Проект 0

Что такое Arduino?

00

DFROBOT
DRIVE THE FUTURE

Что такое Arduino?

Arduino - это платформа для создания прототипов электроники с открытым исходным кодом, основанная на гибком, простом в использовании аппаратном и программном обеспечении. Она предназначена для художников, дизайнеров, любителей и всех тех, кто заинтересован в создании интерактивных объектов или исследовании окружающей среды.
Arduino может взаимодействовать с окружающей средой, получая входящие сигналы от сенсоров, и на основе полученных данных контролировать свет, двигатели или другие исполнительные механизмы. Интегрированная среда разработки Arduino IDE представляет собой кросс-платформенное приложение, написанное на Java, которое состоит из среды разработки для языка программирования Processing/Wiring и кода библиотек. Arduino IDE может работать независимо и взаимодействовать с другим программным обеспечением, таким как Flash, Processing, MaxMSP и т. д. Arduino IDE является открытым исходным кодом, поэтому вы можете бесплатно скачать и поделиться большим количеством интерактивных проектов!

Далее представлен список нескольких проектов, которые дадут вам понимание о возможностях платы Arduino:

// сделать звуковое уведомление о готовности кофе
// email уведомление через мобильный телефон
// мигающая пушистая плюшевая игрушка
// кресло-коляска профессора Ксавье с функцией распознавания голоса и подачи напитков
// оружие из Звездных войн
// монитор с измерением пульса при езде на велосипеде
// робот рисующий картины на снегу

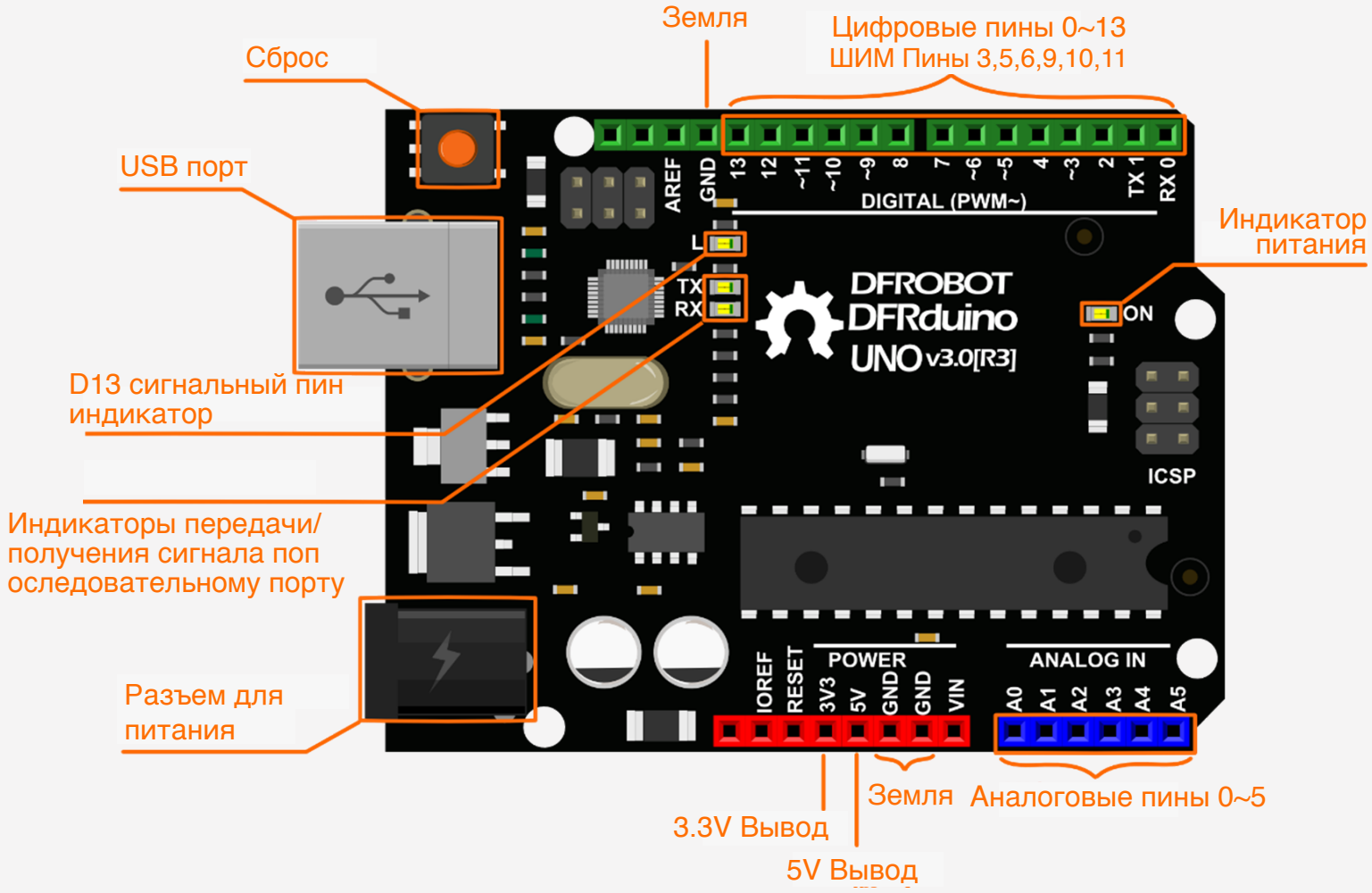
История Arduino

В 2005 году проект Arduino был создан для студентов Института проектирования взаимодействий в итальянском городе Ивреа. На тот момент, для своих проектов, студенты-программисты использовали микроконтроллеры «BASIC Stamp». Они стоили 100$ и считались весьма дорогостоящими для студентов.
Массимо Банзи – один из основателей Ардуино, преподавал в Ивреа. Название «Arduino» происходит от бара в Ивреа, где встречались основатели проекта. Сам бар был назван в честь Ардуино – короля Италии с 1002 до 1014 года.
Колумбийский студент Эрнандо Барраган внес свой вклад в развитие аппаратного обеспечению дизайна сборки. После завершения монтажной платформы исследователи работали над тем, чтобы сделать ее более легкой, дешевой и доступной для
сторонников открытого программного обеспечения. В итоге школа закрылась, поэтому эти исследователи, включая Дэвида Куартели, продвинули эту идею. Она заключалась в том, чтобы создать Arduino, каким мы его знаем сегодня.

Arduino UNO

А сейчас, давайте, поближе познакомимся с микроконтроллером Arduino и найдем пины входа/выхода и встроенные светодиоды.

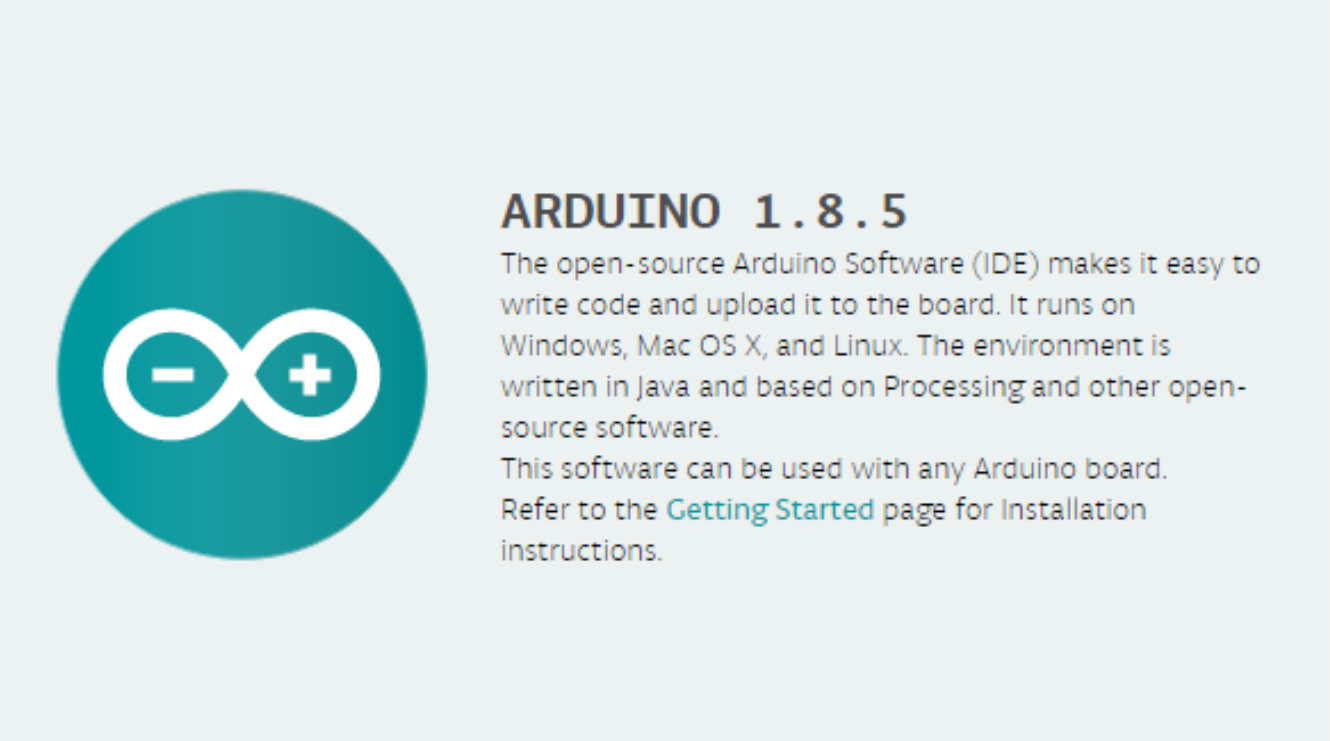
- Пины входа/выхода, цифровые пины 0-13, аналоговые пины 0-5
- Два варианта питания Arduino – от компьютера через USB-порт, либо от внешнего блока питания 6-12 В через разъем питания.
- 4 светодиода и кнопка сброса. L это встроенный светодиод, который соединяется с цифровым пином 13. TX и RX индикаторы передачи и получения сигнала. Когда мы загружаем скетч в Arduino, эти два светодиода



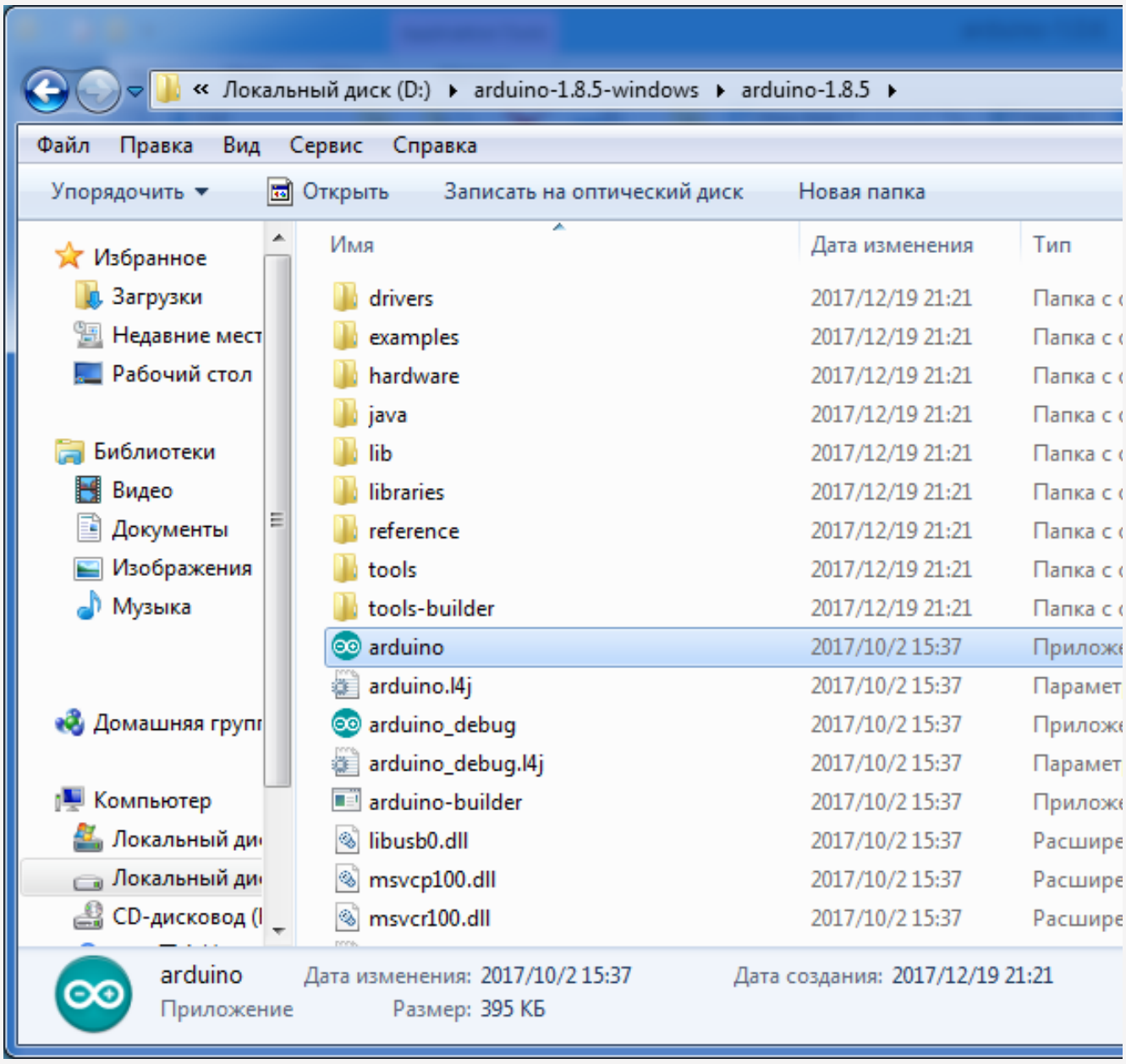
Первое использование

1. Загрузите Arduino IDE

Перейдите по ссылке:
http://arduino.cc/en/Main/Software
загрузите установочный файл в зависимости от вашей операцио нной системы

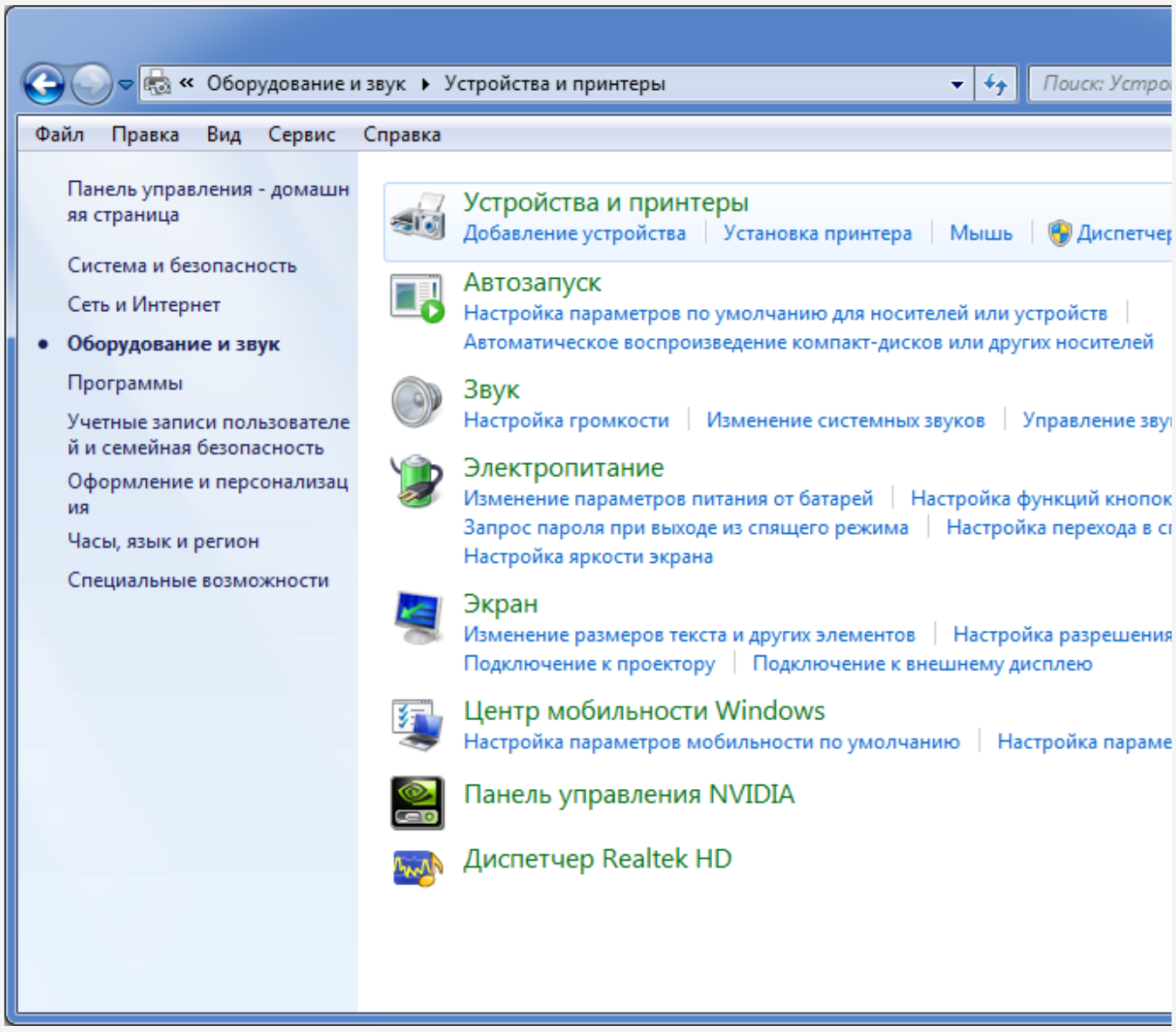


Для пользователей Windowsинструкция по установки подана ниже. Пользователи Mac и Linux, могут напрямую использовать скетч Arduino просто нажав на файл.

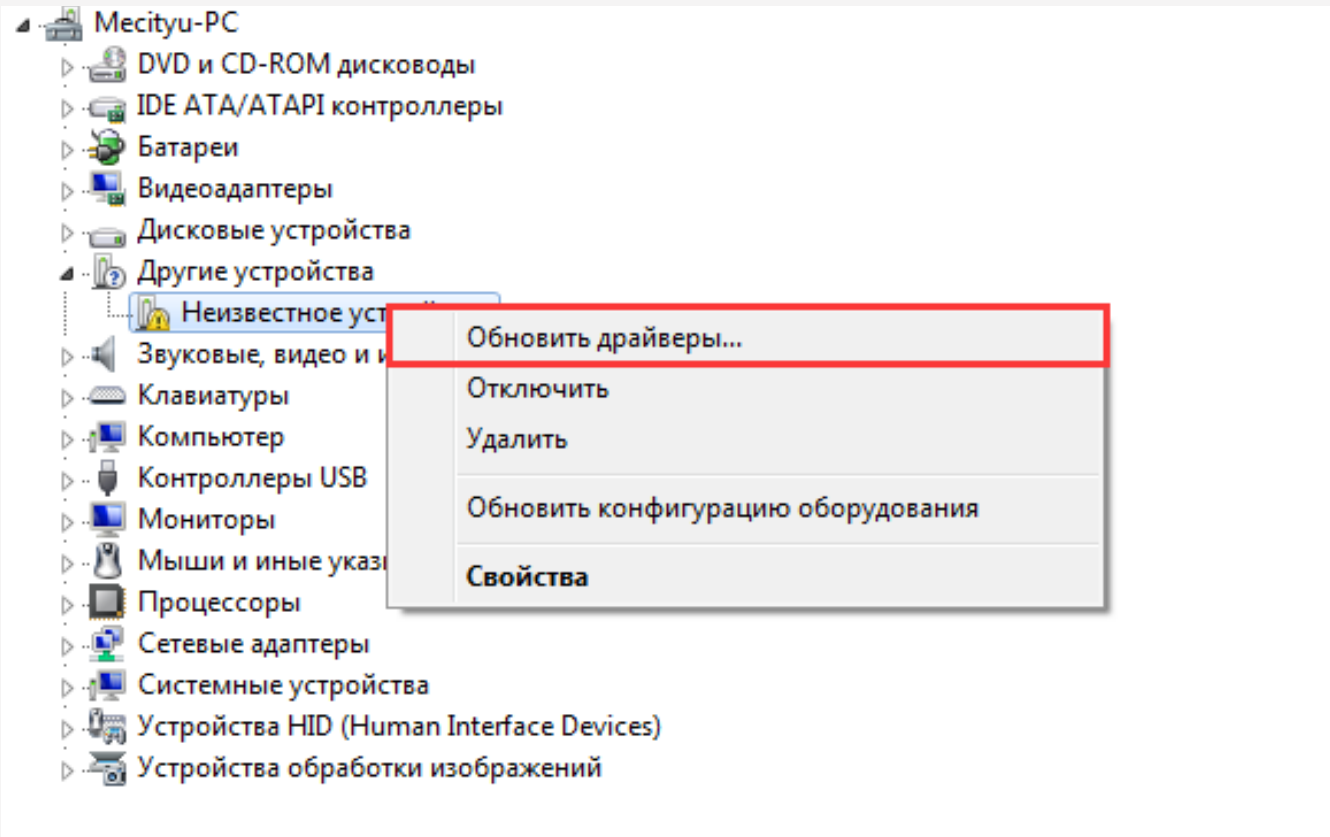


2.Установка драйверов

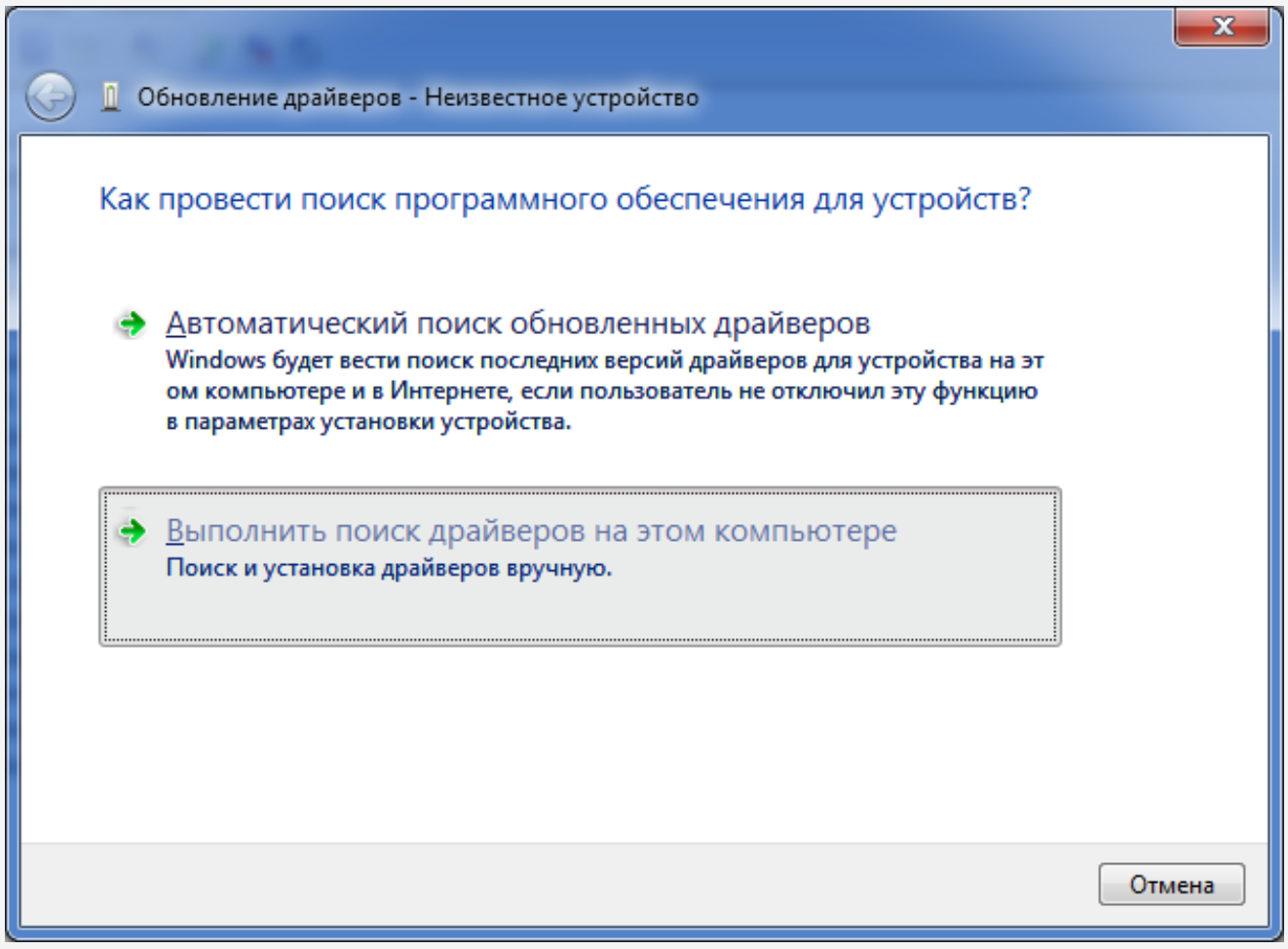
Устанавливая драйвера для Arduino Uno на Windows 7, Vista или на XP: Подключите плату и подождите, пока Windows начнет процесс установки драйвера.
Через несколько секунд, несмотря на все усилия, процесс потерпит неудачу, но не паникуйте! Нажмите на меню «Пуск» и откройте панель управления



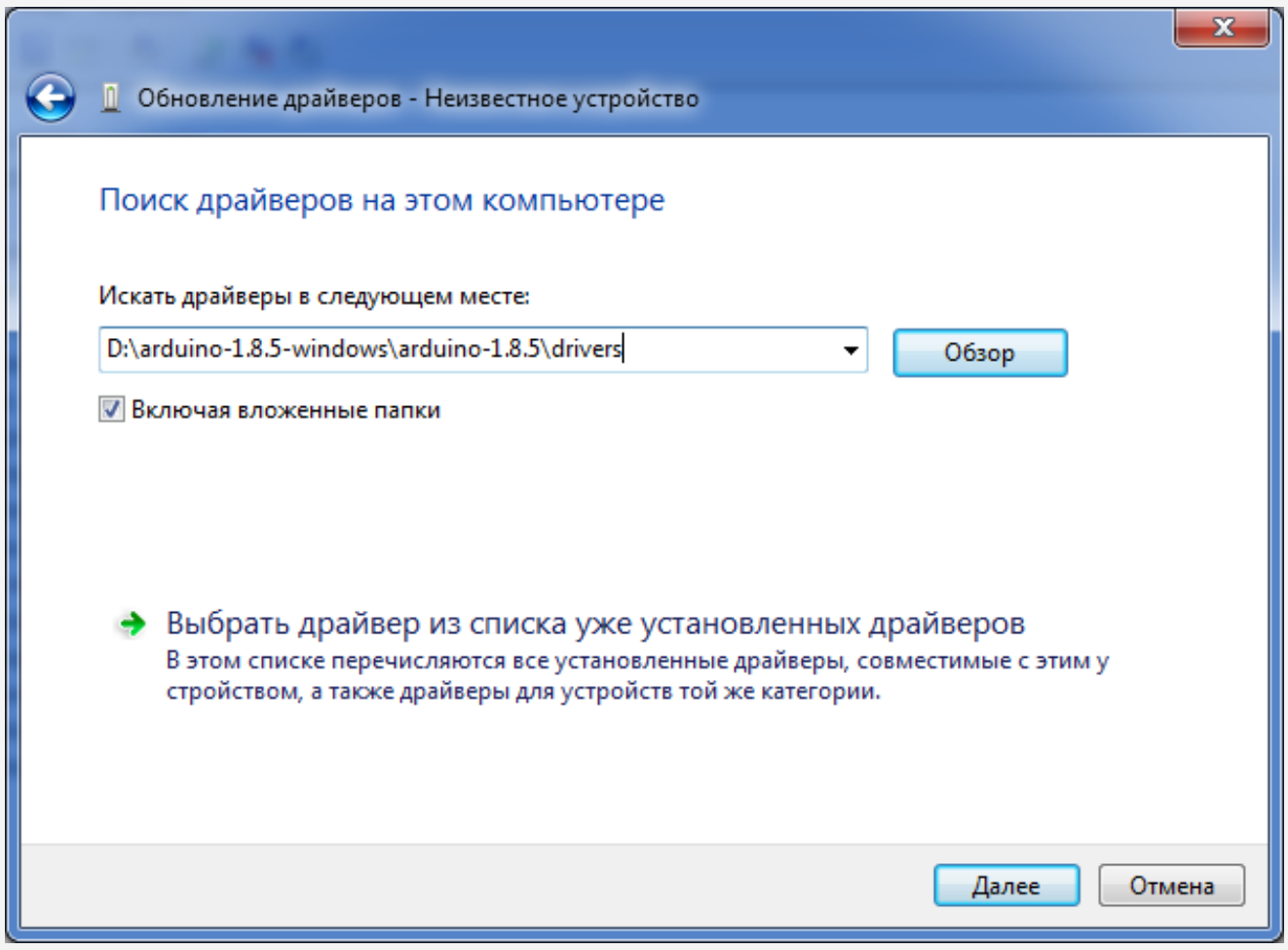
Найдите “Неизвестное устройство” (Unknown device) нажмите на него правой кнопкой мыши и выберите “Обновить драйвера” (Update Driver Software)



Выберите “Выполнить поиск драйверов на этом компьютере” (Browse my computer for driver software), чтобы найти драйвера вручную

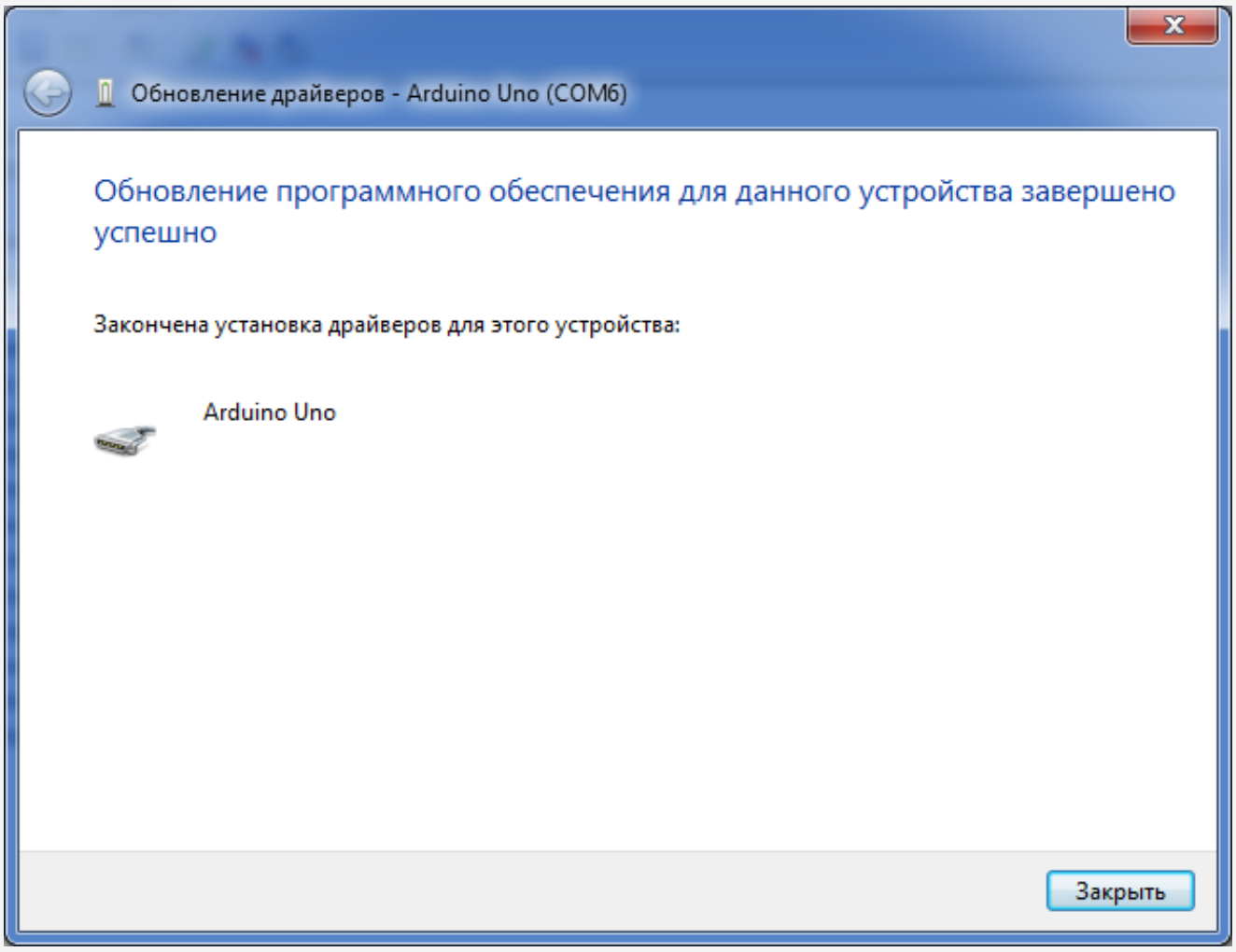


Нажмите “Обзор” (Browse) и укажите директорию, куда вы установили Arduino IDE, а точнее её дочернюю папку "drivers". Ставим галочку “Включая вложенные подпапки” и кликаем “Далее” (Next).



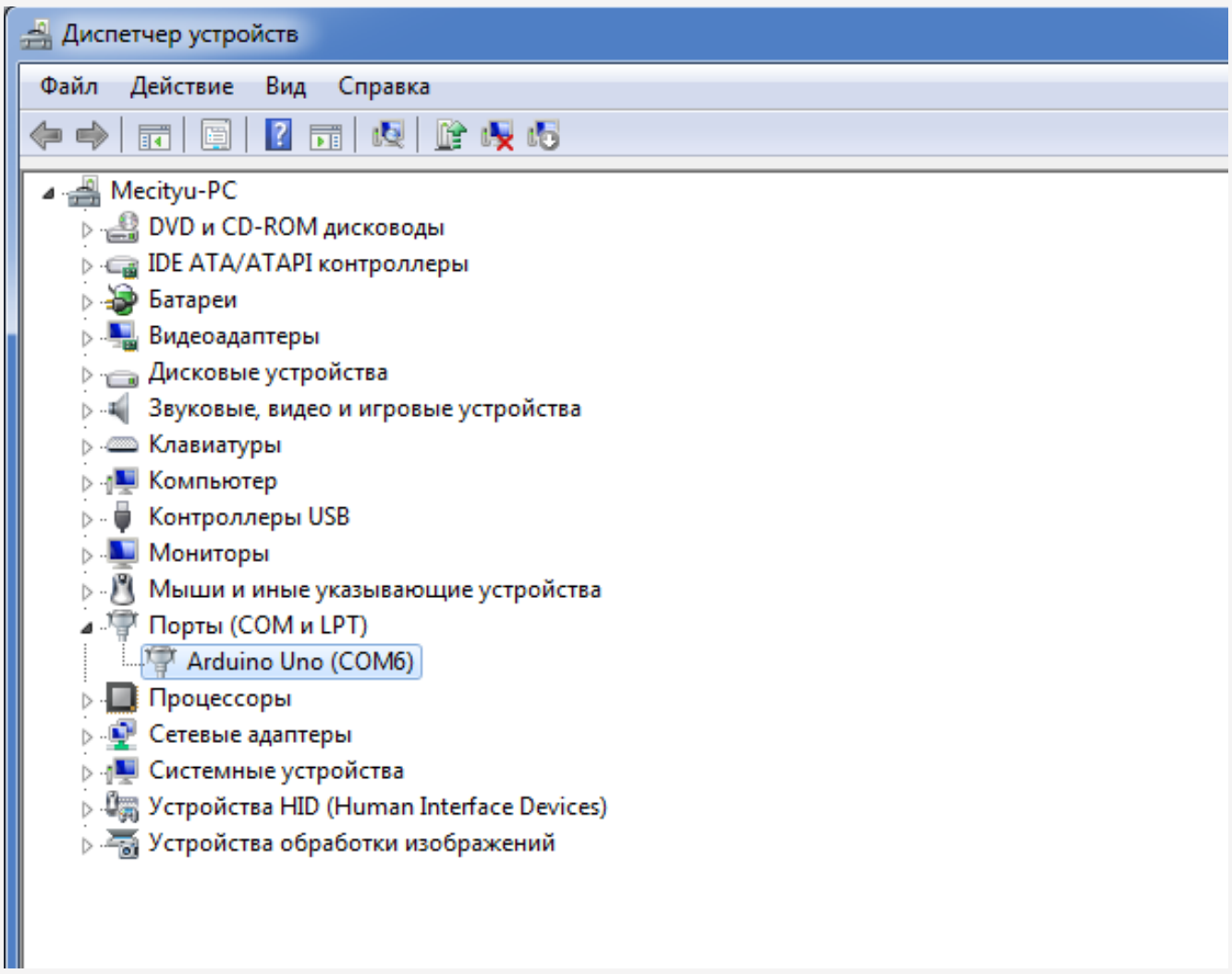
Этот диалог указывает на успешную установку. Это то, что вы должны увидеть. Если нет, проверьте заново все этапы и попробуйте снова

Если вы вернетесь в Диспетчер устройств, устройство Arduino теперь будет распознано вашим компьютером.



