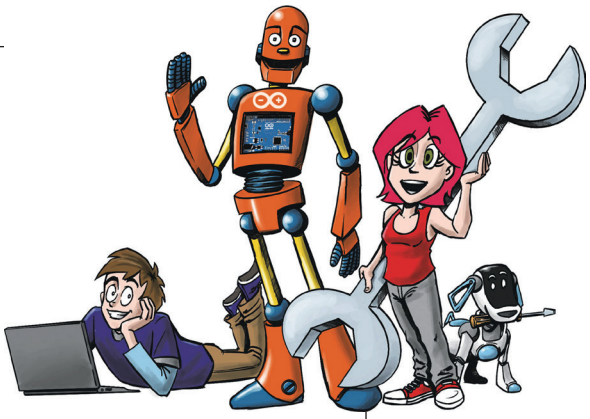


Содержание

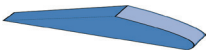
Введение 9

Что такое микросхема? 9
Как пишутся программы? 10
Материалы 11
Начало работы 11
Заключение 16
Несколько заданий 17

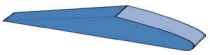
Мигай, мигай, огонек 19

Установка программного обеспечения 19
Наша первая программа 22
Наша вторая программа: гирлянда со светодиодами 27
Наша третья программа: аппарат Морзе 36
Наша четвертая программа: игра «Горячий провод» 39
Заключение 48
Несколько вопросов 49
...и несколько заданий 49

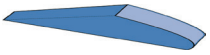
1



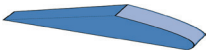
Arduino говорит 51
Отправка первого текста 51
Заключение 59
Вопрос 60
...и задание на сегодня 60



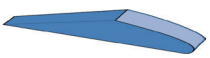
Сенсоры – интерфейсы для мира 61
Что такое датчик? 62
Включить светодиоды 63
Заключение 72
Несколько вопросов 73
...и несколько заданий 73



Моторы – движение с Arduino 75
Двигатель постоянного тока – веселое вращение 76
Эффективно управлять двигателем 79
Сервоприводы 81
Заключение 86
Несколько вопросов 86
...и несколько заданий 86



Чтение исходного кода других разработчиков 87
Документация 87
Загадочный исходный код 88
Заключение 91
Несколько вопросов 91
...и задание 91



ЖК-дисплей – отображение данных на самом Arduino 93
Что такое ЖК-дисплей? 94
Заключение 98

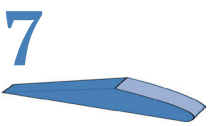
Несколько вопросов..... 98
...и задание..... 98

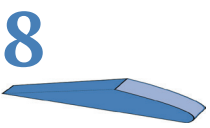
Arduino и мультиметр..... 99
Для начала история из жизни..... 99
Какие сведения нам нужны?..... 100
Изучаем потенциометр..... 105
Заключение..... 108
Несколько вопросов..... 108
...и несколько заданий..... 108

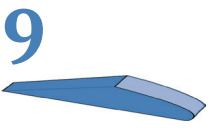
Arduino online..... 109
HTML – ворота в интернет..... 109
«Сеть, нам нужна сеть»..... 111
Заключение..... 114
Несколько вопросов..... 115
...и несколько заданий..... 115

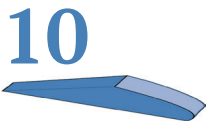
Клавиатура с Arduino Leonardo..... 117
Первые шаги с Leonardo..... 118
Первая маленькая клавиатура..... 120
Ключ обеспечения секретности..... 124
Заключение..... 126
Несколько вопросов..... 126
...и несколько заданий..... 127

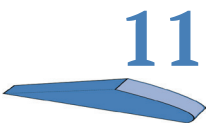
Взгляд за пределы IDE..... 129
C++, сердце Arduino..... 129
Перенос программы на Arduino..... 134
Программирование AVR..... 137
Заключение..... 142
Несколько вопросов..... 142
...и несколько заданий..... 142





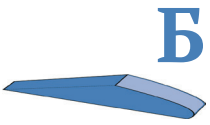




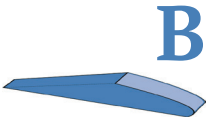


Не забудь меня – использование EEPROM 145
Общая информация о EEPROM 145
Что можно запрограммировать в EEPROM? 147
Проект: черный ящик 147
Заключение 156
Несколько вопросов 156
…и несколько заданий 156

Установка IDE 157
Установка 157

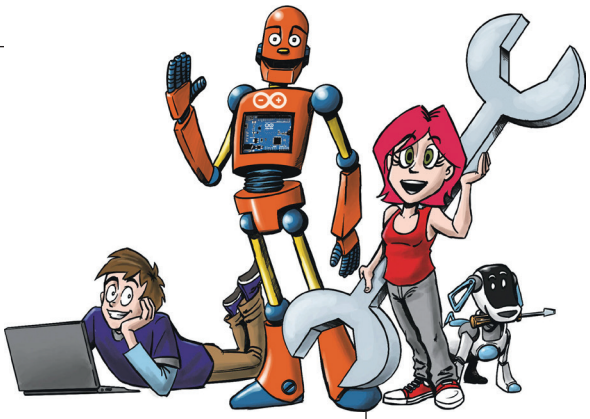


Ответы 159



Список материалов 165

Предметный указатель 167



Введение

Ты уже давно мечтаешь начать программировать или разобраться, из каких деталей состоит компьютер? После прочтения этой книги ты вряд ли сможешь самостоятельно собрать компьютер, но некоторые вещи будут тебе по силам.

Во введении рассказывается:

- что такое микросхема, микроконтроллер и Arduino;
- как написать программу;
- какие материалы тебе понадобятся для этой книги.

В процессе чтения ты легко освоишь основы электроники!

Что такое микросхема?

