pi

Глава 2. Знакомство с Arduino

2.1. Платформа Arduino

Arduino — это платформа, состоящая из аппаратно-программных средств для построения систем автоматики и робототехники. *Аппаратной основой* платформы является плата с размещенным на ней микроконтроллером и разведенными (т. е. распаянными) по фиксированной электрической схеме контактами. *Программная часть* представлена средой разработки Arduino IDE.

Платформа обладает несколькими особенностями, делающими ее популярной во всем мире.

Аппаратная часть платформы имеет *открытую архитектуру*. Архитектура аппаратной части устройства называется открытой, если опубликована ее спецификация, т. е. подробное описание, составляющие, схема и применение каждого компонента. Спецификация позволяет любому производителю создать копию продуктов для платформы, тем самым делая их более доступными. Кроме того, появляется возможность создавать улучшенные, более эффективные версии плат и модулей или новые совместимые устройства. Все это влияет на цены, удерживая их на уровне, доступном широкому кругу пользователей. И наконец, открытая спецификация — это возможность самостоятельного ремонта при должных навыках и умениях.

Оригинальные платы Arduino изготавливаются четырех типов в зависимости от габаритов и количества контактов, доступных для подключения (входы и выходы). Приведем самые распространенные из них.

1. *Arduino Uno*: стандартный размер, 20 контактов (рис. 2.1).
2. *Arduino Mega*: увеличенный размер, 70 контактов (рис. 2.2).
3. *Arduino Nano*: уменьшенный размер, 22 контакта (рис. 2.3).
4. *Arduino Micro*: миниатюрная версия, 20 контактов (рис. 2.4), отсутствует DATA-интерфейс USB (требуется внешний программатор, USB обеспечивает только питание платы).

Кроме основных плат с размещенным микроконтроллером, существуют дополнительные платы. Они расширяют возможности Arduino добавлением новых типов интерфейса или дополнитель-

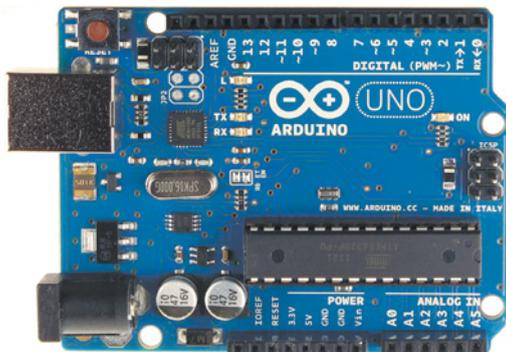


Рис. 2.1. Плата Arduino Uno

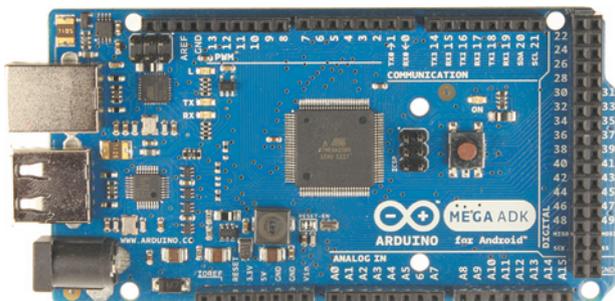


Рис. 2.2. Плата Arduino Mega

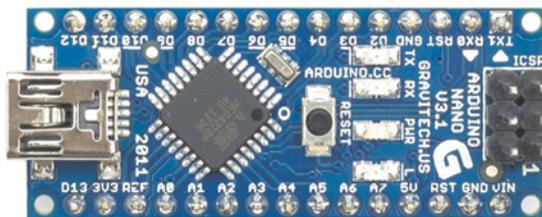


Рис. 2.3. Плата Arduino Nano

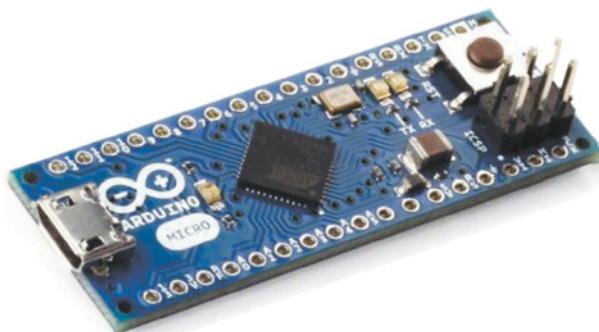


Рис. 2.4. Плата Arduino Micro

ных микросхем. Иногда это зависимые контроллеры, обеспечивающие обмен данными по другим протоколам и стандартам, чем основной микроконтроллер. Такие платы называются *платами расширения* или *щитами (Shield)*. Они устанавливаются непосредственно на плату Arduino Uno или Arduino Mega, занимая все или почти все контакты основной платы, которые дублируются на плате расширения. Подробнее с платами расширения вы познакомитесь в разделе 4.6.

Вопросы

1. Что такое Arduino? Из каких частей состоит данная платформа?
2. Что такое открытая архитектура аппаратной части? Чем полезна открытая архитектура для конечного пользователя?
3. Какие бывают форматы Arduino? Перечислите особенности каждого вида плат.

Это интересно!

Название платформы Arduino Uno связано с местом ее рождения. Официальные платы разрабатываются и производятся в Италии. В переводе с итальянского языка uno означает «один». Arduino Uno — эталонная плата, базовая версия для остальных модификаций.

2.2. Контакты платы Arduino Uno

В книге здесь и далее рассматривается плата *Arduino/Genuino Uno*. Это самая распространенная плата стандартного размера. Она управляется микроконтроллером ATmega328, имеющим характеристики: 2 Кб оперативной памяти, 1 Кб памяти на энергонезависимом носителе (EEPROM, аналог BIOS у персонального компьютера), 32 Кб памяти для загрузки программ, из которых 0,5 Кб занято самим загрузчиком. Входное напряжение от источника питания составляет 7–12 В. Подключенные датчики и модули управляются током 40 мА для основного рабочего напряжения 5 В и 50 мА для дополнительного напряжения 3,3 В. Тактовая частота микропроцессора равна 16 МГц — это почти в 4 раза меньше, чем тактовая частота процессоров первого поколения Pentium для персональных компьютеров!

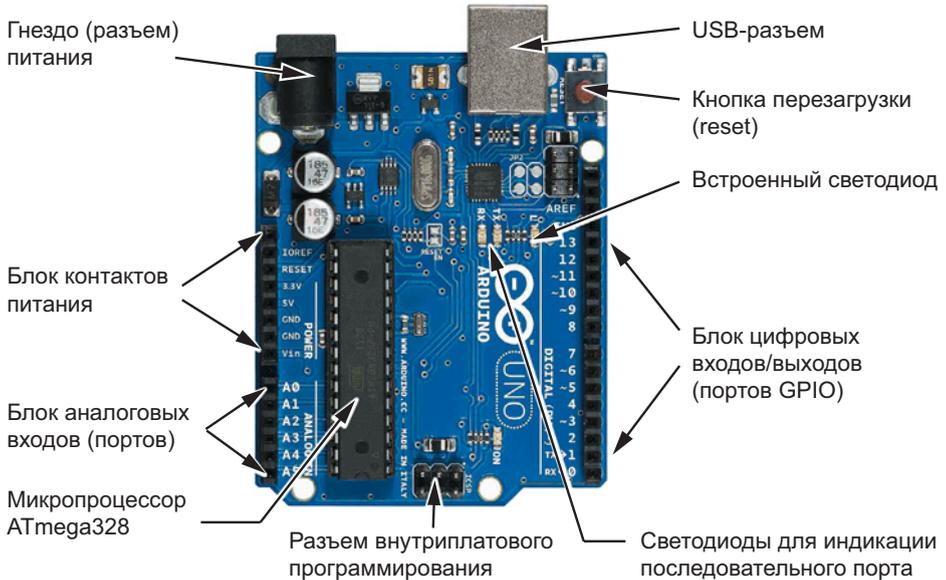


Рис. 2.5. Контакты платы Arduino Uno

Микроконтроллер предназначен решать узкоспециализированные задачи, обычно не требующие сложных вычислений, поэтому характеристик ATmega328 достаточно для эффективного управления автоматизацией и взаимодействия с внешней средой с помощью датчиков.

На основной плате (рис. 2.5) расположены три группы *контактов (портов)*:

- 1) контакты питания платы;
- 2) контакты управления: цифровые входы/выходы (Digital Input/Output или GPIO — General Purpose Input and Output);
- 3) аналоговые входы (Analog Input).

Контакты питания расположены в отдельном блоке, находящемся неподалеку от гнезда внешнего питания, подключаемого к плате с помощью стандартного штекера диаметром 2,1 мм.

В блок входят:

- *IOREF* — с этого вывода плата расширения получает информацию о рабочем напряжении Arduino Uno (5 или 3,3 В). На основе этой информации плата расширения выбирает необходимый источник питания или режим работы преобразователей уровней (5 или 3,3 В соответственно);
- *RESET* — с этого вывода платы расширения получают сигнал на сброс состояния, т. е. перезагрузку, чтобы старые

и уже ненужные данные не превращались в компьютерный мусор при последующих запусках программы;

- *3.3V* — вывод напряжения 3,3 В от стабилизатора напряжения на плате Uno. Используется для питания датчиков или подключения шины питания макетной платы.

Шина — это электронный канал, связывающий несколько входов и выходов. Физически обычно представляет собой медную полосу, к которой подведены несколько выводов-ответвлений. Максимальный ток, потребляемый датчиками от этого вывода, составляет 5 мА;

- *5V* — вывод напряжения 5 В от стабилизатора напряжения на плате Uno. Чаще всего используется для питания датчиков или подключения шины питания макетной платы. Максимальный допустимый ток, потребляемый датчиками от этого вывода, составляет 40 мА;
- *GND* — земляные выводы. Используется стандартное обозначение из электроники. *Земля (GND, ground)* — это отрицательный вывод однополярного источника питания. В роли положительного выступает вывод *5V* или *3,3V*, т. е. *шина питания*;
- *VIN* — одновременно вход и выход внешнего питания. С него можно получить дополнительный ток или подать питание на другие устройства с тем напряжением, которое приходит на плату без преобразования в 5 В, например в диапазоне 12 В для драйверов моторов.

Arduino Uno имеет 20 контактов для *управления и получения информации* с датчиков, модулей и сервоприводов. Из них 14 — это цифровые входы/выходы, остальные шесть — аналоговые входы. Каждый из цифровых контактов может работать в качестве входа или выхода в зависимости от потребностей проекта. Уровень напряжения на них ограничен 5 В, а максимальная величина тока — 40 мА.

Рядом с блоком контактов питания расположен блок аналоговых входов. Они обозначены символами от А0 до А5. *Аналоговый сигнал* (рис. 2.6) — это непрерывно изменяющееся напряжение, которое поступает на вход с датчиков, считывается платой и преобразуется с помощью встроенного АЦП (аналого-цифрового преобразователя). Он может быть представлен 1024 уровнями, т. е. имеет вид 10-битного числа. Физически это означает изменение напряжения в диапазоне от 0 до 5 В.

Цифровой сигнал принимает только два значения (рис. 2.7): логическую единицу (HIGH, 5 В) и логический ноль (LOW, 0 В).

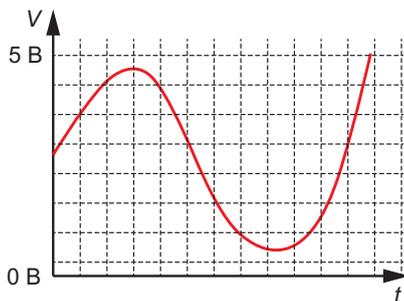


Рис. 2.6. Аналоговый сигнал

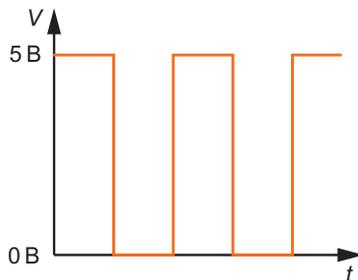


Рис. 2.7. Цифровой сигнал

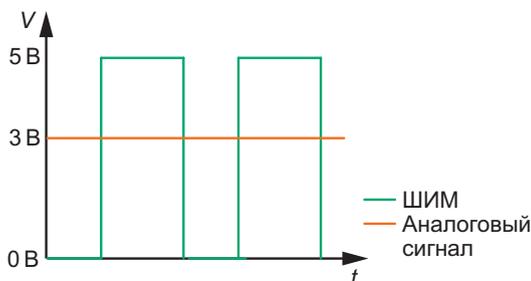


Рис. 2.8. График представления аналогового сигнала с помощью ШИМ

В блоке *цифровых контактов* расположены следующие группы контактов, различающиеся по дополнительным функциям.

1. Последовательный интерфейс. Используется для последовательного соединения нескольких плат Arduino или связи с другими микроконтроллерами:

- RX←0 — для приема данных платой Arduino Uno;
- TX→1 — для передачи данных на другую плату.

2. Управление внешними прерываниями — контакты 2 и 3. Значения напряжения на них служат в качестве флагов — отметок о наступлении определенного события. При большой или, наоборот, отсутствующей нагрузке микропроцессор прерывает выполнение текущей функции и запускает функцию прерывания, которая становится приоритетной.

3. Контакты 3, 5, 6, 9, 10, 11, поддерживающие широтно-импульсную модуляцию (ШИМ; рис. 2.8) — операцию вывода изменяющегося аналогового значения с помощью цифровых устройств. Рядом с этими контактами стоит значок ~.

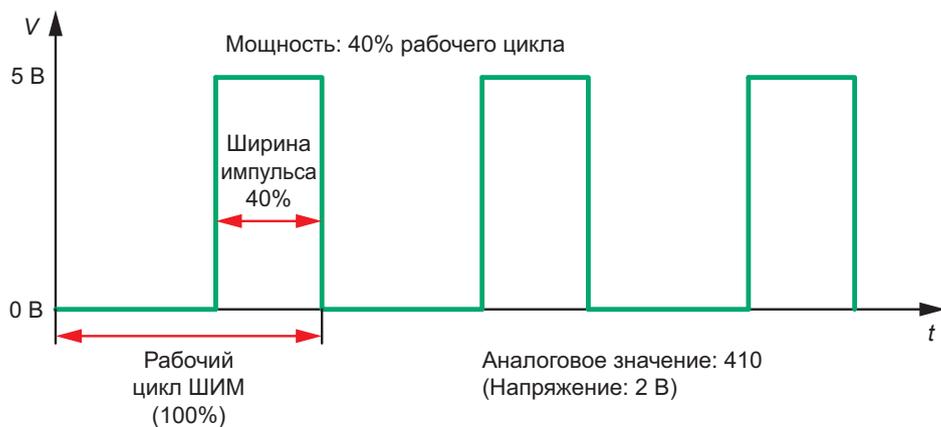


Рис. 2.9. Принцип действия ШИМ

Дело в том, что микроконтроллер ATmega328 не способен выдавать непрерывные изменения сигнала. Он имеет только два состояния: HIGH и LOW, т. е. может выдавать только цифровой сигнал. Этого недостаточно для работы, например, с динамиками, так как в этом случае никогда не будет достигаться нужное значение частоты звука.

Принцип ШИМ заключается в постоянном переключении с определенной частотой между логическим нулем и единицей. Длительность включения максимального значения, т. е. **ширина импульса** (рис. 2.9), зависит от величины аналогового сигнала: чем больше его величина, тем шире импульс, и наоборот. Похожий принцип часто встречается при создании полупрозрачных изображений, например наклеиваемых на стекла общественного транспорта, когда непрозрачные точки чередуются на картинке с пустым местом.

Называть сигнал, полученный с помощью ШИМ, аналоговым нельзя, но его достаточно для полноценной работы с устройствами. Например, светодиод, подключенный к порту с ШИМ, на самом деле мерцает, а не горит, но это незаметно человеческому глазу.

4. Контакт 13 — к нему напрямую подключен встроенный светодиод. При отправке сигнала HIGH светодиод загорается, при сигнале LOW гаснет.

5. Интерфейс SPI — последовательный периферийный интерфейс. Обеспечивает синхронизацию передачи данных между устройствами (отправку и получение), что дает высокую скорость обмена между платой и датчиками. Одно из устройств выступает *ведущим (master)*, второе — *ведомым (slave)*. Например, при под-

ключении дальногомера он получает роль ведомого, а Uno — ведущего. Контакты:

- 10 — SS (Slave select) — ведущий выбирает ведомого;
- 11 — MOSI (Master Out, Slave In) — выход ведущего, вход ведомого;
- 12 — MISO (Master In, Slave Out) — вход ведущего, выход ведомого;
- 13 — SCK (Serial Clock) — с него последовательный тактовый сигнал отправляется на ведомое устройство.

6. *GND* — дублирует GND блока питания. Этот контакт выведен для удобства.

7. Аналоговый контакт *AREF* — определение границы допустимого напряжения. Сюда подается информация, если требуется поднять верхнюю границу аналогового сигнала выше 5 В. Так же можно изменить опорное напряжение (5 В или 3,3 В) на напряжение, подаваемое на *AREF*.

Кроме перечисленных блоков, на плате Arduino Uno есть еще несколько важных элементов:

- *собственная кнопка перезагрузки* — выполняет сброс программы и значений переменных в памяти устройства;
- *разъем питания* — гнездо под штекер диаметром 2,1 мм, стандартное для внешних зарядных устройств или блоков питания на аккумуляторах или батарейках типа «Крона», AA или AAA;
- *USB-разъем* — может использоваться для двух целей: в качестве разъема для подачи питания от компьютера или мобильного аккумулятора либо как *последовательный порт (Serial Port)*, работающий по протоколу UART. Протокол *UART* (УАПП, универсальный асинхронный приемопередатчик) — старейший и самый распространенный протокол передачи данных. Именно через него на компьютер будет поступать информация с датчиков и результаты работы Arduino, а также будут загружаться скетчи и передаваться команды с компьютера. USB-разъем Arduino защищен от перегрузок: в случае подачи на плату напряжения, превышающего допустимое, порт будет отключен и никакого короткого замыкания не произойдет;
- *разъем для внутрисхемного программирования (ICSP)* — небольшой разъем для тонкой настройки ATmega, используемый для записи непосредственно в EEPROM.

Вопросы

1. Где на плате Arduino Uno расположен блок контактов питания? Перечислите функции его контактов. Каково назначение напряжения 5 В и 3,3 В? Ответ запишите в виде таблицы в тетради.
2. Что такое аналоговый сигнал и какие значения он принимает? Где расположены аналоговые входы на плате Arduino Uno?
3. Что такое цифровой сигнал? С помощью какой операции цифровой сигнал можно представить как аналоговый? Какие контакты позволяют это сделать? Для чего это требуется?
4. Подумайте и обоснуйте, для чего необходимо внешнее прерывание. Приведите пример.

Это интересно!

Проекты под управлением Arduino принято относить к робототехнике, однако чаще всего речь идет об автоматике и автоматизации. Кстати, то же самое касается проектов на LEGO Mindstorms EV3.

Практические задания

Задание 1. Найдите в Интернете примеры плат на платформе Arduino разных производителей. Кратко опишите, чем они различаются (3–4 предложения о каждой плате). Например, платы Arduino LilyPad, Arduino Esplora или Amperka Strela.

Задание 2. При наличии деталей соберите блок питания для Arduino.

Компоненты (рис. 2.10):

- блок для четырех элементов питания типа АА, 1х;
- штекер для разъема питания с клеммником, 1х;
- элементы питания типа АА, 4х;
- отвертка РНО (крестовая), 1х;
- ножницы, 1х;
- изоляционная лента, 1х.



Рис. 2.10. Компоненты для сборки блока питания Arduino



Внимание

Батарейки (или аккумуляторы) пока отложите в сторону — работать со вставленными элементами питания нельзя! От блока отходят два провода: красный и черный. В электронике принято использовать эти цвета для обозначения питания (красный) и земли (черный).

1) Возьмите штекер и ослабьте винты клеммника, повернув их отверткой против часовой стрелки.



Рис. 2.11. Клеммник со вставленными проводами

2) Вставьте зачищенный конец черного провода в гнездо со знаком «минус», а красного — в гнездо со знаком «плюс» (рис. 2.11). Внутри клеммника должны оказаться только медные концы проводов без изоляции (оболочки). Если длина оголенной части проводов недостаточна, зачистите провод. Если провод на конце «распушился», закрутите его.

3) Затяните винты клеммника, чтобы провода держались в нем прочно. Сделайте виток изолянты вокруг места соединения.

4) Вставьте батарейки в блок для элементов питания, соблюдая полярность. Автономное питание для будущего прототипа готово.



Внимание!

Если провода у входа в штекер нагреваются, немедленно вытащите батарейки и разберите конструкцию! Это означает, что где-то случилось короткое замыкание. Кроме того, оголенные части проводов не должны выступать из клеммника во избежание короткого замыкания. Это очень опасно!

2.3. Макетные платы

Для подключения к Arduino дополнительных компонентов применяются *макетные платы*. По способу подключения компонентов они бывают двух типов:

- 1) для прототипирования с помощью пайки (рис. 2.12);
- 2) для прототипирования без пайки (рис. 2.13).

Первый способ позволяет надежно закреплять компоненты, но в случае ошибки их сложно демонтировать. Кроме того, есть

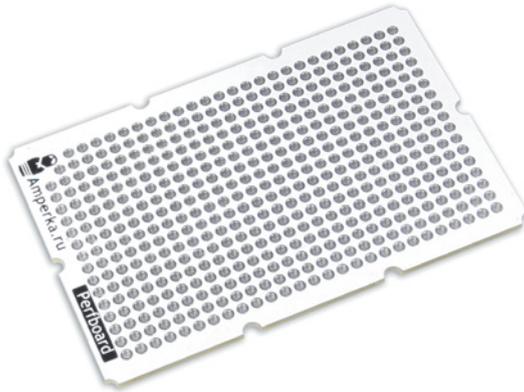


Рис. 2.12. Макетная плата для прототипирования с помощью пайки

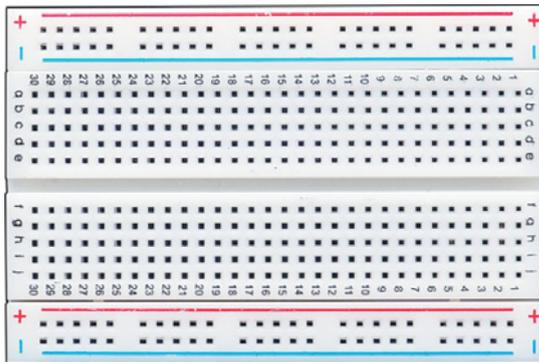


Рис. 2.13. Макетная плата для прототипирования без пайки

риск ожога или вдыхания вредных веществ при неумелом обращении с паяльником.

Для прототипирования без пайки используются макетные платы с уже объединенными в шины контактами, которые значительно расширяют возможности подключения дополнительных компонентов к основной плате с микроконтроллером.

Макетные платы типа *miniBoard* для пайки часто входят в состав плат расширения (щитов). Непосредственно на щитах также могут быть размещены макетные платы с готовыми шинами, имеющие на нижней поверхности клейкую прослойку. В таком случае подключенные элементы закреплены почти так же надежно, как при первом способе, особенно при соединении не проводами, а жесткими перемычками.

Как видите, *аппаратная часть платформы Arduino нацелена на максимальную совместимость не только с компонента-*

ми внутри платформы, но и с другими устройствами, взаимодействие с которыми происходит через стандартные интерфейсы.

Программная часть платформы состоит из открытого (свободного) программного обеспечения. Это означает, что все коды программной среды разработки Arduino IDE выложены в открытом доступе и могут быть использованы, изменены и усовершенствованы без каких-либо ограничений. Подобный подход дал возможность многим разработчикам-энтузиастам создать дополнительное программное обеспечение, например модуль графического программирования ArduBlock или среду S4A, в которой мы впоследствии будем работать. Подробнее программная часть платформы будет рассмотрена в следующем разделе.



Вопросы

1. Какие виды макетных плат вы знаете?
2. Какие функции выполняют контакты, записанные в таблице? Заполните таблицу.

Контакт	Функция
IOREF	
RESET	
3.3V	

Контакт	Функция
5V	
GND	
VIN	



Запомните

- ◆ Открытая архитектура ◆ Аналоговый сигнал
- ◆ Цифровой сигнал ◆ Широтно-импульсная модуляция (ШИМ)
- ◆ Ширина импульса

Это интересно!

Дополнительные компоненты (датчики и модули) для Raspberry Pi совместимы с платформой Arduino, но при основном напряжении 3,3 В.

Глава 3. Программное обеспечение Arduino

3.1. Среда разработки Snap4Arduino (S4A)

Знакомство с платой и программированием мы предлагаем начать с графического блочного программирования на языке Snap!, хотя официальной программной составляющей платформы Arduino является среда программирования Arduino IDE.

Прежде чем начать, заглянем в историю этого языка визуального программирования. **Snap!** разработали Дженс Мониг, Брайан Харви и команда студентов Калифорнийского университета, Беркли (<http://snap.berkley.edu/>). Их вдохновили два языка программирования, от которых исследователи взяли самое лучшее: графический язык Scratch, один из наиболее распространенных языков обучения программированию в мире, и Scheme, язык — наследник известнейшего языка программирования LISP, созданного для моделирования искусственного интеллекта. От первого языка Snap! взял удобный интерфейс, перетаскивание блоков на рабочую область и анимацию, а от второго — способы работы с объектами и процедурами.

Snap! считается модификацией языка Scratch 1.2, но имеет следующие преимущества:

- простое создание пользовательских блоков (стало доступно только в Scratch 2.0);
- использование списков первого порядка;
- использование процедур первого порядка;
- сохраняемые состояния программы (возможность паузы).

Исполняемые фрагменты кода на Snap! называются **скриптами** (от англ. Script — *сценарий*).

По аналогии с языком Scratch программы и подпрограммы в Snap! называются **спрайтами**. Они включают в себя все используемые материалы: костюмы, звуки, скрипты. Если в проекте существует несколько спрайтов, то их совокупность называется **пакетом спрайтов**.

Еще один компонент, который потребуется при работе в S4A и пришедший из Scratch, — это **сцена**. Сценой называют про-

странство, где отображаются персонажи и графические объекты, с которыми происходит действие (анимация).

Имейте в виду, что плата Arduino поддерживает только один поток данных. Это значит, что она выполняет команды последовательно и не способна к параллельному выполнению нескольких задач. Существует небольшая хитрость: команды ставятся на паузу с сохранением состояния и запускаются поочередно. Это обеспечивает псевдопараллельные вычисления. В результате возможности Arduino значительно расширяются.

Дальше — больше. Реальные проекты, в отличие от большинства учебных, создаваемых на уроках информатики, зачастую включают в себя компоненты, написанные на разных языках программирования. Обработчик (приложение, в котором происходит сборка и загрузка кода) написан на JavaScript (JS). Это основной язык программирования сайтов, ведь именно он заставляет двигаться динамические части большинства страниц в Сети.

Работа с языком Snap! по программированию роботов и автоматике на Arduino выполняется в специальной среде, о которой упоминалось ранее, — Snap4Arduino. Она была разработана командой Берната Ромагозы в Барселоне.

Теперь, если кто-то вам скажет, что визуальный язык программирования не может быть по-настоящему полезным и производительным, смело отправляйте его на эту страницу нашей книги или на сайт создателей Snap!

Установка S4A

Работать с графическим языком программирования Snap! можно в разных оболочках. Вы будете использовать свободно распространяемое программное обеспечение S4A (Snap for Arduino).

Для работы в этой среде необходимо подготовить вашу плату Arduino, загрузив в нее специальную прошивку S4A, которая обеспечит постоянную связь по COM-порту в режиме ожидания команд от компьютера. Делается это при *первом* подключении платы к компьютеру. По тому же принципу работают все альтернативные прошивки.

Это очень простая операция:

1. Скачайте файл прошивки на сайте S4A (<http://s4a.cat/downloads/S4AFirmware15.in>).
2. Скачайте установщик стандартной среды **Arduino IDE**. Это программное обеспечение распространяется бесплатно, поэтому вы без проблем можете скачать себе установочный файл, перейдя

Download the Arduino IDE



Рис. 3.1. Загрузка Arduino IDE. Выбор операционной системы

на официальный сайт Arduino: <https://www.arduino.cc/en/Main/Software>

В перечне справа выберите операционную систему, установленную на вашем компьютере (рис. 3.1). Если у вас установлена ОС Windows, но нет прав администратора (ограниченная учетная запись), кликните по второй строке.

На странице загрузки будет предложено сделать пожертвование разработчикам среды. Если вы хотите скачать установщик без взноса, кликните на **Just Download** (рис. 3.2).

Если у вас нет компьютера или требуется, чтобы среда программирования для роботов была всегда с собой, то можно найти бесплатные приложения в Google Play, например **ArduinoDroid**, или в App Store, например **ArduinoCode**.

Также можно воспользоваться веб-версией среды программирования, доступной по адресу <http://create.arduino.cc/editor> (рис. 3.3).

3. Откройте скачанный ранее файл прошивки S4A. Автоматически запустится среда Arduino IDE. Выберите требуемый тип платы в подменю **Инструменты** → **Плата** → **Arduino/Genuino Uno** (рис. 3.4).

Загрузите прошивку, нажав на кнопку **Загрузить** . Когда внизу появится сообщение об окончании загрузки, выйдите из среды Arduino IDE.

4. Установите среду S4A на компьютер с сайта проекта <http://s4a.cat/>, выбрав установщик, соответствующий вашей операционной системе (рис. 3.5).

Support the Arduino Software

Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). Learn more on how your contribution will be used.

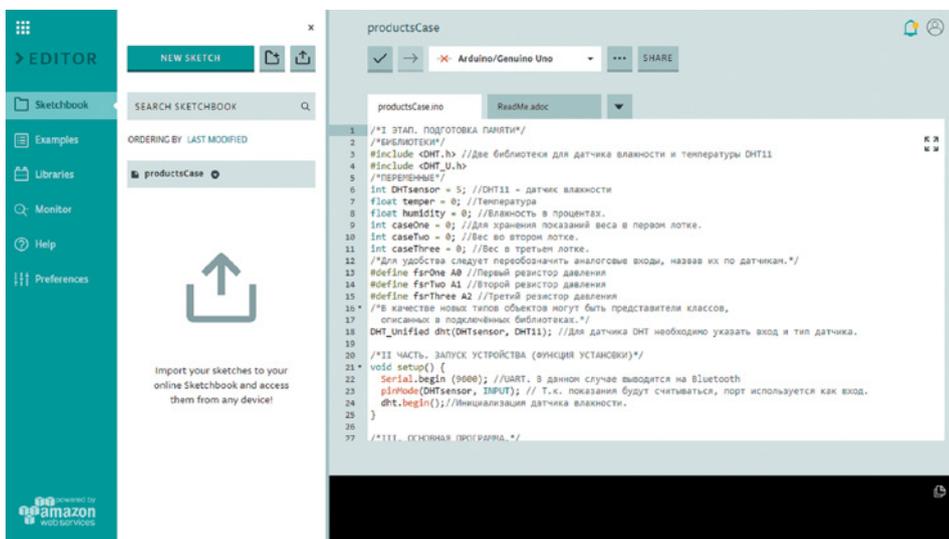


SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED **8 713 645** TIMES. IMPRESSIVE! THIS IDE IS NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS. HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING IT TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLES, CLONES, AND EVEN COUNTERFEIT. YOU CAN HELP ACCELERATE THE DEVELOPMENT OF THE ARDUINO IDE BY CONTRIBUTING TOWARDS THE EFFORT OF MAKING IT BETTER.

\$3 **\$5** **\$10** **\$25** **\$50** **OTHER**

JUST DOWNLOAD **CONTRIBUTE & DOWNLOAD**

Рис. 3.2. Размер пожертвования при скачивании



productsCase

✓ → Arduino/Genuino Uno ... SHARE

productsCase.ino ReadMe.doc

```

1 /*1 ЭТАП. ПОДГОТОВКА ПАмяТИ*/
2 #include <math.h>
3 #include <DHT.h> //Две библиотеки для датчика влажности и температуры DHT11
4 #include <DHT_U.h>
5 #PERЕМЕННЫЕ?
6 int DHTsensor = 5; //DHT11 - датчик влажности
7 float temp = 0; //Температура
8 float humidity = 0; //Влажность в процентах.
9 int caseOne = 0; //Для хранения показаний веса в первом лотке.
10 int caseTwo = 0; //Вес во втором лотке.
11 int caseThree = 0; //Вес в третьем лотке.
12 /*Для удобства следует переобозначить аналоговые входы, назвав их по датчикам.*/
13 #define fsrOne A0 //Первый резистор давления
14 #define fsrTwo A1 //Второй резистор давления
15 #define fsrThree A2 //Третий резистор давления
16 /* В качестве новых типов объектов могут быть представлены классы,
17 описанные в подключаемых библиотеках.*/
18 DHT_Unified dht(DHTsensor, DHT11); //Для датчика DHT необходимо указать вход и тип датчика.
19
20 /*II часть. ЗАПУСК УСТРОЙСТВА (ФУНКЦИЯ УСТАНОВКИ)*/
21 void setup() {
22   Serial.begin(9600); //UART. В данном случае выводится на Bluetooth
23   pinMode(DHTsensor, INPUT); // Т.е. показания будут считываться, порт используется как вход.
24   dht.begin(); //Инициализация датчика влажности.
25 }
26
27 /*III. ОСНОВНАЯ ПРОГРАММА.*/

```

Import your sketches to your online Sketchbook and access them from any device!

powered by amazon

Рис. 3.3. Веб-версия среды программирования

В дальнейшем для программирования платы Arduino Uno на языке Snap! потребуется запустить всего лишь программу S4A.

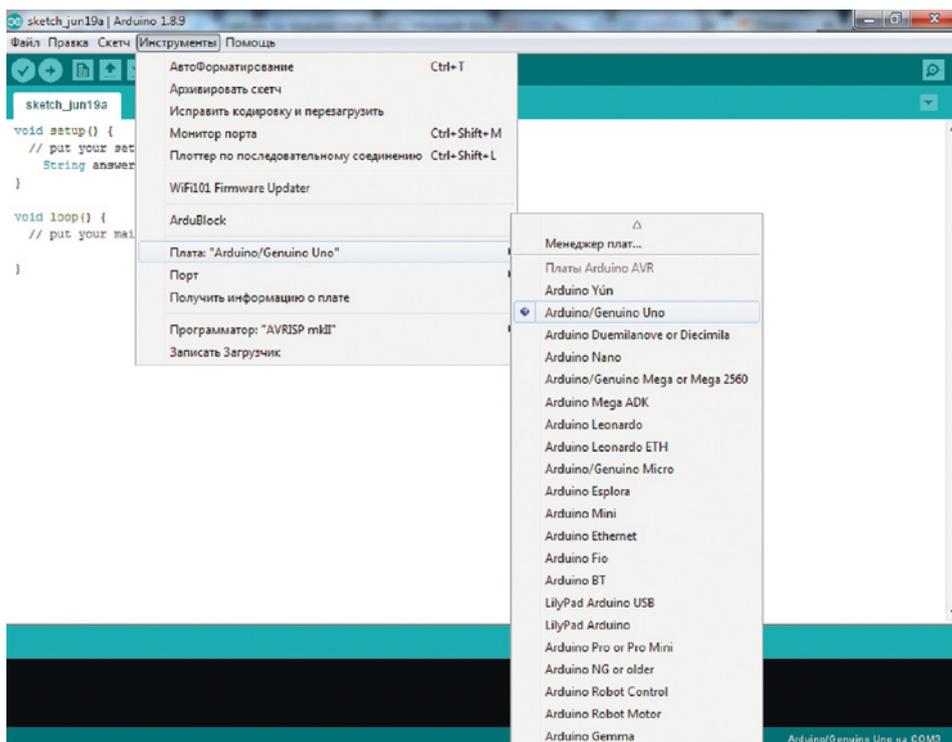


Рис. 3.4. Выбор платы

About Docs Android Changelog Downloads Kit FAQ Team

Download and Install

- 1 Download and install **S4A**
- 2 Install our firmware into your **Arduino**
- 3 Enjoy programming your **world!**

Installing S4A requires you to install software both in your PC and your [Arduino](#) board. Here you'll find the detailed steps to get it up and running.

Installing S4A into your computer

S4A works in the three major consumer operating systems. Download and install the one that fits your configuration:

- [Windows](#)
- [Mac](#)
- [Linux \(Debian\)](#)
- [Linux \(Fedora\)](#) (version 1.5)
- [Raspbian \(Debian for RaspberryPi\)](#) (version 1.5)

Рис. 3.5. Установка среды S4A



Примечание

Теперь вы можете приступить к выполнению практических заданий и знакомству с периферией Arduino. Для этого переходите к главе 4. Мы специально собрали параграфы, посвященные ПО для Arduino, в одном месте, чтобы вы могли пользоваться книгой и как справочником.



Вопросы

1. Перечислите, на каких языках программирования основан Snap!
2. Можно ли применять несколько языков программирования в одном проекте?

3.2. Среда разработки Arduino IDE

В предыдущей главе говорилось, что программная часть платформы состоит из среды разработки Arduino IDE и дополнительного программного обеспечения (ПО), созданного энтузиастами и компаниями, производящими Arduino-совместимые аппаратные компоненты.

Среда разработки Arduino IDE является свободным ПО и распространяется авторами платформы. Ее назначение — программирование микроконтроллера ATmega. На официальном сайте (<https://www.arduino.cc/en/Main/Software>) доступны установщики для разных операционных систем, включая бинарные файлы для Linux. Также предоставляется безустановочный архив для пользователей Windows, не имеющих прав администратора.

Среда разработки позволяет писать, проверять, компилировать и загружать программный код в платы Arduino через UART-интерфейс. Этот интерфейс можно обеспечить физическим соединением через USB, Bluetooth, Wi-Fi, LAN в зависимости от возможностей установленных или встроенных аппаратных модулей. Для платы Arduino Uno без дополнительных плат расширения доступно подключение только с помощью USB. Сами программы пишутся на языке Wiring — специальной модификации языков программирования C/C++, разработанной для взаимодействия микроконтроллеров и периферийного электрического оборудования. Подробнее о том, почему был выбран именно этот язык, и особенностях Wiring мы расскажем позднее.

Программы, написанные в редакторе кода среды Arduino IDE, называются *скетчами* (от англ. *sketch* — набросок). Для

их хранения был придуман формат .INO, получивший название от последних букв слова «Arduino». При сохранении программы Arduino IDE создает для каждого скетча папку проекта. Дело в том, что ранние версии среды использовали расширение .PDE — файла исходного кода на языке Processing, основанного на Java, но близкого по синтаксису к C/C++. Это объясняется тем, что среда Arduino IDE развивалась из среды Processing IDE с заменой языка Processing на язык Wiring.

В таком файле содержались функции, константы и дополнительные конструкции. Программы Processing назывались «скетчи» (*sketches*) и предназначались для программирования изображений, анимации и взаимодействия внешних элементов. Каждый такой файл сохранялся в одноименной собственной папке проекта вместе со всеми свойствами и дополнительными картинками. Это позволяло легко переносить готовые решения визуального окружения (например, программу, с помощью графиков отображающую температуру и влажность в помещении). Из курса информатики вам известно, что некоторые программы могут ссылаться на подпрограммы — файлы других программ в рамках этого же проекта с определенным набором часто вызываемых функций. Таким образом, в папке может содержаться не один скетч, а несколько. Поэтому место их хранения было названо *скетчбуком* (в переводе с англ. книга набросков), а процесс создания — *скетчингом* (т.е. создание набросков). При работе с микроконтроллерами содержимое одного скетчбука — это все, что загружается в память устройства непосредственно из среды разработки.

Вопросы

1. Что такое скетч, скетчбук и скетчинг?
2. В каком формате сохраняются скетчи в последних версиях среды разработки Arduino IDE?
3. На каком языке пишутся программы для Arduino?

3.3. Работа в Arduino IDE

После установки среду нужно запустить на компьютере. При этом откроется рабочее пространство. Для каждого нового скетча открывается отдельное окно. Окно содержит следующие основные зоны (рис. 3.6).

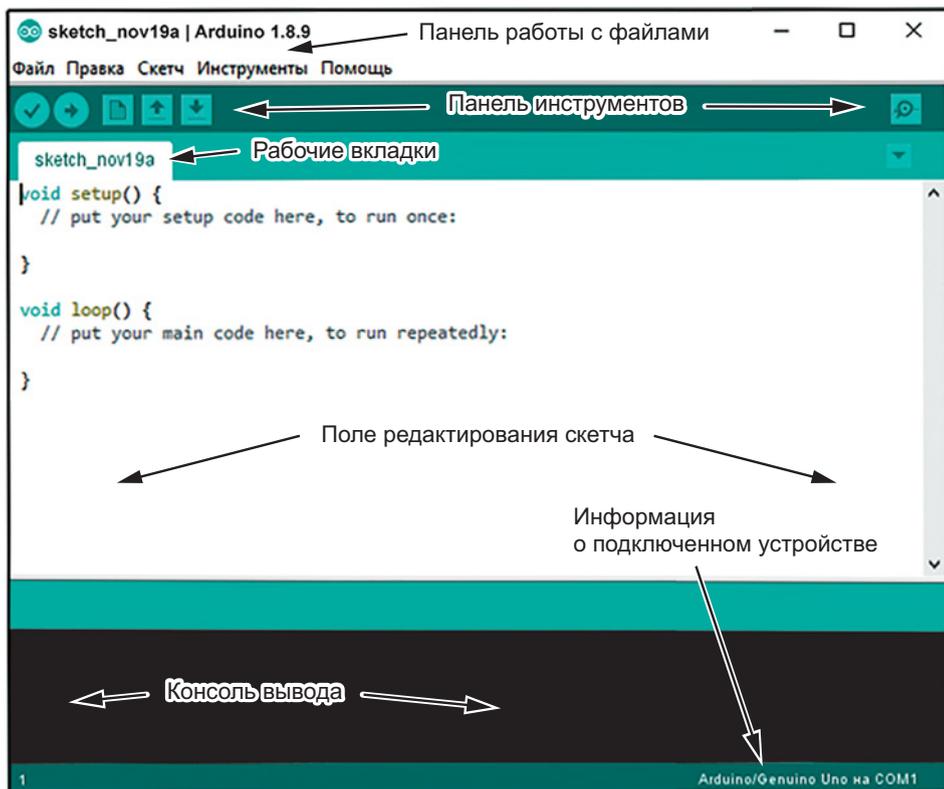


Рис. 3.6. Окно скетча

1. Панель работы с файлами. Это стандартная верхняя панель приложений, которая содержит вкладки:

- «Файл». Функции: открытие, закрытие, сохранение, вывод на печать, папка примеров, недавние проекты, настройки среды.
- «Правка». Кроме поиска и замены, присутствует полезная функция копирования кода в HTML, что позволяет вставлять его с подсветкой синтаксиса на страницы форума или блога. Несмотря на то что нумерация строк в среде Arduino IDE по умолчанию не отображается, перейти к нужной строке (например, с допущенной ошибкой) можно с помощью соответствующей команды (нумерацию можно включить в настройках).