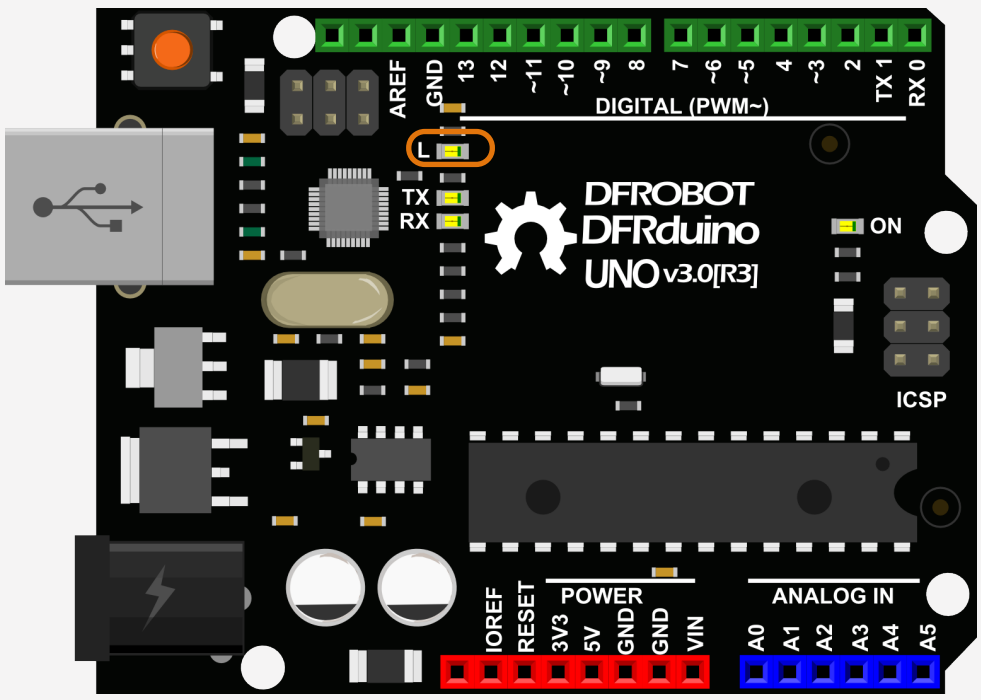
Перейдите в «Диспетчер устройств»> «Порты (COM и LPT)». Вы должны увидеть «Arduino Uno (COM #)». Это COM-порт, используется компьютером для передачи данных на ваш Arduino. В нашем примере компьютер соединяется с Arduino на COM36.

Запомните ваш COM номер, так как он потребуется нам в дальнейшем.

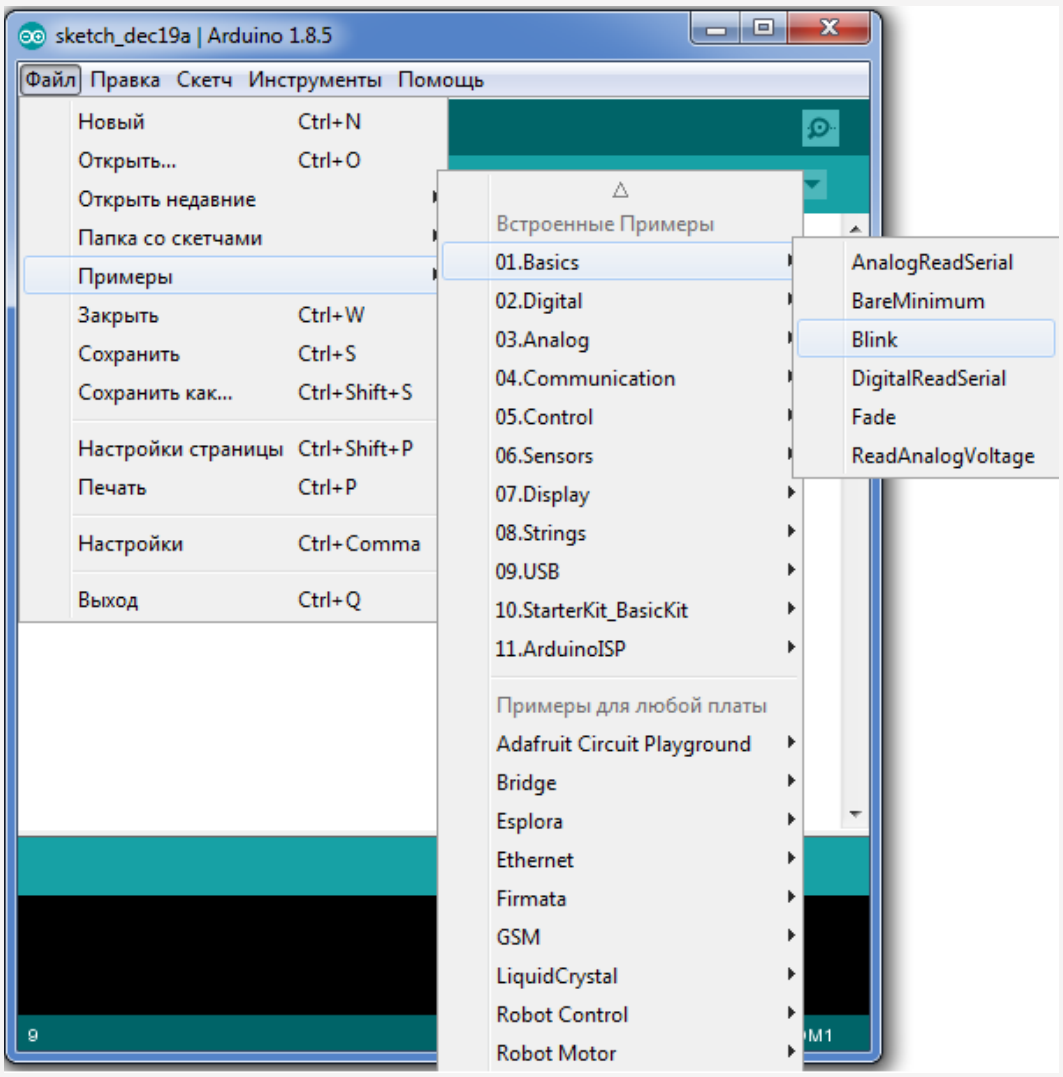


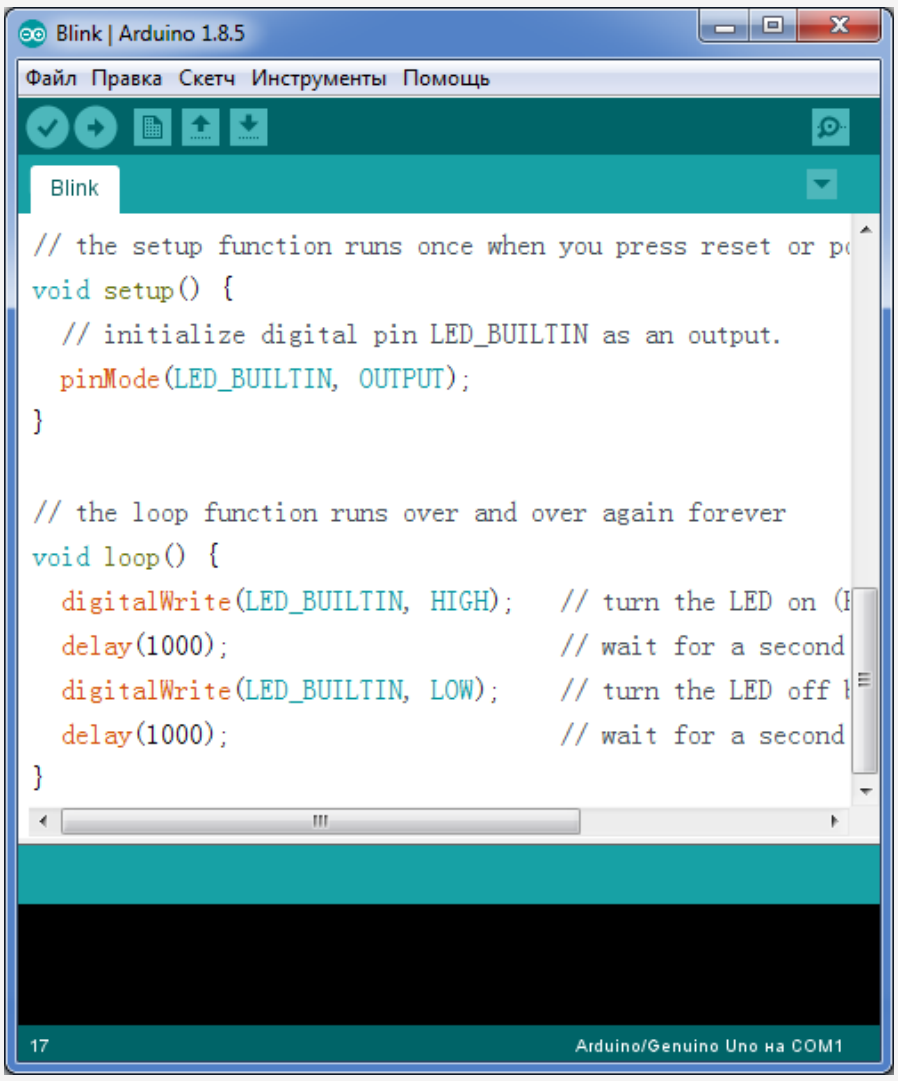
4. Загрузка программы Blink (Мигание)

Откройте Arduino IDE и ознакомьтесь с каждым значком, чтобы узнать их функции. Здесь мы будем использовать очень простой пример кода “Blink”, чтобы пройти весь процесс и проверить, работает ли контроллер.



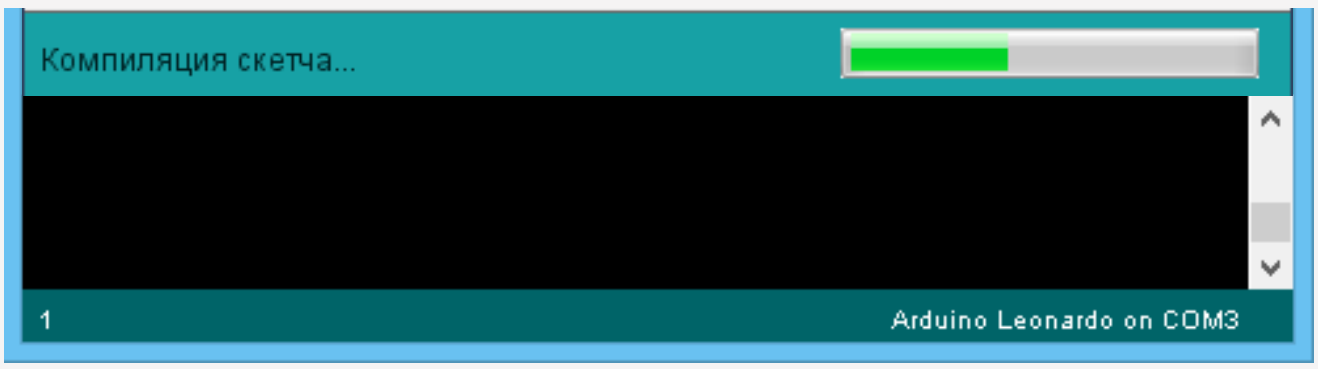
Откройте пример скетча мигание светодиода. Выберете Файл (File) > Образцы (Examples) > 01.Basics > Blink



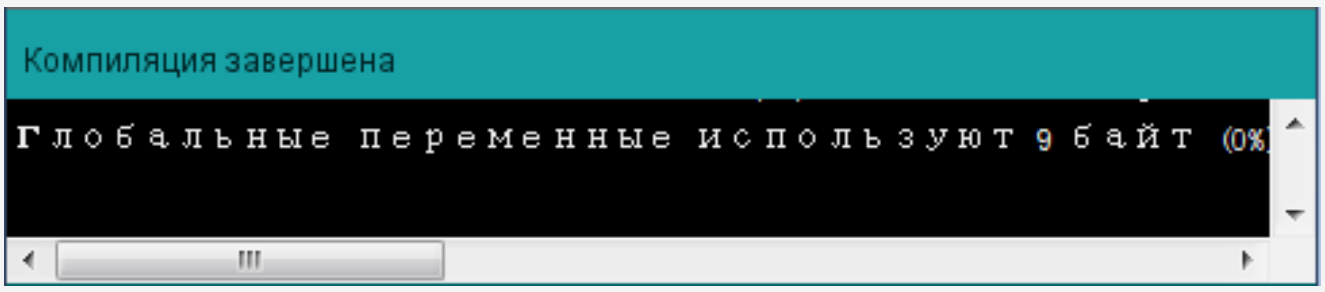


Нажмите «Проверить», чтобы скомпилировать ваш код. IDE переводит код из текста, написанного на языке программирования в инструкции, понятные контроллеру. Этот процесс называется компиляцией

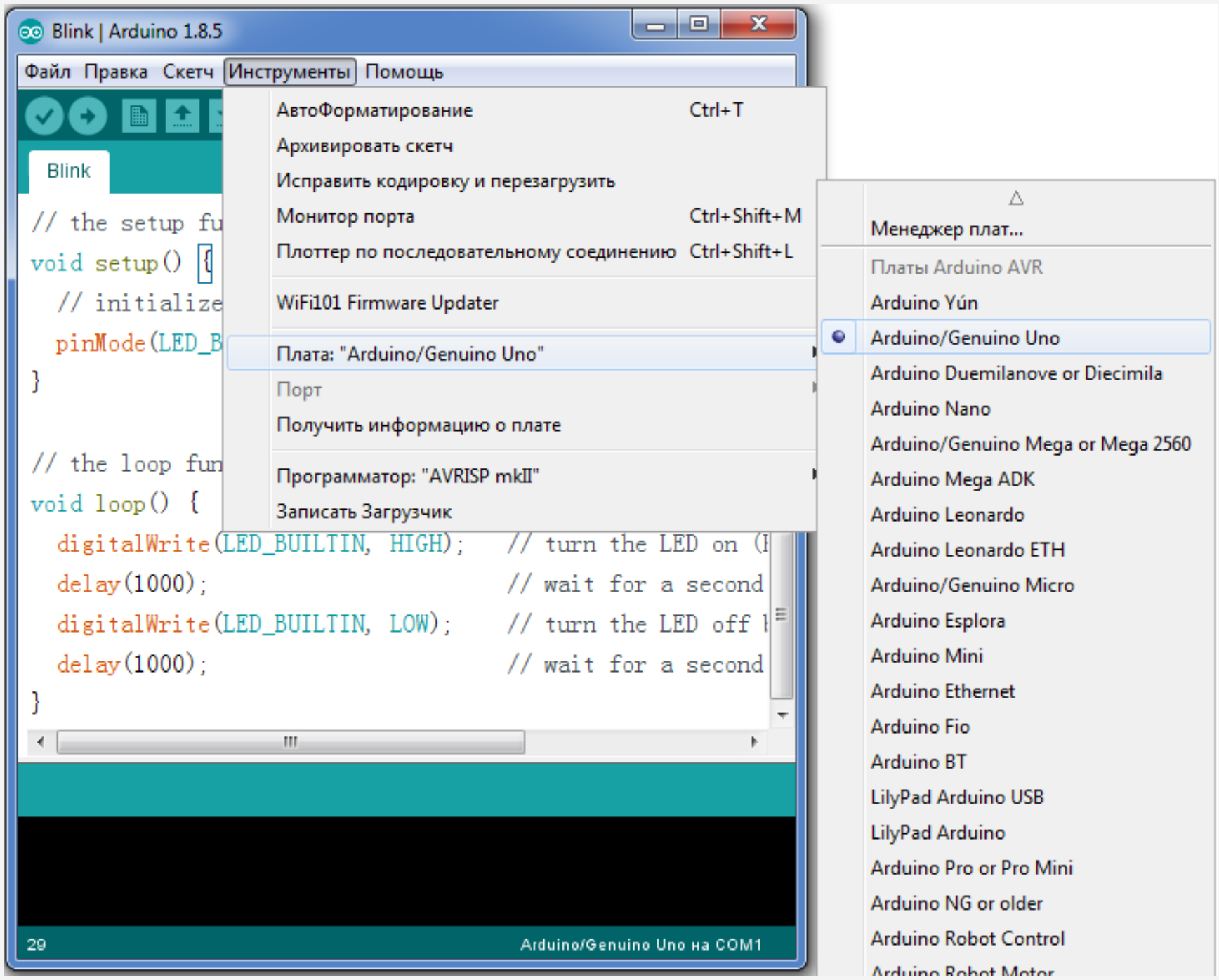
Проверка...



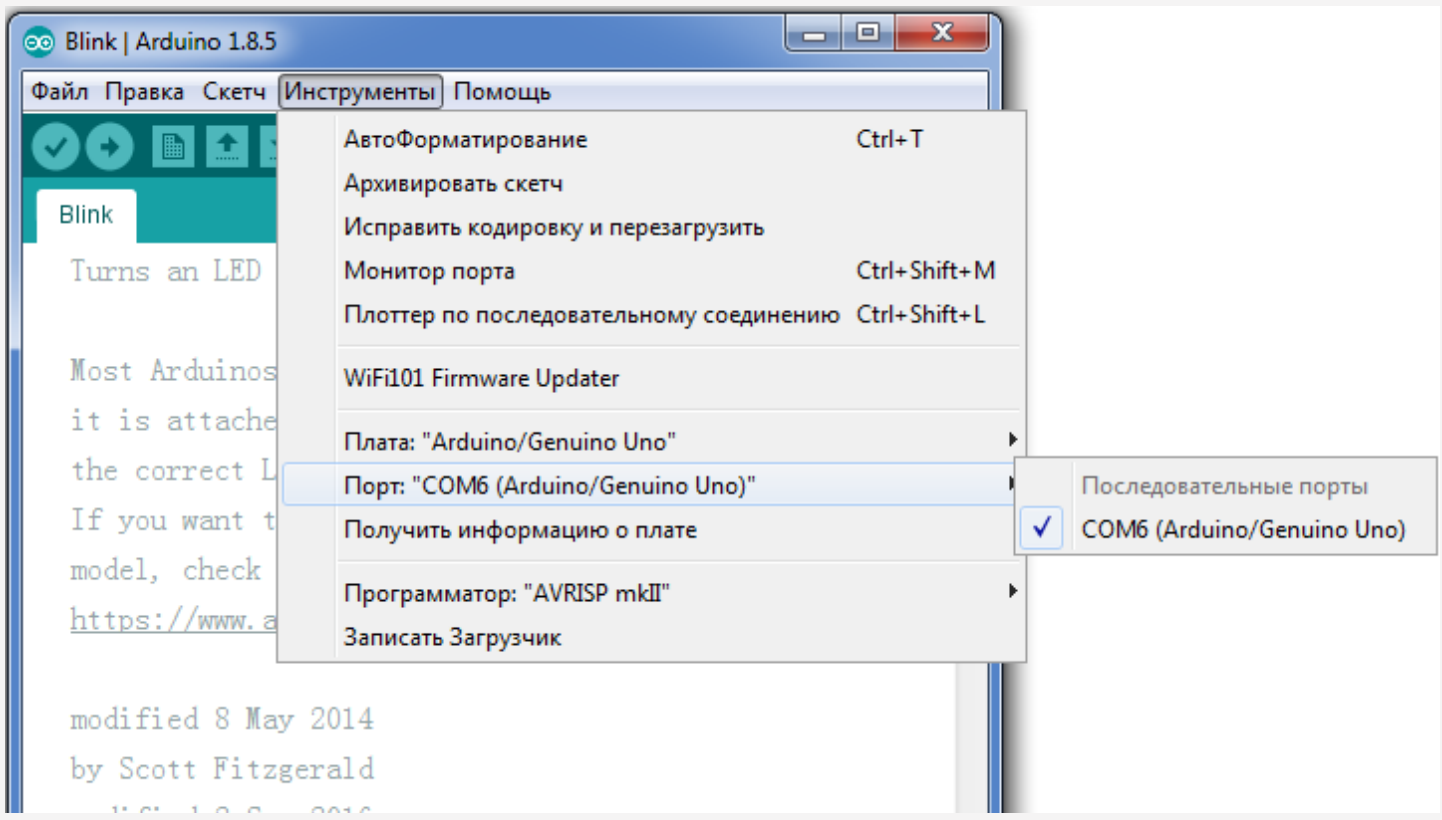
Закончено!



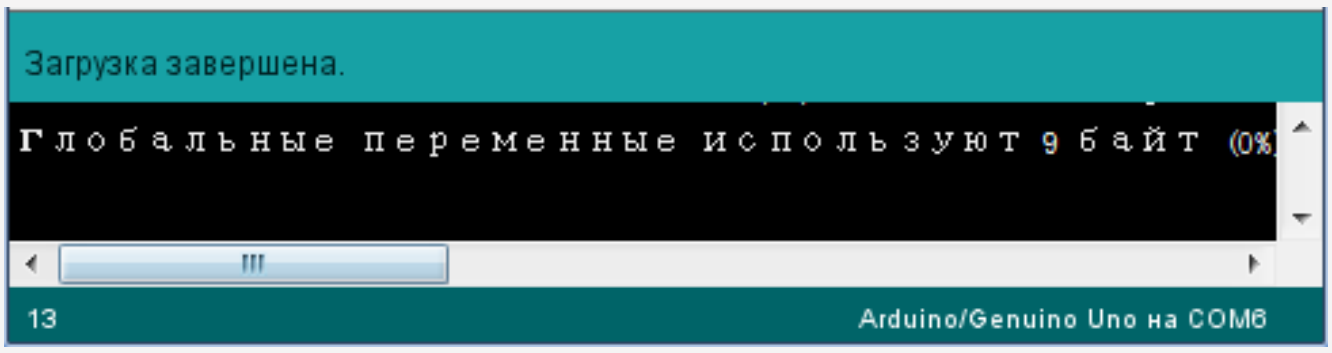
Код, который мы используем, должен быть без ошибок. Если в коде есть ошибки, он не будет загружен на плату.
Пора скачать код на Arduino! Выберите микропроцессор, выбрав “Плата > Arduino UNO”



Далее выберите ваш порт COM, выбрав “Последовательный порт” и выберите номер порта COM, который вы видели ранее. В нашем примере используется COM36.



Нажмите “Загрузить”, чтобы отправить инструкции через USB кабель в Arduino



После того, как он будет закончен, Arduino автоматически запустит код, и встроенный светодиод начнет мигать, как запрограммировано!

После загрузки кода, необходимо выполнить следующие шаги:
Проверить код > Выбрать Плату и Порт > Загрузить
…после этого ваш скетч загрузится

Проект 1

Мигание светодиода

01

DFROBOT
DRIVE THE FUTURE

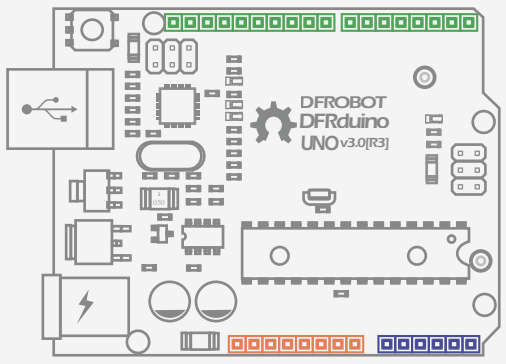
Давайте начнем!

Давайте отправимся в наше приключение Arduino! В первом уроке вы узнаете основы таких компонентов, как светодиоды, кнопки и резисторы, в том числе нагрузочные и понижающие резисторы. Кроме того, вы начнете писать скетчи Arduino для управления светодиодом с помощью Arduino.

Мигание светодиода

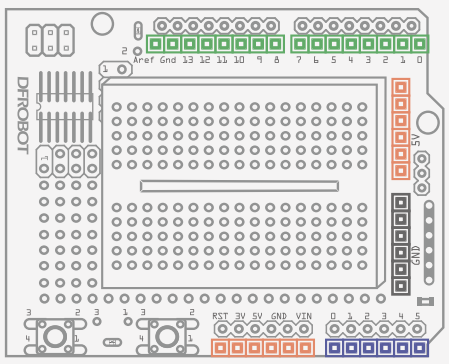
В первом проекте мы снова используем скетч Blink (Мигание), но с соответствующим светодиодом вместо встроенного светодиода. Благодаря этому мы можем иметь четкое представление о том, как работает светодиод и как они могут использоваться в цепи.

Необходимые компоненты



x1

DFRduino UNO R3



x1

Плата расширения

Перемычки M/M

x2

Резистор 220R Ом

x1

Светодиод 5MM

x1

* ВОЗМОЖНО, ВАМ ПОНАДОБИТСЯ ВЫБРАТЬ ДРУГОЙ РЕЗИСТОР В ЗАВИСИМОСТИ ОТ СВЕТОДИОДА, КОТОРЫЙ ВЫ БУДЕТЕ ИСПОЛЬЗОВАТЬ. ВО ВТОРОЙ ЧАСТИ ЭТОГО УРОКА, МЫ ОБЪЯСНИМ, КАК РАССЧИТАТЬ ЗНАЧЕНИЕ СОПРОТИВЛЕНИЯ.

* DFRDUINO – ARDUINO СОВМЕСТИМАЯ ПЛАТЫ ФИРМЫ DFROBOT, ИМЕЮЩАЯ ФУНКЦИОНАЛ ТАКОЙ ЖЕ, КАК И ОРИГИНАЛЬНАЯ ПЛАТА ARDUINO

Аппаратное обеспечение

Чтобы построить схему, вам нужно взять Arduino и уложить плату расширения поверх него.
Соединительные штырьки на нижней части платы расширения должны вставляться в разъемы на плате Arduino. Будьте аккуратны и осторожны, чтобы не согнуть их.
Очистите клейкую ленту с задней стороны макетной платы и затем вставьте ее в Prototype Shield. Теперь вы можете настроить схему в соответствии с рисунком ниже.

Дважды проверьте ориентацию светодиодных ножек на цепи. Светодиоды поляризованы (будут работать только при правильном размещении в цепи). Длинная ножка светодиода подключается к Vcc. (В этом примере 10), а короткая ножка подключается к GND.

Когда вы закончите цепь, подключите контроллер Arduino и компьютер с помощью прилагаемого USB-кабеля

Стандартной практикой является использование проводов различной цветной изоляции для вашей собственной справки, но использование различных комбинаций цветов не останавливает работу схемы.

Обычно красный провод показывает источник питания (Vcc), черный провод указывает на заземление (GND), зеленый провод показывает цифровые контакты, синий провод показывает аналоговые контакты, а белый - другие.

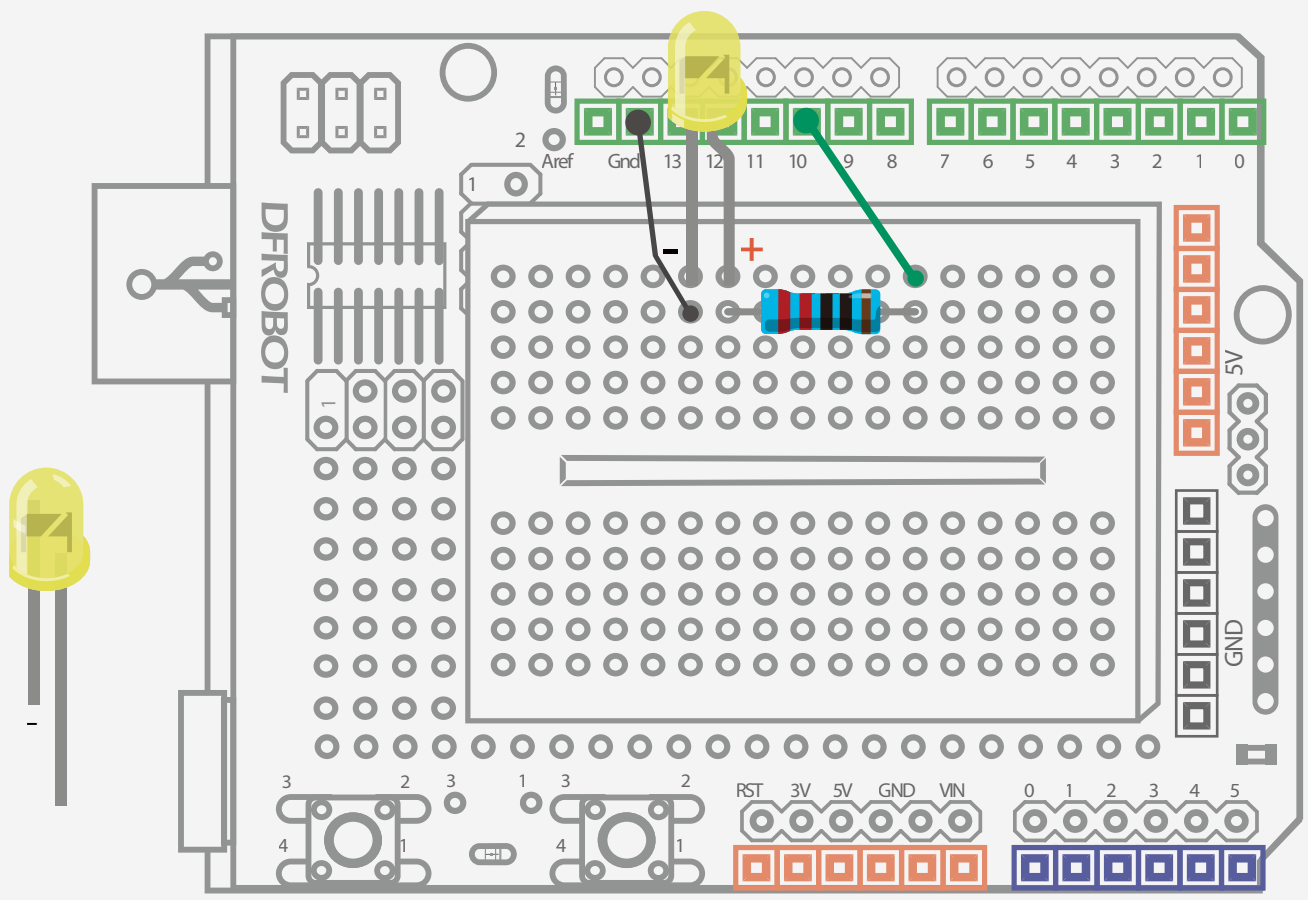


Рис. 1-1 Цепь мигающего светодиода

Скетч Arduino

Откройте IDE Arduino и введите код в качестве примера кода 1-1. (Мы настоятельно рекомендуем вам вводить код вместо копирования и вставки, чтобы вы могли развивать свои навыки кодирования и знать язык кодирования наизусть.)

Пример кода 1-1:

```
//Проект -- Мигание светодиодом
/*
    Описание: включать и выключать светодиод каждую секунду.
*/
int ledPin = 10;
void setup() {
    pinMode(ledPin, OUTPUT);
}
void loop() {
    digitalWrite(ledPin,HIGH);
    delay(1000);
    digitalWrite(ledPin,LOW);
    delay(1000);
}
```

Когда вы закончите вводить код, нажмите «Проверить», чтобы проверить, можно ли скомпилировать код. Если в коде нет ошибок, нажмите «Загрузить», чтобы загрузить код на микроконтроллер. Теперь ваш встроенный светодиод должен мигать и включаться.

Код

Комментарии :

//Проект – мигание светодиодом

многострочные комментарии :

/* текст между этими двумя символами будет прокомментирован; компилятор проигнорирует текст, и текст появится серым * /

Объявление переменных :

int ledPin = 10;

тип переменных

имя переменных

Любая строка кода, которая содержит «//», перед ней не компилируется компилятором. IDE Arduino указывает на это, автоматически поворачивая строку кода. Это позволяет вам писать простые описания на английском языке кода, который вы написали, и облегчит понимание кода для другим людям. Мы называем это комментированием.

Подобно однострочным комментариям, любой текст между / * и * / будет игнорироваться компилятором. Напомним еще раз, Arduino IDE превратит этот текст в серый цвет, чтобы показать, что он закомментирован. Это полезный способ аннотации кода.

Объявление переменных - полезный способ именования и сохранения значений для последующего использования программой. Целые числа (int) представляют числа от -32768 до 32767. В приведенном выше примере у нас есть объявление переменной int равной 10. «ledPin» - это имя переменной, которое мы выбрали. Конечно, вы можете назвать его чем угодно, а не ledPin, но лучше назвать переменную в соответствии с ее функцией. Здесь переменная ledPin указывает, что светодиод подключен к выходу 10 Arduino. Объявление переменной должно заканчиваться (;)

Что такое переменная

Переменная - это место хранения фрагмента данных. Она имеет имя, значение и тип. В приведенном выше примере «int» (целое) является типом, «ledPin» - это имя, а «10» - это значение. В этом примере мы объявляем переменную с именем ledPin типа «int» (целое), что означает, что светодиод подключен к цифровому выходу 10. Переменные отображаются в виде оранжевого текста в скетче.

Целые числа могут хранить числа от -32768 до 32767. Вы должны ввести или объявить переменные перед их использованием. Позже в программе вы можете просто ввести имя переменной, а не вводить данные снова и снова в каждой строке кода, после чего ее значение будет искать и использовать IDE.

Когда вы пытаетесь назвать переменную, запустите ее буквой, сопровождаемой буквой, числом или подчеркиванием. Язык, который мы используем (C), чувствителен к регистру. Существуют определенные имена, которые вы не можете использовать, такие как «main», «if», «while», поскольку у них есть собственная предварительно назначенная функция в среде IDE.

Не забывайте, что имена переменных чувствительны к регистру!

Функция setup()

void setup () {}

Функция не возвращает значение

Пустые скобки

В этом примере в функции setup () есть только одна строка

Формат функции выглядит следующим образом

пин режим (входа, выхода) ;

Pin

Mode (OUTPUT/INPUT)

Функция setup() считывается Arduino при запуске скетча. Он используется для инициализации переменных, режимов вывода, инициализации библиотек и т.д. Функция настройки запускается только после каждого включения или сброса платы Arduino.

Эта функция используется для определения поведения работы цифрового контакта. Цифровые контакты определяются как вход (INPUT) или выход (OUTPUT). В приведенном выше примере вы можете увидеть скобки, содержащие два параметра: переменную (ledPin) и ее поведение (OUTPUT).

«PinMode» настраивает указанный цифровой вывод, чтобы вести себя как вход или выход. Он имеет два параметра: «Pin»: номер контакта, режим которого вы хотите установить «режим»: INPUT, OUTPUT или INPUT_PULLU.
Если вы хотите установить цифровой пин 2 в режиме ввода, какой код вы бы набрали?
Ответ: pinMode (2 INPUT)

Функции

Сегментирующий код в функции позволяет программисту создавать модульные фрагменты кода, которые выполняют определенную задачу, а затем возвращаются в область кода, из которой функция была “вызвана”. Типичный случай для создания функции - это когда нужно выполнять одно и то же действие несколько раз в программе.

В скетче Arduino, есть две требуемые функции – это setup() и loop(). Другие функции должны создаваться вне скобок этих двух функций.

Отличия между INPUT и OUTPUT

INPUT (вход) - сигнал, полученный из внешней среды в Arduino, например, через кнопку. OUTPUT (выход) - это сигнал, который отправляется из Arduino во внешнюю среду, например, светодиод и зуммер.

Если мы прокрутим дальше, мы увидим основную часть кода. Это цикл.

```
void loop() {
    digitalWrite(ledPin,HIGH);
    delay(1000);
    digitalWrite(ledPin,LOW);
    delay(1000);
}
```

Формат функции выглядит следующим образом

digitalWrite (пин, значение) ;

пин | значение HIGH/LOW

Программа Arduino должна включать функцию setup () и loop (), иначе это не сработает.

После создания функции setup (), которая инициализирует и устанавливает начальные значения, функция loop () делает именно то, что предлагает ее название, то есть – цикл, позволяя вашей программе что-то изменять и реагировать. Используйте его для активного контроля платы Arduino. Здесь мы хотим контролировать светодиод постоянно и выключать каждую секунду

Как мы можем это сделать?

В этом проекте мы хотим, чтобы светодиод включался в течение 1 секунды, а затем выключался на 1 секунду и повторял это действие снова и снова. Как мы можем выразить это в коде?

Посмотрите на loop () в первом выражении.

Это связано с другой функцией: digitalWrite ().

```
digitalWrite(ledPin,HIGH);
```

digitalWrite сообщает значение HIGH(включено) или LOW(выключено) цифровому пину. Если контакт сконфигурирован как OUTPUT с pinMode (), его напряжение будет установлено на соответствующее значение: для HIGH 5 В (или 3,3 В на платах 3,3 В) и для LOW 0 В (земля). Обратите внимание, что digitalWrite () применяется только при условии, что pinMode () установлен как OUTPUT. Почему? Читайте дальше.

Отношения между pinMode(), digitalWrite() и digitalRead()

Если pinMode настраивает цифровой пин, чтобы вести себя как вход, вы должны использовать функцию digitalRead (). Если пин сконфигурирован как выход, тогда вы должны использовать digitalWrite ().

ПРИМЕЧАНИЕ. Если вы не установите pinMode () в OUTPUT и подключите светодиод к выходу, при вызове digitalWrite (HIGH) светодиод может выглядеть тусклым. Это связано с тем, что без объяснения установки pinMode (), digitalWrite () будет использовать внутренний резистор подтягивания, который действует как большой резистор, ограничивающий ток.

Например: Светодиод, Зуммер	Например: Кнопка
pinMode(pin,OUTPUT)	pinMode(pin,INPUT)
digitalWrite(pin,HIGH/LOW)	digitalRead(pin)

Далее:

```
delay(1000);
```

delay() делет паузу в программе на определенный отрезок времени (в милисекундах). (1000 милисекунд равны 1 секунде)

Аппаратное обеспечение

Макетная плата

Макетная плата - это способ построения и тестирования цепей без необходимости постоянного пайки. Компоненты вставляются в гнезда на макетной плате, а затем для соединения используются дополнительные провода «перемычки».