Наверняка у тебя уже есть представление о том, что из себя представляет микросхема. По-английски их называют микрочипами (*microchip*) или просто чипами (*chips*), и я дальше иногда тоже буду их так называть. Обычно это маленький черный квадратик, расположенный на плате, например на системной плате компьютера. Микросхема, которую ты будешь программировать, выглядит немного иначе. Она прямоугольная, а не квадратная, и у нее намного меньше выводов (металлических ножек по краям чипа). Кроме того, эти выводы крупнее, чем у обычных микросхем, которые ты мог встречать раньше.

Что такое микроконтроллер?

Микроконтроллер – это микросхема, которая уже содержит в себе все необходимые элементы (комплектующие). Если

провести аналогию, то системная плата компьютера – это микроконтроллер, а оперативная память на ней – одно из комплектующих. Точно так же оперативная память содержится внутри нашего микроконтроллера. Микроконтроллеры часто называют *computer-on-chip* – «однокристальный компьютер» по-русски.

Что такое Arduino?

Чтобы тебе было легче освоить программирование микроконтроллера, существует так называемый проект Arduino. В нем есть готовые платы с микроконтроллером и собственным программным обеспечением для создания программ. Как ясно из заголовка книги, здесь пойдет речь о программировании микросхем на плате Arduino.

Проект Arduino предоставляет готовую плату с подходящей средой разработки программ на персональном компьютере (ПК). Раньше разработчикам приходилось самим мастерить платы, чтобы иметь возможность научиться их программировать.

Как пишутся программы?

К сожалению, программирование осуществляется не голосом, а с помощью текста, набираемого на компьютере. Этот текст пишется не на русском языке, а с помощью специальных знаков и нескольких английских слов. Но пусть это тебя не пугает. Ты будешь учиться программировать с помощью довольно сложного языка программирования C++. Чтобы новичкам было легче им овладеть, создатели Arduino разработали упрощенный *диалект* (то есть вариант) этого языка программирования. C++ основан на нескольких словах и множестве знаков (символов), которые выглядят очень загадочно. В скобках указывается, что означает такой знак, как, например, ++ (приращение) или % (деление по модулю).

Со временем ты запомнишь, что значит каждый из этих символов, и сможешь безошибочно их использовать. В программном коде ниже показано, что можно сделать с помощью C++. Я написал эту программу для террариумного регулятора температуры, создание которого находится пока на начальной стадии:

```
#include "dimmen.h"
#include "kern_temperatur.h"
#include "terra_temperatur_class.h"
```

```
#include "class_cool.h"
void setup() {
 pinMode(13, OUTPUT);   pinMode(12,OUTPUT);
 Serial.begin(9600);   ADMUX = 0xC8;    delay(10);
}

void loop() {
 Cooler cooler(13); delay(100); bool hot = kern_temp(17);
 if (hot) cooler.start();
 else{
 cooler.stop();
 }
 delay(500); }
```

Также для программирования тебе понадобится программное обеспечение для ПК, которое можно найти на сайте arduino.cc. Подробную информацию по установке смотри в приложении А или в следующей главе.

Рассмотрим теперь устройство, которое нам необходимо для этой книги. Наверно, перед тобой сейчас лежит Arduino Uno – самая простая и удобная для изучения плата Arduino. Эта плата подключается к компьютеру через кабель USB. При составлении схемы плата должна быть отключена от кабеля USB, подключать кабель можно только тогда, когда схема готова. Пока мы не будем плату программировать, лишь использовать в качестве источника питания для схем.

Материалы

Чтобы продолжить, нам потребуются некоторые материалы. Это светодиод (деталь, которая излучает свет), разноцветные проводники-перемычки (черный и красный проводники обычно применяются для подключения питания), плата Arduino, резистор и так называемая *макетная плата*, на которой это все соединяется.

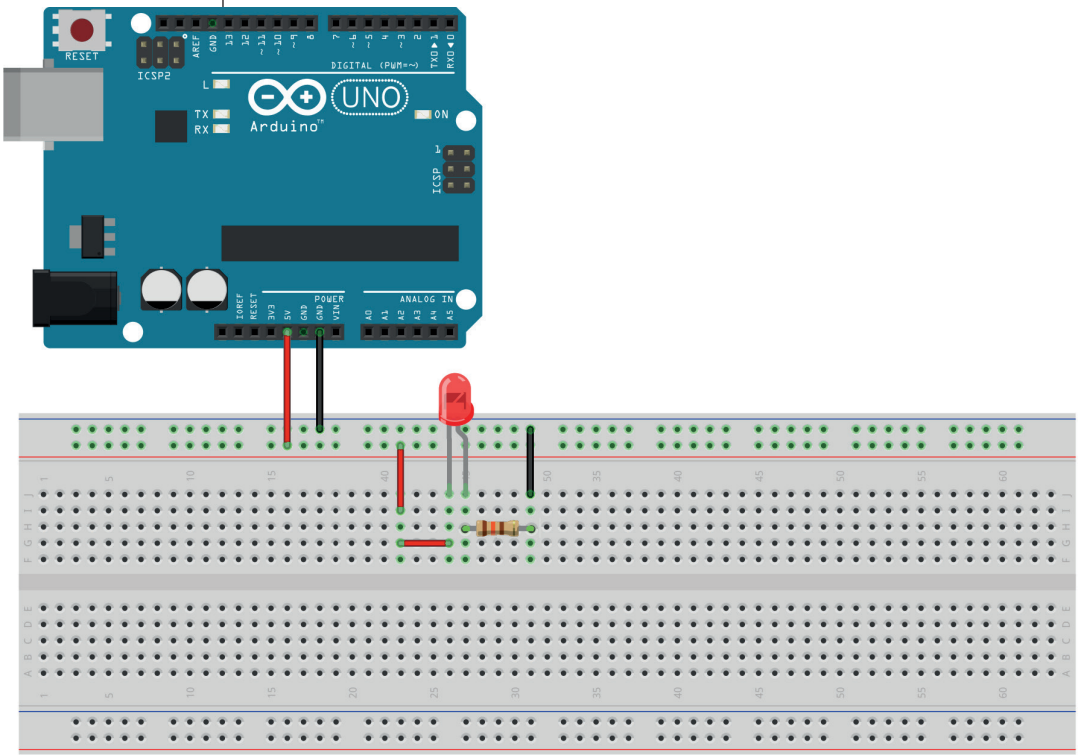
Все необходимые материалы перечислены в приложении Б в конце этой книги.

