СОДЕРЖАНИЕ

Что такое Arduino 5
Установка Arduino IDE 8
1 Светодиод. Мигаем светодиодом 12
2. Кнопка. Обрабатываем нажатие кнопки на примере зажигания светодиода. Боремся с дребезгом 15
3 Потенциометр. Показываем закон Ома на примере яркости светодиода 20
4 Светодиодная шкала 10 сегментов. Крутим потенциометр, меняем количество светящихся светодиодов 23
5 RGB-светодиод. Широтно-импульсная модуляция. Переливаемся цветами радуги 28
6 Семисегментный индикатор одноразрядный. Выводим цифры 33
7 Матрица 4-разрядная из 7-сегментных индикаторов. Делаем динамическую индикацию 36
8 Микросхема сдвигового регистра 74HC595. Управляем матрицей из 4 разрядов, экономим выводы Ардуино 42
9 Матрица светодиодная 8x8 46
10 Пьезоизлучатель. Управляем пьезоизлучателем: меняем тон, длительность, играем Имперский марш 49
11 Транзистор MOSFET. Показываем усилительные качества транзистора. На примере электродвигателя изменяем обороты 54
12 Реле. Управляем реле через транзистор 57
13 Фоторезистор. Обрабатываем освещенность, зажигая или гася светодиоды 61
14 Датчик температуры аналоговый LM335. Принцип работы, пример работы 65
15 Индикатор LCD1602. Принцип подключения, вывод информации на него 68
16 Графический индикатор на примере Nokia 5110 72
17 Сервопривод. Крутим потенциометр, меняем положение 76
18 Джойстик. Обрабатываем данные от джойстика. Управление Pan/Tilt Bracket с помощью джойстика 80
19 Шаговый двигатель 4-фазный, с управлением на ULN2003 (L293) 84
20 Датчик температуры DS18B20 88
21 Датчик влажности и температуры DHT11 92

22 Датчики газов. Принцип работы, пример работы 96
23 Ультразвуковой датчик расстояния HC-SR04. Принцип работы, подключение, пример 99
24 3-осевой гироскоп + акселерометр на примере GY-521 103
25 ИК-фотоприемник и ИК-пульт. Обрабатываем команды от пульта 106
26 Часы реального времени. Принцип работы, подключение, примеры 110
27 SD-карта. Чтение и запись данных 116
28 Считыватель RFID на примере RC522. Принцип работы, подключение, примеры 120
29 Работа с Интернетом на примере Arduino Ethernet shield W5100 126
30 Беспроводная связь. Модуль Wi-Fi ESP8266 131
31 Беспроводная связь. Модуль Bluetooth HC-05 137
32 Беспроводная связь. Модуль GSM/GPRS SIM900 142
33 Модуль GPS. Принцип работы, подключение, примеры 147
Встроенные функции языка Arduino 152

Что такое Arduino

Arduino – это электронный конструктор и удобная платформа быстрой разработки электронных устройств для новичков и профессионалов. Платформа пользуется огромной популярностью во всем мире благодаря удобству и простоте языка программирования, а также открытой архитектуре и программному коду. Плата Arduino состоит из микроконтроллера Atmel AVR и элементов обвязки для программирования и интеграции с другими схемами. На многих платах присутствует линейный стабилизатор напряжения +5 В или +3,3 В. Тактирование осуществляется на частоте 16 или 8 МГц кварцевым резонатором (в некоторых версиях – керамическим резонатором). В микроконтроллер предварительно прошивается загрузчик Boot-Loader, поэтому внешний программатор не нужен. Устройство программируется через USB без использования программаторов.

Существует несколько версий платформ Arduino. Версия Leonardo базируется на микроконтроллере ATmega32u4. Uno, Nano, Duemilanove построены на микроконтроллере Atmel ATmega328. Старые версии платформы Diecimila и первая рабочая Duemilanoves были разработаны на основе Atmel ATmega168. Arduino Mega2560, в свою очередь, построена на микроконтроллере ATmega2560. А самые последние версии Arduino Due – на базе микропроцессора Cortex.

Версия UNO (рис. 1) является одной из самых популярных и широко используемой для небольших проектов.