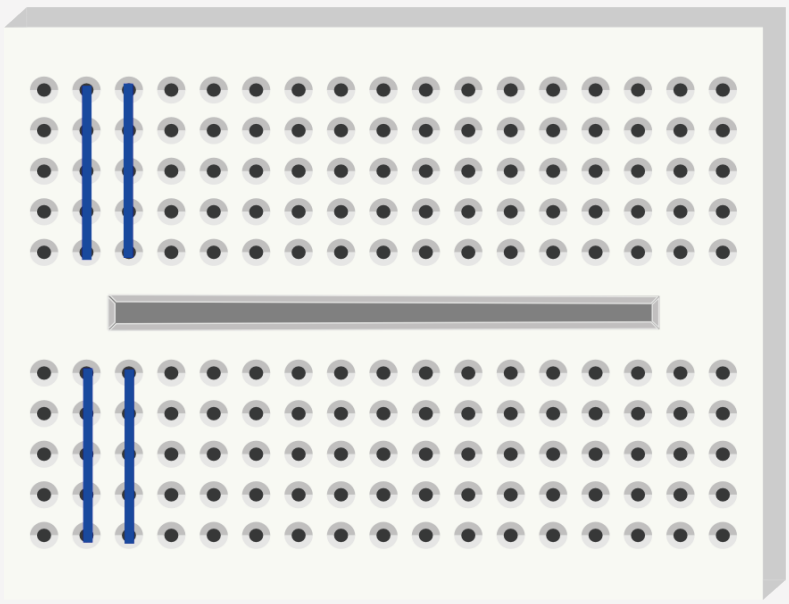


Рис. 1-2 Направление проводимости макетной платы

Макетная плата имеет две колонки, каждая с 17 полосками соединений. На этом рисунке черные линии показывают, как подключается верхняя строка. Тоже самое для каждой другой строки. Два столбца изолированы друг от друга.

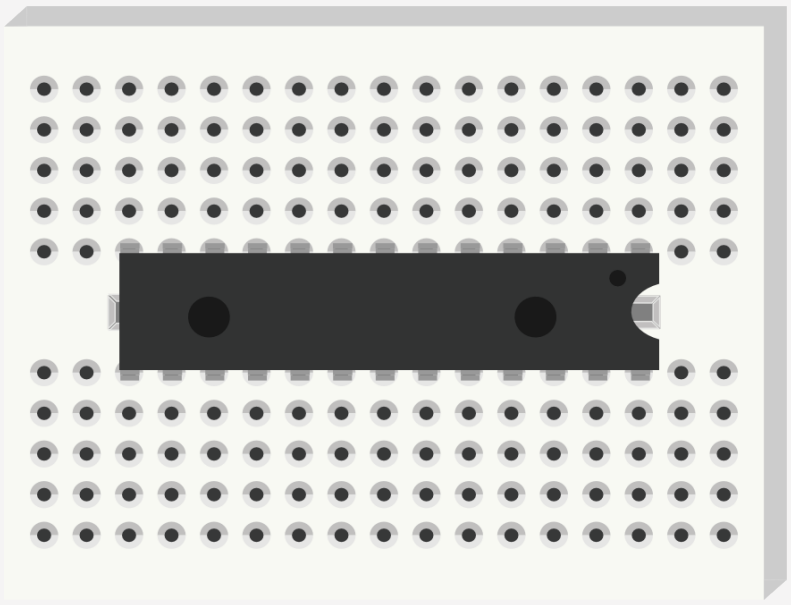


Рис. 1-3. Установление DIP-чип.

Резисторы

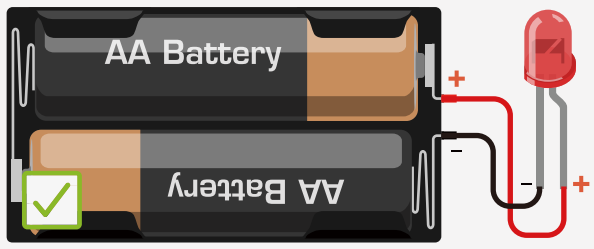
Как следует из названия (англ. resist – оказывать сопротивление (прим. перев.)), резисторы сопротивляются потоку электричества. Чем выше значение резистора, тем больше будет сопротивление и меньше электрического тока. Единица сопротивления называется Омом, который обычно сокращается до Ω (греческая буква Омега).
В отличие от светодиодов, резисторы не поляризованы (не имеют положительного и отрицательного вывода) - они могут быть подключены в любом направлении. Обычно для светодиода требуется напряжение 2 В и ток 35 мА для освещения, поэтому с резистором 100 Ом вы сможете контролировать поток электроэнергии. При сопротивлении меньшем 10 Ом будет превышение по току.
Если вы хотите прочитать значение сопротивления из цветового кода резистора, посетите этот веб-сайт для таблиц калькуляции http://www. 21ic.com/tools/compo nent/ 201003/54192.html

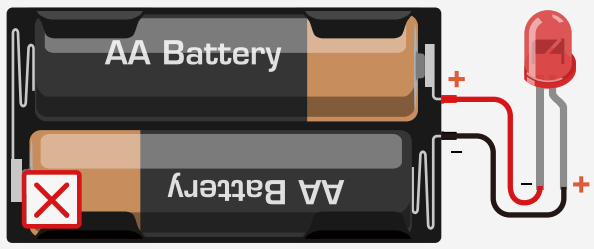
Читаем цветовые кольца резистора

Значение резистора будет отмечено на внешнем корпусе ваших резисторов, но что мы должны делать в случае потери метки и отсутствия измерительных инструментов? Ответ заключается в том, чтобы прочитать значение резистора. Это группа цветных колец вокруг резистора. Подробности в онлайн доступе для тех, кто в этом заинтересован.
Вот онлайн-калькулятор для расчета значения резистора пятицветного кольца: http://ww-w.21ic.com/ tools/compo nent/ 201003/54192.htm

Светодиоды

Светодиод (светодиод) представляет собой двухпроводный полупроводниковый источник света. Это основной диод pn-перехода, который испускает свет при активации. Светодиоды поляризованы, поэтому их нужно подключить правильно. Если вы положите отрицательный вывод светодиода в источник питания и положительный вывод на землю, компонент не будет работать, как показано на диаграмме справа.
В вашем комплекте вы также можете найти светодиоды с 4 выводами. Это RGB-светодиод с 3 встроенными в него основными цветными светодиодами. Его мы рассмотрим позже.





Проект 2

Сигнал бедствия S.O.S.

02

DFROBOT
DRIVE THE FUTURE

Давайте построим генератор кода Морзе с цепью, которую мы построили на первом уроке. Азбука Морзе - это метод передачи текстовой информации в виде серии тонов, огней или кликов. Мы можем использовать медленное мигание и быстрое мигание светодиода вместо точек и тире для обозначения букв алфавита. Например, SOS. Согласно азбуке Морзе, «S» представлен тремя точками, которые мы можем представить с медленным миганием, тогда как «O» представлено тремя тире, которые мы можем представить с быстрым миганием.

Пример кода 2-1:

```
int ledPin = 10;
void setup() {
      pinMode(ledPin, OUTPUT);
}
void loop() {
// 3 быстрых мигания для "S"
      digitalWrite(ledPin,HIGH);
      delay(150);
      digitalWrite(ledPin,LOW);
      delay(100);

      digitalWrite(ledPin,HIGH);
      delay(150);
      digitalWrite(ledPin,LOW);
      delay(100);

      digitalWrite(ledPin,HIGH);
      delay(150);
      digitalWrite(ledPin,LOW);
      delay(100);

       delay(100);    //100 миллисекунд разрыв между
каждой буквы
```

```
//3 быстрых мигания для “0”
digitalWrite(ledPin,HIGH);
delay(400);
digitalWrite(ledPin,LOW);
delay(100);

digitalWrite(ledPin,HIGH);
delay(400);
digitalWrite(ledPin,LOW);
delay(100);

digitalWrite(ledPin,HIGH);
delay(400);
digitalWrite(ledPin,LOW);
delay(100);

delay(100);
  //Задержка в 100 миллисекунд между каждой буквой

//3 быстрых мигания для “S” снова
digitalWrite(ledPin,HIGH);
delay(150);
digitalWrite(ledPin,LOW);
delay(100);

digitalWrite(ledPin,HIGH);
delay(150);
digitalWrite(ledPin,LOW);
delay(100);

digitalWrite(ledPin,HIGH);
delay(150);
digitalWrite(ledPin,LOW);
delay(100);

    delay(5000);        // ждать 5 секунд, чтобы повторить следующий S.O.S. сигнал
}
```

КОД

Для этого требуется много повторяющейся работы. Есть ли способ лучше? Взгляните на следующий пример кода.

Пример кода 2-2:

```
//Второй проект - сигнал S.O.S
int ledPin = 10;
void setup() {
        pinMode(ledPin, OUTPUT);
}
void loop() {
        // 3 быстрых мигания для «S» снова
        for(int x=0;x<3;x++){
            digitalWrite(ledPin,HIGH);          // настроить светодиод на включение
            delay(150);                         // Задержка 150 миллисекунд
            digitalWrite(ledPin,LOW);           // настроить светодиод на отключение
            delay(100);                         // задержка 100 миллисекунд
        }

        //Задержка в 100 миллисекунд между каждой
буквой
        delay(100);

        //3 быстрых мигания для “O”
        for(int x=0;x<3;x++){                   //настроить светодиод на включение
            digitalWrite(ledPin,HIGH);          // задержка 400 миллисекунд
            delay(400);                         // настроить светодиод на отключение
            digitalWrite(ledPin,LOW);           // задержка 100 миллисекунд
            delay(100);
        }

        //Задержка в 100 миллисекунд между каждой
буквой
        delay(100);
```

```
    // 3 быстрых мигания для представления “S” снова
    for(int x=0;x<3;x++){
            digitalWrite(ledPin,HIGH);          // настроить светодиод на включение
            delay(150);                         // Задержка 150 миллисекунд
            digitalWrite(ledPin,LOW);           // настроить светодиод на отключение
            delay(100);                         // задержка 100 миллисекунд
     }

    // ждать 5 секунд, чтобы повторить следующий сигнал S.O.S
    delay(5000);
}
```

После загрузки кода вы увидите, что светодиод мигает сигналом S.O.S и с периодичностью 5 секунд. Если вы установите элементы в водонепроницаемый футляр, вы можете использовать его для парусного спорта или похода!

КОД

Первая часть двух скетчей идентична: мы инициализировали переменную и сконфигурировали цифровой пин 10 как INPUT. В основном коде loop () вы можете найти строки, похожие на последний проект, для включения и отключения светодиода. Разница в том, что основной код содержит 3 независимых блока операторов.

Первый блок конфигурирует 3 контакта как OUTPUT

```
for(int x=0;x<3;x++){
      digitalWrite(ledPin,HIGH);     // настроить светодиод на включение
      delay(150);                    // Задержка 150 миллисекунд
      digitalWrite(ledPin,LOW);      // настроить светодиод на отключение
      delay(100);                    // задержка 100 миллисекунд
 }
```

Конструкция "for"

Конструкция «for» используется для повторения блока положений, заключенного в скобки. Счетчик прироста обычно используется для увеличения и завершения цикла. Конструкция «for» полезна для любой повторяющейся операции и часто используется в комбинации для работы с массивами данных/пинов.

заголовоке цикла «for» есть три части:

Итак, для конструкции «for» в скетче

```
for(int x=0;x<3;x++){
      ……
}
```

Шаг 1: Инициализировать значение переменной x как 0

Шаг 2: Оцените, если x меньше 3.

Шаг 3: Если он действителен, выполните следующую инструкцию

Шаг 4: x увеличивается и становится равным 2.

Шаг 5: Повторите шаг 2 меньше 3.

Шаг 6: Повторите шаг 3

...

Когда x = 3, условие x <3 ложно, и программа пропускает код.

Составим уравнение x <3, чтобы цикл повторялся 3 раза (от 0 до 2).

Если бы мы хотели, чтобы цикл повторялся 100 раз, мы можем использовать следующий код: for (int x = 0; x <100; x ++) {}

Некоторые операторы сравнения, такие как «>», «<», часто используются в условных операторах программирования. Они будут рассмотрены более подробно в следующем разделе.

Некоторые общие операторы, которые вы можете использовать

«<» - пример оператора сравнения. Операторы сравнения позволяют Arduino сравнивать два значения. Ниже приведены примеры использования часто используемых операторов сравнения:

- == (равно)
- ! = (не равно)
- <(меньше)
- > (больше)
- <= (меньше или равно)
- > = (больше или равно)

Остерегайтесь случайного использования одного знака равенства (например, если (x = 10)). Единый знак равенства является оператором присваивания и устанавливает x до 10 (помещает значение 10 в переменную x). Знак двойного равенства означает вместо этого оператор сравнения (например, если (x == 10)),
Кроме того, будьте осторожны, чтобы между <= или> = не было пробела.

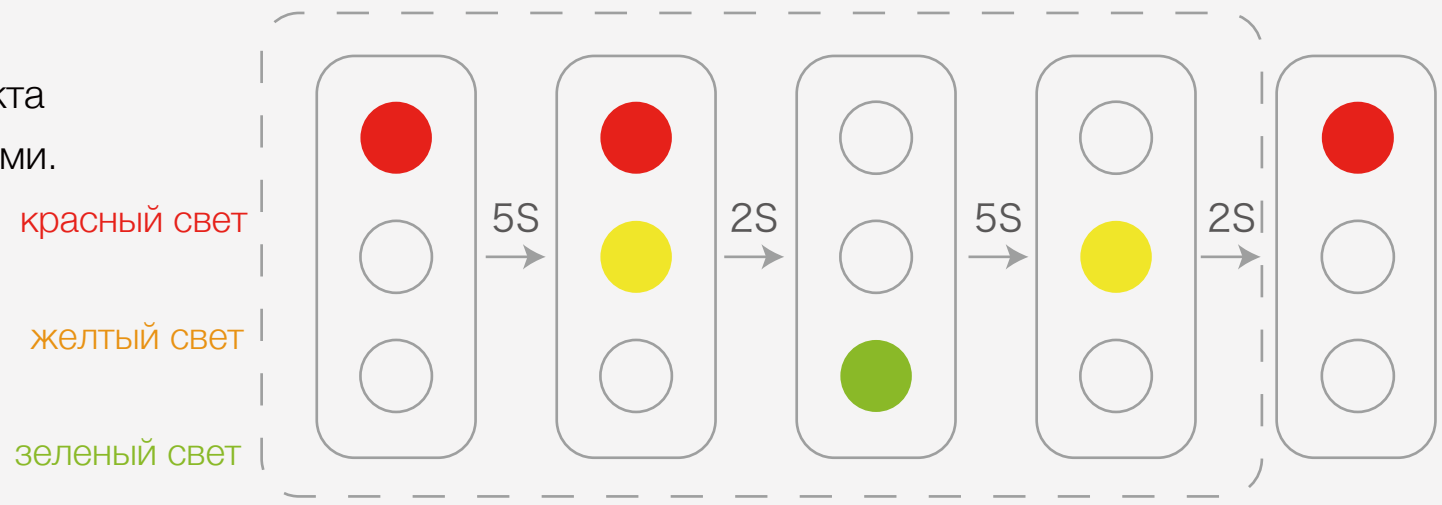
Вы также можете использовать такие арифметические операции, как: сложение, вычитание, умножение и деление (+ - * /).

Теперь вы должны знать, как работает цикл «for». В коде есть три цикла. Первый "for" цикл повторяется 3 раза, а длительное мигание чередуется 3 раза, давая выход из 3 точек, представляющих букву S в азбуке Морзе. Второй цикл for также повторяется 3 раза, а временное мигание чередуется 3 раза, давая выход из 3 тире, представляя букву O в азбуке Морзе. Третий для цикла идентичен первому, а также выводит S в азбуке Морзе.

Вы можете заметить, что в разных блоках кода есть разные переменные x, но они не мешают друг другу. Потому что он имеет ограниченную область действий в определенном блоке функции. Это аналог глобальных переменных, которые необходимо объявить в верхней части кода из функции setup () и loop (), и вы сможете использовать их в любом месте кода.

Упражнение

Давайте сделаем светофор, используя 3 цифровых контакта для управления 3 светодиодами.



Проект 3

Интерактивный светофор

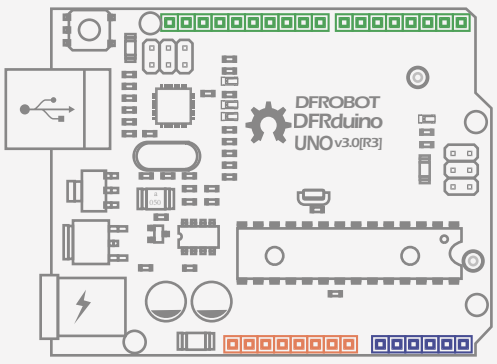
03

DFROBOT
DRIVE THE FUTURE

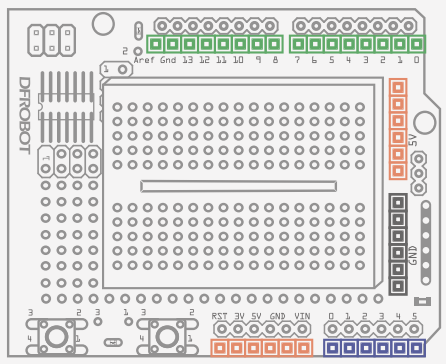
Интерактивный светофор

В этом уроке вы начнете свой первый интерактивный проект Arduino, запустив светофор с управляемыми кнопками. Когда кнопка нажата, свет будет изменяться для прохода пешеходов.

Необходимые компоненты



DFduino UNO R3 x1



Плата расширения x1

Перемычки “мама-мама” x13

Резистор 220 Ом x6

Нажимная кнопка x1

Светодиод 5ММ x2

Светодиод 5ММ x2

Светодиод 5ММ x1

* Почему 5 светодиодов с 6 резисторами? Дополнительный резистор является понижающим резистором для кнопки.

* После этого мы больше не будем перечислять основную плату Arduino, макетную плату, плату расширения или перемычки в списке компонентов, поскольку они будут необходимы для каждого проекта.

Цепь

Следуйте приведенной ниже схеме подключения, чтобы построить свою цепь.
Используйте только прилагаемый USB-кабель для питания Arduino после сборки схемы. Этот кабель обеспечивает устойчивое напряжение 5В к Arduino. Если вы используете другой источник питания, есть вероятность, что напряжение может быть слишком высоким, что может привести к перегрузке компонентов.

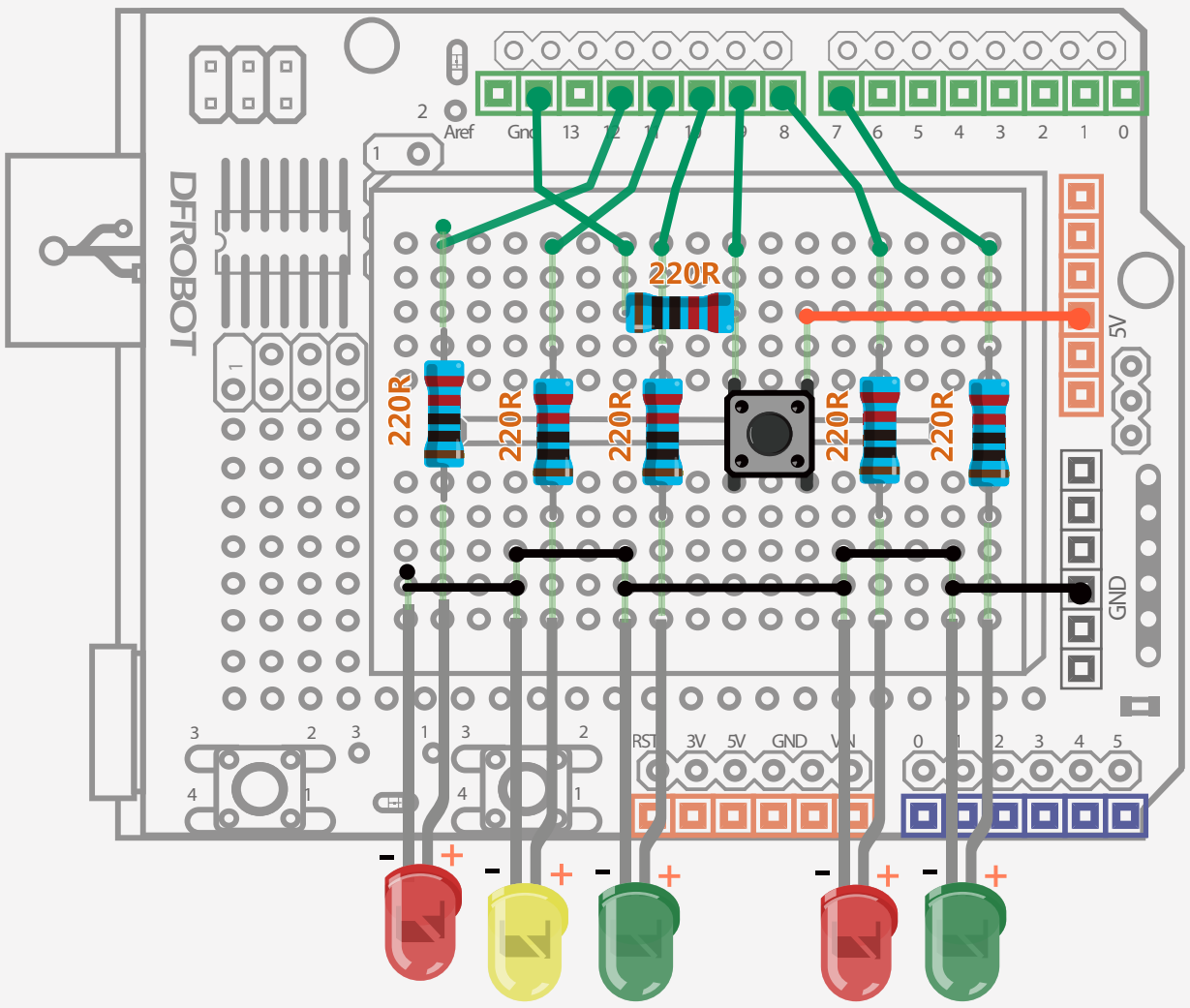


Рис. 3-1 Схема подключения светодиодного мигания

Код

Скетч первоначально из “Beginning Arduino”

Пример кода 3-1:

```
// PROJECT 3 Интерактивный светофор
int carRed = 12;        //настроить светофор
int carYellow = 11;
int carGreen = 10;
int button = 9;         //пин кнопки
int pedRed = 8;         //настроить свет для пешеходов
int pedGreen = 7;
int crossTime = 5000;               //время прохождения пешеходов
unsigned long changeTime;       //время нажатия кнопки

void setup() {
        //настраиваем все светодиоды как выходные
        pinMode(carRed, OUTPUT);
        pinMode(carYellow, OUTPUT);
        pinMode(carGreen, OUTPUT);
        pinMode(pedRed, OUTPUT);
        pinMode(pedGreen, OUTPUT);
           pinMode(button, INPUT);          //настройка кнопки в качестве
входного
             digitalWrite(carGreen, HIGH); //инициализировать зеленый
светофор на
            digitalWrite(pedRed, LOW);      //инициализировать красный
пешеходный свет
}

void loop() {
        int state = digitalRead(button);
        //проверяем, нажата ли кнопка, и если прошло 5 секунд после ее
последнего нажатия.
        if(state == HIGH && (millis() - changeTime)> 5000){
        //выполняем функцию изменения светодиода
        changeLights();
        }
}
void changeLights() {
```

```
    digitalWrite(carGreen, LOW);    //выключен зеленый светофор
    digitalWrite(carYellow, HIGH);   //желтый светофор на
    delay(2000);                    //ждать 2 секунды

    digitalWrite(carYellow, LOW);   //желтый светофор выключен
    digitalWrite(carRed, HIGH);      //красный светофор на
     delay(1000);                        //ждать 1 сек по соображениям
безопасности

    digitalWrite(pedRed, LOW);      //красный свет пешехода
выключен
    digitalWrite(pedGreen, HIGH);  //светлый пешеходный свет

    delay(crossTime);               //время для пересечения улицы

      //мигать зеленым пешеходным светом, чтобы скоро сообщить
пешеходам
    for (int x=0; x<10; x++) {
        digitalWrite(pedGreen, HIGH);
        delay(250);
        digitalWrite(pedGreen, LOW);
        delay(250);
    }
    digitalWrite(pedRed, HIGH);     //красный пешеходный свет
    delay(500);

    digitalWrite(carRed, LOW);      //отключен красный светофор
     digitalWrite(carYellow, HIGH);   //желтый светофор при задержке
(1000)
    delay(1000);
    digitalWrite(carYellow, LOW);   //желтый светофор выключен
    digitalWrite(carGreen, HIGH);    //зеленый светофор на

     changeTime = millis()           // записываем продолжительность
с момента последнего изменения
    //вернуться к циклу основного кода
}
```

После загрузки скетча взгляните на то, как изменяется светодиод. Во-первых, зеленый свет включился, чтобы позволить автомобилям проехать, а красный зажегся для пешеходов. Когда вы нажмете кнопку, пешеходный светофор поменяет свет с красного на зеленый, и светофор изменится с зеленого на зеленый и красный. Затем происходит задержка, позволяющая пешеходам переходить улицу.

Когда происходит отсрочка задержки, зеленый пешеходный индикатор мигает, чтобы уведомить пешеходов. Когда это завершается, свет возвращается в исходное состояние с включенным зеленым светофором и красным светом для пешеходов.

Вышеприведенные коды выглядят сложными, но на самом деле они не являются таковыми на практике.

Если у вас возникнут какие-либо трудности в выполнении этого задания, попробуйте сначала нарисовать схему, подобную той, что находится в домашней работе проекта 2. Это поможет вам лучше понять коды. Удачи!

Код

Основываясь на двух предыдущих проектах, большинство кодов должно иметь для вас значение. Коды начинаются с набора объявлений переменных, но мы использовали новый термин. Это объясняется ниже:

```
unsigned long changeTime;
```

Прежде чем мы будем использовать int для хранения целых чисел от -32768 до 32767. Комманда long, которую мы используем в этом проекте, может хранить целые числа от -2147483648 до 2147483647. Unsigned long не может хранить отрицательные числа. Поэтому он хранит целые числа от 0 до 4294967295.
Если мы используем int, мы можем перейти на предел 32 секунды (32768 мл сек) и рискнуть ошибками при запуске скетча.
В соответствии с этим нам нужна другая команда для хранения целых чисел и для исключения отрицательных чисел, например unsigned long, чей предел составляет 49 дней.
Затем мы вводим функцию setup () для настройки светодиода и кнопки.

```
pinMode(button, INPUT);
```

Мы хорошо знакомы с функцией pinMode (), представленой в Проекте 1. Её отличие от проекта со светодиодом заключается в том, что кнопка должна быть установлена как INPUT. В функции setup (), пожалуйста, инициализируйте пешеход и светофор:
digitalWrite (carGreen, HIGH); // инициализация включение зеленого света в светофоре
digitalWrite (pedRed, LOW); // включение пешеходного красного света

Первая строка основного скетча - проверить состояние кнопки в pin9.

```
int state = digitalRead(button);
```

Может ли поле переменной быть бесконечно большим?

Почему некоторые переменные хранят большие объемы данных, а некоторые не могут? Это зависит от емкости хранилища «int» - оно намного меньше, чем «unsighned long».
Ваш Arduino точно также, как компьютер и микроконтроллер имеет ограниченное пространство для хранения. Максимальное пространство для хранения основного чипа Arduino UNO (Atmega 328) составляет 32 тыс., следовательно, если мы можем сохранить пространство для хранения, мы обязательно должны это сделать.

Общие переменные с которыми вы столкнетесь:

Тип данных	RAM	Диапазон
boolean	1 байт	0 ~ 1 (True or False)
char	1 байт	-128 ~ 127
unsigned char	1 байт	0~255
int	2 байт	-32768 ~ 32768
unsigned int	2 байт	0 ~ 65535
long	4 байт	-2147483648 ~ 2147483647
unsigned long	4 байт	0 ~ 4294967295
float	4 байт	-3.4028235E38 ~ 3.4028235E38
double	4 байт	-3.4028235E38 ~ 3.4028235E38

Существуют различные типы переменных. Int и long для целых чисел, char - для символов, float и double - для переменных с десятичной точкой.

В функции setup () есть новая команда «digitalRead ()»()!

digitalRead (pin)

pin

Эта команда используется для считывания состояния цифрового вывода, будь то высокий (1) или низкий (0). Команда имеет один параметр: «pin number».

Эта команда передает сигнал, считываемый в состояние переменной для дальнейшего суждения.

Когда стоит значение HIGH или 1, это означает, что кнопка нажата. Когда стоит значение LOW или 0, это означает, что кнопка не нажата.

Мы можем проверить значение состояния, чтобы проверить, нажата ли кнопка.

```
if(state == HIGH && (millis()
changeTime)> 5000) {
   //carry out light-changing
command
        changeLights();
}
```

Здесь мы используем команду if() для проверки условий.

```
if(condition){
   Statement;
}
```

Если условие, указанное в скобках, выполняется, оно выполняет инструкцию. Если нет, программа пропускает код.

Другими словами, когда выражение «if» возвращает TRUE, код в скобках запускается. Если оно возвращает FALSE, код в скобках пропускается.В выше приведенных кодах первое условие, является ли переменное состояние HIGH (или нет). Когда нажимается кнопка, состояние становится HIGH. Второе условие - это возврат значения для «millis ()» минус, что для изменения времени больше 5000.

Мы используем «&&» для подключения двух условий. Это логический оператор, показывающий, что мы хотим, чтобы оба вышеуказанных условия выполнялись одновременно.

«Mills ()» - это команда, которая возвращает количество миллисекунд, так как плата Arduino начала запускать текущую программу. Эта камера будет переполняться (возвращаться к нулю) примерно через 50 дней.Здесь мы используем его для вычисления, если есть перерыв более 5 секунд при нажатии кнопки более одного раза. Если он короче 5 секунд, он пропускает код, чтобы избежать ошибок, вызванных случайным нажатием кнопки.

Это команда внутри команды «if ()».

```
changeLights();
```

Это функция, созданная вне функции loop (). Когда мы хотим использовать её, нам просто нужно указать имя функции. Она не возвращает значение и не нуждается в передаче параметров. Когда она будет использоваться, программа запустит функцию и вернется к основному коду. Старайтесь не пропустить круглую скобку при использовании этой функции, поскольку она не будет распознана.

Логические операции

Другие булевые операторы:

- && ——логика и (Истинно, только если оба условия верны)
- || —— логическая или (Истинно, если любое условие истинно)
- ! —— не логическая (Истинно, если условие ложно)

Компоненты

Нажимная кнопка

Используемая нами кнопка имеет 4 контакта. Когда вы нажимаете кнопку или поворачиваете рычаг, они соединяют два контакта вместе
так что электричество может проходить через них. На самом деле существуют только две электрические связи; внутри контактов коммутатора 1 и 4 соединены вместе, как и 2 и 3. Маленькие тактильные переключатели, которые используются в этом уроке, имеют четыре соединения. У вас может быть один с двумя контактами, но они работают одинаково.

Кнопка может включать и выключать электричество, протекающее по цепи. В проекте, когда он нажат, пин D9 определяет HIGH (включено), в противном случае он остается LOW(отключено).

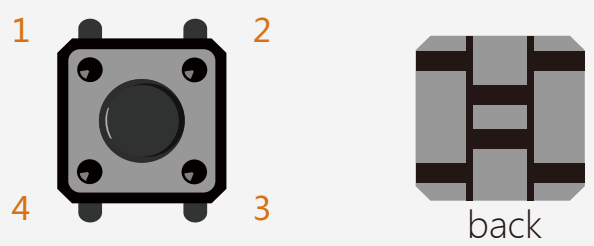


Рис. 3-2. Структура кнопки (спереди и сзади)

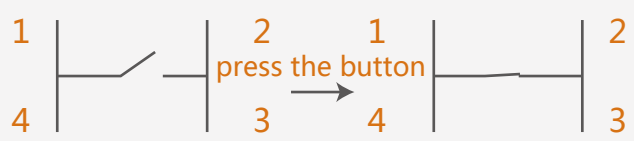


Рис. 3-3. Принципиальная схема кнопки

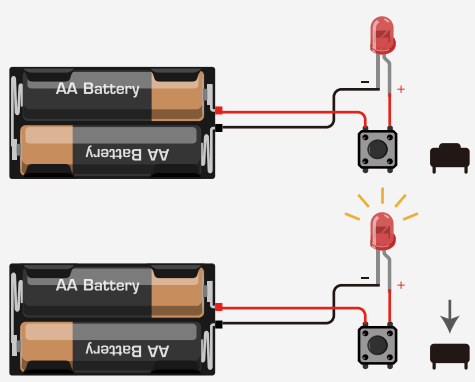
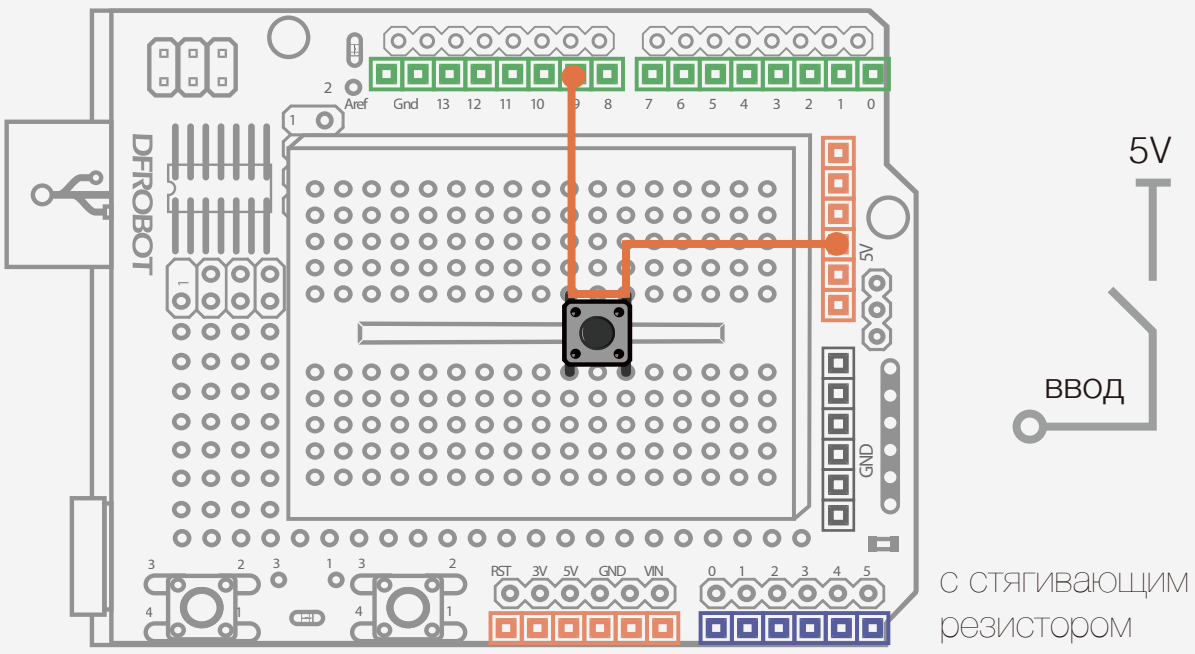


Рис. 3-4

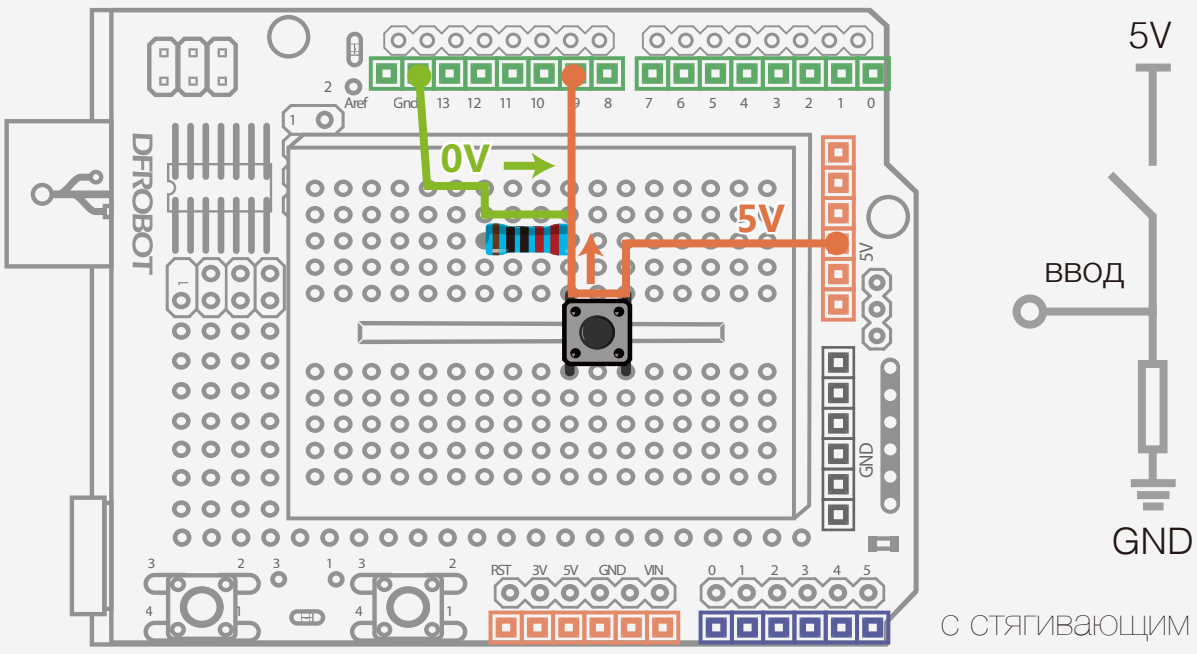
Что такое стягивающий резистор?

В электронных логических цепях используются стягивающие резисторы для обеспечения того, чтобы входы в ардуино устанавливались на ожидаемых уровнях логики, если внешние устройства отключены. Стягивающий резистор «тянет» напряжение провода, к которому он подключен, к земле, когда другие компоненты на линии неактивны.