Начало работы

Попробуем выполнить несложную задачу с Arduino. Предлагаю сначала просто зажечь светодиод. Это можно сделать двумя способами: используя в качестве источника тока либо Arduino, либо батарею. Мы будем пользоваться

первым способом, применяя, как говорили, здесь плату Arduino только как источник питания.

Для знакомства со светодиодом построй электрическую схему по первому рисунку в этой книге. При этом обрати внимание на следующее: красный проводник подключается к выводу Arduino с надписью VCC, а черный – к выводу с надписью GND, то есть к плюсу и минусу питания. И наконец: красный проводник (положительный полюс) присоединяется к длинному выводу светодиода, а черный (отрицательный полюс) – к резистору и, через него, к короткому выводу светодиода.



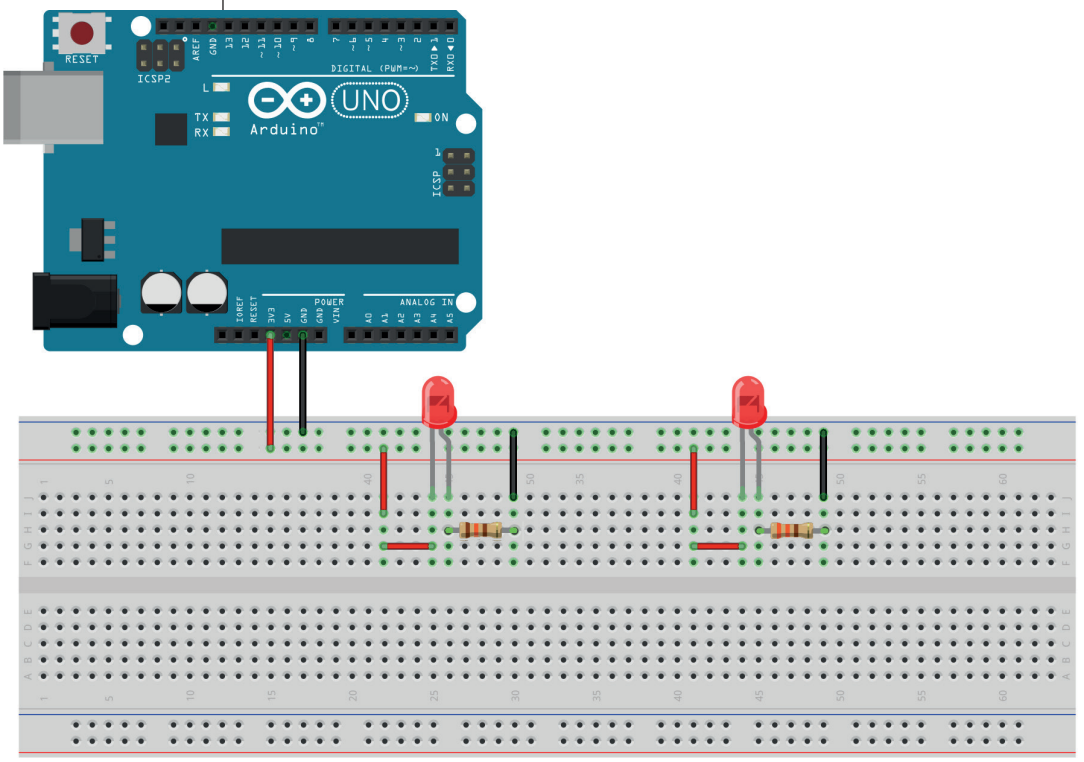
Таким образом, у тебя есть уже четыре важные детали для схемы подключения светодиода, которая будет часто встречаться в книге: Arduino, макетная плата, сам светодиод с резистором и, наконец, соединительные проводники (перемычки). И хотя последняя деталь самая простая, без нее ты не сможешь построить ни одну схему. Только при пайке плат можно обойтись без перемычек, но и здесь есть исключения, когда они бывают необходимы.

Резистор

К сожалению, подключать светодиоды напрямую к источнику питания, подобно лампочке, нельзя. Из-за большой силы тока они бы просто сгорели. Поэтому необходима еще одна небольшая деталь – *резистор*. Он служит для того, чтобы ослабить ток и предотвратить перегрев чувствительного компонента. Резистор выглядит как очень маленькая трубка с длинными блестящими выводами по концам, на которую нанесено несколько цветных полосок. От многих других компонентов резисторы отличаются тем, что не важно, в каком направлении они подключены относительно плюса и минуса источника питания.

Резисторы стоят относительно недорого (около 2–3 рублей за штуку), так что на 100 рублей у тебя выйдет не менее 30–50 штук. Они бывают разной величины сопротивления (номинала), которое измеряется в омах.

В примере, который мы собрали выше, необходим резистор сопротивлением 130 Ом. Оставим один светодиод подключенным, как было ранее, а второй подключим через резистор с другим сопротивлением – 330 Ом.



В этом примере левый светодиод горит ярче, чем правый, то есть чем меньше сопротивление резистора, тем яркость светодиода больше. Кстати, не важно, с какой стороны к светодиоду подключается резистор – их можно было бы включить между длинным концом и красным проводом.

А теперь решающий вопрос: как определить значение сопротивления резистора, не используя измерительные приборы? Если присмотреться, можно заметить у каждого резистора несколько разноцветных колец, которые образуют свой цветовой код. С помощью таблицы ниже можно определить цветной код для нужной величины сопротивления резистора.

Если ты правильно держишь резистор, золотое или серебряное кольцо всегда должно располагаться справа.