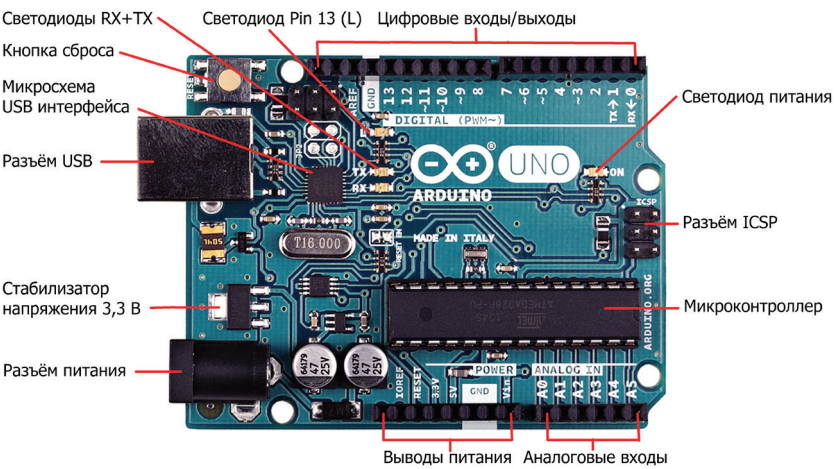

Рис. 1. Плата Arduino UNO

Характеристики платы Arduino UNO показаны в табл. 1.1.

Таблица 1.1

Микроконтроллер	ATmega328
Рабочее напряжение	5 В
Напряжение питания (рекомендуемое)	7–12 В
Напряжение питания (предельное)	6–20 В
Цифровые входы/выходы	14 (из них 6 могут использоваться в качестве ШИМ-выходов)
Аналоговые входы	6
Максимальный ток одного вывода	40 мА
Максимальный выходной ток вывода 3.3 В	50 мА
Flash-память	32 КБ (ATmega328), из которых 0,5 КБ используются загрузчиком
SRAM	2 КБ (ATmega328)
EEPROM	1 КБ (ATmega328)
Тактовая частота	16 МГц

Каждый из 14 цифровых выводов может работать в качестве входа или выхода. Уровень напряжения на выводах ограничен 5 В. Максимальный ток, который может отдавать или потреблять один вывод, составляет 40 мА. Все выводы сопряжены с внутренними подтягивающими резисторами (по умолчанию отключенными) номиналом 20–50 кОм. Помимо этого, некоторые выводы Ардуино могут выполнять дополнительные функции:

- последовательный интерфейс: выводы 0 (RX) и 1 (TX);
- внешние прерывания: выводы 2 и 3;
- ШИМ: выводы 3, 5, 6, 9, 10 и 11 могут выводить 8-битные аналоговые значения в виде ШИМ-сигнала;
- интерфейс SPI: выводы 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK);
- светодиод: 13. Встроенный светодиод, подсоединенный к выводу 13.

В Arduino Uno есть 6 аналоговых входов (A0–A5), каждый из которых может представить аналоговое напряжение в виде 10-битного числа (1024 различных значения). По умолчанию измерение напряжения осуществляется относительно диапазона от 0 до 5 В. Тем не менее верхнюю границу этого диапазона можно изменить, используя вывод AREF и функцию analogReference(). Некоторые из аналоговых входов имеют дополнительные функции:

- TWI: вывод A4 или SDA и вывод A5 или SCL.

В Arduino Uno есть восстанавливаемые предохранители, защищающие USB-порт компьютера от коротких замыканий и перегрузок. Несмотря на то что большинство компьютеров имеют собственную защиту, такие предохранители обеспечивают дополнительный уровень защиты. Если от USB-порта потребляется ток более 500 мА, предохранитель автоматически разорвет соединение до устранения причин короткого замыкания или перегрузки.

Установка Arduino IDE

Разработка собственных приложений на базе плат, совместимых с архитектурой Arduino, осуществляется в официальной бесплатной среде программирования Arduino IDE. Среда предназначена для написания, компиляции и загрузки собственных программ в память микроконтроллера, установленного на плате Arduino-совместимого устройства. Основой среды разработки является язык Processing/Wiring – это фактически обычный C++, дополненный простыми и понятными функциями для управления вводом/выводом на контактах. Существуют версии среды для операционных систем Windows, Mac OS и Linux.

Последнюю версию среды Arduino можно скачать со страницы загрузки официального сайта http://arduino.cc/en/Main/Software.

Рассмотрим установку Arduino IDE на компьютере с операционной системой Windows. Отправляемся на страницу http://arduino.cc/en/Main/Software, выбираем версию для операционной системы Windows и скачиваем архивный файл. Он содержит все необходимое, в том числе и драйверы. По окончании загрузки распаковываем скачанный файл в удобное для себя место.