Рядом с текстовым названием функции правки дается альтернативное сочетание клавиш. Их запоминание значительно ускоряет процесс написания кода, но не является обязательным.

- «Скетч». Содержит основные команды компилятора, в том числе проверку и загрузку скетча в плату.

Загрузка с помощью программатора требуется для замены загрузчика самой Arduino, т. е. переписывания защищенной области памяти.

Экспорт бинарного файла — при этой операции скетч представляется в виде двоичных данных, однако не загружается в память микроконтроллера, а сохраняется в скетчбуке для передачи, изменения или загрузки в Arduino с помощью других программ.

Подключение библиотек упрощает написание кода. Подробнее они будут рассмотрены вместе с особенностями языка Wiring.

Добавление файла в скетчбук — операция прикрепления звуков, картинок, электронных таблиц и последующей работы с ними в скетче.

- «Инструменты»:

Автоформатирование в среде реализовано слабо и сводится к соблюдению новых строк при открытии фигурных скобок.

Архивирование скетча в отдельный zip-архив с датой в названии позволяет соблюдать *контроль версий* — сохранение разных стадий готовности программного продукта (проекта) для возможности восстановления последней работоспособной версии в случае возникновения ошибок.

Исправление кодировки убирает некорректно отображаемые символы. Например, восстанавливает ASCII (стандартную кодировку для строк ATmega) из текста, написанного в Windows-1251 или UTF-8. К сожалению, Arduino IDE не поддерживает вывод кириллицы.

Монитор порта — это окно диалога между пользователем и Arduino (рис. 3.7). В нем отображается информация, поступающая от платы по протоколу UART на скорости, указанной в правом нижнем углу окна монитора порта. Отправляемые данные переводятся в коды символов ASCII. Символ конца строки добавляется в код, отправляемый на плату, и может быть использован в программе. Значения скорости UART в мониторе порта и внутри программы должны совпадать.

Плоттер по последовательному соединению — это инструмент, позволяющий в режиме реального времени строить графики по данным, поступающим от портов Arduino. Arduino IDE поддерживает множество различных плат, каж-

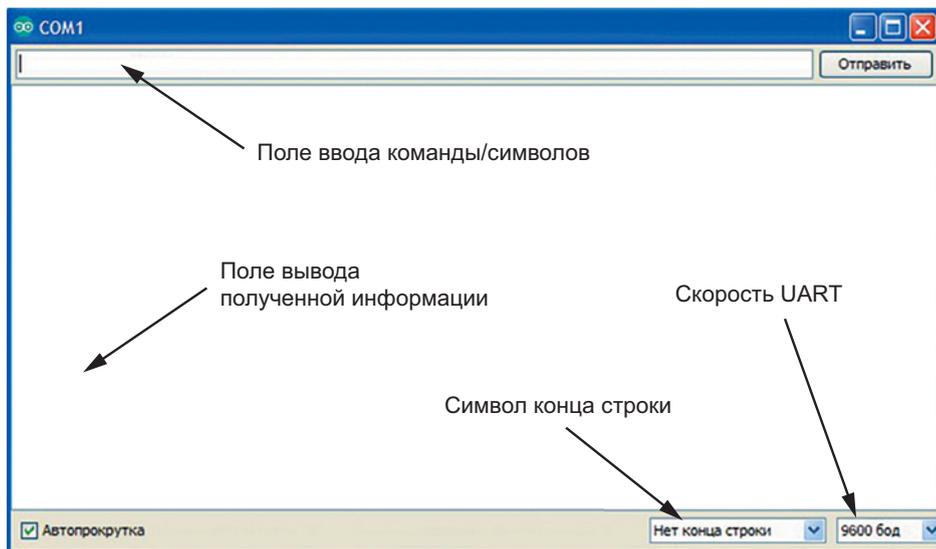


Рис. 3.7. Монитор порта

дая из которых имеет свои особенности (например, модель микропроцессора и объем памяти). Поэтому важно, чтобы название платы в среде и порт, к которому она подключена, совпадали с реальными.

Обычно Arduino IDE автоматически определяет плату и порт. Для USB-подключения порту присваивается название COM X, где X — цифровой номер порта. Каждый тип микроконтроллера, установленного на плате, требует программатор, который ему соответствует.

- «Помощь». Этот раздел состоит в основном из ссылок на онлайн-справку на английском языке.

2. Панель инструментов. На ней размещаются основные кнопки, используемые при работе с программой.

	Проверка скетча, записанного в поле ввода. Среда компилирует (собирает для данного микропроцессора) программу, проверяя на наличие ошибок.
	Компилирование программы и загрузка ее на подключенное устройство. Arduino сразу же запустит полученную программу.
	Создание нового скетча. Открывается новая вкладка. В качестве названия используется текущая дата.
	Открытие сохраненного ранее скетча.

	<i>Сохранение открытого скетча.</i> Не забывайте использовать эту функцию перед закрытием среды, чтобы не потерять достигнутый результат!
	<i>Монитор порта.</i> Запускает диалоговое окно аналогично команде на панели работы с файлами.

3. Панель скетчей (рабочие вкладки). Открытые скетчи отображаются в виде вкладок. По умолчанию названием несохраненной вкладки является текущая дата. Переключаться между вкладками можно с помощью комбинации клавиш `Ctrl+Alt+Left` или `Ctrl+Alt+Right`. Новая вкладка создается сочетанием клавиш `Ctrl+Shift+N` или выбором соответствующей вкладки. В меню можно также переименовать открытый скетч.

4. Поле редактирования скетча. Сюда записывается исходный код программы на языке Wiring. В редакторе имеется поддержка подсветки синтаксиса, в том числе стандартных имен и функций. Нумерация строк не отображается, однако ведется; их количество можно увидеть в нижнем левом углу окна Arduino IDE.

5. Консоль вывода. Консоль выводит сообщения компилятора об ошибках в коде, проблемах подключения к плате и загрузке в устройство, а также о состоянии свободной памяти Arduino. При выводе указывается номер строки с ошибкой, который можно использовать при поиске проблемного места в процессе отладки.

6. Панель общей информации. Слева находится информация о количестве строк в коде, справа — название платы и порт, к которому она подключена. При компилировании или загрузке с ошибкой панель меняет свой цвет на красный, сигнализируя о проблеме.

Мы рассмотрели все основные функции среды разработки Arduino IDE. В дальнейшем этот материал может пригодиться в качестве руководства пользователя.

Вопросы

1. Какие основные операции с файлами доступны в среде Arduino IDE?
2. Что такое экспорт бинарного файла и чем он отличается от загрузки файла на устройство?
3. Чем различаются монитор порта и плоттер по последовательному соединению? Что подразумевается под этими понятиями?
4. Какие команды доступны на панели инструментов?

3.4. Альтернативное программное обеспечение для Arduino

Для смартфонов и планшетных компьютеров существуют аналоги среды разработки, также распространяемые бесплатно. Например, для гаджетов под управлением операционной системы Android доступна программа ArduinoDroid.

Иногда при перечислении наборов для прототипирования упоминается созвучное Arduino название — *Espruino*. Эта платформа входит в семейство Arduino-совместимых платформ. Все платы семейства могут быть подключены друг к другу, потому что имеют один и тот же интерфейс: цифровые входы и выходы, одинаковое напряжение. Их различие заключается в микропроцессоре, наборе размещенных на плате компонентов или просто в производителе. Например, плата Iskra JS идентична Arduino Uno во всем, кроме микропроцессора: вместо ATmega328 на ней установлен более мощный ARM Cortex-M4. Для программирования таких плат используется среда разработки Espruino Web IDE, встраиваемая прямо в браузер Google Chrome.

Кроме официальных сред разработки, для Arduino/Espruino разработано дополнительное программное обеспечение. В большинстве случаев оно распространяется по свободной лицензии, как, например, установленная вами ранее S4A.

Рассмотрим три бесплатные программы, которые могут пригодиться при прототипировании.

1. *Fritzing* — платформа виртуального прототипирования (<http://fritzing.org/home/>; рис. 3.8). Она доступна для Windows, Linux и Mac OS. На этой платформе можно собирать виртуальный прототип и получать принципиальные схемы ваших проектов, а также основу кода для их управления. Процесс прототипирования состоит в перемещении моделей на рабочее пространство и подключении их к макетной плате.

Чтобы соединить два контакта, нужно зажать указатель мыши над одним из контактов и перетащить его на другой контакт. Встроенная коллекция материалов постоянно пополняется новыми моделями от известных производителей. В программе есть поддержка Raspberry Pi и других платформ. Она позволяет собрать проект при отсутствующих физических деталях или избежать случайного короткого замыкания при некорректной сборке.

2. Модуль графического программирования *ArduBlock* (<https://sourceforge.net/projects/ardublock/files/>; рис. 3.9). Является дополнительным модулем для среды разработки Arduino IDE, а не

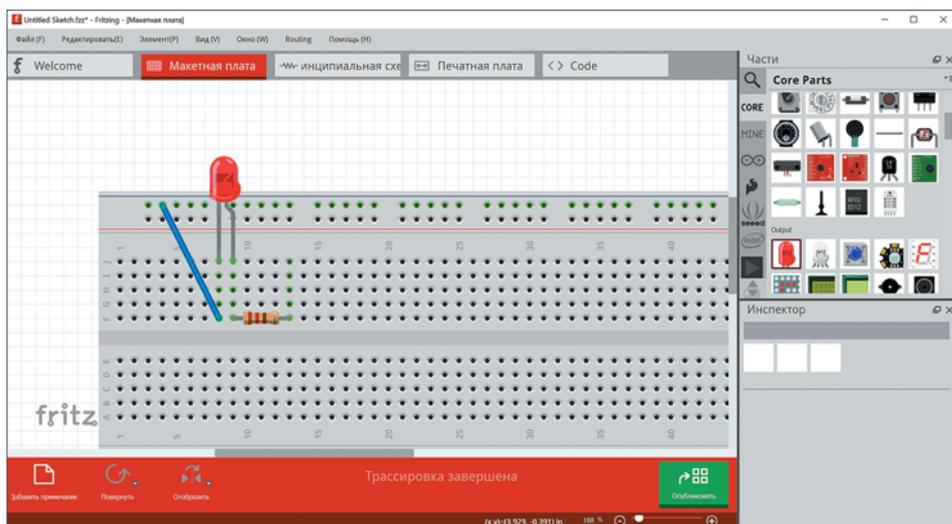


Рис. 3.8. Платформа виртуального прототипирования Fritzting

самостоятельной программой. ArduBlock — это аналог языков Blockly, Scratch или LabVIEW, используемых при программировании роботов LEGO Mindstorms EV3 и других знакомых вам платформ. Он позволяет собирать программу для Arduino из графических блоков, которые при этом автоматически конвертируются в исходный код на Wiring. Вы познакомитесь подробнее с ним в следующих главах.

3. Среда графического программирования *FLProg* (<http://flprog.ru/load/>; рис. 3.10). Она доступна для установки на Windows и Linux. Программа российских разработчиков способна показать связь Arduino и производственной робототехники. FLProg нацелена на программирование с помощью языков *FBD (Function Block Diagram)* (рис. 3.11) и *Ladder Diagram (LD, LAD, PKC)*, т.е. графического языка с построением цепей из триггеров и языка релейной логики соответственно. Второй язык ориентирован на инженеров по автоматизации на производствах.

В итоге программа выдает понятный и легко расшифровываемый любым электронщиком результат. Это значит, что обслуживать готовый продукт (воплощенный в жизнь прототип) смогут непрограммисты. FLProg обеспечивает также взаимодействие с внешними устройствами (например, со станками по электрической сети предприятия).

Проекты, собранные на основе Arduino, легко тиражировать, поскольку детали стоят недорого. Однако они не обладают такой же прочностью и точностью, как дорогое профессиональное обо-

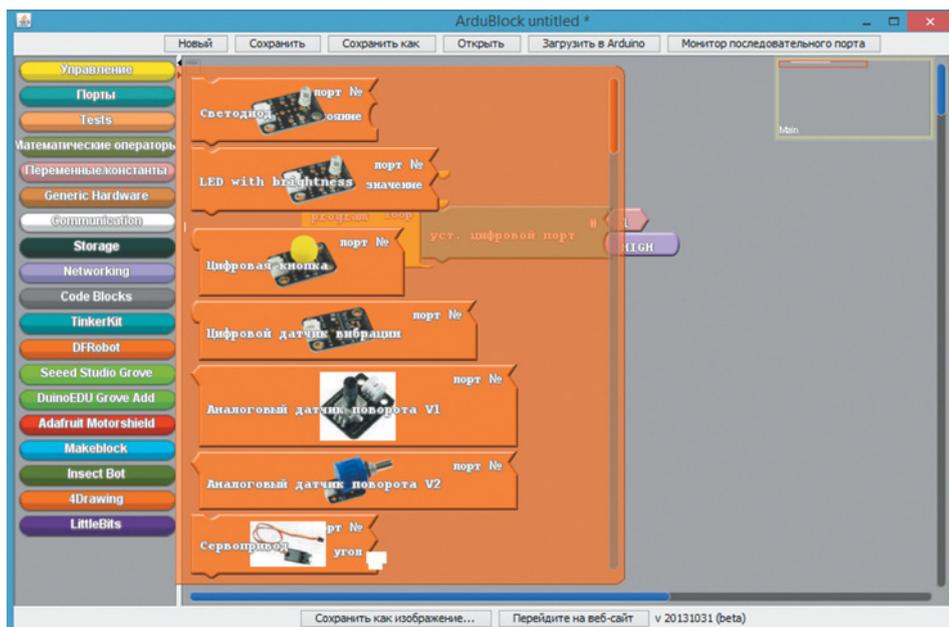


Рис. 3.9. Модуль графического программирования ArduBlock

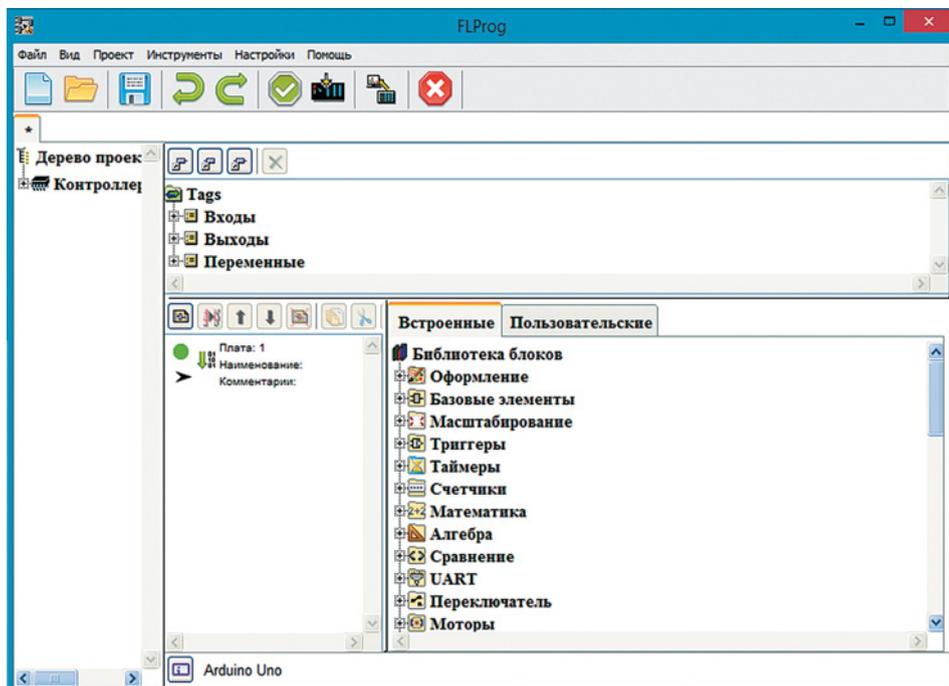


Рис. 3.10. Среда графического программирования FLProg

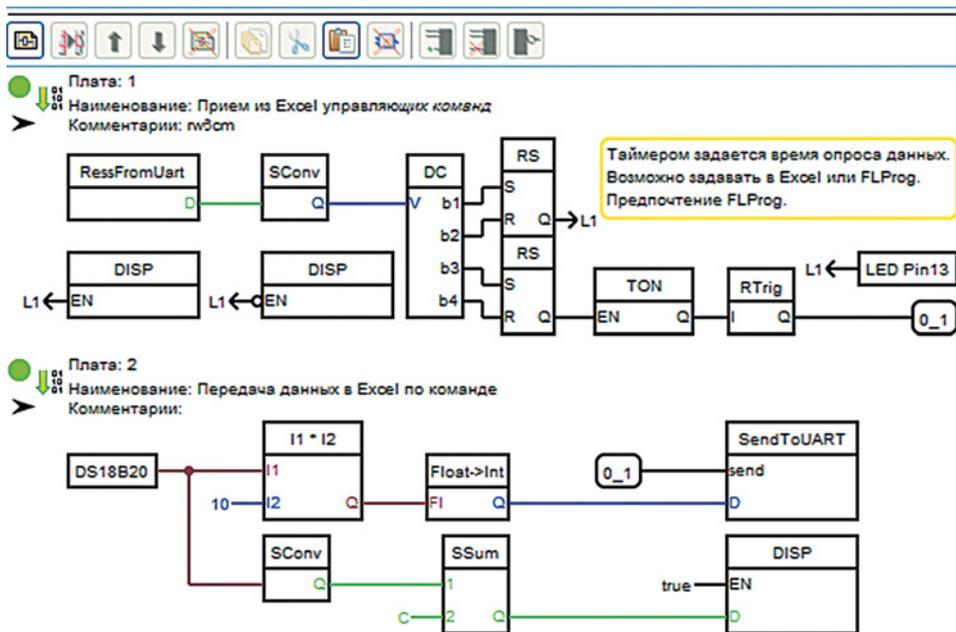


Рис. 3.11. Производственный язык программирования FBD

рудование, поэтому их часто используют как решения только для малого бизнеса, где их мощностей достаточно.

❓ Вопросы

1. Что такое Espruino? Перечислите основные особенности этой платформы.
2. Обоснуйте, почему Arduino — универсальная платформа.

❗ Запомните

- ◆ Среда разработки Arduino IDE
- ◆ Скетч
- ◆ Скетчбук
- ◆ Скетчинг
- ◆ Экспорт бинарного файла
- ◆ Контроль версий
- ◆ Монитор порта
- ◆ Плоттер по последовательному соединению

🔧 Практические задания

Задание 1. Напишите краткий обзор (по одному абзацу) трех программных продуктов для Arduino, распространяемых бесплатно. Для поиска можете воспользоваться базой приложений от разработчиков SourceForge (<https://sourceforge.net/>), центром приложений Ubuntu, магазином приложений Windows, Google Play Market или AppStore и ключевым словом *Arduino*.

Задание 2. Подготовьте иллюстрации к инструкции по подключению светодиода к плате Arduino Uno в программе Fritzing, выполняя следующие пошаговые инструкции.

- 1) Запустите программу Fritzing и создайте новый файл.
- 2) Работа во вкладке «Макетная плата».
 - В разделе **Части** во вкладке **CORE** выберите резистор на 220 Ом (рис. 3.12) и перетащите его на рабочую область.
 - Установите его на макетной плате, как показано на рис. 3.13, перетащив мышью на нужное место.
 - Выберите в разделе **Части** вкладку **Поиск**. Введите в поле запроса **Arduino UNO** и нажмите клавишу **Enter**. Перетащите элемент с надписью **UNO** на рабочую область (рис. 3.14).
 - Найдите красный светодиод, используя запрос **RED LED**. Перетащите элемент на рабочую область (рис. 3.15).
 - Подключите светодиод. У него есть два вывода разной длины, имеющие различное назначение. Традиционно длинный вывод — это **анод**, положительный контакт, по которому светодиод получает питание. Короткий вывод светодиода — это **катод**, отрицательный контакт, т. е. земля схемы. Подключите светодиод к макетной плате, как показано на рис. 3.16.
 - Соедините землю платы Arduino Uno с шиной макетной платы, обозначенной синим цветом. Для этого зажмите левую клавишу мыши на порте GND и не отпуская протяните провод до нужного контакта макетной платы (рис. 3.17). Если случайно протянут лишний провод, удалите его, на-

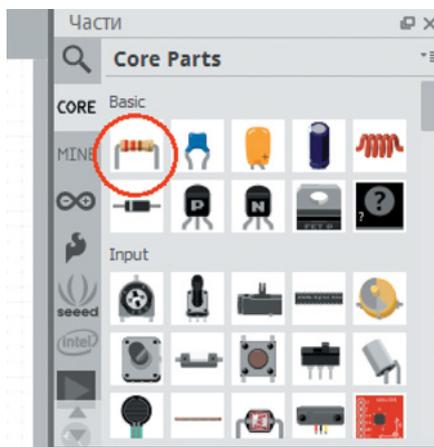


Рис. 3.12. Выбор резистора

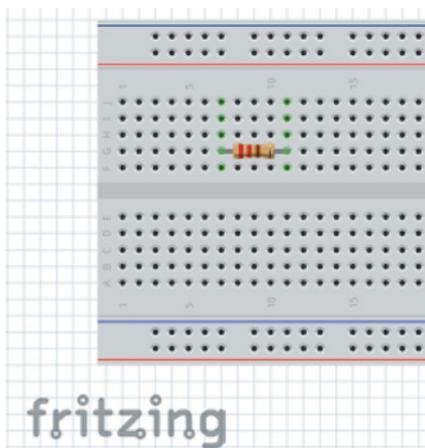


Рис. 3.13. Подключение резистора



Рис. 3.14. Выбор элемента Uno

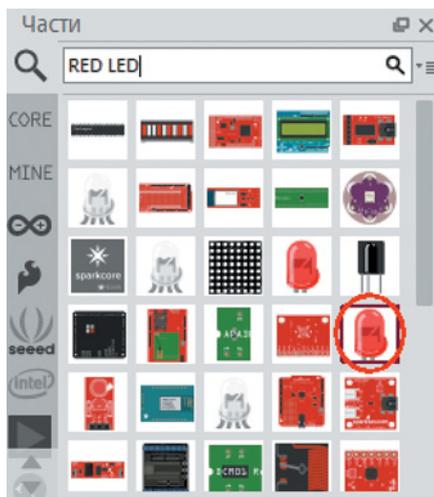


Рис. 3.15. Выбор красного диода

жав на провод правой клавишей мыши и выбрав из контекстного меню пункт **Delete Wire** («Удалить провод»).

- Подайте управляемое питание на светодиод. Пусть Arduino Uno управляет светодиодом через цифровой порт №7. Протяните провод от порта №7 к ряду отверстий макетной платы, в котором установлен левый вывод резистора. В результате получится последовательное соединение резистора и светодиода (рис. 3.18).
- Оформите цвета проводов. Для этого нажмите правой клавишей мыши на провод и выберите пункт **Изменить цвет**.

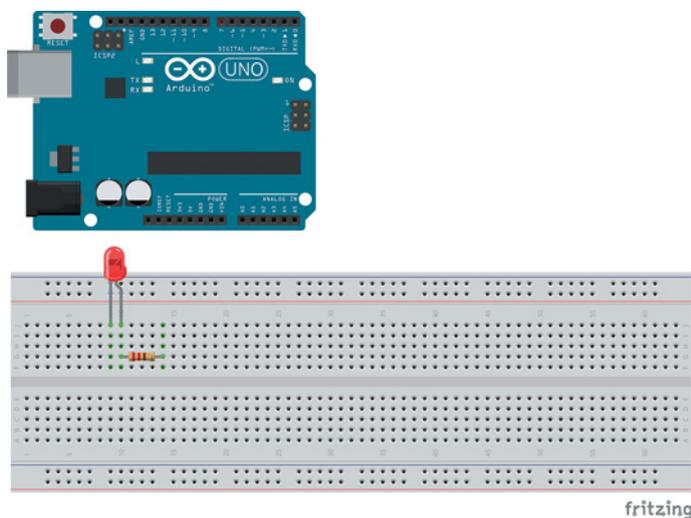


Рис. 3.16. Подключение светодиода к макетной плате

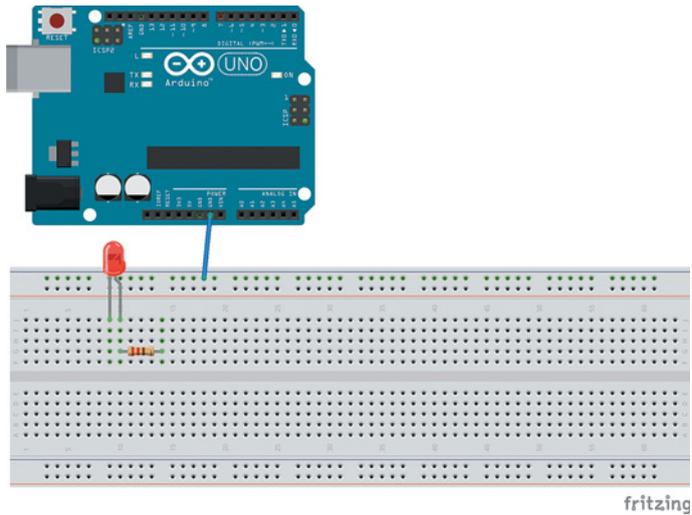


Рис. 3.17. Подключение земли Arduino Uno к земляной шине на макетной плате

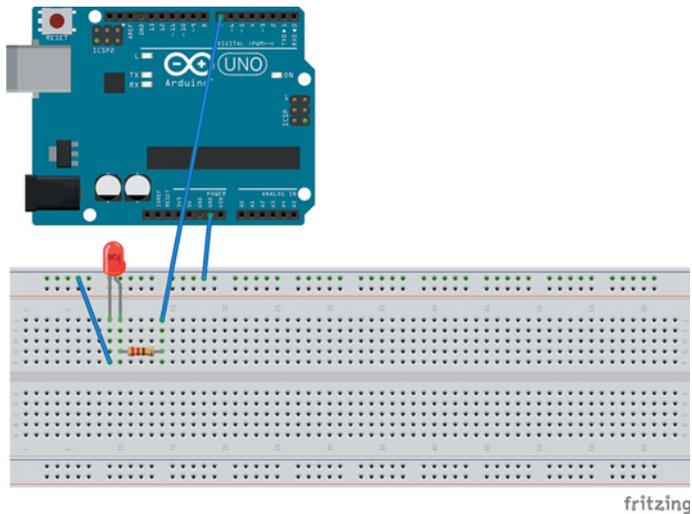


Рис. 3.18. Последовательное соединение резистора и светодиода

Пусть питающий провод будет красного цвета, а земля — черного.

Для того чтобы провод не закрывал части платы и проект выглядел аккуратно, необходимо добавить несколько точек изгиба. Для этого нажмите правой клавишей мыши на провод и выберите **Добавить точку изгиба**. За нее провод можно сдвинуть в сторону. Добавьте несколько точек и уложите провод вдоль боковых сторон платы (рис. 3.19).

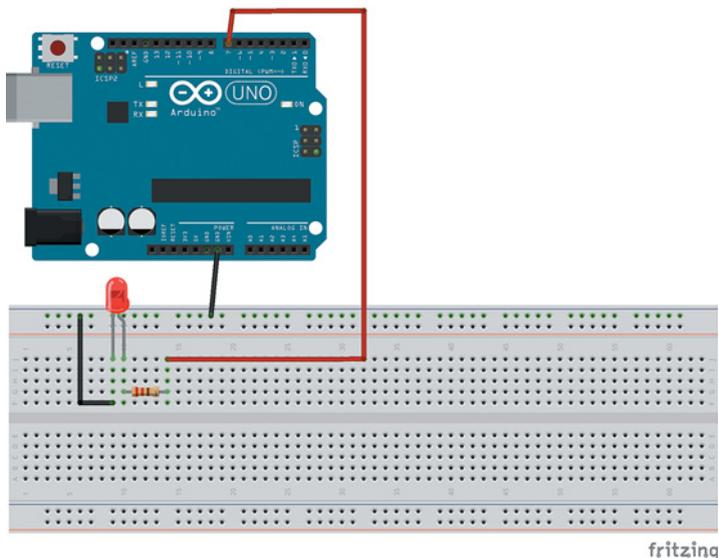


Рис. 3.19. Добавление точек изгиба провода

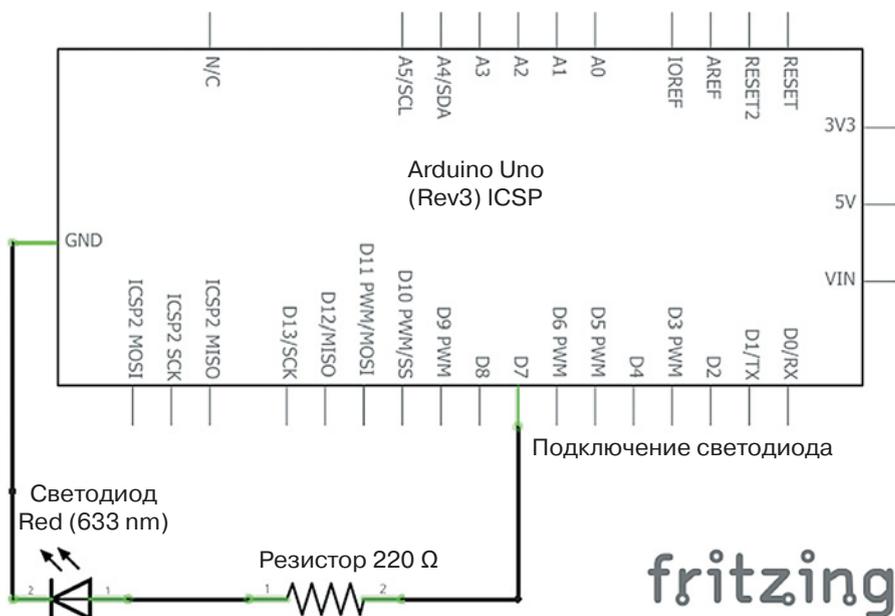


Рис. 3.20. Принципиальная схема

3) Перейдите во вкладку **Принципиальная схема**. Здесь вы увидите, как выглядит подключение с точки зрения электрики (рис. 3.20).

- Для того чтобы изменить название элемента схемы, достаточно нажать на него дважды.

- Элементы схемы можно перетаскивать по рабочей области, чтобы добиться наилучшего размещения. Надписи можно двигать отдельно от элементов.
 - Для соединения действуют те же правила, что во вкладке **Макетная плата**.
 - Элементы схемы можно поворачивать. Для этого нужно выбрать соответствующий пункт из контекстного меню правой клавиши мыши.
 - Надписи можно поворачивать аналогичным способом.
 - Оформите свою схему.
- 4) Вкладка **Печатная плата** необходима тем, кто решит закрепить контакты с помощью пайки, не используя для конечного устройства макетную плату, т. е. создать свою собственную.
 - 5) Вкладка **Code** похожа на среду Arduino IDE. В ней тоже можно написать скетч и загрузить его в плату Arduino (рис. 3.21).
 - 6) Сохраните проект. Программа Fritzing использует собственный формат файлов с расширением **.fzz**.

Проект можно опубликовать в онлайн-галерее среды или вывести на печать. При переходе во вкладки **Макетная плата** или **Принципиальная схема** в меню **Файл** становятся доступными функции экспорта, включая форматы изображений PNG, JPEG, SVG и сохранение в виде PDF-файла. В будущем вы можете аналогично оформлять собственные сложные проекты для публичной защиты или сайта.

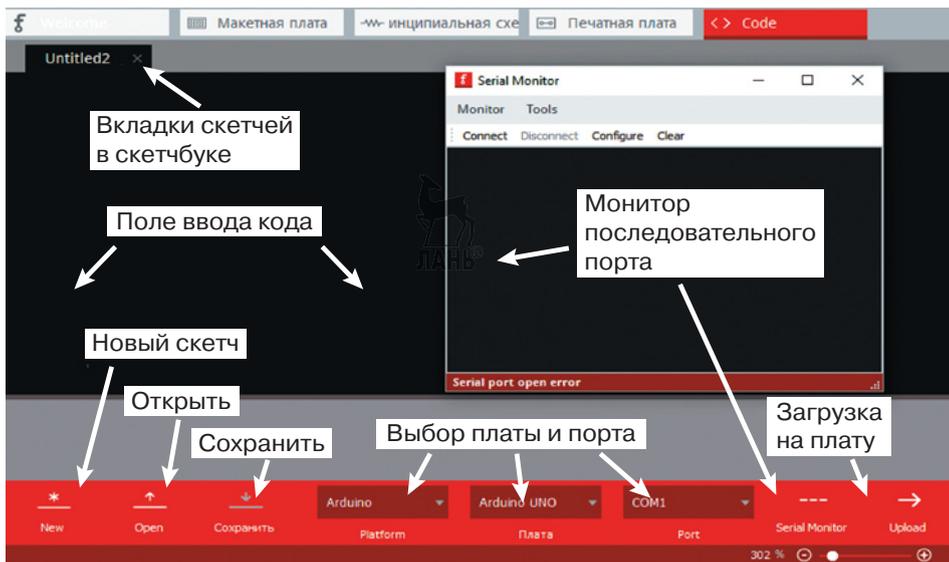


Рис. 3.21. Вкладка Code

Глава 4. Периферия Arduino

4.1. Виды периферийного оборудования

Кроме основной платы с микропроцессором (например, Arduino Uno), для прототипирования требуется *периферийное оборудование* — аппаратная часть, позволяющая получать информацию из окружающего мира или выводить ее. Некоторые физические элементы платформы Arduino уже были рассмотрены ранее.

Существует несколько типов классификаций такого оборудования, например:

1. По способу взаимодействия с окружающей средой:

- *сенсоры* — выдают информацию о положении, освещенности и т. д.;
- *механические компоненты* — совершают какие-либо механические действия (например, сервоприводы);
- *электрические компоненты* — элементы электрических цепей, необходимые для изменения напряжения, сопротивления или силы тока в цепи, для взаимодействия с другими компонентами или внешними устройствами (например, с бытовой электрической сетью);
- *устройства индикации (аудио и видео)* — выводят информацию с помощью звука, цвета, текста или картинки;
- *сетевые интерфейсы* — обеспечивают связь с другими сложными устройствами (например, с платой Bluetooth XBee);
- *источники питания* — внешние источники (например, блок питания на четыре элемента типа АА или один типа «Крона»), позволяющие сделать прототип автономным или обеспечить питание с помощью бытовой электрической сети.

2. По сложности состава:

- однокомпонентные устройства;
- многокомпонентные устройства (сложные датчики и модули);
- платы расширения.



Вопросы

1. По каким признакам классифицируются физические компоненты платформы Arduino? Какие типы устройств участвуют во взаимодействии с миром? Какие компоненты относятся к каждому из этих типов?
2. Какие типы выделяют при классификации по сложности состава?
3. Какие части платформы называются однокомпонентными?

4.2. Однокомпонентные устройства

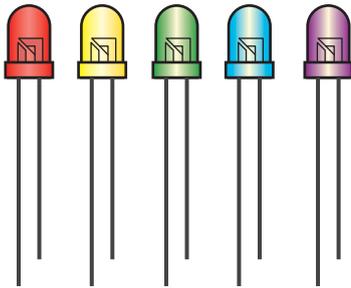


Рис. 4.1. Светодиоды

К *однокомпонентным устройствам* относятся устройства, состоящие из одного элемента. Для их подключения к плате с микропроцессором обычно требуются дополнительные электрические компоненты, чтобы обеспечить необходимое напряжение или сопротивление. Они также являются элементарными компонентами электрической схемы. Примерами однокомпонентных устройств являются

светодиод (рис. 4.1), тактовая кнопка (рис. 4.2), потенциометр (рис. 4.3), а также зуммер (пьезоизлучатель; рис. 4.4), фоторезистор (рис. 4.5) и резистор.



Рис. 4.2.
Тактовая кнопка



Рис. 4.3.
Потенциометр

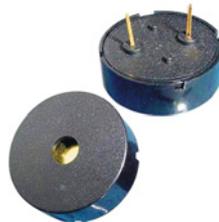


Рис. 4.4.
Зуммер (пьезоизлучатель)



Рис. 4.5.
Фоторезистор

Резистор — это радиоэлектронный элемент, широко применяемый в схемотехнике. Его основное назначение — ограничивать величину тока или напряжения в электрической цепи, чтобы обеспечить нормальный режим работы остальных компонентов электрической схемы, например транзисторов, диодов, светодиодов, микросхем и др. Переменный резистор называется *потенциометром*. На рис. 4.6 показаны условные обозначения резисторов на электрических схемах.

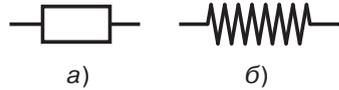


Рис. 4.6. Обозначение резисторов на электрических схемах:

- а) в России и Европе;
б) в США

Это интересно!

Кстати, все условные обозначения, принятые в РФ, можно найти в специальном стандарте «ГОСТ 2.728-74 ЕСКД (Единая система конструкторской документации). Обозначения условные графические в схемах».

Резисторы — это наиболее распространенные пассивные элементы схемотехники. Они различаются по номиналу и мощности рассеяния. При построении схем применяют последовательное, параллельное и смешанное соединение резисторов.

При последовательном соединении n резисторов с сопротивлениями R_1, \dots, R_n общее сопротивление равно их простой сумме:

$$R_{\text{общ}} = R_1 + R_2 + \dots + R_n.$$

При параллельном соединении n резисторов их общее сопротивление выражается более сложной формулой:

$$R_{\text{общ}} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \dots + \frac{1}{R_n}}.$$

Иначе ее можно записать так:

$$\frac{1}{R_{\text{общ}}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \dots + \frac{1}{R_n}.$$

Упрощенные формулы для параллельного соединения двух и трех резисторов:

$$R_{\text{общ}} = \frac{R_1 \cdot R_2}{R_1 + R_2}; \quad R_{\text{общ}} = \frac{R_1 \cdot R_2 \cdot R_3}{(R_2 \cdot R_3) + (R_1 \cdot R_3) + (R_1 \cdot R_2)}.$$

Таким образом, при последовательном соединении резисторов их общее сопротивление увеличивается, а при параллельном — уменьшается.



Рис. 4.7. Маркировка резисторов

Для указания номинала применяется специальная стандартная маркировка резисторов в виде четырех или пяти цветных колец на корпусе резистора (рис. 4.7).

Цвет колец определяется следующими *правилами* (табл. 4.1):

- золотые и серебряные кольца размещаются только на третьей и четвертой позициях;
- каждому цвету соответствует конкретное число;
- допуск показывает класс качества резистора, т.е. на сколько реальные значения сопротивления могут отличаться от заявленного номинала (в процентах).

Таблица 4.1

Цвет	Первое кольцо	Второе кольцо	Множитель	Допуск, %
Черный	Нет	0	1	Не бывает
Коричневый	1	1	10	± 1
Красный	2	2	100	± 2
Оранжевый	3	3	1000	Не бывает
Желтый	4	4	10 000	Не бывает
Зеленый	5	5	100 000	$\pm 0,5$
Синий	6	6	1 000 000	$\pm 0,25$
Фиолетовый	7	7	10 000 000	$\pm 0,1$
Серый	8	8	100 000 000	$\pm 0,05$
Белый	9	9	1 000 000 000	Не бывает
Золотой	Не бывает	Не бывает	0,1	± 5
Серебряный	Не бывает	Не бывает	0,01	± 10

Для примера возьмем резистор с четырьмя кольцами разного цвета:

Первое кольцо	Второе кольцо	Множитель	Допуск, %
Коричневый	Красный	Оранжевый	Золотой
1	2	1000	5

Цифры ставятся подряд, следовательно, речь идет о 12 единицах. Далее следует разобраться с множителем. Оранжевый соответствует 1000, значит, кило-. Итого: 12 кОм. Последний шаг — определение допуска, т.е. на сколько значение сопротивления может отличаться от номинала. Золотое кольцо говорит, что не более чем на 5% в плюс или минус от номинала.

Иногда в электронике встречаются резисторы с пятью кольцами. В этом случае двухзначное число изменяется на трехзначное. Их используют для экономии места, когда требуется указать наиболее точное значение. Например:

Первое кольцо	Второе кольцо	Третье кольцо	Множитель	Допуск, %
Коричневый	Красный	Оранжевый	Оранжевый	Золотой
1	2	3	1000	± 5

Данный резистор имеет номинал 123 кОм и допуск в $\pm 5\%$.



Вопросы

1. Что такое резистор?
2. Как определить номинал резистора по цветовой маркировке?
3. Существует ли разница между последовательным и параллельным соединением резисторов?

Это интересно!

Для быстрого запоминания формулы закона Ома есть секретный способ: закройте пальцем неизвестный элемент и получите готовую формулу (рис. 4.8).



Рис. 4.8. Способ запоминания формулы закона Ома



Практические задания

1. Определите номинал резистора, изображенного на рис. 4.9.
2. Зарисуйте цветные маркировки для следующих номиналов: 100 Ом, 1 кОм, 10 кОм, 200 Ом.



Рис. 4.9

4.3. Простые упражнения для Arduino и S4A

В данном разделе преобладает практическая составляющая. Предлагаем вам выполнить несколько небольших познавательных заданий с возрастающей сложностью. Вы познакомитесь с базовыми радио- и электронными компонентами в действии.

Мигание светодиодом

Мигание светодиодом — своеобразный «Hello, World!» для робототехников. Это многолетняя и приятная традиция, которая буквально означает, что вы здороваетесь с миром электроники!

Компоненты:

- Плата Arduino Uno, 1x;
- Макетная плата для прототипирования BreadBoard Half, 1x (рис. 4.10);
- Светодиод, 1x;
- Резистор, 220 Ом, 1x;
- Черный провод с концами типа штекер, 1x;
- Красный провод с концами типа штекер, 1x;
- USB-кабель, 1x.