Когда переключатель на линии разомкнут, он имеет высокий импеданс и действует так, как будто он отключен. Поскольку другие компоненты работают так, как будто они отключены, цепь работает так, как будто она отключена, а нагрузочный резистор подключает провод к логическому уровню HIGH. Когда другой компонент на линии активен, он будет отменять уровень логики HIGH, установленный подтягивающим резистором. Подтягивающий резистор гарантирует, что провод находится на определенном уровне логики, даже если к нему не подключены активные устройства.



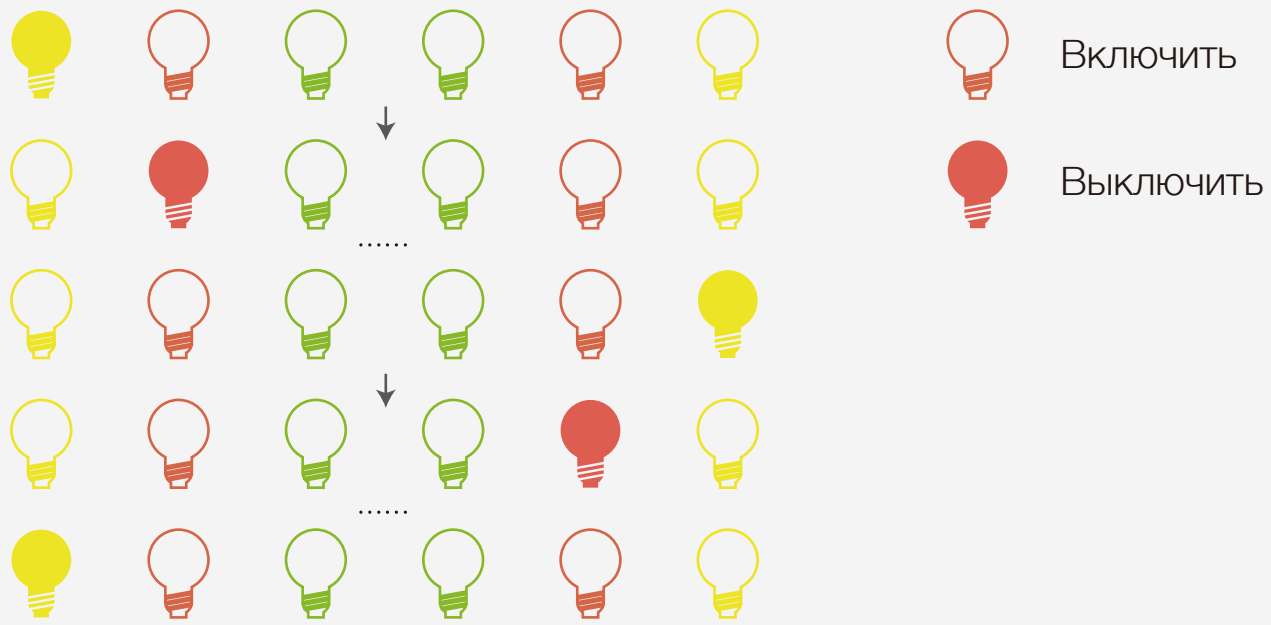
с стягивающим резистором



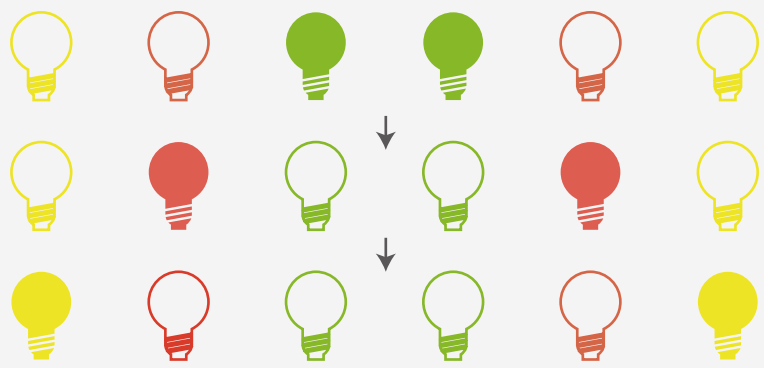
с стягивающим резистором

Домашнее задание

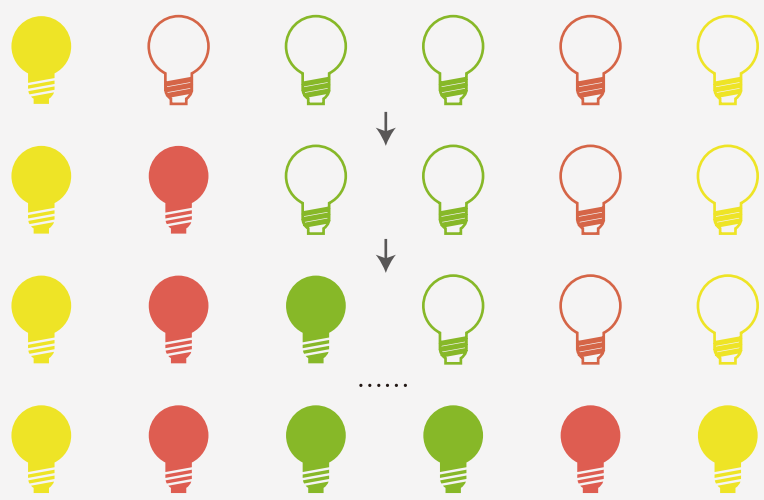
1、Выберите 6 светодиодов любого цвета и сделайте светодиодное табло



2、Когда вы закончите, попробуйте подсвечивать столбец светодиодов посередине и пропускать свет к любому краю.



3、Или свет столбца слева направо.



Проект 4

Проект 4

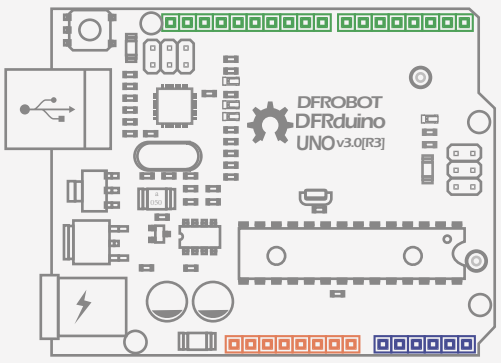
04

DFROBOT
DRIVE THE FUTURE

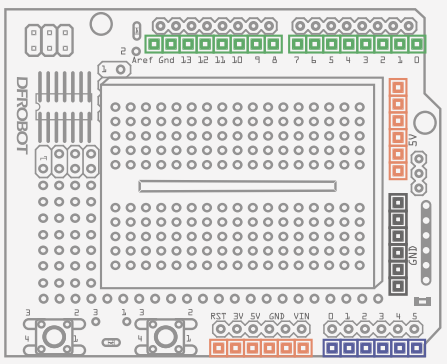
“Дыхание” светодиода

На предыдущих уроках мы узнали, как включать и выключать светодиод с помощью программирования Arduino. Также можно контролировать яркость вашего светодиода. На вашем контроллере имеется 6 цифровых контактов с надписью «~». Это означает, что эти контакты могут использовать сигнал PWM (ШИМ). Мы построим светодиодный фейдер RGB при помощи PWM (ШИМ) управляя яркостью светодиодов.

Компоненты



DFduino UNO R3 x1



Плата расширения x1

Перемычки “мама-мама” x2

220 Ом x1 x1

Светодиод 5ММ (Зеленый) x1

Цепь

Схема подключения такая же, как и в Проекте 1. Если вы неуверенны в схеме подключения, вернитесь к Проекту 1.

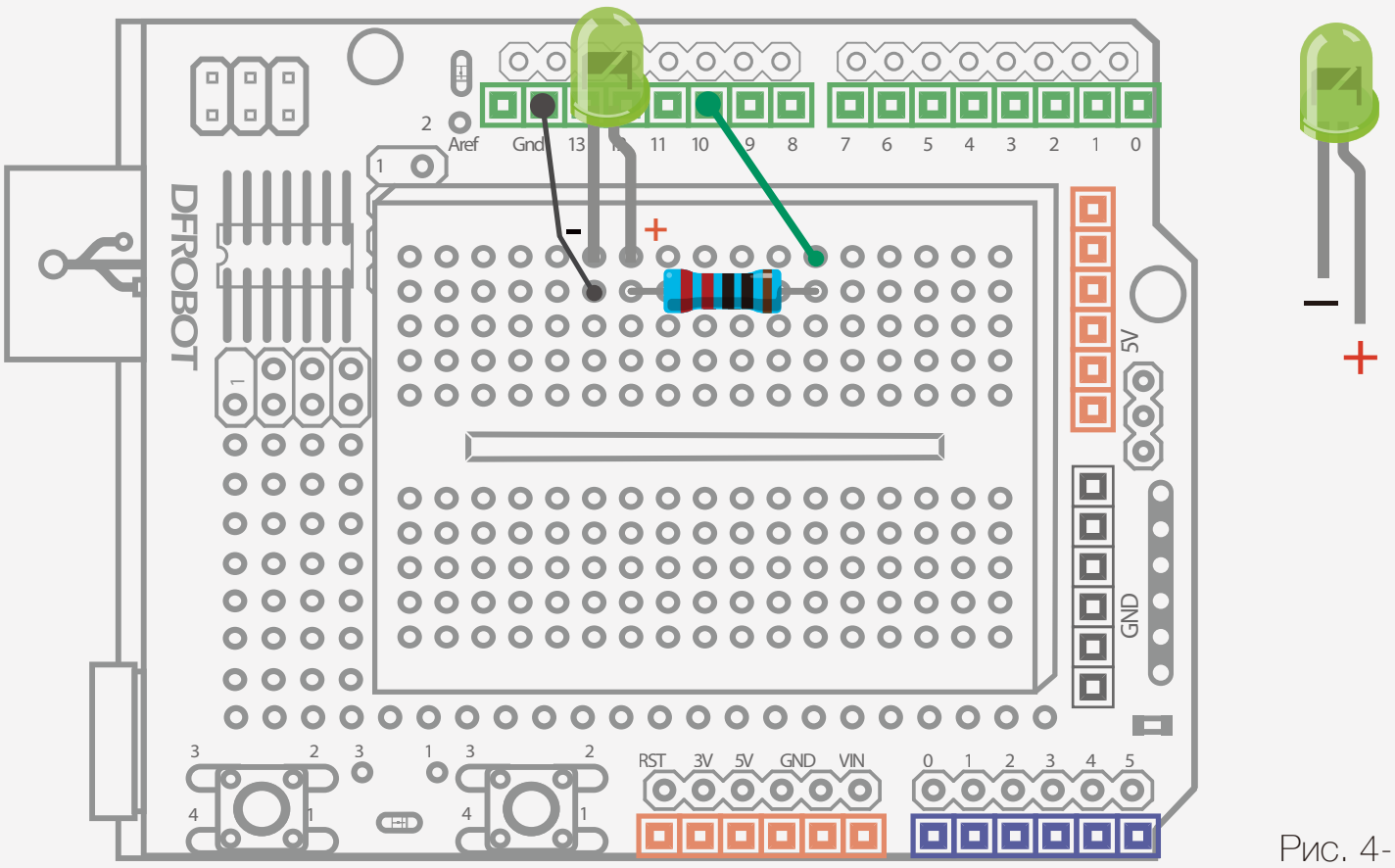


Рис. 4-1 Схема “дыхания” светодиода

Код Arduino

Пример кода 4-1 :

```
//Project 4
int ledPin = 10;

void setup() {
    pinMode(ledPin,OUTPUT);
}

void loop(){
    fadeOn(1000,5);
    fadeOff(1000,5);
}

void fadeOn(unsigned int time,int increament){
      for (byte value = 0 ; value < 255; value+=increament){
        analogWrite(ledPin, value);
        delay(time/(255/5));
        }
}

void fadeOff(unsigned int time,int decreament){
       for (byte value = 255; value >0; value-=decreament){
        analogWrite(ledPin, value);
        delay(time/(255/5));
        }
}
```

Вы увидете, как светодиод становится ярче и постоянно затухает после загрузки кода.

Код

С большей частью кода, мы уже знакомы, а именно, инициализацией объявлений переменных, настройкой пинов, настройкой цикла «for», а также вызова функции.

В основном коде мы используем только 2 функции. У вас будет четкое представление после проверки одного из них, как показано ниже.

```
void fadeOn(unsigned int time,int increament){

   for (byte value = 0 ; value < 255; value+=increament){

     analogWrite(ledPin, value);
     delay(time/(255/5));
     }
}
```

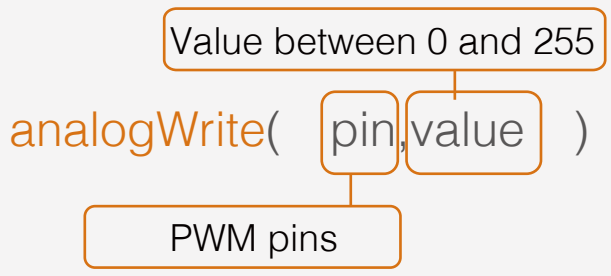
Функция fadeOn () имеет 2 параметра, «int time» для времени и «int increment» для возрастающих значений. Существует список for(), который повторяет программу. Условие «значение(value) <255», а нарастание яркости определяется путем увеличения.

Это новая команда в функции «for ()».

analogWrite(ledPin, value)

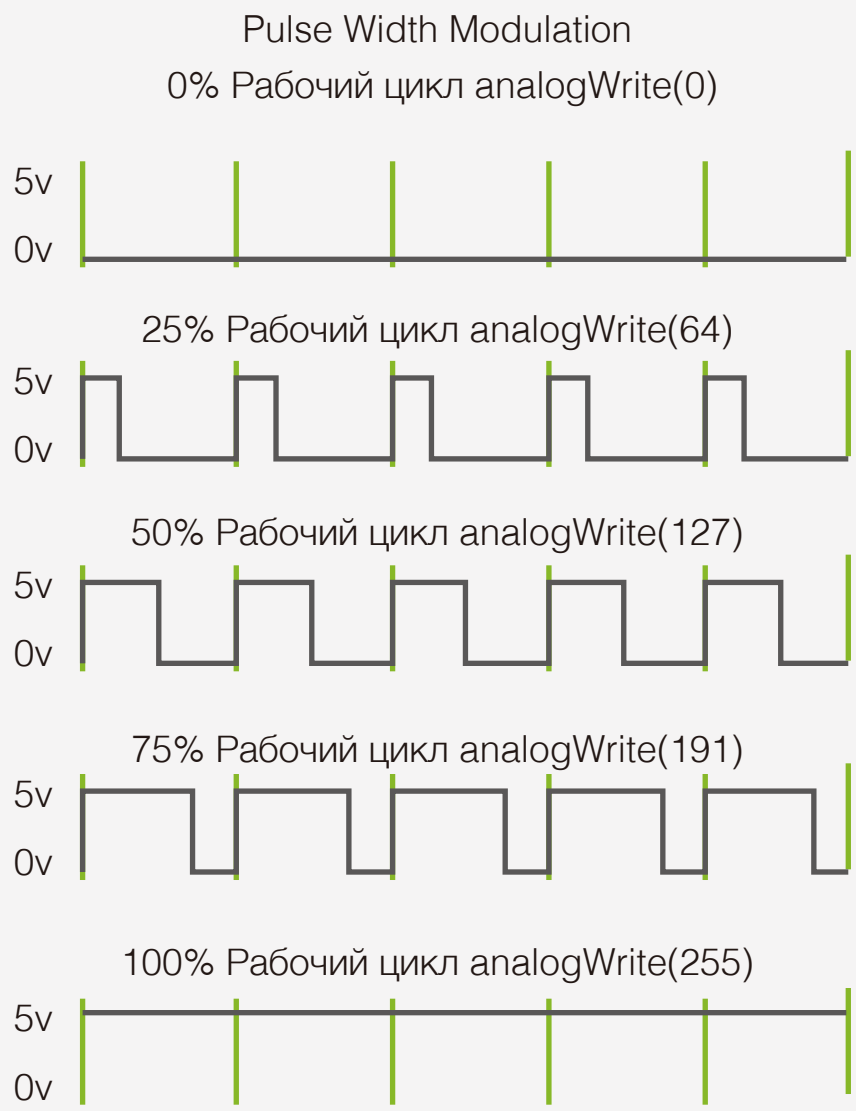
Как мы можем отправлять аналоговые значения к цифровому выводу? Мы используем пины, отмеченные знаком «~» на конце, такие как D3, D5, D6, D9, D10 и D11, для вывода переменного количества мощности на светодиод. Эта технология контроля мощности известна как «широтно-импульсная модуляция» или сокращенно ШИМ.

Формат команды analogWrite выглядит следующим образом:



Функция analogWrite () означает присвоение значения ШИМ аналогового значения от 0 до 255.

Примерно каждые 1/500 секунды пины ШИМ выводят импульс с использованием цифровых сигналов. Контролируя длину включения и выключения, он создает эквивалентный эффект от напряжения от 0 до 5 вольт. Длительность импульса называется «ширина импульса», поэтому PWM (ШИМ) относится к широтно-импульсной модуляции.
Давайте подробнее рассмотрим ШИМ.



Зеленая линия - это цикл импульса. Процент длины высокого напряжения и низкого напряжения соответствует значению функции analogWrite () известен как «Рабочий цикл» (“Duty Cycle”). Рабочий цикл первого импульса равен 0. Таким образом, значение равно 0, а яркость светодиода равна 0, что эквивалентно выключению. Чем дольше сигнал HIGH, тем ярче светодиод. Рабочий цикл последнего импульса составляет 100%. Таким образом, значение равно 255, а яркость светодиода - 255. Аналогично, 50% составляет половину яркости, а 25% относительно тускнеет.

ШИМ широко используется для управления яркостью света. Мы также используем его для управления скоростью вращения двигателей, например, колес транспортных средств, приводимых в движение двигателями.

Эта глава окончилась! Хотя мы использовали то же оборудование, что и в Проекте 1, Arduino запускает другую программу, что приводит к совершенно другому эффекту.Мир Arduino захватывает, надеемся, для вас он тоже увлекателен!

упражнение

1. Создайте мерцающий эффект пламени с помощью светодиодов, контролируя значение ШИМ в случайном порядке. Накройте его бумагой, и ночью она станет маленькой лампой.
Материалы:
1x красный светодиод
2x желтых светодиода
1x резистор 220 Ом
Используйте «random ()» для настройки уровня яркости. Мы предлагаем вам сначала инициализировать уровень яркости и изменить его в пределах случайного значения, например, (120) +135. Таким образом, светодиод может меняться в небольшом количестве, как свет пламени!

Вы можете найти источники в приведенных ниже ссылок для получения дополнительных объяснений по различным командам.
https://www.arduino.cc/en/Reference/HomePage

2. Попробуйте более сложный проект: Управляйте светодиодом двумя кнопками, чтобы один сделать ярче, а другой - более тусклым.

Источник: http://www.geek-workshop.com/thread-1054-1-1.html

Проект 5

Цветной RGB-светодиод

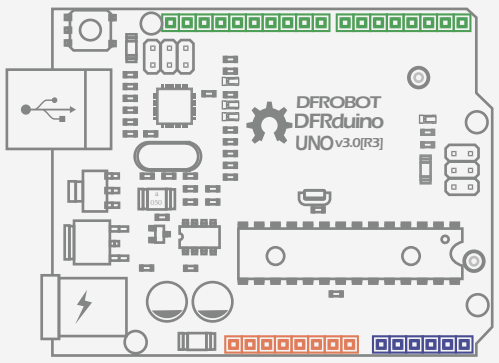
05

DFROBOT
DRIVE THE FUTURE

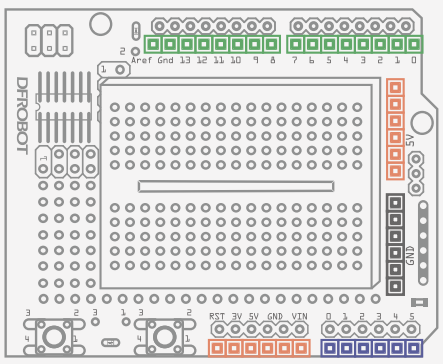
Цветной RGB-светодиод

Начнем с нового компонента: RGB-светодиода. Этот компонент сочетает в себе красный, синий и зеленый светодиоды и может отображать различные цвета, настраивая различные значения каждого света. Компьютерный монитор использует много RGB- светодиодов для отображения изображения. Мы научимся создавать разные цвета с помощью RGB-светодиода в этом уроке.

Компоненты



DFduino UNO R3 x1



Плата расширения x1

Перемычки “мама-мама” x4

Резистор 220 Ом x3

RGB-светодиод x1

Цепь

Перед построением схемы попробуйте определить, является ли ваш RGB -светодиод общим катодом или общим анодом. Если вы не знаете, как это сделать, перейдите к последней части этого урока. В этом проекте мы предполагаем, что вы используете обычный RGB-светодиод катода.

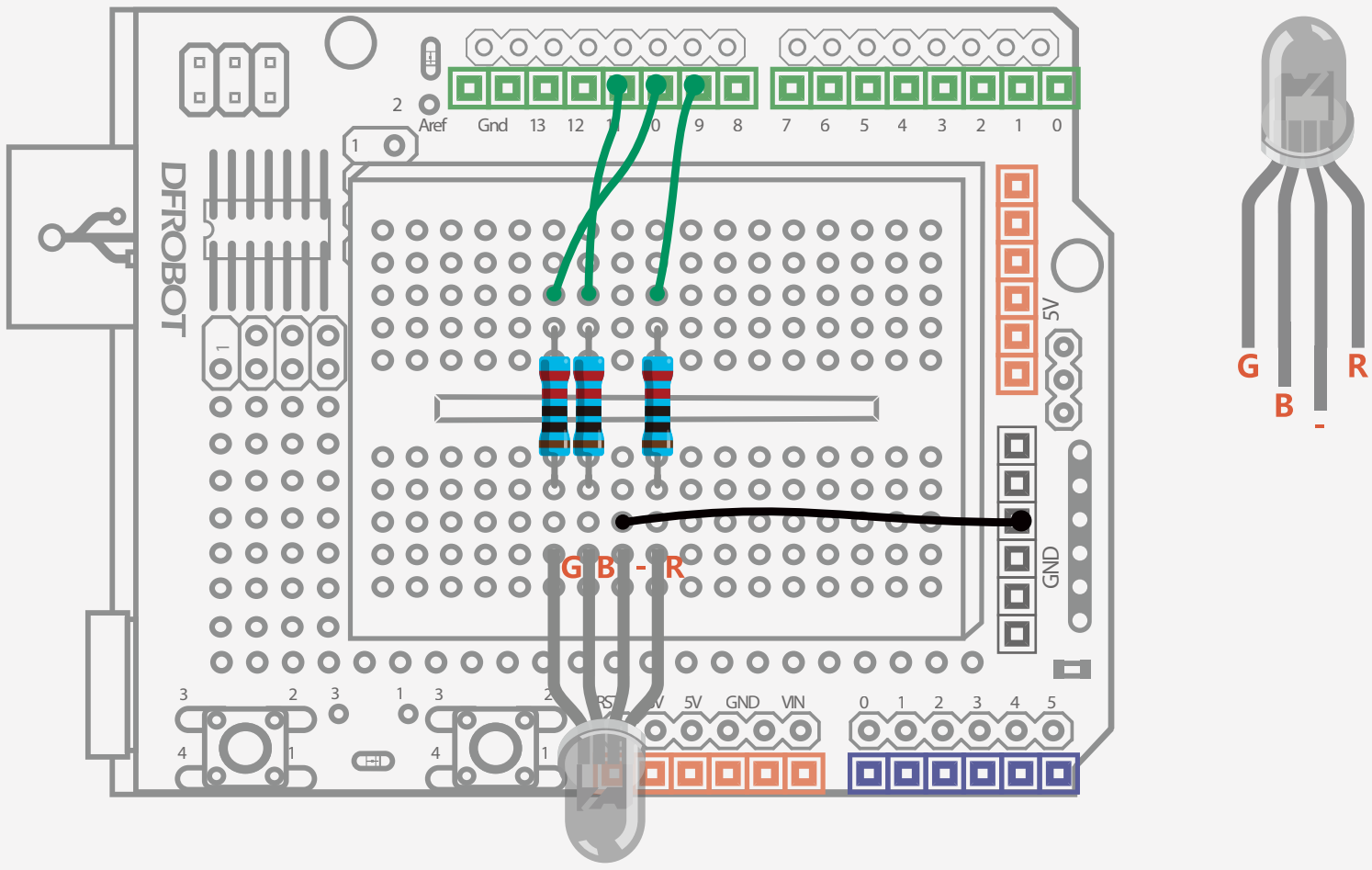


Рис. 5-1. Схема цветного RGB-светодиода

Пример кода 5-1

//ПРОЕКТ 5 RGB-СВЕТОДИОД :

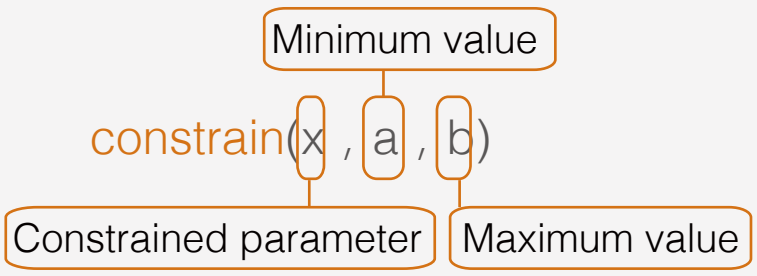
```
//PROJECT 5 RGB LED
int redPin = 9;
int greenPin = 10;
int bluePin = 11;

void setup(){
    pinMode(redPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(bluePin, OUTPUT);
}

void loop(){
     //R:0-255 G:0-255 B:0-255
     colorRGB(random(0,255),random(0,255),random(0,255));
    delay(1000);
}

void colorRGB(int red, int green, int blue){
        analogWrite(redPin,constrain(red,0,255));
        analogWrite(greenPin,constrain(green,0,255));
        analogWrite(bluePin,constrain(blue,0,255));
}
```

После загрузки этого кода вы должны увидеть RGB-светодиод со случайными цветами.

Код

Во-первых, мы подключим 3 светодиода, содержащиеся в RGB-светодиоде, на 3 ШИМ-пина, чтобы мы могли настроить их на разные цвета, объявив 3 контакта в OUTPUT. Основная часть этой программы заключается в создании новой команды: colorRGB (), которая имеет 3 параметра для установки значения красного, зеленого и синего света между 0 и 255.
Таким образом, когда мы хотим настроить цвет, мы можем просто назначить значения этой команде, а не постоянно повторять команду analogWrite ().

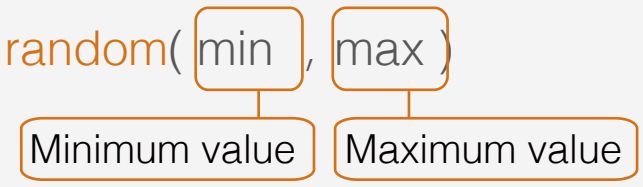
Здесь мы введем constrain () и random ().
Попробуйте найти их с помощью сайтов, о которых мы упоминали в последнем домашнем задании, и посмотрим, сможете ли вы их понять.
Формат функции ограничения:



Для функции «constrain ()» требуются три параметра: x, a и b.«x» здесь является номером ограничения, «a» является минимумом, а «b» - максимальным.Если значение меньше «a», оно вернется к a. Если оно больше, чем «b», оно вернется к «b».

Красный, зеленый и синий - наши ограниченные параметры. Они ограничены между 0 и 255 (что попадает в диапазон значений ШИМ). Значения генерируются случайным образом, используя функцию «random ()».

Формат «random ()» выглядит следующим образом:



Первой переменной этой функции является минимальное значение, а второе - максимальное. Поэтому в этой программе мы настраиваем как «random (0,255)».

Компоненты

RGB-светодиод

RGB-светодиод имеет четыре провода. Если вы используете RGB-светодиод с общим катодом, есть один вывод, который приведет к положительному соединению каждого из одиночных светодиодов и одному выходу, который подключен ко всем трем отрицательным сторонам светодиодов. Вот почему он называется общим катодом. Внешне нет никакой разницы между общим катодными и общими анодами RGB-светодиодами, однако вам нужно обратить внимание при назначении значений цвета. Например, для общего катода красный RGB-светодиод B-0. Для общего анодного RGB-индикатора красный цвет R-0, G-255, B-255. Как мы можем настроить RGB-светодиод на разные цвета?
Устанавливая разные значения уровней яркости 3 основным цветам, используя функцию «analogWrite (значение)», вы можете настроить любой цвет, который вам нравится!

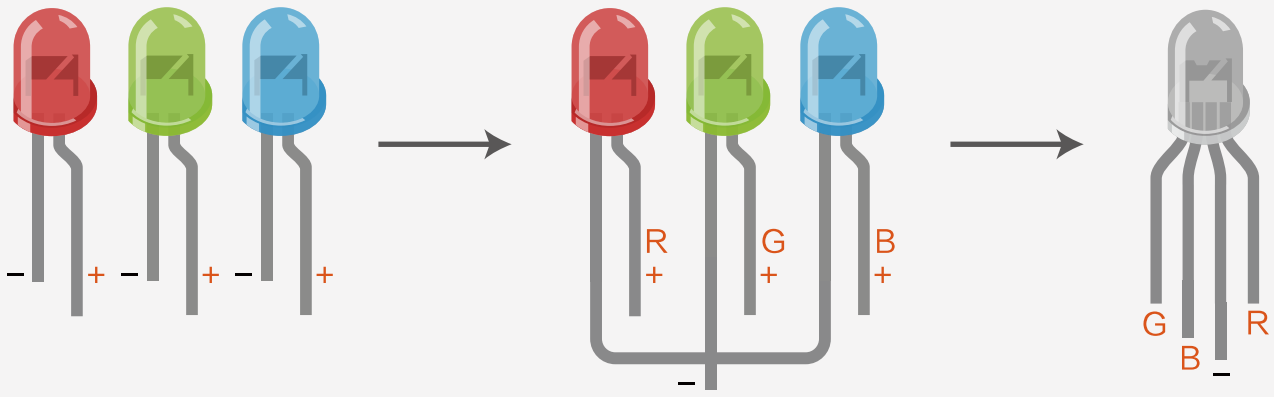


Fig. 5-2 How 3 LEDs form A RGB LED (общими)

Рис. 5-3. Повторное смешиваниекрасного, зеленого и синего для достижения различных цветов

Вы можете настроить цвета 255x255x255 (16777216), назначив разные значения ШИМ на этих 3-х светодиодах!

Красный	Зеленый	Синий	Цвет
255	0	0	Красный
0	255	0	Зеленый
0	0	255	Синий
255	255	0	Желтый
0	255	255	Бирюзовый
255	0	255	Пурпурно красный
255	255	255	Белый

Рисунок 5-1. Цвета, генерируемые из комбинированных значений ШИМ различных светодиодов

Разница между общими анодами и общими катодными светодиодами RGB

В чем разница между общим анодом и общим катодом RGB-светодиодах в приложении? Согласно приведенному ниже рисунку, нет различия между общим анодом и катодом с точки зрения их внешнего вида. Однако в их применении есть два ключевых отличия:

(1) Различные соединения: для общего анода порт должен быть подключен к 5 В, но не GND, иначе светодиод не загорится.

(2) Совместимость цветов: общий анод RGB LED полностью отличается от RGB-светодиода.

RGB-светодиод декодирует в обратном порядке: «R-0, G-255 и B-255».

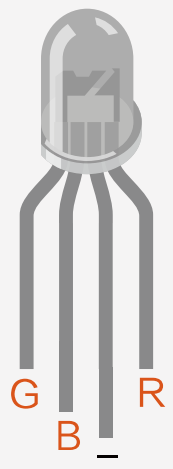


Рис. 5-4 Схема катодного RGB-светодиода

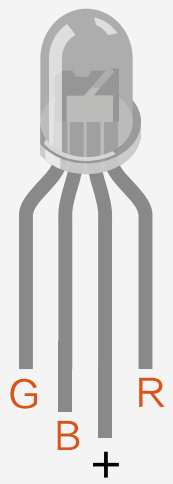


Рис. 5-5 Схема анодного RGB-светодиода

Упражнение

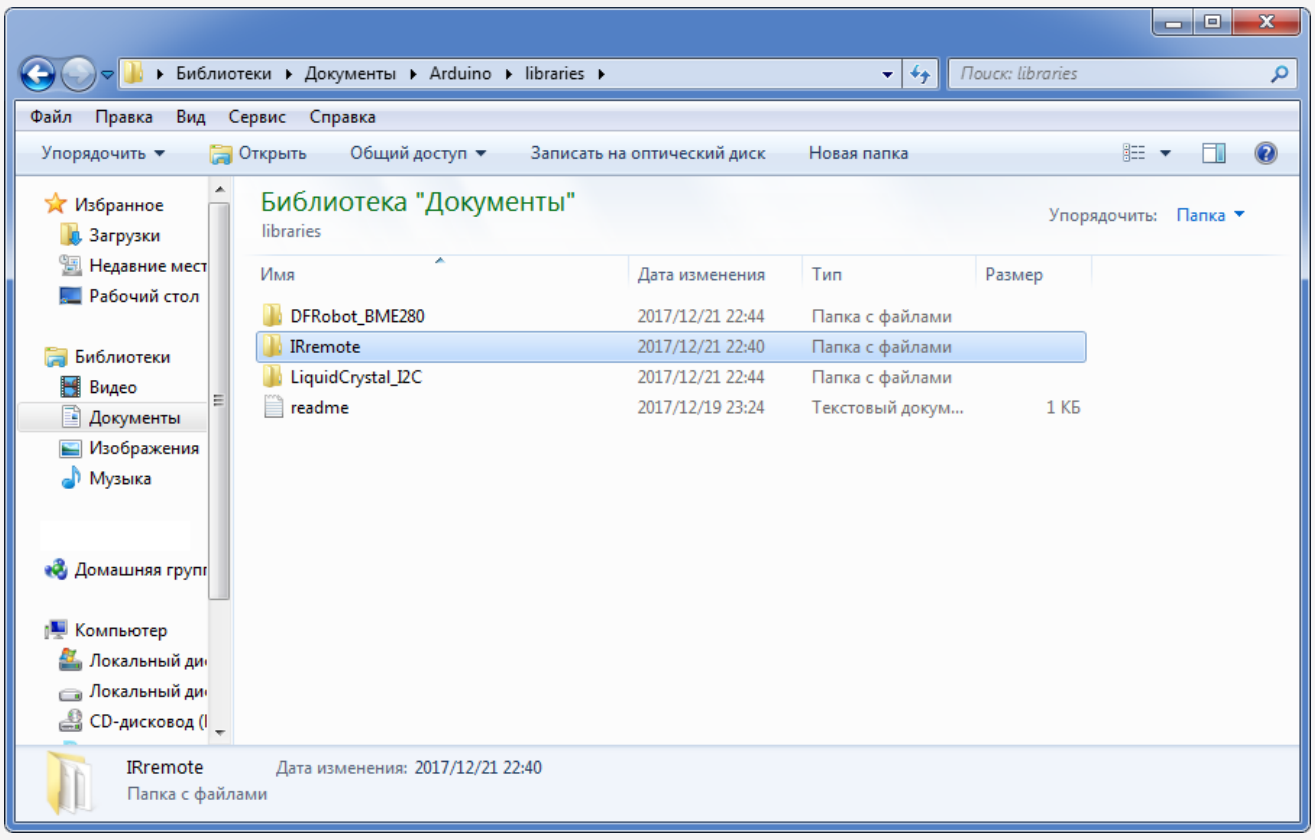
1. На основе вышеприведенной программы создайте последовательность изменения цвета. Попробуйте настроить различные цвета, играя со значениями.
СОВЕТ. Вам нужно изменить переменные «colorRGB ()».

2. Создайте радужную последовательность светодиодов и сделайте каждый цвет затухающим, чтобы переход между каждым ними стал более плавным.

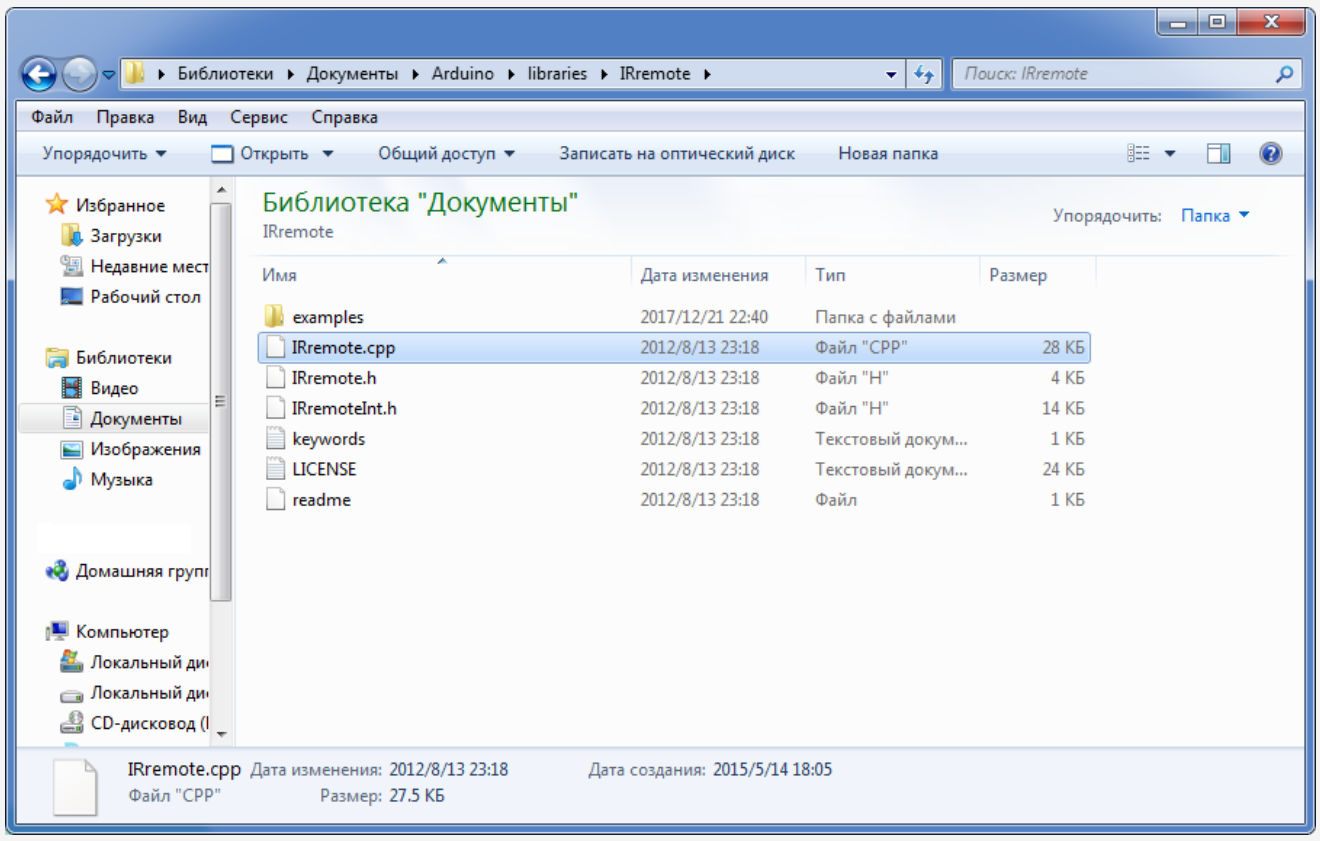
Как загружать библиотеку

Загрузите библиотеку из Интернета и разархивируйте ее в каталог «libraries» внутри каталога IDE Arduino на вашем компьютере.