Теперь необходимо установить драйверы. Подключаем Arduino к компьютеру. На контроллере должен загореться индикатор питания – зеленый светодиод. Windows начинает попытку установки драйвера, которая заканчивается сообщением «Программное обеспечение драйвера не было установлено». Открываем Диспетчер устройств. В составе устройств находим значок Arduino Uno – устройство отмечено восклицательным знаком. Щелкаем правой кнопкой мыши на значке Arduino Uno и в открывшемся окне выбираем пункт **Обновить драйверы** и далее пункт **Выполнить поиск драйверов на этом компьютере**. Указываем путь к драйверу – ту папку на компьютере, куда распаковывали скачанный архив. Пусть это будет папка drivers каталога установки Arduino – например, C:\arduino-1.0\drivers. Игнорируем все предупреждения Windows и получаем в результате сообщение **Обновление программного обеспечения для данного устройства завершено успешно**. В заголовке окна будет указан и COM-порт, на который установлено устройство. Теперь можно запускать Arduino IDE.

Среда разработки Arduino (см. рис. 2) состоит из:

- редактора программного кода;
- области сообщений;

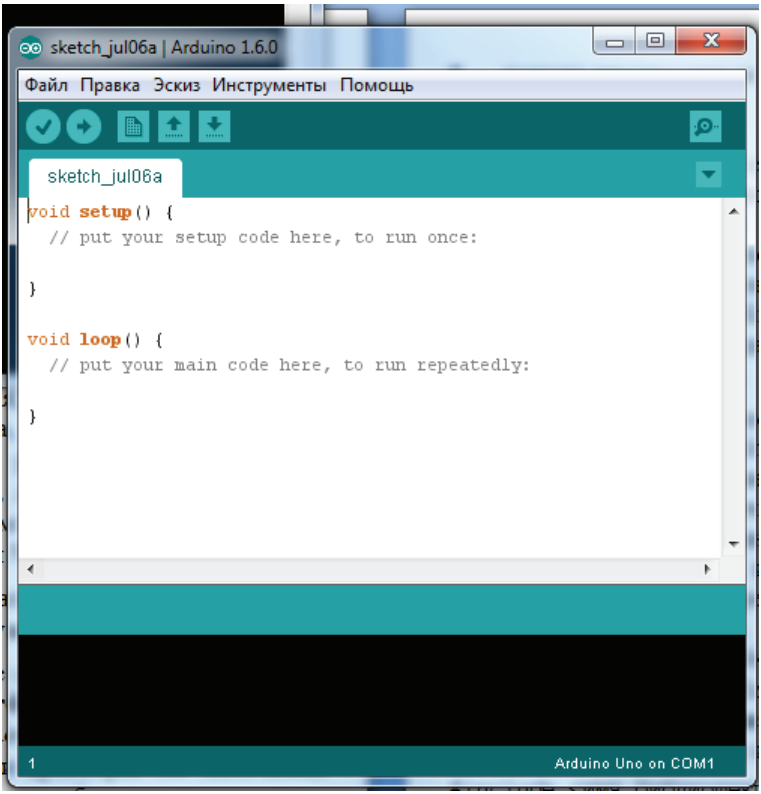

Рис. 2. Среда Arduino IDE

- окна вывода текста;
- панели инструментов с кнопками часто используемых команд;
- нескольких меню.

Программа, написанная в среде Arduino, носит название *скетч*. Скетч пишется в текстовом редакторе, который имеет цветовую подсветку создаваемого программного кода. Во время сохранения и экспорта проекта в области сообщений появляются пояснения и информация об ошибках. Окно вывода текста показывает сообщения Arduino, включающие полные отчеты об ошибках и другую информацию. Кнопки панели инструментов позволяют проверить и записать программу, создать, открыть и сохранить скетч, открыть мониторинг последовательной шины.

Разрабатываемым скетчам дополнительная функциональность может быть добавлена с помощью библиотек, представляющих собой специальным образом оформленный программный код, реализующий некоторый функционал, который можно подключить к создаваемому проекту. Специализированных библиотек существует множество. Обычно библиотеки пишутся так, чтобы упростить решение той или иной задачи и скрыть от разработчика детали программно-аппаратной реализации. Среда Arduino IDE поставляется с набором стандартных библиотек. Они находятся в подкаталоге libraries каталога установки Arduino. Необходимые библиотеки могут быть также загружены с различных ресурсов. Если библиотека установлена правильно, то она появляется в меню **Эскиз | Импорт библиотек**. Выбор библиотеки в меню приведет к добавлению в исходный код строчки

```
#include <имя библиотеки.h>
```

Эта директива подключает заголовочный файл с описанием объектов, функций и констант библиотеки, которые теперь могут быть использованы в проекте. Среда Arduino будет компилировать создаваемый проект вместе с указанной библиотекой.