Все контактные отверстия на боковых шинах соединены горизонтально по всей длине макетной платы. Шина «+» (питание) обозначена красной линией, а шина «-» (GND, земля) — синей. В центре платы находятся два блока основных контактов, соединенных между собой в ряды по пять отверстий в каждом.

Светодиод (рис. 4.11) — это электроэлемент, который светится при прохождении через него электрического тока. Собственное его сопротивление очень мало, поэтому для ограничения тока через светодиод необходимо последовательно с ним включать резистор, иначе светодиод быстро перегорит. Для удобства определения выводов ножки светодиода имеют разную длину. Напряжение подается на длинную ножку (анод, «+»), а земля подключается к короткой (катод, «-»).

Сборка схемы (рис. 4.12)

1. Подключите катод (короткую ножку) черным проводом к земле (порт GND на Arduino).
2. Подключите красным проводом анод через резистор 220 Ом к питанию с управлением — цифровому порту № 12. Светодиод будет светиться, когда на цифровом порту появится напряжение.

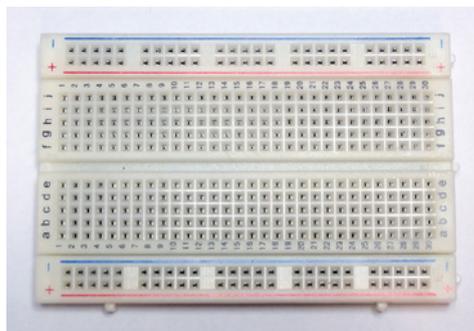


Рис. 4.10. Макетная плата BreadBoard Half

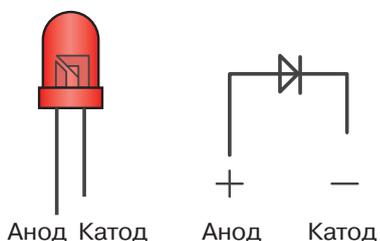


Рис. 4.11. Светодиод и его изображение на схеме

Резистор может находиться как со стороны земли, так и со стороны питания.

Мы подготовили схему во Fritzing.

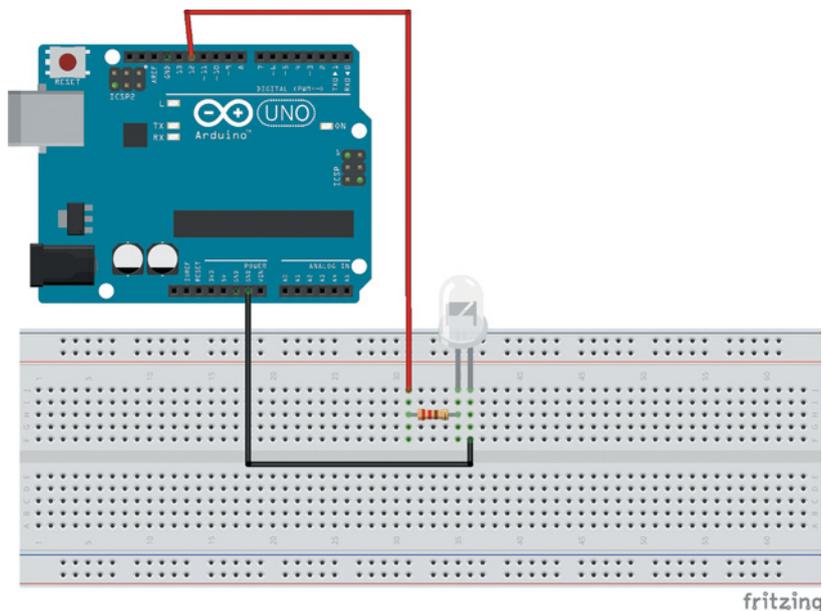


Рис. 4.12. Схема подключения светодиода и резистора

Это интересно!

Arduino Uno позволяет использовать для тех же целей встроенный светодиод. В программе нужно будет изменить лишь номер порта на 13.

Программа

Спрайт и сцена в среде S4A представлены на рис. 4.13.

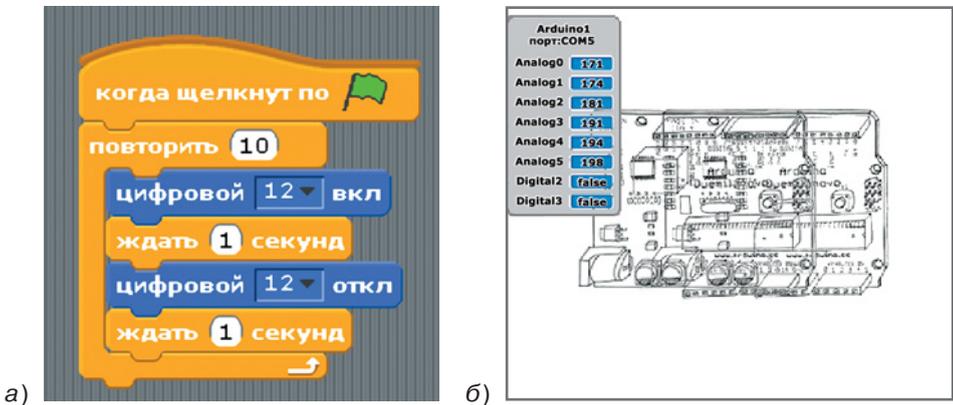


Рис. 4.13. Спрайт (а) и сцена (б) в среде S4A

Соедините Arduino с компьютером с помощью USB-кабеля и нажмите на зеленый флажок, чтобы запустить спрайт.

Маячок с убывающей яркостью

Как уже говорилось в разделе, посвященном знакомству с платой, не любой порт Arduino поддерживает широтно-импульсную модуляцию. Если вы хотите регулировать напряжение, вам подойдут контакты, помеченные символом тильда «~». На Arduino Uno это цифровые входы/выходы № 3, 5, 6, 9, 10 и 11.

Компоненты:

- Плата Arduino Uno, 1x;
- Макетная плата BreadBoard Half, 1x;
- Светодиод, 1x;
- Резистор, 220 Ом, 1x;
- Провод с концами типа штекер, 2x;
- USB-кабель, 1x.

Сборка схемы

Сборка очень похожа на сборку в предыдущем задании, только на этот раз нужен порт с поддержкой ШИМ. Пусть это будет цифровой выход № 9 (рис. 4.14).

Микроконтроллер преобразует уровень в диапазоне от 0 до 255 в напряжение от 0 до 5 В.

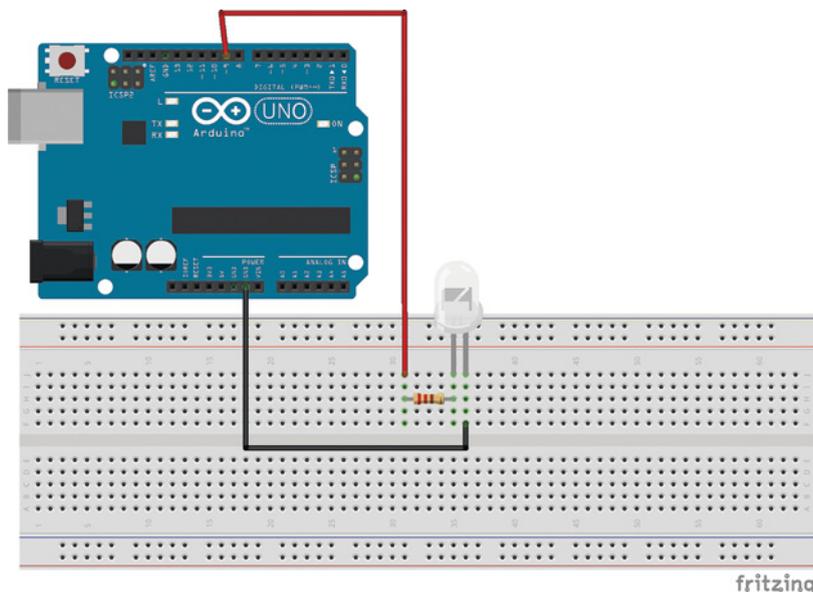


Рис. 4.14. Сборка схемы



Внимание!

Это выполняется с помощью ШИМ, и полученный сигнал не является настоящим аналоговым сигналом. Например, 85 — это $1/3$ от 255, а значит, $1/3$ от 5 В, т. е. 1,66 В.

Программа

Спрайт для мигания маячка с нарастающей яркостью изображен на рис. 4.15.



Рис. 4.15. Спрайт для мигания маячка с нарастающей яркостью

Попробуйте изменить значения по своему вкусу. Перейдем к более интересным заданиям!

Светофор, срабатывающий по кнопке

Компоненты:

- Плата Arduino Uno, 1x;
- Макетная плата BreadBoard Half, 1x;
- Светодиод, 3x (советуем взять светодиоды разных цветов);
- Резистор, 220 Ом, 3x;
- Резистор, 10 кОм, 1x;
- Провод с концами типа штекер, 8x;
- Тактовая кнопка, 1x;
- USB-кабель, 1x.



Рис. 4.16. Условное обозначение тактовой кнопки на электрической схеме

Тактовая кнопка (рис. 4.16) — это контакт, замыкающий цепь при давлении на толкатель, т. е. ключ; подключается через стягивающий или подтягивающий резистор. Для предотвращения короткого замыкания используются резисторы номиналом от 10кОм.

В электрической цепи возникают различные шумы — произвольные изменения напряжения. Они могут внести ошибку в считываемую информацию, когда важно четко различать два состояния: логический нуль (0 В для Arduino) и логическую единицу (5 В). Чтобы убрать шумы, используют дополнительные резисторы. Для гарантии отсутствия напряжения при разомкнутой цепи рядом со входом ставится стягивающий резистор (рис. 4.17). В результате шум будет стекать в землю, т. е. резистор «стягивает» напряжение до нуля.

Если же надо «подтянуть» напряжение до логической единицы, пока внешняя цепь разомкнута, используют подтягивающий резистор. Через него подается питание (рис. 4.18).

Сборка схемы

Сборка данной модели похожа на сборку в предыдущих заданиях (рис. 4.19). Наличие ШИМ на контакте необязательно. В схеме используется стягивающий резистор.

Логика программы

На сцене в среде S4A должны находиться три объекта:

- Светофор
- Плата Arduino
- Кнопка

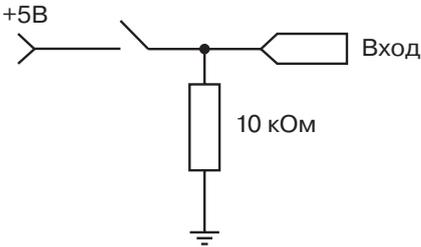


Рис. 4.17. Подключение стягивающего резистора

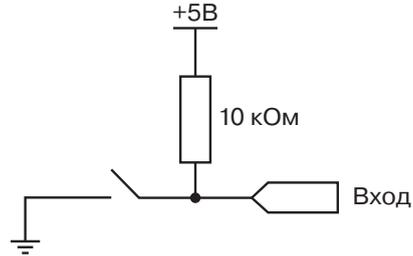


Рис. 4.18. Подключение подтягивающего резистора

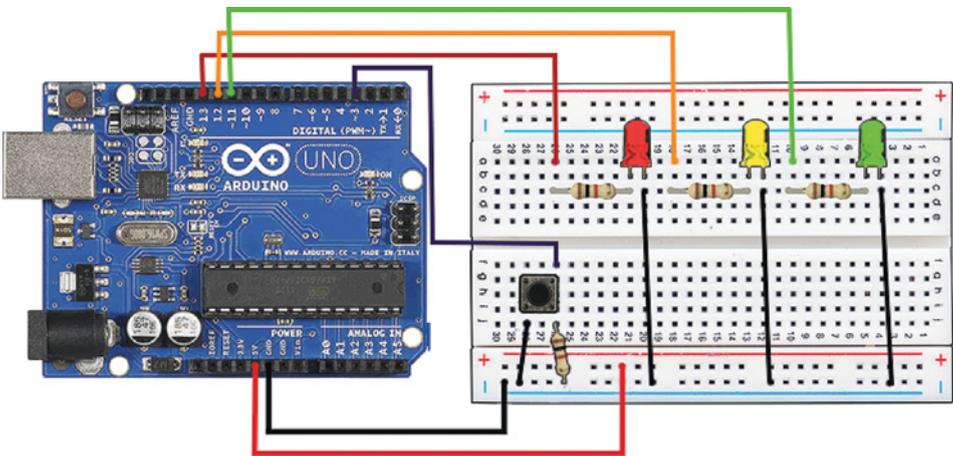


Рис. 4.19. Сборка схемы

При запуске программы должен гореть красный цвет. При щелчке по кнопке красный светодиод гаснет, и на 2 секунды зажигается желтый, а затем зеленый, который горит 3 секунды. После этого снова зажигается красный. Все это происходит как на сцене, так и на макетной плате.

Программа

На самом деле здесь будут задействованы несколько спрайтов — для каждого объекта.

1. Спрайт для кнопки представлен на рис. 4.20.
2. Спрайт «Светофор» содержит три костюма — это фонарики красного, желтого и зеленого цветов, которые должны поочередно появляться. Их смена управляется фрагментом программы, представленным на рис. 4.21.

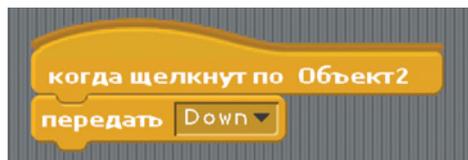


Рис. 4.20. Спрайт для кнопки

3. Осталось обеспечить связь с аппаратной частью проекта. Для управления светодиодами на макетной плате напишите следующий фрагмент для объекта **Arduino Uno** (рис. 4.22).

Испытайте программу. Получилось?

Усложните ее, разместив на плате кнопку, и запрограммируйте ее так, чтобы светодиоды зажигались после нажатия не нарисованной кнопки, а физической. Код программы, конечно, изменится (рис. 4.23).

Самостоятельно измените программу так, чтобы по щелчку мыши нарисованный светофор и светодиоды работали синхронно.



Рис. 4.21. Фрагмент программы для смены костюмов (цветов)



Рис. 4.22. Фрагмент программы для управления светодиодами

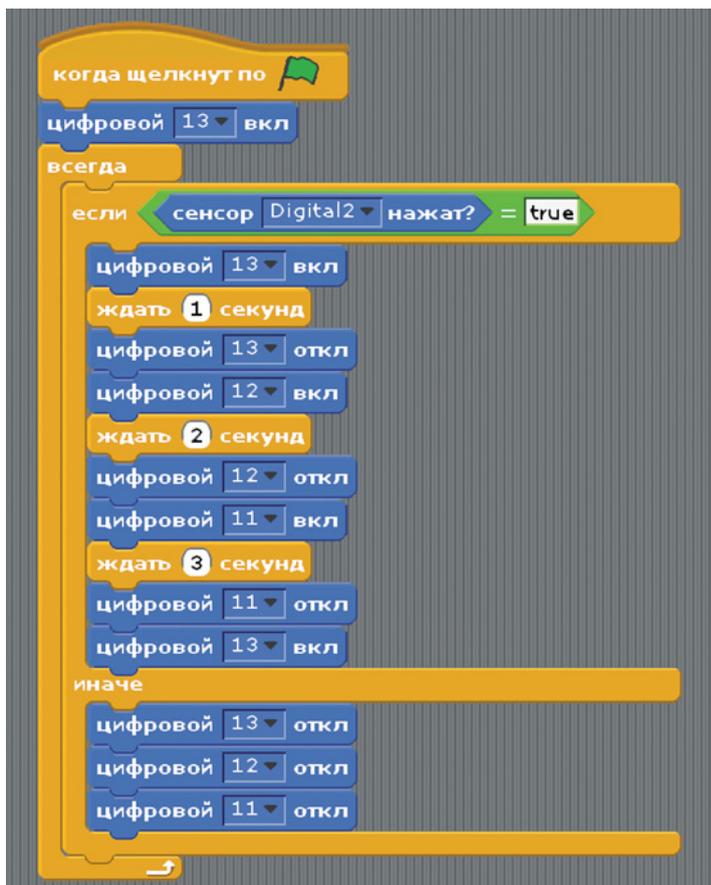


Рис. 4.23. Программа для работы светодиодов по нажатию физической кнопки

Управление движением с помощью потенциометра. Упражнение «Краб»

Компоненты:

- Плата Arduino Uno, 1x;
- Макетная плата BreadBoard Half, 1x;
- Потенциометр, 1 кОм, 1x;
- Провод с концами типа штекер, 2x;
- USB-кабель, 1x.

Перед тем как вы познакомитесь с потенциометром, реализуйте такую задачу: на сцене находится краб, который ползет вдоль

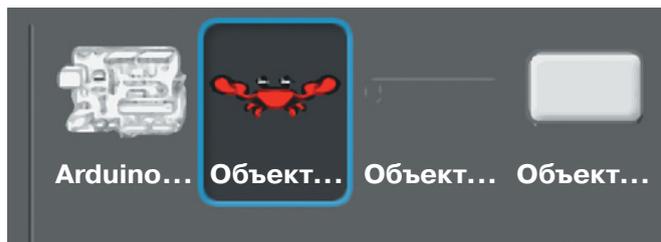


Рис. 4.24. Четыре объекта



Рис. 4.25. Расположение объектов на сцене

трубы. Запрограммируйте его перемещение в трубе по нажатию кнопки.

Всего мы имеем четыре спрайта — по количеству объектов (рис. 4.24). Расположение объектов на сцене будет примерно таким, как показано на рис. 4.25. В спрайте с кнопкой будет следующий фрагмент программы (рис. 4.26):

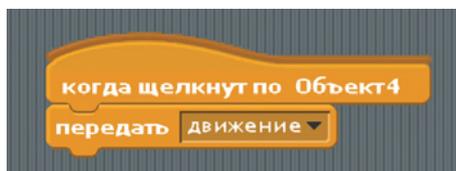


Рис. 4.26. Фрагмент программы для кнопки

Нарисуйте трубу самостоятельно с помощью встроенного графического редактора. Удобнее сначала нарисовать большую трубу, затем подобрать масштаб. Как видите, труба оказалась слишком большой, поэтому установите процентное соотношение от исходного размера посредством команд (рис. 4.27).

В спрайте с крабом пропишите исходную позицию, внешний вид (выбор костюма краба) и команду движения (рис. 4.28).



Рис. 4.27. Установка размеров трубы

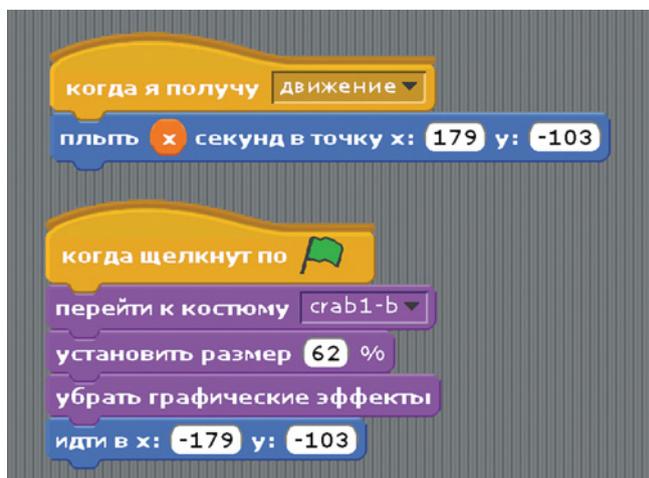


Рис. 4.28. Спрайт для краба

Усложните задачу, создав неудобства крабу. Для этого сделайте так, чтобы диаметр трубы изменялся с помощью ползунка, затрудняя проход по трубе. Чем уже труба, тем труднее крабу ползти и тем ниже его скорость.

Нарисуйте ползунок (рис. 4.29) — это будет новый объект.

Логика спрайта для ползунка следующая. Создается переменная x , в которую помещается координата мышки по оси x .

Обратите внимание, что начальная позиция ползунка (0, 134), а значение переменной x меняется только тогда,



Рис. 4.29. Ползунок

когда указатель мышки находится на ползунке. Если указатель смещается с ползунка, то спрайту с крабом передается сообще-



Рис. 4.30. Код для ползунка

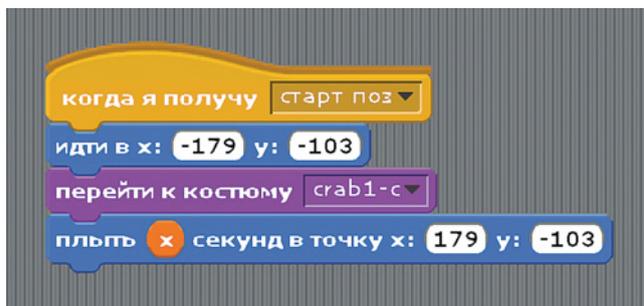


Рис. 4.31. Смена костюма краба

ние с командой: двигаться со скоростью, зависящей от диаметра трубы. Пусть эта скорость равна x единицам в секунду.

Пропишите для нового объекта код, приведенный на рис. 4.30. При этом в спрайте краба происходит смена костюма краба (рис. 4.31).

В итоге сцена получает вид, изображенный на рис. 4.32.

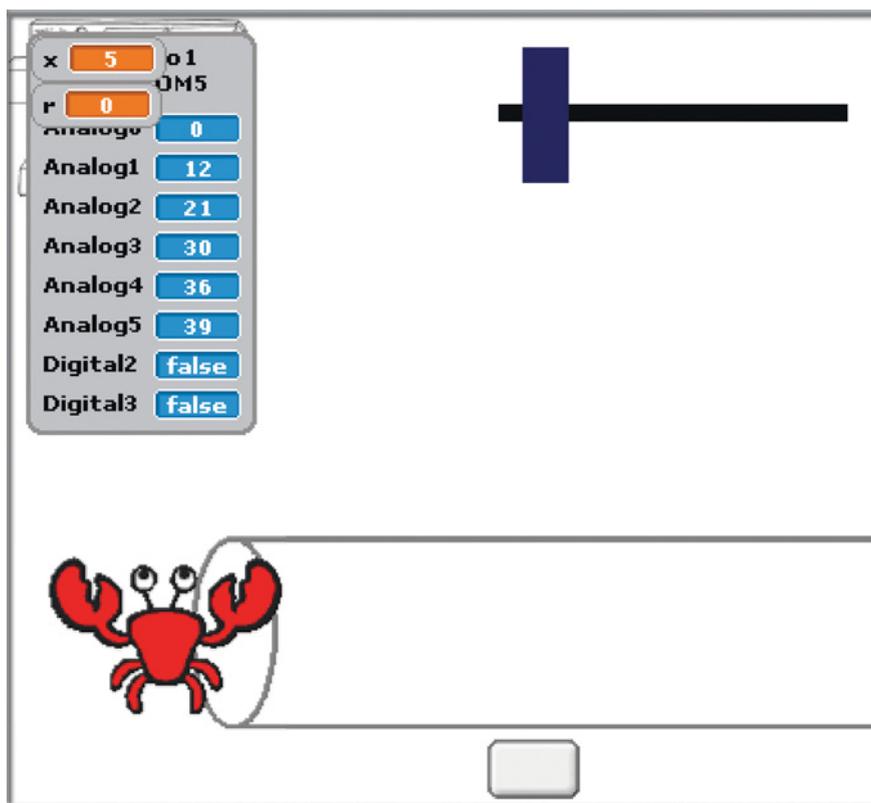


Рис. 4.32. Окончательный вид сцены

Теперь познакомимся с потенциометром и подключим его к макетной плате. *Потенциометр* — это переменный резистор с регулируемым сопротивлением. Потенциометры используются в робототехнике как регуляторы различных параметров: громкости звука, мощности, напряжения и т. п. Условное обозначение потенциометра на электрической схеме представлено на рис. 4.33.

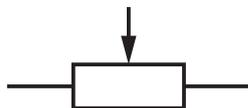


Рис. 4.33. Условное обозначение потенциометра на электрической схеме

Сборка

Замените ползунок реальным потенциометром, который будет выполнять ту же роль — изменять диаметр трубы. Итак, подключим потенциометр к макетной плате (рис. 4.34).

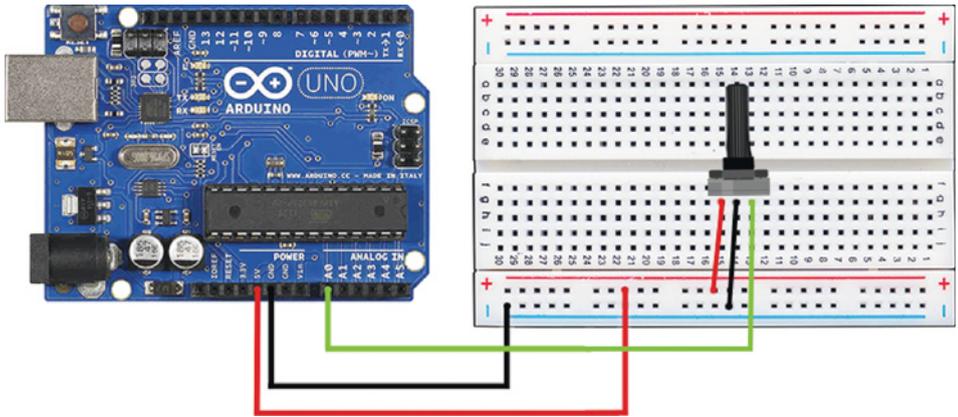


Рис. 4.34. Сборка схемы с потенциометром

Программа

Создайте еще одну переменную r для записи значения с аналогового входа A0 (рис. 4.35).

Разместите еще одну кнопку и в ее спрайте пропишите, что при щелчке по ней должно записываться текущее значение напряжения с потенциометра (рис. 4.36).

В переменную r запишите напряжение с потенциометра. Микроконтроллер выдаст число в диапазоне от 0 до 1023, пропорциональное углу поворота ручки; его необходимо уменьшить, чтобы использовать для анимации. Один из вариантов — разделить полученное значение на 35 и округлить результат до целочисленного. Таким образом, вы провели аналогию между ползун-

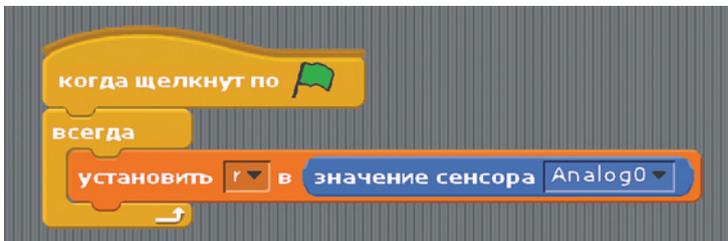


Рис. 4.35. Создание переменной r

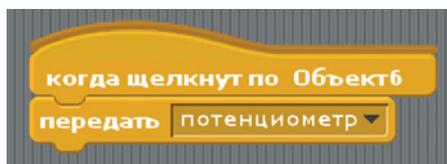


Рис. 4.36. Спрайт для новой кнопки

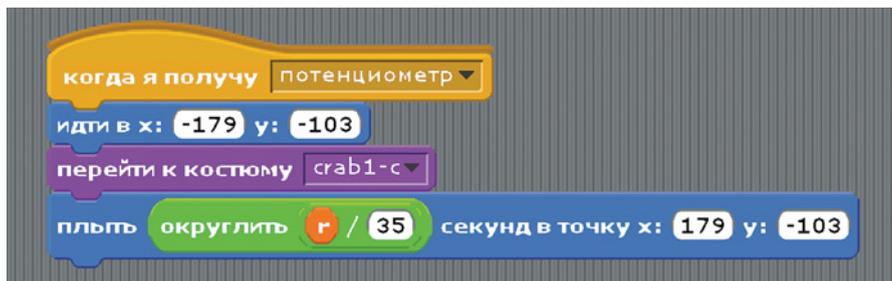


Рис. 4.37. Фрагмент программы на спрайте с крабом

ком, регулирующим диаметр трубы, и работой потенциометра. На спрайте с крабом разместите следующий фрагмент программы (рис. 4.37).

Управление яркостью светодиода с помощью потенциометра

Компоненты:

- Плата Arduino Uno, 1x;
- Макетная плата BreadBoard Half, 1x;
- Светодиод, 1x;
- Резистор, 220 Ом, 1x;
- Потенциометр, 1x;
- Провод с концами типа штекер, 6x;
- USB-кабель, 1x.

В следующей модели от поворота ручки потенциометра будет зависеть яркость светодиода. Это также одна из базовых схем робототехники, а точнее, автоматики.

Сборка схемы

Соедините контакт «земля» потенциометра, а также катод светодиода с шиной «-» на макетной плате. Затем подключите эту шину к контакту GND на Arduino (рис. 4.38). При таком подключении задействуется меньше входов, чем при прямом подключении компонентов к Uno, и от макетной платы к контроллеру тянется меньше проводов.

Неважно, какой из крайних контактов потенциометра будет подключен к 5V, а какой — к GND: поменяется только направление, в котором нужно крутить ручку потенциометра для увеличения напряжения. Запомните, что сигнал считывается со среднего контакта!

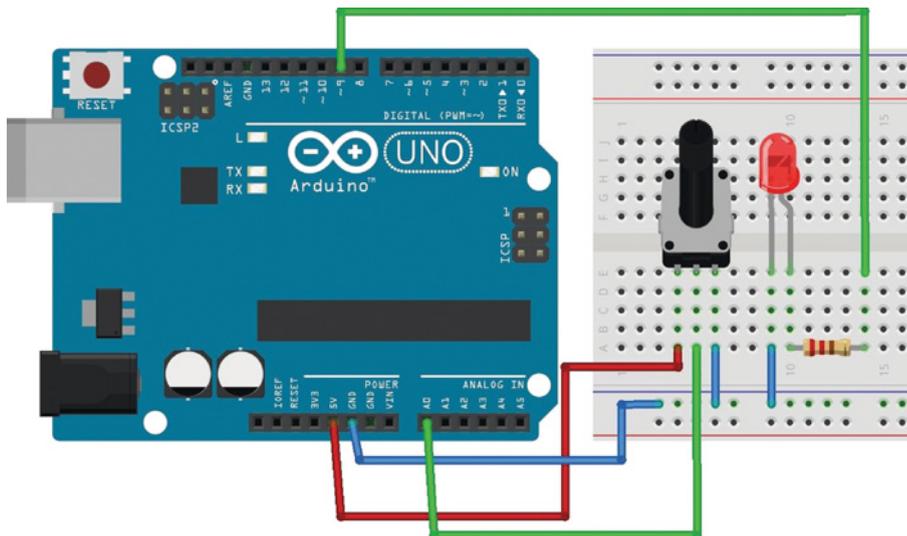


Рис. 4.38. Сборка схемы

Обозначения «+» и «-» на макетной плате не обязывают вас использовать эти шины только как шины питания, однако чаще всего они используются именно так и маркировка помогает не ошибиться.

Логика программы

В разделе переменных объявляются две переменные: *rotat* и *brightness*. Напряжение с потенциометра считывается и сохраняется в *rotat*. Микроконтроллер выдает число в диапазоне от 0 до 1023, пропорциональное углу поворота ручки. В переменную *brightness* записывается полученное ранее значение *rotat*, деленное на 4 (рис. 4.39), так как светодиод имеет 256 шагов яркости (от 0 до 255).

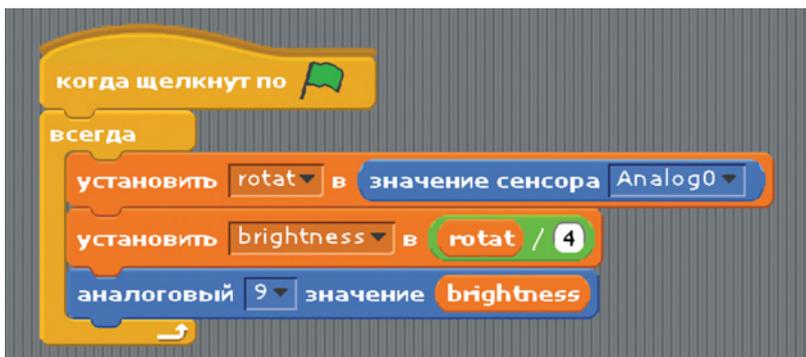


Рис. 4.39. Работа с переменными

Работа с фоторезистором. Упражнение «Робот»

Компоненты:

- Плата Arduino Uno, 1x;
- Макетная плата BreadBoard Half, 1x;
- Фоторезистор, 1x;
- Резистор, 10 кОм, 1x;
- Провод с концами типа штекер, 4x;
- USB-кабель, 1x.

Фоторезистор — это компонент (однокомпонентный датчик), сопротивление которого изменяется в зависимости от интенсивности падающего на него света. Условное обозначение фоторезистора на электрической схеме показано на рис. 4.40.

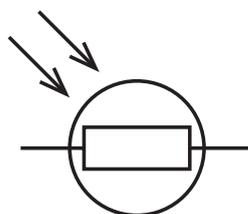


Рис. 4.40. Условное обозначение фоторезистора на электрической схеме

Сборка

Установим фоторезистор на макетной плате (рис. 4.41) и соединим один его вывод с землей, а другой — с аналоговым входом А0.

Далее смоделируем такую задачу. В комнате есть два окна, на одном из которых мы установим жалюзи. Они могут открываться и закрываться по щелчку по кнопке. В комнате обитает робот, который боится темноты. Как только жалюзи начинают закрываться, робот передвигается ко второму окну, на котором нет жалюзи. Когда жалюзи на первом окне открываются, он возвращается к нему, так как света от первого окна больше, чем от второго.

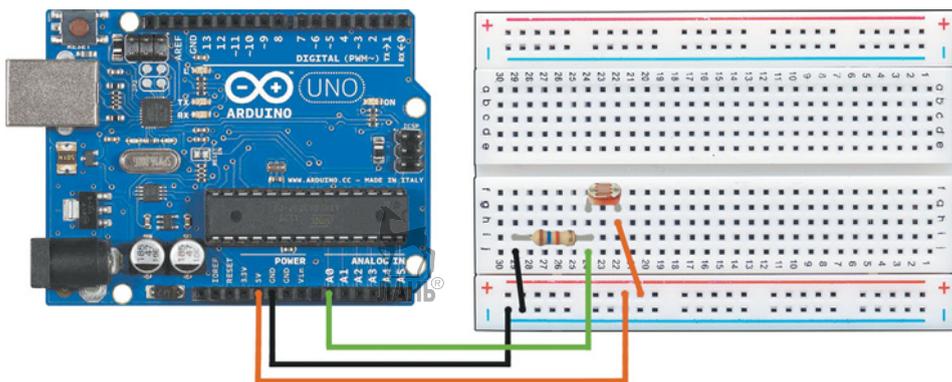


Рис. 4.41. Подключение фоторезистора к макетной плате

Программа

Оформите сцену, добавив фон, и разместите три спрайта: жалюзи, кнопку и робота (рис. 4.42).

Спрайт «жалюзи» содержит 12 костюмов, отличающихся набором секций (рис. 4.43). У робота два костюма (рис. 4.44).

Объявите переменную x , которая будет содержать 1, если жалюзи требуется закрыть, и 0, если их требуется открыть. Для этого для кнопки пропишите фрагмент программы, представленный на рис. 4.45.

В спрайте с жалюзи пропишите код, представленный на рис. 4.46.

При запуске программы задаются начальные значения для спрайтов (робота и жалюзи), затем после щелчка по кнопке передается сообщение, в результате которого происходит закрытие или открытие жалюзи (в зависимости от значения переменной x). Далее роботу передается сообщение, в котором размещен следующий фрагмент программы (рис. 4.47).

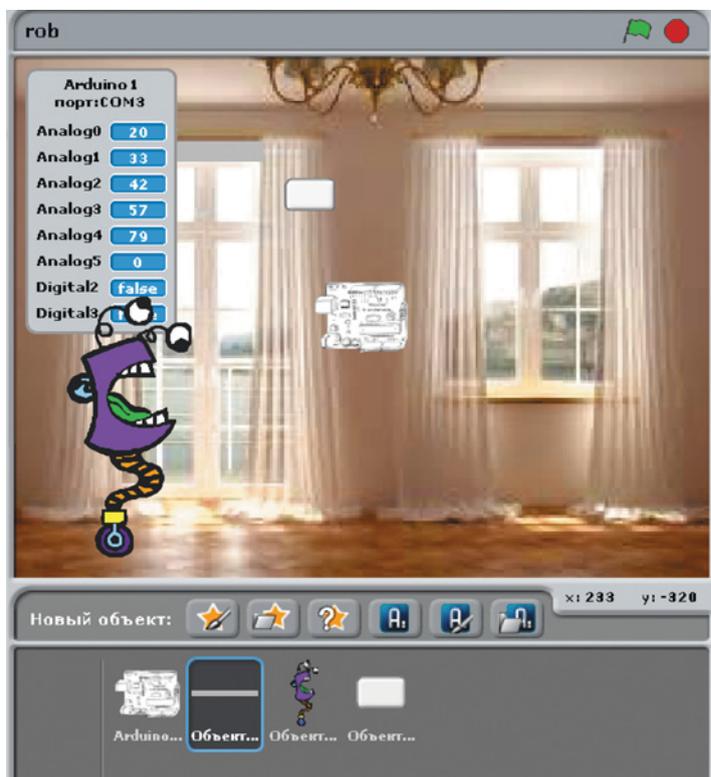


Рис. 4.42. Три спрайта для трех объектов и стандартный спрайт Arduino



Рис. 4.43. Костюмы для спрайта «жалюзи»

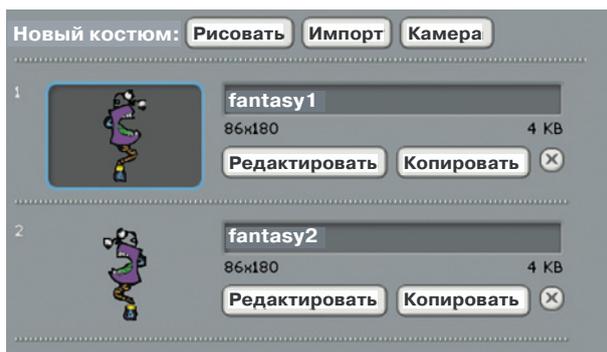


Рис. 4.44. Костюмы для объекта «Робот»

Усложните программу: данные освещенности поступают не от нарисованного (виртуального) окна, а от фоторезистора, размещенного на макетной плате. Для этого создайте переменную *ok1*, в которую будет считываться значение напряжения с фоторезистора. Пропишите фрагмент программы, показанный на рис. 4.48.

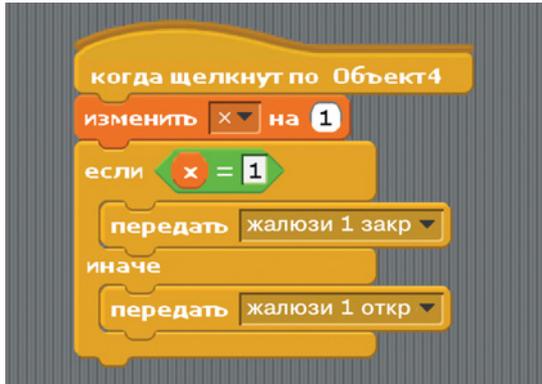


Рис. 4.45. Фрагмент программы для кнопки

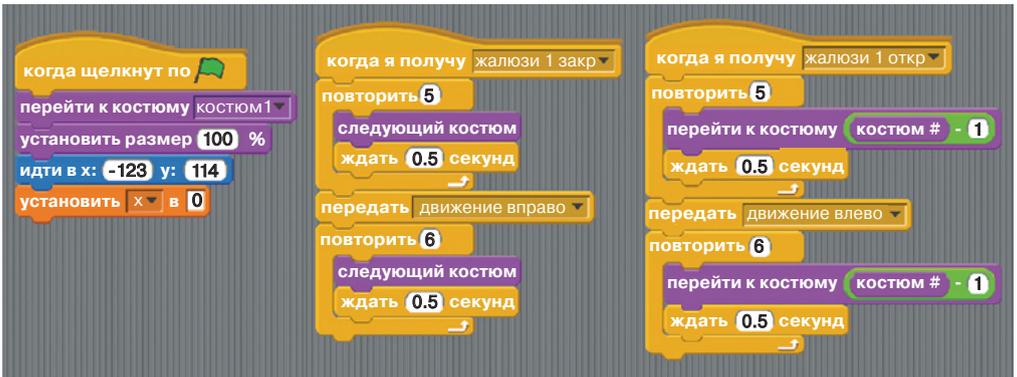
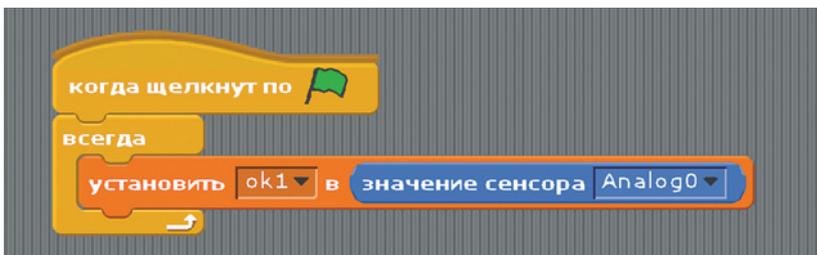


Рис. 4.46. Код в спрайте с жалюзи



Рис. 4.47. Фрагмент программы для сообщения роботу

Рис. 4.48. Считывание напряжения с фоторезистора в переменную *ok1*

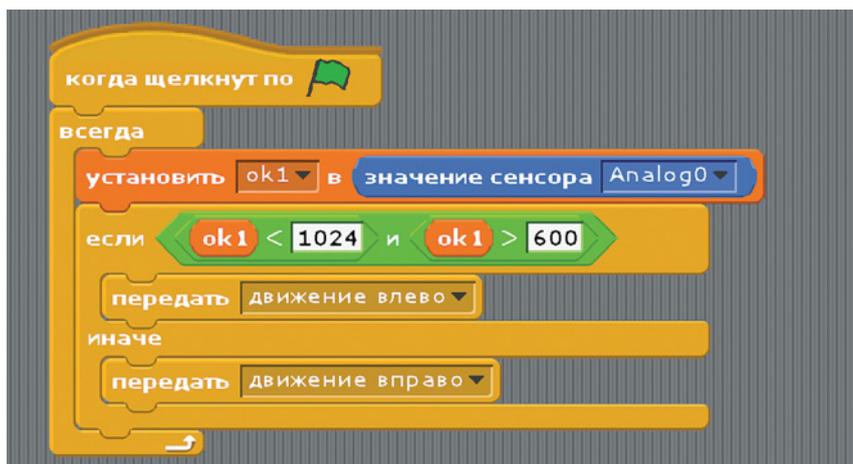


Рис. 4.49. Реакции робота на положение жалюзи



Рис. 4.50. Тестирование программы

В микроконтроллеры ATmega 328, применяемые в Arduino Uno, встроен *аналого-цифровой преобразователь (АЦП)* с разрешением 10 бит. АЦП позволяет считывать напряжение, подаваемое на аналоговые входы, и получать на выходе с ШИМ значения от 0 до 1023 (всего 1024 градации).