Помните, что внутри папки будет «.cpp» файл, «.h» и часто файл «keywords.txt». Убедитесь, что они находятся в одном каталоге, иначе среда разработки Arduino не распознает библиотеку.



Затем откройте программу в примере. Проверьте, нужно ли вам менять контакты в программе, если это необходимо.



Проект 6

Сигнализация

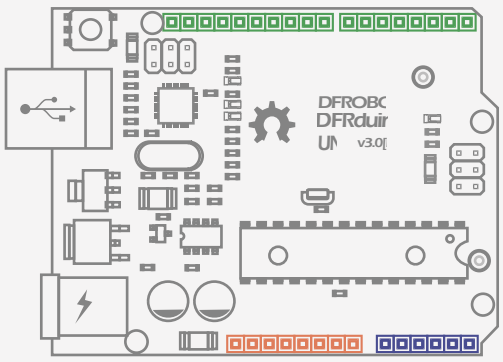
06

DFROBOT
DRIVE THE FUTURE

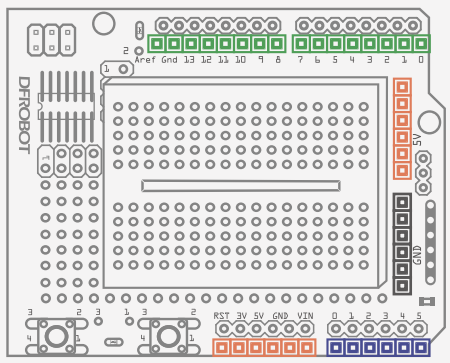
Сигнализация

Испытаем новый компонент: зуммер! Он генерирует звуки разных частот с использованием синусоидальных волн. Если вы подключите светодиод с той же синусоидальной волной, вы можете сделать свой собственный сигнал.

Компоненты

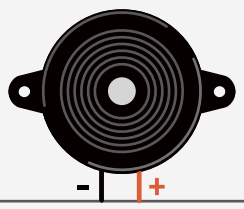


DFduino UNO R3 x1



Плата расширения x1

Перемычки “мама-мама” x2



Зуммер x1

Аппаратное соединение

Выполните следующие подключения. Обратите внимание, что длинная нога на зуммере положительная, а более короткая нога - отрицательная. Подключите отрицательный провод к GND (земле) и положительный вывод к контакту 8..

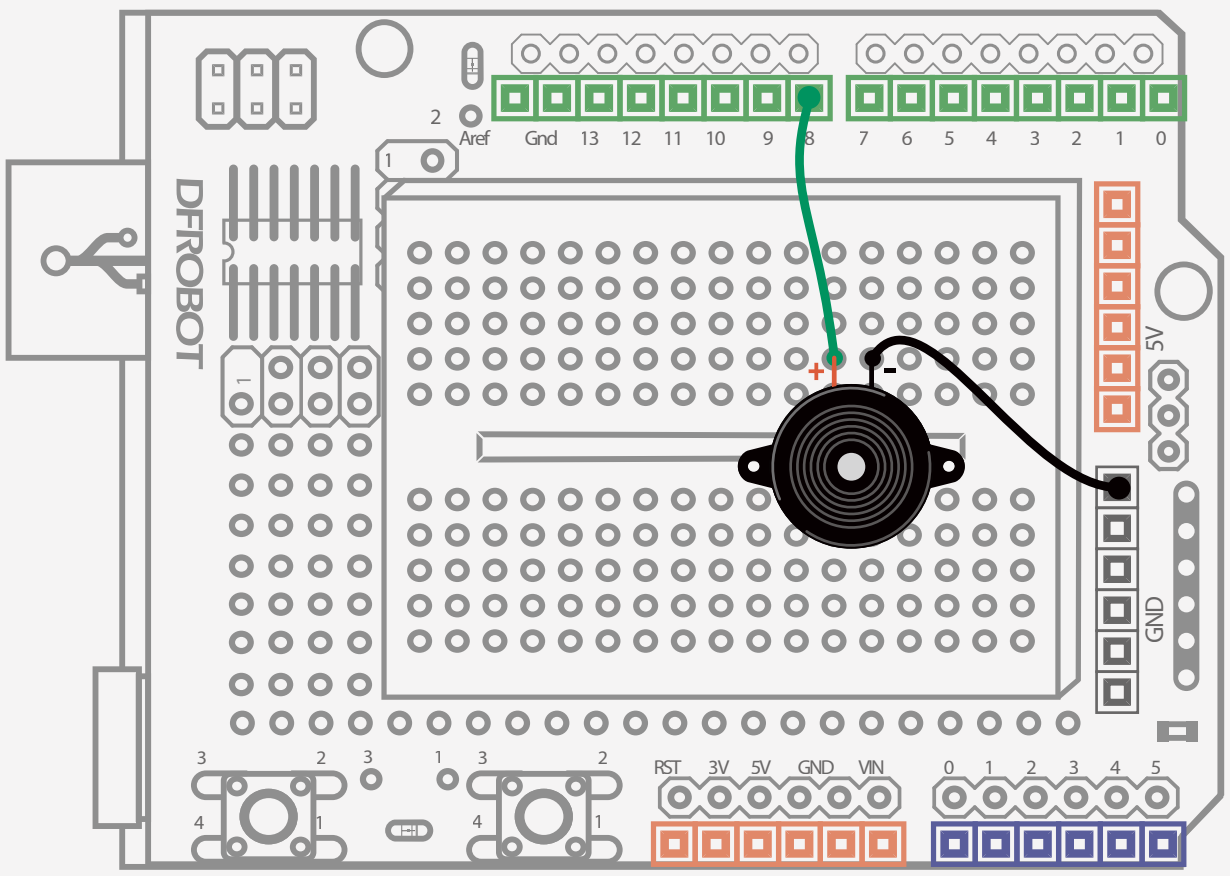


Рис. 6-1 Схема будильника

Код

Пример кода 6-1 (из “Beginning Arduino”)

Пример кода 6-1:

```
//Проект 6 Сигнализация
float sinVal;
int toneVal;

void setup(){
    pinMode(8, OUTPUT);
}

void loop(){
    for(int x=0; x<180; x++){
        // преобразование синусоидального угла в радианную меру
        sinVal = (sin(x*(3.1412/180)));
        // генерирование звука разных частот по синусоидальной величине
        toneVal = 2000+(int(sinVal*1000));
        // Установка частоты для 8 пин-выхода
        tone(8, toneVal);
        delay(2);
    }
}
```

После загрузки кода вы можете услышать сигнал высокого и низкого уровня.

Код

Сначала определите две переменные:

```
float sinVal;
int toneVal;
```

«Float» - это тип данных для чисел с плавающей запятой (число, которое имеет десятичную точку).

Числа с плавающей запятой часто используются для приближения аналоговых и непрерывных значений, поскольку они имеют большее разрешение, чем целые числа. Здесь мы используем переменную «float» для хранения синусоидальных значений. Синусоидальная волна изменяется довольно равномерно в форме волны, выкачивая ее на звуковые частоты. Следовательно, toneVal получает значения из sinVal и преобразует его в частоты

```
for(int x=0; x<180; x++){}
```

Нам нужно использовать формулу «3.1412 / 180» », чтобы преобразовать значение угла из градусов в радианы, потому что единица« sin »представляет собой радиан вместо угла.

```
sinVal=(sin(x*(3.1412/180)));
```

Затем мы меняем это значение на частоту звукового сигнала:

```
toneVal = 2000+(int(sin-
Val*1000));
```

“sinVal "- это переменная типа float , значение с десятичной точкой. Мы не хотим, чтобы наша частота имела десятичную точку, поэтому нам нужно изменить ее значение на целое, написав команду, как показано ниже:

```
int(sinVal*1000)
```

Человеческие уши могут слышать звук с частотой от 20 Гц до 20 кГц, поэтому мы умножаем значение raw на 1000 и прибавляем 2000, чтобы присвоить значение для «tone-Val» и получить диапазон от 2000 до 3000.

```
tone(8, toneVal);
```

Здесь мы вводим три функции, относящиеся к тону:

1、tone(pin,frequency)

«Pin» - это цифровой пин, подключенный к зуммеру. Значение частоты в Гц.

2、tone(pin,frequency, duration)

Параметр «duration» измеряется в миллисекундах. Если не будет длительности, зуммер будет продолжать звучать на разных частотах.

3、noTone(pin)

Функция «noTone (pin)» заключается в том, чтобы закончить звук с определенного пина.

Компоненты

Зуммер

Зуммер - это электронный компонент, который может генерировать звук. Обычно существуют два типа: пьезо-зуммеры и магнитные зуммеры.

Разница между активными и пассивными зуммерами

Пьезо и магнитные зуммеры далее подразделяются на два типа: активные и пассивные зуммеры. Основное различие заключается в разных требованиях к их входному сигналу. В этом случае «активные» и «пассивные» относятся не к источникам питания, а к источникам колебаний.Активный зуммер имеет свой собственный источник колебаний - он жужжит, когда он включенАктивный зуммер имеет простую схему генератора, которая изменяет постоянного тока на импульсный сигнал определенной частоты. Активные зуммеры содержат специальную пленку под названием «молибден», магнитное поле от колебания зуммера. После включения питания он начинает звучать.

Активные зуммеры не поляризованы, что означает, что вы можете подключить их любым способом, и они будут работать.

В этот комплект включены активные магнитные зуммеры.

Пассивный зуммер не имеет собственного осциллятора, поэтому ему нужно использовать квадратную волну от 2 кГц до 5 кГц, чтобы запускать ее вместо простого использования постоянного тока.Пассивные зуммеры поляризованы, поэтому их нужно подключить правильно: они имеют длинный провод (анод) и короткий провод (катод)

Для новичков с пассивными зуммерами легче работать.

Если вы хотите исследовать зуммеры дальше, вот некоторые идеи проекта:Пассивные звуковые сигналы хороши для различных музыкальных эффектов.

Существует множество приложений. Например, зуммеры используются в инфракрасных, ультразвуковых датчиках в мониторинге для оповещения о приближающихся объектах; в температурных датчиках для сигнализации о повышении температуры; в газовых датчиках для сигнализации об утечки газа. Помимо сигналов тревоги, зуммеры также могут использоваться в качестве музыкальных инструментов, использующих разные частоты для формирования нот.Зуммеры потрясают, не так ли?

Упражнение

Создать сигнлизацию с красным светодиодом.
1. Настройте нашу цепь так, чтобы светодиод включался в унисон с функцией «sin», чтобы интенсивность света менялась со звуком.

2. Сможете ли вы сделать дверной звонок используя полученные знания из Проекта 3? Звуковой сигнал должен звучать, когда кнопка нажата.

Проект 7

Сигнализатор температуры

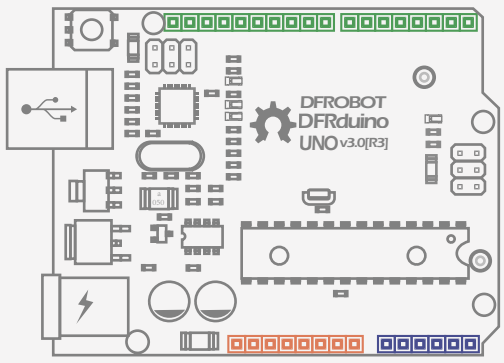
07

DFROBOT
DRIVE THE FUTURE

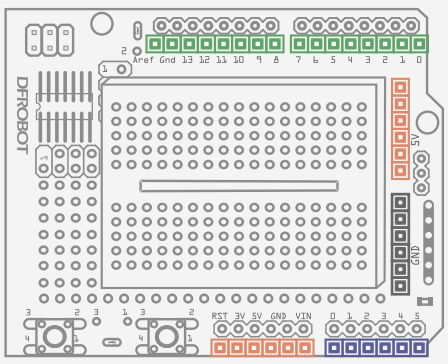
Сигнализатор температуры

В предыдущей цепи мы добавили датчик температуры, для того чтобы зуммер издавал звук, когда температура достигнет определенного градуса. Это наш первый проект с использованием исполнительного механизма, реагирующего на датчик.

Компоненты

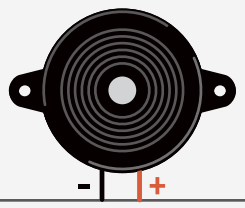


DFduino UNO R3 x1



Плата расширения x1

Перемычки “мама-мама” x5



Зуммер x1

Датчик температуры x1

На основе цепи Проекта 6 подключите датчик температуры к LM35, как показано ниже. Штыри подключены к 5 В, Аналоговый вывод 0 и GND (земля).

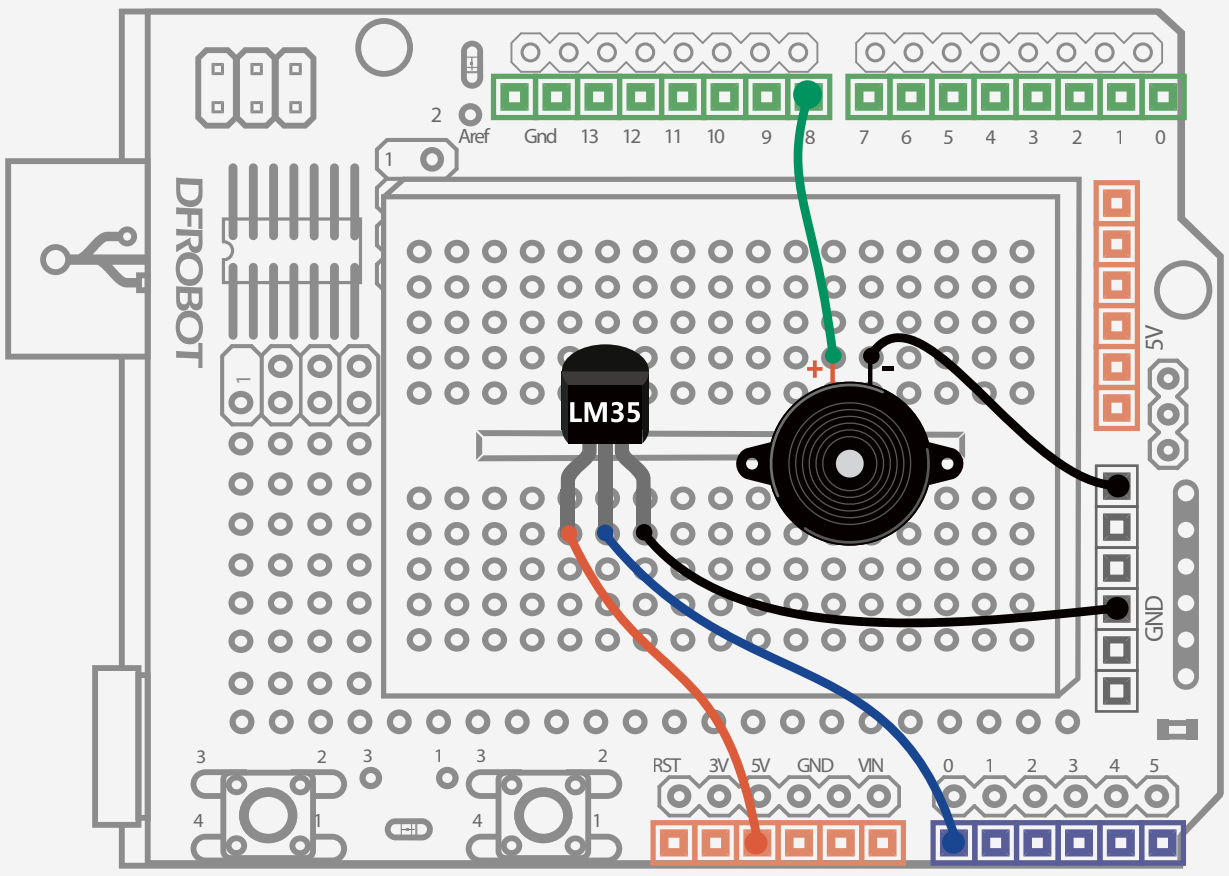


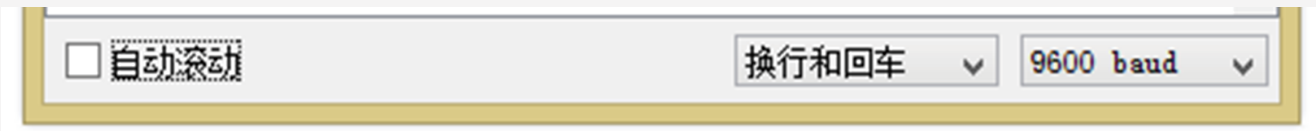
Рис. 7-1 Схема температурной тревоги

Пример кода 7-1 :

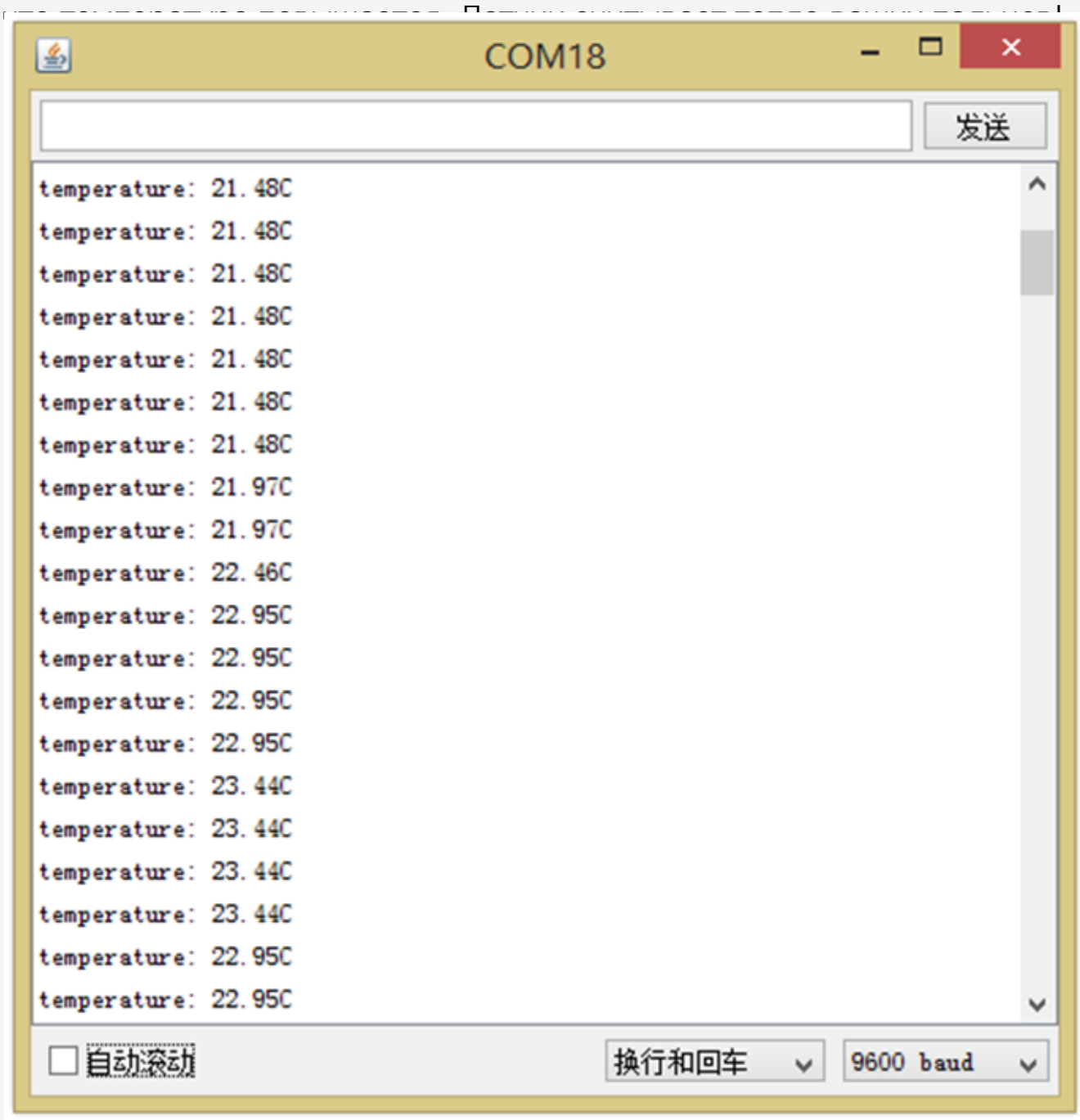
```
//Проект 7 Температурная тревога
float sinVal;
int toneVal;
unsigned long tepTimer ;

void setup(){
   pinMode(8, OUTPUT);              // настроить пин зуммера
   Serial.begin(9600);               //настраиваем скорость передачи до
9600 бит / с
}
void loop(){
   int val;                        //сохраняем значение LM35
   double data;                    //сохраняем значение LM35
   val=analogRead(0);        //сохранить преобразованное значение
температуры
   data = (double) val * (5/10.24);    // Преобразование значения
напряжения в значение температуры
   if(data>27){   //Если температура выше 27, зуммер начинает звучать
        for(int x=0; x<180; x++){
          //Преобразование функции sin в радиан
          sinVal = (sin(x*(3.1412/180)));
          //Используйте функцию sin для генерации частоты звука
          toneVal = 2000+(int(sinVal*1000));
          //Настройка пина 8
          tone(8, toneVal);
          delay(2);
        }
    } else {                              // Если температура ниже 27,
выключите зуммер
        noTone(8);                        //Отключить зуммер
    }
    if(millis() - tepTimer > 50   // Каждые 500 мс, последовательный порт
(Serial Port) выводит значение температуры.
          tepTimer = millis();
          Serial.print("temperature: "); // Последовательный порт выводит
температуру
          Serial.print(data);                // Последовательный порт выводит
значение температуры
          Serial.println("C");               // Последовательный порт выводит
единицу измерения
      }
}
```

После успешного добавления кода откройте Serial Monitor Arduino IDE.

Set the baud rate 9600.



Прочитайте значение температуры из последовательного порта (Serial Port). Если вы положите пальцы на датчик LM35, вы сразу обнаружите,



В соответствии с программой, когда температура достигает +27°C – начинает звучать зуммер. Если температура опускается ниже +27°C – зуммер останавливается.

Большинство вышеперечисленных кодов такие же, что и в Проекте 6. Почти все синтаксисы были упомянуты в предыдущих проектах. Надемся, у вас теперь есть представление о переменных и функциях.

Мы инициализируем 3 переменные в верхней части программы.

```
float sinVal;
int toneVal;
unsigned long tepTimer ;
```

Третья переменная «tepTimer» представляет собой тип данных unsigned long для хранения значений времени и исходящей температуры из последовательного порта.Почему “unsigned long”?Так как машина будет работать относительно длительное время, мы выберем длинное целое число, и поскольку оно не может хранить отрицательные числа, оно будет неподписанным.Почему в первой строке функции «setup ()» мы настраиваем только зуммер, как выходной режим и игнорируем температурный датчик LM35?LM35 использует аналоговые значения.

Аналоговые значения не нужно настраивать для «pinmode». «PinMode» используется только для цифровых контактов.

Существует множество функций для связи по последовательному порту:

```
Serial.begin(9600);
```

Эта функция предназначена для инициализации скорости передачи (скорости передачи данных) последовательного порта. Обычно настройка неисправности в серийном мониторе работает для большинства приложений, но некоторые беспроводные модули имеют конкретную потребность в скорости передачи данных.В функции «loop ()» мы объявляем две переменные: «val» и «data» наверху. Это переменные в ограниченном объеме, поэтому они запускаются только внутри отдельного блока кода.

```
val=analogRead(0);
```

Это новая функция «analogRead (pin)».

```
analogRead(pin)
```

Эта функция считывает значение с указанного аналогового вывода. Цифровые контакты в Arduino подключены к 10-байтовому аналого-цифровому преобразователю, поэтому напряжение между 0 и 5 В преобразуется в значение от 0 до 1023. Каждое значение соответствует значению напряжения. Здесь значение напряжения считываемой температуры выдает диапазон от 0 до 1023. Каждые 10 мВ соответствуют 1 градусу для датчика температуры LM35.

```
data = (double) val * (5/10.24);
```

Из значения напряжения, считываемого через датчик, диапазон составляет от 0 до 1023. Таким образом, мы делим его на 1024 части и умножаем результат на 5, чтобы преобразовать его в значение напряжения. Поскольку 10 мВ соответствует 1 степени, нам нужно умножить это, чтобы получить значение температуры в двойном типе данных и присвоить его переменной данных.

Последовательный порт

Последовательный порт позволяет Arduino взаимодействовать с внешним миром, передавая и получая данные. В каждом микроконтроллере Arduino имеется по меньшей мере один последовательный порт, отдельно подключенный к цифровому выводу 0 (прием RX / данных) и аналоговый вывод 1 (передача TX / передача данных). Цифровой контакт 1 и 0 нельзя использовать для функции ввода / вывода при использовании последовательного порта. Вы загружаете код в Arduino через последовательный порт. При загрузке кода USB будет подключать цифровой вывод RX и аналоговый вывод TX. Пины RX и TX не могут принимать другие сигналы во время этого, или будут помехи. Лучшим способом использования этих двух контактов является вставка компонентов после загрузки кода.

В следующей программе мы вычисляем условие с помощью конструкции “if/else”.

Формат «if / else»:

```
if (expression) {
  Statement 1;
} else{
   Statement 2;
}
```

Эта функция называется условной. Она работает следующим образом: Выделено выражение. Если условия выражения верны, выполняется условие 1, и положение 2 пропускается. Если выражение ложно, выполняется положение 2, а положение 1 пропускается. Любое положение 1 или 2 должно выполняться, но одновременное выполнение запрещено. Проще говоря, это Arduino принимает решение между двумя предопределенными переменными.

```
if(data>27){
 for(int x=0; x<180; x++){
      ……
  }
} else {
      ……
}
```

Если температура выше +27˚C, программа запускает первую часть программы, следуя инструкциям if, чтобы активировать зуммер. Если нет, она запускает конструкцию else, чтобы остановить зуммер. Помимо обнаружения изменения температуры для нашей тревоги, нам также необходимо отобразить температуру, которую Arduino считывает через последовательный порт. Нам нужно снова использовать функцию «millis ()» для отправки данных каждые 500 мс. (более подробную информацию см. в Проекте 3). После того, как последовательный порт получил данные, как мы можем отображать его на Serial Monitor?

```
Serial.print(val);
Serial.println(val);
```

print () работает для преобразования «val» в считываемый ASCII-формат (стандартный текст), выводимый из последовательного порта.

Существуют различные форматы для этой функции:
1. номера выводятся как число например: Serial.print (78); выход "78"
2. Выходы плавающего типа данных в виде числа с плавающей запятой максимум 2 цифры после десятичной точки например: Serial.print (1,23456); выходы "1.23"
3. Добавьте одинарный кавычек к символу и добавьте двойную кавычку в строку символов. например Serial.print («N»); выходы "N"

Serial.print(“Hello world.”);
outputs “Hello world.”

Другая общая команда - «Serial.write ()». Он не выводится в формате ASCII, а в байтовом формате. Проверьте ссылку на Arduino.cc, если вам интересно узнать больше

```
Serial.print(data);
```

Являются ли данные символьной строкой? Почему она выводит числа?Ответ заключается в том, что мы объявили переменную в функции настройки, чтобы назначить ей номер.

Компоненты

Датчик температуры LM35

LM35 является очень распространенным температурным датчиком с точностью до + 0,25 ° C.

Он имеет 3 пина

«+ Vs» - это мощность
«Vout» - выход напряжения
«GND» земля.
Формула расчета выглядит так:
Vout = 10 мВ / ˚C * T˚C (диапазон температур + 2 ° C ~ 40 ° C)

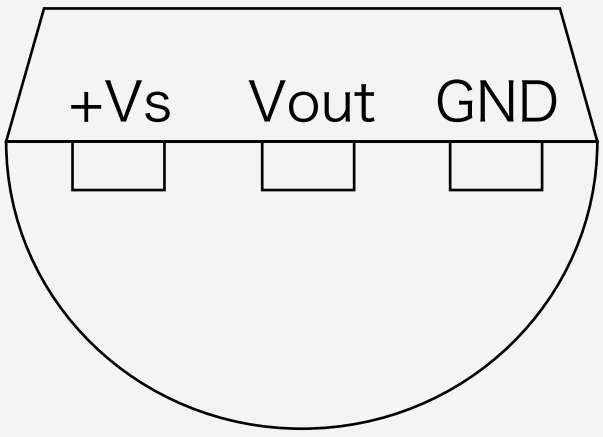
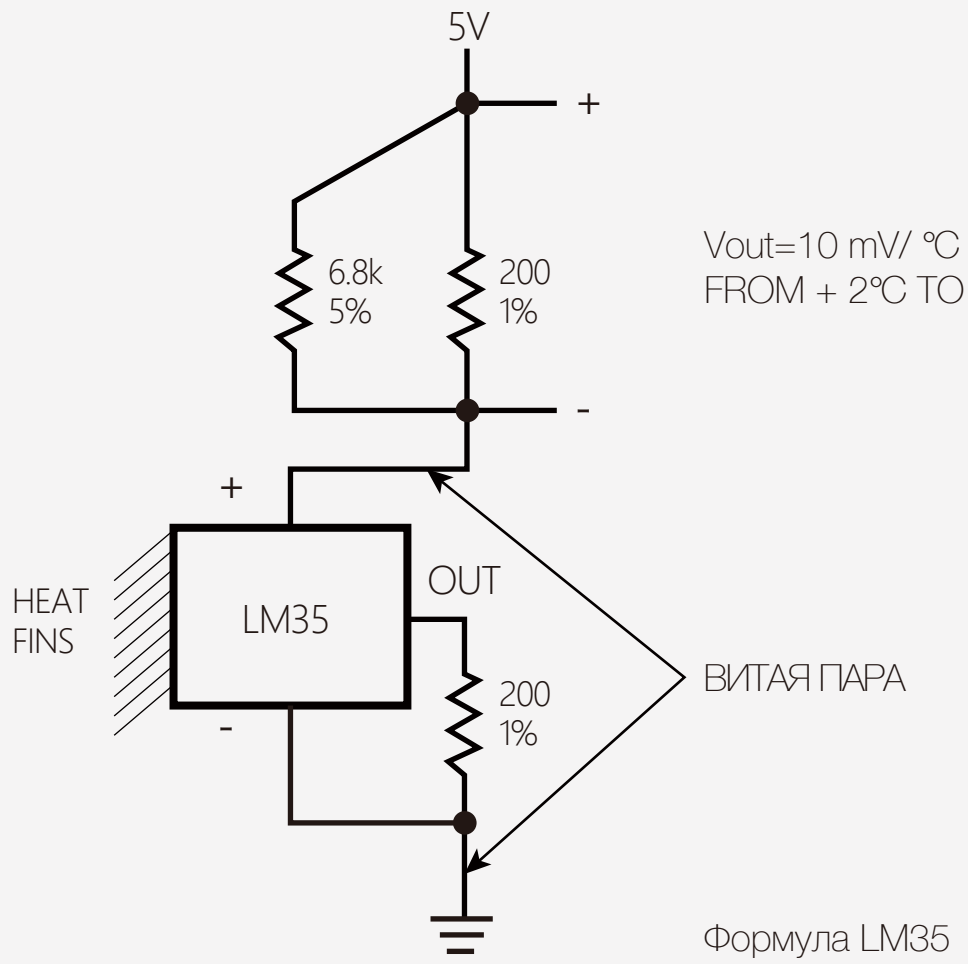


Схема пинов LM35

Если вы хотите узнать больше об этом компоненте, вы можете ознакомиться с листом данных. Он дает дополнительную информацию о методе преобразования температурных данных в напряжение.Полезный ресурс о технических особенностях компонентов можно найти здесь: http://www.alldatasheet.com/



Vout=10 mV/ ℃ (TAMBIENT+1 ℃)
FROM + 2℃ TO +40℃

Формула LM35

Упражнение

Добавьте светодиод в проект выше. Когда температура находится в определенном диапазоне, включите светодиод и произведите звук зуммера.
Вы можете установить разные цветные светодиоды и различные звуковые сигналы для разных температурных диапазонов
Например :
- Когда температура ниже +10°C или выше +35°C, загорается красный светодиод, и зуммер издаёт звук высокой частоты.

- Когда температура падает между +25°C и +35°C, загорается желтый светодиод, а зуммер производит звук низкой частоты.

- Когда температура падает между +10°C и +25°C, загорается зеленый светодиод, и зуммер выключается.

Проект 8

Датчик вибрации

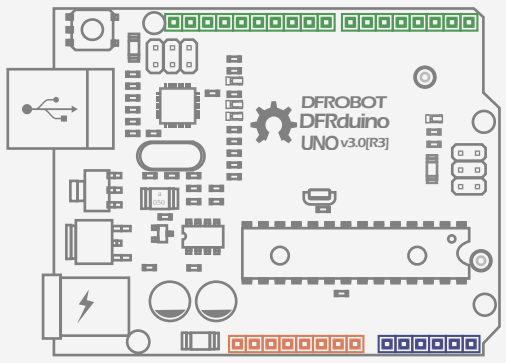
08

DFROBOT
DRIVE THE FUTURE

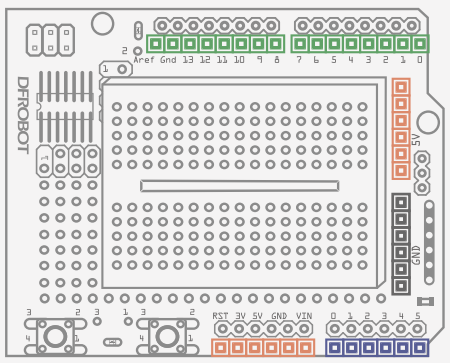
Датчик вибрации

В этом проекте мы будем использовать датчик наклона, входящий в комплект. Датчик наклона распознает основные движения и ориентацию в пространстве. Он содержит два контакта и небольшой металлический шар. Когда он удерживается в определенной ориентации, шар соединяет два контакта и завершает схему. Мы также добавили светодиод в этот проект. Когда датчик обнаруживает движение, светодиод HIGH (включен). Когда движение не обнаружено, светодиод LOW (отключен).

Компоненты



DFduino UNO R3 x1

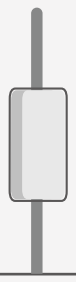


Плата расширения x1

Перемычки “мама-мама” x5

Светодиод 5ММ x1

Резистор 220 Ом x2



Датчик наклона x1

Цепь

Датчик наклона ведет себя так же, как кнопка. Вам необходимо добавить стягивающий резистор к датчику, чтобы убедиться, что цепь отключена, когда сигнал не обнаружен.
Вам также необходимо добавить к светодиоду резистор, ограничивающий ток.