Цвет	1-е кольцо	2-е кольцо	3-е кольцо (множитель)	4-е кольцо (крайнее справа – допуск)
Серебряное			0,01	10%
Золотое			0,1	5%
Черное	–	0	1	
Коричневое	1	1	10	
Красное	2	2	100	
Оранжевое	3	3	1 К = 1000	
Желтое	4	4	10К	
Зеленое	5	5	100К	
Синее	6	6	1М	
Фиолетовое	7	7	10М	
Серое	8	8	100М	
Белое	9	9	1Г	

Везде, где в таблице стоит буква «К», подразумеваются три нуля. Потому что К значит «кило», то есть тысяча (1К соответствует 1000, 10К соответствуют 10 000). М(ега) и Г(ига) значат 1 000 000 и 1 000 000 000.

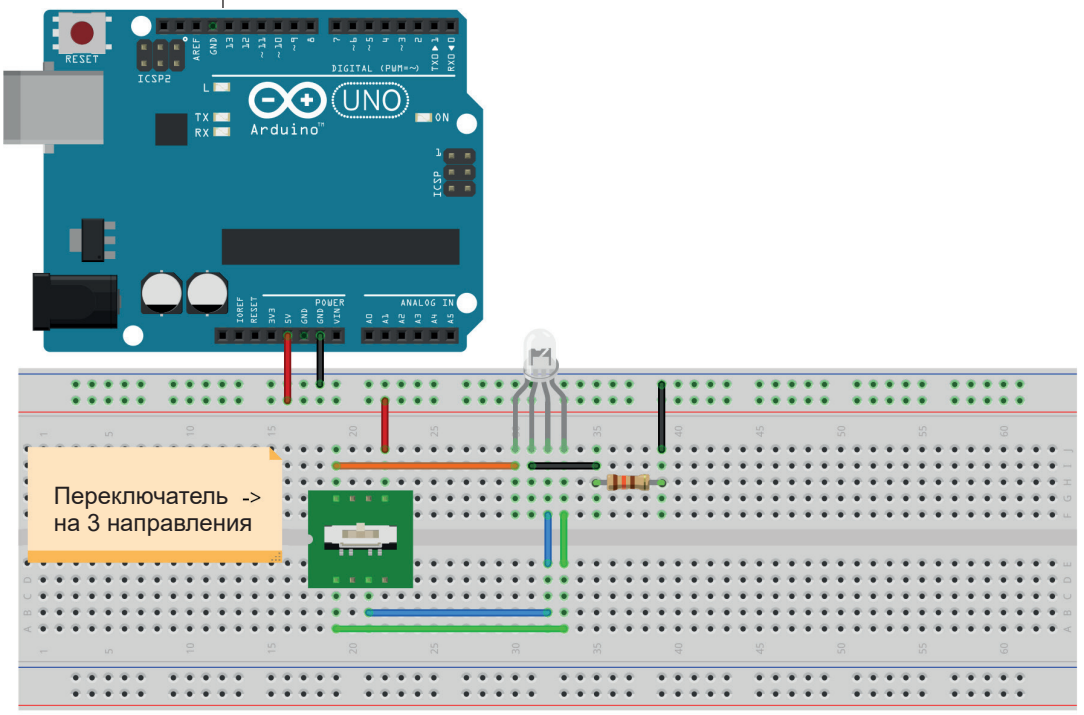
Золотое или серебряное кольцо обозначает точность (допуск). Допуск показывает, на сколько может отклоняться значение сопротивления. Чаще всего встречаются резисторы с допуском 5 %, у которого кольцо золотого цвета.

Расчет примера:

Если нам нужен резистор на 130 Ом, у него будут следующие цвета: 1-е кольцо = коричневый, 2-е кольцо = оранжевый и 3-е кольцо = коричневый, то есть 13 (1-е и 2-е кольца) × 10 (множитель) равно (=) 130 (Ом).

Схема с разноцветным светодиодом

Построй следующую схему.



Это схема со светодиодом, который может гореть тремя разными цветами (как правило, красным, зеленым и синим). В этой схеме тебе понадобится резистор на 130 Ом, так как мы подключаем схему к Arduino в качестве источника питания (вывод VCC ⇒ красный, вывод GND ⇒ черный). Если двигать ползунок размещенного на схеме переключателя на три направления, то светодиод будет последовательно загораться одним из трех цветов.

Здесь есть один общий отрицательный полюс и три положительных полюса для каждого цвета светодиода.

Заключение

Теперь ты знаешь…

- как подключать светодиоды и что такое резистор;
- как подключать разноцветные светодиоды.

И ты знаешь...

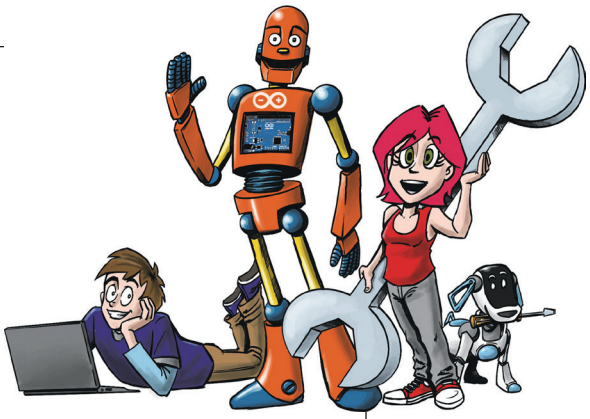
- как сделать так, чтобы светодиод сгорел.

Несколько заданий

Чтобы закрепить знания, выполни следующие задания.

1. Подключи три светодиода в ряд (+ 3 резистора на 330 Ом каждый).
2. Если ты знаешь, что такое мостовая схема включения диодов (обычных выпрямительных), то используй ее, чтобы зажечь светодиод, независимо от того, как он подключен к полюсу.
3. Если ты не справился с предыдущим заданием, изучи мостовой выпрямитель.