Условимся, что если значение переменной находится в диапазоне от 600 до 1023, то жалюзи закрыты и робот передвигается к соседнему окну, иначе (значение переменной < 600) жалюзи открыты. Пропишите код (рис. 4.49).

Включите фонарик и протестируйте свою программу, направляя его свет на фоторезистор и затем отодвигая фонарик в сторону (рис. 4.50).

Полезно знать, что...

Картинки в Интернете, включая фотографии и рисунки, являются объектами авторского (международного) права. Их использование в собственных проектах может быть незаконно. Особенно если вы захотите продемонстрировать свой проект за пределами собственного дома.

Однако существуют ресурсы, которые можно законно использовать, соблюдая ряд указанных условий. Такой контент отмечен символом международной лицензии для творческих сообществ — **Creative Common (CC) Licence**. Сегодня существует уже второй стандарт этой авторской лицензии. Рядом с опубликованным материалом ставятся специальные водяные знаки или символы условий:

-  **CC BY — Attribution:** вы можете использовать, изменять произведение, создавать новое на его основе, даже для коммерческих проектов, однако обязаны указать имя автора исходного произведения.
-  **CC SA — Share Alike:** вы можете использовать и изменять произведение, но, выкладывая свое творение, должны распространять его только под той же лицензией, что и оригинал.
-  **CC NC — Non-Commercial:** вы можете использовать и изменять произведение для учебной, научной и любой иной некоммерческой деятельности.
-  **CC ND — No Derivative Works:** вы можете использовать контент, однако создание производных работ запрещено. Это значит, что картинку нельзя перекрашивать, применять к ней фильтры и тому подобное. Такое условие часто ставят фотографы-ретушеры и некоторые художники, которые не хотят, чтобы настроение и посыл их работы, переданный в свете и цвете, были изменены.



Рис. 4.51. Плашка с условиями распространения продукта

Условий может быть сразу несколько. Обычно они наносятся на специальные плашки. Один из вариантов показан на рис. 4.51.

Особое место занимают произведения, маркируемые **CC0**. Данная лицензия означает передачу произведения в общественное достояние и может быть применена в том числе к программному обеспечению. Мы уже говорили, что Arduino — свободная платформа, аналогичное бывает и в отношении фотографий, рисунков и музыки. Такие файлы можно использовать, транслировать, изменять даже без указания авторства, причем совершенно законно. Главное, не выдавать за свое собственное творчество, если вы не создали производную работу.

Терменвокс

В этом упражнении имитируется звучание музыкального инструмента терменвокс: изменение высоты звука, издаваемого зуммером, на основе информации об интенсивности света от фоторезистора, получаемой бесконтактным путем. Настоящий терменвокс работает тоже бесконтактно, однако принципы его работы сложнее.

Это интересно!

Терменвокс был создан нашим соотечественником Львом Сергеевичем Терменом в Петрограде в 1920 году. Изначально изобретатель дал инструменту имя этеротон («звук из воздуха»).

Существует еще одна интересная вариация инструмента, созданного японским последователем Льва Термена — Масами Такеучи. Его версия терменвокса заключена в корпус-матрешку.

Компоненты:

- Плата Arduino Uno, 1x;
- Макетная плата BreadBoard Half, 1x;
- Резистор, 10 кОм, 1x;
- Провод с концами типа штекер, 6x;
- Фоторезистор, 1x;
- Зуммер (пьезоизлучатель), 1x;
- USB-кабель, 1x.



Рис. 4.52. Изображение зуммера на электрической схеме

Зуммер (пьезоизлучатель, пьезоэлектрический звукоизлучатель) — это компонент, который на основе обратного пьезоэлектрического эффекта воспроизводит звуки определенных частот. Под действием электрического поля происходит механическая деформация материала, вызывающая появление звуковой волны. Обозначение зуммера на электрической схеме приведено на рис. 4.52.

Сборка

Соедините компоненты, как показано на рис. 4.53.

Программа

Программа для терменвокса достаточно простая, поэтому вы можете составить ее самостоятельно (рис. 4.54). Для получения нужной частоты звука разделите значение, поступающее на аналоговый вход от фоторезистора, на константу 4,1. Поскольку для вывода используется порт с ШИМ, потребуется использовать команду «аналоговый».

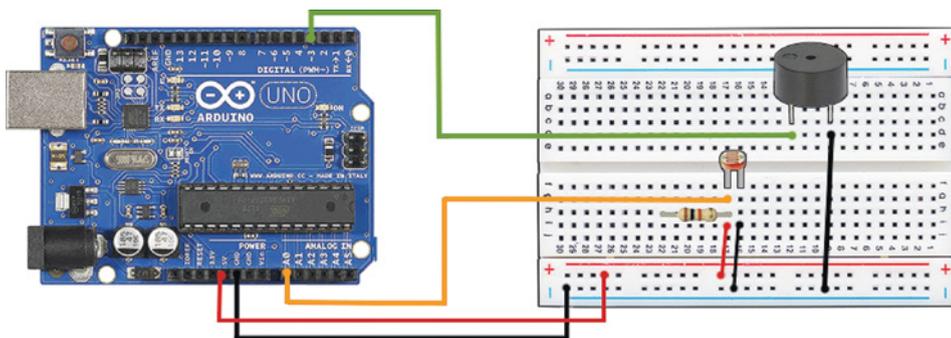


Рис. 4.53. Подключение компонентов



Рис. 4.54. Программа для терменвокса

Ночной светильник

Компоненты:

- Плата Arduino Uno, 1x;
- Макетная плата BreadBoard Half, 1x;
- Резистор, 10 кОм, 1x;
- Резистор, 220 Ом, 1x;
- Светодиод, 1x;
- Провод с концами типа штекер, 10x;
- Фоторезистор, 1x;
- Потенциометр, 1x;
- USB-кабель, 1x.

Это упражнение тоже достаточно простое. Соберите и запрограммируйте модель ночника с автоматическим включением: когда уровень освещенности в помещении падает ниже установленно-

го пользователем порогового значения освещенности помещения, включается светодиод.

Освещенность (E) — это точная физическая величина, измеряемая в люксах (1 люкс равен 1 люмену на квадратный метр) и характеризующая освещение поверхности световым потоком, который на нее падает. Чем сильнее источник света, тем выше освещенность поверхности. При расчете освещенности помещения важно знать габариты помещения: чем оно больше, тем больше требуется источников света. Так, например, в двух одинаковых по площади комнатах, но с разной высотой потолка, освещенность будет разная: где потолок ниже, там расстояние от лампы, висящей на потолке, до пола или человека меньше, а значит, и степень освещенности будет разной. Освещенность помещения измеряется люксометром.

Однако в проектах, связанных с Arduino, в качестве понятия освещенности ошибочно принимают показания, получаемые с фоторезистора — основного компонента люксометра. На самом деле это не является достаточно корректным, поскольку при таких измерениях чаще всего не учитываются размеры помещения и угол падения света на поверхность.

Сборка схемы

Наверное, вы уже догадались, как должна выглядеть схема (рис. 4.55).

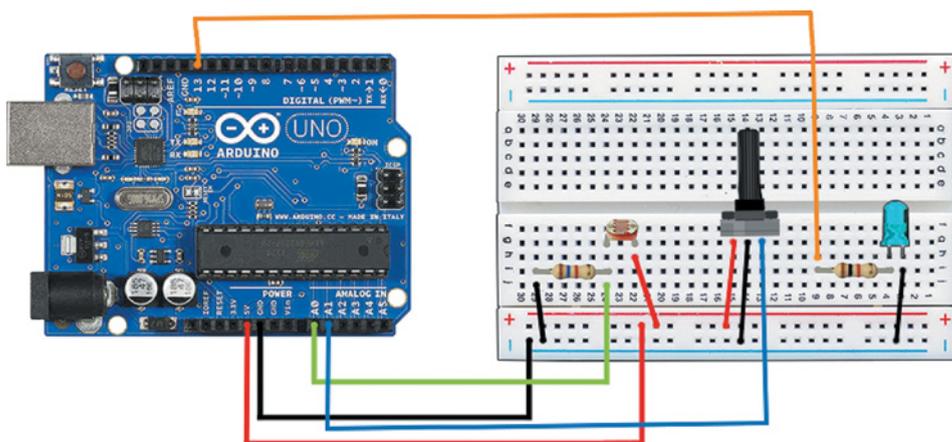


Рис. 4.55. Сборка схемы



Задание

Составьте принципиальную электрическую схему, вспомнив условные обозначения всех используемых компонентов.

Программа

Нужно только задать условия для микроконтроллера (рис. 4.56).



Рис. 4.56. Условия для микроконтроллера

Подобная задача — одна из самых распространенных и простых задач автоматизации. Чаще всего ее решение реализуют без использования сложных микроконтроллеров, обходясь блоком фотореле.

RGB-светодиод

Компоненты:

- Плата Arduino Uno, 1x;
- Макетная плата BreadBoard Half, 1x;
- Резистор, 10 кОм, 1x;
- Резистор, 220 Ом, 3x;
- RGB-светодиод, 1x;
- Провод с концами типа штекер, 7x;
- USB-кабель, 1x;
- Фоторезистор, 1x.

В этом упражнении вы познакомитесь с компонентом, который уже можно отнести к сложным. RGB-светодиод управляется тремя цветовыми каналами. Его название вам знакомо из курса информатики.

RGB (Red, Green, Blue) — это аддитивная цветовая модель, используемая для кодирования цвета путем регулирования по трем каналам интенсивности основных цветов (красного, зеленого и синего соответственно). Такая модель широко применяется для web-графики, ретуши и т. д. Для полиграфических целей чаще используют систему **Panton (PMS, Pantone Matching System)** — стандартизированную систему подбора цвета, основанную на смешении различных красок.

В отличие от обычного светодиода у RGB-светодиода целых три анода — по одному на каждый основной цвет (рис. 4.57). Их следует соединять с контактами Arduino, поддерживающими ШИМ, чтобы иметь возможность получить любой цвет по желанию. Кстати, в RGB-модели в качестве максимального значения используется число 255, а минимального — 0, поэтому для получения нужного цвета не придется производить дополнительные манипуляции с переводом значений. Катод обычно общий. Иногда встречаются RGB-светодиоды с общим анодом.

Сборка схемы

Соедините компоненты так, как показано на рис. 4.58.

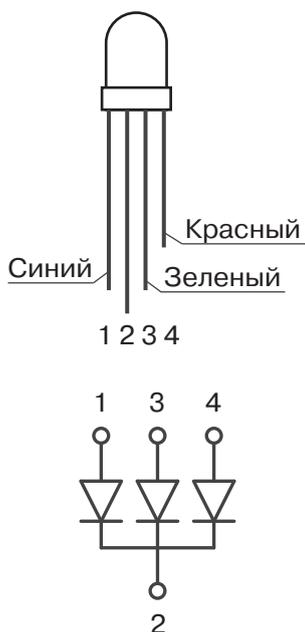


Рис. 4.57. RGB-светодиод с общим катодом

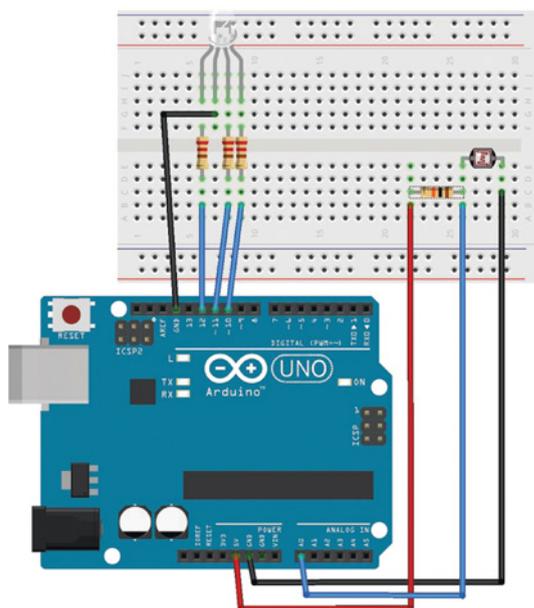


Рис. 4.58. Сборка схемы

Программа

Для написания программы получите данные с фоторезистора (Analog0) и поставьте условие, что при значении 600 (светло) светодиод загорается синим. Кроме того, пусть на сцену программы S4A выводится фон, соответствующий освещенности (рис. 4.59).

На вкладке **Костюмы** выберите пункт **Рисовать** и в редакторе создайте три фона с надписями для трех режимов освещенности.

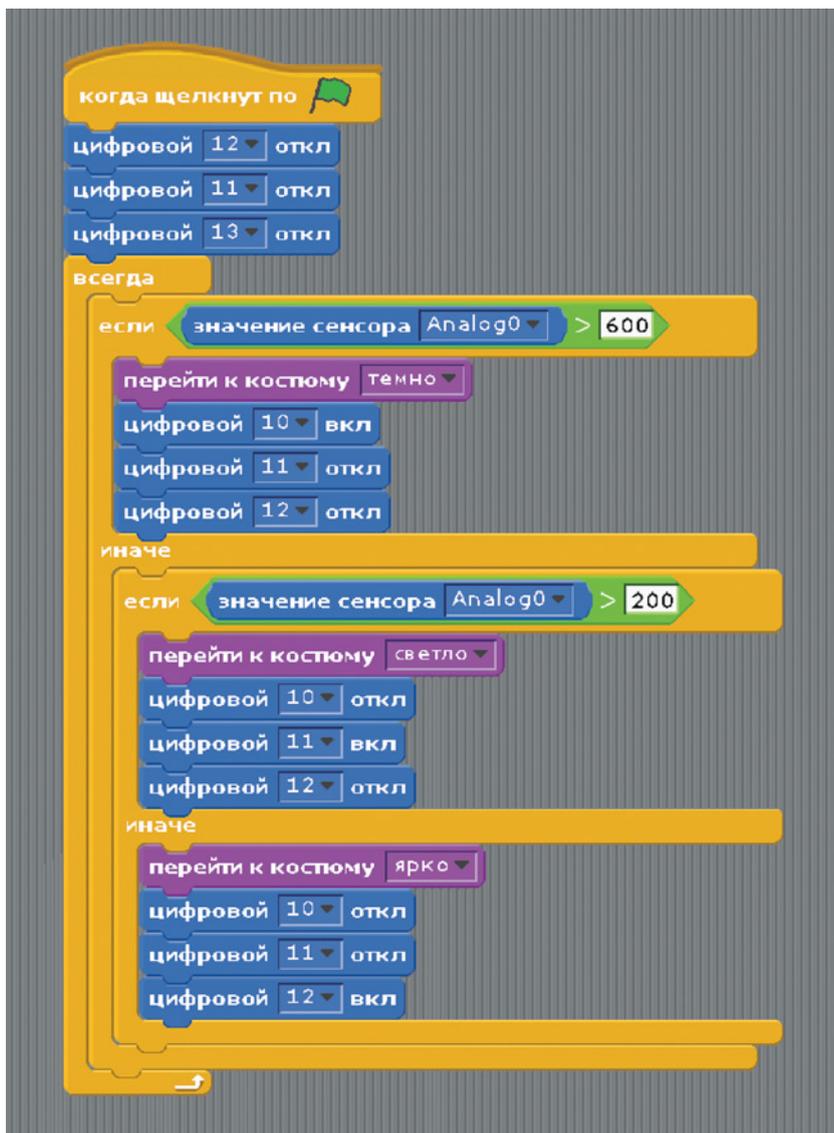


Рис. 4.59. Фон с надписями для трех режимов освещенности

Сахарница

Компоненты:

- Плата Arduino Uno, 1x;
- Макетная плата BreadBoard Half, 1x;
- Резистор, 220 Ом, 1x;
- Датчик наклона HDX, 1x;
- Провод с концами типа штекер, 2x;
- Провод с одним концом типа штекер и одним концом типа гнездо, 2x;
- USB-кабель, 1x.

Датчик наклона HDX основан на элементе, состоящем из корпуса и металлического шарика, перекатывающегося внутри (рис. 4.60). При наклоне сенсора в одну из сторон шарик перекачивается в эту сторону и замыкает цепь. Таким образом, этот датчик позволяет судить только о наличии наклона в какую-то сторону, но не о величине угла наклона. С помощью этого датчика можно отслеживать торможение, падение, вибрацию.

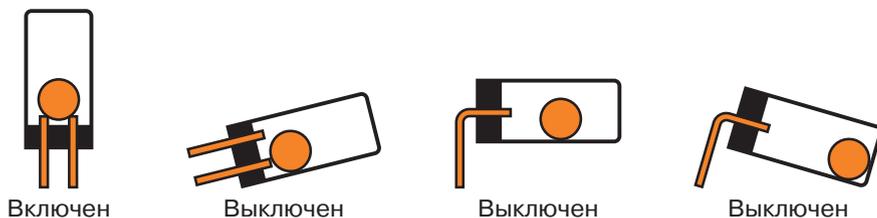


Рис. 4.60. Датчик наклона

Программа

Создайте модель: при наклоне датчика HDX виртуальная сахарница с дозатором наклоняется и сахар высыпается. При исходном положении датчика сахарница возвращается в вертикальное положение.

Для начала разместите на сцене два спрайта — сахарницу и сахар (рис. 4.61). Сахарница имеет один костюм, сахар — целых три. Вы можете нарисовать их самостоятельно с помощью встроенного графического редактора (рис. 4.62).

Создайте переменную y , которая будет увеличиваться на единицу при наклоне или вибрации. Создайте еще переменную x , которая будет содержать количество секунд, в течение которых датчик находится в вертикальном положении, т. е. когда $y = 0$. Если количество секунд превышает 2, то сахарницу нужно вернуть в вертикальное положение. Создайте также переменную z , к которой мы вернемся чуть позже. Для этого в спрайте с Arduino

пропишите следующий фрагмент программы (рис. 4.63). Для поворота сахарницы сбросьте значения x и y в 0. Затем начните проверку наклона датчика. Если он наклонен, то измените значение y на 1 («да»). При первом срабатывании датчика отправьте сообщение сахарнице с командой *повернуть объект*. При актив-

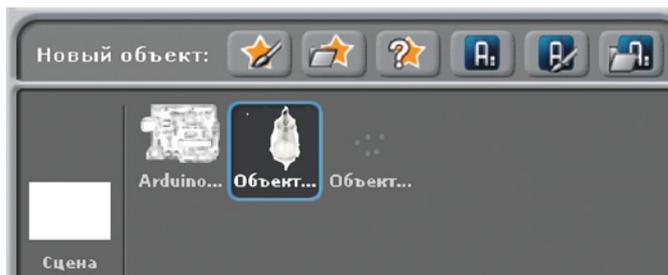


Рис. 4.61. Объекты «Сахарница» и «Сахар»

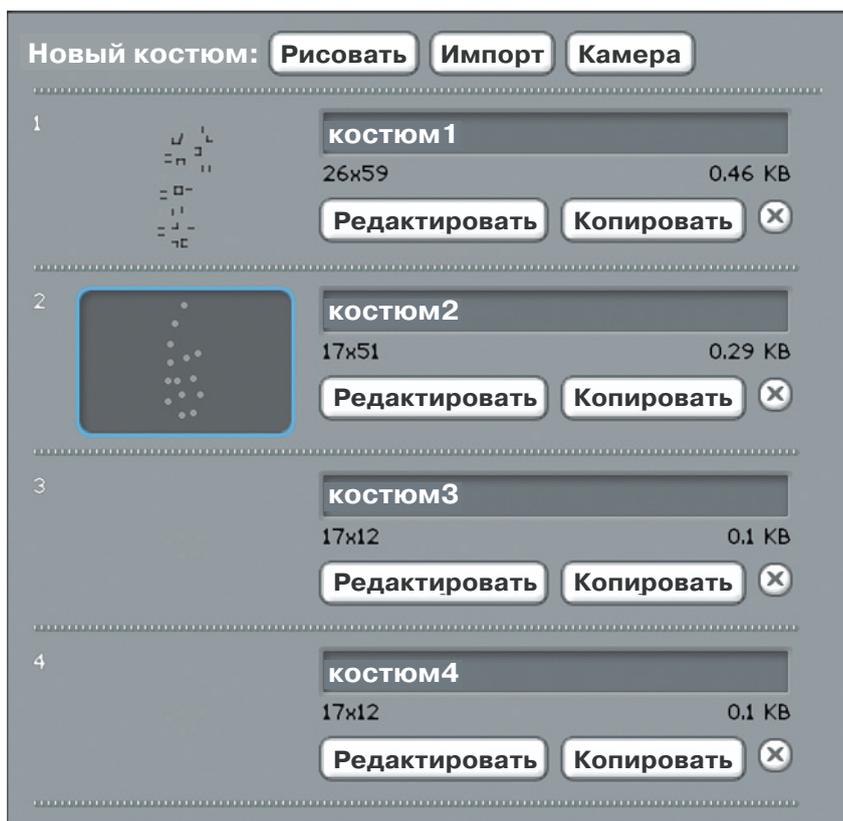


Рис. 4.62. Костюмы для сахара



Рис. 4.63. Фрагмент программы для датчика наклона

ном же состоянии датчика ($y > 1$) спрайту с сахаром постоянно дается команда о смене костюма (что имитирует процесс высыпания). За поворот сахарницы уже будет отвечать переменная z . Правило изменения z будет описано ниже.

При запуске программы задается вертикальное положение сахарницы, при получении сообщения о повороте происходит изменение положения. В спрайте с сахарницей должен быть размещен фрагмент программы, приведенный на рис. 4.64.

В каждый момент, когда $z = 1$, происходит наклон сахарницы и высыпание сахара. Затем запускается таймер. Если время несрабатывания датчика (отсутствие изменения положения) более двух секунд, то сахарница возвращается в положение по умолчанию, т. е. в вертикальное. В спрайте с сахаром напишите фрагмент, приведенный на рис. 4.65.

В итоге сцена должна выглядеть, как на рис. 4.66.



Рис. 4.64. Фрагмент программы в спрайте с сахарницей



Рис. 4.65. Фрагмент программы в спрайте с сахаром



Рис. 4.66. Окончательный вид сцены

Сборка схемы

Датчик наклона подключается к цифровому порту №12. Общий вид установки компонентов представлен на рис. 4.67.

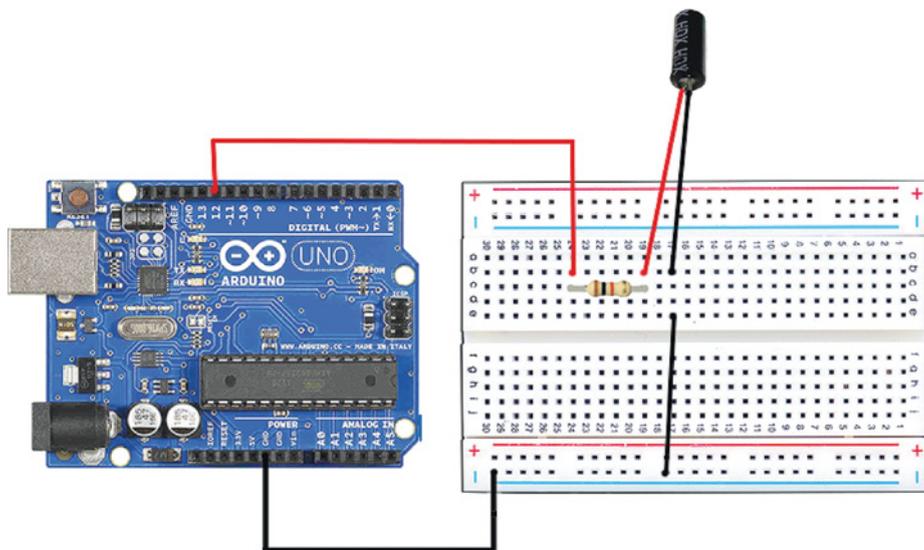


Рис. 4.67. Сборка схемы

Итак, вы выполнили десять упражнений и познакомились с самыми часто применяемыми компонентами для Arduino. Теперь перейдем к сложным датчикам.

4.4. Модули и сложные датчики

К *многокомпонентным периферийным устройствам* относятся модули, сложные датчики, моторы и сервоприводы.

Модуль — это готовая электрическая схема, размещенная на отдельной плате и состоящая из нескольких соединенных элементов. Например, схема датчика освещенности состоит из фоторезистора и обычного резистора на 100 кОм. Использование модулей значительно упрощает работу с платформой, позволяя выбирать готовые решения без страха ошибиться в характеристиках электрических компонентов и собирать проект из наборов. Это удобно для тех, кто хочет заниматься программированием проектов, иногда даже пропуская этап прототипирования. Зачастую при размещении модулей используются специальные платы или переходники вместо макетных плат (для прототипирования или пайки). Таким образом модули просто «насаживаются» в предназначенные для них гнезда, как при работе с платами закрытого типа.

Примером являются Тройка-модули и им подобные (рис. 4.68). Чаще всего они подключаются к плате Arduino с помощью тройного шлейфа от группы контактов SVG, где S (signal) — канал передачи сигнала, например, измерения или управления; V (voltage) — обеспечение модуля питанием (5 В, реже 3,3 В); G (ground) — шина «земли».

Сложные датчики — это устройства, имеющие несколько каналов ввода / вывода, т. е. несколько элементарных датчиков. Например, ультразвуковой дальномер (рис. 4.69) состоит из сонара, излучающего ультразвук, и специального микрофона, определяющего время возвращения отраженного звука (эха). Принцип эхолокации заключается в расчете расстояния до объекта по времени прихода эха с учетом плотности среды (например, воздуха или воды).

На самом деле грань между двумя этими типами многокомпонентных периферийных устройств настолько тонка, что очень часто игнорируется.

Моторы и приводы активно используются не только в спортивной робототехнике, но и для обеспечения подвижности частей различных шлагбаумов, часов, каруселей и многого другого. Существует даже мнение, что робот может считаться роботом только в случае, если он двигается. Это не совсем верно, поскольку робот может быть центром управления систем и при этом его части остаются неподвижны.

Сервопривод (servo) — это привод с управлением через отрицательную обратную связь. Такая связь позволяет точно узнать угол

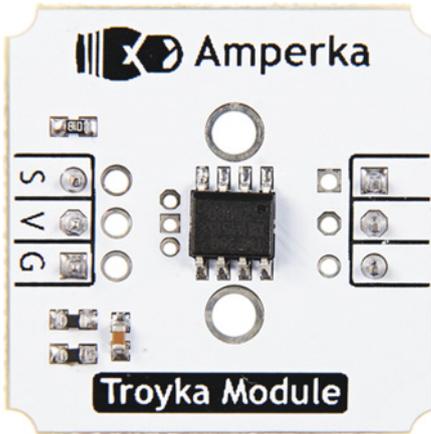


Рис. 4.68. Аналоговый термометр (модуль)



Рис. 4.69. Ультразвуковой дальномер (сложный датчик)

поворота, скорость и наличие внешнего усилия. Каждый сервопривод состоит из двигателя постоянного тока, потенциометра и управляющей платы (со встроенным энкодером). Таким образом, сервопривод может не только обрабатывать поворот на нужный угол с нужной скоростью, но также измерять угол поворота и усилие, приложенное к рычагу, установленному на приводе.

Сервоприводы различаются по мощности (с разным крутящим моментом), форм-фактору и прочности, по величине хода (максимальному углу поворота) и набору крыльев (рычаги, «мельничные» крылья и др.).

Самый распространенный сервопривод среди поклонников Arduino — это маленький сервопривод FS90 с диапазоном угла поворота 180° и крутящим моментом $1,32 \text{ кг} \cdot \text{см}$ при питании 5 В (рис. 4.70).

Шаговые двигатели — это синхронные электродвигатели переменного тока, в которых ток, подаваемый на одну из обмоток статора, вызывает появление вращающегося магнитного поля (рис. 4.71). При этом ротор вращается синхронно с этим полем. При подаче импульса тока происходит дискретное угловое перемещение. Следующий импульс тока вызывает очередной поворот ротора и т. д. Количество шагов, на которые делится круг, чаще всего программно задается пользователем. Также можно задать скорость и направление вращения. Шаговые двигатели редко подключают напрямую, потому что они требуют напряжения, отличного от стандартных 5 В . В связи с этим их запитывают



Рис. 4.70. Сервопривод FS90 с углом поворота 180°

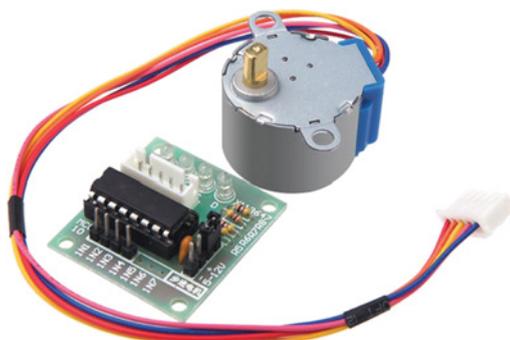


Рис. 4.71. Шаговый двигатель с драйвером

через драйвер мотора — особый модуль, на котором установлен блок управления и преобразователь напряжения.

И сервоприводы, и шаговые двигатели управляются с помощью ШИМ.

Еще один тип модулей, который часто встречается в проектах на Arduino и автоматизации в целом, — это *модули экранов*.

Кратко опишем их основные типы.

1. *LCD* (Liquid Crystal Display) — жидкокристаллический (ЖК) дисплей, обычно монохромный с подсветкой (рис 4.72). Наиболее распространенный вид — знакосинтезирующий, т. е. отображающий только символы, размещенные на нескольких строках. Некоторые поддерживают кириллицу, однако можно создавать и собственные символы в виде массивов. Такие экраны могут

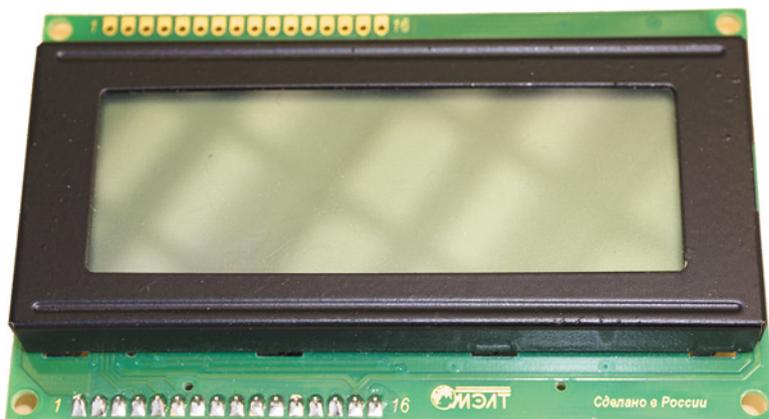


Рис. 4.72. Монохромный жидкокристаллический дисплей



Рис. 4.73. Цветной жидкокристаллический дисплей

иметь встроенный I²C-интерфейс, но встречаются модели и без него. Протокол I²C позволяет обмениваться информацией с платой Arduino, задействуя меньше физических каналов связи, и, следовательно, занимать меньше контактов платы.

2. TFT LCD (TFT, от англ. thin-film transistor — тонкопленочный транзистор) — цветной жидкокристаллический дисплей с активной матрицей (рис. 4.73). Способен показывать не только символы, но и графические объекты. Матрицы различаются по своим характеристикам, дополнительно может быть установлена сенсорная сетка. Модули с TFT-дисплеями для Arduino часто оснащают собственными плеерами и устройствами для чтения карт памяти.

3. LED-дисплей (Light-emitting diode display) — экран, состоящий из светодиодов (рис. 4.74). Такие дисплеи бывают одного цвета или RGB с возможностью программирования смены оттенков и интенсивности подсветки. Отлично подходят для вывесок и приборов, работающих постоянно, потому что имеют маленькое энергопотребление.

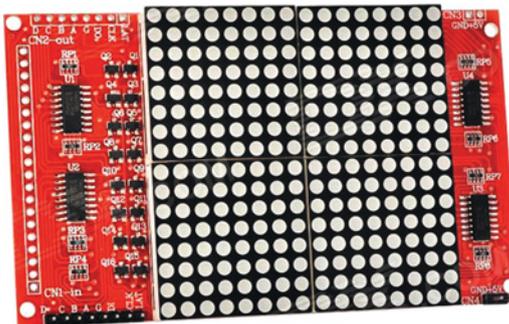


Рис. 4.74. Экран, состоящий из светодиодов

Стоит отметить, что модули и датчики используются для создания обратной связи. Дело в том, что автомат может выполнять действия многократно по одному алгоритму, не зависящему от окружающей среды. В отличие от них роботы выбирают ветвь выполнения программы в зависимости от условий, например от показаний освещенности. Кроме того, роботы должны сами, т. е. без участия оператора (человека), выбирать поведение, что тоже важно.

Управление с обратной связью (анализ окружающей обстановки посредством обработки информации с датчиков и сенсоров) осуществляется в *замкнутой системе управления*.

В противовес ей существует *открытая система управления*, при которой конечное решение принимается оператором: человеком или внешним устройством.



Вопросы

1. Что подразумевается под многокомпонентными устройствами? В чем различие между простыми и сложными датчиками? Что такое модуль? Приведите примеры.
2. В чем различие между сервоприводом и шаговым двигателем?
3. Приведите примеры использования различных типов экранов.

4.5. Применение модулей и S4A

В этом разделе вы познакомитесь с работой датчика уровня жидкости и сервопривода.

Сигнализатор затопления

Платформа Arduino является открытой, поэтому датчики многих производителей, представленные на рынке, имеют идентичный вид и функционал. Датчик уровня (рис. 4.75) имеет такую же группу контактов SVG (S+), как, например, и Тройка-модули.

Контакт с обозначением «-» подключается к земле (GND), средний — к питанию 5 В, а контакт S — к аналоговому входу, например А0. Когда датчик совершенно сухой, напряжение на его выходе и показания на аналоговом входе равны нулю. Чем глубже датчик погружен в воду, тем выше показания (от 0 до 1023).

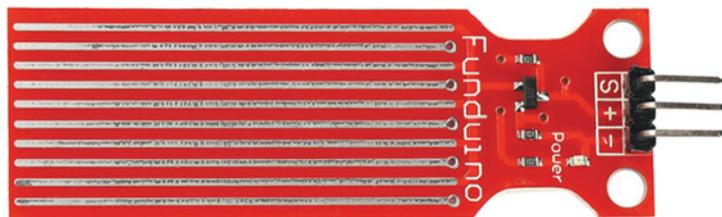


Рис. 4.75. Аналоговый датчик уровня жидкости

Компоненты:

- Плата Arduino Uno, 1x;
- Макетная плата BreadBoard Half, 1x;
- Резистор, 220 Ом, 3x;
- Светодиод, 1x;
- RGB-светодиод, 1x;
- Датчик уровня жидкости, 1x;
- Провод с концами типа штекер, 4x;
- Провод с одним концом типа штекер и одним концом типа гнездо, 3x;
- USB-кабель, 1x.

Сборка

Соберите модель сигнализатора затопления согласно рис. 4.76.

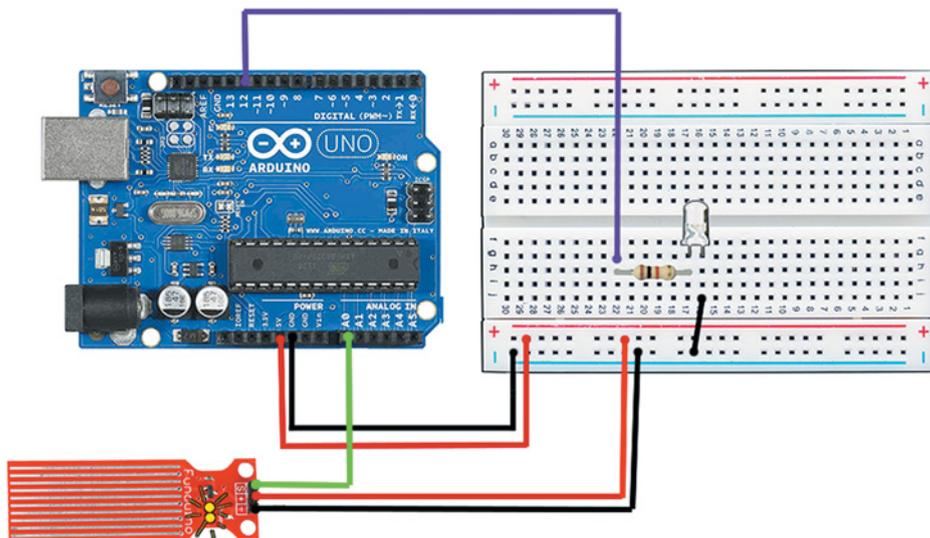


Рис. 4.76. Сборка модели



Рис. 4.77. Функция включения светодиода при превышении порогового значения

Программа

Напишите программу для автоматического включения светодиода при попадании воды на датчик. Для начала необходимо выполнить калибровку датчика, т. е. узнать, какое значение показаний на аналоговом порту соответствует полному погружению датчика в воду.

Создайте переменную x . Она будет использоваться для хранения значения, получаемого с датчика погружения. Задайте собственный пороговый уровень воды и измерьте значение. После того как вы узнали показания датчика при его увлажнении, можно добавить в спрайт функцию автоматического включения светодиода при превышении заданного порогового значения и его выключения, если уровень снизился (рис. 4.77).

Усложните модель, встроив в нее RGB-светодиод. В зависимости от степени погружения датчика в жидкость цвет светодиода должен меняться (рис. 4.78):

- максимальная степень — красный;
- средняя — синий;
- слабая — зеленый.

При подключении RGB-светодиода не забудьте ввести в схему дополнительные резисторы (рис. 4.79)!

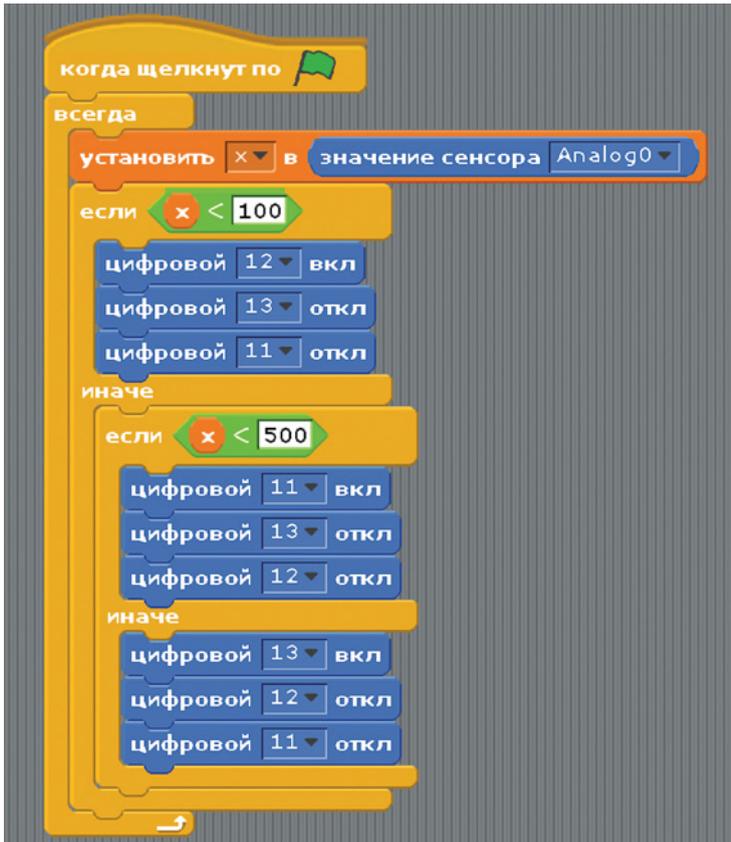


Рис. 4.78. Цвет RGB-светодиода зависит от степени погружения датчика в воду

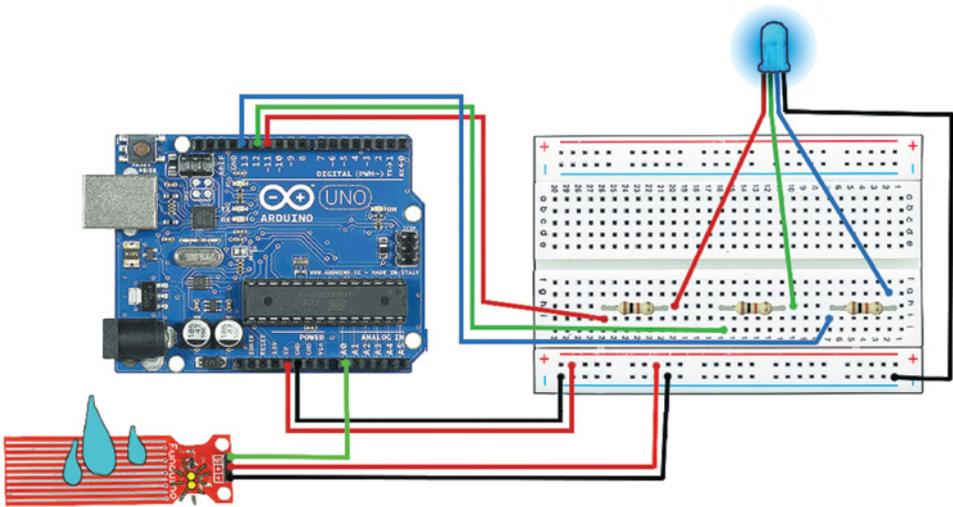


Рис. 4.79. Подключение RGB-светодиода

Сервопривод

Сервопривод — это мотор, положением вала которого можно управлять. От обычного мотора он отличается тем, что ему можно точно задать в градусах положение, в которое встанет вал.

Порядок подключения сервопривода к Arduino описан в табл. 4.2.

Таблица 4.2

Провод сервопривода	Порт Arduino
Черный	GND
Красный	5V
Желтый (оранжевый)	Любой цифровой порт с поддержкой ШИМ

Компоненты:

- Плата Arduino Uno, 1x;
- Макетная плата BreadBoard Half, 1x;
- Потенциометр, 1x;
- Микросервопривод FS90, 1x;
- Провод с концами типа штекер, 6x;
- USB-кабель, 1x.