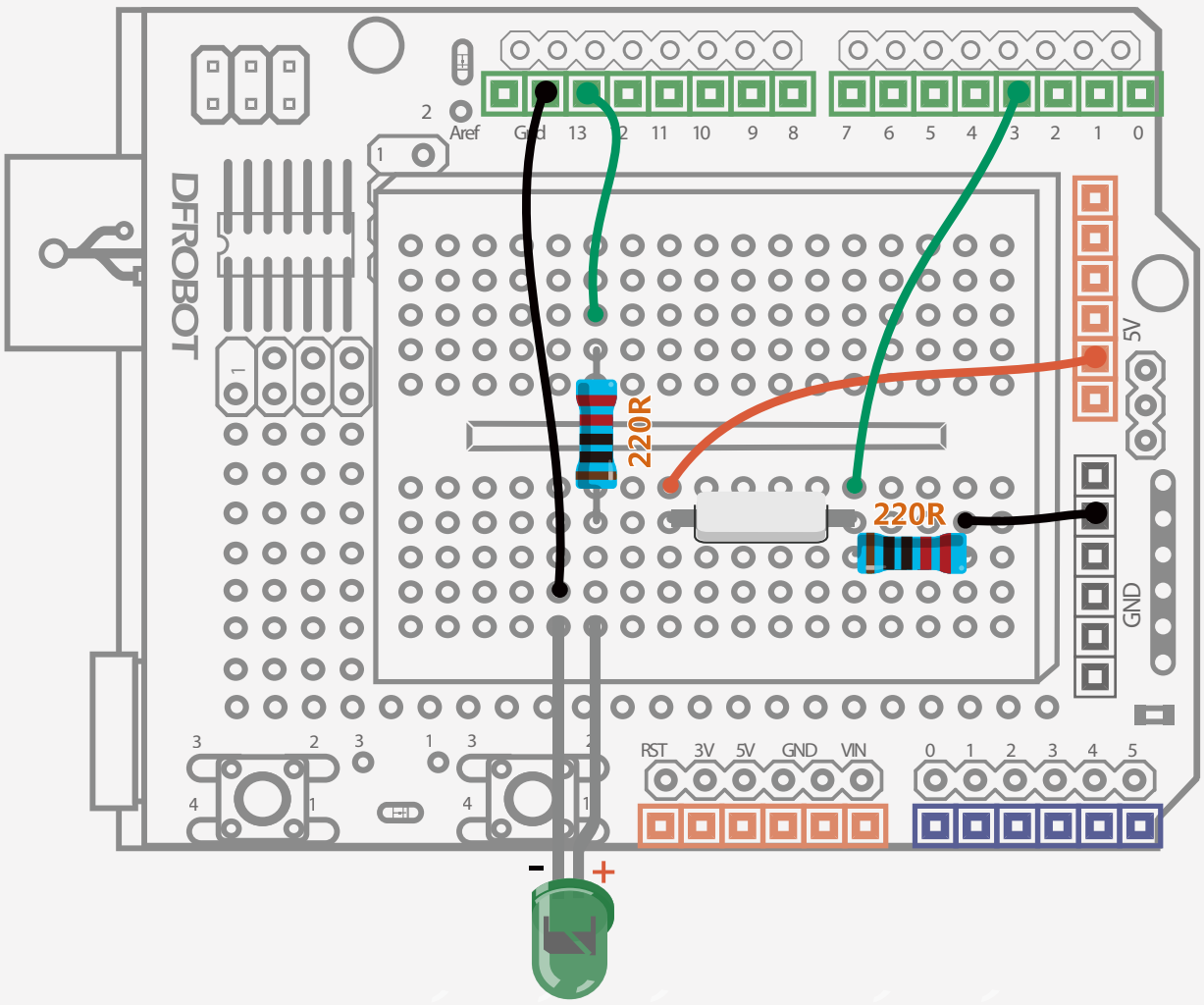


Схема цепи датчика наклона

Код

Пример кода 8-1

```
//проект 8 - датчик вибрации

int SensorLED = 13;                    //определить светодиодный цифровой
пин 13
int SensorINPUT = 3;                   подключить датчик наклона для
прерывания 1 в цифровом пине 3
unsigned char state = 0;

void setup() {
  pinMode(SensorLED, OUTPUT);      //настроить светодиод как исходящий
режим
  pinMode(SensorINPUT, INPUT);     //настроить датчик наклона в качестве
режима ввода
//когда низкое напряжение изменяется на высокое напряжение, оно
вызывает прерывание 1 и запускает функцию мигания
  attachInterrupt(1, blink, RISING);
 }

void loop(){
      if(state!=0){                          // положение if не равно 0
        state = 0;                           // присвоить значение состояния 0
        digitalWrite(SensorLED,HIGH);  // включить светодиод
        delay(500);                            // задержка на 500 мс
      }
      else{
        digitalWrite(SensorLED,LOW);   // если нет, выключите светодиод
     }
}

void blink(){                                // функция прерывания blink ()
state++;                                     // после запуска прерывания, состояние
продолжает увеличиваться


}
```

Ожидаемый результат:
Когда мы встряхиваем плату, светодиод HIGH (включен). Когда мы перестанем дрожать, светодиод LOW (выключен).

Код

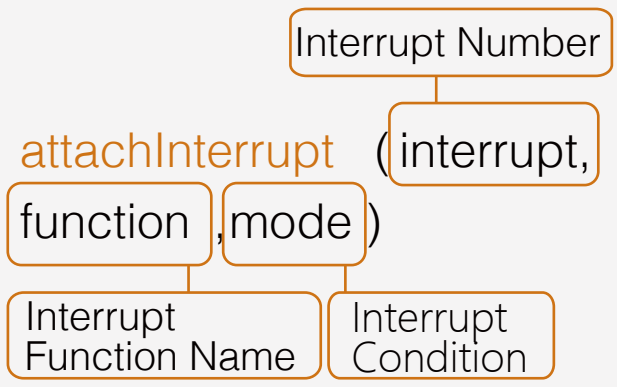
В этом разделе мы рассмотрим функцию прерывания, которая была использована в коде. Программа работает следующим образом: при отсутствии прерывания программы код продолжает работать, а светодиод остается LOW (выключен). Когда происходит внешнее событие и активируется датчик наклона, например, кто-то трясет плату, программа запускает функцию blink (), и состояние начинает увеличиваться. Когда конструкция if обнаруживает, что состояние больше не 0, оно вызывает светодиод HIGH (включено). В то же время он сбрасывает состояние в 0 и ждет следующего прерывания. Если нет прерывания, светодиод LOW (выкл.).

Что такое прерывание?

Представьте, что вы смотрите дома телевизор и звонит телефон. Вам нужно прекратить смотреть телевизор и поднять трубку. После того, как звонок закончился, вы продолжаете смотреть телевизор. В этом случае вызов является прерыванием, и звонок телефона является условием.

Функция attach Interrupt указывает на именованную функцию или процедуру обслуживания прерываний (ISR) для вызова при возникновении прерывания. Это заменяет любую предыдущую функцию, которая была привязана к прерыванию. Большинство плат Arduino имеют два внешних прерывания: цифры 0 (на цифровом выводе 2) и 1 (на цифровом выводе 3). Различные платы имеют разные прерывистые контакты.Проверьте ссылки на arduino.cc для получения дополнительной информации: http://arduino.cc/en/ Reference / AttachInterrupt

В «attachInterrupt» есть 3 параметра,



interrupt:

Номер прерывания (int) - либо 0, либо 1. Если он равен 0, вы должны подключиться.

function:

+ Функция вызывается при возникновении прерывания

+ Функция не имеет параметров и ничего не возвращает

+ Поскольку delay () и millis () оба полагаются на прерывания, они не будут работать во время работы функции.

+ Функция не может считывать значения из последовательного порта. Вы можете потерять данные, подключенные к последовательному порту.

mode:

«Mode» определяет, когда прерывание должно быть запущено. Четыре константы предопределены как допустимые значения:

«LOW» для запуска прерывания каждый раз, когда сигнал на входе низкий,

«CHANGE» для запуска прерывания каждый раз, когда сигнал на входе меняет значение ,

«FALLING», когда сигнал на входе изменяется от HIGH to LOW.

Давайте вернемся к программе:

attachInterrupt(1, blink, RISING);

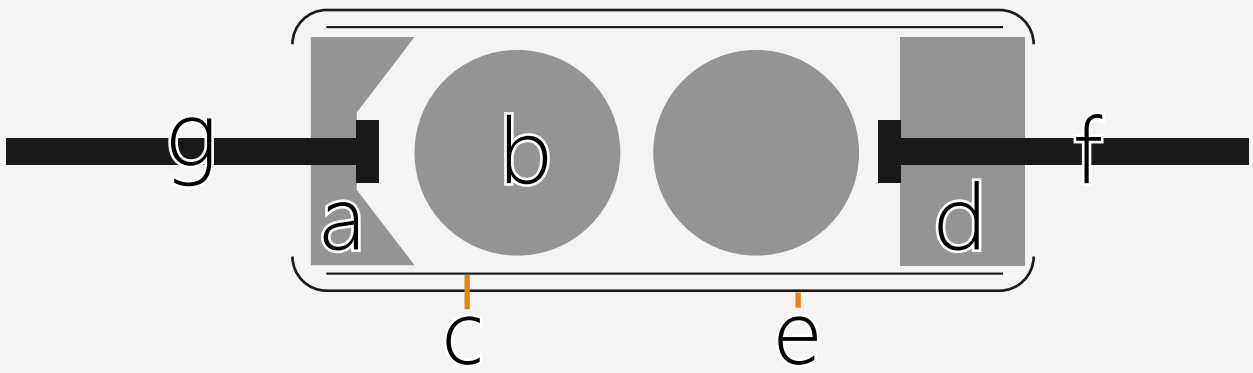
1. attachInterrupt(1,"because the tilt sensor is connected to digital pin 3.
2. Blink" is our interrupt function
3. «ВОСПРОИЗВЕДЕНИЕ», чтобы вызвать, когда пин идет от LOW до HIGH.

Почему мы выбрали «RISING»? Когда датчик наклона не обнаруживает никакого сигнала, контакт 3 LOW. Когда он обнаруживает сигнал, он соединяется с 5 вольтами, и это изменяет контакт от LOW до HIGH.

Компоненты

Датчик наклона

Датчик наклона имеет несколько названий: шариковый переключатель, вибрационный переключатель и т. д. Хотя у него разные названия, принципы работы одинаковы. Проще говоря, это переключатель, состоящий из цилиндра с небольшим металлическим шариком внутри. Когда металлический шар катится к любому краю цилиндра, они касаются одного из контактных штифтов, и цепь завершена. Изучите схему для более подробной информации.



a. Бронзовая оболочка b. Бронзовый шарик c. Бронзовая трубка

d. PC set e. Термочувствительная трубка

f. Бронзово-кондуктивный пин

g. Phosphor Copper Pinch Cock

Рис. 8-2 Схема вибрационного датчика

Проект 9

Светочувствительный светодиод

09

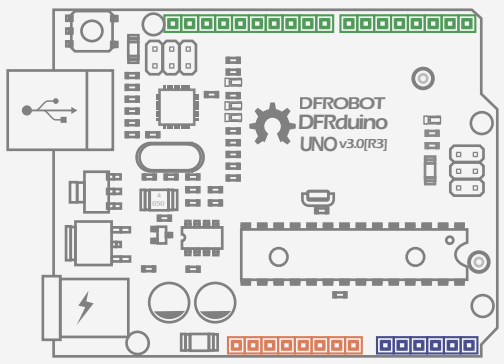
DFROBOT
DRIVE THE FUTURE

Светочувствительный светодиод

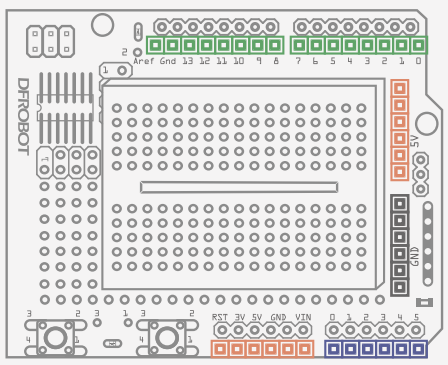
Знакомимся с новым сенсорным компонентом: фотодиодом. Проще говоря, когда датчик обнаруживает свет, его сопротивление меняется. Чем сильнее свет в окружающей среде, тем ниже будет значение сопротивления, которое будет считывать фотодиод. Читая значение сопротивления фотодиода, мы можем выработать окружающее освещение в окружающей среде. Фотодиод, входящий в комплект для начинающих, является типичным датчиком освещенности.

В этом проекте мы сделаем автоматический свет, который может отрегулировать себя в соответствии с окружающим освещением вокруг него. Когда он темный, фотодиод обнаруживает изменение и запускает свет, и наоборот.

Компоненты



DFduino UNO R3 x1



Плата расширения x1

Перемычки “мама-мама” x5

Светодиод 5 ММ x1

Датчик внешней освещенности x1

Резистор 220 Ом x1

Резистор 10 кОм x1

Цепь

Имейте в виду, что фотодиоды поляризованы, как светодиоды, поэтому они будут работать только при правильном подключении.Фотодиод должен быть подключен к резистору 10 кОм, а не к резистору 220 Ом.

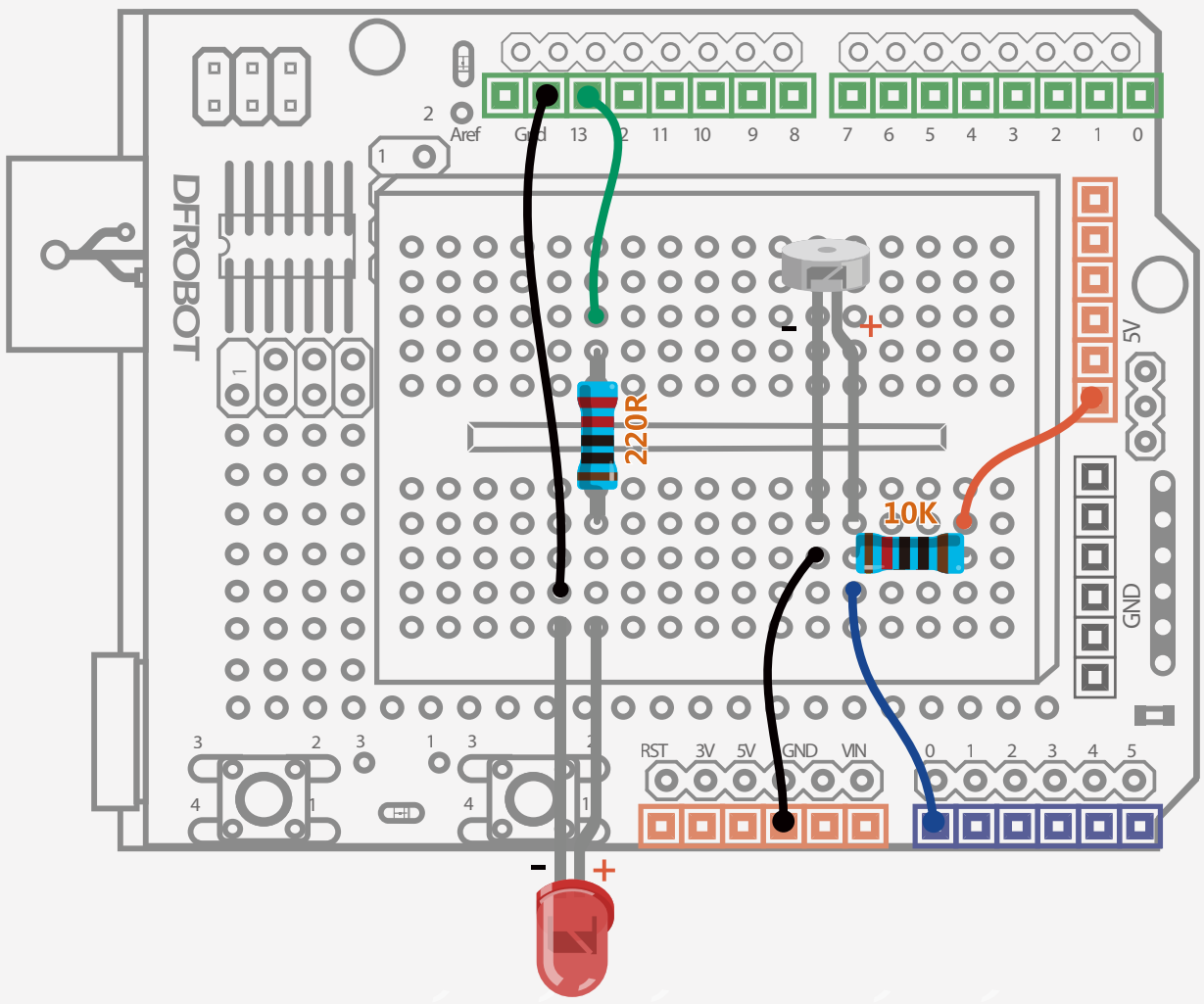


Схема фотодиодной цепи

Пример кода 9-1 :

```
//Проект 9 - Зажигание лампу
int LED = 13;                    //определить светодиодный цифровой
контакт 13
int val = 0;                     //определить значение напряжения
фотодиода в цифровом выводе 0
void setup(){
    pinMode(LED,OUTPUT);      // Настройка светодиода в качестве режима
вывода
    Serial.begin(9600);       // Настройка скорости передачи 9600
}

void loop(){
    val = analogRead(0);   // Чтение значения напряжения от 0 -1023
    Serial.println(val);   // считывание значения напряжения с
последовательного монитора
    if(val<1000){          // Если ниже 1000, выключите светодиод
        digitalWrite(LED,LOW);
    }else{                 // В противном случае включите светодиод
        digitalWrite(LED,HIGH);
    }
    delay(10);             // задержка на 10 мс
}
```

После загрузки кода вы можете светить фонариком на фотодиод, чтобы изменить уровень освещенности окружающей среды. Когда фонарик не светится, светодиод должен загореться. Когда фонарик светится, светодиод должен погаснуть.

Код

Очень краткое описание программы:
Подобно температурному датчику LM35, фотодиод считывает аналоговый сигнал, поэтому нам не нужно определять «pinMode» в «serial.begin».
Мы берем данные аналогового тока из фотодиода и сравниваем его со значением 1024, чтобы сделать его цифровым. Вы можете изменить это значение, если хотите. Попробуйте сыграть с последовательным монитором и посмотреть, какие выходы дает фотодиод. Затем используйте число, которое вы получите здесь, в качестве числа сравнения, чтобы изменить чувствительность цепи.

Цепь

Фотодиод

Фотодиод - полупроводниковое устройство, преобразующее свет в ток.

Ток генерируется, когда фотоны поглощаются фотодиодом. Чем сильнее свет окружающей среды, тем ниже будет значение сопротивления, которое выводит фотодиод. Аналоговое значение в диапазоне от 0 до 1023 соответствует значению напряжения в диапазоне от 0 до 5 В.Входное напряжение Vin (5 В) подключено к 2 резисторам. Измеряя напряжение R2, как показано ниже, вы можете получить значение сопротивления фотодиода.В нашем проекте R1 является резистором 10k, а R2 - фотодиодом. Значение сопротивления R2 в темноте настолько велико, что оно почти достигает 5V. Когда поглощаются фотоны, значение R2 будет уменьшаться, равно как и значение его напряжения. Для этого проекта предпочтительнее использовать фиксированный резистор в диапазоне от 1k до 10k, в противном случае коэффициент деления напряжения не очевиден. Вот почему в этом проекте мы использовали резистор 10k для R1.

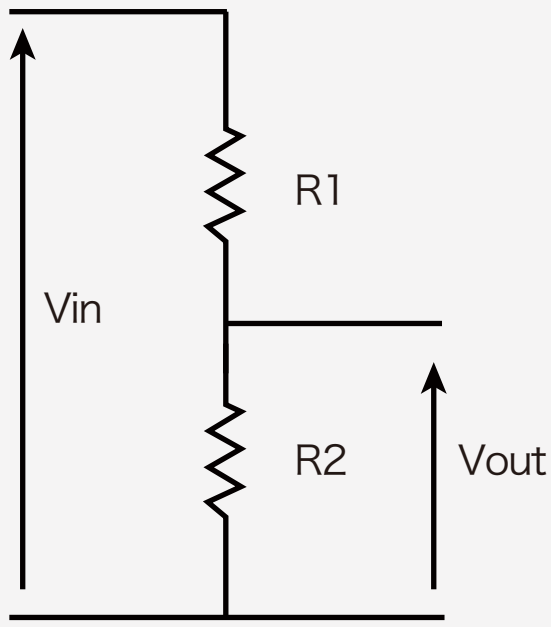


Схема делителя напряжения

$$Vout = \frac{R2}{R1+R2} \times Vin$$

Формула делителя напряжения

Проект 10

Как подключать сервопривод

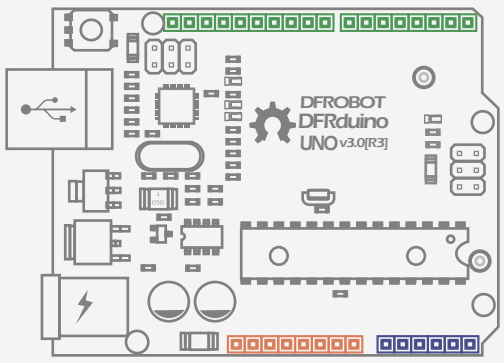
10

DFROBOT
DRIVE THE FUTURE

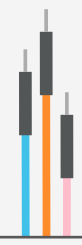
Как подключить сервопривод

Сервоприводы идеально подходят для встроенных электронных приложений, поскольку они могут точно перемещаться в определенную позицию. Большинство сервоприводов могут поворачиваться на 180 градусов максимум. Некоторые даже более крупные могут поворачиваться на 360 градусов. Они могут использоваться на мобильных платформах для обнаружения устройств, таких как камеры и детекторы интеллектуальных транспортных средств, или в роботизированных суставах.

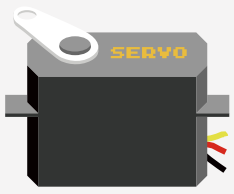
Компоненты



DFduino UNO R3 x1



Перемычки M/M x3



Сервопривод x1

Цепь

Сервопривод имеет три провода. Цвет проводов колеблется между сервоприводами, но красный провод всегда равен 5В, а земля будет либо черным, либо коричневым. Другим проводом является сигнальный провод, и он обычно оранжевый или желтый. Сигнальный провод подключен к цифровому пину 9.

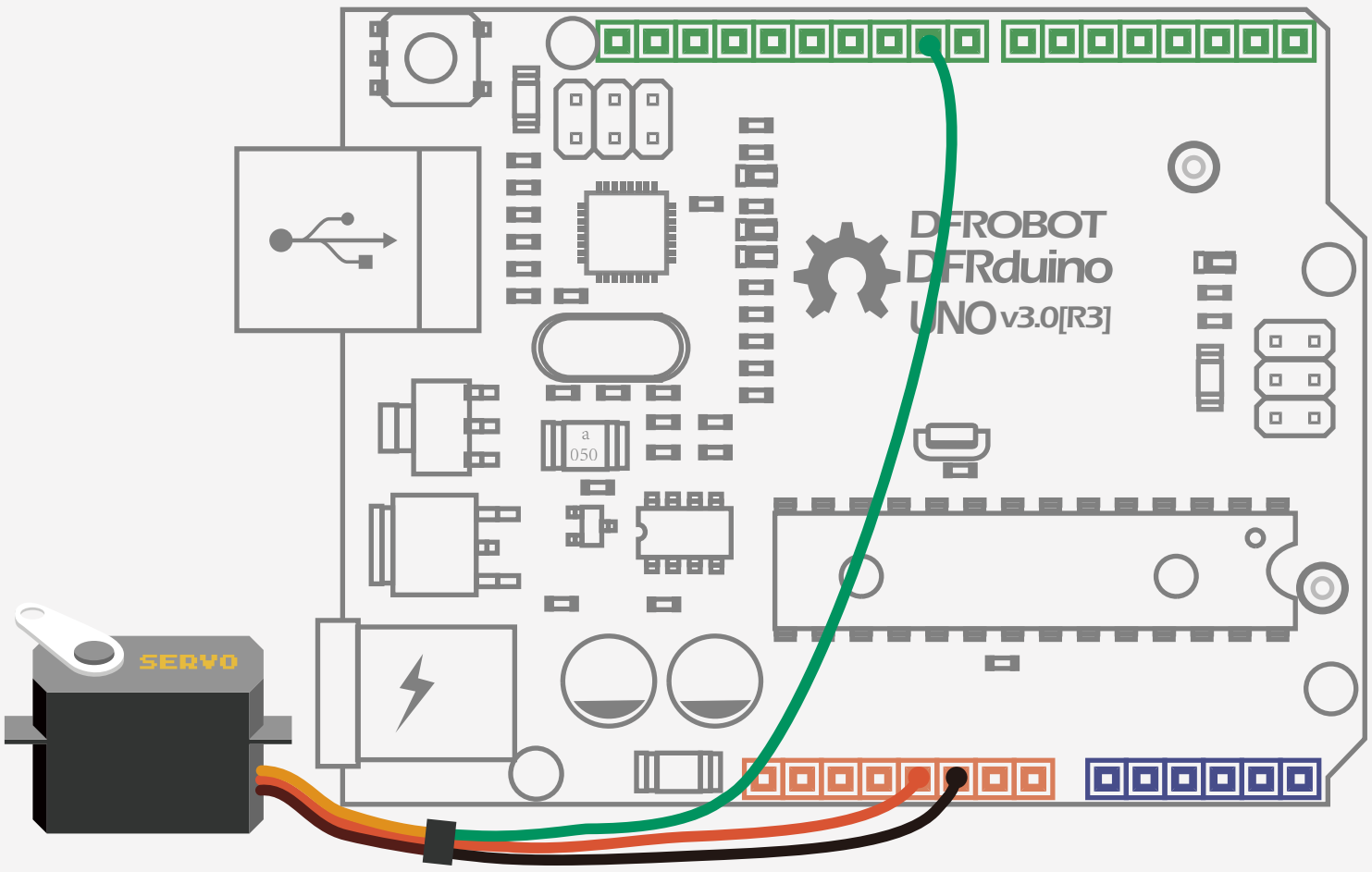


Схема сервопривода, подключенного к Arduino

Код

Пример кода 10-1

```
// Проекта 10 Серво
#include <Servo.h>          // объявить о вставке библиотеки Servo.h
Servo myservo;              // создаем объект серво для управления
сервоприводом
int pos = 0;                // переменная pos для сохранения положения
сервопривода
void setup() {
   myservo.attach(9);       // присоединяем сервопривод к цифровому пину
9.
}

void loop() {
    for(pos = 0; pos < 180; pos += 1){      //серво поворачивается от 0 до 180
с шагом 1 градус
        myservo.write(pos);  //передаем сервоуправление в положение в
переменной 'pos'
        delay(15);              //ждем 15сек для достижения позиции
сервопривода
}
   for(pos = 180; pos>=1; pos-=1) {         //серво поворачивается от 180 до 0
с шагом 1 градус
        myservo.write(pos);  //передаем сервоуправление в положение в
переменной 'pos'
        delay(15);              //ждет 15 мс, чтобы сервопривод достиг позиции
  }
}
```

После загрузки скетча вы увидите, что сервопривод перемещается взад и вперед от 0 до 180 градусов.

Код

Скетч начинается с вставки библиотеки <Servo.h>.

```
#include <Servo.h>
```

Эта библиотека уже находится в Arduino IDE. Определите его, открыв Arduino-1.0.5 / libraries / Servo / Servo.h. Библиотеки представляют собой коллекции новых команд, которые были упакованы вместе, чтобы было легко включить их в свои скетчи. Arduino поставляется с несколькими полезными библиотеками, такими как серво-библиотека, используемая в этом примере, которая может использоваться для взаимодействия с более продвинутыми устройствами.

Нам нужно создать имя в коде для сервопривод:

В функции setup () There есть еще одна команда.

```
myservo.attach(9);
```

Объявление функций в серво-библиотеке немного отличается от объявления других функций. Нам нужно объявить различные функции в библиотеке, включая объявление серво объекта и определение функции. Как и в библиотеке, вам нужно указать объект, чтобы программа могла его идентифицировать. Формат библиотечной функции приведен ниже.Не пропустите знак точки («.») Между словом «мой серво» и «прикрепить». myservo - это объект servo, который мы назвали ранее.

Таким образом, мы вызываем:

```
attach(pin) ;
```

digital pins

attach (pin) назначает контакт. Мы можем использовать любой цифровой вывод, кроме 0 и 1. В этом проекте мы выбрали цифровой вывод 9.

В основной программе для есть 2 положения. Первый начинается с 0, затем вращается на 180 градусов с шагом в 1 градус. Второй начинается с 180 градусов и возвращается к 0 с шагом в 1 градус.

```
myservo.write(pos);
```

Как и предыдущая команда, вы должны объявить имя для этой команды. Параметры этой функции - это угол. Единица градуса.

Если вы хотите узнать больше о функциях в серво-библиотеке, посетите веб-сайт arduino:
http://ardui-no.cc/ru/reference/servo
или посетите веб-сайт DFRobot:
www.dfrobot.com

Проект 11

Управляемый сервопривод

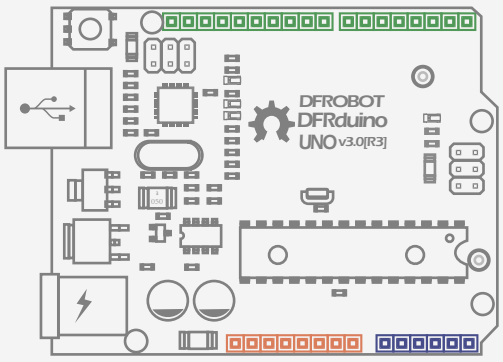
11

DFROBOT
DRIVE THE FUTURE

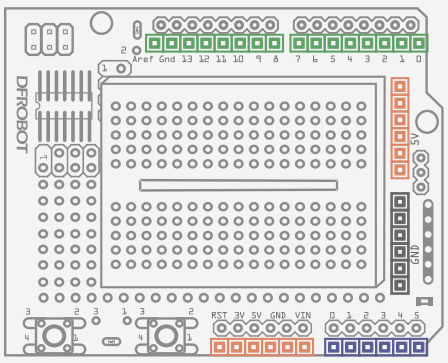
Управляемый свветодиод

Мы научились превращать сервопривод в конкретный положение с внешнего сигнала. В новом проекте мы будем использовать потенциометр для управления сервоприводом. Вы также можете изменить эту схему, заменив потенциометр на, такого как переключатель наклона, или замените привод на светодиод. Погрузитесь и используйте свое воображение!

Компоненты



DFduino UNO R3 x1

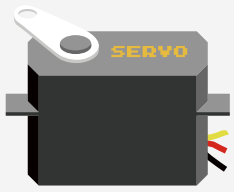


Плата расширения x1

Перемычки “мама-мама” x3

Перемычки F/M x3

10K Потенциометр x1



Сервопривод x1

Цепь

Она немного отличается от нашего предыдущего проекта, поскольку мы используем потенциометр. Потенциометры иногда называют переменными резисторами, и они такими являются. Вращая ручку, вы изменяете электрическое сопротивление компонента. Он имеет 3 пина: два по бокам и один сверху. Подключите боковые штыри к 5V и GND на Arduino соответственно, а один штифт на противоположной стороне потенциометра к аналоговому выводу 0 на Arduino.

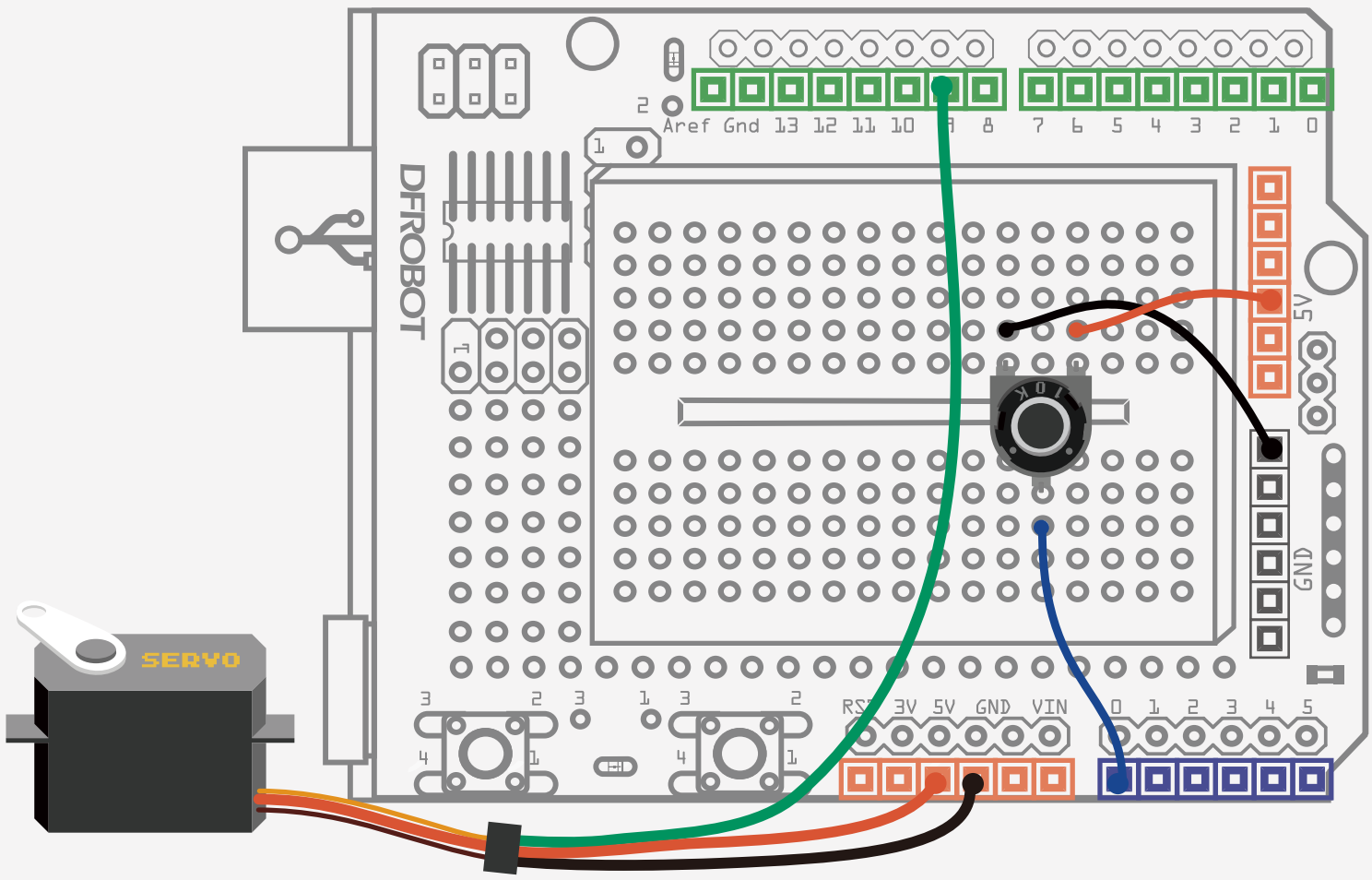


Рис. 11-1 Схема управляемого сервопривода

Код

Пример кода 11-1

```
// Проект 11 управляемый сервопривод

#include <Servo.h>        // вставляем библиотеку Servo.h
Servo myservo;            // создание серво объекта для управления
сервоприводом

int potpin = 0;           //подключить потенциометр к цифровому pin0
int val;                  //значение переменной для считывания значения с
аналогового вывода

void setup() {
  myservo.attach(9);      //Прикрепите сервопривод на контакте 9 к серво
объекту.
}

void loop() {
  val = analogRead(potpin);        //использует значение потенциометра
(значение от 0 до 1023)
  val = map(val, 0, 1023, 0, 179);    // масштабируем его, чтобы использовать
его с серво (значение от 0 до 180)
  myservo.write(val);              // устанавливает положение сервопривода
в соответствии с масштабированным значением
  delay(15);                       // ждать 15 мс, чтобы перейти в
определенную позицию

}
```

После загрузки скетча вы можете увидеть поворот сервопривода в соответствии с положением потенциометра.

Код

Сначала мы вставляем библиотеку <Servo.h>, а затем определим потенциометр на аналоговом пину 0, чтобы прочитать его значение.

Теперь перейдем к функции «map». Формат функции карты выглядит следующим образом:

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

Функция «map» повторно отображает число из одного диапазона в другой. То есть значение «from-Low» будет отображаться в «toLow», значение «fromHigh» to «toHigh», значения между промежуточными значениями и т. Д.

Параметры

value: число для отображения

fromLow: нижняя граница текущего диапазона значений

fromHigh: верхняя граница текущего диапазона значений

toLow: нижняя граница целевого диапазона значения

toHigh: верхняя граница целевого диапазона значения

Обратите внимание, что «нижние границы» любого диапазона могут быть больше или меньше «верхних границ», поэтому функция «map ()» может использоваться для изменения диапазона чисел, например:

```
y = map(x, 1, 50, 50, 1);
```

Функция также хорошо обрабатывает отрицательные числа, так что этот пример также действителен и работает хорошо:

```
y = map(x, 1, 50, 50, -100);
val = map(val, 0, 1023, 0, 179);
```

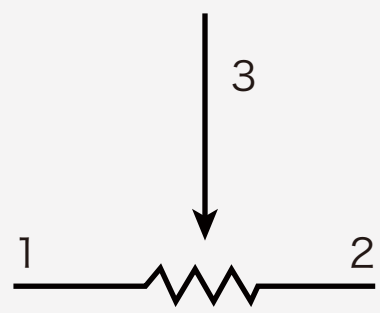
Вернемся к нашему скетчу: мы сопоставляем аналоговое значение от 0 до 1023 до значения сервопривода от 0 до 180.

Цепь

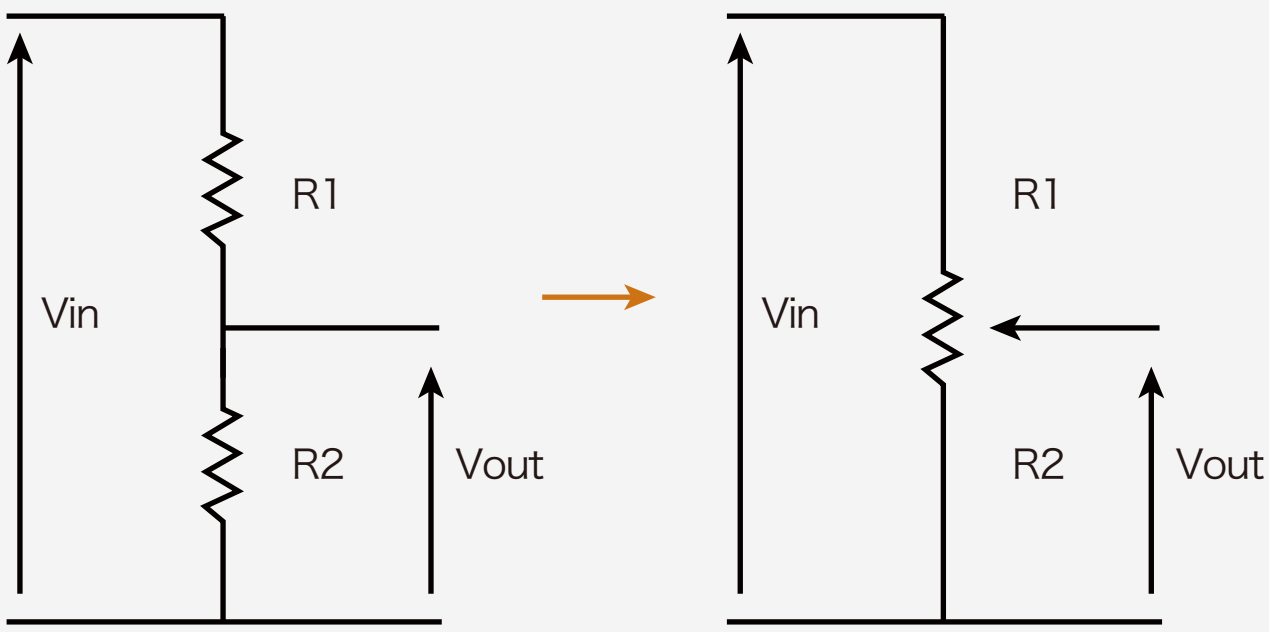
Потенциометр

Потенциометр представляет собой простую ручку, которая обеспечивает переменное сопротивление rangin от 0 до 10k Ω, которое Arduino может читать как аналоговое значение

В этом проекте мы подключили три провода к Arduino. Первый пошел на землю с одного из внешних штифтов потенциометра. Второй пошел от 5 вольт к другому внешнему пину потенциометра. Третий пошел от аналогового входа 2 к среднему выходу потенциометра.