Ты познакомился с введением к этой книге. В следующих главах мы будем строить и применять гораздо более интересные схемы. Вперед, к новым открытиям!



1
Мигай, мигай, огонек

В этой главе ты узнаешь, как запрограммировать Arduino. Кроме того, ты научишься использовать кнопки и узнаешь о возможностях их применения вместе с Arduino.

Мы разберем следующие вопросы:

- установка программного обеспечения, подключение Arduino;
- как включить и выключить светодиоды; как остановить Arduino на время;
- чтение состояния кнопки (замкнута или разомкнута);
- как составить план собственного проекта.

В конце главы мы спланируем и запрограммируем небольшой проект: мигающая гирлянда с различными функциями. Во второй главе мы продолжим работу над гирляндой и сможем управлять ей и менять ее функции через компьютер.

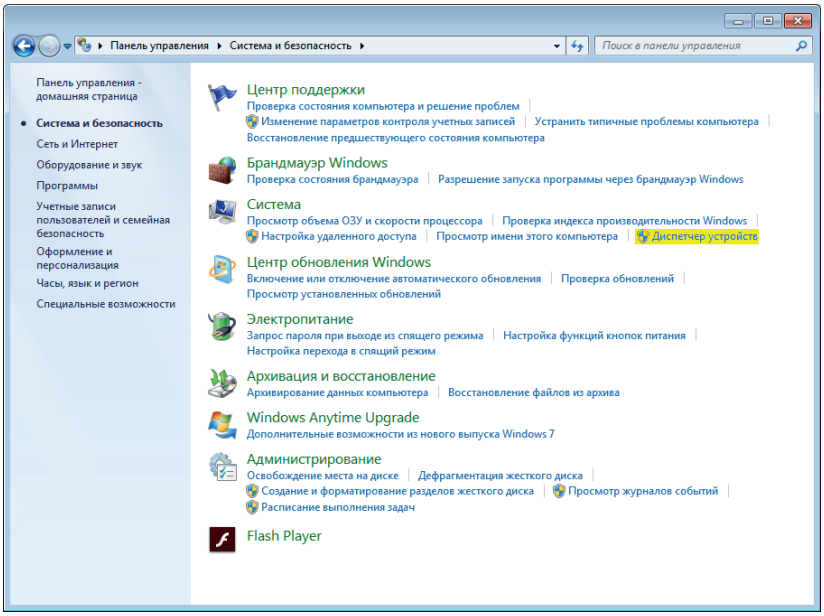
Установка программного обеспечения

Установка программного обеспечения не должна вызывать у тебя никаких затруднений. Сначала нужно скачать

программное обеспечение здесь: http://arduino.cc/en/Main/Software. На момент написания этой книги актуальная версия 1.8.5.

Для тех, кто пользуется Windows, на сайте имеется выбор: самоустанавливающийся EXE-архив или обычный архив в формате ZIP. Мы остановимся на втором варианте. Скачанный архив распакуй с помощью любого архиватора (например, Winrar или 7Zip, можно использовать и собственные возможности Windows). Содержащуюся там папку под названием **arduino** (маленькими буквами с добавлением номера версии) можно сохранить, например, на рабочем столе. В папке находится так называемая IDE, программа для создания исходного кода и его преобразования в «машинный язык». По сути, IDE представляет собой обычную программу для работы с текстом (текстовый редактор), которая различным цветом помечает команды программирования.

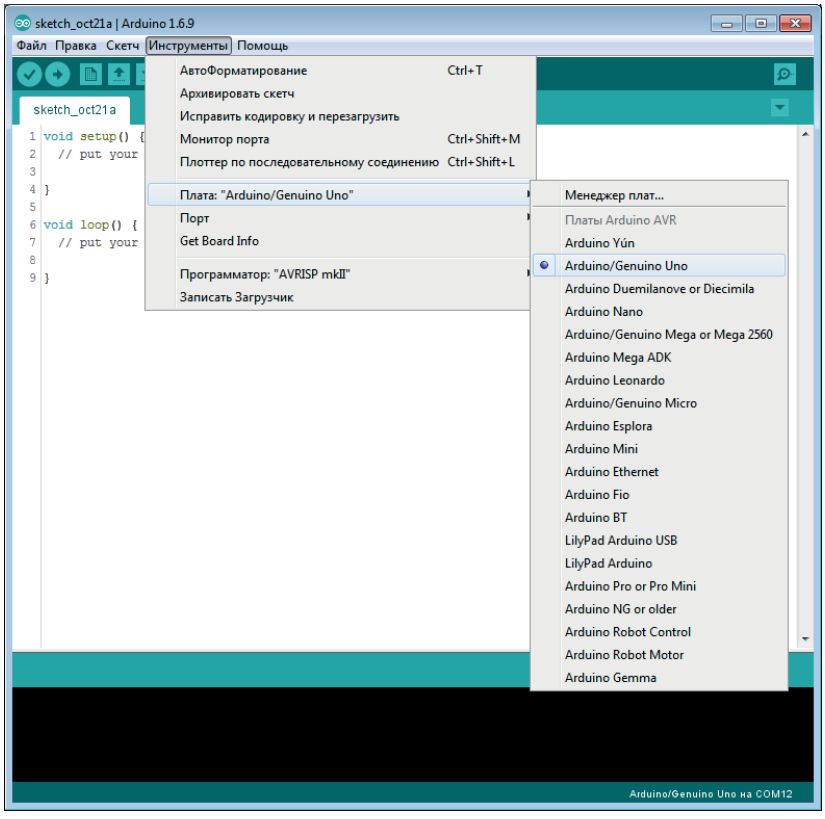
Теперь перейдем к самому сложному этапу – установке драйвера. Для операционной системы Windows нужно установить драйвер, который позволит опознавать подключенную плату Arduino и ее программировать. Подключи к USB-кабелю Arduino Uno и подожди. Через некоторое время должно появиться окошко, в котором предложат найти или обновить драйверы. Проигнорируй его и открой панель управления. Затем нажми на раздел **Система и безопасность**. Там выбери пункт меню **Система | Диспетчер устройств**.