Сборка

Подключите сервопривод к плате Arduino Uno, как показано на рис. 4.80.

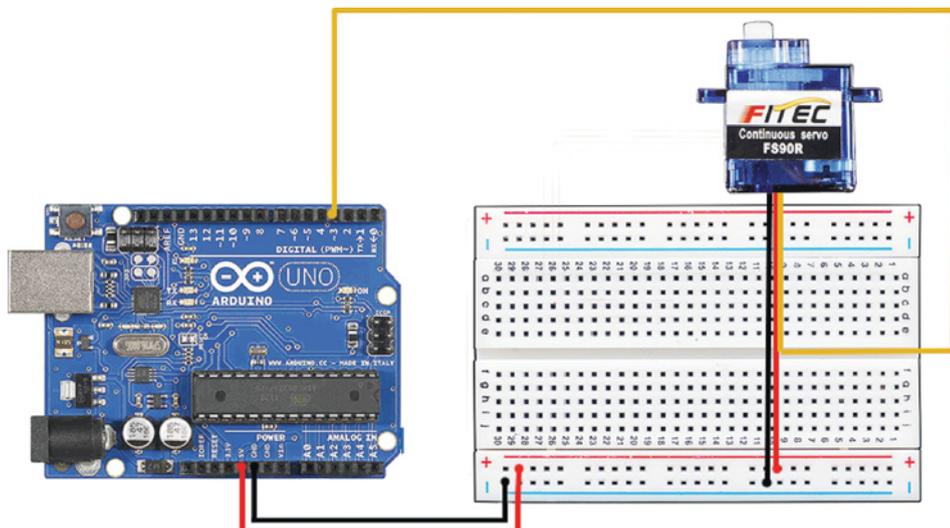


Рис. 4.80. Подключение сервопривода



Рис. 4.81. Код для запуска сервопривода

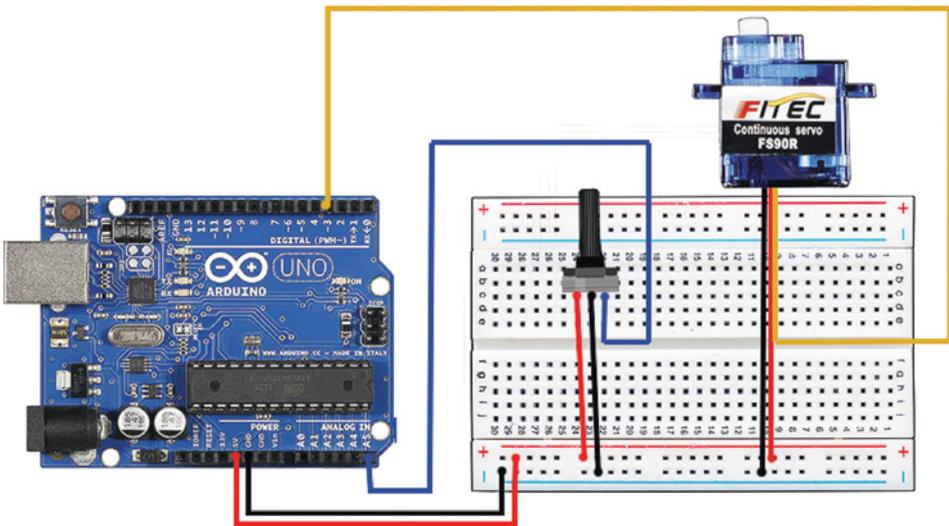


Рис. 4.82. Установка потенциометра

Программа

Для запуска потребуется записать простой код в спрайт платы (рис. 4.81).

Что произошло при запуске кода? Почему цикл не может выполняться всегда? Имейте в виду, что сервоприводы различаются по углам поворота: 180° и 360° , и это следует учитывать в робототехнике при решении конкретных задач.

Углом поворота также можно управлять вручную с помощью потенциометра. Установите его на макетную плату и подключите к модели (рис. 4.82).

С аналогового входа Arduino можно считать значения от 0 до 1023. В схеме с потенциометром задействуются все эти значения. В то же время на цифровой выход Arduino с помощью ШИМ можно подать значения от 0 до 255, а в случае с сервоприводом в команду движения вообще можно передавать значения только до 179, т. е. потребуется пропорционально преобразовать значения из отрезка [0;1023] в значения из отрезка [0;179].

Самостоятельно допишите спрайт, основываясь на уже знакомых вам моделях.

4.6. Платы расширения

Самый сложный тип периферийного оборудования Arduino — это *платы расширения*. Их еще называют *щитами* (shield). Они также являются многокомпонентными устройствами, но существенно отличаются от модулей, поэтому их выделяют в отдельную категорию. Каждая плата расширения представляет собой сложную электрическую схему с возможностью подключения к ней однокомпонентных и многокомпонентных периферийных устройств. В щит иногда бывает встроен дополнительный микроконтроллер, берущий на себя управление специализированными компонентами, чтобы уменьшить нагрузку на микроконтроллер Arduino.

Платы расширения делятся на несколько типов:

- *платы расширения портов*, например Mega Proto Shield или Troyka Shiled; Multiservo Shield (рис. 4.83), расширяют возможности основной платы добавлением портов, иногда собственного формата. К этим портам можно быстро подключать совместимые модули (такие, как Troyka-модули)

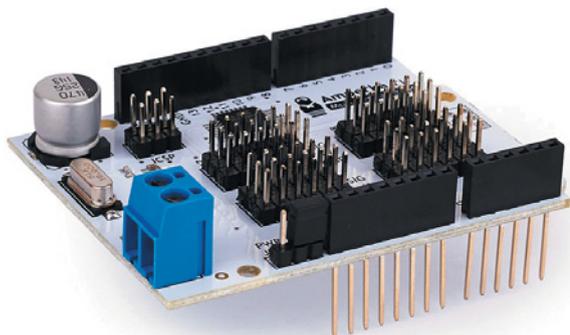


Рис. 4.83. Multiservo Shield

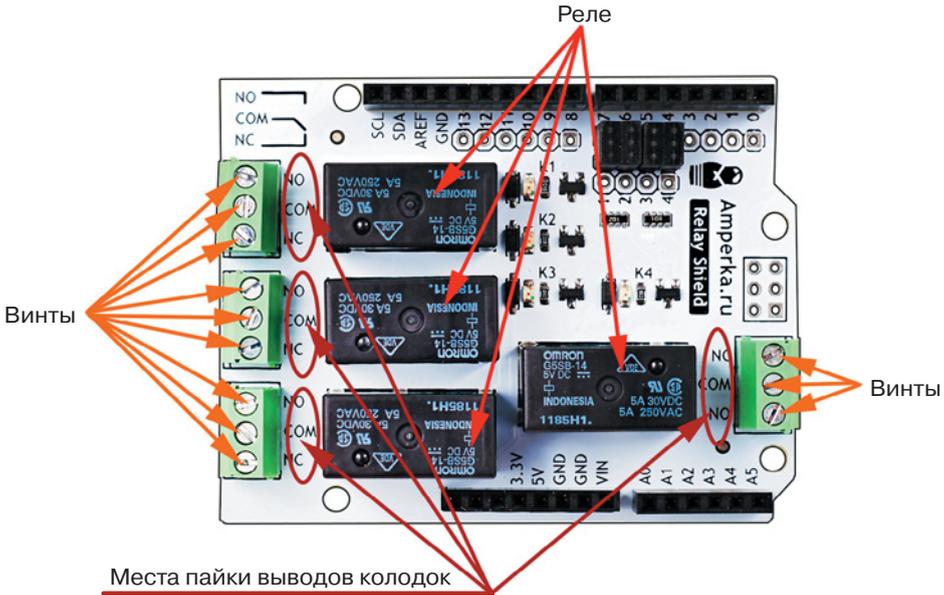


Рис. 4.84. Relay Shield

с помощью всего одного шлейфа (несколько скрепленных проводов);

- платы для взаимодействия с бытовой электрической сетью, например Power Shield или Relay Shield (рис. 4.84), преобразуют напряжение внешнего источника питания, т. е. снабжают слаботочную плату Arduino интерфейсом для управления устройствами сетей с высоким напряжением, например бытовой техникой или потолочным светильником.



Внимание

Работа с бытовым напряжением опасна для здоровья и даже жизни! Ни в коем случае не прикасайтесь руками или отверткой с неизолированной ручкой к винтам контактных колодок, а также местам пайки выводов колодок и реле. НИКОГДА не работайте с платой, если вилка устройства подключена к бытовой сети.



- платы сетевых интерфейсов обеспечивают передачу данных по сети: по последовательному порту через физиче-



Рис. 4.85. Плата GPRS Shield

ские порты других протоколов взамен miniUSB на основной плате:

- *платы беспроводного интерфейса* (например, Wireless Shield, GPRS Shield или Wi-Fi Shield) позволяют подключать Arduino к беспроводным локальным и глобальным сетям для удаленного получения информации с датчиков и модулей в реальном времени, а также для организации сети управления; широко используются при создании гаджетов «умного дома». Плата GPRS Shield (рис. 4.85) позволяет, например, получать данные о критической ситуации с помощью SMS (смс) или через информационные системы оповещения мобильных операторов;
- *платы проводного интерфейса* (например, Ethernet Shield) позволяют включать Arduino в проводные локальные сети или Интернет при наличии доступа к нему из локальной сети;
- *платы взаимодействия с человеком* (например, EasyVR Shield) обеспечивают удобный интерфейс управления, например голосовое управление (распознавание речи).

Щиты устанавливаются на основную плату и обычно занимают все порты. Они могут быть соединены друг с другом (например, Тройка Shield на Wireless Shield), как слои в сэндвиче (рис. 4.86). Именно за это сходство прототипы на Arduino иногда называют сэндвичами.

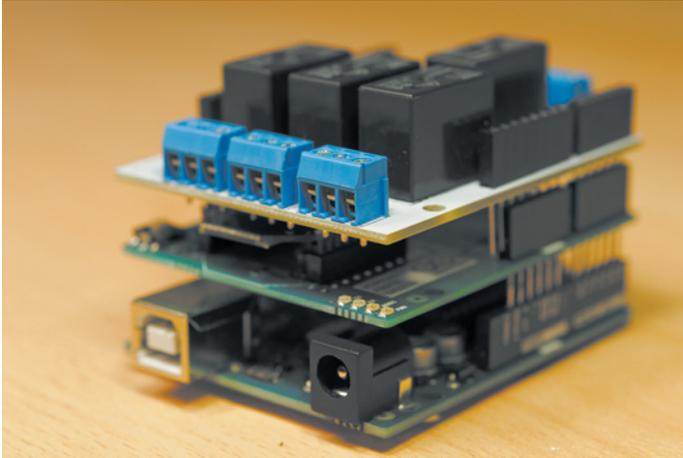


Рис. 4.86. Сэндвич

Практическое задание

Прделаем небольшую практическую работу, чтобы лучше познаться с платами расширения.

1. Возьмите основную плату Arduino Uno и плату расширения Troyka Shield. Соедините их так, чтобы названия выходов на Troyka Shield и Uno совпадали (рис. 4.87).

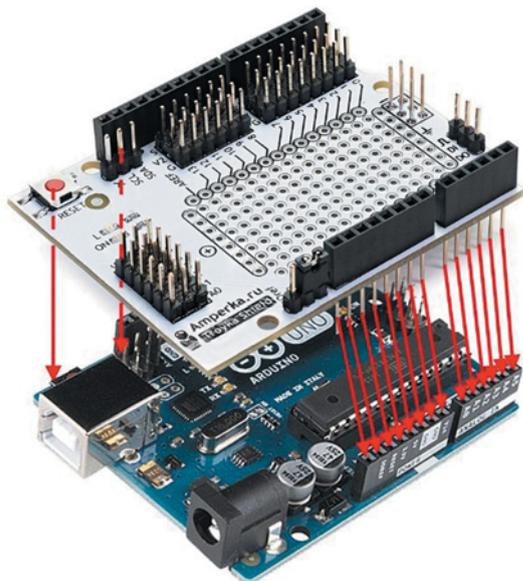


Рис. 4.87. Соедините плату Arduino Uno и Troyka Shield так, чтобы названия выходов на них совпадали



Рис. 4.88. Перемычка соединяет V2 и 5V

Убедитесь, что перемычка рядом с изображением матрешки на плате Troyka Shield соединяет V2 и 5V (рис. 4.88). Это значит, что на плате будет использоваться напряжение 5 В, необходимое для питания Troyka-модулей.

Рядом с блоком цифровых входов/выходов на Troyka Shield расположены *дублирующие порты* собственного формата. Наличие двух дополнительных контактов позволяет подключить питание и землю к модулю без дополнительных проводов. Аналогичный блок дублирующих портов для аналоговых входов расположен на короткой стороне платы.

2. Самостоятельно подключите датчик наклона (рис. 4.89) к дублирующему цифровому порту (входу) № 2, руководствуясь правилом сопоставления буквенных и цветовых обозначений.

Обобщая вышеизложенную информацию, стоит выделить три типа взаимодействия прототипов на Arduino с внешним миром:

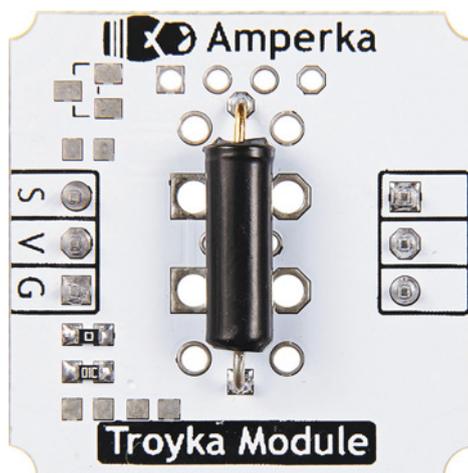


Рис. 4.89. Датчик наклона

- 1) *автономное* — снятие и обработка показаний, вывод с помощью индикации;
- 2) *локальное* — с устройствами бытовой электрической сети, по Wi-Fi, Bluetooth или иному интерфейсу в рамках одного помещения/здания;
- 3) *глобальное* — с помощью сетевых интерфейсов при наличии выхода в сеть Интернет.

Первый тип используется в учебных моделях или в самодостаточных устройствах, не требующих получения данных от других внешних самостоятельных устройств (например, кодовый замок сейфа или устройство автоматического полива растений). Им необходимы лишь данные, которые они получают от собственных датчиков.

Второй тип наиболее распространен в центрах управления «умный» дом. Для них необходим сбор информации о состоянии подключенных в сеть бытовых приборов, гаджетов и элементов бытовой электрической сети, например для управления освещением.

Третий тип встречается как в «умных» домах, так и в небольших устройствах, размещенных в труднодоступных местах или требующих постоянного удаленного доступа к снимаемым ими показаниям. Кроме того, глобальное взаимодействие позволяет корректировать управление устройствами, например, в зависимости от прогноза погоды или для включения оповещения о чрезвычайных ситуациях.

Вопросы

1. Перечислите типы плат расширения и свойства, характерные для каждого типа. Можно ли подключать к Arduino несколько щитов одновременно? В чем различия платы расширения и многокомпонентного устройства.
2. Нарисуйте схему типов взаимодействия прототипов с внешним миром. В качестве дополнительных ветвей выпишите, где применяется каждый тип.

Запомните

- ◆ Периферийное оборудование ◆ Модуль ◆ Сложные датчики
- ◆ Плата расширения (щит)



Практические задания

Задание 1. Сделайте платформу для следования по линии.

Подключите к Arduino Uno платы расширения Motor Shield и Тройка Shield. Сигнал, управляющий движением платформы, должен поступать на мотор после обработки ШИМ. Для реализации остановки используйте показания цифровых датчиков линии.

Присоедините к моторам колеса. Расположите датчик линии по центру на передней стороне платформы на высоте 2 см от пола.

Линию можно распечатать в виде картинки с трассой из Интернета либо сделать с помощью черной изоленты.

Задание 2. Включение лампочки.

Компоненты:

- плата Arduino Uno, 1x;
- плата расширения Relay Shield, 1x;
- провод с патроном и вилкой, 1x;
- острогубцы (кусачки), 1x;
- светодиодная лампа (20 Вт), 1x;
- крестовая отвертка размера ph0, 1x.

Сборка

1. Ослабьте клеммы первого реле.
2. Перекусите одну из жил двухжильного провода, который идет к патрону лампы (рис. 4.90). В этот разрез вы подключите плату Arduino, которая будет служить ключом в электрической цепи. Другая, цельная, жила подключается к земле (GND).
3. Зачистите изоляцию проводника на месте разреза так, чтобы оголенная часть полностью скрылась внутри клеммника. Если провод на конце «распушился», скрутите его.
4. Один конец обрезанного провода, идущий к вилке, вставьте в клемму с надписью N.C. (normally closed) — нормально замкнутый контакт (в рабочем состоянии замкнут, т. е. не разрывает цепь). Затяните клемму. Внутри должны оказаться только медные концы провода без изоляции (рис. 4.91).
5. Другой конец, идущий к патрону, вставьте в клемму с надписью N.O. (normally opened) — нормально разомкнутый контакт (в рабочем состоянии цепь разомкнута). Проверьте провод и затяните клемму (см. рис. 4.91).

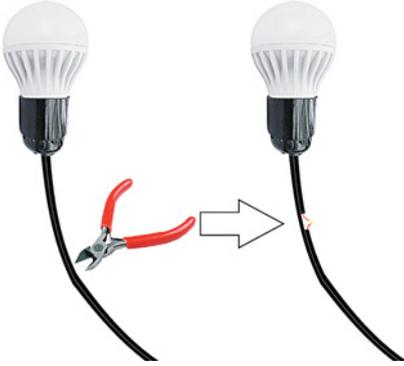


Рис. 4.90. Разрежьте одну жилу двухжильного провода, который идет к патрону для лампы

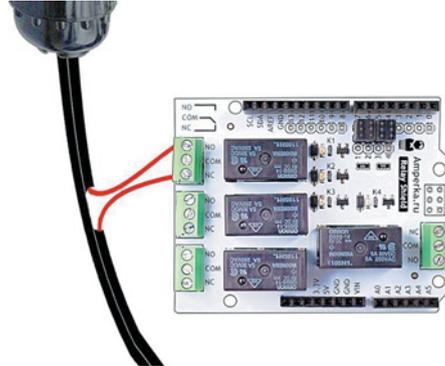


Рис. 4.91. Подключение лампы к плате Relay Shield

Еще раз проверьте контакты — нет ли оголенных проводов, ведь это может привести к короткому замыканию! И помните об опасности работы с бытовой сетью!

Программа

Для управления лампочкой с помощью Arduino вы можете использовать различные варианты спрайтов для S4A. Например, пропишите для спрайта с платой код, показанный на рис. 4.92.

При щелчке по флажку будет производиться запись в переменную *trig* (от англ. *trigger* — переключатель). При нажатии



Рис. 4.92. Код для спрайта с платой

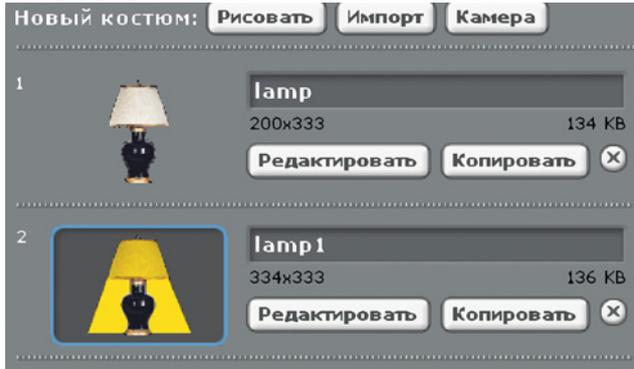


Рис. 4.93. Костюм для объекта lamp



Рис. 4.94. Код для спрайта светильника

клавиши «Пробел» триггер меняет значение на противоположное. Таким образом вы управляете переключателем лампы через цифровой порт № 13.

Добавьте визуальное оформление. Для этого возьмите в библиотеке объект **lamp** и создайте ему второй костюм (**lamp1**) для включенного состояния (рис. 4.93).

В спрайт светильника пропишите следующий код (рис. 4.94). Измените программу по собственному вкусу.

Глава 5. Язык программирования Wiring

5.1. Введение в язык Wiring

Из третьей главы вы знаете, что в первых версиях программной среды Arduino IDE использовался язык программирования Processing. Он основан на Java — одном из самых востребованных современных языков программирования. На нем можно писать серьезные проекты, например большие программы или игры (кстати, популярный движок браузерных игр Unity написан на Java). Однако у Java и его модификаций, включая Processing, существует большая проблема с распределением ресурсов и требованиями к ним.

Микроконтроллеры имеют небольшую свободную память и слабые вычислительные мощности, поэтому было принято решение создать Wiring — *модификацию языка C++*. Точнее, Wiring является *фреймворком* (англ. *framework*) — набором библиотек и правил, частично меняющих основные конструкции и синтаксис языка. Этот язык менее требователен к ресурсам и более стабилен, поскольку оснащен автоматической проверкой ошибок обращения к памяти. C++, в свою очередь, является идейным развитием и упрощением синтаксиса языка программирования C, используемого при программировании управляющих микроконтроллеров на производственных роботах. Wiring получил от создателей новые встроенные функции для простого (с точки зрения программиста) обращения к электронным компонентам прототипа. Новый язык сохранил в себе функциональность предшественников, а снижение сложности написания программ позволило приблизиться к программированию микропроцессоров множеству людей по всему миру. Дополнительными плюсами языка являются проверка и компиляция программы на компьютере, выявление ошибок до этапа загрузки на прототип, а также большое количество готовых решений или их элементов в свободном доступе.

Wiring — это язык высокого уровня, т. е. нацелен на взаимодействие с человеком и программами. Одновременно он способен обращаться к аппаратным компонентам как язык низкого уровня и реализует парадигмы (свод нерушимых правил) объектно ориентированного программирования (ООП) и процедурного программирования (ПП). Следование принципам ООП подразумева-

ет, что Wiring работает с совокупностями объектов, объединяемых по общему, описанному программистом признаку, например принадлежностью к одной аппаратной категории или компоненту. Причем функции и компоненты учитывают наследование. Например, если объявлено получение информации с одной ножки датчика, значит, существует и весь датчик. Парадигма процедурного программирования иллюстрируется наличием нескольких скетчей в одном проекте, т. е. вынесением нескольких функций в подпрограммы (процедуры).

На самом деле под Wiring понимают не только язык программирования, но и платформу полностью, включая одноименный язык, среду разработки и поддерживающие их платы.

Вы познакомитесь с языком Wiring постепенно. Читать код на текстовых языках программирования очень просто. Для этого надо лишь мысленно переводить названия команд с английского языка на русский. Сначала вы познакомитесь с основой языка, а затем, чтобы легче привыкнуть к новому формату, будете читать код, набранный ранее с помощью графических блоков в плагине ArduBlock, синтаксис которого очень похож на уже привычный для вас S4A.



Вопросы

1. На каком языке программирования основан Wiring и почему? Перечислите преимущества, которые новый язык получил от предшественника.
2. Дайте определение языка программирования Wiring. Как в написанных на нем программах отражаются парадигмы ООП и ПП? Расшифруйте сокращения ООП и ПП.

Это интересно!

На языке C++ пишут операционные системы, драйверы устройств, движки браузеров, программы и игры, использующие потенциал системы Arduino полностью, так как данный язык имеет прямую связь с аппаратным обеспечением.

5.2. Программы на языке Wiring: библиотеки и переменные

Программы на языке Wiring строго структурированы. Код всегда начинается с объявления библиотек.

Библиотека — это набор специфических и часто используемых для определенных целей функций. Их полностью описывает раз-

работчик. Пользователь может вызвать функцию, указав лишь ее название и не прописывая весь алгоритм. Если используются команды только из стандартной библиотеки, ее не объявляют. Например, при написании скетча для работы с RFID-метками (как ключи от домофона или стикер на одежде в магазине) необходимо подключить библиотеку Wire.

1. #include <Wire.h>

Название библиотеки всегда заключается в угловые скобки. Служебное слово *include* (включить, встроить) позволяет встроить в итоговый проект файл библиотеки. *Служебные слова* — это ключевые слова, зарезервированные для конкретного языка программирования. Они являются названиями функций, описанных в библиотеках, включая стандартную библиотеку.

Далее необходимо объявить глобальные переменные. *Переменная* — это место, отведенное в памяти компьютера для хранения данных, необходимых для выполнения программы или полученных ею в процессе работы. Чаще всего под переменной понимают некий минимальный объект: одно целое число, одно вещественное число (с запятой), одно слово, одну строку и т. д. *Глобальные переменные* доступны из любого места программы, *локальные* — только из функции, где они объявлены. Для Arduino это особенно важно, поскольку память под глобальные переменные выделяется при включении устройства и записанные в ней значения сохраняются (если их не переписывают принудительно) до выключения питания независимо от количества повторов при исполнении программы роботом.

В Wiring предусмотрены следующие типы данных:

- логический тип (boolean);
- символ (char);
- 8-битное беззнаковое число (byte, от 0 до 255 в десятичной системе);
- целочисленный тип (int);
- целочисленный беззнаковый тип (unsigned int);
- 16-битное беззнаковое число (word, от 0 до 65 535);
- целые числа в расширенном диапазоне (long, от -2 147 483 648 до 2 147 483 647);
- беззнаковые целые числа в расширенном диапазоне (unsigned long);
- вещественное число (float);
- вещественное число в расширенном диапазоне (double);
- массив (array);
- пустой тип (void).

Глобальную переменную можно задать тремя способами.

1. Объявить переменную, указав ее тип и название, не присваивая конкретного значения:

```
3. int x, y, z;
```

2. Объявить переменную, указав тип, название и присвоив исходное значение (инициализировать), которое может быть изменено в ходе выполнения программы:

```
5. int oneWire=10;
```

```
6. float luxMeter=A5;
```

Обратите внимание на 5-ю и 6-ю строки кода. Хорошим тоном считается написание названий переменной на английском языке со строчной буквы, а написание второй и последующих неразрывных частей — с заглавных, если название составное. Пробелы в названиях переменных недопустимы.

А откуда появилось странное число A5? В данных строках переменным присваиваются названия портов, используемых подключенными датчиками. Это облегчает чтение программы в дальнейшем, так как будет сразу известно, где в коде 10 означает число, а где — порт датчика. Для цифровых входов и выходов можно применять номера контактов, для аналоговых — букву A и порядковый номер контакта. В программных строках 5 и 6 прописано, что цифровой выход № 10 будет называться устройством **oneWire**¹, а аналоговый вход № 5 — люксметром (датчиком для измерения освещенности).

```
8. char letter ="a";
```

```
9. char warningMes []="Alarm!";
```

Обратите внимание на объявление символьных переменных. Тип *char* равен строго одному символу. Чтобы ввести или сохранить строку, необходимо создать массив символьных переменных, добавив к названию квадратные скобки. Внутри скобок может быть пусто (компилятор автоматически поставит значение, равное количеству символов + символ конца строки «/0» в ASCII) или введено число (тогда в память будет занесено строгое количество букв без одной). Если число внутри скобок превосходит необходимое, то будет выделена память, в которую может попасть компьютерный мусор. При вводе строки без квадратных скобок в память будет записан только первый символ.

¹ OneWire («один провод») — протокол, предусматривающий передачу данных по одной шине (каналу) от и к устройству. Физически применяется два провода: один для сброса остаточного напряжения (земля), другой для сигнала и питания. Обычно он подключается через резистор на 2,2 кОм. Пример устройства oneWire — пара «домофон и ключ-планшет».

3. Объявить переменную, указав тип, название и присвоив исходное значение, которое не может быть изменено в ходе выполнения программы. Для этого используется служебное слово *const* — константа. Она не может быть изменена никакой командой.

```
11. const int voltage = 220;
```

Иногда константы задают так:

```
2. #define ground 0
```

На самом деле здесь переменная не объявляется, а «подменяется». Служебное слово *define* переводится как «переобозначить». В строке задается правило, по которому слово «ground» будет восприниматься платой как 0. Этот способ удобно использовать для отладки программы с тестовым значением, чтобы, например, заменить в массивах все символы «a» на «b». В сложных математических программах с помощью *define* объявляют константу-коэффициент (например, лямбду), чтобы иметь возможность быстро изменить ее значение.



Вопросы

1. Каковы опорные правила написания программ на Wiring?
2. Что такое библиотека? Перечислите виды библиотек. Дайте определение стандартным словам и переменным. Как можно объявить переменную в языке Wiring? Приведите конкретные примеры.

5.3. Основные функции в языке Wiring

В языке Wiring для Arduino в любой программе всегда присутствуют две функции: *setup()* и *loop()*. Мы опишем их и другие основные функции, используемые в языке Wiring, на примерах программирования простейших операций: мигание светодиодом при включении платы, работа тактовой кнопки, потенциометра, пьезоизлучателя.

Функции — это логически полные блоки кода, выполняющие определенные операции. Они равнозначны этапам использования оборудования, как это встречалось вам в любой «умной» электронике. Например, современные телевизоры при включении показывают логотип производителя, затем работают в режиме показа / переключения каналов.

Функция *setup()*

Эта функция (ее название переводится как «установка») работает один раз после запуска Arduino, устанавливая конкретные режимы используемых портов, скорость последовательного порта и другие параметры, необходимые для инициализации прототипа. При срабатывании она не выдает никакого значения и не требует данных для запуска, поэтому используется тип *void* (в пер. с англ. — пустой) и аргумент в скобках отсутствует. Именно здесь задается скорость обмена данными по последовательному порту (UART). Для этого используется функция *Serial.begin()* из стандартной библиотеки.

```
13. void setup() {  
14.     Serial.begin(9600);  
15. }
```

Для чего нужна точка? Сама функция — это *begin()*, т. е. «начать (связь)». *Serial* — это объект, по отношению к которому применяется функция (serial port — последовательный порт). В скобках дан аргумент, который необходимо присвоить в качестве значения скорости.

Как вы уже знаете, цифровой выход № 13 управляет встроенным светодиодом. Чтобы его задействовать, нужно установить режим порта как выхода (OUTPUT).

```
13. void setup() {  
14.     Serial.begin(9600);  
15.     pinMode(13, OUTPUT);  
16. }
```

Функция *pinMode()* стандартной библиотеки имеет два аргумента: номер порта и его режим. В английском языке «pin» означает «контакт», слово «mode» переводится как «режим». У цифровых портов Arduino бывает два режима: вход (INPUT, напряжение считывается) и выход (OUTPUT, напряжение подается), причем одновременно доступен только один из них.

Функция *loop()*

Дословно ее название переводится как «петля». Функция *loop()* работает аналогично циклу и запускается сразу после работы функции *setup()*, повторяясь снова и снова до тех пор, пока не будет отключено питание платы. Здесь располагается основная часть программы и реализуется взаимодействие с периферийным оборудованием. Функция *loop()* не возвращает никаких значений

после исполнения (кроме изменения переменных внутри себя) и не требует никаких данных для запуска, поэтому она имеет тип *void* и пустой аргумент.

Светодиод подключен к цифровому выходу №13, сигнал на котором может принимать значения логического нуля (LOW) или логической единицы (HIGH), что соответствует состояниям «не горит» и «горит».

```
18. void loop () {
19.     digitalWrite(13, HIGH);
20. }
```

Функция *digitalWrite()* устанавливает сигнал, посылаемый на контакт. В переводе *digital* означает «цифровой», *write* — «записать». В аргументах функции *digitalWrite()* требуется указать два значения: номер порта и значение (LOW или HIGH).

При запуске программы Arduino включит светодиод. При каждом повторении функции *loop()* микроконтроллер будет посылать сигнал о включении. Для получения мигания светодиод выключается с помощью команды *digitalWrite (13, LOW)*.

```
18. void loop () {
19.     digitalWrite(13, HIGH);
20.     digitalWrite(13, LOW);}
```

Однако глаз не заметит это мигание, потому что микроконтроллер обрабатывает команды очень быстро. Нужно увеличить длительность включенного и выключенного состояний светодиода, введя паузу. Для паузы при выполнении программы используется команда *delay()*:

```
18. void loop () {
19.     digitalWrite(13, HIGH);
20.     delay(500);
21.     digitalWrite(13, LOW);
22.     delay(500);
23. }
```

В качестве аргумента функции *delay()* указывается количество миллисекунд, на которые необходимо включить паузу. Достаточно 500 мс = 0,5 с.

Удалите лишние строки, как показано ниже, сформировав следующий скетч:

```
1. void setup (){
2.     Serial.begin(9600);
3.     pinMode(13, OUTPUT);}
```

```

4.     }
5.
6. void loop () {
7.     digitalWrite(13, HIGH);
8.     delay(500);
9.     digitalWrite(13, LOW);
10.    delay(500);
11.    }

```

Скомпилируйте и загрузите скетч на плату. После загрузки сразу начнется выполнение программы: встроенный светодиод будет равномерно мигать.

Команда *digitalWrite()* записывала значение в цифровой порт №13 для вывода обработанной информации. Для получения данных от цифровых датчиков требуется команда *digitalRead()* с единственным аргументом — номером порта, к которому подключен датчик. Слово «read» означает «считать», т. е. команда выполняет считывание с порта.

Примером цифрового датчика является тактовая кнопка. В нажатом состоянии она передает логическую единицу, в ненажатом состоянии — нуль. Устанавливая кнопку на макетной плате, при сборке руководствуйтесь рисунком 5.1.

Подключите к макетной плате красным проводом питание (5 В) и черным проводом землю от контактов Arduino 5V и GND соответственно. Установите на плате резистор на 10 кОм, соединив один из его выводов с правой нижней ножкой кнопки, а другой — с заземленной шиной. Правую верхнюю ножку кнопки соедините с цифровым входом № 2 платы Arduino. Закончите подключение, соединив шину питания с ножкой кнопки, а шину земли — с резистором (см. рис. 5.1).

Необходимо назначить режим порта в скетче. Для этого добавьте строку:

```
4.     pinMode(2, INPUT);
```

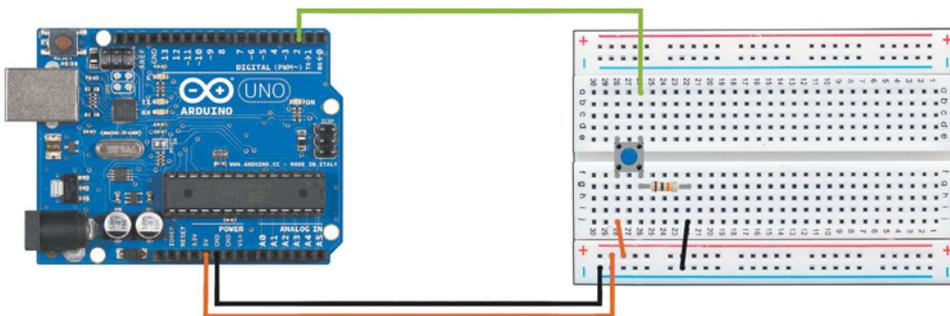


Рис. 5.1. Установка кнопки на макетную плату

Информация будет считываться с порта, поэтому он работает в режиме входа.

В тело функции `loop()` добавьте команду для считывания сигнала. Однако, чтобы отобразить ее в окне монитора последовательного порта, необходимо использовать функцию `Serial.println()`:

```
12. Serial.println(digitalRead(2));
```

Загрузите скетч в устройство и запустите монитор порта. В нем будут отображаться показания, получаемые с тактовой кнопки. Жажмите кнопку и проверьте, что на экране отображается 1.

Для аналоговых устройств используются команды, содержащие слово `analog`, т.е. «аналоговый». Примером аналогового устройства является *потенциометр* — переменный резистор (от лат. *resisto* — сопротивляюсь; пассивный элемент электрической цепи), сопротивление которого регулируется поворотом ручки.

Сборку выполняйте в соответствии с рис. 5.2. Установите потенциометр на макетной плате. Обычно потенциометр имеет три контакта. Красным проводом соедините левый контакт с шиной питания (5 В) на макетной плате. Правый контакт соедините черным проводом с шиной земли. Остается подключить физический канал передачи данных на плату Arduino Uno. Соедините желтым проводом с концами типа штекер средний контакт потенциометра и аналоговый порт A0 платы Arduino.

В данном случае строку выбора режима порта добавлять в программу не нужно, поскольку аналоговые порты работают только в режиме входа. Поэтому сразу начинайте с вывода данных с потенциометра на монитор порта:

```
13. Serial.println(analogRead(A0));
```

Рассмотрите строку 13. Внешняя функция `println()` — это вывод на экран (`print`), завершающийся символом перевода на новую строку (`ln` от англ. *line* — линия). Эту функцию применяют

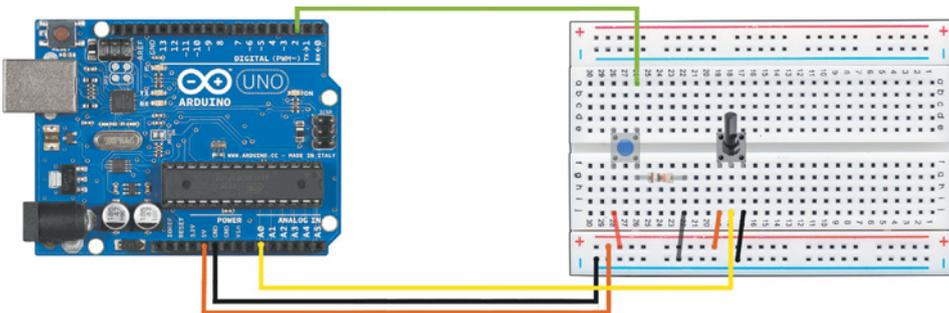


Рис. 5.2. Установка потенциометра

в отношении объекта `Serial` (последовательный порт) и числового аргумента, представленного как результат работы функции считывания показаний аналогового порта `A0`.

Как видите, в качестве аргумента функции может выступать другая функция.

В скетче подряд прописаны две строки с выводом числового значения на экран:

```
12. Serial.println(digitalRead(2));
13. Serial.println(analogRead(A0));
```

Дополните их текстовым сопровождением. У функции `println` аргументы должны быть одного типа, следовательно, нужно создать строковую переменную (типа *String*). Применяйте также *явное преобразование типов данных*. Для этого в языке Wiring достаточно написать название типа как функцию `String()`, а в качестве ее аргумента взять то, что необходимо представить в новом формате:

```
12. String answer = "Button" + String(digitalRead(2));
13. Serial.println(answer);
14. Serial.println(analogRead(A0));
```

Среда Arduino IDE не имеет русскоязычной локализации, т. е. в ней недоступно использование символов кириллицы. Результат выполнения программы можно проверить в окне монитора порта после загрузки скетча на устройство.



Вопросы

1. Что такое функции?
2. Какие функции всегда должны присутствовать в программе на языке Wiring? Что происходит в каждой из них? Сколько раз они выполняются?

Это интересно!

В электронике принято использовать цвета для обозначения функции провода. Так, для провода, подающего питание на устройство, выбирают красный цвет, для обозначения земли — черный, а информационного сигнала — желтый. Также существует способ легкого запоминания буквенных обозначений:

- *GND* — *грунт* — *земля*;
- *V* — *вольтаж* — *питание*;
- *S* — *сигнал* — *информационный сигнал*.

5.4. Функции Wiring и ШИМ. Работа со звуками

Широтно-импульсная модуляция (ШИМ, см. главу 2) помогает переводить цифровые выходы в псевдоаналоговые. Поэтому в языке Wiring существует стандартная функция *analogWrite()*.

Для демонстрации работы ШИМ воспользуемся зуммером (пьезоэлектрическим звукоизлучателем). У него всего два контакта: плюс и минус. Закрепите зуммер на макетной плате (рис. 5.3). Красным проводом соедините контакт зуммера на стороне «+» с цифровым ШИМ-выходом № 3 платы Arduino Uno. Второй контакт зуммера соедините черным проводом с шиной земли на макетной плате.

Переведите цифровой порт № 3 в режим выхода:

```
5. pinMode(3, OUTPUT);
```

Существует два способа воспроизведения звука:

- 1) с помощью функции *tone()* стандартной библиотеки;
- 2) с помощью библиотеки Arduino Tone Library.

Способ 1. Функция *tone()* генерирует сигнал в форме прямоугольной волны заданной частоты. В стандартном виде ШИМ включается на 50% -й рабочий цикл. Функция может иметь два или три аргумента: порт с поддержкой ШИМ, частота звука в герцах и длительность в миллисекундах. Если длительность не указана, функция работает до поступления команды *noTone()*, аргументом которой является номер порта. Загрузите скетч на плату и проверьте выполнение.

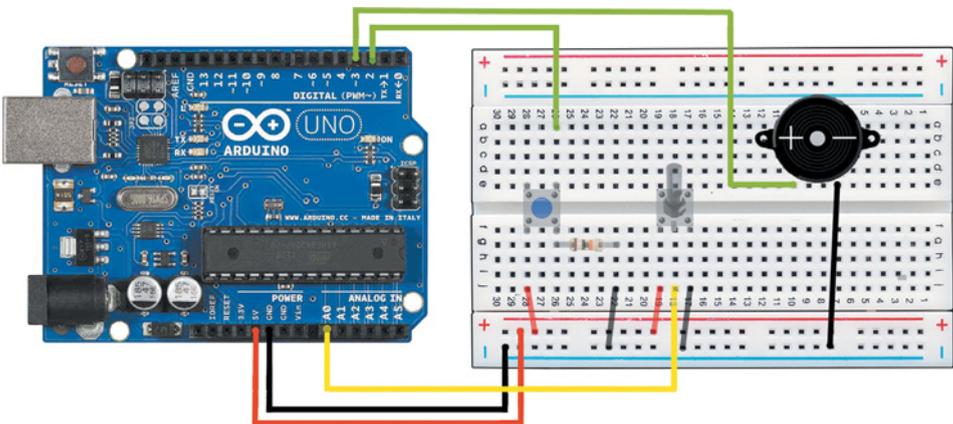


Рис. 5.3. Закрепление зуммера на макетной плате

```

15.     tone(3, 500); //частота 500 Гц (герц)
16.     noTone(3);
17.     tone(3, 700, 500); //частота 700 Гц; 0,5 секунды

```

Здесь появляется однострочный комментарий. Он не обрабатывается компилятором и не влияет на выполнение скетча микроконтроллером. Дополните начало программы подписью, оформленной как многострочный комментарий вида */* абзац текста */*:

```

1.  /* Первый скетч для Arduino.
2.  Плата: Arduino Uno
3.  Компоненты: встроенный светодиод, тактовая кнопка,
4.  потенциометр, зуммер.
5.  */

```

Свяжите между собой работу двух компонентов. Пусть частота тона зуммера зависит от показаний на входе А0, полученных с потенциометра. Для удобства используйте глобальную или локальную переменную, например:

```
23.     int freq = analogRead(A0);
```

Теперь требуется передать это значение зуммеру:

```
24.     tone (3, freq, 500);
```

После загрузки и выполнения скетча слышны паузы, возникающие из-за команд, написанных для управления светодиодом. Удалите лишние команды и ограничение длительности звучания тона в строке 24. Итоговый скетч должен выглядеть так:

```

1.  void setup () {
2.      pinMode(3, OUTPUT);
3.      }
4.
5.  void loop () {
6.      int freq = analogRead(A0);
7.      tone (3, freq);
8.  }

```

Загрузите его на Arduino Uno и убедитесь, что частота звука изменяется при повороте ручки потенциометра.