Перед загрузкой скетча требуется задать необходимые параметры в меню **Инструменты | Плата** (Tools | Board) (рис. 3) и **Инструменты | Последовательный порт** (рис. 4).

Современные платформы Arduino перезагружаются автоматически перед загрузкой. На старых платформах необходимо нажать кнопку перезагрузки. На большинстве плат во время процесса загрузки будут мигать светодиоды RX и TX.

При загрузке скетча используется загрузчик (bootloader) Arduino – небольшая программа, загружаемая в микроконтроллер на плате. Она позволяет загружать программный код без использования дополнительных аппаратных средств. Работа загрузчика распознается по миганию светодиода на цифровом выводе D13.

Монитор последовательного порта (Serial Monitor) отображает данные, посылаемые в платформу Arduino (плату USB или плату последовательной шины).

Теперь, когда мы немного узнали об Arduino и среде программирования Arduino IDE, перейдем к практическим занятиям – экспериментам.

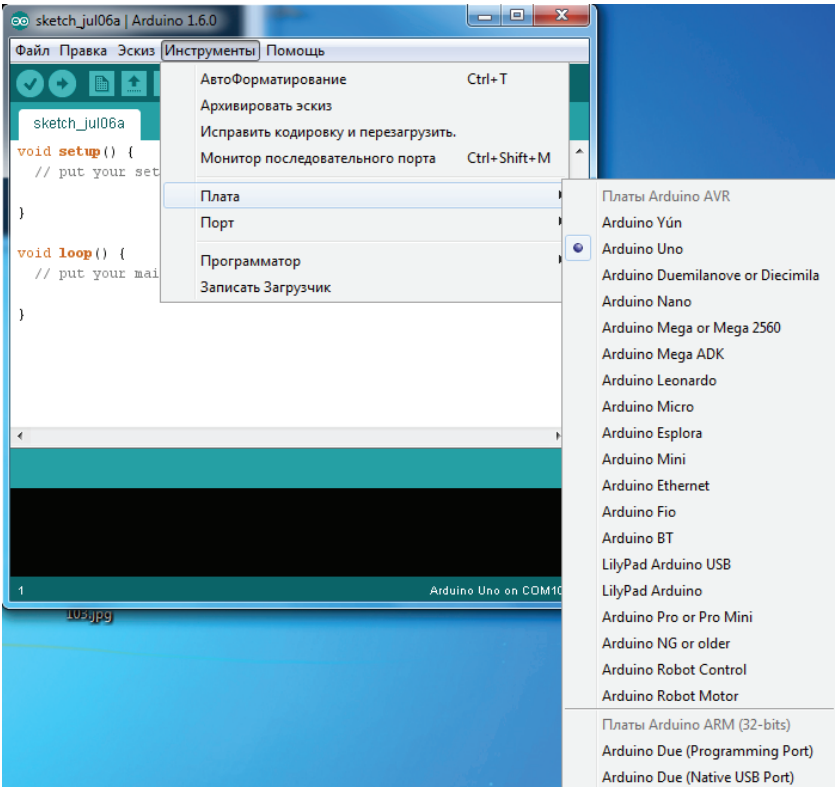

Рис. 3. Выбор Arduino платы

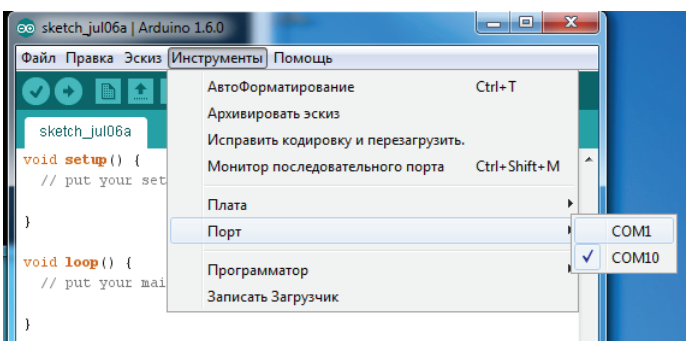

Рис. 4. Выбор порта подключения платы Arduino

1 Светодиод. Мигаем светодиодом

В этом эксперименте мы научимся управлять светодиодом. Заставим его мигать.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- светодиод;
- резистор 220 Ом;
- провод папа-папа.