В открывшемся окне нужно выбрать пункт **Порты (COM и LPT)**. Теперь там должно отображаться название платы Arduino.

Нажатием правой кнопки мыши на этом названии выбери **Обновить драйвер**, а в открывшемся окне – **Выполнить поиск драйверов на этом компьютере**. Теперь тебе нужно найти папку на рабочем столе, в которую ты распаковал IDE (в дальнейшем мы будем называть ее просто **Arduino**, хотя обычно она еще дополняется номером версии IDE). В подпапке **Drivers** (**...\Arduino\Drivers**) ты найдешь файл с названием *Arduino.inf*. Выбери его, и установка драйвера завершена. (Но честно, быстрее можно установить всю операционную систему Linux, чем один драйвер для Windows.)

Итак, драйвер установлен. Теперь ты можешь через USB-кабель подключить Arduino к компьютеру и приступить к программированию. Но сначала нам предстоит настроить в IDE нужную плату Arduino. Если ты используешь рекомендуемую здесь Arduino Uno, выбери ее в меню **Инструменты | Плата...**, см. рисунок:



Затем необходимо определить правильный порт: отключи плату Arduino Uno (если ты ее уже подключил), посмотри порты в **Инструменты | Последовательный порт...** и запиши их. Затем подключи Arduino Uno заново. Выбери порт, который появился. Теперь можно начать программировать.

Наша первая программа

Наша первая программа поможет проверить, правильно ли подключен Arduino.

```
void setup() {}
void loop() {}
```

Этот текст (исходный код) тебе нужно ввести в IDE и потом кликнуть на значок со стрелкой вправо в верхнем левом углу, под строкой меню. Если в строке над черным полем в нижней части IDE после всех преобразований появится надпись «Загрузка завершена», то Arduino настроен правильно, в противном случае фон окрасится оранжевым цветом с надписью «Проблема загрузки в плату...». Очень часто такая ошибка возникает, когда настроено все правильно, но плата или порт в IDE выбраны неверно, а также тогда, когда плату отключили от USB на время составления схемы и забыли включить обратно. В случае если все подключено, но ошибка все-таки возникает, повтори инструкции выше или см. приложение А.

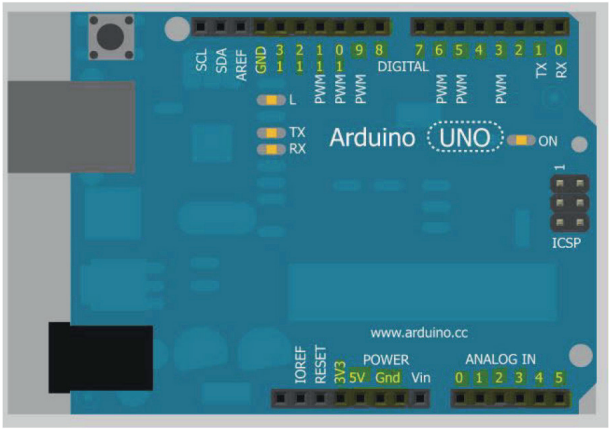
Программа для Arduino, так называемый *скетч* (sketch), всегда состоит, как минимум, из двух частей: «setup()» и «loop()». Код, который записан в части setup, исполняется один раз при запуске или при сбросе контроллера. Код в части loop, напротив, исполняется в бесконечном цикле. Исходный код, который ты пишешь сам после заголовков «setup()» и «loop()», заключается в фигурные скобки. Почему это так, ты узнаешь в следующей главе в разделе «Как работают функции».

Наша первая программа должна управлять светодиодом и включать его, а позднее он будет мигать.

```
void setup() {
pinMode(13,OUTPUT); //Пин 13 на выход
digitalWrite(13,HIGH); //Пин 13 включить
}
void loop() {}
```

Все, что стоит за двумя косыми линиями, в программе всегда расценивается как комментарий, то есть написанный за ними текст не имеет функции, а служит только для того, чтобы код было проще читать.

В Arduino все выводы микросхемы контроллера выведены на небольшие металлические контакты в черных гнездах по бокам платы Arduino Uno (на рисунке ниже отмечены желтым цветом). У микросхем выводы часто называют английским словом «пины», потому что там они действительно похожи на иголочки (*pin* – иголка). Выводы платы Arduino по традиции тоже так иногда называют, и, встретив его в этой книге, помните, что *пин* означает то же самое, что просто *вывод*. Каждым выводом-пином платы Arduino программа может управлять по отдельности.



Рядом с каждым выводом на плате написан номер, по которому ты можешь его настроить. В примере я использую пин 13, потому что на нем уже установлен светодиод на плате. Этот код тебе необходимо загрузить в Arduino, как показано в примере выше, и светодиод сразу загорится зеленым, красным или желтым цветом (цвет зависит от изготовителя платы).

В коде выше ты можешь увидеть, что команда `pinMode` является *функцией*. В скобках указываются *параметры* функции – сведения, которые необходимы для ее работы. Здесь первым параметром является номер вывода, вторым – состояние. Это может быть `OUTPUT` (выход) или `INPUT` (вход). Состояние «вход» нам понадобится далее в разделе «Чтение входов», а пока будем работать с состоянием «выход».

```
void setup() {
pinMode(13,OUTPUT); //Пин 13 на выходе
digitalWrite(13,HIGH); //Пин 13 включить
}
void loop() {}
```

Выходы всегда что-то переключают, например светодиоды, входы всегда что-то читают, например состояние кнопки (нажата или не нажата). Функция всегда заканчивается точкой с запятой (;), чтобы транслятор знал, где конец команды. Транслятор (что по-английски означает «переводчик», иначе его называют компилятором) переводит (транслирует или компилирует) исходный код на машинный язык, то есть в двоичную систему счисления, чтобы его понял микроконтроллер. (Двоичная система – это единицы и нули, с которыми работает компьютер и о которых ты, вероятно, уже слышал.)