Способ 2. Чтобы применить второй способ, добавьте библиотеку *Tone* в среду Arduino IDE. Для этого скачайте zip-архив с библиотекой (<https://github.com/bhagman/Tone/archive/master.zip>). Затем выберите вкладку «Скетч», подменю «Подключить библиотеку» и пункт «Добавить .ZIP-библиотеку». После добавления ар-

хива в среду выберите из списка в том же подменю библиотеку *Tone*. В основном эту библиотеку используют для работы с обычным динамиком.

Первая строка скетча должна приобрести вид:

```
1. #include <Tone.h>
```

У подключенной библиотеки, как и у стандартной, существуют собственные специфические функции (по-другому называемые *методами*). Их требуется применять к объектам. Например:

```
1. #include <Tone.h> //Подключенная дополнительная библиотека
2. Tone zum; //Создали объект zum типа Tone
3.
4. void setup (){
5.   zum.begin(3); //Подготовили порт 3 для передачи сигнала
6. }
7.
8. void loop () {
9.   zum.play(NOTE_D1, 500); //Проиграли ноту «ре» объектом zum
10.  delay (1000);
11. }
```

Все изученные *команды языка Wiring* можно разбить на две категории:

- 1) *команды низкого уровня* (взаимодействующие с аппаратной частью и «говорящие на ее языке»);
- 2) *команды высокого уровня* (функции библиотек, а также команды, взаимодействующие только с программными компонентами среды, и т. д.).

Приведенных в этом разделе функций и команд хватит для начального изучения платформы. Для продвинутых проектов могут дополнительно понадобиться следующие команды и функции:

<i>Serial.available()</i>	Возвращает логическую переменную после проверки доступности последовательного порта
<i>Serial.read()</i>	Считывает ровно один байт из последовательного интерфейса (порта)
<i>Serial.write()</i>	Осуществляет побайтовый вывод данных из буфера (временной памяти) последовательного порта
<i>millis()</i>	Значение времени, прошедшего от последнего вызова функции, в миллисекундах

Продолжение табл.

<code>micros()</code>	Значение времени, прошедшего после запуска программы, в микросекундах
<code>randomSeed(seed)</code>	Устанавливает начальное число для функции поиска псевдослучайного числа. Для лучшего результата в качестве аргумента используют сигнал с аналогового входа
<code>random(min, max)</code>	Генерирует псевдослучайное число в заданных границах
<code>pow(a,b)</code>	Возведение числа a в степень b
<code>abs(x)</code>	Модуль числа x
<code>min(x,y)/max(x,y)</code>	Выбор минимального/максимального из двух значений x и y
<code>sqrt(x)</code>	Взятие квадратного корня из x
<code>sin(x)/cos(x)</code>	Вычисление синуса/косинуса (аргумент в радианах)



Вопросы

1. С помощью каких команд выполняется чтение и запись данных цифрового датчика? аналогового датчика? Что требуется указать в теле функции `setup()` при подключении датчиков? Какие режимы порта при этом используются?
2. Что делает стандартная функция `delay()`? В каких единицах отображается ее результат и как она связана со светодиодами?
3. Как вывести значения, полученные от датчиков, на монитор последовательного порта? Как оформляются комментарии в программе?
4. Что такое команды низкого уровня и команды высокого уровня.



Запомните

- ◆ Wiring ◆ Библиотека ◆ Служебные слова ◆ Переменная
- ◆ Функции



Практические задания

Задание 1. Напишите скетч, выводящий на монитор последовательного порта знаменитую фразу: «Hello, world!»

Задание 2. Напишите скетч, выводящий на одной строке показания потенциометра и время, прошедшее после включения платы.

Это интересно!

Среди Arduino-совместимых платформ особое место занимает *Espruino*, которая, как вы уже знаете, использует язык программирования **JavaScript**. Работа с ним имеет ряд преимуществ. Данный язык предназначен для написания специальных *сценариев* работы с объектами. Сценарий JavaScript отличается от программ на языке Wiring. Во-первых, он не имеет строгой последовательной структуры расположения элементов в сценарии. Во-вторых, компилируется и выполняется построчно, а не весь сразу. Такая особенность JavaScript полезна при использовании языка в качестве функционального дополнения HTML — гипертекстовой разметки страниц в Интернете. Даже если скорость передачи была низкой или произошла ошибка в получении данных, пользователь увидит часть, которая успела загрузиться.

Веб-разработка тесно связана с дизайном. Зачастую описание действий сайта выполняет дизайнер. Например, ему может понадобиться вставить слайд-шоу или создать плавные переходы между вкладками при наступлении определенных событий. Для того чтобы дизайнер справился собственными силами, JavaScript создан легким для понимания и применения. Поэтому гаджеты с управлением под JS удобно связываются с различными веб-сервисами и отлично подходят для создания метеостанций или «умных» камер наблюдения.

5.5. Графические блоки и код в ArduBlock

Первое, что нужно сделать, чтобы использовать графическое программирование в среде Arduino IDE, — установить дополнительный **плагин (plug-in)**. Это независимо компилируемый модуль, который представляет собой расширение функций и встраивается в основную программу.

ArduBlock — это одновременно и графический язык программирования, и плагин, переводящий программы, написанные на этом языке, в код на Wiring.

Итак, установите плагин, следуя инструкции:

1. Скачайте плагин ArduBlock с веб-страницы разработчиков: <http://sourceforge.net/projects/ardublock/files/> Выберите предпоследнюю или последнюю версию. Мы рекомендуем найти версию максимального объема среди последних пяти, поскольку она является стабильной и содержит больше блоков. На момент напи-

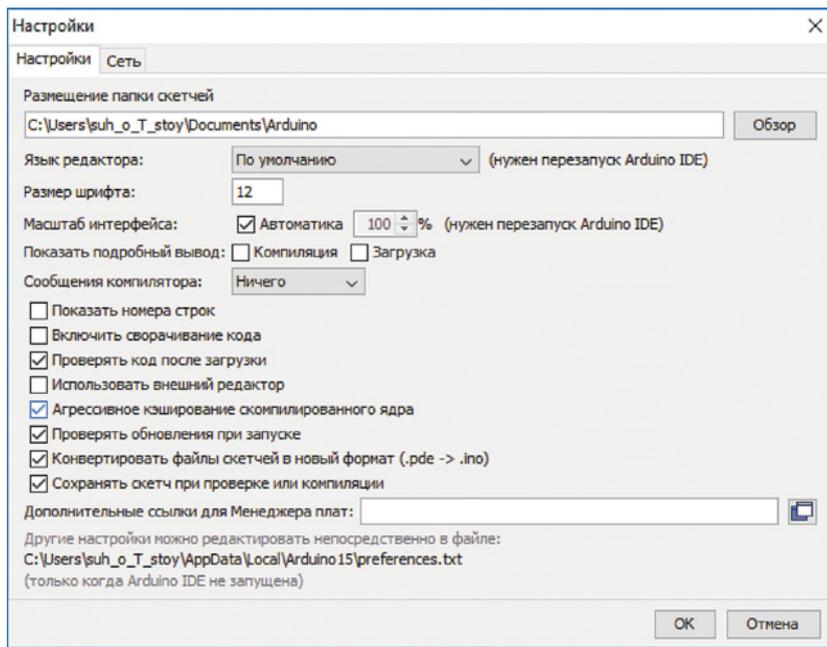


Рис. 5.4. Путь к папке скетчей Arduino

сания книги самой полной была версия от 2 июля 2014 года: <https://sourceforge.net/projects/ardublock/files/ardublock-beta-20140702.jar/download> Более новые версии имели меньше блоков.

2. Когда файл с плагином ArduBlock загрузится, переименуйте его в *ardublock-all.jar*

3. Откройте среду Arduino IDE.

4. Зайдите в меню «Файл» → «Настройки» и посмотрите путь к папке скетчей Arduino. Например, в операционной системе Windows эта папка будет располагаться в папке Documents (рис. 5.4).

5. Откройте проводник и в нем папку с документами Arduino. В этой папке создайте папку *tools*, внутри нее — папку *ArduBlockTool*, а внутри нее — папку *tool* и уже в эту папку *tool* поместите файл *ardublock-all.jar*. Таким образом, полный путь к файлу будет выглядеть так: *.../Documents/Arduino/tools/ArduBlockTool/tool/ardublock-all.jar*

6. Перезагрузите IDE. В разделе «Инструменты» появится пункт ArduBlock (рис. 5.5).

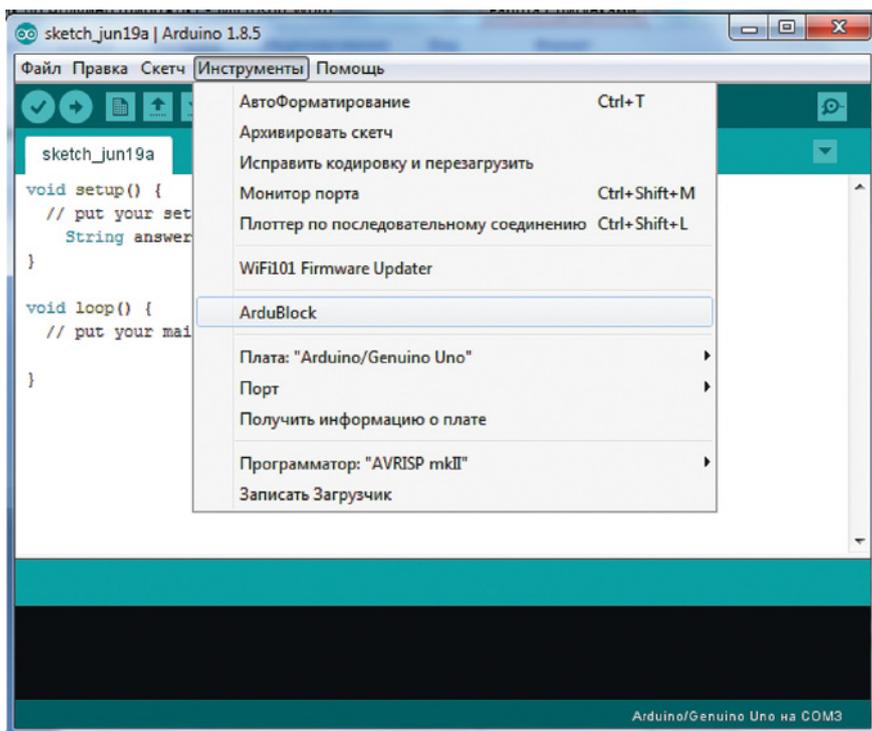


Рис. 5.5. Пункт ArduBlock в разделе «Инструменты»



Примечание

Случается, что программа не выполняет конвертацию из ArduBlock в Wiring. Причиной этого может быть нестабильная текущая версия. Например, если на данный момент версия 1.8 не конвертирует программу в текстовый код, необходимо установить более раннюю версию плагина, например 1.7.7.

Выполните несколько упражнений, которые вам уже знакомы, но с применением нового инструмента.

Подключение датчика уровня жидкости

Большинство блоков в ArduBlock проиллюстрированы картинкой (фотографиями датчиков и модулей), что упрощает процесс поиска нужной команды. В качестве примера рассмотрим программу поворота на 180° сервопривода, подключенного к порту № 5, с задержкой в 2 секунды (рис. 5.6).



Рис. 5.6. Программа поворота сервопривода

Когда вы закончите составление программы из графических блоков, сохраните файл. Нажмите кнопку **Save**. В появившемся диалоговом окне (рис. 5.7) выберите место для сохранения вашей программы в формате **.abp** (ArduBlock Program).

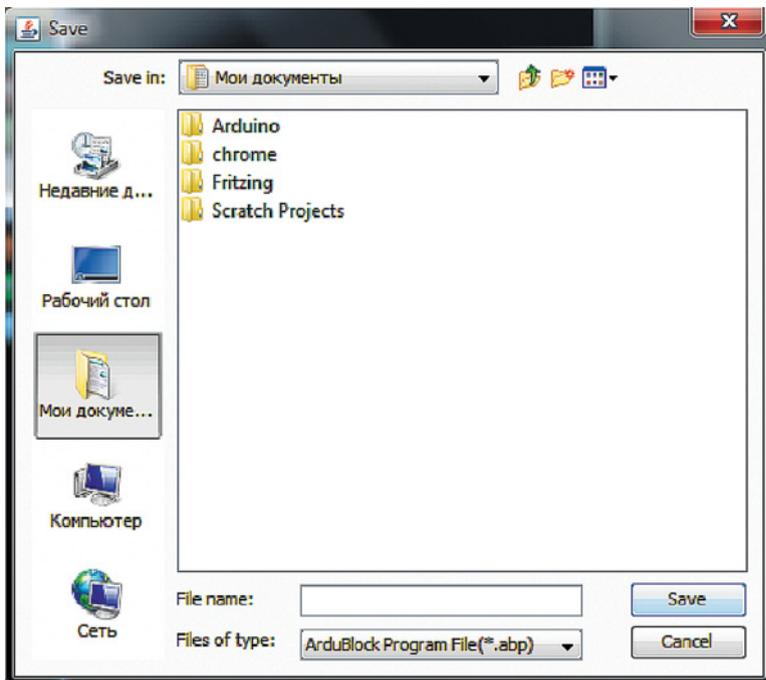


Рис. 5.7. Сохранение программы в формате .abp

Для загрузки скетча на плату нажмите на кнопку **Upload**. Ваша программа будет переведена в код на Wiring, который отобразится в готовом виде в редакторе среды Arduino IDE. В нашем случае вы получите следующий код:

```

1. #include<Servo.h> // Подключение библиотеки для работы
2. // с сервоприводом
3. Servo servo1; // Подключение библиотеки для работы
4. // с сервоприводом
5.
6. void setup() { // Функция установки
7.   servo1.attach(5); // Привязка (англ. attach) сервопривода
8. // к цифровому выходу № 5
9. // с поддержкой ШИМ
10. }
11.
12. void loop() { // Функция повтора, основной цикл
13.   servo1.write(0); // Запись (англ. write) градуса угла (0)
14. // для объекта сервопривод
15.   delay(3000); // Ожидание (delay) 3 с объектом servo1
16.   servo1.write(270); // запись значения угла в 270 градусов
17.   delay(1500); // Ожидание 1.5 с
18. }
```

Управление потенциометром

Дополните предыдущую схему, регулируя ШИМ с помощью потенциометра. Вытащите соответствующий блок из меню компонентов и добавьте в цикл (рис. 5.8).

Кликнув правой клавишей по блоку, можно вызвать контекстное меню, в котором предлагается создать комментарий или кло-



Рис. 5.8. Добавление потенциометра

нировать элемент. Добавленный комментарий отразится позднее и в коде.

```
1. #include <Servo.h>
2. Servo servo1;
3.
4. void setup()
5. {
6.   servo1.attach(5);
7. }
8. void loop()
9. {
10.  int potent=analogRead(A1); //инициация переменной с записью
11.                             //в нее значения, считываемого
12.                             //с аналогового порта A1.
13.  potent=map(potent,0,1023,0,180);
14.    //создание карты, т.е. границ изменения
15.    //входящих значений и необходимых границ на выходе
16.  servo1.write(potent); //запись значения угла поворота
17.                        //сервопривода из полученного
18.                        //в переменную potent
19.  delay(2); //пауза для ожидания поворота сервопривода
20. }
```

В данном случае работа с графическими блоками кажется сложнее, чем работа с текстом, поскольку используется мало команд, а блоки занимают большую часть экрана. Однако удобство использования графического языка легко продемонстрировать на примере следующего упражнения.

Работа с LCD-дисплеем

Компоненты:

- плата Arduino Uno, 1x;
- LCD-дисплей 1602, 1x;
- USB-кабель, 1x;
- провод с одним концом типа штекер и одним концом типа гнездо, 4x;
- макетная плата BreadBoard Half, 1x.

LCD-дисплеи размерности 16×2 (маркировка 1602) на базе контроллера HD44780 находятся в ряду самых простых, доступных и востребованных дисплеев для разработки различных электронных устройств. Их можно встретить как в устройствах, собранных на коленке для личного использования или просто из интереса,

№ контакта	Обозначение	Назначение
1	VSS	Питание контроллера (контакт земли)
2	VDD	Питание контроллера (контакт питания)
3	VO	Вывод управления контрастом
4	RS	Выбор регистра
5	RW	Чтение/запись (режим записи выбирается при соединении контакта с землей)
6	E	Enable (строб по спаду)
7–10	DB0–DB3	Младшие биты 8-битного интерфейса
11–14	DB4–DB7	Старшие биты интерфейса
15	A	Анод (+) питания подсветки
16	K	Катод (-) питания подсветки

так и в промышленно изготовленных системах автоматизации, таких как, например, автоматы для приготовления кофе.

На дисплее имеется 16-контактный (16pin) разъем для подключения (рис. 5.9). Выводы промаркированы на тыльной стороне платы. Некоторые производители меняют местами 1-й и 15-й контакты, о чем всегда свидетельствует надпись.

Прежде чем обсуждать подключение дисплея к Arduino через I²C-интерфейс, давайте поговорим о самом протоколе I²C.

I²C/ПС (*Inter-Integrated Circuit*) — этот протокол изначально создавался для связи интегральных микросхем внутри электронного устройства. Разработка принадлежит фирме Philips. В основе протокола I²C лежит использование 8-битной шины для связи блоков в управляющей электронике и системы адресации, благодаря которой можно связываться по одним и тем же проводам с несколькими устройствами. Мы просто поочередно передаем данные то одному, то другому устройству, добавляя к пакетам данных идентификатор нужного элемента.

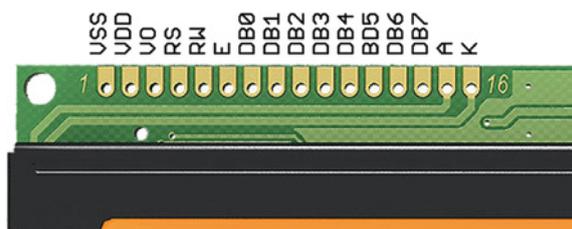


Рис. 5.9. Разъем для подключения LCD-монитора

Самая простая схема I²C может содержать одно ведущее устройство (чаще всего это микроконтроллер ATmega на плате Arduino) и несколько ведомых (например, LCD-дисплей). Каждое устройство имеет адрес в массиве адресов от 7 до 127. В одной схеме не должно быть двух устройств с одинаковым адресом.

Плата Arduino Uno поддерживает протокол I²C на аппаратном уровне. Обычно для подключения устройств по данному протоколу используются аналоговые контакты A4 и A5.

В работе протокола I²C можно выделить несколько преимуществ:

- для работы требуется всего две линии — SDA (линия данных) и SCL (линия синхронизации);
- возможность подключения большого количества ведомых устройств;
- сокращение времени разработки;
- набор устройств управляется всего одним микроконтроллером;
- возможное число микросхем, подключаемых к одной шине, ограничивается только нагрузочной способностью Arduino Uno;
- высокая степень сохранности данных обеспечивается подавляющим всплески фильтром, встроенным в схемы интерфейса;
- простая процедура диагностики возникающих сбоев, быстрая отладка неисправностей;
- шина уже интегрирована в плату Arduino Uno, поэтому не нужно разрабатывать дополнительно шинный интерфейс.

Несмотря на большое количество достоинств, протокол имеет и недостатки:

- емкостное ограничение на линии — 400 пФ;
- трудное программирование контроллера I²C, если к шине подключено несколько различных устройств;
- при большом количестве устройств возникают трудности локализации сбоя, если одно из устройств ошибочно устанавливает состояние низкого уровня.

Самый быстрый и удобный способ использования LCD-дисплея с Arduino — это покупка готового экрана со встроенной поддержкой протокола I²C. Но таких экранов не очень много и стоят они недешево. В то же время стандартных экранов на рынке предлагается множество. Поэтому самым доступным и популярным на сегодня вариантом является покупка и установка отдельного I²C-модуля — переходника (интерфейса), который изображен на рис. 5.10.

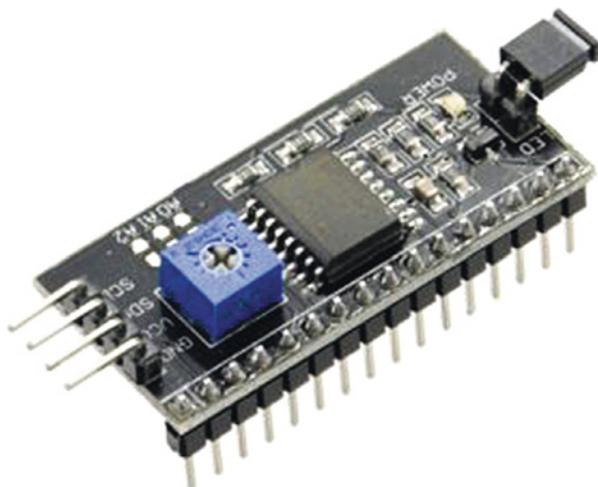


Рис. 5.10. Модуль I²C для LCD 1602 Arduino



Рис. 5.11. Подключенный модуль I²C

На одной стороне модуля расположены выводы I²C: земля, питание и SDA, SCL для передачи данных, а на другой — разъемы внешнего питания. И конечно, на плате есть множество ножек, с помощью которых модуль припаивается к стандартным выводам экрана (рис. 5.11). Если вы работаете с платой для прототипирования без пайки, то модуль придется подключать дополнительными проводами.

Для подключения к плате Arduino используются I²C-выходы. Если нужно, подключается внешнее питание для подсветки. С помощью встроенного подстроечного резистора можно подрегулировать настраиваемые значения контрастности экрана (J).

На рынке можно встретить LCD-модули 1602 с уже припаянными переходниками — этот вариант самый простой! Если же вы купили отдельный переходник, припаяйте или подключите его к модулю.

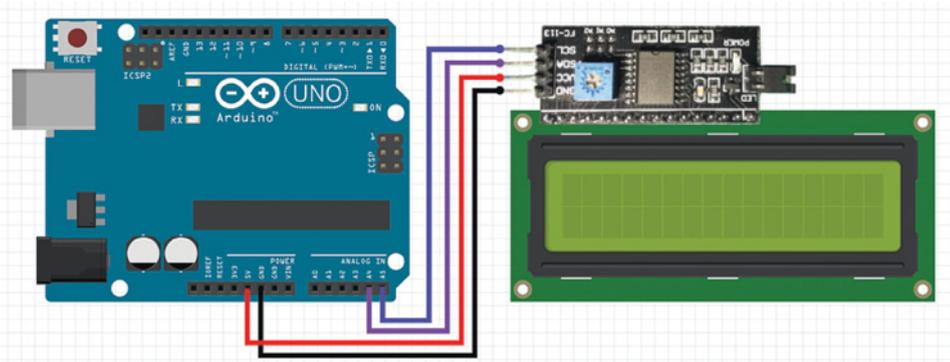


Рис. 5.12. Подключение LCD с поддержкой I²C к плате Arduino Uno



Рис. 5.13. Блоки для программирования LCD-дисплея

После подключения можно приступать к программированию LCD-дисплея через ArduBlock.

Жидкокристаллический монитор с поддержкой I²C подключается к плате Arduino Uno четырьмя проводами: два провода для данных и два для питания (рис. 5.12):

- 1) вывод GND подключается к GND на ведущей плате;
- 2) вывод VCC подключается к контакту 5V;
- 3) SCL подключается к контакту A5 аналоговой группы;
- 4) SDA подключается к контакту A4 аналоговой группы.

Откройте среду программирования ArduBlock и соберите блоки, изображенные на рис. 5.13. Команда **print** позволяет писать сообщение, т. е. печатать его на экране. **Line** — номер строки дисплея, начиная с 0, **Char** — номер символа в строке. В графе **address** с недавнего времени вместо «стандартного» адреса 0x27 стали указывать 0x3F (чаще всего) или 0x5C.

Сохраните скетч и нажмите на **Загрузить в Arduino**. Итоговый код программы в IDE:

```
1. #include <Wire.h>
2. #include <LCD.h>
3. #include <LiquidCrystal_I2C.h>
4.
5. LiquidCrystal_I2C lcd_I2C_3F(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
6. /* Здесь прописывается, что был подключен объект типа
7. дисплей на конкретные адреса и с настройками. Это требуется,
8. так как все данные физически идут по одной шине. */
9. void setup()
10. {
11.   lcd_I2C_3F.begin (16, 2); //Объявление размера экрана
12.   lcd_I2C_3F.setBacklight(HIGH); //Включение подсветки
13.   lcd_I2C_3F.setCursor( (1) - 1, (1) - 1); //Установка каретки
14.                                     //(курсора)
15.   lcd_I2C_3F.print( "Hello" ); //Вывод сообщения
16. }
17.
18. void loop()
19. {
20. }
```

Для взаимодействия Arduino с LCD 1602 по шине I²C вам потребуются как минимум две библиотеки:

- 1) библиотека Wire.h для работы с I²C уже имеется в стандартной программе Arduino IDE. Wire — это интерфейс для «общения» по одной шине;
- 2) библиотека LiquidCrystal_I2C.h, которая включает в себя большое разнообразие команд для управления LCD-дисплеем по шине I²C и позволяет сделать скетч проще и короче.

После подключения дисплея нужно установить еще библиотеку LiquidCrystal_I2C.h. Для этого скачайте файл по ссылке в формате архива: https://bitbucket.org/fmalpartida/new-liquid-crystal/downloads/LiquidCrystal_V1.2.1.zip

Затем в среде Arduino IDE импортируйте эту библиотеку: **Скетч** → **Подключить библиотеку** → **Добавить библиотеку .ZIP** — и выберите скачанный архив.

5.6. Практические задания по Wiring

Теперь, когда вы привыкли к чтению текстового программного кода, перейдем к его написанию.

Фоторезистор

Компоненты:

- плата Arduino Uno, 1x;
- фоторезистор, 1x;
- резистор, 100 кОм, 1x;
- USB-кабель, 1x;
- провод с двумя концами типа штекер, 5x;
- макетная плата BreadBoard Half, 1x.

Подключите фоторезистор, предварительно подготовив макетную плату (подключив питание и землю). Вставьте выводы фоторезистора в отверстия разных рядов макетной платы. Добавьте рядом резистор на 100 кОм, как показано на рис. 5.14.

Соедините общую точку фоторезистора и резистора на 100 кОм с аналоговым входом А0 платы Arduino Uno.

Фоторезистор будет изменять сопротивление электрической цепи в зависимости от уровня освещенности помещения: чем больше света в помещении, тем ниже уровень аналогового сигнала.

Проверьте результат с помощью скетча:

1. void setup () {
2. Serial.begin(9600);
3. }
4. }
5. void loop () {
6. Serial.print ("Light:");
7. Serial.println(analogRead(A0));
8. }

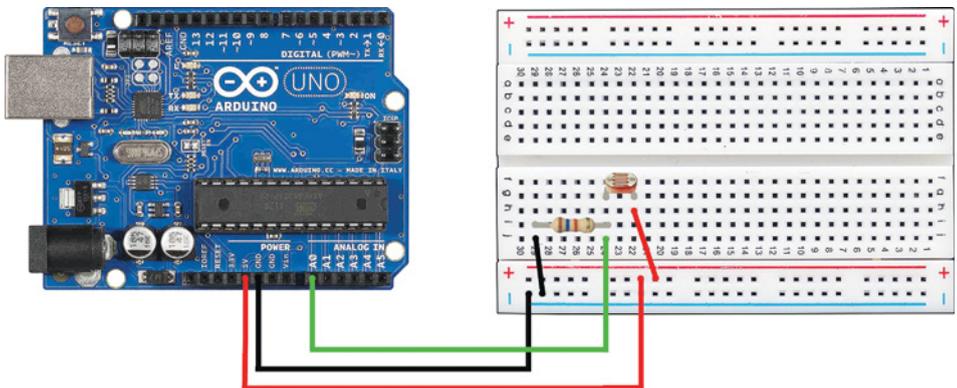


Рис. 5.14. Подключение фоторезистора

Дальномер

Компоненты:

- плата Arduino Uno, 1x;
- ультразвуковой дальномер HC-SR04, 1x;
- USB-кабель, 1x;
- красный провод с одним концом типа штекер и одним концом типа гнездо, 2x;
- зеленый провод с одним концом типа штекер и одним концом типа гнездо, 2x;
- провод с двумя концами типа штекер, 2x.

Подключите к Arduino Uno сложный цифровой датчик — ультразвуковой дальномер HC-SR04 (рис. 5.15). Соедините контакт VCC (питание) на дальномере и контакт 5V на плате Arduino. Красным проводом (если нет черного провода) соедините контакт GND дальномера с контактом земли на основной плате (см. рис. 5.15).

Сонар в дальномере HC-SR04 имеет обозначение **Trig**, а микрофон — **Echo**. Зеленым проводом соедините контакт **Trig** дальномера и цифровой порт №9 платы Arduino. Вторым зеленым проводом подключите контакт **Echo** дальномера к цифровому порту №8. Всё, дальномер подключен (рис. 5.16).

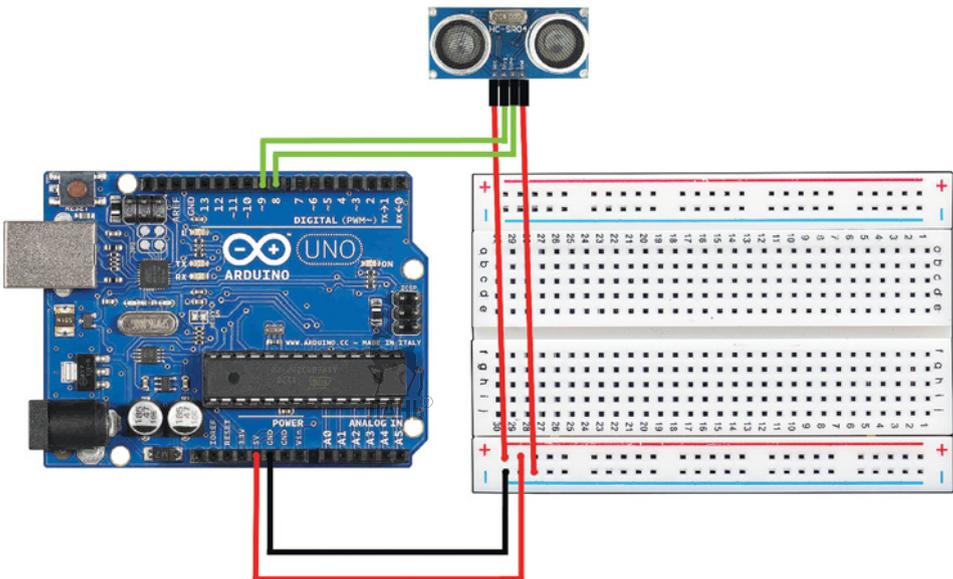


Рис. 5.15. Схема подключения дальномера

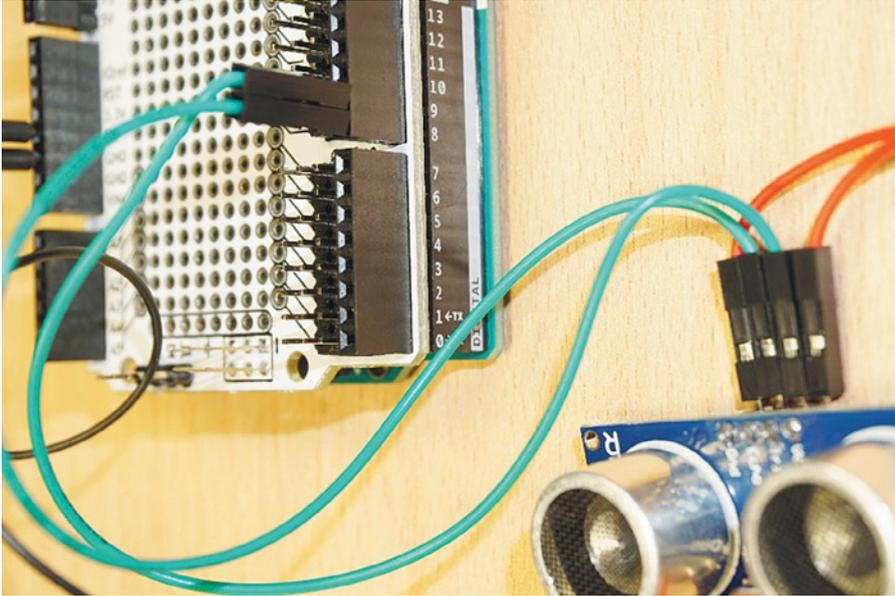


Рис. 5.16. Дальномер подключен

Скетч для измерений будет выглядеть так:

```

1. void setup () {
2.     Serial.begin (9600);
3.     pinMode(9, OUTPUT); //Trig
4.     pinMode(8, INPUT); //Echo
5. }
6.
7. void loop () {
8.     digitalWrite(9, LOW); //очистка сигнала во избежание ошибок
9.     delayMicroseconds(2); //пауза в микросекундах
10.    digitalWrite(9, HIGH); //начало отправки ультразвукового сигнала
11.    delayMicroseconds(10); //длительность сигнала составляет 10 мкс
12.    digitalWrite(9, LOW); //выключение сонара
13.    int duration = pulseIn(8, HIGH); //подсчет секунд до
                                     //прихода отраженного сигнала
14.    float cm = duration / 58;
15.    /* 58 — это константа, обусловленная физическим строением
16.    дальномера данной модели. На самом деле необходимо
17.    делить на 29 два раза (прямой и обратный путь). */
18.    Serial.println(cm);
19.    delay(1000);
20. }

```

Шаговый двигатель

Шаговый двигатель нельзя подключить непосредственно к выводам Arduino — на выходах микроконтроллера недостаточно нагрузочной способности по току и напряжению. По этой причине необходим усилитель управляющих сигналов — драйвер мотора. Обычно это отдельная плата.

Компоненты:

- плата Arduino Uno, 1x;
- шаговый двигатель 28BYJ-48 (5V), 1x;
- драйвер мотора ULN2003, 1x;
- USB-кабель, 1x;
- провод с одним концом типа штекер и одним концом типа гнездо, 2x;
- тройной шлейф для драйвера мотора, 1x.

Мы будем запитывать драйвер мотора от платы Arduino (через преобразователь). Однако он позволяет использовать и внешнее питание, причем более мощное, например 9 В. На рис. 5.17 изображен общий вид соединения Arduino Uno, шагового двигателя и драйвера мотора. Порядок соединения расписан в табл. 5.1.

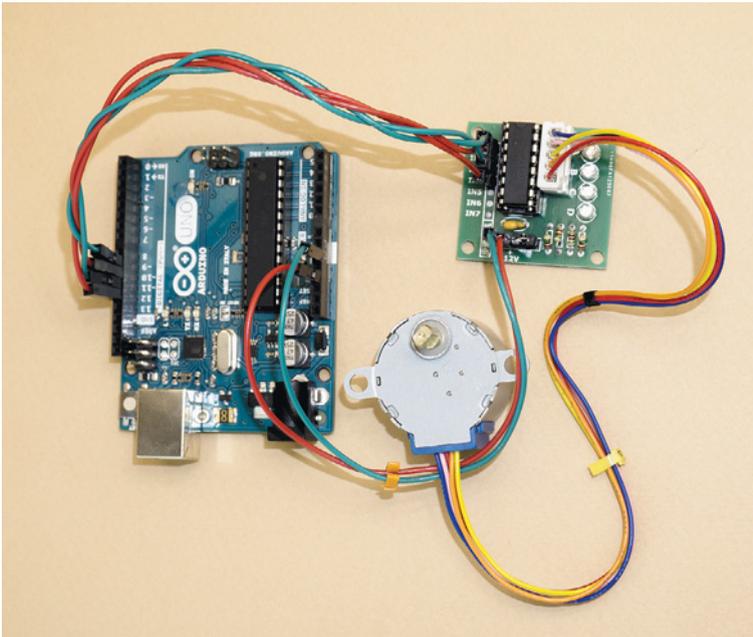


Рис. 5.17. Соединение Arduino Uno, шагового двигателя и драйвера мотора

Таблица 5.1

Драйвер мотора	Arduino Uno
IN1	D8
IN2	D9
IN3	D10
IN4	D11

Перепечатайте представленный ниже скетч и запустите его. Поэкспериментируйте с длительностями *delay()*.

```
1. int in1 = 8;
2. int in2 = 9;
3. int in3 = 10;
4. int in4 = 11;
5.
6. void setup() {
7. //Задание ролей портов:
8. pinMode(in1, OUTPUT);
9. pinMode(in2, OUTPUT);
10. pinMode(in3, OUTPUT);
11. pinMode(in4, OUTPUT);
12.
13. }
14.
15. void loop() {
16. digitalWrite(in1,HIGH);
17. digitalWrite(in2,HIGH);
18. digitalWrite(in3,LOW);
19. digitalWrite(in4,LOW);
20. delay(5);
21. digitalWrite(in1,LOW);
22. digitalWrite(in2,HIGH);
23. digitalWrite(in3,LOW);
24. digitalWrite(in4,LOW);
25. delay(5);
26.
27. digitalWrite(in1,LOW);
28. digitalWrite(in2,HIGH);
29. digitalWrite(in3,HIGH);
30. digitalWrite(in4,LOW);
31. delay(5);
32. digitalWrite(in1,LOW);
33. digitalWrite(in2,LOW);
34. digitalWrite(in3,HIGH);
35. digitalWrite(in4,LOW);
36. delay(5);
37.
```

```
38.    digitalWrite(in1,LOW);
39.    digitalWrite(in2,LOW);
40.    digitalWrite(in3,HIGH);
41.    digitalWrite(in4,HIGH);
42.    delay(5);
43.
44.    digitalWrite(in1,LOW);
45.    digitalWrite(in2,LOW);
46.    digitalWrite(in3,LOW);
47.    digitalWrite(in4,HIGH);
48.    delay(5);
49.
50.    digitalWrite(in1,HIGH);
51.    digitalWrite(in2,LOW);
52.    digitalWrite(in3,LOW);
53.    digitalWrite(in4,HIGH);
54.    delay(5);
55.
56.    digitalWrite(in1,HIGH);
57.    digitalWrite(in2,LOW);
58.    digitalWrite(in3,LOW);
59.    digitalWrite(in4,LOW);
60.    delay(5);
61. }
```

Как видите, код получился слишком громоздкий, но с помощью специальной библиотеки его можно сильно сократить и упростить. Ниже приведен тот же самый фрагмент программы, но написанный в IDE с использованием библиотеки Stepper для шаговых двигателей:

```
1. #include <Stepper.h> //Подключение библиотеки для двигателя
2.
3. Stepper myStepper(2048,8,9,10,11); //Инициализация объекта
4. //типа шаговый двигатель с 2048 шагами, подключенного
5. //к контактам 8-11
6.
7. void setup() {
8.     myStepper.setSpeed(10); //Установка скорости
9. }
10.
11. void loop() {
12.     myStepper.step(2048); //Сделать полный оборот вперед
13.     delay(1000);
14.     myStepper.step(-2048); //Сделать полный оборот назад
15.     delay(1000);
16. }
```

Датчик температуры и влажности DHT11

DHT11 — это цифровой датчик, состоящий из термистора (резистора, сопротивление которого изменяется под влиянием изменений температуры) и емкостного датчика влажности. Он недорогой и имеет следующие характеристики: питание 3,5–5 В; точность измерения температуры 2% в рабочем диапазоне 0–50 °С, диапазон измерения влажности 20–95% с точностью 5%.

Компоненты:

- плата Arduino Uno, 1x;
- датчик температуры и влажности DHT11, 1x;
- USB-кабель, 1x;
- провод с одним концом типа штекер и одним концом типа гнездо, 3x;
- провод с концами типа штекер, 3x.

Подключение датчика DHT11 представлено на рис. 5.18.

Для работы с датчиком необходимо скачать и установить специальную библиотеку (<https://istarik.ru/file/stDHT.zip>).

Скетч будет выглядеть следующим образом:

1. `#include <stDHT.h>`
- 2.
3. `DHT sens(DHT11); //Указать датчик DHT11, DHT21 или DHT22`
4. `//(несколько датчиков вписывать не нужно).`

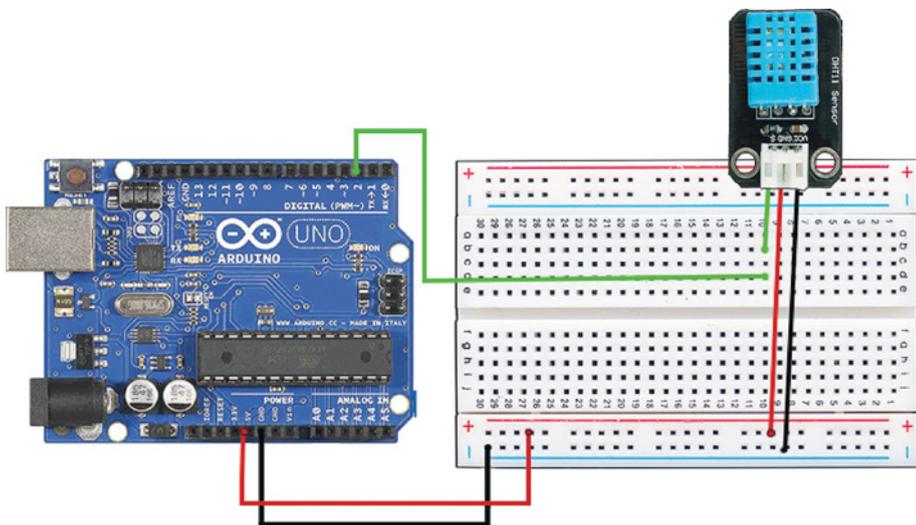


Рис. 5.18. Подключение датчика DHT11

```
5. //Подключать можно только одинаковые датчики, т.е. нельзя
6. //использовать одновременно DHT11 и DHT22!
7.
8. void setup()
9. {
10.     Serial.begin(57600); //Открытие порта на скорости 57 600 бод.
11.     pinMode(2, INPUT);
12. }
13.
14. void loop()
15. {
16.     int t = sens.readTemperature(2); //Считывание в переменную t
17.     //показаний температуры с датчика,
18.     //названного sens.
19.     int h = sens.readHumidity(2);    //Считывание в переменную h
20.     //показаний влажности.
21.     delay(2000);
22.
23.     //Вывод результатов в последовательный порт:
24.     Serial.print("Hum: ");
25.     Serial.print(h);
26.     Serial.print(" %");
27.     Serial.print("Temp: ");
28.     Serial.print(t);
29.     Serial.println(" C ");
30. }
```

Выполните компиляцию и загрузку скетча на устройство. Затем откройте монитор последовательного порта. В нем должны отобразиться данные в режиме реального времени.