Потенциометр работает аналогично делителю напряжения в Проекте 9. Потенциометр разделен на 2 резистора валом. Поворачивая вал потенциометра, мы меняем количество сопротивлений на внешней стороне стеклоочистителя, которое соединено с центральным штифтом потенциометра. Это изменяет относительную «близость» этого контакта до 5 вольт и землю, что дает нам другой аналоговый вход. Когда вал повернут полностью в одном направлении, на штифт направляется 0 вольт, и мы читаем 0. Когда вал повернут полностью в другом направлении до верхнего предела, есть 5 вольт, идущих к pin, и мы читаем 1023. В промежутке «analogRead ()» возвращает число от 0 до 1023, которое пропорционально величине напряжения, подаваемого на штырь.



Проект 12

Интерактивный регулируемый RGB-светодиод

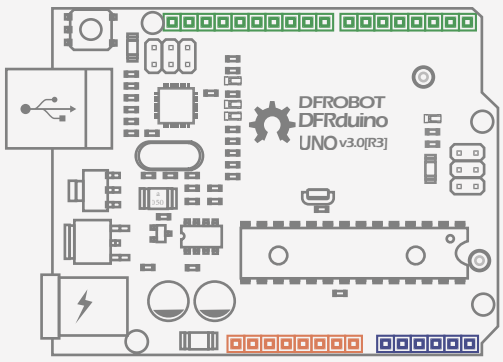
12

DFROBOT
DRIVE THE FUTURE

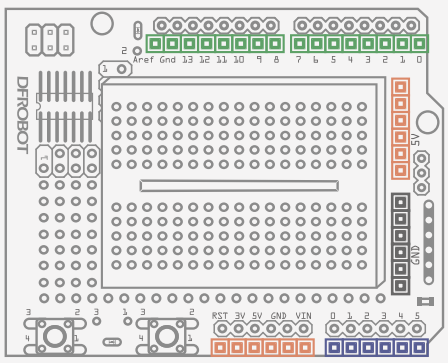
Интерактивный регулируемый RGB-светодиод

В Проекте 5 мы узнали о том, как настроить RBG-светодиод на различные цвета. На этот раз мы попытаемся сделать его интерактивным, добавив 3 потенциометра, чтобы вы могли выбрать любой цвет, который вы хотите для своего освещения дома.

Компоненты



Основная плата DFRduino Uno x1



Плата расширения x1

Перемычки “мама-мама” x13

Резистор 220 Ом x3

10K Потенциометр x3

5MM RGB-светодиод x1

Цепь

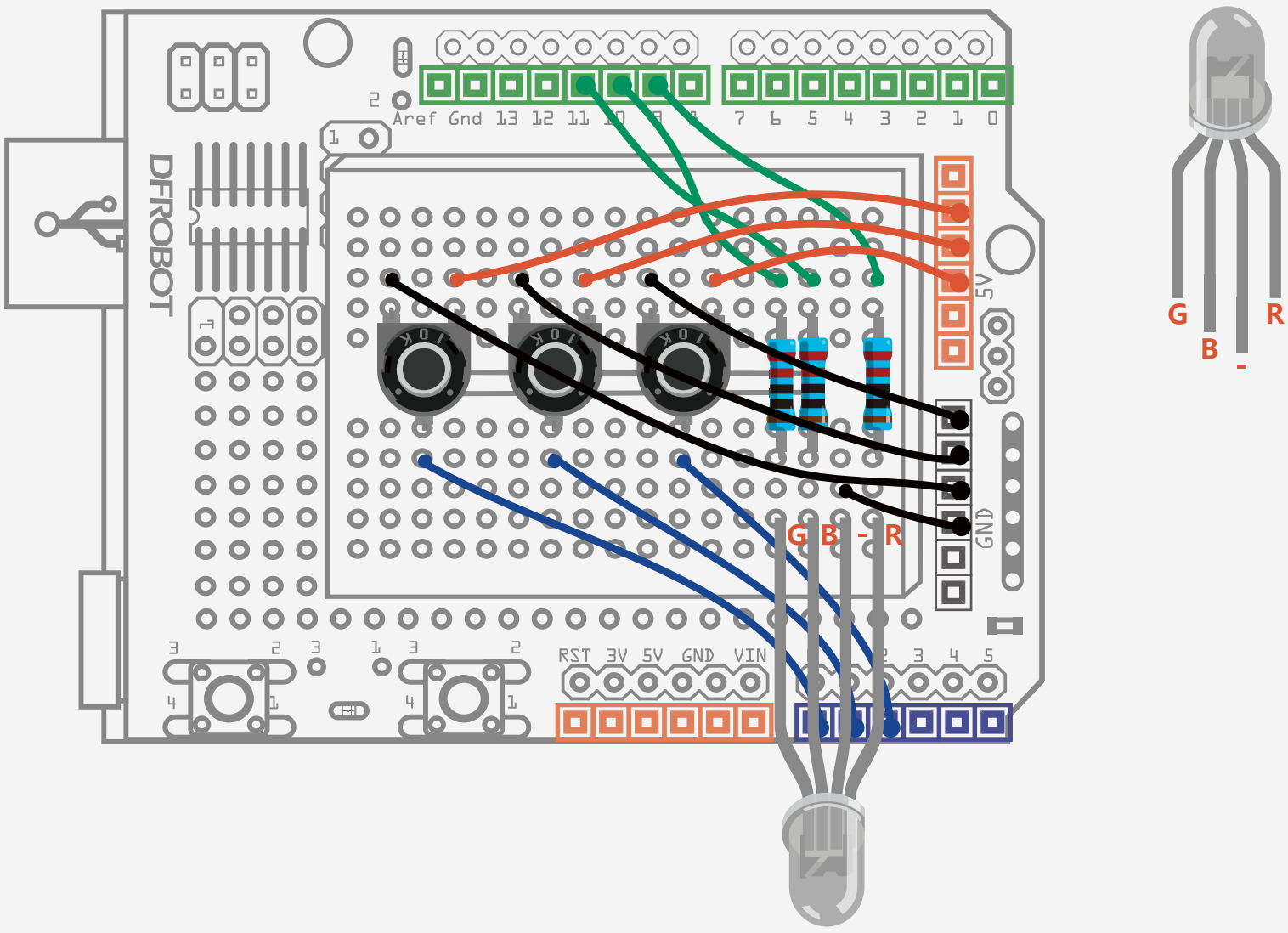


Рис. 12-1. Схема цепи интерактивно-регулируемого RGB-светодиода

Код

```
//Пример кода 12-1:
int redPin = 9;                    // R - цифровой 9
int greenPin = 10;                 // G - цифровой 10
int bluePin = 11;                  // B - цифровой 11
int potRedPin = 0;                 // потенциометр 1 - аналоговый 0
int potGreenPin = 1;               // потенциометр 2 - аналоговый 1
int potBluePin = 2;                // потенциометр 3 - аналоговый 2
void setup(){
  pinMode(redPin,OUTPUT);
  pinMode(greenPin,OUTPUT);
  pinMode(bluePin,OUTPUT);
  Serial.begin(9600);                    // Инициализировать
последовательный порт
}
void loop(){
  int potRed = analogRead(potRedPin);        //  считывание значения с
потенциометра красного светодиода
  int potGreen = analogRead(potGreenPin);    //  считывание значения из
потенциометра зеленого светодиода
  int potBlue = analogRead(potBluePin);      // считывание значения из
потенциометра синего светодиода
  int val1 = map(potRed,0,1023,0,255);       // сопоставьте значение напряжения
от 0 ~ 1023 до аналогового значения в диапазоне от 0 ~ 255
  int val2 = map(potGreen,0,1023,0,255);
  int val3 = map(potBlue,0,1023,0,255);

  //печатать значение красного, зеленого и синего светодиодов из
последовательного порта
  Serial.print("Red:");
  Serial.print(val1);
  Serial.print("Green:");
  Serial.print(val2);
  Serial.print("Blue:");
  Serial.println(val3);
  colorRGB(val1,val2,val3);  // настроить аналоговое значение для светодиода RGB
}
}

//определить функцию colorRGB
void colorRGB(int red, int green, int blue){
  analogWrite(redPin,constrain(red,0,255));
  analogWrite(greenPin,constrain(green,0,255));
  analogWrite(bluePin,constrain(blue,0,255));
}
```

После загрузки скетча, вы можете изменить различные комбинации красного, зеленого и синего цветов на RGB-светодиоде

Проект 13

Самодельный вентилятор

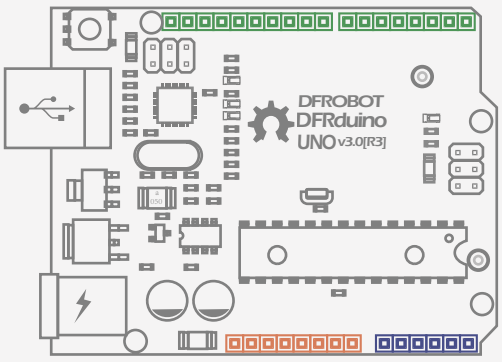
13

DFROBOT
DRIVE THE FUTURE

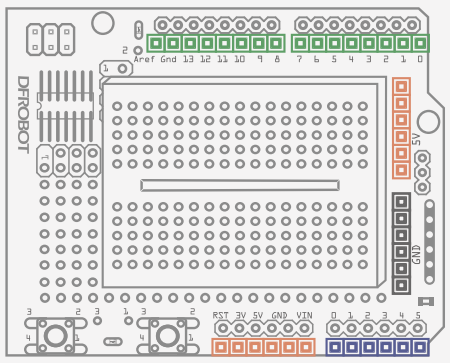
Самодельный вентилятор

В этом проекте мы будем использовать реле и двигатель, чтобы сделать небольшой вентилятор. Реле представляет собой электрически управляемый переключатель, который позволяет включать или выключать цепь с использованием напряжения и / или тока, намного превышающего, чем может обрабатывать Arduino.

Компоненты



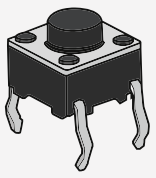
DFduino UNO R3 x1



Плата расширения x1

Перемычки “мама-мама” x9

Светодиод 5ММ x1

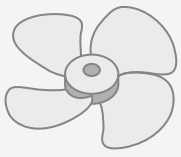


Нажимная кнопка x1

Резистор 220 Ом x2

Реле x1

130 Мотор x1



Крыльчатка x1

Проводка/Подключение

Подключите кнопку с помощью стягивающего резистора 220 Ом, чтобы удерживать логический сигнал около нуля вольт, когда кнопка отключена. Реле имеет 6 контактов. Контакт 1 и 2 на реле - это входные сигналы и отдельно подключены к цифровому контакту 3 и GND на Arduino. Штырьки 3, 4, 5 и 6 на реле являются выходными сигналами, но в это время мы будем использовать только контакт 4 и контакт 6. Реле аналогично кнопке, которая также имеет 2 соединения.

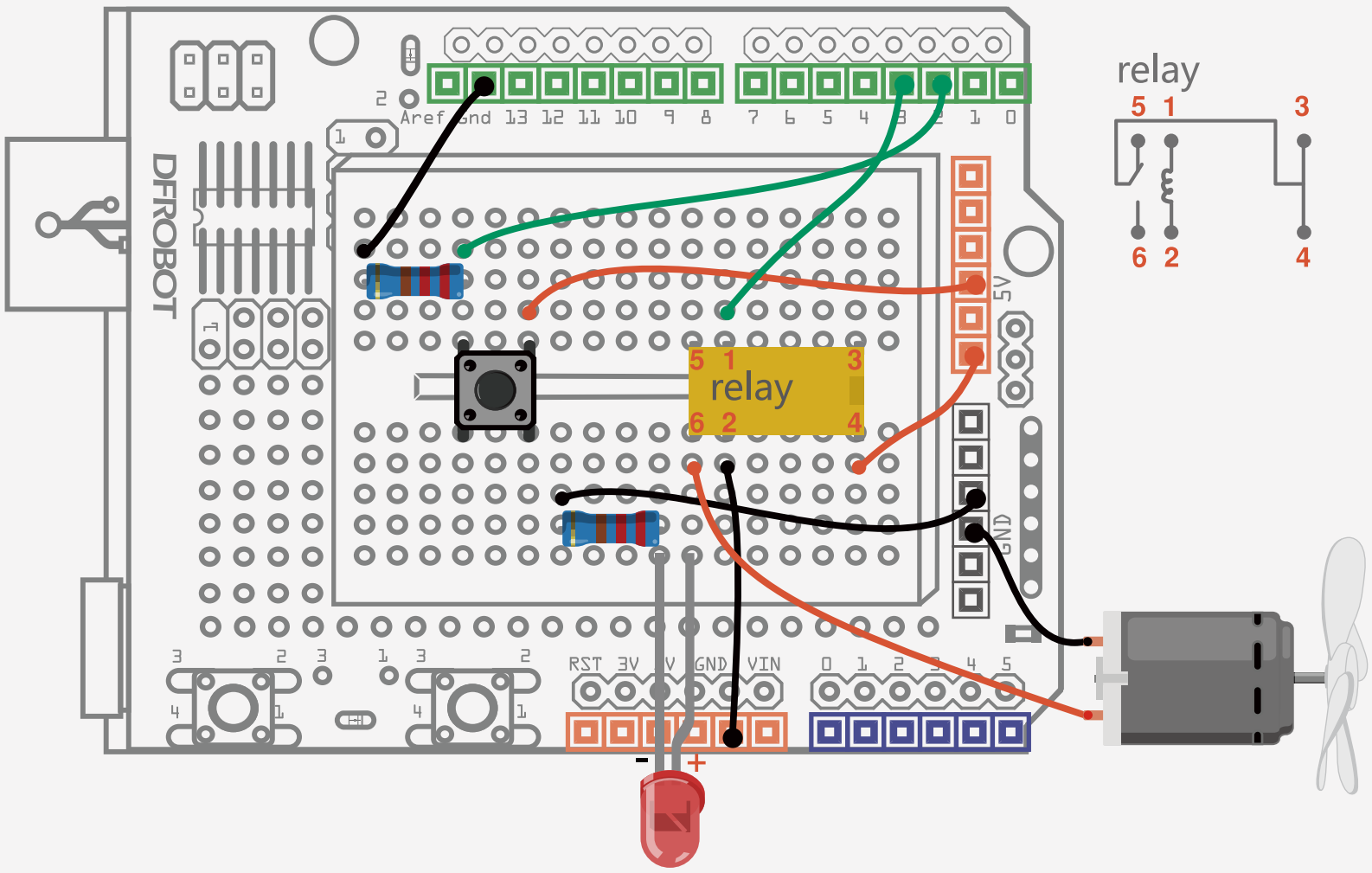


Рис. 13-1 Схема соединений для самодельного вентилятора

Код

Пример кода 13-1:

```
//Проект тринадцать - Arduino для управления работой вентилятора
int buttonPin = 2;                    // int buttonPin = 2;
int relayPin =  3;                    // int relayPin = 3;
int relayState = HIGH;                // int relayState = HIGH;
int buttonState;                      // записываем текущее состояние кнопки
int lastButtonState = LOW;            //  записываем состояние последней
кнопки
long lastDebounceTime = 0;
long debounceDelay = 50;              //исключить время отладки

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(relayPin, OUTPUT);

digitalWrite(relayPin, relayState);   // настроить начальное состояние реле
}

void loop() {
  int reading = digitalRead(buttonPin);     //читаем значение кнопки

  // однажды обнаруживает изменение состояния, записывает время
  if (reading != lastButtonState) {
    lastDebounceTime = millis();
  }

// ждать 50 мс, чтобы оценить, является ли оно тем же самым состоянием, что и
последнее состояние
// если они разные, измените состояние кнопки
// если состояние кнопки высокое (нажато), измените состояние реле
  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading != buttonState) {
      buttonState = reading;

      if (buttonState == HIGH) {
        relayState = !relayState;
      }
    }
  }
  digitalWrite(relayPin, relayState);

  // изменение последнего состояния кнопки
  lastButtonState = reading;
}
```

После загрузки скетча, вы можете контролировать реле и светодиод с помощью кнопки.

Код

Дребезг контакта при нажатии кнопки

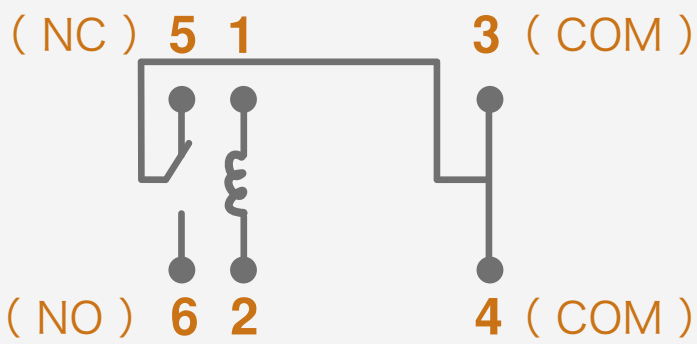
```
if (reading != lastButtonState) {
  lastDebounceTime = millis();
}
if ((millis() - lastDebounceTime) > debounceDelay) {
  if (reading != buttonState) {
    ......
  }
}
```

Когда сигналы принимаются Arduino, программа не работает на них немедленно - она проверяет правильность сигнала и ждет определенное время для подтверждения. Если сигнал правильный, программа будет работать соответственно.Причина, по которой программа проверяет правильный сигнал, состоит в том, что при нажатии кнопки происходит отскок кнопки - дребезг контакта . Он может генерировать неправильный сигнал, поэтому мы тестируем его для решения проблемы на оборудовании.

Аппаратное обеспечение

Реле

Реле представляет собой переключатель с электрическим приводом, который позволяет включать или выключать цепь, используя напряжение и / или ток, намного превышающий уровень, который обычно может обрабатывать Arduino. Между цепью низкого напряжения, управляемой Arduino и высоковольтной цепью, нет связи - реле изолирует цепи друг от друга.
Давайте посмотрим на внутреннюю структуру реле:



Реле имеют 6 контактов. Штырьки 1 и 2 подключены к цифровому выводу и GND. Мы используем эти 2 контакта для питания реле. Между контактом 1 и контактом 2 имеется катушка. Когда цепь HIGH, ток течет в катушке, генерирует магнитное поле, замыкает контакты переключателя и соединяет контакт NO (нормально разомкнутый) с COM (общим). Когда цепь LOW, ток не протекает в катушке, поэтому NC (нормально замкнутый) подключается к общему контакту. Мы подключаем Pin 4 и Pin 6 для управления включением и выключением реле и светодиодом.

Разница между двигателем постоянного тока, шаговыми двигателями и сервоприводами

Двигатели постоянного тока (постоянного тока) - это устройства, которые меняют электрическую энергию на кинетическую энергию. Когда вы подаете питание на двигатель постоянного тока, он начнет вращаться непрерывно, пока эта мощность не будет удалена. Когда вы переключаете полярности, он вращается в противоположном направлении. Двигатель работает непрерывно при высоких оборотах (оборотов в минуту). Эти обороты вала двигателя не могут управляться под определенным углом, но вы можете контролировать скорость. Поскольку вращение происходит так быстро, использовать его для транспортных средств нецелесообразно.Шаговый двигатель имеет комплект передач на двигателе постоянного тока для снижения скорости и увеличения крутящего момента. Это делает его более практичным для использования в транспортных средствах. Его скорость может контролироваться PWM.

Сервопривод также является двигателем. Он контролирует положение двигателя системой обратной связи, как мы видели в рассмотренных сервопроектах. Сервоприводы практичны для использования в роботизированном оружии.

Проект 14

ИК-пульт дистанционного управления светодиодом

14

DFROBOT
DRIVE THE FUTURE

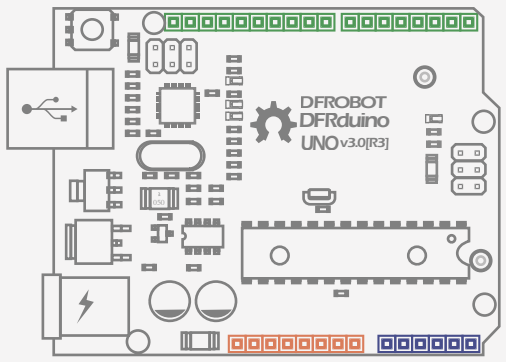
ИК-пульт дистанционного управления светодиодом

Инфро-красный приемник или ИК-приемник - это аппаратное устройство, которое отправляет информацию с инфракрасного пульта дистанционного управления на другое устройство посредством приема и декодирования сигналов. В общем случае приемник выводит код, чтобы однозначно идентифицировать инфракрасный сигнал, который он получает. Этот код затем используется для преобразования сигналов с пульта дистанционного управления в формат, который может быть понят другим устройством. Поскольку инфракрасный свет является светом, для него требуется видимость прямой видимости для

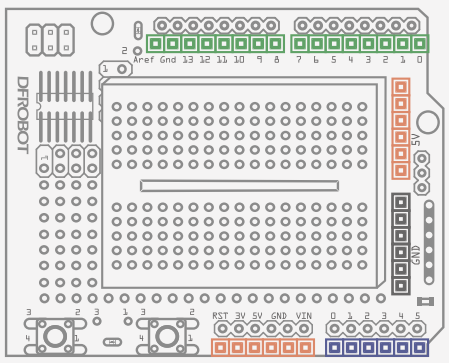
наилучшая возможная операция, но все же может отражаться на таких предметах, как стекло и стены. Плохо размещенные ИК-приемники могут привести к так называемому «туннельному видению», где рабочий диапазон пульта дистанционного управления уменьшается, поскольку они установлены так далеко назад в шасси устройства.

Задание для разминки

Компоненты

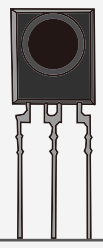


DFduino UNO R3 x1



Плата расширения x1

Перемычки “мама-мама” x3



ИК-приемник x1

ИК-пульт дистанционного управления x1

Подключение

Обратите внимание, что провод «V-out» инфракрасного приемника должен быть подключен к цифровому пину 11 на Arduino.

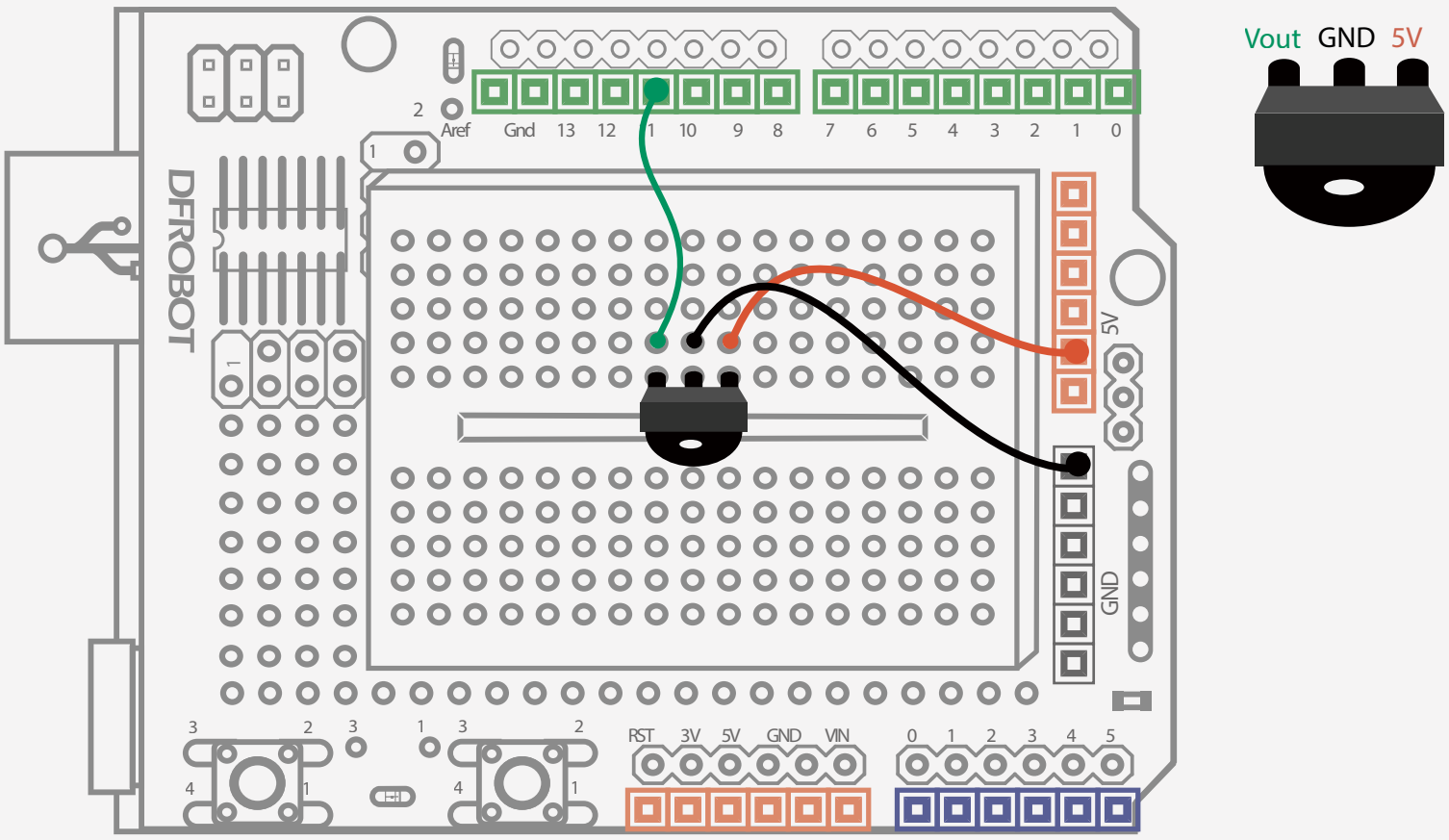


Рис. 14-1 Схема ИК-приемник

Код

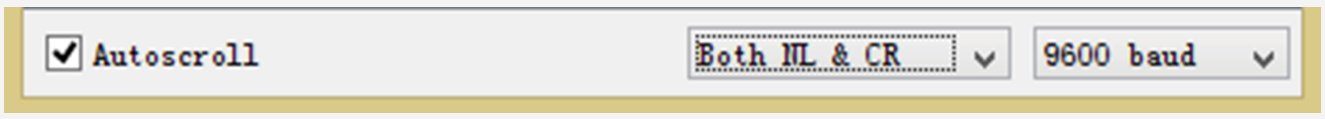
Сначала нужно вызвать библиотеку «IRremote». Разархивируйте RAR и сохраните его в каталоге файлов Arduino. Запустите «IRrecvDemo» в примере.
Если вы не знаете, как вызвать библиотеку в среде Arduino, обратитесь к упражнению в Project 5.

Этот скетч от «IRrecvDemo» в примерах библиотеки «IRremote».

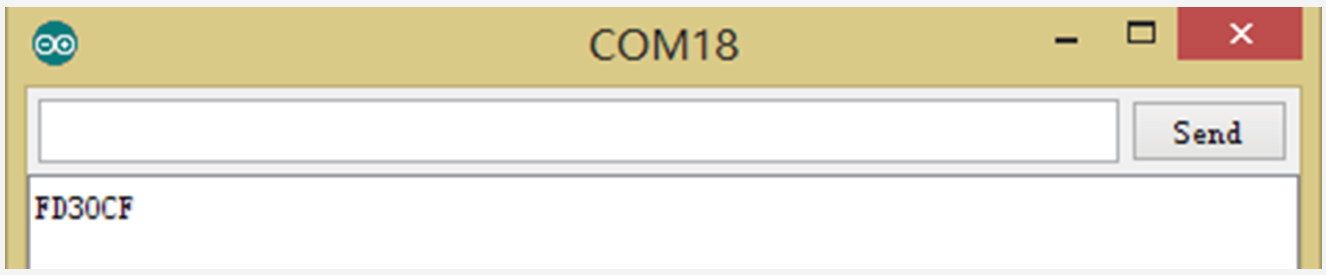
Пример кода 14-1 :

```
//проект четырнадцать инфракрасный приемник
#include <IRremote.h>        //вставляем библиотеку IRremote.h
int RECV_PIN = 11;             //определить вывод RECV_PIN 11
IRrecv irrecv(RECV_PIN);       //определить RECV_PIN как инфракрасный
приемник
decode_results results;        //определение результатов переменных для
сохранения результата инфракрасного

void setup(){
  Serial.begin(9600);            //настраиваем скорость передачи в бодах 9600
  irrecv.enableIRIn();           //Загружаем инфракрасное декодирование
}

void loop() {
    //проверяем, принимают ли данные декодирования и сохраняют их в
переменных результатах
    if (irrecv.decode(&results)) {
        // печать данных, полученных в шестнадцатеричном формате
        Serial.println(results.value, HEX);
        irrecv.resume();  // ожидание следующего сигнала
  }
}
```

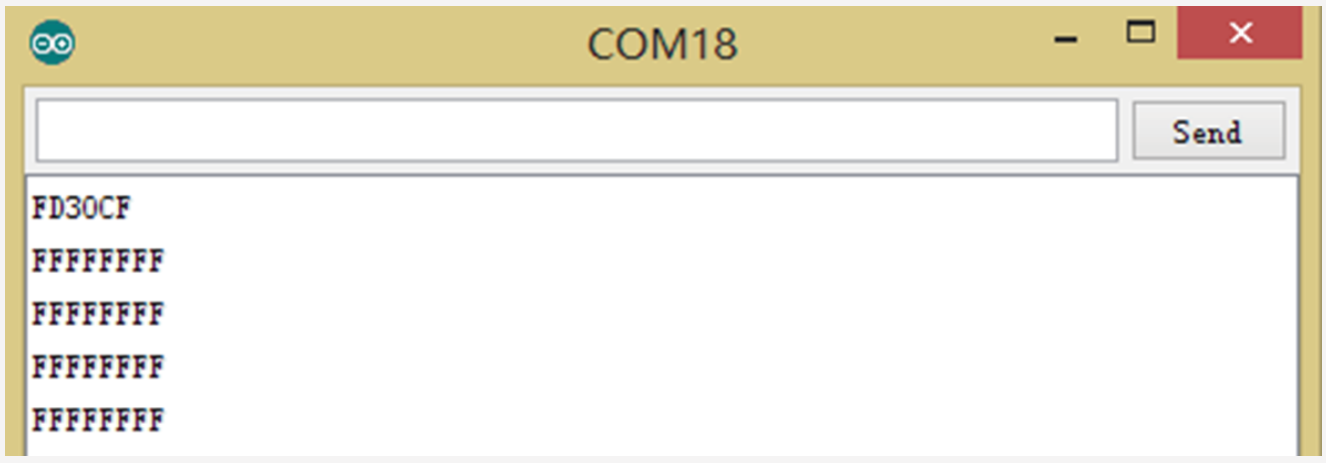
После загрузки кода откройте последовательный монитор IDE Arduino и настройте скорость передачи в бодах 9600 в соответствии с Serial.begin (9600).



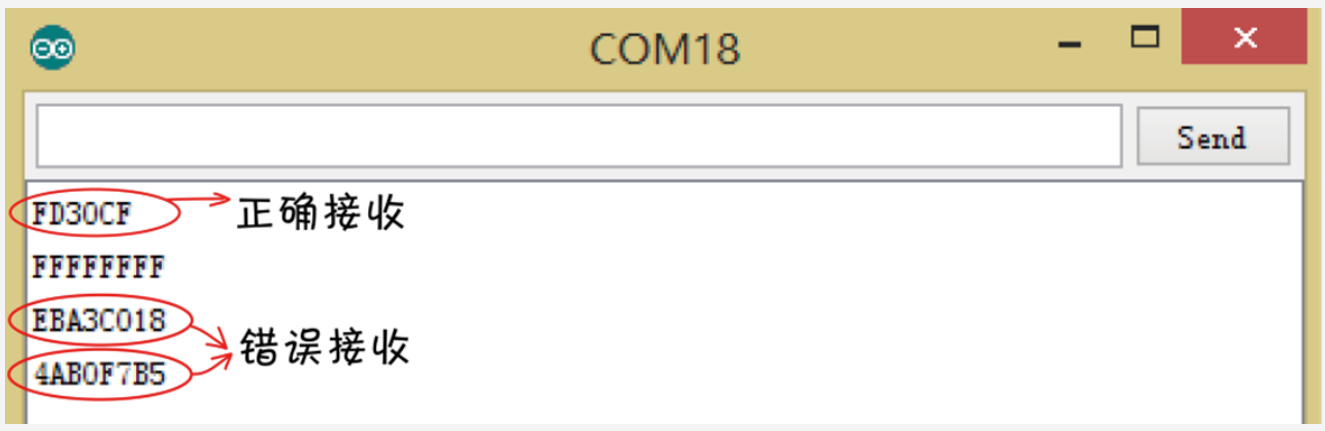
После настройки нажмите кнопку на пульте дистанционного управления в направлении инфракрасного приемника. Каждая кнопка имеет шестнадцатеричный код. Мы можем видеть код на серийном мониторе, независимо от того, какую кнопку мы нажимаем. Например, мы нажимаем кнопку «0», полученный шестнадцатеричный код - FD30CF.



Если вы продолжаете нажимать одну кнопку, последовательный монитор читает «FFFFFFFF»



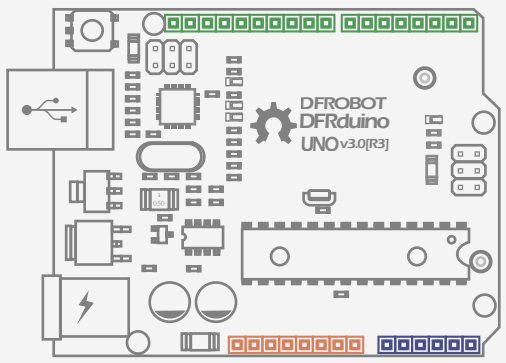
Если он правильно принят в последовательном порту, код должен быть шестизначным, начиная с FD. Если контроллер не отправляет сигнал в инфракрасный приемник, он может получить неправильный код, как мы видим ниже.



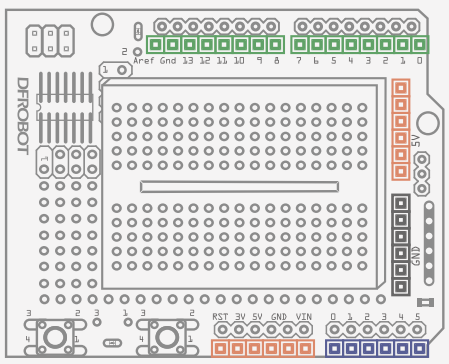
Первоначальное инфракрасное декодирование слишком сложно манипулировать, поэтому мы используем библиотеку, которая построена другими, без ее полного понимания. Поскольку у нас появилась идея декодирования инфракрасного сигнала, сделаем светодиод с инфракрасным управлением.

Светодиод на дистанционном управлении

Компоненты

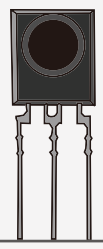


Основная плата DFRduino Uno R3 x1



Основная плата DFRduino Uno R3 x1

Перемычки “мама-мама” x3



ИК-приемник x1

ИК-пульт дистанционного управления x1

Светодиод 5ММ x1

Резистор 220 Ом x1

Подключение

На основе предыдущей схемы добавьте светодиод и резистор. Подключите светодиод к цифровому пину 10 и светодиоду сигнала инфракрасного приемника к цифровому пину 11.