Команда `pinMode()` необходима, чтобы Arduino знал, как использовать вывод – на выход или на вход. Вторая команда, `digitalWrite()`, включает состояние вывода. В `digitalWrite()` во втором параметре ты можешь указать либо `HIGH` («высокий уровень», близкий к питанию 5 вольт), тогда пин будет выдавать ток, либо `LOW` («низкий уровень», близкий к нулю), тогда пин будет принимать ток от внешнего источника.

Ты, наверное, уже заметил, что в информатике используется много английских слов. Английский язык играет важную роль в программировании. И специальные термины Arduino, например `HIGH` (здесь в значении «высокий уровень»), и большинство языков программирования, а также технических описаний компонентов схем «составлено» на английском. Поэтому занятия английским для программиста и электронщика будут совсем не лишними.

Как сделать так, чтобы светодиод мигал?

Ты можешь записать в цикл loop функцию `digitalWrite()` с чередующимися `HIGH` и `LOW`, но это не будет работать. Вот попробуй с этим кодом:

```
void setup() {
pinMode(13,OUTPUT); //Только один раз, поскольку мы хотим
                    //использовать пин 13 только как выход
}
void loop() {
digitalWrite(13,HIGH);
digitalWrite(13,LOW);
}
```

Сам по себе код должен работать, и он работает! Но поскольку контроллер функционирует с частотой 16 миллионов герц (герц означает одно колебание в секунду; если команда длится один такт, вывод переключается 16 миллионов раз в секунду, соответственно, состояние вывода меняется каждые 0,0000000625 секунды), человеческий глаз не способен увидеть реакцию светодиода. Необходимо немного увеличить интервал. Для этого используется команда `delay()` (*delay* – задержка). Эта команда приостанавливает контроллер на заданное количество миллисекунд (1000 мс = 1 секунда). Чтобы светодиод менял состояние один раз в секунду, нам понадобится следующий код:

```
void setup() { pinMode(13,OUTPUT);
}
void loop() {
digitalWrite(13,HIGH); //Включить светодиод
delay(1000); //Подождать 1 секунду
digitalWrite(13,LOW); //Выключить светодиод
delay(1000); //И только через 1 с снова включить
             //и цикл начинается сначала
}
```

Итак, твой первый проект – мигающий светодиод – выполнен. Можешь сохранить его через **Файл | Сохранить...** под любым именем. Arduino IDE сохраняет все файлы проектов с исходным кодом в одном месте, так что ты сможешь использовать их позже.

Что такое переменные и для чего они нужны

Итак, у тебя есть первый «работающий» код, который обеспечивает цикличное управление выводами. Наверняка тебе интересно, как можно упорядочить, например, 50 выводов (в Arduino Mega их более 50 штук)? В Arduino есть возможность дать каждому выводу свое имя. Для этого есть разные способы, здесь мы будем присваивать имена с помощью *переменных*. Содержание переменной может меняться, отсюда и название. Ее противоположность – *константа*, которая никогда не меняется. В переменных можно установить номера выводов, которыми мы хотим управлять. Для этого нужна числовая переменная, так называемая *целочисленная переменная* (integer, `int`). Предыдущий пример с переменными выглядит так:

```
int led = 13; //Определить переменные
int pause = 1000;
void setup() {pinMode(led,OUTPUT);
```

```
}
void loop() {
digitalWrite(led,HIGH); //Пин, который установлен для светодиода,
                        //переключается
delay(pause); //Подождать количество мс, которое установлено в паузе
digitalWrite(led,LOW);
delay(pause);
}
```

Если компилятор встречает выражение вида `тип_переменной имя_переменной = показатель;`, переменная инициализируется. В таблице приведены типы переменных, которые мы будем использовать.

Тип переменной	Содержание	Пример
int	Целое число от –32 768 до 32 767	9
float	Число с плавающей запятой	9.4
byte	Целое число от 0 до 255	50
char	Буква (символ)	'e'
bool (или boolean)	Булево значение, другими словами – «правда» (true) или «ложь» (false)	true/false или 1/0
long	Целое число от –2 147 483 648 до 2 147 483 647	262000

Важно, что имена переменных не могут начинаться с цифры и не содержат пробела (вместо него можно использовать символ подчеркивания, например `langer_variablenname`). Помни, что переменная во всем тексте программы пишется одинаково, с учетом прописных и строчных букв (led, LED и Led – это три разные переменные!). Буквы в именах употребляются только из английского алфавита, из других языков не допускается.