5.7. Дополнительные задания для самостоятельной работы

Шар с предсказаниями

Это задание идет после упражнения «Работа с LCD-дисплеем».

Компоненты:

- плата Arduino Uno, 1x;
- датчик вибрации SW-520D, 1x;
- USB-кабель, 1x;
- светодиод красный, 1x;
- светодиод зеленый, 1x;

- провода;
- макетная плата BreadBoard Half, 1x.

Запрограммировать «шар» так, чтобы он случайным образом «отвечал» на поставленный вопрос «Да» или «Нет». Работа шара очень проста: если он находится в состоянии покоя, то ничего не происходит, но стоит постучать по датчику, как загорается либо красный, либо зеленый светодиод.

Усложнение задачи 1

Шар должен выдавать сообщение на экран в виде текста на LCD-дисплее.

Автоповорот

Запрограммировать картинку на экране таким образом, чтобы она поворачивалась в соответствии с положением датчика-акселерометра (рис. 5.19).

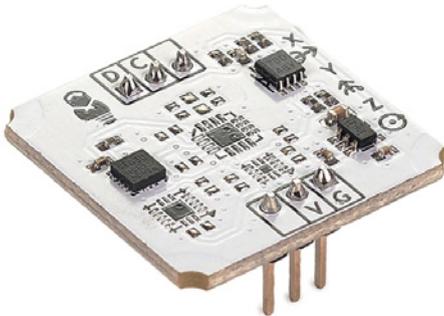


Рис. 5.19. Акселерометр

Акселерометр позволяет определять ускорение, действующее в направлении осей x , y , z , и применяется для определения ориентации объекта в пространстве: углов крена и тангажа. Кстати, впоследствии эти знания можно использовать при создании дронов.

Компоненты:

- плата Arduino Uno, 1x;
- плата расширения Тройка Shields, 1x;
- акселерометр (Тройка-модуль), 1x;
- USB-кабель, 1x;
- провода;
- макетная плата BreadBoard Half, 1x.

Акселерометр общается с управляющей электроникой по протоколу I²C / TWI. Подключение выполняется двумя трехпровод-

ными шлейфами, идущими в комплекте с модулем. При подключении модуля к Arduino удобно использовать модуль *Troyka Shield*, на котором уже выведены дополнительные контакты.

Для получения данных с акселерометра требуется библиотека *Troyka-IMU*. Она скрывает в себе все тонкости протокола, посредством которого передаются данные с акселерометра, и предоставляет простые и понятные функции для вывода значений.

Реклама «Бегущая строка»

Запрограммировать на цифровом табло бегущий текст.

Компоненты:

- плата Arduino Uno, 1x;
- светодиодный дисплей, 8x8, 1x;
- USB-кабель, 1x;
- провода;
- макетная плата BreadBoard Half, 1x.

Подключить дисплей можно, руководствуясь схемой на рис. 5.20 и табл. 5.2 соответствия контактов.

Для работы необходимо установить библиотеку *FrequencyTimer*.

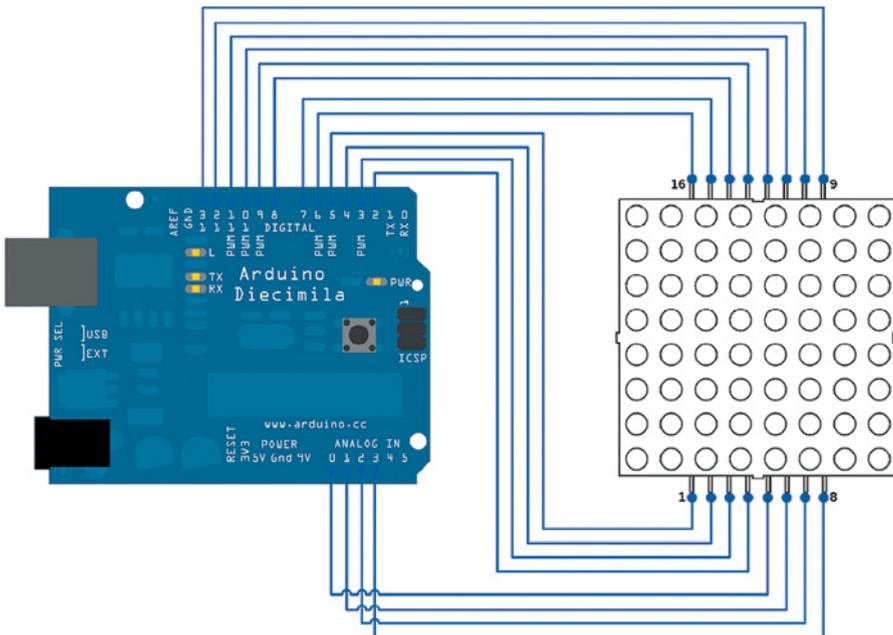


Рис. 5.20. Подключение светодиодного дисплея

Таблица 5.2

Контакты матрицы	Контакты платы Arduino Uno
Pin 1	A1
Pin 3	D5
Pin 4	D11
Pin 6	A2
Pin 7	D7
Pin 9	D4
Pin 10	D6
Pin 12	A0
Pin 13	D10
Pin 15	D12
Pin 16	A3
Pin 18	D2
Pin 19	D8
Pin 21	D3
Pin 22	D13
Pin 24	D9

5.8. Проект «Развитие моторики»



Рис. 5.21. Ваш проект может выглядеть так

В этом разделе дан пример робототехнического проекта (рис. 5.21). При его выполнении вам понадобятся все ваши знания, полученные в предыдущих главах книги, в том числе о работе датчиков и программировании. Если хотите, используйте его в качестве возможного шаблона для оформления собственных новых проектов.

Цель: создание системы развития мелкой моторики у детей дошкольного возраста.

Маленькие дети только учатся управлять собственным телом. То, что взрослым кажется простым и не требующим специального внимания, для них может представлять определенные трудности. Нейронные связи в организме ребенка еще только формируются, поэтому нужна тренировка, чтобы научиться нажимать на предметы пальцами с разной силой и в разных комбинациях.

Задачи:

1. Создание тренажера для развития мелкой моторики у детей дошкольного возраста.
2. Формулирование и описание игры:
 - 1) методики (правил игры, участия и описания);
 - 2) программы для тренажера, соответствующие требованиям пункта 1).

Ключевые слова: дети, моторика, здоровье, игры, развитие.

Области применения:

- *дома:* для развития мелкой моторики у младших членов семьи, для развития собственных навыков, а также для тренировки пианистов;
- *в образовательных учреждениях:* для развития моторики у детей в группах детского сада.

Компоненты и инструменты:

- компьютер (минимальные требования): ОС Windows XP, Vista, Windows 7, Windows 8, Windows 8.1, Windows 10 (32/64) / Linux Mint, Ubuntu, Fedora / Mac OS X; оперативная память не менее 512 Мб; процессор — 1,1 ГГц (или быстрее); свободное пространство на диске — 200 Мб;
- среда программирования Arduino IDE;
- плата Arduino Uno;
- соединительные провода с двумя штырьковыми контактами, 8х;
- соединительные провода с концами типа штырек и гнездо, 6х;
- плата расширения Wireless Proto Shield;
- модуль беспроводной связи Bluetooth Bee (или XBee);
- макетная плата BreadBoard Half;
- резистор давления диаметром 12 мм, 3х;
- резистор, 10 кОм, 3х;
- NiMH-аккумулятор типа «Крона» и кабель питания для батарейки «Крона» или импульсный блок питания с USB-разъемом (рекомендуется);
- USB-кабель типа А-В;
- изоляционная лента или скотч;
- ножницы.

Для корпуса:

- готовый набор для сборки скворечника или деревянная шкатулка для декупажа, в которую сможет поместиться сэндвич из Arduino Uno и Wireless Proto Shield и отдельная макетная плата; у нас нашлась кормушка для птиц (см. рис. 5.21);
- дрель;
- наждак или наждачная бумага для обработки краев.

Для своего проекта вы можете взять модули других производителей, имеющие аналогичные характеристики, поскольку Arduino — открытая аппаратная платформа.

Исследование

В проектах всегда все начинается с выявления проблемы, постановки вопроса и поиска уже существующих решений, которые потом потребуются или адаптировать, или отвергнуть как неоптимальные. Чтобы на это решиться, нужно провести тщательный анализ.

Итак, этап исследования включает в себя:

- краткое изучение направления работы;
- исследование существующих разработок по заданной тематике;
- анализ материалов и выявление принципов, применяемых в этих разработках;
- критическую оценку существующих решений;
- формулирование гипотезы о собственном оптимальном решении.

Как видите, план большой, однако подобное исследование может занять в формате тезисов не более 5–10 страниц.

Например, для данного проекта потребуются:

- выписать определения (моторика, развитие и т. д.);
- изучить, какие методики применяются, сузив область до методик, включающих применение игр и игрушек;
- выяснить, какие игрушки наиболее популярны и какие рекомендуют психологи; из чего эти игрушки сделаны (пластик, дерево, металл), они статичны или динамичны;
- выписать, почему электронные игрушки из дешевых материалов могут быть не только вредны, но и бесполезны из-за низкой чувствительности датчиков и несоответствия заявленным параметрам;
- доказать, что прямой контакт с хорошо закрепленными датчиками даст лучший результат, и ввести ограничения и требования к будущему продукту.

Сборка

1. *Установка платы расширения Wireless Proto Shield на плату Arduino Uno (рис. 5.22).*

2. *Установка модуля беспроводной связи Bluetooth Bee (рис. 5.23) на специальную группу контактов на плате Wireless Proto Shield.*

3. *Подключение питания к макетной плате.* Соедините красным проводом любой контакт на шине питания «+» (рядом



Рис. 5.22. Сэндвич из платы расширения Wireless Proto Shield и платы Arduino Uno

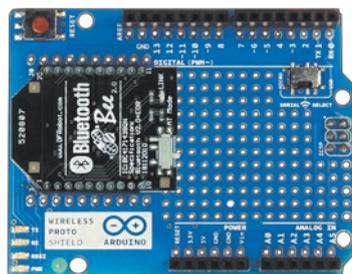


Рис. 5.23. Модуль беспроводной связи Bluetooth Bee

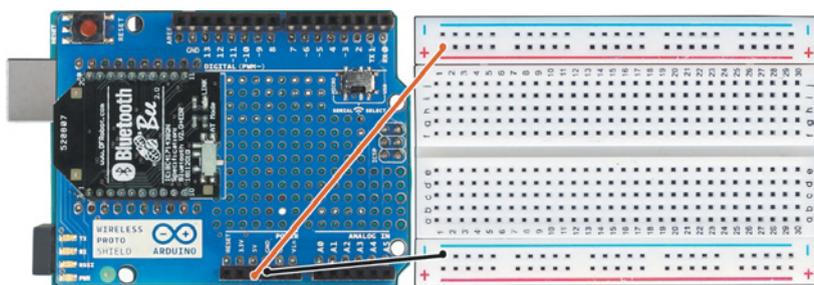


Рис. 5.24. Поддача питания на макетную плату

с красной линией) на макетной плате с контактом 5V на плате Wireless Proto Shield. Учитывая будущие подключения, удобнее задействовать шину, расположенную наверху макетной платы (рис. 5.24).

4. Подключение земли к макетной плате. Соедините черным проводом любой контакт на шине земли «-» (рядом с синей линией) на макетной плате с контактом GND на плате Wireless Proto Shield. Теперь удобнее использовать шину, расположенную внизу макетной платы (см. рис. 5.24).

5. Следующим шагом будет сборка простенькой электрической схемы тренажера. Она будет состоять из трех делителей напряжения, построенных совершенно одинаково: из резистора давления R_d и обычного резистора R на 10 кОм, соединенных последовательно (рис. 5.25). Выходное напряже-

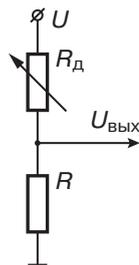


Рис. 5.25. Электрическая схема одного делителя напряжения

ние снимается с их общей точки и подается в виде аналогового сигнала на управляющую электронику, например Arduino.

Это интересно!

Резистор давления изменяет свое сопротивление в зависимости от силы, приложенной к его диску: чем больше сила давления, тем меньше сопротивление R_d . При отсутствии давления сопротивление резистора давления превышает 1МОм. Резистор давления может измерять вес в диапазоне от 100 г до 10 кг, но с невысокой точностью, поэтому для сборки электронных весов он не годится.

Сила тока в такой схеме рассчитывается согласно закону Ома:

$$I = \frac{U}{R_d + R},$$

где I — сила тока; U — напряжение питания; R_d — сопротивление резистора давления; R — сопротивление второго резистора.

Напряжение $U_{\text{ВЫХ}}$, которое будет поступать на плату Arduino Uno с резистора R , можно вычислить по формуле:

$$U_{\text{ВЫХ}} = IR = \frac{U}{R_d + R} \cdot R$$

или

$$U_{\text{ВЫХ}} = U \cdot \frac{R}{R_d + R}.$$

Кстати!

Этот пункт тоже следует описать в документации проекта, как и все применяемые формулы.

Итак, приступим к сборке на макетной плате первого делителя напряжения. Присоедините любой из выводов резистора давления к шине питания на макетной плате (рис. 5.26). Другой вывод подключите к контакту j1 (см. рис. 5.26).

Затем вставьте выводы резистора R (на 10 кОм) в контакты f1 и e1. После этого соедините черным проводом контакт d1 и шину земли (рис. 5.27).

Осталось подключить выход делителя напряжения к плате Arduino. Соедините проводом контакт g1 на макетной плате и аналоговый вход A0 (Analog №0) на плате Wireless Proto Shield (рис. 5.28).

Аналогично соберите еще два делителя напряжения (рис. 5.29).

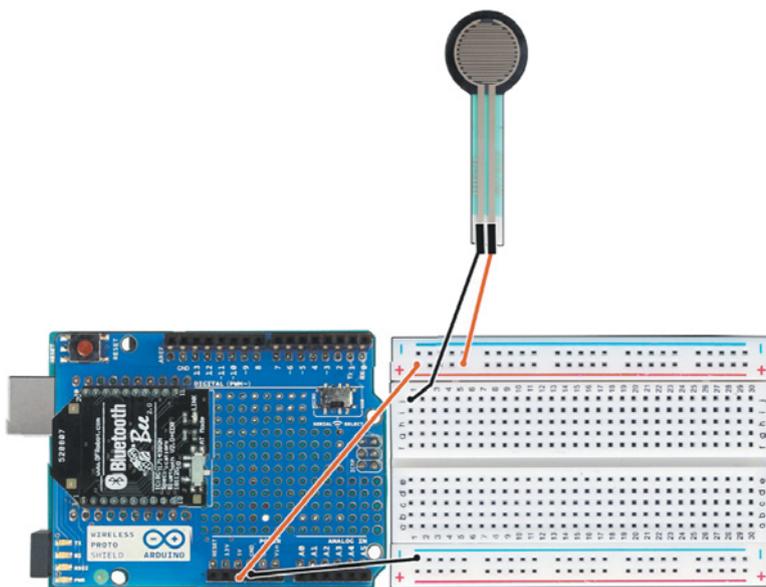


Рис. 5.26. Подключение первого резистора давления

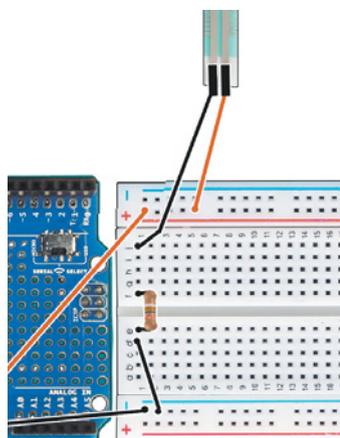


Рис. 5.27. Подключение резистора R

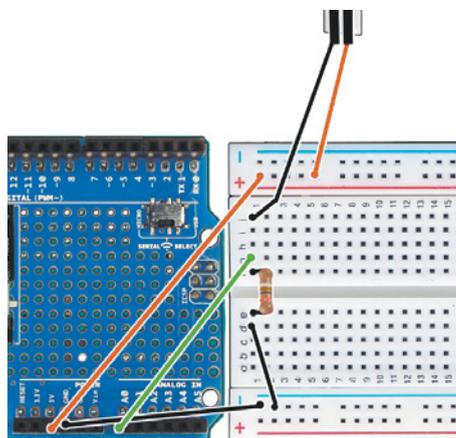


Рис. 5.28. Подключение выхода делителя напряжения к плате Arduino

В таблице ниже расписан порядок подключения компонентов тренажера:

	Делители напряжения		
	Первый	Второй	Третий
R_D	j1 и шина «+»	j11 и шина «+»	j21 и шина «+»
R	f1 и e1	f11 и e11	f21 и e21
Выход	g1 и A0	g11 и A1	g21 и A2

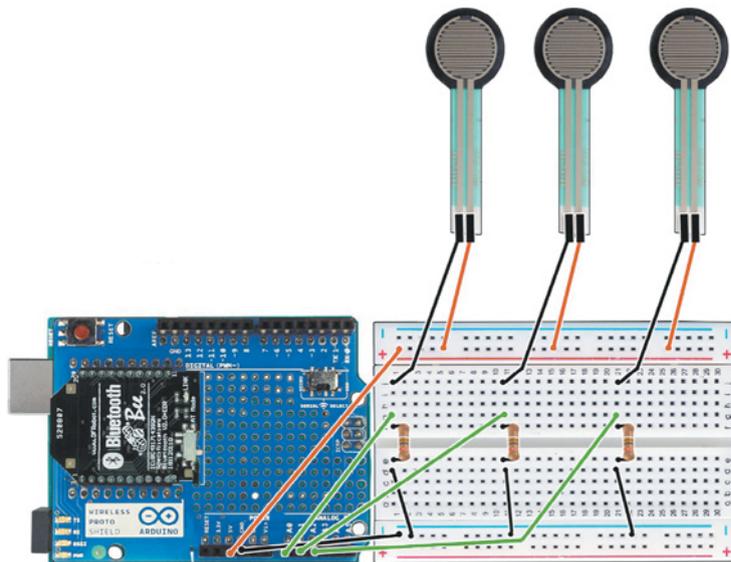


Рис. 5.29. Подключение второго и третьего делителей напряжения

6. Дизайн корпуса. Первым делом нужно обеспечить вентиляцию — устройство не должно перегреваться! Для этого проделайте в стенках корпуса отверстия для резисторов давления. Сбоку нужно еще одно отверстие, чтобы завести внутрь кабель питания (рис. 5.30). Обработайте края наждачной бумагой или надфилем, чтобы пользователь не получил занозы при игре.

Аккуратно поместите сэндвич из плат Arduino и Wireless Pro-to Shield и макетную плату в корпус, выведя наружу три резистора давления (рис. 5.31). Зафиксируйте их изолентой так, чтобы ребенку было удобно нажимать.

Для питания устройства подсоедините к тренажеру проводом батарейку типа «Крона» или воспользуйтесь блоком питания



Рис. 5.30. Домик для тренажера



Рис. 5.31. Теперь всё на месте!

и USB-кабелем. Батарейка или аккумулятор типа «Крона» обеспечивают автономную, хотя и недолгую, работу проекта. Можно взять аккумулятор типа PowerBank.

Добавьте элементы декора, например посадите на жердочку плюшевую птичку. Спрячьте изоленту и замаскируйте видимую часть проводов датчиков.

Проверьте, что соединения не нарушены и ничто не касается плат. Если все в порядке, то сборка завершена и можно подключать тренажер к компьютеру для программирования.

Программирование

Для резистора давления нет специфических библиотек — обычно применяется обработка значения сигнала, поступающего на аналоговый вход платы Arduino. Для использованной в проекте схемы сборки верны следующие значения:

Сопротивление резистора давления, R_d	Сила нажатия, Н	Выходное напряжение, В	Значение функции <code>analogRead()</code>
30 кОм	0,2	1,3	266
6 кОм	1	3,1	634
1 кОм	10	4,5	921
250 Ом	100	4,9	1003

Применение модуля беспроводной связи тоже не требует дополнительных библиотек, поскольку он на физическом уровне реализует другой канал для протокола последовательного порта UART (УАПП, универсальный асинхронный приемопередатчик), используемого при USB-подключении.

1. /*1 ЭТАП. ПОДГОТОВКА ПАМЯТИ*/
2. /*ПЕРЕМЕННЫЕ*/
3. `int firstSensor = 0;` //Для хранения значения давления
4. //на первый датчик.
5. `int secondSensor = 0;` //Для хранения значения давления
6. //на второй датчик.
7. `int thirdSensor = 0;` //Для хранения значения давления
8. //на третий датчик.
9. `String textline;` //Название силы нажатия.
- 10.
11. /*Для удобства следует переобозначить аналоговые входы,
12. назвав их по датчикам.*/
13. `#define fsrOne A0` //Первый резистор давления.
14. `#define fsrTwo A1` //Второй резистор давления.
15. `#define fsrThree A2` //Третий резистор давления.
- 16.

```
17.
18. /*II ЧАСТЬ. ЗАПУСК УСТРОЙСТВА (ФУНКЦИЯ УСТАНОВКИ)*/
19. void setup() {
20.   Serial.begin (9600); //UART. В данном случае выводится
21.                       //на Bluetooth, переключается физически.
22. }
23.
24. /*III. ОСНОВНАЯ ПРОГРАММА.*/
25.
26. /*ДОПОЛНИТЕЛЬНАЯ ФУНКЦИЯ РАНЖИРОВАНИЯ ДАВЛЕНИЯ*/
27. void pressure (int value) { //Будет возвращать строку, а на вход
28.                             //получает значение от датчика
29. /* После слов необходимо добавить пробелы, чтобы по ним
30. разделять данные в приложении на Android */
31.   if(value<250) textline = "слабо ";
32.   else if(value>=250 & value<500) textline = "нормально ";
33.   else if(value>=500) textline = "сильно ";
34.   Serial.print(textline);
35. }
36.
37. /*ДОПОЛНИТЕЛЬНАЯ ФУНКЦИЯ ОПРОСА ДАТЧИКОВ*/
38. void info(){
39.   firstSensor = analogRead(fsrOne); //Считывание показаний
40.                                     //с первого датчика.
41.   secondSensor = analogRead(fsrTwo); //Считывание показаний
42.                                       //со второго датчика.
43.   thirdSensor = analogRead(fsrThree); //Считывание показаний
44.                                       //с третьего датчика.
45. //Вывод ранжированных значений с трех датчиков
46. //в одну строку.
47.   pressure(firstSensor);
48.   pressure(secondSensor);
49.   pressure(thirdSensor);
50.   Serial.println(); //Перевод каретки на новую строку.
51. }
52. /*2. ОСНОВНАЯ ФУНКЦИЯ ПОВТОРА*/
53. void loop() {
54.   info();
55.   delay(5000); //Пауза 5 секунд.
56. }
```

При прошивке через USB-кабель убедитесь, что рычажок «SERIAL SELECT» на плате Wireless Proto Shield находится в положении USB. После прошивки переведите рычажок в положение «MICRO».

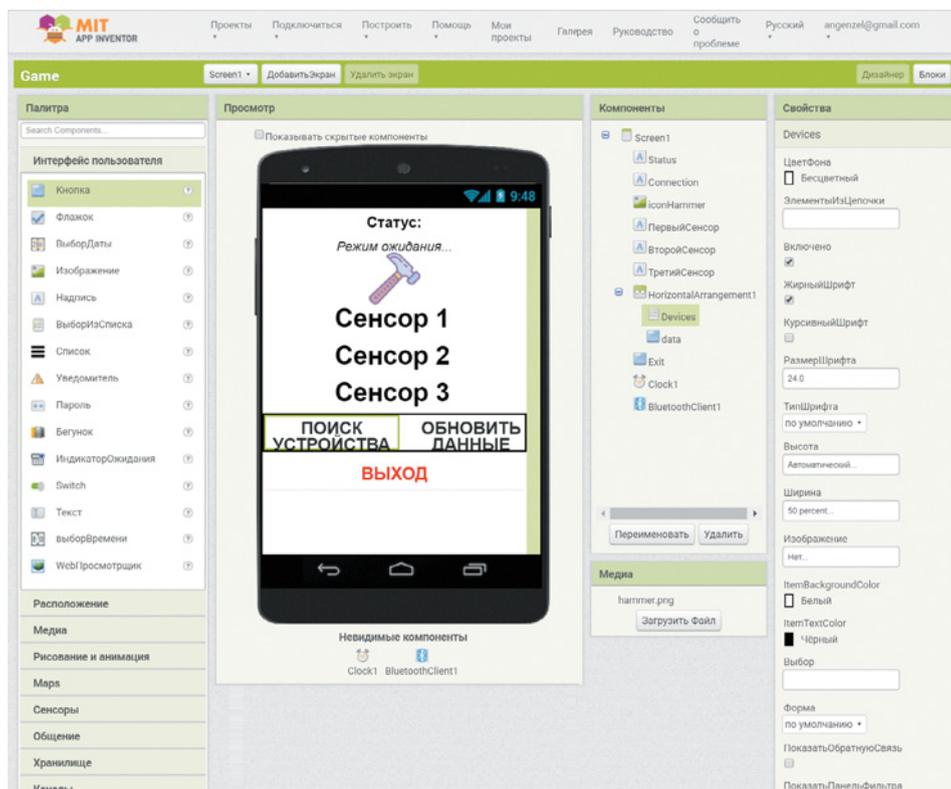


Рис. 5.32. Интерфейс программы

Для получения данных и доступа к последовательному порту устройства с помощью мобильного телефона установите на смартфон приложение Game. Вы можете его скачать на страничке этой книги на сайте издательства «Лаборатория знаний», а можете создать аналог самостоятельно с помощью web-приложения **AppInventor2**. Это не потребует дополнительных знаний. Свое приложение мы разместили в галерее: ai2.appinventor.mit.edu/?galleryId=6606960705798144

Интерфейс программы изображен на рис. 5.32.

Сначала рассмотрим код. Он написан на визуальном языке Blockly, по структуре очень похожем на Scratch или Snap!. Блоки приложения выглядят следующим образом. Сначала нужно создать пустой лист, куда будут переноситься принимаемые с устройства данные:

инициализировать глобальную data в  создать пустой лист

Затем идут блоки, отвечающие за запрос списка доступных к подключению устройств и непосредственно подключения (рис. 5.33).

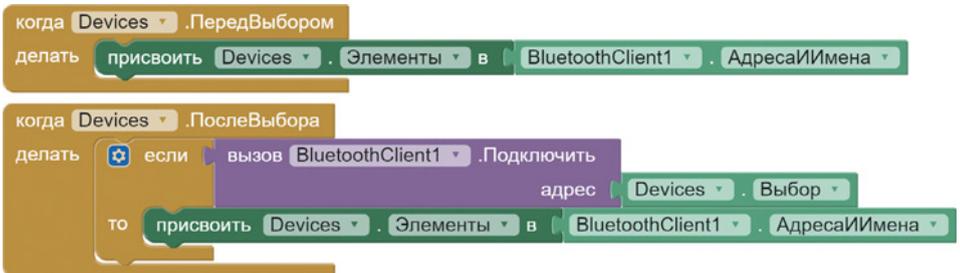


Рис. 5.33. Блоки, отвечающие за запрос списка

Когда устройство подключено, программа должна показать соответствующую информацию (рис. 5.34).

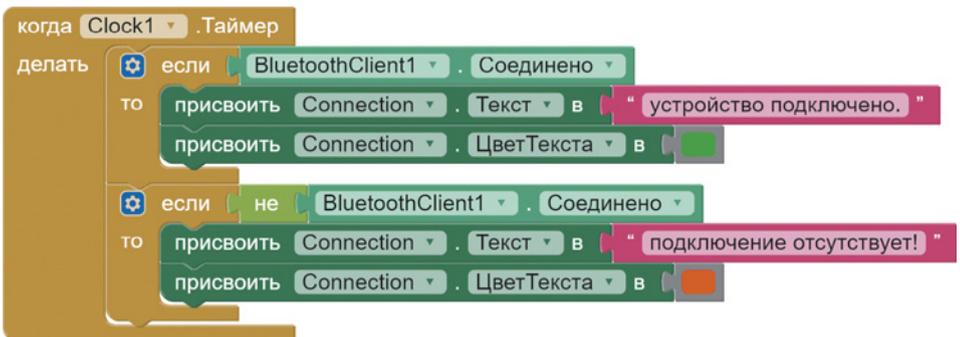


Рис. 5.34. Устройство подключено

Самая важная часть программы — это получение данных по Bluetooth от собранного тренажера. Когда пришедшая строка будет считана, ее следует сразу же разбить на элементы, используя в качестве разделителя пробел. Затем информация по каждому элементу выводится в соответствующую строку на экране (рис. 5.35).

Правилом хорошего тона считается наличие явной кнопки выхода из приложения:



Вернемся на вкладку **Дизайнер** и рассмотрим подробнее внешний вид экрана приложения (рис. 5.36). Он будет единственным, но в AppInventor2 вы можете настраивать экраны для разных

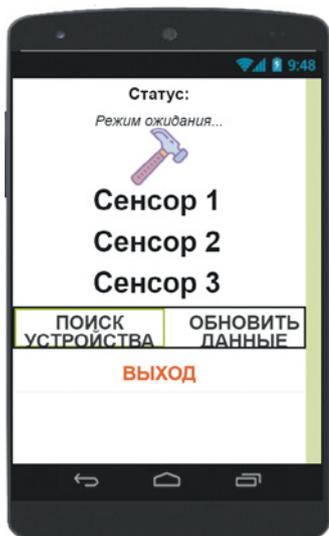


Рис. 5.36. Внешний вид экрана приложения

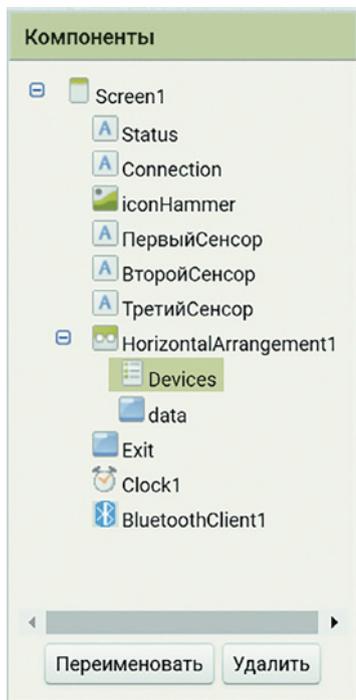


Рис. 5.37. Панель компонентов

событий, тогда при нажатии кнопки или другом действии изображение на экране телефона будет полностью меняться.

В рассматриваемом проекте мы поместили на экран индикатор подключения устройств по Bluetooth (Status), три надписи, которые будут меняться в зависимости от полученных данных (с номерами сенсоров) и три кнопки: кнопку поиска доступных устройств (Devices), открывающую соответствующий экран, кнопку запроса данных (data) о нажатии при подключенном тренажере и кнопку закрытия приложения (Exit). На панели компонентов это выглядит, как показано на рис. 5.37.

Вы можете отредактировать приложение прямо на сайте или создать его с нуля. Для подключения найдите в списке доступных устройств Bluetooth Bee. В качестве кода по умолчанию для всех моделей установлено 0000 или 1234.

Запустите устройство и проверьте его работоспособность.

Дополнительные задания

Придумайте игры с ребенком и сделайте специальные карточки. Что еще можно делать с данным устройством? Как бы вы его могли улучшить?

В табличном процессоре создайте дневник тренировок. Выпишите ответы на следующие вопросы:

1. Как часто вам самим удается правильно распределять силу? А испытываемому (ребенку)?
2. Есть ли связь между номером попытки в день и точностью выполнения задания?

Глава 6. Применение робототехники в различных сферах

6.1. Робототехника в современном мире

Современные робототехнические платформы способны к локальному и глобальному взаимодействию, поэтому робототехника получила широкое применение практически во всех сферах жизнедеятельности. Рассмотрим некоторые области, использующие средства *интеллектуальной автоматизации* (с выбором сценария в зависимости от внешних факторов).

Робототехника на производстве сегодня является основой производительности и безопасности.

Автоматизированные линии на заводах и фабриках (рис. 6.1) часто объединены в единую сеть. Раньше оборудование было автономным, остановить физически независимые друг от друга автоматы (например, печатающие обложки и печатающие страницы книг) можно было только вручную. Сегодня же, получив



Рис. 6.1. Автоматизированная линия

сигнал об ошибке или, например, перегреве одного из автоматов, управляющий блок остановит зависящие компоненты линии или снизит темп производства, чтобы успеть устранить ошибку без серьезных потерь.

Автоматические системы проверки электронных и механических компонентов внедрены в производство повсеместно. Самая известная компания, производящая подобные робототехнические решения, — *Siemens*. Большинство таких систем программируется на языке C — основе языка Wiring.

Кроме того, роботизированные системы используются при *обслуживании производственных помещений*, например для удаленного управления освещением, его интенсивностью; для контроля показаний датчиков влажности, загазованности и вредных веществ в воздухе. Это позволяет сделать труд работников безопасным и комфортным и хранить сырье и продукцию в подходящих условиях.

Стоит отметить, что автоматизированные системы обслуживания помещений также используются в вузах, некоторых школах и других общественных местах; они поддерживают определенную атмосферу (температуру, освещение и т. д.) в музеях, книжных хранилищах и библиотеках, управляют праздничной подсветкой и светомузыкальными фонтанами (рис. 6.2). Ведущей в этой области является немецкая фирма *Schneider Electric*, применяющая для программирования *графический язык*, который можно также использовать для программирования плат Arduino (например, с помощью среды *FLProg*).

Автоматизированные робототехнические системы применяются для *обслуживания зданий и управления коммунальным хозяйством* целых микрорайонов. Роботы, в зависимости от температуры окружающей среды, сезона и других факторов, управляют нагревом воды в системе отопления, регулируют напор холодной воды, проверяют состояние крыш, записывают номера припаркованных во дворе машин, распознают лица злоумышленников. При поломке лифта, пожаре или попытках взлома они автоматически вызывают соответствующие специальные службы и делают многое другое. Такое управление, чаще всего встроенное в само здание, получило название «*умное здание*» (*smart building*). Системы, используемые локально и устанавливаемые единственным пользователем в собственной квартире без внедрения в конструкции (стены), называются привычным и прочно вошедшим в обиход словосочетанием «*умный дом*» (*smart home*).

В *социальной сфере* робототехнические системы применяются для обработки результатов опросов, быстрой и качественной



Рис. 6.2. Автоматическое управление светомузыкальными фонтанами

проверки результатов голосований и контроля на избирательных участках. Автоматические записи в очереди, формирование заказов в интернет-магазинах, на складах и многое другое физически основано на робототехнике.

Робототехника широко применяется в *медицине*: от исследования и проверки новых формул лекарств до создания бionicеских протезов. Системы тревожных кнопок и автоматизированного оповещения социально незащищенных групп граждан встраиваются как компоненты «умного дома». Однако робототехнические системы не ограничены помещениями. Например, подключенные к единой сети терминалы контроллеров в общественном транспорте, автоматизированные системы контроля оплаты проезда и даже турникеты в метро являются частью большой *роботизированной системы города*.

В *транспортной сфере* роботизированные системы применяются для решения проблем *логистики* — управления материальными и людскими потоками. Так осуществляется, например, контроль за движением междугородных поездов, составов метро, наземного городского транспорта, включая информацию на остановках о приближении автобусов и троллейбусов (рис. 6.3),



Рис. 6.3. Информационное табло на остановке общественного транспорта

почтовым сообщением между населенными пунктами, доставкой грузов и т. п. Многие из этих систем обмениваются информацией друг с другом, в результате чего работа контролируемых ими сфер становится более эффективной и менее затратной.

В *сельскохозяйственном секторе* робототехнические комплексы применяются для определения состояния почвы, автоматического полива и подкормки культур, проверки качества продукции (например, молока, мяса или яиц) и условий ее хранения.

Сегодня в этом секторе существуют полностью автоматизированные сельскохозяйственные производства. Прослеживается тенденция к роботизации небольших хозяйств. Однако в отличие от крупных производителей фермы широко применяют робототехнические решения *на основе Arduino и совместимых платформ*.

В *военно-промышленном комплексе* робототехническими решениями оснащают не только современную технику, включая беспилотные самолеты, вертолеты и танки, но и производство маскировочных комплектов, специальных тканей и прочего хозяйственного обеспечения армии.

Робототехника является одной из прикладных областей *искусственного интеллекта*. Оснащение робототехнических си-

стем программными решениями, способными реагировать на ситуации, не предусмотренные программистом, является одной из главных задач современности. Сегодня уже существуют медицинские роботы, способные самостоятельно составить лекарство, и роботы, умеющие самостоятельно построить и преодолеть сложный маршрут по пересеченной местности.

Робототехника прочно вошла во все сферы нашей жизни. Робототехнические платформы открытого типа (Arduino, Raspberry Pi и др.) позволяют самостоятельно реализовать относительно простые решения, оптимизирующие многие рутинные процессы. Использование в этих платформах языка программирования, родственного языку программирования в большинстве серьезных производственных систем, позволяет создавать компоненты, которые можно интегрировать как дополнительные функции в реальном производственном оборудовании.



Вопросы

1. В чем заключается особенность интеллектуальной автоматизации?
2. В каких сферах применяется робототехника? Ответ поясните примерами. Что общего у производственной робототехники и Arduino?
3. Что такое «умный дом»? На примере собственного дома или школы определите, какие компоненты относятся к «умному зданию», а какие — к «умному дому».
4. Как применяется робототехника на общественном транспорте? Приведите примеры.
5. Как применяется робототехника в сельскохозяйственном секторе? Приведите примеры.
6. Какие задачи решают роботизированные системы на основе алгоритмов искусственного интеллекта?



Практические задания

Задание 1. Сельскохозяйственный сектор. Придумайте и реализуйте проект автоматического полива растений. *Подсказка:* могут понадобиться подвешенный резервуар, гибкая трубка с зажимом, сервомотор и датчик влажности почвы.

Задание 2. Производство. Придумайте и реализуйте проект управления поведением автомата в зависимости от изменяющихся условий (задымленность, температура) и времени суток.

Задание 3. Социальная сфера. Придумайте и реализуйте проект кабинки для голосования за одного из двух кандидатов, которая автоматически подсчитывает голоса избирателей. *Подсказка:* для учета голосов можно применить тактовые кнопки или два датчика линии.

6.2. Arduino и производственные языки

Специальные графические языки программирования, применяемые в производственной робототехнике, требуют установки среды программирования **FLProg**. Свежая версия программы доступна на сайте разработчика и распространяется бесплатно. Прямая ссылка на страницу загрузки: <https://flprog.ru/category/downloads/distrutives/actualversion/>¹

Выберите установщик вашей операционной системы, установите программу в путь по умолчанию и запустите ее.

Создайте проект в программе **FLProg**, в котором индикаторами состояния освещения в помещении будут светодиод и пьезодинамик (зуммер), управляемые программой на одном из языков программирования производственных автоматов.

Физическое подключение представлено в табл. 6.1

Таблица 6.1

Компонент	Порт Arduino
Фоторезистор	Аналоговый вход № 0
Кнопка	Цифровой вход № 1
Светодиод	Цифровой выход № 3 с поддержкой ШИМ
Зуммер	Цифровой выход № 5 с поддержкой ШИМ

1. Создайте новый проект (рис. 6.4).

В появившемся диалоговом окне выберите язык программирования FBD и микроконтроллер Arduino Uno из выпадающего списка (рис. 6.5).

FBD (Function Block Diagram — диаграммы функциональных блоков) — это графический язык программирования стандарта МЭК 61131-3, использующий наборы библиотечных блоков. Блоки представляют собой подпрограммы, функции или функциональные компоненты: триггеры, таймеры, счетчики, блоки обра-

¹ Уважаемые ребята! Обращаем ваше внимание на написание `distrutives` в ссылке.