Светодиод – это полупроводниковый прибор, преобразующий электрический ток непосредственно в световое излучение. По-английски светодиод называется light emitting diode, или LED. Цветовые характеристики светодиодов зависят от химического состава использованного в нем полупроводника. Светодиод излучает в узкой части спектра, его цвет чист, что особенно ценят дизайнеры. Светодиод механически прочен и исключительно надежен, его срок службы может достигать 100 тысяч часов, что почти в 100 раз больше, чем у лампочки накаливания, и в 5–10 раз больше, чем у люминесцентной лампы. Наконец, светодиод – низковольтный электроприбор, а стало быть, безопасный.

Светодиоды поляризованы, имеет значение, в каком направлении подключать их. Положительный вывод светодиода (более длинный) называется анодом, отрицательный – катодом. Как и все диоды, светодиоды позволяют току течь только в одном направлении – от анода к катоду. Поскольку ток протекает от положительного к отрицательному, анод светодиода должен быть подключен к цифровому сигналу 5 В, а катод должен быть подключен к земле.

Мы будем подключать светодиод к цифровому контакту D10 Arduino последовательно с резистором. Светодиоды должны быть всегда соединены последовательно с резистором, который выступает в качестве ограничителя тока. Чем больше значение резистора, тем больше он ограничивает ток. В этом эксперименте мы используем резистор номиналом 220 Ом. Схема подключения приведена на рис. 1.1.

Как подобрать ограничительный резистор и как будет влиять номинал резистора на яркость светодиода, мы рассмотрим в эксперименте 3.

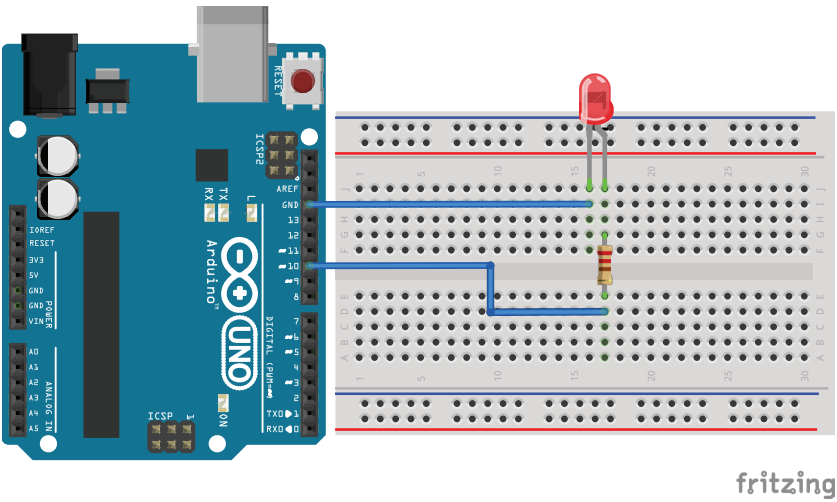

Рис. 1.1. Схема подключения светодиода

Светодиод последовательно с резистором подключаем к цифровому выводу Arduino D10. По умолчанию все выводы Arduino сконфигурированы как входы. Мы собираемся использовать вывод Arduino как выход, поэтому необходимо его переконфигурировать, выдав контроллеру соответствующую команду.

```
pinMode(10,OUTPUT);
```

Для мигания светодиода необходимо попеременно с определенным интервалом подавать на вывод Arduino сигналы HIGH (высокий уровень или 1) и LOW (низкий уровень или 0). Интервал изменения сигнала на выходе D10 Arduino будем устанавливать с помощью команды delay(), задерживающей выполнение скетча на заданное время в миллисекундах (мс).

Скетч эксперимента приведен в листинге 1.1.

Листинг 1.1

```
const int LED=10;     // вывод для подключения светодиода 10 (D10)
void setup()
   {
   // Конфигурируем вывод подключения светодиода как выход (OUTPUT)
   pinMode(LED, OUTPUT);
   }
void loop()
```

```
{
  // включаем светодиод, подавая на вывод 1 (HIGH)
  digitalWrite(LED,HIGH);
  // пауза 1 сек (1000 мс)
  delay(1000);
  // выключаем светодиод, подавая на вывод 0 (LOW)
  digitalWrite(LED,LOW);
  // пауза 1 сек (1000 мс)
  delay(1000);
}
```

Порядок подключения:

1. Длинную ножку светодиода (анод) подключаем к цифровому выводу D10 Arduino, другую (катод) – через резистор 220 Ом к выводу GND (см. рис. 1.1).
2. Загружаем в плату Arduino скетч из листинга 1.1.
3. Наблюдаем процесс мигания светодиода.