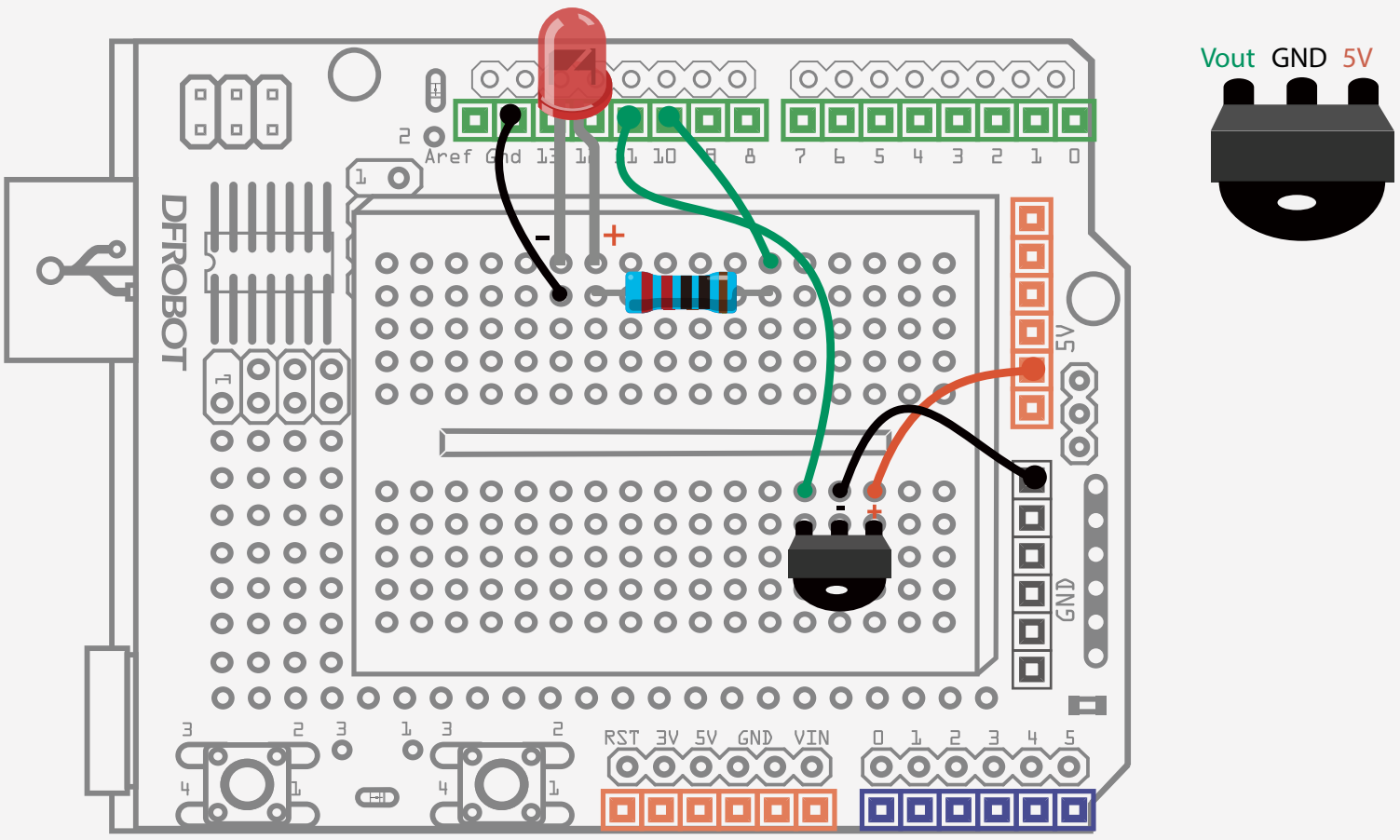


Рис. 14-2. Схема цепи ИК-пульта дистанционного управления.

Код

мы не предлагаем вам вводить код шаг за шагом. вы можете пересмотреть исходный код и посмотреть, что добавлено.

Пример кода 14-2:

```
#include <IRremote.h>
int RECV_PIN = 11;
int ledPin = 10;                    // Светодиодный - цифровой 10
boolean ledState = LOW;             // ledState для хранения состояния светодиода
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup(){
  Serial.begin(9600);
  irrecv.enableIRIn();
  pinMode(ledPin,OUTPUT);           // определять светодиод как выходной сигнал
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);

   //после получения кода от кнопки питания состояние светодиода изменяется от
HIGH до LOW или от LOW до HIGH.
    if(results.value == 0xFD00FF){
        ledState = !ledState;               //задний ход
        digitalWrite(ledPin,ledState);    //изменение состояния светодиода
     }
    irrecv.resume();
  }
}
```

Код

Определение инфракрасного приемника совпадает с последним скетчем.

```
#include <IRremote.h>       //вставляем библиотеку IRremote.h
int RECV_PIN = 11;            //Определим вывод
RECV_PIN 11
IRrecv irrecv(RECV_PIN);      //определить RECV_PIN как инфракрасный
receiver
decode_results results;       //определение переменных результатов
для сохранения результата инфракрасного приемника
int ledPin = 10;              //Светодиодный - цифровой 10
boolean ledState = LOW;       //Ledstate используется для хранения
статуса светодиода
```

В функции setup () мы используем последовательный порт для загрузки инфракрасного декодирования и настройки pinMode цифровых контактов. В основной программе мы тестируем, принимаем инфракрасный сигнал и сохраняем данные в переменной результатов.

```
if (irrecv.decode(&results))
```

Как только Arduino получает данные, программа выполняет две функции: сначала проверяет, получен ли инфракрасный код от кнопки питания.

```
if(results.value == 0xFD00FF)
```

Во-вторых, чтобы изменить состояние светодиода.

```
ledState = !ledState;           //Кувырок
digitalWrite(ledPin,ledState);  //Изменение соответствующего статуса светодиода
```

Возможно, вы не знакомы с «!». «!» - логическое НЕ.

«LedState» - это противоположное состояние «ledState». «!» Используется только в переменной, которая содержит только 2 состояния или логический тип переменной.Затем Arduino будет ждать следующего сигнала.

Упражнение

1.Объедините проект вентилятора с текущим проектом. Добавьте еще одну функцию в мини-контроллер для управления светодиодом и вентилятором.

2.Сделайте самодельный вентилятор с дистанционным управлением, например: небольшая цифра, которая может перемещаться с сервоприводами, управляемыми инфракрасными сигналами.

Проект 15

ИК-пульт дистанционного управления светодиодным модулем

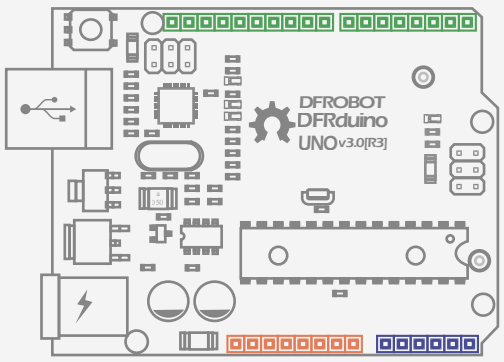
15

DFROBOT
DRIVE THE FUTURE

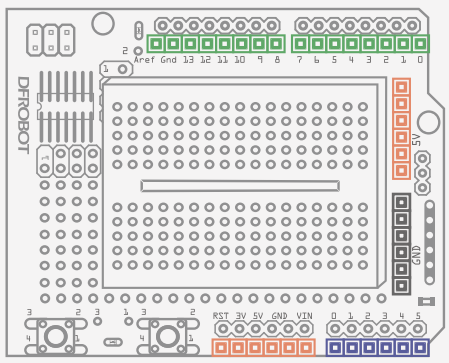
IR Remote Controlled LED Module

8-segment LED is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays. By controlling each LED on the segment connected to a digital pin, numbers can be displayed on this LED.

Компоненты



DFduino UNO R3 x1



Плата расширения x1

Перемычки “мама-мама” x10

Перемычки “папа-мама” x8

8-Segment LED x1

Подключение

7-сегментный индикатор имеет 10 контактов. 5 контактов в верхнем положении подключены к цифровому выходу 2 к цифровому выходу 5. Другие 5 контактов в нижнем положении с десятичной точкой подключены к цифровому выходу 6-9. 8 резисторов включены для ограничения тока для светодиодов.

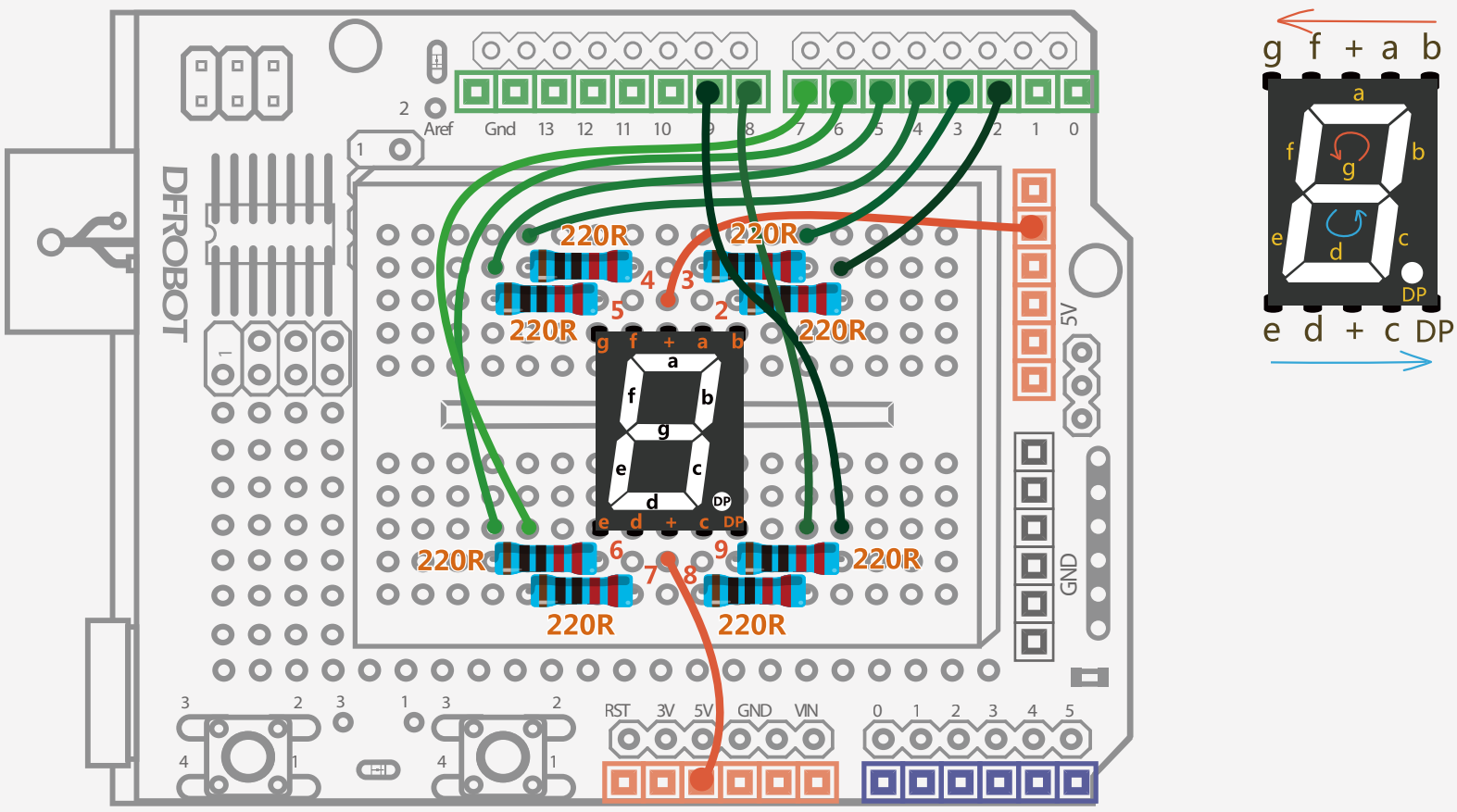


Рис. 15-1. Схема цепи светодиодного дисплея

Код

Пример кода 15-1:

```
//Проект 15 - цифровой дисплей трубки
void setup(){
for(int pin = 2 ; pin <= 9 ; pin++){      // определять цифровой вывод 2-9
как вывод
      pinMode(pin, OUTPUT);
      digitalWrite(pin, HIGH);
  }
}

void loop() {
    // отображать номер 0
    int n0[8]={0,0,0,1,0,0,0,1};
    //отображает массив n0 [8] в цифровом выводе 2-9
    for(int pin = 2; pin <= 9 ; pin++){
       digitalWrite(pin,n0[pin-2]);
    }
    delay(500);

    // отображать номер1
    int n1[8]={0,1,1,1,1,1,0,1};
    // отображать массив n1 [8] в цифровом выводе 2-9
    for(int pin = 2; pin <= 9 ; pin++){
       digitalWrite(pin,n1[pin-2]);
    }
    delay(500);

    // отображать номер 2
    int n2[8]={0,0,1,0,0,0,1,1};
    // отображать массив n2 [8] в цифровом выводе 2-9
    for(int pin = 2; pin <= 9 ; pin++){
       digitalWrite(pin,n2[pin-2]);
    }
delay(500);

    // отображать номер 3
    int n3[8]={0,0,1,0,1,0,0,1};
    // отображать массив n3 [8] в цифровом выводе 2-9
    for(int pin = 2; pin <= 9 ; pin++){
       digitalWrite(pin,n3[pin-2]);
    }
    delay(500);

    // отображать номер 4
    int n4[8]={0,1,0,0,1,1,0,1};
```

```
  // отображать массив n4 [8] в цифровом выводе 2-9
  for(int pin = 2; pin <= 9 ; pin++){
    digitalWrite(pin,n4[pin-2]);
  }
  delay(500);

  // отображать номер 5
  int n5[8]={1,0,0,0,1,0,0,1};
  // отображать массив n5 [8] в цифровом выводе 2-9
  for(int pin = 2; pin <= 9 ; pin++){
    digitalWrite(pin,n5[pin-2]);
  }
  delay(500);

  // отображать номер 6
  int n6[8]={1,0,0,0,0,0,0,1};
  // отображать массив n6 [8] в цифровом выводе 2-9
  for(int pin = 2; pin <= 9 ; pin++){
    digitalWrite(pin,n6[pin-2]);
  }
  delay(500);

  // отображать номер 7
  int n7[8]={0,0,1,1,1,1,0,1};
  // отображать массив n7 [8] в цифровом выводе 2-9
  for(int pin = 2; pin <= 9 ; pin++){
    digitalWrite(pin,n7[pin-2]);
  }
  delay(500);

  // отображать номер 8
  int n8[8]={0,0,0,0,0,0,0,1};
  // отображать массив n8 [8] в цифровом выводе 2-9
  for(int pin = 2; pin <= 9 ; pin++){
    digitalWrite(pin,n8[pin-2]);
  }
  delay(500);

  // отображать номер 9
  int n9[8]={0,0,0,0,1,1,0,1};
  // отображать массив n9 [8] в цифровом выводе 2-9
  for(int pin = 2; pin <= 9 ; pin++){
    digitalWrite(pin,n9[pin-2]);
  }
  delay(500);
}
```

Цепь

8-сегментный индикатор

8-сегментный индикатор имеет 10 контактов: a, b, c, d, e, f, g и DP сегменты (десятичная точка). Каждый из них подключен к цифровым контактам. Контролируя каждый светодиод на подключенном сегменте, на этом светодиоде могут отображаться номера. Другой пин - это общий пин, подключенный к GND (общий катод) или 5V (общий анод).

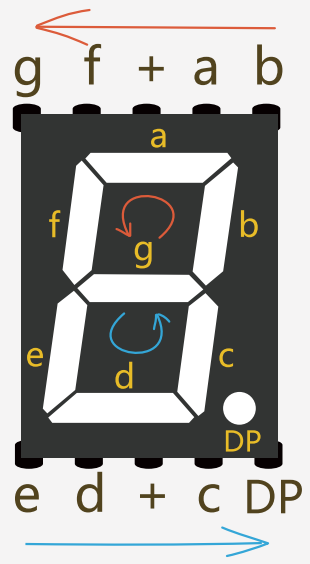


Рис. 15-2 Инструкции для выводов

Расхождения катодов и анодов

Все катоды (отрицательные клеммы) или все аноды (положительные клеммы) сегментных светодиодов подключены и выведены на общий вывод; это называется «общим катодом» или «общим анодным» устройством. Общий вывод светодиодов катодов подключен к GND. Когда вы настраиваете этот пин HIGH, он активирует светодиод. Общий вывод анодных светодиодов подключен к 5V. Когда вы настраиваете этот пин LOW, он выключает светодиод.

Код

8-сегментный светодиод подключен к 8 цифровым пинам, поэтому нам необходимо определить 8 цифровых пинов с петлями «FOR», чтобы упростить работу. Сегменты b, a, f, g, e, d, c, DP соответствуют цифровым пинам от 2 до 9.

```
for(int pin = 2 ; pin <= 9 ; pin++){
    pinMode(pin, OUTPUT);
    digitalWrite(pin, HIGH);
}
```

Установите пины от 2 до 9 в качестве «OUTPUT» и инициализируйте их HIGH.

Основная программа - отобразить число от 0 до 9. Посмотрим на код отображения 0.

```
int n0[8]={0,0,0,1,0,0,0,1};
```

Здесь мы вводим понятие массива. Массив представляет собой набор переменных, к которым обращаются с номером индекса. Мы объявляем массив типа «int», называем его «n0» и назначаем 8 значений для его инициализации. Обратите внимание, что массивы - это нулевые индексы. При объявлении массива типа для хранения нулевого символа требуется еще один элемент, кроме вашей инициализации. Ссылаясь на инициализацию массива выше, четвертый элемент массива имеет индекс 3 (n0 [3]), а его значение равно 1. Восемь элементов массива имеют индекс 7 (n0 [7]), а его значение равно 1 . 8 внутри квадратных скобок (n0 [8]) означает 8 элементов.

После определения массива мы вводим цикл «for».

```
for(int pin = 2; pin <= 9 ; pin++){
    digitalWrite(pin,n0[pin-2]);
}
```

Цикл «for» должен назначать значения HIGH / 1 или LOW / 0 для 2-9 пинов. Например, когда pin = 2, n0 [pin-2] равен n0 [0], который указывает на первый элемент массива со значением 0 / LOW. Таким образом, сегмент b включен. Когда циклы for идут к пину = 3, сегмент включен. Когда циклы for идут к пину = 4, сегмент f включен и т. д.

Процесс цикла ниже:

pin= 2 → n0[0] = 0 → digitalWrite(2,0) → b сегмент включен

pin= 3 → n0[1] = 0 → digitalWrite(3,0) → a сегмент включен

pin= 4 → n0[2] = 0 → digitalWrite(4,0) → f сегмент включен

pin= 5 → n0[3] = 0 → digitalWrite(5,1) → g сегмент отключен

pin= 6 → n0[4] = 0 → digitalWrite(6,0) → e сегмент включен

pin= 7 → n0[5] = 0 → digitalWrite(7,0) → d сегмент включен

pin= 8 → n0[6] = 0 → digitalWrite(8,0) → c сегмент включен

pin= 9 → n0[7] = 1 → digitalWrite(9,1) → DP сегмент отключен

Теперь он отображает 0. Он отображает от 1 до 9 на основе массива. Просто попробуйте его вычеркнуть, и статус понятен когда свет включается и выключается.

Более простой метод основывается на понимании кода 1. Это будет показано на следующей странице.

Код 2

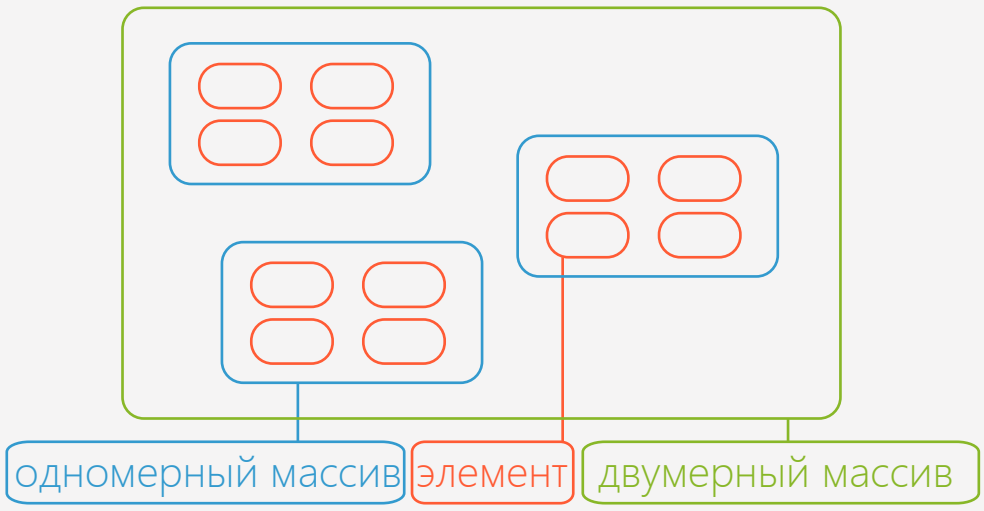
В этом разделе мы представим еще более простой способ манипулирования циклами для чисел от 0 до 9 в 8-сегментном светодиоде. Выше в скетче мы создали 10 массивов для отображения от 0 до 9. Это одномерный массив. Как насчет создания двумерного массива, чтобы сделать его проще?

Пример кода 15-2

```
// Пункт 11 - цифровой дисплей
int number[10][8] =
{
  {0,0,0,1,0,0,0,1},     //отображение 0
  {0,1,1,1,1,1,0,1},     //отображение 1
  {0,0,1,0,0,0,1,1},     //отображение 2
  {0,0,1,0,1,0,0,1},     //отображение 3
  {0,1,0,0,1,1,0,1},     //отображение 4
  {1,0,0,0,1,0,0,1},     //отображение 5
  {1,0,0,0,0,0,0,1},     //отображение 6
  {0,0,1,1,1,1,0,1},     //отображение 7
  {0,0,0,0,0,0,0,1},     //отображение 8
  {0,0,0,0,1,1,0,1}      //отображение 9
};
void numberShow(int i){                          //вызов этой функции для
отображения чисел
   for(int pin = 2; pin <= 9 ; pin++){
       digitalWrite(pin, number[i][pin - 2]);
    }
}
void setup(){
   for(int pin = 2 ; pin <= 9 ; pin++){    //  определять цифровые выходы от
2 до 9 в качестве вывода
       pinMode(pin, OUTPUT);
       digitalWrite(pin, HIGH);
  }
}

void loop() {
   for(int j = 0; j <= 9 ; j++){
     numberShow(j);                          //вызывать функцию numberShow ()
для отображения 0-9display 0-9.
       delay(500);
    }
}
```

Код 2

Одномерный массив состоит из элементов, а двумерный массив состоит из одномерных массивов.



В первом скетче мы манипулировали 10 массивами, каждый из которых состоял из 8 элементов, соответствующих значениям в сегменте от b до DP. В скетче мы будем манипулировать 10 одномерными массивами вместе, чтобы сформировать двумерный массив.

число[0][0]

```
int число[10][8] =
{
  {0,0,0,1,0,0,0,1},    // display0
  {0,1,1,1,1,1,0,1},    // display1
  {0,0,1,0,0,0,1,1},    // display2
  {0,0,1,0,1,0,0,1},    // display3
  {0,1,0,0,1,1,0,1},    // display4
  {1,0,0,0,1,0,0,1},    // display5
  {1,0,0,0,0,0,0,1},    // display6
  {0,0,1,1,1,1,0,1},    // display7
  {0,0,0,0,0,0,0,1},    // display8
  {0,0,0,0,1,1,0,1}     // display9
};
```

число[9][7]

Это двумерный массив. Он начинается с символа «null». Поэтому число [0] [0] является первым элементом в первом массиве, значение которого равно 0. Число [9] [7] - это восемь элементов в десятом массиве, значение которого равно 1.

```
void numberShow(int i){
// эта функция предназначена для
отображения чисел
   for(int pin = 2; pin <= 9 ; pin++)
        digitalWrite(pin, number[i][pin - 2]);
}

void loop() {
   for(int j = 0; j <= 9 ; j++){
        numberShow(j);
// вызывать функцию numberShow () для
отображения от 0 до 9
        delay(500);
   }
}
```

В основной программе циклы «for» манипулируют переменной «j» с циклом от 0 до 9. Как только «j» получает значение со значением, функция «numberShow ()» работает соответственно.Функция «numberShow ()» работает следующим образом:

Когда программа начинается с j = 0, «numberShow (j)» - «numberShow (0)». Когда он переходит к функции «numberShow ()», значение i равно 0, а начальное значение этого вывода равно 2. Значение digitalWrite (0 - digitalWrite (2, число [0] [0]), значение которого равно 0. digitalWrite (2,0) означает настройку вывода 2 как LOW, поэтому светодиод сегмента b HIGH. После этого петли переходят к выводу = 3, pin = 4, ..., пока контакт = 9, когда весь цикл цикла и 0 на 8-сегментном светодиоде.

Сравните, как мы отображали 0 в первом скетче.

```
// display number 0
int n0[8]={0,0,0,1,0,0,0,1};
// display the array of n0[8] in digital pin 2-9
for(int pin = 2; pin <= 9 ; pin++){
  digitalWrite(pin,n0[pin-2]);
 }
```

Принцип назначения значения для вывода с 2 по 9 для управления сегментом от b до DP является тем же самым.

Когда цикл numberShow (0) закончен, он возвращается к функции цикла цикла.

1→ numberShow(1)→ i =1 → number[1][pin-2] → display 1

j=2 → numberShow(2) → i=2 → number[2][pin-2] → display 2

j=3 → numberShow(3) → i=3 → number[3][pin-2] → display 3

……

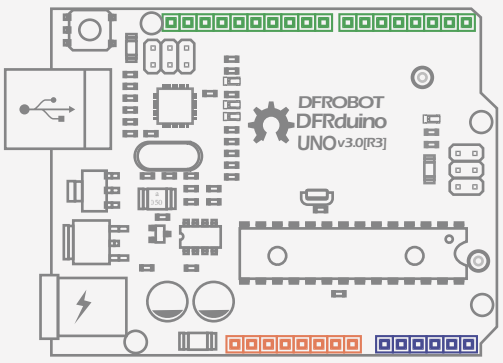
j=3 → numberShow(9) → i=9 → number[9][pin-2] → displays 9

Вышеприведенный анализ для кодов. Подумайте только о различиях между одномерным массивом и двухмерным массивом и о том, как работают коды.

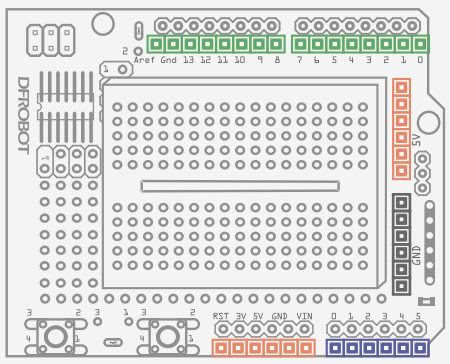
Комбинированное применение светодиодного модуля и ИК-приемника должно быть достигнуто после обработки их принципов работы. Совместимый контроллер Arduino обрабатывает и передает сигнал, передаваемый с пульта дистанционного управления Mini, в ИК-приемник на светодиодный модуль.

Номера с 0 до 9 на мини-пульте дистанционного управления должны отображаться в соответствующих положениях модуля светодиодов. Кроме того, возможны уменьшающиеся и увеличивающиеся функции.

Компоненты



DFduino UNO R3 x1

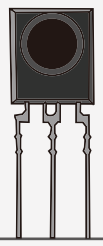


Плата расширения x1

Перемычки М/М x13

Резистор 220 Ом x8

7-сегментный светодиод x1



ИК-приемник x1

ИК-пульт дистанционного управления x1

Подключение

В этом проекте мы собираемся просто объединить аппаратное обеспечение проекта 14 и проекта 15. Есть несколько изменений. Если есть проблемы со светодиодными модулями, вернитесь назад и просмотрите Project 15.

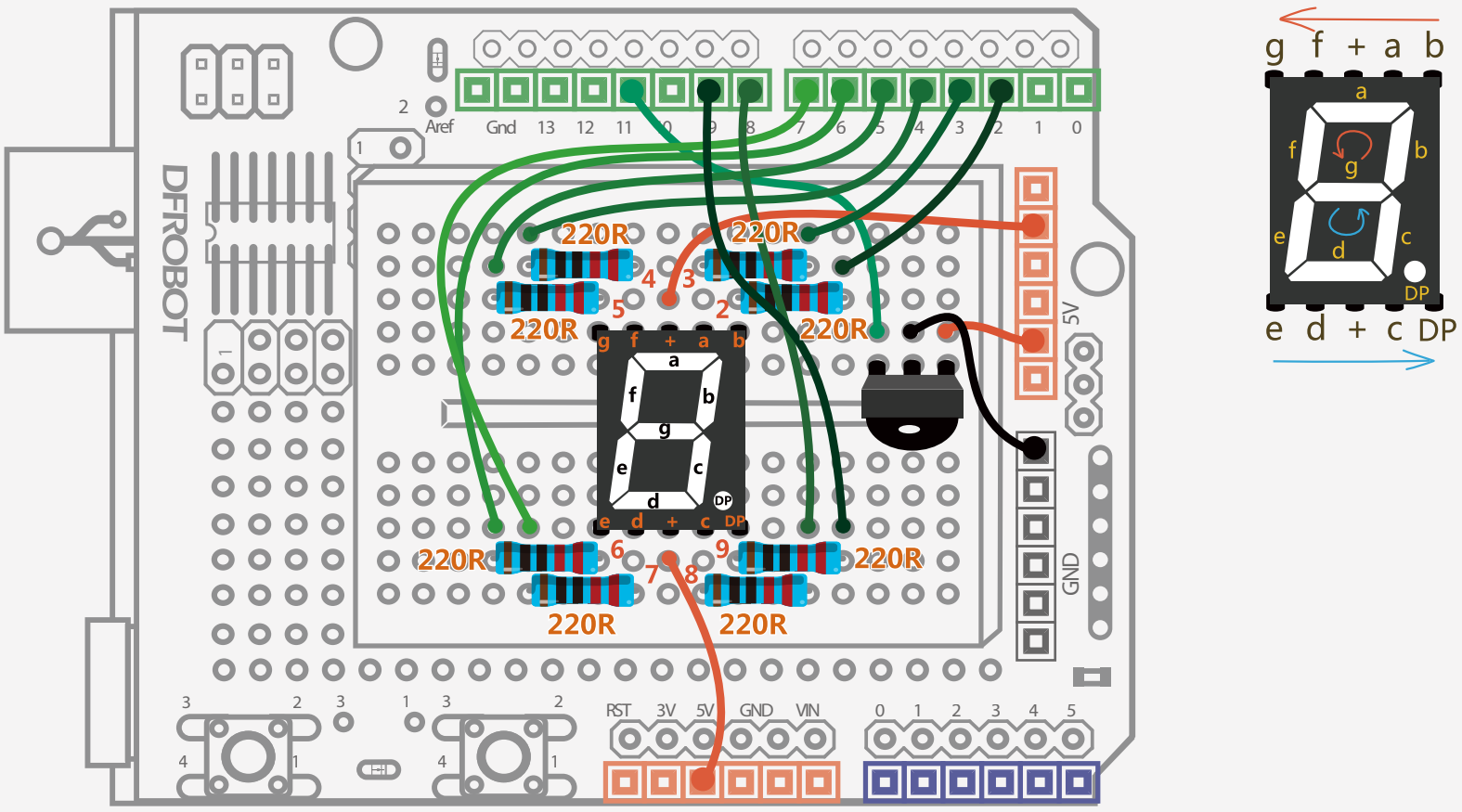


Рис. 15-3. ИК-пульт дистанционного управления.

Код

Пример кода 13-1

```
Проект 13 - ИК-модуль с дистанционным управлением
#include <IRremote.h>     // Вызовите библиотеку IRremote.h.
int RECV_PIN = 11;          // Dim RECV_PIN как 11
IRrecv irrecv(RECV_PIN);    // Установите RECV_PIN (Pin-out 11) в качестве
порта ИК-приемника.
decode_results results;     // Устанавливаем результаты как позицию
хранения ИК-записей.
int currentNumber = 0;     // Эта переменная предназначена для хранения
текущего номера.

long codes[12]=             // Массив предназначен для хранения IR-кодов,
поставляемого ИК-пультом.
{
  0xFD30CF,0xFD08F7,     // 0 ,1
  0xFD8877,0xFD48B7,     // 2 ,3
  0xFD28D7,0xFDA857,     // 4 ,5
  0xFD6897,0xFD18E7,     // 6 ,7
  0xFD9867,0xFD58A7,     // 8 ,9
  0xFD20DF,0xFD609F,     // + ,-
};

int number[10][8] =         // Массив для хранения чисел, отображаемых на
светодиодном модуле.
{
 {0,0,0,1,0,0,0,1},//0
 {0,1,1,1,1,1,0,1},//1
 {0,0,1,0,0,0,1,1},//2
 {0,0,1,0,1,0,0,1},//3
 {0,1,0,0,1,1,0,1},//4
 {1,0,0,0,1,0,0,1},//5
 {1,0,0,0,0,0,0,1},//6
 {0,0,1,1,1,1,0,1},//7
 {0,0,0,0,0,0,0,1},//8
 {0,0,0,0,1,1,0,1},//9
};

void numberShow(int i) {   //  Функция работает для отображения номеров
на модуле СИД.
for(int pin = 2; pin <= 9 ; pin++){
  digitalWrite(pin, number[i][pin - 2]);
  }
}
void setup(){
  Serial.begin(9600);         // Установите скорость передачи в бодах как
9600.
  irrecv.enableIRIn();         // Запуск ИК-декодирования.
```

```
for(int pin = 2 ; pin <= 9 ; pin++){    //В качестве выходного режима
установите Pin-out 2 - 9.
  pinMode(pin, OUTPUT);
  digitalWrite(pin, HIGH);
  }
}
    void loop() {
  //Оцениваем, принимаются ли данные декодирования и хранятся
данные в переменных результатах.
  if (irrecv.decode(&results)) {
      for(int i = 0; i <= 11; i++){
        // Оценим, принимаются ли ИК-коды от кнопки 0 до 9.
        if(results.value == codes[i]&& i <= 9){
            numberShow(i);                    //Отображение от 0 до 9
соответствующее положение светодиодного модуля.
            currentNumber = i;                //Присвоить значение,
отображаемое для переменная currentNumber.

            Serial.println(i);
            break;
        }

        // Оценим, принимаются ли уменьшающиеся ИК-коды, а текущее
значение не равно 0.
        else if(results.value == codes[10]&& currentNumber != 0){
            currentNumber--;                  //Уменьшить текущее значение.
            numberShow(currentNumber);  //Светодиодный модуль
отображает уменьшающееся значение.
            Serial.println(currentNumber);   //Последовательный порт
выводит уменьшающееся значение.
            break;
        }

        //Оцениваем, принимаются ли уменьшающиеся ИК-коды, а
текущее значение не равно 9.
        else if(results.value == codes[11]&& currentNumber != 9){
            currentNumber++;                  //Текущее значение
увеличивается.
            numberShow(currentNumber);  //Светодиодный модуль
отображает увеличенное значение.
            Serial.println(currentNumber);   //Последовательный порт
выводит увеличивающееся значение.
            break;
        }
    }

  Serial.println(results.value, HEX);          //Просмотр ИК-кодов через
монитор последовательного порта
  irrecv.resume();                             //Подождите следующего
сигнала.
  }
}
```

Рисунок 15-4. Инструкции по нажатию кнопки

Коды загружены, попробуйте нажать несколько кнопок, как показано на рисунке 15-4, чтобы проверить состояние светодиодного модуля.

Обзор кода

Код ниже начинается с общих заявлений для ИК-приемника. Просто скопируйте проект 12.

```
#include <IRremote.h>     // Вызовите библиотеку Iremotw.h.
int RECV_PIN = 11;          // DimRECV_PIN как 11.
IRrecv irrecv(RECV_PIN);    // Установите RECV_PIN (Pin-out 11) в качестве порта ИК-приемника.
decode_results results;     // Устанавливаем результаты в качестве места хранения записей ИК-записей.
int currentNumber = 0;     // Эта переменная предназначена для хранения текущего номера.
```

Здесь есть дополнительная переменная: «currentNumber». Эта переменная предназначена для хранения текущих значений. Число, увеличивающее и уменьшающееся, соответствует соответствующей ссылке.

Для хранения этих ИК-кодов применяется массив. «0x-» представляет шестнадцатеричный. Заменяемые ИК-коды работают для реализации дополнительных функций с помощью кнопок addsl на пульте дистанционного управления.

Первый шаг :

Необходимо учитывать три условия. Во-первых, 7-сегментный светодиодный модуль должен отображать от 0 до 9 при нажатии соответствующих кнопок нажатия. Во-вторых, каждый раз, когда нажимается «Назад», количество отображаемых номеров уменьшается на 1 каждый раз, пока оно не достигнет 0. В-третьих, каждый раз, когда нажата кнопка «Вперед», отображается номер увеличивается на 1 каждый раз, пока не достигнет 0.Чтобы принять решение по вышеуказанным условиям, мы принимаем утверждение if. Что отличается от предыдущих примеров кода, мы используем if ... else if.i Какая разница? Оценочное выражение обычно следует за другим, если не для else. Как бы то ни было, ни то, ни другое, если не работает независимо, но оператор if обязательно.

Вернитесь к кодам для трех условий, указанных ниже:

```
if (results.value == codes[i]&& i <= 9)
if (results.value == codes[10]&& currentNumber != 0)
if (results.value == codes[11]&& currentNumber != 9)
```

Первое «if» оценивает первое условие, отображающее от 0 до 9. Оценка основана на том, получены ли данные от результатов. Значение - это ИК-коды кодов массивов [0] ~ кодов [9].Второе «if» оценивает второе условие: принимаются ли команды « Backward » (код [10] = 0xFD20DF), а текущий номер отображается не 9.Другой вопрос - как подтвердить элементы массива. Следовательно, перед оценкой «if» должен быть цикл «за», чтобы иметь переменный цикл от 0 до 11.После подтверждения определенных ИК-кодов программа начнет выполнение второго события. Должен быть соответствующий исполняемый код, следующий за каждым оператором if.Ввод кода теперь завершен. Это самый сложный раздел во всех проектах, которые мы продемонстрировали, и это трудно понять в первый раз, но вы можете расслабиться – практика приносит мастерство.

Упражнения

Создайте дистанционный контроллер, который работает на основе этого проекта.
Используя эти приложения, вы можете сделать подвижного игрушечного человека. Вы можете использовать сервоприводы, описанные в проектах, упомянутых ранее, и использовать разные кнопки на пульте дистанционного управления, чтобы сервомашины перемещались под различные углы. Используйте свое воображение и получайте удовольствие.