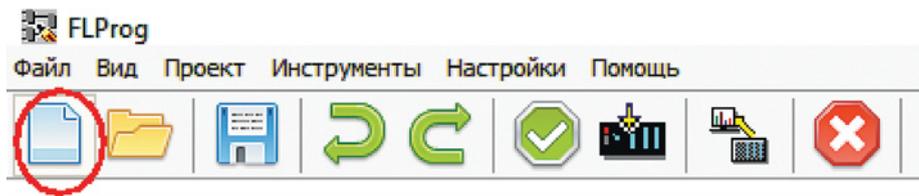


Рис. 6.4. Создание нового проекта

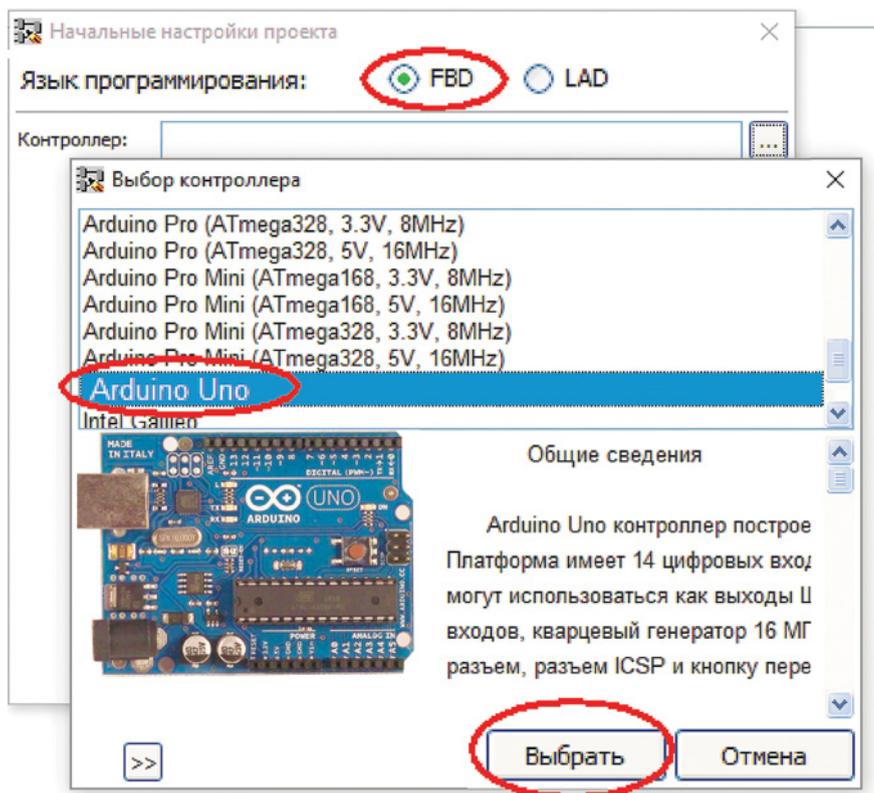


Рис. 6.5. В появившемся диалоговом окне выберите язык программирования FBD и микроконтроллер Arduino Uno из выпадающего списка

ботки аналогового сигнала, математические операции, логические элементы (И, НЕ, ИЛИ) и т.д. Из блоков составляются цепи, внутри которых команды выполняются строго последовательно. Результат вычисления записывается во внутреннюю переменную или подается сразу на выход контроллера. Данный язык используется при программировании производственных автоматов многими фирмами, включая Siemens.

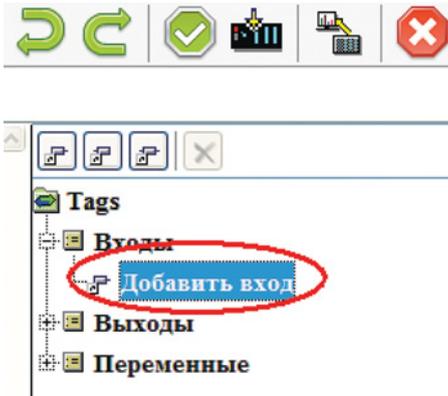


Рис. 6.6. Добавление нового входа для обработки сигнала фоторезистора

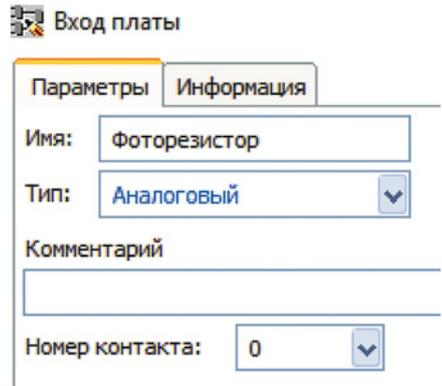


Рис. 6.7. Присвойте входу имя **Фоторезистор**, укажите тип и номер контакта

В среде для выбора доступен еще язык *LAD (Ladder Diagram, LD, PKC)* — язык релейной (лестничной) логики. Подходит для замены логических схем, выполненных на релейной технике. Язык ориентирован на инженеров по автоматизации, поэтому логические операции в нем представлены в виде электрических цепей с замкнутыми или разомкнутыми контактами, пара контактов отождествляется с переменной. Для нашего проекта использование этого языка нецелесообразно.

Подтвердите выбор языка командой **Выбрать**.

2. Добавьте новый вход для обработки сигнала фоторезистора, выбрав **Добавить вход** (рис. 6.6). В открывшемся окне параметров присвойте входу имя **Фоторезистор**, укажите тип и номер контакта (рис. 6.7).

После нажатия кнопки **Готово** перетащите появившееся название входа на рабочую область (рис. 6.8).

3. По аналогии добавьте вход для кнопки и перетащите элемент на рабочую область (рис. 6.9).

4. Объявите выход, предназначенный для светодиода. Добавлять входы и выходы можно также с помощью специальных кнопок (рис. 6.10). Присвойте выходу имя **Светодиод**, выберите из раскрывающегося списка порт с поддержкой ШИМ, укажите номер порта. Перетащите элемент на рабочую область.

5. Для добавления пьезодинамика выберите соответствующий компонент из списка встроенных блоков и перетащите его на рабочую область (рис. 6.11).

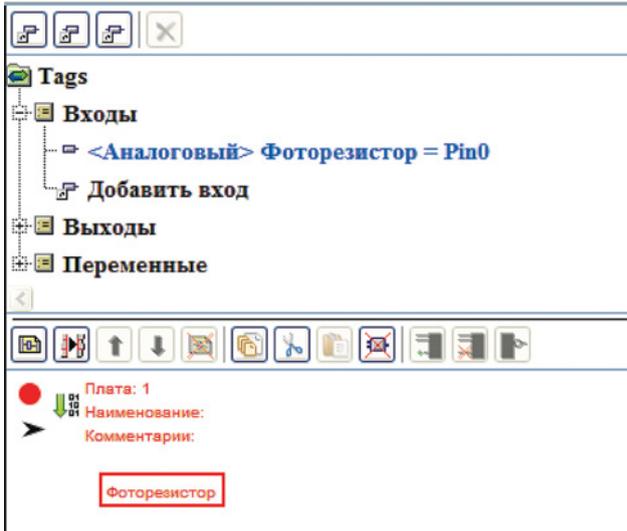


Рис. 6.8. Перетащите появившееся название входа на рабочую область

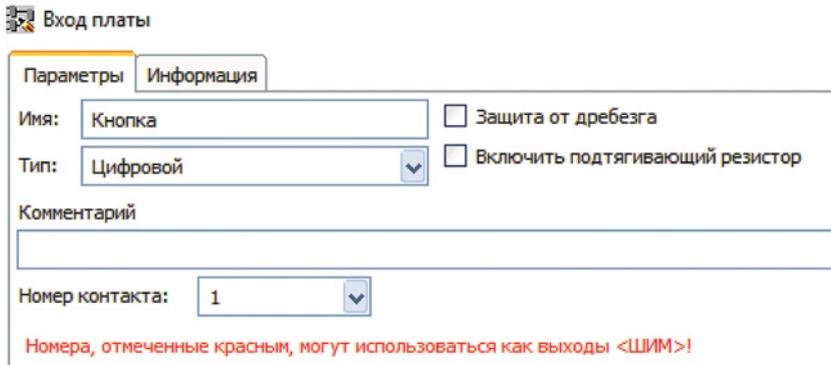


Рис. 6.9. Добавьте вход для кнопки и перетащите элемент на рабочую область

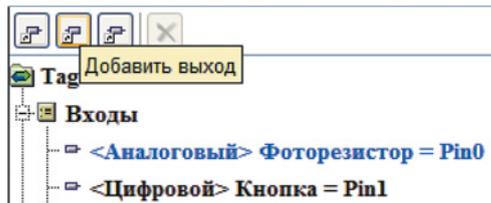


Рис. 6.10. Добавлять входы и выходы можно также с помощью специальных кнопок

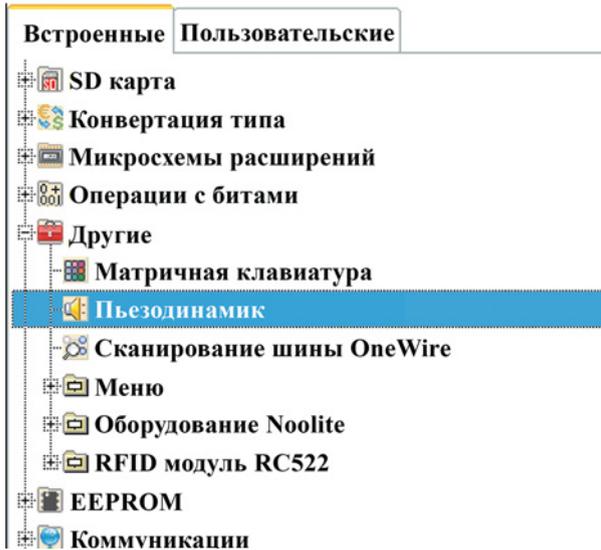


Рис. 6.11. Добавление пьезодинамика

Двойным нажатием клавиши по блоку **Buzzer** (пьезодинамик) откройте окно параметров. Выберите номер контакта и установите значения, как показано на рис. 6.12.

6. Фоторезистор, являясь аналоговым датчиком, передает на плату значения 0–1023. ШИМ может выдавать сигнал в градации значений от 0 до 255. Частота сигнала пьезодинамика имеет диа-

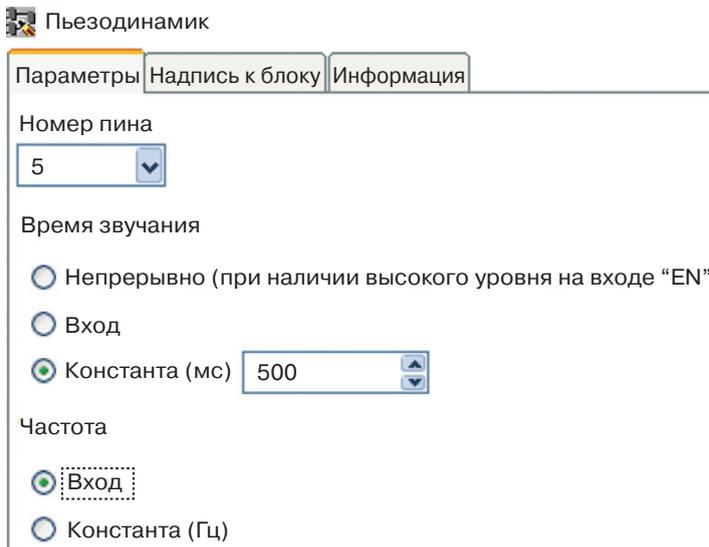


Рис. 6.12. Выбор контакта и установка значений для пьезодинамика

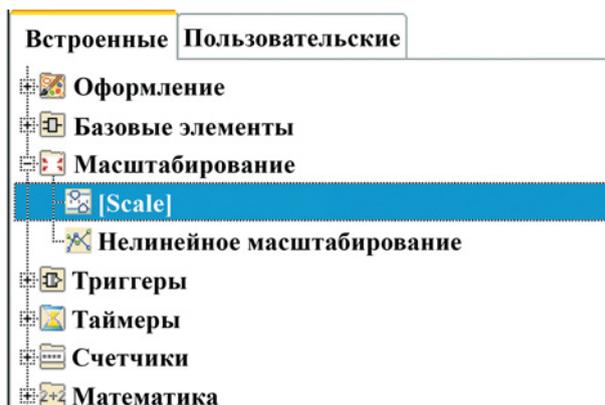


Рис. 6.13. Встроенный блок линейного масштабирования Scale

пазон 20–20 000 Гц, и ее конкретное значение нужно передавать числом.

Для корректной работы прототипа необходимо преобразовать значения с применением масштабирования.

Поскольку здесь наблюдается линейная зависимость между сигналом, получаемым от фоторезистора, и желаемым напряжением на выходе с ШИМ, используйте встроенный блок линейного масштабирования **Scale** (рис. 6.13). Перетащите два блока на рабочую область.

Откройте окно параметров первого блока масштабирования и проверьте, что там введены значения, показанные на рис. 6.14. Этот блок будет использоваться для светодиода.

Откройте окно параметров второго блока масштабирования (рис. 6.15). Для того чтобы сигналы пьезодинамика различались на слух, выберите частоты в диапазоне от 300 до 800 Гц.

Перейдите во вкладку **Надпись к блоку** (рис. 6.16) и введите в поле название, чтобы не перепутать блоки в дальнейшем.



Рис. 6.14. Окно параметров первого блока масштабирования

Масштабирование

Параметры	Надпись к блоку	Информация	
Верхний предел входного сигнала	<input type="radio"/> Вход	<input checked="" type="radio"/> Константа	<input type="text" value="1023"/>
Нижний предел входного сигнала	<input type="radio"/> Вход	<input checked="" type="radio"/> Константа	<input type="text" value="0"/>
Верхний предел выходного сигнала	<input type="radio"/> Вход	<input checked="" type="radio"/> Константа	<input type="text" value="800"/>
Нижний предел выходного сигнала	<input type="radio"/> Вход	<input checked="" type="radio"/> Константа	<input type="text" value="300"/>

Рис. 6.15. Окно параметров второго блока масштабирования

Масштабирование

Параметры	Надпись к блоку	Информация
<input checked="" type="checkbox"/> Показывать		
Выравнивание		
<input checked="" type="radio"/> Влево	<input type="radio"/> По центру	<input type="radio"/> Вправо
<input type="text" value="Частота зуммера"/>		

Рис. 6.16. Вкладка «Надпись к блоку»

7. Соедините блоки в схему (рис. 6.17). Сигнал с фоторезистора должен подаваться в блоки масштабирования. Каждый блок масштабирования имеет два порта: **Input (I)** — вход, **Quit (Q)** — выход. К выходу неподписанного блока масштабирования подключите блок светодиода.

У блока **Buzzer** два входа: **F** (foreign) — внешний вход, на который подаются данные из другого блока, и **En** (encoder) — ключ разрешения, на который поступают только два значения: логический ноль или логическая единица. Когда на **En** подана единица, зуммер работает, если ноль — он выключен. Подключите ко входу **F** блока пьезодинамика выход второго блока масштабирования, а ко входу **En** — кнопку (рис. 6.18).

8. Выполните проверку проекта (рис. 6.19).

9. Скомпилируйте проект (рис. 6.20). При нажатии кнопки программа **FLProg** запустит среду **Arduino IDE**, в поле редактора которой будет внесен проект, переведенный на язык **Wiring**. Скомпилируйте и загрузите скетч на плату **Arduino Uno** обычным способом.

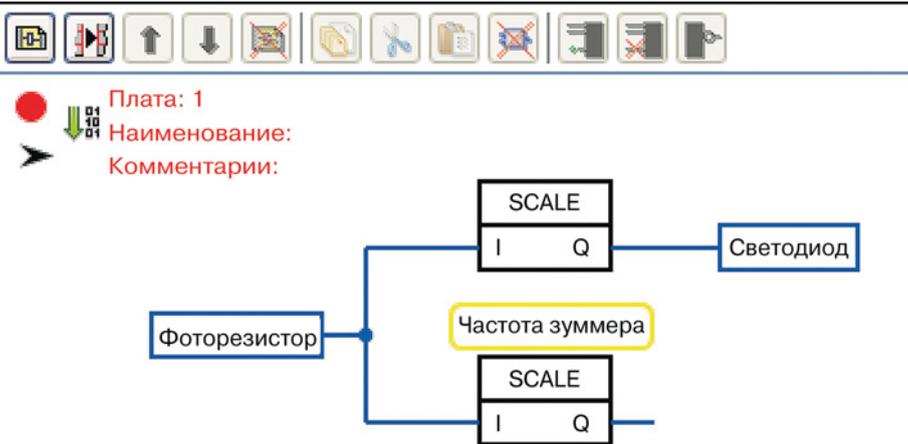


Рис. 6.17. Схема соединения блоков

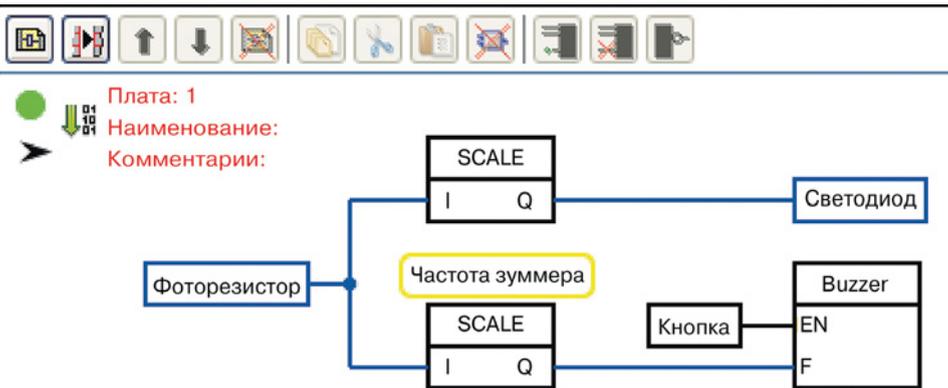


Рис. 6.18. Подключение блока Buzzer

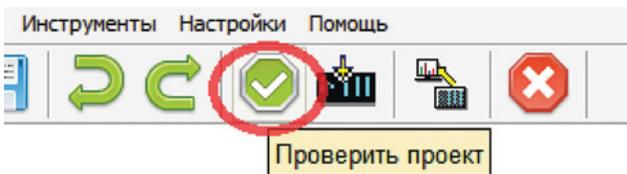


Рис. 6.19. Проверка проекта

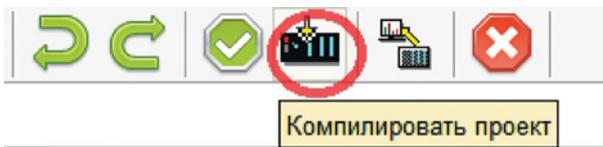


Рис. 6.20. Компиляция проекта

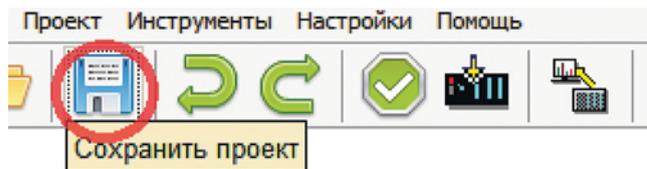


Рис. 6.21. Сохранение проекта в FLProg

10. Закройте среду Arduino IDE и сохраните проект в FLProg (рис. 6.21). Файл будет сохранен в собственном формате программы с расширением .flp.

6.3. Оформление робототехнических проектов

Предисловие. В этом разделе будет очень много «умных слов» и серьезных разговоров, которые так любят взрослые. Но они нужны, чтобы взрослые принимали вас на равных.

Вы научились использовать широкие возможности платформы Arduino и готовы заниматься настоящими инженерными проектами. Но одних знаний о программировании и сборке недостаточно! Надо учиться правильно их представлять.

Инженерная культура предусматривает опору на законодательную базу, т.е. всевозможные стандарты, регламентирующие инженерную деятельность, поэтому здесь и далее мы будем ссылаться на международные и государственные стандарты. Их соблюдение помогает инженерам понимать друг друга, а заказчикам и потребителям быть уверенными в качестве результата. В ГОСТ Р 54869-2011 «Проектный менеджмент. Требования к управлению проектом» даны следующие определения:

Проект — это комплекс взаимосвязанных мероприятий, направленный на создание уникального продукта или услуги в условиях временных и ресурсных ограничений.

Продукт проекта — измеримый результат, который должен быть получен в ходе реализации проекта.

Цель любого проекта — создание максимально эффективного решения при минимальных затратах ресурсов. Это значит, что инженер тщательно изучает область проблемы и продумывает такое решение, которое бы помогало достигнуть требуемого результата с минимально возможными затратами материала и времени. Конечно, снизить стоимость продукта возможно не всегда: ино-

гда требуется жертвовать финансовыми вложениями ради улучшения характеристик продукта.

Любой проект обладает следующими характерными особенностями, отличающими его от любого другого вида и формы деятельности:

- проблема проекта четко сформулирована;
- результат проекта всегда уникален и измерим;
- проект ограничен по времени;
- ресурсы проекта, включая время и команду, всегда ограничены;
- проект должен иметь строго сформулированное техническое задание;
- проект «проживает» все этапы жизни без исключения;
- обладает сопроводительной документацией.

Совокупность сопроводительной документации проекта, отображающей каждый этап его жизни, объединяется в *инженерную книгу* или *инженерный журнал*. Он полностью отражает ваш проект. Его составление поможет вам производить оценку собственного проекта, не терять важные идеи и систематизировать деятельность.

В связи с тем что инженерная книга отображает этапы жизни проекта, ее создание идет параллельно созданию проекта. Последним этапом является оформление готовой инженерной книги.

Это интересно!

IEC 81346 (2009). Промышленные системы, установки и оборудование и промышленная продукция. Принципы структурирования и кодированные обозначения.

ГОСТ Р ИСО 15926-1-2008. Промышленные автоматизированные системы и интеграция. Интеграция данных жизненного цикла для перерабатывающих предприятий, включая нефтяные и газовые производственные предприятия.

ГОСТ 34.601-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания.

ГОСТ Р 57100-2016/ISO/IEC/IEEE 42010. Системная и программная инженерия. Описание архитектуры.

ГОСТ Р ИСО 9241-210-2012. Эргономика взаимодействия человек-система. Часть 210. Человекоориентированное проектирование интерактивных систем.

ГОСТ Р ИСО 9001. Системы менеджмента качества. Требования.

ГОСТ Р ИСО/МЭК 15288. Информационная технология. Системная инженерия. Процессы жизненного цикла систем.

В зависимости от того, какое требуется решение, определяются характер деятельности и, как следствие, особенности этапов жизненного цикла проекта. Так, если решается техническая задача, речь идет об инженерной деятельности, результатом которой всегда являются продукты и изделия.

Продуктом робототехнических проектов являются, несомненно, *искусственные системы* — системы, которые созданы человеком и состоят из одного или нескольких следующих элементов: технические средства (робототехническая платформа), программные средства (ПО для моделирования, среды программирования и т. д.), люди, процессы (например, процесс оценки), процедуры (например, инструкции оператора в управляемом раунде игры), основные средства и природные ресурсы (например, вода, объекты живой природы). Процессы жизненного цикла подобных систем регламентированы в ГОСТе Р ИСО/МЭК 15288-2005 «Информационная технология. Системная инженерия. Процессы жизненного цикла систем», а концепция самих циклов описана в Приложении В этого документа. Итак, *жизненный цикл систем* проходит следующие *стадии*:

- 1) замысла;
- 2) разработки;
- 3) производства;
- 4) применения;
- 5) поддержки применения;
- 6) прекращения применения и списания.

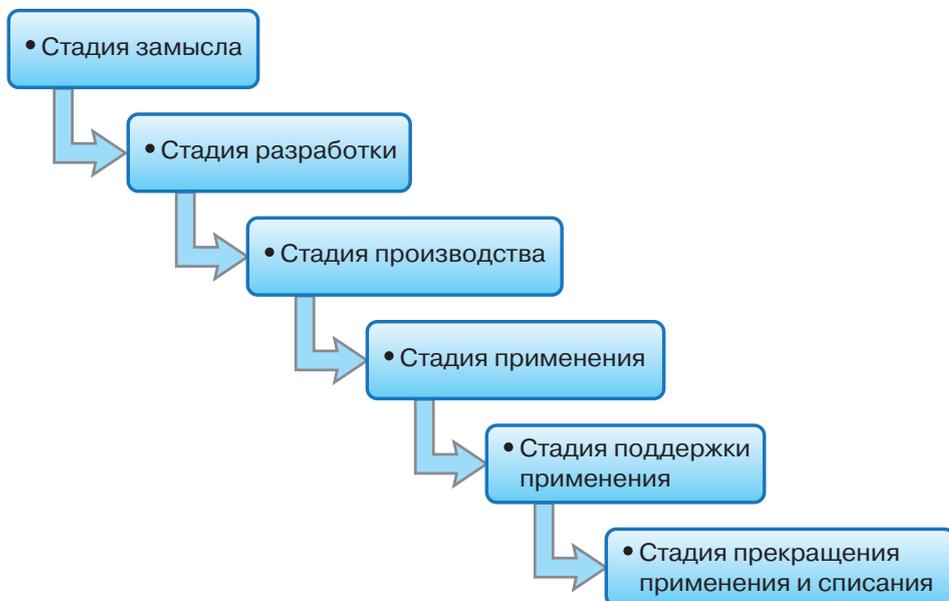
Для школьных робототехнических проектов этот список может выглядеть так:

- 1) стадия замысла;
- 2) разработка;
- 3) сборка и программирование;
- 4) участие в соревнованиях и конкурсах;
- 5) ремонт и совершенствование робота перед следующими этапами соревнований и конкурсов;
- 6) разборка робота на детали.

Методология определяет подход к работе команды, построению взаимодействия. Кратко рассмотрим два подхода, наиболее распространенных в инженерной деятельности.

Первая методология является традиционной. Она называется *каскадной (или waterfall) методологией разработки* и предусматривает постепенное выполнение частей проекта в рамках одной системы, разделяя задачи по различным отделам внутри компании и ставя им соответствующие сроки. Таким образом, в дан-

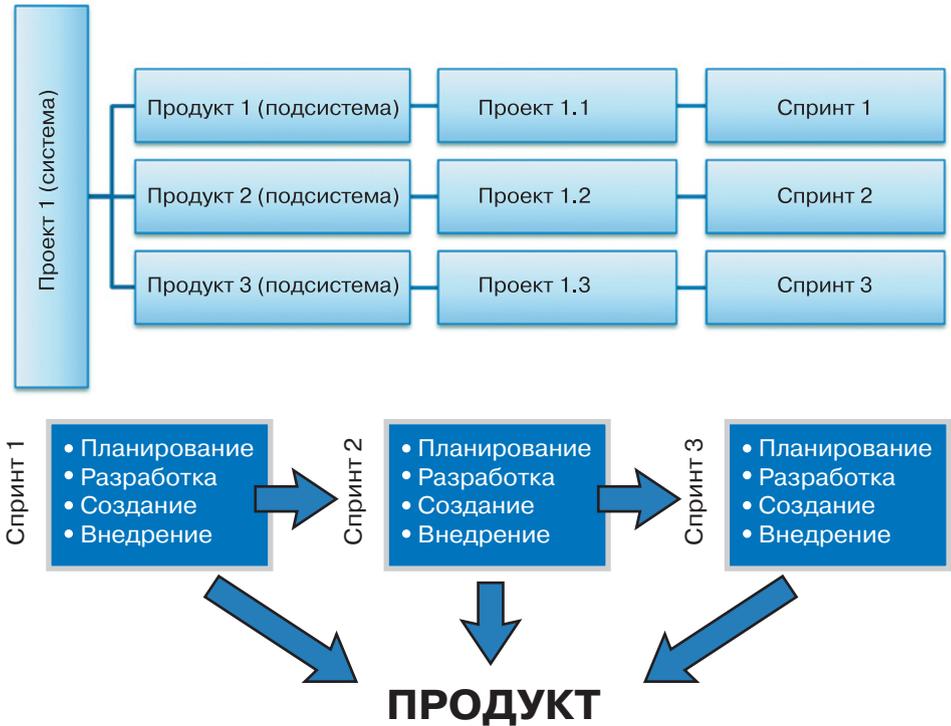
ной методологии составляется график, единый для всей системы. Этапы выполняются последовательно.



Здесь создается продукт в целом, на подсистемы раскладываются лишь части согласно функциям имеющихся на предприятии отделов (например, конструкторский отдел и отдел, занимающийся непосредственно производством). Такая методология удобна для долговременных проектов, которые выполняются строго установленным и неизменным числом участников с определенной квалификацией (команда исполнителей строго фиксирована) и строгой формулировкой от заказчика, получаемой в начале работы.

С течением времени появляются новые техники и методологии разработки, которые позволяют усовершенствовать данный процесс. С 2000 года широкую популярность приобрела новая *гибкая методология разработки (agile)*, в которой присутствуют аналогичные стадии. Однако проект разбивается на кратковременные подпроекты, результатом которых является промежуточный продукт — функционирующий, имеющий собственную цену и пользу. Это может быть устройство, программа или, например, деталь. Впоследствии такой продукт будет включен в итоговый продукт. Обычно каждый подпроект длится 1–4 недели — этот временной отрезок называют *спринтом*. Спринт содержит практически все этапы классической модели, кроме утилизации. Внедрение продукта спринта может быть необязательным. После

каждого спринта проводится обсуждение результатов и работы команды, т. е. оценка собственных ошибок и достижений. Затем команда может быть перераспределена, изменен график работы или внесены иные изменения в формат или подход. Кроме того, гибкая методология предусматривает ежедневные короткие (около 15 минут) совещания для организации деятельности и своевременного отслеживания прогресса. Чаще всего для реализации методологии agile используется подход SCRUM:



В ходе работы свои коррективы может вносить и заказчик, например, по результатам применения продукта одного из первых спринтов. Обратная связь осуществляется постоянно. Проект, создаваемый с применением методологии agile, более гибок и позволяет вносить изменения. Также эта методология хороша для команд с непостоянным составом или возможной сменой ролей внутри команды, в условиях невозможности точного планирования на длительный срок (например, больше чем на учебную четверть, состоящую из четырех-пяти недель).

Существуют и другие методологии разработки, о которых вы можете узнать самостоятельно. Этапы жизненного цикла систем определяют этапы работы команды над проектом вне зависимости от выбранной методологии. Если вы овладеете различны-

ми методологиями разработки, включая распределение ролей и взаимосвязь этапов и действий в них, вы сможете эффективно справляться с любым проектом любого реального заказчика. Более того, получив опыт, вы сможете выступить в качестве собственного заказчика — создать собственный стартап.

Прежде чем вы приступите к плотной, детальной работе над каждым пунктом, рассмотрим, что представляют собой данные этапы и что требуется на каждом из них от инженеров.

ЭТАП 1. Постановка и осознание проблемы

Правильно сформулированная проблема влияет на постановку целей, задач и, как следствие, на достигнутые результаты. Эта истина знакома вам из любых учебных проектов, начиная от информационно-поисковых и заканчивая научно-исследовательскими работами (НИР). Проблема проекта для робототехнических соревнований вытекает из отсутствия эффективного и/или оптимального решения с применением регламентированных средств при условиях ограниченности ресурсов и выдвигаемых требований в ситуации, описанной в регламенте игры/сезона. Дополнительными факторами, влияющими на постановку проблемы, выступают требования, предъявляемые к команде или индивидуальному участнику его учителем или образовательным учреждением, или иной организацией. Для проектов, не связанных с соревнованиями, проблема возникает из рассматриваемых в теме конференции или конкурса тезисов, из поставленных вами самими вопросов, ваших интересов.

Однако формулирования проблемы недостаточно. Для преобразования проблемы в инженерную задачу необходимо понимание ее как системы, состоящей из требований и условий, т. е. вводных данных, и противоречия, которое необходимо разрешить. Зачастую двигателем выступает вопрос: «А почему так нельзя?» Действительно, почему? Попробуйте понять, как, не меняя идеи, станет можно, при каких условиях.

ЭТАП 2. Выбор стратегии решения

Когда задача ясна, выполняется поиск возможных решений. На данном этапе необходимо сформулировать основную идею, за счет которой ваш проект будет лучшим. Сначала анализируются все доступные стратегии и способы решения поставленной проблемы. Затем из предложенных выбирается та единственная, что наиболее выгодна с точки зрения ресурсов и прибыли, планируемых

результатов. В любом случае, для выбора стратегии вам придется провести некоторое *исследование конкретной области (analysis)*.

ЭТАП 3. Требования и ограничения для выбранного решения

Здесь начинается проработка конкретной выбранной стратегии. В качестве требований выступают официальные пункты из правил соревнований или конкурса либо требования к школьным проектам. Техника безопасности при работе с электрическими компонентами и инструментами тоже входит в перечень требований. Дополнительные требования выдвигаются непосредственно членами команды, учителями и руководителями учреждений.

Также в регламенте указываются ограничения. Они могут касаться как количества человек в команде и их возраста, так и технических компонентов. Особое внимание следует уделить временным ограничениям, т.е. срокам: длительность четверти, сроки выставки и т.д.

Кроме указанных требований и ограничений, существует целый ряд вопросов, которые не указываются в столь явном виде, но подразумеваются для каждого инженерного проекта. Во-первых, вы должны определить степень надежности своего проекта. Подумайте, нужен ли он только для выполнения одной задачи в ближайшее время или конструкция может использоваться в течение нескольких лет? И где она будет использоваться: в помещении или на улице? Во-вторых, задать минимальные характеристики готового робота. Например, если робот должен убирать мусорные баки, то его механизмы должны удерживать вес мусора с учетом веса бака. В-третьих, следует обратить внимание, что роботы не обязательно должны быть автономными, они могут управляться через Интернет или просто подключаться по локальной сети к рабочей станции оператора (например, охранника). Информационные системы могут предусматривать взаимодействия типа *человек–система* или *система–система*, т.е. вы должны также предусмотреть требования и ограничения, выдвигаемые к UI (user interface/пользовательскому интерфейсу, «как выглядит») и частично к UX (user experience/опыту взаимодействия, ощущения пользователя от использования, «что делает»). *Пользовательский интерфейс* — это физические и технические методы взаимодействия с пользователем, направленные в основном на обмен информацией (ввод и вывод) с системой в удобном для человека виде (например, кнопки). *Опыт взаимодействия* — это реализация соответствия ожидания пользователя

от произведенного им действия и результата, действительно реализуемого системой (например, ожидание, что нажатие большой красной кнопки приведет к запуску устройства). Таким образом, UI — это инструмент, видимое выражение для UX. Эти два компонента, которые никогда не бывают разделимы, крайне важны для взаимодействия человек–система. Если же предусматривается взаимодействие система–система, то речь может идти о создании сообществ («роя»), требованием для которых будет избежание столкновения или реализация следования. Другим примером взаимодействия система–система является управление компонентами «умного» дома.

ЭТАП 4. Формулирование концепции решения

Итак, когда область решения четко ограничена, можно переходить к формулированию шагов конкретного выбранного решения. На данном этапе пока не рассматриваются конкретные ресурсы. Здесь составляются блок-схемы алгоритмов поведения робота и алгоритмы действий операторов, если это требуется. На этом этапе появляется четкое представление о действиях робота. Таким образом, на четвертом этапе окончательно реализуется модель UX.

ЭТАП 5. Моделирование архитектуры

Архитектура искусственной системы — это абстрактное представление системы, включающее в себя идеализированные модели компонентов и модели взаимодействия между ними. Все элементы в ней взаимосвязаны, но являются всего лишь абстрактными моделями, что позволяет в дальнейшем на этапе производства (реализации) изменять состав ресурсов, заменять датчики на аналогичные, выбирать доступные или более выгодные по габаритам устройства и платы. К компонентам архитектуры относятся модели датчиков, устройств ввода-вывода, компьютеров, интерфейсов, каналов передачи информации, протоколы, интерфейсы, драйверы и т. д.

Моделирование — это процесс представления чего-либо с сохранением существенных для исследования/решения задачи свойств и связей. Например, при моделировании архитектуры системы для модели датчика освещенности важно лишь его функциональное назначение.

Моделирование производится в специальных компьютерных средах или на бумаге путем построения схем. Можно использо-

вать специфические языки. Для снижения итоговой стоимости проекта желательно пользоваться свободным программным обеспечением.

Чаще всего создаются многоуровневые модели, на верхнем уровне которых система разделена на функциональные подсистемы специального назначения. Каждая такая подсистема в свою очередь раскладывается на подсистемы, причем разделение производится по каналу с наименьшим потоком информации. Этот процесс называется *декомпозицией системы*. В случае использования agile-методологии каждая подсистема становится целью одного спринта.

Архитектура системы должна быть:

- надежна;
- ремонтпригодна — работоспособность в случае неисправности можно восстановить за минимальное время при относительно небольших затратах ресурсов;
- диагностируема — в случае возникновения неполадки ее можно обнаружить;
- тестируема — есть возможность определить ее правильную работу;
- проста в обслуживании и эксплуатации;
- безопасна;
- наращиваема и открыта — иметь возможность расширения и замены частей.

И самое главное — архитектура системы должна полностью реализовывать концепцию выбранного решения в установленных условиях и ограничениях.

ЭТАП 6. Ресурсная база

Ресурсная база определяется многими факторами и напрямую связана с ограничениями, к которым относятся не только пункты, указанные в правилах мероприятия, но и возможности вашего образовательного учреждения или ваши собственные. Помните, что ресурсы бывают материальные, информационные, финансовые, человеческие и временные. Важно учитывать их все.

На данном этапе производятся анализ и оценка имеющихся и требуемых ресурсов, возможность дополнительных закупок в рамках доступных финансовых ресурсов, распределяются обязанности.

ЭТАП 7. Техническое задание

Здесь происходит распределение задач по функциональным областям и областям деятельности между участниками. Это очень ответственный шаг, и крайне важно, чтобы распределение проводилось в согласовании с опытным руководителем проекта, например учителем.

Этапы 6 и 7 выполняются итерационно до тех пор, пока не будет достигнуто оптимальное распределение ресурсов, ролей и задач.

Ну вот теперь вы полностью погрузились в терминологию инженерных проектов и можете создать настоящий стартап, посвященный производственным роботам на основе любимейшей вам платформы Arduino. Вперед!

ЗАКЛЮЧЕНИЕ

Уважаемые ребята!

Вы прошли нелегкий, но интересный путь от знакомства с платой до изучения профессиональных тонкостей работы инженера. Давайте остановимся и оглянемся назад. Что же вы узнали?

Сначала вы открыли для себя интересный мир открытых робототехнических платформ, узнали, что такое прототипирование. На макетных платах вы собрали собственные прототипы, используя различные электронные компоненты: от однокомпонентных устройств до модулей и плат расширения с собственными микропроцессорами. На примере открытой платформы Arduino вы изучили принцип передачи информации изменением напряжения, принцип широтно-импульсной модуляции, физическую сторону протоколов UART, I²C, работу устройств OneWire и многое другое. В ваших руках горки плат и кучки проводов становились полноценными полезными устройствами!

При изучении курса вы познакомились с особенностями трех языков программирования, принадлежащих к различным группам. Знание графического пропедевтического языка программирования Snap! позволило без особых усилий и затрат времени получить результат работы программы, а также изучить различие между открытыми и замкнутыми (автономными) системами управления роботом. Перейдя к текстовому объектно ориентированному программированию на языке Wiring, вы стали на несколько ступеней ближе к популярному языку программирования C++, применяемому профессиональными программистами при написании операционных систем и других программ. Изучив язык стандартной среды Arduino IDE, вы с легкостью освоите синтаксис любого нового текстового языка. Затем вы познакомились с третьим языком — FBD. Оказалось, что графические языки — это не только обучающие инструменты, но и возможность программировать серьезную производственную робототехнику инженерами-электронщиками.

Теория имеет малую ценность без практики. В отличие от прохождения курсов, предусматривающих только учебные задания и компоненты, вы, выполняя небольшие упражнения и проекты, также изучали электрические схемы и обозначения, являющиеся универсальными для всех специалистов мира. В дальнейшем это послужит вам хорошим подспорьем при чтении схем проектов по

автоматике и телемеханике! К тому же, теперь вы умеете составлять и собственные схемы. Вы узнали, что автоматизированные системы применяются повсюду: в городской инфраструктуре, в медицине, ВПК, сфере развлечений и даже в ходе проведения выборов.

Чтобы систематизировать знания и применить их в жизни для достижения целей, вы изучили понятие жизненного цикла проектов и узнали, что для правильного оформления проектной документации следует опираться на государственные и международные стандарты (ГОСТы и др.).

Довольно большой путь, не правда ли? Мы гордимся вами. Вы доказали миру и в первую очередь себе, что способны на многое. Вас ограничивает лишь ваша фантазия. Значит, преград не существует. Творите, конструируйте, программируйте! А серия книг и проектов «Робофишки» обязательно вам в этом поможет.

Успехов, дорогие друзья!

Минимальные системные требования определяются соответствующими требованиями программ Adobe Reader версии не ниже 11-й либо Adobe Digital Editions версии не ниже 4.5 для платформ Windows, Mac OS, Android и iOS; экран 10"

Электронное издание для дополнительного образования

Серия: «РОБОФИШКИ»

**Салахова Алёна Антоновна, Феокистова Ольга Александровна,
Александрова Наталья Алексеевна, Храмова Марина Викторовна**

**ARDUINO®.
ПОЛНЫЙ УЧЕБНЫЙ КУРС.
ОТ ИГРЫ К ИНЖЕНЕРНОМУ ПРОЕКТУ**

Для детей среднего и старшего школьного возраста

Ведущий редактор *Т. Г. Хохлова*. Художники *В. А. Прокудин, Я. В. Соловцова*
Технический редактор *Т. Ю. Федорова*. Корректор *И. Н. Панкова*
Компьютерная верстка: *О. Г. Лапко*

Подписано к использованию 17.03.20.
Формат 155×225 мм

Издательство «Лаборатория знаний»
125167, Москва, проезд Аэропорта, д. 3

Телефон: (499) 157-5272, e-mail: info@pilotLZ.ru, <http://www.pilotLZ.ru>

Юные инженеры и их преподаватели!

Наш полный учебный курс, посвященный робототехнике и платформе **Arduino**[®], познакомит вас с особенностями аппаратного обеспечения и программирования **Arduino Uno**[®]. Кроме того, вы приобщитесь к электронике – научитесь читать, составлять и собирать действующие схемы из электронных компонентов. Наконец, мы покажем вам, как правильно оформлять инженерные проекты.

- ◆ В пособии много внимания уделено языкам программирования. Вы научитесь программировать на графических языках **Snap!** и **ArduBlock**, изучите текстовый язык **Wiring**, познакомитесь с производственными языками программирования.
- ◆ Пособие выполнено в формате последовательно выстроенных тем, сопровождаемых вопросами, практическими заданиями и проектами.

В первую очередь данный курс может быть встроен в урочный процесс в качестве модуля **«Робототехника»**. Также он будет полезен для внеурочной деятельности или самостоятельного изучения.

Курс может рассматриваться в качестве третьей ступени изучения робототехники после курсов С. А. Филиппова **«Уроки робототехники. Конструкция. Движение. Управление»** и Ю. А. Винницкого и К. Ю. Полякова **«Конструируем роботов на ScratchDuino[®]. Первые шаги»** издательства «Лаборатория знаний». Проектная часть курса может быть расширена серией книг **«РОБОФИШКИ. Конструируем роботов на Arduino[®]»**.

info@pilotLZ.ru

arduino@pilotLZ.ru

www.pilotLZ.ru

 vk.com/roboLz