Теперь мы можем поэкспериментировать с периодом мигания светодиода, меняя в скетче значения задержки в функции delay(). На странице http://arduino-kit.ru/0001 вы можете посмотреть видеоурок данного эксперимента и скачать скетч, представленный в листинге 1.1.

2 Кнопка. Обрабатываем нажатие кнопки на примере зажигания светодиода. Боремся с дребезгом

Это эксперимент по работе с кнопкой. Мы будем включать светодиод по нажатии кнопки и выключать по отпускании кнопки. Рассмотрим понятие дребезга и программные методы его устранения.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- кнопка;
- светодиод;
- резистор 220 Ом;
- резистор 10 кОм;
- провода папа-папа.

В данном эксперименте мы будем использовать контакт D2 Arduino в качестве входа. Это позволяет подключить к нему кнопку для взаимодействия с проектом в режиме реального времени. При использовании Arduino в качестве входов используют pull-up- и pull-down-резисторы, чтобы вход Arduino не находился в «подвешенном» состоянии (в этом состоянии он будет собирать внешние наводки и принимать произвольные значения), а имел заранее известное состояние (0 или 1). Резисторы pull-up подтягивают вход к питанию +5 В, pull-down-резисторы подтягивают вход к GND. Кроме этого, pull-up- и pull-down-резисторы гарантируют, что кнопка не создаст короткого замыкания между +5 В и землей при нажатии. В нашем эксперименте для подключения кнопки мы будем использовать pull-down-резистор. Схема подключения представлена на рис. 2.1. Когда кнопка отключена, вход D2 будет подтянут к «земле» через резистор номиналом 10 кОм, который будет ограничивать поток тока, и на входном контакте будет установлено значение напряжения `LOW`. При нажатии на кнопку входной контакт напрямую связан с 5 В. Большая часть тока будет протекать по пути наименьшего сопротивления через замкнутую кнопку, и на входе генерируется уровень `HIGH`. При

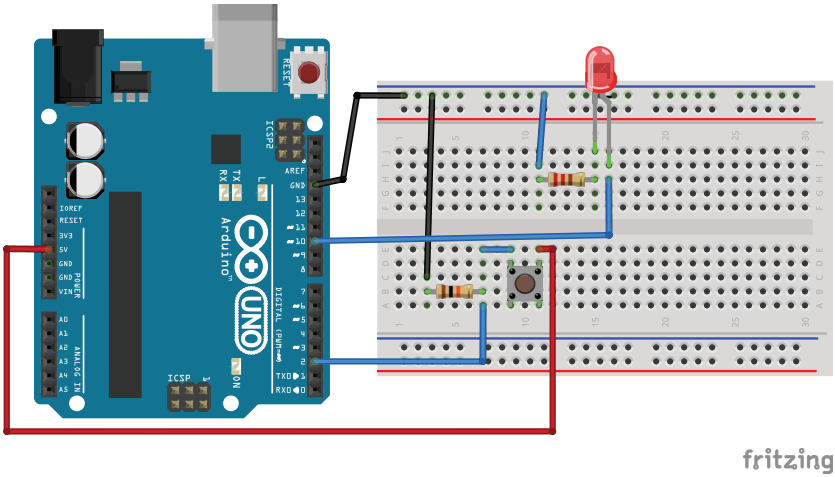

Рис. 2.1. Схема подключения кнопки и светодиода

нажатии на кнопку включаем светодиод, при отпускании – гасим. Код данного скетча приведен в листинге 2.1.

Листинг 2.1

```
const int LED=10;     // Контакт 10 для подключения светодиода
const int BUTTON=2;   // Контакт 2 для подключения кнопки
void setup()
  {
    // Сконфигурировать контакт светодиода как выход
    pinMode (LED, OUTPUT);
    // Сконфигурировать контакт кнопки как вход
    pinMode (BUTTON, INPUT);
  }
void loop()
  {
  if (digitalRead(BUTTON) == LOW)
     {
     // включаем светодиод, подавая на вывод 1 (HIGH)
     digitalWrite(LED, HIGH);
     }
  else
     {
     // выключаем светодиод, подавая на вывод 0 (LOW)
     digitalWrite(LED, LOW);
     }
  }
```

Порядок подключения:

1. Длинную ножку светодиода (анод) подключаем к цифровому выводу D10 Arduino, другую (катод) – через резистор 220 Ом к выводу GND (см. рис. 2.1).
2. Один вход кнопки подключаем к +5 В, другой – через резистор 10 кОм к GND, выход кнопки подключаем к входу D2 Arduino (см. рис. 2.1).
3. Загружаем в плату Arduino скетч из листинга 2.1.
4. При нажатии на кнопку светодиод должен гореть, при отпускании – потухнуть.