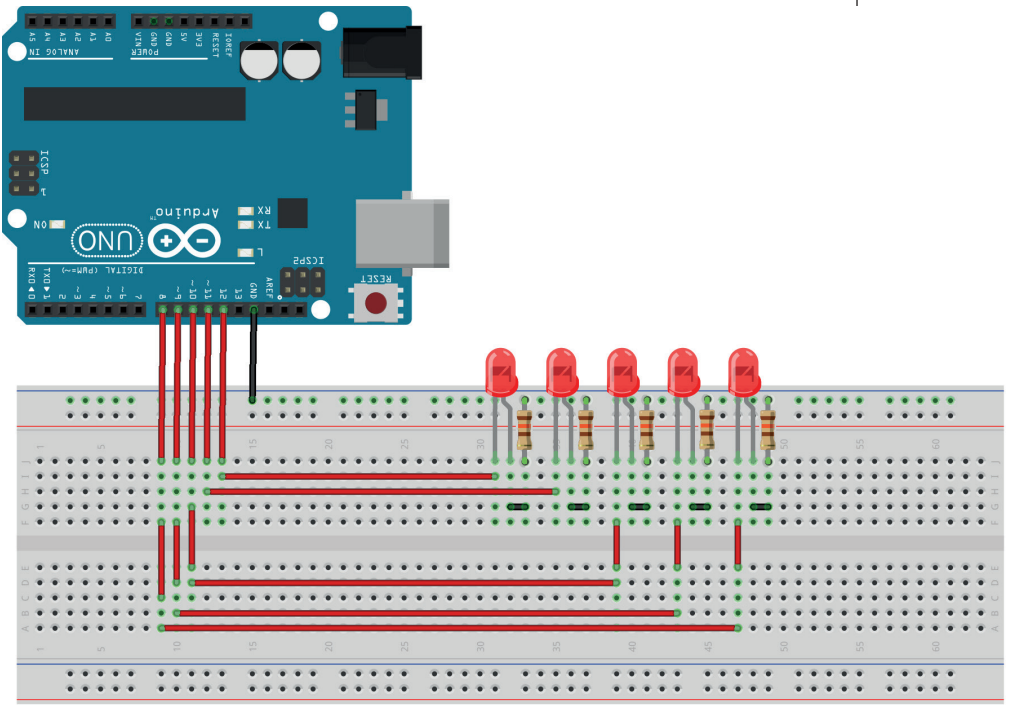
Для отдельных переменных необходимо помнить следующие моменты. Переменные всегда имеют определенный диапазон значений, который нельзя превышать (например, от 0 до 255 у типа `byte`). Числа с плавающей запятой (в школе их называют вещественными, или действительными, числами) должны быть написаны с разделительной точкой (.) перед дробной частью, поскольку так принято в Америке (например, 3,14 должно выглядеть как 3.14). В переменной `char` буква всегда должна стоять в верхних одинарных кавычках (‘A’), а в `bool` ты можешь использовать только указанные в таблице значения `true` или `false`. В следующей главе ты узнаешь, для чего используется `bool` (`boolean`).

Как упорядочить переменные? Каждый микроконтроллер содержит определенное количество RAM-памяти, то есть оперативной памяти

микросхемы. Ее можно разделить на небольшие единицы-ячейки. В зависимости от размера переменной резервируется определенное количество ячеек памяти, и в них записывается конкретное число. Поэтому необходимо всегда иметь достаточный объем свободной оперативной памяти, потому что в противном случае программа может сработать неправильно. Небольшое утешение: все представленные в книге программы без проблем работают на Arduino Uno. Некоторые скетчи из-за собственных расширений могут заполнить всю память, но на это будет отдельное указание в тексте.

Наша вторая программа: гирлянда со светодиодами

Теперь давай сделаем с помощью переменных кое-что полезное, а именно небольшую гирлянду со светодиодами. Для этого необходимо минимум пять светодиодов, чтобы можно было что-то заметить. Я предлагаю использовать выводы с 12-го по 8-й. Здесь решающее значение имеет соединение перемычками. Если у тебя макетная плата, можно сделать так:



Все это довольно просто: ты соединяешь вывод 8 с крайним слева светодиодом и поочередно следующий вывод со следующим светодиодом. Таким образом, у нас уже готово соединение, и сейчас мы проведем следующий эксперимент.

Сначала давай разработаем цикл, который будет заставлять мигать один светодиод и затем переходить к следующему и т. д. В исходном коде для этого нет ничего нового, и он относительно прост:

```
int pin1 = 8;
int pin2 = 9;
int pin3 = 10;
int pin4 = 11;
int pin5 = 12;

void setup() {
pinMode(pin1,OUTPUT)
pinMode(pin2,OUTPUT)
pinMode(pin3,OUTPUT)
pinMode(pin4,OUTPUT)
pinMode(pin5,OUTPUT) }
void loop() {
//Заставь Pin 1 мигать
digitalWrite(pin1,HIGH);
delay(100);
digitalWrite(pin1,LOW);
delay(100);
//Pin 2
digitalWrite(pin2,HIGH);
delay(100);
digitalWrite(pin2,LOW);
delay(100);
//Pin 3
digitalWrite(pin3,HIGH);
delay(100);
digitalWrite(pin3,LOW);
delay(100);
//Pin 4
digitalWrite(pin4,HIGH);
delay(100);
digitalWrite(pin4,LOW);
delay(100);
//Pin 5
digitalWrite(pin5,HIGH);
delay(100);
digitalWrite(pin5,LOW);
delay(100);
}
```

Как работают функции

Вот видишь, ничего нового, но, пожалуй, это слишком длинный код для такого простого эффекта. Как насчет того, чтобы его укоротить? Это можно сделать с помощью функций. Термин уже упоминался выше, а теперь мы самостоятельно напишем функцию. В нашем случае функция – это просто выполнение нескольких команд (по сути, тоже функций, например `digitalWrite()`), собранных в одном вызове. С помощью функций можно упорядочить код (кстати, `setup()` и `loop()` – тоже функции). Такой метод – техника так называемого *структурного программирования*. Также сюда относятся циклы и переходы на метку, причем в этой книге я переходы не использую, потому что на сегодняшний день они являются устаревшими и вредят коду (такой код еще называют «спагетти-код» – без шуток, это реальный термин!).

Вернемся к функциям. Рассмотрим одну функцию, которую ты уже знаешь: `void digitalWrite(int,int)`. Так можно коротко обобщить функции, их названия, а также типы передаваемых в нее параметров (в нашем случае два целых числа типа `int`). Перед функцией идет тип возвращаемого значения, в нашем случае `void` (в переводе «пустота» – функция ничего не возвращает, но указать это все равно требуется).

Таким образом, функцию можно переписать так:

```
Тип_возвращаемого значения Название_функции(параметр){код_функции;}
```

Вот функция для сокращения нашего длинного кода:

```
void blink(int pin,int msek) {
digitalWrite(pin,HIGH);
delay(msek);
digitalWrite(pin,LOW);
delay(msek);
}
```

Это наша первая функция. В круглых скобках указываются параметры, то есть какой вывод должен быть включен и как долго. В фигурных скобках идет код, который относится к функции.

> После названия функции и фигурных скобок точка с запятой не ставится.