Усложним задачу – будем переключать состояние светодиода (включен/выключен) при каждом нажатии кнопки. Загрузим на плату Arduino скетч из листинга 2.2.

Листинг 2.2

```
const int LED=10;         // Контакт 10 для подключения светодиода
const int BUTTON=2;       // Контакт 2 для подключения кнопки
int tekButton = LOW;      // Переменная для сохранения текущего состояния кнопки
int prevButton = LOW;     // Переменная для сохранения предыдущего состояния
                          // кнопки
boolean ledOn = false;    // Текущее состояние светодиода (включен/выключен)

void setup()
   {
   // Сконфигурировать контакт светодиода как выход
   pinMode (LED, OUTPUT);
   // Сконфигурировать контакт кнопки как вход
   pinMode (BUTTON, INPUT);
   }
void loop()
  {
  tekButton=digitalRead(BUTTON);
  if (tekButton == HIGH && prevButton == LOW)
     {
     // нажатие кнопки - изменить состояние светодиода
     ledOn=!ledOn;
     digitalWrite(LED, ledOn);
     }
prevButton=tekButton;
}
```

При нажатии кнопки светодиод должен изменять свое состояние. Но это будет происходить не всегда. Виной тому – дребезг кнопок.

Кнопки представляют из себя механические устройства с системой пружинного контакта. Когда вы нажимаете на кнопку вниз, сигнал не просто меняется от низкого до высокого, он в течение нескольких миллисекунд меняет значение от одного до другого, прежде чем контакты плотно соприкоснутся и установится значение HIGH. Микроконтроллер зафиксирует все эти нажатия, потому что дребезг неотличим от настоящего нажатия на кнопку. Устранить влияние дребезга можно программно. Алгоритм следующий:

1. Сохраняем предыдущее состояние кнопки и текущее состояние кнопки (при инициализации LOW).
2. Считываем текущее состояние кнопки.
3. Если текущее состояние кнопки отличается от предыдущего состояния кнопки, ждем 5 мс, потому что кнопка, возможно, изменила состояние.
4. После 5 мс считываем состояние кнопки и используем его в качестве текущего состояния кнопки.
5. Если предыдущее состояние кнопки было LOW, а текущее состояние кнопки HIGH, переключаем состояние светодиода.
6. Устанавливаем предыдущее состояние кнопки для текущего состояния кнопки.
7. Возврат к шагу 2.

Добавляем к нашему скетчу подпрограмму устранения дребезга. Получаем код, показанный в листинге 2.3.

Листинг 2.3

```
const int LED=10;        // Контакт 10 для подключения светодиода
const int BUTTON=2;      // Контакт 2 для подключения кнопки
int tekButton = LOW;     // Переменная для сохранения текущего состояния кнопки
int prevButton = LOW;    // Переменная для сохранения предыдущего состояния
                         // кнопки
boolean ledOn = false;   // Текущее состояние светодиода (включен/выключен)

void setup()
   {
   // Сконфигурировать контакт светодиода как выход
   pinMode (LED, OUTPUT);
   // Сконфигурировать контакт кнопки как вход
   pinMode (BUTTON, INPUT);
   }
// Функция сглаживания дребезга. Принимает в качестве
// аргумента предыдущее состояние кнопки и выдает фактическое.
boolean debounce(boolean last)
```

```
   {
   boolean current = digitalRead(BUTTON);  // Считать состояние кнопки,
   if (last != current)                    // если изменилось...
      {
      delay(5);                            // ждем 5 мс
      current = digitalRead(BUTTON);       // считываем состояние кнопки
      return current;                      // возвращаем состояние кнопки
      }
   }

void loop()
   {
   tekButton = debounce(prevButton);
   if (prevButton == LOW && tekButton == HIGH)   // если нажатие...
      {
      ledOn = !ledOn;           // инвертировать значение состояния светодиода
      }
   prevButton = tekButton;
   digitalWrite(LED, ledOn);     // изменить статус состояния светодиода
 }
```

Загружаем скетч в плату Arduino и проверяем работу. Теперь все работает нормально, каждое нажатие кнопки приводит к изменению состояния светодиода.

На странице http://arduino-kit.ru/0002 вы можете посмотреть видеоурок данного эксперимента и скачать скетчи, представленные в листингах 2.1, 2.2, 2.3.

3 Потенциометр. Показываем закон Ома на примере яркости светодиода

В этом эксперименте мы познакомимся с потенциометром и будем управлять яркостью светодиода и изменением сопротивления потенциометра.

Необходимые компоненты:

- контроллер Arduino UNO;
- плата для прототипирования;
- потенциометр 2 кОм;
- светодиод;
- резистор 220 Ом;
- провода папа-папа.

В эксперименте 1 для подключения светодиода к цифровому выходу мы использовали ограничительный резистор номиналом 220 Ом. Сейчас мы рассмотрим, как подобрать ограничительный резистор и как будет влиять номинал резистора на яркость светодиода.

Самым главным уравнением для любого инженера-электрика является закон Ома. Закон Ома определяет отношения между напряжением, током и сопротивлением в цепи.

Закон Ома определяется следующим образом:

$$V = I \times R,$$

где V – напряжение в вольтах; I – ток в амперах; R – сопротивление в омах.

В электрической схеме каждый компонент имеет некоторое сопротивление, что снижает напряжение. Светодиоды имеют предопределенное падение напряжения на них и предназначены для работы в определенном значении тока. Чем больше ток через светодиод, тем ярче светодиод светится, до предельного значения. Для наиболее распространенных светодиодов максимальный ток составляет 20 мА. Обычное значение падения напряжения для светодиода – около 2 В.

Напряжение питания 5 В должно упасть на светодиоде и резисторе, поскольку доля светодиода 2 В оставшиеся 3 В должны упасть

на резисторе. Зная максимальное значение прямого тока через светодиод (20 мА), можете найти номинал резистора.

$R = V/I = 3/0{,}02 = 150$ Ом.

Таким образом, со значением резистора 150 Ом ток 20 мА протекает через резистор и светодиод. По мере увеличения значения сопротивления ток будет уменьшаться. 220 Ом немного более, чем 150 Ом, но все же позволяет светиться светодиоду достаточно ярко, и резистор такого номинала очень распространен. Если мы будем увеличивать номинал резистора, то будем уменьшать ток, проходящий через светодиод, и, соответственно, яркость светодиода.

Для изменения яркости светодиода мы будем использовать потенциометр. Потенциометры являются переменными делителями электрического напряжения. Как правило, это резистор с подвижным отводным контактом (движком). Они бывают разных размеров и форм, но все имеют три вывода. Номинал потенциометра определяет сопротивление между крайними выводами, оно неизменно, поворотом ручки мы изменяем сопротивление между средним и крайним выводами от 0 до номинала потенциометра либо от номинала до нуля.

В эксперименте потенциометр мы подключаем последовательно с резистором 220 Ом, чтобы не уменьшить значения ограничивающего резистора для светодиода до нуля и не сжечь светодиод. Схема подключения представлена на рис. 3.1.

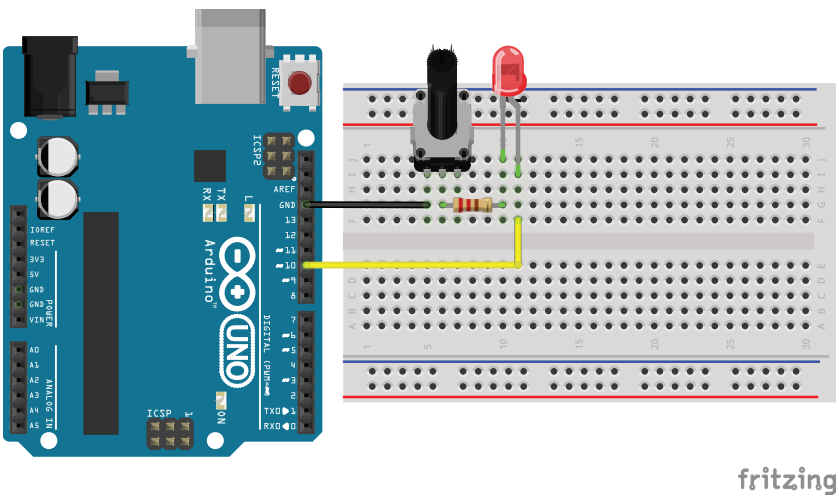

Рис. 3.1. Схема подключения потенциометра и светодиода